kolektiv autorů

Linux

Dokumentační projekt
3. aktualizované vydání

Computer Press
Brno
2003

3. aktualizované vydání

LINUX

Dokumentační projekt

Překlad oficiální linuxové dokumentace

Příručka uživatele

Příručka správce operačního systému

Příručka správce sítě

Praktické návody:

konfigurace ● jádro ● vypalování CD ● intranet ● server ● Bash ● DVD

NOVÉ

Apache ● software ● tisk ● fonty ● modemy ● přechod a koexistence s Windows ● hudba a zvuk ● MP3 ● digitální fotoaparáty ● zabezpečení ●

computer press

Linux – Dokumentační projekt

kolektiv autorů

3. aktualizované vydání.
Vydavatelství a nakladatelství Computer Press®,
Nám. 28. dubna 48, 635 00 Brno, http://www.cpress.cz

ISBN 80-7226-761-2

Prodejní kód: K0819

Překlad: Jiří Veselský, Ludvík Roubíček, Marek Kocan
Odborná korektura: Pavel Janík
Jazyková korektura: Eva Bublová
Vnitřní úprava: Petr Klíma
Sazba: René Kašík
Rejstřík: René Kašík

Obálka: IMIDEA
Komentář na zadní straně obálky: Ivo Magera
Technická spolupráce: Jiří Matoušek, Pavlína Bauerová
Odpovědný redaktor: Ivo Magera
Technický redaktor: Petr Klíma
Produkce: Petr Baláš

Tato publikace je volně dostupná na adrese www.cpress.cz/knihy/K0819

Computer Press, a.s., Nám. 28. dubna 48, 635 00 Brno
tel.: 546 122 111, fax: 546 122 112

Objednávejte na: **www.knihy.cpress.cz**
distribuce@cpress.cz

Bezplatná telefonní linka: **800 555 513**

Dotazy k vydavatelské činnosti směřujte na: **knihy@cpress.cz**
Objednávat můžete na adresách vydavatelství nebo přímo na: **www.knihy.cpress.cz**

Máte-li zájem o pravidelné zasílání informací o knižních novinkách do Vaší e-mailové schránky, zašlete nám jakoukoli, i prázdnou zprávu na adresu **novinky@cpress.cz.**

Obsah

ČÁST I

Příručka uživatele

Originál: http://www.ibiblio.org/pub/Linux/docs/linux-doc-project/users-guide/

Tato příručka obsahuje vše, co potřebujete znát pro práci s operačním systémem Linux, což je volně šířitelná obdoba operačního systému Unix pro osobní počítače. Popisuje základní příkazy operačního systému Unix a rovněž se zabývá některými specifickými vlastnostmi Linuxu. Manuál je určen zejména pro začátečníky, avšak zkušenější uživatelé zde rovněž naleznou spoustu užitečných informací.

Poděkování

Autor by rád poděkoval následujícím lidem za jejich neocenitelnou pomoc při psaní a korekci této příručky. Jsou to:

Linus Torvalds, autor operačního systému Linux, který poskytl řadu podkladů.

Karl Fogel mi pomohl s vytvářením dokumentace k operačnímu systému Linux a napsal podstatné části kapitol 8 a 9.

Maurizio Codogno napsal podstatnou část kapitoly 11.

David Channon napsal dodatek A věnovaný editoru `vi`.

Společnost **Yggdrasil Computing, Inc.** poskytla štědrou podporu pro realizaci této příručky.

Společnost **Red Hat Software** rovněž dobrovolně podpořila realizaci této příručky.

Typografické konvence použité v této části

Tučné písmo — Používá se pro nové pojmy, varovné poznámky a klíčová slova u programovacích jazyků.

Italika — Používá se ke zdůraznění textu.

Skloněné neproporcionální písmo — Používá se k označení meta-proměnných, zejména v příkazovém řádku. Například „`ls -l` *foo*“, kde foo by mělo být nahrazeno jménem souboru, jako například `/bin/cp`.

`Neproporcionální písmo` — Používá se k reprezentaci textů vypisovaných na obrazovce. Dále se používá při výpisu kódů (zpravidla vytvořených v jazyku C), skriptů a výpisu obecných souborů, jako jsou konfigurační soubory. Aby nemohlo dojít k nedorozumění, budou tyto výpisy uváděny v rámečcích.

Výrazy v rámečku — Označují klávesy, které se mají stisknout. Uvidíte je nejčastěji ve formě: „Pro pokračování stiskněte klávesu Enter.“

Tento symbol bude signalizovat zvláštní upozornění nebo nebezpečí. Odstavce takto označené čtěte se zvláštní pozorností.

Tento symbol bude označovat text týkající se zvláštních pokynů pro uživatele systému X Window.

Tímto symbolem jsou označeny odstavce, které obsahují důležité informace a měly by být čteny obzvlášť pečlivě.

KAPITOLA 1

Úvod

Kdo by si měl přečíst tuto knihu

Patříte mezi ty, kteří by si měli přečíst tuto knihu? Pokud neznáte přímou odpověď, pokuste se odpovědět na jiné otázky. Získali jste někde operační systém Linux a nevíte, co máte dělat dál? Jste uživateli jiného operačního systému než Unix a uvažujete o používání systému Linux? Chcete se dozvědět, co lze od tohoto operačního systému očekávat?

Máte-li k dispozici tuto knihu a přitom jste na některou z předcházejících otázek odpověděli „ano“, pak byste si ji měli přečíst. Každý, kdo si nainstaloval na svůj osobní počítač operační systém Linux (což je volně šiřitelná obdoba operačního systému Unix vytvořená panem Linusem Torvaldsem) a kdo neví, jak jej má efektivně využívat, by si měl tuto knihu přečíst. Je v ní je popsána většina základních příkazů operačního systému Unix a rovněž jsou zde uvedeny pokročilejší techniky pro práci se systémem. Rovněž se zmíníme o výkonném editoru GNU Emacs a několika dalších důležitých aplikacích.

Co byste měli udělat, než začnete tuto knihu číst

V této knize se předpokládá, že máte zajištěn přístup k operačnímu systému typu Unix a že tímto operačním systémem je Linux, tedy operační systém běžící na osobních počítačích s procesorem Intel.* Posledně jmenovaný předpoklad však není nezbytný – budeme upozorňovat na případy, kdy se ostatní unixové operační systémy od systému Linux liší.

Operační systém Linux je dostupný v mnoha formách, jež se nazývají distribuce. Předpokládá se, že máte nainstalovánu kompletní distribuci, jako je SlackWare, Redhat nebo Debian. Mezi jednotlivými distribucemi jsou jisté rozdíly, ale většinou nejsou důležité. Může se stát, že příklady uvedené v této knize se budou lišit od vaší distribuce. Ničeho se neobávejte, většinou půjde o odlišnosti zanedbatelné, se kterými si snadno poradíte. Pokud byste narazili na závažné rozdíly, neváhejte a napište o nich autorovi.

Pokud máte přístupová práva jako superuživatel (tedy například jako správce operačního systému nebo ten, kdo provedl instalaci), měli byste si pro sebe vytvořit normální uživatelský účet.

Potřebné informace najdete v instalační příručce. Jestliže nemáte přístupová práva jako super uživatel, pak požádejte o vytvoření účtu správce systému.

* Poznámka korektora: Operační systém Linux je v současné době k dispozici nejen pro procesory Intel, ale i SPARC, Alpha, ARM, PowerPC, pro stanice SGI a jiné.

Na čtení této knihy byste měli mít dostatek času a trpělivosti. Naučit se pracovat s operačním systémem Linux není jednoduché – většina lidí považuje ostatní operační systémy za jednodušší (jako například Macintosh Operating System). Jakmile se však naučíte základům, budete pracovat velmi snadno a efektivně. Unix je výkonný operační systém a umožňuje snadno provádět i velmi složité úlohy.

Navíc se v této knize předpokládá, že jste seznámeni se základní počítačovou terminologií. I když není tento předpoklad nezbytným, bude se vám kniha číst snadněji. Měli byste například vědět, co je to spustitelný program nebo co je to textový soubor. Pokud to nevíte, budete potřebovat někoho, kdo vám při studiu tohoto operačního systému poradí a pomůže.

Kdy se můžete čtení této knihy vyhnout

Jakýkoliv počítačový program se nejlépe naučíte tak, že si jej budete zkoušet na počítači. Většina lidí přijde na to, že čtení manuálů bez možnosti bezprostředně program nebo operační systém používat není příliš užitečné. Operační systém Unix se nejlépe naučíte tak, že jej budete používat. Nevyhýbejte se experimentování – můžete sice leccos pokazit, ale stále máte možnost provést novou instalaci. Většině havárií způsobených vaší nezkušeností se můžete vyhnout důsledným zálohováním.

Operační systém Unix nelze používat tak intuitivně, jako většinu ostatních operačních systémů. Proto byste si měli přečíst alespoň kapitoly 4, 5 a 6.

Jednu z možností, jak se vyhnout čtení této knihy, představují tzv. manuálové stránky, jež poskytují dokumentaci on-line.

Jak číst tuto knihu

Při čtení této knihy vám doporučujeme, abyste si bezprostředně vyzkoušeli to, co si právě přečtete. Pokud jste s některou tématikou již seznámeni, můžete příslušné odstavce přeskočit. Některá témata vám budou připadat zajímavá, jiná triviální, či dokonce nudná. Možná máte tolik sebejistoty, že budete některé příkazy zkoušet přesto, že přesně nevíte, jakou mají funkci. I to je způsob, jak se naučit pracovat s operačním systémem, i když poněkud riskantní.

Mnozí lidé ztotožňují operační systém Unix s příkazovým interpretem. Příkazový procesor je program, kterým lze řídit vše, co se v operačním systému Unix odehrává. Operační systém Unix je ve skutečnosti velmi složitý – většina jednoduše zadaných příkazů spustí spoustu procesů, o nichž uživatel zpravidla ani neví. V této knize probereme základní pravidla pro používání příkazového procesoru a také se zmíníme o důležitých programech, které jsou součástí operačního systému Unix nebo Linux.

Tato kapitola je spíše pseudokapitolou, která předkládá informace o obsahu knihy. Další kapitoly se zabývají tématy dle následujícího seznamu:

Druhá kapitola se zabývá vznikem operačních systémů Unix a Linux. Dále předkládá informace o nadaci Free Software Foundation a o projektu GNU.

Ve **třetí kapitole** je vysvětleno, jak začít a skončit práci s počítačem, co se stane při startování a při ukončení činnosti operačního systému. Většina těchto informací není pro práci s operačním systémem Linux důležitá. Jsou to ale informace užitečné a zajímavé.

Kapitola 4 představuje úvod k práci s příkazovým procesorem operačního systému Unix. Jedná se o prostředí, v němž se provádějí příkazy a spouštějí programy. Dozvíte se zde o základních příkazech a programech, které musíte znát, pokud máte používat operační systém Unix.

V **kapitole 5** se budeme zabývat systémem X Window. Ten představuje primární grafické uživatelské rozhraní pro operační systémy typu Unix a některé distribuce jej používají jako základní uživatelské prostředí.*

V **kapitole 6** se budeme zabývat pokročilejšími technikami při práci s příkazovým procesorem, zejména těmi, které vaši práci zefektivní.

V **sedmé kapitole** jsou stručně popsány příkazy operačního systému Unix. Čím více nástrojů budete umět používat, tím efektivnější bude vaše práce.

Kapitola 8 popisuje věhlasný editor Emacs. Jedná se o velký program, který většinu nástrojů operačního systému Unix integruje do jediného prostředí.

Devátá kapitola je věnována konfiguraci operačního systému Unix. Pomocí této konfigurace si můžete nastavit vlastnosti systému tak, aby se vám co nejlépe pracovalo.

V **desáté kapitole** se zmíníme o možnostech, které má uživatel operačního systému Unix k dispozici, chce-li komunikovat s ostatními počítači, případně se sítí Internet. Najdete zde informace o elektronické poště a o systému World Wide Web.

Jedenáctá kapitola popisuje některé komplikovanější příkazy.

V **kapitole 12** se dozvíte, jak se při práci s operačním systémem Unix nebo Linux snadno vyhnout chybám.

Dokumentace k operačnímu systému Linux

Tato uživatelská příručka (původní název zní: „*The Linux Users' Guide*“) je určena začátečníkům. V rámci projektu „*Linux Documentation Project*“ se však připravují knihy pro pokročilejší uživatele.

Další knihy týkající se operačního systému Linux

Dokumentace k operačnímu systému Linux dále obsahuje tyto knihy: „*Installation and Getting Started*“ – příručka popisující instalaci systému Linux. „*The Linux System Administrator's Guide*“ – dokumentace týkající se správy operačního systému Linux. „*The Linux Kernel Hackers' Guide*“ – kniha popisující jádro systému Linux a způsoby jeho modifikace. „*The Linux Network Administration Guide*“ – příručka týkající se instalace, konfigurace a používání síťových spojení.

Praktické návody – dokumenty HOWTO

Kromě knih napsaných v rámci projektu „*Linux Documentation Project*“ byla vytvořena řada krátkých dokumentů ve formě souborů týkajících se jednotlivých témat kolem operačního systému Linux. Tyto soubory s ev originále nazývají „*HOWTO*“ a do této knihy byly zařazeny na dvě desítky nejzajímavějších a nejdůležitějších z nich. Najdete je v části IV jako „*Praktické návody*“. Například dokument „*Sdílení modemů*“ popisuje možnosti sdílení modemu pod Linuxem.

Soubory s praktickými návody jsou dostupné v několika formách – jako ucelené knihy (například „*The Linux Bible*“ nebo „*Dr. Linux*“), nebo je můžete získat prostřednictvím diskusní skupiny comp.os.linux.answers. Soubory se rovněž nacházejí na mnoha serverech FTP nebo WWW. Centrálním místem obsahujícím informace o operačním systému Linux je *http://www.linux.org*.

* Poznámka korektora: V současné době používá systém X Window většina používaných distribucí operačního systému Linux.

Co je to „Linux Documentation Project"?

Cílem projektu *„Linux Documentation Project"* je shromáždit, utřídit a v jednotné formě prezentovat dokumentaci týkající se operačního systému Linux. Pracují na něm lidé po celém světě. Impuls k projektu dal Lars Wirzenius.

Předpokládá se, že v rámci projektu *„Linux Documentation Project"* budou postupně vydány knihy týkající se všech témat kolem operačního systému Linux. Pokud budete mít jakékoliv náměty na zlepšení této dokumentace, dejte prosím vědět autorovi této knihy.

Operační systémy

Primární funkce operačního systému spočívá v tom, že poskytuje podporu pro realizaci počítačových programů. Proto například můžete používat svůj oblíbený editor a vytvářet dokumenty. Bez podpory operačního systému by editor nemohl pracovat – editor potřebuje ke své činnosti interakci s vaším terminálem, s vašimi soubory a dalším technickým vybavením počítače.

Nyní si položme otázku: Proč je nutné psát a číst knihy o operačních systémech, které jsou pouhou podporou vašich aplikací? Každý operační systém obsahuje spoustu programů pro správu, konfiguraci, řízení spojení s ostatními systémy, zálohování a podobně. Pokud jde o operační systém Linux, najdete zde řadu „miniaplikací", jež vám pomohou pracovat efektivněji. Jestliže tedy nepoužíváte váš počítač pouze k práci s jednou nebo několika málo aplikacemi, je znalost operačního systému velmi důležitá.

Operační systém (nejčastěji označován zkratkou „OS") může být jednoduchý a minimalizovaný (například DOS) nebo velký a složitý (například OS/2 nebo VMS). Operační systémy typu Unix patří ke středně velkým systémům. Poskytují o něco více zdrojů a prostředků než první jednoduché operační systémy, a to v takové formě, aby s jejich pomocí bylo možné řešit prakticky všechny úlohy.

Původně byl operační systém Unix navržen jako zjednodušení operačního systému Multics. Filosofie operačního systému Unix spočívá v tom, že by se veškeré funkce měly rozdělit do malých částí, tedy relativně jednoduchých programů.[1] Nové funkční vlastnosti lze získat vhodnou kombinací těchto jednoduchých programů. Samozřejmě se stále objevují nové a nové obslužné programy, které lze snadno integrovat do vašich nástrojů, a tak můžete váš operační systém neustále rozšiřovat. Když jsem například psal tento dokument, používal jsem následující programy: `fvwm` pro obsluhu a konfiguraci systému X Window, editor `Emacs` pro vytvoření textu, `xdvi` pro prohlížení textu před tiskem, `LATEX` pro formátování textu, `dvips` pro přípravu tisku a `lpr` pro vlastní tisk. Kdyby například existoval jiný program pro prohlížení textu před tiskem, pak bych jej mohl použít, aniž bych změnil ostatní programy. V tomto okamžiku na mém počítači běží současně třicet osm programů (většina z nich se nachází v neaktivním stavu „sleep", dokud nebudou aktivovány k provedení nějakého úkolu).

Při používání operačního systému Unix budete jistě chtít minimalizovat množství práce potřebné k realizaci každého běžně prováděného úkolu. Operační systém Unix vám pro tento účel nabízí spoustu nástrojů. Abyste je mohli efektivně používat, budete muset poměrně přesně vědět, jakou funkci každý nástroj má. Pokud byste nad uskutečněním každého jednoduchého úkolu strávili ho-

1 Tato filosofie byla ve skutečnosti určena technickým vybavením počítačů, na kterých běžely první verze operačního systému Unix. Časem se ukázalo, že Unix dobře funguje i na počítačích s vyspělejším technickým vybavením. I dnes, po dvaceti pěti letech, je původní filosofie operačního systému Unix zcela vyhovující.

dinu nebo dokonce více, pak by vaše práce nebyla příliš produktivní. V této knize se dozvíte, jak a v které situaci využívat nástroje obsažené v operačním systému Unix a jak tyto nástroje kombinovat za účelem řešení složitějších úloh.

Klíčovou částí operačního systému je tzv. **jádro**. Ve většině operačních systémů, jako je Unix, OS/2 nebo VMS, plní jádro systému takové funkce, jako je spouštění programů, přidělování systémových zdrojů, přidělování času procesoru současně běžícím programům a podobně. Samozřejmě platí, že i jádro systému je program. Ten běží na vašem počítači jako první program po jeho nastartování, kdy realizuje všechny konfigurační funkce, a jako poslední program před vypnutím počítače, kdy realizuje všechny potřebné funkce k zastavení systému.

Co je Unix

Historie operačního systému Unix

V roce 1965 pracovaly společnosti Bell Telephone Laboratories (divize AT&T) a General Electric na projektu „MAC of MIT“, jehož cílem bylo vytvořit operační systém Multics. Později se společnost Bell Telephone Laboratories rozhodla od spolupráce odstoupit, ale v důsledku toho neměla k dispozici kvalitní operační systém.

Pánové Ken Thompson a Dennis Ritchie se rozhodli navrhnout operační systém, který by společnosti Bell Telephone Laboratories vyhovoval. Ken Thompson tento návrh realizoval při vytváření vývojového prostředí na počítači PDP-7. Další výzkumný pracovník společnosti Bell Telephone Laboratories, pan Brian Kernighan, dal novému operačnímu systému název Unix.

Později zveřejnil pan Dennis Ritchie programovací jazyk C. V roce 1973 byl Unix kompletně přepsán do jazyka C (původní systém byl vytvořen v assembleru[1]). V roce 1977 byl operační systém Unix převeden z počítače PDP na nový počítač s použitím procesu, jež se nazývá „**porting**“. Tato akce byla uskutečnitelná právě proto, že byl operační systém Unix přepsán v jazyce C.

Koncem sedmdesátých let byla společnosti AT&T protimonopolním úřadem zakázána činnost v oblasti počítačového průmyslu. Proto se společnost rozhodla za velmi výhodných finančních podmínek převést licenci na operační systém Unix na některé university. Unix se tedy stal populárním především v akademických kruzích, avšak postupem času se začal prosazovat i v komerční sféře. Dnešní podoba Unixu se zcela liší od verze z roku 1970. Existují dvě základní varianty: System V od společnosti USL (Unix System Laboratories, dnes jej vlastní Novell[2]) a BSD (Berkeley Software Distribution).

Poslední verze USL má označení SVR4[3] (čtvrtá verze), zatímco poslední verze od BSD má označení 4.4. Kromě těchto základních verzí však existuje spousta dalších verzí operačního systému Unix. Komerční verze jsou zpravidla odvozeny od jedné z verzí USL nebo BSD. Existuje však spousta verzí operačního systému Unix, které kombinují vlastnosti obou základních verzí.

Ceny současných komerčních verzí operačního systému Unix pro počítače s procesorem Intel se pohybují od 500 do 2000 dolarů.

---

1 Assembler je základní programovací jazyk, který je úzce spjat s konkrétním typem počítače. Programy napsané v jazyce assembler zpravidla nejsou přenositelné mezi různými počítačovými platformami.

2 Tato verze operačního systému Unix byla nedávno prodána společnosti Novell. Předtím byla verze USL vlastněna společností AT&T.

3 SVR4 je zkratkou pro „System V“ verze 4. (System V Release 4)

Historie operačního systému Linux

Autorem operačního systému Linux je pan Linus Torvalds. Jeho původní verze byla v průběhu asi deseti let zdokonalována bezpočtem lidí na celém světě. Linux představuje verzi operačního systému Unix pro osobní počítače s procesorem Intel, Alpha a další a byl kompletně vytvořen znovu – na jeho vývoji se společnosti Unix System Laboratories a Berkeley Software Distribution vůbec nepodílely. Zajímavé je, že se na vývoji operačního systému Linux podíleli lidé na celém světě – od Austrálie po Finsko a všichni doufali, že se Linux podaří uvést do podoby schopné konkurovat ostatním operačním systémům typu Unix.

Vlastní projekt operačního systému Linux začal výzkumem vlastností procesoru 386. Systém je navržen tak, aby maximálně využíval všech vlastností tohoto procesoru.

Na operační systém Linux se vztahují licenční podmínky GPL (GNU General Public Licence). Tato licence se vztahuje na veškeré programové vybavení produkované nadací Free Software Foundation a jejím cílem je zabránit komukoliv omezovat distribuční práva ostatních. Jinými slovy, podle licenčních podmínek GPL si můžete účtovat za distribuci programového vybavení GNU kolik chcete, ale nesmíte nikomu nařizovat, za jaký poplatek je má distribuovat dál. Dále musíte s každou distribucí programového vybavení GNU zpřístupnit zdrojové kódy[4], což je velmi užitečné pro programátory. Pak si totiž každý může například modifikovat operační systém Linux a dále distribuovat tuto modifikovanou verzi – opět za předpokladu, že ji bude distribuovat podle licenčních podmínek GPL.

Operační systém Linux podporuje většinu programového vybavení napsaného pro Unix, včetně systému X Window. Tento systém byl vytvořen na Massachusetts Institute of Technology tak, aby umožňoval operačním systémům Unix vytvářet grafická okna a interaktivně spolupracovat mezi sebou. Dnes platí, že je systém X Window implementován pro každou verzi operačního systému Unix.

Kromě standardů, které představují verze System V a BSD, existuje sada standardizačních dokumentů publikovaná úřadem pro standardizaci IEEE, která má označení **POSIX**. Operační systém Linux splňuje většinu podmínek uvedených v dokumentech POSIX-1 a POSIX-2. Přitom má v mnoha aspektech vlastnosti podobné systému BSD, ale má také vlastnosti, které najdete u systému System V. Stručně řečeno, pokud jde o standard, představuje operační systém Linux kombinaci (a většina lidí se domnívá, že velmi dobrou) všech tří standardů.

Většina programového vybavení dodávaná s operačním systémem Linux pochází od nadace Free Software Foundation a je součástí projektu GNU. Hlavním cílem projektu GNU je jednak zpřístupnit programové vybavení původně napsané pro operační systém Unix uživatelům ostatních operačních systémů, jednak vytvořit přenositelný operační systém, který bude mít prakticky stejné vlastnosti jako Unix. Přenositelný znamená, že tento operační systém poběží na všech počítačových platformách (ať jsou vybaveny procesorem Intel, PowerPC, Alpha či jiným). Tento operační systém má označení Hurd. Hlavní rozdíl mezi operačním systémem Linux a operačním systémem Hurd nespočívá v uživatelském rozhraní, ale v programátorském rozhraní – Hurd představuje moderní operační systém, zatímco Linux je operačním systémem založeným na filosofii stařičkého Unixu.

---

4 Zdrojový kód programu je textový soubor obsahující příkazy pro konkrétní programovací jazyk. Příslušným překladačem jej lze přeložit do strojového kódu pro konkrétní počítač. Strojový kód je soubor, který již není snadno čitelný a modifikovatelný tak jako zdrojový kód.

Předcházející odstavce by mohly vzbudit dojem, že se o operační systém Linux stará pouze pan Linus Torvalds. V raných dobách existence tohoto systému se například o překladač GCC (základní překladač jazyka C pro Linux) a o knihovny Linux C staral H. J. Lu. V každé distribuci operačního systému Linux naleznete v adresáři `/usr/src/linux/` soubor `CREDITS` obsahující seznam všech lidí, kteří se významnou měrou podíleli na jeho vývoji.

Dnešní podoba operačního systému Linux

První číslo ve verzi operačního systému Linux představuje číslo hlavní revize. V době, kdy byla napsána tato kniha (únor 1996) je k dispozici teprve první verze. Druhé číslo představuje méně podstatné revize. Pokud je druhé číslo sudé, jedná se o stabilní verzi. Jestliže je liché, jedná se o vývojovou verzi. Ve vývojových verzích bývá spousta chyb, které „odvážní" uživatelé postupně odhalují a programátoři postupně opravují. Jakmile jsou všechny závažné nedostatky z vývojové verze odstraněny, prohlásí se vývojová verze za stabilní a začne se pracovat na nové vývojové verzi. V únoru 1996 měla stabilní verze číslo 1.2.11 a poslední vývojová verze 1.3.61.*

Operační systém Linux je velkým systémem obsahujícím spoustu chyb, které se postupně odstraňují. Ve stabilních verzích se však vyskytují chyby velmi zřídka a souvisejí hlavně s nestandardním technickým vybavením počítače. Doposud jsme hovořili pouze o chybách jádra systému. Operační systém však zdaleka není jen jádro systému, ale obsahuje spoustu obslužných programů.

Ani velmi zkušený uživatel často není schopen určit, zda je původ chyby v obslužném programu nebo v jádře systému. Také je velmi nesnadné rozeznat, zda jsou některé „divné" věci chybou operačního systému, nebo novou vlastností. Doufáme, že vám tato kniha pomůže orientovat se právě v takových situacích.

Několik málo otázek a odpovědí na ně

Než se pustíme do další kapitoly, odpovězme si na několik důležitých otázek.

Otázka: Jak se má vyslovovat Linux?

Odpověď: Podle autora operačního systému Linux se má samohláska „i" vyslovovat krátce (jako například ve slově minimum). Linux by se měl rýmovat se slovem Minix, což je další verze operačního systému Unix. Samohláska „u" by se měla vyslovovat ostře, jako například ve slově „rule", a nikoliv měkce, jako například ve slově „ducks". Obecně by se mělo slovo Linux rýmovat se slovem „cynics". V našich zemích se Linux vyslovuje „linuks".

Otázka: Proč by se mělo pracovat s operačním systémem Linux?

Odpověď: Proč ne. Linux je obecně levnější než ostatní operační systémy a je méně problematický než většina komerčních operačních systémů. Nemusí být pravda, že zrovna Linux je tím nejlepším operačním systémem pro vaši konkrétní aplikaci. Pokud má však někdo zájem o používání operačního systému typu Unix na osobním počítači a o aplikace vytvořené pro Unix, pak je Linux pravděpodobně nejlepším vysoce výkonným systémem.

Komerční programové vybavení pro operační systém Linux

Pro operační systém Linux existuje spousta komerčního programového vybavení. Nejdůležitější roli hraje systém Motif, což je uživatelské rozhraní pro systém X Window připomínající Microsoft Windows. Na základě systému Motif byla vytvořena řada komerčního programového vybavení.

* Poznámka korektora: V současné době (únor 2003) je poslední stabilní verze 2.4.20 a vývojová verze 2.5.59.

Dnes můžete koupit prakticky cokoliv ve verzi pro operační systém Linux – od oblíbeného textového procesoru Word Perfect až po Maple, univerzální nástroj pro matematické a fyzikální modelování.

Možná se budete divit, jak lze skloubit operační systém Linux, jenž je distribuován podle licenčních podmínek GPL (GNU General Public Licence, viz dodatek B), s komerčním programovým vybavením. Podmínky GPL platí pouze pro jádro systému, zatímco na ostatní aplikace vytvořené pro Linux se vztahují jiné podmínky GNU, *„Library General Public Licence"* (viz dodatek B). Podle těchto podmínek smějí poskytovatelé programového vybavení prodávat své aplikace a přitom nemusejí distribuovat zdrojové kódy.

Uvědomte si prosím, že uvedené dva dokumenty představují něco jako autorská práva, ale v žádném případě nepředstavují licenci na užívání. Tyto dokumenty nikterak nevymezují způsoby, kterými můžete dané programové vybavení používat, ale vymezují pravidla, kterými se musíte řídit při distribuci tohoto programového vybavení. V tom spočívá hlavní myšlenka filosofie nadace Free Software Foundation a platí i pro Linux: na používání operačního systému Linux neexistuje žádná licence.

KAPITOLA 3

Začínáme pracovat s operačním systémem Linux

Pravděpodobně máte jisté zkušenosti s používáním jednouživatelského operačního systému, jako je DOS nebo OS/2. Chcete-li pracovat v těchto operačních systémech, nemusíte se identifikovat – předpokládá se, že jste jediným uživatelem operačního systému, který má přístup ke všem jeho zdrojům. Na druhé straně, operační systém Unix je víceuživatelským systémem. To znamená, že v daném okamžiku může mít k systému přístup více uživatelů najednou.

Aby mohl operační systém Unix komunikovat s každým uživatelem odděleně, musí jej identifikovat procesem, který se nazývá „**přihlášení se**“ (logging in). Když zapnete počítač, proběhne spousta inicializačních procesů a až potom je počítač schopen komunikovat s uživatelem. Protože je tato příručka věnována operačnímu systému Linux, vysvětlíme si, co se odehrává v průběhu zavádění tohoto operačního systému Linux.

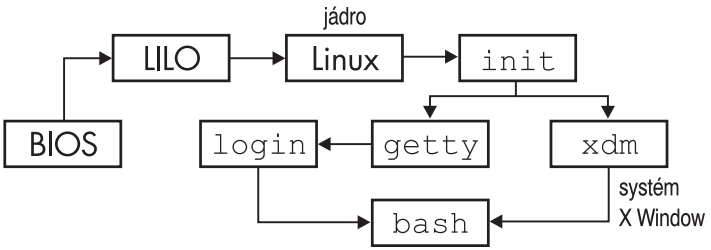
Pokud nepoužíváte operační systém Linux na osobním počítači s procesorem Intel, pak se vás nebudou některé informace v této kapitole týkat. Většinu užitečných informací najdete v oddílu Zapnutí počítače.

Jestliže se pouze zajímáte o používání vašeho počítače, pak kromě oddílu Činnost uživatele nemusíte tuto kapitolu číst.

Zapnutí počítače

Po zapnutí počítače s procesorem Intel se jako první provede program zvaný BIOS. BIOS je zkratkou pro Basic Input/Output System. Tento program je trvale uložen ve zvláštní paměti počítače. Realizuje některé základní testy a identifikuje technické vybavení, jako je pevný disk či disketová jednotka. Po identifikaci disku najde tzv. zaváděcí sektor (boot sector) a spustí kód, který v něm nalezne. Jestliže byl nalezen disk, ale žádný zaváděcí sektor, vypíše BIOS následující hlášení:

```
Non-system disk or disk error
```



Obrázek 3.1 – Cesta, po které dospěje osobní počítač s procesorem Intel k příkazovému řádku. Proces `init` může, ale také nemusí realizovat zavedení systému X Window. Pokud systém X Window zavádí, spustí nejdříve program `xdm`. Jinak spustí příslušný počet procesů `getty`.

Pak ovšem stačí disk bez zaváděcího sektoru vyjmout, stisknout kteroukoliv klávesu a zaváděcí proces bude pokračovat.

Pokud není v disketové jednotce vložena disketa, BIOS vyhledá hlavní zaváděcí záznam MBR (master boot record) na pevném disku (závisí na nastavení BIOSu). Ten obsahuje kód (zaváděcí program), jehož úkolem je zavést operační systém. Pro operační systém Linux se tento program jmenuje LILO (LInux LOader). Linux Loader je nejpoužívanějším bootovacím programem pro Linux, ale je možno použít i GRUB Loader, SYSLinux nebo i NT Loader. To znamená, že program LILO je uložen právě v diskové oblasti MBR. Nyní budeme předpokládat, že se zavádí operační systém Linux. (U vaší konkrétní distribuce může proces zavádění operačního systému Linux probíhat poněkud odlišně – proto si pečlivě pročtěte dokumentaci k vaší distribuci. Další užitečné informace naleznete v dokumentaci k zaváděcímu programu LILO.)

Jak Linux přebírá kontrolu nad počítačem

Nejdříve tedy předá BIOS kontrolu zaváděcímu programu LILO a ten pak předá kontrolu **jádru** operačního systému Linux (Linux kernel). Jádro je centrálním programem celého systému a řídí všechny ostatní programy. V prvním kroku jádro realizuje přepnutí procesoru do tzv. chráněného módu. Procesor 80386[1], jež řídí váš počítač, může pracovat ve dvou módech: v tzv. „reálném módu“ a v tzv. „chráněném módu“. Operační systém DOS, stejně jako BIOS, funguje v reálném módu. Vyspělejší operační systémy však pracují v chráněném módu. Proto platí, že Linux po svém zavedení zcela nahradí BIOS.

Pokud jde o počítače s jinými procesory, probíhá tato fáze zavádění systému Linux poněkud jinak. Zpravidla nedochází k přepínání do chráněného módu a jen několik málo procesorů vyžaduje komplikované zavádění operačního systému podobné kombinaci BIOS – LILO. Po zavedení jádra pracuje Linux na všech počítačích stejně bez ohledu na procesor.

V dalším kroku se Linux pokouší identifikovat technické vybavení počítače, na kterém běží. Musí znát informace o pevném disku, zda je nebo není k dispozici myš, zda je nebo není váš počítač připojen k počítačové síti a podobně. Linux si není schopen po vypnutí počítače tyto údaje pamatovat, proto je pokaždé zjišťuje znovu. Naštěstí nemusíte údaje o technickém vybavení vašeho počítače zadávat – Linux si je zjistí sám.

1 Pod pojmem procesor 80386 budeme dále rozumět všechny 32bitové procesory Intel, tedy i 80486, Pentium, Pentium Pro, Pentium MMX a další. Místo 80386 budeme také používat označení 386.

V průběhu zavádění vypisuje operační systém Linux řadu hlášení. Podrobněji se jimi budeme zabývat v oddílu Hlášení jára systému. Asi by se vám nelíbilo, pokud byste museli při každém startu odpovídat na spoustu dotazů týkajících se technického vybavení a konfigurace vašeho počítače. Na několik důležitých dotazů tohoto druhu však budete muset odpovědět při instalaci systému. Budete-li mít jakékoliv problémy, pročtěte si dokumentaci k vaší distribuci.

Jak jsme se již zmínili, jádro řídí všechny ostatní programy. Jestliže tedy úspěšně identifikuje veškeré technické vybavení počítače, spustí jiný program, který již začne dělat něco užitečného. Tento program má název `Proces init`. (Všimněte si, že pro název programu jsme použili jiný font. `Neproporcionální font` budeme používat k označení programů, adresářů i obecných souborů.) Po nastartování programu `Proces init` se jádro stane jakýmsi „manažerem“, avšak již není aktivním programem.

Jestliže chceme vědět, co dělá počítač po zavedení jádra, musíme studovat Proces init. Ten prochází mnoha komplikovanými stádii, která nejsou na všech počítačích stejná. Operační systém Linux má mnoho verzí programu `Proces init` – každá z nich realizuje své cíle různými způsoby. Také záleží na tom, zda je váš počítač zapojen do sítě a jakou distribuci operačního systému Linux používáte. Nyní si uveďme některé základní procesy, jež `init` realizuje:

- Zpravidla proběhne kontrola souborového systému. Co je to souborový systém? Je to způsob organizace a ukládání souborů na pevném disku. Linux při kontrole souborového systému zjišťuje, které části disku jsou již použity a které jsou volné. Souborový systém se podobá rejstříku nebo štítkovému katalogu v knihovně. Bohužel se stává, že například při výpadku elektrické energie dojde k poškození některých souborů a pak je používání některých částí pevného disku konfliktní. `Proces init` proto spouští další program označený jako `fsck`, jenž se pokouší vadné soubory a konfliktně obsazené části pevného disku opravit.
- Pokud je váš počítač připojen k síti, spustí se několik speciálních programů, jež zajišťují komunikaci vašeho počítače s jinými počítači.
- Na pevném disku mohou zůstat některé dočasné soubory, které jsou jinak neužitečné. Ty mohou být programem `Proces init` zrušeny.
- Zpravidla také dojde k aktualizaci systémového času. Operační systém Unix předpokládá, že je čas definován jako UTC (Universal Coordinated Time), známý také jako střední greenwichský čas. Přitom je čas ve vašem počítači pravděpodobně nastaven na lokální – to znamená, že některý program musí přečíst lokální čas nastavený ve vašem počítači a korigovat jej na čas UTC.

Jakmile `Proces init` ukončí všechny uvedené aktivity, přejde do stavu, jenž lze označit jako „rodič“ všech procesů v systému Unix. Pod pojmem proces si lze jednoduše představit běžící program. Protože jeden program může být spuštěn dvakrát nebo vícekrát, mohou existovat dva procesy nebo více procesů realizujících konkrétní program.

V operačním systému Unix tedy platí, že proces je instancí programu. Vytváří se systémovým voláním (službou poskytovanou jádrem systému) nazvaným „fork“ (rozvětvení). Pojem „fork“ se používá proto, že se původně jeden proces při spuštění dalšího procesu rozdělí na dva oddělené procesy. Typickým příkladem programu, který běží jako několikanásobný proces, je program `getty`. `getty` je důležitý program, jenž voláním programu `login` umožňuje uživateli přihlásit se do systému.

Činnost uživatele

Jak se přihlásit do systému

Než začnete používat operační systém Unix, musíte se identifikovat. Procesu identifikace se říká „**přihlášení do systému**“ (`login`). Jedná se o proces, ve kterém se operační systém přesvědčí, zda má uživatel oprávnění systém používat. Systém se vás zeptá na jméno účtu (account name) a heslo (password). Jméno účtu je zpravidla podobné vašemu jménu – pravděpodobně jste jej obdrželi od vašeho systémového správce nebo jste si vytvořili vlastní, pokud jste sami systémovým správcem. (Informace o vytváření jména účtu naleznete v příručce *„Příručka správce operačního systému Linux“.*)

Po dokončení všech zaváděcích procesů byste měli na svém monitoru spatřit hlášení podobné následujícímu. Nejnovější verze Linuxu se hlásí takto (první řádek je pouze uvítací zprávou – může zde být cokoliv jiného):

```
RedHat Linux release 5.1 (Manhattan) with all updates. Kernel 4.1.108 on an Intel
mousehouse login:
```

Je však možné, že vás systém uvítá něčím úplně jiným. Místo nudné textové obrazovky se vám může představit nějaká grafická obrazovka. Avšak i v tomto případě budete vyzváni k přihlášení se do systému a proběhnou stejné procedury identifikace uživatele. Pokud se vám objevila zmíněná grafická obrazovka, pak jste zřejmě uživateli systému X Window. To znamená, že budete pracovat v systému oken, o kterém se podrobněji zmíníme v kapitole 5. Všimněte si, že veškerý výklad o systému X Window bude v této knize označován velkým písmenem X na levém okraji textu.

Jako jméno účtu budeme používat smyšlené jméno `larry`. Kdykoliv uvidíte v následujícím textu jméno `larry`, měli byste místo něj dosadit své vlastní jméno účtu. Jména účtu by měla být založena na skutečném jménu uživatele; ve větších operačních systémech typu Unix se zpravidla používají příjmení uživatelů, kombinace jména a příjmení nebo kombinace jména a číslic. Pro uživatele se jménem Larry Greenfield by se mohly použít například tyto kombinace: `larry`, `greenfie`, `lgreenfi` nebo `lg19`.

Pro bližší vysvětlení, jméno `mousehouse` je „jménem“ počítače, na kterém pracuji. Je pravděpodobné, že jste při instalaci operačního systému Linux použili jiné, podobně duchaplné jméno. Jméno počítače není důležité; v této knize budeme používat již uvedené jméno `mousehouse`, případně `lionsden`. Jestliže tedy zadáte své jméno účtu a stisknete klávesu Enter, objeví se výzva k zadání hesla:

```
mousehouse login: larry
Password:
```

Na druhém řádku očekává operační systém Linux zadání hesla (password). Při zadávání hesla neuvidíte na obrazovce, co píšete, proto zadávejte heslo pečlivě. Chybně zadaný znak můžete zrušit, ale opět nebudete vidět, který znak jste zrušili. Jestliže se na vás zrovna někdo dívá, nezadávejte heslo příliš pomalu. Mohlo by tak dojít k vyzrazení vašeho hesla a nepovolaná osoba by jej mohla zneužít. Jestliže zadáte heslo chybně, budete mít možnost zadat je znovu.

Pokud jste zadali heslo správně, objeví se zpravidla na vaší obrazovce nějaká zpráva, jež se většinou nazývá „zpráva dne“. Tato zpráva může obsahovat cokoliv – její znění zadává systémový správce.

Jako další se objeví tzv. výzva příkazového řádku (prompt). Výzva příkazového řádku je řetězec indikující skutečnost, že systém očekává zadání příkazu. Mohla by vypadat takto:

```
/home/larry#
```

Pokud jste uživateli systému X Window, pak zřejmě uvidíte výzvu podobnou výše uvedené v nějakém okně na obrazovce. Okno je obdélníková orámovaná oblast obrazovky, která v tomto případě imituje terminál. Pokud chcete zadat příkaz v příkazovém řádku v okně, musí být okno aktivní – stačí například do tohoto okna přesunout kurzor myši.

Jak ukončit práci s počítačem

Chcete-li skončit práci s počítačem, pak rozhodně nestačí pouze počítač vypnout. Pokud to uděláte v operačním systému Linux, s největší pravděpodobností ztratíte nějaká data nebo poškodíte nějaké důležité soubory.

Na rozdíl od operačního systému DOS, kde lze počítač vypnout vypínačem nebo restartovat tlačítkem Reset, je v operačním systému Linux nutné provést řadu kroků, které zajistí bezpečné ukončení systému. Uveďme si hlavní důvod. Operační systém Linux používá tzv. **paměť cache** pro diskové operace. To znamená, že jistá část souborů uložených na pevném disku je umístěna do paměti RAM[3]. K synchronizaci mezi pamětí RAM a pevným diskem dochází přibližně každých třicet sekund. Jestliže se má vypnout nebo restartovat počítač, pak je nutné, aby se část paměti RAM obsahující disková data uložila.

Máte-li tedy ukončit práci s počítačem a přitom jste normálně přihlášeni (t.j. zadali jste jméno uživatelského účtu a heslo), pak se musíte odhlásit. K tomuto účelu slouží příkaz `logout`. Napište v příkazovém řádku `logout` a stiskněte klávesu Enter. Klávesou Enter se odesílá každý příkaz. Než tuto klávesu stisknete, můžete chybně zadané znaky opravovat, případně celý příkaz zrušit a zadat nový.

```
RedHat Linux release 5.1 (Manhattan) with all updates.
Kernel 4.1.108 on an Intel
mousehouse login:
```

Nyní se může do systému přihlásit další uživatel.

Vypnutí počítače

Jste-li jediným uživatelem počítače, pak po odhlášení můžete počítač odpojit od zdroje elektrické energie.[4] V takovém případě se přihlašujte jako uživatel se speciálním jménem účtu root. Účet root je účtem systémového správce, který má zajištěn přístup ke každému souboru v systému. Pokud hodláte vypnout počítač, pak musíte obdržet heslo od systémového správce. (Jste-li jediným uživatelem počítače, pak jste systémovým správcem právě vy! Proto nikdy nezapomeňte heslo!)

---

3 Rozdíl mezi pamětí RAM a pevným diskem lze přirovnat ke krátkodobé a dlouhodobé paměti. Po vypnutí počítače se veškeré informace v paměti RAM ztratí, zatímco informace uložené na pevném disku zůstávají uchovány. Na druhé straně platí, že přístup k informacím na pevném disku je nesrovnatelně pomalejší, než přístup k informacím v paměti RAM.

4 Chcete-li šetřit některé komponenty technického vybavení vašeho počítače, pak vypínejte počítač jen když nebudete na počítači pracovat delší dobu (například večer). Časté zapínání a vypínání počítače zbytečně namáhá některé jeho součásti.

Přihlášení a odhlášení by mohlo vypadat takto:

```
mousehouse login: root
Password:
Last login: Sun 5 11:14:57 on tty1
/# shutdown -h now

Broadcast message from root (tty1) Sun 14:52:32 1998...

The system is going down for system halt NOW!
INIT:Switching to runlevel: 0
INIT: sending processes the TERM signal
The system is halted
System Halted
```

Po zadání příkazu `shutdown -h now` proběhnou jisté procedury, které připraví systém na vypnutí nebo reset.

Na závěr ještě uveďme krátkou poznámku pro pohodlnější uživatele. Místo procesu přihlašování a odhlašování jako uživatel root lze použít příkaz `su`. Jako běžný uživatel zadejte v příkazovém řádku příkaz `su` a stiskněte klávesu Enter. Systém si vyžádá heslo uživatele root a pak vám přidělí všechna jeho privilegia. Nyní můžete bez problémů systém ukončit příkazem `shutdown -h now`.

Hlášení jádra systému

Když poprvé startujete počítač, objeví se na obrazovce spousta hlášení popisující technické vybavení vašeho počítače. Uvedená hlášení vypisuje jádro operačního systému Linux. V tomto oddílu si některá důležitá hlášení podrobněji popíšeme.

Přirozeně platí, že se hlášení jádra systému budou lišit, což je dáno jednak typem počítače a jednat distribucí operačního systému Linux. V následujícím textu budou popsána hlášení platná pro můj počítač. Některá mají obecnou platnost, jiná jsou více specifická. Musím poznamenat, že můj počítač má minimální konfiguraci, pokud jde o technické vybavení, proto dále neuvidíte příliš mnoho informací týkajících se speciálního technického vybavení. Následující hlášení pocházejí z verze operačního systému Linux 1.3.55. Tato verze patří v době, kdy píši tuto knihu, k nejnovějším.

1. V prvním kroku operační systém Linux identifikuje grafickou kartu, aby mohl vybrat nejvhodnější font a jeho velikost. (Čím menší font je zvolen, tím větší počet hlášení se vejde na jednu obrazovku.) Linux se vás může zeptat, jaký font má používat, nebo použije standardní „kompilovaný“ font (compiled font).[5]

   ```
   Console: 16 point font, 400 scans
   Console: colour VGA+ 80x25, 1 virtual console (max 63)
   ```

   V předcházejícím příkladu se vlastník počítače v době kompilace rozhodl pro větší, standardní font. Také si všimněte zastaralého tvaru slova „colour“. Linus se evidentně naučil špatnou verzi angličtiny.

2. V dalším kroku zobrazí jádro operačního systému zprávu o výkonnosti vašeho počítače. Výkonnost se měří v jednotkách „BogoMIPS“. Zkratka MIPS (million instructions per se-

5 Kompilace je proces, při kterém se instrukce srozumitelné člověku přeloží do instrukcí srozumitelných počítači. Pojmem kompilovaný se zde rozumí „začleněný do programu“.

cond) označuje počet instrukcí, které je počítač schopen provést za sekundu. „BogoMIPS“ je zkratkou „bogus MIPS“ (bogus=podvodný, falešný) a označuje, kolikrát za sekundu neudělá počítač absolutně nic. Protože cyklus, kterým se výkonnost počítače měří, nedělá ve skutečnosti nic, nelze uvedené číslo považovat za objektivní ukazatel výkonnosti.* Operační systém Linux toto číslo používá v případě, kdy potřebuje vyčkat na odezvu nějakého zařízení.

```
Calibrating delay loop.. ok - 33.28 BogoMIPS
```

3. V třetím kroku vypíše jádro systému informace o použití paměti:

```
Memory: 23180k/24576k available (544 kernel code, 384k reserved,468k data)
```

Z této informace vyplývá, že má počítač 24 MB operační paměti. Část této paměti byla rezervována pro jádro. Zbytek může být využíván ostatními programy. Uvedené informace se týkají paměti **RAM**, která může být používána k uchovávání dat, jen když je počítač zapnut. Počítač však má k dispozici permanentní paměť zvanou **pevný disk**. Obsah pevného disku zůstává uchován i tehdy, když je váš počítač vypnut.

4. V průběhu zavádění operačního systému provádí Linux řadu testů technického vybavení a o výsledcích těchto testů vypisuje průběžné zprávy.

```
This processor honours the WP bit even when in supervisor mode. Good.
```

5. V dalším kroku se Linux zabývá konfigurací sítě. Následující zpráva by měla být popsána v manuálu *„Příručka správce sítě“*.

```
Swansea University Computer Society NET3.033 for Linux 1.3.50
IP Protocols: ICMP, UDP, TCP
```

Popis konfigurace sítě přesahuje rámec této dokumentace, a proto se jím nebudeme zabývat.

6. Operační systém Linux podporuje jednotku pro výpočet v pohyblivé řádové čárce FPU (floating point unit). Jedná se o speciální integrovaný obvod (nebo přímo součást procesoru 80486DX), jenž realizuje aritmetické operace s neceločíselnými operandy. Některé z těchto integrovaných obvodů mohou být špatné a když se je operační systém Linux pokusí detekovat, dojde k jeho zhroucení – počítač přestane být funkční. Pokud nastane tento případ, uvidíte na obrazovce zprávu:

```
Checking 386/387 coupling...
```

V případě, že používáte procesor 486DX, uvidíte tuto zprávu:

```
Checking 386/387 coupling... Ok, fpu using exception 16 error reporting.
```

Máte-li starší procesor 386 s matematickým koprocesorem 387, uvidíte zprávu:

```
Checking 386/387 coupling... Ok, fpu using irq13 error reporting.
```

7. Nyní proběhne další test, jenž testuje instrukci „halt“.

* Poznámka korektora: Velikost tohoto čísla může být např. ovlivněna typem procesoru, jeho schopností předvídat cykly a také např. místem v paměti, kde se testovací cyklus provádí.

```
Checking 'hlt' instruction... Ok.
```

8. Po dokončení počáteční konfigurace vypíše operační systém Linux řádek, kterým identifikuje sám sebe. Na tomto řádku je informace o verzi systému, verzi překladače GNU C Compiler, kterým bylo jádro přeloženo, a kdy byl překlad realizován.

```
Linux version 1.3.55 (root@mousehouse) (gcc version 2.7.0)
Sun Jan 7 14:56:26 EST 1996
```

9. Jako další se nastartuje ovladač sériového portu, který zjistí informace o příslušném technickém vybavení. **Ovladač** (driver) je součást jádra operačního systému, která řídí nějaké zařízení, zpravidla periferní. Ovladač je odpovědný za detaily komunikace mezi procesorem a periferním zařízením. Ovladače umožňují, aby se programátoři mohli při psaní aplikací soustředit na vlastní aplikaci a nemuseli se starat o takové problémy, jako je například tisk na tiskárně.

```
Serial driver version 4.11 no serial optins enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
tty02 at 0x03e8 (irq = 4) is a 16450
```

V tomto případě byly nalezeny tři sériové porty. Sériový port je ekvivalentní portu COM v operačním systému DOS. Jedná se o zařízení běžně používané ke komunikaci s modemy nebo myší.

Uvedený výpis sděluje, že sériový port s pořadovým číslem 0 (`COM1`) má adresu `0x03f8`. Když port přeruší činnost jádra (zpravidla proto, aby sdělil systému, že chce přenášet data), použije přerušení IRQ 4. Přerušení IRQ představuje další prostředek pro komunikaci mezi programovým vybavením a periferním zařízením. Každý sériový port má také svůj řídící integrovaný obvod. Často se používají obvody 16450, mohou se však také vyskytnou obvody 16550 nebo 820.**

10. Nyní přichází na řadu paralelní port. Paralelní port se nejčastěji připojuje k tiskárně a v operačním systému Linux začínají jména paralelních portů dvojicí písmen lp. lp je zkratkou pro Line Printer (řádková tiskárna), i když v poslední době by se mělo spíše jednat o zkratku pro Laser Printer (laserová tiskárna). Samozřejmě platí, že je operační systém Linux schopen komunikovat s každým typem paralelní tiskárny – jehličkovou, bublinkovou i laserovou.***

```
lp0 at 0x03bc, (polling)
```

Uvedená zpráva říká, že v systému byl nalezen jeden paralelní port a že pro něj bude použit standardní driver.

11. V dalším kroku identifikuje operační systém Linux ovladače pevných disků. Podle následující zprávy byly identifikovány dva pevné disky IDE:

```
hda: WDC AC2340, 325MB, w/127KB Cache, CHS=1010/12/55
hda: WDC AC2850F, 814MB, w/64KB Cache, LBA, CHS=827/32/63
```

---

* Poznámka korektora: V novějších jádrech Linuxu je obsažen i test na procesory, které obsahují i tzv. chybu „F00F".

** Poznámka korektora: V novějších počítačích je téměř vždy obvod 16550A.

*** Poznámka korektora: V poslední době se objevují i tzv. GDI-tiskárny, jejichž podpora není ještě zcela dokonalá, protože jejich výrobci nejsou ochotni poskytovat programátorům specifikaci komunikačního protokolu.

12. Dále provede jádro kontrolu disketových jednotek. Podle následujícího příkladu má počítač dvě disketové jednotky: jednotku „A“ pro diskety 5 1/4 palce a jednotku „B“ pro diskety 3 1/2 palce. Operační systém Linux přidělí disketové jednotce „A“ jméno `fd1` a disketové jednotce „B“ jméno `fd0`.

```
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M
floppy: FDC 0 is a National Semiconductor PC87306
```

13. Dalším ovladačem v případě mého počítače je ovladač SLIP. Vypíše následující zprávu o své konfiguraci:

```
SLIP: version 0.8.3-NET3.019-NEWTTY (dynamic channels, max=256)
  (6 bit encapsulation enabled)
CSLIP: code copyright 1989 Regents of the University of
  California
```

14. Dále jádro ověří všechny pevné disky, které byly identifikovány. Prohledá různé diskové oddíly a identifikuje ty části, na kterých se nachází operační systém, aby zabránil případné interferenci s jinými operačními systémy. V následujícím příkladě jsou identifikovány dva pevné disky: `hda` se čtyřmi oddíly a `hdb` s jedním oddílem:

```
Partition check:
hda: hda1 hda2 hda3 hda4
hdb: hdb1
```

15. V konečné fázi provede operační systém Linux připojení (mount) hlavního souborového systému platného pro kořenový oddíl (root partition). Kořenový oddíl je část pevného disku, ve které je umístěn operační systém. Když operační systém připojuje hlavní souborový systém, zpřístupní jej uživateli:

```
VFS: Mounted root (ext2 filesystem) readonly.
```

KAPITOLA 4

Příkazový interpret operačního systému Unix

Příkazy operačního systému Unix

Když se poprvé přihlásíte do operačního systému Unix, objeví se na obrazovce **výzva příkazového řádku** (prompt), která by mohla vypadat takto:

```
/home/larry#
```

Podle názvu „výzva příkazového řádku“ lze usuzovat, že systém na tomto místě očekává zadání příkazu. Každý příkaz operačního systému Unix představuje posloupnost písmen, čísel a znaků. Nepatří sem však mezery. Platnými příkazy operačního systému Unix jsou například: `mail`, `cat` nebo `CMU_is_Number-5`. Některé znaky není v příkazech povoleno používat – zmíníme se o nich později. Poznamenejme, že Unix má vlastnost označovanou jako „**case-sensitive**“ (doslova citlivý na malá a velká písmena).[1] To znamená, že `cat` a `Cat` jsou různé příkazy.

Příkazový řádek je zobrazován speciálním programem, který se nazývá **příkazový interpret** (shell). Příkazový interpret přijímá příkazy a realizuje je. Příkazové interprety mají svůj vlastní programovací jazyk a programy napsané v tomto jazyce se nazývají skripty příkazového interpretu (shell scripts).

Pro operační systémy Unix existují dva hlavní typy příkazových interpretů: Bourne shell a C shell. (V současnosti existuje více než desítka různých příkazových interpretů – např. v distribuci Red Hat Linux 5.1 je pro každého uživatele k dispozici pět příkazových interpretů – `ash`, `bash`, `tcsh`, `zsh` a `pdksh` – ze kterých si může vybrat příkazem `chsh`). Příkazový procesor Bourne shell byl nazván podle svého autora, kterým je Steven Bourne. Ten vytvořil původní příkazový procesor pro operační systém Unix a většina příkazových procesorů vytvořených od té doby končí písmeny sh. Tím se indikuje, že další příkazové procesory jsou rozšířením původního příkazového interpretu. Dnes existuje spousta implementací původního příkazového interpretu Bourne shell.

1 Některé operační systémy, jako je OS/2 nebo Windows NT, sice zachovávají v názvech velká a malá písmena, ale jinak mezi nimi nerozlišují. V praxi platí, že se v operačních systémech Unix nepoužívají příkazy nebo obecná jména, která by se lišila pouze malými a velkými písmeny (například různé příkazy cat a Cat).

Jiný typ představují tzv. C shelly (původně je navrhl pan Bill Joy), jež se rovněž hojně používají. Obecně platí, že příkazové interprety typu Bourne shell se používají z důvodu kompatibility s originálním příkazovým interpretem, zatímco C shell se používá pro interaktivní zpracování příkazů.

Příkazové interprety typu C shell se vyznačují tím, že mají lepší vlastnosti pro interaktivní práci, ale mají poměrně nepružné vlastnosti z hlediska programátorského.

Operační systém Linux se distribuuje s příkazovým interpretem `bash`, který byl vytvořen nadací Free Software Foundation. `bash` je zkratkou pro Bourne Again Shell, což tradičně představuje spíše špatnou slovní hříčku typickou pro operační systém Unix. Jedná se o vylepšený příkazový procesor Bourne shell. Má podobné programátorské vlastnosti jako Bourne shell a také má spoustu dobrých vlastností převzatých z příkazového interpretu C shell. `bash` je implicitním příkazovým interpretem používaným v operačním systému Linux.

Když se přihlásíte do systému Linux, pak je to právě program `bash`, který zobrazuje výzvu příkazového řádku. Příkazový procesor `bash` vás bude provázet po celou dobu práce s počítačem.

Typické příkazy operačního systému Unix

Prvním příkazem, se kterým se seznámíme, je příkaz `cat`. Máte-li jej použít, napište `cat` a stiskněte klávesu Enter.

```
/home/larry# cat
```

Jestliže je nyní kurzor na následujícím řádku, pak jste zadali příkaz `cat` správně. Existuje několik způsobů, jak příkaz `cat` zadat. Některé jsou správné, jiné nikoliv.

- Předpokládejme, že jste se spletli:

  ```
  /home/larry# ct
  ct: command not found
  /home/larry#
  ```

Tímto způsobem vás příkazový interpret upozornil na to, že nemůže najít program, který by se jmenoval „`ct`“. Nabídne vám další výzvu příkazového řádku a očekává další příkaz. Připomeňme, že operační systém Unix je citlivý na velká a malá písmena. Příkaz `CAT` je rovněž chybný.

- Před příkaz můžete uvést libovolný počet mezer, například:[2]

  ```
  /home/larry# _ cat
  ```

  Takto zadaný příkaz je v pořádku a program `cat` se spustí.

- V příkazovém řádku také můžete pouze stisknou klávesu Enter. Ničeho se neobávejte, nic se nestane.

Předpokládejme, že jste spustili program `cat`. Asi jste zvědavi, co dělá. Pokud si myslíte, že se jedná o nějakou hru, pak asi budete zklamáni. Tento program je velice užitečným nástrojem, i když se to na první pohled nezdá. Napište jakýkoliv text a stiskněte Enter. Pak na obrazovce uvidíte:

```
/home/larry# cat
Help! I'm stuck in a Linux program!
Help! I'm stuck in a Linux program!
```

2 _ indikuje mezeru.

(Skloněným písmem je označeno to, co jsem v programu `cat` napsal.) Zdá se, že program pouze kopíruje text. To je někdy užitečné, ale program `cat` toho umí mnohem více. Pro tento okamžik program cat ukončíme a později si uvedeme příklady, kdy je mnohem užitečnější.

K ukončení většiny příkazů operačního systému Unix se používá kombinace kláves Ctrl+D[3]. Touto kombinací odešlete znak konce souboru EOF (end-of-file). Alternativně lze znak EOF považovat za konec textu, avšak v této knize jej budeme považovat za konec souboru. Znak EOF signalizuje programům operačního systému Unix, že jste ukončili zadávání dat. Pro program `cat` to znamená, že další data nemá zpracovávat a ukončí se.

Také si můžete vyzkoušet program sort. Jak napovídá jeho název, jedná se o program pro třídění. Jestliže zapíšete několik řádků a pak stisknete Ctrl+D, zobrazí se vám tyto řádky uspořádány. Programy tohoto druhu se nazývají **filtry**. Fungují tím způsobem, že nejdříve akceptují nějaký text, provedou s ním nějaké operace (například třídění) a pak modifikovaný text vypíší na obrazovce (nebo na jiném výstupním zařízení). Programy cat a sort jsou poněkud neobvyklé filtry, cat je neobvyklý v tom, že přečte text a neprovádí v něm žádné změny. Program `sort` je neobvyklý v tom, že čte řádky a neprovádí nic, dokud nepřečte znak EOF. Mnoho filtrů zpracovává data řádek po řádku – přečtou řádek, provedou nějaké modifikace a zobrazí řádek modifikovaný.

Pomozte si sami

V operačních systémech Unix existuje příkaz `man`,[4] jenž zobrazuje tzv. manuálové stránky. Originální manuálové stránky jsou pouze v anglickém jazyce, ale na jejich překladu se již pracuje. Zadejte například příkaz:

```
/home/larry# man cat
CAT(1)                                                    CAT(1)
NAME
   cat - concentrate files and print on the standard output
SYNOPSIS
   cat [-benstuvAET] [--number] [--number-nonblank]
   [--squeeze-blank] [--show-nonprinting] [--show-ends]
   [--show-tabs] [--show-all] [--help] [--version] [--file...]
DESCRIPTION
   This manual page documents the GNU version of cat.       cat
```

Uvedený výpis představuje asi jednu stránku informací o příkazu `cat`. Nepředpokládejte, že budete okamžitě všemu rozumět. Manuálové stránky zpravidla předpokládají, že má jejich čtenář jisté znalosti o operačním systému Unix, tedy znalosti, které vy zatím nemáte. Když čtete manuálovou stránku, pak se v dolním řádku objeví blok s textem jako „`--more--`“ nebo „`Line 1`“. Jedná se o nápovědu, že k danému tématu existuje více informací.

Man po zobrazení první stránky vyčkává, co se rozhodnete dělat dále. Chcete-li pokračovat dále v prohlížení manuálových stránek, stiskněte klávesu Spacebar – pak se vám zobrazí následující stránka. Pokud chcete program man ukončit, stiskněte klávesu Q. Pak se vrátíte do příkazového řádku příkazového procesoru, který bude očekávat zadání dalšího příkazu.

---

3 Podržte stisknutou klávesu Ctrl a stiskněte klávesu D.

4 Program `man` také zobrazí informace o systémovém volání, podprogramu, formátu souboru a mnoho dalších informací. V originálních verzích operačního systému Unix obsahuje manuálová stránka stejné informace jako tištěná dokumentace. Pro tento okamžik se spokojíme s nápovědou k syntaxi příkazu.

Programu man lze také předat jisté argumenty, jež řídí jeho funkce. Předpokládejme například, že si chcete přečíst informace o všem, co souvisí s formátem Postscript (Postscript je řídící jazyk pro tisk na tiskárnách od firmy Adobe). Pak zadejte příkaz `man -k ps` nebo `man -k Postscript`. Zobrazí se vám seznam všech příkazů, systémových volání a další dokumentace pojednávající o tomto tématu. Tento postup je velmi užitečný v případě, kdy chcete nalézt nějaký nástroj, ale nevíte, kde jej hledat, nebo nevíte, zda vůbec existuje.

Ukládání informací

Programy patřící do kategorie filtrů jsou velmi užitečné, zejména pokud patříte mezi pokročilejší uživatele. Zůstává zde však jeden problém. Jak ukládat, uchovávat a aktualizovat informace? Za tímto účelem nabízí operační systém Unix **soubory** a **adresáře**.

Adresář je něco jako pořadač, který místo svazků papíru obsahuje soubory. Velké pořadače mohou obsahovat několik dalších menších pořadačů. V operačním systému Unix se systém adresářů a souborů nazývá souborový systém. Zpočátku je každý souborový systém tvořen jediným adresářem, který se nazývá kořenový nebo také hlavní adresář. Uvnitř tohoto adresáře se mohou vyskytovat další adresáře a uvnitř nich se opět mohou vyskytovat adresáře (ty se někdy nazývají podadresáře). V každém adresáři mohou být uloženy soubory.[5]

Každý soubor a každý adresář má své vlastní jméno. Jméno může být buď krátké a může se shodovat se jménem adresáře nebo souboru uloženého jinde, nebo dlouhé (tzv. jméno včetně cesty), které je v rámci souborového systému na daném disku unikátní. Příkladem krátkého jména může být `joe`, zatímco odpovídajícím dlouhým jménem (t.j. jménem včetně cesty) je `/home/larry/joe`. Posloupnost znaků v dlouhém jménu ukončená znakem / se nazývá cesta. Cesta může být dekódována do posloupnosti adresářů. Následující příklad ilustruje, jak systém dospěje k souboru `/home/larry/joe`:

```
/home/larry/joe
```

První znak / indikuje kořenový adresář.

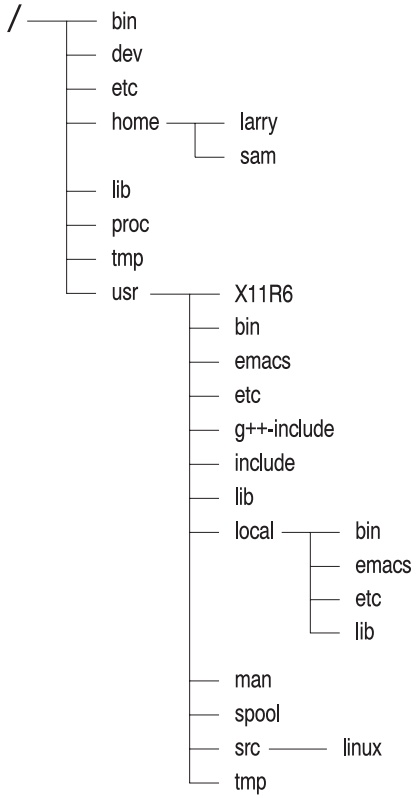
Následuje adresář se jménem `home`, který je podadresářem kořenového adresáře.

Další je adresář `larry`, který je podadresářem adresáře `home`.

Soubor `joe` je uvnitř adresáře `larry`. Cesta může odkazovat buď na adresář, nebo na soubor – `joe` tedy může být jak adresářem, tak i souborem. Všechny položky před krátkým jménem musejí být adresáře.

Existuje jednoduchý způsob, jak strukturu adresářů vizualizovat. Tato vizualizace se nazývá stromový diagram a většina uživatelů asi zná stromové diagramy z operačního systému DOS (vytvářejí je známé programy pro správu souborů, jako je NORTON COMMANDER). Stromový diagram můžete vytvořit programem tree nebo v programu Midnight Commander, linuxové obdobě Norton Commandera. Typickou stromovou strukturu adresářů používanou v operačním systému Linux znázorňuje obrázek 4.1. Poznamenejme, že zde uvedený diagram není kompletní. Úplná struktura operačního systému Linux obsahuje přes 8000 souborů! Zrovna tak platí, že zde mohou být uvedeny adresáře, jež se nevyskytují ve vaší instalaci a naopak, ve vaší instalaci mohou být adresáře, které v uvedeném obrázku uvedeny nejsou.

5 Počet podadresářů může být omezen, ale v některých systémech nemusí. V operačním systému Linux jsem bez problémů vytvořil podadresáře až do úrovně 10.



Obrázek 4.1 – Typická (zkrácená) stromová struktura adresářů v operačním systému Unix

Prohledávání adresářů pomocí příkazu ls

Nyní máte jistě představu o tom, jak jsou v operačním systému Unix organizovány adresáře a soubory. Samozřejmě budete potřebovat nějaké nástroje, pomocí nichž budete moci se soubory a adresáři manipulovat. Prvním důležitým nástrojem je příkaz `ls`. Jedná se o příkaz, který zobrazuje seznam souborů. Nyní zadejte příkaz `ls` na úrovni příkazového řádku:

```
/home/larry# ls
/home/larry#
```

Žádný seznam se nezobrazil, což je v pořádku. Operační systém Unix pracuje přesně. Příkazu `ls` nebyly předány žádné parametry, proto se žádný seznam nezobrazil.

Jak jsme se již zmínili, při instalaci operačního systému Linux se nainstaluje více než 8000 souborů. Kde jsou tyto soubory uloženy? Abychom pochopili používání příkazu `ls`, musíme si vysvětlit pojem aktuální adresář. Podle výzvy příkazového řádku poznáte, že aktuálním adresářem je `/home/larry`. Zde však nejsou uloženy žádné soubory, proto příkaz `ls` nezobrazil žádný seznam. Zkuste si zobrazit seznam souborů a adresářů uložených v hlavním adresáři:

```
/home/larry# ls /
bin     etc     lostfound       proc    tmp
boot    home    misc            root    usr
dev     lib     mnt             sbin    var
/home/larry#
```

V předcházejícím příkazu „ls /“ jsme programu ls předali parametr „/“. Prvním slovem v příkazu je jméno příkazu a další slova jsou parametry. Obecně platí, že parametry modifikují funkce příkazů. V případě příkazu ls se jako první parametr uvádí jméno adresáře, jehož obsah má příkaz ls zobrazit. Některé příkazy mají speciální parametry, kterým se říká volby (options) nebo přepínače (switches). Vyzkoušejte si následující příkaz:

```
/home/larry# ls -F /
bin/    etc/    lostfound/      proc/   tmp/
boot/   home/   misc/           root/   usr/
dev/    lib/    mnt/            sbin/   var/
/home/larry#
```

Parametr `-F` se nazývá **volba**. Volba je speciální parametr, který začíná pomlčkou a modifikuje způsob provádění programu. V případě příkazu `ls` je parametr `-F` určen k rozlišení položek seznamu na adresáře, speciální soubory, programy a ostatní soubory. Každá položka končící znakem / je adresář. Později budeme hovořit o příkazu ls, který je opravdu velmi univerzální, podrobněji.

Nyní byste si měli udělat malé cvičení. Především se naučte, jak příkaz ls pracuje. Vyzkoušejte si zobrazit obsahy jiných adresářů podle obrázku 4.1. Některé z nich budou prázdné, jiné budou obsahovat spoustu souborů. Doporučuji, abyste si vyzkoušeli příkaz ls s parametrem -F i bez něj. Například adresář /usr/local by mohl vypadat takto:

```
/home/larry# ls /usr/local
Acrobat3        com     etc     include lib     man     share   teTeX
bin             doc     games   info            libexec sbin    src
/home/larry#
```

Druhé cvičení bude více obecné. Spousta příkazů v operačním systému Unix se spouští podobně jako příkaz `ls`. Také mají nějaké parametry a nějaké volby, jež se zpravidla uvádějí jako jednoznakové řetězce za pomlčkou. Na rozdíl od příkazu `ls` však některé příkazy vyžadují parametry a/nebo volby. K vyjádření syntaxe příkazů budeme používat následující formu:

`ls [-aRF] [`*`adresář`*`]`

Když budeme popisovat nový příkaz, použijeme k jeho definici podobný syntaktický zápis. Prvním slovem je příkaz (v tomto případě `ls`). To, co následuje za příkazem, jsou parametry. Volitelné parametry jsou uzavřeny do hranatých závorek. Tzv. meta-proměnné jsou vyznačeny skloněným písmem – jsou to slova, která musejí být nahrazena skutečnými parametry. V uvedeném příkladu je meta-proměnnou *`adresář`*. Ta by měla být nahrazena jménem skutečného adresáře.

V případě voleb platí jiná pravidla. V syntaktické definici příkazu jsou uzavřeny v hranatých závorkách. V příkazu můžete použít kteroukoliv volbu nebo kteroukoliv jejich kombinaci. V případě příkazu `ls` máte tedy možnost použít osm kombinací voleb. Porovnejte výstup příkazů `ls -R` a `ls -F`.

Aktuální adresář a příkaz cd

`pwd`

Používání adresářů může být poněkud těžkopádné – asi by se vám nelíbilo, kdybyste museli pokaždé vypisovat celou cestu, kdykoliv byste si chtěli zpřístupnit nějaký soubor. V operačním systému Unix je zaveden pojem aktuálního adresáře. V příkladech, které jsme doposud použili, je aktuálním adresářem `/home/larry`. Pokud chcete zjistit, jaký je skutečný aktuální adresář, zadejte příkaz `pwd` (print working directory). V některých případech zobrazuje výzva příkazového řádku i jméno počítače. To je užitečné pouze tehdy, když je počítač zapojen do sítě, ve které se vyskytuje více počítačů.

```
mousehouse> pwd
/home/larry
mousehouse>
```

`cd [`*`adresář`*`]`

Jak jste se mohli přesvědčit, příkaz `pwd` zobrazí aktuální adresář.[6] Příkaz `pwd` je tedy velmi jednoduchý. Většina příkazů v operačním systému Unix funguje implicitně v aktuálním adresáři, který lze změnit pomocí příkazu `cd`. Vyzkoušejte si například příkazy:

```
/home/larry# cd /home
/home# ls -F
larry/ sam/ shutdown/ steve/ user1/
/home#
```

Pokud vynecháte volitelný parametr *`adresář`*, pak se navrátíte do vašeho domovského adresáře, kterým je v našem případě `/home/larry`. Jinak se dostanete do adresáře specifikovaného prostřednictvím parametru. Například:

```
/home# cd
/home/larry# cd /
/# cd home
/home# cd /usr
/usr# cd local/bin
/usr/local/bin#
```

Jak jste si asi všimli, příkaz `cd` umožňuje specifikovat buď **absolutní cestu,** nebo **relativní cestu**. Absolutní cesta začíná znakem / a musí obsahovat všechny adresáře vedoucí k výslednému adresáři, do kterého se chcete přepnout. Relativní cesta se vztahuje k vašemu aktuálnímu adresáři. V předcházejícím příkladu jsme v adresáři `/usr` zadali relativní adresář `local/bin` – `local` je adresář pod adresářem `usr` a `bin` je adresář pod adresářem `local`. Podobně jsme použili relativní adresář v příkazu `cd home`.

V operačním systému Unix (podobně jako v operačním systému DOS) existují dva speciální adresáře, jež se používají pouze jako relativní. Jsou to adresáře „`.`“ a „`..`“. Adresář „`.`“ specifikuje aktuální adresář a adresář „`..`“ specifikuje nadřazený adresář. Jedná se tedy o zkratky, které existují v každém adresáři, ale ne vždy mají přesně ten význam, jak jsme jej popsali. Například hlavní adresář má jako nadřazený adresář sebe sama.

6 Někdy se také používá termín pracovní adresář. V této knize se budeme držet termínu aktuální adresář.

Soubor `./chapter-1` v aktuálním adresáři by měl být identický se souborem `chapter–1`. Některé příkazy vyžadují zadání typu `./chapter-1`, ale to není častý případ. Ve většině případů jsou `./chapter-1` a `chapter-1` identické specifikace souboru.

Adresář „`..`“ je velmi užitečný při „pohybu“ v adresářové struktuře směrem nahoru:

```
/usr/local/bin# cd ..
/usr/local# ls -F
archives/      bin/    emacs@ etc/    ka9q/   lib/    tcl@
/usr/local# ls -F ../src
cweb/          linux/ xmris
/usr/local#
```

V tomto případě jsme se přepnuli do nadřazeného adresáře pomocí `cd ..` a pak jsme zobrazili obsah adresáře `/usr/src` z adresáře `/usr/local` pomocí příkazu `ls -F ../src`.

Pro domovský adresář existuje zkratka ~. Vyzkoušejte si následující příkaz:

```
/usr/local# ls -F ~/
/usr/local#
```

Opět jsme se přesvědčili, že v domovském adresáři nejsou žádné soubory. Užitečnost zkratky ~ pro domovský adresář vynikne zejména v případě, kdy začneme se soubory manipulovat.

Vytváření a odstraňování adresářů

`mkdir` *`adresář1`* [*`adresář2 ... adresářN`*]

Vytváření vlastních adresářů je v operačním systému Unix opravdu jednoduché a představuje velmi užitečný nástroj pro organizaci a údržbu souborů. K vytvoření adresáře slouží příkaz `mkdir` (make directory).

Uveďme si jednoduchý příklad, abychom pochopili, jak příkaz `mkdir` funguje.

```
/home/larry# ls -F
/home/larry# mkdir report-1993
/home/larry# ls -F
report-1993/
/home/larry# cd report-1993
/home/larry/report-1993#
```

Příkaz `mkdir` může akceptovat více než jeden parametr. Předpokládá, že každý z nich představuje jméno adresáře, který se má vytvořit. Opět můžete specifikovat celou cestu nebo relativní adresář. V předcházejícím příkladu jsme použili relativní adresář `report-1993`.

```
/home/larry/report-1993# mkdir /home/larry/report-1993/chap1
  ~/report-1993/chap2
/home/larry/report-1993# ls -F
chap1/  chap2/
/home/larry/report-1993#
```

`rmdir` *`adresář1`* [*`adresář2 ... adresářN`*]

Jakýmsi opakem k příkazu `mkdir` je příkaz `rmdir` (remove directory). `rmdir` funguje přesně jako `mkdir`. Podívejte se na následující příklad:

```
/home/larry/report-1993# rmdir chap1 chap3
rmdir: chap3: No such file or directory
/home/larry/report-1993# ls -F
chap2/
/home/larry/report-1993# cd ..
/home/larry# rmdir report-1993
rmdir: report-1993: Directory not empty
/home/larry#
```

Jak jste si zřejmě všimli, příkaz `rmdir` odmítnul odstranit adresář, který buď neexistuje nebo není prázdný. Uvědomte si, že adresář `report-1993` obsahuje podadresář `chap2`, proto nelze adresář `report-1993` odstranit.

Manipulace se soubory

Práce s adresáři je sice zajímavá, ale stále neřeší náš problém s ukládáním, aktualizací a udržováním informací. Tento problém se řeší prostřednictvím **souborů**.

Vytváření a editování souborů se budeme věnovat v několika následujících kapitolách.

Základními příkazy pro manipulaci se soubory jsou v operačním systému Unix příkazy `cp` (copy), `mv` (move) a `rm` (remove).

Příkaz pro kopírování souborů cp

```
cp [-i] zdroj cíl
cp [-i] soubor1 soubor2 ... souborN cílový adresář[7]
```

Příkaz `cp` je v operačním systému Unix velmi důležitým a výkonným příkazem. V jedné sekundě vám umožní přemístit tolik informací, kolik jich středověký mnich přemístil za celý rok.

Jestliže nemáte na svém pevném disku dostatek prostoru, buďte při používání příkazu `cp` opatrní. Nikdo nechce vidět zprávu „Disk full“, když pracuje s důležitými soubory. Příkaz `cp` je dále schopen přepsat důležité soubory bez jakéhokoliv varování – tomuto nebezpečí se budeme věnovat podrobněji.

Nejprve se zaměříme na první definici příkazu `cp`. Prvním parametrem příkazu `cp` je soubor, který se má kopírovat (tzv. zdrojový soubor, source file). Druhým parametrem je soubor, který má být kopií prvního (tzv. cílový soubor, destination file). Při kopírování můžete vytvářet soubory s novým jménem nebo můžete soubory se stejným jménem kopírovat do jiných adresářů. Vyzkoušejte si následující příklady:

```
/home/larry# ls -F /etc/passwd
/etc/passwd
/home/larry# cp /etc/passwd .
/home/larry# ls -F
passwd
/home/larry# cp passwd frog
/home/larry# ls -F
frog passwd
/home/larry#
```

7 Příkaz `cp` má v předloze dva řádky, neboť význam druhého parametru může být rozdílný v závislosti na počtu parametrů.

Prvním příkazem `cp` se z adresáře `/etc` zkopíroval soubor `passwd` (jenž obsahuje jména všech uživatelů systému Unix spolu se zakódovanými hesly) do mého domovského adresáře. Toto platí pouze za předpokladu, že nepoužíváme tzv. stínová hesla, kdy jsou zakódovaná hesla uložena v souboru `/etc/shadow`. Příkaz `cp` neruší zdrojový soubor, proto jsem neudělal nic, co by nějak poškodilo systém. Nyní existují v mém systému dvě kopie souboru `passwd`, jedna v adresáři `/etc` a druhá v mém domovském adresáři `/home/larry` a obě kopie mají jméno `passwd`.

Třetí kopii jsem vytvořil příkazem `cp passwd frog`. Nyní jsou v mém systému tři kopie souboru: `/etc/passwd`, `/home/larry/passwd` a `/home/larry/frog`. Posledně vytvořená kopie má jiné jméno, ale obsahuje stejná data.

Příkaz `cp` je schopen kopírovat soubory mezi adresáři, a to v tom případě, kdy je prvním parametrem jméno souboru a druhým parametrem je jméno adresáře. Pak zůstane jméno cílového souboru shodné se jménem zdrojového souboru.

Kopírovat soubory a přitom měnit jejich jména lze jen tehdy, když se v příkazu `cp` jako parametry uvedou jména souborů. S příkazem `cp` je spojeno jedno nebezpečí. Když například zadáte příkaz `cp /etc/passwd /etc/group`, vytvoří příkaz `cp` nový soubor `group`, jehož obsah bude identický s obsahem souboru `/etc/passwd`. Pokud by však soubor `/etc/group` již existoval, pak jej přepíšete bez jakéhokoliv varování a bez možnosti starou verzi souboru `/etc/group` obnovit.

Podívejme se na další příklad použití příkazu `cp`:

```
/home/larry#   ls -F
frog passwd
home/larry#    mkdir passwd_version
home/larry#    cp frog passwd passwd_version
home/larry#    ls -F
frog           passwd  passwd_version/
home/larry#    ls -F passwd_version
frog passwd
home/larry#
```

Jak jsem nyní použil příkaz `cp`? Je evidentní, že příkaz `cp` může akceptovat více než dva parametry (nyní jsem příkaz `cp` použil podle druhé definice). Příkaz `cp` v předcházejícím příkladu zkopíroval všechny uvedené soubory (`frog` a `passwd`) do adresáře `passwd_version`, který je uveden jako poslední parametr. V podstatě platí, že příkaz `cp` je schopen akceptovat jakýkoliv počet parametrů, přičemž prvních n-1 považuje za jména souborů a poslední považuje za jméno adresáře, do kterého se mají uvedené soubory kopírovat.

Jestliže kopírujete v daném okamžiku více než jeden soubor, pak jej nemůžete přejmenovat – kopírované soubory si uchovají svá původní jména. Co se stane, když zadáte příkaz `cp frog passwd toad`, kde `frog` a `passwd` jsou soubory a přitom `toad` není adresář? Vyzkoušejte si takový příkaz a uvidíte.

Odstranění souborů pomocí příkazu rm

`rm [-i] soubor1 soubor2 ... souborN`

Nyní, když jsme se naučili, jak kopírovat a vytvářet soubory (později si probereme jiné způsoby vytváření souborů), bude jistě užitečné vědět, jak soubory rušit. Ve skutečnosti je to velmi jednoduché. Příkaz pro odstraňování souborů má název `rm` a zruší všechny soubory, jejichž jména se uvedou jako parametry.

Například:

```
/home/larry# ls -F
frog passwd passwd_version/
/home/larry# rm frog toad passwd
rm: toad: No such file or directory
/home/larry# ls -F
passwd_version/
/home/larry#
```

Jak asi vidíte, příkaz `rm` je velmi nevlídný. Nejen, že se vás nezeptá, zda má uvedené soubory skutečně zrušit, ale provede zrušení existujících souborů, i když vlastně příkaz nebyl správně zadán (soubor `toad` neexistoval). Příkaz `rm` může být skutečně nebezpečný. Podívejme se na rozdíl mezi následujícími dvěma posloupnostmi příkazů:

```
/home/larry# ls -F
toad frog/
/home/larry# ls -F frog
toad
/home/larry# rm frog/toad
/home/larry#
```

```
/home/larry# rm frog toad
rm: frog is a directory
/home/larry# ls -F
frog/
/home/larry#
```

Pátý řádek v první posloupnosti a první řádek v druhé se liší jediným znakem. Uvedený příklad má ilustrovat, že opomenutí jediného znaku může v operačním systému Unix způsobit, že provedete něco úplně jiného, než jste původně chtěli. Proto je velmi důležité, abyste raději několikrát zkontrolovali zapsaný příkaz, než stisknete klávesu Enter.

Přesouvání souborů pomocí příkazu mv

`mv [-i]` *`starý nový`*
`mv [-i]` *`soubor1 soubor2 ... souborN adresář`*

Nakonec se podívejme na příkaz pro tzv. přesouvání souborů. Příkaz má název `mv` a je podobný příkazu `cp`. Na rozdíl od příkazu `cp` však zdrojové soubory po jejich zkopírování odstraňuje. Příkaz `mv` je tedy jakousi kombinací příkazů `cp` a `rm`. Uveďme si na následujícím příkladu, jak příkaz `mv` funguje:

```
/home/larry# cp /etc/passwd .
/home/larry# ls -F
passwd
/home/larry# mv passwd frog
/home/larry# ls -F
frog
/home/larry# mkdir report
/home/larry# mv frog report
/home/larry# ls -F
```

```
report/
/home/larry# ls -F report
frog
/home/larry#
```

Jak jste si pravděpodobně všimli, příkaz `mv` soubor přejmenuje, pokud je druhým parametrem jméno souboru. Jestliže je druhým parametrem jméno adresáře, pak se soubor přesune do nového adresáře a přitom zůstane uchováno jeho původní jméno.

Podobně jako v případě příkazu `cp` musíte být při používání příkazu `mv` velmi opatrní. Příkaz neprovádí kontrolu, zda cílový soubor existuje nebo ne. Kdyby například v mém adresáři `report` již existoval soubor `frog`, pak by se příkazem `mv frog report` nejdříve zrušil soubor `/report/frog` a pak by se nahradil obsahem souboru `~/frog`.

Ve skutečnosti existuje možnost, jak přimět příkazy `rm`, `cp` a `mv`, aby vás upozorňovali na možnost přepsání či zrušení souborů. Všechny tyto příkazy akceptují volbu `-i`. Po uvedení této volby bude uživatel před každým zrušením souboru upozorněn. Dokonce můžete použít tzv. **alias** (druhé jméno) a zařídit, aby vás například příkaz `rm` upozorňoval na zrušení souboru, aniž uvedete volbu `-i`. O tom však budeme diskutovat až v oddíle Vytváření aliasů v kapitole 9.

KAPITOLA 5

Systém X Window

Tato kapitola se týká pouze uživatelů systému X Window. Jestliže máte před sebou barevnou grafickou obrazovku s mnoha okny a kurzorem, kterým lze pohybovat pouze prostřednictvím myši, pak jste zřejmě uživateli systému X Window. Pokud však máte před sebou obrazovku s bílým textem na černém pozadí, pak momentálně systém X Window nemáte aktivován. Budete-li chtít systém X Window aktivovat, přečtěte si následující oddíl.

Spuštění a ukončení systému X Window

Spuštění systému X Window

Systém X Window můžete spustit i v případě, kdy se automaticky nenastartuje ihned po zavedení operačního systému. Existují dva příkazy, kterými lze nastartovat systém X Window: `startx` a `xinit`. Nejdříve vyzkoušejte příkaz `startx`. Pokud příkazový procesor ohlásí, že takový příkaz neexistuje, pak zkuste `xinit`. Pokud ani v tomto případě nedojde ke spuštění systému X Window, pak jej pravděpodobně nemáte nainstalován. Prostudujte si příslušnou dokumentaci.

Jestliže se příkaz spustí, ale po nějakém čase se opět vrátíte do příkazového řádku příkazového procesoru, pak je systém X Window sice instalován, ale není nakonfigurován. I v tomto případě si budete muset důkladně přečíst dokumentaci k instalaci systému X Window.

Ukončení systému X Window

V závislosti na tom, jak je váš systém X Window konfigurován, existují dva možné způsoby, jak systém X Window ukončit. První závisí na tom, zda váš systém X Window řídí správce oken či nikoliv. Pokud ano, můžete systém X Window ukončit prostřednictvím nabídky (viz oddíl Nabídky). Chcete-li si zobrazit nabídku, klepněte myší na pozadí pracovní plochy.

Pak se vám mezi jinými objeví položka „Exit Window Manager“ nebo „Exit X“ nebo jiná položka obsahující slovo „Exit“. Pokuste se najít tuto položku (počítejte s tím, že zde může existovat více než jedna nabídka – vyzkoušejte různá tlačítka myši) a ukončete systém X Window.

Druhá metoda ukončení systému X Window spočívá v tom, že se k řízení systému X Window využije speciálního okna `xterm`. V takovém případě byste měli najít okno nazvané „login“ nebo „system `xterm`“. Přesuňte kurzor myši do tohoto okna a zadejte příkaz `exit`.

Jestliže se systém X Window automaticky spustil, když jste se přihlásili do systému, pak by vás jedna z výše uvedených metod ukončení systému X Window měla automaticky odhlásit. Jestliže jste však systém X Window nastartovali z příkazového řádku, pak se do tohoto příkazového řádku po ukončení systému X Window opět vrátíte (pokud se chcete odhlásit, zadejte příkaz `logout`).

Co je systém X Window?

Systém X Window je distribuované grafické uživatelské prostředí původně vyvinuté v akademické instituci Massachusetts Institute of Technology. Později byl předán skupině distributorů s názvem „The X Consortium“. Zde se dodnes udržuje a nadále vyvíjí.

Systém X Window (někdy zkracován jako „X“[1]) se každých několik málo let distribuuje v nové verzi, uváděné jako „release“. Poslední verze má označení X11R6, tedy „release“ 6. Číslo 11 značí hlavní verzi systému X Window, avšak toto označení je trvalé, protože se nová verze neplánuje.

V souvislosti se systémem X Window existují dva důležité pojmy, se kterými byste se měli seznámit. Program, jenž běží pod systémem X Window, se nazývá **klient**. Například `xterm` je klient, který se spustí v okamžiku, kdy se přihlašujete do systému. Program, který poskytuje služby ostatním programům typu klient, se nazývá **server**. Server například kreslí okno pro program `xterm` a zajišťuje komunikaci s uživatelem.

Protože jsou server a klient dva odlišné programy, je možné zajistit, aby server běžel na jednom počítači a klient na jiném. Vzhledem k tomu, že grafické uživatelské rozhraní dodržuje jistý standard, můžete v systému X Window spustit program na vzdáleném počítači (třeba na druhém konci světa) a přitom můžete grafické rozhraní tohoto programu vidět na svém počítači.

Dalším důležitým termínem, se kterým byste se měli seznámit, je **správce oken** (window manager). Jedná se o speciální program typu klient, který určuje pozici jednotlivých oken na pracovní ploše systému X Window a řídí způsoby, kterými uživatel například přemisťuje okna. Samotný server v podstatě pro uživatele nic nedělá. Pouze představuje rozhraní mezi uživatelem a klientem.

Co se nachází na pracovní ploše X Window

Když poprvé nastartujete systém X Window, zároveň se nastartuje řada dalších programů. Jako první se nastartuje server. Další v pořadí se nastartují některé programy typu klient – jejich pořadí a typ závisí na distribuci a není standardizován. Je však pravděpodobné, že mezi těmito klienty je správce oken (buď program `fvwm` nebo `twm`), terminálové okno `xterm` a hodiny `xclock`.

Program XClock

`xclock [-digital] [-analog] [-update` *seconds*`] [-hands` *color*`]`

Program `xclock` (hodiny) funguje přesně tak, jak asi očekáváte. Zobrazuje ciferník s vteřinovou ručičkou a malou i velkou ručičkou v malém okně.

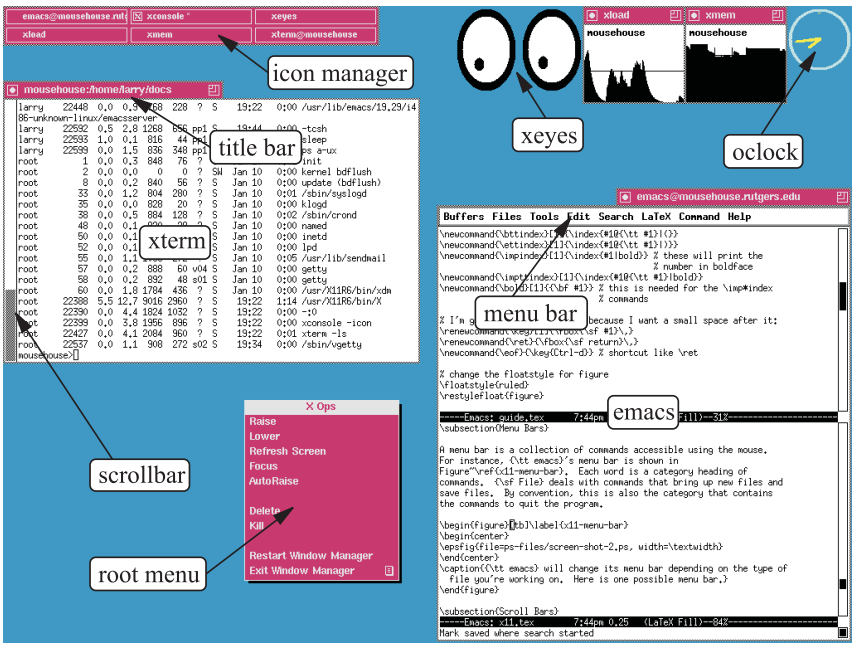
Prostřednictvím myši nemůžete vlastnosti okna s hodinami příliš modifikovat. Nanejvýš můžete měnit jeho velikost. Pokud však program xclock spustíte z příkazového řádku, pak můžete prostřednictvím jeho parametrů nastavit různé vlastnosti. Když zadáte příkaz `xclock -digital`, zobrazí se digitální hodiny. Pomocí parametru `-update` můžete nastavit, zda se má vteřinová ručička aktualizovat každou sekundu (`-update 1`) nebo například každých pět sekund (`update -5`).

Chcete-li vědět více o programu `xclock` a jeho parametrech, podívejte se do manuálových stránek. Stačí v příkazovém řádku zadat `man xclock`. Budete-li chtít spustit více programů `xclock` najednou, přečtěte si oddíl Současný běh úloh, ve kterém budeme diskutovat o provozování více úloh současně.

Jestliže máte program `xclock` spuštěn na popředí (což je běžný způsob v provozování programů) a chcete jej ukončit, stiskněte kombinaci kláves Ctrl-C.

---

1 Existuje několik přijatelných způsobů, jak nazývat systém X Window. Nejčastěji se vyskytuje zkratka „X“ nebo název „X Window“.



Obrázek 5.1 – Příklad standardní obrazovky systému X Window. V tomto příkladu uživatel spustil program twm. Standardní hodiny byly nahrazeny jiným programem zvaným `oclock`.

Program xterm

Okno s příkazovým řádkem uvnitř (tedy v našem případě okno s výzvou `/home/larry#`) je řízeno programem, jenž se nazývá xterm. Což je zvláštní a přitom komplikovaný program. Na první pohled se zdá, že toho moc neumí, ale jeho prostřednictvím lze vykonat spoustu užitečné práce. Emuluje terminál, a proto v něm lze spouštět všechny textově orientované aplikace operačního systému Unix. Také obsahuje vyrovnávací paměť uchovávající jistou množinu příkazů – proto se můžete snadno vracet k dříve zadaným příkazům (podrobněji se tímto tématem budeme zabývat v oddílu Posuvné lišty).

Celá tato kniha je zaměřena na příkazy zadávané v příkazovém řádku terminálu, proto je program xterm v systému X Window tak důležitý. Chcete-li něco zadávat v okně tohoto programu xterm, musí být toto okno aktivní. To znamená, že musíte do okna terminálu přesunout kurzor myši. Způsob, kterým se nastavuje aktivní okno obecně, závisí na správci oken.

Program xterm představuje jednu z možností, jak v systému X Window spouštět více programů současně. Programy běžící v systému X Window jsou standardními programy operačního systému Unix, proto mohou být spuštěny z terminálového okna. Protože dlouhodobé programy spouštěné z terminálového okna zablokují program `xterm` po celou dobu svého běhu, spouští se takové programy zpravidla na pozadí. Více informací o tomto tématu uvedeme v oddílu Současný běh úloh.

Správce oken

V operačním systému Linux existují dva běžně používané programy pro správu oken.* První z nich, twm, je zkratkou pro „Tab Window Manager“. Druhý program je menší a má název fvwm („F(?) Virtual Window Manager“). Oba programy jsou konfigurovatelné, což znamená, že si uživatel může definovat význam ovládacích klíčů a způsob práce s myší.

Konfiguraci programu twm je věnován oddíl Konfigurace programu twm a konfiguraci programu fvwm probereme v oddílu Konfigurace programu fvwm.

Jak se vytvářejí nová okna

Při vytvoření nového okna existují tři možné varianty, jež realizuje správce oken. V poslední době se objevuje spousta nových správců oken, např. `fvwm95`, `gnome` nebo `kwm`. Správce oken může být nakonfigurován tak, že se zobrazí rám nového okna a uživatel má možnost okno umístit kamkoliv na obrazovku. Tento způsob umístění okna se nazývá **ruční umístění** (manual placement). Jestliže se před vámi objeví rám okna, jednoduše jej pomocí kurzoru myši nastavte na požadovanou pozici a stiskněte levé tlačítko.

Je také možné, že správce oken umístí nové okno na pracovní ploše systému X Window dle „svého uvážení“. Takové umístění okna se nazývá **náhodné umístění** (random placement).

Třetí varianta spočívá v tom, že se okno náležící jisté aplikaci pokaždé otevře na stejné pozici. Správce oken může být nakonfigurován tak, aby pro vybrané aplikace otevíral okna s předem definovanou velikostí na předem definované pozici.

Jako příklad může sloužit program `xclock`, kdy asi budete chtít zobrazovat hodiny v horním levém rohu pracovní plochy systému X Window.

Aktivní okno

Správce oken řídí spoustu důležitých nastavení. Vás bude určitě zajímat, jak nastavit dané okno (například terminálové okno) tak, abyste v něm mohli zadávat příkazy. Aktivní okno je nejčastěji určeno pozicí kurzoru myši. Jestliže je kurzor myši umístěn na jednom z terminálových oken,[2] pak v tomto okně můžete zadávat příkazy z klávesnice. V tom spočívá rozdíl od jiných operačních systémů, jako je OS/2, Microsoft Windows nebo Macintosh, kde musíte na dané okno (chcete-li, aby bylo aktivní) klepnout myší. V systému X Window obvykle platí, že když kurzor myši opustí rámec okna, pak v tomto okně již nemůžete zadávat příkazy.

Poznamenejme, že oba správce oken (tedy program twm a fvwm) lze konfigurovat tak, že se aktivace oken bude realizovat stejně jako například v operačním systému OS/2. Popis konfigurace správce oken najdete v příslušné dokumentaci, nebo můžete požadované konfigurace dosáhnout metodou pokusů a omylů.

Přemisťování oken

V systému X Window lze rovněž konfigurovat způsob, kterým mají být okna přemisťována. Moje osobní konfigurace programu twm obsahuje tři způsoby, jak posouvat okna po pracovní ploše. Nejznámější způsob spočívá v tom, že se kurzor myši umístí na titulní (hlavní) lištu okna, stiskne kterékoliv tlačítko (levé, pravé nebo střední[3]), a pak se okno přesune na novou pozici. Je ovšem velmi pravděpodobné, že je váš systém X Window konfigurován tak, aby se okna přemisťovala pomocí levého tlačítka (tak, jak je tomu u ostatních operačních systémů).

---

2 V daném okamžiku můžete spustit několik programů `xterm` současně.

3 Některé osobní počítače mají dvoutlačítkovou myš. Pak můžete prostřední tlačítko simulovat současným stisknutím pravého a levého tlačítka.

* Poz. korektora: V současné době je jich již mnohem více.

Jiný způsob přemisťování oken může spočívat v tom, že se využívá některé klávesy na klávesnici. Například moje vlastní konfigurace umožňuje stisknout klávesu Alt, nastavit kurzor myši kamkoliv do rámce okna a pak po stisknutí levého tlačítka okno přesunout na novou pozici.

I zde platí, že si konfiguraci správce oken můžete měnit metodou pokusů a omylů, nebo si prostudujte příslušnou dokumentaci. Budete-li se pokoušet interpretovat konfigurační soubor správce oken, přečtěte si oddíl Konfigurace programu twm nebo oddíl Konfigurace programu fvwm.

Překrývání oken

Protože v systému X Window může být otevřeno několik oken najednou, má smysl hovořit o překrývání oken. Přestože okna a samotná pracovní plocha systému X Window jsou dvourozměrné, mohou se okna navzájem překrývat, jako by představovaly třírozměrné útvary. To znamená, že vybrané okno může být zobrazeno v popředí a částečně nebo úplně překrývat okna na pozadí. V souvislosti s překrýváním oken existuje několik operací:

- **Vyzvednutí** (raising) okna do popředí. Toho lze zpravidla dosáhnout klepnutím jistým tlačítkem myši na titulní lištu vybraného okna. Podle konfigurace správce oken to může být kterékoliv tlačítko; také je možné dosáhnout vyzvednutí okna do popředí pomocí více než jednoho tlačítka.
- **Zasunutí** (lowering) okna do pozadí. Toho lze obvykle dosáhnout klepnutím jiného tlačítka myši na titulní lištu okna. Dokonce lze nakonfigurovat správce oken tak, že k vyzvednutí i zasunutí okna se použije totéž tlačítko myši – je-li okno v popředí, pak se zasune do pozadí a je-li v pozadí, vyzvedne se do popředí.
- **Cyklické** procházení všech oken. Správce oken může být konfigurován tak, že po stisknutí jisté klávesy se postupně objevuje ve stanoveném pořadí každé otevřené okno v popředí.

Transformace okna do ikony

Nyní se podívejme na další operace, které umožňují transformovat (nebo také skrýt) okno do ikony. V závislosti na konfiguraci správce oken lze tohoto efektu dosáhnout několika různými způsoby. Při používání správce oken twm si většina lidí nakonfiguruje tzv. **správce ikon** (icon manager). Jedná se o speciální okno, které obsahuje seznam jmen všech ostatních oken otevřených na pracovní ploše. Pokud klepnete jistým tlačítkem myši na konkrétní jméno, dané okno se transformuje na ikonu. Okno je stále aktivní, ale nevidíte jej. Pomocí jiného tlačítka můžete provést inverzní operaci – ikona se zpět transformuje na okno.

Uvedené vlastnosti mohou být velmi užitečné. Předpokládejme například, že máte otevřenu spoustu terminálových oken pro příležitostnou komunikaci se vzdálenými počítači. Kdyby zůstala všechna okna otevřena, měli byste na pracovní ploše nepřehledno. Protože však s těmito terminálovými okny pracujete pouze příležitostně, můžete je transformovat do ikon a získat tak větší přehled. Jediné malé nebezpečí spočívá v tom, že zapomenete na některé okno transformované do ikony a zbytečně si otevřete pro stejný účel nové okno.

Pokud jde o umisťování ikon, lze správce oken nakonfigurovat tak, aby se ikony automaticky řadily na spodní okraj pracovní plochy.

Jak měnit velikost oken

V systému X Window existuje několik způsobů, jak měnit velikost oken. Tyto způsoby opět závisejí na konfiguraci správce oken. První a asi nejpoužívanější metoda spočívá v tom, že klepnete na rámeček ohraničující okno levým tlačítkem myši, držíte je stisknuté a rozměr okna nastavíte na požadovanou velikost (tento způsob je běžný i u ostatních operačních systémů, jako je Microsoft Windows nebo OS/2).

Další metoda je založena na speciálním tlačítku umístěném v titulní liště okna. Na obrázku 5.1 si můžete všimnout malého tlačítka na pravé straně titulní lišty každého okna. Stačí klepnout levým tlačítkem myši na toto tlačítko a nastavit rozměr okna na požadovanou velikost. Jakmile má okno příslušné rozměry, levé tlačítko myši uvolněte.

Nastavení maximální velikosti okna

Většina správců oken podporuje tzv. maximalizaci oken, tedy nastavení rozměrů okna tak, aby pokrylo celou obrazovku. Při použití správce oken `twm` lze dokonce nastavit takovou konfiguraci, aby bylo možné maximalizovat pouze vertikální rozměr, horizontální rozměr nebo oba rozměry najednou. Tento proces je v programu `twm` označován jako „zooming" (zvětšování), ale častěji se dává přednost termínu „maximization". Různé aplikace reagují na změnu rozměrů okna různě. Například `xterm` vám poskytne větší pracovní prostor a nezvětšuje přitom velikost fontů.

Obecně lze říci, že proces maximalizace oken není bohužel v systému X Window standardizován.

Nabídky

Dalším úkolem správce oken je řídit funkci nabídek a umožnit tak uživateli rychle realizovat úkony, které se stále dokola opakují. Například lze vytvořit nabídku, která umožní automaticky a rychle spustit editor emacs nebo terminál `xterm`. Obecně platí, že programy běžící v systému X Window můžete spouštět buď z terminálového okna, což je pomalejší a nepohodlné, nebo z vhodně nakonfigurovaných nabídek, což je rychlé a pohodlné.

Různé nabídky mohou být zpřístupněny klepnutím myši v základním okně (často se pro toto okno používá označení pracovní plocha), tedy okně, které nelze přemisťovat a které obsahuje všechna ostatní okna. Pozadí základního okna je implicitně šedé[4]. Chcete-li vyvolat nabídku, stačí umístit kurzor myši na pozadí základního okna a stisknout tlačítko. V nabídce si opět pomocí kurzoru myši vyberte požadovanou položku (aniž uvolníte tlačítko myši) a uvolněním tlačítka ji spusťte.

Atributy systému X Window

Existuje spousta programů, jež využívají vlastností systému X Window. Některé programy, jako je editor emacs, mohou být spuštěny jako textově orientované aplikace a zároveň mohou být spuštěny v prostředí systému X Window. Existuje však řada programů, které mohou běžet pouze v prostředí X Window.

Geometrie

Existuje několik aspektů, které mají programy běžící pod systémem X Window společné. První z nich lze označit jako geometrii. Atributy geometrie popisují velikost a umístění okna a tvoří je čtyři komponenty.

- Horizontální rozměr, zpravidla měřený v grafických bodech. (Grafický bod je nejmenší jednotka na obrazovce, které lze přiřadit barvu. Systém X Window běžně pracuje s rozlišovací schopností 1024 bodů horizontálně a 768 bodů vertikálně.) Některé aplikace (`xterm` nebo `emacs`) měří velikost okna ve znacích. Například šířka okna je dána počtem znaků na řádku (nejčastěji osmdesát), který může aplikace zobrazit.

4 Existuje program xfishtank, který na pozadí základního okna simuluje akvárium a také spousta dalších programů pro změnu pozadí v systému X Window.

- Vertikální rozměr, opět obvykle měřený v grafických bodech. Rovněž je možné stanovit vertikální rozměr ve znacích.
- Horizontální vzdálenost od okrajů obrazovky. Například číslo `+35` znamená, že levý okraj okna bude vzdálen 35 grafických bodů od levého okraje obrazovky. Na druhé straně číslo `-50` bude znamenat, že pravý okraj okna bude vzdálen padesát grafických bodů od pravého okraje obrazovky. Obecně platí, že není možné inicializovat okno mimo rámec obrazovky, i když je pak možné okno mimo rámec obrazovky přesunout.
- Vertikální vzdálenost od horního nebo dolního okraje obrazovky. Kladná vertikální vzdálenost je měřena od horního okraje obrazovky a záporná je měřena od spodního okraje obrazovky. Všechny čtyři komponenty definující geometrii okna se uvádějí prostřednictvím jediného řetězce. Například `503x73-78+0` znamená, že okno bude mít šířku 503 grafických bodů, výšku 73 grafických bodů a že bude umístěno vpravo bezprostředně pod horním okrajem obrazovky. Obecný tvar řetězce pro specifikaci geometrie okna je tedy: `hsizexvsize+hplace+vplace`.

Display

Každé aplikaci v systému X Window je přiřazen tzv. display, což je jméno obrazovky, kterou řídí server systému X Window. Displej se skládá ze tří komponent:

- Jméno počítače, na kterém server běží. Při samostatné instalaci operačního systému Linux běží server vždy na stejném systému jako klient. Ve většině případů může být jméno počítače vynecháno.
- Číslo serveru, který na daném počítači běží. Na každém jednotlivém počítači může běžet několik serverů systému X Window současně (v případě operačního systému Unix to není příliš častý případ). Pak každý z nich musí mít unikátní číslo.
- Číslo obrazovky. Systém X Window podporuje servery řídící v daném okamžiku více než jednu obrazovku. Některé aplikace je účelné provozovat na více než jednom monitoru a pak je nutné, aby server řídil více monitorů. Není totiž efektivní, aby byl pro každý monitor spuštěn zvláštní server.

Uvedené tři atributy lze specifikovat prostřednictvím jediného řetězce v následujícím formátu: *`machine:server-number.screen-number.`*

Například moje aplikace mají nastavenu hodnotu pro display jako `:0.0`. To znamená, že systém pracuje s první obrazovkou, řídí jej první server a běží na lokálním počítači. Kdybych však používal vzdálený počítač, pak bude display nastaven jako `mousehouse:0.0`.

Implicitně se řetězec definující hodnotu display čte ze systémové proměnné `DISPLAY` (viz oddíl Systémové proměnné) a může být předefinován parametry příkazového řádku pro spuštění systému X Window (viz tabulka 5.2). Chcete-li se přesvědčit, zda je systémová proměnná nastavena, zadejte příkaz `echo $DISPLAY`.

Společné vlastnosti

Systém X Window představuje uživatelské grafické rozhraní, které lze prakticky ve všech aspektech konfigurovat. Je téměř nemožné říci, jak jeho jednotlivé komponenty fungují, protože to především závisí na konfiguraci.

jméno	následováno čím	příklad
`-geometry`	geometrií okna	`xterm -geometry 60x24+0+90`
`-display`	display, ve kterém se má program objevit	`xterm -display lionsden:0.0`
`-fg`	primární barva popředí	`xterm -fg yellow`
`-bg`	primární barva pozadí	`xterm -bg blue`

Tabulka 5.2 – Standardní volby pro programy běžící pod systémem X Window

Každá vlastnost může být překonfigurována, změněna nebo úplně nahrazena jinou. Proto je obtížné jednoznačně popsat chování jednotlivých prvků tvořících toto rozhraní. Zatím jsme popsali to, co popsat lze: různé správce oken a možnosti jejich konfigurace.

Situaci ještě více komplikuje skutečnost, že různé aplikace jsou založeny na různých grafických knihovnách. Pro tyto knihovny se vžil název „knihovny přípravků" – widget sets. Spolu se systémem X Window se distribuuje „Athena", grafický systém vyvinutý v Massachusetts Institute of Technology. Systém Athena se běžně používá ve volně šiřitelném programovém vybavení. Má tu nevýhodu, že nevytváří příliš dobře vyhlížející grafické objekty a jeho používání je poměrně těžkopádné.

Další populární knihovnou je „Motif". Motif patří ke komerčním produktům a má podobné vlastnosti jako uživatelské grafické rozhraní používané v operačním systému Microsoft Windows. Tuto knihovnu využívá spousta komerčních a některé volně šiřitelné aplikace. Populární program pro prohlížení stránek WWW Netscape také využívá knihovnu Motif.

Pokusme se nyní popsat některé společné vlastnosti systému X Window.

Tlačítka

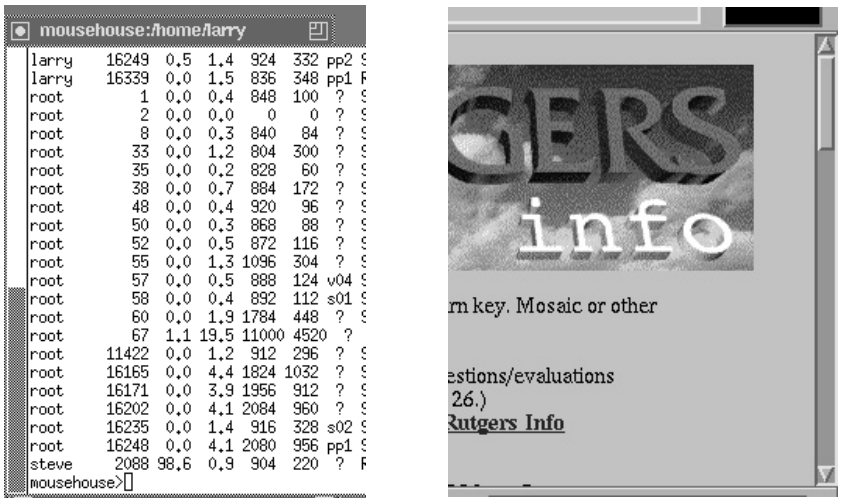
Tlačítka jsou prvky grafického rozhraní, jež se nejsnadněji používají. Stačí na tlačítko přesunout kurzor myši a klepnout levým tlačítkem. Tlačítka systémů Athena a Motif fungují stejně, rozdíly mezi nimi jsou pouze kosmetické.

Nabídkové lišty

Nabídková lišta je soubor příkazů zpřístupněných pomocí myši. Na obrázku 5.3 je například zobrazena nabídka editoru `emacs`. Každé slovo v nabídce představuje záhlaví jisté množiny dalších příkazů. Například položka `File` obsahuje příkazy pro zavádění a ukládání souborů. Podle zažité konvence zde najdete i příkaz k ukončení editoru.

Chcete-li zpřístupnit některý příkaz z nabídky, přesuňte kurzor myši na příslušnou nabídku (například `File`) a stiskněte levé tlačítko myši (držte je stále stisknuté). Pak se vám zobrazí další nabídka příkazů. Chcete-li si jeden z nich vybrat, přesuňte kurzor myši na požadovaný příkaz a uvolněte tlačítko myši. Některé nabídkové lišty vyžadují, abyste na požadované položce klikli myší. Pak program vyčkává, až klepnete na nějaký příkaz. Také můžete klepnout mimo nabídku a tím dáte programu najevo, že žádný příkaz realizovat nemá.

Obrázek 5.3 – Editor Emacs mění nabídkovou lištu podle toho, s jakým typem souboru zrovna pracujete. Zde je ukázka lišty obsahující nabídky.



Obrázek 5.4 – Na levé straně tohoto terminálového okna je ukázka posuvné lišty vytvořené v systému Athena. Na druhé straně je ukázka posuvné lišty (v okně programu Netscape) vytvořené systémem Motif.

Posuvné lišty

Posuvná lišta je nástroj umožňující zobrazit pouze část dokumentu, zatímco zbytek dokumentu je mimo okno. Například na obrázku 5.4 zobrazuje terminálové okno dolní třetinu textu. Díky posuvným lištám se také snadno určí, jaká část dokumentu je právě zobrazena. Tmavá část posuvné lišty ukazuje relativní pozici právě zobrazené stránky textu a zároveň její relativní velikost. Jestliže se celý text vleze na obrazovku, je celá posuvná lišta zobrazena tmavě. Jestliže je na obrazovce zobrazena polovina textu, pak bude zobrazena tmavě polovina posuvné lišty.

Vertikální posuvné lišty mohou být zobrazeny jak v pravém okraji okna, tak i v levém. Horizontální posuvné lišty bývají zpravidla zobrazeny u dolního okraje okna.

Posuvné lišty v systému Athena

Posuvné lišty v systému Athena fungují jinak, než u ostatních systémů. Každé tlačítko myši má jinou funkci. Má-li se text rolovat nahoru (tj. má-li se zobrazit text nad momentálně zobrazeným textem), pak stačí klepnout pravým tlačítkem myši, přičemž je kurzor myši umístěn kdekoliv v posuvné liště. Pokud chcete text rolovat dolů, pak klepněte levým tlačítkem myši.

Dále si můžete pomocí prostředního tlačítka myši zobrazit text v konkrétní pozici. Stačí klepnout prostředním tlačítkem myši na odpovídající pozici v posuvné liště.

Posuvné lišty v systému Motif

Posuvné lišty v systému Motif fungují téměř stejně jako v operačním systému Windows nebo Macintosh. Příklad posuvné lišty je na obrázku 5.4. Všimněte si, že na začátku a konci posuvné lišty jsou šipky. Ty jsou určeny pro „jemné“ rolování textu. Pokud na jedné z těchto šipek klepnete levým nebo prostředním tlačítkem myši, posune se text jen o několik málo řádků. Pravé tlačítko myši v tomto případě neplní žádnou funkci.

Pokud jde o používání myši uvnitř posuvné lišty, existují významné rozdíly mezi systémy Athena a Motif. Pravé tlačítko myši nemá žádný význam. Klepnutí levým tlačítkem myši nad momentální pozicí tmavé části posuvné lišty způsobí posun textu nahoru. Podobně platí, že klepnutí levým tlačítkem myši pod momentální pozicí tmavé části lišty způsobí posun textu dolů. Dále lze zobrazovaným textem přímo manipulovat tak, že se klepne levým tlačítkem myši na tmavou část posuvné lišty, drží se tlačítko stisknuté a nastaví se pozice tmavé části na požadované místo. Jakmile se tlačítko uvolní, požadovaná pozice se zobrazí.

Prostřední tlačítko má podobnou funkci jako u systému Athena. Zobrazí se ta část textu, která svou pozicí relativně odpovídá pozici kurzoru myši v posuvné liště.

Práce s operačním systémem Unix

Operační systém Unix je velmi výkonný nástroj pro ty, kdo jej umí používat. V této kapitole si popíšeme pokročilejší techniky práce s příkazovým procesorem `bash`.

Pseudoznaky

V předcházející kapitole jsme se naučili pracovat s příkazy pro manipulaci se soubory. Příležitostně se může stát, že budete chtít pracovat s více soubory najednou. Například budete chtít zkopírovat všechny soubory začínající posloupností „data“ do adresáře `~/backup`. Můžete to udělat tak, že použijete několikrát příkaz `cp`, nebo v jednom příkazu `cp` uvedete seznam všech souborů, které se mají kopírovat. Oba způsoby jsou těžkopádné a jistě vám seberou hodně času. Navíc máte velkou šanci udělat chybu.

Na následujícím příkladě si uvedeme daleko elegantnější postup:

```
/home/larry/report# ls -F
1993-1 1994-1 data1 data5
1993-2 data-new data2
/home/larry/report# mkdir ~/backup
/home/larry/report# cp data* ~/backup
/home/larry/report# ls -F ~/backup
data-new data1 data2 data5
/home/larry/report#
```

Z uvedeného příkladu vidíte, že po zadání hvězdičky zkopíroval příkaz `cp` všechny soubory začínající řetězcem data do adresáře `~/backup`. Zkuste uhodnout, jak bude fungovat příkaz `cp d*w ~/backup`

Co se ve skutečnosti stalo?

Rozhodně dobrá otázka. Příkazový interpret `bash` je schopen interpretovat jisté znaky, kterým se říká pseudoznaky (wildcards). Znak hvězdička (*) je interpretován tak, že za něj dosadí jakoukoliv posloupnost znaků. Proto příkaz z našeho příkladu `cp data* ~/backup` byl interpretován jako příkaz `cp data-new data1 data2 data5 ~/backup`.

Abychom ještě lépe ilustrovali interpretaci znaku hvězdička, uvedeme si nový jednoduchý příkaz `echo`, který zkopíruje své parametry jako text na obrazovce.

```
/home/larry# echo Hello!
Hello!
/home/larry# echo How are you?
How are you?
/home/larry# cd report
/home/larry/report# ls -F
1993-1 1994-1 data1 data5
1993-2 data-new data2
/home/larry/report# echo 199*
1993-1 1993-2 1994-1
/home/larry/report# echo *4*
1994-1
/home/larry/report# echo *2*
1993-2 data2
/home/larry/report#
```

Jak sami vidíte, příkazový procesor rozšířil hvězdičku, vybral všechny soubory vyhovující specifikaci s hvězdičkou a předal je programu, který byl zadán ke spuštění. Naskýtá se přirozená otázka. Co se stane, když specifikaci s hvězdičkou nevyhovuje žádný soubor? Vyzkoušejte si například příkaz `echo /rc/fr*og` a uvidíte – příkazový procesor předá specifikaci přesně tak, jak byla zadána (žádnou interpretaci hvězdičky neprovede).

Ostatní příkazové procesory, například tcsh, se chovají jinak – nepředávají přesnou specifikaci, ale vypíší zprávu `No match`. Uveďme si příklad s příkazovým procesorem `tcsh`:

```
mousehouse>echo /rc/fr*og
echo: No match.
mousehouse>
```

Také vás asi napadá otázka, zda je možné pomocí příkazu `echo` zobrazit například `data*` (a nikoliv seznam souborů vyhovujících specifikaci `data*`). Řešení je jednoduché, stačí požadovaný text uzavřít do uvozovek. Například:

```
/home/larry/report# echo "data*"
data*
/home/larry/report#
```

```
mousehouse>echo "data*"
nebo    data*
mousehouse>
```

Znak otazník

Kromě znaku hvězdičky je příkazový procesor schopen interpretovat jako speciální znak také znak otazník. Uvede-li se ve specifikaci otazník, pak to znamená, že má být nahrazen právě jedním znakem. Například příkaz `ls /etc/??` zobrazí všechny soubory v adresáři `/etc`, jejichž jména se skládají právě ze dvou znaků.

Jak ušetřit čas pomocí příkazu bash

Editování příkazového řádku

Někdy se stane, že napíšete v příkazovém řádku příkazového interpretu `bash` dlouhý příkaz a pak si všimnete, že jste udělali chybu. Samozřejmě můžete postupně smazat všechny znaky, až se dostanete k chybně zadanému znaku, ale pak budete muset celý zbytek příkazu znovu psát. Příkazový interpret `bash` umožňuje používat kurzorové klávesy (šipka vlevo a šipka vpravo), pomocí kterých si můžete nastavit kurzor na chybně zadaný znak a opravit jej.

Dále existuje řada speciálních klávesových zkratek, pomocí nichž můžete příkazový řádek editovat. Tyto klávesové zkratky jsou většinou podobné příkazům používaným v editoru Emacs. Například kombinace Ctrl+T provede výměnu dvou sousedních znaků.[1] Popis většiny důležitých klávesových zkratek najdete v kapitole 8, která je věnována editoru Emacs.

Doplňování příkazů a jmen souborů

Další vynikající vlastnost příkazového procesoru bash spočívá v tzv. doplňování příkazových řádků. Podívejme se na následující příklad.

```
/home/larry# ls -F
this-is-a-long-file
/home/larry# cp this-is-a-long-file shorter
/home/larry# ls -F
shorter        this-is-a-long-file
/home/larry#
```

Samozřejmě je velmi nepohodlné a namáhavé vypisovat každý znak jména souboru `this-is-a-long-file` pokaždé, když chcete tento soubor zpřístupnit. Vytvořte si takový soubor, například příkazem `cp /etc/passwd this-is-a-long-file`. Nyní si ukážeme, jak zadat předcházející příkaz rychleji a s menším rizikem, že uděláme chybu.

Místo vypisování celého jména souboru zadejte `cp th` a pak stiskněte klávesu Tab. Jako mávnutím kouzelného proutku se vám celé jméno souboru objeví v příkazovém řádku a vám již stačí napsat `shorter`. Příkazový procesor `bash` neumí bohužel číst vaše myšlenky, proto jméno souboru `shorter` musíte napsat sami.

Když stisknete klávesu Tab, příkazový interpret `bash` se pokusí najít soubor, který začíná znaky, jež jste doposud napsali. Když například napíši `/usr/bin/ema` a pak stisknu Tab, příkazový procesor bash najde soubor `/usr/bin/emacs`. Když však napíši `/usr/bin/ld` a pak stisknu klávesu Tab, příkazový interpret pípne, aby mě upozornil, že našel více souborů. Skutečně, v adresáři `/usr/bin` jsou soubory `ld`, `ldd` a `ld86`.

Jestliže se pokoušíte doplnit jméno souboru a příkazový procesor pípne, pak můžete znovu stisknout klávesu Tab a objeví se vám seznam všech nalezených souborů. Proto nemusíte znát přesně jména svých souborů, příkazový procesor vám je vždy tímto způsobem připomene.

1 Zkratka Ctrl+T znamená, že máte stisknout klávesu Ctrl, držet ji stisknutou, a pak stisknout klávesu T.

Standardní vstup a standardní výstup

Zkuste si provést příkaz, který vypíše seznam všech souborů v adresáři /usr/bin: ls /usr/bin. Adresář /usr/bin obsahuje velké množství souborů, proto po zobrazení jejich jmen zůstane na obrazovce jen několik a ostatní „utečou“ nahoru. Jak tento problém řešit?

Standardní vstup a výstup

Operační systém Unix nabízí programátorům velmi jednoduchý způsob zápisu na terminál. Pokud nějaký program něco vypisuje na vaši obrazovku, pak používá tzv. **standardní výstup** (standard output). Standardní výstup se zkracuje zkratkou stdout a slouží k předávání informací uživateli. Na druhé straně existuje **standardní vstup** (standard input), který slouží k předávání informací od uživatele. Samozřejmě je možné, aby program komunikoval s uživatelem bez použití standardního vstupu a výstupu, ale většina programů, kterými se zabýváme v této knize, standardní vstup a standardní výstup používá.

Například příkaz ls vypisuje seznam adresářů a souborů na standardní výstup, jenž je normálně spojen s terminálem. Příkazový interpret bash čte vámi zadané příkazy ze standardního vstupu.

Programy také mohou zapisovat do tzv. **standardního chybového výstupu** (standard error). Standardní chybový výstup je téměř výlučně spjat s terminálem, proto mohou být uživatelé informováni o případných chybách.

V následujících odstavcích si ukážeme, jak využívat standardní vstup a standardní výstup v konstrukcích, kterým se říká přesměrování vstupu, přesměrování výstupu a roura vstupu/výstupu.

Přesměrování výstupu

Velmi důležitou vlastností operačního systému Unix je schopnost přesměrovat výstup. Tato vlastnost vám umožní uložit výsledky příkazu do souboru nebo vytisknout na tiskárně. Jestliže chcete například přesměrovat výstup z příkazu ls /usr/bin do souboru, přidejte na konec příkazu znak > a za něj uveďte jméno souboru (nejlépe neexistujícího). Například:

```
/home/larry# ls
/home/larry# ls -F /usr/bin > listing
/home/larry# ls
listing
/home/larry#
```

Jak vidíte, seznam souborů se nezobrazil na terminálu, ale místo toho se vytvořil ve vašem domovském adresáři nový soubor listing. Zkusme se na tento soubor podívat pomocí příkazu cat. Jestli si vzpomínáte, příkaz cat se jevil jako zbytečný příkaz, který zkopíroval na obrazovku (standardní výstup) to, co jste napsali (standardní vstup). Příkaz cat také umí vypsat obsah souboru na standardní výstup, pokud tento soubor uvedete jako parametr:

```
/home/larry# cat listing
 ...
/home/larry#
```

Soubor listing nyní obsahuje přesný výstup příkazu ls /usr/bin, tedy seznam všech souborů v adresáři /usr/bin. To je v pořádku, ale stále to neřeší náš původní problém.[2]

2 Pro netrpělivé čtenáře poznamenejme, že zde mohou použít program more. Než se však k němu dostaneme, musíme ještě několik věcí vysvětlit.

Nyní je příkaz `cat` o něco zajímavější – lze jej použít spolu s přesměrováním výstupu. Co asi udělá příkaz `cat listing > newfile`? Přesměrování `> newfile` pro příkazový interpret znamená toto: „vezmi celý výstup z příkazu a ulož jej do nového souboru“. Výstup z příkazu `cat listing` je ovšem soubor `listing`. Jinými slovy, pravě jsme popsali jeden ze způsobů kopírování souborů, i když ne příliš efektivní.

Co asi udělá příkaz `cat > fox`? Příkaz `cat` čte data ze standardního vstupu (v tomto případě terminálu) a kopíruje je na standardní výstup, dokud nenarazí na znak konce souboru Ctrl+D. V tomto případě bude standardní vstup přesměrován do souboru `fox`. Nyní nám příkaz `cat` slouží jako základní editor:

```
/home/larry# cat > fox
The quick brown fox jumps over the lazy dog.
```

Stiskni Ctrl+D

Právě jste vytvořili soubor se jménem `fox` obsahující větu „`The quick brown fox jumps over the lazy dog.`“ Další důležitou funkcí příkazu `cat` je spojování souborů dohromady. Příkaz `cat` bude vypisovat každý soubor, který mu byl předán jako parametr. Proto například příkaz `cat listing fox` nejprve vypíše seznam souborů adresáře `/usr/bin` a nakonec vypíše naši hloupou větu. Příkaz `cat listing fox > listandfox` vytvoří nový soubor tvořený obsahem souborů `listing` a `fox`.

Přesměrování vstupu

Tak, jako je možné přesměrovat standardní výstup, je možné přesměrovat i standardní vstup. Místo toho, aby program četl z klávesnice, bude číst ze souboru. Protože se přesměrování vstupu vztahuje k přesměrování výstupu, je přirozené pro něj vyhradit znak `<`. Tento znak se také uvádí za příkazem, který chcete realizovat.

Přesměrování vstupu je užitečné zejména v případě, kdy máte soubor obsahující data a program, který data čte se standardního vstupu. Na druhé straně platí, že většina programů vyžaduje specifikaci souboru se vstupními daty, proto se přesměrování vstupu nepoužívá tak často, jako přesměrování výstupu.

Roura

Řada příkazů v operačním systému Unix produkuje velké množství informací. Jak jsme si již ukázali, příkaz `sp /usr/bin` vyprodukuje příliš mnoho informací, než aby mohly být zobrazeny na terminálu. Abyste si mohli prohlédnout všechny informace z příkazů, jako je `ls /usr/bin`, budete muset použít další důležitý příkaz operačního systému Unix, který má název `more`.[3] Příkaz `more` způsobí pozastavení výpisu na obrazovku, jakmile se celá obrazovka zaplní. Například příkaz `more < /etc/rc` zobrazí stejným způsobem soubor `/etc/rc` jako příkaz `cat`, ale s tím rozdílem, že výpis pozastaví po každém naplnění obrazovky.

Příkaz `more` také můžete použít ve tvaru `more /etc/rc`, což představuje normální způsob jeho použití.

Co je však platné, že příkaz `more` umožňuje zobrazit více informací, než se vleze na obrazovku? Příkaz `more < ls /usr/bin` nebude fungovat, protože přesměrování vstupu funguje pouze se soubory a nikoliv s příkazy. Budete tedy muset postupovat takto:

3 Název `more` pochází z nápovědy tohoto programu, kterou byl původně řetězec --more--. V mnoha verzích operačního systému Linux se vyskytuje identický příkaz, který postupně programátoři zdokonalovali.

```
/home/larry# ls /usr/bin > temp-ls
/home/larry# more temp-ls
...
/home/larry# rm temp-ls
```

Naštěstí nabízí operační systém Unix mnohem elegantnější způsob. Stačí, když zadáte příkaz `ls /usr/bin | more`. Znak | indikuje tzv. **rouru**. Roura v operačním systému Unix řídí tok dat.

Užitečnost roury ještě více vzrůstá ve spojení s dalšími nástroji, kterým se říká **filtry**. Filtr je program, který čte standardní vstup, nějakým způsobem jej modifikuje a odešle jej na standardní výstup. Příkladem filtru je právě příkaz `more` – čte data ze standardního vstupu, zobrazuje je na standardní výstup (obrazovku) a umožňuje vám prohlédnout celý soubor. `more` však není úplně dokonalý filtr, protože jeho výstup není vhodný pro to, aby mohl být předán jako vstup jinému programu.

Mezi další filtry patří příkazy `cat`, `sort`, `head` a `tail`. Chcete-li například přečíst pouze prvních deset řádků výstupu z příkazu `ls`, můžete použít příkaz `ls /usr/bin | head`.

Současný běh úloh

Jak řídit úlohy

Řízení úloh (job control) představuje možnost zajistit, aby daný proces (proces není v podstatě nic jiného, než běžící program) běžel na pozadí nebo naopak, aby běžel na popředí. Jinými slovy, zřejmě potřebujete nástroje k tomu, abyste mohli dlouhodobě běžící program přesunout do pozadí a tak mohli pracovat na jiných věcech, a přitom měli možnost do programu běžícího na pozadí kdykoliv zasahovat. Tímto nástrojem je v operačním systému Unix příkazový procesor. Ukážeme si, jak jej využívat k řízení úloh.

V souvislosti s řízením úloh jsou v operačním systému Unix vyhrazena dvě důležitá klíčová slova. První z nich je `fg` pro běh programů v popředí a druhé je `bg` pro běh programů na pozadí. Abychom vyzkoušeli, jak fungují, použijeme příkaz `yes`:

```
/home/larry# yes
```

Po spuštění příkazu `yes` se na levé straně obrazovky vypisuje velkou rychlostí písmeno „y“.[4] Pokud byste chtěli příkaz `yes` zastavit, stiskli byste klávesu Ctrl+C. Místo toho však stiskněte klávesu Ctrl+Z. Nyní se vám bude zdát, že se provádění příkazu `yes` zastavilo. Na obrazovce se však objeví následující zpráva:

```
[1]+ Stopped          yes
```

To znamená, že proces `yes` byl pozastaven. Pozastavený proces můžete obnovit pomocí příkazu `fg`, který jej aktivuje v popředí a program poběží dále. Zatímco je program pozastaven, můžete realizovat další příkazy. Než zadáte příkaz `fg`, zkuste si zadat několikrát jiný příkaz, například `ls`.

Jakmile se příkaz `yes` „vrátí“ do popředí, bude pokračovat ve výpisu písmen „y“ stejnou rychlostí jako na počátku po jeho spuštění. Nemusíte mít obavy, že se výstup z programu, který byl pozastaven, někam ztratí. Je-li program pozastaven uvedeným způsobem, pak až do okamžiku, kdy jej obnovíte, žádný výstup neprodukuje. Nyní stiskněte klávesu Ctrl+C a program `yes` zrušte.

4 Možná se vám zdá příkaz `yes` divný. Řada příkazů však potřebuje potvrdit nějakou volbu, proto v operačním systému Unix takový příkaz existuje.

Nyní se vraťme k předcházející zprávě:

```
[1]+ Stopped          yes
```

Číslo v hranatých závorkách je **pořadové číslo úlohy**, na které se můžete odvolávat, kdykoliv budete chtít na běhu této úlohy něco změnit. Číslo úlohy se zavádí z toho důvodu, aby existovalo nějaké rozlišení mezi více současně běžícími úlohami. Znak „+“ za hranatými závorkami indikuje, že uvedená úloha je aktuální úlohou, tedy úlohou, jež byla jako poslední převedena z popředí na pozadí. Jestliže zadáte příkaz `fg`, pak do popředí převedete právě tu úlohu, která je označena znakem „+“. Slovo `Stopped` znamená, že úloha byla pozastavena a nachází se ve zvláštním stavu – není zrušena, ale také není aktivní. Operační systém Linux ji uvedl do speciálního stavu, kdy může pokračovat v realizaci, jakmile bude zadán příslušný příkaz. Jako poslední je uvedeno jméno procesu, tedy v našem případě `yes`.

Než pokročíme dále, zrušme tuto úlohu a nastartujme ji, tentokrát jiným způsobem. Příkaz ke zrušení úlohy se jmenuje `kill` (doslova zabít, zničit) a používá se následujícím způsobem:

```
/home/larry# kill %1
[1]+ Stopped yes
/home/larry#
```

Zpráva, která se objevila na obrazovce (říká, že proces byl pozastaven) je zavádějící. Abychom zjistili, v jakém stavu se proces nachází (t.j. zda je aktivní, zmrazen nebo pozastaven), zadáme příkaz `jobs`:

```
/home/larry# jobs
[1]+ Terminated yes
/home/larry#
```

Až zde máte napsáno, v jakém stavu se úloha `yes` skutečně nachází – byla ukončena a nikoliv pozastavena. Je také možné, že zadáte příkaz `jobs` a vůbec žádná zpráva se nevypíše. To znamená, že na pozadí opravdu neběží žádná úloha. Jestliže jste právě zrušili nějakou úlohu a příkaz `jobs` nic nevypíše, pak jste ji zrušili úspěšně. Obvykle se ale objeví zpráva, že úloha byla ukončena (`Terminated`).

Nyní spusťme příkaz `yes` znovu, tentokrát následujícím způsobem:

```
/home/larry# yes > /dev/null
```

Pokud jste četli oddíl o přesměrování vstupu a výstupu, pak asi tušíte, že příkaz `yes` bude odesílat svůj výstup do souboru `/dev/null`. Soubor `/dev/null` má v operačním systému Unix speciální význam. Představuje „černou díru“, do které se ztratí jakékoliv množství informací, aniž by například hrozilo, že se zaplní váš pevný disk. Nemáte-li dostatek fantazie, pak si představte, že je ve vašem počítači vyvrtána díra, kterou proudí informace ven a ztrácejí se v hlubinách vesmíru.

Po zadání uvedeného příkazu se vám nevrátí nápověda příkazového řádku, ani se nebudou na obrazovce vypisovat písmena „y“. I když je výstup z příkazu `yes` přesměrován do souboru `/dev/null`, úloha stále běží na popředí. Jako obvykle ji můžete pozastavit pomocí klávesy Ctrl+Z. Pak se vám samozřejmě znovu objeví výzva příkazového řádku:

```
/home/larry# yes > /dev/null
[Příkaz "yes" běží, nyní stiskněme Ctrl+Z]
[1]+ Stopped          yes > /dev/null
/home/larry#
```

Je nějaká možnost přesunout úlohu do pozadí tak, aby dále běžela a aby se nám objevila výzva příkazového řádku? K tomuto účelu slouží příkaz `bg`:

```
/home/larry# bg
[1]+ yes > /dev/null &
/home/larry#
```

Nyní byste měli věřit následujícímu tvrzení: po zadání příkazu `bg` běží proces `yes > /dev/null` dále, a to na pozadí. Poznáte to podle toho, že když v příkazovém řádku zadáte nějaký příkaz, například `ls` nebo `stuff`, bude jeho realizace poněkud zpomalena. Jiný efekt úlohy běžící na pozadí nemají. Nadále můžete v příkazovém řádku zadávat jakékoliv příkazy a proces `yes > /dev/null` poběží na pozadí a přitom bude výstup odesílat do „černé díry".

Existují dvě možnosti, jak proces běžící na pozadí zrušit (doslova „zabít"). Buď použijete příkaz `kill`, nebo přesunete proces do popředí a ukončíte jej pomocí klávesy Ctrl+C. Vyzkoušíme si druhý způsob, abychom lépe pochopili vztah mezi příkazy `fg` a `bg`:

```
/home/larry# fg
yes > /dev/null
[Nyní běží proces opět v popředí. Stiskněte klávesu Ctrl+C a ukončete jej.]
/home/larry#
```

Jako další si vyzkoušíme spustit současně více procesů, například:

```
/home/larry# yes > /dev/null &
[1] 1024
/home/larry# yes | sort > /dev/null &
[2] 1026
/home/larry# yes | uniq > /dev/null
[nyní stiskněte klávesu Ctrl+Z]
[3]+ Stopped yes | uniq > /dev/null
/home/larry#
```

Určitě jste si všimli, že jsme na konec prvních dvou příkazů zapsali znak `&`. Pokud na konec příkazu zadáte znak `&`, příkazový procesor spustí úlohu hned od počátku na pozadí. Je to mnohem jednodušší, než spustit úlohu normálním způsobem, stisknout klávesu Ctrl+Z a pak zadávat příkaz `bg`. První dvě úlohy jsme tedy přímo spustili na pozadí. Třetí úloha je v tomto okamžiku pozastavena a tedy neaktivní. Jistě jste zpozorovali, že počítač nyní reaguje na další příkazy pomaleji, protože dvě běžící úlohy spotřebovávají jistý čas procesoru.

Zrušme nyní druhou úlohu, protože ta patrně výrazně snižuje výkon vašeho počítače. Mohli bychom použít příkaz `kill %2`, ale to by bylo příliš jednoduché. Místo toho zadejme následující příkazy:

```
/home/larry# fg %2
yes | sort > /dev/null
[Stiskněte klávesu Ctrl+C]
/home/larry#
```

Uvedený příklad demonstruje, že příkaz `fg` akceptuje parametry začínající znakem `%`. Ve skutečnosti jsme mohli použít následující posloupnost příkazů:

```
/home/larry# %2
yes | sort > /dev/null
[Stiskněte klávesu Ctrl+C]
/home/larry#
```

První příkaz bude opět fungovat, protože příkazový procesor interpretuje číslo úlohy jako požadavek, že má být úloha přenesena do popředí. Čísla úloh odlišuje od ostatních čísel právě pomocí znaku %. Nyní zadejte příkaz `jobs`, abyste věděli, které úlohy momentálně běží:

```
/home/larry# jobs
[1]- Running yes > /dev/null &
[3]+ Stopped yes | uniq > /dev/null
/home/larry#
```

Znak „-“ znamená, že úloha číslo 1 byla do pozadí přesunuta jako první v pořadí a bude jako druhá v pořadí přesunuta do popředí, pokud zadáte příkaz `fg` bez parametrů. Znak „+“ znamená, že úloha číslo 3 bude do popředí přesunuta jako první v pořadí, pokud zadáte příkaz `fg` bez parametrů. Uvedené implicitní pořadí můžete změnit takto:

```
/home/larry# fg %1
yes > /dev/null
[nyní stiskněte klávesu Ctrl+Z]
[1]+ Stopped yes > /dev/null
/home/larry#
```

Pomocí uvedených příkazů jste pozastavili úlohu číslo 1 a změnili jste také pořadí priorit všech úloh. K jakým změnám došlo? Na to vám odpoví příkaz `jobs`:

```
/home/larry# jobs
[1]+ Stopped yes > /dev/null
[3]- Stopped yes | uniq > /dev/null
/home/larry#
```

Nyní jsou obě úlohy pozastaveny (v obou případech jsme použili klávesu Ctrl+Z). Úloha číslo 1 je nyní na pozici první, pokud jde o pořadí, v jakém budou úlohy implicitně přenášeny do popředí. Je tomu tak z toho důvodu, že jsme úlohu nejprve přesunuli do popředí a pak ji pozastavili. Znak „+“ vždy označuje tu úlohu, která byla jako poslední pozastavena, když předtím běžela na popředí. Úlohu číslo 1 můžeme opět aktivovat tímto způsobem:

```
/home/larry# bg
[1]+ yes > /dev/null &
/home/larry# jobs
[1]- Running yes > /dev/null &
[3]+ Stopped yes | uniq > /dev/null
/home/larry#
```

Všimněte si, že úloha 1 nyní běží a u druhé úlohy se objevil znak „+“. Nakonec obě úlohy zrušíme, protože se už nemůžeme dále dívat na to, jak snižují výkon počítače:

```
/home/larry# kill %1 %2
[3] Terminated yes | uniq > /dev/null
/home/larry# jobs
[1]+ Terminated yes > /dev/null
/home/larry#
```

Jistě jste si všimli zpráv upozorňujících na ukončení úloh – jak se zdá, nic neumírá tiše. Tabulka 6.1 obsahuje stručný přehled o příkazech, které se vztahují k řízení úloh.

Teorie řízení úloh

Především je nutné pochopit, že úlohy řídí příkazový procesor. Ve skutečnosti v systému neexistují programy jako fg, bg, jobs, & nebo kill. Tyto programy jsou vnitřními příkazy příkazového procesoru a jsou jeho součástí. (Výjimku někdy tvoří příkaz `kill`, který je v některých operačních systémech Unix implementován jako externí program. Pokud však jde o Linux, příkaz `kill` je integrován v příkazovém procesoru bash). Tento mechanismus řízení úloh je logický. Každý uživatel chce mít k realizaci úloh svůj prostor. Protože každý uživatel pracuje se svým příkazovým procesorem, je logické, aby příkazový procesor úlohy řídil. Odtud dále vyplývá, že čísla úloh se vztahují právě k jednomu uživateli. Moje úloha číslo 1 bude zřejmě zcela odlišná od vaší úlohy číslo 1, a to i v případě, kdy budeme oba přihlášeni k témuž systému. Ve skutečnosti platí, že přihlásíte-li se do systému více než jednou, bude každý spuštěný příkazový procesor vlastnit unikátní datové struktury pro řízení úloh. Jako jediný uživatel tedy můžete mít několik zcela různých úloh s pořadovým číslem 1 běžících v různých příkazových interpretech.

`fg %job`	Tento příkaz je příkazem příkazového procesoru a přesouvá úlohu do popředí. Chcete-li zjistit, která úloha se přesune implicitně jako první, zadejte příkaz `jobs` a podívejte se, která úloha je označena znakem „+". Parametry: volitelným parametrem je číslo úlohy. Implicitní je úloha označená znakem „+".
`&`	Pokud se znak „&" uvede na konec příkazového řádku, bude úloha spuštěna automaticky přímo na pozadí. Pro takto spuštěný proces platí všechny metody řízení úloh, které jsme zde uvedli.
`bg %job`	Tento příkaz je příkazem příkazového procesoru a přesouvá úlohu do pozadí. Chcete-li zjistit, která úloha se přesune implicitně jako první, zadejte příkaz `jobs` a podívejte se, která úloha je označena znakem „+". Parametry: volitelným parametrem je číslo úlohy. Implicitní je úloha označená znakem „+".
`kill %job PID`	Tento příkaz je příkazem příkazového procesoru a ukončuje úlohu, ať běží na pozadí nebo byla pozastavena. Pokaždé byste měli použít jako parametr číslo úlohy, kterému předchází znak `%`. Parametry: buď číslo úlohy, kterému předchází znak `%`, nebo PID (identifikační číslo procesu), pak znak `%` není nutné uvádět. Na jednom příkazovém řádku může být uvedeno více úloh, které se mají ukončit.
`jobs`	Tento příkaz je příkazem příkazového procesoru. Vypisuje informace o běžících úlohách a o úlohách, které byly pozastaveny.
Ctrl+C	Klávesová zkratka vyhrazená k bezpodmínečnému ukončení úlohy. Běžně se používá k ukončení úlohy běžící v popředí. Je však nutné upozornit na skutečnost, že ne všechny programy na tuto klávesovou zkratku reagují.
Ctrl+Z	Klávesová zkratka vyhrazená k pozastavení úlohy. Opět platí, že některé programy ji ignorují. Jakmile je jednou úloha pozastavena, může být znovu přesunuta do popředí nebo ukončena.

Tabulka 6.1 – Přehled příkazů a kláves používaných k řízení úloh

Jediný bezpečný způsob, jak identifikovat proces, představuje identifikační číslo procesu (process identification number, zkratka PID). Každá úloha běžící v systému má svoje unikátní identifikační číslo procesu. Dva různí uživatelé mohou k odkazu na proces použít totéž identifikační číslo a pak mají jistotu, že se jedná o tentýž proces (samozřejmě za předpokladu, že jsou přihlášeni k témuž systému).

Podívejme se na další příkaz, který vám pomůže lépe pochopit význam identifikačních čísel procesů. Příkaz `ps` vypíše seznam všech běžících nebo pozastavených procesů, včetně vašeho příkazového procesoru. Tento příkaz má také několik voleb, z nichž nejdůležitější jsou `a`, `u` a `x`. Zadáte-li volbu `a`, pak se vám zobrazí seznam všech procesů, t.j. procesů, které spustili i ostatní uživa-

telé. Pomocí volby `x` se zobrazí seznam těch procesů, které nejsou nijak svázány s terminálem.[5] Poslední volba `u` zajistí, že se vám zobrazí další užitečné informace o běžících nebo pozastavených procesech.

Jestliže chcete získat komplexní představu, co se ve vašem systému odehrává, zadejte příkaz `ps -aux`. (V novějších verzích je možno znaménko „–“ vynechat a použít pouze příkaz `ps aux`.) Pak se můžete podívat, kolik paměti každý proces potřebuje (sloupec `%MEM`) a jak zatěžuje procesor (sloupec `%CPU`). Ve sloupci `TIME` je uveden celkový čas procesoru, který spuštěný proces spotřeboval.

Ještě jedna poznámka o identifikačním číslu procesu. Kromě parametru `%číslоúlohy` můžete v příkazu `kill` použít identifikační číslo procesu. Spusťte příkaz `yes > /dev/null` na pozadí, pak spusťte příkaz `ps` a podívejte se na identifikační číslo náležící procesu `yes`. Toto identifikační číslo můžete také použít ke zrušení procesu `yes`.[6]

Jestliže začnete programovat v jazyku C, pak se brzy dozvíte, že příkazy pro řízení úloh jsou interaktivní verze systémových volání `fork` a `execl`. Jedná se o příliš složité mechanismy, než abychom je zde mohli popsat. Pokud budete vytvářet programy, které jsou schopny spouštět více procesů, pak se s těmito systémovými funkcemi budete muset seznámit v příslušné dokumentaci.

Virtuální konzoly

Operační systém Linux podporuje tzv. virtuální konzoly. Jedná se o metodu, která budí dojem, že váš počítač není jedním počítačem ale několika počítači najednou. Linux je schopen spustit několik terminálů současně a přitom jsou všechny spojeny s jedním jádrem. Naštěstí je používání virtuálních konzol jednou z nejjednodušších záležitostí v operačním systému Linux. Chcete-li si je vyzkoušet, stiskněte klávesu Alt a pak klávesu F2.[7]

Najednou se vám objeví nová výzva k přihlášení se do systému. Nepodléhejte panice. Nyní pracujete s virtuální konzolou číslo 2. Přihlaste se a proveďte několik příkazů, abyste se přesvědčili, že nově nastartovaný příkazový procesor skutečně funguje. Budete-li se chtít vrátit do virtuální konzoly číslo 1, stiskněte opět klávesu Alt a pak F1. Nebo můžete vytvořit třetí konzolu pomocí kláves Alt a F3.

Operační systém Linux má implicitně povoleno šest virtuálních konzol. Pokud budete chtít vědět jak, prostudujte si manuál *„Příručka správce operačního systému Linux“*. Budete muset udělat nějaké úpravy v souborech v adresáři `/etc`. Šest virtuálních konzol však většině uživatelů stačí.

Začnete-li používat virtuální konzoly, brzy zjistíte, že jsou ideálním nástrojem k realizaci mnoha úkonů současně. Na první konzole můžete například provozovat editor Emacs, na druhé programy pro komunikaci s Internetem a na třetí můžete mít spuštěn příkazový procesor, kdybyste chtěli spustit ještě nějaký další program.

5 To má smysl jen v případě jistých systémových programů, které nekomunikují s uživatelem prostřednictvím klávesnice.

6 Obecně je mnohem jednodušší zrušit úlohu pomocí jejího čísla.

7 Ujistěte se, že pracujete s textovou konzolou. V případě systému X Window by uvedená kombinace kláves nemusela fungovat.

KAPITOLA 7

Malé a výkonné programy

V čem spočívá síla operačního systému Unix

Síla operačního systému Unix spočívá v používání malých a jednoduchých příkazů, které se sami o sobě zdají neužitečné. Pokud se je naučíte spojovat dohromady, vytvoříte systém, jenž je mnohem pružnější a výkonnější než všechny ostatní operační systémy. V této kapitole se budeme zabývat příkazy `sort`, `grep`, `more`, `cat`, `wc`, `spell`, `diff` a `tail`. Z názvů těchto programů nelze bohužel intuitivně předpokládat, jakou funkci plní.

Nejdříve se budeme věnovat každému příkazu odděleně a nakonec uvedeme několik příkladů, jak je spojovat a používat dohromady.[1]

Práce se soubory

Kromě příkazů `cd`, `mv` a `rm`, o kterých jsme se zmínili v kapitole 4, existují další příkazy určené k manipulaci se soubory (a nikoliv s daty, jež soubory obsahují). Patří mezi ně `touch`, `chmod`, `du` a `df`. Žádný z těchto příkazů se nestará o obsah souborů – pouze mění některé jejich atributy, se kterými operační systém Unix pracuje.

Uveďme si seznam atributů, se kterými uvedené příkazy manipulují:

- Časové údaje. S každým souborem jsou spjaty tři časové údaje.[2] První časový údaj obsahuje informaci o době vytvoření souboru, druhý o době jeho poslední modifikace a třetí o době, kdy byl naposledy zpřístupněn.
- Vlastník souboru. Každý soubor v operačním systému Unix je vlastněn některým uživatelem.
- Skupina. Každý soubor je také spojen se skupinou uživatelů. Nejčastěji se tato skupina nazývá `users`, což znamená, že soubor je sdílen každým uživatelem přihlášeným do systému.
- Přístupová práva. Každý soubor má přístupová práva (pro která se někdy používá označení privilegia). Jedná se o mechanismus, jenž určuje, kdo může k danému souboru přistupovat, kdo může spouštět dané programy a podobně. Každé z přístupových práv může být odděleně přiřazeno vlastníku souboru, skupině nebo všem uživatelům operačního systému.

1 Poznamenejme, že výklad uvedených příkazů nebude zcela vyčerpávající. Podrobnosti naleznete v manuálových stránkách.

2 Starší souborové systémy v operačním systému Linux uchovávaly pouze jeden časový údaj, protože byly odvozeny od operačního systému Minix. Pokud máte takový souborový systém, pak pro vás budou některé informace neaktuální.

`touch` *`soubor1 soubor2 ... souborN`*

Příkaz `touch` provede aktualizaci časových údajů každého souboru uvedeného jako parametr – nastaví čas vytvoření na aktuální čas. Pokud soubor uvedený jako parametr neexistuje, program touch jej vytvoří. Je také možné specifikovat čas, který má být souborům přiřazen. Podrobnosti si nastudujte v manuálových stránkách.

`chmod [-Rfv]` *`mód soubor1 soubor2 ... souborN`*

Příkaz umožňující změnit přístupová práva se nazývá `chmod` (change mode). Než se pustíme do jeho popisu, musíme si probrat, jaká přístupová práva operační systém Unix bere v úvahu. Každý soubor je spojen se skupinou přístupových práv. Podle těchto přístupových práv operační systém Unix určuje, zda může být soubor čten, zda do něj může být zapisováno, nebo, jedná-li se o spustitelný program, může být spuštěn. V následujících odstavcích budeme hovořit o přístupových právech uživatelů. Každý program, jenž uživatel spustí, má shodná přístupová práva. Pokud přesně nevíte, co program dělá, mohou nastat problémy s bezpečností systému.

Operační systém Unix rozlišuje tři různé typy uživatelů. Prvním z nich je vlastník souboru. Tím je osoba, která má právo v souvislosti s tímto souborem používat příkaz `chmod`. Druhý typ představuje skupina. Většina souborů ve vašem systému by měla mít přiřazen atribut „`users`“, a tak by měla být přístupna každému normálnímu uživateli. Chcete-li znát hodnotu atributu skupina, zadejte příkaz `ls -l` *`file`*.

Nakonec operační systém identifikuje ty osoby, které nejsou ani vlastníky souboru, ani členové jeho skupiny. Pro ty je vyhrazen atribut „`other`“ (ostatní).

Typické nastavení přístupových práv je: pro vlastníka právo číst soubor a zapisovat do souboru, pro skupinu právo číst soubor a pro ostatní jsou všechna práva potlačena. Může se však stát, že skupina má právo soubor číst i do něj zapisovat a přitom vlastník nemá žádná práva.

Nyní si vyzkoušejme, jak pomocí příkazu `chmod` změnit některá přístupová práva. Nejdříve si vytvořte nový soubor, řekněme pomocí příkazu `cat` nebo pomocí editoru Emacs. Implicitně je vám přiděleno právo soubor číst i právo do něj zapisovat. Práva ostatních uživatelů jsou určena podle toho, jak je nastaven váš systém a jak je nastaven proces přihlašování. Ujistěte se, že nově vytvořený soubor jste schopni číst pomocí příkazu `cat`. Nyní si odeberte právo číst tento soubor pomocí příkazu `chmod u-r` *`filename`*. Parametr `u-r` se interpretuje jako „user minus read“. Když se nyní pokusíte soubor přečíst, obdržíte chybové hlášení `Permission denied!`. Chcete-li získat zpět právo soubor číst, zadejte příkaz `chmod u+r` *`filename`*.

Přístupová práva k adresářům používají stejné atributy (pro čtení, zapisování a spouštění), ale poněkud odlišným způsobem je interpretují. Právo číst znamená, že uživatel, skupina nebo ostatní mohou vypisovat seznam souborů v adresáři. Právo zapisovat znamená, že uživatel, skupina nebo ostatní mohou přidávat soubory do adresáře nebo rušit soubory. Právo spouštět znamená, že uživatel může zpřístupňovat soubory v adresáři i v jeho podadresářích. Pokud nemá uživatel žádná práva, nemůže ani použít příkaz `cd`.

Při používání příkazu `chmod` se specifikuje, kdo má k danému souboru přístupová práva (uživatel, skupina, ostatní nebo všichni). Dále se specifikují atributy, které definují, co se s daným souborem může dělat. Znaménko plus indikuje udělení daného práva, znaménko mínus popření. Pomocí znaménka rovná se (=) se specifikují přesná přístupová práva. Přípustná přístupová práva jsou read (čtení), write (zápis) a execute (spouštění).

Pokud se použije v příkazu chmod volba R, pak se změna přístupových práv týká všech souborů v adresáři a všech podadresářů (R je označením pro rekurzivní). Jestliže se použije volba f, pak se příkaz chmod pokusí změnit přístupová práva i v případě, že uživatel není vlastníkem daného souboru. Zadá-li se volba v, pak bude příkaz chmod podrobně vypisovat informace o všech krocích, které realizuje.

Systémová statistika

Příkazy uvedené v tomto oddílu jsou určeny k výpisu informací o systému nebo o jeho částech.

`du [-abs] [cesta1 cesta2 ... cestaN]`

Příkaz du je zkratkou pro „disk usage“ (využívání disku). Zobrazuje informaci o velikosti diskového prostoru, který je přidělen danému adresáři a všem jeho podadresářům. Samotný příkaz du zobrazuje seznam, jehož položky uvádějí u každého adresáře velikost diskového prostoru (který adresář obsazuje), kolik prostoru obsazuje aktuální adresář a jako poslední udává velikost diskového prostoru spotřebovaného aktuálním adresářem a všemi jeho podadresáři. Pokud příkazu du předáte nějaké parametry (jméno souboru nebo adresáře), pak vám vypíše uvedené informace o tomto souboru či adresáři.

Použije-li se volba a, pak se vypíší uvedené informace jak o souborech, tak i o adresářích. Po zadání volby b se informace o velikosti diskového prostoru nebudou uvádět v kilobajtech, ale přímo v bajtech. Jeden bajt je ekvivalentní jednomu znaku v textovém souboru.

Pokud se použije volba s, pak příkaz du vypíše zmíněné informace o adresářích uvedených jako parametry a nikoliv o jejich podadresářích.

`df`

Příkaz df je zkratkou pro „disk filling“. Sumarizuje množství použitého diskového prostoru. Pro každý souborový systém (připomeňme, že souborový systém zajímá buď samostatný disk, nebo samostatný diskový oddíl) vypíše informaci o celkovém množství diskového prostoru, informaci o velikosti použité části, o volné kapacitě a nakonec o celkové kapacitě souborového systému.

Paradoxně se může stát, že se vám zobrazí kapacita větší než 100 %. Je to způsobeno tím, že operační systém Unix rezervuje pro každý souborový systém jistý prostor pro superuživatele. Proto je zajištěno, že i když uživatel zaplní celý disk, zůstává na disku něco málo volného místa, aby se mohly realizovat některé operace.

Pro většinu uživatelů nemá příkaz df žádné užitečné volby.

`uptime`

Příkaz uptime dělá přesně to, co byste podle jeho názvu očekávali. Vypisuje celkový čas, který uplynul od posledního zavedení operačního systému.

Dále příkaz uptime uvádí informaci o aktuálním čase a o tzv. „load average“. Termínem „load average“ se míní průměrný počet úloh čekajících na spuštění v průběhu daného časového intervalu. Příkaz uptime uvádí tyto hodnoty pro poslední minutu, posledních pět minut a posledních deset minut. Jestliže se hodnota „load average“ blíží k nule, pak to znamená, že je systém téměř nevytížen. Hodnota blízká k jedničce znamená, že je systém plně vytížen, ale není zahlcen. Vysoké hodnoty jsou výsledkem skutečnosti, že v systému běží několik programů současně.

Příkaz uptime patří k několika málo programům operačního systému Unix, které nemají parametry a žádné volby. (V nových verzích programu uptime, který je součástí balíku procps, je akceptována volba V, která je určena pro zobrazení verze balíku procps).

`who`

Příkaz `who` slouží k zobrazení seznamu momentálně přihlášených uživatelů systému. Pokud se k příkazu přidají parametry `am i`, pak se zobrazí informace o aktuálním uživateli.

`w [-f] [`*`uživatelské jméno`*`]`

Příkaz `w` zobrazuje informace o momentálních uživatelích operačního systému a informace o tom, co dělají. V podstatě tento příkaz kombinuje příkazy `uptime` a `who`. Záhlaví výstupu z příkazu `w` je přesně stejné jako v případě příkazu `uptime` a každý řádek obsahuje informace o uživateli, například kdy se přihlásil. Kolonka `JCPU` ukazuje celkový čas procesoru, který uživatel spotřeboval, zatímco kolonka `PCPU` ukazuje celkový čas procesoru spotřebovaný jejich momentálně běžící úlohou.

Jestliže u příkazu `w` uvedete volbu `f`, pak se nezobrazí vzdálený systém, ze kterého je daný uživatel přihlášen (ve výpisu bude chybět kolonka `FROM`).

Co obsahují soubory

V operačním systému Unix existují dva hlavní příkazy pro výpis obsahu souborů – `cat` a `more`. Již jsme o nich hovořili v kapitole 6.

`cat [-nA] [`*`soubor1 soubor2 ... souborN`*`]`

Příkaz `cat` nepatří mezi uživatelsky přátelské příkazy. Nečeká, až si přečtete obsah souboru a spíše se používá v souvislosti s rourami. Má však několik užitečných voleb. Například volba `n` zajistí, že všechny řádky vypisovaného souboru budou číslovány. Při zadání volby `A` se budou řídící znaky zobrazovat jako normální znaky a ne jako podivné sekvence paznaků. Opět připomínáme, že kompletní seznam voleb akceptovatelných příkazem `cat` naleznete v manuálových stránkách. Pokud se v příkazovém řádku neuvede ani jeden parametr, použije příkaz `cat` standardní vstup.

`more [-l] [+`*`linenumber`*`] [`*`file1 file2 ... fileN`*`]`

Příkaz `more` je mnohem užitečnější a budete jej často používat zejména k zobrazování souborů ve formátu ASCII. Jedinou zajímavou volbou je `l`, po jejímž uvedení příkazu `more` sdělíte, že nechcete interpretovat znak Ctrl-L jako znak „nová stránka“. `more` zahájí zobrazení od specifikovaného čísla řádku (*`linenumber`*).

Protože je more interaktivním příkazem, uvádíme v následujícím seznam příkazů, kterými lze jeho činnost řídit.

Mezerník	Zobrazí se následující stránka.
`d`	Text se posune o 11 řádků, což je přibližně polovina stránky.
`/`	Vyhledání regulárního výrazu. Zatímco regulární výraz může být opravdu komplikovaný, můžete zadat textový řetězec, který se má vyhledat. Zadáte-li například `/toad` Enter, vyhledá se v celém textu řetězec „`toad`“. Jestliže uvedete jen / Enter, pak se vyhledá další výskyt posledně zadaného řetězce.
`n`	Také tento příkaz je určen k vyhledání posledně zadaného regulárního výrazu.
`: n`	Pokud jste zadali prostřednictvím parametrů více než jeden soubor, zahájí se prohlížení následujícího souboru.
`: p`	Prohlížení se nastaví na předcházející soubor.
`q`	Program `more` se ukončí.

`head [-lines] [file1 file2 ... fileN]`

Příkaz `head` zobrazí prvních deset řádků každého z uvedených souborů nebo prvních deset řádků ze standardního vstupu, pokud žádný soubor není jako parametr v příkazovém řádku uveden. Jakákoliv numerická hodnota uvedená jako volba změní implicitní nastavení deset. Například `head -15 frog` zobrazí prvních patnáct řádků souboru `frog`.

`tail [-lines] [file1 file2 ... fileN]`

Podobně jako příkaz `head` zobrazuje příkaz `tail` jen část souboru. Normálně zobrazuje posledních deset řádků souboru nebo posledních deset řádků ze standardního vstupu. Také akceptuje volbu pro nastavení počtu zobrazených řádků.

`file [file1 file2 ... fileN]`

Příkaz `file` se pokouší identifikovat formát souborů uvedených v seznamu v příkazovém řádku. Protože ne všechny soubory mají příponu charakterizující formát souboru nebo neobsahují posloupnosti znaků, podle kterých by se dal formát identifikovat, pokouší se příkaz `file` provádět některé základní testy a tak odhadnout, co soubor obsahuje.

Buďte opatrní, často se může stát, že je formát souboru identifikován chybně.

Informační příkazy

Následující oddíl je věnovaný příkazům, které mění soubor, provádějí jisté operace se souborem nebo zobrazují některé statistické informace o souboru.

`grep [-nvwx] [-number] expression [file1 file2 ... fileN]`

Jeden z nejužitečnějších příkazů operačního systému Unix je příkaz `grep` (generalized regular expression parser). Jeho jméno se zdá být příliš nadsazené na to, že jde o program, jenž umí pouze vyhledávat regulární výrazy v textu. Následující příklad demonstruje nejjednodušší způsob používání příkazu grep:

```
/home/larry# cat animals
Animals are very interesting creatures. One of my favorite animals is
the tiger, a fearsome beast with large teeth.
I also like the lion---it's really neat!
/home/larry# grep iger animals
the tiger, a fearsome beast with large teeth.
/home/larry#
```

Z uvedeného příkladu vyplývá jedna nevýhoda. I když příkaz vypsal všechny řádky obsahující hledané slovo, neuvedl, kde se dané slovo v souboru vyskytuje (neuvedl číslo řádku).

Někdy to může stačit, v závislosti na tom, co potřebujete udělat. Když například hledáte chyby ve výstupu z programu, stačí zadat příkaz `a.out | grep error`, kde `a.out` je jméno vašeho programu.

Jestliže však chcete přesně vědět, kde se hledaný regulární výraz v souboru nachází, zadejte volbu `n`. Pak bude příkaz `grep` zobrazovat i pořadová čísla řádků. Volbu `v` použijte tehdy, když budete chtít zobrazit seznam všech řádků neobsahujících daný regulární výraz.

K dalším vlastnostem příkazu `grep` patří to, že implicitně vyhledává neúplná slova. V předcházejícím příkladu jsme zadali neúplné slovo `iger` a `grep` našel slovo `tiger`. Jestliže má program `grep` pracovat pouze s celými slovy, zadejte volbu `w`. Po zadání volby `x` bude `grep` pracovat s celými řádky.

Pokud neuvedete v příkazu `grep` žádné parametry, bude prohledávat standardní vstup.

`wc [-clw] [`*`soubor1 soubor2 ... souborN`*`]`

`wc` je zkratka pro „word count" (počet slov). Tento příkaz spočítá slova, znaky a řádky v souborech uvedených v seznamu. Pokud se v příkazovém řádku neuvedou jako parametry žádné soubory, pracuje příkaz `wc` se standardním vstupem.

Pro příkaz `wc` existují tři volby: `c` (character – znak), `l` (line – řádek) a `w` (word – slovo). Pomocí volby se specifikuje, co má program `wc` počítat. Pokud například zadáte příkaz `wc -cw`, pak spočítá znaky a slova, ale nikoliv řádky. Příkaz zadaný bez voleb spočítá vše – znaky, slova i řádky.

Jedna z aplikací příkazu `wc` spočívá v tom, že lze s jeho pomocí určit počet souborů v adresáři: `ls | wc -w`. Pokud chcete například vědět, kolik souborů končí s příponou `.c`, pak zadejte příkaz `ls *.c | wc -w`.

`spell [`*`soubor1 soubor2 ... souborN`*`]`

`spell` je velmi jednoduchý program operačního systému Unix pro kontrolu pravopisu textových souborů. Zpravidla kontroluje pravopis americké angličtiny.[3] Program `spell` je tedy filtr, který prochází soubor ve formátu ASCII a na výstupu vypisuje slova, která považuje za pravopisně nesprávná. `spell` pracuje se soubory uvedenými v příkazovém řádku jako parametry, jinak zpracovává standardní vstup.

Pravděpodobně máte k dispozici i dokonalejší program pro kontrolu pravopisu `ispell`. Program `ispell` navíc nabízí možnosti opravy chybně napsaného slova. Pokud se v příkazovém řádku specifikuje soubor, pak se program `ispell` ovládá pomocí nabídky, jinak pracuje jako klasický filtr.

Chcete-li se dozvědět o programu ispell více, podívejte se do manuálových stránek.

`cmp `*`soubor1`*` [`*`soubor2`*`]`

Příkaz `cmp` porovnává obsahy dvou souborů. První musí být uveden jako parametr v příkazovém řádku, druhý je buď specifikován jako parametr, nebo se čte ze standardního vstupu. Příkaz `cmp` je velmi jednoduchý a jeho úkolem je ukázat, kde se dva soubory liší.

`diff `*`soubor1 soubor2`*

Jedním z nejkomplikovanějších standardních programů operačního systému Unix je příkaz `diff`. GNU verze programu `diff` má více než dvacet voleb! Jedná se o zdokonalenou verzi programu `cmp`, která ukazuje nejen kde se soubory liší, ale také v čem se liší.

Nebudeme probírat kompletní možnosti programu `diff`, protože to překračuje rámec této knihy. Místo toho se zaměříme na základní operace. Krátce řečeno, program `diff` akceptuje jako parametry dva soubory a zobrazí rozdíly mezi nimi na principu „řádek po řádku". Uveďme si příklad:

```
/home/larry# cat frog
Animals are very interesting creatures. One of my favorite
  animals is
the tiger, a fearsome beast with large teeth.
I also like the lion---it's really neat!
```

3 I když existuje několik verzí tohoto programu pro evropské jazyky, distribuce operačního systému Linux většinou obsahuje kontrolu pravopisu americké angličtiny.

```
/home/larry# cp frog toad
/home/larry# diff frog toad
/home/larry# cat dog
Animals are very nteresting creatures. One of my favorite
  animals is
the tiger, a fearsome beast with large teeth.
I also like the lion---it's really neat!
/home/larry# diff frog dog
1c1,2
< Animals are very interesting creatures. One of my favorite
  animals is
---
> Animals are very nteresting creatures. One of my favorite
  animals is
>
3c4
< I also like the lion---it's really neat!
---
> I also like the lion---it's really neat!
/home/larry#
```

Jak vidíte v našem příkladu, program `diff` neprodukuje žádný výstup, pokud jsou soubory identické. Pokud se porovnávaly dva různé soubory, pak řádek `1c1,2` říká, že byl porovnán první řádek levého souboru (`frog`) s prvním a druhým řádkem pravého souboru (`dog`) a že zde byly nalezeny rozdíly. Pak byly porovnány řádky 3 (v souboru `frog`) a 4 (v souboru `dog`) a i zde byl nalezen rozdíl. Může se zdát podivné, že se nejdříve porovnávají řádky s různým pořadovým číslem. Je to však z toho důvodu, aby program pracoval efektivněji.

`gzip [-v#] [`*`soubor1 soubor2 ... souborN`*`]`
`gunzip [-v] [`*`soubor1 souborX (xÎ{1,2,...,N})`*`]`
`zcat [`*`soubor1 soubor2 ... souborN`*`]`

Tyto tři programy se používají ke kompresi a dekompresi dat.

Příkaz `gzip` (GNU zip) je program, který čte původní soubory a produkuje soubory menší. Přitom soubory specifikované v příkazovém řádku ruší a nahrazuje je komprimovanými soubory. Komprimované soubory mají stejná jména, ale navíc mají příponu „`gz`“.

`tr` *`řetězec1 řetězec2`*

Příkaz `tr` (translate characters) pracuje pouze se standardním vstupem – neakceptuje žádné soubory jako parametry. Jeho dvěma parametry jsou dva řetězce. Příkaz nahradí všechny výskyty znaků řetězce *`řetězec1`* na vstupu odpovídajícím znakem z řetězce *`řetězec2`*. Kromě relativně jednoduchého tvaru tohoto příkazu, například `tr frog toad`, může příkaz `tr` akceptovat poměrně složité příkazy. Uvedeme si příklad, kdy se pomocí příkazu `tr` převádějí malá písmena na velká:

```
/home/larry# tr [:lower:] [:upper:]
this is WEIRD sentence.
THIS IS WEIRD SENTENCE.
/home/larry#
```

Program `tr` je dosti složitý a často se využívá v malých programech pro příkazové procesory.

KAPITOLA 8

Editace souborů v editoru Emacs

Co je to Emacs?

Abyste mohli dělat něco rozumného s počítačem, potřebujete mít možnost ukládat texty do souborů a měnit texty, které jsou v souborech uloženy. Takové úkony vám umožní realizovat textový editor. Emacs je jeden z nejpopulárnějších editorů na světě – zejména proto, že i začátečníkům umožňuje bez problémů pracovat s textovými soubory. Klasický editor dodávaný s operačním systémem Unix, `vi`, probereme v dodatku A.

Než se začnete učit pracovat s editorem Emacs, musíte si najít nějaký soubor obsahující obyčejný text a zkopírovat si jej do domovského adresáře.[1] Na začátek by bylo riskantní editovat originální soubor, mohl by obsahovat důležité informace. Editor Emacs se spouští jednoduše:

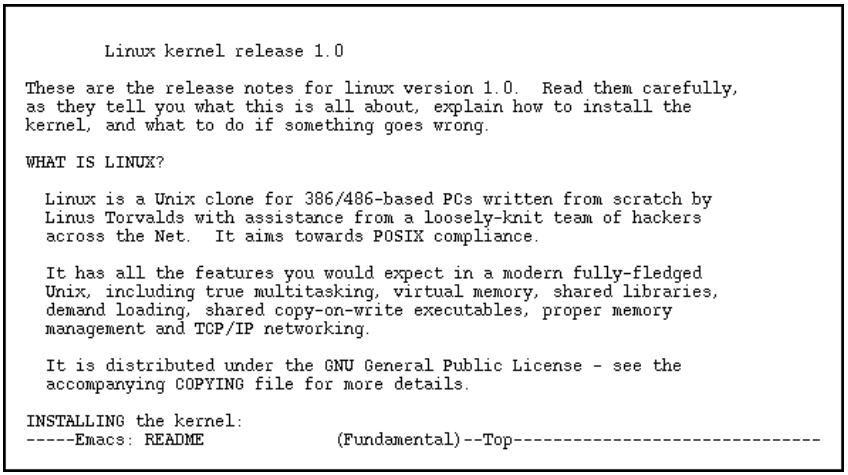
```
/home/larry# emacs README
```

Pokud jste se rozhodli zkopírovat soubor `/etc/rc`, `/etc/inittab` nebo jiný, pak samozřejmě jako parametr uvedete jméno tohoto souboru – například `emacs rc`.

Spuštění editoru Emacs může mít různý efekt. Záleží to na tom, v jakém prostředí jej spouštíte. V textovém terminálu zobrazujícím pouze textové znaky vytvoří nastartovaný editor Emacs okno přes celou obrazovku. Pokud jej spustíte v systému X Window, vytvoří si editor své vlastní okno. Budeme předpokládat, že jste spustili editor Emacs v textovém terminálu. Vše, co zde budeme uvádět, bude platit i pro editor Emacs spuštěný v okně systému X Window. V systému X Window nezapomínejte přesunout kurzor myši na okno obsahující editor, jinak nebudete moci nic psát.

Vaše obrazovka (nebo okno v systému X Window) by měla vypadat přibližně tak, jak je zobrazena na obrázku 8.1. Většina obrazovky obsahuje text vašeho souboru. Obzvláště důležité jsou poslední dva řádky, zejména chcete-li se naučit pracovat s tímto editorem. Řádek obsahující dlouhé posloupnosti pomlček je řádek specifikující mód editoru (modeline).

1 Například `cp /usr/src/linux/README ./README`



Obrázek 8.1 – Editor Emacs byl právě nastartován příkazem `emacs README`

Na uvedeném obrázku vidíte v předposledním řádku slovo „Top“. Může zde být také uvedeno slovo „All“ a mohou se zde také vyskytovat malé rozdíly, což záleží na verzi editoru. Někteří uživatelé mají v tomto řádku zobrazen aktuální čas. Pod uvedeným řádkem se nachází informační a **příkazový řádek**, někdy označovaný jako „minibuffer“, jindy jako „echo area“. Informační řádek slouží editoru ke komunikaci s uživatelem, k vypisování zpráv a někdy zde jeho prostřednictvím budete zadávat příkazy. Budou se zde objevovat takové pokyny, jako: „`For information about GNU project and its goals, type C-h C-p.`“ Zatím takové pokyny ignorujte, budeme se jimi zabývat později.

Než opravdu změníte text v nějakém souboru, musíte se naučit „pohybovat“ textem a pohybovat kurzorem. Kurzor by měl být umístěn na začátku souboru v levém horním rohu obrazovky. Chcete-li kurzor posunout o jeden znak dopředu, stiskněte kombinaci kláves C-F (stiskněte klávesu Ctrl, držte ji stisknutou a stiskněte klávesu F). Kurzor se posune o jeden znak dopředu. Pokud budete obě klávesy stále držet stisknuty, bude se pohyb kurzoru opakovat. Všimněte si, že když kurzor dospěje na konec řádku, automaticky se přesune na začátek následujícího řádku. Opačného směru pohybu kurzoru dosáhnete pomocí kláves C-B. Pokud chcete posunout kurzor o jeden řádek dolů, použijte klávesy C-N, pokud jej chcete posunout o jeden řádek nahoru, stiskněte klávesy C-P[2].

Používání klávesy Ctrl zajišťuje nejrychlejší způsob, jak při editování textu pohybovat kurzorem. Editor Emacs používá takové kombinace kláves, které při psaní udržují vaše ruce nad klávesnicí. Jestliže jste však zvyklí používat kurzorové klávesy, budou také fungovat.

Pokud pracujete v systému X Window, můžete k nastavení pozice textového kurzoru použít kurzor myši. Stačí kurzor myši nastavit na požadovanou pozici v textu a klepnout levým tlačítkem. Tento způsob je však velmi pomalý (jednou rukou musíte opustit klávesnici, uchopit myš, nastavit kurzor a pak se na klávesnici zase vrátit) a nedoporučujeme si na něj zvykat. Většina uživatelů, kteří pracují s editorem Emacs dlouhodobě, používá k pohybu kurzoru výhradně klávesnici.

[2] Zajisté jste si již všimli, že příkazy editoru Emacs jsou většinou realizovány pomocí kombinace dvou kláves.

Pomocí kláves C-P a C-B nastavte kurzor opět na začátek souboru, tedy horní levý roh obrazovky. Nyní podržte klávesy C-B stisknuty poněkud déle. Uslyšíte pípnutí a v informačním řádku uvidíte zprávu „`Beginning of buffer`“.

V tomto okamžiku se asi podivíte. Co je to buffer?

Když editor Emacs zpracovává soubor, nepracuje ve skutečnosti se souborem přímo. Zkopíruje si jeho obsah do speciální pracovní oblasti, která se nazývá **buffer** – zde můžete obsah souboru modifikovat. Až ukončíte editaci souboru, dáte příkaz, aby editor buffer uložil. Do té doby zůstane obsah původního souboru nezměněn a změny se provádějí pouze uvnitř pracovní oblasti editoru.

Nyní jsme tedy připraveni obsah bufferu modifikovat. Vše, co jsme dělali doposud, bylo „nedestruktivní“, což znamená, že jsme obsah bufferu neměnili. Předpokládejme, že chcete do editovaného souboru zapsat znak „X“. Jakmile stisknete klávesu X, změní se předposlední řádek obrazovky. Editor Emacs registruje změny v editovaném textu a jakmile nějaké nastanou, informuje o tom uživatele pomocí dvou hvězdiček před slovem Emacs. Pak bude uvedený řádek vypadat asi takto:

```
--**-Emacs: some_file.txt     (Fundamental)--Top--------------------
```

Hvězdičky zůstanou zobrazeny do té doby, než obsah bufferu uložíte. Obsah bufferu můžete v průběhu editace ukládat průběžně vícekrát – stačí stisknout klávesy C-X C-S (stiskněte klávesu Ctrl, držte ji stisknutou a pak stiskněte klávesu „X“ a „S“). Průběžné ukládání obsahu bufferu má význam především při vytváření nebo modifikaci velkých souborů.

Nyní si uvedeme seznam několika málo příkazů používaných v editoru Emacs spolu s těmi, které jsme již popsali. Je na vás, abyste si jejich používání procvičili. Než pokročíme dále, měli byste s těmito příkazy být důkladně seznámeni.

C-F	Posunutí kurzoru o jeden znak doprava.
C-B	Posunutí kurzoru o jeden znak doleva.
C-N	Posunutí kurzoru o jeden řádek dolů.
C-P	Posunutí kurzoru o jeden řádek nahoru.
C-A	Umístění kurzoru na začátek řádku.
C-E	Umístění kurzoru na konec řádku.
C-V	Zobrazení následující stránky.
C-L	Zobrazení stránky s aktuálním řádkem uprostřed.
C-D	Zrušení znaku, na kterém je umístěn kurzor.
C-K	Zrušení textu od pozice kurzoru do konce řádku.
C-X C-S	Uložení obsahu bufferu do odpovídajícího souboru.
Backspace	Zrušení znaku vlevo od kurzoru.

Používání editoru pod systémem X Window

Pokud chcete rychle editovat soubory pod systémem X Window, máte poněkud ulehčenou situaci. Editovaný text je zobrazen pod nabídkou funkcí editoru, proto je práce s editorem intuitivnější.

V textovém módu uvedená nabídka není dostupná.

Když poprvé spustíte editor Emacs pod systémem X Window, obsahuje nabídka čtyři hlavní položky: `Buffers`, `File`, `Edit` a `Help`. Potřebujete-li si některou z položek hlavní nabídky zpřístupnit, nastavte kurzor myši na tuto položku, stiskněte levé tlačítko myši (držte je stále stisknuté). Pak přesuňte kurzor na požadovanou funkci a tlačítko myši uvolněte. Jestliže žádnou funkci vybrat nechcete, přesuňte kurzor myši mimo nabídku a opět uvolněte tlačítko myši.

Nabídka `Buffers` obsahuje seznam různých souborů, které v této relaci s editorem Emacs editujete. Nabídka `File` obsahuje příkazy pro zavádění a ukládání souborů. Funkce této nabídky podrobněji popíšeme později. Nabídka `Edit` obsahuje některé nejdůležitější příkazy pro editování textového souboru a prostřednictvím nabídky `Help` si můžete interaktivně zobrazovat vybrané části dokumentace k editoru.

Jistě jste si všimli, že u každé funkce v nabídce je uvedena ekvivalentní kombinace kláves pro realizaci této funkce. Na jedné straně můžete funkce vyvolávat pomocí myši a menu, na druhé straně můžete k tomuto účelu používat ekvivalentní kombinace kláves, což je samozřejmě rychlejší.

Editování více souborů současně

V daném okamžiku je editor Emacs schopen pracovat s více soubory. Ve skutečnosti platí, že počet souborů, které mohou být současně uloženy v bufferech editoru, je omezen pouze velikostí paměti počítače. Po zadání příkazu Ⓒ-Ⓧ Ⓒ-Ⓕ se do bufferu editoru Emacs zavádí nový soubor. Když tento příkaz zadáte, v příkazovém řádku se objeví výzva k zadání jména souboru:

```
Find file: ~/
```

Syntaxe používaná k zadání jména souboru je stejná, jako v příkazovém řádku příkazového procesoru; lomítka se používají k oddělení adresářů a podadresářů, znak ~ je interpretován jako váš domovský adresář. I zde funguje možnost automatického **doplňování jména souboru** – zadáte-li dostatek znaků k tomu, aby mohlo být jméno souboru jednoznačně identifikováno, pak stačí stisknout klávesu Tab a zbytek jména souboru se doplní automaticky (nebo se zobrazí seznam jmen všech souborů, která zadané specifikaci vyhovují). Podobnou funkci jako klávesa Tab má klávesa mezerník. Necháme na vás, abyste zjistili, v čem se funkce vyvolané těmito klávesami liší. Máte-li zadáno jméno souboru, který chcete editovat, stiskněte klávesu Enter; editor zavede obsah souboru do své pracovní oblasti a zobrazí jej na obrazovce. V terminologii editoru Emacs se tento proces nazývá vyhledání souboru. Najděte si nějaký další soubor a popsaným způsobem jej zaveďte do editoru. Nyní máte v editoru nový buffer. Budeme předpokládat, že původní má název `some_file.txt` a nový má název `another_file.txt`. Zdá se, že se váš první buffer ztratil. Pravděpodobně se divíte, kam se poděl.

Neobávejte se, stále zůstává uvnitř editoru a k jeho opětovnému zobrazení stačí stisknout klávesovou kombinaci Ⓒ-Ⓧ Ⓑ. Pak se vás editor v příkazovém řádku zeptá, do kterého bufferu se má přepnout. Nabídne implicitní buffer, t.j. buffer, jehož obsah se zobrazí po stisknutí klávesy Enter (nemusíte jméno bufferu uvádět). Implicitní buffer je ten, který jste „opustili“ jako poslední. Jestliže tedy pracujete se dvěma soubory a často mezi nimi přepínáte, používejte klávesovou kombinaci Ⓒ-Ⓧ Ⓑ a nemusíte při každém přepnutí zadávat jméno souboru. Samozřejmě můžete jméno uvádět, ale bude vás to zdržovat.

I při přepínání mezi buffery lze použít funkci k doplňování jména – stejně jako v případě vyhledávání souboru. Po stisknutí klávesy Tab doplní editor automaticky jméno bufferu, je-li schopen jej z doposud zadaných znaků jednoznačně identifikovat. Kdykoliv jste v příkazovém řádku vyzváni k zadání jména, vyzkoušejte si, zda je editor schopen jméno automaticky doplnit. Automatické doplňování vám ušetří spoustu času a editor je schopen je realizovat pokaždé, když má vybrat nějakou položku z nějakého předdefinovaného seznamu.

Vše, co jste se naučili o příkazech pro editování v prvním bufferu platí i v druhém. Pokročme dále – v novém bufferu změňte nějaký text, ale neukládejte jej (pomocí kláves C-X C-S). Nyní předpokládejme, že nechcete provedené změny uložit a že chcete buffer zrušit. K tomu slouží příkaz C-X K. Nejdříve se vás editor zeptá, který buffer chcete zrušit. Stisknete-li klávesu Enter, zruší se implicitní buffer (což je asi nejčastější případ). Editor se vás znovu zeptá, zda chcete buffer opravdu zrušit, a když zadáte „yes“ a stisknete Enter, pak jej konečně zruší.

Nyní byste se měli důkladně procvičit v zavádění souborů, jejich modifikaci, ukládání, přepínání mezi buffery a rušení bufferů. Ujistěte se, že needitujete nějaké důležité soubory, které by mohly poškodit funkci vašeho systému.[3] Vytvořte si alespoň pět bufferů a vyzkoušejte si přepínání mezi nimi.

Ukončení práce s editorem

Když ukončíte práci s editorem Emacs, ujistěte se, že všechny buffery, které mají být uloženy jsou skutečně uloženy. Pak můžete editor ukončit pomocí kláves C-X C-C. Někdy se vás editor po zadání kláves C-X C-C v příkazovém řádku na něco zeptá. Nebuďte znepokojeni a odpovězte normálním způsobem. Jestliže se domníváte, že se budete do editoru chtít později vrátit, nepoužívejte klávesy C-X C-C, ale klávesu C-Z. Pak bude aktivita editoru pouze pozastavena. K opětovné aktivaci editoru můžete použít známý příkaz `fg`. Pozastavení editoru je mnohem efektivnější, než jej stále dokola ukončovat a znovu startovat, zejména když opakovaně editujete tytéž soubory.

V systému X Window má funkce C-Z jiný efekt – provede transformaci okna s editorem do ikony. Podrobně jsme tento mechanismus popsali v kapitole 5. Existují tedy dvě možnosti, jak transformovat okno obsahující editor do ikony – buď normálním způsobem pomocí správce oken, nebo pomocí klávesy C-Z. V systému X Window však k opětovné aktivaci editoru nelze použít příkaz fg – musíte použít správce oken.

Klíče Meta

Doposud jsme při práci s editorem Emacs používali kombinace kláves s klávesou Ctrl. Existují však další možné kombinace, a to s klávesou Meta. Těmto kombinacím se v terminologii editoru Emacs říká meta-klíče. Bohužel ne všechny klávesnice mají klávesu Meta umístěnu ve stejné poloze a některé ji nemají vůbec. V případě klávesnic u počítačů IBM PC je klávesa Meta totožná s klávesou Alt.

Klávesu Meta si můžete otestovat. Stiskněte tu klávesu, o které si myslíte, že by mohla být klávesou Meta, a zároveň stiskněte klávesu X. Pokud se v příkazovém řádku objeví malá nápověda (zpravidla M-X), pak jste kýženou klávesu nalezli. Stiskněte klávesu C-G a znovu se vrátíte do bufferu editoru Emacs.

3 Pokud nejste přihlášeni jako uživatel root, neměli byste být schopni vašemu systému ublížit, ale stejně buďte opatrní.

Jestliže se vám v příkazovém řádku nic neobjeví, stále existuje řešení. Místo klávesy Meta můžete použít klávesu Escape. Avšak v tomto případě ji nedržte stisknutou – stiskněte ji, uvolněte a zadejte další klávesu reprezentující danou funkci. Vyzkoušejte si naši oblíbenou kombinaci, tedy Escape a pak „X“. Nyní se vám v příkazovém řádku nápověda určitě objeví. Opět stiskněte klávesu C-G. Klávesa C-G je v editoru Emacs určena ke zrušení čehokoliv, co nehodláte realizovat. Editor zpravidla vyšle zvukový signál, aby vás upozornil, že danou funkci rušíte.[4]

Označení M-X je analogické označení C-X (kde x je klávesa reprezentující nějakou funkci). Pokud jste nalezli skutečný klíč Meta, pak jej používejte. Jinak budete muset používat popsanou sekvenci s klávesou Escape. Kdekoliv v následujícím textu uvidíte M-X, pak to znamená, že máte použít meta-klíč.

Práce s bloky textu

Editor Emacs, tak jako prakticky každý editor, umožňuje pracovat s bloky textu. Abyste tyto funkce mohli využívat, musíte mít prostředek pro definování **začátku bloku** a **konce bloku**. V editoru Emacs se definice bloku realizuje pomocí nastavení dvou pozic v bufferu, které se označují jako mark (značka) a point. Chcete-li v bufferu nastavit začátek bloku, nastavte kurzor na požadovanou pozici a stiskněte klávesu C-SPC (SPC je zkratka pro mezerník). V příkazovém řádku se objeví zpráva „Mark set“.[5] Nyní je začátek bloku nastaven. Editor nepoužívá žádné zvýrazňovací prostředky k tomu, aby byl začátek bloku viditelný.

A co konec bloku? Konec bloku je definován aktuální pozicí kurzoru. Proto se pro něj používá v terminologii editoru Emacs termín point. Point zrovna tak například znamená místo, od kterého se má vkládat text při kopírování nebo vkládání bloků. Nastavením začátku bloku a přesunutím kurzoru na kteroukoliv jinou pozici v textu je definován začátek a konec bloku. Tento blok se nazývá **region**. Region je tedy vždy část textu mezi značkou a aktuální pozicí kurzoru.

Pouhá definice regionu nezpřístupňuje blok ke kopírování. Nejdříve musíte blok „**zkopírovat**“ (copy), abyste jej pak mohli „**nalepit**“ (paste) někam jinam. Má-li se blok zkopírovat, stiskněte klávesu M-W. Nyní je blok uložen ve speciálním bufferu editoru Emacs. Jestliže nyní chcete blok nalepit někam jinam, nastavte si kurzor na požadovanou pozici a stiskněte C-Y.

Pokud chcete blok textu přesunout (a ne zkopírovat), pak místo příkazu M-W použijte příkaz C-W. Tímto způsobem se blok po přenesení do speciálního bufferu vymaže. Když zjistíte, že jste blok nechtěli vymazat, stiskněte klávesu C-Y a blok se obnoví. Místo, do kterého editor Emacs ukládá části textu, se nazývá **kill-ring**. U jiných editorů se tato oblast nazývá „clipboard“ nebo „paste buffer“.

Existuje ještě jeden způsob, jak vystřihovat a nalepovat text: kdykoliv stisknete klávesu C-K, dojde ke zrušení textu od pozice kurzoru do konce řádku a přitom se zrušený text také ukládá do oblasti kill-ring. Pokud takto zrušíte více než jeden řádek, uloží se všechny řádky do oblasti kill-ring a vy máte možnost je později nalepit najednou.

Někdy je tento způsob přesouvání a kopírování textu rychlejší, než používat systém s označováním začátku a konce bloku. Záleží jen na vás, kterému způsobu dáte přednost.

---

4 Příležitostně se může stát, že jedno stisknutí klávesy C-G nepřesvědčí editor Emacs, že chcete opravdu přerušit činnost, kterou právě děláte. Stačí však trvat na svém a pak se editor vrátí do předcházejícího módu.

5 Na některých počítačích příkaz C-SPC nebude fungovat. Místo toho použijte C-@.

Vyhledávání a náhrada řetězců

V editoru Emacs máte několik možností, jak vyhledávat text. Některé způsoby jsou komplikované a nemá smysl je zde popisovat. Nejjednodušší a nejčastěji používaná metoda je tzv. inkrementální vyhledávání označované jako „isearch“ (incremental search). Předpokládejme, že potřebujete vyhledat řetězec „gadfly“ v následujícím textu:

```
I was growing afraid that we would run out of gasoline, when my passenger ex-
claimed
''Gadzooks! There's gadfly in here!''.
```

Nastavte pozici kurzoru na začátek textu nebo na místo, o kterém víte, že předchází hledanému řetězci, a zadejte příkaz Ⓒ-Ⓢ. Tím nastavíte editor Emacs do módu vyhledávání. Nyní začněte zadávat řetězec, který chcete vyhledat. Jakmile však napíšete první znak, tedy „g“ „přeskočí“ kurzor na první výskyt písmene „g“ v textu. Jestliže máte v editoru text uvedený v našem příkladu, pak kurzor skočí na začátek slova „growing“. Nyní napište písmeno „a“ (druhé písmeno ve slově „gadfly”) a editor přesune kurzor na slovo „gasoline“, protože vyhledal první výskyt dvojice znaků „ga“. Když zadáte další písmeno „d“, skočí kurzor na slovo „gadzooks“ a nakonec po zadání písmene „f“ skočí na hledané slovo „gadfly“. Přitom jste nemuseli zadat kompletní hledaný řetězec.

Při postupném vyhledávání funguje editor Emacs tak, že po každém zadání dalšího znaku hledaného řetězce vyhledá první výskyt toho slova, jehož začátek je shodný s doposud zapsanými znaky. Jakmile zadáte tolik znaků, aby vyhledávání bylo jednoznačné, můžete funkci ukončit stisknutím klávesy Enter. Jestliže se domníváte, že hledaný řetězec je nad aktuální pozicí kurzoru, pak zadejte příkaz Ⓒ-Ⓡ, čímž inicializujete zpětné vyhledávání.

Pokud naleznete první výskyt zadaného řetězce, ale zajímá vás jeho další výskyt, pak znovu zadejte příkaz Ⓒ-Ⓢ. Editor Emacs vyhledá následující výskyt zadaného řetězce a tak můžete pokračovat dále. Když editor výskyt hledaného řetězce nenalezne, vypíše zprávu, že řetězec nenašel a začne znovu vyhledávat od začátku bufferu. Podobná pravidla platí pro příkaz Ⓒ-Ⓡ.

Nyní si popsané funkce vyzkoušejte. Najděte si soubor s anglickým textem a vyhledejte v něm výskyt slova „the“. Pak pomocí opakované funkce Ⓒ-Ⓢ najděte další výskyty. Všimněte si, že editor přitom vyhledá i jiné řetězce, například „them“, protože vyhovují zadané specifikaci. Jestliže chcete vyhledat pouze řetězec „the“, pak na konec hledaného řetězce budete muset přidat mezeru. Při editování hledaného řetězce můžete používat standardní editační klávesy Backspace a Delete. Chcete-li vyhledávání ukončit, vždy použijte klávesu Enter.

Editor Emacs také umožňuje nahradit výskyt jednoho řetězce jiným. Tento proces se v terminologii editoru Emacs označuje jako **query-replace**. Proces zahájíte tak, že stisknete Ⓜ-Ⓧ a napíšete `query-replace` a stisknete Enter.

I v případě zadávání příkazů funguje v editoru Emacs doplňování, a proto stačí, když například napíšete „query-re“ a stisknete klávesu Tab. Předpokládejme, že chcete řetězec „gadfly“ nahradit řetězcem „housefly“. V příkazovém řádku „`Query replace:`“ zadejte „gadfly“ a stiskněte Enter. Jak budete opět vyzváni k zadání řetězce, kterým se má původní řetězec nahradit – zadejte „housefly“. Editor Emacs bude nyní procházet text, zastavovat kurzor na každém výskytu řetězce „gadfly“ a bude se vás ptát, zda má nalezený řetězec nahradit. Pokud ano, stiskněte klávesu Ⓨ, pokud ne, stiskněte klávesu Ⓝ. Jestliže je pro vás předcházející výklad příliš komplikovaný, popsané funkce si vyzkoušejte. Bude to mít pro vás větší význam, než kdybyste předcházející odstavce četli desetkrát.

Vnitřní funkce editoru Emacs

Všechny funkce editoru Emacs vyvolané stisknutím nějaké kombinace kláves mají nějaký název, kterému editor „rozumí". Například klávesa C-P znamená pro editor Emacs provedení vnitřní funkce `previous-line`. Všechny vnitřní funkce mohou být volány prostřednictvím svého jména, a to po stisknutí kláves Alt+X. Když například zapomenete, jaký klíč máte použít k nastavení kurzoru na předcházející řádek, stačí zadat: M-X `previous-line` a Enter. Vyzkoušejte si to a přesvědčte se, že C-P a M-X `previous-line` realizují tutéž funkci.

Autor editoru Emacs postupoval tak, že nejdříve definoval celou množinu vnitřních funkcí editoru a pak teprve navrhl klávesové zkratky pro realizaci nejčastěji používaných funkcí. Někdy je jednodušší použít explicitní volání funkce prostřednictvím M-X, než si pamatovat klávesovou zkratku svázanou s touto funkcí. Například funkce `query-replace` je u některých verzí editoru Emacs svázána s klávesou M-%. Kdo si ale má pamatovat takovou divnou kombinaci? Pokud budete náhradu řetězců provádět extrémně často, pak má samozřejmě smysl si klávesovou zkratku pamatovat.

Většina kláves, které stisknete, jsou písmena, číslice, případně další znaky, které se mají vkládat do textového bufferu. Každá z těchto kláves je **svázána** s funkcí, jež má jméno `self-insert-command`. Tato funkce nedělá nic jiného, než že dané písmeno nebo číslici vloží do bufferu. Kombinace kláves, například s klávesou Ctrl, jsou obecně svázány s jinými funkcemi – pohyb kurzoru a podobně. Například kombinace C-V je svázána s funkcí `scroll-up`, což znamená, že se editovaný text posune o jednu obrazovku dolů.

Jak ale postupovat, když do textu chcete vložit řídící znak? Konec konců, řídící znaky jsou také znaky ASCII, i když zřídkakdy používané. Může se stát, že budete chtít do textu takový znak vložit. Proto v editoru Emacs existuje prostředek, který zabrání editoru interpretovat kombinaci klávesy s klávesou Ctrl jako příkaz. Klávesa C-Q[6] je svázána se speciální funkcí, která má název `quoted-insert`. Jediné, co funkce `quoted-insert` dělá, je, že přečte následující klávesu a vloží ji do textového bufferu bez interpretace. Pokud chcete do textu vložit samotný znak C-Q, pak zadejte C-Q dvakrát.

V editoru Emacs existují některé funkce, jež nejsou svázány s žádnou kombinací kláves. Když například píšete dlouhý text, pak asi nebudete chtít ukončovat každý řádek klávesou Enter. Po editoru Emacs můžete chtít, aby to dělal za vás (po editoru Emacs můžete chtít cokoliv). Příkaz, který takovou funkci realizuje, má název `auto-fill-mode`. Není však implicitně svázán s žádnou kombinací kláves. Jestliže chcete tento příkaz inicializovat, zadejte `M-X auto-fill-mode`. Jak jsme si již řekli, „M-X" je klíč umožňující vyvolat vnitřní funkci editoru jménem. Takto byste mohli vyvolat i funkci `next-line` (následující řádek) nebo `previous-line` (předcházející řádek), ale to by bylo velmi neefektivní, protože uvedené funkce jsou svázány s klávesami C-N a C-P.

Mimochodem, když se po vyvolání funkce `auto-fill-mode` podíváte na předposlední řádek, uvidíte zde na pravé straně slovo Fill. Dokud se zde toto slovo vyskytuje, bude editor Emacs ukončovat řádky za vás. Když funkci „`M-X auto-fill-mode`" vyvoláte znovu, automatické ukončování řádků přestane fungovat – funkce je vytvořena jako přepínač.

Může se vám zdát, že zadávání funkcí prostřednictvím jejich dlouhých jmen není příliš pohodlné. Naštěstí i v tomto případě v editoru Emacs funguje doplňování jmen (stejně jako doplňování jmen souborů). Proto platí, že zřídkakdy budete muset vypisovat celé jméno funkce, písmeno po písmenu. Pokud si nejste jisti, zda bude editor schopen doplnit jméno, stiskněte klávesu Tab. V horším případě se vám objeví znak Tab, v lepším případě editor provede doplnění.

---

6 Pro kombinaci kláves C-Q se používá označení „klávesa", protože představuje jediný znak z tabulky ASCII.

Nápověda v editoru Emacs

Editor Emacs má velmi rozsáhlou nápovědu. Tak rozsáhlou, že se o ní zmíníme jen velmi stručně. Nejvyšší úroveň nápovědy se vyvolá stisknutím kombinace kláves Ⓒ-Ⓗ a nějakého písmene. Například Ⓒ-Ⓗ Ⓚ vyvolá nápovědu vztahující se ke kombinacím kláves (budete vyzváni k zadání kombinace kláves a pak se objeví text s vysvětlením, jakou funkci kombinace kláves realizuje). Ⓒ-Ⓗ Ⓣ vyvolá výukový program editoru Emacs. K důležitým kombinacím kláves patří Ⓒ-Ⓗ Ⓒ-Ⓗ, kdy se vám objeví „nápověda o nápovědě“. Zde se dozvíte vše o systému nápovědy. Když znáte jméno funkce editoru Emacs (například `save-buffer`), ale nemůžete si vzpomenout na kombinaci kláves k jejímu vyvolání, použijte Ⓒ-Ⓗ Ⓦ („`where - is`”) a zadejte jméno funkce. Zrovna tak máte možnost zjistit podrobné informace o dané funkci – zadejte Ⓒ-Ⓗ Ⓕ a pak jméno funkce.

Mějte na paměti, že editor Emacs je sice schopen doplňovat jména funkcí, avšak nemůžete na tuto vlastnost příliš spoléhat, pokud k nějaké funkci (jejíž název přesně neznáte) potřebujete nápovědu. Jestliže si myslíte, že můžete odhadnout slovo, kterým funkce začíná, zkuste je napsat a stiskněte Tab. Uvidíte, zda editor byl schopen funkci identifikovat. Pokud ne, zkuste něco jiného. Totéž platí pro jména souborů. I když si nemůžete vzpomenout, jak se jmenuje soubor, který jste editovali před mnoha měsíci, můžete název odhadnou a zkusit, co na to editor odpoví. Doplňování jmen je tedy nejen užitečné v tom, že šetří čas, ale pomáhá vám nalézt to, co jste již zapomněli, nebo to, co si přesně nepamatujete.

Existuje několik dalších znaků, které můžete zadat po příkazu Ⓒ-Ⓗ a tak získat nápovědu různým způsobem. Nejčastěji však budete používat Ⓒ-Ⓗ Ⓚ, Ⓒ-Ⓗ Ⓦ a Ⓒ-Ⓗ Ⓕ. Až budete blíže seznámeni s editorem Emacs, vyzkoušejte si například Ⓒ-Ⓗ Ⓐ. Pak se vás editor zeptá na řetězec a mezi všemi jmény funkcí nalezne to, které daný řetězec obsahuje. (Písmeno „a“ je zkratkou pro „apropos“ nebo „about“.)

Jiný zdroj informací o editoru Emacs nabízí systém pro čtení dokumentace v hypertextovém formátu **Info**. Ten můžete inicializovat přímo z editoru Emacs tak, že stisknete kombinaci kláves Ⓒ-Ⓗ Ⓘ. Pak se vám objeví základní stránka systému Info, ve které najdete další pokyny, jak postupovat při vyhledávání informací.

Pracovní módy editoru Emacs

Každý buffer editoru Emacs je spjat s tzv. módem.[7] Módy byly zavedeny z toho důvodu, že například při psaní zprávy pro elektronickou poštu má uživatel jiné požadavky, než při psaní programu v jazyku C. Kdyby měl editor splňovat všechny požadavky najednou, bylo by jeho používání velmi komplikované.

Proto se autor editoru Emacs[8] rozhodl řešit tuto situaci pomocí módů. Chování editoru se mění podle toho, s jakým módem je konkrétní buffer spjat. Jednotlivé módy se od sebe liší vazbou mezi kombinacemi kláves a funkcemi, ale jsou zde i jiné rozdíly.

Nejzákladnějším módem je mód `fundamental`, který nemá žádné speciální funkce. Uvedeme zde, co o základním módu říká samotný editor Emacs:

```
Fundamental mode:
Major mode not specialized for anything in particular.
Other major modes are defined by comparison with this one.
```

7 Aby nebyla situace tak jednoduchá, existují zde hlavní módy (Major Modes) a vedlejší módy (Minor Modes). Zatím však pro nás nejsou podstatné.

8 Autor editoru Emacs je Richard Stallman.

Právě uvedenou informaci jsem získal takto: zadal jsem příkaz C-X B (což znamená vyvolání funkce `switch-to-buffer`) a zadal jsem „foo“ jako jméno bufferu, do kterého se chci přepnout. Protože buffer s takovým jménem nebyl doposud vytvořen, editor jej vytvořil a přepnul se do něj. Implicitně v něm nastavil základní mód (`fundamental-mode`). Všechny názvy módů mají tvar `<modename>-mode` a pomocí funkce „M-X“ lze pro každý buffer specifikovat mód explicitně právě pomocí jeho názvu. Abych získal více informací o základním módu, zadal jsem příkaz C-H M. Nakonec se zobrazila výše uvedená informace o základním módu.

Od základního módu je odvozen mírně užitečnější mód `text-mode` (textový mód), který má dva speciální příkazy: M-S pro funkci `center-paragraph` a M-S pro funkci `center-line`. Příkaz M-S znamená, že máte stisknout klávesu Shift, držet ji stisknutou, a pak stisknout klávesu Shift a „S“.

Nyní si můžete vyzkoušet vytvořit nový buffer a definovat v něm textový mód. Pak stiskněte kombinaci kláves C-H M. Objeví se vám informace o textovém módu. Možná nebudete všemu rozumět, ale jistě zde najdete užitečné informace.

V dalších oddílech si probereme některé z nejpoužívanějších módů. Budete-li s nimi pracovat, nezapomeňte na kombinaci kláves C-H M, kdykoliv budete chtít znát o aktuálním módu nějaké podrobnosti.

Programovací módy

Mód pro jazyk C

Jestliže použijete editor Emacs pro psaní programů v jazyku C, můžete po něm chtít, aby prováděl automatické odsazování. Soubory s příponou „.c“ nebo „.h“ zavádí editor Emacs automaticky v módu „c-mode“. To znamená, že jsou k dispozici některé speciální funkce vhodné pro psaní programů v jazyku C. Klávesa Tab je v módu `c-mode` svázána s funkcí `c-indent-command`. To znamená, že se po stisknutí klávesy Tab nevloží do textu znak Tab, ale dojde k automatickému odsazení řádku podle kontextu ve vytvářeném programu. Z toho vyplývá, že editor Emacs má jisté znalosti o syntaxi programů napsaných v jazyku C. V žádném případě však nepočítejte s tím, že vás bude editor upozorňovat na chyby v programu!

Navíc editor předpokládá, že jsou předcházející řádky odsazeny správně. Pokud v předcházejícím řádku chybí závorka, středník, složená závorka či cokoliv jiného, pak editor provede odsazení chybně. To je pro vás signál, že jste na něco zapomněli a můžete předcházející řádek opravit.

Uvedenou vlastnost editoru Emacs můžete využít ke kontrole oddělovačů ve zdrojovém programu. Nemusíte celý program číst od začátku a pracně hledat chybu, ale stačí zahájit odsazování řádků od začátku souboru pomocí klávesy Tab. Když dojde k nesprávnému odsazení, zkontrolujte předcházející řádek. Jinými slovy, v tomto směru za vás editor Emacs může udělat spoustu práce.

Mód pro jazyk Scheme

Tento mód pro vás bude užitečný jen tehdy, když používáte programovací jazyk Scheme. Tento jazyk není tak běžně používaným jazykem jako jazyk C nebo Pascal, ale v poslední době jeho popularita vzrůstá, a proto se mu budeme také věnovat. Většina toho, co platí pro jazyk Scheme, platí i pro jazyk Lisp.

Aby to nebylo tak jednoduché, v editoru Emacs jsou definovány dva módy pro programovací jazyk Scheme a každý uživatel se může rozhodnout, který mu bude lépe vyhovovat. Zaměříme se na popis módu s názvem `cmuscheme` a později, v oddíle o konfiguraci editoru Emacs, si vysvětlíme rozdíly mezi oběma módy. Nebuďte při čtení následujících řádků zneklidněni, když se váš edi-

tor Emacs bude chovat trochu jinak, než zde bude popisováno. V editoru lze konfigurovat prakticky vše a různé distribuce operačního systému Linux obsahují různě konfigurované editory Emacs.

V editoru Emacs můžete inicializovat interaktivní proces Scheme pomocí příkazu `M-X run-scheme`. Takto vytvoříte buffer nazvaný „`*scheme*`“, ve kterém se nachází obvyklý příkazový řádek jazyka Scheme. V tomto řádku můžete zadávat výrazy v jazyku Scheme ukončené klávesou Enter, jazyk Scheme je bude vyhodnocovat a bude zobrazovat příslušné odpovědi. Tímto způsobem můžete, máte-li interaktivně komunikovat s procesem Scheme, zadat definice všech svých funkcí a aplikací v příkazové řádku. Jiná situace nastane v případě, že máte zdrojový kód zapsaný v nějakém samostatném souboru. Pak by mohlo být jednodušší editovat tento soubor a definice odeslat prostřednictvím bufferu Scheme.

Jestliže do editoru Emacs zavedete soubor s příponou „`.ss`“ nebo „`.scm`“, pak se automaticky nastartuje mód **Scheme mode**. Pokud se z nějakých důvodů tento mód nenastartuje, můžete jej aktivovat prostřednictvím příkazu M-X `scheme-mode`. Tento mód ovšem není totožný s bufferem, ve kterém běží proces Scheme. V módu `scheme-mode` však máte k dispozici některé příkazy pro komunikaci s uvedeným bufferem.

Jestliže ve zdrojovém kódu jazyka Scheme máte vytvořenu definici nějaké funkce, pak ji stisknutím kláves C-C C-E můžete odeslat do bufferu, ve kterém běží proces Scheme. Pokud stisknete klávesy C-C M-E, pak se po odeslání definice funkce dostanete přímo do uvedeného bufferu a zde můžete zadávat interaktivní příkazy. Klávesy C-C C-L jsou určeny k zavedení souboru s kódem v jazyku Scheme (fungují pro buffer s procesem Scheme i pro buffer se zdrojovým kódem). Stejně jako u ostatních programovacích jazyků je klávesa Tab určena k odsazování řádků.

Pokud pracujete v bufferu, ve kterém běží proces Scheme, můžete používat klávesy M-P a M-N k zobrazení předcházejících nebo následujících příkazů (tzv. **input history**).

Předpokládejme, že ladíte nějakou funkci, řekněme '`rotate`', pro kterou jste použili argumenty, například:

```
> (rotate '(a b c d e))
```

pak můžete tento příkaz znovu zavést do příkazového řádku pomocí M-P. Nemusíte znovu v příkazovém řádku Scheme zadávat dlouhou sekvenci znaků a ušetříte tak spoustu času.

Editor Emacs má speciální módy jen pro několik málo programovacích jazyků. Patří mezi ně jazyk C, C++, Lisp a Scheme. (V současné době je dostupný mód snad pro každý programovací jazyk – např. Java, Python nebo JavaScript.)

Mód pro elektronickou poštu

Prostřednictvím editoru Emacs můžete také vytvářet a odesílat zprávy pro elektronickou poštu. K tomu je určen speciální buffer, tzv. „mail buffer“, který se inicializuje prostřednictvím klávesy C-X M. Nejdříve vyplníte položky „`To:`“ a „`Subject:`“ a pak se pomocí kombinace kláves C-N přenesete přes oddělovací čáru do pole, ve kterém můžete napsat zprávu. Oddělovací čáru nikdy needitujte ani nerušte, protože pak by editor Emacs nebyl schopen elektronickou zprávu odeslat. Oddělovací čáru používá k odlišení záhlaví od textu zprávy.

V poli pro text zprávy (pod oddělovací čarou) můžete uvést cokoliv. Po dokončení zprávy ji odešlete pomocí kláves C-C C-C. Editor Emacs obsah bufferu odešle.

Jak zvýšit efektivitu práce s editorem Emacs

Zkušení uživatelé editoru Emacs jsou až fanatičtí, pokud jde o zvyšování efektivity práce s editorem. Ve skutečnosti někdy stráví více času zvyšováním efektivity, než kolik pak ušetří. I když nechci, abyste se stali takovými fanatiky, existuje několik opatření, pomocí kterých se vám bude s editorem Emacs pracovat snadněji. Někteří zkušení uživatelé pohlížejí na začátečníky jako na hlupáky, protože neznají všechny „triky“ s editorem. Já osobně tento druh elitářství odsuzuji, protože jsem také kdysi byl začátečníkem. Nyní se však opět věnujme editoru.

Pro pohyb kurzoru existují některé další klávesy. Víme, že pro pohyb kurzoru o jeden znak doprava je určena kombinace kláves C-F. Chcete-li „poskočit“ s kurzorem o celé slovo, zadejte M-F. Vyzkoušejte si, co udělá příkaz M-B. To ale není vše. Pokud zapisujete věty tak, aby za poslední tečkou byly vždy dvě mezery, můžete pomocí klávesy M-E přeskakovat celé věty. Dvě mezery jsou podmínkou, protože jinak by editor nebyl schopen konec věty identifikovat. Opět si vyzkoušejte, jak funguje kombinace kláves M-A.

Možná si to ani neuvědomujete a používáte opakovaně kombinace kláves C-F k posunu kurzoru na konec řádku. Připomínáme, že k tomuto účelu je určena kombinace kláves C-E a k nastavení kurzoru na začátek řádku je určena kombinace kláves C-A. Možná, že často používáte kombinaci kláves pro posun kurzoru na následující řádek, a přitom byste měli použít kombinaci kláves C-V, kdy se vám zobrazí následující stránka. Zrovna tak využívejte kombinaci kláves M-V.

Pokud jste někde u konce řádku a všimnete si, že jste někde na začátku udělali chybu, pak nepoužívejte klávesy Backspace nebo Delete, abyste se dostali k chybnému místu. Museli byste jinak dobře napsaný řádek psát znovu. Místo toho používejte kombinaci kláves M-B, C-B, případně C-F, opravte chybu a pak se pomocí C-E vraťte na konec řádku.

Pokud zadáváte jméno souboru, nezadávejte je nikdy celé. Zadejte jen nezbytný počet znaků, které soubor jednoznačně identifikují. Pak vám editor Emacs zbývající znaky po stisknutí klávesy Tab nebo mezerníku automaticky doplní. Šetřte sebe a ne procesor!

Když píšete nějaký dlouhý text, používejte funkci pro automatické ukončování řádků. Kombinace kláves M-Q vyvolá funkci `fill-paragraph` a lze jej použít ve všech textových módech. Také jej můžete použít k „zarovnání“ již napsaného odstavce. Stačí nastavit kurzor na nějakou pozici uvnitř odstavce a stisknout M-Q.

Někdy je užitečné použít kombinaci kláves C-X U, kdy se editor pokusí vrátit zpět provedené změny. Editor Emacs v tomto případě odhadne, kolik změn má vrátit, a jeho odhad je obvykle velmi inteligentní. Kombinace kláves C-X U můžete použít opakovaně až do okamžiku, kdy editor vrátí poslední změnu, kterou si „pamatuje“.

Konfigurace editoru Emacs

Editor Emacs je tak velký a tak komplexní program, že má i svůj vlastní programovací jazyk. Vlastnosti editoru můžete modifikovat pomocí vlastních programů. Programovací jazyk integrovaný v editoru Emacs se jmenuje Emacs Lisp a je dialektem jazyka Lisp. Pokud máte s jazykem Lisp nějaké zkušenosti, pak se vám bude zdát opravdu přátelským. Pokud nemáte, ničeho se neobávejte. V tomto oddíle se nebudeme programováním v editoru Emacs zabývat příliš do hloubky a pokud se s jazykem Emacs Lisp budete chtít seznámit podrobněji, pak si prostudujte stránky dostupné v systému Info.

Většina funkcí editoru Emacs je definována pomocí kódu napsaného v jazyku Emacs Lisp.[9] Většina z těchto souborů je distribuována spolu s editorem a kolektivně se nazývají „Emacs Lisp library". Umístění této knihovny závisí na tom, jakým způsobem je editor Emacs instalován ve vašem systému. Nejčastěji jej můžete najít v adresáři: `/usr/lib/emacs/lisp`, `/usr/lib/emacs/19.19/lisp` a podobně. Číslo `19.19` značí verzi editoru Emacs a může se lišit od verze ve vašem systému.

Informace o uložení knihovny je uložena v interní proměnné **load-path** editoru Emacs, proto ji nemusíte pracně hledat v souborovém systému. Jestliže chcete zjistit hodnotu této proměnné, musíte ji **vyhodnotit**. To znamená, že musíte aktivovat interpretr Emacs Lisp. V editoru Emacs existuje speciální mód pro vyhodnocování výrazů jazyka Lisp zvaný **lisp-interaction-mode**. S tímto módem je obvykle spjat buffer „`*scratch*`". Pokud se vám jej nepodaří nalézt, vytvořte nový buffer s jakýmkoliv jménem a zadejte příkaz M-X `lisp-interaction-mode`.

Nyní se nacházíte v pracovním prostoru pro interaktivní komunikaci s interpretrem Emacs Lisp. Zadejte příkaz:

```
load-path
```

a pak stiskněte C-J. V módu interaktivní komunikace je kombinace kláves C-J svázán s funkcí `eval-print-last-sexp`. Slovo „sexp" znamená „s-expression" (**s-výraz**), což znamená vyváženou skupinu závorek – to je opravdu řečeno velmi zjednodušeně, ale brzy budete tušit, k čemu jsou takové výrazy užitečné při práci z jazykem Emacs Lisp. V každém případě po vyhodnocení proměnné `load-path` obdržíte zprávu, která bude vypadat přibližně takto:

```
load-path Ctrl+j
("/usr/lib/emacs/site-lisp/vm.5.35" "/home/kfogel/elithp"
"/usr/lib/emacs/site-lisp" "/usr/lib/emacs/19.19/lisp")
```

Toto hlášení nebude vypadat v každém systému stejně, protože závisí na způsobu instalace editoru Emacs. Uvedený příklad pochází z mého počítače s procesorem 386, na němž běží operační systém Linux. Jak vyplývá z předcházejícího výpisu, proměnná `load-path` je seznam řetězců. Každý řetězec uvádí adresář, který by mohl obsahovat soubory náležící do systému Emacs. Když potřebuje editor Emacs zavést soubory obsahující kód Lisp, hledá tyto soubory v uvedených adresářích v uvedeném pořadí. Pokud je v proměnné `load-path` uveden adresář, který v souborovém systému neexistuje, editor jej ignoruje.

Ve fázi startování se editor Emacs pokouší nalézt soubor `.emacs` ve vašem domovském adresáři. Proto platí, že pokud chcete mít svoji vlastní konfiguraci editoru Emacs, měli byste používat soubor `.emacs`. Nejvýznamnější konfigurační nastavení se týkají vazeb mezi kombinaci kláves a funkcemi, proto se jim nyní budeme věnovat.

```
(global-set-key "\Ctrl+cl" 'goto-line)
```

Funkce `global-set-key` má dva argumenty: prvním z nich je kombinace kláves a druhým je funkce, která se má po stisknutí tohoto klíče realizovat. Slovo „`global`" znamená, že uvedená vazba mezi kombinacemi kláves a funkcí bude platit ve všech hlavních módech. Existuje jiná funkce, `local-set-key`, jež nastavuje vazbu mezi klíčem a funkcí pro jediný buffer. V uvedeném příkladu jsme nastavili vazbu mezi kombinaci kláves C-C a funkcí `goto-line`. Při definici kombinace kláves se musí použít řetězec, což znamená, že se kombinace kláves musí uzavřít do uvozovek. Speciální syntaxe „`\Ctrl+<char>`" znamená, že se má stisknout klávesa Ctrl, držet stisknutá, a pak se má stisknout klávesa `<char>`. Podobně platí, že „`\Alt+<char>`" indikuje kombinaci s klávesou Meta.

9 Někdy se neoficiálně nazývá „Elisp".

Konfigurace vazby mezi kombinaci kláves a funkcí vypadá jednoduše, ale kde získat informace o funkci „goto-line“? Nebo naopak, předpokládejme, že chci kombinaci kláves C-C L svázat s funkcí, která umožní přeskočit na řádek specifikovaného pořadového čísla, ale jak mám zjistit jméno takové funkce?

V tomto okamžiku využijete vynikajících vlastností systému nápovědy, který je integrován v editoru Emacs. Jakmile se jednou rozhodnete, jaký druh funkce chcete použít, můžete použít editor Emacs k „vystopování“ jejího jména. Jedna sice rychlá, ale ne příliš přímočará metoda spočívá v tom, že se využije schopnosti editoru Emacs doplňovat jména funkcí. Měli byste si pamatovat, že kombinace kláves C-H F vyvolá nápovědu popisující funkci (t.j. vyvolá funkci `describe-function`). Pak stiskněte Tab, aniž cokoliv zadáte. Tak „donutíte“ editor Emacs doplnit prázdný řetězec. Jinými slovy, editor vypíše jména všech funkcí, které jsou v něm interně definovány. Uvedená operace bude chvíli trvat, protože takových funkcí je v editoru definováno opravdu mnoho.

Nyní stiskněte kombinaci kláves C-G, čímž funkci `describe-function` přerušíte. Nyní v editoru existuje buffer nazvaný „`*Completions*`“, který obsahuje požadovaný seznam všech interních funkcí editoru Emacs. Přepněte se do tohoto bufferu a pomocí postupného vyhledávání najděte funkci, kterou chcete použít. Můžete například využít předpokladu, že funkce pro přechod na řádek specifikovaného čísla bude ve svém názvu obsahovat slovo „`line`“. Proto zadejte vyhledávání slova „`line`“ a najdete tak všechny eventuality.

Vhodnější metoda spočívá v tom, že použijete kombinaci kláves C-H A (t.j. vyvoláte funkci `command-apropos`), kdy se vám zobrazí všechny funkce obsahující specifikovaný řetězec. Výstup z funkce `command-apropos` se poněkud hůře třídí, než výstup z funkce `describe-function`. Vyzkoušejte si obě metody a vyberte tu, která vám bude lépe vyhovovat.

Samozřejmě se může stát, že budete hledat funkci, která v editoru Emacs neexistuje. V takovém případě si ji budete muset napsat sami. Nebudeme zde popisovat, jak se v jazyku Emacs Lisp programuje. Doporučujeme prostudovat si příklady v knihovně Emacs Lisp a přečíst si stránky systému Info popisující jazyk Emacs Lisp. Pokud znáte někoho, kdo programování v jazyku Emacs Lisp ovládá, jistě vám pomůže.

Definice vlastních funkcí editoru Emacs není nic těžkého. Abych vás trochu navnadil, za poslední rok jsem jich bez problémů vytvořil 131. Vyžaduje to trochu zkušeností, ale programování v jazyku Emacs Lisp zvládnete poměrně rychle.

Další konfigurační možnosti spočívají v tom, že se v souboru `.emacs` nastaví hodnoty jistých proměnných. Přidejte například do vašeho souboru `.emacs` následující řádek a pak znovu nastartujte editor Emacs:

```
(setq inhibit-startup-message t)
```

Editor Emacs kontroluje ve fázi startování proměnnsou `inhibit-startup-message` a podle její hodnoty se rozhodne, zda zobrazovat jisté informace (o verzi editoru, zárukách a podobně) nebo ne. Uvedený výraz jazyka Lisp používá příkaz `setq` k nastavení proměnné `inhibit-startup-message` na hodnotu „t“ což je speciální hodnota v jazyce Lisp pro „true“. Opakem je hodnota „nil“ což je speciální hodnota pro „false“. V mém konfiguračním souboru `.emacs` se nacházejí některá nastavení, jež možná shledáte užitečnými:

```
(setq case-fold-search nil) ; gives case insensitivity in searching
;; make C programs ident the way I like them to:
(setq c-indent-level 2)
```

První výraz nastavuje způsob vyhledávání na tzv. case-insensitive, což znamená, že se při vyhledávání nerozlišují malá a velká písmena (a to dokonce i tehdy, když hledaný řetězec obsahuje buď pouze malá, nebo pouze velká písmena). V druhém výrazu se nastavuje odsazování řádků při psaní programů v jazyku C na hodnotu 2. Je to poněkud méně, než je implicitní nastavení, ale to záleží na individuálním vkusu a na tom, jaká hodnota odsazení učiní váš program napsaný v jazyku C čitelnějším.

V jazyku Lisp se komentářový řádek označuje znakem „;“. Editor Emacs ignoruje vše, co se nachází za tímto znakem. Výjimku tvoří případ, kdy se znak „;“ nachází uvnitř řetězce. Například:

```
;; Tyto dva řádky jsou v jazyku Lisp ignorovány, ale
;; následující s-výraz bude plně vyhodnocen:
(setq some-literal-string "An awkward pause; for no purpose.")
```

Doporučuje se, abyste změny ve zdrojových souborech pro jazyk Lisp komentovali, protože jinak například po šesti měsících určitě zapomenete, jakou změnu jste dělali. Komentář na celý řádek uvádějte dvojicí znaků `;;`. Pak bude editor Emacs správně provádět odsazování řádků.

To, co jsme si řekli o vyhledávání interních funkcí editoru Emacs, platí i pro vyhledávání proměnných. Chcete-li vytvořit seznam všech proměnných, zadejte příkaz Ⓒ-Ⓗ-Ⓒ (describe-variable), nebo použijte Ⓒ-Ⓗ Ⓒ-Ⓐ (apropos). Použijete-li druhou možnost, pak musíte počítat s tím, že vám příkaz vyhledá funkce a proměnné dohromady.

Soubory se zdrojovým kódem v jazyku Emacs Lisp mají implicitní příponu „`.el`“, například „`c-mode.el`“. Aby však mohl kód vytvořený v jazyku Emacs Lisp běžet rychleji, umožňuje editor provést jakousi **předkompilaci** (soubory označované jako „**byte-compiled**“) a pak mají tyto soubory implicitní příponu „`.elc`“. Výjimku, pokud jde o implicitní příponu, tvoří soubor `.emacs`, jenž příponu „`.el`“ nepotřebuje, neboť jej editor hledá automaticky ve fázi startování.

Chcete-li zavést interaktivně soubor s kódem v jazyku Lisp, použijte příkaz `M-X load-file`. Editor vás vyzve k zadání jména souboru a pak specifikovaný soubor zavede do své pracovní oblasti. Jestliže chcete zavést soubor „zevnitř“ jiného souboru napsaného v jazyku Lisp, postupujte takto:

```
(load "c-mode") ; tento příkaz zavede buď soubor c-mode.el, nebo c-mode.elc
```

Editor Emacs nejdříve k uvedenému jménu přidá příponu `.elc` a pokusí se jej nalézt v některém adresáři specifikovaném v proměnné `load-path`. Pokud jej nenajde, přidá příponu `.el` a hledání zopakuje. Jestliže není úspěšný ani v tomto případě, pak použije přímo řetězec předaný funkci `load`. Překlad můžete realizovat prostřednictvím příkazu `Alt+X byte-compile-file`. Pokud však zdrojový soubor často aktualizujete, pak to zpravidla nemá smysl. Soubor .emacs nikdy nepřekládejte, ani mu nepřidávejte příponu `.el`.

Bezprostředně po zavedení souboru `.emacs` vyhledá editor Emacs soubor `default.el` a zavede jej. Tento soubor je obvykle umístěn v adresáři uvedeném v proměnné `load-path` s názvem `site-lisp` nebo `local-elisp` či v nějakém podobném adresáři (podívejte se na proměnnou `load-path`). Uživatelé, kteří provádějí údržbu editoru Emacs používají soubor `default.el` k nastavení globálních konfigurací, které pak platí pro každého uživatele v systému. Soubor `default.el` by také neměl být kompilován, protože se v něm často provádějí změny.

Jestliže váš osobní soubor `.emacs` obsahuje nějakou chybu, pak se editor Emacs nebude pokoušet načíst soubor `default.el`, ale zastaví svou činnost a vypíše hlášení: „`Error in init file`“. Uvidíte-li takové hlášení, pak jste pravděpodobně udělali nějakou chybu v souboru `.emacs`.

V souboru `.emacs` se ještě vyskytuje jeden druh výrazů. Knihovna Emacs Lisp někdy nabízí více sad programů, které realizují tytéž funkce různým způsobem. To znamená, že musíte specifikovat tu sadu, která se má používat (implicitní sada nemusí být pro vás vždy tou nejlepší). Tyto sady se například uplatňují v oblasti interaktivního rozhraní pro jazyk Scheme. S editorem Emacs se standardně distribuují dvě sady funkcí interaktivního rozhraní pro jazyk Scheme: `xscheme` a `cmuscheme`.

```
prompt>Is /usr/lib/emacs/19.19/lisp/*scheme*
/usr/lib/emacs/19.19/lisp/cmuscheme.el
/usr/lib/emacs/19.19/lisp/cmuscheme.elc
/usr/lib/emacs/19.19/lisp/scheme.el
/usr/lib/emacs/19.19/lisp/scheme.elc
/usr/lib/emacs/19.19/lisp/xscheme.el
/usr/lib/emacs/19.19/lisp/xscheme.elc
```

Já osobně dávám přednost sadě `cmuscheme` před sadou `xscheme`, avšak editor Emacs implicitně používá sadu `xscheme`. Jak „donutit“ editor Emacs, aby byl nakonfigurován v souladu s mým přáním? Do souboru `.emacs` jsem vložil následující řádky:

```
;; notice how the expression can be broken across two lines. Lisp
;; ignores whitespaces, generally:
(autoload 'run-scheme "cmuscheme"
"Run an inferior Scheme, the way I like it". t)
```

Funkce `autoload` akceptuje jako argument jméno funkce (začínající apostrofem ') a „sdělí“ editoru Emacs, že je tato funkce definována v jistém souboru. Jméno tohoto souboru se předává jako druhý argument, tedy řetězec (buď s příponou „`.el`“, nebo „`.elc`“). Soubor pak bude vyhledán v adresářích definovaných v proměnné `load-path`.

Zbývající argumenty jsou volitelné, ale jsou nezbytné v tomto případě: Třetí argument je dokumentačním řetězcem pro specifikovanou funkci. Pokud použijete funkci `describe-function`, pak se jako popis k vyhledané funkci objeví právě tento řetězec.

Čtvrtý argument „sdělí“ editoru Emacs, že specifikovaná funkce může být volána interaktivně, t.j. pomocí kombinace kláves Alt+X. V tomto případě je to velmi důležité, protože jedině tak lze spustit proces Scheme v prostředí editoru Emacs pomocí příkazu `Alt+x run-scheme`.

Nyní, když je funkce `run-scheme` definována jako automaticky zaveditelná, co se stane, když se zadá příkaz `Alt+x run-scheme`? Editor Emacs vyhledá funkci `run-scheme`, zjistí, zda je automaticky zaveditelná a zavede specifikovaný soubor (v našem případě „`cmuscheme`“). Protože kompilovaný soubor `cmuscheme.elc` existuje, editor jej zavede. Tento soubor musí definovat funkci `run-scheme`, jinak dojde při jeho zavádění k chybě. Naštěstí tuto funkci skutečně definuje, proto jde vše hladce a já mám k dispozici své oblíbené rozhraní pro jazyk Scheme.[10]

Automatické zavedení funkce zajišťuje, že bude editor Emacs schopen funkci najít, až ji budete potřebovat. Navíc můžete nad automaticky zaváděnými funkcemi získat jistou kontrolu. Dále platí, že automaticky zaveditelné funkce šetří paměť spotřebovanou editorem Emacs, protože se tyto funkce zavádějí až v okamžiku, kdy jsou potřebné. Řada příkazů není ve fázi startování editoru ve skutečnosti definována. Místo toho jsou nastaveny jako automaticky zaveditelné funkce. Pokud takový příkaz nezadáte, nikdy se odpovídající funkce nezavede. Uvedená vlastnost automatického zavádění funkcí je pro editor Emacs (a hlavně pro uživatele) „životně“ důležitou. Kdyby editor ve

10 Jistě jste si všimli, že o rozhraní cmuscheme jsme hovořili již dříve. Proto byste si měli zavedení sady cmuscheme vyzkoušet.

fázi startování zaváděl všechny funkce, trvala by tato fáze asi dvacet minut a celá dostupná paměť vašeho počítače by byla obsazena. O automatické zavádění implicitních funkcí se nemusíte starat, editor Emacs je realizuje sám.

Kde získat další informace

V této kapitole jsme zdaleka neuvedli vše, co byste měli znát o editoru Emacs. Je zde zhruba jedno procento informací. Budete-li editor intenzivně využívat, pak budete potřebovat znát spoustu dalších „triků“ šetřících čas. Nejjednodušší bude, když vyčkáte, až budete muset řešit nějaký konkrétní problém. Pak vyhledáte funkci, která váš problém vyřeší.

Snad nejdůležitější je, abyste uměli plně využívat systém nápovědy integrovaný v editoru Emacs. Řekněme například, že budete chtít vložit do editovaného textu obsah jiného souboru. Snadno uhodnete, že asi existuje funkce `insert-file` (vložení souboru). Máte-li svou domněnku potvrdit, použijte příkaz Ⓒ-Ⓗ Ⓕ. V příkazovém řádku zadejte jméno funkce, kterou hledáte a o které si chcete přečíst nějaké informace. Protože víte, že editor Emacs je schopen doplňovat jména funkcí, můžete jako „počáteční“ odhad uvést řetězec „insert“. Pak stačí stisknout klávesu Tab. Objeví se vám seznam všech funkcí obsahujících řetězec „insert“ a funkce „`insert-file`“ je jedna z nich.

Nyní si můžete přečíst spoustu informací o funkci `insert-file` a také ji můžete použít – stačí zadat příkaz `Alt+x insert-file`. Chcete-li také zjistit, zda je tato funkce svázána s nějakým klíčem, zadejte příkaz `Ctrl+h w insert-file` a stiskněte Enter. Čím lépe budete znát vlastnosti systému nápovědy v editoru Emacs, tím snáze naleznete potřebné informace. Máte-li navíc dostatek erudice zkoumat nové věci a ochotu učit se, ušetříte mnoho času.

Jestliže si chcete objednat kopii manuálu k editoru Emacs a/nebo manuál „*Emacs Lisp Programming*“, pošlete objednávku na adresu:

Free Software Foundation
675 Mass Ave
Cambridge, MA 02139
USA

Oba tyto manuály jsou distribuovány v elektronické podobě spolu s editorem Emacs a jsou čitelné prostřednictvím systému Info (použijte klávesu Ⓒ-Ⓗ Ⓘ, pomocí které inicializujete rozhraní mezi editorem Emacs a systémem Info). Na druhé straně, cena za uvedené manuály je opravdu mírná a navíc se získané peníze budou věnovat na vývoj kvalitního volně šířitelného programového vybavení. Také chceme poznamenat, že pomocí kombinace kláves Ⓒ-Ⓗ Ⓒ-Ⓒ si můžete zobrazit informace o licenčních podmínkách platných pro editor Emacs. Jsou určitě zajímavější, než si teď myslíte, a pomohou vám vyjasnit si pojem „volné programové vybavení“. Pokud si myslíte, že pojem „volné programové vybavení“ znamená, že daný program nic nestojí, pak si licenční podmínky rychle přečtěte.

KAPITOLA 9

Konfigurace operačního systému Unix

Konfigurace příkazového interpretu bash

Filosofie operačního systému Unix se od filosofie jiných operačních systémů podstatně liší v jedné věci. Autoři Unixu se nepokoušeli předvídat všechny potřeby všech uživatelů. Místo toho se pokusili navrhnout operační systém tak, aby si každý uživatel pracovní prostředí svého operačního systému snadno upravil sám podle svých potřeb. Konfigurace jednotlivých programů operačního systému Unix se definuje prostřednictvím tzv. **konfiguračních souborů**. Někdy jsou označovány jako „ini-files" nebo „rc files" nebo dokonce jako „dot files" (jedná se zpravidla o soubory, jejichž jména začínají tečkou). Pokud si vzpomínáte, tak jména souborů začínající znakem „." nejsou normálně příkazem `ls` zobrazována.

Nejdůležitější konfigurační soubory jsou ty, které používá příkazový interpret. Implicitním příkazovým procesorem v operačním systému Linux je `bash` a právě jemu bude věnována tato kapitola. Než začneme popisovat, jak příkazový interpret `bash` konfigurovat, podívejme se na soubory, které `bash` vyhledává.

Inicializace příkazového interpretu bash

Existuje několik různých způsobů, jak příkazový procesor `bash` spustit. Tzv. **login-shell** se automaticky spouští poté, co se přihlásíte do systému.

Další způsob spočívá ve spuštění **interaktivního příkazového procesoru** (interactive shell). To je jakýkoliv příkazový procesor, který se prezentuje příkazovým řádkem. Příkazový interpret, jenž se spustí bezprostředně po přihlášení se do systému, je také interaktivní. Jiným příkladem interaktivního příkazového interpretu je program xterm spuštěný z prostředí X Window.

Existují také **neinteraktivní příkazové interprety.** Tyto procesory se používají k vykonávání příkazů uvedených v souboru, jako jsou dávkové soubory s příponou .BAT v operačním systému MS-DOS. Analogií v operačním systému Unix jsou tzv. **skripty**. Skript příkazového procesoru je něco jako miniprogram. Jsou sice výrazně pomalejší než kompilovaný program, ale snadno se vytvářejí a modifikují.

Příkazové procesory v operačním systému Unix používají v závislosti na typu následující inicializační soubory:

Typ příkazového procesoru	inicializační soubor
Interaktivní příkazový interpret spuštěný při přihlášení se do systému	`.bash_profile`
Interaktivní příkazový procesor	`.bashrc`
Neinteraktivní příkazový procesor	skript příkazového procesoru

Inicializační soubory

Protože většina uživatelů chce mít stejné uživatelské prostředí bez ohledu na to, jaký typ příkazového procesoru se spustí, začneme konfiguraci tím, že do konfiguračního souboru `.bash_profile` vložíme jednoduchý příkaz „source ~/.bashrc“. Příkaz source „sdělí“ příkazovému procesoru, že má jeho argument interpretovat jako skript. To znamená, že se při každém spuštění skriptu `.bash_profile` také spustí skript `.bashrc`.

Nyní budeme přidávat příkazy do našeho souboru `.bashrc`. Do souboru `.bash_profile` se zadávají pouze příkazy, které se mají spouštět při přihlášení se do systému.

Vytváření aliasů

Jakým způsobem lze měnit konfigurační nastavení? Hned uvedeme příklad, který si do svého konfiguračního souboru `.bashrc` zadává devadesát procent uživatelů operačního systému Unix:

```
alias ll="ls -l"
```

Příkaz definuje tzv. **alias** (druhé jméno) příkazu. V našem případě se jméno příkazu `ll` expanduje na příkaz příkazového procesoru `"ls -l"`. Za předpokladu, že příkazový procesor bash přečetl uvedenou definici ve vašem konfiguračním souboru `.bashrc`, pak má příkaz `ll` stejný efekt jako příkaz `ls -l`. Přitom ušetříte polovinu stisknutých kláves. Když zadáte v příkazovém řádku `ll` a stisknete Enter, příkazový procesor zadaný příkaz rozvine podle definice a pak realizuje příkaz `ls -l`. Ve skutečnosti v systému příkaz `ll` neexistuje, ale příkazový procesor `bash` jej automaticky transformuje na platný program.

Některé příklady aliasů jsou uvedeny na této straně dole. Můžete si je vložit do vašeho souboru `.bashrc`. Zvláště zajímavý je první z nich. Je-li definován alias `ls="ls -F"`, pak po každém zadání příkazu `ls` bude automaticky aplikovat volbu `-F`. Platí, že se alias nikdy nepokusí rekurzivně rozšířit sám sebe. Uvedený příklad demonstruje nejčastěji používaný způsob automatického přidávání voleb do příkazů.

Všimněte si znaku „#“. Ve skriptech příkazového procesoru se tento znak používá k označení komentáře a příkazový procesor zbytek řádku za znakem # ignoruje.

Dále jste si mohli všimnout několika dalších věcí. Především se v některých definicích nevyskytují uvozovky, například v definici `pu`. Platí totiž, že pokud se na pravé straně znaku rovná se (=) vyskytuje jediné slovo, pak se uvozovky nemusejí uvádět.

Nic by se nestalo, kdyby zde uvozovky byly uvedeny. Určitě však uvozovky používejte v případě, když budete definovat nové jméno pro příkaz s volbami a/nebo argumenty. Například:

```
alias rf="refrobnicate -verbose -prolix -wordy -o foo.out"
alias ls="ls -F"              # give characters at end of listing
alias ll="ls -l"              # special ls
alias la="ls -a"
alias ro="rm  *~; rm.*~"      # removes backup files created by Emacs
alias rd="rmdir"              # saves typing!
alias md="mkdir"
alias pu=pushd         # pushd, popd, and dirs weren't covered
alias po=popd          # manual---you might want to look them up
alias ds=dirs          # in the bash manpage
# these all are just keyboard shortcuts
alias to="telnet cs.oberlin.edu"
alias ta="telnet altair.mcs.anl.gov"
alias tg="telnet wombat.gnu.ai.mit.edu"
alias tko="talk kold@cs.oberlin.edu"
alias tjo="talk jimb@cs.oberlin.edu"
alias mroe="more"             # spelling correction!
alias moer="more"
alias email="emacs -f rmail"  # my mail reader
alias ed2="emacs -d floss:0 -fg \"grey95\" -bg \"grey50\""
                                    # one way of invoking emacs
```

Asi se vám zdá, že poslední alias má divně umístěné uvozovky:

```
alias ed2="emacs -d floss:0 -fg \"grey95\" -bg \"grey50\""
```

Jak asi tušíte, chtěl jsem umístit uvozovky do samotných voleb. Proto jsem musel před každou uvozovku vložit obrácené lomítko. Příkazový procesor pak takovou uvozovku nebude interpretovat obvyklým způsobem.

Všimněte si také dvou definic pro opravu chybně zadaného příkazu `more`. Pokud omylem zadám „`mroe`“ nebo „`moer`“, bude příkazový procesor „vědět“, že jsem chtěl zadat „`more`“. Aliasy neinterferují s argumenty předávanými programům. To znamená, že například příkaz

```
/home/larry# mroe hurd.txt
```

bude fungovat dobře.

Určitě platí, že správné používání definice aliasů představuje polovinu konfiguračních možností, které jsou u příkazových procesorů k dispozici. Při práci v operačním systému Unix si všímejte, které příkazy často zadáváte, a pak si pro ně pořiďte druhá jména, tedy zkratky. Zjistíte, že se vám pak bude pracovat mnohem radostněji.

Systémové proměnné

V souboru `.bashrc` se definují další důležitá konfigurační nastavení prostřednictvím systémových proměnných (environment variables). Co jsou to systémové proměnné? Pojďme na to z druhé strany: předpokládejme, že čtete dokumentaci k programu `fruggle` a že narazíte na následující věty:

> *Fruggle normally looks for its configuration file, .frugglerc, in the user's home directory. However, if the environment variable FRUGGLEPATH is set to a different filename, it will look there instead.*

Každý program běží v nějakém **prostředí** a to je definováno příkazovým interpretem, jenž program volá.[1] Lze si představit, že prostředí existuje uvnitř příkazového interpretu. Programátoři mají k dispozici speciální funkci pro získání hodnoty systémové proměnné a program `fruggle` tuto funkci využívá. To znamená, že ověří hodnotu v systémové proměnné `FRUGGLEPATH`. Pokud není tato systémová proměnná definována, pak program použije soubor `.frugglerc` ve vašem domovském adresáři. Pokud však je definována, program `fruggle` použije její hodnotu místo implicitního souboru `.frugglerc`.

Nyní si uveďme ukázku, jak změnit prostředí v příkazovém procesoru `bash`:

```
/home/larry# export PGPPATH=/home/larry/secrets/pgp
```

Pod příkazem `export` si můžete představit následující větu: „Exportuj tuto proměnnou do prostředí, ze kterého budu spouštět program, tak, aby proměnná byla z tohoto programu viditelná.“ Později uvidíte, že jsou i jiné důvody pro používání příkazu `export`.

Uvedenou systémovou proměnnou používá program `pgp` pro šifrování, jehož autorem je Phil Zimmerman. Program `pgp` implicitně používá váš domovský adresář jako adresář, ve kterém hledá jisté soubory (šifrovací klíče). Také tento adresář využívá k vytvoření dočasných pracovních souborů. Nastavením systémové proměnné `PGPPATH` jsem změnil pracovní adresář na `/home/larry/secrets/pgp`. Abych zjistil přesné jméno této systémové proměnné, musel jsem si přečíst manuál k programu `pgp`. Systémové proměnné se zpravidla uvádějí velkými písmeny a končí na „`PATH`“.

Je užitečné vědět, jak hodnotu systémové proměnné zjistit:

```
/home/larry# echo $PGPPATH
.pgp
/home/larry#
```

Všimněte si, že jsme před jméno systémové proměnné uvedli znak `$`. Jedině tak lze zjistit hodnotu systémové proměnné. Pokud byste znak dolaru neuvedli, dostali byste následující výpis:

```
/home/larry# echo PGPPATH
PGPPATH
/home/larry#
```

Znak dolaru se používá k vyhodnocení systémové proměnné, avšak pouze v kontextu s příkazovým procesorem – přesněji s příkazovým procesorem, jenž realizuje interpretaci příkazu. V jakých případech realizuje příkazový procesor interpretaci?

Proměnná	Obsahuje	Příklad
`HOME`	Váš domovský adresář	`/home/larry`
`TERM`	Typ vašeho terminálu	`xterm, vt100, console`
`SHELL`	Cesta k vašemu příkazovému procesoru	`/bin/bash`
`USER`	Jméno vašeho účtu	`larry`
`PATH`	Seznam adresářů, ve kterých se automaticky vyhledávají programy ke spuštění	`/bin:/usr/local/bin:/usr/bin/X11`

Tabulka 9.1 – Některé důležité systémové proměnné

1 Nyní vidíte, proč jsou příkazové procesory tak důležité. Představte si, že byste museli definovat celé prostředí, kdykoliv budete spouštět nějaký program!

Je to v těch případech, kdy zadáváte příkaz z příkazového řádku nebo kdy příkazový procesor čte příkazy z nějakého souboru, například ze souboru `.bashrc`.

Existuje další důležitý příkaz, pomocí kterého lze získat informace o prostředí. Tímto příkazem je `env`. Po zadání příkazu `env` se vám zobrazí seznam všech systémových proměnných. Jste-li uživateli systému X Window, pak bude tento seznam velmi dlouhý, proto používejte příkaz `env | more`.

Některé systémové proměnné jsou opravdu důležité, proto jsou uvedeny v tabulce 9.1. Tyto systémové proměnné se automaticky definují po přihlášení se do systému. Není proto nutné je nastavovat v souboru `.bashrc` nebo `.bash_login`.

Nyní se blíže podívejme na systémovou proměnnou `TERM`. Abychom jí mohli porozumět, podívejme se zpět do historie operačního systému Unix. Operační systém musí znát jisté údaje o vaší konzole, aby mohl realizovat takové funkce, jako je zápis znaků na obrazovku, pohyb kurzoru v textovém řádku a podobně. V počátcích rozvoje výpočetní techniky výrobci terminálů průběžně rozšiřovali jejich vlastnosti: nejdříve inverzní video, později znaky pro evropské jazyky, dokonce i první funkce pro kreslení (připomínáme, že jde o dobu, kdyse ještě nikomu ani nesnilo o grafických oknech a myších). Každá nová vlastnost však přinášela programátorům problémy: jak měli vědět, co terminál podporuje a co ne? Jak měli podporovat nové vlastnosti a přitom „neodepsat“ staré terminály?

V operačním systému Unix najdete odpověď na tyto otázky v souboru `/etc/termcap`. Soubor `/etc/termcap` obsahuje seznam všech terminálů, o kterých váš operační systém „ví,“ a dále informace, jakým způsobem se má řídit kurzor. Pokud systémový správce dostane nový terminál, pak pouze do souboru `/etc/termcap` přidá záznam vztahující se k novému terminálu a nemusí pracně konfigurovat celý operační systém Unix. Někdy je situace ještě jednodušší. Kdysi se stal terminál vt100 od firmy Digital Equipment Corporation jakýmsi pseudostandardem, který převážná většina moderních terminálů respektuje a umí emulovat.

V operačním systému Linux je někdy hodnota systémové proměnné `TERM` nastavena jako console, což znamená emulaci terminálu vt100 s některými speciálními funkcemi.

Další proměnná, `PATH`, je rovněž kritickou systémovou proměnnou z hlediska funkčnosti příkazového procesoru. Zde uvádím nastavení na mém počítači:

```
/home/larry# env | grep ^PATH
PATH=/home/larry/bin:/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/TeX/bin
/home/larry#
```

Systémová proměnná `PATH` obsahuje seznam adresářů oddělených dvojtečkou, ve kterých systém automaticky vyhledává spustitelné programy. Když například zadám příkaz `ls` a stisknu klávesu Enter, bude příkazový procesor `bash` hledat program `ls` nejdříve v adresáři `/home/larry/bin`, který jsem vytvořil pro ukládání mých vlastních programů.

Program `ls` jsem však nenapsal já (ten byl pravděpodobně napsán ještě před tím, než jsem se narodil), proto zde příkazový procesor tento příkaz nenašel. Jako další prohledává příkazový procesor adresář /bin. A zde program `ls` našel. Protože soubor `ls` je spustitelný program, přestane příkazový procesor dále hledat a spustí jej. Může se stát, že v jiném adresáři bude také uložen spustitelný program `ls` (například v adresáři `/usr/bin`), ale příkazový procesor jej nespustí, pokud mu to výslovně nepřikážete:

```
/home/larry# /usr/bin/ls
```

Systémová proměnná PATH existuje hlavně proto, abychom nemuseli při každém spouštění nějakého programu zadávat celou cestu k tomuto programu. Zadáte-li tedy jakýkoliv příkaz, prohledá příkazový procesor všechny adresáře uvedené v systémové proměnné PATH. Když jej najde, spustí jej, a pokud ne, vypíše následující zprávu:

```
/home/larry# clubly
clubly: command not found
```

Všimněte si, že moje systémová proměnná PATH neobsahuje aktuální adresář, tedy „.“. Pokud by obsahovala, pak by vypadala takto:

```
/home/larry# echo $PATH
.:/home/larry/bin:/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/TeX/bin
/home/larry#
```

Zda zadávat nebo nezadávat aktuální adresář do systémové proměnné PATH je předmětem diskuse v kuloárech okolo operačního systému Unix. Problém tkví v tom, že aktuální adresář v systémové proměnné PATH by mohl představovat „díru“ v bezpečnosti systému. Předpokládejme, že se přepnete do adresáře, ve kterém někdo nechal virový program „Trójský kůň“ a nazval jej ls. Vy nic zlého netušíte a zadáte příkaz ls, což je přirozené, chcete-li se seznámit s novým adresářem. Protože je aktuální adresář uveden v systémové proměnné PATH jako první, spustí příkazový procesor právě tuto verzi příkazu ls. I když jste nechtěli udělat nic špatného, rozpoutali jste ve vašem systému virovou nákazu. Proti takovým haváriím neexistuje dost účinná ochrana. Virový program může spustit osoba, která ani nemá privilegia uživatele root. Stačí, aby měla právo zapisovat do adresáře, ve kterém se virový program nachází. Dokonce to může být její domovský adresář.

Pokud jde o váš systém, je pravděpodobné, že jeden uživatel neklade druhému různé nástrahy ve formě „Trojských koňů“ a že kolektiv uživatelů je založen na přátelských a kolegiálních vztazích. Avšak ve velkých systémech s mnoha uživateli (jako jsou například universitní počítače), mohou být desítky programátorů, které byste nejraději ani nepotkali. Z toho vyplývá, že zařazení aktuálního adresáře do systémové proměnné PATH závisí na konkrétní situaci.[2]

Skutečný způsob, kterým je nastavena systémová proměnná PATH v mém počítači, je dostatečně ilustrativní. Zde je uvedeno nastavení v souboru .bashrc:

```
export PATH=${PATH}:.:${HOME}/bin:/bin:/usr/bin:/usr/local/bin:/usr
  /bin/X1,1:/usr/TeX/bin
```

Zde jsem využil skutečnosti, že proměnná HOME je nastavena před tím, než příkazový procesor bash přečte můj soubor .bashrc. Systémová proměnná PATH je tedy nastavena prostřednictvím systémové proměnné HOME. Složené závorky („{...}“) představují další úroveň uvozovek. Oddělují to, co se vyhodnotí po znaku $, proto bude příkazový procesor moci jednoznačně identifikovat následující část příkazu (tedy „/bin“ v tomto případě). Zde uvádíme další příklad efektu, který vyvolají složené závorky:

```
/home/larry# echo ${HOME}foo
/home/larryfoo
/home/larry#
```

2 Pamatujte si, že program z aktuálního adresáře můžete vždy spustit explicitně, tedy například „./foo“.

Bez uvedení složených závorek se nevypíše nic, protože neexistuje proměnná prostředí `HOMEfoo`:

```
/home/larry# echo $HOMEfoo
/home/larry#
```

Nyní se podívejme na další zvláštní konstrukci v uvedeném příkazu. Jaký je význam řetězce „$PATH"? Tento řetězec zařadí do proměnné prostředí `PATH` hodnotu proměnné prostředí `PATH` dříve definovanou. Kde byla nastavena původní hodnota? Soubor `/etc/profile` slouží jako jistý typ globálního souboru `.bash_profile`, kde se nacházejí nastavení společná pro všechny uživatele systému. Jeden soubor s globálním nastavením má jistou výhodu. Má-li například systémový správce přidat do proměnné prostředí `PATH` důležitou cestu, stačí, když to udělá v souboru s globální platností a nemusí opravovat inicializační soubory všech uživatelů. Proto je proměnná prostředí `PATH` již nastavena a vy ji můžete použít.

Prostřednictvím jisté proměnné prostředí můžete také specifikovat, jak má vypadat váš příkazový řádek. Tato proměnná prostředí má označení jako `PS1`. Předpokládejme, že dáváte přednost tomu, aby se stále zobrazovala cesta do aktuálního adresáře. Pak nastavte proměnnou prostředí `PS1` takto:

```
export PS1='$PWD#'
```

Jak asi tušíte, jsou zde ve skutečnosti použity dvě proměnné prostředí. První, která se nastavuje, je `PS1` a druhou je `PWD`. Proměnná prostředí `PWD` může znamenat buď „Print Working Directory", nebo „`PATH` to Working Directory". Vyhodnocení proměnné prostředí však probíhá uvnitř jednoduchých uvozovek. Jednoduché uvozovky slouží k vyhodnocení vnitřního výrazu. Výsledkem tohoto vyhodnocení je proměnná prostředí `PWD`. Kdybyste použili výraz `export PS1=$PWD`, pak by se vám stále zobrazoval ten adresář, který byl aktuální v době vyhodnocení tohoto výrazu. Trochu jsme pohled na vyhodnocování proměnných prostředí zkomplikovali, ale to není tak důležité. Pouze si pamatujte, že budete-li chtít zobrazovat v příkazovém řádku aktuální adresář, budete muset použít jednoduché uvozovky.

Možná, že budete chtít podobný příkazový řádek jako v operačním systému MS-DOS. Pak zadejte export `PS1='$PWD>'`. Chcete-li mít v příkazovém řádku jméno vašeho systému, zadejte `PS1="hostname"'>'`.

V posledním příkladu jsme použili nový typ uvozovek, tzv. zpětné uvozovky. Tyto uvozovky ve skutečnosti nic nechrání. Výraz uzavřený ve zpětných uvozovkách se vyhodnotí jako příkaz a výstup se objeví na místě zpětných uvozovek.

Vyzkoušejte si příkazy `echo` `` `ls` `` nebo `wc` `` `ls` ``. Čím více budete seznámeni s příkazovým procesorem, tím užitečnější vám budou uvedené techniky.

V souboru `.bashrc` je mnoho dalších možností, jak konfigurovat váš příkazový procesor, ale zde není dostatek prostoru všechny prodiskutovat. Další podrobnosti si nastudujte v manuálových stránkách k příkazovému procesoru `bash` nebo se zeptejte zkušených uživatelů. Dále uvádíme kompletní soubor `.bashrc`, abyste si jej mohli nastudovat. Představuje typický standard, i když je proměnná prostředí `PATH` poněkud dlouhá.

```
# some random stuff:
ulimit -c unlimited
export history_control=ignoredups
export PS1='$PWD>'
umask 022
# application-specific PATHs:
```

```
export MANPATH=/usr/local/man:/usr/man
export INFOPATH=/usr/local/info
export PGPPATH=${HOME}/.pgp
# make the main PATH:
homepath=${HOME}:~/bin
stdpath=/bin:/usr/bin:/usr/local/bin:/usr/ucb:/etc:/usr/etc:/urs
  /games
pubpath=/usr/public/bin:/usr/gnusoft/bin:/usr/local/contribs/bin
softpath=/usr/bin/X11:/usr/local/bin/X11:/usr/TeX/bin
export PATH=.:${homepath}:${stdpath}:${pubpath}:${softpath}
# Technically, the curly braces were not necessary, because the
  colons
# were valid delimiters; nevertheless, the curly braces are a good
# habit to get into, and they can't hurt.
# aliases
alias ls="ls -CF"
alias fg1="fg %1"
alias fg2="fg %2"
alias tba="talk sussman@tern.mcs.anl.gov"
alias tko="talk kold@cs.oberlin.edu"
alias tji="talk jimb@totoro.bio.indiana.edu"
alias mroe="more"
alias moer="more"
alias ll="ls -l"
alias la="ls -a"
alias ro="rm *~; rm.*~"
alias rd="rmdir"
alias pu=pushd
alias po=popd
alias ds=dirs
alias to="telnet cs.oberlin.edu"
alias ta="telnet altair.mcs.anl.gov"
alias tg="telnet wobat.gnu.ai.mit.edu"
alias email="emacs -f rmail"
alias ed2="emacs -d floss:0 -fg \"grey95\" -bg \"grey50\""
function gco
{
gcc -o $1 $1.c -g
}
```

Inicializační soubory systému X Window

Většina lidí dává při své práci přednost grafickému uživatelskému prostředí, kterým je v opeačním systému Unix systém X Window. Jestliže jste seznámeni s operačním systémem Macintosh nebo Microsoft Windows, pak vám systém X Window bude připadat velmi známý. Méně známé vám však budou připadat konfigurační možnosti, které systém X Window nabízí.

V případě operačního systému Macintosh nebo Microsoft Windows se konfigurace realizuje přímo v grafickém uživatelském prostředí. Když chcete například změnit barvu pozadí, klepnete myší na novou barvu nějakého speciálního inicializačního grafického programu. V systému X Window se implicitní hodnoty řídí textovými soubory, které můžete přímo editovat – jinými slovy, chcete-li například zadat novou barvu pozadí, musíte její jméno uvést v jistém inicializačním souboru.

Nikdo nepopírá, že inicializační metody v systému X Window jsou poněkud těžkopádnější než u komerčních programů. Domnívám se, že tendence zachovávat textově orientované inicializační metody dokonce i v grafickém uživatelském prostředí tkví v tom, že například systém X Window vytvořila poměrně nesourodá skupina programátorů, kteří až příliš lpí na tradicích dodržovaných v operačních systémech typu Unix. Tyto tendence se mohou v příštích verzích systému X Window změnit (alespoň doufám, že se změní), ale nyní se budeme zabývat inicializací prostřednictvím textových souborů. Tak alespoň máte k dispozici velmi flexibilní a přesnou kontrolu nad konfigurací.

Nejdůležitější konfigurační soubory pro systém X Window jsou tyto:

`.xinitrc` Skript, který systém X Window spouští ve fázi startování.

`.twmrc` Soubor, který čte správce oken `twm`.

`.fvwmrc` Soubor, který čte správce oken `fvwm`.

Všechny uvedené soubory by měly být uloženy ve vašem domovském adresáři.

Soubor `.xinitrc` je jednoduchý skript příkazového procesoru, jenž se automaticky spouští ve fázi startování systému X Window. Může dělat vše, co mohou dělat ostatní skripty, avšak nejvíce ze všeho má smysl jej použít k nastartování různých programů systému X Window a k nastavení parametrů. Posledním příkazem v souboru .xinitrc je obvykle příkaz pro spuštění správce oken, například `/usr/bin/X11/twm`.

Jaký druh konfiguračních nastavení má smysl v souboru `.xinitrc` uvádět? Snad nějaká volání programu `xsetroot`, čímž si můžete nastavit pozadí okna a vlastnosti kurzoru. Dále volání programu `xmodmap`, jenž předá serveru[3] informace o tom, jak má interpretovat signály z vaší klávesnice. Všechny ostatní programy, které se mají spustit pokaždé spolu se systémem X Window (například `xclock`).

Zde uvádím některé řádky z mého souboru `.xinitrc`. Váš konfigurační soubor bude jistě vypadat jinak, proto je považujte za pouhý příklad.

```
#!/bin/sh
# The first line tells the operating system which shell to use in
# interpreting this script. The script itself ought to be marked as
# executable; you can make it so with "chmod +x ~/.xinitrc".
# xmodmap is a program for telling the X server how to interpret your
# keyboard's signals. It is *definitely* worth learning out. You
# can do "man xmodmap", "xmodmap -help", "xmodmap -grammar", and more.
# I don't guarantee that the expressions below will mean anything on
# your system (I don't even guarantee that they mean anything on
# mine):
xmodmap -e 'clear Lock'
xmodmap -e 'keycode 176 = Control_R'
xmodmap -e 'add control = Control_R'
xmodmap -e 'clear Mod2'
xmodmap -e 'add Mod1 = Alt_L Alt_R'
# xset is a program for setting other parameters of the X server:
xset m 3 2 & # mouse parameters
xset s 600 5 & # screen saver prefs
xset s noblank # ditto
xset fp+ /home/larry/x/fonts # for cxterm
```

3 Server představuje hlavní proces systému X Window, tedy program, se kterým musejí všechny ostatní programy běžící pod systémem X Window komunikovat, pokud chtějí využívat grafické prostředí. Tyto ostatní programy se nazývají klienti a systém jako celek bývá označován termínem systém „klient-server“.

```
# To find out more, do "xset -help"
# Tell the X server to superimpose fish.cursor over fish.mask, and use
# the resulting pattern as my mouse cursor:
xsetroot -cursor /home/lab/larry/x/fish.cursor /home/lab/larry/x/fish.mask &
# a pleasing background pattern and color:
xsetroot -bitmap /home/lab/larry/x/pyramid.xbm -bg tan
# todo: xrdb here? What about .Xdefaults file?
# You should do "man xsetroot", or "xsetroot -help" for
# more information on the program above.
# A client program, the imposing circular color-clock by Jim Blandy:
/usr/local/bin/circles
# Maybe you'd like to have a clock on your screen at all time:
/usr/bin/X11/xclock -digital &
# Allow client program running at occs.cs.oberlin.edu to display
# themselves here, do the same thing for juju.mcs.anl.gov:
xhost occs.cs.oberlin.edu
xhost juju.mcs.anl.gov
# You can simply tell the X server to allow clients running on any
# other host (a host being a remote machine) to display here, but this
# is a security hole -- those clients can be run by someone else,
# and watch your keystrokes as you type your password or something!
# However, if you wanted to do it anyway, you could use a "+" to stand
# for all possible hostnames, instead of a specific hostname, like
# this:
# xhost +
# And finally, run the window namager:
/usr/bin/X11/twm
# Some people prefer other window manager. I use twm, but fvwm is
# often distributed with Linux too:
# /usr/bin/X11/fvwm
```

Všimněte si, že některé programy běží na pozadí. Jsou to ty programy, jejichž zadání je ukončeno znakem `&`. Jiné programy se na pozadí nespouštějí. Rozdíl spočívá v tom, že se startují současně se systémem X Window a běží na pozadí, dokud systém X Window neukončíte. Jiné se vykonají bezprostředně a ihned skončí – jedním z nich je `xsetroot`, který pouze nastaví základní okno a chování kurzoru a pak skončí.

Jakmile se nastartuje správce oken, načte svůj vlastní inicializační soubor. Tento inicializační soubor řídí takové věci, jako je nastavení nabídek, pozice oken, řízení ikon a další. Pokud používáte jako správce oken program `twm`, pak tímto inicializačním souborem je soubor `.twmrc`, který se nachází ve vašem domovském adresáři. Jestliže však používáte `fvwm`, pak je inicializačním souborem soubor `.fvwmrc`. V následujících oddílech se budeme zabývat pouze těmito dvěma, protože se běžně distribuují s operačním systémem Linux.

Konfigurace programu twm

Soubor `.twmrc` není skript příkazového procesoru – je napsán v jazyku speciálně vyvinutém pro program `twm`.[4] Při konfiguraci prostřednictvím souboru `.twmrc` si uživatelé nejraději hrají s nastavením oken (barvy a podobně) a nabídek. Zde uvádíme příklad konfiguračního souboru `.twmrc`:

4 Toto je jedna z odpuzujících vlastností inicializačních souborů: některé z nich mají svůj vlastní příkazový jazyk. To znamená, že uživatel musí být velmi zdatný a rychle se naučit další jazyk. Předpokládám, že takové vymyšlenosti mohly být zajímavé v ranných dobách operačního systému Unix, kdy programátoři stále toužili po něčem novém. Dnes je ale vyčerpávající učit se stále dokola nové a nové syntaxe jazyků, které konec konců slouží jedinému programu.

```
# Set colors for various parts of windows. This has a great
# impact no the "feel" of your environment.
Color
{
BorderColor "OrangeRed"
BorderTitleForeground "Black"
BorderTitleBackground "Black"
TitleForeground "black"
TitleBackground "gold"
MenuForeground "black"
MenuBackground "LightGrey"
MenuTitleForeground "LightGrey"
MenuTitleBackground "LightSlateGrey"
MenuShadowColor "black"
IconForeground "DimGray"
IconBackground "Gold"
IconBorderColor "OrangeRed"
IconManagerForeground "black"
IconManagerBacground "honeydew"
}
# I hope you don't have a monochrome system, but if you do...
Monochrome
{
BorderColor "black"
BorderTitleForeground "black"
BorderTitleBackground "white"
TitleForeground "black"
TitleBackground "white"
}
# I created beifang.bmp with the program "bitmap". Here I tell twm to
# use it as the default highlight pattern on windows' title bars:
Pixmaps
{
TitleHighlight "/home/larry/x/beifang.bmp"
}
# Don't worry about this stuff, it's only for power users :-)
BorderWidth 2
TitleFont "-adobe-new century schoolbook-bold-r-normal--14-140-75-75
  -p-87-iso8859-1"
MenuFont "6x13"
IconFont "lucidasans-italic-14"
ResizeFont "fixed"
Zoom 50
RandomPlacement
# These programs will not get a window titlebar by default:
NoTitle
{
"stamp"
"xload"
"xclock"
"xlogo"
"xbiff"
"xeyes"
```

```
"oclock"
"xoid"
}
# "AutoRaise" means that a window is brought to the front whenever the
# mouse pointer enters it. I find this annoying, so I have turned
# off. As you can see, I inherited my .twmrc from people who also
# did not like autoraise.
AutoRaise
{
"nothing" # I don't like auto-raise # Me either # nor I
}
# Here is where the mouse button functions are defined. Notice the
# pattern: a mouse button pressed on the root window, with no modifier
# key being pressed, always brings up a menu. Other locations usually
# result in window manipulation of some kind, and modifier keys are
# used in conjuction with mouse buttons to get at the more
# sophisticated window manipulations.
#
# You don't have to follow this pattern in your own .twmrc -- it's
# entirely up to you how you arrange your environment.
# Button = KEYS : CONTEXT : FUNCTION
# ----------------------------------
Button1 = : root : f.menu "main"
Button1 = : title : f.raise
Button1 = : frame : f.raise
Button1 = : icon : f.iconify
Button1 = m : window : f.iconify
Button2 = : root : f.menu "stuff"
Button2 = : icon : f.move
Button2 = m : window : f.move
Button2 = : title : f.move
Button2 = : frame : f.move
Button2 = s : frame : f.zoom
Button2 = s : window : f.zoom
Button3 = : root : f.menu "z"
Button3 = : title : f.lower
Button3 = : frame : f.lower
Button3 = : icon : f.raiselower
# You can write your own functions; this one gets used in the menu
# "windowops" near the end of this file:
Function "raise-n-focus"
{
f.raise
f.focus
}
# Okay, below are the actual menus reffered to in the mouse button
# section. Note that many of these menu entries themselves call
# sub-menus. You can have as many levels of menus as you want, but be
# aware that recursive menus don't work. I tried it.
menu "main"
{
"Vanilla" f.title
"Emacs" f.menu "emacs"
```

```
"Logins" f.menu "logins"
"Xlock" f.menu "xlock"
"Misc" f.menu "misc"
}
# This allows me to invoke emacs on several different machines. See
# the section on .rhosts files for more information about how this
# works:
{
"Emacs" f.title
"here" !"/usr/bin/emacs &"
"" f.nop
"phylo" !"rsh phylo \"emacs -d floss:0\" &"
"geta" !"rsh geta \"emacs -d floss:0\" &"
"darwin" !"rsh darwin \"emacs -d floss:0\" &"
"ninja" !"rsh ninja \"emacs -d floss:0\" &"
"indy" !"rsh indy \"emacs -d floss:0\" &"
"oberlin" !"rsh cs.oberlin.edu \"emacs -d floss.life.uiuc.edu:0\" &"
"gnu" !"rsh gate-1.gnu.ai.mit.edu \"emacs -d floss.life.uiuc.edu:0\" &"
}
# This allows me to invoke xterms on several different machines. See
# the section on .rhosts files for more information about how this
# work:
menu "logins"
{
"Logins" f.title
"here" !"/usr/bin/X11/xterm -ls -T `hostname` -n `hostname` &"
"phylo" !"rsh phylo \"xterm -ls -display floss:0 -T phylo\" &"
"geta" !"rsh geta \"xterm -ls -display floss:0 -T geta\" &"
"darwin" !"rsh darwin \"xterm -ls -display floss:0 -T darwin\" &"
"ninja" !"rsh ninja \"xterm -ls -display floss:0 -T ninja\" &"
"indy" !"rsh indy \"xterm -ls -display floss:0 -T indy\" &"
}
# The xlock screensaver, called with various options (each of which
# gives a different pretty picture):
menu "xlock"
{
"Help" !"xlock -mode hop &"
"Qix" !"xlock -mode qix &"
"Flame" !"xlock -mode flame &"
"Worm" !"xlock -mode worm &"
"Swarm" !"xlock -mode swarm &"
"Hop NL" !"xlock -mode hop -nolock &"
"Qix NL" !"xlock -mode qix -nolock &"
"Flame NL" !"xlock -mode flame -nolock &"
"Worm NL" !"xlock -mode worm -nolock &"
"Swarm NL" !"xlock -mode swarm -nolock &"
}
# Miscellaneous programs I run occassionally:
menu "misc"
{
"Xload" !"/usr/bin/X11/xload &"
"XV" !"/usr/bin/X11/xv &"
"Bitmap" !"/usr/bin/X11/bitmap &"
```

```
"Tetris" !"/usr/bin/X11/xtetris &"
"Hextris" !"/usr/bin/X11/xhextris &"
"XRoach" !"/usr/bin/X11/xroach &"
"Analog Clock" !"/usr/bin/X11/xclock -analog &"
"Digital Clock" !"/usr/bin/X11/xclock -digital &"
}
# This is the one I bound to the middle mouse button:
menu "stuff"
{
"Chores" f.title
"Sync" !"/bin/sync"
"Who" !"who | xmessage -file - -columns 80 -lines 24 &"
"Xhost +" !"/usr/bin/X11/xhost + &"
"Rootclear" !"/home/larry/bin/rootclear &"
}
# X functions that are sometimes convenient
menu "x"
{
"X Stuff" f.title
"Xhost +" !"xhost + &"
"Refresh" f.refresh
"Source .twmrc" f.twmrc
"(De)Iconify" f.iconify
"Move Window" f.move
"Resize Window" f.resize
"Destroy Window" f.destroy
"Window Ops" f.menu "windowops"
"" f.nop
"Kill twm" f.quit
}
# This is submenu from above:
menu "windowops"
{
"Window Ops" f.title
"Show Icon Mgr" f.showiconmgr
"Hide Icon Mgr" f.hideiconmgr
"Refresh" f.refresh
"Refresh Window" f.winrefresh
"twm version" f.version
"Focus on Root" f.unfocus
"Source .twmrc" f.twmrc
"Cut File" f.cutfile
"(De)Iconify" f.iconify
"DeIconify" f.deiconify
"Move Window" f.move
"ForceMove Window" f.forcemove
"Resize Window" f.resize
"Raise Window" f.raise
"Lower Window" f.lower
"Raise or Lower" f.raiselower
"Focus on Window" f.focus
"Raise-n-Focus" f.function "raise-n-focus"
"Destroy Window" f.destroy
"Kill twm" f.quit
}
```

Věřte mi, že to není zdaleka nejobsáhlejší soubor `.twmrc`, jaký jsem kdy viděl. Je vysoce pravděpodobné, že nějaký příklad souboru .twmrc bude obsažen ve vaší distribuci systému X Window. Prohledejte adresář `/usr/lib/X11/twm/` nebo `/usr/X11/lib/X11/twm`. Zde byste měli uvedený soubor nalézt.

Často dělají uživatelé chybu a zapomínají uvádět znak `&` na konec příkazů. Pokud systém X Window „zatuhne“ právě když spustíte nějaký příkaz, pak pravděpodobně tkví příčina v tom, že jste zapomněli uvést znak `&`. V takovém případě ukončete systém X Window kombinací kláves Ctrl-Alt-Backspace, opravte soubor `.twmrc` a spusťte systém X Window znovu.

Konfigurace programu fvwm

Pokud používáte jako správce oken program `fvwm`, prohledejte tento adresář:

```
/usr/lib/X11/fvwm/ nebo /usr/X11/lib/X11/fvwm.
```

Zde najdete nějaké příklady konfiguračních souborů.

Poznámka: O programu `fvwm` nic nevím. Asi bych byl schopen něco vyčíst z příkladů konfiguračních souborů, ale zůstal bych pouze čtenářem a těžko bych dokázal něco vysvětlovat. Zájemcům o program fvwm a jeho konfiguraci doporučuji přečíst si příslušné manuálové stránky a prostudovat příklady konfiguračních souborů nacházejících se ve výše zmíněných adresářích.

Ostatní inicializační soubory

Za zmínku stojí následující inicializační soubory:

`.emacs`	Inicializační soubor editoru Emacs. Editor jej čte ve fázi startování.
`.netrc`	Soubor obsahující implicitní jména a hesla pro `ftp`.
`.rhosts`	Zpřístupňuje váš účet vzdáleným systémům.
`.forward`	Inicializační soubor pro automatické přesměrování elektronické pošty.

Inicializační soubor pro editor Emacs

Pokud používáte editor Emacs jako primární editor, pak má pro vás soubor `.emacs` mimořádný význam. Podrobně jsme jej popsali v kapitole 8.

Implicitní nastavení pro FTP

V souboru `.netrc` můžete mít uložena některá implicitní nastavení pro `ftp`. Následující řádky obsahují příklad takového souboru:

```
machine floss.life.uiuc.edu login larry password fishSticks
machine darwin.life.uiuc.edu login larry password fishSticks
machine geta.life.uiuc.edu login larry password fishSticks
machine phylo.life.uiuc.edu login larry password fishSticks
machine ninja.life.uiuc.edu login larry password fishSticks
machine indy.life.uiuc.edu login larry password fishSticks


machine clone.mcs.anl.gov login fogel password doorm@
machine osprey.mcs.anl.gov login fogel password doorm@
machine tern.mcs.anl.gov login fogel password doorm@
machine altair.mcs.anl.gov login fogel password doorm@
machine dalek.mcs.anl.gov login fogel password doorm@
```

```
machine juju.mcs.anl.gov login fogel password doorm@
machine sunsite.unc.edu login anonymous password larry@cs.oberlin.edu
```

Každý řádek souboru `.netrc` specifikuje jméno počítače, přihlašovací jméno, které se má použít jako implicitní pro tento počítač, a heslo. Uvedená nastavení vám ušetří velké množství času, protože jinak byste při přihlašování se k jednotlivým serverům `ftp` museli pokaždé zadávat dlouhé identifikační řetězce. Pokud se budete přihlašovat k serveru ftp, jehož jméno je uvedeno v souboru `.netrc`, pokusí se program `ftp` aplikovat uživatelské jméno a heslo z tohoto souboru.

Při spouštění programu `ftp` můžete pomocí parametru -n specifikovat, že nechcete použít implicitní nastavení ze souboru .netrc. Příkaz pak bude mít tvar „`ftp -n`“.

Musíte se ujistit, že soubor `.netrc` jste schopni číst pouze vy. K nastavení příslušných přístupových práv použijte program `chmod`. Kdyby soubor `.netrc` mohli číst jiní uživatelé, pak by mohli odhalit vaše hesla platná pro servery ftp uvedené v tomto souboru.

Takový případ by představoval značnou „bezpečnostní díru“. Naštěstí program `ftp` a ostatní programy, které čtou soubor `.netrc`, odmítnou pokračovat ve své činnosti, pokud zjistí, že přístupová práva k tomuto souboru nejsou v pořádku.

Další informace týkající se souboru .netrc si najděte v manuálových stránkách prostřednictvím příkazu „`man .netrc`“ nebo „`man ftp`“.

Povolení snadného vzdáleného přístupu k vašemu účtu*

Jestliže se ve vašem domovském adresáři nachází soubor `.rhosts`, pak lze ze vzdálených počítačů spouštět aplikace na vašem počítači. Uveďme si příklad. Předpokládejme, že jste přihlášeni na počítači `cs.oberlin.edu`. Na počítači `floss.life.uiuc.edu` je správně konfigurován soubor `.rhosts`.

Pak ze svého počítače můžete na počítači `floss.life.uiuc.edu` spustit aplikaci, jejíž výstup bude přesměrován na vaši obrazovku, a dokonce se před tím nebudete muset přihlašovat a zadávat heslo.

Soubor `.rhosts` může vypadat takto:

```
frobnozz.cs.knowledge.edu jsmith
aphrodite.classics.havhaahd.edu wphilps
frobbo.hoola.com trixie
```

Formát souboru je velmi jednoduchý: jméno počítače následované jménem uživatele. Předpokládejme nyní, že uvedený příklad je mým skutečným souborem `.rhosts` na vzdáleném počítači `floss.life.uiuc.edu`. To by znamenalo, že bych mohl spustit program na počítači floss a výstup by mohl být přesměrován na kterýkoliv počítač uvedený v tomto souboru, pokud bych byl přihlášen jako odpovídající uživatel.

Přesný mechanismus, prostřednictvím kterého uživatel uskutečňuje spouštění vzdáleného programu, je realizován programem `rsh`, jehož název je zkratkou pro „remote shell“, tedy „vzdálený příkazový procesor“. Ten nastartuje příkazový procesor na vzdáleném počítači a spustí specifikovaný program. Uveďme si příklad:

```
frobbo$ whoami
trixie
frobbo$ rsh floss.life.uiuc.edu "ls ~"
```

---

* Poznámka korektora: Používání programu `rsh` a `rlogin` je v současné době nahrazeno podobným programem `ssh`, který je bezpečnější.

```
foo.txt mbox url.ps snax.txt
frobbo$ rsh floss.life.uiuc.edu "more ~/snax.txt"
[nyní se zobrazí stránky souboru snax.txt]
```

Uživatel trixie na počítači `floss.life.uiuc.edu`, který má soubor `.rhosts` uvedený v předcházejícím příkladě, umožňuje uživateli trixie na počítači `frobbo.hoola.com` spouštět programy z počítače floss.

Soubor `.rhosts` bude pracovat správně, i když nemáte na všech počítačích stejné uživatelské jméno. Chcete-li „sdělit" vzdálenému počítači, jaké jméno uživatele chcete použít k přihlášení se, použijte při zadání příkazu `rsh` volbu „`-l`". Pokud takový uživatel na vzdáleném počítači existuje a pokud zde existuje soubor `.rhosts` obsahující jméno vašeho (t.j. lokálního) počítače a jméno uživatele, pak se příkaz `rsh` provede úspěšně.

```
frobbo$ whoami
trixie
frobbo$ rsh -l larry floss.life.uiuc.edu "ls ~"
[zde se objeví seznam souborů mého adresáře na počítači floss]
```

Uvedený příklad bude fungovat, pokud má uživatel `larry` na počítači `floss.life.uiuc.edu` takový soubor `.rhosts`, jenž umožňuje uživateli `trixie` z počítače `frobbo.hoola.com` spouštět programy. Není relevantní, zda se jedná nebo nejedná o tutéž osobu. Důležitá jsou pouze jména uživatelů, jména počítačů a záznamy v souboru `.rhosts`.

V souboru `.rhosts` mohou být uvedeny i jiné kombinace – například jméno uživatele následující za jménem vzdáleného počítače se může vynechat a tak umožnit kterémukoliv uživateli na vzdáleném počítači spouštět programy na vašem počítači. To je však spojeno s jistými riziky. Nepovolaná osoba by mohla například zrušit vaše soubory. Pokud se rozhodnete pro takto organizovaný soubor `.rhosts`, pak byste se měli ujistit, že právo číst soubor `.rhosts` máte pouze vy.

Přesměrování elektronické pošty

Kromě jiných souborů můžete také mít soubor `.forward`, který nelze přímo nazvat „inicializačním" souborem. Pokud tento soubor obsahuje adresu elektronické pošty, pak veškeré zprávy elektronické pošty budou odesílány na tuto adresu. Soubor je užitečný v případě, kdy máte účet na několika různých systémech, ale elektronickou poštu chcete číst pouze na jednom.

Na vašem počítači se mohou nacházet i jiné inicializační soubory. Jejich počet se liší podle operačního systému, jenž používáte, a také podle programů, které máte nainstalovány. Jeden ze způsobů, jak získat více informací o inicializačních souborech, spočívá v tom, že si prostudujete soubory ve vašem domovském adresáři začínající tečkou. Není sice garantováno, že všechny takové soubory jsou inicializační, ale převážná většina z nich určitě bude.

Kde si můžete prohlédnout některé příklady

Jestliže na svém počítači máte instalován operační systém Linux a jestliže máte přístup do sítě Internet, pak se můžete přihlásit prostřednictvím služby telnet do systému `floss.life.uiuc.edu`. Přihlašte se jako „guest" a heslo uveďte jako „explorer". Připravili jsme pro vás spoustu příkladů, z nichž většina je uložena v adresáři `/home/kfogel`. Tyto příklady si můžete zkopírovat a prostudovat. Buďte ale opatrní. Počítač floss nepředstavuje zcela zabezpečený systém a když se budete dostatečně snažit, podaří se vám získat přístupová práva uživatele root. Dali jsme přednost důvěře před neustálou ostražitostí a doufáme, že toho nikdo nezneužije.*

* Poznámka korektora: Autor knihy to zřejmě myslel jako vtip.

KAPITOLA 10

Komunikace s ostatními systémy

Moderní operační systémy typu Unix jsou dobře přizpůsobeny ke komunikaci s ostatními počítači, což znamená, že jsou vybaveny prostředky pro práci v síti. Dva různé počítače s operačním systémem Unix si mohou vyměňovat informace mnoha způsoby. Tato kapitola je věnována metodám, pomocí kterých budete moci komunikovat prostřednictvím sítě s ostatními počítači.

Budeme se zabývat elektronickou poštou, zájmovými skupinami a některými základními programy pro komunikaci.

Elektronická pošta

Mezi nejpopulárnější vlastnosti operačního systému Unix patří možnost odesílat a přijímat zprávy elektronické pošty. Máte-li k dispozici elektronickou poštu, nepotřebujete papír, inkoust a pero, obálky, známky a nesrovnatelně pomalejší poštovní službu.

Odesílání elektronické pošty

Potřebujete-li odeslat zprávu elektronickou poštou, stačí zadat příkaz `mail username` a pak zapsat vaši zprávu.

Předpokládejme například, že chcete odeslat zprávu elektronickou poštou uživateli jménem sam:

```
/home/larry# mail sam
Subject: The user documentation
Just testin out the mail system.
EOT
/home/larry#
```

Program `mail` je velmi jednoduchý. Podobně jako program `cat` čte text ze standardního vstupu řádek po řádku, dokud nenarazí na znak „konec textu“ jenž je reprezentován klávesou Ctrl-D. To znamená, že po dokončení textu zprávy stisknete Enter a pak Ctrl-D.

Příkaz `mail` představuje nejrychlejší způsob, jak odeslat zprávu elektronické pošty, a je užitečný zejména ve spojení s prostředky pro přesměrování vstupu/výstupu či rourami. Jestliže chcete například odeslat soubor `report1` uživateli jménem „Sam“, můžete použít příkaz `mail sam < report1` nebo dokonce „`sort report1 | mail sam`“.

Používání příkazu `mail` je však spojeno s velkou nevýhodou. Jestliže uděláte v nějakém řádku chybu, pak ji již nemůžete opravit. Proto vám doporučuji (pokud nepoužijete možnost s přesměrováním vstupu/výstupu) k odesílání elektronické pošty používat editor Emacs. Postup jsme popsali v oddílu Pracovní módy editoru Emacs.

Čtení zpráv elektronické pošty

`mail [`*`user`*`]`

Program `mail` poskytuje poněkud těžkopádnou možnost číst zprávy elektronické pošty. Zadáte-li příkaz `mail` bez jakýchkoliv parametrů, objeví se vám následující hlášení:

```
/home/larry# mail
No mail for larry
/home/larry#
```

Předpokládejme, že chcete odeslat zprávu elektronickou poštou sami sobě, abyste si mohli vyzkoušet způsoby jejího čtení:

```
/home/larry# mail larry
Subject: Frogs!
and toads!
EOT
/home/larry# echo "snakes" | mail larry
/home/larry# mail
Mail version 5.5. 6/1/90 Type ? for help.
"/usr/spool/mail/larry" 2 messages new
>N 1 larry Tue Aug 30 18:11 10/211 "Frogs!"
N 2 larry Tue Aug 30 18:12 10/211
&
```

Příkazový řádek uvnitř programu pro čtení elektronické pošty je uvozen znakem ampersand („&"). Umožňuje specifikovat spoustu jednoduchých příkazů a po zadání znaku ? a Enter zobrazuje stručnou nápovědu.

Základními příkazy pro práci s programem mail jsou:

`t` *`message-list`*	Na obrazovce se zobrazí zprávy uvedené v seznamu *`message-list`*.
`d` *`message-list`*	Zprávy uvedené v seznamu *`message-list`* se zruší.
`s` *`message-list file`*	Zprávy uvedené v seznamu *`message-list`* se uloží do souboru *`file`*.
`r` *`message-list`*	Odpověď na zprávy – program `mail` zahájí proces sestavování nových zpráv, které budou odeslány každému, kdo vám odeslal zprávu uvedenou v seznamu *`message-list`*.
`q`	Program `mail` se ukončí a uloží každou zprávu, která nebyla zrušena příkazem `d` do souboru mbox ve vašem domovském adresáři.

Jak vypadá seznam *`message-list`*? Skládá se ze seznamu čísel oddělených mezerami, nebo může být vyjádřen ve formě intervalu, tedy například 2-4 (což znamená „2 3 4"). Také můžete uvést uživatelské jméno odesílatele. Například `t sam` zobrazí všechny zprávy odeslané uživatelem sam. Jestliže se seznam *`message-list`* neuvede, zobrazí se vždy poslední zpráva.

Se čtením zpráv elektronické pošty je spojeno několik problémů. Především platí, že pokud má zpráva více řádků, než se vejde na obrazovku, program mail se nezastaví! Takou zprávu budete muset uložit do souboru a pak si ji prohlédnout prostřednictvím příkazu `more`. Dále program mail nemá dobré prostředky pro čtení starých zpráv, tedy zpráv uložených do souborů.

Editor Emacs má prostředek pro čtení zpráv elektronické pošty, jenž se jmenuje `rmail`. V této knize se však programem rmail nebudeme zabývat. V operačním systému Linux jsou navíc k dispozici další programy pro čtení elektronické pošty, jako je `elm` nebo `pine`.*

Jak vyhledat uživatele sítě

Příkaz finger

Příkaz `finger` vám umožní získat informace o ostatních uživatelích vašeho systému nebo o uživatelích sítě Internet. Jméno příkazu nepochybně vzniklo jako zkratka s reklamním podtextem ve firmě AT&T.

`finger [-slpm] [`*`user`*`][`*`@machine`*`]`

Volitelné parametry v příkazu `finger` mohou být mírně matoucí. Pomocí příkazu `finger` můžete získat informace o lokálním uživateli (například „sam“), o jiném počítači (například „@lionsden“), informace o uživateli vzdáleného počítače (například „sam@liondsen“) nebo informace o lokálním počítači (neuvede se žádný parametr).

Příkaz finger má další zajímavou vlastnost. Pokud se pokusíte získat informace o uživateli, jehož jméno přesně neznáte, pokusí se příkaz `finger` najít jméno sám (zkouší různé kombinace založené na původně zadaném jménu). To znamená, že když například zadám příkaz `finger Greenfield`, obdržím zprávu, že účet `sam` existuje pro jméno `Sam Greenfield`.

```
/home/larry# finger sam
Login: sam Name: Sam Greenfield
Directory: /home/sam Shell: /bin/tcsh
Last login Sun Dec 25 14:47 (EST) on tty2
No Plan.
/home/larry# finger greenfie@gauss.rutgers.edu
[gauss.rudgers.edu]
Login name: greenfie In real life: Greenfie
Directory: /gauss/u1/greefie Shell: /bin/tcsh
On since Dec 25 15:19:41 on ttyp0 from tiptop-slip-6439
13 minutes Idle Time
No unread mail
Project: You must be joking!
No Plan.
/home/larry# finger
Login Name Tty Idle Login Time Office Office Phone
larry Larry Greenfield 1 3:51 Dec 25 12:50
larry Larry Greenfield p0 Dec 25 12:51
/home/larry#
```

* Poznámka korektora: Samozřejmě můžete používat i jiné poštovní klienty, např. i Netscape Communicator.

Použijete-li u příkazu `finger` volbu `-s`, pak se vám vždy zobrazí stručný výpis (stejný, který obdržíte, když program `finger` použijete k získání informací o počítači) a pokud použijete volbu `-l`, zobrazí se vám vždy kompletní výpis (i tehdy, když program `finger` použijete k získání informací o počítači). Po zadání volby `-p` se nezobrazí informace ze souborů `.forward`, `.plan` a `.project`. Zadáte-li volbu `-m` a žádáte-li pouze informace o uživateli, pak se vám zobrazí pouze přihlašovací jméno.

Soubory .plan a .project

Nyní bychom měli vysvětlit, jaký je význam souborů `.plan` a `.project`. Jedná se o soubory, které jsou uloženy v domovském adresáři uživatele a jejichž obsah se zobrazí pokaždé, když je na daného uživatele aplikován program `finger`. Soubory `.plan` a `.project` si můžete vytvořit sami – jediné omezení spočívá v tom, že ze souboru .project se zobrazuje pouze první řádek.

Dále platí, že každý, kdo chce aplikovat program `finger` musí mít možnost procházet vaším domovským adresářem (`chmod a+x ~/`) a každý musí být schopen číst soubory `.plan` a `.project` (`chmod a+r ~/.plan ~/.project`).

Používání systémů vzdálenými počítači

`telnet` *`vzdálený systém`*

Hlavní prostředek pro využívání vzdálených systémů založených na operačním systému Unix představuje program Telnet. (V současné době se spíše používá program `ssh`, který na rozdíl od příkazu `telnet` umožňuje šifrovanou komunikaci se vzdáleným systémem.) Používání programu telnet je velmi jednoduché:

```
/home/larry# telnet lionsden
Trying 128.2.36.41...
Connected to lionsden
Escape character is '^]'.
lionsden login:
```

Jak vidíte z následujícího příkladu, po zadání příkazu `telnet` budete vyzváni k přihlášení se ke vzdálenému systému. Uživatelské jméno lze zadat jakékoliv (ovšem heslo musíte znát správné) a pak je vám vzdálený systém k dispozici téměř stejně jako váš lokální systém.

Normální způsob ukončení programu `Telnet` spočívá v odhlášení se, ale je také možnost zadat znak Escape, kterým je zpravidla Ctrl-]. Tak obdržíte nový příkazový řádek uvozený řetězcem `telnet>`. Nyní stačí napsat `quit` a pak stisknout klávesu Enter – spojení se přeruší a program `Telnet` se ukončí. Pokud si to rozmyslíte a nechcete relaci ukončit, stiskněte pouze klávesu Enter.

Pokud jste uživateli systému X Window, pak si pro komunikaci se vzdáleným počítačem otevřete nové terminálové okno – například prostřednictvím příkazu „`xterm -title "lionsden" -e telnet lionsden &`“. Uvedený příkaz otevře nové terminálové okno, ve kterém automaticky poběží program `Telnet`. Jestliže takový příkaz budete používat častěji, pak doporučujeme, abyste si pro něj vytvořili alias.

Přenášení souborů

`ftp` *`vzdálený systém`*

Normální způsob přenášení souborů zprostředkovává v operačním systému Unix program `ftp`, což je zkratka pro „**file transfer protocol**“. Po zadání příkazu `ftp` budete vyzváni, abyste se přihlásili ke vzdálenému systému téměř stejným způsobem, jako v případě programu `telnet`. Pak se vám zobrazí speciální příkazový řádek.

Nyní máte k dispozici několik příkazů běžných v operačním systému Unix, jež můžete aplikovat v příkazovém řádku programu `ftp`. Například příkaz `cd` i zde slouží k přepínání se mezi adresáři a příkaz `ls` slouží k zobrazení seznamu souborů uložených v aktuálním adresáři.

Navíc máte k dispozici dva důležité příkazy: `get` a `put`. Příkaz `get` je určen k přenosu souborů ze vzdáleného počítače do vašeho lokálního počítače a příkaz `put` slouží k opačnému úkolu. Oba příkazy jako implicitní používají váš domovský adresář a lokální adresář na vzdáleném počítači (který můžete měnit prostřednictvím příkazu `cd`).

S používáním programu `ftp` je spojen jeden problém. Spočívá v rozlišení mezi znakovými a binárními soubory. Protokol pro přenos dat programem `ftp` je velmi starý a implicitně předpokládá, že přenášené soubory jsou textové. Aplikuje-li se tento implicitní přenos na binární soubory, pak budou s největší pravděpodobností po přenosu poškozeny. Před přenosem binárního souboru proto používejte příkaz `binary`.

Program `ftp` ukončíte příkazem `bye`.

Putování po stránkách WWW

WWW, neboli World Wide Web (doslova „celosvětová pavučina“) zřejmě představuje nejpopulárnější způsob využívání Internetu. Skládá se ze stránek, z nichž každá je spojena s tzv. lokátorem URL (**uniform resource locator**). Lokátory URL jsou komické řetězce následujícího tvaru: `http://www.rutgers.edu/`. Stránky jsou vytvořeny v jazyku HTML (**hypertext markup language**).

Jazyk HTML umožňuje autorům stránek WWW začlenit do dokumentů odkazy na kterékoliv jiné stránky WWW (nebo obrázky) umístěné kdekoliv jinde v celosvětovém systému WWW. Když uživatel čte nějaký dokument, pak se pouhým klepnutím na odkaz (prezentovaný klíčovým slovem nebo tlačítkem) přenese na jinou stránku WWW, která může být uložena v počítači na druhém konci světa.

`netscape [`*`url`*`]`

Nejpopulárnějším programem pro prohlížení stránek WWW je v operačním systému Linux program Netscape od firmy Netscape Corporation. Program Netscape může běžet pouze pod systémem X Window. *Pozn.: V poslední době jsou populární i další prohlížeče jako Mozilla, Kongueror, Galeon a další.

Program Netscape je velmi jednoduchý, pokud jde o ovládání. Je založen na knihovně Motif a připomíná program napsaný pro Microsoft Windows (samozřejmě, že pro Microsoft Windows také existuje verze programu Netscape). Odkazy na další stránky WWW jsou v programu Netscape zobrazeny modře. Stačí na odkaz klepnout levým tlačítkem myši a vzápětí se vám zobrazí nová stránka WWW.

Operační systém Linux podporuje i jiné programy pro prohlížení stránek WWW, například program lynx, což je textově orientovaný program, proto není schopen zobrazovat obrázky a jiné grafické prvky dokumentů WWW. Je ovšem schopen pracovat pod systémem X Window.

`lynx [`*`url`*`]`

Naučit se pracovat s programem `lynx` je poněkud obtížnější, než naučit se pracovat s programem `Netscape`. Při práci si budete muset zvyknout na používání kurzorových kláves. Klávesy „šipka nahoru“ a „šipka dolů“ jsou určeny k přepínání mezi odkazy na dané stránce WWW, klávesa „šipka doprava“ zobrazí novou stránku (na kterou odkazuje zvýrazněný odkaz) a klávesa „šipka doleva“ je určena k zobrazení předcházející stránky. K ukončení programu `lynx` použijte klávesu Q. Program `lynx` má mnohem více příkazů, jejichž kompletní popis najdete v manuálových stránkách.

Zábavné příkazy

Většina lidí, která má co do činění s operačním systémem Unix, asi nebude souhlasit s titulem této kapitoly. Někteří se dokonce rozčílí, protože si u každého příkazu musejí pamatovat až desítky parametrů a najednou se jim někdo snaží namluvit, že by používání takových příkazů mohlo být zábavné. V nadpisu této kapitoly se spíš odráží až sarkastický humor autorů operačního systému Unix. Dále uvedené příkazy totiž nemají ekvivalentní příkazy v operačním systému MS-DOS. Přitom jsou velmi výkonné, a když je zatvrzelí příznivci operačního systému MS-DOS chtějí používat, musejí si je koupit (speciální verze těchto příkazů pro MS-DOS byly dodatečně vytvořeny). Protože operační systém Unix odjakživa soupeří s operačními systémy od firmy Microsoft, jsou touto skutečností příznivci operačního systému Unix pobaveni.

V této kapitole se budeme zabývat příkazy `find` (příkaz pro vyhledávání skupin souborů v adresářové struktuře), `tar` (příkaz pro archivaci souborů nebo celých adresářů), `dd` (příkaz pro kopírování) a `sort` (příkaz pro třídění souborů). Předem je nutno upozornit na skutečnost, že tyto příkazy nejsou standardizovány. Zaměříme se na popis verzí náležejících do projektu GNU, protože právě tyto verze jsou implementovány v operačním systému Linux a právě tyto verze pravděpodobně budou (tak jako ostatní programy vytvořené v rámci projektu GNU) v blízké budoucnosti představovat standard. Používáte-li jiný operační systém typu Unix, pak nezapomeňte konfrontovat příslušné manuálové stránky.

Příkaz find

Všeobecné poznámky

Mezi doposud probranými příkazy jsou takové, které umožňují rekurzivní procházení stromovou strukturou adresářů. Příkladem jsou příkazy `ls -R` nebo `rm -R`. Příkaz `find` je také rekurzivní. Kdykoliv byste měli něco dělat s jistým typem souborů v adresáři a ve všech jeho podadresářích, vzpomeňte si na příkaz `find`. V jistém smyslu platí, že vyhledávání souborů příkazem `find` představuje spíše vedlejší efekt.

Základní struktura příkazu `find` je následující:

`find` *`cesta`* `[...]` *`výraz`* `[...]`

Uvedená syntaxe platí pouze pro verzi GNU. Ostatní verze neumožňují specifikovat více než jednu cestu (path), což z praktického hlediska není tak důležité. Hrubé vysvětlení funkce příkazu je následující: prostřednictvím prvního parametru zadáte, odkud má prohledávání začít (parametr Cesta; u verze GNU nemusíte tento parametr vůbec uvádět a prohledávání pak začne od aktuálního adresáře), a druhý parametr (Výraz) určuje typ vyhledávání.

Standardní chování příkazu `find` je poněkud lstivé a stojí za to tomuto faktu věnovat trochu pozornosti. Předpokládejme, že vaším domovským adresářem je adresář garbage a že obsahuje soubor foobar. Řekněme, že náhodou napíšete příkaz `find . -name foobar`. Tento příkaz by měl podle všech předpokladů vyhledat soubory nazvané `foobar` – avšak nic se nestane. Potíž je v tom, že program `find` je tzv. tichý (silent) příkaz. Pouze vrátí 0 (ať už nějaký soubor najde nebo ne), nebo vrátí záporné číslo, pokud nastal nějaký problém. Uvedený případ nenastane, pokud používáte operační systém Linux, ale je dobré jej mít na paměti.

Výrazy

Výraz může být rozdělen do čtyř různých skupin klíčových slov: *volby*, *testy*, *akce* a *operátory*. Každé klíčové slovo může vracet hodnotu true nebo false a přitom může produkovat nějaký vedlejší efekt. Rozdíl mezi skupinami klíčových slov popisuje následující seznam:

volby celkově ovlivňují operaci vyhledávání a netýkají se zpracování souboru. Příkladem volby je `-follow`, která „sdělí“ příkazu `find`, aby procházel symbolické odkazy. Volby vždy vracejí hodnotu true.

testy jsou skutečnými testy. Například test `-empty` ověřuje, zda je soubor prázdný. Testy mohou vracet hodnotu true nebo false.

akce produkují jako vedlejší efekt jméno programu. Také mohou vracet hodnotu true nebo false.

operátory ve skutečnosti nevracejí žádnou hodnotu (dle konvence vracejí hodnotu true) a používají se ke skládání výrazů. Příkladem je operátor `-or`, jenž jako výsledek produkuje logickou operaci OR nad operandy, kterými jsou dva výrazy. Jsou-li vedle sebe uvedeny dva výrazy bez operátoru, pak se implicitně aplikuje operátor `-and`.

Poznamenejme, že program `find` spoléhá na syntaktickou analýzu příkazu, kterou realizuje příkazový procesor. To znamená, že všechna klíčová slova musejí být oddělena nezobrazitelnými znaky a zejména platí, že spousta znaků musí být uvozena znaky Escape – jinak by je příkazový procesor nemohl správně interpretovat. Jako znaky Escape mohou být použita obrácená lomítka, jednoduché nebo dvojité uvozovky. V později uvedených příkladech se jednoznaková klíčová slova uvozují obráceným lomítkem, protože je to nejjednodušší.

Volby

V následujícím seznamu uvádíme přehled všech voleb, které je schopen příkaz `find` akceptovat (připomínáme, že se jedná o verzi GNU). Nezapomeňte, že volby vždy vracejí hodnotu true.

- `-daystart`

  Volba `-daystart` měří dobu, která uplyne od poslední půlnoci. Počítačový nadšenec asi nemůže pochopit, proč má nějaký program takovou volbu, ale programátor, který pracuje od osmi do pěti, ji ocení.

- `-depth`

  Tato volba zajistí, že příkaz `find` bude zpracovávat obsah každého podadresáře před zpracováním adresáře samotného. Abych řekl pravdu, neumím si představit užitečné uplatnění této volby, kromě případu, kdy se emuluje příkaz `rm -F` (podadresáře nemůžete zrušit, dokud obsahují nějaké soubory).

- `-noleaf`

  Volba `-noleaf` vypíná optimalizaci, která indikuje, že adresář obsahuje o dva podadresáře méně, než by měl obsahovat. Kdyby byl svět dokonalý, pak by bylo možné odkazovat se na všechny adresáře z prostředí každého jejich podadresáře (pomocí volby `..`), pomocí odkazu `.` uvnitř samotného adresáře a prostřednictvím jeho skutečného jména z jeho nadřazeného adresáře.

  To znamená, že na každý adresář musejí existovat alespoň dva odkazy (jeden z adresáře samotného a jeden z adresáře nadřazeného) a případně další odkazy ze všech podadresářů. V praxi se však může stát, že symbolické odkazy a distribuované souborové systémy mohou toto pravidlo porušit.[1]

- `-maxdepth` *`levels`*, `-mindepth` *`levels`*

  Parametry `levels` jsou nezáporná čísla, jež udávají maximální a minimální úroveň podadresářů, které má příkaz `find` prohledávat. Uveďme příklady: `-maxdepth 0` indikuje, že příkaz `find` má být realizován pouze pro argumenty uvedené v příkazovém řádku a že se žádné podadresáře prohledávat nemají. `-mindepth 1` zakazuje zpracování příkazu pro argumenty uvedené v příkazovém řádku, zatímco ostatní soubory v podadresářích budou zpracovány.

- `-version`

  Po zadání parametru `-version` se pouze vypíší informace o verzi programu find.

- `-xdev`

  Parametr `-xdev` má poněkud zavádějící označení. Pro program `find` předává informaci o tom, že se dané zařízení (tedy souborový systém) nemá prohledávat. Volba `-xdev` je velmi užitečná v případě, kdy se má vyhledávat něco v hlavním souborovém systému. Hlavní souborový systém většinou obsazuje malou diskovou oblast. Kdyby se použil příkaz `find /` bez parametru `-xdev`, pak by se prohledávala celá adresářová struktura!

Testy

První dva testy jsou velmi jednoduché: `-false` vždy vrací hodnotu false a `-true` vždy vrací hodnotu true. K dalším testům, které nevyžadují specifikaci hodnoty, patří test `-empty` (vrací hodnotu true, pokud je daný soubor prázdný) a dále dvojice `-nouser / -nogroup`, kdy test vrací hodnotu true, právě když se v souboru `/etc/passwd` nebo `/etc/group` nevyskytuje záznam identifikující vlastníka souboru jako uživatele nebo skupinu. Jedná se o postižení situace, kdy je v systému zrušen účet uživatele, ale soubory jím vlastněné jsou stále uloženy někde v souborovém systému. Podle Murphyho zákonů jsou takové soubory neobyčejně velké.

Samozřejmě je možné vyhledávat soubory vlastněné jistým uživatelem nebo jistou skupinou. Odpovídající testy jsou `-uid nn` a `-gid nn`. Naneštěstí nelze přímo zadat uživatelské jméno, ale musí se vždy uvést jeho identifikační číslo `nn`.

Je povoleno použít zápis identifikačního čísla ve tvaru `+nn`, což znamená „ostře větší než“ nebo `-nn`, což znamená „ostře menší než“. Ve spojení s identifikačními čísly uživatelů se tato možnost jeví jako hloupá, ale v souvislosti s ostatními testy může být užitečná.

Další užitečnou volbou je volba `-type c`, která vrací hodnotu true, pokud je soubor typu `c`. Mnemotechnické zkratky jsou stejné, jako v případě příkazu `ls`: **b** pro binární soubor, **c** pro znakový soubor, **d** pro adresáře, **p** pro pojmenované roury, **l** pro symbolické odkazy a **s** pro sokety (sockets). Obyčejné soubory jsou specifikovány prostřednictvím zkratky **f**. K testu `-type` se vztahuje

1 Distribuované souborové systémy umožňují, aby se soubory lokalizované někde jinde jevily jako lokální.

test `-xtype`, který je podobný, ale chová se jinak v případě symbolických odkazů – pokud není zadána volba `-follow`, ověřuje se místo vlastního symbolického odkazu soubor, na který odkaz odkazuje.

Testy `-inum nn` a `-links nn` ověřují, zda je číslo i-uzlu příslušné k danému souboru nn nebo zda má soubor nn symbolických odkazů. Test `-size nn` má hodnotu true, jestliže je pro soubor alokováno nn bloků po 512 bajtech. Dnes již však neplatí, že se velikost souboru vždy měří (například po zadání příkazu `ls -s`) v blocích po 512 bajtech – Linux například používá bloky po 1024 bajtech. Proto je možné číslo `nn` doplnit znakem `b` (pak se měří velikost v bajtech) nebo k (pak se měří velikost v kilobajtech).

Bity specifikující přístupová práva se testují pomocí testu `-perm` *`mode`*. Pokud argumentu *`mode`* nepředchází žádné znaménko, pak bity specifikující přístupová práva musejí přesně souhlasit. Pokud předchází znaménko `-`, pak musejí být nastavena příslušná přístupová práva, ale o ostatních se nic nepředpokládá. Pokud předchází znak +, pak je podmínka splněna, právě když je kterýkoliv z bitů nastaven. Důležité je, že hodnota mode musí být uvedena buď symbolicky, nebo jako oktalové číslo, tak jako v případě příkazu `chmod`.

Následující skupina testů se vztahuje k času, kdy byl soubor naposledy použit. Takové testy jsou zejména užitečné tehdy, když dojde k zaplnění diskového prostoru a uživatel potřebuje vyhledat staré soubory, které lze zrušit. Staré a zapomenuté soubory se těžko hledají a příkaz `find` vám dává naději, že se vám je podaří nalézt. Test `-atime nn` vrací hodnotu true, pokud byl daný soubor zpřístupněn před `nn` dny, `-ctime nn` vrací hodnotu true, když byly atributy přístupových práv daného souboru změněny před `nn` dny (například příkazem `chmod`). Test `-mtime nn` vrací hodnotu true, když byl soubor naposledy modifikován před `nn` dny. Užitečným bude zřejmě test `-newer file`, který vrací hodnotu true, když byl uvažovaný soubor modifikován později než soubor uvedený jako parametr `file`. GNU-verze příkazu `find` také akceptuje testy `-anewer` a `-cnewer`, které se chovají podobně jako test `-newer`, a dále testy `-amin`, `-cmin` a `-mmin`, které pracují s časem uvedeným v minutách.

V neposlední řadě se musíme zmínit o testu `-name pattern`. Tento test vrací hodnotu true, pokud jméno souboru vyhovuje šabloně uvedené prostřednictvím parametru `pattern`. Pro šablonu platí prakticky stejná pravidla jako pro používání pseudoznaků ve jménech souborů, například při používání příkazu `ls`. Jistě si pamatujete, že příkazový procesor je schopen zvláštním způsobem interpretovat tzv. pseudoznaky (hvězdičku a otazník). Avšak test `-name foo*` nevrátí tu hodnotu, kterou byste očekávali. Místo toho budete muset použít test `-name foo` nebo `-name "foo*"`. Dobře si tuto situaci zapamatujte, většina uživatelů používá nesprávný test. Je zde ještě jeden rozdíl oproti příkazu `ls` – tečky na začátku nejsou interpretovány. Pokud se potřebujete vyrovnat s tímto problémem, použijte test `-path pattern`, který se o tečky a zpětná lomítka při porovnávání cest nestará.

Akce

Jak jsme si již řekli, argumenty charakterizující akce slouží k inicializaci nějaké operace. Přesto mezi tyto parametry patří například akce `-prune`, která nic neaktivuje, pouze realizuje krok dolů ve stromové struktuře adresářů. Většinou je tato akce spjata s parametrem `-fstype`, prostřednictvím kterého se realizuje výběr mezi různými souborovými systémy. Ostatní akce mohou být rozděleny do dvou širokých kategorií.

- Akce, které něco tisknou. Je zřejmé, že implicitní akcí bude akce `-print`. Ta v průběhu činnosti programu find tiskne jména souborů, které se právě zpracovávají (ovšem za předpokladu, že ostatní podmínky uvedené v příkazovém řádku vracejí hodnotu true). Akce `-print` má několik variant. První z nich, `-fprint file`, používá soubor uvedený jako pa-

rametr `file` místo standardního výstupu. Další, `-ls`, zobrazuje jméno aktuálního souboru jako příkaz `ls -dils`. Akce `-printf format` se chová podobně jako funkce `printf()` v jazyku C (v této variantě lze specifikovat formát výstupu). Totéž realizuje akce `-fprintf file`, ale do souboru uvedeného prostřednictvím parametru `file`. Tato akce rovněž vždy vrací hodnotu true.

- Akce, které aktivují nějaký příkaz. Jejich syntaxe je poněkud zvláštní, a proto se na ně podívejme podrobněji.

  ```
  -exec command \;
  ```

  Akce `-exec` spustí příkaz zadaný prostřednictvím parametru `command` a vrací hodnotu true, pokud má příkaz status ukončení 0. Za příkazem je uvedena dvojice znaků „`\;`“, protože jinak by příkaz `find` nebyl schopen rozeznat, kde příkaz `command` končí (trik s uvedením akce `-exec` na konec příkazu `find` není aplikovatelný). Proto se příkaz `command` ukončuje znakem „`;`“, což je zároveň znak k oddělování příkazů v jazyku příkazového procesoru. Bez znaku „`\`“ by středník byl příkazovým procesorem na základě syntaktické analýzy odstraněn a do příkazu `find` by se již nedostal. Proto se zde znak „`\`“ musí uvést. Další věc, kterou si musíte zapamatovat, spočívá ve způsobu specifikace jména aktuálního souboru uvnitř příkazu `command`. Jméno souboru musí být uvedeno pomocí dvojice složených závorek `{}`. Některé starší verze programu `find` vyžadují, aby bylo jméno odděleno netisknutelnými znaky (white spaces), například mezerami. To však není příliš šikovné, proto je ve verzi GNU zavedena konstrukce se složenými závorkami umožňující generovat řetězce. Asi vás napadá otázka, zda musejí být složené závorky uzavřeny do uvozovek. Podle mých zkušeností nemusejí – ani v případě, že použijete příkazový procesor `bash`, ani v případě, že použijete příkazový procesor `tcsh`. Proto zůstanou v příkazu `find` řetězce uvedené ve složených závorkách příkazovým procesorem nedotčeny.

  ```
  -ok command \;
  ```

  Akce `-ok` se chová podobně jako akce `-exec` s tím rozdílem, že pro každý vybraný soubor je uživatel vyzván k potvrzení příkazu (tj. zda se má provést nebo ne). Pokud odpověď začíná písmenem „y“ nebo „Y“, příkaz se provede a akce vrátí hodnotu true. Jinak se akce neprovede a vrácená hodnota je false.

Operátory

V příkazu `find` lze použít spoustu operátorů. Následující seznam je uvádí v pořadí s klesající prioritou.

```
\( expr \)
```

Kulaté závorky mění prioritu operandu. Závorkám musí předcházet znak „`\`“, protože v příkazovém procesoru mají speciální význam.

```
! expr
-not expr
```

Operátory `!` a `-not` mění pravdivostní hodnotu výrazu `expr`. Je-li hodnota `expr` true, změní se na false a naopak. Znak „`!`“ nemusí být oddělen obráceným lomítkem nebo uvozovkami, protože je následován netisknutelným znakem (mezerou).

```
expr1 expr2
expr1 -a expr2
expr1 -and expr2
```

Všechny tři varianty odpovídají logické operaci AND, přičemž se nejčastěji používá první varianta. Jestliže je první výraz `expr1` vyhodnocen jako false, pak se již druhý výraz nevyhodnocuje.

```
expr1 -o expr2
expr1 -or expr2
```

Obě varianty odpovídají logické operaci OR. Je-li první výraz `expr1` vyhodnocen jako true, pak se již druhý výraz `expr2` nevyhodnocuje.

```
expr1 , expr2
```

Uvedený operátor má poněkud speciální význam. Oba výrazy `expr1` i `expr2` se vyhodnotí (samozřejmě se všemi vedlejšími účinky) a hodnota konečného výrazu je nastavena na hodnotu výrazu `expr2`.

Příklady

Jak vyplývá z předcházejících seznamů, příkaz `find` má spoustu parametrů. Existuje však několik často používaných konstrukcí s příkazem `find`, které stojí za to si zapamatovat. Některé z nich si uvedeme jako příklady.

```
% find .-name foo\* -print
```

První příklad vyhledá všechny soubory, jejichž jména začínají řetězcem `foo`. Pokud se mají hledat soubory, jejichž jména mají řetězec `foo` někde uvnitř, pak je vhodné místo `foo` použít „`*foo*`“.

```
% find /usr/include/ -xtype f -exec grep foobar \
        /dev/null {}\;
```

V druhém příkladu se rekurzivně provádí příkaz `grep`, a to od adresáře `/usr/include`. V tomto případě se zajímáme o obyčejné soubory a symbolické odkazy, které na obyčejné soubory odkazují. Proto jsme použili test `-xtype`. V mnoha případech je jednodušší tento test nepoužít, zejména tehdy, kdy jsme si jisti, že žádný binární soubor neobsahuje požadovaný řetězec. A proč jsme použili příkaz `/dev/null`? Jedná se o trik, který příkaz `grep` „přinutí“ vypsat jméno souboru, pro který nebylo splněno výběrové kritérium. Příkaz `grep` je aplikován na každý soubor jinak, a proto „nepovažuje“ za nezbytné vypisovat jméno souboru. Ale nyní jsou zde dva soubory – aktuální soubor a soubor `/dev/null`. Jiná možnost spočívá v použití roury a příkazu `xargs`. Když jsem ji zkusil, zničil jsem si celý souborový systém.

```
%find / -atime +1 -fstype ext2 -name core \
        -exec rm {}\;
```

Tento příklad představuje klasickou úlohu pro crontab. Zruší ze souborového systému `ext2` všechny soubory s názvem `core`, které nebyly posledních 24 hodin zpřístupněny. Je možné, že někdo tyto soubory používá v souvislosti s programem `gdb` k ladění, ale je málo pravděpodobné, že je bude používat po 24 hodinách.

```
% find /home -xdev -size +500k -ls > piggies
```

Další příklad umožňuje zjistit, kdo vlastní soubory větší než 500 kilobajtů a kdo tedy zatěžuje souborový systém. Poznamenejme, že pokud se zajímáme pouze o jeden souborový systém, pak argument `-xdev` není nutný.

Slovo na závěr

Mějte na paměti, že příkaz `find` spotřebuje velmi mnoho času, pokud má provádět operace s každým souborem v celém souborovém systému. Proto je nutné počet operací vhodně optimalizovat, zejména když se na vašem systému pravidelně realizují úlohy pro údržbu prostřednictvím crontab. Jako poučný příklad vezměme následující: Předpokládejme, že chceme zrušit soubory končící na `.BAK`, přitom chceme změnit přístupová práva všech adresářů na 771 a přístupová práva souborů končících na `.sh` změnit na 755. Přitom chceme ještě připojit souborový systém NFS na telefonní linku, a v tomto systému nechceme žádné soubory kontrolovat. Proč bychom měli psát tři různé příkazy? Nejefektivněji uvedenou úlohu splníme takto:

```
% find .\( -fstype nfs -prune \) -o \
       \( -type d -a -exec chmod 771 {}\; \) -o \
       \( -name "*.BAK" -a -exec /bin/rm {}\; \) -o \
       \( -name "*.sh" -a -exec chmod 755 {}\; \)
```

Asi se vám bude takový příkaz zdát nesrozumitelný, ale když si jej pozorně prohlédnete, zjistíte, že je logický. Pamatujte si, že příkazy tohoto typu neprovádějí nic jiného, než že vyhodnocují výrazy, jejichž hodnota je buď false, nebo true. Samozřejmě mají vedlejší účinky – ty se realizují tehdy, když příkaz `find` musí vyhodnotit část obsahující výraz s akcí `-exec`, což nastane právě tehdy, když je levá strana výrazu vyhodnocena jako pravdivá (true). Když je například právě zpracovávaným souborem adresář, pak se bude realizovat první akce `-exec` a přístupová práva adresáře budou změněna na 771. Ostatní části příkazu budou vynechány. Budete-li takové příkazy aplikovat prakticky, přestanou se vám zdát nesrozumitelné a budete je používat zcela přirozeně.

Archivační program tar

Úvod

Tar je obecně použitelný program pro archivaci souborů, který je schopen sloučit (spakovat) velké množství souborů do jediného archivního souboru, přičemž zachovává veškeré informace o souborech, jako jsou uživatelská práva. Jméno `tar` je zkratkou „tape archive“, protože tento nástroj byl původně používán pro archivaci na magnetické pásky. Jak však uvidíme, používání příkazu `tar` není zdaleka omezeno pouze na záložní soubory pro magnetické pásky.

Hlavní funkce

Formát příkazu `tar` je následující:

`tar` *`functionoptions files...`*

kde *`function`* je jednoznaková indikace operace, která se má provést; *`options`* je seznam (jednoznakových) voleb pro tuto operaci a *`files`* je seznam souborů, jež se mají spakovat nebo rozpakovat. (Všimněte si, že argument function není oddělen od options mezerou.) Parametr `function` může být:

- `c` pro vytvoření nového archivního souboru
- `x` pro extrakci souborů z archivního souboru
- `t` pro výpis obsahu archivního souboru
- `r` pro přidání souborů na konec archivního souboru
- `u` pro aktualizaci souborů, které jsou novější než soubory v archivním souboru
- `d` pro porovnání souborů v archivním souboru se soubory v souborovém systému

Nejčastěji budete používat parametr `c`, `x` a `t`; ostatní budete používat jen zřídka.

Volby

Nejčastěji používané volby `options` jsou tyto:

- `v` pro tisk podrobné informace v průběhu pakování a rozpakování
- `k` pro zachování existujících souborů při rozpakování, tj. žádný existující soubor nebude přepsán souborem z archivního souboru
- `f` *`filename`* pro specifikaci jména archivního souboru

Další volby popíšeme v následujícím oddílu později.

Příklady

Ačkoliv se syntaxe příkazu `tar` zdá být na první pohled složitá, je jeho praktické použití velmi jednoduché. Předpokládejme, že máme adresář `mt` obsahující následující soubory:

```
rutabaga% ls -l mt
-rw-r--r-- 1 root root 24 Sep 21 1993 Makefile
-rw-r--r-- 1 root root 847 Sep 21 1993 README
-rw-r--r-- 1 root root 9220 Nov 16 19:03 mt
-rwxr-xr-x 1 root root 2775 Aug 7 1993 mt.1
-rw-r--r-- 1 root root 6421 Aug 7 1993 mt.c
-rw-r--r-- 1 root root 3948 Nov 16 19:02 mt.o
-rw-r--r-- 1 root root 11204 Sep 5 1993 st_info.txt
```

Nyní chceme spakovat pomocí příkazu `tar` obsah tohoto adresáře do jediného archivního souboru. Použijeme tedy příkaz:

```
tar cf mt.tar mt
```

Prvním argumentem v příkazu `tar` je operace (zde `c` pro vytvoření) následovaná volbou `options`, kde jsme použili `f mt.tar` a specifikovali tak jméno archivního souboru `mt.tar`. Jako poslední je uveden seznam souborů – pokud se místo seznamu uvede název adresáře, `tar` spakuje všechny soubory v tomto adresáři.

Poznamenejme, že prvním argumentem musí být písmeno, označující operaci, následované seznamem voleb „options“. Proto není důvod dávat před první argument pomlčku, jak to vyžaduje většina systémů Unix. Příkaz `tar` v systému Linux však tuto pomlčku připouští, proto by mohl mít posledně uvedený příklad tvar:

```
tar -cf mt.tar mt
```

V některých verzích musí být typ operace (jako je `c`, `t`, nebo `x`) na prvním místě, v jiných verzích na pořadí písmen nezáleží. Jistý přehled o průběhu pakování (nebo rozpakování) získáte prostřednictvím volby `v` – pak se bude každý soubor ukládaný do archivního souboru vypisovat na obrazovce. Například:

```
rutabaga% tar cvf mt.tar mt
mt/
mt/st_info.txt
mt/README
mt/mt.1
mt/Makefile
mt/mt.c
mt/mt.o
mt/mt
```

Použijete-li volbu `v` vícekrát, zobrazovaná informace bude podrobnější:

```
rutabaga% tar cvvf mt.tar mt
drwxr-xr-x root/root 0 Nov 16 19:03 1994 mt/
-rw-r--r-- root/root 11204 Sep 5 13:10 1993 mt/st_info.txt
-rw-r--r-- root/root 847 Sep 21 16:37 1993 mt/README
-rwxr-xr-x root/root 2775 Aug 7 05:50 1993 mt/mt.1
-rw-r--r-- root/root 24 Sep 21 16:03 1993 mt/Makefile
-rw-r--r-- root/root 6421 Aug 7 09:50 1993 mt/mt.c
-rw-r--r-- root/root 3948 Nov 16 19:02 1994 mt/mt.o
-rw-r--r-- root/root 9220 Nov 16 19:03 1994 mt/mt
```

Tyto podrobné informace jsou důležité zejména tehdy, když chcete zkontrolovat, zda `tar` realizuje pakování dle vašich představ. U některých verzí programu `tar` musí být volba `f` uvedena jako poslední. Je to z toho důvodu, že se za volbou `f` předpokládá jméno souboru. Pokud neuvedete volbu `f`, předpokládá `tar` (z historických důvodů), že má použít zařízení `/dev/rmt0`, což je první magnetopásková jednotka.

Nyní můžeme soubor `mt.tar` předat někomu jinému a ten si jej může rozpakovat na svém počítači. K rozpakování by měl použít následující příkaz:

```
tar xvf mt.tar
```

Tento příkaz vytvoří adresář `mt` a umístí do něj všechny soubory, které jsme dříve uvedli – s týmiž přístupovými právy jako v původním systému. Nové soubory budou vlastněny uživatelem, který provedl příkaz `tar xvf` – pokud ovšem nepoužijete tento příkaz jako root (pak je původní vlastník zachován). Parametr `x` zajistí rozpakování souborů a volba `v` opět zobrazí soubory, které se ukládají na disk:

```
courgette% tar xvf mmt.tar
mt/
mt/st_info.txt
mt/README
mt/mt.1
mt/Makefile
mt/mt.c
mt/mt.o
mt/mt
```

Všimněte si, že tar zachoval cestu každého souboru relativní k poloze v původní struktuře adresářů. To znamená, že když jsme vytvářeli archivní soubor pomocí příkazu `tar cf mt.tar mt`, jediný soubor, který jsme specifikovali místo seznamu souborů, byl adresář `mt` obsahující soubory. Proto tar uložil do archivního souboru samotný adresář a soubory, které se v něm nacházely. Při rozpakování se napřed vytvořil adresář `mt` a pak do něj byly uloženy soubory – tedy přesně opačný proces, než jaký probíhal při vytváření archivního souboru. tar implicitně rozpakuje všechny soubory relativně k pracovnímu adresáři, v němž jej spustíte. Pokud se například pokusíte spakovat obsah adresáře `/bin` příkazem:

```
tar cvf bin.tar /bin
```

dostanete varovné hlášení:

```
tar: Removing leading / from absolute path names in the archive.
```

To znamená, že soubory jsou ukládány v archivním souboru uvnitř adresáře `/bin`. Když tento soubor rozpakujete, adresář `/bin` bude vytvořen jako podadresář vašeho pracovního adresáře, v němž spouštíte `tar` – ne jako absolutní adresář `/bin`. Toto je velmi důležitý mechanismus, který byl vymyšlen za účelem ochrany před fatálními chybami při rozpakování pomocí příkazu `tar`. Jinak byste, podle předchozího příkladu, mohli přepsat soubory v adresáři `/bin`, a tak zničit systém.

Pokud byste však opravdu chtěli rozpakovat takový archivní soubor do adresáře `/bin`, museli byste mít jako pracovní adresář nastaven `/`. Výše uvedený mechanismus můžete potlačit pomocí volby `p`, ale nedoporučuje se to. Jiný způsob, jak vytvořit soubor `mt.tar`, spočívá v tom, že se přepnete do adresáře `mt` pomocí příkazu `cd` a zadáte příkaz:

```
tar cvf mt.tar *
```

Potom ovšem nebude podadresář `mt` ukládán do archivního souboru a při rozpakování budou soubory ukládány přímo do vašeho pracovního adresáře. Proto doporučujeme vždy pakovat soubory tak, aby archivní soubor obsahoval podadresář, jak jsme ukázali pomocí příkazu `tar cvf mt.tar mt`. Pak se vždy před rozpakováním vytvoří adresář pro uložení souborů a nemůže se stát, že přepíšete soubory ve svém pracovním adresáři. Navíc zbavíte osobu, která provádí rozpakování, starostí s vytvářením adresáře pro ukládání souborů a ušetříte jí dost času. Samozřejmě existuje mnoho situací, při nichž nebude vhodné doporučený postup dodržovat, ale všeobecně se takový postup považuje za jakousi etiketu při práci s programem `tar`.

Při vytváření archivních souborů můžete uvést seznam souborů nebo adresářů, které se mají do archivního souboru uložit. V prvním příkladu jsme použili příkaz `tar` pro jediný adresář a ukázali jsme použití hvězdičky, kterou příkazový procesor rozšíří na seznam všech souborů v pracovním adresáři. Před rozpakováním archivního souboru pomocí příkazu `tar` je vhodné se podívat, co je jeho obsahem. Tak se například dozvíte, zda budete muset vytvořit adresář pro uložení souborů vlastními silami, nebo bude vytvořen automaticky. K prohlížení obsahu archivního souboru použijte příkaz:

```
tar tvf tarfile
```

Ten zobrazí obsah archivního souboru `tarfile`. Poznamenejme, že pokud použijete funkci t, stačí uvést jednu volbu `v` a obdržíte podrobnou informaci o obsahu archivního souboru:

```
courgette% tar tvf mt.tar mt
drwxr-xr-x root/root 0 Nov 16 19:03 1994 mt/
```

```
-rw-r--r-- root/root 11204 Sep 5 13:10 1993 mt/st_info.txt
-rw-r--r-- root/root 847 Sep 21 16:37 1993 mt/README
-rwxr-xr-x root/root 2775 Aug 7 05:50 1993 mt/mt.1
-rw-r--r-- root/root 24 Sep 21 16:03 1993 mt/Makefile
-rw-r--r-- root/root 6421 Aug 7 09:50 1993 mt/mt.c
-rw-r--r-- root/root 3948 Nov 16 19:02 1994 mt/mt.o
-rw-r--r-- root/root 9220 Nov 16 19:03 1994 mt/mt
```

Žádné rozpakování v tomto případě neproběhne, ale pouze se zobrazí informace o obsahu archivního souboru. Zde vidíme, že jména souborů jsou doplněna jménem adresáře `mt`, a proto při rozpakování bude tento adresář nejdříve vytvořen a pak teprve do něj budou ukládány soubory. Příkaz `tar` můžete rovněž použít k extrakci jednotlivých souborů z archivního souboru. Potom použijte příkaz ve tvaru:

```
tar xvf tarfile files
```

kde `files` je seznam souborů, které se mají rozpakovat. Jak jsme si již ukázali, pokud neuvedeme žádný seznam, `tar` rozpakuje všechny soubory z archivního souboru.

Jestliže specifikujete soubory, které se mají rozpakovat, musíte uvést jejich plná jména včetně cesty tak, jak jsou uvedena v archivním souboru. Chceme-li například rozpakovat soubor `mt.c` z výše uvedeného archivního souboru `mt.tar`, použijeme následující příkaz:

```
tar xvf mt.tar mt/mt.c
```

který nejdříve vytvoří podadresář `mt` a do něj pak umístí soubor `mt.c`. Program `tar` má mnohem více možností, než které jsme zde uvedli. Ty, o nichž jsme se zde zmínili, budete zřejmě používat nejčastěji. Verze GNU implementovaná operačnímu systému Linux má řadu přípon a představuje ideální nástroj pro pořizování záložních kopií. Více informací naleznete v manuálové stránce k příkazu `tar`.

Jak používat program tar spolu s programem gzip

Program tar neprovádí při ukládání dat do archivního souboru žádnou komprimaci. Jestliže vytvoříte soubor `tar` ze tří souborů o velikosti 200 KB, pak výsledný soubor bude mít velikost 600 KB. Proto patří k běžné praxi komprimovat soubory `tar` pomocí programu `gzip` (nebo starším programem `compress`). Komprimovaný soubor tar můžete vytvořit pomocí následujících příkazů:

```
tar cvf tarfile files...
gzip -9 tarfile
```

Provedení takových příkazů je však těžkopádné a vyžaduje, abyste měli na disku dostatek místa pro nekomprimovaný soubor `tar`, než spustíte `gzip`.

Mnohem efektivnější způsob spočívá v tom, že se využije možnost programu tar zapisovat archivní soubor do standardního výstupu. Pokud použijete místo jména archivního souboru pomlčku (`-`), pak bude `tar` číst data ze standardního vstupu nebo zapisovat do standardního výstupu. K vytvoření komprimovaného souboru tar tedy můžeme použít příkaz:

```
tar cvf - files... | gzip -9 > tarfile.tar.gz
```

Zde vytváří `tar` archivní soubor ze souborů uvedených v seznamu `files`, zapisuje jej do standardního výstupu; `gzip` čte data ze standardního vstupu, komprimuje je a zapisuje do svého standardního výstupu. Nakonec jsme přesměrovali komprimovaný soubor `tar` do `tarfile.tar.gz`. Podobně bychom mohli k rozpakování takového souboru použít příkaz:

```
gunzip -9c tarfile.tar.gz | tar xvf -
```

Zde `gunzip` provede dekomprimaci uvedeného archivního souboru a zapisuje výsledek do standardního výstupu sloužícího jako standardní vstup pro program `tar`, jenž soubory rozpakuje a ukládá. Není práce v systému Unix zábavná? Samozřejmě, že jsou oba výše uvedené příkazy poněkud těžkopádné. Naštěstí GNU-verze programu tar má možnost uvést volbu `z`, která zajistí automatické vytváření nebo rozpakování komprimovaných archivních souborů `tar`. (Diskusi o použití volby `z` jsme úmyslně zařadili až na toto místo, abyste si uvědomili její výhody.) K vytvoření a rozpakování archivních souborů máte tedy možnost používat následující příkazy:

```
tar cvzf tarfile.tar.gz files...
```

a

```
tar xvzf tarfile.tar.gz
```

Poznamenejme, že byste měli soubory vytvořené tímto způsobem označovat pomocí přípony `.tar.gz`, aby byl zřejmý jejich formát. Volba funguje i ve spojení s dalšími parametry, jako je například `t`. Možnost používat volbu `z` podporuje pouze GNU-verze programu `tar`. Jestliže používáte `tar` na jiných systémech Unix, musíte pro realizaci stejného úkolu zadávat delší příkazy, jak jsme uvedli dříve. Téměř všechny verze operačního systému Linux používají verzi GNU.

Nyní využijeme svých znalostí a napíšeme krátké skripty pro příkazový procesor, které vám mohou sloužit jako návod pro vytváření a rozpakování souborů `tar`. V příkazovém procesoru bash doplňte následující řádky do souboru `.bashrc`:

```
tarc () {tar czvf $1.tar.gz $1 }
tarx () {tar xzvf $1 }
tart () {tar tzvf $1 }
```

Budete-li nyní chtít vytvořit komprimovaný archivní soubor obsahující soubory z jednoho adresáře, stačí zadat příkaz:

```
tarc directory
```

Výsledný archivní soubor bude mít název `directory.tar.gz`. (Přesvědčte se, že jste neuvedli před jménem adresáře lomítka (`/`) – jinak bude archivní soubor vytvořen jako `.tar.gz` uvnitř daného adresáře.) K zobrazení obsahu archivního souboru stačí zadat příkaz

```
tart file.tar.gz
```

a k jeho rozpakování příkaz

```
tarx file.tar.gz
```

Triky při používání programu tar

Protože tar ukládá do archivního souboru informace o vlastnictví souborů, přístupových právech, adresářové struktuře i symbolických odkazech, velmi dobře se hodí ke kopírování nebo přesouvání celých adresářových stromů z jednoho místa na druhé v rámci jednoho systému (a dokonce i mezi systémy, jak uvidíte). Použijete-li syntaxi s pomlčkou (`-`), budete zapisovat výsledný soubor do standardního výstupu, který může být čten jako standardní vstup a rozpakován kdekoliv.

Předpokládejme například, že máme adresář obsahující dva podadresáře: `from-stuff` a `to-stuff`. Adresář `from-stuff` obsahuje celý strom dalších podadresářů s mnoha soubory, symbolickými odkazy atd. a bylo by velmi obtížné zkopírovat jej pomocí příkazu `cp`. Ke zkopírování celého adresáře `from-stuff` do adresáře `to-stuff` však můžeme použít následující příkazy:

```
cd from-stuff
tar cf - .| (cd ../to-stuff; tar xvf -)
```

Velice jednoduché a elegantní! Začínáme v adresáři `from-stuff`, kde vytvoříme soubor tar obsahující celý adresář, a zapíšeme jej do standardního výstupu. Tento výstup pak čte podřízený příkazový procesor (příkazy uvedené v závorkách), který se nejdříve přepne do adresáře `../to-stuff` a pak spustí příkaz `tar xvf`, jenž čte ze standardního vstupu. Přitom se žádný archivní soubor `tar` neukládá na disk – data jsou přímo posílána z jednoho procesu tar do druhého. Druhý proces tar používá volbu `v` pro zobrazení informací o každém souboru, který se ukládá, a vy můžete kontrolovat, zda kopírování obsahu adresářů probíhá správně.

Pomocí tohoto „triku“ můžete ve skutečnosti přenášet celé adresáře mezi jednotlivými systémy – stačí vložit vhodný příkaz `rsh` mezi příkazy uzavřené v závorkách. Pak vzdálený příkazový procesor spustí `tar`, který bude číst archivní soubor jako standardní vstup. (Poznamenejme, že verze GNU příkazu `tar` má možnost automaticky číst nebo zapisovat soubory `tar` z/do jiných počítačů v síti; informace najdete v příslušné manuálové stránce.)

Program dd

Existuje jakási legenda, podle které v ranných dobách vývoje operačního systému Unix potřebovali programátoři program pro binární kopírování dat mezi jednotlivými zařízeními. Protože velmi spěchali, „vypůjčili“ si syntaxi příkazu používanou v operačním systému na počítačích IBAlt+360 a později vyvinuli rozhraní konzistentní s ostatními příkazy operačního systému Unix. Nevím, zda je tato legenda pravdivá, ale pěkně se vypráví.

Volby

Přes svůj legendou opředený původ není program `dd` úplně jiný než ostatní příkazy operačního systému Unix. Ve skutečnosti představuje filtr, který implicitně čte data ze standardního vstupu a zapisuje je na standardní výstup. Pokud zadáte na terminálu pouze příkaz `dd`, pak bude program `dd` tiše očekávat vstup z klávesnice.

Syntaxe příkazu `dd` je následující:

```
dd [if=file] [of=file] [ibs=bytes] [obs=bytes]
   [bs=bytes] [cbs=bytes] [skip=blocks] [seek=blocks]
   [count=blocks] [conv={ascii, ebcdic, ibm, block,
   unblock, lcase, ucase, swab, noerror, notrunc, sync}]
```

Všechny volby mají tvar *option=value*. Před a za znakem „=“ se nesmí objevit mezera, což je nepříjemné, protože v takové situaci neprovede příkazový procesor doplnění jména souboru, pokud se uvedou pseudoznaky. Verze příkazového procesoru bash v operačním systému Linux je však dostatečně inteligentní, proto nemusíte mít obavy. Všechny výše uvedené číselné hodnoty (`bytes` a `blocks`) mohou být následovány multiplikativní konstantou. Pro tyto konstanty lze použít následující zkratky: **b** pro bloky (multiplikativní konstanta 512), **k** pro kilobajty (multiplikativní konstanta 1024), **w** pro slova (multiplikativní konstanta 2) a prostřednictvím **xm** se číselná hodnota násobí konstantou **m**.

Význam voleb příkazu `dd` je popsán v následujícím seznamu.

- `if=`*`filein`* a `of=`*`fileout`* jsou volby specifikující jméno vstupního a výstupního souboru. Výstupní soubor je zkrácen na velikost danou volbou `seek`. Pokud není klíčové slovo `seek` uvedeno, pak je hodnota `seek` rovna nule (soubor je před provedením operace zrušen). Volba `notrunc` (viz dále) však může toto implicitní chování příkazu `dd` změnit.
- `ibs=`*`nn`* a `obs=`*`nn`* jsou volby specifikující počet bajtů, které se mají najednou přečíst a najednou zapsat. Domnívám se, že implicitní hodnoty jsou jeden blok (t.j. 512 bajtů), ale nejsem si tím tak docela jist. Uvedené parametry jsou velmi důležité v případě, kdy se jako vstup nebo výstup používají speciální zařízení – například při čtení ze sítě je hodnota `ibs` nastavena na 10 kilobajtů, zatímco disketa 3,5 palce má přirozenou délku bloku 18 kilobajtů. Nevhodné nastavení těchto hodnot může nejen značně prodloužit realizaci programu, ale také může vést k neodstranitelným chybám. Proto buďte opatrní.
- `bs=`*`nn`* je volba, která předefinuje nastavení `ibs` a `obs` – obě hodnoty se nastaví na stejnou konstantu *nn*.
- Volba `cbs=`*`nn`* nastavuje vyrovnávací paměti pro konverzi na *nn* bajtů. Vyrovnávací paměť se používá při konverzi z formátu ASCII na EBCDIC nebo při konverzi mezi textovými a binárními soubory a podobně. Například soubory vytvořené pod operačním systémem VMS mají zpravidla velikost bloku 512 bajtů, proto byste měli například při čtení dat zapsaných na magnetickou pásku v operačním systému VMS nastavit hodnotu cbs na 1 bajt.
- `skip=`*`nbl`* a `seek=`*`nbl`* jsou volby, které „sdělí“ programu `dd`, kolik bloků má „přeskočit“ při čtení vstupu a při zápisu na výstup. Samozřejmě platí, že druhá volba má smysl jedině tehdy, když je specifikována konverze notrunc. Velikost bloků je dána volbami `ibs` a `obs`. Uvědomte si, že pokud nezadáte hodnotu ibs a zadáte hodnotu `skip=1b`, pak ve skutečnosti dojde k „přeskočení“ 512x512 bajtů, což je 256 kilobajtů.
- Volbou `count=`*`nbl`* se nastavuje, kolik bloků se má ze vstupu kopírovat. Velikost bloku je přitom dána hodnotou `ibs`. Uvedená volba spolu s předcházející volbou je užitečná v případě, kdy chcete obnovit co nejvíce bajtů z porušeného souboru – „přeskočíte“ nečitelnou část souboru a zbytek zkopírujete do nového souboru.
- Volba `conv=`*`conversion`*`,[`*`conversion`*`,...]` je určena ke konverzi podle specifikace. Možné konverze jsou následující: `ascii` – konverze formátu EBCDIC na formát ASCII; `ebcdic` nebo `ibm` – konverze z formátu ASCII na EBCDIC (jednoznačná konverze z formátu EBCDIC na formát ASCII neexistuje; první představuje standardní konverzi a druhá funguje lépe, pokud se má soubor tisknout na tiskárně IBM); `block` – vyplňuje mezerami záznamy ukončené znakem newline na délku uvedenou prostřednictvím volby `cbs`; `unblock` – opačná konverze ke konverzi `block`; `lcase` – konverze z velkých písmen na malá; `ucase` – konverze z malých písmen na velká; `swab` – konverze, při které se každý pár vstupních bajtů zamění (chcete-li například použít na počítači s procesorem Intel soubor obsahující celá čísla, který byl vytvořen na počítači s procesorem 680x0, budete takovou

konverzi potřebovat); `noerror` – program `dd` bude pokračovat v činnosti i poté, co nastane nějaká chyba; `sync` – každý vstupní blok se doplní znaky NUL tak, aby měl délku definovanou prostřednictvím volby `ibs`.

Příklady

Dříve či později se setkáte s případem, kdy budete chtít pod operačním systémem Linux vytvořit svoji první disketu. Jak zapsat data na disketu bez souborového systému MS-DOS? Řešení je jednoduché:

```
% dd if=disk.img of=/dev/fd0 obs=18k count=80
```

V uvedeném příkladu jsem se rozhodl nepoužít volbu `ibs`, protože nevím jaká je optimální volba pro velikost bloku na pevném disku. Nemůže však uškodit, když se místo volby `obs` použije volba `bs` – pak je proces kopírování dokonce rychlejší. Všimněte si nastavení počtu sektorů pro zápis (18 kilobajtů je velikost sektoru, proto je hodnota `count` nastavena na `80`). Pro disketovou jednotku se v operačním systému Linux používá jméno zařízení `/dev/fd0`.

Další užitečné použití příkazu `dd` je v oblasti zálohování, zejména při zapojení počítače do sítě. Předpokládejme, že máte počítač alfa a že na počítači beta je magnetopásková jednotka `/dev/rst0` obsahující soubor, který chcete zkopírovat. Dále předpokládejme, že máte stejná přístupová práva na obou počítačích a že na pevném disku počítače beta není dostatek místa pro kopii uvažovaného souboru. Pak můžete použít příkaz:

```
% rsh beta 'dd if=/dev/rst0 ibs=8k obs=20k' | tar xvBf -
```

Celou operaci takto realizujete „na jeden průchod“. Zajisté jste si všimli, že se v příkazu využila schopnost příkazového procesoru číst data z magnetopáskové jednotky. Velikosti vstupních a výstupních bloků jsou nastaveny na implicitní hodnoty, tedy 8 kilobajtů pro čtení a 20 kilobajtů pro zápis do sítě ethernet.

Poznámka na závěr: zapomněl jsem říci, že označení příkazu `dd` je zkratkou pro výraz „data duplicator“.

KAPITOLA 12

Chyby, skryté závady a další nepříjemnosti

Jak předcházet chybám

Řada lidí na nejrůznějších fórech dává najevo zklamání z operačního systému Unix. Jejich zklamání zpravidla vyplývá z toho, že jej neumějí efektivně využívat – obvykle se jedná o uživatele dříve zvyklé na operační systémy s pohodlnou obsluhou, jako je Microsoft Windows, Macintosh Operating System a podobně. Naopak lidé, kteří zvládli filosofii operačního systému Unix a jsou schopni v něm efektivně pracovat, na něj nedají dopustit. Cení si zejména toho, že jen málokterý příkaz vyžaduje v průběhu své realizace nějakou komunikaci s uživatelem, a proto v operačním systému vše běží hladce, rychle a bez problémů.

Právě skutečnost, že například příkazy `rm` a `mv` nikdy nevyžadují potvrzení, zda mají skutečně zrušit specifikované soubory, vede často k problémům. Proto si nyní projděme malý seznam, v němž uvádíme zásady, jak se takovým a podobným problémům vyhnout.

- Pořizujte si záložní kopie. Tato výzva platí zejména pro uživatele, kteří používají systém sami. Každý systémový administrátor by měl pravidelně pořizovat záložní kopie – dostatečný interval je asi jeden týden. Podrobnosti najdete v *„Příručce správce operačního systému Linux“*.
- Každý uživatel by si měl pořizovat své vlastní záložní kopie. Pokud používáte více jak jeden systém, pak si uchovávejte aktualizované kopie všech svých souborů na každém systému. Jestliže máte přístup k disketové jednotce, pak si na ni ukládejte kopie důležitých souborů. Přinejhorším si kopie důležitých souborů uchovávejte alespoň v oddělených adresářích.
- Důkladně si promyslete, zda jste správně zadali „destruktivní“ příkazy, jako je `mv`, `rm` a `cp`. Zrovna tak si dejte pozor na přesměrování (>) souborů – i to může být nebezpečné a vyžaduje zvláštní pozornost. Nevinně vyhlížející opomenutí může způsobit katastrofu:

  ```
  /home/larry/report# cp report-1992 report-1993 backups
  ```

  Stačí vynechat poslední parametr a neštěstí je hotovo (místo zálohy máte zrušen jeden soubor):

  ```
  /home/larry/report# cp report-1992 report-1993
  ```

- Autor rovněž doporučuje na základě vlastních zkušeností neprovádět zálohování ani další úlohy údržby pozdě v noci, kdy jste unaveni a snadno uděláte chybu. Jestli o půl druhé v noci zjistíte, že máte příliš plný disk, pak jej nechte plným a nezačínejte hned rušit soubory – takovou práci si nechte až na ráno, kdy budete vyspaní a odpočatí.
- Používejte takovou výzvu příkazového řádku, která vás bude informovat o aktuálním adresáři. Pokud takovou výzvu nepoužíváte, rovněž hrozí nebezpečí. Následující příklad nám zaslal člen diskusní skupiny `comp.unix.admin`[1], který si nevšiml, že není v adresáři `/tmp`:

```
mousehouse> pwd
/etc
mousehouse> ls /tmp
passwd
mousehouse> rm passwd
```

  Série právě uvedených příkazů by vás mohla učinit velmi nešťastnými při pohledu na to, jak si rušíte soubor `passwd`, bez kterého se nemůže nikdo do operačního systému Unix přihlásit.

Chyba není ve vás

Naneštěstí pro programátory na celém světě platí, že ne všechny problémy pocházejí z chyb uživatelů. Operační systémy Unix a Linux jsou velmi komplikované a všechny známé verze mají chyby či skryté závady. Některé z těchto závad lze jen velmi těžko odstranit, protože se projevují jen za velmi speciálních okolností.

V souvislosti s chybami programového vybavení se používá termín „bug“ (doslova štěnice). Asi nejvýstižnější překlad bude „skrytá chyba“, protože se jedná o chybu, o které autoři programového vybavení neví. Chyba tohoto druhu se odhalí až při testování daného programu, někdy až po velmi dlouhé době.

Co dělat, když objevíte skrytou chybu

Když vám počítač dá chybnou odpověď (nejdříve důkladně prověřte, zda je odpověď opravdu chybná) nebo nějaký program zhavaruje, pak lze uvažovat o skryté chybě.

Skrytou chybu může obsahovat program, který stále běží, přesto že by měl skončit – i v tomto případě byste však měli nejdříve zkontrolovat, zda neprovádí příliš složité a zdlouhavé operace. Než použijete nový příkaz, důkladně si prostudujte příslušnou dokumentaci (nejlépe manuálové stránky).

Některá hlášení vám budou připadat jako skryté chyby, i když ve skutečnosti skrytými chybami nebudou. Prostudujte si oddíl Hlášení jádra systému a příslušnou dokumentaci, než učiníte nějaké závěry o skrytých chybách.

Například taková hlášení, jako „`disk full`“ nebo „`lp0 in fire`“ neznamenají, že je program vadný, ale že je něco v nepořádku s vaším technickým vybavením – nedostatek diskového prostoru nebo špatně připojená tiskárna.

1 Jedná se o mezinárodní diskusní skupinu, která řeší otázky kolem správy operačního systému Unix.

Pokud nemůžete najít něco v dokumentaci, pak je to chybou dokumentace a měli byste autora programového vybavení kontaktovat a o nedostatku informovat. Zrovna tak je chybou dokumentace, když popisuje jiné vlastnosti programového vybavení, než jaké ve skutečnosti jsou.[2] Je-li něco nekompletního nebo nejasného v dokumentaci, pak se jedná o chybu dokumentace.

Naopak, jestliže nejste schopni porazit šachový program `gnuchess`, pak je to tím, že algoritmus tohoto programu je opravdu dobrý a neznamená to nutně, že je ve vašem mozku skrytá chyba.

Jak ohlásit chybu

Jakmile jste si jisti, že jste odhalili skrytou chybu, je nutné zajistit, aby se hlášení o ní dostalo na správné místo. Pokuste se zjistit, co chybu způsobilo a pokuste se rekonstruovat všechny okolnosti, za jakých chyba nastala. Jestli se vám chybu nepodaří znovu „vyvolat“ pročtěte si zprávy z diskusní skupiny `comp.os.linux.help` nebo `comp.unix.misc`. Také si důkladně pročtěte manuálové stránky k danému programu.

Při odesílání zpráv o skryté chybě se dává přednost elektronické poště. Pokud nemáte možnost odesílat zprávy prostřednictvím elektronické pošty, kontaktujte osobu, od které máte operační systém Linux nebo zkuste kontaktovat někoho, kdo přístup k elektronické poště má. Také se můžete obrátit na komerčního dodavatele operačního systému Linux. Ten má určitě zájem na tom, aby se skryté chyby rychle odstraňovaly. Mějte na paměti, že nikdo není povinen odstraňovat chyby, dokud s ním nemáte podepsanou příslušnou smlouvu.

Když odesíláte hlášení o chybě, pak uveďte všechny informace a okolnosti, za kterých se chyba projevila. Dodržujte dále uvedené zásady:

- Popište, co si myslíte, že má program dělat a co ve skutečnosti dělá. Například: „Program vrací hodnotu 5, když mu byl zadán výraz 2+2“. Nebo „Program ohlásil `segment violation -- core dumped`“. Je velmi důležité popsat, co se stalo, aby mohl autor chybu odstranit.
- Uveďte všechna nastavení proměnných prostředí.
- Uveďte verzi jádra operačního systému (najdete ji v souboru `/proc/version`) a verzi systémových knihoven (podívejte se do adresáře `/lib`, nebo pošlete výpis obsahu tohoto adresáře).
- Popište, jak jste program spustili, a popište, co jste dělali, když k chybě došlo.
- Uveďte všechny okolnosti, za kterých k chybě došlo. Řekněme například, že příkaz `w` některému uživateli nezobrazí informace o aktuálním procesu. Nestačí napsat: „Příkaz `w` nefunguje pro jistého uživatele“. Chyba může nastat proto, že jméno uživatele má osm znaků nebo proto, že se přihlásil prostřednictvím sítě. Správná informace bude tedy znít takto: „Příkaz `w` nezobrazuje informace o aktuálním procesu uživateli greenfie, když se přihlásí prostřednictvím sítě.“.
- Buďte zdvořilí. Většina lidí pracuje obětavě na volném programovém vybavení z vlastní iniciativy, vlastní dobré vůle a především proto, aby vám předložili program, který potřebujete. Nebuďte na lidi, kteří pracují od rána do noci, hrubí – právě hrubé osočování dokázalo odradit nejednoho nadějného programátora od práce na operačním systému Linux.

2 To bude asi také platit o tomto manuálu.

Technické informace

Editor `vi` (vyslovuje se [ví áj]) je jediným editorem, který se nachází v každé instalaci operačního systému Unix. Původně byl vytvořen na univerzitě California v Berkeley, jeho různé verze se nacházejí ve všech distribucích operačního systému Unix a je také součástí distribuce operačního systému Linux. Editor `vi` se poměrně těžko učí, ale má mnoho velmi výkonných funkcí. Obecně se doporučuje začátečníkům editor Emacs, protože se mnohem snáze používá. Avšak uživatelé, kteří používají více počítačových platforem, raději pracují s editorem `vi`.

Abychom pochopili, proč klávesa K znamená posun kurzoru o jeden řádek nahoru a proč editor pracuje ve třech odlišných módech, musíme se podívat do historie. Pokud máte pokušení naučit se pracovat s editorem `vi`, pak můžete považovat tento dodatek za učebnici, která vás provede veškerými základy. Také zde předkládáme přehled příkazů, který můžete považovat za referenční příručku.

Dokonce i v případě, že editor `vi` nebude editorem pro vaši běžnou práci, není čas věnovaný jeho základům úplně zbytečný. Je téměř jisté, že v operačním systému typu Unix, který používáte, je editor `vi` instalován. Někdy je nezbytné použít editor `vi` při instalaci jiných programových produktů, například editoru Emacs. Spousta nástrojů operačního systému Unix, aplikací a her používá jistou podmnožinu příkazů editoru `vi`.

Stručná historie editoru vi

První textové editory byly orientovány na zpracování textových souborů řádek po řádku a používaly se na neinteligentních terminálech. Typickým editorem, který takto funguje, je editor **Ed**. Editor Ed je velmi výkonný a využívá velmi málo zdrojů počítače. V porovnání s editorem Ed nabízí editor `vi` uživateli vizuální alternativu s mnohem širší množinou příkazů.

Editor `vi` se startuje stejně jako řádkový editor `ex`. Platí, že editor `ex` je vlastně editor `vi` spuštěný ve speciálním módu. Vizuální složka editoru `ex` může být inicializována z příkazového řádku pomocí příkazu editoru `vi` nebo přímo z editoru `ex`.

Editor `ex/vi` byl vyvinut na univerzitě California v Berkeley a jeho autorem je William Joy. Původně byl dodáván jako nepodporovaný obslužný program. Oficiálně byl zařazen až v operačním systému System V Unix od firmy AT&T. Postupně se stal velmi populárním a dodnes konkuruje moderním celoobrazovkovým editorům.

V důsledku své popularity se editor `vi` objevil v mnoha verzích a dnes existují verze pro většinu operačních systémů (i jiných, než Unix). Cílem této kapitoly není popsat všechny dostupné příkazy editoru `vi` a jejich modifikace. Právě v důsledku toho, že vzniklo mnoho verzí editoru `vi`, není množina jeho příkazů standardizována. Řada klonů editoru `vi` dokonce nepodporuje některé původní příkazy.

Jestliže máte nějaké zkušenosti s editorem ed, pak se editor vi naučíte mnohem snáze. I když editor vi nehodláte používat, budou se vám základní znalosti hodit zejména v nouzových situacích.

Stručný výklad příkazů editoru Ed

Cílem tohoto oddílu je naučit vás pracovat s editorem ed. Byl navržen tak, aby se jej každý snadno naučil. Než jej budete moci používat, budete muset chvíli trénovat. Při probírání jednotlivých příkazů si hned vyzkoušejte uvedené příklady – tak se naučíte s editorem pracovat velmi rychle.

Vytvoření souboru

Editor ed je schopen editovat v daném okamžiku pouze jeden soubor. Vyzkoušejte si následující příklad a vytvořte si svůj první soubor prostřednictvím editoru ed.

```
/home/larry# ed

a

This is my first text file using Ed.
This is really fun.
.
w firstone.txt
q
/home/larry#
```

Nyní si můžete obsah souboru ověřit pomocí příkazu cat nebo more:

```
/home/larry# cat firstone.txt
```

Předcházející příklad ilustruje spoustu důležitých aspektů. Po spuštění editoru se objeví prázdný řádek. Příkaz a je určen k přidání textu do souboru. Pokud chcete ukončit zadávání textu, zapište do prvního sloupce na novém řádku tečku. Chcete-li text uložit, pak zadejte příkaz w a jméno souboru. Samotný příkaz q činnost editoru ukončí.

Nejdůležitější je poznatek, že editor pracuje ve dvou módech – v **textovém módu** a v **příkazovém módu**. Svoji činnost editor zahajuje v příkazovém módu, kdy lze na prázdných řádcích zadávat příkazy. Ty jsou definovány prostřednictvím jisté množiny znaků.

Jak editovat existující soubor

Pokud chcete do existujícího souboru přidat řádek, postupujte dle následujícího příkladu:

```
/home/larry# ed firstone.txt

a

This is a new line of text.

q
```

Když zkontrolujete soubor firstone.txt pomocí příkazu cat, zjistíte, že nový řádek byl vložen mezi původní první a druhý řádek. Jak editor ed pozná, kam má vložit nový textový řádek?

Po načtení souboru si editor `ed` uchovává informaci o aktuálním řádku. Příkaz a vkládá nový text za aktuální řádek. V editoru `ed` také můžete vkládat nový řádek před aktuální řádek a to prostřednictvím příkazu `i`.

Nyní je patrné, že editor `ed` pracuje s textem řádek po řádku. Všechny příkazy mohou být aplikovány na specifikovaný řádek.

Dále uveďme příklad, jak přidat textový řádek na konec souboru:

```
/home/larry# ed firstone.txt
$a
The last line of text.
.
w
q
```

Modifikátor příkazu `$` „sdělí“ editoru `ed`, že má přidat řádek za poslední řádek stávajícího textu. Pokud byste chtěli přidat textový řádek za první řádek, použijte modifikátor `1`. Nyní máte k dispozici příkazy, pomocí kterých lze vkládat nový text před nebo za řádek specifikovaného čísla.

Jak se dozvíme, který řádek je aktuální? Stačí zadat příkaz `p` a zobrazí se obsah aktuálního řádku. Pokud chcete změnit aktuální řádek na hodnotu 2, pak postupujte dle následujícího příkladu:

```
/home/larry# ed firstone.txt
2p
q
```

Podrobnosti o číslování řádků

Ukázali jsme si, jak prostřednictvím příkazu `p` zobrazit obsah aktuálního řádku. Také víme, že pro příkazy existují modifikátory ve formě pořadových čísel řádků. Chceme-li vypsat obsah druhého řádku, stačí zadat příkaz:

2p

Existují také jiné speciální modifikátory, prostřednictvím kterých lze měnit nastavení aktuálního řádku. Znak dolaru `$` se používá pro poslední řádek textu. Chcete-li vypsat obsah posledního řádku, zadejte:

```
$p
```

Pro aktuální číslo řádku se používá speciální modifikátor tečka. Obsah aktuálního řádku prostřednictvím tohoto modifikátoru lze vypsat takto:

```
.p
```

Možná se vám takový příkaz zdá zbytečný. Když však často měníte obsah aktuálního řádku, pak zjistíte, jak je příkaz užitečný.

Pokud chcete zobrazit text v rozsahu řádků od 1 do 2, pak musíte specifikovat rozsah:

```
1,2p
```

První číslo odkazuje na počáteční řádek, jenž se má zobrazit, a druhé číslo odkazuje na poslední řádek, jenž se má zobrazit. Aktuální řádek se po provedení tohoto příkazu nastaví na druhou hodnotu.

Další příklad ukazuje, jak zobrazit obsah souboru od prvního řádku po aktuální řádek:

```
1,.p
```

Také můžete zobrazit obsah souboru od aktuálního řádku po řádek poslední:

```
.,$p
```

Nyní budete jistě umět zadat příkaz, který zobrazí obsah celého souboru.

Dále si ukážeme, jak zrušit první dva řádky souboru:

```
1,2d
```

Příkaz `d` ruší text řádek po řádku. Chcete-li zrušit celý text, stačí zadat:

```
1,$d
```

Pokud provedete v editovaném textu velké množství změn a nechcete obsah souboru uložit, pak je nejlepší editor ukončit bez zápisu.

Většina uživatelů nepoužívá editor `ed` jako svůj hlavní editor. Moderní editory jsou celoobrazovkové a nabízejí mnohem pružnější množinu příkazů. Editor `ed` představuje dobrý úvod k editoru `vi` a pomůže vám pochopit, odkud příkazy editoru `vi` pocházejí.

Stručný výklad příkazů editoru vi

Cílem tohoto oddílu je naučit vás pracovat s editorem `vi`. Předpokládáme, že nemáte s editorem `vi` žádné zkušenosti a probereme si deset nejzákladnějších příkazů. Tyto příkazy vám budou stačit k realizaci nejnutnějších kroků při editování souboru a další příkazy se pak snadno naučíte podle svých potřeb. Při probírání jednotlivých příkazů si hned vyzkoušejte uvedené příklady – tak se naučíte s editorem pracovat velmi rychle.

Jak spustit editor vi

Chcete-li spustit editor `vi`, zadejte jméno programu a jako parametr uveďte jméno editovaného textového souboru. Objeví se vám obrazovka, na jejíž levé straně bude zobrazen sloupec znaků tilda (~). Editor `vi` se nyní nachází v příkazovém módu – cokoliv napíšete, bude považováno za příkaz a nikoliv za vstupní text. Jestliže chcete zadávat text, musíte nejdříve zadat příkazy. Pro vkládání textu existují tyto dva základní příkazy:

- `i` vložení textu vlevo od kurzoru
- `a` přidání textu vpravo od kurzoru.

Protože se v tomto okamžiku nacházíte na začátku prázdného souboru, nezáleží na tom, který z uvedených příkazů použijete. Zadejte tedy jeden z nich a zapište následující text. (Jedná se o báseň, jejímž autorem je Augustus DeMorgan. Je uvedena v manuálu *„The Unix Programming Environment“,* který napsali pan B.W. Kernighan a R. Pike.):

```
Great fleas have little fleas<Enter>
  upon their backs to bite 'em,<Enter>
And little fleas have lesser fleas<Enter>
  and so and infinitum.<Enter>
And great fleas themselves, in turn,<Enter>
  have greater fleas to go on;<Enter>
While these again have greater still,<Enter>
  and greater still, and so on.<Enter>
<Esc>
```

Všimněte si, že k ukončení vkládaného textu se používá klávesa Esc. Pak editor opět přejde do příkazového módu.

Příkazy pro pohyb kurzoru

- `h` posunutí kurzoru o jeden znak doleva
- `j` posunutí kurzoru o jeden řádek dolů
- `k` posunutí kurzoru o jeden řádek nahoru
- `l` posunutí kurzoru o jeden znak doprava

Uvedené příkazy mohou být opakovány tak, že danou klávesu budete držet stisknutou. Nyní si vyzkoušejte pohyb kurzoru všemi směry. Pokud se pokusíte posunout kurzor na neexistující pozici (například když stisknete klávesu K a přitom je kurzor na prvním řádku souboru), pak obrazovka blikne nebo se ozve zvukový signál. Ničeho se neobávejte, váš soubor nebude v takovém případě nijak poškozen.

Jak rušit text

- `x` zrušení znaku na pozici kurzoru
- `dd` zrušení řádku

Posuňte kurzor na druhý řádek a nastavte jeho pozici pod apostrof ve slově `'em`. Stiskněte klávesu X a apostrof zmizí. Nyní stiskněte klávesu I (přepnete tak editor `vi` do módu, ve kterém se vkládá text) a zapište „`th`“. Nakonec stiskněte klávesu Esc.

Uložení souboru

- `:w` uložení souboru (na disk)
- `:q` ukončení editoru `vi`

Nejdříve se přesvědčte, že jste v příkazovém módu – stiskněte klávesu Esc. Nyní zadejte `:w`. V tomto okamžiku se vámi vytvořený text uloží do diskového souboru.

Příkaz pro ukončeni editoru `vi` je reprezentován příkazem `:q`. Jestliže chcete zkombinovat uložení souboru a ukončení editoru, pak použijte příkaz `:wq`. Pro příkaz `:wq` existuje ekvivalentní zkratka `ZZ`. Zkratka `ZZ` se bude hodit zejména programátorům. Podívejme se na typickou posloupnost akcí, které provádí programátor při ladění programu: spuštění programu; zjištění problému; editování souboru obsahujícího zdrojový kód programu; provedení změn v souboru a jeho uložení na disk; přeložení programu; spuštění programu; ...a tak stále dokola. Pak bude programátor příkaz `ZZ` zřejmě používat velmi často. Ve skutečnosti není příkaz `ZZ` přesným ekvivalentem příkazu `:wq`. Příkaz `ZZ` totiž neprovede uložení souboru, pokud v textu neprovedete žádné změny, zatímco příkaz `:wq` soubor před ukončením editoru soubor vždy uloží.

Jestliže jste udělali nějaké změny, ale pak nechcete soubor ukládat, použijte příkaz `:q!` (nezapomeňte předtím stisknout klávesu Esc). Pokud byste zapomněli znak „`!`“ editor `vi` vám nepovolí ukončit práci bez uložení souboru.

Co bude následovat

Deset příkazů, které jsme v předcházejících odstavcích probrali, stačí k tomu, abyste byli schopni pracovat s editorem vi. Tyto příkazy však představují pouze základ práce s editorem. Existují další příkazy, například pro kopírování textu z jednoho místa na druhé, pro přesouvání textu z jednoho souboru do druhého, pro konfiguraci editoru a podobně. V editoru vi lze aplikovat asi 150 příkazů.

Pokročilejší techniky práce s editorem vi

Velká výhoda editoru `vi` spočívá v tom, že jej můžete používat, i když znáte jen několik málo základních příkazů. Většina uživatelů je nadšena, že stačí znát jen pár příkazů, ale jakmile s editorem pracují déle, potřebují k realizaci některých úloh další příkazy.

V následujícím oddílu se předpokládá, že se uživatel seznámil se základními příkazy uvedenými v předcházejícím oddílu. Nyní si probereme další příkazy – od kopírování textů až po definici maker.

Také uvedeme oddíl o konfiguraci editoru `vi`, kde se dozvíte, jak nastavit některé vlastnosti editoru, aby byla vaše práce co nejefektivnější. Následující odstavce jsou zaměřeny spíše na popis příkazů a nejsou již tolik zaměřeny na jejich procvičování. Proto doporučujeme, abyste si probírané příkazy průběžně procvičovali sami.

Nebudeme uvádět zcela vyčerpávající seznamy příkazů editoru `vi`, ale zaměříme se na příkazy nejčastěji používané zejména z praktického hlediska. I když si nakonec zvolíte pro svou běžnou práci jiný editor, budou se vám nabyté znalosti vždy hodit.

Příkazy pro změnu polohy kurzoru

K nejzákladnějším funkcím editoru patří příkazy pro změnu polohy kurzoru. Zde uvádíme jejich přehled.

h	posunutí kurzoru o jeden znak doleva
j	posunutí kurzoru o jeden řádek dolů
k	posunutí kurzoru o jeden řádek nahoru
l	posunutí kurzoru o jeden znak doprava

Některé implementace editoru vi také umožňují používat kurzorové klávesy.

w	posunutí kurzoru na začátek následujícího slova
e	posunutí kurzoru na konec následujícího slova
E	posunutí kurzoru na konec následujícího slova před mezeru
b	posunutí kurzoru na začátek předcházejícího slova
0	posunutí kurzoru na začátek řádku
^	posunutí kurzoru na první slovo v aktuálním řádku
$	posunutí kurzoru na konec řádku
<CR>	posunutí kurzoru na začátek následujícího řádku
-	posunutí kurzoru na začátek předcházejícího řádku
G	posunutí kurzoru na konec souboru
1G	posunutí kurzoru na začátek souboru
nG	posunutí kurzoru na řádek s pořadovým číslem n
Ctrl-Shift-G	zobrazení čísla aktuálního řádku
%	posunutí kurzoru na odpovídající hranatou závorku
H	posunutí kurzoru na první řádek obrazovky
M	posunutí kurzoru na prostřední řádek obrazovky
L	posunutí kurzoru na spodní řádek obrazovky
nl	posunutí kurzoru na sloupec n

Jestliže kurzor dospěje na spodní nebo horní řádek obrazovky, pak se zobrazený text vždy příslušným směrem posune. Nyní uvedeme alternativní příkazy, které umožňují posouvání zobrazovaného textu na obrazovce (tzv. rolování).

Ctrl-f	posunutí textu o stránku nahoru
Ctrl-b	posunutí textu o stránku dolů
Ctrl-d	posunutí textu o půl stránky dolů
Ctrl-u	posunutí textu o půl stránky nahoru

Právě uvedené příkazy jsou určeny k pohybu kurzoru. Některé z nich mohou být modifikovány pomocí čísla, které se uvede před příkaz. Předřazené číslo n znamená, že se příkaz bude opakovat n-krát.

Uveďme si tvar příkazu pro posunutí kurzoru o n pozic doleva:

nl	posunutí kurzoru o n pozic (znaků) doprava

Jestliže chcete například zařadit před text větší počet mezer, můžete použít podobnou modifikaci příkazu i pro vkládání textu. Zadejte počet mezer, pak příkaz i následovaný mezerou a nakonec stiskněte klávesu ESC.

ni opakované vložení textu n-krát

Příkazy odkazující na řádky mohou jako modifikátor použít číslo řádku. Ukázkovým příkladem je příkaz G:

1G posunutí kurzoru na první řádek textového souboru

Editor `vi` má velké množství příkazů, které lze použít ke změně pozice kurzoru – od jednoduchých příkazů, kde se kurzor posune o jednu pozici, až po složitější příkazy, kdy se kurzor nastavuje na předem specifikovanou pozici. Také lze zadat nastavení kurzoru na specifikovaný řádek při spouštění editoru, například:

```
vi +10 myfile.tex
```

Uvedený příkaz otevře soubor myfile.tex a umístí kurzor na desátý řádek od začátku souboru.

Vyzkoušejte si příkazy uvedené v tomto oddíle a procvičte se v jejich používání. Většina uživatelů používá pouze jistou podmnožinu uvedených příkazů. V dalším oddílu si probereme příkazy umožňující text měnit.

Modifikace textu

Nyní je naším cílem měnit obsah textového souboru a editor vi za tímto účelem nabízí spoustu příkazů.

V tomto oddíle budeme probírat příkazy určené k editování textu, rušení textu a podobně. Až oddíl dočtete, budete mít dostatek znalostí potřebných k vytvoření jakéhokoliv textového souboru. Následující oddíly jsou pak zaměřeny na další užitečné příkazy.

Při vkládání textu lze vložit více textových řádků prostřednictvím klávesy Enter. Předpokládejme, že jste udělali chybu a že jste stále na řádku, ve kterém se chyba vyskytuje.

Pak můžete použít klávesu Backspace a vrátit se k místu, kde se nachází chyba. Pokud jde o klávesu Backspace, různé implementace editoru vi se chovají různě. Některé z nich pouze posunují kurzor zpět a zapsaný text nechávají nedotčený. Jiné text postupně zprava ruší. Některé implementace dokonce umožňují používat v módu zadávání textu kurzorové klávesy. To vše však nepatří k normálnímu chování editoru `vi`. Pokud je text viditelný a použijete klávesu Esc (přitom jste na řádku, ve kterém jste použili klávesu Backspace), pak budou znaky za kurzorem zrušeny. Funkci klávesy Backspace si budete muset vyzkoušet, abyste zjistili, jak se tato klávesa ve vaší konkrétní implementaci editoru vi chová.

a	Přidání textu od aktuální pozice kurzoru
A	Přidání textu na konec řádku
i	Vložení textu vlevo od pozice kurzoru
I	Vložení textu vlevo od prvního výskytu zobrazitelného znaku (non-white character) v aktuálním řádku
o	Otevření nového řádku a vložení textu za aktuální řádek
O	Otevření nového řádku a vložení textu před aktuální řádek

Nyní máme několik příkazů pro vkládání textu a dále si uvedeme příkazy pro zrušení textu. Editor `vi` má několik příkazů pro zrušení textu, které mohou být spojeny s modifikátorem.

X	zrušení znaku, na jehož pozici je umístěn kurzor
dw	zrušení textu od aktuální pozice kurzoru do konce slova
dd	zrušení aktuálního řádku
D	zrušení textu od aktuální pozice kurzoru do konce řádku

Prostřednictvím modifikátorů se síla příkazů zvýší. Následující příklady jsou pouze podmnožinou všech možností:

nx	n-krát opakované zrušení znaku, na jehož pozici je umístěn kurzor
ndd	zrušení n řádků
dnw	zrušení n slov (stejnou funkci má příkaz ndw)
dG	zrušení textu od aktuální pozice kurzoru do konce souboru
d1G	zrušení textu od aktuální pozice kurzoru do začátku souboru
d$	zrušení textu od aktuální pozice kurzoru do konce řádku
dn$	zrušení textu od aktuální pozice kurzoru do konce n-tého řádku

Uvedené příklady ukazují, že operace zrušení textu mohou být velmi efektivní. Je to zejména evidentní tehdy, když tyto operace použijete ve spojení s příkazy pro změnu polohy kurzoru. Poznamenejme, že příkaz `D` tvoří výjimku, protože ignoruje jakékoli modifikátory.

Příležitostně nastanou situace, kdy budete chtít provedené změny vrátit zpět. Následující příkazy jsou určeny pro obnovení textu:

u	zrušení posledně provedeného příkazu
U	zrušení všech změn provedených v tomto řádku
:e!	obnova stavu souboru od okamžiku, kdy byl naposledy uložen

Editor `vi` umožňuje nejen vracet zpět provedené změny, ale také vracet zpět vrácené změny. Například pomocí příkazu `5dd` zrušíte pět řádků a pak je obnovíte pomocí příkazu `u`. Když použijete příkaz `u` znovu, budou řádky opět zrušeny.

Nové verze editoru `vi` nabízejí příkazy, které umožňují editovat text v tzv. přepisovacím módu. Znamená to, že chcete-li měnit nějaký text, pak jej nemusíte nejdříve rušit.

r c	přepsání znaku na pozici kurzoru znakem
R	přepsání textu novým textem
c w	změna textu v aktuálním slově
c $	změna textu od aktuální pozice kurzoru do konce řádku
c n w	změna následujících n slov (totéž jako ncw)
c n $	změna do konce n-tého řádku
c	změna do konce řádku (totéž jako c$)
c c	změna aktuálního řádku
S	v textu, který právě píšete, nahradí aktuální znak
n s	v textu, který právě píšete, nahradí n aktuálních znaků

Série příkazů realizujících změny vám umožní zadat řetězec znaků, který musí být ukončen klávesou Esc.

Příkaz c w platí od aktuální pozice kurzoru ve slově do konce slova. Když použijete příkaz realizující změnu a přitom zadáte „vzdálenost“, editor `vi` zobrazí znak `$` na pozici, do které bude změna provedena. Nový text může být delší nebo kratší než text původní.

Kopírování a přesouvání částí textu

Pro přesouvání textu existuje řada příkazů, jejichž kombinací lze dosáhnout kýženého efektu. V tomto oddíle popíšeme pojmenované a nepojmenované buffery spolu s příkazy umožňujícími „vystřihnout“ a „nalepit“ text.

Základní kopírování textu probíhá ve třech krocích:

1. Kopírování textu do bufferu („**vystřižení**“ textu).
2. **Nastavení kurzoru** na cílovou pozici.
3. Kopírování textu z bufferu na novou pozici („**nalepení**“ textu).

K vystřižení textu do nepojmenovaného bufferu použijte některý z následujících příkazů:

y y	Přesunutí kopie aktuálního řádku do nepojmenovaného bufferu.
Y	Přesunutí kopie aktuálního řádku do nepojmenovaného bufferu.
c y y	Přesunutí kopie následujících n řádků do nepojmenovaného bufferu.
c Y	Přesunutí kopie následujících n řádků do nepojmenovaného bufferu.
y w	Přesunutí slova do nepojmenovaného bufferu.
y n w	Přesunutí n slov do nepojmenovaného bufferu.
n y w	Přesunutí n slov do nepojmenovaného bufferu.
y $	Přesunutí textu od aktuální pozice do konce řádku.

Nepojmenovaný buffer je dočasná oblast paměti, jejíž obsah může být zrušen prostřednictvím jiných často používaných příkazů. Někdy se stane, že jistou část textu budete potřebovat po dlouhou dobu. V takovém případě byste měli použít pojmenovaný buffer. Editor `vi` nabízí 26 pojmenovaných bufferů. Jako identifikační jméno bufferu se používá jednoznakové písmeno. Pro rozlišení pojmenovaného bufferu editor `vi` používá znak `"`. Při odkazu na pojmenovaný buffer se používají malá písmena (pokud má být obsah bufferu zcela nahrazen) nebo velká písmena (pokud má být obsah bufferu doplněn). Uveďme si příklady:

" a y y	Přesunutí aktuálního řádku do pojmenovaného bufferu a.
" a Y	Přesunutí aktuálního řádku do pojmenovaného bufferu a.
" b y w	Přesunutí aktuálního slova do pojmenovaného bufferu b.
" B y w	Přidání aktuálního slova k obsahu pojmenovaného bufferu b.
" b y 3 w	Přesunutí následujících tří slov do pojmenovaného bufferu b.

Ke zkopírování („nalepení”) obsahu bufferu použijte příkaz `p`:

p	Zkopírování obsahu nepojmenovaného bufferu VPRAVO od pozice kurzoru.
P	Zkopírování obsahu nepojmenovaného bufferu VLEVO od pozice kurzoru.
n P	Zkopírování obsahu nepojmenovaného bufferu n-krát VLEVO od pozice kurzoru.
" a p	Zkopírování obsahu pojmenovaného bufferu a VPRAVO od pozice kurzoru.
" b 3 P	Zkopírování obsahu pojmenovaného bufferu b třikrát VLEVO od pozice kurzoru.

Jestliže používáte editor `vi` v terminálovém okně systému X Window, pak máte k dispozici ještě jeden prostředek pro kopírování textů. Oblast textu, kterou chcete kopírovat, vyznačte kurzorem myši (levé tlačítko myši přitom držte stisknuté). Vyznačená oblast textu bude zobrazena inverzně a přitom se automaticky přenese do speciálního bufferu vyhrazeného pro systém X Window. Po-

kud chcete text „nalepit“, stačí stisknout prostřední tlačítko myši. Nezapomeňte, že předtím musíte editor vi přepnout do vkládacího módu, protože jinak by mohl být vstup interpretován jako příkaz a výsledek by byl nepředvídatelný. Prostřednictvím stejné techniky lze kopírovat jednotlivá slova. Slovo se přenese do bufferu dvojnásobným klepnutím levým tlačítkem myši. Obsah bufferu se změní pouze tehdy, když se vyznačí nová oblast textu.

Přesouvání textu se rovněž odehrává ve třech krocích:

1. **Zrušení textu** a jeho současné zkopírování do pojmenovaného nebo nepojmenovaného bufferu.
2. **Nastavení kurzoru** na cílovou pozici.
3. **Kopírování textu** z pojmenovaného nebo nepojmenovaného bufferu na novou pozici („nalepení“ textu).

Proces je stejný jako kopírování textu, avšak v prvním kroku se text zruší. Když se provede příkaz `dd`, přesune se zrušený řádek do nepojmenovaného bufferu. Pak můžete obsah bufferu „nalepit“ stejným způsobem, jako při kopírování textu.

" a d d	zrušení řádku a přesunutí do pojmenovaného bufferu a
" a A d d	zrušení čtyř řádků a přesunutí do pojmenovaného bufferu a
d w	zrušení slova a přesunutí do nepojmenovaného bufferu. Další příklady na zrušení textu jsme uvedli v oddíle o modifikaci textu.

V případě, že systém zhavaruje, obsahy pojmenovaného i nepojmenovaných bufferů se ztratí. Obsah editovaného bufferu však může být obnoven.

Vyhledávání a nahrazování textů

Editor `vi` má spoustu příkazů pro vyhledávání řetězců. Můžete vyhledávat jednotlivé znaky, ale také můžete při vyhledávání použít regulární výrazy.

Hlavní vyhledávací příkazy jsou `f` a `t`:

f c	vyhledání následujícího znaku c vpravo od kurzoru
F c	vyhledání následujícího znaku c vlevo od kurzoru
t c	přesunutí kurzoru na pozici vlevo od následujícího znaku c
T c	přesunutí kurzoru na pozici vpravo od předcházejícího znaku c (V některých verzích editoru vi je tento příkaz shodný s příkazem Fc.)
;	opakování posledního příkazu f, F, t nebo T
,	stejný příkaz jako ;, ale mění směr vyhledávání původního příkazu

Jestliže hledaný znak neexistuje, upozorní vás editor `vi` zvukovým nebo jiným signálem.

Editor `vi` rovněž umožňuje v editovaném textu hledat řetězec:

/`str`	vyhledání specifikovaného řetězce od aktuální pozice kurzoru
?`str`	vyhledání specifikovaného řetězce před aktuální pozicí kurzoru
n	opakování předcházejícího příkazu / nebo ?
N	opakování předcházejícího příkazu / nebo ?, přičemž se změní směr vyhledávání

Když použijete příkaz / nebo ?, „vyčistí“ se spodní řádek obrazovky. Pak v tomto řádku zadáte hledaný řetězec a stisknete klávesu Enter.

Řetězec uvedený za příkazy / a ? může být regulárním výrazem. V regulárním výrazu se prostřednictvím pseudoznaků popisují jisté množiny znaků. Pseudoznaky používané v editoru `vi` jsou následující: `.`, `*`, `[`, `]`, `^` a `$`. Následující seznam specifikuje význam pseudoznaků:

`.`	vyhovuje jakýkoliv znak kromě znaku „nového řádku“
`\`	uvozuje jakýkoliv speciální znak
`*`	vyhovuje výskytu předcházejících znaků
`[]`	vyhovuje právě jeden znak uvedený v hranatých závorkách
`^`	vyhovuje právě jeden znak, jen pokud se nachází na začátku řádku
`$`	vyhovuje znak předcházející konci řádku
`[^]`	vyhovují znaky, které nejsou uvedeny v hranatých závorkách
`[-]`	vyhovují znaky z uvedeného rozsahu

Nejjednodušší způsob, jak se naučíte používat regulární výrazy, spočívá v tom, že je budete používat. Vyzkoušejte si následující příklady:

`c.pe`	vyhovují slova jako cope, cape, caper a podobně
`c\.pe`	vyhovují slova jako c.pe, c.per a podobně
`sto*p`	vyhovují slova jako stp, stop, stoop a podobně
`cat.*n`	vyhovují slova jako carton, cartoon a podobně
`xyz.*`	vyhovují xyz do konce řádku
`^The`	vyhovují všechny řádky začínající na The

`atime$`	vyhovují všechny řádky končící na atime
`^Only$`	vyhovují všechny řádky, ve kterých se vyskytuje pouze slovo Only
`b[aou]rn`	vyhovují slova barn, born nebo burn
`Ver[D-F]`	vyhovují slova VerD, VerE nebo VerF
`Ver[^1-9]`	vyhovují slova začínající na Ver, po kterých nenásleduje žádná číslice
`the[ir][re]`	vyhovují slova their, therr, there a theie
`[A-Za-z][A-Za-z]*`	vyhovuje jakékoliv slovo

Editor vi používá příkazový mód `ex` k realizaci operací vyhledávání a náhrady řetězců. Všechny příkazy začínající dvojtečkou představují požadavek v módu `ex`.

Příkazy pro vyhledání a náhradu řetězce umožňují, aby byly realizovány v jistém řádkovém rozsahu. Uživatel může vyžadovat, aby byl před náhradou řetězce vyzván k potvrzení, zda se má konkrétní výskyt řetězce nahrazovat nebo ne. Uveďme si obecný tvar příkazu pro náhradu řetězce a několik příkladů:

...syntaxe obecného příkazu :<start>,<finisn>s/<find>/<replace>/g

:1,$s/the/The/g	vyhledá všechny výskyty slova the a nahradí je slovem The
:%s/the/The/g	znak % znamená „celý soubor“. Tento příkaz provede stejnou funkci jako předcházející příkaz
:.,5s/^.*//g	zruší obsah editovaného souboru od aktuálního řádku po pátý řádek
:%s/the/The/gc	nahradí všechny výskyty slova the slovem The a před každou náhradou si vyžádá potvrzení
:%s/^..../g	zruší první čtyři znaky každého řádku

Jak je z předcházejících příkladů patrné, jsou příkazy pro vyhledávání a náhradu řetězců velmi výkonné, zejména když se kombinují s regulárními výrazy. Pokud se direktiva `g` v příkazu neuvede, provede se náhrada pouze u prvního výskytu hledaného řetězce.

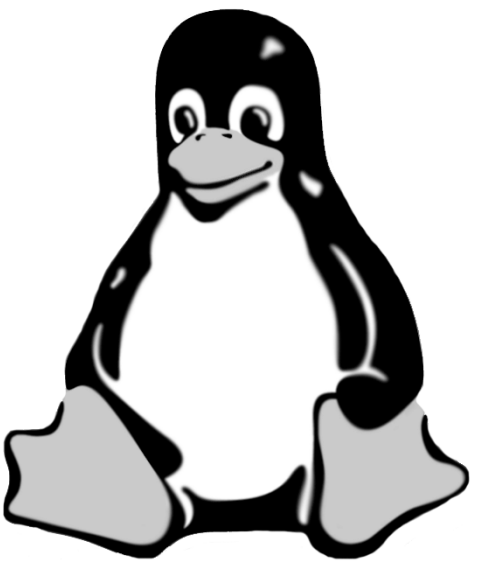
Někdy se může stát, že původní vyhledávaný řetězec se má použít v nahrazujícím řetězci. Za tímto účelem nabízí editor `vi` některé speciální znaky:

:1,5s/help/&ing/g	v prvních pěti řádcích nahradí slovo help slovem helping
:%s/ */&&/g	v celém souboru zdvojnásobí počet mezer mezi slovy

Používání kompletního řetězce má v editoru `vi` jisté omezení, protože k stanovení rozsahu náhrady editor používá kulaté závorky `( a )`. V takovém případě je nutno použít znak `\`:

:s/^\(.*\):.*/\1/g	zruší vše, co následuje po dvojtečce včetně dvojtečky samé
:s/\(.*\):\(.*\)/\2:\1/g	vymění slova na obou stranách dvojtečky

Posledně uvedené příklady asi budete muset číst velmi pozorně. Editor `vi` nabízí velmi výkonné příkazy, které moderní editory zpravidla nemají. Za to se ovšem platí jistá cena – naučit se všechny příkazy editoru vi není vůbec snadné. Pokud jsme vás neodradili, pokuste se znovu si projít uvedené příklady a uvidíte, že se používání editoru `vi` pro vás stane zcela přirozenou záležitostí.



ČÁST II

Příručka správce operačního systému

Originál: http://www.tldp.org/LDP/sag/

KAPITOLA 1

Úvod

Na počátku byl soubor nesličný a pustý, a prázdno se vznášelo nad povrchem bitů. A Ruka Autorova dosedla na povrch klávesnice – i řekl Autor: „Budiž slova!“ A byla slova.

Příručka správce operačního systému Linux popisuje ty aspekty používání operačního systému, jež se vztahují k jeho správě. Je určena lidem, kteří o správě operačního systému neví zhola nic (ptají se teď, co to je), avšak zvládají přinejmenším základy jeho běžného užívání. Nenaleznete zde návod, jak Linux instalovat. Instalace systému je podrobně popsaná v dokumentu „Průvodce instalací a začátky“. Další informace o dokumentaci k systému Linux jsou uvedeny níže.

Administrací (správou) systému rozumíme všechny činnosti, které je nutno pravidelně vykonávat, aby počítačový systém zůstal v provozuschopném stavu. Zahrnuje například zálohování souborů (a v případě potřeby jejich obnovování), instalaci nových programů, vytváření uživatelských účtů (a jejich mazání v případě, že jsou nepotřebné), kontroly a opravy případných poškození systému souborů a další. Když si počítač představíte jako dům, pak by správou systému byla jeho údržba. Ta by zahrnovala například úklid, zasklívání rozbitých oken a další podobné věci.

Příručka je strukturována tak, že většinu kapitol můžete číst nezávisle na sobě. Když například hledáte nějaké informace o zálohování, stačí, když si přečtete příslušnou kapitolu. Přesto manuál zůstává především učebnicí a jakýmsi průvodcem, a můžete jej číst od začátku do konce.

Nelze předpokládat, že by tato knížka pokryla celou problematiku administrace systému. Správce systému bude potřebovat řadu další dokumentace operačního systému Linux. Koneckonců, administrátor je v podstatě jenom uživatel, který má zvláštní práva a povinnosti. Velmi významným pramenem jsou manuálové stránky, po kterých by správce měl sáhnout pokaždé, když si není funkcí některého příkazu zcela jist. Nevíte -li, jaký příkaz použít, vyzkoušejte příkaz **apropos**. Další podrobnosti viz manuálová stránka tohoto příkazu.

Tato příručka je zaměřena především na operační systém Linux, ale v obecných principech může být užitečná i pro správce jiných unixových systémů. Bohužel je mezi různými verzemi Unixu tolik rozdílů (a o správě systému to platí dvojnásob), že není prakticky možné postihnout všechny známé varianty. Je totiž obtížné – vezmeme-li v potaz způsob, jakým se Linux vyvíjí – pokrýt i všechny možnosti jen tohoto operačního systému.

Neexistuje jediná oficiální distribuce Linuxu od jediného výrobce. Různí lidé používají různá nastavení a konfigurace. Navíc si řada uživatelů vytváří své vlastní. Proto tato kniha není zaměřená na některou z konkrétních distribucí. V rámci možností se v příručce snažíme upozornit na některé odlišnosti a objasnit i jiné možné alternativy.

[1] Porozumění představuje v případě Linuxu základní podmínku úspěchu. Tato kniha by mohla být pouhý seznam návodů – jenže co byste dělali tváří v tvář problému, na nějž byste zde návod nenalezli. Pokud vám ale nabídneme vysvětlení principů, nejsou návody nutné – vyplynou samy ze znalosti věci.

Než abychom podali strohý seznam „pěti jednoduchých kroků“ pro řešení každého úkolu, dáváme přednost popisu základních principů, tedy objasnění toho, jak věci doopravdy fungují. V knize proto najdete hodně informací, které nejsou nezbytné pro každého. Takovéto části jsou v textu označeny a v případě, že používáte systém s předem nastavenou konfigurací, můžete je klidně přeskočit. Pochopitelně, přečtete-li si knihu celou, proniknete do systému hlouběji, a pak by pro vás mohly být o něco příjemnější i jeho používání a jeho správa[1].

Tak jako vše ostatní spojené s vývojem Linuxu, byla i tato práce založena na principu dobrovolnosti. Pustili jsme se do ní, protože jsme si mysleli, že by to mohla být zábava. Dalším důvodem byl pocit, že je potřeba tuto práci udělat. Přesto – jako konečně u každé dobrovolné práce – jsou určité hranice nasazení a úsilí, které můžete vynaložit. Navíc vás omezuje také to, kolik vědomostí a zkušeností máte. Přirozeně, manuál není tak dobrý, jak by mohl být v případě, že by přišel někdo s kouzelnou hůlkou a dobře zaplatil za jeho napsání. Pak by bylo možné strávit i několik dalších let jeho zdokonalováním. Samozřejmě si myslíme, že je celkem povedený, nicméně berte to jako varování.

Je jeden konkrétní bod, ve kterém jsme manuál dost „ořezali“ – není v něm vyčerpávajícím způsobem popsaná řada věcí, které již jsou podrobně zdokumentované v jiných volně dostupných příručkách. Vztahuje se to zvlášť na dokumentaci k jednotlivým programům. Neuvádíme například všechny podrobnosti použití programu **mkfs**. Popisujeme jenom funkci programu a pouze tolik z jeho dalších možností, kolik je potřeba pro dosažení účelu této knihy. Laskavého čtenáře, jenž hledá podrobnější informace, odkazujeme na onu další dokumentaci. Převážná většina dokumentů, na které se odvoláváme v odkazech, je součástí úplné sady dokumentace k operačnímu systému Linux.

O této části

Poděkování

Joanna děkuje

Lars se snažil o napsání co nejlepší příručky a já jako současný správce bych ráda v jeho snaze pokračovala. Budeme vděčni za všechny vaše nápady jak ji vylepšit. Gramatické a věcné chyby, nápady týkající se nových oblastí, o které by bylo možno knihu rozšířit, opakující se části, informace o rozdílech mezi různými verzemi Unixu – to všechno jsou připomínky, které se zájmem očekáváme. Kontaktní informace najdete prostřednictvím služby World Wide Web na adrese *http://www.iki.fi/viu/*.

Při práci na této knize nám přímo či nepřímo pomáhalo mnoho lidí. Rádi bychom zvlášť poděkovali Mattu Welshovi za inspiraci a vedení projektu LDP; Andy Oramovi za to, že nás znovu a znovu zaměstnával řadou velmi podnětných připomínek; Olafu Kirschovi za to, že nám dokázal, že vše lze zvládnout; Adamu Richterovi z Yggdrasil a dalším za to, že nám ukázali, že tato práce může být zajímavá i pro jiné lidi.

Stephen Tweedie, H. Peter Anvin, Rémy Card a Theodore Ts'o odvedli kus práce, kterou jsme si formou odkazů a referencí „zapůjčili" (tím pádem je naše kniha na pohled tenčí a o to víc působivá) – sem patří porovnání souborových systémů xia a ext2, seznam zařízení nebo popis souborového systému ext2. Tyto části jsme z knihy vyřadili. Za toto jsme vděční vůbec nejvíc a zároveň se velmi omlouváme za předchozí verze manuálu, které občas v některých oblastech postrádaly odpovídající úroveň.

Kromě toho patří náš dík Marku Komarinskému za jeho materiály z roku 1993 i mnoho dalších sloupků v Linux Journalu, jež se týkaly problematiky správy systému. Jsou velmi informativní a inspirující.

Dostali jsme množství užitečných připomínek od velkého počtu dalších lidí. Díky malé černé díře v našem archivu nelze dohledat všechna jména, takže alespoň některá z nich (v abecedním pořadí): Paul Caprioli, Ales Cepek, Marie-France Declerfayt, Dave Dobson, Olaf Flebbe, Helmut Geyer, Larry Greenfield a jeho otec, Stephen Harris, Jyrki Havia, Jim Haynes, York Lam, Timothy Andrew Lister, Jim Lynch, Michael J. Micek, Jacob Navia, Dan Poirier, Daniel Quinlan, Jouni K. Seppänen, Philippe Steindl, G. B. Stotte. Omlouváme se všem, na které jsme zapomněli.

Stephen děkuje

Coby nový správce této příručky bych chtěl poděkovat Larsovi a Joanně za jejich práci.

V příručce jako je tato se téměř vždy najdou přinejmenším drobné nepřesnosti. Kromě toho se v ní objeví části, které postupně zastarávají. Pokud cokoliv z toho postřehnete, pošlete mi laskavě e-mail na adresu bagpuss@debian.org. Akceptuji připomínky v jakémkoliv formátu – diff, text, HTML, cokoliv. Nikomu nechci bránit pomoci mi s prací nad tímto textem.

Mnohokrát děkuji Helen Topping Shawové za práci s červenou tužkou, díky níž je příručka daleko lepší, než by byla jinak.

Aktuální stránky příručky najdete na adrese http://peope.debian.org/~bagpuss.

Přehled operačního systému Linux

A viděl Bůh vše, což učinil, a aj, bylo velmi dobré.
Genesis 1:31[2]

Tato kapitola podává zevrubný přehled o operačním systému Linux. V první části jsou popsány nejdůležitější ze služeb, jež systém nabízí. Další části se bez přílišných podrobností zabývají programy, které popsané služby realizují. Cílem kapitoly je podat výklad principů systému jako celku s tím, že každá část bude podrobněji probrána později, na jiném místě knihy.

Různé části operačního systému

Operační systém typu Unix se skládá z *jádra* a *systémových programů*. Kromě toho pro různou běžnou práci existují *aplikační programy*. Jádro je srdcem operačního systému[2]. Udržuje záznamy o souborech na disku, spouští programy, řídí jejich současný běh, přiděluje paměť a další technické prostředky různým procesům, přijímá a odesílá pakety z a do počítačové sítě a tak dál. Jádro systému samotné toho dělá velmi málo, ale poskytuje základní služby různým nástrojům, pomocí kterých mohou být realizovány všechny ostatní služby. Jádro rovněž hlídá, aby nikdo nemohl přistupovat k hardwarovým zařízením přímo. Když chtějí uživatelé a procesy používat technické prostředky, musí používat nástroje, které nabízí jádro systému[3]. Tímto způsobem je zabezpečená i vzájemná ochrana uživatelů. Nástroje jádra systému, o nichž byla řeč, lze využívat prostřednictvím *systémových volání*. Podrobnější informace o systémových voláních uvádí sekce 2 manuálových stránek.

Systémové programy realizují služby, které se vyžadují od operačního systému. Využívají při tom nástroje, které nabízí jádro systému. Systémové i všechny ostatní programy běží jakoby „na povrchu“ jádra. Říká se tomu *uživatelský režim*. Rozdíl mezi systémovými a aplikačními programy je v jejich určení. Pomocí aplikačních programů mohou uživatelé dělat některé užitečné věci (popřípadě se bavit – je-li aplikace, kterou si zrovna spustili, počítačová hra). Systémové programy jsou potřebné k tomu, aby systém vůbec fungoval. Textový editor je aplikace, **mount** je systémový

[2] Často bývá jádro považováno za celý operační systém, to ale není pravda. Operační systém nabízí mnohem více služeb, než jen čisté jádro.

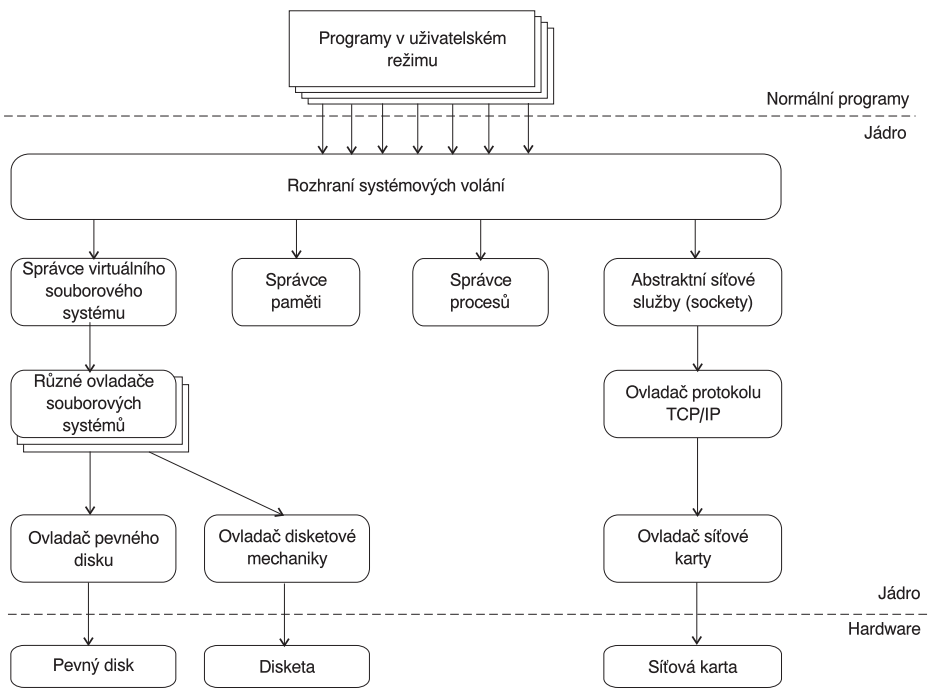
[3] Vždycky jsem si to představoval jako jistou formu zapouzdření, což by mohlo být dostatečně názorné vysvětlení pro ty, kteří jsou odkojeni objektově orientovaným programováním.

program. Hranice mezi aplikačními a systémovými programy je často dost neostrá a je důležitá jen pro zásadové „kategorizéry“.

Součástí operačního systému mohou být i překladače programovacích jazyků a jejich knihovny (v případě Linuxu překladač GCC a knihovna jazyka C), nicméně tomu tak nemusí být. Další částí systému je dokumentace, někdy dokonce i některé hry. Tradičně se za operační systém pokládá obsah jeho instalační pásky či instalačních disků. Pokud jde o systém Linux, není uvedená definice zcela jasná, protože na mnoha serverech FTP po celém světě existuje množství různých instalací systému.

Důležité části jádra systému

Jádro Linuxu sestává z několika důležitých subsystémů. Jsou to části řízení procesů, správy paměti, ovladačů technických prostředků, ovladačů souborových systémů, správy sítě a různé další kusy a kousky. Některé z nich jsou zobrazeny na obrázku 2.1.



Obrázek 2.1 – Důležité části jádra Linuxu

Snad nejdůležitějšími subsystémy (bez nichž nic jiného nefunguje) jsou správce paměti a správce procesů. Subsystém správy paměti zajišťuje přidělování paměťových oblastí a odkládacího prostoru jednotlivým procesům, částem jádra a vyrovnávací paměti. Subsystém správy procesů vytváří procesy a přepínáním mezi aktivními procesy, které využívají procesor, zabezpečuje multitasking.

Jádro systému na nejnižší úrovni obsahuje ovladače pro všechny druhy technických zařízení, které operační systém podporuje. Vzhledem k tomu, že na světě existuje celá řada různých typů hardwaru, je počet ovladačů zařízení velký. Je ale mnoho jinak podobných zařízení, která se často liší pouze v tom, jak spolupracují s programy. Takovéto podobnosti umožňují definovat obecné třídy ovladačů, jež podporují podobné operace. Každý člen takovéto třídy má stejné rozhraní k ostatním částem jádra. Liší se v tom, jak tyto operace implementuje. Například všechny ovladače disků vypadají pro zbytek jádra podobně. To znamená, že všechny znají operace jako „inicializuj diskovou jednotku“, „čti sektor N“ a „zapiš sektor N“.

Některé softwarové služby, jež poskytuje jádro samotné, mají rovněž podobné vlastnosti. Proto mohou být také rozdělené do tříd. Kupříkladu různé síťové protokoly byly vyčleněny do jednoho programového rozhraní – knihovny „BSD socket library“. Dalším příkladem je vrstva *virtuálního souborového systému* (VFS). Ta odděluje operace souborového systému od jejich implementace. Každý typ souborového systému obstarává implementaci určité množiny operací, společné všem systémům souborů. Když se některý z prvků systému pokouší využít určitý souborový systém, žádost jde přes VFS. Ten ji směruje k požadovanému ovladači konkrétního systému souborů.

Nejdůležitější služby v unixovém systému

Tato část popisuje některé významnější služby systému Unix, avšak opět bez větších podrobností. Všechny služby budou později podrobně vysvětleny v dalších kapitolách.

Proces init

Nejdůležitější služby v systému Unix poskytuje **Proces init**. Spuštění **Procesu init** jako prvního z procesů je v každém unixovém systému posledním krokem, který provede jádro systému při zavádění. Po spuštění **Proces init** pokračuje v proceduře zavádění systému. Vykonává různé úkoly, které se při spouštění systému obvykle provádí (kontroluje a připojuje souborové systémy, spouští démony a tak dále).

Přesný seznam úloh, které **Proces init** při zavádění dělá, závisí na verzi tohoto programu i operačního systému. **Proces init** často obstarává takzvaný *jednouživatelský režim*. V jednouživatelském režimu se do systému nemůže nikdo přihlásit a příkazový interpret může z konzoly používat pouze root. Běžným režimem práce je *víceuživatelský režim*. Tyto režimy práce některé systémy Unix zobecňují do takzvaných *úrovní běhu*. Jednouživatelský a víceuživatelský režim tak představují dvě různé úrovně, na kterých může systém běžet. Kromě nich mohou existovat i další, například úroveň, při které se na konzole spustí grafické rozhraní X Window a podobně.

Linux povoluje až 10 úrovní běhu, 0 – 9, standardně však bývají definovány jen některé. Úroveň 0 představuje zastavení systému. Úroveň 1 je jednouživatelský režim. Úroveň 6 je restart systému. Chování dalších úrovní závisí na tom, jak je definuje vámi používaná distribuce a v různých distribucích se výrazně liší. Podíváte-li se na soubor `/etc/inittab`, může vám napovědět, jaké jsou předdefinované úrovně a jak jsou nastaveny.

V běžné situaci **Proces init** kontroluje, zda fungují procesy **getty** (umožňující uživatelům připojit se do systému) a adoptuje procesy – sirotky. Sirotci jsou procesy, jejichž rodičovské procesy byly z různých důvodů ukončeny – říká se, že umřely. V systému Unix *musí* být *všechny* procesy součástí jediné hierarchické stromové struktury. Proto musí **Proces init** sirotky adoptovat.

Když se systém vypíná, **Proces init** zodpovídá za ukončení všech ostatních procesů, odpojení všech souborových systémů, zastavení procesoru a za vše ostatní, co má podle dané konfigurace udělat.

Přihlášení z terminálů

Přihlášení uživatelů prostřednictvím terminálů (připojených na sériové linky) a konzoly (v případě, že neběží X Window) obstarává program **getty**. **Proces init** spouští zvláštní instanci **getty** pro každý terminál, ze kterého se bude možno do systému přihlásit. Program **getty** dále čte zadávané uživatelské jméno a spouští program **login**, jenž čte přístupové heslo. Jestli jsou uživatelské jméno a heslo správné, spustí program **login** příkazový interpret neboli *shell*. Když je příkazový interpret ukončen – jakmile se uživatel odhlásí ze systému, nebo když je program **login** ukončen proto, že nesouhlasí uživatelské jméno a heslo – **Proces init** to zjistí a spustí pro daný terminál novou instanci programu **getty**. Samotné jádro systému nemá vůbec přehled o přihlašování uživatelů do systému. Všechno kolem toho obstarávají systémové programy.

Syslog

Jádro systému i mnoho systémových programů hlásí různé chyby, vypisuje varování a jiná hlášení. Velmi často je důležité, aby bylo možno tyto zprávy prohlížet později, dokonce i s velkým časovým odstupem. Je tedy vhodné je zapisovat do nějakých souborů. Program, který to má na starosti, se jmenuje **syslog**. Lze jej nastavit tak, aby třídil zprávy a hlášení do různých souborů, a to podle původce, případně stupně významnosti. Hlášení jádra systému jsou obvykle směrována do jiného souboru než hlášení jiných procesů a programů. Jsou většinou významnější a je potřeba číst je pravidelně, aby bylo možné rozeznat případné problémy v zárodku.

Periodické vykonávání příkazů: cron a at

Uživatelé i správci systému často potřebují spouštět některé programy pravidelně. Například administrátor systému, který musí sledovat zaplněnost disku, by mohl chtít pravidelně spouštět příkaz, jenž by „vyčistil“ adresáře dočasných souborů (`/tmp` a `/var/tmp`). Program by odstranil starší dočasné soubory, které po sobě programy z různých důvodů korektně nesmazaly.

Takovéto služby nabízí program **cron**. Každý uživatel má vlastní soubor `crontab`, jenž obsahuje seznam příkazů, které chce vlastník spustit, a časy, kdy se mají tyto příkazy provést. Démon **cron** má na starosti spouštění těchto příkazů v požadovaném čase.

Služba **at** je podobná službě **cron**, provede se ale jenom jednou. Příkaz je vykonán v určeném čase, ale jeho spouštění se neopakuje.

Podrobnější informace viz manuálové stránky cron(1), crontab(1), crontab(5), at(1) a atd(8).

Grafické uživatelské rozhraní

Unix a Linux nezačleňují uživatelská rozhraní do jádra systému. Místo toho je implementují pomocí programů uživatelské úrovně. To se týká jak textového módu, tak grafického uživatelského prostředí.

Díky takovémuto řešení je samotný systém flexibilnější. Má to ale nevýhodu v tom, že je na druhou stranu velmi jednoduché implementovat pro každý program různá uživatelská rozhraní. Důsledkem je, že se takovýto systém uživatelé pomaleji učí.

Grafické prostředí, které Linux používá primárně, se nazývá „X Window System“ (zkráceně X). Ale ani X přímo neimplementuje uživatelské rozhraní. X Window pouze zavádí systém oken, tedy sadu nástrojů, pomocí kterých může být grafické uživatelské rozhraní implementované. Nad tímto systémem pracují správci oken, populární jsou například fvwm, icewm, blackbox a windowmaker. Dále existují dva rozšíření správci pracovního prostředí – Gnome a KDE.

Komunikace prostřednictvím počítačové sítě

Komunikace pomocí počítačové sítě je propojení dvou nebo více počítačů tak, že mohou komunikovat navzájem každý s každým. V současnosti používané metody propojování a komunikace jsou docela komplikované, ale výsledný efekt stojí za to.

Operační systémy Unix mají řadu síťových funkcí. Většinu základních služeb – služby souborových systémů, tisky, zálohování a podobně, lze využívat i prostřednictvím sítě. To ulehčuje správu systému a umožňuje centralizovanou administraci. Zachovávají se výhody mikropočítačové technologie i přínos distribuovaných systémů (nižší náklady a lepší odolnost vůči poruchám).

Tato kniha se komunikací prostřednictvím počítačové sítě zabývá jenom zběžně. Podrobnosti o této problematice, včetně základního popisu principů počítačových sítí, přináší *Příručka správce sítě*.

Přihlášení do systému ze sítě

Přihlášení do systému ze sítě funguje trochu odlišně než běžné přihlášení přes terminál. Pro každý terminál, prostřednictvím kterého je možné se přihlásit, je vyhrazená samostatná fyzická sériová linka. Pro každého uživatele, který se přihlašuje prostřednictvím sítě, existuje jedno samostatné virtuální síťové spojení, nicméně těchto spojení může být velký počet[4]. Proto není možné, aby běžely samostatné procesy **getty** pro všechna možná virtuální spojení. Kromě toho existuje několik různých způsobů přihlášení prostřednictvím sítě. Dva nejdůležitější způsoby v sítích TCP/IP jsou **telnet** a **rlogin**[5].

Síťová přihlášení mají místo řady procesů **getty** jednoho démona pro každý ze způsobů připojení (**telnet** a **rlogin** mají každý vlastního démona). Tento démon vyřizuje všechny přicházející žádosti o přihlášení. Dostane-li takovouto žádost, spustí svou novou instanci. Nová instance pak obsluhuje tuto jedinou žádost a původní instance nadále sleduje další příchozí žádosti o přihlášení. Nová instance pracuje podobně jako program **getty**.

Síťové souborové systémy

Jednou z nejužitečnějších věcí, kterou lze využít díky síťovým službám, je sdílení souborů pomocí *síťového souborového systému*. Jeden z nejběžněji používaných typů se nazývá Network File System, zkráceně NFS, a byl vyvinut společností Sun.

V síťovém souborovém systému jsou všechny operace se soubory, které dělá program na jednom počítači, odesílány prostřednictvím počítačové sítě na jiný počítač. Pro program, který běží na lokálním počítači, vzniká iluze, že soubory, které se nachází na vzdáleném počítači, jsou ve skutečnosti umístěny na počítači, na němž tento program běží. Takovýmto způsobem je velmi jednoduché sdílet informace, navíc lze používat již existující programy bez toho, že by je bylo potřeba měnit.

Další oblíbenou metodu sdílení souborů představuje Samba, http://www.samba.org. Tento systém umožňuje sdílet soubory i s Microsoft Windows (prostřednictvím „Okolních počítačů), a nabízí i sdílení tiskáren.

---

[4] Přinejmenším jich může být hodně. Propustnost sítě je stále choulostivý rys, takže existují praktické limity maximálního počtu síťových připojení.

[5] V současnosti považuje většina správců programy **telnet** a **rlogin** za nepříliš bezpečné a upřednostňují **ssh** (secure shell), který přenášená data šifruje a znesnadňuje tak odposlechnutí citlivých informací, jako jmen a hesel. Rozhodně doporučujeme tento program použít.

Pošta

Elektronická pošta je obvykle tím nejdůležitějším způsobem počítačové komunikace. Elektronický dopis je uložený v textovém souboru se zvláštním formátem. K jeho odeslání nebo přečtení se používají speciální programy.

Každý uživatel systému má vlastní *schránku na příchozí poštu*. Je to soubor určitého formátu, ve kterém jsou uloženy všechny nově příchozí zprávy. Když někdo odesílá poštu, program zjistí adresu poštovní schránky příjemce a připojí dopis k jeho souboru s příchozí poštou. Jestli je schránka příjemce na jiném počítači, je dopis odeslán na tento stroj a ten se bude snažit doručit jej do schránky příjemce.

Systém elektronické pošty se skládá z několika typů programů. Doručení pošty do místních nebo vzdálených poštovních schránek má na starosti první z nich – *agent pro přenos pošty* (MTA), například **sendmail** nebo **smail**. Uživatelé používají ke čtení pošty množství různých programů, takzvaných *uživatelských poštovních agentů* (MUA), například **pine**, **mutt** nebo **elm**. Poštovní schránky uživatelů jsou obvykle uloženy v adresáři `/var/spool/mail`.

Tisk

Tiskárnu může současně využívat pouze jeden uživatel. Nesdílet tiskárny mezi uživateli je ale dost neekonomické. Tiskárnu proto řídí program, jenž realizuje takzvanou *tiskovou frontu*. Všechny tiskové úlohy všech uživatelů systému jsou zařazeny do fronty. Hned, jak tiskárna ukončí jednu úlohu, automaticky se jí odesílá další v pořadí. Uživatelé si nemusí zabezpečovat frontu požadavků na tisk organizačně a odpadá i nutnost soupeřit a handrkovat se o přístup k tiskárně[6].

Program pro obsluhu tiskové fronty navíc *ukládá* (takzvaný *spool*) všechny tiskové výstupy na disk, takže pokud je tisková úloha ve frontě, je text uložen v nějakém souboru. Tento mechanismus aplikačním programům umožňuje rychle odeslat tiskové úlohy programu, jenž tiskovou frontu obsluhuje. Aplikace sama tak může pokračovat ve své práci. Nemusí čekat, než se úloha, která se právě tiskne, ukončí. To je v mnoha případech skutečně výhodné. Umožní vám to například zahájit tisk jedné verze dokumentu, přičemž nemusíte čekat, než se tisk ukončí, a můžete mezitím pracovat na nové, zcela pozměněné verzi.

Organizace systému souborů

Souborový systém je rozdělen na mnoho částí. Obvykle jsou hierarchicky uspořádané a nejvýše stojí kořenový souborový systém root. Souborový systém root obsahuje adresáře `/bin`, `/lib`, `/etc`, `/dev` a několik dalších; souborový systém `/usr`, který obsahuje programy a data, jež se nemění. Souborový systém `/var` obsahuje data, jež se naopak často mění (například logovací soubory). Posledním je souborový systém `/home`. Svá data a soubory si v něm ukládají uživatelé systému. Rozdělení souborového systému na jednotlivé svazky může být jiné. Závisí zejména na hardwarové konfiguraci systému a rozhodnutích jeho správce. Všechna data a soubory mohou být nakonec uloženy i v jediném systému souborů.

O něco podrobněji je organizace souborového systému popsána v kapitole 4, velmi podrobně ji definuje Filesystem Hierarchy Standard, http://www.pathname.com/fhs.

> Některé další detaily týkající se uspořádání systému souborů popisuje kapitola 3. Ještě více podrobností o tomto tématu přináší dokument Linux Filesystem Standard.

[6] Místo toho vytvoří novou frontu *u* tiskárny, kde čekají, až se jejich úloha vytiskne, protože ještě nikdo nedokázal napsat tiskový systém, který by přesně věděl, kdy co vytiskne. Pro mezilidské vztahy na pracovišti to představuje jednoznačný přínos.

Přehled adresářové struktury

Dva dny nato seděl Pú na své větvi, pohupoval nohama a tam,
vedle něj, stály čtyři hrníčky medu. . .
A. A. Milne

Kapitola popisuje důležité části standardní struktury adresářů operačního systému Linux, která je založená na standardu Filesystem Hierarchy Standard. Načrtneme v ní také běžný způsob rozdělení struktury adresářů do samostatných souborových systémů (svazků) s odlišnými účely a tento způsob rozdělení zdůvodníme. Ne všechny distribuce Linuxu tento standard dodržují, je však dostatečně univerzální, aby stačil pro základní přehled.

Pozadí

Tato kapitola volně vychází z normy Filesystem Hierarchy Standard (FHS)[7] verze 2.1, která je pokusem zavést jisté konvence do organizace adresářového stromu operačního systému Linux[8]. Výhodou přijetí takovéto normy je, že když bude vše na svém obvyklém místě, bude jednodušší psát programy a přenášet na Linux software z jiných platforem. Zároveň to ulehčí správu počítačů, na kterých běží operační systém Linux. I když neexistuje autorita, která by vývojáře, programátory a distributory donutila přizpůsobit se této normě, je její podpora v současnosti součástí většiny (ne -li všech) distribucí Linuxu. Není vhodné se neřídit standardem FHS, nejsou-li pro to velmi závažné důvody. Norma FHS se snaží sledovat tradice Unixu i současné trendy jeho vývoje. Motivací je snaha o usnadnění přechodu na Linux pro uživatele, kteří mají zkušenosti s jinými systémy Unix a naopak.

Tato kapitola není tak detailní jako samotná norma FHS. Správce systému – chce-li plně proniknout do problematiky systémů souborů – by si měl přečíst i normu FHS.

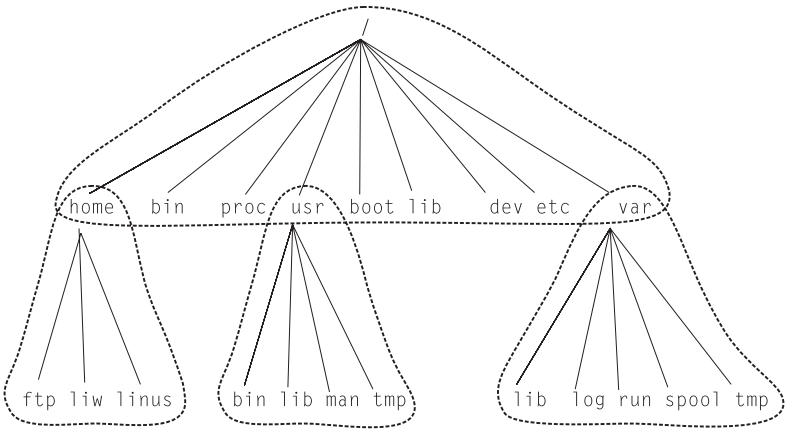
Kapitola rovněž nepopisuje podrobnosti týkající se všech typů souborových systémů. Nebylo také cílem popsat všechny adresáře a soubory, ale nabídnout čtenáři přehled o celém systému z perspektivy systému souborů. Podrobnější informace o popisovaných souborech jsou k dispozici na jiných místech této knihy, případně na manuálových stránkách.

Celou stromovou strukturu adresářů je možné rozdělit na menší části, každá z těchto částí může být umístěna na vlastním disku nebo samostatné diskové oblasti. Tak se lze jednoduše přizpůsobit omezením velikosti disků a zároveň usnadnit zálohování i ostatní úkoly spojené se správou systému. Nejdůležitější z těchto částí jsou kořenový souborový systém a dále souborové systémy `/usr`, `/var` a `/home` (viz obrázek 3.1). Každý systém souborů má jiné určení. Adresářová stromo-

[7] http://www.pathname.com/fhs

[8] A všech dalších unixových systémů.

vá struktura byla navržena tak, aby fungovala i v síti počítačů s operačním systémem Linux. Uživatelé a programy tak mohou pomocí sítě sdílet některé části systémů souborů, a to buď prostřednictvím zařízení určených pouze pro čtení (například CD-ROM), nebo pomocí sítě se systémem NFS.



Obrázek 3.1 – Části adresářové struktury Unixu. Přerušované čáry označují hranice diskových oblastí

Smysl jednotlivých částí adresářové struktury je popsán v dalším textu.

- Kořenový souborový systém je specifický pro každý počítač. Obecně je uložen na lokálním disku (avšak může to být i virtuální disk v paměti RAM – takzvaný „ramdisk“, nebo síťová disková jednotka). Kořenový svazek obsahuje soubory nutné pro zavedení systému a jeho uvedení do stavu, ve kterém mohou být připojeny ostatní souborové systémy. Obsah kořenového souborového systému postačuje pro práci v jednouživatelském režimu. Na tomto svazku jsou rovněž uloženy nástroje pro opravy poškozeného souborového systému a pro obnovení ztracených souborů ze záloh.
- Souborový systém `/usr` obsahuje všechny příkazy, knihovny, manuálové stránky a jiné soubory, jejichž obsah se nemění a které uživatel potřebuje při běžném provozu. Žádný ze souborů na tomto svazku by neměl být specifický pro daný počítač. Rovněž by se neměl při normálním provozu měnit. Tyto podmínky zaručují, že soubory uložené v souborovém systému `/usr` bude možné efektivně sdílet v síti. Sdílení tohoto svazku je výhodné jak z hlediska nákladů – šetří se tím místo na disku (v souborovém systému `/usr` mohou být uloženy stovky megabajtů dat), tak z hlediska usnadnění správy systému (například při instalaci novější verze aplikace se pak mění pouze systém `/usr` na hostitelském počítači a ne na každé stanici zvlášť). Je-li souborový systém `/usr` na lokálním disku, může být připojen pouze pro čtení. To snižuje pravděpodobnost poškození systému souborů při havárii systému.
- Souborový systém `/var` obsahuje soubory, které se v čase mění. Tedy především sdílené adresáře pro elektronickou poštu, systém news, tiskárny, logovací soubory, formátované manuálové stránky a dočasné soubory. Historicky bývaly všechny soubory, které jsou nyní uloženy v systému `/var`, v souborovém systému `/usr`. To však znemožňovalo připojit svazek `/usr` pouze pro čtení.
- Souborový systém `/home` obsahuje domovské adresáře uživatelů, tedy všechna „reálná data“. Vyčlenění uživatelských domovských adresářů do vlastní adresářové stromové struktu-

ry nebo samostatného souborového systému ulehčuje zálohování. Ostatní části adresářového stromu totiž buď nevyžadují zálohování vůbec, nebo – vzhledem k tomu, že se nemění tak často – se zálohují jenom zřídka. Velký souborový systém `/home` je dobré rozdělit na několik menších částí hierarchicky nižší úrovně a rozlišit je jménem, například `/home/students` a `/home/staff`.

Různé části, na které je hierarchická adresářová struktura rozčleněna, byly v našem přehledu označeny jako souborové systémy. Není ale žádný zvláštní důvod k tomu, aby ve skutečnosti ležely na samostatných oddělených svazcích. Všechny by mohly být nakonec i v jediném souborovém systému. Takové řešení má význam hlavně pro malé jednouživatelské systémy, kdy je prioritou jednoduchost. Celá stromová adresářová struktura může být rozdělena na souborové systémy i jinak. Způsob jejího rozčlenění závisí na tom, jak velký je disk a jak velký diskový prostor bude vyhrazen pro různé účely. Jedinou věcí, na kterou je potřeba dbát, jsou standardní unixová *jména*. Ta je potřeba zachovat. I když bude adresář `/var` a `/usr` ve stejné diskové oblasti, musí být zachována standardní jména, jako např. `/usr/lib/libc.a` nebo `/var/log/messages`. Totéž platí i v případě, že se například přesune adresář `/var` do adresáře `/usr/var` a na původním místě přesunutého adresáře bude symbolický odkaz na adresář `/usr/var`.

Struktura souborového systému Unixu sdružuje soubory podle jejich účelu, tedy všechny příkazy na jednom místě, data na jiném, dokumentace na dalším a tak dál. Alternativou by bylo sdružovat soubory podle toho, ke kterému programu patří. Pak by mohly být například všechny soubory pro program Emacs v jednom adresáři, všechny soubory pro TEX v jiném a podobně. Problém druhého přístupu je v tom, že je velmi obtížné sdílet soubory (adresář určitého programu často obsahuje jak statické soubory, které lze sdílet, tak soubory, jejichž obsah se mění, a ty sdílet nelze). Rovněž by bylo velmi složité dohledávat v rámci celého systému soubory určitého typu, například manuálové stránky aplikací uložené na mnoha různých místech. Programátory by jistě strašila noční můra – jak v takovémto případě vytvořit programy, které by byly schopné v případě potřeby manuálové stránky všech aplikací nalézt.

Souborový systém `root`

Kořenový svazek root by obecně měl být malý, protože obsahuje velmi kritické soubory. U malého souborového systému, který se mění jenom zřídka, je menší pravděpodobnost poškození. Poškození souborového systému root většinou znamená, že operační systém nebude možné zavést. Tento problém lze řešit pouze pomocí speciálních opatření (například zavedením systému z diskety), a to by chtěl riskovat asi málokdo.

Obecně by kořenový adresář neměl obsahovat žádné soubory, snad kromě standardního obrazu systému. Ten se obvykle jmenuje `/vmlinuz`. Všechny ostatní soubory by měly být uloženy v podadresářích kořenového adresáře, obvykle tímto způsobem:

`/bin`	Příkazy potřebné při zavádění systému, které mohou použít i normální uživatelé.
`/sbin`	Stejně jako `/bin`, příkazy ale nejsou určeny pro normální uživatele, i když i ti je mohou použít, je-li to nutné a je-li to povoleno. Tento adresář se obvykle nenachází v cestě normálních uživatelů, má jej ale v cestě root.
`/etc`	Konfigurační soubory specifické pro daný počítač.
`/root`	Domovský adresář superuživatele, obvykle nepřístupný ostatním uživatelům.
`/lib`	Sdílené knihovny pro programy v kořenovém souborovém systému.

`/lib/modules`	Zaveditelné moduly jádra systému – zvláště ty, které jsou potřeba pro zavedení systému při zotavení po havárii (například síťové ovladače a ovladače pro souborový systém).
`/dev`	Soubory zařízení. O některých nejběžnějších souborech zařízení hovoříme v kapitole 5.
`/tmp`	Dočasné soubory. Programy, které se spouští až po zavedení systému, by měly používat místo adresáře `/tmp` adresář `/var/tmp`, protože je velmi pravděpodobné, že leží na větším disku.
`/boot`	Soubory, jež používá zavaděč operačního systému, například LILO. Často se zde ukládají obrazy jádra (místo v kořenovém adresáři). V případě, že jich máte víc, může obsah adresáře `/boot` značně narůst a pak bude lepší mít jej v samostatném souborovém systému. Tím se také zajistí, že obrazy jádra budou uloženy na prvních 1 024 cylindrech disku IDE[9].
`/mnt`	Přípojné místo pro dočasná připojení dalších systémů souborů správcem systému. Nepředpokládá se, že by tento adresář využívaly pro automatická připojení souborových systémů programy. Adresář `/mnt` může být rozdělen na podadresáře (například `/mnt/dosa` pro disketovou mechaniku používanou v souborovém systému MS-DOS a `/mnt/exta` pro tutéž mechaniku využívanou se souborovým systémem ext2 a podobně).
`/proc`, `/usr`, `/var`, `/home`	Přípojná místa pro další souborové systémy[10].

Adresář `/etc`

Adresář `/etc` obsahuje mnoho souborů. Některé z nich jsou popsány v dalším textu. Pokud jde o ostatní, měli byste nejdřív zjistit, ke kterému programu patří, a pak si přečíst manuálové stránky k tomuto programu. V adresáři `/etc` je uloženo také hodně síťových konfiguračních souborů, které jsou popsány v *Příručce správce sítě*.

`/etc/rc` nebo `/etc/rc.d` nebo `/etc/rc?.d`

Skripty nebo adresáře skriptů, které se spouští při startu, nebo v případě, že se mění úroveň běhu systému. Podrobnosti viz kapitola 9.

`/etc/passwd`

Databáze uživatelů systému s položkami, v nichž je uloženo uživatelské jméno i skutečné jméno uživatele, domovský adresář, šifrované heslo a některé další informace. Formát je popsán v manuálové stránce programu **passwd**. V moderních systémech se zašifrovaná hesla obvykle nacházejí v souboru `/etc/shadow`. V souboru `passwd` se tedy nacházejí kdejaké informace *vyjma* hesel – z historických a praktických důvodů ale není přejmenování žádoucí.

`/etc/fdprm`

Tabulka parametrů disketové jednotky. Popisuje, jak vypadají různé formáty disket. Používá ji program **setfdprm**. Víc informací uvádí manuálová stránka programu **setfdprm**.

`/etc/fstab`

Seznamy souborových systémů připojovaných automaticky při startu příkazem **mount -a** (ve skript `/etc/rc`, nebo nějakém ekvivalentu). V systému Linux obsahuje rovněž informace o od-

[9] Omezení na prvních 1 024 cylindrů dnes už většinou neplatí. Moderní BIOSy a nové verze LILO umožňují tento limit obejít prostřednictvím logického adresování bloků (LBA). Viz manuálová stránka programu **lilo**.

[10] Konkrétně systém `/proc` ovšem fyzicky na disku neexistuje, viz dále část věnovaná tomuto souborovému systému.

kládacích oblastech, které používá příkaz **swapon -a**. Podrobnější informace viz část 6.8.5 a manuálové stránky příkazu **mount**. Samotný soubor `fstab` má typicky svou vlastní manuálovou stránku v sekci 5.

`/etc/group`

Soubor podobný souboru `/etc/passwd`, ale místo uživatelů popisuje pracovní skupiny. Podrobnější informace viz manuálová stránka **group**.

`/etc/inittab`

Konfigurační soubor **Procesu init**.

`/etc/issue`

Soubor obsahuje výstup programu **getty**, který se zobrazí před výzvou pro přihlášení uživatele. Obvykle obsahuje stručný popis systému nebo uvítací hlášení. Obsah určuje správce systému.

`/etc/magic`

Konfigurační soubor programu **file**. Obsahuje popisy různých formátů souborů, podle kterých pak program **file** tyto typy rozpoznává. Více informací najdete v manuálových stránkách pro **magic** a **file**.

`/etc/motd`

Takzvaná *zpráva pro tento den* – automatický výstup na terminál uživatele po úspěšném přihlášení do systému. Obsah volí správce systému. Často se využívá pro předávání informací (například upozornění na plánovaná zastavení systému apod.) všem uživatelům systému.

`/etc/mtab`

Seznam aktuálně připojených souborových systémů. Jeho obsah po zavedení systému a připojení určených souborových systémů prvotně nastavují inicializační skripty, v běžném provozu pak automaticky příkaz **mount**. Používá se v případech, kdy je potřeba zjistit, které souborové systémy jsou připojené, například při zadání příkazu **df**.

`/etc/shadow`

Soubor stínových hesel uživatelů v systémech, které mají nainstalovanou podporu stínových hesel. Kódovaná stínová hesla jsou z bezpečnostních důvodů přenesena ze souboru `/etc/passwd` do souboru `/etc/shadow`, který může číst pouze superuživatel. Snižuje se tak pravděpodobnost odhalení některého z přístupových hesel. Pokud vám systém umožňuje tento soubor použít (většina to umožňuje), pak to rozhodně udělejte.

`/etc/login.defs`

Konfigurační soubor příkazu **login**. Obvykle bývá popsán v sekci 5 manuálu.

`/etc/printcap`

Podobně jako u souboru `/etc/termcap`, až na to, že soubor je určený pro tiskárny. Odlišná je i jeho syntaxe. Bývá popsán v sekci 5 manuálu.

`/etc/profile, /etc/csh.login, /etc/csh.cshrc`

Soubory spouštěné při přihlášení uživatele nebo při startu systému interprety příkazů Bourne shell nebo C shell. Umožňují správci systému stanovit globální nastavení stejná pro všechny uživatele. Viz manuálové stránky k příslušným interpretům příkazů.

`/etc/securitty`

Soubor identifikuje zabezpečené terminály, tedy terminály, ze kterých se může přihlašovat superuživatel. Typicky jsou v seznamu uvedeny pouze virtuální konzoly, takže je nemožné (nebo přinejmenším těžší) získat oprávnění superuživatele přihlášením se po modemu nebo ze sítě. Nepovolujte přihlášení superuživatele přes síť. Rozumnější je přihlásit se jako běžný uživatel, a pak získat práva superuživatele příkazem **su** nebo **sudo**.

`/etc/shells`

Soubor, jenž uvádí seznam důvěryhodných interpretů příkazů. Příkaz **chsh** umožňuje uživatelům změnit příkazový interpret spuštěný při přihlášení, a to pouze na některý z interpretů uvedený v tomto souboru. Proces **ftpd**, jenž běží na hostitelském počítači a poskytuje služby FTP pro klientské počítače, rovněž kontroluje, zda je uživatelův příkazový interpret uveden v tomto souboru a nedovolí připojit se klientům, jejichž příkazový interpret v tomto seznamu uveden není.

`/etc/termcap`

Databáze vlastností terminálu. Popisuje, kterými escape sekvencemi se řídí různé typy terminálů. Každý program je napsaný tak, že místo přímého výstupu escape sekvence, jež by fungovala pouze s konkrétním typem terminálu, hledá v tabulce `/etc/termcap` sekvenci, která odpovídá tomu, co chce program na terminálu zobrazit. Pak může většina programů správně obsluhovat většinu typů terminálů. Více informací uvádí manuálové stránky pro `termcap`, `curs-termcap` a `terminfo`.

Adresář `/dev`

Adresář `/dev` obsahuje speciální soubory všech zařízení. Speciální soubory jsou pojmenované podle určitých konvencí, které popisujeme v kapitole 5. Speciální soubory se vytváří v průběhu instalace operačního systému, v běžném provozu pak skriptem `/dev/MAKEDEV`. Podobný je skript `/dev/MAKEDEV.local`. Ten upravuje a používá správce systému, když vytváří čistě lokální speciální soubory a odkazy. Lokální speciální soubory jsou ty, které nejsou vytvořeny standardním postupem pomocí skriptu `MAKEDEV`, typicky například speciální soubory pro některé nestandardní ovladače zařízení.

Souborový systém `/usr`

Souborový systém `/usr` je často dost velký, protože jsou v něm instalované všechny programy. Všechny soubory v systému `/usr` jsou obvykle instalované přímo z distribuce systému Linux. Všechny další lokálně instalované programy se ukládají do adresáře `/usr/local`. To umožňuje správci systému instalovat vyšší verze Linuxu z nové verze distribuce nebo i úplně jiné distribuce operačního systému bez toho, že by bylo potřeba současně instalovat všechny programy znovu. Některé z podadresářů adresáře `/usr` jsou popsány níže, ty méně významné neuvádíme. Více informací přináší popis standardu FHS.

`/usr/X11R6`

Všechny soubory systému X Window. Soubory pro X nejsou integrální součástí operačního systému z důvodů zjednodušení vývoje a instalace X. Adresářová struktura `/usr/X11R6` je podobná stromu, který je vytvořen pod adresářem `/usr` samotným.

`/usr/bin`

Zde se nachází téměř všechny uživatelské příkazy. Některé další příkazy jsou uloženy v adresáři `/bin` nebo `/usr/local/bin`.

`/usr/sbin`

Obsahuje ty příkazy pro správu systému, které nejsou potřeba přímo v kořenovém souborovém systému (například převážná většina serverových programů).

`/usr/share/man, /usr/share/info, /usr/share/doc`

Manuálové stránky, informační dokumenty GNU, případně různé jiné soubory s dokumentací.

`/usr/include`

Hlavičkové soubory pro programovací jazyk C. Z důvodů zachování konzistence by měly být spíš v adresáři `/usr/lib`, ale z historických důvodů jsou umístěny ve zvláštním adresáři.

`/usr/lib`

Datové soubory pro programy a subsystémy, které se nemění. Jsou zde rovněž uloženy některé globální konfigurační soubory. Jméno `lib` je odvozeno od anglického slova „library" (knihovna). Původně totiž byly v adresáři `/usr/lib` uloženy knihovny podprogramů.

`/usr/local`

Místo pro lokálně instalovaný software a další soubory. Originální distribuce by zde neměla nic instalovat, tento adresář slouží čistě správci systému. Díky tomu si může být jistý, že aktualizace distribuce mu nepřepíší žádné jím instalované programy.

Souborový systém `/var`

Systém `/var` obsahuje data, která se při běžném provozu systému mění. Soubory jsou specifické pro každý systém, a proto se data mezi jinými počítači v síti nesdílí.

`/var/cache/man`

Vyrovnávací paměť pro manuálové stránky, které jsou formátovány na požádání. Zdrojové texty manuálových stránek jsou obvykle uloženy v adresáři `/usr/share/man/man?` (kde `?` je příslušná sekce manuálu, viz manuál pro příkaz `man, sekce 7`). Některé stránky se dodávají v předem formátované verzi a jsou pak uloženy v adresáři `/usr/share/man/cat*`. Jiné stránky je třeba při prvním prohlížení naformátovat. Formátované verze jsou pak uloženy právě v adresáři `/var/cache/man`. Další uživatel, který si chce stejné stránky prohlížet, tak nemusí čekat na jejich opakované formátování.

`/var/games`

Jakákoliv proměnná data her instalovaných v adresáři `/usr` – pro případ, že by byl svazek `/usr` připojen jen pro čtení.

`/var/lib`

Soubory, které se při normálním provozu systému mění.

`/var/local`

Měnící se data pro programy instalované v adresáři `/usr/local` (tedy programy instalované správcem systému). Upozorňujeme, že lokálně instalované programy by měly podle potřeby používat i ostatní podadresáře nadřazeného adresáře `/var`, například `/var/lock`.

`/var/lock`

Soubory zámků. Většina programů dodržuje určitou konvenci a vytváří v adresáři `/var/lock` zámky. Tím dávají ostatním programům najevo, že dočasně využívají některé zařízení nebo soubor. Jiné programy, které by chtěly stejné zařízení či soubor ve stejném okamžiku používat, se o to nebudou pokoušet.

`/var/log`

Adresář obsahuje logovací soubory různých programů, zejména programu **login** (do souboru `/var/log/wtmp` se zaznamenávají všechna přihlášení a odhlášení uživatelů systému) a **syslog** (do souboru `/var/log/messages` ukládá všechna hlášení jádra systému a systémových programů). Velikost souborů v adresáři `/var/log` dost často nekontrolovaně roste, proto se musí v pravidelných intervalech mazat.

`/var/mail`

Standardem FHS navrhované umístění poštovních schránek. Podle toho, nakolik váš systém respektuje FHS, mohou být schránky stále umístěny ve `/var/spool/mail`.

`/var/run`

Adresář, do něhož se ukládají soubory obsahující informace o systému, jež platí až do jeho dalšího zavedení. Tak například soubor `/var/run/utmp` obsahuje informace o momentálně přihlášených uživatelích systému.

`/var/spool`

Adresáře pro elektronickou poštu, systém news, tiskové fronty a další subsystémy, které využívají metodu spoolingu a princip řazení úloh do fronty. Každý z těchto subsystémů má v tomto adresáři svůj vlastní podadresář, například news se ukládají do `/var/spool/news`. Pokud systém není plně kompatibilní s FHS, může ukládat poštovní schránky ve `/var/spool/mail`.

`/var/tmp`

Do adresáře `/var/tmp` se ukládají velké dočasné soubory a dočasné soubory, které budou existovat déle než ty, které se ukládají do adresáře `/tmp`. (Avšak správce systému by měl dbát na to, aby stejně jako v adresáři `/tmp`, ani v adresáři `/var/tmp` nebyly uloženy velmi staré dočasné soubory.)

Souborový systém `/proc`

Systém souborů `/proc` je vlastně imaginárním souborovým systémem. Ve skutečnosti na disku neexistuje. Místo toho jej v paměti vytváří jádro systému. Ze systému souborů `/proc` lze získávat různé aktuální informace o systému (původně o procesech – z toho je odvozeno jeho jméno). Některé z významnějších souborů a adresářů popisujeme níže. Samotný souborový systém `/proc` je podrobněji popsán na manuálové stránce *proc*.

`/proc/1`

Adresář s informacemi o procesu číslo 1. Každý z procesů má v adresáři `/proc` vlastní podadresář, jehož jméno je stejné jako identifikační číslo procesu.

`/proc/cpuinfo`

Různé informace o procesoru. Například typ, výrobce, model, výkon a podobně.

`/proc/devices`

Seznam ovladačů zařízení konfigurovaných pro aktuálně běžící jádro systému.

`/proc/dma`

Informuje o tom, které kanály DMA jsou právě využívány.

`/proc/filesystems`

Souborové systémy konfigurované v jádru systému.

`/proc/interrupts`

Informuje o tom, která přerušení jsou využívána a kolikrát nastala.

`/proc/ioports`

Informuje o tom, které ze vstupně-výstupních portů se momentálně využívají.

`/proc/kcore`

Obraz fyzické paměti systému. Má velikost odpovídající velikosti fyzické paměti systému. Ve skutečnosti ale samozřejmě nezabírá takovéto množství paměti, protože jde o soubor generovaný „na požádání“, tedy pokaždé jenom v okamžiku, kdy k němu různé programy přistupují. Uvědomte si, že soubory souborového systému `/proc` nezabírají ve skutečnosti (než je zkopírujete na nějaké jiné místo na disku) vůbec žádný diskový prostor.

`/proc/kmsg`

Výstupní hlášení jádra systému. Zde uložená hlášení dostává i program **syslog**.

`/proc/ksyms`

Tabulka symbolů jádra systému.

`/proc/loadavg`

Statistika zatížení systému – tři celkem nic neříkající indikátory toho, kolik práce systém momentálně má.

`/proc/meminfo`

Informace o využití paměti, jak fyzické, tak virtuální.

`/proc/modules`

Informuje o tom, které moduly jádra jsou právě zavedeny v paměti.

`/proc/net`

Informace o stavu síťových protokolů.

`/proc/self`

Symbolický odkaz do adresáře procesů toho programu, který zrovna přistupuje k souborovému systému `/proc`. Když k systému souborů `/proc` současně přistupují dva různé procesy, budou mít přidělené dva různé odkazy. Tímto způsobem se mohou programy pohodlně a jednoduše dostat k vlastnímu adresáři.

`/proc/stat`

Různé statistiky týkající se systému. Například počet výpadků stránek od zavedení systému a podobně.

`/proc/uptime`

Informuje o tom, jak dlouho systém běží.

`/proc/version`

Verze jádra systému.

Všechny výše uvedené soubory jsou normálně čitelné textové soubory, ne vždy jsou ale formátovány příliš přehledně. Existuje celá řada programů, které nedělají prakticky nic jiného, než že načtou některý z těchto souborů a zobrazí jej v přehlednějším formátu. Například program **free** přečte soubor `/proc/meminfo` a v něm uvedené údaje převede z bajtů na kilobajty (a ještě něco málo přidá).

Soubory zařízení

V této kapitole popisujeme, co jsou to soubory zařízení, a jak se vytvářejí. Uvádíme také některé běžné soubory zařízení. Kanonický seznam souborů zařízení najdete v souboru `/usr/src/linux/Documentation/devices.txt` v případě, že máte nainstalovány zdrojové kódy jádra. Dále uvedené informace platí pro jádro 2.2.17.

Skript MAKEDEV

Po instalaci operačního systému bude většina souborů zařízení již vytvořena a budou připraveny k použití. Pokud byste náhodou potřebovali nějaký, který neexistuje, vyzkoušejte nejprve skript **MAKEDEV**. Obvykle jej najdete jako `/dev/MAKEDEV`, případná kopie (nebo odkaz) může být i v `/sbin/MAKEDEV`. Pokud se nenachází ve spouštěcí cestě, budete muset cestu definovat explicitně.

Typicky se tento skript používá následujícím způsobem:

```
# /dev/MAKEDEV -v ttyS0
create ttyS0   c 4 64 root:dialout 0660
```

Tímto příkazem vytvoříte znakové zařízení `/dev/ttyS0` s hlavním číslem 4 a vedlejším číslem 64, jehož vlastníkem bude root, skupina dialout, a práva budou 0660.

Konkrétně `ttyS0` je sériový port. Podle hlavního a vedlejšího čísla zařízení se orientuje jádro. Jádro přistupuje k zařízením výhradně pomocí čísel, což by znesnadňovalo orientaci normálních smrtelníků – proto používáme soubory s určitými názvy. Práva 0660 znamenají právo pro čtení a zápis vlastníkovi (root) a skupině (dialout). Ostatní uživatelé žádná práva k souboru nemají.

Příkaz mknod

Jako preferovaný způsob vytváření neexistujících zařízení se doporučuje skript **MAKEDEV**. V některých případech ale tento skript nemusí znát zařízení, které chcete vytvořit. Pak použijete příkaz **mknod**. Abyste jej mohli použít, musíte znát hlavní a vedlejší číslo zařízení, které chcete vytvořit. Základní zdroj těchto informací představuje soubor `devices.txt` v dokumentaci zdrojových kódů jádra.

Předpokládejme pro ilustraci, že naše verze skriptu **MAKEDEV** neumí vytvořit zařízení `/dev/ttyS0`, proto je budeme vytvářet příkazem **mknod**. Z dokumentace k jádru se dozvíme, že má mít hlavní číslo 4 a vedlejší číslo 64. Teď už tedy víme všechno, abychom mohli zařízení vytvořit:

```
# mknod /dev/ttyS0 c 4 64
# chown root.dialout /dev/ttyS0
# chmod 0644 /dev/ttyS0
# ls -l /dev/ttyS0
crw-rw---- 1 root dialout 4, 64 Oct 23 18:23 /dev/ttyS0
```

Jak vidíte, v tomto případě je vytvoření zařízení o dost složitější. Ukázali jsme si nicméně celý potřebný postup. Je velmi nepravděpodobné, že byste někdy museli ručně vytvářet zařízení `/dev/ttyS0`, nicméně princip je stejný pro libovolná zařízení.

Seznam zařízení

Následující seznam není v žádném případě vyčerpávající ani podrobný. Řada následujících zařízení bude fungovat pouze v případě, že jádro obsahuje přeložené moduly pro jejich podporu. Podrobnosti o konkrétních zařízeních se dozvíte v dokumentaci jádra.

`/dev/dsp`

Digital Signal Processor. V zásadě jde o rozhraní mezi programy, produkujícími zvuk, a mezi zvukovou kartou. Jde o znakové zařízení s hlavním číslem 14 a vedlejším 3.

`/dev/fd0`

První disketová mechanika. Pokud máte to štěstí a vlastníte více disketových mechanik, budou číslovány sekvenčně. Jde o blokové zařízení s hlavním číslem 2 a vedlejším 0.

`/dev/fb0`

První framebuffer. Framebuffer je abstrakční vrstva mezi programy a grafickým hardwarem. Znamená to, že programy se nemusejí starat o to, jaký konkrétní grafický hardware používáte, stačí jim umět komunikovat s API framebufferu, které je dobře definováno a standardizováno. Jde o znakové zařízení s hlavním číslem 29 a vedlejším 0.

`/dev/hda`

Master disk na primárním IDE řadiči. Slave disk na primárním řadiči bude `/dev/hdb`. `/dev/hdc` a `/dev/hdd` jsou master a slave na sekundárním IDE řadiči. Každý disk je rozdělen na oddíly. Oddíly 1 – 4 jsou primární, oddíly 5 a výše jsou logické oddíly na rozšířených oddílech. Zařízení, popisující jednotlivé oddíly, jsou tedy pojmenována poněkud složitěji – například `/dev/hdc9` je 9. oddíl (logický oddíl na rozšířeném oddílu) master disku na sekundárním IDE rozhraní. Přidělení hlavních a vedlejších čísel je také trochu složitější. Na prvním IDE řadiči představují všechny oddíly zařízení s hlavním číslem 3. Master disk `hda` má vedlejší číslo 0, slave disk `hdb` vedlejší číslo 64. U každého oddílu se pak jeho číslo přičítá k vedlejšímu číslu zařízení, tedy například `/dev/hdb5` má hlavní číslo 3 a vedlejší 69 (64+5). Disky na sekundárním IDE se obsluhují stejně, hlavní číslo je ale 22.

`/dev/ht0`

První pásková IDE jednotka. Další jednotky se číslují `ht1` atd. Jsou to znaková zařízení s hlavním číslem 37 a vedlejším počínaje 0 pro `ht0`, 1 pro `ht1` atd.

`/dev/js0`

První analogový joystick. Další joysticky se číslují `js1`, `js2` atd. Digitální joysticky se označují `djs0`, `djs1` atd. Jde o znaková zařízení s hlavním číslem 13. Analogové joysticky pak mají vedlejší číslo 0 až 127 (což je dost i pro toho nejfanatičtějšího hráče). Digitální joysticky začínají vedlejším číslem 128.

`/dev/lp0`

První paralelní tiskárna. Další tiskárny jsou `lp1`, `lp2` atd. Jde o znaková zařízení s hlavním číslem 6 a vedlejším počínaje 0.

`/dev/loop0`

První loopback zařízení. Tato zařízení slouží k připojování souborových systémů, které nejsou umístěny na nějakém blokovém zařízení, například na disku. Pokud budete například chtít připojit obraz CD ve formátu iso9660 aniž byste jej napřed vypalovali na CD, můžete to udělat pomocí loopback zařízení. Obvykle je tento postup pro uživatele transparentní a provádí se příkazem **mount**. Podívejte se na manuálové stránky příkazů **mount** a **losetup**. Loopback zařízení jsou bloková zařízení s hlavním číslem 7 a vedlejším číslem počínaje 0.

`/dev/md0`

První skupina metadisků. Metadisky se týkají zařízení RAID (Redundant Array of Independent Disk). Podrobnosti naleznete v různých dokumentech LDP, které se vztahují k RAID. Metadisky jsou bloková zařízení s hlavním číslem 9 a vedlejším počínaje 0.

`/dev/mixer`

Součást zvukového ovladače OSS (Open Sound System). Podrobnosti viz dokumentace k OSS, http://www.opensound.com. Jde o znakové zařízení s hlavním číslem 14 a vedlejším 0.

`/dev/null`

Černá díra, kam můžete posílat data, která už nechcete nikdy vidět. Cokoliv zapsaného do `/dev/null` se ztratí. Toto zařízení může být užitečné, pokud například chcete spustit nějaký příkaz a nezajímá vás jeho výstup. Jde o znakové zařízení s hlavním číslem 1 a vedlejším 3.

`/dev/psaux`

Port myši PS/2. Znakové zařízení s hlavním číslem 10 a vedlejším 1.

`/dev/pda`

IDE disk na paralelním rozhraní. Pojmenovávají se podobně jako disky na interním IDE rozhraní (`/dev/hd*`). Jde o bloková zařízení s hlavním číslem 45. Vedlejší čísla si zasluhují trochu vysvětlení. První zařízení je `/dev/pda` a má vedlejší číslo 0. Oddíly na tomto disku získáte přičtením čísla oddílu k vedlejšímu číslu. Jednotlivá zařízení mohou obsahovat maximálně 15 oddílů (na rozdíl od 63 u interních IDE disků). Vedlejší čísla oddílů na `/dev/pdb` začínají od 16, na `/dev/pdc` od 32 a na `/dev/pdd` od 48. Vedlejší číslo zařízení `/dev/pdc6` tedy bude 38 (32+6). Tento mechanismus nás omezuje na maximálně 4 paralelní disky po patnácti oddílech.

`/dev/pcd0`

CD-ROM mechanika na paralelním rozhraní. Mechaniky se číslují od 0 nahoru. Všechny mají hlavní číslo 46, `/dev/pcd0` má vedlejší číslo 0, další mechaniky pak 1, 2 atd.

`/dev/pt0`

Pásková jednotka na paralelním rozhraní. Pásky nemají oddíly a číslují se tedy prostě sekvenčně. Jsou to znaková zařízení s hlavním číslem 96, vedlejší číslo je 0 pro `pt0`, 1 pro `pt1` atd.

`/dev/parport0`

Nízkoúrovňový paralelní port. Většina zařízení připojených na paralelní port používá vlastní ovladač. Toto zařízení slouží k přímému přístupu na paralelní port. Jde o znakové zařízení s hlavním číslem 99 a vedlejším číslem od 0 nahoru.

`/dev/random a /dev/urandom`

Generátory náhodných čísel. `/dev/random` je nedeterministický generátor, takže následující vygenerovanou hodnotu není možné odvodit z hodnot předchozích. Náhodná čísla se generují na základě hardwarové „entropie“ systému. Pokud momentálně není entropie systému dostatečná, zařízení se zablokuje a nedovolí přečíst další hodnotu, než se nasbírá dostatek entropie. Zařízení `/dev/urandom` funguje podobně. Standardně generuje náhodná čísla na základě entropie systému, pokud ale není dostatek entropie, použije algoritmus generování pseudonáhodných čísel. Tento přístup je považován za méně bezpečný pro významné funkce, například při generování kryptografických klíčů. Pokud vyžadujete vysokou bezpečnost, použijte zařízení `/dev/random`, pokud vám jde o rychlost, použijte `/dev/urandom`. Jde o znaková zařízení s hlavním číslem 1 a vedlejším 8 (`/dev/random`) a 9 (`/dev/urandom`).

`/dev/zero`

Generátor nul. Při každém čtení z tohoto zařízení dostanete nulu. Může to být užitečné například tehdy, chcete-li vytvořit soubor určité délky a nezáleží vám na jeho obsahu. Jde o znakové zařízení s hlavním číslem 1 a vedlejším číslem 5.

KAPITOLA 5

Disky a jiná média

Na prázdném disku lze hledat věčně.

Při instalaci a aktualizaci systému vás u disků čeká hodně práce. Je potřeba vytvořit souborové systémy, do kterých se budou ukládat soubory, a vyhradit na discích prostor pro různé části systému.

Tato kapitola popisuje všechny tyto úvodní činnosti. Když tuto práci jednou podstoupíte a systém nastavíte, obvykle to už nebudete muset dělat znovu. Výjimkou je používání disket. K této kapitole se také budete vracet pokaždé, když budete přidávat nový pevný disk nebo pokud budete chtít optimálně vyladit diskový subsystém.

Mezi základní úkoly při správě disků patří:

- Formátování pevného disku. Formátování disku je posloupností několika různých dílčích činností (jako je například kontrola výskytu vadných sektorů), které tento disk připravují na další použití. (V současnosti je většina nových pevných disků formátovaná výrobcem a jejich formátování není nutné.)
- Rozdělení pevného disku na oblasti. Když chcete disk využívat pro několik činností, o kterých se nepředpokládá, že by se vzájemně ovlivňovaly, můžete jej rozdělit na samostatné diskové oblasti. Jedním z důvodů pro rozdělení disku na oblasti je instalace a provozování různých operačních systémů na jednom disku. Jinou výhodou rozdělení pevného disku na oblasti je oddělení uživatelských souborů od souborů systémových. Tím se zjednoduší zálohování a sníží se pravděpodobnost poškození systémových souborů.
- Vytvoření souborového systému (vhodného typu) na každém disku nebo diskové oblasti. Z pohledu operačního systému nestačí disk pouze zapojit, Linux jej může používat až poté, co na něm vytvoříte nějaký souborový systém. Pak lze na disk ukládat soubory a přistupovat k nim.
- Vytvoření jediné stromové struktury připojením různých souborových systémů. Systémy souborů se připojují buď automaticky, nebo manuálně – podle potřeby. (Ručně připojené souborové systémy se obvykle musí také ručně odpojit.)

Kapitola 7 obsahuje informace o virtuální paměti a diskové vyrovnávací paměti. Pracujete-li s disky, měli byste být s jejich principy obeznámeni.

Dva druhy zařízení

Unix, a tedy i Linux, zná dva různé typy zařízení. Jednak bloková zařízení s náhodným přístupem (například disky) a jednak znaková zařízení (například pásky a sériové linky), která mohou být buď sériová, nebo s náhodným přístupem. Každému z podporovaných zařízení odpovídá v systé-

mu souborů jeden soubor zařízení. Když se čtou či zapisují data z anebo do souboru zařízení, přenáší se ve skutečnosti na zařízení, které tento soubor reprezentuje. Takže pro přístup k zařízením nejsou nutné žádné zvláštní programy a žádné zvláštní programovací techniky (jako například obsluha přerušení nebo cyklické dotazování sériového portu). Když například chcete vytisknout nějaký soubor na tiskárně, stačí zadat:

```
$ cat jméno_souboru > /dev/lp1
$
```

a obsah souboru se vytiskne (soubor musí mít přirozeně nějakou formu, kterou tiskárna zná). Avšak vzhledem k tomu, že není příliš rozumné, aby několik uživatelů současně kopírovalo soubory na jedinou tiskárnu, pro tiskové úlohy se běžně používá zvláštní program (nejčastěji `lpr`). Tento program zajistí, že se v určitém okamžiku bude tisknout pouze jeden soubor. Ihned po ukončení jedné tiskové úlohy automaticky pošle na tiskárnu další úlohu. Podobný mechanismus přístupu vyžaduje většina zařízení. Takže ve skutečnosti se běžný uživatel o soubory zařízení skoro vůbec nemusí zajímat.

Protože se zařízení chovají jako soubory souborového systému (uložené v adresáři `/dev`), lze velmi lehce zjistit, které soubory zařízení existují. Lze použít například příkaz `ls` nebo jiný vhodný program. V prvním sloupci výstupu příkazu `ls -l` je uveden typ souboru a jeho přístupová práva. Chcete-li si prohlédnout sériové zařízení systému, zadáte:

```
$ ls -l /dev/ttyS0
crw-rw-r-- 1 root dialout 4, 64 Aug 19 18:56 /dev/ttyS0
$
```

Podle prvního písmene prvního sloupce, tedy písmene „c“ v řetězci „crw-rw-r--“, informovaný uživatel pozná o jaký typ souboru jde. V tomto případě se jedná o znakové zařízení. U běžných souborů je prvním písmenem „-“, u adresářů je to „d“ a pro bloková zařízení se používá písmeno „b“. Podrobnější informace uvádí manuálová stránka k příkazu `ls`.

Všimněte si, že všechny speciální soubory obvykle existují, i když zařízení samotné není nainstalované. Takže například pouhý fakt, že v systému souborů je soubor `/dev/sda`, neznamená, že skutečně máte pevný disk SCSI. Díky tomu, že všechny speciální soubory po instalaci operačního systému existují, lze zjednodušit instalační programy. Méně složité je tím pádem i přidávání nových hardwarových komponent (pro nově přidávané součásti již není třeba hledat správné parametry a vytvářet speciální soubory).

Pevné disky

Tato část zavádí terminologii, která se v oblasti pevných disků používá. Znáte-li tyto pojmy a principy, můžete ji přeskočit.

Na obrázku 6.1 jsou schematicky znázorněny důležité části pevného disku. Disk se skládá z jedné nebo více kruhových *desek*[11], jejichž *povrchy* jsou pokryty magnetickou vrstvou, na kterou se zaznamenávají data. Každý povrch má svou *čtecí a zapisovací hlavu*, která čte nebo zaznamenává údaje. Desky rotují na společné hřídeli, typická rychlost otáčení je 5 400 nebo 7 200 otáček za minutu. (Pevné disky s vysokým výkonem používají i vyšší rychlosti.) Hlavy se pohybují podél roviny povrchu, spojením tohoto pohybu s rotací povrchu může hlava přistupovat ke všem částem magnetických povrchů.

Procesor a disk spolu komunikují prostřednictvím *řadiče disku*. Díky řadiči se zbytek systému nemusí zajímat o to, jak pracovat s diskem, protože lze použít pro různé typy disků řadiče, jež ma-

[11] Ty jsou vyrobeny z nějakého pevného materiálu, například hliníku. Odtud vzaly pevné disky svůj název.

jí pro ostatní součásti počítače stejné rozhraní. Takže počítač může disku (místo zadávání sérií dlouhých a složitých elektrických signálů, podle nichž se hlava nejdřív přesune na odpovídající místo disku, pak čeká, až se pod ni dostane žádaná pozice a dělá další nepříjemnosti, které jsou pro danou operaci potřeba) jednoduše vzkázat: „Hele, milý disku, dej mně to, co potřebuji". (Ve skutečnosti je sice rozhraní řadiče stejně dost složité, ale rozhodně ne tak složité, jako by bylo v případě, kdyby ostatní prvky systému přistupovaly k disku přímo.) Řadič navíc může dělat některé další operace, obsluhuje například vyrovnávací paměť, automaticky nahrazuje vadné sektory a podobně.

Výše uvedené obvykle každému postačí k pochopení toho, jak hardware pevného disku funguje. Existuje pochopitelně ještě řada dalších částí, například motory, které otáčí diskem a přemisťují hlavy, elektronika, jež řídí operace dalších mechanických částí a tak dále, jež ale většinou nejsou pro pochopení principu práce pevného disku tak důležité.

Magnetické vrstvy disku jsou obvykle rozděleny do soustředných kružnic, kterým se říká *stopy*. Stopy se dělí na *sektory*. Toto rozdělení se používá pro určování místa na disku a přidělování diskového prostoru souborům. Když například chcete na disku najít určité místo, zadáte „souřadnice": vrstva 3, stopa 5, sektor 7. Počet sektorů je většinou pro všechny stopy stejný, ale některé pevné disky mají na vnějších stopách víc sektorů (všechny sektory mají stejnou fyzickou velikost, takže většina z nich je umístěna na delších, vnějších stopách). Typicky bude v jednom sektoru uloženo 512 bajtů dat. Disk samotný pak neumí pracovat s menším množstvím dat, než je jeden sektor.

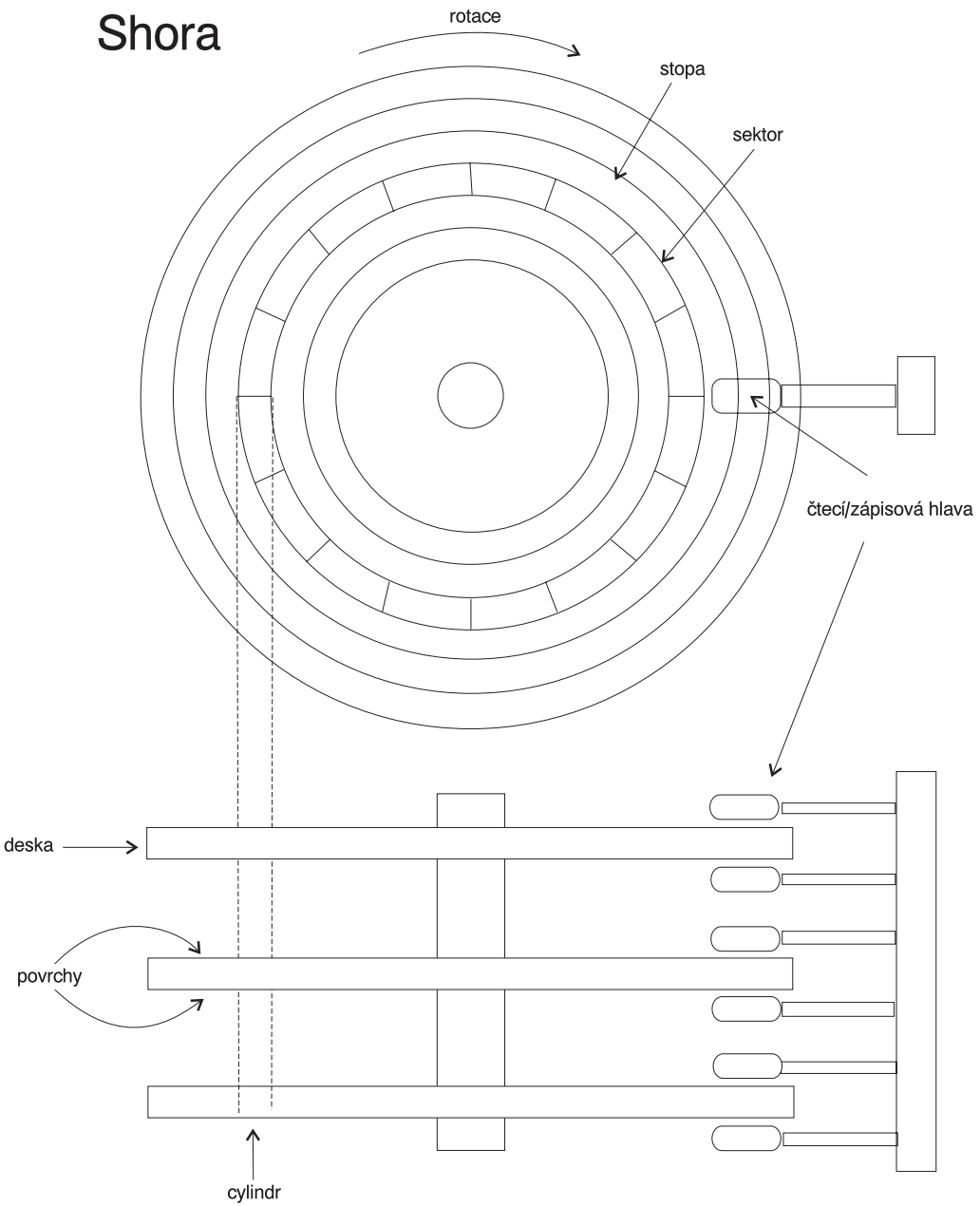
Každý povrch je rozdělen na stopy a sektory stejným způsobem. Takže když je hlava jednoho povrchu nad určitou stopou, hlavy ostatních povrchů se také nachází nad odpovídajícími stopami. Všem těmto stopám dohromady se říká *cylindr*. Přemístění hlavy z jedné stopy (cylindru) na jinou trvá jistou dobu. Takže ukládáním dat, ke kterým se často přistupuje najednou (například data jednoho souboru) na stejný cylindr, se zamezí zbytečnému přesouvání hlav při jejich pozdějším čtení. Snižuje se přístupová doba a zvyšuje výkon. Není ale vždycky možné uložit data na disk tímto způsobem. Souborům, které jsou na disku uloženy na několika místech, se říká *fragmentované*.

Počet povrchů (respektive hlav, což je to samé), cylindrů a sektorů se dost liší. Specifikace jejich počtu se nazývá *geometrií* pevného disku. Geometrie je obvykle uložena ve zvláštní bateriemi zálohované oblasti paměti, které se říká *CMOS RAM*, odkud si ji operační systém načítá vždy během zavádění nebo inicializace ovladače disku.

Naneštěstí má BIOS[12] omezení, které neumožňuje zadat v paměti CMOS počet stop větší než 1 024, což je pro pevné disky velké kapacity příliš málo. Uvedené omezení lze obejít tak, že řadič pevného disku bude lhát ohledně skutečné geometrie disku a bude *překládat adresy* požadované systémem na adresy, které odpovídají realitě. Představme si například pevný disk, jenž má 8 hlav, 2 048 stop a 35 sektorů na stopu[13]. Řadič tohoto disku bude zbytku systému lhát a tvrdit, že má 16 hlav, 1 024 stop a 35 sektorů na stopu, což nepřekračuje omezení v počtu stop. Pak bude při každém požadavku systému na přístup k disku překládat adresu, kterou dostane tak, že počet hlav vydělí dvěma a počet stop dvěma vynásobí. Matematické úpravy budou v praxi složitější, protože skutečná čísla nejsou tak „pěkná", jako v uvedeném příkladě. Ale nezbývá než zopakovat, že detaily nejsou tak důležité pro pochopení principu. Překládání adres zkresluje pohled operačního systému na to, jak je disk ve skutečnosti organizovaný. To je nepraktické, protože nelze pro snížení přístupové doby a zvýšení výkonu použít „trik" s ukládáním všech souvisejících dat na jeden cylindr.

[12] BIOS je speciální software pro řízení počítače, uložený v paměti RAM. Mimo jiné má na starosti počáteční fáze zavádění systému.

[13] Čísla jsou vymyšlená.



Obrázek 5.1 – Schematické znázornění disku

Překlady adres jsou výlučně problémem disků typu IDE. Disky typu SCSI používají sekvenční čísla sektorů. To znamená, že řadič disku SCSI překládá každé sekvenční číslo sektoru na trojici [hlava, cylindr, sektor]. Navíc používá úplně jinou metodu komunikace s CPU, a proto jsou disky SCSI těchto problémů ušetřeny. Uvědomte si ale, že ani u disků SCSI počítač nezná jejich skutečnou geometrii.

Vzhledem k tomu, že operační systém Linux obvykle nezná skutečnou geometrii disku, nebudou se ani jeho souborové systémy pokoušet ukládat soubory na stejné cylindry. Místo toho se pokusí souborům přidělit sekvenčně řazené sektory. Tato metoda téměř vždy zaručí podobný výkon, jakého lze dosáhnout při ukládání souvisejících dat na jeden cylindr. Celá problematika je o něco složitější, řadiče například využívají vlastní vyrovnávací paměti, nebo mechanismus automatického, řadičem řízeného „přednačítání“ sekvenčně řazených sektorů.

Každý pevný disk je v systému reprezentován samostatným speciálním souborem. Typicky mohou být v systému maximálně dva nebo čtyři pevné disky IDE. Zastupují je pak speciální soubory `/dev/hda`, `/dev/hdb`, `/dev/hdc` a `/dev/hdd`. Pevné disky SCSI reprezentují speciální soubory `/dev/sda`, `/dev/sdb` a tak dále. Podobné konvence týkající se názvů speciálních souborů platí i pro pevné disky jiných typů, podrobnosti najdete v kapitole 5. Pamatujte na to, že speciální soubory zastupující pevné disky umožňují přístup k disku jako celku, bez ohledu na diskové oblasti (o kterých budeme hovořit za chvíli). Není proto těžké při práci s disky pochybit. Neopatrnost může v tomto případě vést ke ztrátě dat. Speciální soubory disků se obvykle používají pouze pro přístup k hlavnímu zaváděcímu sektoru disku, o kterém se také zmíníme později.

Diskety

Disketa sestává z pružné membrány pokryté z jedné nebo obou stran podobnou magnetickou substancí, jako pevný disk. Pružný disk samotný nemá čtecí a zápisovou hlavu, ta je součástí disketové mechaniky. Disketa vlastně odpovídá jedné desce pevného disku, ale na rozdíl od pevného disku je vyměnitelná. Jednu disketovou mechaniku lze využít pro přístup k různým disketám, kdežto pevný disk je jedinou nedílnou jednotkou.

Podobně jako pevný disk se i disketa dělí na stopy a sektory. Dvě korespondující si stopy každé strany diskety tvoří cylindr. Počet stop a sektorů je ale pochopitelně o hodně nižší než u pevného disku.

Disketová jednotka umí obvykle pracovat s několika různými typy disket. Například 3,5palcová disketová mechanika může pracovat jak s disketami o velikosti 720 kB, tak 1,44 MB. Vzhledem k tomu, že disketová jednotka musí zacházet s každým typem diskety trochu jinak, musí i operační systém vědět, který typ diskety je zrovna zasunutý v mechanice. Proto existuje pro jednotky pružných disků množství speciálních souborů, a to vždy jeden pro každou kombinaci typu disketové jednotky a typu diskety. Takže soubor `/dev/fd0H1440` pak reprezentuje první disketovou jednotku (`fd0`). Musí to být 3,5palcová mechanika pracující s 3,5palcovými disketami vysoké hustoty záznamu (proto `H` v názvu zařízení) s kapacitou 1 440 kB (proto `1440`). To jsou běžné 3,5palcové diskety HD.

Konstrukce tvorby názvů speciálních souborů zastupujících disketové jednotky je poměrně složitá. Proto má Linux pro diskety i zvláštní typy zařízení. Tato zařízení automaticky detekují typ diskety zasunuté v mechanice. Fungují tak, že se při požadavku na přístup pokouší přečíst první sektor vložené diskety, přičemž postupně zkouší jejich různé typy, až se jim podaří najít ten správný. Samozřejmě, podmínkou je, aby vložená disketa byla nejdříve naformátovaná. Automatická zařízení zastupují speciální soubory `/dev/fd0`, `/dev/fd1` a tak dále.

Parametry, které tato automatická zařízení používají pro přístup k disketám, lze nastavit také programem **setfdprm**. To se může hodit jednak v případě, že používáte diskety, jež nemají běžnou

kapacitu, to znamená, že mají neobvyklý počet sektorů, dále v případě, že autodetekce z neznámých důvodů selže a nebo když vlastní speciální soubor chybí.

Kromě toho, že Linux zná všechny standardní formáty disket, umí pracovat i s množstvím nestandardních formátů. Některé z nich ale vyžadují speciální formátovací programy. Touto problematikou se teď nebudeme zabývat a doporučíme projít si soubor `/etc/fdprm`. Ten blíže specifikuje nastavení, která program **setfdprm** podporuje.

Operační systém musí vědět o tom, že byla disketa v mechanice vyměněna. Jinak by totiž mohl například použít data z dříve vložené diskety, uložená ve vyrovnávací paměti. Bohužel, vodič, jenž se používá k signalizaci výměny diskety, bývá někdy poškozený. Při používání disketové jednotky v systému MS-DOS nebude mechanika vůbec schopná indikovat systému výměnu média, což je ještě horší situace. Jestli jste se někdy setkali s podivnými, zdánlivě nevysvětlitelnými problémy při práci s disketami, jejich příčinou mohla být právě nefunkční indikace výměny média. Jediným způsobem, jak lze tyto problémy odstranit, je nechat disketovou mechaniku opravit. *Pozn. korektora: Vzhledem k dnešním cenám je lepší koupit novou.

Jednotky CD-ROM

Jednotky CD-ROM čtou opticky data z plastických disků. Informace jsou zaznamenány na povrchu těchto disků[14] jako miniaturní „dolíčky“ seřazené v husté spirále, jež začíná uprostřed disku a končí na jeho okraji. Jednotka vysílá laserový paprsek, který čte data z disku tak, že sleduje tuto spirálu. Od hladkého povrchu disku se odráží jinak, než když narazí na dolík. Tímto způsobem lze jednoduše kódovat binární informace. Ostatní je prostě pouhá mechanika.

Ve srovnání s pevnými disky jsou jednotky CD-ROM pomalé. Pevné disky mají průměrnou přístupovou dobu typicky menší než 15 milisekund, kdežto rychlá jednotka CD-ROM bude data vyhledávat s přístupovou dobou řádově v desetinách sekundy. Skutečná rychlost přenosu dat je ale celkem vysoká, zhruba stovky kilobajtů za sekundu. To, že je jednotka CD-ROM „pomalá“, znamená, že ji nelze pohodlně využít jako alternativu pevných disků, i když to samozřejmě možné je. Některé distribuce Linuxu totiž nabízejí takzvané „live“ souborové systémy na CD-ROM. Ty fungují tak, že část souborů se při instalaci nakopíruje na pevný disk a v případě potřeby se čtou přímo z kompaktního disku. Instalace je pak jednodušší, méně časově náročná a ušetří se také značná část diskového prostoru. Jednotky CD-ROM lze s výhodou využít při instalaci nového programového vybavení, protože při instalování není vysoká rychlost určujícím faktorem.

Je několik způsobů, jak se data na disky CD-ROM ukládají. Nejrozšířenější z nich upravuje mezinárodní standard ISO 9660. Tato norma definuje minimální souborový systém, který je ještě o něco primitivnější než systém souborů používaný systémem MS-DOS. Na druhou stranu je souborový systém ISO 9660 tak jednoduchý, že by jej měly bez potíží dokázat mapovat na svůj vlastní souborový systém všechny operační systémy.

Pro běžnou práci v Unixu je ale souborový systém ISO 9660 prakticky nepoužitelný. Proto se zavedlo rozšíření tohoto standardu, kterému se říká „Rock Ridge“. Rock Ridge například umožňuje používat dlouhá jména souborů, symbolické odkazy a mnoho dalších vymožeností, díky kterým se disk CD-ROM tváří víceméně jako unixový souborový systém. Navíc, rozšíření Rock Ridge zachovává kompatibilitu se souborovým systémem ISO 9660, takže je použitelné i v jiných než unixových systémech. Operační systém Linux podporuje jak ISO 9660, tak Rock Ridge. Rozšíření rozezná a dále používá zcela automaticky.

Souborový systém je ale pouze jednou stranou mince. Většina disků CD-ROM často obsahuje data, ke kterým lze přistupovat výhradně pomocí zvláštních programů. Bohužel, většina těchto pro-

[14] Přesněji řečeno na kovovém povrchu uvnitř disku, chráněném vnějším plastikovým povrchem.

gramů není určena pro Linux (možná s výjimkou některých programů, jež běží pod **dosemu**, linuxovým emulátorem systému MS-DOS, nebo pod **wine**, emulátorem Windows[15]). Existuje také komerční program VMWare, který softwarově emuluje celý virtuální počítač[16].

K jednotce CD-ROM se také přistupuje prostřednictvím odpovídajícího speciálního souboru. Je několik způsobů, jak připojit jednotku CD-ROM k počítači: buď rozhraním SCSI, nebo pomocí zvukové karty, nebo rozhraním EIDE. Popis dalších podrobností technického řešení jednotlivých způsobů připojení jednotky CD-ROM je nad rámec této knihy. Pro nás je podstatné, že podobně jako u jednotek pružných disků určuje způsob připojení vždy jiný speciální soubor.

Pásky

Magnetopáskové jednotky používají pásky podobné[17] těm, které se používají pro záznam zvuku na magnetofonových kazetách. Páska je již svou povahou sériovým zařízením – aby bylo možné dostat se k některé její části, je nejdřív nutné projít všechny předchozí záznamy. K údajům na disku lze přistupovat náhodně, takže je možné „skočit" přímo na kterékoliv místo na něm. Sériový přístup k datům na páskách je příčinou toho, že jsou pomalé.

Na druhou stranu jsou pásky relativně levné, což kompenzuje nedostatky v rychlosti. Bez problému je lze vyrobit dost dlouhé, takže je na ně možno uložit velké množství dat. To vše předurčuje pásková zařízení k archivaci a zálohování, u nichž se nevyžaduje vysoká rychlost, ale naopak, využívá se nízkých nákladů a velké kapacity.

Formátování

Formátování je procedura zápisu značek, které se používají na označení stop a sektorů na magnetickém médiu. Předtím, než je disk naformátován, jsou magnetické signály na jeho povrchu neuspořádané, chaotické. Formátování do tohoto chaosu vnáší určitý řád. Načrtnou se – obrazně řečeno – linie, ve kterých vedou stopy a ty se pak rozdělí na sektory. Skutečné podrobnosti jsou trochu jiné, ale to není pro tuto chvíli podstatné. Důležité je to, že disk nelze používat bez toho, že by byl naformátován.

Pokud jde o formátování, je terminologie mírně zavádějící. V systémech MS-DOS a Windows se pod pojmem „formátování" rozumí i proces vytvoření souborového systému (o němž bude řeč později). Takže se obě tyto procedury označují jediným pojmem – obzvlášť u disket. Když je nutné je rozlišit, používá se pro formátování v pravém slova smyslu termín *nízkoúrovňové formátování* a pro vytvoření souborového systému označení *vysokoúrovňové formátování*. Terminologie používaná v unixovém světě označuje obě tyto činnosti tradičními pojmy „formátování" a „vytvoření souborového systému". Nejinak tomu bude i v této knize.

Disky IDE a některé disky SCSI jsou formátovány ve výrobě a není třeba je po připojení formátovat znovu, takže většina lidí se o formátování disků nestará. Vlastní naformátování disku dokonce může disku uškodit, protože některé disky vyžadují zvláštní způsob formátování, který pak umožňuje například automatické přemisťování vadných sektorů.

Disky, které je potřeba formátovat (a jež lze formátovat), vyžadují obvykle speciální formátovací programy, protože rozhraní formátovací logiky zabudované v jednotce se liší od jednoho typu disku k druhému. Takovéto formátovací programy jsou často buď součástí řadiče BIOS, nebo běží pouze pod systémem MS-DOS. Ani jeden z těchto prostředků tedy nelze bez problémů použít v systému Linux.

---

15 Je ironií, že název *wine* znamená „Wine Is Not a Emulator". Přísně vzato je *wine* pouze náhradou API. Podrobnosti viz http://www.winehq.com.

16 Viz http://www.vmware.com.

17 Nicméně zcela odlišné.

V průběhu formátování můžete narazit na chybná místa na disku, kterým se říká *vadné bloky* nebo *vadné sektory*. S vadnými bloky si někdy poradí samotný řadič pevného disku, ale v případě, že se jich objeví víc, je potřeba nějakým opatřením zamezit možnému použití těchto vadných částí disku. Mechanismus, který problém vadných bloků řeší, je součástí souborového systému. Způsob, jakým systém souborů pracuje s informacemi o vadných sektorech, je popsán v dalších částech této kapitoly. Jinou alternativou je vytvoření malé diskové oblasti, která by obsahovala pouze vadné bloky disku. Takovýto alternativní postup je vhodný zejména v případě, že je rozsah vadných sektorů velmi velký. Souborové systémy totiž mohou mít s rozsáhlými oblastmi vadných bloků problémy.

Diskety se formátují programem **fdformat**. Speciální soubor, který se má formátovat, se zadává jako jeho parametr. Následujícím příkazem bychom například zformátovali 3,5palcovou disketu s vysokou hustotou záznamu, vloženou do první disketové jednotky:

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
$
```

Pamatujte si, že když chcete použít zařízení s autodetekcí (například `/dev/fd0`), *musíte* nejdříve nastavit parametry zařízení pomocí programu `setfdprm`. Stejný výsledek jako v prvním příkladě mají příkazy:

```
$ setfdprm /dev/fd0 1440/1440
$ fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
$
```

Je obvykle výhodnější vybrat správný speciální soubor, jenž odpovídá typu diskety. Zapamatujte si, že není rozumné formátovat disketu na vyšší kapacitu, než je ta, pro kterou je určena.

Program `fdformat` zároveň prověří disketu – zkontroluje, zda neobsahuje vadné bloky. Pokud narazí na vadný blok, opakovaně se pokusí vadný blok použít (obvykle to uslyšíte – zvuky, které jednotka při formátování vydává, se dramaticky změní). Je-li na disketě pouze „dočasná“ chyba (špatná úroveň signálu po zápisu znečištěnou zápisovou hlavou, planý poplach a podobně), program `fdformat` nic neřekne. Naopak skutečná chyba – fyzicky poškozený sektor, přeruší proces kontroly povrchu diskety a jejího formátování. Jádro systému zapíše hlášení do logovacího souboru po každé vstupně-výstupní chybě, na kterou při formátování narazí. Tato hlášení se zároveň objeví na konzole. Když běží program `syslog`, zapisují se tato hlášení také do souboru `/var/log/messages`. Samotným programem `fdformat` uživatel nezjistí, kde přesně se vadný blok nachází (obvykle to ani nikoho nezajímá – diskety jsou tak levné, že se ty vadné automaticky vyhazují).

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... read: Unknown error
$
```

Vadné bloky na disku nebo diskové oblasti (i disketě) lze vyhledat pomocí příkazu **badblocks**. Neprovádí formátování disku, takže je možné kontrolovat i disky, na nichž již existuje souborový systém. Níže uvedený příklad hledá chyby na 3,5palcové disketě se dvěma vadnými bloky:

```
$ badblocks /dev/fd0H1440 1440
718
719
$
```

Výstupem příkazu `badblocks` jsou čísla vadných sektorů. Většina souborových systémů umí takovéto vadné bloky označit jako nepoužitelné. Systémy souborů udržují seznam, tabulku vadných sektorů, která se zakládá, když se systém souborů vytváří. Tento seznam pak lze kdykoliv měnit. První kontrola vadných bloků se dělá příkazem `mkfs`, jímž se vytváří souborový systém. Další kontroly se dělají programem `badblocks` a nové chybné bloky se přidávají do seznamu vadných bloků příkazem `fsck`. Příkazy `mkfs` a `fsck` popíšeme později.

Řada moderních disků umí automaticky zjistit výskyt vadných bloků a pokouší se „opravit" je tak, že místo nich použije k těmto účelům zvlášť vyhrazené správné sektory. Mechanismus náhrady chybného bloku správným je pro operační systém transparentní. Pokud se zajímáte o podrobnosti, měly by být popsány v návodu k disku. Avšak i disky tohoto typu by teoreticky mohly selhat, kdyby počet vadných bloků výrazně vzrostl, nicméně s největší pravděpodobností disk „umře" z jiných důvodů daleko dříve.

Diskové oblasti

Pevný disk může být rozdělen na několik *diskových oblastí*. Každá disková oblast se chová tak, jako by byla samostatným diskem. Diskové oblasti mají smysl v případě, že máte jeden pevný disk a chcete na něm používat například dva operační systémy – pak disk rozdělíte na dvě diskové oblasti a každý operační systém bude používat vlastní diskovou oblast a nebude zasahovat do druhé. Takto mohou oba operační systémy v klidu a míru koexistovat na jediném disku. Kdybyste nepoužili rozdělení disku na samostatné diskové oblasti, museli byste zakoupit pevný disk pro každý z operačních systémů.

Diskety se na diskové oblasti nerozdělují. Z technického hlediska to možné je, ale vzhledem k tomu, že mají malou kapacitu, by oblasti byly prakticky využitelné jenom v ojedinělých případech. Ani disky CD-ROM se obvykle nedělí na oblasti, protože je praktičtější je používat jako jeden velký disk a skutečně málokdy je potřeba mít na jednom disku CD-ROM uloženo několik operačních systémů.

Hlavní zaváděcí sektor, zaváděcí sektory a tabulka diskových oblastí

Informace o tom, jak je pevný disk rozdělen na diskové oblasti, je uložena v prvním sektoru (to jest, prvním sektoru první stopy první vrstvy disku). Tento sektor obsahuje takzvaný *hlavní zaváděcí záznam* (master boot record, MBR). Je to sektor, který se načítá a spouští systémem BIOS pokaždé, když se počítač spustí. Zaváděcí sektor disku obsahuje krátký program, jenž načte tabulku diskových oblastí a zjistí, která oblast disku je aktivní (tedy označená jako zaváděcí). Pak přečte první sektor této oblasti, takzvaný *zaváděcí sektor*. (MBR je také zaváděcí sektor, ale má zvláštní postavení, a proto i zvláštní označení.) Zaváděcí sektor na diskové oblasti obsahuje další krátký program, který načítá první část operačního systému, jenž je na této diskové oblasti uložený (samozřejmě za předpokladu, že je disková oblast bootovatelná), a spouští jej.

Metoda dělení disku na diskové oblasti není hardwarově implementovaná a není ani součástí systému BIOS. Jde čistě o konvenci podporovanou většinou operačních systémů. Ale ne všechny operační systémy se chovají podle těchto konvencí, jsou i výjimky. Některé operační systémy sice podporují dělení disku na diskové oblasti, ale v rámci své diskové oblasti používají svou vlastní interní metodu dělení. Tyto typy operačních systémů mohou bez jakýchkoliv zvláštních prostředků spolupracovat s jinými systémy (včetně operačního systému Linux). Naopak, takový ope-

rační systém, jenž rozdělení disku na diskové oblasti nepodporuje, nemůže koexistovat na stejném pevném disku s jiným operačním systémem.

Je dobré si na kousek papíru vypsat tabulku rozdělení disku na oblasti. Kdyby pak náhodou v budoucnu došlo k poškození některé oblasti, nemusíte díky tomuto jednoduchému bezpečnostnímu opatření přijít o všechna data. (Poškozenou tabulku oblastí lze opravit programem **fdisk**.) Důležité informace získáte příkazem **fdisk -l**:

```
$ fdisk -l /dev/hda

Disk /dev/hda: 15 heads, 57 sectors, 790 cylinders
Units = cylinders of 855 * 512 bytes

        Device Boot    Begin   Start   End     Blocks  Id      system
/dev/hda1              1       1       24      10231+  82      Linux swap
/dev/hda2              25      25      48      10260   83      Linux native
/dev/hda3              49      49      408     153900  83      Linux native
/dev/hda4              409     409     790     163305  5       Extended
/dev/hda5              409     409     744     143611+ 83      Linux native
/dev/hda6              745     745     790     19636+  83      Linux native
$
```

Rozšířené a logické diskové oblasti

Původní schéma dělení disků počítačů PC na diskové oblasti umožňuje vytvořit pouze čtyři oblasti. To se záhy v praxi ukázalo jako nedostatečné. Zčásti například proto, že někteří uživatelé chtěli mít na svém počítači i víc než čtyři operační systémy (Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD nebo Windows/NT, a to jsme vyjmenovali jenom některé), ale především proto, že je výhodné mít několik diskových oblastí i pro jeden operační systém. Například z důvodů zlepšení odezvy operačního systému je lepší nevyužívat pro odkládací prostor systému Linux hlavní diskovou oblast operačního systému, ale mít odkládací prostor na samostatné diskové oblasti (viz dále).

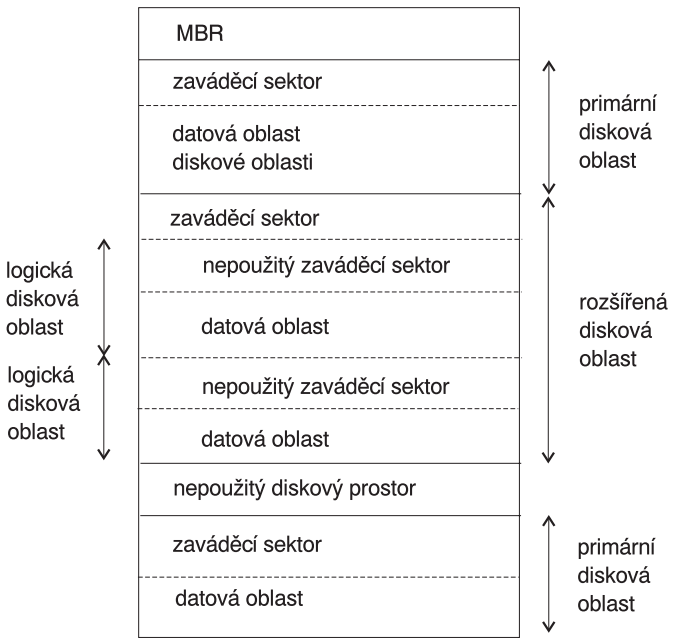
Aby bylo možné omezení počtu diskových oblastí obejít, byly zavedeny takzvané *rozšířené diskové oblasti*. Tento trik umožňuje rozdělit *primární diskové oblasti* na podoblasti. Takto rozdělená primární oblast je onou rozšířenou oblastí a její části (podoblasti) jsou takzvané *logické diskové oblasti*. Chovají se stejně jako primární, ale jsou vytvořeny jiným způsobem. Není mezi nimi ale žádný rozdíl v rychlosti.

Struktura oblastí pevného disku by mohla vypadat například tak, jak je uvedeno na obrázku 5.2. Disk je rozdělen na tři primární diskové oblasti. Druhá z nich je rozdělena na dvě logické. Část disku nepatří vůbec žádné diskové oblasti. Disk jako celek a každá primární oblast má svůj zaváděcí sektor.

Typy diskových oblastí

Tabulky rozdělení disku (jedna v MBR a další na rozšířených diskových oblastech) mají vyhrazený jeden bajt pro každou diskovou oblast a ten identifikuje její typ. Snaží se popsat operační systém, který diskovou oblast používá, nebo její účel. Informační bajt by měl zamezit tomu, aby dva operační systémy pracovaly s jednou diskovou oblastí. Avšak ve skutečnosti většina operačních systémů označení typu diskové oblasti ignoruje. I Linux se například vůbec nezajímá o to, jak je určitá disková oblast označená. Dokonce – což je ještě horší, některé operační systémy tento bajt

používají nesprávně. Například přinejmenším některé verze systému DR-DOS ignorují nejvyšší bit tohoto bajtu.



Obrázek 5.2 – Příklad rozdělení pevného disku na diskové oblasti

Žádný z úřadů pro normalizaci nespecifikoval, co která hodnota bajtu znamená. Některé obecně přijaté hodnoty a odpovídající typy uvádí tabulka 5.1. Obsáhlejší seznam vypíše linuxový program **fdisk**.

0	prázdný	40	Venix 80286	94	Amoeba BBT
1	DOS 12-bit FAT	51	Novell?	a5	BSD/386
2	XENIX root	52	Microport	b7	BSDI fs
3	XENIX usr	63	GNU HURD	b8	BSDI swap
4	DOS 16-bit FAT (<32 M)	64	Novell	c7	Syrinx
5	Extended	75	PC/IX	db	CP/M
6	DOS 16-bit FAT (>= 32 M)	80	Old MINIX	e1	DOS access
7	HPFS/NTFS	81	Linux/MINIX	e3	DOS R/O
8	AIX	82	Linux swap	f2	DOS secondary
9	AIX bootable	83	Linux native	ff	BBT
a	OS/2 Boot Manag	93	Amoeba		

Tabulka 5.1 – Typy diskových oblastí (podle programu fdisk)

Dělení pevného disku na diskové oblasti

Je mnoho programů, které umí vytvářet a mazat diskové oblasti. Součástí většiny operačních systémů je nějaký takový a bývá rozumné používat program, jenž je součástí operačního systému, v jehož prostředí s diskovými oblastmi pracujete, hlavně proto, kdyby dělal něco speciálního, co jiné nedělají. Většinou se tyto programy jmenují **fdisk** (včetně toho, který je součástí distribuce systému Linux) nebo nějak podobně. Detaily týkající se možností linuxového programu **fdisk** podrobně popisuje jeho manuálová stránka. Podobný je příkaz **cfdisk**, který má hezčí, celoobrazovkové uživatelské rozhraní.

V případě, že používáte disky IDE, musí být celá zaváděcí disková oblast (tedy disková oblast, na které jsou uloženy soubory obrazů jádra) na prvních 1 024 cylindrech. To proto, že při zavádění operačního systému (předtím, než systém přechází do chráněného režimu) se k disku přistupuje přes BIOS a ten neumí pracovat s více než 1 024 cylindry. V některých případech je možné používat zaváděcí diskovou oblast, která leží na prvních 1 024 cylindrech jenom částečně. To je možné jenom když na prvních 1 024 cylindrech budou uloženy všechny soubory, které při inicializaci čte systém BIOS. Vzhledem k tomu, že takovéhoto uspořádání je velmi obtížné dosáhnout, *je lepší je vůbec nepoužívat* – nikdy si totiž nemůžete být jistí, zda změna parametrů jádra systému nebo defragmentace disku nezpůsobí, že systém nebude vůbec možné zavést. Proto se raději vždy ujistěte, že je zaváděcí disková oblast vašeho systému celá na prvních 1 024 cylindrech[18].

Některé nové verze systémů BIOS a disků IDE již umí pracovat i s disky, které mají více než 1 024 cylindrů. Máte-li takovýto systém, můžete na tento problém zapomenout. Jestli si v tom nejste zcela jisti, umístěte raději podle doporučení zaváděcí diskovou oblast na prvních 1 024 cylindrů.

Každá disková oblast by měla mít sudý počet sektorů, protože souborové systémy Linuxu používají diskové bloky o velikosti 1 kB, tedy dva sektory. Lichý počet sektorů diskové oblasti způsobí, že poslední sektor bude nevyužitý. Nemělo by to způsobit žádné zvláštní problémy, ale není to příliš elegantní. Některé verze programu **fdisk** vás budou na tento stav upozorňovat.

Při změně velikosti diskové oblasti je nejlepší vytvořit zálohu všeho, co chcete na diskovou oblast zachránit (a ještě lepší je zálohovat úplně všechno), pak diskovou oblast smazat, vytvořit novou a obnovit na ní soubory ze zálohy. Chcete-li zvětšit velikost nějaké diskové oblasti, budete muset pravděpodobně upravit i velikosti (tedy vytvořit zálohy a obnovit soubory) sousedních diskových oblastí.

Vzhledem k tomu, že změny velikostí diskových oblastí jsou dost pracné, je lepší nastavit je správně hned napoprvé. Jinak budete potřebovat efektivní a jednoduchý systém zálohování. Instalujete-li poprvé systém z médií, jež nevyžadují časté zásahy obsluhy (například z CD-ROM – protikladem jsou diskety), je často jednodušší si nejdřív pohrát s různými konfiguracemi. Jelikož zatím na disku nemáte žádná data, která by bylo potřeba zálohovat, není natolik bolestné několikrát změnit velikosti diskových oblastí.

Existuje program pro systém MS-DOS, který se jmenuje **fips**, a ten umí změnit velikosti diskových oblastí systému MS-DOS bez toho, že by bylo potřeba zálohovat, mazat a obnovovat soubory z těchto diskových oblastí. Pro jiné souborové systémy je to zatím nadále nevyhnutelné[19].

Speciální soubory a diskové oblasti

Každá disková oblast a rozšířená disková oblast má svůj vlastní soubor zařízení. Podle konvence pro konstrukci názvů souborů zařízení se číslo diskové oblasti připojí za jméno celého disku s tím,

[18] Toto omezení neplatí pro novější verze programu LILO, které umí pracovat s LBA (Logical Block Addressing). V dokumentaci vaší distribuce byste měli zjistit, zda vaše verze LILO tento mechanismus podporuje.

[19] Program **fips** je součástí většiny distribucí Linuxu. Podobnou funkci, ale s pěknějším rozhraním, nabízí i komerční program Partition Magic. Mějte na paměti, že manipulace s oddíly je vždy riziková – než začnete měnit velikost oddílů, *vždy* si zálohujte důležitá data. Velikosti různých typů oddílů umí změnit i GNU program **parted**, někdy však s jistými omezeními. Než se do této operace pustíte, přečtěte si dokumentaci.

že 1–4 budou primární disková oblast (podle toho, kolik primárních diskových oblastí bylo vytvořeno) a 5–8 jsou logické diskové oblasti (bez ohledu na to, na které z primárních diskových oblastí jsou vytvořeny). Například `/dev/hda1` je první primární disková oblast prvního pevného disku IDE a `/dev/sdb7` je třetí rozšířená disková oblast na druhém pevném disku SCSI.

Souborové systémy

Co jsou to souborové systémy?

Souborový systém tvoří metody a struktury dat, pomocí kterých operační systém udržuje záznamy o souborech na discích a diskových oblastech. Jde tedy o způsob, jakým jsou soubory na disku organizované. Tento termín se ale používá i k označení diskové oblasti nebo disku, na kterém se ukládají soubory, nebo ve významu typu souborového systému. Takže když někdo řekne „mám dva souborové systémy“, může mít na mysli to, že má disk rozdělen na dvě diskové oblasti, nebo to může znamenat, že používá „souborový systém ext2“, tedy určitý typ souborového systému.

Rozdíl mezi diskem či diskovou oblastí a souborovým systémem, který je na nich vytvořený, je dost podstatný. Jen některé programy (mezi nimi – zcela logicky – programy, pomocí kterých se souborové systémy vytváří) pracují přímo se sektory disku nebo diskové oblasti. Jestli na nich byl předtím vytvořený systém souborů, bude po spuštění takovýchto programů zničen nebo vážně poškozen. Většina aplikací ale pracuje se souborovým systémem. Proto je nelze použít na diskové oblasti, na které není vytvořen žádný systém souborů (nebo na které je vytvořen souborový systém nesprávného typu).

Předtím, než bude možné diskovou oblast nebo disk použít, je potřeba je inicializovat – musí se na ně zapsat určité datové struktury. Tento proces se označuje jako *vytvoření souborového systému*.

Většina unixových souborových systémů má podobnou obecnou strukturu, v dalších podrobnostech se ale celkem dost liší. Mezi ústřední pojmy patří *superblok*, *inode*, *datový blok*, *adresářový blok* a *nepřímý blok*. Superblok obsahuje informace o souborovém systému jako celku, například jeho velikost (zrovna u této položky závisí přesné hodnoty na konkrétním souborovém systému). Inode obsahuje všechny informace o souboru kromě jeho jména. Jméno souboru je uloženo v adresáři společně s odpovídajícím číslem inode. Adresářová položka obsahuje jména souborů a čísla inodů, které tyto soubory reprezentují. Inode dále obsahuje čísla datových bloků, v nichž jsou uložená data souboru, který daný inode zastupuje. V inode je ale místo jenom pro několik čísel datových bloků. Když je jich potřeba víc, je dynamicky alokováno víc místa pro další ukazatele na datové bloky. Tyto dynamicky alokované bloky jsou uloženy v nepřímém bloku. Jak jejich název naznačuje, v případě, že je potřeba najít datový blok, musí se nejdřív najít jeho číslo v nepřímém bloku.

Unixové souborové systémy obvykle umožňují vytvořit v souboru *díru*, a to pomocí systémového volání `lseek()` (podrobnosti uvádí příslušná manuálová stránka). Vypadá to tak, že souborový systém předstírá, že je na konkrétním místě souboru uložen blok nulových bajtů, nicméně pro něj není vyhrazen žádný sektor pevného disku (to znamená, že takovýto soubor zabírá o něco méně diskového prostoru). S tím se můžete setkat zvlášť často u malých binárních souborů, sdílených knihoven Linuxu, některých databází a v několika dalších speciálních případech. (Díry jsou implementovány prostřednictvím speciální adresy datového bloku v nepřímém bloku nebo inode. Tato zvláštní adresa znamená, že pro určitou část souboru není alokován žádný datový blok, tedy že je v tomto souboru díra.)

Galerie souborových systémů

Linux podporuje několik typů souborových systémů. Mezi nejdůležitější patří:

minix

Je nejstarší a je považován za nejspolehlivější. Má ale několik omezení – chybí časová razítka, jména souborů mohou být nejvíce 30 znaků dlouhá, souborový systém může mít maximálně 64 MB a další.

xia

Modifikovaná verze souborového systému minix. Nemá omezení v délce jmen souborů a velikosti souborového systému, jinak nepřináší žádné nové rysy. Mezi uživateli není příliš oblíbený, ale jinak má pověst velmi spolehlivého systému.

ext2

Souborový systém, který má nejvíce různých možností ze všech zde uvedených původních souborových systémů pro Linux. V současnosti je také nejpopulárnější. Byl navržen tak, aby byl zpětně kompatibilní, takže nové verze kódu souborového systému nevyžadují nové, opakované vytváření již existujících souborových systémů.

ext

Starší verze ext2, která nebyla zpětně kompatibilní. Lze ji jenom stěží používat v nových instalacích Linuxu – většina uživatelů již přešla na systém souborů ext2.

Kromě toho je podporováno několik souborových systémů jiných operačních systémů, což umožňuje přenášet soubory mezi různými operačními systémy. Tyto cizí, nepůvodní souborové systémy jinak fungují jako původní systémy souborů pro Linux, ale obvykle postrádají některé rysy typické pro Unix, případně mají některá neobvyklá omezení nebo jiné zvláštnosti.

msdos

Souborový systém kompatibilní se souborovým systémem FAT operačního systému MS-DOS (také OS/2 a Windows NT).

umsdos

Rozšiřuje možnosti ovladače souborového systému msdos pro systém Linux. Umí pracovat s dlouhými názvy souborů, zná vlastníky souborů, přístupová práva, odkazy a speciální soubory. To umožňuje používat běžný souborový systém msdos jako by byl originálním linuxovým souborovým systémem. Rovněž není potřeba mít oddělené diskové oblasti pro systémy Linux a MS-DOS.

iso9660

Standardní souborový systém disků CD-ROM. Oblíbené rozšíření „Rock Ridge“ tohoto standardu automaticky zavádí delší jména souborů a další možnosti.

nfs

Síťový souborový systém, který umožňuje sdílení souborových systémů mezi větším počtem počítačů a jednoduchý přístup k souborům každého z nich.

smbfs

Síťový souborový systém pro sdílení dat s MS Windows. Používá síťové protokoly kompatibilní s Windows.

hpfs

Souborový systém operačního systému OS/2.

sysv

Souborové systémy operačních systémů SystemV/386, Coherent a Xenix.

Výběr konkrétního souborového systému závisí na dané situaci. V případě, že jsou kompatibilita nebo jiná omezení nutnou podmínkou výběru některého souborového systému jiného operačního systému, nezbývá než použít tento nepůvodní systém souborů. Pokud nejste ve výběru omezení, bude pravděpodobně nejrozumnější používat ext2, protože má ze všech uvedených systémů souborů nejvíce možností a nemá nedostatky z hlediska výkonu.

Dalším ze souborových systémů je systém souborů `proc`, nejčastěji dostupný prostřednictvím adresáře /proc. Ve skutečnosti ale není souborovým systémem v pravém slova smyslu, i když tak na první pohled vypadá. Systém souborů proc umožňuje přístup k určitým datovým strukturám jádra systému, například k seznamu procesů (odtud jeho jméno). Přizpůsobuje tyto datové struktury tak, že se navenek chovají jako soubory souborového systému. K systému souborů proc pak lze přistupovat všemi běžnými nástroji, které se soubory běžně pracují. Takže kdybyste chtěli znát například seznam všech procesů, zadali byste příkaz:

```
$ ls -l /proc
total 0
dr-xr-xr-x     4 root        root        0 Jan 31 20:37 1
dr-xr-xr-x     4 liw         users       0 Jan 31 20:37 63
dr-xr-xr-x     4 liw         users       0 Jan 31 20:37 94
dr-xr-xr-x     4 liw         users       0 Jan 31 20:37 95
dr-xr-xr-x     4 root        users       0 Jan 31 20:37 98
dr-xr-xr-x     4 liw         users       0 Jan 31 20:37 99
-r--r--r--     1 root        root        0 Jan 31 20:37 devices
-r--r--r--     1 root        root        0 Jan 31 20:37 dma
-r--r--r--     1 root        root        0 Jan 31 20:37 filesystems
-r--r--r--     1 root        root        0 Jan 31 20:37 interrupts
-r--------     1 root        root  8654848 Jan 31 20:37 kcore
-r--r--r--     1 root        root        0 Jan 31 11:50 kmsg
-r--r--r--     1 root        root        0 Jan 31 20:37 ksyms
-r--r--r--     1 root        root        0 Jan 31 11:51 loadavg
-r--r--r--     1 root        root        0 Jan 31 20:37 meminfo
-r--r--r--     1 root        root        0 Jan 31 20:37 modules
dr-xr-xr-x     2 root        root        0 Jan 31 20:37 net
dr-xr-xr-x     4 root        root        0 Jan 31 20:37 self
-r--r--r--     1 root        root        0 Jan 31 20:37 stat
-r--r--r--     1 root        root        0 Jan 31 20:37 uptime
-r--r--r--     1 root        root        0 Jan 31 20:37 version
$
```

(V seznamu bude vždy několik souborů, které neodpovídají žádným procesům. Výstup ve výše uvedeném příkladu byl zkrácen.)

Je důležité si uvědomit, že i když se pro systém proc používá označení „souborový systém“, žádná z jeho částí neleží na žádném z disků. Existuje jenom „v představách“ jádra systému. Kdykoliv k některé části souborového systému proc přistupuje uživatel systému nebo některý z procesů, jádro „předstírá“, že je tato část uložená na disku, ale ve skutečnosti tomu tak není. Takže i když je v souborovém systému proc uložený několikamegabajtový soubor `/proc/kcore`, ve skutečnosti nezabírá na disku žádné místo.

Který souborový systém použít?

Obvykle není moc důvodů používat několik různých souborových systémů. V současné době je nejoblíbenějším souborový systém ext2fs a to je současně pravděpodobně ta nejrozumnější volba. Ve vztahu ke zmiňovaným účetním strukturám, rychlosti, (pochopitelně) spolehlivosti, kompatibilitě a vzhledem k různým jiným důvodům by mohlo být vhodné zvolit i jiný systém souborů. Takovéto požadavky je třeba posuzovat případ od případu[20].

Vytváření souborového systému

Souborové systémy se vytváří a inicializují příkazem **mkfs**. Je to vlastně vždy jiný program pro každý typ souborového systému. Program **mkfs** je jenom koncové rozhraní, program, který spouští některé další programy, podle typu požadovaného souborového systému. Typ souborového systému se volí přepínačem `-t fstype`.

Programy volané příkazem **mkfs** mají nepatrně odlišné rozhraní příkazové řádky. Běžně používané a nejdůležitější volby jsou uvedeny níže. Více informací najdete v manuálových stránkách.

`-t fstype` Typ souborového systému.

-c Vyhledat a ošetřit chybné bloky.

`-l filename` Přečíst seznam vadných bloků ze souboru.

Kdybyste chtěli vytvořit souborový systém ext2 na disketě, zadali byste následující příkazy:

```
$ fdformat -n /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
$ badblocks /dev/fd0H1440 1440 / bad-blocks
$ mkfs -t ext2 -l bad-blocks /dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

V prvním kroku se disketa formátuje (volba -n zakáže její validaci, tedy kontrolu vadných sektorů). Vadné bloky pak vyhledává program **badblocks**, a to s výstupem přesměrovaným do souboru bad-blocks. Nakonec se vytvoří souborový systém a mezi účetní struktury se uloží seznam vadných bloků, ve kterém budou všechny vadné sektory, jež našel program **badblocks**.

Místo příkazu **badblocks** je možno použít parametr -c programu **mkfs** tak, jak to uvádí následující příklad:

```
$ mkfs -t ext2 -c /dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
```

[20] V současné době existuje několik kandidátů, kteří mohou nahradit systém ext2, například reiserfs nebo ext3. Oba tyto souborové systémy podporují „žurnálování". Definice a popis žurnálování je mimo (stávající) rozsah této příručky, velmi stručně řečeno jde ale o mechanismus, který zajišťuje výrazně vyšší odolnost souborového systému proti výpadkům napájení nebo jiným nekorektním formám vypnutí počítače. Snižuje se tak pravděpodobnost ztráty dat a je velmi pravděpodobné, že některý z těchto souborových systémů se v brzké době stane novým standardem.

```
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

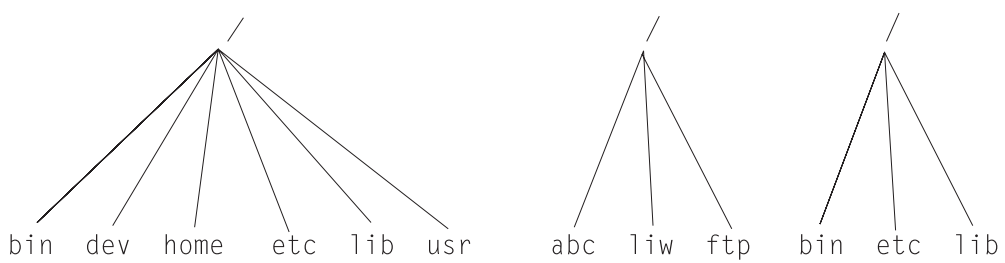
Volba `-c` programu `mkfs` je sice výhodnější než `pbadblocks` pro kontrolu vadných bloků je vždy nutné použít po vytvoření souborového systému.

Postup, kterým se vytváří souborové systémy na pevných discích a diskových oblastech, je stejný jako naznačený postup inicializace souborového systému na disketě, s tím rozdílem, že není nutné je formátovat.

Připojení a odpojení

Souborový systém se musí před použitím *připojit*. Po připojení systému souborů dělá operační systém některé účetní operace, kterými se ověřuje funkčnost připojení. Protože všechny soubory v Unixu jsou součástí jediného hierarchického adresářového stromu, operace připojení souborového systému začlení obsah připojovaného souborového systému do některého z adresářů dříve připojeného systému souborů.

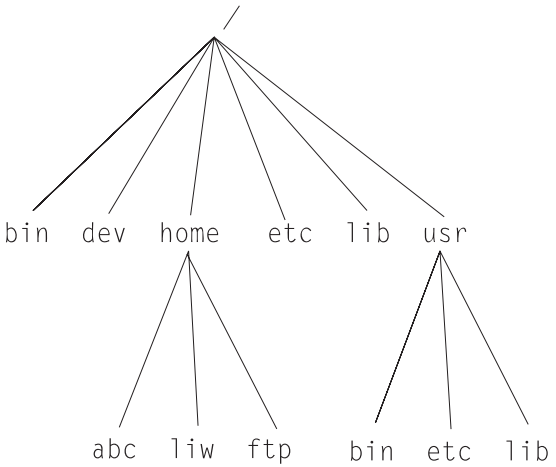
Na obrázku 5.3 jsou například zobrazeny tři samostatné souborové systémy, každý se svým vlastním kořenovým adresářem. Když budou poslední dva souborové systémy připojeny pod adresáře `/home` a `/usr` prvního systému souborů, dostaneme jeden adresářový strom, který je vyobrazen na obrázku 5.4.



Obrázek 5.3 – Tři samostatné souborové systémy

Souborové systémy lze připojit například zadáním následujících příkazů:

```
$ mount /dev/hda2 /home
$ mount /dev/hda3 /usr
$
```



Obrázek 5.4 – Připojené souborové systémy /home a /usr

Příkaz **mount** má dva parametry. Prvním je soubor zařízení odpovídající disku nebo diskové oblasti, na které leží připojovaný souborový systém. Druhým parametrem je adresář, pod nímž bude souborový systém připojen. Po provedení těchto příkazů vypadá obsah obou připojovaných systémů souborů tak, jako by byl součástí hierarchického stromu adresářů /home a /usr. Říkáme, že „/dev/hda2 je připojený do adresáře /home“ a podobně to platí i pro adresář /usr. Když si pak chcete prohlédnout některý ze souborových systémů, procházíte strukturou adresářů, do nichž jsou připojené, jako by to byly jakékoliv jiné adresáře. Je důležité si uvědomit rozdíl mezi souborem zařízení /dev/hda2 a adresářem /home, ke kterému je systém souborů připojen. Speciální soubor zařízení umožňuje přístup k „syrovému“ obsahu pevného disku, kdežto prostřednictvím adresáře /home se přistupuje k souborům, které jsou na tomto disku uloženy. Adresář, ke kterému je souborový systém připojený, se nazývá *bod připojení*.

Operační systém Linux podporuje mnoho typů souborových systémů. Příkaz **mount** se vždy pokusí rozeznat typ připojovaného systému souborů. Lze také použít přepínač -t typ_fs, který přímo specifikuje typ připojovaného souborového systému. Někdy je to potřeba, protože heuristika programu **mount** nemusí vždy pracovat správně. Chcete-li například připojit disketu systému MS -DOS, zadáte příkaz:

```
$ mount -t msdos /dev/fd0 /floppy
$
```

Adresář, ke kterému se souborový systém připojuje, nemusí být prázdný, ale musí existovat. Nicméně v něm uložené soubory budou po dobu připojení nového souborového systému nedostupné prostřednictvím svých názvů. (Soubory, které byly v době připojení otevřené, budou nadále přístupné. Soubory, na něž směřují pevné odkazy z jiných adresářů, budou přístupné prostřednictvím těchto odkazů.) Nehrozí žádné nebezpečí poškození těchto souborů a někdy to navíc může být i užitečné. Někteří uživatelé například rádi používají synonyma /tmp a /var/tmp. Proto si dělají symbolický odkaz adresáře /tmp do adresáře /var/tmp. Předtím, než se při zavádění systému připojí souborový systém /var, je adresář /var/tmp v kořenovém souborovém systému. Po připojení systému /var, bude adresář /var/tmp v kořenovém souborovém systému nepřístupný. V pří-

padě, že by adresář `/var/tmp` v kořenovém souborovém systému neexistoval, bylo by použití dočasných souborů před připojením systému souborů /var nemožné.

Jestli nemáte v úmyslu v připojovaném souborovém systému cokoliv zapisovat, zadejte programu mount přepínač `-r`. Tím se vytvoří připojení pouze pro čtení. Jádro systému pak zabrání každému pokusu o zápis do tohoto souborového systému a nebude ani aktualizovat časy posledního přístupu v inodech souborů. Připojení systému souborů pouze pro čtení je podmínkou u médií, na která nelze zapisovat, například u disků CD-ROM.

Pozorný čtenář si již uvědomil drobný logistický problém. Jakým způsobem se připojuje první souborový systém, tedy kořenový souborový systém (obsahující kořenový adresář celé stromové struktury), když zjevně nemůže být připojený na jiný souborový systém? Odpověď je jednoduchá – je to kouzlo[21]. Kořenový souborový systém se magicky připojuje při zavádění systému a lze se spolehnout na to, že bude připojený vždy. Kdyby z nějakých důvodů souborový systém root nebylo možné připojit, systém se vůbec nezavede. Jméno souborového systému, jenž se připojuje jako kořenový, je buď zkompilované přímo v jádře systému, nebo se nastavuje zavaděčem LILO, případně programem **rdev**.

Kořenový svazek se obvykle nejdříve připojí pouze pro čtení. Pak inicializační skripty spustí program **fsck**, který jej zkontroluje. Když se neobjeví žádný problém, kořenový svazek se připojí znovu, a to tak, že na něj bude možno i zapisovat. Program **fsck** se nesmí spouštět na připojeném souborovém systému s možností zápisu, protože jakékoliv změny v souborovém systému, ke kterým by došlo při kontrole (a opravách) chyb v souborovém systému), způsobí *vážné* potíže. Když je kořenový souborový systém při kontrole připojený pouze pro čtení, program **fsck** může bez obav opravovat všechny chyby, protože operace opětovného připojení souborového systému vyprázdní všechna metadata, která má souborový systém uložená v paměti.

Na řadě systémů se při startu automaticky připojují i jiné souborové systémy. Jsou specifikované v souboru `/etc/fstab`. Podrobnosti týkající se formátu tohoto souboru najdete v manuálové stránce k souboru fstab. Přesné detaily procesu připojování dalších souborových systémů závisí na množství faktorů a je-li potřeba, může je správce systému nastavit, viz kapitola 8.

Když už není třeba mít souborový systém připojený, je možno jej odpojit příkazem **umount**[22]. Program **umount** má jediný parametr, buďto soubor zařízení, nebo bod připojení. Například odpojení adresářů připojených v předchozím příkladu lze provést příkazy:

```
$ umount /dev/hda2
$ umount /usr
$
```

Podrobnější instrukce jak používat příkaz **umount** najdete na manuálové stránce k tomuto programu. Je nutné vždycky odpojit disketovou mechaniku! *Nestačí jenom vytáhnout disketu z mechaniky!* Vzhledem k použití vyrovnávacích pamětí nemusí být data fyzicky zapsána, dokud není disketa odpojena. Předčasné vytažení diskety z mechaniky by mohlo způsobit poškození jejího obsahu. Když z diskety pouze čtete, není její poškození moc pravděpodobné, ale když zapisujete – byť třeba jen omylem – důsledky mohou být katastrofální.

Připojování a odpojování souborových systémů vyžaduje oprávnění superuživatele, takže ho může dělat pouze uživatel root. Je tomu tak například proto, že kdyby měl kterýkoliv z uživatelů právo připojit si jednotku pružných disků na libovolný adresář, nebylo by velmi těžké připojit disketu s virem typu trójského koně „přestrojeného“ za program `/bin/sh`, nebo jiný často používaný příkaz. Avšak dost často je potřeba, aby uživatelé mohli s disketami pracovat. Je několik způsobů, jak jim to umožnit:

[21] Podrobnosti viz zdrojové kódy jádra nebo Kernel Hacker's Guide.

[22] Původně to samozřejmě mělo být **unmount**, nicméně v 70. letech *n* záhadně zmizelo. Pokud byste je našli, prosíme o vrácení do Bell Labs, NJ.

- Sdělit uživatelům heslo superuživatele. To je z hlediska bezpečnosti zjevně špatné, avšak to nejjednodušší řešení. Funguje dobře v případě, že vůbec není potřeba zabývat se zabezpečením systému. To se týká řady osobních systémů – tedy systémů, které nejsou připojeny do počítačové sítě.
- Používat programy jako je **sudo**, jenž umožní uživatelům používat příkaz **mount**. Toto je z hlediska bezpečnosti rovněž špatné řešení, avšak tímto způsobem přímo nepředáváte privilegia superuživatele každému[23].
- Umožnit uživatelům používat balík programů **mtools**, jenž umožňuje manipulovat se souborovými systémy MS-DOS bez toho, že by je bylo potřeba připojovat. Řešení funguje dobře pouze v případě, že jsou diskety systému MS-DOS vším, co budou uživatelé systému potřebovat. V ostatních případech je nevyhovující.
- Zapsat všechny disketové jednotky a pro ně přípustné přípojné body spolu s dalšími vhodnými volbami do seznamu připojovaných systémů v souboru `/etc/fstab`.

Posledně uvedenou alternativu lze realizovat přidáním níže uvedeného řádku do souboru `/etc/fstab`:

```
/dev/fd0 /floppy msdos user,noauto 0 0
```

Význam jednotlivých položek (zleva doprava): soubor zařízení, které se má připojit; adresář, kam se má toto zařízení připojit; typ souborového systému; parametry; četnost zálohování (používá program **dump**); posledním parametrem je pořadí kontroly programem fsck (určuje pořadí, ve kterém by měly být souborové systémy prověřovány při startu systému; 0 znamená nekontrolovat).

Přepínač noauto zakáže automatické připojení při startu systému (tedy nepovolí příkazu **mount –a** toto zařízení připojit). Parametr `user` umožní kterémukoliv uživateli připojit si souborový systém. Z důvodů bezpečnosti zamezí možnosti spouštět programy (jak běžné, tak programy s příznakem `setuid`) a interpretaci souborů zařízení z připojeného souborového systému. Pak si každý uživatel může připojit disketovou jednotku se souborovým systémem msdos tímto příkazem:

```
$ mount /floppy
$
```

Disketu lze odpojit (a musí být odpojena) odpovídajícím příkazem `umount`.

Chcete-li umožnit přístup k několika různým typům disket, musíte zadat několik přípojných bodů. Nastavení pro každý přípojný bod mohou být různá. Například přístup k disketám s oběma typy souborových systémů – MS-DOS i ext2 – byste umožnili přidáním těchto řádků do souboru `/etc/fstab`:

```
/dev/fd0 /dosfloppy msdos user,noauto 0 0
/dev/fd0 /ext2floppy ext2 user,noauto 0 0
```

U souborových systémů MS-DOS (nejenom na disketách) budete pravděpodobně vyžadovat omezený přístup s využitím systémových parametrů `uid, gid a umask`. Ty jsou podrobně popsané v manuálové stránce příkazu `mount`. Následkem neopatrnosti při připojování souborového systému MS-DOS může být totiž to, že kterýkoliv uživatel získá oprávnění číst v připojeném systému souborů kterékoliv soubory, což není moc dobré.

Kontrola integrity souborového systému programem fsck

Souborové systémy jsou poměrně složité struktury a tím také mají sklony k chybovosti. Správnost a platnost souborového systému se kontroluje příkazem **fsck**. Lze jej nastavit tak, aby při kontrole automatic-

[23] Správné nastavení tohoto příkazu si vyžaduje něco intenzivního přemýšlení. Příkaz **sudo** je možné nastavit tak, aby uživateli umožnil provést jen některé operace. Podrobnosti viz manuálové stránky sudo(8), sudoers(5) a visudo(8).

ky opravoval méně závažné chyby a upozorňoval uživatele na výskyt chyb, které nelze odstranit. Naštěstí je kód implementující souborové systémy odladěný velmi dobře, takže problémy vznikají jen zřídka a jsou obvykle zapříčiněny výpadkem napájecího napětí, selháním technického vybavení nebo chybou obsluhy, například nesprávným vypnutím systému.

Většina systémů je nastavena tak, že spouští program **fsck** automaticky při zavádění systému, takže jakékoliv chyby jsou detekovány (a většinou i odstraněny) předtím, než systém přechází do běžného pracovního režimu. Používáním poškozeného systému souborů se totiž jeho stav obvykle ještě více zhoršuje. Jsou-li v nepořádku datové struktury souborového systému, práce se systémem je pravděpodobně poškodí ještě víc, důsledkem čeho mohou být ztráty dat většího rozsahu. Kontrola velkých souborových systémů programem **fsck** chvíli trvá. Když se ale systém vypíná správně, chyby souborových systémů se vyskytují jenom velmi zřídka. Lze pak využít několika triků, díky kterým se lze kontrole (velkých) souborových systémů vyhnout, není-li to nutné. První trik spočívá v tom, že existuje-li soubor `/etc/fastboot`, neprovádí se žádná kontrola. Druhý: souborový systém ext2 má ve svém superbloku speciální příznak, jehož hodnota je nastavena podle toho, jestli byl souborový systém po předchozím připojení odpojen správně, či nikoliv. Podle tohoto příznaku pozná pak program **e2fsck** (verze příkazu **fsck** pro souborový systém ext2), jestli je nutné provádět kontrolu tohoto systému souborů, či nikoliv. V případě, že podle hodnoty příznaku byl daný systém souborů odpojen korektně, kontrola se neprovádí. Předpokládá se, že řádné odpojení systému zároveň znamená, že v souborovém systému nevznikly žádné defekty. To, zda se při proceduře zavádění systému vynechá proces kontroly systému souborů v případě, že soubor `/etc/fastboot` existuje, záleží na nastavení spouštěcích skriptů systému. Trik s příznakem souborového systému ext2 ale funguje vždy, když systém souborů kontrolujete programem **e2fsck**. Kdybyste totiž chtěli programu **e2fsck** přikázat, aby prověřil i korektně odpojený systém souborů, museli byste tento požadavek explicitně zadat odpovídajícím přepínačem. (Podrobnosti o tom jak, uvádí manuálová stránka programu **e2fsck**.)

Automatické prověřování správnosti souborových systémů se provádí jenom u systémů souborů, které se připojují automaticky při startu operačního systému. Manuálně lze program **fsck** použít pro kontrolu jiných souborových systémů, například na disketách.

Najde-li program **fsck** neodstranitelné chyby, připravte se na to, že budete potřebovat buď hluboké znalosti obecných principů fungování souborových systémů a konkrétního typu poškozeného souborového systému zvlášť, nebo dobré zálohy. Druhou možnost lze jednoduše (ačkoli někdy dost pracně) zajistit. Nemáte-li potřebné „know-how“ sami, mohou první alternativu v některých případech zajistit vaši známí, dobrodinci z diskusních skupin o Linuxu na Internetu, popřípadě jiný zdroj technické podpory. Rádi bychom vám poskytli víc informací týkajících se této problematiky, bohužel nám v tom brání nedostatek podrobných znalostí a zkušeností. Jinak užitečný by pro vás mohl být program **debugfs**, jehož autorem je Theodore T'so.

Program **fsck** může běžet pouze na odpojených souborových systémech, v žádném případě ne na připojených (s výjimkou kořenového souborového systému připojeného při zavádění systému pouze pro čtení). To proto, že program přistupuje k „syrovému“ disku, a tak může modifikovat systém souborů bez toho, že by využíval operační systém. Když se vám podaří operační systém tímto způsobem „zmást“, téměř určitě můžete očekávat problémy.

Kontrola chyb na disku programem badblocks

Je vhodné pravidelně kontrolovat výskyt vadných bloků na disku. Dělá se to příkazem **badblocks**. Jeho výstupem je seznam čísel všech vadných bloků, na které program narazil. Tímto seznamem pak můžete „nakrmit“ program **fsck**, který podle něj provede záznamy do datových struktur souborového systému. Podle informací těchto účetních struktur se řídí operační systém a když pak

ukládá na disk data, nepokouší se využívat v seznamu uvedené vadné bloky. V následujícím příkladu je naznačen celý postup:

```
$ badblocks /dev/fd0H1440 1440 > bad-blocks
$ fsck -t ext2 -l bad-blocks /dev/fd0H1440
Parallelizing fsck version 0.5a (5-Apr-94)
e2fsck 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Check reference counts.
Pass 5: Checking group summary information.

/dev/fd0H1440: ***** FILE system WAS MODIFIED *****
/dev/fd0H1440: 11/360 files, 63/1440 blocks
$
```

Je-li v seznamu vadných bloků uveden blok, který je již některým ze souborů využíván, program `e2fsck` se pokusí tento sektor přemístit na jiné místo disku. Když je blok skutečně vadný a nejde jenom o logickou chybu vzniklou při zápisu, bude obsah souboru s největší pravděpodobností poškozený.

Boj s fragmentací

Když se soubor ukládá na disk, nemůže být vždy zapsán do po sobě jdoucích bloků. Soubor, jenž není uložen do souvislé řady za sebou jdoucích bloků, je *fragmentovaný*. Načtení takového souboru pak trvá déle, protože čtecí hlava disku se musí při čtení víc pohybovat. Proto je žádoucí zamezit fragmentaci souborů, i když pro systémy s velkou vyrovnávací pamětí a dopředným čtením nepředstavuje až tak závažnou komplikaci.

Souborový systém ext2 se snaží udržet fragmentaci souborů na minimu. I v případě, že všechny bloky jednotlivých souborů nelze uložit do sektorů, jež jdou po sobě, ukládá je tak, aby byly co nejvíce pohromadě. Systém souborů ext2 navíc vždy efektivně alokuje volné sektory, které jsou nejblíže ke zbylým blokům ukládaného souboru. Používáte-li proto systémy souborů ext2, nemusíte se o fragmentaci příliš starat. Přesto existuje program **defrag**, jenž umí defragmentovat tento souborový systém[24].

Pro systém MS-DOS existuje celá řada defragmentačních programů. Ty přesouvají bloky v souborovém systému tak, aby fragmentaci odstranily. V ostatních souborových systémech se musí defragmentace dělat tak, že se vytvoří záloha souborového systému, který se znovu vytvoří a ze zálohy se obnoví soubory. Doporučení zálohovat souborový systém před jeho defragmentací se týká všech souborových systémů, protože v průběhu procesu defragmentace může dojít k poškození systému souborů i dalším chybám.

Další nástroje pro všechny souborové systémy

Existují i některé další nástroje užitečné pro správu souborových systémů. Program **df** ukazuje volný diskový prostor v jednom či několika souborových systémech. Program **du** ukazuje, kolik diskového prostoru zabírá adresář a všechny soubory v něm uložené. Lze jej s výhodou využít při „honu“ na uživatele, kteří zabírají svými (mnohdy zbytečnými) soubory nejvíc místa na disku.

Program **sync** zapíše všechny neuložené bloky z vyrovnávací paměti (viz část 7.6) na disk. Jen zřídkakdy je potřeba zadávat jej ručně, automaticky to totiž dělá démon **update**. Má to význam

[24] http://www.go.dlr.de/linux/src/defrag-0.73.tar.gz

především v případě nečekaných událostí, například když je proces **update** nebo jeho pomocný proces **bdflush** nečekaně ukončen, nebo v případě, že musíte *ihned* vypnout napájení a nemůžete čekat, než se opět spustí program **update**. Opět upozorňujeme na manuálové stránky. **man** je v Linuxu váš nejlepší přítel. Užitečný je také jeho bratranec **apropos** pro případ, že byste nevěděli, jaký příkaz použít.

Další nástroje pro souborový systém ext2

Kromě programu, kterým se tento systém souborů vytváří (**mke2fs**), a programu pro kontrolu jeho integrity (**e2fsck**), jež jsou přístupné přímo z příkazové řádky, případně přes koncové programy nezávislé na typu souborového systému, existují pro souborový systém ext2 některé další nástroje, jež mohou být při správě systému užitečné.

Program **tune2fs** umí upravit parametry souborového systému. Uvedeme některé z těch významnějších parametrů:

- Maximální počet připojení, po němž program **e2fsck** provede kontrolu souborového systému bez ohledu na to, že je nastaven příznak korektního odpojení. U systémů, které jsou určeny k vývoji nebo testování, je rozumné tento limit snížit.
- Maximální čas mezi kontrolami integrity. Příkaz **e2fsck** hlídá maximální periodu mezi dvěma kontrolami, a provede opět kontrolu i v případě, že je nastaven příznak korektního vypnutí a souborový systém nebyl připojen vícekrát, než je povoleno. Opakované kontroly lze zakázat.
- Počet bloků vyhrazených pro superuživatele. Souborový systém ext2 rezervuje některé bloky pro superuživatele. Když se pak souborový systém jako celek zaplní, není potřeba nic mazat a systém lze v omezené míře spravovat. Rezervovaný počet bloků je implicitně nastaven na 5 %, což u většiny disků stačí k tomu, aby se zamezilo jejich přeplnění. V případě disket nemá tato rezervace smysl.

Více informací najdete v manuálové stránce programu **tune2fs**.

Program **dumpe2fs** vypisuje informace o souborovém systému typu ext2, většinou z jeho superbloku. Na obrázku 5.5 je příklad výstupu tohoto programu. Některé z těchto informací jsou ryze technické a vyžadují hlubší pochopení problematiky fungování tohoto souborového systému. Většina údajů je ale snadno pochopitelná i pro správce – laiky.

```
dumpe2fs 0.5b, 11-Mar-95 for EXT2 FS 0.5a, 94/10/23
Filesystem magic number: 0xEF53
Filesystem state: clean
Errors behavior: Continue
Inode count: 360
Block count: 1440
Reserved block count: 72
Free blocks: 1133
Free inodes: 326
First block: 1
Block size: 1024
Fragment size: 1024
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 360
Last mount time: Tue Aug 8 01:52:52 1995
Last write time: Tue Aug 8 01:53:28 1995
```

```
Mount count: 3
Maximum mount count: 20
Last checked: Tue Aug 8 01:06:31 1995
Check interval: 0
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
Group 0:
Block bitmap at 3, Inode bitmap at 4, Inode table at 5
1133 free blocks, 326 free inodes, 2 directories
Free blocks: 307-1439
Free inodes: 35-360
```

Obrázek 5.5 – Příklad výstupu programu dumpe2fs

Program **debugfs** je nástroj pro ladění souborového systému. Umožňuje přímý přístup k strukturám dat souborového systému uloženým na disku, a lze jej proto použít při opravách disků poškozených natolik, že to nezvládne program **fsck**. Lze jej též využít k obnově smazaných souborů. Když ale chcete použít program **debugfs**, je velmi důležité, abyste skutečně věděli, co děláte. V případě, že zcela neporozumíte některé jeho funkci, se totiž může stát, že všechna svá data zničíte.

Programy **dump** a **restore** lze použít při zálohování souborového systému ext2. Jsou to specifické verze tradičních nástrojů pro zálohování používaných v systému Unix, určené speciálně pro systém souborů ext2. Víc informací o zálohování najdete v kapitole 12.

Disky bez souborových systémů

Ne na všech discích nebo diskových oblastech se vytváří souborové systémy. Například disková odkládací oblast nebude používat žádný souborový systém. Dalším příkladem jsou diskety, jejichž mechaniky se mnohdy používají pro emulaci páskové jednotky. Program **tar** nebo jiný archivační nástroj pak zapisuje přímo na „syrový“ disk bez souborového systému. Zaváděcí diskety systému Linux rovněž nemají souborový systém, pouze holé jádro systému.

To, že se na disku (disketě) nevytvoří souborový systém, má výhodu v tom, že lze využít větší část kapacity disku, protože každý souborový systém má vždy nějakou režii. Navíc lze takto dosáhnout větší kompatibility disků s jinými operačními systémy, například soubor s formátem, jenž používá program **tar**, má stejnou strukturu ve všech operačních systémech, kdežto souborové systémy samotné jsou ve většině operačních systémů různé. Další výhodou je, že disky bez souborových systémů lze v případě potřeby použít velmi rychle (odpadá operace vytváření a validace souborového systému). Zaváděcí diskety Linuxu také nutně nemusí obsahovat souborový systém, ačkoliv to možné je.

Dalším z důvodů proč používat „syrová“ zařízení je možnost vytvořit přesný zrcadlový obraz – kopii disku. Když například disk obsahuje částečně poškozený souborový systém, je vhodné předtím, než se pokusíte chybu opravit, udělat přesnou kopii poškozeného disku. Pak není problém začít znova v případě, že při neúspěšném pokusu opravit chybu poškodíte systém souborů ještě víc. Jedním ze způsobů, jak zrcadlovou kopii disku udělat, je použít program **dd**:

```
$ dd if=/dev/fd0H1440 of=floppy-image
2880+0 records in
2880+0 records out
$ dd if=floppy-image of=/dev/fd0H1440
2880+0 records in
```

```
2880+0 records out
$
```

První příkaz `dd` uloží přesný obraz diskety do souboru `floppy-image`, druhý zapíše tento obraz na další disketu. (Samozřejmě se předpokládá, že uživatel před zadáním druhého příkazu diskety v mechanice vyměnil. Jinak by byly tyto příkazy k ničemu.)

Přidělování diskového prostoru

Způsoby rozdělování disku na diskové oblasti

Rozdělit disk na diskové oblasti tím nejlepším možným způsobem není vůbec jednoduché. A co je nejhorší – neexistuje univerzálně správný způsob, jak toho dosáhnout. Celý problém totiž komplikuje příliš mnoho různých faktorů.

„Tradiční" variantou je mít (relativně) malý kořenový souborový systém, jenž obsahuje adresáře `/bin`, `/etc`, `/dev`, `/lib`, `/tmp` a další programy a konfigurační soubory, které jsou potřeba k tomu, aby se systém spustil a běžel. Vše, co je třeba k tomu, aby mohl být systém uveden do chodu, je právě kořenový souborový systém (na vlastní diskové oblasti nebo na samostatném disku). Důvodem tohoto „tradičního" schématu je fakt, že když je kořenový svazek malý a méně často používaný, je menší pravděpodobnost, že se v případě havárie systému poškodí. S takovýmto uspořádáním je také jednodušší odstranit případné problémy způsobené havárií systému. V dalším kroku se pak vytvoří samostatné diskové oblasti (nebo použijí další disky) pro stromovou strukturu svazku `/usr`, domovské adresáře uživatelů (nejčastěji pod adresářem `/home`) a pro odkládací prostor. Tím, že se vyčlení vlastní disková oblast (disk) domovským adresářům, v nichž jsou uloženy soubory jednotlivých uživatelů, se zjednoduší zálohování, protože programy, které jsou uloženy v adresáři `/usr`, obvykle není potřeba zálohovat. V síťovém prostředí je navíc možné adresář `/usr` sdílet mezi několika počítači (například využitím NFS) a tím snížit celkovou potřebu diskového prostoru. Ta by jinak dosahovala několika desítek či stovek megabajtů, násobeno počtem stanic v síti.

Problém několika samostatných diskových oblastí je v tom, že rozdělují celkové množství volného diskového prostoru na mnoho malých částí. V současnosti, kdy jsou disky a (snad) i operační systémy spolehlivější, stále více uživatelů preferuje možnost mít jenom jednu oblast, ve které jsou uloženy všechny soubory. Na druhou stranu zálohování (a obnovování) malých diskových oblastí je ale méně pracné.

U malých pevných disků (předpokládá se, že neděláte zrovna něco jako vývoj jádra systému) je obvykle nejlepší mít jenom jednu diskovou oblast. U velkých pevných disků je pro změnu výhodnější rozdělit je na několik větších oblastí, pouze pro případ, že by se stalo něco, s čím jste při instalaci systému nepočítali. (Uvědomte si ale, že pojmy „malý" a „velký" se zde používají v relativním smyslu, jedině vaše konkrétní potřeby diskového prostoru rozhodnou o tom, kde bude ležet jejich hranice.)

Máte-li k dispozici několik disků, je vhodné umístit kořenový souborový systém (včetně adresáře `/usr`) na jeden a domovské adresáře uživatelů na druhý disk.

Je dobré připravit se na malé „experimentování" s různými způsoby rozdělení disku na oblasti – kdykoliv, ne pouze při první instalaci systému. Je s tím sice dost práce, protože nezbytnou podmínkou je opakovaná instalace systému poté, co se některý pokus nezdaří[25]. Nicméně je to jediný způsob, jak si ověřit správnost rozdělení disku.

[25] To už není pravda, protože existují způsoby, jak přesouvat diskové oddíly a připojovací body bez reinstalace systému, nicméně (momentálně) je to mimo záběr této příručky. Pokud máte v této oblasti zkušenosti a znalosti, co kdybyste je sepsali a ušetřili mi práci?

Nároky na diskový prostor

Distribuce Linuxu, kterou budete instalovat, vám obvykle nějakým způsobem sdělí, kolik diskového prostoru je potřeba pro různé konfigurace operačního systému. Programy, které se instalují dodatečně, se budou většinou chovat stejně. Díky tomu si můžete udělat představu o nárocích na diskový prostor a naplánovat si jeho rozdělení. Měli byste se ale připravit i na budoucnost a vyhradit si nějaké místo navíc pro věci, na které si vzpomenete později.

Prostor, který byste měli vyčlenit pro soubory uživatelů systému, závisí na tom, co budou dělat. Většina lidí totiž obvykle spotřebuje tolik diskového prostoru, kolik je jenom možné. Nicméně množství, které jim skutečně stačí, je velmi různé. Někteří uživatelé vystačí s psaním textů a spokojeně přežijí i s několika megabajty. Jiní dělají složitou editaci grafických souborů a budou potřebovat gigabajty volného prostoru.

Mimochodem, když budete při odhadech nároků na diskový prostor srovnávat velikosti souborů v kilobajtech nebo megabajtech a diskový prostor v megabajtech, uvědomte si, že tyto dvě jednotky mohou být různé. Někteří výrobci disků rádi tvrdí, že kilobajt je 1 000 bajtů a megabajt je 1 000 kilobajtů, kdežto zbytek počítačového světa používá pro oba koeficienty číslo 1 024. Proto váš 345MB pevný disk bude mít ve skutečnosti pouze 330 MB.

O přidělování odkládacího prostoru hovoříme v části 7.5.

Příklady rozvržení diskového prostoru

Autor manuálu dlouho používal 109MB pevný disk. V současné době má k dispozici pevný disk o velikosti 330 MB. V dalším textu bude vysvětleno, jak a proč byly tyto disky rozděleny na jednotlivé diskové oblasti.

Disk o velikosti 109 MB byl velmi často „přerozdělován“ mnoha různými způsoby podle toho, jak se měnily konkrétní potřeby a používané operační systémy. Popíšeme dva typické scénáře. Při prvním z nich byly na jednom disku instalovány operační systémy MS-DOS a Linux. První diskový oblast o velikosti asi 20 MB byla vyhrazena pro systém MS-DOS, některý z kompilátorů jazyka C, textový editor, několik dalších utilit a rozpracovaný program. Několik megabajtů volného diskového prostoru zbylo proto, aby nevznikaly pocity klaustrofobie. Odkládacímu prostoru systému Linux bylo vyhrazeno 10 MB na samostatné diskové oblasti a zbytek, tedy 79 MB, na další diskové oblasti byl vyčleněn pro soubory operačního systému Linux. Rozdělovat takovýto prostor na samostatné oblasti pro kořenový souborový systém, `/usr` a domovské adresáře `/home` nemá praktický význam.

Když pak nebylo třeba systému MS-DOS, změnil jsem rozdělení disku tak, že odkládací prostor používal 12 MB a zbytek byl opět jeden souborový systém.

Disk o velikosti 330 MB lze rozdělit na několik diskových oblastí následujícím způsobem:

5 MB	kořenový souborový systém
10 MB	odkládací oblast
180 MB	systém `/usr`
120 MB	systém `/home`
15 MB	nevyužitá oblast

Neobsazenou diskovou oblast lze využívat na různé experimenty, při nichž je potřeba mít samostatný diskový segment, například při testování různých distribucí Linuxu, nebo srovnávání rych-

losti souborových systémů a podobně. Jinak lze neobsazenou diskovou oblast rovněž využít jako odkládací prostor (hlavně v případě, že máte rádi hodně otevřených oken)[26].

Zvětšování diskového prostoru pro Linux

Zvětšení diskového prostoru vyhrazeného pro Linux je jednoduché. Především v případě, že se instalují nové pevné disky (popis instalace disků jde ale nad rámec této knihy). Je-li to nutné, je potřeba disky zformátovat. Pak se podle výše uvedeného postupu vytvoří diskové oblasti a souborový systém a přidají se odpovídající řádky do souboru `/etc/fstab` kvůli tomu, aby se nové disky připojovaly automaticky.

Tipy, jak ušetřit místo na disku

Nejlepším způsobem, jak ušetřit diskový prostor, je vyvarovat se instalování nepotřebných programů. Většina distribucí Linuxu při instalaci nabízí možnost výběru balíků programů, které se mají instalovat. Podle této nabídky byste měli být schopni analyzovat své potřeby a pak pravděpodobně zjistíte, že většinu z nich nebudete potřebovat. Tím se ušetří hodně místa na disku, protože některé z programů a aplikačních balíků jsou dost objemné. I když zjistíte, že budete některou aplikaci nebo i celý balík programů potřebovat, určitě nebudete muset instalovat všechny jejich součásti. Obvykle není potřeba instalovat například on line dokumentaci, stejně jako některé ze souborů Elisp pro GNU verzi programu Emacs, některé z fontů pro X11 či některé knihovny programovacích jazyků.

V případě, že nemůžete některé balíky programů trvale odinstalovat, můžete využít kompresi. Komprimační programy jako **gzip** nebo **zip** zabalí (a rozbalí) jednotlivé soubory, případně celé skupiny souborů. Program **gzexe** transparentně komprimuje a dekomprimuje programy tak, že to uživatel v běžném provozu nepostřehne (nepoužívané programy se komprimují a rozbalí se, až když jsou potřeba). V současné době se testuje systém DouBle, který transparentně komprimuje všechny soubory v souborovém systému. (Znáte-li program Stacker pro MS-DOS, princip je podobný.)

[26] Tato část je poměrně neaktuální. Většinou se dnes používají disky o velikosti řádově gigabajty. I tak se můžete uvedeným popisem přibližně řídit (vynásobte údaje příslušným koeficientem tak, ať odpovídají vašemu hardwaru). Plánuje se aktualizace textu s ohledem na větší diskové kapacity.

KAPITOLA 6

Správa paměti

Minnet, jag har tappat mitt minne,
är jag svensk eller finne, kommer inte ihĺg[27].
(Bosse Österberg)

V této kapitole jsou popsány hlavní rysy systému správy paměti operačního systému Linux, tedy subsystémy virtuální paměti a diskové vyrovnávací paměti. Kapitola popisuje jejich účel, funkce a další podrobnosti, které správce systému musí mít na paměti.

Co je virtuální paměť?

Operační systém Linux podporuje *virtuální paměť*, to znamená, že používá disk jako rozšíření paměti RAM. Tím se efektivní velikost využitelné paměti odpovídajícím způsobem zvětší. Jádro systému zapisuje obsah právě nevyužívaných paměťových bloků na pevný disk a paměť se tak může využívat pro jiné účely. Když pak přijde požadavek na její původní obsah, bloky z disku se načtou zpět do paměti. To vše probíhá z pohledu uživatele zcela transparentně. Programy běžící pod Linuxem vidí pouze, že je k dispozici hodně paměti a nestarají se o to, že její část je občas uložená na disku. Přirozeně, čtení a zápis na pevný disk je pomalejší (zhruba o tři řády), než využití reálné paměti, takže programy neběží tak rychle. Část disku, která se využívá jako virtuální paměť, se nazývá *odkládací prostor*.

Linux může použít jako odkládací prostor nejen normální soubor uložený v souborovém systému, ale i diskové oblasti. Předností diskových oblastí je rychlost, výhodou odkládacího souboru je jednodušší možnost změny celkové velikosti odkládacího prostoru. Není přitom totiž potřeba měnit rozdělení celého pevného disku, kdy navíc hrozí nutnost kompletní reinstalace systému. Víte-li, jak velký odkládací prostor budete potřebovat, zvolte odkládací prostor na zvláštní diskovou oblast. Pokud si nároky nejste zcela jisti, zvolte nejdřív odkládací prostor v souboru. Když budete systém nějakou dobu používat, budete schopni odhadnout, kolik odkládacího prostoru skutečně potřebujete. Až budete mít ohledně předpokládané velikosti požadovaného odkládacího prostoru jasno, vytvoříte pro něj zvláštní diskovou oblast.

Měli byste rovněž vědět, že Linux umožňuje využívat současně několik odkládacích oblastí, případně několik odkládacích souborů. Takže když potřebujete pouze příležitostně větší množství odkládacího prostoru, je lepší (místo trvale vyhrazené rezervy) nastavit další soubor navíc.

Poznámka k terminologii používané v oblasti operačních systémů: v odborných kruzích se obvykle rozlišuje mezi odkládáním (anglicky *swapping*), tedy zapsáním celého procesu do odkládacího prostoru na disku, a stránkováním (anglicky *paging*), tedy zapisováním pouhých částí pevné velikosti (obvykle několik kilobajtů) najednou. Stránkování je obecně výkonnější a je to metoda, kterou používá i operační systém Linux. Tradiční terminologie systému Linux ale používá pojem odkládání (swapping)[28].

[27] Švédská pijácká písnička, přibližně: „Paměť, ztratil jsem paměť, jsem Švéd nebo Fin, nemůžu si vzpomenout."

[28] Poměrně zbytečně to značně dráždí část počítačových teoretiků.

Vytvoření odkládacího prostoru na disku

Odkládací soubor je běžný soubor a není pro jádro systému ničím zvláštní. Jediná vlastnost, která má pro jádro význam, je, že odkládací soubor nemá díry a že je připraven pro použití programem **mkswap**. Musí být navíc (z důvodů implementace) uložen na lokálním disku, takže nemůže být uložen v souborovém systému, který je připojen pomocí NFS.

Zmínka o dírách je důležitá. Odkládací soubor rezervuje určitý diskový prostor, takže jádro systému pak může rychle odložit stránku paměti bez toho, že by muselo absolvovat celou proceduru alokace diskového prostoru, která se používá pro běžný soubor. Jádro využívá pouze ty sektory, které byly odkládacímu souboru skutečně přiděleny. Protože díra v souboru znamená, že tomuto místu souboru nejsou přiděleny žádné diskové sektory, nemohlo by je jádro dost dobře využít.

Jeden ze způsobů, kterým lze vytvořit odkládací soubor bez prázdných míst, je:

```
$ dd if=/dev/zero of=/extra-swap bs=1024 count=1024
1024+0 records in
1024+0 records out
$
```

kde /extra-swap je jméno odkládacího souboru, jehož velikost je uvedena za parametrem count=. Ideální je zvolit velikost jako násobek 4, protože jádro systému zapisuje do odkládacího prostoru *stránky paměti*, které jsou 4 kilobajty velké. Nebude-li velikost násobkem 4, může být posledních pár kilobajtů nevyužitých.

Samostatná odkládací disková oblast rovněž není ničím neobvyklým. Vytvoří se stejně, jako každá jiná disková oblast. Jediným rozdílem je, že se používá jako holé zařízení, tedy bez souborového systému. Je dobré označit ji jako typ 82 („Linux swap“). I když to jádro systému striktně nevyžaduje, vnese to do seznamu diskových oblastí řád.

Poté, co vytvoříte diskovou oblast pro odkládací prostor nebo odkládací soubor, je potřeba zapsat na jejich začátek signaturu, která obsahuje některé administrativní informace, a s níž jádro pracuje. Provede se to příkazem **mkswap** tímto způsobem:

```
$ mkswap /extra-swap 1024
Setting up swapspace, size = 1044480 bytes
$
```

Odkládací prostor zatím není využitý. Sice existuje, ale jádro systému jej jako virtuální paměť zatím nezná.

Při zadávání příkazu `mkswap` byste měli být velice opatrní, protože program nekontroluje, zda se soubor nebo disková oblast nevyužívá k jiným účelům. *Příkazem `mkswap` proto můžete lehce přepsat důležité soubory nebo celé diskové oblasti!* Naštěstí budete tento příkaz potřebovat pouze když instalujete operační systém.

Systém správy paměti v Linuxu omezuje velikost swapovacího souboru na přibližně 127 MB (z různých technických příčin je skutečná maximální velikost (4 096-10)*8*4 096 = 133 890 048 bajtů = 127,6875 MB). Můžete nicméně vytvořit až 8 swapovacích souborů a získat tak celkem 1 GB swapovacího prostoru[29].

Toto omezení už momentálně neplatí, tato kapitola je naplánována k předělání Opravdu Brzo (tm). Pro nová jádra a nové verze programu `mkswap` závisí limity na architektuře. U architektury i386 a kompatibilních je limit 2 GB, u jiných architektur je jiný. Podrobnosti viz manuálová stránka mkswap(8).

[29] Gigabajt sem, gigabajt tam, za chvíli se začneme bavit o fyzické paměti.

Využívání odkládacího prostoru

Využívání nově vytvořeného odkládacího prostoru lze zahájit příkazem **swapon**. Příkaz sdělí jádru systému, že odkládací prostor, jehož úplná cesta se zadává jako parametr příkazu, lze od této chvíle používat. Takže když budete chtít začít využívat dočasný soubor jako odkládací prostor, zadáte tento příkaz:

```
$ swapon /extra-swap
$
```

Odkládací prostory lze využívat automaticky poté, co budou zapsány v souboru `/etc/fstab`, například:

```
/dev/hda8   none   swap   sw   0   0
/swapfile   none   swap   sw   0   0
```

Spouštěcí skripty vykonávají příkaz `swapon -a`, jenž zahájí odkládání do všech odkládacích prostorů uvedených v souboru `/etc/fstab`. Takže příkaz `swapon` se obvykle používá jenom když je potřeba použít odkládací prostor navíc.

Příkazem `free` lze monitorovat využívání odkládacích prostorů. Příkaz zobrazí celkové množství odkládacího prostoru, který je v systému využíván:

```
$ free
             total       used       free     shared    buffers
Mem:         15152      14896        256      12404       2528
-/+ buffers:            12368       2784
Swap:        32452       6684      25768
$
```

V prvním řádku výstupu (`Mem:`) se zobrazuje velikost fyzické paměti. Sloupec `total` neukazuje velikost fyzické paměti, kterou využívá jádro systému, ta má obvykle asi jeden megabajt. Ve sloupci `used` je zobrazeno množství využívané paměti (v druhém řádku je vynechána velikost vyrovnávací paměti). Sloupec `free` udává celkové množství nevyužité paměti. Sloupec `shared` ukazuje paměť sdílenou několika procesy – platí čím více, tím lépe. Ve sloupci `buffers` je zobrazena aktuální velikost vyrovnávací paměti.

V posledním řádku (`Swap:`) jsou analogické informace pro odkládací prostor. Když jsou v tomto řádku samé nuly, není odkládací prostor systému aktivovaný.

Stejné informace lze získat příkazem `top`, nebo z údajů v souborovém systému `proc`, přesněji v souboru `/proc/meminfo`. Obvykle je ale dost obtížné získat informace o využití jednoho konkrétního odkládacího prostoru.

Odkládací prostor lze vyřadit z činnosti příkazem `swapoff`. Příkaz pravděpodobně využijete pouze pro vyřazení dočasných odkládacích prostorů. Všechny stránky, které jsou uloženy v odkládacím prostoru, se po zadání příkazu `swapoff` nejdříve načtou do paměti. Když není dostatek fyzické paměti, do které by se načetly, budou uloženy do některého z jiných odkládacích prostorů. Když není ani dostatek virtuální paměti na odložení všech načítaných stránek odkládacího prostoru, jenž má být vyřazen z činnosti, začnou problémy. Po delší době by se operační systém měl zotavit, ale mezitím bude prakticky nepoužitelný. Proto byste měli předtím, než vyřadíte některý odkládací prostor z činnosti, zkontrolovat (například příkazem `free`), jestli máte dostatek volné paměti.

Všechny odkládací prostory, které se aktivují automaticky příkazem `swapon -a`, lze vyřadit z činnosti příkazem `swapoff -a`. Příkaz vyřadí z činnosti odkládací prostory uvedené v souboru `/etc/fstab`. Odkládací prostory přidané ručně zůstanou nadále v činnosti.

Někdy mohou nastat situace, že se využívá příliš mnoho odkládacího prostoru, i když má systém dostatek volné fyzické paměti. Může se to stát například v situaci, kdy jsou v jednom okamžiku velké nároky na virtuální paměť, ale po chvíli je ukončen některý větší proces, který využívá větší část fyzické paměti, a ten paměť uvolní. Odložená data se ale nenačítají do paměti automaticky a zůstanou uložena na disku až do doby, než budou potřeba. Fyzická paměť by tak mohla zůstat dost dlouho volná, nevyužitá. Není potřeba se tím znepokojovat, ale je dobré vědět, co se v systému děje.

Sdílení odkládacího prostoru s jinými operačními systémy

Virtuální paměť používá mnoho operačních systémů. Vzhledem k tomu, že každý ze systémů využívá svůj odkládací prostor jenom když běží (tedy nikdy ne několik systémů současně), odkládací prostory ostatních operačních systémů pouze zabírají místo na disku. Pro operační systémy by bylo efektivnější sdílet jediný odkládací prostor. I to je možné, ale je potřeba si s tím pohrát. V textu *Tips-HOWTO* je uvedeno několik praktických rad, jak implementovat sdílení odkládacího prostoru.

Přidělování odkládacího prostoru

Možná někdy narazíte na radu, že máte přidělovat dvakrát tolik odkládacího prostoru, než máte fyzické paměti. To je ovšem pouhá pověra. Uvádíme proto správný postup:

- Zkuste odhadnout, jaké budete mít nároky na paměť, tedy největší množství paměti, které budete pravděpodobně v jednom okamžiku potřebovat. To je dané součtem paměťových nároků všech programů, které poběží současně.

  Když například chcete, aby běželo grafické uživatelské rozhraní X Window, měli byste mu přidělit asi 8 MB paměti. Kompilátor gcc vyžaduje několik megabajtů (kompilace některých souborů by mohla mít neobvykle velké nároky na paměť, jež by mohly dosáhnout až několika desítek megabajtů, ale běžně by měly stačit zhruba čtyři megabajty). Jádro samotné bude využívat asi jeden megabajt paměti, běžné interprety příkazů a jiné menší utility několik stovek kilobajtů (řekněme asi jeden megabajt dohromady). Není potřeba být v odhadech úplně přesný, stačí udělat velmi hrubý odhad, ale ten by měl být spíše pesimistický.

  Je důležité si uvědomit, že když bude systém současně používat více uživatelů, budou paměť RAM potřebovat všichni. Avšak budou-li současně ten samý program používat dva uživatelé, celková potřeba paměti obvykle nebude dvojnásobná, protože stránky kódu a sdílené knihovny budou v paměti pouze jednou.

  Pro správný odhad nároků na paměť jsou užitečné příkazy **free** a **ps**.

- K odhadu podle předchozího kroku připočítejte nějakou rezervu. To proto, že odhady paměťových nároků programů budou velmi pravděpodobně nedostatečné – je možné, že na některé aplikace, které budete chtít používat, zapomenete. Takto budete mít jistotu, že pro tento případ máte nějaké to místo navíc. Mělo by stačit pár megabajtů. (Je lepší vyčlenit příliš mnoho, než moc málo odkládacího prostoru. Ale není potřeba to přehánět a alokovat celý disk, protože nevyužitý odkládací prostor zbytečně zabírá místo. V dalších částech

této kapitoly bude uvedeno, jak přidat další odkládací prostor.) Vzhledem k tomu, že se lépe počítá s celými čísly, je dobré hodnoty zaokrouhlovat směrem nahoru, řádově na další celé megabajty.

- Na základě těchto výpočtů budete vědět, kolik paměti budete celkem potřebovat. Takže když odečtete velikost fyzické paměti od celkových nároků na paměť, dozvíte se, kolik odkládacího prostoru musíte celkem vyčlenit. (U některých verzí Unixu se musí vyčlenit i paměťový prostor pro obraz fyzické paměti, to znamená, že velikost paměti vypočítaná podle kroku 2 představuje skutečné nároky na odkládací prostor a neodečítá se velikost fyzické paměti.)
- Je-li vypočítaná velikost odkládacího prostoru o hodně větší než dostupná fyzická paměť (více než dvakrát), měli byste raději více investovat do fyzické paměti. Jinak bude výkon systému příliš nízký.

Vždy je dobré mít v systému alespoň nějaký odkládací prostor, i když podle výpočtů žádný nepotřebujete. Linux používá odkládací prostor poněkud agresivněji, snaží se mít tolik volné fyzické paměti, kolik je jenom možné. Linux navíc odkládá paměťové stránky, které se nepoužívají, i když se fyzická paměť zatím nevyužívá. Tím se totiž zamezí čekání na odložení v případě akutní potřeby, data se odkládají dříve, v době, kdy je disk jinak nevyužitý.

Odkládací prostor lze rozdělit mezi několik disků. To může v některých případech zlepšit výkon systému, jenž v tomto směru závisí na relativních rychlostech disků a jejich přístupových modelech. Lze samozřejmě experimentovat s několika variantami, ale mějte vždy na paměti to, že je velmi lehké u takovýchto pokusů pochybit. Neměli byste také věřit tvrzením, že některá z možností je lepší než ostatní, protože to nemusí být vždy pravda.

Vyrovnávací paměť

Čtení z disku[30] je ve srovnání s přístupem k fyzické paměti velmi pomalé. Navíc se v běžném provozu velmi často z disku načítají stejná data několikrát během relativně krátkých časových intervalů. Například v případě elektronické pošty se nejdříve musí načíst došlá zpráva; když na ni chcete odpovědět, načte se ten samý dopis do editoru; pak stejná data načte i poštovní program, který je kopíruje do souboru s došlou, případně odeslanou poštou a podobně. Nebo si zkuste představit, jak často se může zadávat příkaz **ls** na systému s mnoha uživateli. Jediným načtením informací z disku a jejich uložením do paměti do doby až je nebude potřeba, lze zrychlit všechny operace čtení z disku s výjimkou prvního čtení. Paměť vyhrazená pro tyto účely, tedy pro ukládání z disku načítaných a na disk zapisovaných dat, se nazývá *vyrovnávací paměť*.

Paměť je naneštěstí omezený a navíc vzácný systémový prostředek, takže vyrovnávací paměť nemůže být obvykle dost velká (nemohou v ní být uložena úplně všechna data, která by chtěl někdo používat). Když se vyrovnávací paměť zaplní, data, jež se nepoužívala nejdéle, se zruší a takto uvolněná vyrovnávací paměť se využije na ukládání nových dat.

Vyrovnávací paměť funguje i při zápisu na disk. Jednak proto, že data, která se zapisují na disk, se velmi často brzo opakovaně načítají (například zdrojový kód nejprve uložíme do souboru a pak jej opět načítá překladač), takže je výhodné uložit data zapisovaná na disk i do vyrovnávací paměti. Druhou výhodou je to, že uložením dat do vyrovnávací paměti (aniž by se okamžitě zapsala na disk) se zlepší odezva programu, jenž data zapisuje. Zápis dat na disk pak může probíhat na pozadí bez toho, že by se tím zpomalovaly ostatní programy.

Diskové vyrovnávací paměti používá většina operačních systémů (i když je možné, že se jim říká nějak jinak). Ne všechny ale pracují podle výše uvedených principů. Některé fungují jako *write-*

[30] Samozřejmě s výjimkou ramdisku, ze zjevných důvodů.

through: data se zapisují na disk ihned (ovšem přirozeně zůstanou rovněž uložena ve vyrovnávací paměti). Je-li zápis odložen na pozdější dobu, označují se tyto vyrovnávací paměti jako *write-back*. Tento systém je samozřejmě efektivnější, je ale také o něco více náchylný k chybám. V případě havárie systému, případně výpadku proudu v nevhodném okamžiku, či vytažení diskety z disketové mechaniky předtím, než jsou data čekající ve vyrovnávací paměti na uložení zapsána, se změny uložené ve vyrovnávací paměti obvykle ztratí. Navíc by takováto událost mohla zapříčinit chyby v souborovém systému (je-li na disku či disketě vytvořen), jelikož mezi nezapsanými daty mohou být i důležité změny účetních informací samotného souborového systému.

Proto by se nikdy nemělo vypínat napájení počítače dřív, než správně proběhla procedura zastavení systému (viz kapitola 8). Rovněž by se neměla vytahovat disketa z mechaniky předtím, než byla odpojena příkazem **umount** (byla-li předtím připojená), případně předtím, než program, jenž přistupoval na disketu, signalizuje, že práci s disketovou mechanikou ukončil a než přestane svítit kontrolka mechaniky pružného disku. Příkaz **sync** vyprázdní vyrovnávací paměť, tedy uloží všechna nezapsaná data na disk. Lze jej použít vždy, když chcete obsah vyrovnávací paměti bezpečně uložit na disk. V tradičních systémech Unix existuje program **update**, který běží na pozadí a spouští příkaz **sync** každých 30 sekund. V těchto systémech proto není obvykle potřeba příkaz **sync** používat ručně. V systému Linux běží další démon **bdflush**, jenž dělá něco podobného jako program **sync**, ale častěji a ne v takovém rozsahu. Navíc se tímto způsobem vyhnete náhlému „zamrznutí“ systému, které může někdy příkaz **sync** při větším rozsahu vstupně-výstupních operací způsobit.

V systému Linux se program **bdflush** spouští příkazem **update**. V případě, že je démon **bdflush** z jakéhokoliv důvodu neočekávaně ukončen, jádro systému o tom podá varovné hlášení. Obvykle ale není důvod se ničeho obávat. Proces **bdflush** lze spustit ručně (příkazem **/sbin/update**).

Ve skutečnosti se do vyrovnávací paměti neukládají celé soubory, ale bloky, což jsou nejmenší jednotky při vstupně-výstupních diskových operacích (v Linuxu mají nejčastěji velikost 1 kB). Tímto způsobem se do vyrovnávací paměti ukládají i adresáře, superbloky, další účetní data souborového systému a data z disků bez souborových systémů.

O efektivitě vyrovnávací paměti prvotně rozhoduje její velikost. Malá vyrovnávací paměť je téměř zbytečná. Bude v ní uloženo tak málo dat, že se vyrovnávací paměť vždy vyprázdní předtím, než mohou být data opakovaně použita. Kritická velikost záleží na tom, jaké množství dat se z disků čte a na disky zapisuje a jak často se k těmto údajům opakovaně přistupuje. Jediným způsobem, jak to zjistit, je experimentovat.

Má-li vyrovnávací paměť pevnou velikost, není výhodné ji mít příliš velkou. To by mohlo kriticky zmenšit dostupnou systémovou paměť a zapříčinit časté odkládání (jež je rovněž pomalé). Aby byla reálná paměť využívána co nejefektivněji, Linux automaticky využívá veškerou volnou paměť RAM jako vyrovnávací paměť. Naopak, operační systém vyrovnávací paměť automaticky zmenšuje, když běžící programy požadují víc fyzické paměti.

O vyrovnávací paměť se v Linuxu nemusíte nijak starat, vše probíhá automaticky. Kromě dodržování korektních postupů vypínání počítače a odpojování médií si jí vůbec nemusíte všímat.

V systému Linux není potřeba dělat nic zvláštního pro to, aby bylo možné vyrovnávací paměť využívat. Systém ji totiž používá zcela automaticky. Kromě správného postupu při zastavení systému a vyjímání disket z mechaniky se prakticky o vyrovnávací paměť vůbec nemusíte starat.

Spouštění a zastavování systému

Start me up
Ah... you've got to... you've got to
Never, never never stop
Start it up
Ah... start it up, never, never, never
You make a grown man cry,
you make a grown man cry
(Rolling Stones)

Tato kapitola popisuje, co se děje po tom, co je systém Linux spuštěn, a jak jej správně zastavit. Při nedodržení správných postupů může dojít k poškození nebo ztrátě souborů.

Zavádění a ukončení práce systému – přehled

Zapnutí počítače a následné zavedení operačního systému se označuje jako *bootování*. Ten termín původně vznikl z anglické fráze „pull yourself up by your own bootstraps", tedy vytáhnout sebe sama za jazyk vlastních bot, což obrazně vystihuje proces, který probíhá při zapnutí počítače – nicméně realita je podstatně méně zábavná.

V průběhu zavádění počítač nejdříve nahraje krátký kód, takzvaný *zavaděč*, jenž pak zavede a spustí samotný operační systém. Zavaděč je obvykle uložen na předem určeném místě pevného disku nebo diskety. Důvodem pro rozdělení procedury zavádění systému do dvou kroků je to, že samotný operační systém je velký a složitý, kdežto samotný zavaděč je velmi krátký (má několik stovek bajtů). Tím se zamezí nežádoucímu komplikování firmwaru.

Různé typy počítačů provádí úvodní sekvence zavádění systému různě. Pokud jde o počítače třídy PC, ty (přesněji jejich systém BIOS) načítají první sektor pevného disku nebo diskety, kterému

se říká *zaváděcí sektor*. Zavaděč je uložen v tomto prvním – zaváděcím sektoru. Zavaděč pak načítá operační systém z jiného místa na disku, případně i z nějakého jiného média.

Poté, co se operační systém Linux zavede, inicializují se technické prostředky počítače a ovladače jednotlivých zařízení. Pak se spustí **Proces init**, který spouští další procesy, umožňující uživatelům přihlásit se do systému a pracovat v něm. O podrobnostech této části zaváděcího procesu se pojednává dále.

Aby bylo možné systém Linux zastavit, je nejdříve nutné požádat všechny procesy, aby ukončily činnost (zavřely všechny otevřené soubory, popřípadě zařídily další důležité věci – obrazně řečeno „uklidily" po sobě). Potom se odpojí souborové systémy, odkládací prostory a nakonec se na konzole objeví zpráva, že lze počítač vypnout. Jestli se nedodrží správný postup, mohou se stát (a obvykle se stanou) dost nepříjemné věci. Nejzávažnějším problémem je v tomto případě nevyprázdněná vyrovnávací disková paměť souborového systému. Všechna data ve vyrovnávací paměti se totiž ztratí, takže souborový systém na disku bude nekonzistentní a pravděpodobně nepoužitelný.

Zavádění podrobněji

Systém Linux lze zavést buď z disket, z pevného disku nebo z CD. Příslušná kapitola Průvodce instalací uvádí návod, jak systém instalovat všemi výše uvedenými způsoby.

Když počítač PC startuje, systém BIOS provádí různé testy a kontroluje, zda je vše v pořádku[31]. Až pak zahájí skutečné zavádění operačního systému. Vybere zaváděcí diskovou jednotku. Typicky první disketovou mechaniku (je-li v mechanice zasunuta disketa), jinak první pevný disk, pokud je v počítači nainstalován. Pořadí prohledávání lze konfigurovat. Poté se načte první sektor tohoto disku, kterému se říká *zaváděcí sektor*, u pevných disků se označuje jako *hlavní zaváděcí sektor*, protože na pevném disku může být několik diskových oblastí, každá se svým vlastním zaváděcím sektorem.

Zaváděcí sektor obsahuje krátký program (dostatečně krátký na to, aby se vešel do jednoho bloku), jehož úkolem je načíst z disku operační systém a spustit jej. Při zavádění systému z diskety obsahuje její zaváděcí sektor kód, který načte jenom prvních několik stovek bloků (v závislosti na aktuální velikosti jádra) do předem určeného místa v paměti. Na linuxové zaváděcí disketě totiž nebývá vytvořen souborový systém, je na ní uloženo jenom jádro systému v několika po sobě jdoucích sektorech, což proces zavádění systému značně zjednodušuje. Systém lze zavést rovněž z diskety se souborovým systémem, a to pomocí zavaděče systému Linux (anglicky LInux LOader, zkráceně LILO).

Když se systém zavádí z pevného disku, kód obsažený v zaváděcím sektoru disku si prohlédne tabulku diskových oblastí, která je v tomto sektoru také uložená. Pak zaváděcí kód identifikuje aktivní diskovou oblast (oblast, která je označená jako zaváděcí), načte zaváděcí sektor této diskové oblasti a spustí kód v něm uložený. Kód v zaváděcím sektoru diskové oblasti pak dělá v podstatě to samé, co kód obsažený v zaváděcím sektoru diskety. Načte jádro systému ze své diskové oblasti a spustí jej. Avšak v detailech se tyto procedury poněkud liší, protože obecně není výhodné mít zvláštní diskovou oblast čistě pro obraz jádra systému, proto kód v zaváděcím sektoru nemůže jednoduše sekvenčně číst data z disku, jako při zavádění systému z diskety. Existuje několik způsobů řešení tohoto problému. Nejběžnější možností je použít zavaděč operačního systému Linux LILO. (Další detaily tohoto postupu nejsou v této chvíli podstatné. Více informací najdete v dokumentaci zavaděče LILO, která je v tomto směru dokonalejší.)

Když se zavádí operační systém pomocí zavaděče LILO, pokračuje se načtením a spuštěním implicitního jádra. Je také možné nakonfigurovat LILO tak, aby zavedl některý z více obrazů jádra

[31] Tato procedura se označuje jako *Power On Self Test*, zkráceně POST.

systému, nebo i jiný operační systém než Linux. Uživatel systému si tak při zavádění může vybrat, které jádro, případně operační systém, se implicitně zavede při spuštění počítače. Zavaděč LILO lze nakonfigurovat i tak, že při zmáčknutí kláves <Alt>, <Shift> nebo <Control> v okamžiku zavádění systému (tedy při spouštění LILO) se nebude zavádět systém ihned. Místo toho se zavaděč zeptá, který operační systém se bude zavádět. LILO lze nastavit i tak, že se bude při zavádění systému ptát na požadovaný systém, ale s volitelným časovým prodlením, po kterém se zavede implicitně určené jádro.

Zavaděč LILO rovněž umožňuje předat jádru systému řádkové parametry, které se zadávají za jménem jádra nebo operačního systému, jenž se zavádí.

Zavádění systému z diskety i z pevného disku má své výhody. Obecně je zavádění z disku příjemnější, uživatel je ušetřen zbytečných nepříjemností v případě, že v disketách „zcela náhodou" zavládne nepořádek. Kromě toho je zavádění z disku rychlejší. Naopak konfigurace pro zavedení operačního systému z disku může být složitější. Proto mnoho uživatelů dává v první fázi přednost zavádění systému z diskety a až pak, když bude nainstalovaný systém fungovat, doinstalují zavaděč LILO a zavádějí systém z pevného disku.

Jakmile se jádro systému načte do paměti (ať už to znamená cokoliv) a dojde k jeho skutečnému spuštění, stanou se přibližně následující věci:

- Jádro Linuxu se instaluje v komprimovaném tvaru, takže se nejprve samo dekomprimuje. Stará se o to krátký program, jenž je obsažen v začátku obrazu jádra.
- Rozezná-li systém kartu Super VGA, která má nějaké zvláštní textové režimy (jako například 100 sloupců na 40 řádků), zeptá se vás, který z režimů budete používat. V průběhu kompilace jádra systému lze videorežim nastavit – pak se systém při startu nedotazuje. Stejného efektu lze dosáhnout konfigurací zavaděče systému LILO nebo příkazem **rdev**.
- V dalším kroku jádro zkontroluje, jaké další hardwarové komponenty jsou k dispozici (pevné disky, diskety, síťové adaptéry a podobně) a odpovídajícím způsobem nastaví některé ze svých ovladačů zařízení. V průběhu této „inventury" vypisuje jádro zprávy o tom, která zařízení byla nalezena. Například při zavádění systému, jejž používá autor, se vypisují tato hlášení:

```
LILO boot:
Loading linux.
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 3.94 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
lp_init: lp1 exists (0), using polling driver
Memory: 7332k/8192k available (300k kernel code, 384k reserved, 176k da-
ta)
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M
Loopback device init
Warning WD8013 board not found at i/o = 280.
Math coprocessor using irq13 error reporting.
Partition check:
  hda: hda1 hda2 hda3
VFS: Mounted root (ext filesystem).
Linux version 0.99.pl9-1 (root@haven) 05/01/93 14:12:20
```

Přesný formát výstupů se na různých systémech liší, a to v závislosti na hardwarové konfiguraci výpočetního systému, verzi operačního systému a jeho konkrétním nastavení.

- Potom se jádro Linuxu pokusí připojit kořenový svazek. Přípojné místo lze nastavit při kompilaci jádra, později pak příkazem **rdev**, případně zavaděčem LILO. Typ souborového systému je detekován automaticky. Jestli připojení souborového systému selže (například proto, že jste při kompilaci jádra systému zapomněli uvést odpovídající ovladač souborového systému), jádro zpanikaří a systém se v tomto kroku zastaví (beztak mu nic jiného ani nezbývá).

  Kořenový svazek se obvykle připojuje pouze pro čtení (to lze nastavit stejným způsobem, jako místo připojení). Díky tomu se může provést kontrola souborového systému při jeho připojování. Není vhodné prověřovat systém souborů, jenž je připojen pro čtení i zápis.
- Poté spustí jádro systému na pozadí **Proces init**, jenž se nachází v adresáři `/sbin`. **Proces init** bude vždy procesem číslo 1 a jeho úkolem jsou různé operace spojené se startem systému. Přesný postup toho, co jádro systému v tomto kroku dělá, závisí na tom, jak je konfigurováno. Více informací uvádí kapitola 9. Jádro systému v této fázi přinejmenším spustí na pozadí některé nezbytné démony.
- **Proces init** pak přepne do víceuživatelského režimu a spustí procesy **getty** pro virtuální konzoly a sériové linky. Proces **getty** je program, který umožňuje uživatelům přihlásit se prostřednictvím virtuální konzoly nebo sériových terminálů do systému. **Proces init** může rovněž spouštět některé další programy, a to podle toho, jak je konfigurován.
- Poté je zavádění systémů ukončeno a systém normálně běží.

Podrobněji o zastavení systému

I při zastavování operačního systému Linux je důležité dodržovat správný postup. Pokud se správná procedura zastavení systému nedodrží, budou souborové systémy pravděpodobně poškozeny a obsah jednotlivých souborů může být promíchán. To proto, že operační systém využívá diskovou vyrovnávací paměť, jež nezapisuje změny na disk ihned, ale v určitých časových intervalech. To sice významně zvyšuje výkon systému, ale následkem toho je, že pokud se z ničeho nic vypne napájení ve chvíli, kdy vyrovnávací paměť obsahuje množství nezapsaných změn, mohou být data na disku nekonzistentní a souborový systém zcela nefunkční, protože se na disk zapsala jenom některá změněná data.

Dalším z argumentů proti tvrdému vypnutí počítače je to, že v systému ve víceuživatelském režimu může běžet hodně programů na pozadí. Vypnutí ze sítě by pak mohlo mít katastrofální důsledky. Tím, že se při zastavení systému dodržuje korektní postup, se zajistí, že všechny procesy běžící na pozadí svá data včas uloží.

Běh systému Linux se správně ukončí příkazem **shutdown**. Zadává se obvykle jedním ze dvou způsobů.

Když jste přihlášeni v systému jako jediný uživatel, je potřeba před zadáním příkazu **shutdown** ukončit všechny běžící programy, odhlásit se ze všech virtuálních konzolí a přihlásit se na jednu z nich jako superuživatel. Pokud už tak jste přihlášení, je rozumné přepnout se buď do kořenového adresáře nebo do svého domovského adresáře, aby se předešlo problémům při odpojení souborových systémů. Pak můžete zadat příkaz **shutdown -h now**. (Místo parametru `now` můžete zadat symbol plus a nějaké číslo, pak se zastavení systému zpozdí o zadaný počet minut – ve většině případů totiž nejste jediný uživatel systému.)

Druhou alternativou – je-li v systému přihlášeno více uživatelů – je použití příkazu **shutdown -h +*time message***, kde *time* je čas v minutách zbývající do zastavení systému a *message* je stručné sdělení důvodu tohoto opatření.

```
# shutdown -h +10 'Bude instalován nový disk. Systém by
> měl být opět spuštěn za tři hodiny.'
#
```

Takto můžete každého uživatele varovat, že systém bude za deset minut zastaven a že bude lepší se odpojit, než ztratit neuložená data. Varování se zobrazí na každém terminálu, na kterém je někdo přihlášený, včetně všech terminálů `xterm` systému X Window:

```
Broadcast message from root (ttyp0) Wed Aug 2 01:03:25 1995...

Bude instalován nový disk. Systém by měl být opět spuštěn za tři hodiny.

The system is going DOWN for system halt in 10 minutes !!
```

Varování se automaticky opakuje několik minut před zastavením systému v kratších a kratších intervalech, až stanovený čas vyprší.

Když se pak po určeném časovém prodlení rozjede procedura skutečného zastavení systému, odpojí se nejdříve všechny souborové systémy (kromě kořenového), uživatelské procesy (je-li někdo stále přihlášen) se ukončí, běžící démoni se zastaví, všechny připojené svazky se odpojí, obrazně řečeno – všechno ustane. Poté `Proces init` vypíše zprávu, že lze počítač vypnout. Až pak lze sáhnout na síťový vypínač.

V některých případech (ovšem zřídkakdy u správně nakonfigurovaného systému) není možné zastavit systém korektně. Například když jádro systému zpanikaří, havaruje a nefunguje správně, může být zcela nemožné zadat jakýkoliv další příkaz. Pochopitelně, v takovéto situaci je správné zastavení systému poněkud obtížné. Nezbývá než doufat, že se neuložené soubory až tak moc nepoškodí a vypnout napájení. Když nejsou potíže až tak vážné (řekněme, že někdo jenom sekerou rozsekl klávesnici vašeho terminálu) a jádro systému i program `update` stále normálně běží, je obvykle dobré pár minut počkat (dát tím programu `update` šanci vyprázdnit vyrovnávací paměti) a potom jednoduše vypnout proud.

Někteří uživatelé rádi používají při zastavení systému příkaz `sync` zadaný třikrát po sobě, pak počkají, než se ukončí diskové vstupně-výstupní operace a vypnou napájení. Neběží-li žádné programy, je tento postup téměř ekvivalentní zadání příkazu `shutdown` s tím rozdílem, že se neodpojí žádný ze souborových systémů, což ale může způsobit problémy s nastavením příznaku odpojení ukončení u souborových systémů ext2fs. Proto se metoda trojího zadání příkazu sync *nedoporučuje*.

(Pokud vás zajímá, *proč* zrovna třikrát – v raných dobách Unixu se všechny příkazy musely naťukat zvlášť, takže než to člověk udělal potřetí, bylo obvykle dost času na to, aby se ukončila většina diskových vstupně-výstupních operací.)

Znovuzavedení systému

Pod znovuzavedením operačního systému se rozumí jeho opakované zavedení, tedy jeho správné zastavení, vypnutí a opětovné zapnutí napájení počítače. Jednodušší cestou je žádost programu **shutdown** o znovuzavedení systému (místo jeho pouhého zastavení), a to použitím parametru **-r**, tedy například zadáním příkazu **shutdown -r now**.

Většina systémů Linux vykonává příkaz **shutdown -r now** i v případě, že se na systémové klávesnici současně zmáčknou klávesy Ctrl-Alt-Del. Tato trojkombinace obvykle vyvolá znovuzavedení operačního systému. Reakci na stisk kláves Ctrl-Alt-Del lze nastavit, a na víceuživatelském počítači by například bylo lepší před znovuzavedením systému povolit určité časové prodlení. Na-

opak systémy, které jsou fyzicky přístupné komukoliv, by bylo lepší nastavit tak, aby se při zmáčknutí kombinace kláves Ctrl-Alt-Del nedělo vůbec nic.

Jednouživatelský režim

Příkaz **shutdown** lze použít k přepnutí systému do jednouživatelského režimu. V něm se do systému nemůže přihlásit nikdo jiný než superuživatel, jenž může používat konzolu systému. Jednouživatelský mód lze s výhodou využít při plnění některých úkolů spojených se správou systému, které nelze dělat, pokud systém běží v normálním (víceuživatelském) režimu.

Záchranné zaváděcí diskety

Občas se stává, že není možné při zapnutí počítače zavést systém z pevného disku. Například tím, že uděláte chybu při nastavování parametrů zavaděče systému LILO, můžete zavinit, že systém Linux nebude možné zavést. V těchto situacích by přišel vhod nějaký jiný způsob zavedení systému, jenž by fungoval vždy (když samozřejmě funguje hardware). Pro počítače PC je takovou alternativou zavádění systému z diskety.

Většina distribucí operačního systému Linux umožňuje vytvořit takzvanou *záchrannou zaváděcí disketu* již při instalaci systému. Doporučujeme to udělat. Avšak některé takovéto záchranné diskety obsahují pouze jádro systému a předpokládá se, že při odstraňování vzniklých problémů budete používat programy uložené na instalačních médiích distribuce systému. Někdy ale tyto programy nestačí. Například v případě, že budete muset obnovit některé soubory ze záloh, jež jste dělali programem, který není na instalačních discích distribuce systému.

Proto by si správce měl vytvořit vlastní zaváděcí diskety, přizpůsobené konkrétním potřebám. Pokyny jak na to jsou obsaženy v příručce *Bootdisk HOWTO* Grahama Chapmana. Musíte mít přirozeně neustále na paměti, abyste měli záchranné zaváděcí diskety stále aktuální.

Disketovou mechaniku, na níž je připojen kořenový souborový systém, nelze použít k ničemu jinému. To může být nevýhodné v případě, že máte jen jedinou mechaniku. Pokud ale máte dostatek paměti, můžete vytvořit disketu tak, aby se kořenový souborový systém vytvořil v ramdisku. Pak budete moci disketovou mechaniku využít k libovolným dalším činnostem.

Disketovou mechaniku, která se využívá pro připojení superuživatelské zaváděcí diskety, nebudete moci použít k ničemu jinému. Tento problém může být obzvlášť nepříjemný v případě, že máte jenom jednu disketovou mechaniku. Když ale máte dostatek paměti, můžete nastavit zavádění z diskety tak, aby se obsah tohoto superuživatelského zaváděcího disku načítal do virtuálního ramdisku (je proto nutné zvlášť nakonfigurovat jádro systému na zaváděcí disketě). Když se podaří načíst superuživatelskou zaváděcí disketu na ramdisk, lze disketovou mechaniku využít pro připojení jiných disket.

Proces init

Uuno on numero yksi

Tato kapitola popisuje **Proces init**, jenž je vždy prvním uživatelským procesem, který spouští jádro systému. **Proces init** má mnoho důležitých povinností. Spouští například program **getty** umožňující uživatelům přihlásit se do systému, implementuje úrovně běhu systému, stará se o osiřelé procesy a podobně. V této kapitole bude vysvětleno, jak se **Proces init** konfiguruje a jak se zavádí různé úrovně běhu systému.

Proces init přichází první

Proces init je jedním z programů, jež jsou sice absolutně nezbytné k tomu, aby operační systém Linux fungoval, ale kterým obvykle nemusíte věnovat přílišnou pozornost. Součástí každé dobré distribuce systému je i předem nastavená konfigurace **Procesu init**, která vyhovuje většině systémů. Správce pak už obvykle nemusí kolem **Procesu init** nic dělat. O **Proces init** se většinou staráte jenom pokud připojujete nové sériové terminály, modemy pro příchozí volání (takzvané *dial-in* modemy, nikoliv tedy modemy, pomocí kterých se budete připojovat do jiných systémů, to jsou *dial-out* modemy) a když potřebujete změnit implicitní úroveň běhu systému.

Když se zavede jádro systému (načte se do paměti, spustí se, inicializuje ovladače zařízení, datové struktury a podobně), ukončí svou roli v proceduře zavádění operačního systému tím, že spustí první program uživatelské úrovně – **Proces init**. Takže **Proces init** je vždy prvním spuštěným procesem, má tedy vždy číslo procesu 1.

Jádro systému hledá z historických důvodů **Proces init** na několika místech. Jeho správné umístění v systému Linux nicméně je `/sbin/init`. Nenajde-li jádro **Proces init**, pokusí se spustit program `/bin/sh` a pokud neuspěje, skončí neúspěšně i celá procedura startu systému.

Když **Proces init** nastartuje, dokončí proces zavedení systému tím, že se provede několik administrativních úkolů, například kontrola souborových systémů, úklid v adresáři `/tmp`, start různých služeb a spuštění procesu **getty** pro každý terminál nebo virtuální konzolu, prostřednictvím nichž se uživatelé mohou přihlašovat do systému. (Viz kapitola 10.)

Po správném zavedení systému **Proces init** po každém odhlášení uživatele restartuje procesy **getty** pro příslušný terminál. Umožní tím další přihlášení jiných uživatelů. **Proces init** si také osvojuje všechny osiřelé procesy. Když některý z procesů spustí další proces (svého potomka) a později ukončí svou činnost dřív než potomek, vzniká sirotek, který se ihned stává potomkem **Procesu init**. Adopce sirotků má význam především z různých technických důvodů, je ale dobré o ní vědět, protože je pak snadnější pochopit význam položek seznamu procesů a grafů hierarchického stromu běžících procesů[32].

[32] Samotný proces **init** zaniknout nemůže. Nelze jej zabít ani signálem SIGKILL.

Správce systému má na výběr několik verzí programu **Proces init**. Většina distribucí operačního systému Linux používá program **sysvinit** (autorem programu je Miquel van Smoorenburg), jehož předlohou je **Proces init** pro Unix System V. Unix verze BSD používá odlišný **Proces init**. Primárním rozdílem mezi uvedenými verzemi programů jsou úrovně běhu systému. Unix System V je implementuje, kdežto verze BSD nikoliv (alespoň v klasické verzi). Tento rozdíl není podstatný, a tak se podíváme pouze na program **sysvinit**.

Konfigurace Procesu init pro spouštění programu getty – soubor /etc/inittab

Když **Proces init** startuje, načítá konfigurační soubor `/etc/inittab`. Když pak systém Linux běží, **Proces init** tento konfigurační soubor opakovaně načítá pokaždé, když přijme signál HUP[33]. Tato vlastnost umožňuje měnit konfiguraci programu **Proces init** a zajistit, aby se takováto změna projevila i bez toho, že by bylo nutné znovu zavést systém.

Soubor `/etc/inittab` je trochu složitější. Začneme tedy s jednoduchým příkladem konfigurace programu **getty**. Jednotlivé řádky souboru `/etc/inittab` sestávají ze čtyř polí oddělených dvojtečkou:

```
id:úrovně_běhu:akce:proces
```

Uvedené položky budou popsány níže. Soubor `/etc/inittab` může kromě řádkových záznamů obsahovat i prázdné řádky a řádky začínající znakem `#`. Ty `Proces init` ignoruje.

id — První položka identifikuje každý z řádků konfiguračního souboru. U řádků procesů **getty** se tak blíže specifikuje terminál, který daný proces obsluhuje (rozlišuje se číslem následujícím za příslušným názvem speciálního souboru /dev/tty). V řádcích pro ostatní procesy nemá toto pole žádný význam (kromě jeho omezení v délce), avšak mělo by být v celém souboru jedinečné.

úrovně_běhu — Úrovně běhu systému, které připadají pro daný řádek (proces) v úvahu. Úrovně se zadávají jako číslice bez oddělovače. Jednotlivé úrovně běhu systému budou popsány v následujícím odstavci.

akce — Akce, jež se má provést, například respawn (opakovaně spustí příkaz, jenž je uveden v dalším poli pokaždé, když je z různých důvodů ukončen), nebo once (spustí daný příkaz jenom jednou).

proces — Příkaz, který se má spustit.

Chcete-li spustit program **getty** pro první virtuální terminál (`/dev/tty1`) ve všech běžných víceuživatelských úrovních běhu (2–5), vložte do souboru `/etc/inittab` tento řádek:

```
1:2345:respawn:/sbin/getty 9600 tty1
```

První pole identifikuje řádek pro zařízení `/dev/tty1`. Druhá položka určuje, že proces uvedený v posledním poli lze spouštět na úrovni běhu systému číslo 2, 3, 4 a 5. Třetí pole znamená, že tento příkaz by měl být vždy opakovaně spuštěn poté, co se ukončí (aby se po odhlášení uživatele mohl přihlásit kdokoliv jiný). V posledním poli je příkaz, který spouští proces `getty` pro první virtuální terminál[34].

Když budete potřebovat přidat do systému další terminály nebo modemové linky pro příchozí volání, měli byste rozšířit soubor `/etc/inittab` o další řádky, vždy jeden pro každý terminál, respektive modemovou linku. Více informací najdete v manuálových stránkách `init`, `inittab` a `getty`.

---

[33] Například příkazem **kill -HUP 1** jako root.

[34] Různé verze programu **getty** se spouštějí různě. Podívejte se na příslušnou manuálovou stránku a ověřte si, že čtete správnou manuálovou stránku.

Nespustí-li se úspěšně příkaz uvedený v souboru /etc/inittab a je-li v konfiguraci **Procesu init** nastavený jeho restart (akce respawn), bude proces zabírat značnou část systémových zdrojů. **Proces init** totiž požadovaný proces spustí, ten se neprovede úspěšně a ukončí se, **Proces init** jej opakovaně spustí, proces se ukončí, **Proces init** jej spustí, proces se ukončí a tak dál, až donekonečna. Této situaci se předchází tím, že si **Proces init** vede záznamy o tom, jak často se pokoušel určitý příkaz spustit. Je-li frekvence opakovaných pokusů o spuštění procesu příliš vysoká, **Proces init** před dalším pokusem o provedení příkazu vyčká pět minut.

Úrovně běhu systému

Úrovní běhu systému se rozumí určitý stav Procesu init i celého systému. Tento stav určuje, které ze služeb se poskytují. Jednotlivé úrovně se rozlišují čísly, uvedenými v tabulce 8.1. Pokud jde o uživatelsky definované úrovně (2 až 5), neexistují obecně platná pravidla pro jejich používání. Někteří správci systémů pomocí těchto úrovní určují, které subsystémy se spustí, zdali například poběží X, jestli bude systém připojený k síti a podobně. Jiní dávají přednost inicializaci všech subsystémů na všech uživatelsky definovaných úrovních, popřípadě některé ze subsystémů spouští a zastavují samostatně bez toho, že by se měnily úrovně běhu systému. To proto, že počet úrovní je poměrně malý a řízení konfigurace takovýchto systémů pomocí jednotlivých uživatelsky definovaných úrovní běhu systému je tedy příliš hrubé. Správce systému se v této otázce musí rozhodnout sám, avšak nejjednodušší bude řídit se způsobem, jenž implementuje distribuce, ze které byl systém Linux instalován.

0	Zastavení systému.
1	Jednouživatelský režim (pro administraci systému).
2-5	Běžný provoz (definováno uživatelem).
6	Znovuzavedení systému.

Tabulka 8.1 – Čísla úrovní běhu

Úrovně běhu systému se konfigurují v souboru `/etc/inittab` řádkem podobným tomuto:

```
l2:2:wait:/etc/init.d/rc 2
```

V prvním poli je uvedeno libovolné návěští, druhé pole znamená, že tento záznam platí pro úroveň běhu systému číslo 2. Třetí položka (pole wait) říká **Procesu init**, aby spustil příkaz uvedený ve čtvrtém poli jenom jednou, a to při startu dané úrovně běhu systému a pak vyčkal, než se příkaz provede. Samotný příkaz `/etc/init.d/rc` pak spustí všechny procesy a příkazy, které jsou potřebné pro spuštění a ukončení služeb, jimiž se implementuje úroveň běhu číslo 2.

Všechnu dřinu spojenou s nastavováním určité úrovně běhu dělá samotný příkaz uvedený ve čtvrtém poli záznamu. Spouští služby, které zatím neběží, a pozastaví ty, které by na nové úrovni běhu již neměly být poskytovány. Záleží na konkrétní distribuci operačního systému Linux, který z příkazů bude v posledním políčku souboru `/etc/inittab` přesně uveden a které úrovně běhu budou v této konfiguraci implementovány.

Když proces `init` startuje, hledá ten řádek v souboru `/etc/inittab`, jenž specifikuje implicitní úroveň běhu systému:

```
id:2:initdefault:
```

`Proces init` lze také požádat o to, aby se spustil v jiné než běžné úrovni běhu, a to tak, že předáte jádru systému řádkový parametr `single` nebo `emergency`. Parametry lze jádru předat například programem LILO. Tímto způsobem spustíte systém na úrovni 1.

Když už systém běží, lze změnit aktuální úroveň běhu příkazem `telinit`. V případě, že se úroveň běhu systému mění, `Proces init` spouští ten příkaz v souboru `/etc/inittab`, jenž odpovídá nové úrovni běhu systému.

Zvláštní konfigurace v souboru `/etc/inittab`

Soubor `/etc/inittab` má některé speciální funkce, jež umožňují **Procesu init** reagovat i na některé zvláštní situace. Tyto speciální funkce definují zvláštní klíčová slova ve třetím poli záznamu konfiguračního souboru. Několik příkladů:

`powerwait` Umožní **Procesu init** v případě výpadku napájení zastavit systém. Předpokládá se, že systém používá záložní zdroj UPS a software, jenž sleduje UPS a informuje **Proces init** o případném výpadku napájení.

`ctrlaltdel` Nařizuje **Procesu init** znovu zavést systém, když uživatel současně zmáčkne klávesy <Control>-<Alt>-<Delete>. Uvědomte si, že správce systému může reakci na tuto kombinaci nastavit tak, že se místo rebootu provede nějaká jiná akce, že se například – zvlášť když má k systému přístup širší veřejnost – tato klávesová zkratka ignoruje. (Nebo se spustí program **nethack**.)

`sysinit` Příkaz spouštěný při startu systému. Typicky se takto zajistí například smazání adresáře `/tmp`.

Výše uvedený seznam klíčových slov není úplný. Další možnosti i podrobnosti týkající se těch, o kterých se zmiňujeme, uvádí manuálová stránka souboru `inittab`.

Zavádění systému v jednouživatelském režimu

Důležitou úrovní běhu systému je takzvaný jednouživatelský režim (úroveň běhu číslo 1), v němž může počítač používat pouze správce systému. V tomto režimu běží jenom minimum systémových služeb (včetně možnosti přihlášení do systému). Jednouživatelský režim je nutný pro některé úlohy spojené s údržbou systému[35]. Například kontrola konzistence svazku `/usr` programem **fsck** vyžaduje, aby byla disková oblast se souborovým systémem odpojená. Toho ale nelze dosáhnout, pokud nejsou ukončeny téměř všechny systémové služby.

Běžící systém lze přepnout do jednouživatelského režimu příkazem **telinit** a požadavkem na přechod do úrovně běhu 1. Při zavádění systému lze do jednouživatelského režimu přejít zadáním parametru `single` nebo `emergency` na příkazové řádce jádra systému. Jádro předá parametry příkazové řádky **Procesu init**. **Proces init** podle tohoto parametru pozná, že nemá použít implicitní úroveň běhu. (Způsob, kterým se zadává parametr příkazové řádky jádra systému, je podmíněn způsobem, jakým se zavádí operační systém.)

Někdy je potřeba zavést systém v jednouživatelském režimu například proto, aby bylo možné ručně spustit program **fsck** dřív, než se systém pokusí připojit poškozený systém souborů `/usr`, nebo se s ním pokusí jiným způsobem manipulovat. Jakékoliv aktivity na defektním svazku jej s největší pravděpodobností poškodí ještě více, proto by se měla kontrola programem **fsck** udělat co nejdříve.

---

[35] Rozhodně neslouží ke hraní **nethack**.

Zaváděcí skripty, které spouští **Proces init**, automaticky přechází do jednouživatelského režimu pokaždé, když automatická kontrola programu **fsck** při zavádění systému neproběhne úspěšně. Pokouší se tak zabránit systému použít souborový systém, jenž je poškozený natolik, že jej nelze automaticky opravit zmiňovaným programem **fsck**. Takovéto poškození svazku je relativně málo frekventované a jeho příčinou bude pravděpodobně mechanické poškození pevného disku, případně nějaká chyba v experimentální verzi jádra systému. Bude ale lepší, když budete jako správci systému připraveni i na tuto situaci.

Správně nakonfigurovaný systém se před spuštěním příkazového interpretu v jednouživatelském režimu zeptá na přístupové heslo superuživatele. Je to důležité bezpečnostní opatření, protože jinak by bylo možné jednoduše zadat vhodný parametr příkazové řádky zavaděči systému LILO a dostat se tak k systému s oprávněním superuživatele. (Takto nastavený systém se samozřejmě nezavede, když bude důsledkem defektů na systémovém svazku soubor `/etc/passwd` poškozený. Pro tento případ je dobré mít někde po ruce zaváděcí diskety.)

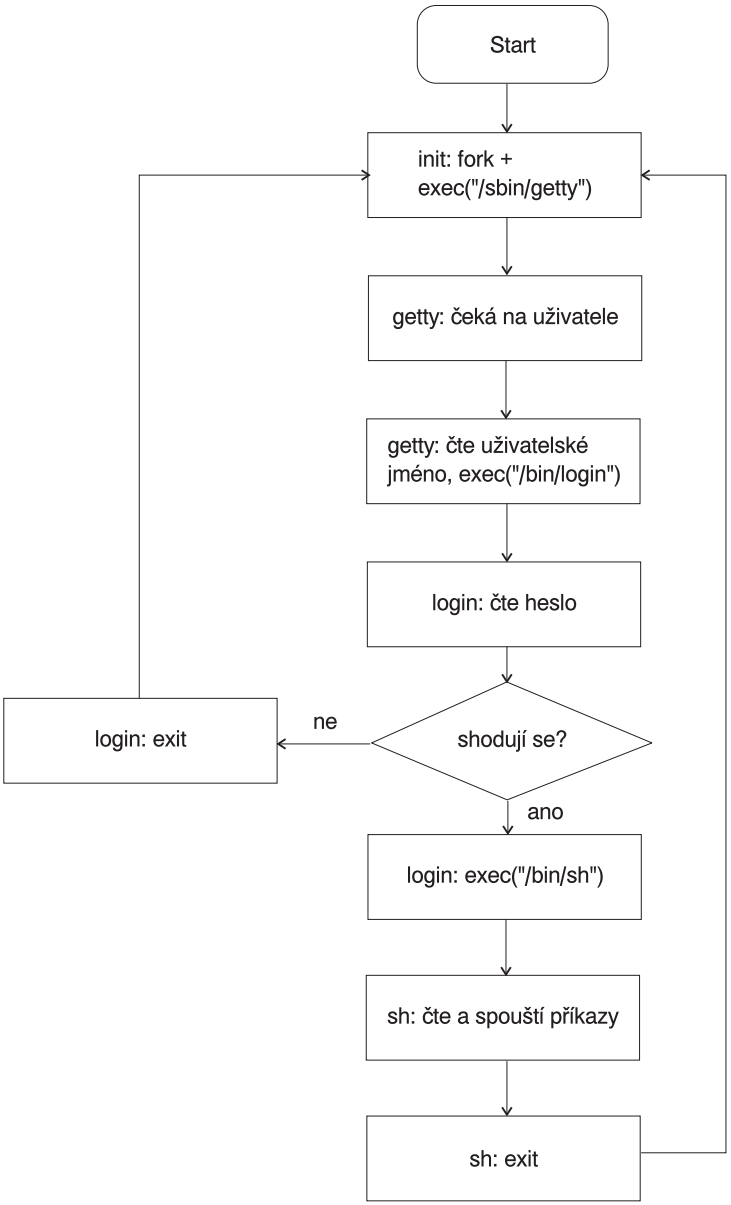
Přihlašování a odhlašování

Nestojím o to být členem klubu,
který přijímá lidi jako jsem já.
(Groucho Marx)

Tato část knihy popisuje, co se v systému děje poté, když se uživatel přihlásí nebo odhlásí. Dále budou detailněji popsány různé interakce některých procesů běžících na pozadí, při zahajování a ukončování sezení používané logovací soubory, konfigurační soubory a další.

Přihlašování přes terminály

Na obrázku 9.1 je graficky zobrazen algoritmus přihlášení uživatele do systému přes terminál. V prvním kroku si **Proces init** ověří, zda běží program **getty** pro dané terminálové spojení (nebo konzolu). Program **getty** sleduje terminál a čeká na uživatele, jenž by mu sdělil, že se chce přihlásit do systému (obvykle tím, že stiskne některou klávesu na klávesnici terminálu). Když proces **getty** zjistí, že uživatel něco napsal na klávesnici, vypíše na obrazovku uvítací zprávu. Ta je uložená v souboru `/etc/issue`. Pak vyzve uživatele, aby zadal své uživatelské jméno a nakonec spustí program **login**. Program **login** dostane zadané uživatelské jméno jako parametr a následně vyzve uživatele, aby zadal přístupové heslo. Je-li heslo zadáno správně, program **login** spustí příkazový interpret vybraný podle nastavení konfigurace pro přihlášeného uživatele. V opačném případě se program **login** jednoduše ukončí, a tím se ukončí i celý proces přihlašování (většinou až poté, co uživatel dostane další možnost zadat správné uživatelské jméno a přístupové heslo). **Proces init** rozpozná, že byla procedura přihlašování ukončena a spustí pro daný terminál novou instanci programu **getty**.



Obrázek 9.1 – Přihlášení přes terminál. Interakce programu **Proces init**, **getty**, **login** a shellu

Je důležité si uvědomit, že jediným novým procesem je ten, jenž vytvoří **Proces init** (použitím systémového volání `fork`). Procesy **getty** a **login** nahrazují právě tento nový proces (systémovým voláním `exec`).

V případě přihlašování po sériových linkách se pro sledování aktivity uživatelů používá zvláštní program proto, že někdy může být (a tradičně bývá) poměrně složité zjistit, kdy je terminál po ne-

činnosti opět aktivní. Program **getty** se rovněž přizpůsobuje přenosové rychlosti a dalším nastavením konkrétního spojení. Takovéto změny parametrů připojení jsou obzvlášť důležité v případě, že systém odpovídá na příchozí modemové žádosti o připojení, kdy se přenosové parametry běžně mění případ od případu.

V současnosti se používá několik různých verzí programů **getty** a **Proces init**. Mají samozřejmě své výhody i nevýhody. Je dobré si přečíst dokumentaci k verzím, které jsou součástí vašeho systému, ale rozhodně neuškodí ani informace o jiných verzích. Další dostupné verze programu lze vyhledat pomocí „Mapy programového vybavení pro Linux“ (The Linux Software Map). V případě, že nemusíte obsluhovat příchozí volání se žádostmi o přihlášení, nebudete se pravděpodobně muset programem **getty** zabývat, avšak podrobnější informace o programu **Proces init** pro vás budou i nadále důležité.

Přihlášení prostřednictvím sítě

Dva počítače, které jsou zapojeny v jedné síti, jsou obvykle propojeny jediným fyzickým kabelem. Když spolu stanice prostřednictvím sítě komunikují, programy, které běží na každé z nich a podílejí se na vzájemné komunikaci, jsou propojeny *virtuálními spojeními*, tedy jakousi sadou imaginárních kabelů. Když spolu aplikace na obou koncích virtuálního spojení komunikují, mají pro sebe vyhrazenou vlastní „linku“. Protože tato linka není skutečná, pouze imaginární, mohou operační systémy na obou počítačích vytvořit i několik virtuálních spojení sdílejících tutéž fyzickou linku. Takto spolu může s využitím jediného kabelu komunikovat několik programů bez toho, že by o ostatních spojeních věděly, nebo se o ně nějakým jiným způsobem staraly. Stejné fyzické médium může být sdíleno i několika počítači. Když pak existuje virtuální spojení mezi dvěma stanicemi, další počítače, které se komunikace neúčastní a sdílí tutéž fyzickou linku, toto spojení ignorují.

Toto byl komplikovaný a možná až příliš odtažitý popis reality. Měl by ale stačit k pochopení důležitého rozdílu mezi přihlášením prostřednictvím sítě a normálním přihlášením přes terminál. Virtuální spojení se vytvoří v případě, že existují dva programy na různých stanicích a přejí si spolu komunikovat. Vzhledem k tomu, že je principiálně možné připojit se z kteréhokoliv počítače v síti na kterýkoliv jiný, existuje velké množství potenciálních virtuálních spojení. Díky tomu není praktické spouštět proces **getty** pro každé potenciální síťové přihlášení do systému.

Proto také existuje jediný proces **inetd** (odpovídající procesu **getty**), který obsluhuje *všechna* síťová připojení. V případě, že proces **inetd** zaregistruje žádost o připojení ze sítě (tedy zaregistruje navázání nového virtuálního spojení s některým jiným počítačem zapojeným v síti), spustí nový proces obsluhující toto jediné přihlášení. Původní proces je nadále aktivní a dále čeká na nové požadavky o připojení.

Aby to nebylo až tak jednoduché, existuje pro připojení ze sítě víc komunikačních protokolů. Dva nejvýznamnější jsou **telnet** a **rlogin**. Kromě připojení do systému existuje i mnoho dalších druhů virtuálních spojení, která lze mezi počítači v síti navázat (síťové služby FTP, Gopher, HTTP a další). Bylo by neefektivní mít zvláštní proces, jenž by sledoval žádosti o navázání spojení pro každý typ připojení (službu). Místo toho existuje jediný proces, který umí rozeznat typ spojení a spustit správný program, jenž pak poskytuje odpovídající služby. Tímto procesem je právě proces **inetd**. Podrobnější informace uvádí *Příručka správce sítě*.

Co dělá program login

Program **login** se stará o autentizaci uživatele (kontroluje, zda bylo zadáno správné uživatelské jméno a přístupové heslo) a počáteční nastavení uživatelského prostředí nastavením oprávnění pro sériovou linku a spuštěním interpretu příkazů.

Částí procedury úvodního nastavení uživatelského prostředí je i vypsání obsahu souboru `/etc/motd` (zkratka pro „message of the day" – zprávu pro tento den) a kontrola nově příchozí elektronické pošty. Tyto kroky lze zakázat vytvořením souboru nazvaného `.hushlogin` v domovském adresáři uživatele.

Existuje-li soubor `/etc/nologin`, jsou přihlášení do systému zakázána. Tento soubor je typicky vytvářen příkazem **shutdown** nebo příbuznými programy. Program **login** kontroluje, jestli tento soubor existuje, a v případě, že je tomu tak, odmítne akceptovat přihlášení a předtím, než se definitivně ukončí, vypíše obsah tohoto souboru na terminál.

Program **login** rovněž zapisuje všechny neúspěšné pokusy o přihlášení do systémového logu (pomocí programu **syslog**). Rovněž zaznamenává úspěšné i neúspěšné pokusy o přihlášení superuživatele. Oba druhy záznamů jsou užitečné při pátrání po případných „vetřelcích".

Momentálně přihlášení uživatelé jsou zapsáni v seznamu `/var/run/utmp`. Tento soubor je platný jenom do dalšího znovuzavedení nebo zastavení systému, protože v průběhu zavádění systému se jeho obsah vymaže. Jinak jsou v souboru `/var/run/utmp` kromě seznamu všech přihlášených uživatelů a používaných terminálů (nebo síťových spojení) uvedeny i další užitečné informace. Příkazy **who**, **w** a další podobné se dívají právě do souboru `/var/run/utmp` a zjišťují, kdo je k systému připojený.

Všechna úspěšná přihlášení jsou zaznamenána do souboru `/var/log/wtmp`. Tento soubor se může bez omezení zvětšovat, proto je potřeba jej pravidelně mazat (například po týdnu) zadáním úkolu démonu **cron**[36]. Soubor wtmp lze procházet příkazem **last**.

Oba soubory `utmp` i `wtmp` mají binární formát (viz manuálová stránka **utmp**), takže je nelze prohlížet bez speciálních programů.

X a xdm

X implementují přihlášení prostřednictvím procesu **xdm**; viz též **xterm -ls**.

Řízení přístupu

Databáze uživatelů je tradičně uložena v souboru `/etc/passwd`. Některé systémy používají takzvaná *stínová hesla*. Přesouvají uživatelská přístupová hesla ze souboru `/etc/passwd` do souboru `/etc/shadow`. Sítě s velkým počtem počítačů, ve kterých se informace o uživatelských účtech sdílí pomocí systému NIS nebo nějakou jinou metodou, mohou databázi uživatelů automaticky kopírovat z jediného centrálního počítače na všechny ostatní stanice.

Databáze uživatelů obsahuje nejenom hesla, ale i některé další informace o uživatelích, například jejich skutečná jména, domovské adresáře a interprety příkazů, jež se implicitně spouští po přihlášení. Je potřeba, aby byly tyto informace o uživatelích v systému obecně dostupné a aby si je mohl každý přečíst. Kvůli tomu se heslo ukládá v zakódovaném tvaru. Má to ale jeden háček. Kdokoliv, kdo má přístup k databázi uživatelů, může s pomocí různých kryptografických metod zkusit hesla uhodnout i bez toho, že by se musel přihlásit k hostitelskému počítači. Systém stínových hesel se snaží zamezit možnosti prolomení přístupových hesel tím, že se přesouvají do jiného sou-

[36] Dobré distribuce to dělají samy.

boru, jenž je přístupný pouze superuživateli (hesla se i tak ukládají v zakódovaném tvaru). Avšak s pozdější instalací systému stínových hesel na systému, který jej nepodporuje, mohou vznikat různé potíže.

Ať tak nebo onak, z bezpečnostních důvodů je důležité pravidelně ověřovat, jestli jsou všechna v systému používaná přístupová hesla netriviální, tedy taková, aby nebylo lehké je uhodnout. Lze použít například program **crack**, jenž zkouší hesla v `/etc/passwd` dekódovat. Heslo, které se mu podaří uhodnout, nelze podle výše uvedeného považovat za spolehlivé. Program **crack** mohou samozřejmě zneužít i případní vetřelci, ale správci systému může jeho pravidelné používání pomoci preventivně omezit výběr nevhodných přístupových hesel. K volbě netriviálního přístupového hesla lze uživatele donutit i programem **passwd**. To je metoda, která je efektivnější především z hlediska zatížení procesoru, protože zpětná analýza zašifrovaných hesel programem **crack** je výpočetně o hodně náročnější.

Databáze skupin uživatelů je uložena v souboru `/etc/group`, u systémů se stínovými hesly případně v souboru `/etc/gshadow`.

Uživatel `root` se obvykle nemůže přihlásit z kteréhokoliv terminálu nebo počítače v síti, pouze z terminálu uvedeného v seznamu `/etc/securetty`. Pak je nutné mít k některému z těchto terminálů fyzický přístup. Avšak takovéto bezpečnostní opatření nelze považovat za dostatečné, protože je možné přihlásit se z kteréhokoliv jiného terminálu jako běžný uživatel a pro změnu uživatelských oprávnění použít příkaz **su**.

Spouštění interpretu příkazů

Při startu každý příkazový interpret automaticky spouští jeden či více předem určených souborů. Různé interprety spouští různé konfigurační soubory. Podrobnější informace najdete v dokumentaci k jednotlivým typům interpretů.

Většina příkazových interpretů nejdříve spustí některý globální soubor, například interpret Bourne shell (`/bin/sh`) a jeho klony spouští soubor `/etc/profile`, až poté spustí soubor `.profile`, jenž je uložen v uživatelově domovském adresáři. Soubor `/etc/profile` umožňuje správci systému nastavit běžné, implicitní uživatelské prostředí, například nastavením proměnné PATH, tak, aby zahrnovalo kromě obvyklých i lokální adresáře s příkazy. Soubor `.profile` zase umožňuje každému z uživatelů upravit si předem nastavené běžné prostředí podle svého vlastního vkusu.

Správa uživatelských účtů

Jaký je rozdíl mezi správcem systému a překupníkem drog?
Žádný, oba měří své prostředky v kilech a oba mají své klienty.
(Starý a otřelý počítačový vtip)

Tato kapitola popisuje způsob vytváření nových uživatelských účtů, změny vlastností těchto účtů a způsoby jejich odstraňování. Různé distribuce systému Linux používají pro tyto úkoly různé nástroje.

Co je to účet?

Používá-li počítač více lidí, je obvykle nutné mezi jednotlivými uživateli rozlišovat. Například proto, aby jejich osobní soubory a data byly osobními v pravém smyslu slova. Identifikace uživatelů je ale důležitá i v případě, že systém využívá v jednom okamžiku pouze jedna osoba, což se týká převážné většiny mikropočítačů[37]. Proto má každý uživatel systému přiděleno jednoznačné uživatelské jméno, které zadává při každém přihlášení.

Avšak pojem „účet" je poněkud širší a zahrnuje – kromě uživatelského jména – i některé další informace o uživateli. Pojmem *uživatelský účet* se označují všechny soubory, zdroje a informace, jež se vztahují k danému uživateli. Běžně se termín „účet" spojuje s bankovním sektorem. V komerčním systému se kolem každého účtu skutečně točí nějaké peníze. Ty mohou z účtu mizet různou rychlostí, podle toho, jak moc uživatel systém zatěžuje. Tak například diskový prostor lze ohodnotit cenou za megabajt uložených dat a den, čas procesoru může mít určitou cenu za sekundu využití a podobně.

Vytváření uživatelských účtů

Samotné jádro systému Linux považuje uživatele systému za pouhé číslo. Každého uživatele lze totiž identifikovat podle jednoznačného celého čísla, takzvaného *identifikačního čísla uživatele* (anglicky *user ID*, zkráceně *UID*). Je tomu tak proto, že počítač umí zpracovat čísla rychleji a jednodušeji než jména v textové formě. Jména v textové podobě (*uživatelská jména*) se udržují ve zvláštní databázi mimo vlastní jádro. Tato databáze obsahuje též další informace o uživatelích systému.

[37] Bylo by docela hloupé, kdyby si má sestra četla mé milostné dopisy.

Když potřebujete vytvořit nový uživatelský účet, musíte přidat informace o novém uživateli do této uživatelské databáze a vytvořit pro něj vlastní domovský adresář. Kromě toho by měl každý nový uživatel absolvovat školení. Je též vhodné nastavit pro nové uživatele přiměřené počáteční nastavení prostředí.

Většina distribucí systému Linux se dodává s programy pro vytváření uživatelských účtů. Správce má dokonce k dispozici hned několik takovýchto programů. Dvě varianty, jejichž uživatelským rozhraním je příkazová řádka, jsou programy **adduser** a **useradd**. Existují i nástroje s grafickým uživatelským rozhraním. Ať už se jako správce systému rozhodnete pro kterýkoliv z programů, oceníte jejich hlavní přínos, tedy to, že omezují manuální práci s nastavováním na minimum, či dokonce úplně. I když na vás při administraci uživatelských účtů čeká mnoho dost spletitých detailů, díky těmto nástrojům vypadá jednoduše. Postup při „ručním" zakládání nových uživatelských účtů uvádí odstavec *Manuální vytváření účtů*.

Soubor /etc/passwd a další informační soubory

Základní databází uživatelů v systému Unix je textový soubor `/etc/passwd` (označovaný jako *password file*), v němž jsou uvedena platná uživatelská jména a další k nim přidružené informace. Každému uživateli odpovídá v souboru jeden záznam – řádek, který je rozdělen na sedm polí, jejichž oddělovačem je dvojtečka. Význam jednotlivých položek je následující:

- Uživatelské jméno.
- Heslo v zakódované podobě.
- Identifikační číslo uživatele.
- Identifikační číslo skupiny.
- Skutečné jméno uživatele, případně popis účtu.
- Domovský adresář.
- Příkazový interpret (nebo program), který se spustí po přihlášení.

Formát jednotlivých políček je podrobněji popsán v manuálové stránce souboru `passwd`.

Každý uživatel systému má k souboru `/etc/passwd` přístup (může jej číst). Může tedy například zjistit přihlašovací jména ostatních uživatelů. To ale znamená, že jsou všem přístupná i hesla ostatních uživatelů (druhé pole každého záznamu). Hesla uložená v souboru `/etc/passwd` jsou zakódovaná, takže teoreticky nevzniká žádný problém. Avšak použitý kódovací algoritmus lze prolomit, zvlášť je-li zvolené heslo jednoduché (například krátké slovo, jež lze najít v nějakém slovníku, jméno nebo příjmení uživatele a podobně). Proto z hlediska bezpečnosti není dobré mít hesla uložená přímo v souboru `/etc/passwd`.

Řada systémů Linux používá systém takzvaných *stínových hesel*. Jde o alternativní způsob uložení uživatelských přístupových hesel, jež se zašifrovaná ukládají do jiného souboru (`/etc/shadow`), který může číst pouze superuživatel. Soubor `/etc/passwd` pak obsahuje v druhém poli pouze speciální znak. Programy, které si potřebují ověřit totožnost uživatele a mají propůjčená přístupová práva vlastníka souboru (pomocí volání jádra setuid), mohou soubor `/etc/shadow` číst. Běžné programy, které používají pouze některé z dalších položek souboru `/etc/passwd`, přístup k heslům uživatelů systému nemají[38].

Výběr čísel uživatelského ID a ID skupiny

Ve většině systémů nezáleží na tom, jaké jsou hodnoty UID a GID. Když ale používáte síťový souborový systém NFS, musíte mít stejná UID a GID na všech systémech v síti. To proto, že i systém

[38] Ano, znamená to tedy, že soubor hesel obsahuje kdeco, kromě hesel. Půvab pokroku...

NFS identifikuje uživatele podle hodnoty UID. Jestliže systém NFS nepoužíváte, můžete vybírat identifikační čísla uživatele a skupiny podle automatického návrhu některého z nástrojů pro správu uživatelských účtů.

Když v síti využíváte NFS, budete si muset zvolit některý z mechanismů synchronizace informací o uživatelských účtech. Jednou z možností je systém NIS – Network Information Service.

Měli byste se také vyvarovat opakovaného přidělování stejných uživatelských čísel (i textových uživatelských jmen), protože nový vlastník UID (případně uživatelského jména) by tak mohl mít přístup k souborům bývalého vlastníka, k jeho elektronické poště a dalším informacím.

Nastavení uživatelského prostředí: adresář /etc/skel

Když jste již pro nového uživatele vytvořili vlastní domovský adresář, můžete nastavit vlastnosti uživatelského prostředí tak, že do domovského adresáře nového uživatele nakopírujete některé soubory z adresáře `/etc/skel`. Správce systému totiž může v adresáři /etc/skel vytvořit konfigurační soubory, jež novým uživatelům vytvoří příjemné základní uživatelské prostředí. Administrátor může například vytvořit soubor `/etc/skel/.profile`, v němž lze nastavením proměnné prostředí EDITOR vybrat některý z textových editorů, jenž by měl pro méně zkušené uživatele přátelské ovládání.

Avšak obvykle se doporučuje mít v adresáři /etc/skel co nejméně souborů, protože by jinak bylo téměř nemožné upravit na větších víceuživatelských systémech při každé změně konfigurační soubory v již existujících uživatelských adresářích. Když se například změní název onoho uživatelsky příjemného editoru, všichni současní uživatelé si musí upravit svůj vlastní soubor .profile. Správce systému by se to mohl pokusit udělat automaticky, například pomocí skriptu, ale takovéto akce téměř pravidelně končí tak, že se nechtěně přepíše či poškodí nějaký jiný soubor.

Vždy když to situace dovolí, je lepší nastavovat globální konfigurace v globálních souborech, jako je /etc/profile. Pak lze nastavení pohodlně upravovat bez toho, že by se muselo měnit vlastní nastavení jednotlivých uživatelů systému.

Manuální vytváření účtů

Nový uživatelský účet lze vytvořit ručně tímto postupem:

- Upravte soubor `/etc/passwd` příkazem **vipw** tak, že do souboru hesel přidáte nový řádek pro nový uživatelský účet. Je nutné dodržovat správnou syntaxi. *Není vhodné upravovat tento soubor přímo běžným editorem!* Program **vipw** soubor `/etc/passwd` uzamkne, takže jej ostatní programy nemohou změnit. Do pole pro heslo vložte znak *, takže zatím nebude možné se prostřednictvím tohoto účtu do systému přihlásit.
- Je-li třeba vytvořit i novou pracovní skupinu, upravte soubor `/etc/group` rovněž programem **vigr**.
- Příkazem **mkdir** pro nového uživatele vytvoříte domovský adresář.
- Do nově vytvořeného domovského adresáře nakopírujte konfigurační soubory z adresáře `/etc/skel`.
- Příkazy **chown** a **chmod** upravte jejich vlastníka a přístupová práva. Užitečný je v tomto případě jejich parametr `-R`. Správná přístupová práva se mohou trochu lišit, a to podle typického využití toho kterého systému, nicméně příkazy v níže uvedeném příkladu obvykle vyhovují většině případů:

```
cd /home/newusername
chown -R username.group .
```

```
chmod -R go=u,go-w .
chmod go= .
```

- Zadejte heslo programem **passwd**.

Poté, co bylo v posledním kroku nastaveno heslo, bude nový účet přístupný. Heslo by se skutečně mělo nastavovat až v posledním kroku, jinak by se mohl uživatel nepozorovaně přihlásit do systému například v době, kdy kopírujete konfigurační soubory.

Někdy je potřeba vytvořit „falešný" účet, který se nebude používat pro přihlašování běžných uživatelů. Například při konfigurování anonymního serveru FTP je výhodné vytvořit účet s uživatelským jménem ftp. Pak si ze serveru může stahovat soubory kdokoliv a pro budoucí (anonymní) klienty se nemusí zřizovat vlastní uživatelské účty. V takovýchto případech obvykle není třeba nastavovat heslo (vynechá se poslední krok výše uvedeného postupu). Je lepší zřizovat takovéto anonymní účty bez nastaveného hesla. Pak k nim má totiž přístup pouze superuživatel.

Změny vlastností uživatelských účtů

Je několik příkazů, kterými lze měnit různé vlastnosti uživatelských účtů (tedy příslušných položek v souboru /etc/passwd):

chfn Mění pole, ve kterém je uloženo skutečné jméno uživatele.

chsh Mění nastavený příkazový interpret.

passwd Mění přístupové heslo.

Superuživatel může pomocí těchto programů změnit vlastnosti kteréhokoliv účtu. Ostatní neprivilegovaní uživatelé mohou měnit pouze vlastnosti svého vlastního účtu. V některých případech je vhodnější běžným uživatelům zakázat možnost používat uvedené příkazy (programem **chmod**), například v případě, že systém používá větší počet méně zkušených začátečníků.

Ostatní změny položek souboru `/etc/passwd` se musí dělat ručně. Když například potřebujete změnit uživatelské jméno, musíte přímo upravit databázi uživatelů `/etc/passwd` (připomínáme, že pouze příkazem **vipw**). Analogicky, když potřebujete přidat či odebrat uživatele z nebo do některé z pracovních skupin, musíte upravit soubor `/etc/group` (příkazem **vigr**). Avšak takovéto úkoly se dělají zřídka a musí se dělat opatrně, protože když například změníte některému z uživatelů jeho uživatelské jméno, nebude mu docházet elektronická pošta a musíte pro něj vytvořit poštovní alias[39].

Zrušení uživatelského účtu

Potřebujete-li zrušit uživatelský účet, smažte nejdříve všechny soubory, které patří uživateli rušeného účtu, včetně poštovní schránky, aliasů pro elektronickou poštu, tiskových úloh, úkolů spouštěných démony **cron** a **at** a všechny další odkazy na tohoto uživatele. Pak odstraňte odpovídající řádek v souborech /etc/passwd a /etc/group (nezapomeňte odstranit uživatelské jméno i ze všech skupin, do nichž byl uživatel zařazen). Předtím, než začnete mazat vše ostatní, je lepší zakázat přístup k rušenému účtu (viz níže). Uživatel tak nebude mít možnost se připojit do systému v době, kdy je jeho účet odstraňován.

Nezapomeňte, že uživatelé systému mohou mít některé soubory uloženy i mimo svůj domovský adresář. Najdete je pomocí příkazu **find**:

```
find / -user username
```

[39] Uživatelské jméno se může změnit například v důsledku sňatku.

Pamatujte na to, že když má váš systém velké disky, poběží výše uvedený příkaz dost dlouho. V případě, že máte v souborovém systému připojeny síťové disky, musíte dávat pozor, abyste tím nezpůsobili problémy s odezvou v síti nebo na serveru.

Součástí některých distribucí systému Linux jsou i zvláštní příkazy, které lze použít při rušení uživatelských účtů. Zkuste na vašem systému vyhledat programy **deluser** či **userdel**. Každopádně není zase tak složité to udělat ručně, navíc tyto programy nemusí najít a odstranit všechny souvislosti.

Dočasné zablokování uživatelského účtu

Občas je potřeba dočasně zablokovat přístup k některému z účtů bez toho, že by bylo nutné jej smazat. Například když uživatel nezaplatil poplatky za využívání systému, nebo když má správce systému podezření, že neznámý útočník prolomil přístupové heslo uživatele některého účtu.

Nejvhodnějším způsobem, jak zamezit přístup k podezřelému účtu, je zaměnit nastavený příkazový interpret na zvláštní program, který na terminál pouze vypíše nějaké hlášení. Když se pak kdokoliv pokusí přihlásit do systému přes zablokovaný účet, neuspěje a dozví se proč. Zprávou lze sdělit uživateli, že se má spojit se správcem systému a domluvit se s ním na řešení vzniklého problému.

Alternativou je změna uživatelského jména, případně hesla. Uživatel se tak ale nedozví, co se vlastně děje. A zmatení uživatelé znamenají i více práce pro správce systému[40].

Nejjednodušší způsob, jak vytvořit program, který by blokoval přístup k účtu, je napsat skript pro program **tail**:

```
#!/usr/bin/tail +2

Tento účet byl z důvodů porušení bezpečnostních opatření zablokován.

Volejte, prosím, číslo 555-1234 a vyčkejte příjezdu mužů v černém.
```

Podle prvních dvou znaků (`#!`) pozná jádro systému, že zbytek řádku je příkaz, který je třeba spustit, aby se skript provedl. V tomto případě je to příkaz `tail`, jenž vypíše vše kromě prvního řádku na standardní výstup.

Je-li uživatel billg podezřelý, že porušuje bezpečnostní opatření, může správce systému udělat něco jako:

```
# chsh -s /usr/local/lib/no-login/security billg
# su - billg

Tento účet byl z důvodů porušení bezpečnostních opatření zablokován.

Volejte, prosím, číslo 555-1234 a vyčkejte příjezdu mužů v černém.

#
```

Pomocí příkazu `su` ve výše uvedeném příkladu se pochopitelně pouze testuje, zda je změna původně nastaveného interpretu funkční.

Takovéto skripty by měly být uloženy ve zvláštním adresáři, aby jejich jména nekolidovala s příkazy jednotlivých uživatelů.

[40] Pokud jste ale BOFH, může s nimi být *spousta* legrace. (Pozn. překladatele: viz http://www.bofh.net

Zálohování

Hardware je nedeterministicky spolehlivý.
Software je deterministicky nespolehlivý.
Lidé jsou nedeterministicky nespolehliví.
Příroda je deterministicky spolehlivá.

Tato kapitola vysvětluje proč, jak a kdy zálohovat a jak ze záloh obnovit data.

Jak je důležité míti zálohy

Vaše data mají určitou cenu. Jejich cena je daná hodnotou vašeho času a cenou úsilí potřebného pro nové vytvoření dat v případě jejich ztráty. To vše lze vyjádřit v penězích, nebo přinejmenším vyvážit zármutkem a slzami. Někdy již totiž ztracené informace nelze obnovit, například když data pocházejí z nějakých neopakovatelných experimentů. Vzhledem k tomu, že informace a data jsou v jistém slova smyslu investicí, měli byste tuto investici chránit a učinit kroky, jež by zabránily jejímu znehodnocení.

V zásadě existují čtyři důvody, jež by mohly vést ke ztrátě dat: závady hardwaru, chyby v programech, jednání lidí nebo přírodní katastrofy[41]. I když je v současnosti hardware relativně spolehlivý, ještě stále se může v podstatě bez příčiny porouchat. Pokud jde o ukládání dat, je nejkritičtější součástí výpočetního systému pevný disk. Ve světě plném elektromagnetického šumu se bláhově spoléhá na to, že miniaturní magnetická políčka na povrchu disku, ve kterých jsou cenné informace uloženy, zůstanou v neporušeném stavu. U programů o spolehlivosti prakticky nelze mluvit, robustní a spolehlivý software je spíše výjimkou, než pravidlem. I lidé jsou dost nespolehliví. Buďto udělají nějakou chybu, nebo jsou zlomyslní a zničí data úmyslně. Přírodu obecně bychom neměli považovat za zlo, ale i když je na nás hodná, může způsobit zkázu. Takže je malým zázrakem, když všechno funguje, jak má.

Řekli jsme, že zálohování je způsobem ochrany investic do dat a informací. Máte-li několik kopií dat, nestane se zase až tak moc, když se některá z verzí zničí (cenou za takovou ztrátu je pouze to, že data musíte obnovit ze zálohy).

Je důležité zálohovat správně. Jako všechno ostatní v reálném světě, dříve nebo později selžou i zálohy. Součástí správného zálohování je i to, že se kontroluje, jestli vůbec funguje. Jistě byste si nechtěli jednoho dne „všimnout“, že se zálohy nedělaly správně[42]. Neštěstí v neštěstí – může se stát, že dojde k nějaké vážné havárii právě v okamžiku, kdy zálohujete a máte pouze jediné zálohovací médium. To se může rovněž poškodit a z úmorné práce zbude (někdy doslova) hromádka popela[43]. Může se také stát, že si v momentě, kdy se pokoušíte obnovit data ze záloh, uvědo-

[41] Pátý důvod pak je „něco jiného“.

[42] Nesmějte se, už se to nejednou stalo.

[43] Já tam byl, já to viděl...

míte, že jste zapomněli zálohovat něco důležitého, například databázi uživatelů systému, která může mít třeba 15 000 položek. To „nejlepší“ nakonec – všechny zálohy mohou fungovat bezchybně, ale v poslední známé páskové jednotce, jež ještě umí přečíst typ pásky, který používáte, je vědro vody.

Když totiž dojde na zálohy, je paranoia v popisu práce.

Výběr média pro zálohování

Co se týče zálohování, je nejdůležitějším rozhodnutím výběr médií. Musíte zvážit náklady, spolehlivost, rychlost, dostupnost a použitelnost.

Cena je poměrně důležitá, protože byste měli mít několikrát větší kapacitu zálohovacích médií, než je velikost zálohovaných dat. Levné médium je obvykle nezbytnost.

Extrémně důležitá je spolehlivost, protože poškozená záloha by rozbrečela i dospělého. Data uložená na zálohovacích médiích musí vydržet bez poškození i několik let. I způsob, kterým médium používáte, má vliv na jeho spolehlivost. Například pevný disk je typicky velmi spolehlivé médium, ovšem z hlediska zálohování je to k ničemu, pokud jej máte ve stejném počítači, ze kterého zálohujete.

Rychlost většinou není příliš důležitá, pokud zálohování může probíhat automaticky. Když nemusíte na zálohování dohlížet, pak nevadí, že trvá dejme tomu dvě hodiny. Ale naopak, nelze-li provést zálohování v době, ve které je počítač jinak nevyužitý (například v průběhu noci), pak může být i rychlost problémem.

Dostupnost je zjevně důležitá, protože nemůžete používat zálohovací médium, které není k dostání. Méně zjevný je důležitý požadavek, aby bylo zálohovací médium dostupné i v budoucnu a případně i pro jiné počítače než ty, které používáte dnes. Jinak se může stát, že nebudete schopni zálohy po nečekané události obnovit.

Použitelnost je nejvíce závislá na tom, jak často se zálohování provádí. Čím jednodušší je zálohování, tím lépe. Zálohování na zvolené médium nesmí být složité, pracné nebo otravné.

Typickými alternativami jsou diskety a pásky. Diskety jsou velmi levné, celkem spolehlivé, ne příliš rychlé, velmi dostupné, ale ne příliš použitelné v případě velkého množství dat. Magnetické pásky jsou levné i drahé, celkem spolehlivé, celkem rychlé, dost dostupné a – podle toho, jaká je jejich kapacita – z hlediska použitelnosti i docela pohodlné.

Existují i jiné možnosti. Obvykle nejsou nejlepší, pokud jde o jejich dostupnost, ale vznikne-li problém, oceníte je více než ostatní možnosti. Řeč je o magneto-optických discích, jež kombinují dobré vlastnosti disket (jejich náhodný, nesekvenční přístup k souborům, možnost rychlého obnovování jednotlivých souborů) a magnetických páskových médií (zálohování velkého objemu dat).

Výběr nástroje pro zálohování

Existuje bezpočet nástrojů, jichž lze při zálohování využít. Mezi ty tradiční, používané při zálohování v systémech Unix, patří programy **tar**, **cpio** a **dump**. Kromě toho lze sáhnout i po velkém množství balíků programů třetích výrobců (šířených zdarma jako freeware i komerčně). Výběr médií pro zálohování má často vliv i na výběr nástroje.

Programy **tar** a **cpio** jsou podobné a z hlediska zálohování prakticky stejné. Oba programy umí ukládat soubory na pásky a obnovovat je, oba jsou schopné využívat téměř všechna média, protože díky ovladačům jádra systému, které mají na starost obsluhu nízkoúrovňových zařízení, se takováto zařízení chovají vůči programům na uživatelské úrovni stejně. Některé unixové verze programů **tar** a **cpio** mo-

hou mít problémy s neobvyklými soubory (symbolickými odkazy, speciálními soubory, soubory s velmi dlouhou cestou a podobně), avšak verze pro operační systém Linux by měly pracovat se všemi soubory korektně.

Program **dump** se liší v tom, že přistupuje k souborovému systému přímo a ne prostřednictvím jeho služeb. Navíc byl napsán speciálně pro zálohování, kdežto programy **tar** a **cpio** jsou primárně určeny pro archivaci souborů, i když je lze s úspěchem použít i jako zálohovací nástroje.

Přímý přístup k souborovému systému má několik výhod. Umožňuje zálohovat soubory bez toho, že by to mělo vliv na jejich časové značky. U programů **tar** a **cpio** byste museli nejdřív připojit souborový systém pouze pro čtení. Přímé čtení dat ze souborového systému je také efektivnější v případech, kdy je potřeba zálohovat všechna data na disku, protože v tomto případě proběhne zálohování s výrazně menším pohybem čtecích hlav. Největší nevýhodou přímého přístupu je, že zálohovací program, který jej používá, je specifický pro každý typ souborového systému. Program **dump** pro Linux rozumí pouze souborovému systému ext2.

Program **dump** má navíc zabudovanou podporu úrovní zálohování, o kterých bude řeč později. U programů **tar** a **cpio** musí být implementovány pomocí jiných nástrojů.

Srovnávání zálohovacích nástrojů třetích výrobců jde nad rámec této knihy. „Mapa programů pro Linux“ (The Linux Software Map) uvádí výčet těch, které jsou jako freeware šířeny zdarma.

Jednoduché zálohování

Při proceduře jednoduchého zálohování se v prvním kroku vytvoří zálohy všech dat a v dalších krocích se pak zálohuje jenom to, co se od posledního zálohování změnilo. Prvnímu kroku se říká *úplné zálohování*, v dalších krocích se tvoří *inkrementální zálohy*. Úplné zálohování je obvykle pracnější než inkrementální, protože se na médium zapisuje víc dat, a proto se úplná záloha nemusí vždy vejít na jednu pásku (nebo disketu). Naopak, obnovování dat z inkrementálních záloh bývá pochopitelně mnohokrát pracnější, než obnovování z úplných záloh. Nicméně inkrementální zálohování lze optimalizovat tak, že se vždy zálohují všechny změny od poslední úplné zálohy. Takto je sice o něco pracnější proces zálohování, ale pak by nemělo být nikdy potřeba obnovovat víc než jednu úplnou a jednu inkrementální zálohu.

Uveďme příklad, kdy chcete data zálohovat každý den a máte k dispozici šest páskových kazet, můžete první z nich použít (řekněme v pátek) pro uložení první úplné zálohy a pásky 2 až 5 pro inkrementální zálohování (od pondělí do čtvrtka). Pak vytvoříte novou úplnou zálohu na pásku číslo 6 (druhý pátek) a začnete znovu s inkrementálním zálohováním na pásky 2 až 5. Nepřepisujte pásku číslo jedna do doby, než se ukončí nové úplné zálohování – v jeho průběhu by se totiž mohlo stát něco neočekávaného. Poté, co vytvoříte novou úplnou zálohu na pásce číslo 6, měli byste uschovat pásku 1 někam jinam pro případ, že by se ostatní zálohovací pásky zničily – například při požáru. Takto vám v případě neočekávané události zůstane alespoň něco. Když pak potřebujete vytvořit další úplnou zálohu, dojdete pro pásku číslo 1 a pásku číslo 6 necháte na jejím místě.

Máte-li víc než šest kazet, můžete ukládat na ty, jež jsou navíc, další úplné zálohy. Pokaždé, když budete dělat úplnou zálohu, použijete tu nejstarší pásku. Takto budete mít úplné zálohy z několika předchozích týdnů, což je dobré, když potřebujete obnovit například některý starší, omylem smazaný, soubor, případně starší verzi nějakého souboru.

Zálohování programem tar

Pomocí programu **tar** lze jednoduše vytvořit úplnou zálohu tímto způsobem:

```
# tar -create -file /dev/ftape /usr/src
tar: Removing leading / from absolute path names in the archive
#
```

Ve výše uvedeném příkladu byla použita syntaxe programu **tar** verze GNU s dlouhými tvary přepínačů. Tradiční verze programu **tar** rozumí pouze jednopísmenným volbám. Verze GNU ale umí pracovat i se zálohami, které se nevejdou na jednu pásku nebo disketu, a s velmi dlouhými cestami k zálohovaným souborům. Ne všechny klasické verze programu **tar** tyto věci zvládají. (Operační systém Linux používá výhradně program **tar** ve verzi GNU.)

Jestli se záloha nevejde na jednu pásku, je potřeba zadat přepínač –multi-volume (-M):

```
# tar -cMf /dev/fd0H1440 /usr/src
tar: Removing leading / from absolute path names in the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
#
```

Nezapomeňte, že před zálohováním je potřeba diskety zformátovat. Stačí, když to uděláte v jiném okně, případně na jiném virtuálním terminálu ve chvíli, kdy program **tar** čeká na vložení další diskety.

Pokaždé, když ukončíte zálohování, měli byste zkontrolovat, zda proběhlo správně. Zadejte příkaz **tar** s parametrem -compare (-d):

```
# tar -compare -verbose -f /dev/ftape
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
....
#
```

Chyba v záloze znamená, že nic nepoznáte až do okamžiku, než budete potřebovat data ze zálohy obnovit.

Je-li kontrola záloh neúspěšná, nebudete moci v případě potřeby původní data z takovýchto záloh obnovit.

Inkrementální zálohování dělá program **tar** s parametrem -newer (-N):

```
# tar -create -newer '8 Sep 1995' -file /dev/ftape /usr/src -verbose
tar: Removing leading / from absolute path names in the archive
usr/src/
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/modules/
usr/src/linux-1.2.10-includes/include/asm-generic/
usr/src/linux-1.2.10-includes/include/asm-i386/
usr/src/linux-1.2.10-includes/include/asm-mips/
usr/src/linux-1.2.10-includes/include/asm-alpha/
usr/src/linux-1.2.10-includes/include/asm-m68k/
usr/src/linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
#
```

Bohužel, program **tar** neumí zjistit změny informací v inode souboru, například změny bitu vyhrazeného pro přístupová práva nebo změny jména souboru. Lze to obejít pomocí programu **find** a srovnávat stav dat uložených v souborovém systému se seznamem souborů, které byly posled-

ně zálohovány. Skripty a programy, které implementují tento způsob zálohování, najdete na různých linuxových FTP serverech.

Obnovování souborů programem tar

Zálohované soubory lze obnovit příkazem **tar** s parametrem `-extract (-x)`:

```
# tar -extract -same-permissions -verbose -file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

Můžete rovněž obnovovat jenom vybrané soubory či adresáře (a všechny soubory a podadresáře, které jsou v nich uloženy) tak, že je uvedete v příkazové řádce:

```
# tar xpvf /dev/fd0H1440 usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
#
```

Když vás zajímá jenom to, které soubory záloha obsahuje, zadejte příkaz **tar** s parametrem --list (-t):

```
# tar -list -file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

Uvědomte si, že program `tar` čte zálohu vždy sekvenčně, takže je při práci s většími objemy dat dost pomalý. Ale jestli používáte páskové jednotky nebo jiná média se sekvenčním přístupem, stejně nemůžete využít různé techniky, jež využívají náhodný přístup.

Program `tar` neumí správně zacházet se smazanými soubory. Potřebujete-li obnovit souborový systém z úplné a inkrementální zálohy a mezi těmito zálohami jste některý ze souborů smazali, po obnovení dat ze záloh bude smazaný soubor znovu existovat. To může být dost závažný problém například v případě, že soubor obsahuje citlivá data, která by již neměla být k dispozici.

Víceúrovňové zálohování

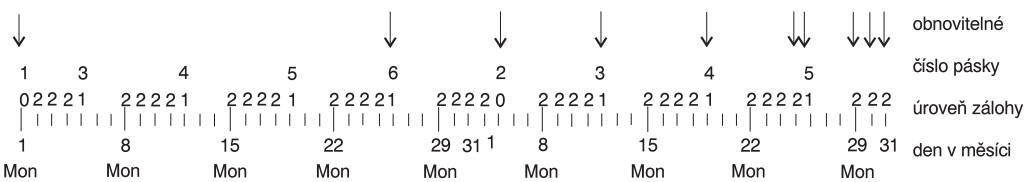
Metoda jednoduchého zálohování, jež byla načrtnuta v předchozím odstavci, je vhodná pro osobní potřebu nebo systémy s malým počtem uživatelů. Pro náročnější podmínky je vhodnější víceúrovňové zálohování.

Metoda jednoduchého zálohování má dvě úrovně: úplné a inkrementální zálohování. Ty lze zobecnit do libovolného počtu dalších úrovní. Úplná záloha by mohla být úrovní 0 a další následné inkrementální zálohy úrovněmi 1, 2, 3 a tak dále. Na každé úrovni inkrementálního zálohování se zálohuje vše, co se změnilo od poslední zálohy stejné nebo nižší úrovně.

Účelem víceúrovňového zálohování je levněji prodloužit historii zálohování dat. V příkladu v předchozím odstavci šla historie zálohování jenom k poslední úplné záloze. Kdybyste měli víc médií, mohli byste ji prodloužit, ale s každou další novou páskou jenom o týden, a to by bylo příliš nákladné. Delší historie vytváření záloh je užitečná, protože omylem smazaných nebo poškozených souborů si často všimnete až po delší době. Navíc jakákoliv verze souboru, i když není zrovna aktuální, je obvykle lepší než žádná.

Víceúrovňovým zálohováním lze historii vytváření archivů prodloužit levněji. Když si například koupíme deset pásek, můžeme používat pásky 1 a 2 na měsíční zálohy (první pátek každého měsíce), pásky 3 až 6 na týdenní zálohy (všechny ostatní pátky – uvědomte si, že v některém měsíci může být pět pátků, takže potřebujete o čtyři pásky víc) a pásky 7 až 10 na denní zálohy (od pondělí do čtvrtka). Takto jsme schopni – pouhým zakoupením čtyř dalších pásek navíc – prodloužit historii zálohování ze dvou týdnů (s využitím všech denních záloh) na dva měsíce. Pravda, během těchto dvou měsíců nemůžeme obnovit všechny verze zálohovaných souborů, avšak to, co obnovit lze, obvykle postačí.

Z obrázku 11.1 je patrné, která úroveň zálohování se používá v ten který den a z kterých záloh je možno na konci měsíce data obnovit.



Obrázek 11.1 – Příklad časového rozvrhu víceúrovňového zálohování

Víceúrovňové zálohování navíc snižuje i čas potřebný k obnovení celého souborového systému. Když potřebujete obnovit celý systém souborů a máte mnoho inkrementálních záloh s monotónně rostoucí řadou úrovní, musíte data pracně obnovovat postupně ze všech médií. Když ale budete používat méně záloh a větší počet úrovní, snížíte počet úrovní potřebných k obnovení celého svazku.

Chcete-li snížit počet médií potřebných k obnovení dat, používejte pro inkrementální zálohy menší rozsah úrovní. Tak se ovšem prodlouží čas zálohování, protože při každém zálohování se ukládá vše, co se změnilo od předchozí úplné zálohy. O něco lepší plán zálohování se uvádí v manuálové stránce programu **dump**. Jeho schéma je patrné z tabulky 11.1. Zkuste použít posloupnost úrovní zálohování: 3, 2, 5, 4, 7, 6, 9, 8, 9 a tak dále. Snížíte tím čas zálohování i obnovování dat. Nejdelší záloha se bude pořizovat po dvou dnech práce. Počet pásek, z nichž se budou obnovovat data, závisí na intervalu mezi úplnými zálohami, ale je rozhodně nižší než u procedury jednoduchého zálohování.

Páska	Úroveň	Zálohuje se (dní)	Obnova z pásek
1	0	-	1
2	3	1	1, 2
3	2	2	1, 3
4	5	1	1, 2, 4
5	4	2	1, 2, 5
6	7	1	1, 2, 5, 6
7	6	2	1, 2, 5, 7
8	9	1	1, 2, 5, 7, 8
9	8	2	1, 2, 3, 7, 9
10	9	1	1, 2, 5, 7, 9, 10
11	9	1	1, 2, 5, 7, 9, 10, 11
...	9	1	1, 2, 5, 7, 9, 10, 11, ...

Tabulka 11.1 – Efektivní zálohovací schéma při více úrovních

Tyto sofistikované postupy zálohování obvykle redukují pracnost, ale na druhou stranu je víc věcí, které jako správce systému musíte uhlídat. Sami se musíte rozhodnout, jestli se vám to vyplatí.

Co zálohovat

Je pochopitelné, že budete chtít zálohovat vše, co se vám na zálohovací média vejde. Výjimkou je software, jejž lze snadno obnovit z instalačních médií[44]. Ale aplikace mohou používat mnoho různých konfiguračních souborů a ty je lepší zálohovat, protože si tím ušetříte nelehkou práci, kterou zabere jejich opakované nastavování. Další významnou výjimkou je souborový systém `/proc`. Obsahuje pouze data, která jádro vytváří automaticky, a proto nemá žádný smysl je zálohovat. Zvláště nežádoucí je zálohovat soubor `/proc/kcore`, protože je to pouze aktuální obraz fyzické paměti, takže je dost velký.

Nerozhodné jsou zprávy news, poštovní schránky, logovací soubory a mnoho dalších dat uložených v adresáři `/var`. Sami se musíte rozhodnout, co pokládáte za důležité.

Typickým příkladem toho, co se musí zálohovat, jsou uživatelské soubory (adresář `/home`) a systémové konfigurační soubory (adresář `/etc` a další konfigurační údaje, které jsou často rozeseté po celém souborovém systému).

Komprimované zálohy

Zálohy zabírají hodně místa na disku, což může stát dost peněz. Nároky na diskový prostor lze minimalizovat komprimováním záloh. Existuje několik způsobů, jak to udělat. Některé programy přímo podporují kompresi, například po zadání parametru `-gzip` (`-z`) programu **tar** verze GNU se jeho výstup spojí pomocí roury s kompresním programem **gzip**. Záloha se pak komprimuje před tím, než se zapíše na zálohovací médium.

Bohužel, komprimované zálohy mohou způsobit vážné komplikace. Z povahy toho, jak komprimace funguje, vyplývá, že když se zapíše jediný bit špatně, bude nepoužitelný i celý zbytek komprimovaných dat. Některé zálohovací programy sice používají zabudované opravné algoritmy, ale žádná z těchto metod si neumí poradit s větším počtem chyb. Je-li tedy záloha komprimovaná

[44] Sami si rozhodněte, co je to „snadno“ – někdo označuje za snadnou instalaci z dvaceti disket.

způsobem, jakým to dělá program **tar** verze GNU, který svůj výstup komprimuje jako celek, může jediná chyba způsobit poškození celé zálohy. Stěžejním rysem zálohování musí být spolehlivost, takže tato metoda komprese není příliš dobrá.

Alternativou je komprimace jednotlivých záloh (souborů). Když se pak stane, že bude některý z nich poškozen, můžete použít jinou zálohu. Pravděpodobnost, že data, která tím ztratíte, se mohou poškodit i jiným způsobem, je stejná, takže na tom nebudete o moc hůř, než kdyby se nezálohovalo vůbec. Metodu komprimace jednotlivých souborů využívá například program **afio** (varianta programu **cpio**).

Komprese nějakou dobu trvá, takže zálohovací program nemusí být schopen data dostatečně rychle zapisovat na pásku. Tento problém lze řešit využitím vyrovnávacích pamětí pro výstup zálohovacích programů (buď interních, je-li zálohovací program natolik chytrý, nebo pomocí jiných programů). Ale i tak by se mohlo stát, že zálohování nebude fungovat správně. S tímto problémem byste se měli setkat jen u velmi pomalých počítačů.

Časové údaje

Čas je iluze. Čas oběda dvojnásob.
(Douglas Adams)

V této kapitole bude vysvětleno, jakým způsobem systém Linux udržuje správný čas a co je potřeba dělat, abyste potížím s nesprávným systémovým časem předešli. Obvykle nebudete muset dělat se systémovým časem nic, ale je užitečné chápat, oč jde.

Časové zóny

Měření času je založeno na převážně pravidelných přírodních jevech jako je střídání světla a tmy, jehož příčinou je rotace planety. Celkový čas mezi dvěma po sobě následujícími periodami je stejný, ale délka denní a noční doby se mění. Jedinou konstantou je poledne.

Poledne je denní doba, ve které se Slunce nachází na své nejvyšší pozici. Vzhledem k tomu, že Země je kulatá[45], je poledne na různých místech zeměkoule v jinou dobu. Z toho vychází představa *místního času*. Lidstvo měří čas v různých jednotkách, jichž většina je rovněž svázána s přírodními jevy, jako je ona kulminace Slunce v poledne. Pokud se nacházíte stále na stejném místě, nezáleží na tom, že je lokální čas na jiných místech jiný.

Když ale potřebujete komunikovat se vzdálenějšími místy, uvědomíte si zároveň, že potřebujete jakýsi společný čas. V dnešní moderní době potřebuje hodně míst komunikovat s různými jinými místy na celé planetě. Proto byl zaveden společný standard měření času. Tomuto společnému času se říká *univerzální čas* (anglicky *universal time*, zkráceně UT nebo UTC), dříve označovaný jako greenwichský čas (Greenwich Mean Time nebo GMT), protože je místním časem v Greenwichi v Anglii. Když spolu potřebují komunikovat lidé, kteří mají různý lokální čas, mohou používat společný univerzální čas. Nevzniká tak zmatek kolem toho, kdy se která věc stala nebo měla stát.

Místním časům se říká časové zóny. I když se z pohledu geografie zdá logické, aby byla místa, která mají poledne ve stejnou dobu, začleněna do stejného časového pásma, neumožňují to hlediska politická. Mnoho zemí z různých důvodů zavádí *letní čas* (anglicky *daylight savings time*, zkráceně DST). Posouvají ručičky hodinek tak, aby měly více denního světla v době, kdy se pracuje, pak je v zimě zase přetáčí zpět. Jiné státy to tak pro změnu nedělají. No a ty, které tak činí, nejsou zajedno v tom, kdy má být čas posunut a mění pravidla z roku na rok. To je důvod, proč jsou konverze časů mezi pásmy poměrně netriviální.

Časová pásma je nejlepší určovat podle polohy nebo podle rozdílu mezi místním a univerzálním časem. Ve Spojených státech a některých dalších zemích mají místní časové zóny své jméno a odpovídající třípísmennou zkratku. Ale tato zkratka není jedinečná a neměla by se používat bez současného označení země. Je lepší mluvit o lokálním čase například v Helsinkách, než o východo-

[45] Podle posledních výzkumů.

evropském čase (zkratka EET, East European Time), protože ne všechny státy východní Evropy leží ve stejném pásmu a nemusí mít stejná pravidla přechodu na letní čas.

Linux má balík programů pro práci s časovými pásmy. Tento software zná všechny existující časové zóny, a navíc jej lze jednoduše přizpůsobit v případě, že se změní pravidla určování času. Takže jediné, co musí správce systému udělat, je vybrat správné časové pásmo. Také každý z uživatelů si může nastavit své vlastní časové pásmo – to je důležité proto, že mnoho z nich používá prostřednictvím sítě Internet počítače v různých zemích. Když se ve vašem místním časovém pásmu změní pravidla přechodu na letní čas, budete muset aktualizovat tomu odpovídající část subsystému pro určování času. Stačí obvykle nastavit časové pásmo systému a upravit datové soubory zvoleného pásma, zbytek už nestojí za řeč.

Hardwarové a softwarové hodiny

Osobní počítač má hardwarové systémové hodiny napájené z baterií. Díky těmto bateriím hodiny fungují, i když je zbytek počítače bez proudu. Hardwarové systémové hodiny lze nastavovat pomocí programu setup v BIOSu, nebo přímo z operačního systému.
Jádro systému Linux udržuje vlastní čas nezávisle na hardwarových hodinách. Při zavádění operačního systému Linux nastaví své vlastní softwarové hodiny na stejný čas, jaký je v tom momentě na systémových hardwarových hodinách. Pak běží oboje hodiny nezávisle. Systém Linux udržuje vlastní čas pomocí softwarových hodin proto, že využívání systémových hardwarových hodin je dost pomalé a složité.

Hodiny jádra systému ukazují vždy univerzální čas. Proto jádro nemusí vůbec nic vědět o časových zónách – jednoduchost přispívá k vyšší spolehlivosti a ulehčuje možnost změny informací o vybraném časovém pásmu. Každý proces si převádí časová pásma sám (pomocí standardních nástrojů, jež jsou součástí balíku pro konverze časových zón).

Hardwarové systémové hodiny mohou být nastaveny na místní či univerzální čas. Je obvykle lepší mít je nastaveny na univerzální čas, protože pak není potřeba měnit nastavení hardwarových systémových hodin na začátku a konci období letního času (univerzální čas je totiž „pořád zimní“). Bohužel některé operační systémy pro PC – včetně systémů MS-DOS, Windows, OS/2 – počítají s tím, že hardwarové hodiny ukazují lokální čas. Systém Linux si umí poradit s oběma způsoby nastavení, ale v případě, že hardwarové hodiny ukazují místní čas, bude potřeba měnit jejich nastavení při přechodu z a na letní čas (jinak by totiž neukazovaly místní čas).

Zobrazení a nastavování času

V systému Debian je systémové časové pásmo určeno symbolickým odkazem `/etc/localtime`. Tento odkaz odkazuje na datový soubor pro dané časové pásmo, popisující místní časovou zónu. Jednotlivé datové soubory pro časová pásma jsou uloženy v adresáři `/usr/lib/zoneinfo`. Jiné distribuce Linuxu mohou parametry pro časová pásma nastavovat odlišně.

Uživatel může změnit své vlastní časové pásmo nastavením proměnné prostředí TZ. Není-li hodnota TZ nastavena, předpokládá se, že uživatel používá nastavení časového pásma systému. Syntaxe nastavení hodnoty proměnné TZ je popsána v manuálové stránce příkazu tzset.

Příkaz **date** ukáže aktuální datum a čas[46]. Například:

```
$ date
Sun Jul 14 21:53:41 EET DST 1996
$
```

[46] Neplést si s příkazem **time**, který *neukáže* aktuální čas.

Znamená to, že je neděle 14. července 1996, asi za deset minut deset večer, to všechno v časovém pásmu označeném „EET DST“, což by mohlo v angličtině znamenat „East European Daylight Savings Time“, tedy východoevropský letní čas. Příkazem **date**$ **date -u**

```
Sun Jul 14 18:53:42 UTC 1996
$
```

Příkazem `date` se také nastavují softwarové hodiny jádra systému:

```
# date 07142157
Sun Jul 14 21:57:00 EET DST 1996
# date
Sun Jul 14 21:57:02 EET DST 1996
#
```

Více podrobností hledejte v manuálové stránce příkazu `date` – syntaxe příkazu je tak trochu tajemná. Čas může nastavovat pouze superuživatel. I když může mít každý z uživatelů nastaveno své vlastní časové pásmo, systémový čas je pro všechny stejný.

Příkaz `date` ukazuje nebo nastavuje jenom softwarové hodiny. Příkaz `clock` synchronizuje systémové hardwarové a softwarové hodiny. Spouští se při zavádění systému, kdy se zjišťuje nastavení hardwarových systémových hodin. Podle nich se pak nastavují hodiny softwarové. Potřebujete-li nastavit oboje, nastavte nejprve softwarové hodiny příkazem `date`, následně hardwarové příkazem `clock -w`.

Parametrem `-u` sdělíte programu `clock`, že hardwarové hodiny ukazují univerzální čas. Přepínač `-u` je potřeba používat správně. V opačném případě bude mít váš systém mírný zmatek v tom, jaký je vlastně přesný čas.

Nastavení hodin se musí dělat opatrně. Mnoho částí operačního systému Unix spoléhá na to, že hodiny fungují správně. Například démon `cron` spouští pravidelně různé příkazy. Změníte-li nastavení hodin, může se stát, že nebude vědět, zda je potřeba některý z programů spustit, či nikoliv. Když na některém starším unixovém systému někdo nastavil hodiny o dvacet let dopředu, démon `cron` se snažil spustit všechny periodicky vykonávané příkazy za celých dvacet let naráz. Aktuální verze programu `cron` si s tímto problémem umí poradit. Přesto byste měli být při změnách času opatrní. Velké časové skoky a skoky vzad jsou nebezpečnější než menší změny a posuny dopředu.

Když jdou hodiny špatně

Softwarové hodiny systému Linux nejdou vždy přesně. V chodu je udržují pravidelná *přerušení časovače* generovaná hardwarem PC. Běží-li v systému příliš mnoho procesů, může trvat obsluha přerušení časovače příliš dlouho a softwarové hodiny se začnou zpožďovat. Hardwarové systémové hodiny běží nezávisle na operačním systému a jsou obvykle přesnější. Jestliže často vypínáte a zapínáte počítač (to je případ většiny systémů, které neslouží jako servery), budete mít obvykle i přesnější systémový čas.

Potřebujete-li upravit nastavení hardwarových hodin, je obyčejně nejjednodušší restartovat systém, spustit setup BIOSu a udělat to tam. Tak lze předejít všem potížím, které by mohly být zapříčiněny změnou systémového času za běhu systému. Nemáte-li možnost upravit čas v nastaveních systému BIOS, nastavte jej příkazy **date** a **clock** (v tomto pořadí), ale připravte se na to, že se možná některé části systému budou chovat podivně, takže budete muset systém opět restartovat.

Počítač připojený k síti (i když jenom pomocí modemu) může automaticky kontrolovat správnost nastavení vlastních hodin tak, že je bude srovnávat s nastavením hodin nějakého jiného počítače.

Jestli se o tomto jiném počítači ví, že jeho hodiny jdou velmi přesně, budou pak mít po připojení přesný čas oba systémy. Systémové časy dvou systémů lze seřídit příkazy **rdate** a **netdate**. Oba uvedené programy kontrolují čas na vzdáleném počítači (příkaz **netdate** i na několika vzdálených stanicích) a podle toho nastaví místní čas lokálního počítače. Počítač, na kterém se bude pravidelně spouštět některý z těchto příkazů, bude mít tak přesný čas, jak přesné budou hodiny vzdáleného počítače. *Pozn. korektora: Další možností je protokol NTP a program **ntupdate**.

Slovníček

Knihovník Neviditelné univerzity se rozhodl sám napomoci obecnému porozumění tak, že napíše orangutansko-lidský slovník. Pracoval na něm celé tři měsíce. Nebylo to lehké. Zatím se dostal jenom ke slůvku „oook“.
(Terry Pratchett, Muži ve zbrani)

Zde je stručný seznam slovních definicí pojmů spojovaných s operačním systémem Linux, zvláště pak se správou systému.

account

Viz účet

aplikační program

Software, který dělá něco užitečného. Výsledek používání aplikačního programu je v podstatě důvodem zakoupení počítače. Viz také systémový program, operační systém.

bad block

Viz vadný blok

bad sector

Viz vadný sektor

boot sektor

Typicky první sektor diskového oddílu. Obsahuje velmi krátký program (řádově stovky bajtů), který zajistí nahrání a spuštění operačního systému.

bootování

Cokoliv, co se děje od okamžiku zapnutí počítače po okamžik, kdy je počítač plně připraven k práci.

bootstrap loader

Velmi krátký program (umístěný typicky v paměti RAM), který přečte určitou oblast disku a předá jí řízení. Data v této oblasti uložená jsou obvykle trochu delší a složitější a dokáží zavést a spustit operační systém.

CMOS RAM

CMOS znamená „Complementary Metal Oxide Semiconductor“. Jde o složitou technologii, výsledkem je ale tranzistor, který si svůj stav pamatuje i po odpojení napájení, takže vzniká určitý typ statické paměti RAM. Tato paměť neztratí svůj obsah po vypnutí počítače.

cylindr

Skupina stop na vícepovrchovém disku, k nimž je možné přistupovat bez přesunu hlaviček. Umístěním dat, k nimž se typicky přistupuje současně, na jednom cylindru zrychluje přístup k disku, protože pohyb hlaviček je pomalejší v porovnání s rychlostí rotace disku.

daylight saving time

Letní čas.

démon

Proces číhající v pozadí – obvykle nenápadně – až do chvíle, než jej něco „rozjede“. Například démon **update** se probudí každých třicet sekund (nebo tak nějak) a vyprázdní buffer vyrovnávací paměti, démon **sendmail** se probere pokaždé, když někdo odesílá poštu.

emergency boot floppy

Viz záchranná disketa

formátování

Přísně vzato rozdělení povrchu disku na stopy, sektory a cylindry. Někdy se (nesprávně) za součást formátování považuje i operace vytvoření souborového systému na naformátovaném médiu.

fragmentace

Situace, kdy soubor není na disku zapsán v za sebou jdoucích blocích. Pokud na disku není dostatek souvislého místa k zapsání celého souboru, může být soubor zapsán ve více celcích. Tento jev se označuje jako fragmentace a vede ke zpomalenému přístupu k souboru.

geometrie (disku)

Kolik má disk povrchů, cylindrů a sektorů.

inkrementální záloha

Zálohovací metoda, při níž se na zálohovací médium zapíší pouze data změněná od poslední úplné zálohy.

inode

Datová struktura obsahující informace o souboru. Každý soubor má svůj inode, soubor je jednoznačně identifikován svým souborovým systémem a číslem svého inode. Každý inode obsahuje následující informace: zařízení, na němž se inode nachází, informace o zamykání, mód a typ souboru, počet odkazů na soubor, identifikátory vlastníka a skupiny souboru, délku souboru, časy přístupu a modifikace, čas modifikace inode a adresy bloků, v nichž je soubor uložen. Asociaci mezi soubory a čísly inode zajišťují adresáře. Inode souboru lze získat přepínačem **-i** příkazu **ls**.

jádro systému

Část operačního systému, která implementuje interakce s technickými prostředky výpočetního systému a sdílení zdrojů. Viz také systémový program.

kořenový souborový systém

Rodič všech souborových systémů v souborovém stromu Unixu. Pokud se nepodaří tento systém připojit, jádro zpanikaří a systém se nezavede.

logický diskový oddíl

Diskový oddíl vytvořený na rozšířeném oddílu, neexistuje fyzicky, pouze v logických softwarových strukturách.

lokální čas

Čas v dané lokalitě, odpovídající otáčení Země kolem osy.

MBR – Master Boot Record

První sektor na disku, kde BIOS hledá zavaděč operačního systému.

MTA – Mail Transfer Agent

Program zodpovídající za doručování e-mailových zpráv. Po přijetí zprávy od MTU nebo jiného MTA ji dočasně uloží, analyzuje adresáta a buď zajistí její lokální doručení, nebo ji předá jinému MTA.

MUA – Mail User Agent

Program umožňující uživateli vytvářet a číst zprávy elektronické pošty. Představuje rozhraní mezi uživatelem a mezi MTA.

NFS – Network File Systém

Protokol vyvinutý společností Sun, popsaný v RFC 1094, který umožňuje přístup k souborům přes síť stejně, jako by byly uloženy lokálně.

operační systém

Software, jenž umožňuje sdílení zdrojů počítačového systému (procesor, paměť, diskový prostor, síť a tak dál) mezi uživateli a aplikačními programy, které uživatelé spouští. Nabízí zabezpečení systému řízením přístupu k němu. Viz také jádro systému, systémový program, aplikační program.

POST – Power On Self-Test

Série diagnostických testů prováděná po zapnutí počítače. Zahrnuje testy paměti, testy hardwaru, testy zapsané konfigurace a podobně.

řadič disku

Hardwarové zařízení, překládající příkazy operačního systému přímo elektronice disku. Představuje abstrakční vrstvu, díky níž operační systém nemusí umět pracovat se všemi možnými typy disků, ale pouze s omezenou množinou řadičů. Typické řadiče jsou IDE a SCSI.

souborový systém

Metody a datové struktury používané operačním systémem k udržování přehledu o souborech na disku nebo oddílu, též způsob organizace disku.

účet

V Unixu mají uživatelé své *účty*. Ty představují jméno a heslo, jimiž se hlásí do systému, domovský adresář pro ukládání souborů, přístupová práva k hardwaru a softwaru – vše dohromady to tvoří účet.

úplná záloha

Zkopírování celého obsahu zálohovaného média na zálohovací médium.

souborový systém

Metody a datové struktury, které používá operační systém pro ukládání záznamů souborů na disk či diskovou oblast; způsob, kterým jsou soubory na disku organizovány. Pojem se používá též pro označení diskové oblasti či disku, kam se soubory ukládají, případně pro určení typu souborového systému.

systémový program

Programy, které implementují vyšší úrovně funkcionality operačního systému, tedy ty vlastnosti systému, které nejsou přímo závislé na hardwaru. Někdy mohou vyžadovat ke spuštění zvláštní oprávnění (například doručování elektronické pošty), ale velmi často jsou považovány za běžnou součást systému (například překladač). Viz také aplikační program, jádro systému, operační systém.

tisková fronta

Soubor nebo soubory, do nichž tiskový démon ukládá úlohy, které se mají vytisknout.

vadný blok

Blok disku (obvykle sektor), na němž nelze spolehlivě uložit data.

vadný sektor

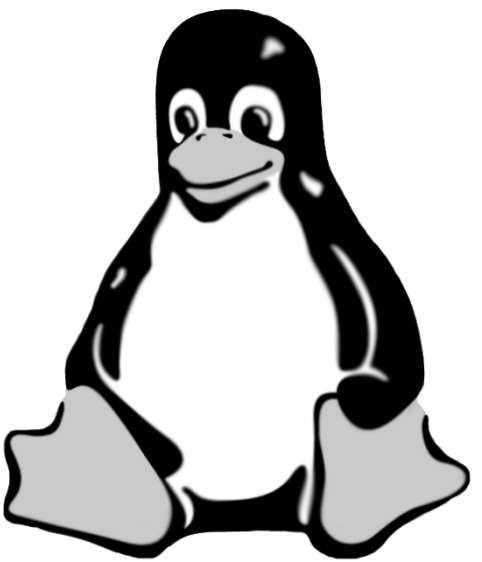
Prakticky totéž jako vadný blok, pouze o něco přesnější určení v případě, že velikost sektoru a bloku jsou různé.

volání systému

Služby, které poskytuje aplikačním programům jádro systému, a způsob, jímž jsou volány. Viz sekci 2 manuálových stránek.

záchranná disketa

Disketa, z níž je systém možné spustit i v případě poškození disku. Většina distribucí umožňuje vytvořit tuto disketu při instalaci, což vřele doporučujeme.



ČÁST III

Příručka správce sítě

Originál: http://www.tldp.org/LDP/nag2/

Úvod

Internet dnes v řadě zemí představuje v mnoha domácnostech běžnou věc. Stále více jinak normálních lidí nastupuje na cestu po informační superdálnici a počítačové sítě se tak dostávají na stejnou úroveň jako televizory nebo mikrovlnné trouby. Internet zažívá neuvěřitelnou míru mediální popularizace a řada společenskovědních oborů začíná považovat web, Usenet a virtuální realitu za nový fenomén internetové kultury.

Počítačové sítě jsou samozřejmě poměrně stará věc. Propojování počítačů a vytváření lokálních sítí bylo běžné už i v prostředích jen s několika počítači, a stejně obvyklé je využití telekomunikačních linek k realizaci dálkových propojování různých systémů. Prudce se rozrůstající společenství celosvětové sítě je nyní běžně dostupné i nekomerčním domácím uživatelům. Cenově přístupné je vytvořit si internetový server nabízející poštovní a konferenční služby prostřednictvím vytáčené nebo ISDN linky, s nástupem technologie DSL (Digital Subscriber Line) a kabelových modemů se tento trend bezpochyby jen rozšíří.

Mluvit o počítačových sítích velmi často znamená mluvit zároveň o Unixu. Unix samozřejmě není jediným operačním systémem se síťovými možnostmi, nebude také navždy představovat vedoucí standard v této oblasti, nicméně již dlouhou dobu se v prostředí počítačových sítí používá a jistě se ještě nějakou dobu používat bude.

Co je na Unixu zejména zajímavé pro domácí uživatele je to, že existuje řada aktivit, jak zajistit zdarma dostupný operační systém unixového typu pro platformu PC. Mezi tyto aktivity patří například 386BSD, FreeBSD a Linux.

Linux je zdarma distribuovaný klon operačního systému Unix, určený pro osobní počítače. V současné době funguje na řadě různých platforem, jako jsou rodina procesorů Intel, ale i procesory Motorola 680x0 a tedy počítače Commodore Amiga a Apple Macintosh, na počítačích Sun SPARC a Ultra-SPARC, Compaq Alpha, MIPS, PowerPC (například nová generace Apple Macintosh) a StrongARM (například rebel.com Netwinder nebo 3Com Palm). Linux byl přenesen i na některé méně rozšířené platformy, například Fujitsu AP-1000 a IBM System 390. Stále probíhá vývoj na přenesení Linuxu na další zajímavé platformy.

Linux byl vyvinut velkým týmem dobrovolníků spolupracujících po Internetu. Celý projekt zahájil v roce 1990 Linus Torvalds, finský vysokoškolák, jako projekt do předmětu Operační systémy. Od té doby se Linux stal plnohodnotným unixovým klonem, na němž běží tak rozdílné aplikace, jako jsou simulační a modelační programy, textové editory, systémy rozpoznávání řeči, webové prohlížeče a celá řada dalších programů včetně řady vynikajících her. Podporována je velmi široká škála hardwaru, a navíc Linux obsahuje úplnou implementaci protokolu TCP/IP včetně SLIP, PPP a firewallů, úplnou implementaci protokolu IPX a celou řadu dalších funkcí a protokolů, které v jiných operačních systémech nejsou. Linux je výkonný, rychlý a zadarmo. Jeho popularita v celém světě silně narůstá.

Samotný operační systém Linux je distribuován za podmínek licence GNU General Public Licence, tedy pod stejnou licencí, pod jakou jsou distribuovány programy vyvíjené Free Software Foundation. Tato licence komukoliv umožňuje programy dále distribuovat a modifikovat (ať už zdarma nebo za úplatu) za podmínky, že všechny modifikace a distribuce budou rovněž dále distribuovatelné zdarma. Termín „free software“ znamená možnost aplikaci volně šířit, nikoliv nutnost šířit ji zdarma.

Účel a orientace této části

Tato kniha vznikla se záměrem poskytnout síťovým administrátorům v prostředí Linuxu jednoduchou referenční příručku. Informace pokrývající prakticky všechny důležité operace prováděné síťovým administrátorem zde naleznou jak začátečníci, tak zkušení uživatelé. Samozřejmě rozsah pokrývané problematiky je tak široký, že není prakticky možné zmínit se o všem, co může být za nějakých okolností významné. Snažili jsme se popsat nejdůležitější a nejběžnější oblasti. Zjistili jsme, že i začátečníci pracující s Linuxem v síti byli schopni podle tohoto textu úspěšně nakonfigurovat síťové prostředí a dozvěděli se dost, aby se mohli věnovat i další specializované problematice.

Existuje celá řada knih a dalších zdrojů informací, kde se můžete dozvědět více o jakékoliv problematice pokryté v této knize (možná s výjimkou některých specificky linuxových funkcí, jako je například nové firewallové rozhraní, které ještě není dobře dokumentováno nikde). Připravili jsme pro vás přehled literatury, která by vám měla pomoci v dalším studiu.

Zdroje informací

Pokud s Linuxem pracujete nově, může pro vás být zajímavá celá řada dalších informací. Pomůže vám přístup k Internetu, i když není nezbytný.

Příručky Linuxového dokumentačního projektu

Linuxový dokumentační projekt (LDP) je skupina dobrovolníků, kteří pracují na tvorbě knih (příruček), dokumentů HOWTO a manuálových stránek, pokrývající širokou oblast témat od instalace po programování jádra. Práce skupiny LDP zahrnuje mimo jiné:

Linux Installation and Getting Started

Matt Welsh a kolektiv. Tato kniha popisuje jak Linux získat a nainstalovat. Obsahuje základní seznámení s Linuxem, s problematikou administrace systému, s X-Window System a se sítěmi.

Linux System Administrators Guide

Lars Wirzenius a Joanna Oja. Tato kniha představuje obecnou příručku administrace linuxového systému a pokrývá témata, jako je vytváření a konfigurace uživatelů, zálohování dat, konfigurace hlavních softwarových balíků a instalace a aktualizace různých programů.

Linux System Administration Made Easy

Steve Frampton. Tato kniha popisuje denní úkony administrace a údržby systému vztahující se k uživatelům.

Linux Programmers Guide

Scott Burkett, Sven Goldt, John D. Harper, Sven van der Meer a Matt Welsh. Tato kniha pokrývá tematiku týkající se vývoje aplikačního softwaru pro Linux.

The Linux Kernel

David A. Rusling. Tato kniha představuje úvod do problematiky jádra Linuxu, jak je vystavěno a jak funguje. Seznamte se s jádrem svého systému.

The Linux Kernel Module Programming Guide

Ori Pomerantz. Tato kniha vysvětluje jak vytvářet moduly jádra.

Další knihy jsou ve stadiu vzniku. Přesnější informace naleznete na webových stránkách LDP na adrese *http://www.linuxdoc.org* nebo na některém z četných zrcadel.

Dokumenty Linux HOWTO (praktické návody) představují vyčerpávající sérii dokumentů, které pokrývají různé aspekty systému – například jak instalovat a nakonfigurovat systém X Window, nebo jak pod Linuxem programovat v assembleru. Můžete je obecně najít v adresáři *HOWTO* na dále uvedených serverech FTP, nebo na webových stránkách některého z mnoha zrcadlících serverů LDP. Seznam dostupných dokumentů naleznete v přehledu literatury na konci této příručky nebo v souboru *HOWTO-INDEX*.

Mohou vás zajímat například dokumenty *Installation HOWTO*, popisující jak Linux nainstalovat, *Hardware Compatibility HOWTO*, obsahující seznam hardwaru, se kterým Linux pracuje, nebo *Distribution HOWTO*, jenž obsahuje seznam dodavatelů, již prodávají Linux na disketách a CD discích.

V přehledu literatury v této příručce jsou uvedeny dokumenty HOWTO, které se vztahují k síťové problematice.

Linux Frequently Asked Questions

Linux Frequently Asked Questions with Answers (FAQ) obsahují široký okruh otázek a odpovědí týkajících se systému. Tento dokument představuje nezbytnou literaturu pro všechny začátečníky.

Dokumentace dostupná přes FTP

Pokud můžete používat anonymní službu FTP, veškerou výše uvedenou dokumentaci můžete získat na řadě různých serverů, například *metalab.unc.edu:/pub/Linux/docs* nebo *tsx-11.mit.edu:/pub/linux/docs*.

Dokumentace dostupná přes WWW

Existuje celá řada webových stránek věnovaná problematice Linuxu. Domovskou stránku dokumentačního projektu najdete na adrese *http://www.linuxdoc.org*.

Open Source Writers Guild (OSWG) je projekt, jehož rozsah zahrnuje mimo jiné i Linux. Výsledkem projektu OSWG je například i tato příručka, cílem projektu je podpora volně distribuované dokumentace. Domovskou stránku tohoto projektu najdete na adrese *http://www.oswg.org:8080/oswg*.

Na obou uvedených adresách najdete hypertextové (a jiné) verze řady dokumentů týkajících se Linuxu.

Dokumentace dostupná komerčně

Literaturu týkající se Linuxu a LDP vydává celá řada nakladatelství a softwarových společností. Dvě z nich jsou:

Specialized Systems Consultants, Inc. (SSC)
http://www.ssc.com/
P.O. Box 55549 Seattle, WA 98155-0549
1-206-782-7733
1-206-782-7191 (FAX)
sales@ssc.com

a

Linux Systems Labs
http://www.lsl.com/
18300 Tara Drive
Clinton Township, MI 48036
1-810-987-8807
1-810-987-3562 (FAX)
sales@lsl.com

Obě společnosti vydávají v knižní podobě vybrané dokumenty HOWTO a další dokumentaci týkající se Linuxu.

Celou sérii knih věnovaných Linuxu vydalo nakladatelství O'Reilly & Associates. Některé z nich vycházejí z LDP, většina je však vydávána nezávisle na LDP. Patří sem například:

- *Running Linux* – Instalační a uživatelská příručka, popisující, jak co nejlépe využít osobní počítač se systémem Linux.
- *Learning Debian GNU/Linux, Learning Red Hat Linux* – Ještě základnější příručka než *Learning Linux*. Tyto knihy obsahují uvedené populární distribuce na CD discích a uvádějí podrobný návod pro jejich instalaci, nastavení a použití.
- *Linux in a Nutshell* – Další kniha z úspěšné série „v kostce" představuje referenční text týkající se Linuxu.

Linux Journal a Linux Magazine

Linux Journal a *Linux Magazine* jsou měsíčně vydávané časopisy pro linuxovou komunitu, které píše a vydává řada linuxových aktivistů. Obsahují články od návodů pro začátečníky až po podrobné popisy interních vlastností jádra. Pokud máte přístup k Usenetu, představují tyto časopisy dobrý způsob jak být v kontaktu s linuxovou komunitou.

Linux Journal je nejstarší a je vydáván společností S.S.C. Incorporated, o níž jsme se už zmínili. Tento časopis najdete i na webu na adrese *http://www.linuxjournal.com*.

Linux Magazine je novější, vydávaný nezávisle. Jeho domovskou stránku najdete na adrese *http://www.linuxmagazine.com*.

Linux Usenet Newsgroups

Pokud máte přístup k Usenetu, můžete využít následujících linuxových skupin:

- *comp.os.linux.announce* – Moderovaná diskusní skupina obsahující informace o nových programech, distribucích, chybách a dalších novinkách. Tuto skupinu by měli sledovat všichni uživatelé Linuxu. Přihlásit se můžete e-mailem na adrese *linux-announce@news.ornl.gov*.
- *comp.os.linux.help* – Skupina věnovaná obecně problematice instalace a použití Linuxu.
- *comp.os.linux.admin* – Diskuse věnovaná administraci systému Linux.
- *comp.os.linux.networking* – Diskuse týkající se síťové problematiky v Linuxu.
- *comp.os.linux.development* – Diskuse věnovaná vývoji jádra a celého systému.
- *comp.os.linux.misc* – Skupina pro všechna témata, která nezapadají do předchozích kategorií.

Existuje i několik diskusních skupin týkajících se Linuxu a vedených v jiných jazycích než v angličtině, například *fr.comp.os.linux* ve francouzštině nebo *de.comp.os.linux* v němčině.

Linuxové poštovní konference

Existuje celá řada specializovaných linuxových poštovních konferencí, kde můžete potkat mnoho lidí ochotných odpovědět vám na vaše otázky.

Nejznámější jsou konference provozované univerzitou Rutgers. Přihlásit se můžete odesláním e-mailu, který bude vypadat takto:

```
To: majordomo@vger.rutgers.edu
Subject: libovolný
Body:

subscribe název-konference
```

Některé z konferencí věnovaných problematice sítí jsou:

- *linux-net* – Diskuse týkající se sítí v Linuxu.
- *linux-ppp* – Diskuse týkající se implementace PPP v Linuxu.
- *linux-kernel* – Diskuse týkající se vývoje jádra. *Konference Linux-kernel je nyní provozována na serveru uger.kernel.org.

Online podpora

Existuje celá řada možností, jak kontaktovat online podporu, kterou poskytují dobrovolníci po celém světě a nabízejí rady a služby, jimiž pomáhají uživatelům s jejich otázkami a problémy.

Open-Source IRC Network je IRC služba věnovaná Open projektům jako jsou Open-Source a Open Hardware. Některé kanály této služby nabízejí online podporu pro Linux. IRC je zkratka z Internet Relay Chat, což je síťová služba, která vám umožňuje interaktivně si povídat s jinými lidmi na Internetu. IRC servery nabízejí různé kanály, na nichž si povídají různé skupiny lidí. Cokoliv, co v daném kanálu napíšete, si budou moci přečíst všichni uživatelé téhož kanálu.

Na Open-Source IRC existuje celá řada aktivních kanálů, kde 24 hodin denně, 7 dní v týdnu najdete lidi, kteří jsou ochotni vám pomoci vyřešit jakékoliv problémy týkající se Linuxu, nebo se kterými si budete moci prostě jen popovídat. Tuto službu můžete využívat, když si nainstalujete IRC klienta, jako je například *irc-II*, připojíte se na server *irc.openprojects.org:6667* a vstoupíte do kanálu *#linpeople*.

Uživatelské skupiny

Po celém světě existuje řada uživatelských skupin, které nabízejí přímou podporu. Řada těchto skupin poskytuje aktivity jako jsou instalační dny, schůzky a semináře, demonstrační noci a jiné společenské události. Uživatelské skupiny představují vynikající způsob, jak se kontaktovat s dalšími uživateli Linuxu ve vašem okolí. Existuje celá řada seznamů těchto skupin. Mezi nejznámější patří:

- Groups of Linux Users Everywhere – *http://www.ssc.com/glue/groups*
- LUG list project – *http://www.nllgg.nl/lugww*
- LUG registry – *http://www.linux.org/users*

Získání Linuxu

Neexistuje žádná jednotná podoba Linuxu, namísto toho existuje řada různých distribucí, jako jsou Debian, RedHat, Caldera, Corel, SuSE nebo Slackware. Každá z těchto distribucí obsahuje vše, co potřebujete ke spuštění úplného systému: jádro, základní nástroje, knihovny, podpůrné soubory a aplikační software.

Distribuce Linuxu můžete získat různými způsoby, například pomocí Internetu. Každá z hlavních distribucí nabízí vlastní FTP a webové servery. Například:

- Caldera – *http://www.caldera.com*, *ftp://ftp.caldera.com*
- Corel – *http://www.corel.com*, *ftp://ftp.corel.com*
- Debian – *http://www.debian.org*, *ftp://ftp.debian.org*
- RedHat – *http://www.redhat.com*, *ftp://ftp.redhat.com*
- Slackware – *http://www.slackware.com*, *ftp://ftp.slackware.com*
- SuSE – *http://www.suse.com*, *ftp://ftp.suse.com*

Na řadě všeobecných archivních FTP serverů jsou zrcadleny různé distribuce Linuxu. Mezi nejznámější servery patří:

metalab.unc.edu:/pub/Linux/distributions/
ftp.funet.fi:/pub/Linux/mirrors/
tsx-11.mit.edu:/pub/linux/distributions/
mirror.aarnet.edu.au:/pub/linux/distributions/

Řadu moderních distribucí je možné instalovat přímo z Internetu. Nicméně pro typickou instalaci musíte nahrát celou řadu programů, takže tuto variantu můžete zvolit pouze pokud máte rychlé a trvalé internetové připojení, nebo pokud chcete aktualizovat stávající instalaci[1].

Linux si můžete koupit na CD disku od celé řady prodejců softwaru. Pokud jej nemají ve vašem oblíbeném počítačovém obchodě, měli byste je požádat, ať si jej objednají. Na CD discích můžete dostat všechny populární distribuce. Někteří prodejci nabízejí celé sady disků, kde každý obsahuje jinou distribuci. Toto řešení je ideální, pokud si chcete vyzkoušet různé distribuce předtím, než se rozhodnete pro svou oblíbenou.

Standardy souborového systému

V minulosti byly různé distribuce a programové balíky určené pro Linux ovlivňovány tím problémem, že neexistoval žádný všeobecně uznávaný způsob organizace souborového systému. To vedlo k nekompatibilitám mezi různými balíky a komplikovalo to uživatelům a administrátorům práci při hledání různých souborů a programů.

K vyřešení této situace byla v srpnu 1993 založena skupina Linux File System Standard Group (FSSTND). Po půl roce debat vydala skupina návrh definující koherentní strukturu souborového systému a udávající umístění základních programů a konfiguračních souborů.

Bylo doporučeno tento standard implementovat v hlavních distribucích Linuxu. Trochu smůla spočívá v tom, že sice ve většině distribucí bylo dosaženo jisté shody s tímto standardem, nicméně jen velmi málo distribucí jej implementuje plně. V celé knize budeme předpokládat, že soubory, o nichž budeme hovořit, jsou umístěny tam, kde mají být podle standardu FSSTND. O možných alternativních umístěních se zmíníme pouze tam, kde existuje dlouhá tradice rozcházející se se standardem.

---

1 A nebo pokud jste extrémně netrpěliví a uvědomujete si, že 24 hodin, jež může trvat stažení souborů z Internetu, je podstatně kratší než 72 hodin, které může trvat dodání objednaného CD.

Standard FSSTND se dále vyvíjel a v roce 1997 byl nahrazen standardem Linux File Hierarchy Standard (FHS). Tento standard se zabývá i multiarchitekturovou problematikou, což standard FSSTND neřešil. FHS můžete nalézt v dokumentaci na všech větších FTP serverech věnovaných Linuxu a na jejich zrcadlech, nebo přímo na domovské stránce na adrese *http://www.pathname.com/fhs*. Koordinátora skupiny Daniela Quinlana můžete kontaktovat na adrese *quinlan@transmeta.com*.

Linux Standard Base

Velké množství distribucí Linuxu představuje výbornou možnost volby pro uživatele, nicméně to zároveň představuje problém pro programátory – zejména pro ty, kteří vyvíjejí komerční programy.

Každý distribuční balík obsahuje své základní knihovny, konfigurační nástroje, systémové aplikace a konfigurační soubory. Bohužel, rozdíly v jejich verzích, názvech a umístěních velmi komplikují snahu zjistit, co která distribuce obsahuje. Tím se stává velice obtížné vyvíjet binární aplikace, které budou spolehlivě fungovat ve všech distribucích.

Jako pomoc při řešení tohoto problému byl založen další projekt, pojmenovaný „Linux Standard Base“. Má za cíl definovat standardní složení distribuce, kterého by se měly všechny hlavní distribuce držet. Pokud programátor vytvoří aplikaci využívající tuto standardní základní platformu, bude aplikace fungovat na všech distribucích, které standard dodržují.

Informace o stavu projektu můžete najít na jeho domovských stránkách na adrese *http://www.linuxbase.org*.

Pokud vám záleží na univerzální použitelnosti programů, zejména u komerčně dodávaných programů, ověřte si, že vámi používaná distribuce se snaží vyhovět standardům podle tohoto projektu.

O této knize

Když se Olaf v roce 1992 přidal k práci na dokumentačním projektu, napsal dvě krátké kapitoly o UUCP a **smailu**, které měly být příspěvkem k příručce systémového administrátora. Vývoj TCP/IP byl v začátcích a když se tyto dvě malé kapitoly začaly rozrůstat, navrhl, že by možná bylo dobré vytvořit samostatnou příručku síťového administrátora. Všichni ho za ten nápad chválili a poradili mu, ať jej realizuje. Tak to udělal a v září roku 1993 vyšla první verze příručky síťového administrátora.

Olaf pokračoval v práci na této příručce a posléze vydal podstatně obsáhlejší verzi. Vince Skahan doplnil původní kapitolu o programu **sendmail**, která je v tomto vydání úplně přepracována vzhledem k novému konfiguračnímu rozhraní programu.

Příručka, kterou čtete, je revize a aktualizace ohlášená nakladatelstvím O'Reilly & Associates a provedená Terry Dawsonem[2]. Terry byl přes 20 let radioamatérem a více než 15 let pracoval v telekomunikačním průmyslu. Byl spoluautorem původního dokumentu NET-FAQ a je autorem a správcem řady dokumentů týkajících se sítí. Terry vždy nadšeně podporoval příručku síťového administrátora a doplnil do ní několik kapitol týkajících se různých síťových oblastí a podílel se i na její postupné aktualizaci tak, jak probíhal vývoj síťových možností Linuxu.

Kapitolu o programu **exim** napsal Philip Hazel[3], který je hlavním vývojářem a správcem tohoto balíku.

[2] Terryho Dawsona můžete kontaktovat na adrese *terry@linux.org.au*.

[3] Philipa Hazela můžete kontaktovat na adrese *ph10@cus.cam.ac.uk*.

Tato kniha je organizována přibližně chronologicky podle postupu, kterým se konfigurují síťové vlastnosti systému. Začíná popisem základní síťové problematiky, zejména sítí na bázi protokolů TCP/IP. Pak přechází od konfigurace TCP/IP na úrovni zařízení přes nastavení firewallu, účtování a konfigurace maškarády, až k nastavení obvyklých aplikací, jako jsou **rlogin** a podobné. Zbývající část knihy je věnována převážně dvěma hlavním aplikacím, které běží na protokolech TCP/IP: elektronické poště a zprávám.

Část věnovaná elektronické poště představuje úvod do obsáhlé problematiky přenosu a směrování pošty a různých adresačních schémat, která můžete potkat. Popisuje konfiguraci a správu programu **exim**, přenosového agenta, který je ideální ve většině případů, kdy se nepoužívá UUCP a programu **sendmail**, který umožňuje řešení i velmi složitých směrovacích situací zahrnujících protokol UUCP.

Samozřejmě žádná kniha nemůže nikdy úplně zodpovědět všechny otázky, které se mohou vyskytnout. Pokud se tedy stane, že se budete držet popsaného postupu a něco stále nebude fungovat, buďte trpěliví. Některé vaše problémy mohou být způsobeny chybou na naší straně (viz část *Oznamování změn* dále v tomto úvodu), mohou být ale způsobeny i změnami v používaných programech. Proto byste měli vždy nejprve využít uvedených informačních pramenů. Existuje velká šance, že se svým problémem nebudete osamoceni a že tedy bude známo řešení nebo alespoň obejití daného problému. Pokud máte možnost, měli byste vždy používat nejnovější jádro a síťové programy, které můžete získat na některém FTP serveru nebo BBS ve vašem okolí. Řada problémů bývá způsobena programy v různém stadiu vývoje, které spolu nedokáží korektně pracovat. Koneckonců, práce na vývoji Linuxu stále probíhají.

Oficiální tištěná verze

Na podzim roku 1993 požádal Andy Oram, který se účastní prací na projektu LDP od samého počátku, Olafa, aby tuto příručku vydal v nakladatelství O'Reilly & Associates. Ten s tím souhlasil, protože nikdy netušil, že se příručka setká s tak velkým úspěchem. Nakonec tedy Olaf a Andy souhlasili s tím, že O'Reilly vydá rozšířenou oficiální verzi této příručky s tím, že Olaf si ponechal svá autorská práva k ní a příručka proto může být volně distribuována. Můžete si tedy vybrat: různé verze tohoto dokumentu můžete najít na serverech zrcadlících obsah LDP, a ty si můžete vytisknout, nebo si můžete koupit tištěnou verzi v nakladatelství O'Reilly.

Proč byste tedy měli platit za něco, co můžete mít zadarmo? Že by se Tim O'Reilly definitivně zbláznil a vydal něco, co si může kdokoliv vytisknout a prodávat sám[4]. Jsou nějaké rozdíly mezi různými verzemi?

Odpovědi na jednotlivé otázky jsou „to záleží", „ne, určitě ne" a „ano i ne". O'Reilly & Associates na sebe vzali riziko vydání této příručky a zdá se, že se jim to vyplatilo, protože nás požádali o spolupráci na novém vydání. Domníváme se, že tento projekt může sloužit jako příklad toho, jak může svět free softwaru a komerční společnost spolupracovat na něčem, z čeho mají prospěch oba. Podle nás služba, kterou O'Reilly poskytuje linuxové komunitě (kromě toho, že si můžete koupit tuto knihu) spočívá také v tom, že přispívá k tomu, aby byl Linux chápán jako něco seriózního – jako životaschopná a užitečná alternativa ke komerčním operačním systémům. Špatné je to knihkupectví, které nemá alespoň jednu polici plnou O'Reillyho knih o Linuxu.

Proč tuto knihu vydali? Zdálo se jim, že to je „jejich" typ knihy. Je to kniha, která by vznikla, kdyby požádali nějakého autora o její napsání. Tempo výkladu, míra podrobností a styl odpovídá tomu, co běžně nabízejí.

4 Pozor! Elektronickou verzi příručky si samozřejmě můžete stáhnout a vytisknout, nicméně knihu z nakladatelství O'Reilly nesmíte prohnat kopírkou a eventuálně prodávat její kopie.

Smyslem licence LDP je to, aby nikdo nezůstal „mimo mísu“. Tuto příručku může vytisknout kdokoliv a nikdo se na vás nebude zlobit, pokud si ji pořídíte jinak. Pokud jste ale vytištěnou verzi neviděli, podívejte se na ni v knihkupectví nebo u kamaráda. Domníváme se, že se vám bude líbit a rádi si ji koupíte.

Jaké jsou rozdíly mezi tištěnou a elektronickou verzí? Andy Oram odvedl skvělou práci při přepisu našich nesourodých poznámek v něco, co stojí za vytištění. (Kromě toho revidoval i další knihy vydané v rámci LDP a svou prací významně přispěl celé linuxové komunitě.)

Jakmile Andy začal původní podobu této příručky upravovat, kniha se rychle vzdalovala od své původní podoby a s každou další konzultací se měnila více a více. Možnost pracovat s profesionálním redaktorem je něco, co by se mělo využívat. V mnoha směrech byl Andyho přínos stejný jako přínos autorů. To platí samozřejmě i o dalších lidech, kteří se podíleli na tom, aby kniha získala podobu, kterou vidíte. Všechny tyto úpravy byly promítnuty zpátky do elektronické verze, takže v jejich obsahu žádné rozdíly nejsou.

Nicméně tištěná verze *je* rozdílná. Je to profesionálně vytištěná vázaná kniha, a i když si dáte tu práci s vytištěním elektronické verze, nedosáhnete stejné kvality a pravděpodobně vás to vyjde dráž než koupě knihy. Navíc je v ní náš amatérský ilustrátorský styl nahrazen profesionální prací pracovníků nakladatelství O'Reilly. Kniha obsahuje rejstřík, který vám usnadní práci s ní. Pokud zamýšlíte celou příručku od začátku do konce přečíst, rozhodně vám doporučujeme její knižní podobu.

Přehled

Kapitola 1 hovoří o historii Linuxu a pokrývá základní síťovou problematiku protokolů UUCP, TCP/IP a dalších, hardwaru i bezpečnosti. Následující kapitoly se věnují konfiguraci Linuxu pro práci v sítích TCP/IP a práci s některými hlavními aplikacemi. Podrobněji o protokolu IP hovoříme v **kapitole 2**, a pak už se pouštíme do opravdové práce. Pokud víte, jak funguje směrování v protokolu IP a jak se provádí zjišťování adres, můžete tuto kapitolu přeskočit.

Kapitola 3 hovoří o úplně základní konfigurační problematice, jako je například sestavení jádra a nastavení síťové karty. Konfigurace sériových portů je popsána samostatně v **kapitole 4**, protože tato problematika se týká nejen TCP/IP.

Kapitola 5 popisuje nastavení počítače pro síť TCP/IP. Obsahuje návod pro konfiguraci samostatného počítače pouze s rozhraním *loopback*, i konfiguraci počítačů připojených k Ethernetu. Popisuje také několik užitečných nástrojů pro testování nastavení. **Kapitola 6** popisuje, jak nakonfigurovat rozlišování jmen a jak vytvořit name server.

Kapitola 7 popisuje, jak vytvořit SLIP spojení a nabízí podrobný popis programu **dip**, který vám umožňuje automatizovat většinu nezbytných kroků. **Kapitola 8** hovoří o PPP a programu **pppd**, démonu služby PPP.

Kapitola 9 navazuje na problematiku bezpečnosti a popisuje, jak vytvořit na Linuxu firewall a jak jej nakonfigurovat pomocí nástrojů **ipfwadm**, **ipchains** a **iptables**. Firewall umožňuje přesně nastavit to, kdo může k vaší síti přistupovat a které její počítače mu budou dostupné.

Kapitola 10 popisuje, jak nastavit IP účtování, takže budete moci zjistit, kde k jakým přenosům dochází a co je vyvolává.

Kapitola 11 hovoří o funkci nazvané IP maškaráda, která umožňuje připojit k Internetu celou síť prostřednictvím jediné adresy IP, takže všechny interní systémy budou před vnějškem skryty.

Kapitola 12 představuje úvod do nastavení některých nejvýznamnějších síťových aplikací, jako jsou **rlogin**, **ssh** a další. Hovoří také o tom, jak jsou různé služby obsluhovány superdémonem

inetd a jak můžete některé z hlediska bezpečnosti významné služby omezit jen na důvěryhodné počítače.

Zbytek knihy představuje podrobného průvodce elektronickou poštou a zprávami. **Kapitola 13** představuje základy problematiky elektronické pošty, například jak vypadá elektronická adresa a jak poštovní systém obsluhuje doručení pošty od odesilatele k adresátovi.

Kapitola 14 a **kapitola 15** hovoří o programech **sendmail** a **exim**, tedy o dvou transportních agentech, které se v Linuxu používají. Věnujeme se záměrně oběma, protože **exim** je jednodušší na instalaci a nastavení, zatímco **sendmail** obsahuje podporu UUCP.

Konvence používané v této části

Všechny příklady uváděné v této části předpokládají, že používáte **sh**-kompatibilní příkazový interpret (shell). Příkazový interpret **bash** je kompatibilní s **sh** a je standardním příkazovým interpretem ve všech distribucích Linuxu. Pokud používáte **csh**, budete muset příklady upravit.

Následuje přehled typografických konvencí, které v knize používáme:

Kurziva se používá pro názvy souborů a adresářů, programů a příkazů, řádkových voleb, e-mailových adres, URL a pro zvýraznění nových pojmů.

Tučné písmo používáme pro názvy počítačů, sítí, uživatelská jména a identifikátory, a příležitostně ke zvýraznění něčeho.

`Neproporcionální` písmo používáme pro výpisy programů, výstupy příkazů a k indikaci proměnných prostředí a klíčových slov, která se v kódu objevují.

`Neproporcionální kurziva` znázorňuje proměnné parametry, klíčová slova nebo text, který musí uživatel nahradit skutečnou hodnotou.

`Neproporcionální tučné` písmo znázorňuje příkazy nebo jiný text, který musí uživatel zapsat.

> **Varování** Takto označený text představuje varování. Popisuje situace, kdy chyba může vést k poškození systému nebo by se velmi obtížně opravovala.

Oznamování změn

Informace uvedené v knize jsme pečlivě testovali a ověřovali, nicméně můžete narazit na funkce, které se nějak změnily nebo i na naši chybu. Budeme rádi, když nás budete informovat o jakýchkoliv nalezených chybách i o doporučeních pro příští vydání. Můžete nám psát na adresu:

O'Reilly & Associates
101 Morris Street
Sebastopol, CA 95472
1-800-988-9938 (USA a Kanada)
1-707-829-0515 (mezinárodní a lokální)
1-707-829-0104 (fax)

Kromě toho nám můžete psát i elektronicky. Pokud se chcete přidat na seznam nakladatelství nebo si chcete nechat poslat katalog, napište na adresu:

info@oreilly.com

Technické otázky a komentáře ke knize posílejte na adresu:

bookquestions@oreilly.com

Tato kniha má svou domovskou stránku, na které najdete příklady, opravy a plány do budoucna. Najdete ji na adrese:

http://www.oreilly.com/catalog/linag2

Další informace o této i dalších knihách najdete na stránkách vydavatelství O'Reilly na adrese

http://www.oreilly.com

Poděkování

Toto vydání příručky síťového administrátora vděčí prakticky za všechno vynikající práci Olafa a Vinceho. Těžko se dá vysvětlit, jakou práci představuje napsání knihy jako je tato, to si musíte vyzkoušet sami. Aktualizace knihy byla náročná činnost, nicméně vycházeli jsme z dobrého základu, a proto to bylo příjemné.

Dále vděčíme mnoha lidem, kteří knihu přečetli a podíleli se na odstranění chyb, ať už technických nebo gramatických (nikdy jsem neslyšel o něčem takovém jako je přívlastek volný a těsný). Phil Hughes, John MacDonald a Eric Ratcliffe odvedli velmi záslužnou (a velmi konzistentní) práci nad obsahem knihy.

Chtěli bychom také poděkovat lidem z nakladatelství O'Reilly, se kterými byla radost pracovat. Byli to Sarah Jane Shangraw, která knize dala podobu, v níž ji vidíte, Maureen Dempsey, která redigovala text, Rob Romano, Rhon Porter a Chris Reilly, autoři obrázků, Hanna Dyer, která navrhla obálku, Alicia Cech, David Futato a Jennifer Niedherst, tvůrci sazby, Lars Kaufman, který navrhl na titulní stranu obrázek pařezu, Judy Hoer, autorka rejstříku a konečně Tim O'Reilly za odvahu tento projekt realizovat.

Vřelé poděkování si zaslouží Andres Sepúlveda, Wolfgang Michaelis, Michael K. Johnson a řada dalších programátorů, kteří strávili mnoho času nad ověřováním informací, které v knize najdete. Phil Hughes, John MacDonald a Eric Ratcliffe se řadou návrhů podíleli na obsahu druhého vydání. Děkujeme také všem, kteří četli první verzi této příručky a poslali nám opravy a připomínky. Snad úplný seznam přispěvatelů najdete v souboru `Thanks` v elektronické distribuci. A konečně, kniha by nemohla vzniknout bez Holgera Grotheho, který zajistil Olafovi připojení k Internetu, potřebné ke vzniku první verze knihy.

Olaf by chtěl poděkovat následujícím skupinám a lidem, kteří byli uvedeni v prvním vydání a sponzorovali ať už jej osobně nebo LDP jako celek. Byly to Linux Support Team, Erlangen, SRN; S.u.S.E. GmbH, Fuerth, SRN; Linux System Labs, Inc., Clinton Twp., USA a RedHat Software, North Carolina, USA.

Terry děkuje své manželce Maggie, která jej podporovala při práci na této knize i navzdory nárokům jejich čerstvě narozeného prvorozeného syna Jacka. Chtěl by také poděkovat všem lidem z linuxové komunity, kteří mu různými způsoby pomohli dostat se na úroveň, kdy sám může pomáhat jiným. „Pomůžu ti, pokud slíbíš, že příště pomůžeš ty někomu jinému."

Síň slávy

Kromě už uvedených osob se na příručce administrátora podílela i celá řada dalších lidí tím, že ji četli a poslali nám různé opravy a doporučení. Všem děkujeme.

Následující seznam uvádí jména těch, jejichž příspěvky dorazily do našich e-mailových schránek: Al Longyear, Alan Cox, Andres Sepúlveda, Ben Cooper, Cameron Spitzer, Colin McCormack, D. J. Roberts, Emilio Lopes, Fred N. van Kempen, Gert Doering, Greg Hankins, Heiko Eissfeldt, J. P. Szikora, Johannes Stille, Karl Eichwalder, Les Johnson, Ludger Kunz, Marc van Diest, Michael

K. Johnson, Michael Nebel, Michael Wing, Mitch D'Souza, Paul Gortmaker, Peter Brouwer, Peter Eriksson, Phil Hughes, Raul Deluth Miller, Rich Braun, Rick Sladkey, Ronald Aarts, Swen Thüemmler, Terry Dawson, Thomas Quinot a Yury Shevchuk.

KAPITOLA 1

Úvod do sítí

Historie

Myšlenka vytváření sítí je pravděpodobně stejně stará jako vlastní telekomunikace. Představte si lidi žijící v době kamenné, kteří si možná mezi sebou předávali zprávy pomocí bubnů. Dejme tomu, že jeskynní člověk A chtěl pozvat jeskynního člověka B na zábavnou hru, která spočívala v tom, že po sobě házeli kamením, ale žil příliš daleko na to, aby B slyšel jeho buben. Jaké měl tedy A volby? Mohl a) dojít za B, b) sehnat si větší buben nebo c) požádat C, který žil v polovině vzdálenosti jejich obydlí, aby zprávu předal. V případě poslední možnosti se jedná o vytvoření sítě.

Samozřejmě že od primitivních snah a prostředků našich předků jsme urazili již dlouhou cestu. Když si dnes chceme domluvit schůzku na sobotní fotbalový zápas[5], máme počítače, které spolu komunikují prostřednictvím soustavy drátů, optických vláken, mikrovln a podobně. V následujícím textu se budeme zabývat způsoby a metodami, kterými je to realizováno, opustíme však záležitosti kolem drátů i kolem fotbalu.

V tomto průvodci si popíšeme sítě na bázi protokolů TCP/IP, které jsou nejpopulárnější jak pro lokální sítě (LAN), tak i pro rozsáhlé sítě (WAN), jako je Internet.

Síť definujeme jako soubor *hostitelů*, kteří spolu mohou vzájemně komunikovat a často se přitom spoléhají na služby vyhrazených hostitelů, přenášejících data mezi jednotlivými účastníky. Hostitelé jsou často počítače, ale neplatí to vždy; někdy tak nazýváme i X-terminály nebo inteligentní tiskárny. Menším seskupením hostitelů říkáme *místa (site)*.

Komunikace by nebyla možná bez určitého druhu jazyka nebo kódu. V počítačových sítích se těmto jazykům souhrnně říká protokoly. Neměli byste si však představovat písemné protokoly, ale spíše vysoce formalizovaný kód chování, které lze pozorovat například při setkání hlav států. Podobně protokoly používané v počítačových sítích nejsou ničím jiným, než velmi přísnými pravidly pro výměnu zpráv mezi dvěma nebo více hostiteli.

Sítě TCP/IP

Moderní síťové aplikace vyžadují sofistikovaný způsob přenosu dat z jednoho počítače na jiný. Pokud spravujete linuxový stroj, který používá řada uživatelů, přičemž všichni současně mohou chtít připojit se ke vzdálenému počítači na jiné síti, musíte mít způsob, jakým budou moci sdílet vaše síťové připojení, aniž by se vzájemně ovlivňovali. Řešením používaným v řadě moderních síťových protokolů je *přepínání paketů* (packet-switching). *Paket* je malý blok dat, který je přenášen z jed-

5 Který se ještě ve své původní podobě hrává v Evropě.

noho počítače na jiný prostřednictvím sítě. K přepínání dochází, když datagram[6] přechází mezi různými linkami v síti. Síť s přepínáním paketů používá jedinou síťovou linku pro mnoho uživatelů, kteří mohou v různých okamžicích posílat pakety od jednoho uživatele k jinému.

Řešení realizované v systému Unix a následně i v celé řadě jiných systémů je nazýváno TCP/IP. Když se mluví o sítích TCP/IP, často narazíte na termín *datagram*, který má svůj specifický technický význam, nicméně často se používá jako ekvivalent slova *paket*. V této části budeme hovořit o základech, na nichž protokoly TCP/IP stojí.

Úvod do sítí TCP/IP

Protokol TCP/IP byl vyvinut v rámci výzkumného projektu financovaného americkou společností DARPA (Defense Advanced Research Projects Agency) v roce 1969. Jednalo se o experimentální síť zvanou ARPANET, která byla uvedena do operačního provozu v roce 1975, poté, co se ukázalo, že jde o úspěšné řešení.

V roce 1983 byla standardizována sada protokolů TCP/IP, kterou museli používat všichni hostitelé v této síti. Když se ze sítě ARPANET nakonec vyvinul Internet (přičemž síť ARPANET sama o sobě zanikla v roce 1990), rozšířilo se používání protokolu TCP/IP i v sítích mimo vlastní Internet. Řada společností má vlastní firemní sítě založené na protokolu TCP/IP a Internet se dostal do pozice, kdy může být klidně považován za obyčejnou konzumní technologii. Těžko dneska přečtete noviny nebo časopis, aby v nich nebyla zmínka o Internetu, který dnes může používat prakticky každý.

Abychom si ukázali konkrétnější využití protokolu TCP/IP, který budeme probírat v následujících statích, vezmeme si za příklad univerzitu GMU (Groucho Marx University), která se nachází někde ve Fredlandu. Většina kateder zde má vlastní lokální sítě, některé sdílejí jedinou a jiné jich zase mají více. Všechny jsou vzájemně propojeny a k Internetu jsou připojeny jednou vysokorychlostní linkou.

Dejme tomu, že váš stroj pracující pod Linuxem je připojen k lokální síti unixových hostitelů na katedře matematiky a jmenuje se **erdos**. Budete-li se chtít připojit k hostiteli na katedře fyziky, který se jmenuje řekněme **quark**, zadáte následující příkaz:

```
$ rlogin quark.physics
Welcome to the Physics Department at GMU
(ttyq2) login:
```

Na výzvu zadáte vaše uživatelské jméno, třeba `andres,` a vaše heslo. Pak se vám spustí příkazový interpret[7] na hostiteli `quark`, se kterým můžete pracovat stejným způsobem, jako byste seděli u konzoly tohoto systému. Až tento interpret opustíte, vrátíte se do prostředí vašeho vlastního počítače. Právě jste použili jednu z okamžitých interaktivních aplikací, které poskytuje protokol TCP/IP: vzdálené přihlášení.

Když jste připojeni ke stroji **quark**, budete asi chtít spouštět také nějaké aplikace s grafickým uživatelským rozhraním, například textový editor, kreslicí program nebo třeba prohlížeč WWW. Systém X Windows představuje plně síťově orientované grafické uživatelské prostředí, a je k dispozici pro řadu různých počítačových systémů. Aby příslušná aplikace věděla, že chcete mít její okna zobrazena na obrazovce vašeho hostitele, je třeba nastavit proměnnou prostředí DISPLAY:

```
$ DISPLAY=erdos.maths:0.0
$ export DISPLAY
```

6 Pozn. překladatele: *Datagram* zde můžeme chápat jako jiný termín pro paket. Drobné a víceméně formální rozdíly v tom, co je co, nás nemusí trápit.

7 Interpret nebo *shell* je řádkové rozhraní k operačnímu systému Unix. Podobá se příkazovému řádku DOSu v prostředí Microsoft Windows, je však mnohem výkonnější.

Když nyní spustíte požadovanou aplikaci, spojí se s X-serverem na vašem stroji a ne na stroji **quark** a zobrazí všechna svá okna na vaší obrazovce. K tomu je samozřejmě nutné, aby na počítači **erdos** běžel systém X11. Pro nás je teď zajímavé, že protokol TCP/IP umožňuje hostitelům **quark** a **erdos** vzájemnou výměnu paketů, což budí dojem, že pracujete v jednom systému. Síť je zde takřka transparentní.

Další velice důležitou vlastností sítí TCP/IP je souborový systém NFS (Network File System). Také on přispívá k transparentnosti sítě, protože umožňuje připojovat adresářové struktury z jiných hostitelů, jako kdyby to byly lokální souborové systémy, takže pak vypadají jako normální adresáře na vašem počítači. Například všechny domovské adresáře uživatelů mohou být na centrálním serveru, ze kterého si je připojují všichni ostatní hostitelé v lokální síti. V důsledku toho se uživatelé mohou přihlásit na libovolný stroj a získají vždy stejný domovský adresář. Podobně je možné sdílet velké objemy dat (například databáze, dokumentaci nebo aplikační programy) mezi mnoha počítači tak, že jediná kopie dat bude udržována na serveru a ostatní počítače k ní budou mít přístup. K souborovému systému NFS se ještě vrátíme v kapitole 14.

Samozřejmě, že to jsou jen příklady využití sítí založených na protokolu TCP/IP. Možnosti jsou takřka neomezené.

Nyní se podrobněji podíváme na způsob, jakým funguje protokol TCP/IP. Jeho znalost vám pomůže lépe pochopit jak a proč máte nakonfigurovat váš počítač. Začneme hardwarem a pomalu budeme postupovat směrem vzhůru.

Ethernety

Typu hardwaru, který se nejčastěji používá v lokálních sítích, říkáme *Ethernet*. V nejjednodušším případě je tvořen jedním kabelem, ke kterému jsou jednotliví hostitelé připojeni prostřednictvím konektorů, odboček nebo transceiverů. Instalace jednoduchého Ethernetu není příliš drahá, což je společně s jejich přenosovou rychlostí 10, 100 nebo i 1 000 megabitů za sekundu důvodem jejich oblíbenosti.

Ethernety existují ve třech provedeních, které nazýváme *tlustý*, *tenký* a *kroucená dvoulinka*. Tlustý a tenký Ethernet používají koaxiální kabel, který se liší šířkou a způsobem, jakým k němu lze hostitele připojit. Tenký Ethernet používá konektor „BNC“ ve tvaru písmene T, který vložíte do kabelu a zapojíte do zadní stěny vašeho počítače. Tlustý Ethernet vyžaduje vyvrtání malé díry do kabelu a připojení transceiveru za pomoci „vampire tap“. Tenký a tlustý Ethernet může být dlouhý maximálně 200, respektive 500 metrů, a proto se jim také říká 10Base-2 a 10Base-5. Termín „base“ je zkratka od „baseband modulation“ a jednoduše znamená, že data jsou posílána přímo na kabel bez použití modulace[8]. Číslo na začátku znamená přenosovou rychlost v megabitech za sekundu a číslo na konci je maximální povolená délka kabelu ve stovkách metrů.

Kroucená dvoulinka je kabel tvořený dvěma dvojicemi měděných vodičů a obvykle vyžaduje použití specializovaného hardwaru – *aktivního rozbočovače*. Kroucená dvoulinka se označuje také jako 10BaseT, kde „T“ znamená „twisted pair“. 100megabitová verze se pak označuje jako 100BaseT.

Pro připojení nového počítače k tenkému Ethernetu je třeba alespoň na několik minut přerušit síťové spojení, protože musíte kabel přeříznout, abyste do něho mohli vložit konektor. Připojení počítače k tlustému Ethernetu je poněkud komplikovanější, typicky však nemusíte přerušit síť. Síť založená na kroucené dvoulince to má ještě jednodušší. Používá zařízení označované jako rozbočo-

8 Pozn. překladatele: Kromě „base“ přenosů se používají ještě „broad“ přenosy – od slovíčka broadband, při nich jsou data při odesílání modulována na nějakou nosnou frekvenci a při příjmu se musí demodulovat. Příkladem této technologie je 10Broad36, Ethernetová kabeláž vedená 75Ω koaxiálním kabelem (tedy klasickým „televizním kabelem“), používaná zejména v technologických systémech.

vač (hub), který slouží jako stykový uzel. Počítače můžete k rozbočovači připojovat a odpojovat, aniž byste jakkoliv ovlivnili ostatní.

Většina lidí upřednostňuje pro malé sítě tenký Ethernet, protože je velice levný: síťové karty seženete už za 500 korun (některé společnosti je dnes prakticky vyhazují) a kabel stojí jen několik korun za metr. Nicméně pro rozsáhlejší instalace je vhodnější tlustý Ethernet nebo kroucená dvoulinka. Ethernet na katedře matematiky univerzity GMU byl nejprve řešen tlustým koaxiálem, protože se jednalo o rozvod na poměrně velkou vzdálenost a nebylo nutné přerušovat síť při každém přidávání nového počítače. Ve většině instalací je dnes nejpopulárnější kroucená dvoulinka. Ceny rozbočovačů klesají a menší zařízení je dnes možné koupit za cenu přijatelnou i pro domácí uživatele. Instalace kroucenou dvoulinkou je pro rozsáhlejší sítě výrazně levnější a se samotným kabelem se pracuje mnohem snáze, než s koaxiálními kabely v jiných typech instalací. Správci sítě na katedře matematiky GMU tak v příštím roce plánují přechod na kroucenou dvoulinku, protože se jedná o moderní technologii, která jim ulehčí práci při přidávání nových počítačů a odpojování starých.

Jednou z nevýhod ethernetové technologie je omezená délka kabelu, což vylučuje jeho použití pro jiné než lokální sítě. Nicméně pomocí opakovačů, mostů a směrovačů může být pospojováno několik ethernetových segmentů. Opakovače prostě jen kopírují signály mezi dvěma nebo více segmenty, takže všechny segmenty dohromady působí dojmem, jako by se jednalo o jediný segment. Vzhledem k požadavkům na časování nemohou mezi libovolnými dvěma počítači v síti ležet více než čtyři opakovače. Mosty a směrovače jsou chytřejší. Analyzují příchozí data a předávají je dále pouze v případě, kdy příjemce není připojen na stejném segmentu jako odesilatel.

Ethernet funguje jako sběrnice, po které může hostitel posílat pakety (nebo *rámce*) až do velikosti 1 500 bajtů jinému hostiteli ve stejném Ethernetu. Adresa hostitele je dlouhá šest bajtů a je napevno zakódována do firmwaru jeho ethernetové desky. Takovéto adresy jsou zpravidla zapisovány ve tvaru dvojic hexadecimálních číslic oddělených dvojtečkami, například: **aa:bb:cc:dd:ee:ff**.

Rámec poslaný jednou stanicí vidí všechny připojené stanice, ale pouze cílová stanice si ho vyzvedne a zpracuje. Pokud se ve stejnou dobu pokusí vysílat dvě stanice, dojde ke *kolizi*. Elektronika síťových karet kolizi velmi rychle detekuje a řeší ji tak, že obě stanice vysílání přeruší a po náhodně zvolené době to zkusí znovu. Určitě uslyšíte hodně o tom, jak problematické jsou kolize na Ethernetu a že díky nim je využitelnost Ethernetu pouhých 30 % jeho přenosové kapacity. Kolize jsou na Ethernetu *normální* jev a na velmi zatížených sítích vás nemůže překvapit, že kolize představují až 30 % přenosů. Realističtější odhad využitelnosti přenosové kapacity je 60 %, a pokud vám to stačí, nemusí vás kolize trápit[9].

Další typy hardwaru

U větších instalací, jako je například Groucho Marx University, se zpravidla kromě Ethernetu používá ještě jiný typ vybavení. Existuje a používá se celá řada dalších komunikačních technologií.Všechny popsané protokoly jsou Linuxem podporovány, nicméně vzhledem k omezení délky tohoto textu je popíšeme jen velmi stručně. Pro řadu různých protokolů existují dokumenty HOWTO, které je popisují podrobně, takže pokud vás zajímá něco, o čem tady nebudeme hovořit, můžete vyzkoušet tyto dokumenty.

Na univerzitě GMU jsou všechny lokální sítě jednotlivých kateder připojeny k univerzitní páteři, což je optický kabel s rozhraním FDDI (*Fiber Distributed Data Interface*). Toto rozhraní používá při přenosu dat úplně jiný přístup, který spočívá v rozeslání několika *tokenů* a stanice může vysílat data pouze tehdy, pokud vlastní nějaký token. Hlavní výhodou technologie s předáváním to-

9 O této problematice hovoří Ethernet FAQ na adrese *http://www.faqs.org/faqs/LANs/ethernet-faq/*, podrobné historické a technické informace naleznete na stránkách Charlese Spurgeona věnovaných Ethernetu na adrese *http://wwwhost.ots.utexas.edu/ethernet/*.

kenů je to, že nedochází ke kolizím. Protokol tak může mnohem snáze využít celou kapacitu přenosového média, která je v případě FDDI 100 megabitů za sekundu. Technologie FDDI založená na optickém kabelu může mít maximální délku kabelu mnohem větší než u klasických „drátěných" technologií. Délkový limit je 200 km, což je ideální při propojování více budov ve městě, nebo jako v případě GMU, k propojení mnoha budov univerzitního areálu.

Podobně pokud se používá vybavení společnosti IBM, velmi pravděpodobně bude instalována síť IBM Token Ring. Token Ring se v některých lokálních sítích používá místo Ethernetu a nabízí podobné výhody jako FDDI, protože plně využívá přenosovou kapacitu média (která je ovšem v tomto případě nižší, 4 Mb/s nebo 16 Mb/s), je ovšem levnější, protože místo optického kabelu používá klasický kabel. V Linuxu se sítě Token Ring konfigurují prakticky stejně jako Ethernet, proto se jim nebudeme specificky věnovat.

I když dnes už to je méně pravděpodobné než dříve, můžete potkat i jiné technologie lokálních sítí, například ArcNet nebo DECNet. Obě Linux podporuje, nebudeme se však jimi zabývat.

Mnoho národních sítí provozovaných telekomunikačními operátory podporuje protokoly s přepínáním paketů. Pravděpodobně nejpopulárnější je standard nazvaný X.25. Tuto službu nabízí mnoho takzvaných veřejných datových sítí, jakou je například Tymnet v USA, Austpac v Austrálii nebo Datex-P v Německu. Standard X.25 definuje sadu protokolů, kterými komunikují terminálová zařízení (například počítače) s komunikačními zařízeními (přepínač X.25). X.25 vyžaduje synchronní datovou linku, a tedy i speciální synchronní sériové komunikační zařízení. Linku X.25 můžete používat i na normálním sériovém portu, pokud použijete speciální zařízení označované jako PAD (Packet Assembler Disassembler). PAD je externí zařízení, které obsahuje asynchronní sériový port a synchronní sériový port. Obstarává komunikaci protokolem X.25, takže se k němu mohou připojit i velmi jednoduchá terminálová zařízení. Protokol X.25 se často používá k přenosu jiných síťových protokolů, například TCP/IP. Protože IP pakety nelze jednoduše namapovat na pakety X.25 (nebo naopak), jsou IP pakety před odesláním jednoduše zapouzdřeny do paketů X.25 a pak odeslány po síti. Pro Linux je dostupná experimentální implementace protokolu X.25.

Novější protokol často nabízený telekomunikačními společnostmi se označuje *Frame Relay*. Protokol Frame Relay má s protokolem X.25 společnou řadu technických podrobností, jinak se ale více podobá protokolu IP. Stejně jako X.25 potřebuje Frame Relay speciální synchronní sériové zařízení. Vzhledem k této podobnosti podporuje řada karet oba protokoly. Alternativou nevyžadující žádný speciální interní hardware je opět použití externího zařízení označovaného jako Frame Relay Access Device (FRAD), které zapouzdřuje ethernetové pakety do paketů Frame Relay a posílá je po síti. Frame Relay je ideální pro přenos TCP/IP mezi různými lokalitami. Linux obsahuje ovladače, které podporují některé typy interních Frame Relay karet.

Pokud potřebujete vysoce rychlý spoj, který bude přenášet různé typy dat, například digitalizovaný obraz nebo zvuk i normální data, bude vás zřejmě zajímat technologie ATM (Asynchronous Transfer Mode). ATM je nová síťová technologie navržená speciálně k poskytování dobře spravovatelných rychlých linek s malým zpožděním, které umožňují řízení kvality služby. Řada telekomunikačních společností buduje infrastrukturu linek ATM, protože jim umožní přenesení různých síťových služeb na jednu platformu a ušetří tak náklady na správu a údržbu. ATM se velmi často používá k přenosu protokolu TCP/IP. Podpoře ATM v Linuxu se věnuje dokument Networking -HOWTO.

Radioamatéři často používají své vybavení k vzájemnému propojení svých počítačů; tomu říkáme *paketové rádio*. Jeden z protokolů používaný paketovými rádii se jmenuje AX.25, který je volným derivátem protokolu X.25. Protokol AX.25 se pak používá k přenosu protokolu TCP/IP i jiných. AX.25 stejně jako X.25 vyžaduje ke své práci synchronně pracující sériový hardware nebo externí zařízení označované jako Terminal Node Controller, který konvertuje pakety vysílané asynchron-

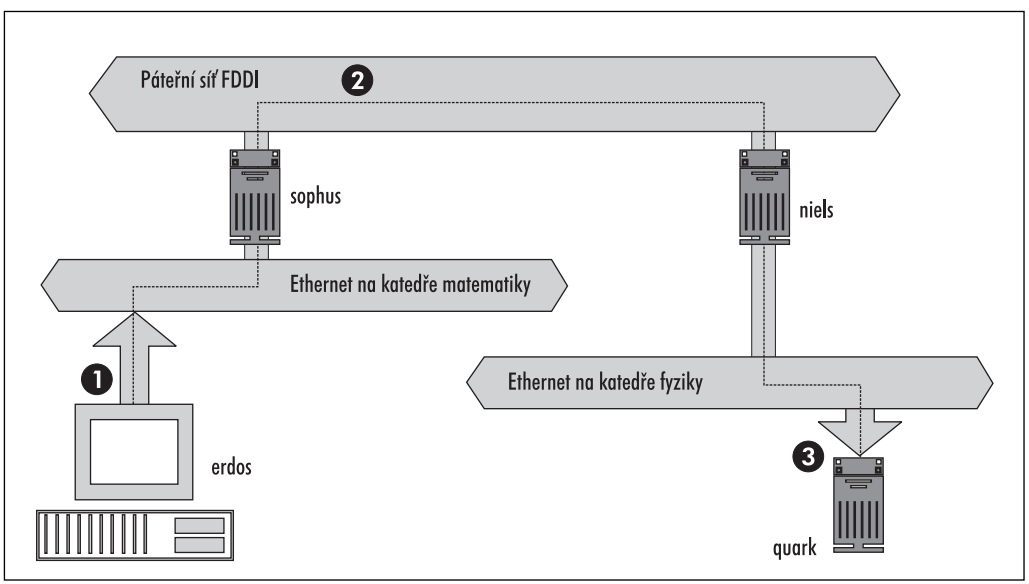
ní linkou na linku synchronní. Existuje celá řada různých karet podporujících paketové rádio, obecně se označují jako „Z8530 SCC based“, což je název nejpopulárnějšího řadiče, od nějž byly odvozeny. Protokolem AX.25 se často přenášejí ještě protokoly NetRom a Rose, což jsou protokoly síťové vrstvy. Protože využívají protokolu AX.25, mají stejné požadavky na hardware. Linux obsahuje úplnou podporu protokolů AX.25, NetRom a Rose. Dobrým zdrojem informací o implementaci těchto protokolů je dokument AX.25-HOWTO.

Další typ spojení vyžaduje připojení k centrálnímu systému pomalou, nicméně levnou sériovou linkou (telefonní, ISDN a podobně). To vyžaduje další protokol pro přenos paketů, například SLIP nebo PPP, které si popíšeme dále.

Internet Protocol

Samozřejmě nechcete, aby byla vaše síť omezena jen na Ethernet nebo na dvoubodovou datovou linku. V ideálním případě budete chtít komunikovat s hostitelským počítačem bez ohledu na to, k jaké fyzické síti je připojen. Ve větších instalacích, jakou mají například na Groucho Marx University, existuje zpravidla několik oddělených sítí, které jsou nějakým způsobem propojeny. Na GMU fungují na katedře matematiky dva Ethernety: první je tvořen rychlými počítači, které využívají profesoři a absolventi, a druhý spojuje pomalejší počítače, na kterých pracují studenti. Obě sítě jsou připojeny k univerzitní páteřní síti FDDI.

Toto připojení zajišťuje vyhrazený hostitel, takzvaná brána, která obsluhuje příchozí a odchozí pakety tím způsobem, že je kopíruje mezi dvěma Ethernety a optickým kabelem sítě FDDI. Sedíte -li například na katedře matematiky a chcete se z vašeho linuxového stroje připojit k počítači **quark**, který je připojen do lokální sítě na katedře fyziky, nemůže síťový software poslat pakety přímo hostiteli **quark** na katedře fyziky, protože není připojen do stejného Ethernetu. Musí se spolehnout na bránu, která funguje jako dopravce. Brána (její jméno je **sophus**) pak za pomoci páteřní sítě tyto pakety předá partnerské bráně **niels** na katedře fyziky, která je doručí cílovému počítači. Na obrázku 1.1 je znázorněn tok dat mezi počítači **erdos** a **quark**.



Obrázek 1.1 – Tři kroky při posílání datagramu z počítače erdos počítači quark

Tomuto schématu doručování dat ke vzdálenému hostiteli se říká směrování a paketům v tomto kontextu říkáme datagramy. Aby to bylo jednodušší, je výměna datagramů řízena protokolem, který je nezávislý na použitém hardwaru: protokolem IP, iternetovým protokolem. Tímto protokolem a otázkami týkajícími se směrování se budeme podrobněji zabývat v kapitole 2.

Hlavní užitek protokolu IP spočívá v tom, že spojuje fyzicky rozdílné sítě do jedné na pohled homogenní sítě. Tomu říkáme internetworking (spojování více počítačových sítí do jedné) a výsledné „metasíti" říkáme internet. Všimněte si zde jemného rozdílu mezi slovy internet a Internet. Druhé z nich je název konkrétního globálního internetu.

Samozřejmě, že protokol IP vyžaduje adresové schéma, které je nezávislé na hardwaru. Toho dosáhneme tím, že každému hostiteli přidělíme jedinečné 32bitové číslo nazývané IP adresa. IP adresa se zpravidla zapisuje jako čtyři desítková čísla (každé z nich udává jednu 8bitovou část) oddělená tečkami. Například počítač **quark** by mohl mít IP adresu **0x954C0C04**, což bychom zapsali jako **149.76.12.4**. Tomuto formátu zápisu také říkáme tečková notace. Velmi často se označuje také jako IPv4 (jako Internet Protocol, Version 4), protože se připravuje nová verze IPv6, která nabízí podstatně pružnější možnosti adresování plus řadu dalších funkcí. Nicméně bude trvat ještě alespoň rok od vydání této knihy, než se protokol IPv6 dostane k normálnímu použití.

Všimněte si, že nyní tady máme tři různé typy adres: nejprve je zde název hostitele, **quark**, potom jeho IP adresa a konečně hardwarová adresa typu 6bajtové ethernetové adresy. Všechny tyto adresy si musí určitým způsobem odpovídat, takže když zadáte **rlogin quark**, může být síťovému softwaru předána IP adresa hostitele **quark** a když protokol IP doručí data do Ethernetu katedry fyziky, musí nějakým způsobem zjistit, která ethernetová adresa této IP adrese odpovídá.

K tomuto tématu se vrátíme v kapitole 2. Nyní nám bude stačit, když si zapamatujeme, že tomuto procesu získávání adresy mapováním názvu hostitele na IP adresu říkáme rozlišování názvu hostitele *(*hostname resolution*) a mapování na hardwarovou adresu říkáme* rozlišování adresy *(*address resolution*).*

Serial Line Internet Protocol

Na sériových linkách je de facto standardem protokol SLIP, neboli *Serial Line IP*. Modifikovaný typ protokolu SLIP se nazývá CSLIP, neboli *compressed SLIP*, a provádí kompresi IP hlaviček, aby bylo možné lépe využít relativně malou šířku pásma, kterou poskytují sériové linky. Dalším sériovým protokolem je PPP, neboli *Point-to-Point Protocol*. Jedná se o modernější verzi protokolu SLIP a obsahuje řadu funkcí, které jej dělají zajímavým. Jeho hlavní výhodou oproti protokolu SLIP je to, že může přenášet nejen datagramy protokolu IP, ale prakticky jakýkoliv protokol.

Transmission Control Protocol

Samozřejmě, že odesláním datagramů z jednoho hostitele na druhý celý proces nekončí. Přihlásíte-li se k počítači **quark**, chcete, aby bylo spojení mezi vaším procesem *rlogin* na hostiteli **erdos** a příkazovým interpretem na hostiteli **quark** spolehlivé. Proto musí odesilatel informaci posílanou z jednoho počítače na druhý rozdělit na pakety a příjemce pak z nich musí znovu sestavit proud znaků. Vypadá to sice jednoduše, ale celý tento proces v sobě zahrnuje několik náročných úkolů.

Předně je důležité vědět, že cílem protokolu IP nikdy nebylo zajistit spolehlivost. Představte si, že by deset lidí připojených k vaší ethernetové síti začalo stahovat zdrojový kód poslední verze prohlížeče Netscape z FTP serveru GMU. Síťový provoz, který by tyto operace vyvolaly, by brána nemusela zvládnout, protože je příliš pomalá a má nedostatek paměti. Když teď pošlete z počítače **quark** nějaký paket, nemusí mít brána **sophus** dostatek místa ve vyrovnávací paměti a tím pá-

dem ho nebude moci předat dál. Protokol IP řeší tento problém tak, že příslušný paket zruší. Paket je tak neodvolatelně ztracen. Proto závisí na komunikujících hostitelích, aby kontrolovali integritu a úplnost dat a v případě výskytu chyby přenos opakovali.

K tomu se používá další protokol zvaný TCP, neboli *Transmission Control Protocol*, který vytváří spolehlivou službu nad protokolem IP. Základní vlastností protokolu TCP je, že za pomoci protokolu IP vytváří iluzi jediného spojení mezi příslušnými dvěma procesy na vašem hostiteli a vzdáleném počítači, takže se nemusíte starat jakým způsobem a po jaké trase vaše data ve skutečnosti putují. TCP spojení v podstatě funguje jako dvousměrná roura, do které mohou oba procesy zapisovat i z ní číst. Lze to přirovnat k telefonnímu hovoru.

Protokol TCP rozpozná koncové body takového spojení podle IP adresy příslušných dvou hostitelů a čísla takzvaného *portu* na každém z hostitelů. Na porty je možné se dívat jako na přípojky síťových spojení. Máme-li znovu použít podobnost s telefonním spojením, lze IP adresu přirovnat ke směrovému číslu (směrují čísla na určité město) a čísla portů pak k místním telefonním číslům (směrují čísla na jednotlivé telefony). Jeden hostitel může poskytovat mnoho různých služeb, které jsou od sebe odlišeny čísly portů.

V našem příkladu otevře klientská aplikace (**rlogin**) port na hostiteli **erdos** a spojí se s portem 513 na hostiteli **quark**, o kterém se ví, že na něm poslouchá server **rlogind**. Tím je navázáno TCP spojení. Za pomoci tohoto spojení provede server **rlogind** autorizaci uživatele a potom spustí příkazový interpret. Standardní vstup a výstup tohoto příkazového interpretu jsou přesměrovány na TCP spojení, takže cokoliv napíšete na vašem počítači, bude proudem TCP přeneseno a předáno na standardní vstup příkazového interpretu.

User Datagram Protocol

Samozřejmě, že protokol TCP není jediným uživatelským protokolem v sítích TCP/IP. I když je vhodný pro aplikace typu **rlogin**, jeho vysoká režie jej neumožňuje používat pro aplikace typu NFS. Místo toho se používá spřízněný protokol UDP, neboli *User Datagram Protocol*. Podobně jako TCP i protokol UDP dovoluje aplikaci kontaktovat službu na určitém portu na vzdáleném počítači, ale nenaváže s ní spojení. Místo toho lze pomocí tohoto protokolu poslat cílové službě samostatné pakety – odtud je odvozen název tohoto protokolu.

Předpokládejme, že chcete přečíst nějaký malý objem dat z databázového serveru. Potřebujete minimálně tři datagramy k navázání spojení protokolem TCP, další tři k odeslání a potvrzení trochy dat a ještě tři k ukončení spojení. S protokolem UDP dosáhneme prakticky stejného efektu s pouhými dvěma datagramy. UDP je protokol bez navazování spojení, takže po nás nevyžaduje otevírat a zavírat relaci. Prostě zabalíme data do datagramu a pošleme je na server. Server vytvoří odpověď, rovněž ji zabalí do datagramu a pošle nám ji zpět. I když je to jednodušší a efektivnější než protokol TCP pro přenosy malých objemů dat, protokol UDP se neumí vyrovnat se ztrátou datagramu. Je čistě otázkou aplikace, například name serveru, aby se s takovou situací uměl vyrovnat.

Více o portech

Na porty je možné nahlížet jako na přípojky síťových spojení. Pokud chce nějaká aplikace nabízet určitou službu, připojí sama sebe k určitému portu a čeká na klienty (říká se tomu také *odposlouchávání* portu). Klient, který chce tuto službu využívat, si alokuje nějaký port na svém hostiteli a připojí se k příslušnému portu serveru na vzdáleném hostiteli. Jeden a týž port může být otevřen na více počítačích, na jednom počítači však může mít jeden port otevřena jen jedna aplikace.

Důležitou vlastností portů je, že jakmile bylo navázáno spojení mezi klientem a serverem, může se k portu serveru připojit jiná instance serveru a odposlouchávat další klienty. To umožňuje například současné přihlášení ke stejnému hostiteli při využití stejného portu 513. Protokol TCP je schopen tato spojení rozlišit, protože všechny přicházejí z různých portů nebo hostitelů. Pokud se například dvakrát přihlásíte k hostiteli **quark** z počítače **erdos**, potom bude první klient **rlogin** využívat port 1 023 a druhý klient port 1 022. Oba však budou připojeni ke stejnému portu 513 na hostiteli **quark**.

Tento příklad ukazuje využití portů jako přístupových bodů, kdy se klient připojuje ke specifickému portu, aby získal určitou službu. Aby klient znal správné číslo portu, musí mezi administrátory obou systémů existovat určitá dohoda ohledně přiřazování těchto čísel. U služeb, které jsou široce používány, jako je například rlogin, musí být tato čísla spravována centrálně. O to se stará organizace IETF (*Internet Engineering Task Force*), která pravidelně vydává RFC nazvané *Přiřazená čísla* (*Assigned Numbers*, RFC-1700). Kromě jiného popisuje čísla portů přiřazená všeobecně známým službám. K mapování služeb na čísla portů používá Linux soubor nazvaný `/etc/services`.

Je třeba poznamenat, že i když spojení využívající protokoly TCP i UDP spoléhají na porty, nedochází mezi nimi ke konfliktům. To znamená, že se například TCP port 513 liší od UDP portu 513. Ve skutečnosti slouží tyto konkrétní porty jako vstupní body pro dvě různé služby, jmenovitě **rlogin** (TCP) a **rwho** (UDP).

Knihovna soketů

V operačních systémech Unix je software, který obsluhuje veškeré výše popsané úkoly a protokoly, většinou součástí jádra, což platí i pro Linux. Ve světě Unixu je nejběžnějším programovým rozhraním knihovna *Berkeley Socket Library*. Její název je odvozen od oblíbené analogie, která nazírá na porty jako na zásuvky (socket) a připojování k portu chápe jako zapojování. Poskytuje volání `bind`, kterým se specifikuje vzdálený hostitel, přenosový protokol a služba, ke které se program může připojit nebo ji poslouchat (pomocí volání `connect`, `listen` a `accept`). Knihovna soketů je však obecnější v tom, že poskytuje nejenom třídu soketů založených na protokolu TCP/IP (sokety typu *AF_INET*), ale také třídu, která obsluhuje spojení s lokálním počítačem (třída *AF_UNIX*). Některé implementace mohou obsluhovat také jiné třídy, jako například protokol XNS (*Xerox Networking System*) nebo X.25.

V Linuxu je knihovna soketů součástí standardní knihovny jazyka C zvané *libc*. V současné době podporuje sokety typu `AF_INET` a `AF_INET6` pro protokoly TCP/IP, `AF_UNIX` pro lokální komunikaci a dále `AF_IPX` pro protokoly sítě Novell, `AF_X25` pro protokol X.25, `AF_ATMPVC` a `AF_ATMSVC` pro sítě ATM, a `AF_AX25`, `AF_NETROM` a `AF_ROSE` pro paketové rádio. Vyvíjí se i podpora dalších protokolů, která bude dostupná časem.

Sítě v Linuxu

Jakožto výsledek soustředěného úsilí programátorů z celého světa by Linux nemohl existovat bez globální počítačové sítě. Takže nikoho nepřekvapí, že již v raných stadiích vývoje začalo několik lidí pracovat na síťových schopnostech tohoto operačního systému. Práce na začlenění protokolu TCP/IP začala na podzim roku 1992, když Ross Biro a další vytvořili projekt, který získal označení Net-1.

Poté co Ross v květnu 1993 opustil práci na vývoji tohoto protokolu, začal Fred van Kempen pracovat na nové implementaci, přičemž přepracoval hlavní části kódu. Výsledkem byla verze Net-2. První veřejná verze Net-2d byla hotová v létě 1992 (jako součást jádra verze 0.99.10) a od té do-

by byla udržována a rozšiřována různými lidmi, zejména Alanem Coxem[10], který stál u zrodu verze Net-2Debugged. Po důkladném otestování a mnoha vylepšeních kódu přišla po vydání Linuxu verze 1.0 změna názvu na Net-3. Kód Net-3 byl dále vyvíjen pro Linux 1.2 a Linux 2.0. Jádra verze 2.2 a vyšší používají knihovnu Net-4, která v současné době představuje standard.

Síťový kód Net-4 podporuje celou řadu zařízení a různých funkcí. Mezi standardní protokoly knihovny Net-4 patří SLIP a PPP (pro síťový provoz po sériových linkách), PLIP (pro paralelní linky), IPX (pro sítě kompatibilní s Novell NetWare, o této problematice budeme hovořit v kapitole 15), Appletalk (pro sítě počítačů Apple) a AX.25, NetRom a Rose (pro rádiové sítě). Mezi další standardní funkce této knihovny patří IP firewally, IP účtování (budeme o nich hovořit v kapitole 9 a 10) a IP maškaráda (kapitola 11). Podporován je IP tuneling a různé techniky směrování. Je podporována celá řada ethernetových zařízení stejně jako některá FDDI, Token Ring, Frame Relay, ISDN a ATM zařízení.

Kromě toho je podporována celá řada dalších funkcí, které zvyšují univerzálnost Linuxu. Mezi tyto funkce patří implementace souborového systému SMB, který spolupracuje s aplikacemi jako jsou *lanmanager* a Microsoft Windows. Tuto implementaci, nazvanou Samba, napsal Andrew Tridgell. Dále Linux obsahuje implementaci novellového protokolu NCP[11].

Různé směry vývoje

V různých dobách probíhaly práce na vývoji různých síťových funkcí systému Linux.

Fred pokračoval ve vývoji knihovny Net2-Debugged poté, co byla prohlášena za oficiální síťovou implementaci. Vývoj vedl k verzi Net-2e, která měla přepracovanou strukturu síťové vrstvy. Největším přínosem Net-2e bylo rozhraní DDI (*Device Driver Interface*). V současné době už není knihovna Net-2e dále vyvíjena.

Další implementace protokolu TCP/IP pochází od Matthiase Urlichse, který napsal ovladač ISDN pro Linux a FreeBSD. Za tím účelem integroval do jádra Linuxu část síťového kódu BSD. Ani na tomto projektu se už dnes nepracuje.

V implementaci síťových funkcí v jádře Linuxu došlo k mnoha změnám, a protože vývoj stále probíhá, i nadále k nim dochází. Někdy to znamená, že změna se musí projevit i v jiných programech, například v nástrojích pro konfiguraci sítě. I když to dnes nepředstavuje takový problém jako v minulosti, i dnes se může stát, že pokud aktualizujete jádro, budete muset aktualizovat i nástroje pro konfiguraci síťových funkcí. Naštěstí vzhledem k velkému počtu dnes existujících distribucí je to poměrně snadná úloha.

Síťová implementace Net-4 je velmi vyzrálá a používá se na řadě míst po celém světě. Velké úsilí bylo věnováno vylepšení výkonu této implementace, takže dnes může být směle porovnána s nejvýkonnějšími implementacemi dostupnými pro srovnatelné platformy. Linux je velmi oblíben u poskytovatelů internetového přístupu a velmi často se v těchto firmách používá k vytvoření levných a spolehlivých webových serverů, poštovních serverů a serverů zpráv. O vývoj Linuxu je velký zájem, a tak probíhá paralelně se změnami síťových technologií, například současné verze jádra již podporují novou generaci protokolu IP, IPv6, tak, jak je navržen jako standard.

Kde získáte kód

Dnes už zní velmi zvláštně, že v začátcích síťové implementace v Linuxu vyžadovalo standardní jádro k začlenění síťové podpory celou řadu různých patchů. V současné době probíhají práce na vývoji síťových funkcí současně s vývojem jádra. Poslední stabilní jádro lze získat na adrese *ftp.kernel.org* v adresáři `/pub/linux/kernel/v2.x`, kde *x* je sudé číslo. Poslední experimentální verzi

10 Alana můžete kontaktovat na adrese *alan@lxorguk.ukuu.org.uk*.

11 NCP je protokol, pomocí nějž komunikují souborové a tiskové servery v síti Novell.

naleznete na stejném místě, číslo této verze je ale liché. Na celém světě jsou zrcadla tohoto serveru. Jen velmi těžko si dnes někdo umí představit Linux bez integrované síťové podpory.

Údržba vašeho systému

V průběhu celé knihy se budeme zabývat především otázkami instalace a konfigurace. Správa však znamená mnohem víc – po zavedení služby je třeba ji také udržovat v provozu. Většina služeb vyžaduje jen nepatrnou údržbu, ale u některých z nich, jako je pošta a news, vyžaduje zachování aktuálnosti systému provádění rutinních úkolů. Tyto úkoly budeme probírat v dalších kapitolách.

Absolutní minimum údržby spočívá v pravidelné kontrole systémových a aplikačních logovacích souborů, zda neobsahují chybové stavy nebo neobvyklé události. Běžně se to dělá za pomoci dvojice administrativních skriptů, které se pravidelně spouštějí démonem **cron**. Tyto skripty obsahují distribuce zdrojových kódů některých hlavních aplikací, jako je například **inn** nebo C News. Je třeba si je jen upravit tak, aby vyhovovaly vašim potřebám a preferencím.

Výstup každé úlohy démona **cron** by měl být poslán poštou na účet **administrátora**. Mnoho aplikací implicitně posílá chybové zprávy, statistiky využití nebo souhrny logovacích souborů na účet **root**. To má smysl pouze tehdy, pokud se jako uživatel **root** přihlašujete často, ale výhodnější je za pomoci poštovního aliasu, který popisujeme v kapitolách 14 a 15, přesměrovat poštu pro uživatele **root** na váš osobní účet.

Bez ohledu na to, jak pečlivě váš systém nastavíte, v důsledku Murphyho zákonů se časem *zaručeně* vynoří nějaký problém. Proto znamená údržba systému i neustálou připravenost řešit stížnosti. Lidé zpravidla očekávají, že správce systému bude k zastižení přinejmenším prostřednictvím e-mailové adresy **root**, ale pro kontaktování lidí, kteří jsou odpovědní za určitý aspekt údržby, jsou běžně používány také jiné adresy. Například stížnosti na špatně fungující poštu budou adresovány uživateli **postmaster** a problémy se systémem news je možné oznamovat na adresách **newsmaster** nebo **usenet**. Pošta na účet **hostmaster** by měla být přesměrována člověku, který má na starosti základní síťové služby, a pokud provozujete name-server, tak také server služby DNS.

Bezpečnost systému

Dalším velice důležitým aspektem správy systému v síťovém prostředí je ochrana systému a uživatelů před vetřelci. Nedbale spravované systémy jsou vhodným cílem pro škodolibé uživatele, jejichž rozsah útoků je poměrně široký: od uhodnutí hesla až po sledování provozu na Ethernetu a způsobené škody mohou sahat od padělaných poštovních zpráv až po ztrátu dat nebo narušení soukromí vašich uživatelů. Když budeme hovořit o souvislostech, za kterých k nim může dojít, zmíníme se o některých konkrétních problémech a běžných ochranách proti nim.

Tato stať probírá některé příklady a základní postupy zajištění systémové bezpečnosti. Samozřejmě, že zde uvedená témata nemohou důkladně postihnout všechny bezpečnostní problémy, s nimiž se můžete setkat; slouží především k ilustraci problémů, které se mohou vyskytnout. Proto je nezbytně nutné přečíst si nějakou dobrou knihu pojednávající o bezpečnosti, zejména pak v systému propojeném do sítě.

Základem bezpečnosti systému je dobrá systémová administrace. Ta zahrnuje kontrolu vlastnictví a přístupových práv ke všem životně důležitým souborům a adresářům a sledování používání privilegovaných účtů. Program COPS například ověří, zda váš souborový systém a běžné konfigurační soubory neobsahují neobvyklá přístupová práva nebo jiné anomálie. Také je rozumné používat sadu programů pro správu hesel, které prosadí jistá pravidla pro zadávání hesel takovým způ-

sobem, aby bylo těžké je uhodnout. Například sada shadow vyžaduje, aby bylo heslo dlouhé minimálně pět písmen a obsahovalo kromě malých a velkých písmen i jiné znaky.

Při zpřístupňování určité služby přes síť jí vždy přidělte nejnižší práva, což znamená, že jí nedovolíte provádět věci, které pro své fungování nepotřebuje. Programům byste měli například povolit přepnout se na účet **root** nebo některý jiný privilegovaný účet jen tehdy, pokud to je opravdu nezbytně nutné. V případě, že chcete používat některou službu jen omezeným způsobem, neváhejte ji nakonfigurovat s co největším omezením práv tak, jak to konkrétní situace dovolí. Chcete -li například povolit bezdiskovým počítačům zavádění operačního systému z vašeho počítače, musíte zprovoznit službu TFTP (*Trivial File Transfer Protocol*), aby tyto počítače mohly z adresáře `/boot` nahrát základní konfigurační soubory. Pokud byste ale používali službu TFTP bez omezení přístupu, umožnili byste komukoliv na světě nahrát si z vašeho systému libovolný soubor, který je veřejně čitelný. Proč tedy neomezit služby protokolu TFTP jen na adresář `/boot`[12]?

Možná také budete chtít omezit určité služby pouze na uživatele určitých počítačů, řekněme z vaší lokální sítě. V kapitole 12 si představíme nástroj **tcpd**, který to umožňuje pro různé síťové aplikace. O ještě pokročilejších metodách omezení přístupu na určité počítače nebo služby budeme hovořit také v kapitole 9.

Dále je nutné se vyhnout „nebezpečným“ aplikacím. Samozřejmě, že každý software, který používáte, může být svým způsobem nebezpečný, protože může obsahovat chyby, jichž mohou chytří lidé využít k získání přístupu k vašemu systému. K takovýmto věcem dochází, a není proti nim úplná ochrana. Tento problém se týká volně šiřitelného softwaru stejně jako komerčních produktů[13]. Nicméně programy, které vyžadují speciální přístupová práva, jsou logicky nebezpečnější než jiné, protože každá díra v nich může mít drastické důsledky[14]. Pokud nějaký síťový program instalujete jako setuid, dvakrát zkontrolujte dokumentaci, abyste na nic nezapomněli a náhodou tak nevytvořili bezpečnostní trhlinu.

Další oblast, kde byste měli být opatrní, jsou programy, které umožňují přihlášení nebo provádění příkazů jen s omezenou autentikací. Programy **rlogin**, **rsh** a **rexec** jsou všechny velice užitečné, nicméně provádějí jen velmi jednoduchou autentikaci požadavku. Autentikace je založena na tom, že se důvěřuje jménu vzdáleného systému zjištěnému u name serveru (o name serverech budeme hovořit později). Tyto záznamy je však možné podvrhnout. V dnešní době by mělo být standardním postupem všechny **r** služby vypnout a nahradit je nástroji **ssh**. Tyto nástroje obsahují mnohem spolehlivější metody autentikace a nabízejí i další služby, jako je například šifrování a komprimace.

Nikdy nelze vyloučit, že vaše opatření selžou, bez ohledu na to, jak opatrní jste byli. Proto byste měli zajistit včasné odhalení vetřelců. Kontrola systémových logovacích souborů je dobrý začátek, ale vetřelec bude zřejmě natolik chytrý, že po sobě smaže všechny zřetelné stopy. Existují však nástroje, jako například **tripwire**, napsaný Gene Kimem a Gene Spaffordem, které umožňují kontrolovat, zda nedošlo ke změně obsahu nebo přístupových práv životně důležitých systémových souborů. Program **tripwire** vypočítá pro tyto soubory několik silných kontrolních součtů a uloží je v databázi. Při dalším spuštění se kontrolní součty vypočítají znovu a kontroluje se, zda se oproti uloženým hodnotám nezměnily.

---

12 K tomuto tématu se vrátíme v kapitole 12.

13 Existovaly komerční unixové systémy (tedy systémy, za které se platí spousta peněz), které obsahovaly skript setuid-root, jenž umožňoval uživatelům získat práva uživatele **root** pomocí jednoduchého standardního triku.

14 V roce 1998 způsobil zablokování velké části Internetu červ RTM, který mimo jiné využíval chyby v různých programech včetně programu **sendmail**. Trvalo dlouho, než byly všechny tyto chyby odstraněny.

KAPITOLA 2

Problematika sítí TCP/IP

V této kapitole se seznámíme s konfiguračními parametry, které budete při připojování linuxového stroje k síti TCP/IP potřebovat, budeme hovořit o IP adresách, názvech a směrování. Popíšeme si nezbytné pozadí, které je nutné k pochopení toho, co se vlastně při instalaci dělá. Samotné konfigurační postupy a používané nástroje pak popíšeme v dalších kapitolách.

Podrobnější informace o protokolech TCP/IP najdete ve třísvazkové knize *Internetworking with TCP/IP* Douglase R. Comera (Prentice Hall). Konfiguračním záležitostem se věnuje kniha *TCP/IP Network Administration* (O'Reilly).

Síťová rozhraní

Kvůli různorodosti vybavení, které se v síťovém prostředí používá, zavádí implementace TCP/IP abstraktní rozhraní, přes nějž se přistupuje k hardwarovým zařízením. Rozhraní nabízí skupinu operací společných pro všechna možná hardwarová zařízení, které se vesměs týkají odesílání a příjmu paketů.

Pro každé přítomné síťové zařízení musí být v jádře přístupné příslušné rozhraní. Například ethernetové karty se v jádře označují jako `eth0` a `eth1`, PPP zařízení (viz kapitola 8) jako rozhraní `ppp0` a `ppp1`, FDDI zařízení jako `fddi0` a `fddi1`. Tato jména rozhraní se používají při konfiguraci, kdy potřebujete říct, pro které fyzické zařízení má daný konfigurační příkaz platit, a nemají žádný další význam.

Než může být zařízení použito v sítích TCP/IP, musí mít přidělenu IP adresu, která slouží k jeho identifikaci při komunikaci se zbytkem světa. Adresa nemá nic společného s názvy, zmíněnými v předchozím odstavci. Pokud bychom samotné zařízení přirovnali ke dveřím, pak adresa je jednoduše jmenovka, která je na dveřích nalepena.

Lze také nastavit řadu jiných parametrů, například maximální velikost datagramu, se kterým umí dané fyzické zařízení pracovat – tzv. *MTU, Maximum Transfer Unit*. Většina parametrů je naštěstí implicitně nastavena na rozumně použitelné hodnoty.

IP adresy

Již v předchozí kapitole jsme se zmínili o tom, že adresy, kterým rozumí síťový protokol IP, jsou 32bitová čísla. Každému počítači musí být v rámci celého síťového prostředí přiděleno jedinečné

číslo[15]. Pokud provozujete lokální síť, která s ostatními sítěmi nekomunikuje na bázi protokolu TCP/IP, můžete tato čísla přidělit podle vlastního uvážení. Existují rozsahy IP adres, které jsou určeny právě pro přidělování na takovýchto privátních sítích. Tyto adresy jsou uvedeny v tabulce 2.1. Nicméně systémům připojeným k Internetu jsou čísla přidělována centrálně, konkrétně institucí Network Information Center, zkráceně NIC[16].

Třída	Adresy
A	10.0.0.0 až 10.255.255.255
B	172.16.0.0 až 172.31.0.0
C	192.168.0.0 až 192.168.255.0

Tabulka 2.1 – Rozsahy IP adres rezervované pro privátní použití

IP adresy jsou pro přehlednost rozděleny do čtyř 8bitových čísel, kterým říkáme oktety. Například počítač **quark.physics.groucho.edu** má IP adresu **0x954C0C04**, která je zapsána jako **149.76.12.4**. Tomuto formátu se také někdy říká tečková notace.

Dalším důvodem této symboliky je rozdělení IP adres na číslo sítě, které je obsaženo v levých oktetech, a na číslo hostitele, které je zapsáno v pravých oktetech. Když požádáte centrum NIC o přidělení IP adres, nebude vám přidělena adresa pro každého hostitele, kterého hodláte používat. Místo toho obdržíte jediné číslo sítě a pak na základě vlastního uvážení můžete přidělovat hostitelům ve vaší síti všechny IP adresy v rámci získaného rozsahu IP adres.

V závislosti na velikosti sítě je nutné mít i různě velkou část hostitele. Protože mají různí uživatelé různé nároky na velikost sítí, existuje několik tříd sítí, které definují různá rozdělení IP adres:

Třída A Třída A obsahuje sítě od **1.0.0.0** do **127.0.0.0**. Číslo sítě je obsaženo v prvním oktetu. Takto je k dispozici 24bitová část hostitele, která umožňuje až 1,6 milionů hostitelů.

Třída B Třída B obsahuje sítě od **128.0.0.0** do **191.255.0.0**; číslo sítě je obsaženo v prvních dvou oktetech. To umožňuje 16 320 sítí, z nichž každá může mít až 65 024 hostitelů.

Třída C Třída C zahrnuje sítě od **192.0.0.0** do **223.255.255.0**, kde číslo sítě je obsaženo v prvních třech oktetech. To umožňuje vytvořit téměř 2 miliony sítí s až 254 hostiteli.

Třída D, E a F Adresy, které spadají do intervalu od **224.0.0.0** do **254.0.0.0,** jsou buď experimentální, nebo jsou rezervovány pro speciální použití. Nespecifikují žádnou síť. Právě z tohoto rozsahu má přiděleny adresy například služba IP Multicast, která umožňuje posílat data současně na více počítačů.

Vrátíme-li se zpět k příkladu z první kapitoly, zjistíme, že adresa **149.76.12.4**, tedy adresa počítače **quark**, spadá do třídy B a označuje hostitele **12.4** na síti **149.76.0.0**.

Ve výše uvedeném seznamu jste si možná všimli, že v každém oktetu nebyly v příslušné části hostitele povoleny všechny hodnoty. Je to tím, že oktety s čísly **0** a **255** jsou rezervovány pro speciální účely. Adresa, ve které jsou všechny bity hostitelské části nulové, odkazuje na síť a adresa, v níž jsou všechny bity hostitelské části rovny jedné, se nazývá vysílací adresa. Vysílací adresa současně odkazuje na všechny hostitele v konkrétní síti. Tedy adresa **149.76.255.255** není konkrétní adresa nějakého hostitele, ale označuje všechny hostitele v síti **149.76.0.0**.

15 V současné době se na Internetu nejčastěji používá IP protokol verze 4. Velké úsilí bylo věnováno vytvoření náhrady, protokolu IP verze 6. IPv6 používá odlišné adresní schéma a delší adresy. Linux obsahuje implementaci protokolu IPv6, není však dosud ve stadiu, abychom ji zde mohli popsat. Samotná podpora protokolu IPv6 v jádře je v pořádku, nicméně řada aplikací musí být upravena, aby mohla s tímto protokolem pracovat. Zůstaňte s námi.

16 Typicky vám IP adresu přidělí poskytovatel internetového připojení, kterému za konektivitu k Internetu platíte. Můžete nicméně kontaktovat přímo NIC a vyžádat si své vlastní adresy prostřednictvím e-mailu na adresu *hostmaster@internic.net* nebo pomocí formuláře na adrese *http://www.internic.net*.

Dále existují ještě různé rezervované síťové adresy. Dvě z nich jsou **0.0.0.0** a **127.0.0.0**. První z nich se nazývá *implicitní směr*, druhá se nazývá *lokální adresa*. Implicitní směr souvisí se způsobem směrování IP datagramů, které budeme probírat dále.

Síť **127.0.0.0** je rezervována pro lokální přenosy na vašem počítači. Adresa **127.0.0.1** je typicky přiřazena speciálnímu rozhraní na vašem hostiteli, které se nazývá *loopback interface* a chová se jako uzavřený obvod. Každý IP paket předaný protokolem TCP nebo UDP je vrácen zpět, jakoby právě přišel z nějaké sítě. To umožňuje vyvíjet a testovat síťový software, aniž byste používali skutečnou síť. Další užitečnou aplikací je případ, kdy chcete použít síťový software na samostatném počítači. Není to zase tak neobvyklé, jak by se mohlo zdát; například mnoho systémů UUCP nemá vůbec žádné IP propojení, ale přesto chtějí provozovat systém zpráv INN. Aby systém INN pod Linuxem správně fungoval, vyžaduje použití loopback rozhraní.

Část adres z každé třídy je vyhrazena jako „rezervované“ nebo „privátní“ adresy. Tyto adresy jsou rezervovány pro použití na privátních sítích a v Internetu se nesměrují. Typicky se používají v organizacích, které si budují vlastní síť a jsou užitečné i pro malé domácí sítě. Rezervované adresy jsou uvedeny v tabulce 2.1.

Rozlišování adres

Když teď víte, jak se vytváří IP adresy, bude vás asi zajímat, jak se s jejich pomocí v ethernetové nebo tokenringové síti adresují různí hostitelé. Tyto protokoly koneckonců identifikují počítače vlastními adresami, které nemají nic společného s IP adresami, že? Správně.

Proto potřebujeme mechanismus pro mapování IP adres na adresy fyzické sítě. Tento mechanismus se nazývá *Address Resolution Protocol*, zkráceně ARP, a není omezen pouze na Ethernet nebo Token Ring, ale používá se i u jiných typů sítí, například u rádiových sítí s protokolem AX.25. Myšlenka protokolu ARP je stejná jako když se někdo pokouší najít pana X mezi 150 různými lidmi: osoba, která jej hledá, prostě zakřičí dost nahlas, aby ji všichni slyšeli a spoléhá na to, že pan X se ozve (pokud je přítomen). Jakmile se ozve, víme, kdo to je.

Když chce ARP najít ethernetovou adresu odpovídající příslušné IP adrese, použije ethernetový mechanismus vysílání, při kterém je datagram současně adresován na všechny stanice v síti. Vyslaný datagram, odeslaný protokolem ARP, obsahuje dotaz na hledanou IP adresu. Každý hostitel, který tento datagram přijme, ji porovná se svou vlastní IP adresou a pokud souhlasí, vrátí dotazujícímu se hostiteli odpověď pomocí protokolu ARP. Nyní může dotazující se hostitel získat z odpovědi ethernetovou adresu počítače, který mu odpověděl.

Možná se ptáte, jak může nějaký počítač najít jiný jen podle jeho internetové adresy, když může jít o počítač někde na druhém konci světa. Odpověď na tuto otázku souvisí se *směrováním*, tedy s nalezením fyzického umístění počítače v síti. O tomto problému budeme hovořit v další části.

Podívejme se nyní na protokol ARP. Když hostitel získá ethernetovou adresu, uloží ji do vyrovnávací paměti protokolu ARP, takže když bude dotyčnému hostiteli posílat znovu nějaký datagram, nemusí se na ni znovu dotazovat. Tuto informaci by však nebylo rozumné uchovávat navždycky; ethernetová karta vzdáleného hostitele může být například z důvodu technických problémů vyměněna, čímž by byla data protokolu ARP neplatná[17]. Proto jsou po určité době záznamy z vyrovnávací paměti protokolu ARP vymazány a vynutí se tak nové dotazování.

Někdy je také nutné nalézt IP adresu, která odpovídá určité ethernetové adrese. K tomu dochází například v situaci, kdy chce bezdiskový počítač zavést operační systém ze síťového serveru, což je v lokálních sítích docela běžná situace. Bezdiskový klient však o sobě nemůže poskytnout vů-

17 Pozn. překladatele: Ještě pravděpodobnější situace je ta, že typicky počítačům v lokální síti jsou IP adresy přidělovány dynamicky (například protokolem DHCP), takže fyzicky jeden a týž počítač se stejnou síťovou kartou může mít jednu IP adresu dnes a jinou zítra.

bec žádné informace – kromě své ethernetové adresy! Jediné, co může udělat, je poslat bootovacímu serveru zprávu se žádostí o sdělení své IP adresy. Pro tento účel existuje další protokol, který se jmenuje *Reverse Address Resolution Protocol*, zkráceně RARP. Společně s protokolem BOOTP slouží k zajištění procedury zavedení operačního systému bezdiskových klientů v síti.

Směrování protokolu IP

Nyní se budeme věnovat otázce jak nalézt počítač, jemuž mají být data poslána, pouze podle jeho IP adresy. Různé části adresy jsou zpracovávány různě a je tedy váš úkol nastavit správně příslušné soubory, které udávají, jak mají být jednotlivé části adresy chápány.

IP sítě

Když někomu píšete dopis, uvedete obvykle na obálce úplnou adresu, která obsahuje stát, město, kraj, PSČ a podobně. Poté co ho vhodíte do poštovní schránky, pošta jej doručí adresátovi: konkrétně ho pošle do uvedeného státu, zde ho místní pošta odešle do příslušného kraje a tak dále. Výhoda tohoto hierarchického schématu je poměrně zřejmá: ať už pošlete dopis kamkoliv, bude vždy místní poštmistr zhruba vědět směr, kterým má dopis poslat dál a přitom se nemusí starat o to, kudy bude dopis putovat z úřadu, na který jej doručí on.

Sítě IP jsou strukturovány obdobným způsobem. Celý Internet se skládá z mnoha sítí, kterým říkáme *autonomní systémy*. Každý takovýto systém provádí veškeré směrování v rámci svých členských systémů, takže se úkol doručení datagramu redukuje na nalezení cesty k síti cílového systému. Znamená to, že jakmile datagram zachytí *libovolný* hostitel v cílové síti, bude další zpracování provedeno výhradně touto sítí.

Podsítě

Tato struktura vyplývá z rozdělení IP adres na část hostitelskou a část síťovou tak, jak bylo popsáno dříve. Cílová síť je implicitně odvozena ze síťové části IP adresy. Hostitelé se shodnou síťovou částí adresy by se tak měli nacházet ve stejné síti[18].

Obdobné schéma má smysl i *uvnitř* sítě, protože vlastní síť se může skládat ze stovek menších sítí, kde jsou nejmenšími jednotkami fyzické sítě, například Ethernety. Protokol IP proto umožňuje rozdělit síť IP na několik *podsítí*.

Podsíť na sebe přebírá odpovědnost za doručení datagramů pro daný rozsah IP adres. K její identifikaci slouží síťová část IP adresy, jako tomu bylo u tříd A, B nebo C. Nyní je však síťová část rozšířena tak, aby obsahovala i některé bity z hostitelské části. Počet bitů, které jsou interpretovány jako číslo podsítě, udává takzvaná *maska podsítě* nebo *síťová maska*. Jedná se také o 32bitové číslo, které určuje bitovou masku síťové části IP adresy[19].

Příkladem takové sítě je školní síť Groucho Marx University. Má síťovou adresu třídy B 149.76.0.0, a proto je její síťová maska 255.255.0.0.

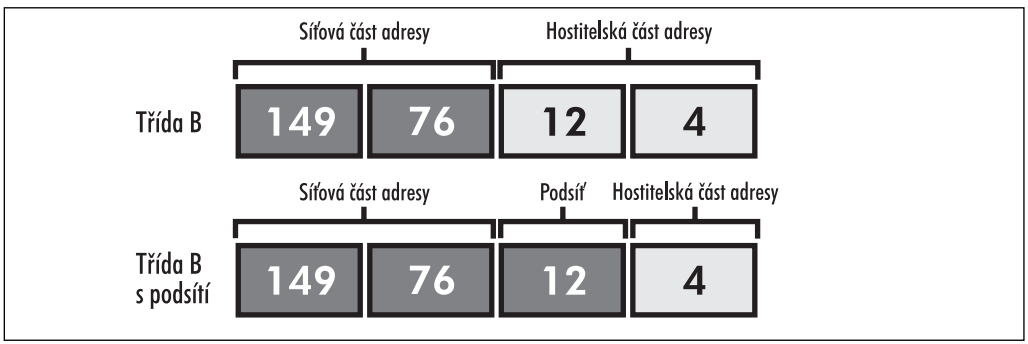
---

[18] Autonomní systémy jsou poněkud obecnější. Mohou v sobě zahrnovat více jednotlivých IP sítí.

[19] Pozn. překladatele: Síťová maska (představíme-li si ji ve dvojkové soustavě) začíná zleva jedničkami přes všechny bity, které označují síťovou část adresy a pokračuje až do konce nulami pro hostitelskou část adresy. Proto máme pro adresu třídy B (první dva bajty jsou síť a druhé dva bajty hostitel) masku 255.255.0.0 (první dva bajty jedniček a druhé dva bajty nul). Rozdělení mezi síťovou a hostitelskou částí adresy je možné nejen na hranici oktetů. Můžeme tak adresu třídy C (tři bajty síť, jeden bajt hostitel) rozdělit maskou například 255.255.255.224 na osm podsítí (první tři bity posledního oktetu jsou adresa podsítě), kde má každá k dispozici 32 hostitelských adres (méně významných pět bitů posledního oktetu). Ve shodě s tím co už bylo řečeno je nicméně z těchto 32 adres fyzicky využitelných jen 30, protože adresa s nulovými posledními pěti bity bude adresou celé konkrétní podsítě a adresa s posledními pěti bity jedničkovými bude vysílací adresou dané podsítě.

Vnitřně je školní síť GMU složena z několika menších sítí, které tvoří například lokální sítě různých kateder. Proto přidělený rozsah IP adres je rozdělen na 254 podsítí, od adresy 149.76.1.0 po adresu 149.76.254.0. Například katedra teoretické fyziky má přidělenu adresu sítě 149.76.12.0. Páteř školní sítě je samostatná síť s adresou 149.76.1.0. Tyto podsítě sdílejí stejnou síťovou adresu v rámci třídy B a třetí oktet slouží k jejich rozlišení. Používají tedy masku podsítě 255.255.255.0.

Obrázek 2.1 ukazuje rozdílnou interpretaci adresy 149.76.12.4, tedy adresy počítače **quark**, je-li chápána jako běžná adresa třídy B, nebo je-li chápána jako adresa v rámci podsítí.



Obrázek 2.1 – Rozdělení adresy třídy B na podsítě

Je vhodné poznamenat, že vytváření podsítí slouží pouze k vnitřnímu dělení sítě. Podsítě jsou vytvářeny vlastníkem sítě (nebo jejím správcem). Podsítě jsou často vytvářeny kvůli existující struktuře, ať už fyzické (mezi dvěma Ethernety), administrativní (mezi dvěma katedrami) nebo geografické (mezi různými lokalitami), a pravomocemi nad jednotlivými podsítěmi je pověřena nějaká kontaktní osoba. Nicméně tato struktura však odráží pouze vnitřní chování sítě a pro okolní svět je neviditelná.

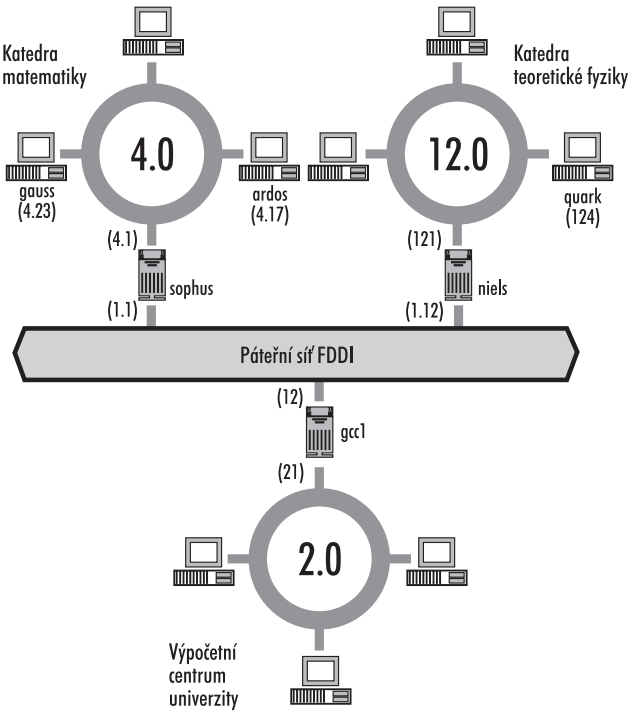
Brány

Vytváření podsítí nemá jen organizační výhody, je často přirozeným důsledkem hardwarové struktury. „Výhled“ hostitele dané fyzické sítě, jako například Ethernetu, je velmi omezený: jedinými hostiteli, se kterými lze komunikovat přímo, jsou hostitelé v dané síti. Ke všem ostatním hostitelům může být přistupováno jen s pomocí specializovaných počítačů, takzvaných bran. Brána je hostitel, který je současně spojen se dvěma nebo více fyzickými sítěmi a jenž je nastaven pro přenášení paketů mezi nimi.

Obrázek 2.2 znázorňuje část síťové topologie sítě univerzity GMU. Počítače připojené ke dvěma různým podsítím jsou znázorněny se dvěma IP adresami.

Aby protokol IP poznal, že se hostitel nachází ve fyzicky lokální síti, musí být různým fyzickým sítím přiřazeny různé IP sítě. Například síťová adresa 149.76.4.0 je vyhrazena pro hostitele lokální sítě katedry matematiky. Když se posílá datagram na počítač **quark**, síťový software na serveru **erdos** okamžitě z IP adresy 149.76.12.4 pozná, že cílový hostitel je připojen k jiné fyzické síti, a proto je dosažitelný pouze pomocí brány (implicitní bránou je **sophus**).

Vlastní brána **sophus** je propojena se dvěma rozdílnými podsítěmi: s katedrou matematiky a s páteřní sítí univerzity. Ke každé z nich přistupuje za pomoci různých rozhraní eth0, respektive fddi0. Jakou adresu mu ale přidělíme? Máme mu dát adresu z podsítě 149.76.1.0 nebo 149.76.4.0?



Obrázek 2.2 – Část síťové topologie univerzity GMU

Odpověď zní: obě. Brána **sophus** musí mít přiřazenu adresu 149.76.1.1 pro použití na podsíti 149.76.1.0 a rovněž adresu 149.76.4.1 pro použití na podsíti 149.76.4.0[20]. Brána musí mít přidělenu jednu IP adresu pro každou síť, ke které je připojena. Tyto adresy – společně s odpovídající síťovou maskou – jsou svázány s rozhraním, přes které se s danou sítí komunikuje. Proto bude mít brána **sophus** přiřazena rozhraní a adresy tak, jak to ukazuje následující tabulka.

Rozhraní	Adresa	Síťová maska
`eth0`	149.76.4.1	255.255.255.0
`fddi0`	149.76.1.1	255.255.255.0
`lo`	127.0.0.1	255.0.0.0

Poslední řádek popisuje lokální rozhraní lo, o kterém jsme se již zmínili.

Jemnou odlišnost mezi přidělením adresy hostiteli nebo jeho rozhraní je možné ignorovat. Na hostitele, kteří se nachází jen v jedné síti, například hostitel **erdos**, se budete obecně odkazovat pomocí jeho IP adresy, i když třeba přísně vzato tato adresa odpovídá ethernetovému rozhraní, jemuž je daná IP adresa přidělena. Tento rozdíl je ale důležitý v případě, kdy se odkazujete na bránu.

20 Pozn. překladatele: První (a analogicky i druhá) adresa samozřejmě nemusí být 146.76.1.*1*, ale může to být 146.76.1.*cokoliv*. Bývá nicméně zvykem, že se branám přiřazují adresy „z některého kraje“ rozsahu adres dané podsítě, tedy buď od *x.x.x.1* nahoru nebo od *x.x.x.254* dolů.

Směrovací tabulka

Nyní se zaměříme na způsob, jakým protokol IP vybírá bránu, kterou použije k doručení datagramu do vzdálené sítě.

Ukázali jsme si, že u datagramu určeného serveru **quark** zkontroluje odesílající hostitel **erdos** cílovou adresu, načež zjistí, že se nenachází na stejné lokální síti. Proto jej pošle na implicitní bránu **sophus**, která je teď postavena před stejný úkol. Brána **sophus** pozná, že hostitel **quark** není na žádné ze sítí, s nimiž je přímo spojena, takže musí najít další bránu, na kterou by datagram poslala. Správnou volbou bude brána **niels**, což je brána katedry fyziky. Nicméně aby mohla brána **sophus** správně asociovat cílovou síť s odpovídající branou, potřebuje k tomu nějaké dodatečné informace[21].

Protokol IP pro tento účel používá tabulku, která asociuje brány se sítěmi, jež lze prostřednictvím dané brány dosáhnout. Navíc musí tabulka obsahovat záznam „pro všechno ostatní“ (takzvanou *implicitní trasu*) – tou bude brána přiřazená síti 0.0.0.0. Této adrese vyhovují všechny neznámé sítě a tak jsou pakety pro síť, u níž není trasa explicitně definována, posílány implicitní trasou. Pro bránu **sophus** by mohla směrovací tabulka vypadat takto[22]:

Síť	Síťová maska	Brána	Rozhraní	Poznámka
149.76.1.0	255.255.255.0	-	`fddi0`	páteřní síť, jsme připojeni přímo
149.76.2.0	255.255.255.0	149.76.1.2	`fddi0`	síť počítačového centra a její brána
149.76.3.0	255.255.255.0	149.76.1.3	`fddi0`	síť nějaké katedry (třeba Automatizace)
149.76.4.0	255.255.255.0	-	`eth0`	síť naší katedry, k té jsme připojeni přímo přes Ethernet
149.76.5.0	255.255.255.0	149.76.1.5	`fddi0`	síť další cizí katedry (třeba Elektrických pohonů)
...	...	...	. . .	tady budou odkazy na sítě a brány všech kateder univerzity
149.76.1.0	0.0.0.0	149.76.1.2	`fddi0`	přes počítačové centrum máme přístup na Internet, jejich brána proto slouží i jako implicitní

Při uvádění trasy pro síť, na niž je brána **sophus** připojena přímo, nepotřebujete žádnou další bránu, proto je na těchto místech místo adresy brány uvedena pomlčka.

Proces zjištění, které trase odpovídá konkrétní cílová adresa, je čistě matematická operace. Celý proces je velmi jednoduchý, jeho pochopení však vyžaduje znalost aritmetiky a logiky ve dvojkové soustavě. Trasa danému cíli odpovídá tehdy a jen tehdy, jestliže adresa sítě binárně ANDována se síťovou maskou je rovna cílové adrese binárně ANDované se síťovou maskou.

Překlad předchozí věty: Trasa odpovídá tehdy, jestliže všechny bity adresy sítě definované síťovou maskou (začínáme zleva a zajímají nás ty bity, kde je v síťové masce jednička) jsou stejné jako bity na shodných pozicích v adrese cílového počítače.

Když protokol IP hledá nejlepší trasu k danému cíli, může ve směrovací tabulce najít více vyhovujících položek. Přinejmenším víme, že implicitní trasa vyhovuje pro jakýkoliv cíl, nicméně pro nám známé cíle vyhovují i jiné trasy. Jak protokol pozná, kterou trasu použít? V této situaci hraje významnou roli síťová maska. I když danému cíli vyhovuje více tras, jedna má síťovou masku delší a druhá kratší (rozuměj: jedna má v síťové masce více jedniček, druhá méně). Čím delší je maska, tím přesněji vyhovuje cílové adrese, a proto se při směrování datagramů vždy volí trasa s nejdelší síťovou maskou. Implicitní trasa má masku se všemi bity nulovými, zatímco masky lokálních podsítí mají 24 jedničkových bitů. Proto budou datagramy pro lokální podsítě předávány přísluš-

21 Pozn. překladatele: I když to na obrázku 2.2 nebylo, nezapomínejme, že na páteřní síti univerzity nejsou jen brány **sophus** a **niels**, ale i padesát dalších bran Katedry elektrických strojů a přístrojů, Katedry automatizace a dokonce i Katedry jazyků, Katedry společenských věd a všeho dalšího včetně rektorátu. A chudák **sophus** se přece nebude ptát každé z nich, zda ve své „diecézi“ nemají požadovaný cílový stroj.

22 Pozn. překladatele: Z tabulky je vidět, jak důležité je přidělovat branám adresy *inteligentně* a hlavně *konzistentně*. Na síti s padesáti podsítěmi a odpovídajícím počtem bran byste se totiž jinak z vytváření směrovacích tabulek zbláznili.

ným branám, protože jejich síťová adresa je delší. Jediné datagramy odesílané implicitní branou budou ty, které nevyhovují žádné jiné položce směrovací tabulky.

Směrovací tabulky mohou být vytvářeny různými způsoby. U malých sítí LAN je většinou nejefektivnější vytvořit je ručně a při procesu zavádění systému je předat protokolu IP pomocí příkazu **route** (viz kapitola 5). V rozsáhlejších sítích jsou vytvářeny a modifikovány za běhu systému pomocí *směrovacích démonů*; tyto démony běží na centrálních hostitelích sítě a prostřednictvím směrovacích protokolů si vyměňují informace pro výpočet optimálních tras mezi členskými sítěmi.

V závislosti na velikosti sítě mohou být použity různé směrovací protokoly. Ke směrování uvnitř autonomních systémů (jako je školní síť Groucho Marx) budou použity *interní směrovací protokoly*. Nejvýznamnějším z nich je protokol RIP, *Routing Information Protocol*, který je implementován v BSD démonu **routed**. Ke směrování mezi autonomními systémy slouží *externí směrovací protokoly*, jako je například EGP (*External Gateway Protocol*) nebo BGP (*Border Gateway Protocol*). Tyto protokoly (stejně jako RIP) jsou implementovány v démonu **gated** z Cornellské univerzity.

Hodnoty metrik

Dynamické směrování založené na protokolu RIP vybírá nejlepší směrování k cílovému hostiteli nebo síti na základě počtu skoků, to znamená počtu bran, kterými musí datagram projít, než dosáhne cíle. Čím méně má trasa skoků, tím lépe ji protokol RIP ohodnotí. Dlouhé trasy s 16 nebo více skoky jsou považovány za nepoužitelné a jsou zrušeny.

Chcete-li použít protokol RIP na vnitřní správu směrovacích informací ve vaší místní síti, musíte mít na všech hostitelích spuštěn program **gated**. Při procesu zavádění systému zkontroluje programem **gated** všechna aktivní síťová rozhraní. Pokud existuje více než jedno aktivní rozhraní (nepočítaje lokální rozhraní), bude předpokládat, že „jeho“ počítač předává pakety mezi několika sítěmi a bude aktivně vyměňovat a vysílat směrovací informace. V opačném případě bude jen pasivně přijímat všechny aktualizace protokolem RIP a bude aktualizovat lokální směrovací tabulku.

Při vysílání informace z lokální směrovací tabulky vypočte program **gated** délku trasy z takzvané *metriky*, která je ve směrovací tabulce součástí dat jednotlivých položek. Metrika je hodnota, kterou při konfiguraci směrování nastavuje správce systému a měla by odrážet skutečnou „cenu“ použití dané trasy[23]. Proto by měla být metrika trasy na podsíť, s níž je brána přímo propojena, vždy rovna nule, zatímco trasa procházející dvěma bránami by měla mít metriku rovnu dvěma. S metrikami se nicméně nemusíte zatěžovat, pokud nebudete používat protokol RIP nebo program **gated**.

Protokol ICMP

Součástí protokolu IP je ještě další protokol, o kterém jsme doposud nemluvili. Jde o protokol ICMP (*Internet Control Message Protocol*), jehož prostřednictvím odesílá síťový kód jádra zprávy o chybách jiným počítačům. Předpokládejme například, že jste znovu přihlášeni na hostiteli **erdos** a chcete se za pomoci telnetu připojit na port 12 345 serveru **quark**, nicméně na tomto portu tohoto počítače žádný proces neposlouchá. Jakmile dorazí na tento port serveru **quark** nějaký paket, síťová vrstva to pozná a okamžitě vrátí prostřednictvím protokolu ICMP hostiteli **erdos** zprávu, v níž bude uvedeno chybové hlášení Port Unreachable (port je nedosažitelný).

Protokol ICMP obsahuje poměrně velký počet zpráv, z nichž se řada týká různých chybových stavů. Velmi zajímavou zprávou je zpráva Redirect message (přesměrování zprávy). Generuje ji smě-

23 Cenu trasy můžeme v nejjednodušším případě odvodit z počtů skoků v jednotlivých trasách. Ve složitějších případech (alternativní propojení více linkami s různou rychlostí, použití linek, kde platíme za objem přenesených dat a podobně) představuje optimální nastavení metrik mimořádné složité umění.

rovací modul v případě, že zjistí, že ho nějaký systém používá jako bránu v případě, že k požadovanému cíli existuje i kratší cesta. Například po restartu systému může být směrovací tabulka brány **sophus** neúplná a obsahuje pouze trasy na síť katedry matematiky, na páteř FDDI a implicitní směr na centrální bránu počítačového centra (**gcc1**). Proto bude každý paket pro server **quark** poslán na bránu **gcc1** místo aby byl poslán přímo na bránu **niels**, což je brána katedry fyziky. Při přijetí takového datagramu si brána **gcc1** všimne, že jde o špatnou volbu trasy, pošle paket na bránu **niels** a zároveň vrátí bráně **sophus** pomocí protokolu ICMP zprávu o přesměrování, která bude obsahovat výhodnější cestu.

Teď to vypadá, že jde o velmi chytrý způsob, jak se vyhnout manuálnímu nastavování všech tras s výjimkou těch nejzákladnějších. Ale ne vždy je dobré spoléhat na dynamická směrovací schémata, ať už jde o protokol RIP nebo ICMP přesměrování. Zpráva o přesměrování u protokolu ICMP nebo protokolu RIP umožňují jen malou, případně nulovou kontrolu toho, zda je směrovací informace skutečně autentická. Takto mohou zlomyslní uživatelé přerušit dopravu v celé síti nebo případně provést i něco horšího. Proto všechny verze linuxového síťového kódu chápou zprávy o přesměrování pouze jako zprávy týkající se konkrétního počítače, nikoliv celé sítě. Tím se minimalizuje dopad eventuálního útoku pouze na směrování k jedinému počítači a nikoliv na směrování pro celou síť. Na opačné straně to ale vede k nadbytečnému provozu, pokud k přesměrování dochází oprávněně, protože pro každý počítač na cílové síti bude generována samostatná ICMP zpráva o přesměrování. Obecně je dnes považováno za nevychovanost spoléhat při směrování na ICMP a jeho zprávy o přesměrování trasy.

Rozlišení jmen hostitele

Jak už jsme řekli, je adresování v sítích na bázi protokolu TCP/IP (přinejmenším ve verzi IPv4) prováděno prostřednictvím 32bitových čísel. Pokud byste si však chtěli zapamatovat více takovýchto adres, strávili byste nad nimi poměrně hodně času. Proto se hostitelé označují „běžnými" jmény, jako je třeba hostitel **gauss** nebo hostitel **strange**. Povinností aplikace pak je, aby našla IP adresu odpovídající danému jménu. Tento proces se označuje jako *rozlišení jména hostitele*.

Když chce aplikace IP adresu odpovídající jménu nějakého hostitele, použije knihovní funkce `gethostbyname(3)` a `gethostbyaddr(3)`. Typicky byly tyto a další příbuzné funkce poskytovány knihovnou *resolveu*, v Linuxu jsou součástí standardní knihovny *libc*. Hovorově se všechny funkce této kategorie označují jako „resolver" – „ten, který rozlišuje". Informace o konfiguraci resolveru najdete v kapitole 6.

U malých sítí jako je Ethernet nebo skupiny sítí Ethernet není příliš obtížné udržovat tabulky s názvy hostitelů a jejich adresami. Tyto informace jsou obvykle uchovávány v souboru pojmenovaném `/etc/hosts`. Při přidávání nebo odebírání hostitele nebo při změně adresy stačí na všech hostitelích v síti aktualizovat soubor `hosts`. Je jasné, že u sítí s více než jen několika počítači by byl takovýto proces dost náročný.

V Internetu byly adresy všech počítačů původně rovněž uchovávány v jediné databázi nazvané `HOSTS.TXT`. Tento soubor byl spravován institucí Network Information Center (NIC) a všechny zúčastněné systémy si ho musely stáhnout a nainstalovat. Jak se síť rozrůstala, objevilo se v souvislosti s tímto schématem několik problémů. Kromě administrativní režie spojené s pravidelnou instalací souboru `HOSTS.TXT` se neúnosně zvětšovalo i zatížení serverů, které ho distribuovaly. Mnohem horší ale bylo, že všechny názvy musely být registrovány v centru NIC, čímž se zajistilo, aby se některý název neobjevil dvakrát.

Proto bylo v roce 1994 zavedeno nové schéma překladu jmen na adresy: *Domain Name System*. DNS navrhl Paul Mockapetris a tento mechanismus řeší oba výše zmíněné problémy. Podrobnosti o DNS uvádíme v kapitole 6.

KAPITOLA 3

Konfigurace síťového hardwaru

Až dosud jsme se bavili o síťových rozhraních a všeobecně o problematice protokolu TCP/IP, ale zatím jsme si neřekli, co se doopravdy děje, když „síťový kód“ jádra přistupuje k nějakému hardwarovému zařízení. Abychom si to mohli přesně popsat, musíme si nejprve něco říct o koncepci rozhraní a ovladačů.

Nejprve máte samozřejmě vlastní hardware, například ethernetovou, FDDI nebo tokenringovou kartu: to je laminátová deska přecpaná spoustou mrňavých čipů, která je zasunuta v nějakém slotu vašeho počítače. Takové věci obecně říkáme fyzické zařízení.

Abyste mohli síťovou kartu používat, musí být v jádru Linuxu k dispozici speciální funkce, které rozumí způsobu, jakým je k tomuto zařízení přistupováno. Software, který tyto funkce implementuje, označujeme jako *ovladač zařízení*. Linux obsahuje ovladače zařízení pro řadu různých síťových karet: ISA, PCI, MCA, EISA, paralelní port, PCMCIA a nejnověji i USB.

Ale co myslíme tím, když říkáme, že ovladač „obsluhuje“ zařízení? Vraťme se zpět k ethernetové kartě. Ovladač musí být nějakým způsobem schopen komunikovat s logikou hardwarové karty: musí jí posílat příkazy a data, a ta by mu měla na oplátku doručit všechna přijatá data.

U počítačů PC se tato komunikace odehrává v oblasti vstupně-výstupní paměti, která je namapována na registry karty, a/nebo prostřednictvím sdílené paměti nebo přímého přístupu do paměti. Všechny příkazy a data vyslané jádrem do karty musí těmito registry projít. Vstupně-výstupní a paměťové adresy jsou obecně určeny zadáním počáteční neboli *bázové adresy*. Typické bázové adresy pro ethernetové karty na sběrnici ISA jsou `0x280` nebo `0x300`. Kartám na sběrnici PCI se obvykle jejich vstupně-výstupní adresy přidělují automaticky.

Obvykle není třeba si dělat příliš starosti s nastavením hardwarového zařízení, například jeho bázové adresy, protože jádro se při zavádění systému pokusí detekovat pozici karty. Tento proces se nazývá *automatické detekce* a znamená to, že jádro přečte několik paměťových nebo vstupně-výstupních adres a porovná přečtená data s tím, co by získalo, kdyby na nich byla nainstalována určitá ethernetová karta. Mohou však existovat ethernetové karty, které se nepodaří detekovat automaticky; bývá to případ levných ethernetových karet, jež nejsou dokonalými klony standardních karet jiných výrobců. Jádro systému se navíc bude při zavádění pokoušet detekovat pouze jediné ethernetové zařízení. Pokud používáte více než jednu kartu, musíte jádru o ostatních kartách explicitně říci.

Dalším takovým parametrem, který možná budete muset jádru sdělit, je nastavení kanálu přerušení. Hardwarové komponenty obvykle vyvolají přerušení jádra v okamžiku, kdy potřebují být obsluhovány – například pokud dorazí nějaká data nebo pokud dojde k nějaké neobvyklé situaci.

U počítačů PC se sběrnicí ISA se mohou přerušení vyskytnout na jednom z 15 kanálů přerušení, které jsou očíslovány 0, 1 a 3 až 15. Číslo přerušení, které je hardwarové komponentě přiděleno, se nazývá *interrupt request number*, zkráceně IRQ[24].

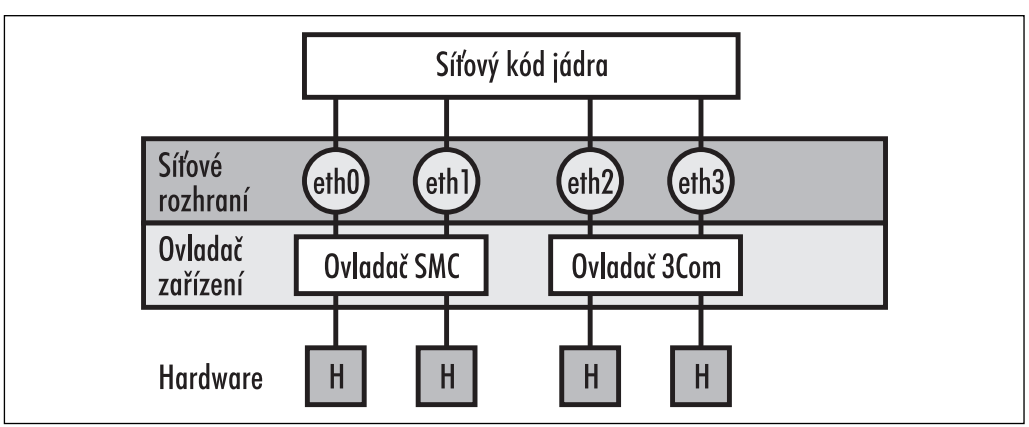
V kapitole 2 jsme si řekli, že jádro přistupuje k zařízení pomocí speciální softwarové konstrukce, takzvaného *rozhraní*. Ta poskytují abstraktní množinu funkcí, totožnou pro různé typy hardwarových zařízení, například funkce pro posílání nebo příjem datagramu.

Rozhraní jsou identifikována prostřednictvím svých názvů. Ve většině operačních systémů unixového typu jsou síťová rozhraní implementována jako speciální soubory zařízení v adresáři `/dev/`. Zadáte-li příkaz `ls -las /dev/`, uvidíte, jak soubory zařízení vypadají. Ve sloupci přístupových práv (tedy ve druhém sloupci) si můžete všimnout, že údaj začíná písmenem a nikoliv pomlčkou, jako u normálních souborů. Toto písmeno indikuje typ zařízení. Nejběžnější zařízení jsou typu `b`, který znamená, že jde o *blokové zařízení*, které najednou obsluhuje celé bloky zapisovaných a čtených dat, a zařízení typu `c`, což jsou *znaková zařízení*, jež s daty pracují znak po znaku. Tam, kde ve výpisu příkazu `ls` normálně vidíte délku souboru, jsou v případě zařízení uvedena dvě čísla: hlavní a vedlejší číslo zařízení. Tato čísla označují skutečné zařízení, s nímž je soubor ovladače spjat.

Každý ovladač zařízení si v jádře registruje jednoznačné hlavní číslo zařízení. Každá instance konkrétního zařízení pak má v rámci tohoto hlavního čísla své jedinečné vedlejší číslo. Například pro rozhraní `tty` jsou soubory `/dev/tty*` znaková zařízení (to poznáme podle písmene `c`) a všechny mají hlavní číslo `4`, ale `/dev/tty1` má vedlejší číslo `1`, `/dev/tty2` má vedlejší číslo `2` a tak dále. Soubory zařízení jsou velmi užitečné pro řadu typů zařízení, mohou však situaci komplikovat, pokud se snažíme najít a otevřít nějaké nepoužívané zařízení.

Názvy rozhraní jsou interně definovány v jádru, a nejedná se o soubory zařízení v adresáři `/dev`. Některá typická jména zařízení jsou uvedena dále v části Prohlídka síťových zařízení v Linuxu. Přiřazení rozhraní zařízením obvykle závisí na pořadí, v němž jsou zařízení konfigurována. Například první nainstalovaná ethernetová karta obdrží název eth0, další bude eth1 a tak dále. Jinak jsou obsluhována rozhraní SLIP, která se přiřazují dynamicky. Kdykoliv se naváže spojení protokolem SLIP, bude sériovému portu přiřazeno rozhraní.

Schéma uvedené na obrázku 3.1 se pokouší ukázat vztahy mezi hardwarem, ovladači zařízení a rozhraními.



Obrázek 3.1 – Vztah mezi ovladači, rozhraními a hardwarem

24 Přerušení číslo 2 a 9 jsou totožná, protože architektura IBM PC používá dva řadiče přerušení s osmi kanály, zapojené v kaskádě. Sekundární řadič je připojen na kanál IRQ 2 primárního řadiče.

Při zavádění systému zobrazí jádro detekovaná zařízení a rozhraní, která se mu podařilo nainstalovat. Následuje část výpisu typické obrazovky při zavádění systému:

```
.
.
This processor honors the WP bit even when in supervisor mode.\
    Good.
Swansea University Computer Society NET3.035 for Linux 2.0
NET3: Unix domain sockets 0.13 for Linux NET3.035.
Swansea University Computer Society TCP/IP for NET3.034
IP Protocols: IGMP,ICMP, UDP, TCP
Swansea University Computer Society IPX 0.34 for NET3.035
IPX Portions Copyright (c) 1995 Caldera, Inc.
Serial driver version 4.13 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16550A
tty01 at 0x02f8 (irq = 3) is a 16550A
CSLIP: code copyright 1989 Regents of the University of California
PPP: Version 2.2.0 (dynamic channel allocation)
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line discipline registered.
eth0: 3c509 at 0x300 tag 1, 10baseT port, address 00 a0 24 0e e4 e0, \
    IRQ 10.
3c509.c:1.12 6/4/97 becker@cesdis.gsfc.nasa.gov
Linux Version 2.0.32 (root@perf) (gcc Version 2.7.2.1)
#1 Tue Oct 21 15:30:44 EST 1997
 .
 .
```

Výpis ukazuje, že jádro bylo přeloženo se zapnutým protokolem TCP/IP, a obsahuje ovladače protokolů SLIP, CSLIP a PPP. Třetí řádek odspodu uvádí, že byla detekována ethernetová karta 3C509 a byla nainstalována jako rozhraní `eth0`. Máte-li jiný typ ethernetové karty – například kapesní adaptér D-Link – vypíše jádro zpravidla řádek začínající názvem tohoto zařízení – v případě karty D-Link to bude `dl0` – za ním pak následuje typ detekované karty. Pokud máte nainstalovanou síťovou kartu, ale nevidíte žádnou takovou zprávu, znamená to, že jádro není schopno správně vaši kartu detekovat. Tímto problémem se budeme zabývat později v části *Automatická detekce ethernetových zařízení*.

Konfigurace jádra

Většina distributorů Linuxu dodává zaváděcí diskety, které pracují se všemi běžnými typy hardwaru počítačů PC. Obecně je jádro na těchto disketách vysoce modulární a obsahuje prakticky všechny myslitelné ovladače. Je to vynikající řešení pro instalační diskety, nicméně dlouhodobě takové jádro asi nebudete chtít. Není nic vtipného na tom mít disk plný ovladačů zařízení, které jsou vám na nic. Typicky si proto sestavíte své vlastní jádro, jež bude obsahovat pouze ty ovladače, které skutečně chcete nebo potřebujete. Tím ušetříte něco diskového prostoru a zkrátíte čas pro překlad jádra.

Každopádně chcete-li používat systém Linux, měli byste ovládat sestavování jádra. Chápejte to jako důkaz vaší způsobilosti, potvrzení toho, proč jsou volně dostupné programy tak výkonné – máte k dispozici jejich zdrojový kód. Nedívejte se na to jako na „musím zkompilovat jádro“, ale raději jako na „*můžu* zkompilovat jádro“. Základy jsou vysvětleny v knize Matta Welshe *Running Linux*, vydané nakladatelstvím O'Reilly[25]. Proto se v této stati zmíníme pouze o těch konfiguračních volbách, které ovlivňují nastavení sítí.

25 Pozn. překladatele: A také v dokumentu *Linux Kernel HOWTO*, který je součástí této knihy.

Důležitá věc, kterou stojí za to zopakovat i zde, je způsob, jakým funguje číslování verzí jádra. Jádra Linuxu jsou číslována následujícím způsobem: `2.2.14`. První číslo udává *hlavní verzi jádra*. Toto číslo se mění, když dojde k velké a významné změně architektury jádra. Hlavní verze se tak změnila z 1 na 2, když byla implementována podpora i jiných procesorů než jen Intel. Druhé číslo je *vedlejší číslo verze*. V zásadě to je nejdůležitější číslo, které nás zajímá. V komunitě linuxových vývojářů byl přijat standard, že *sudá* vedlejší čísla označují *produkční* nebo *stabilní* jádra, zatímco *lichá* čísla představují *vývojová* nebo *nestabilní* jádra. Na počítači, na němž vám záleží, byste měli používat stabilní verze, protože ty jsou mnohem pečlivěji testovány. Vývojová jádra používáte, pokud chcete experimentovat s nejnovějšími funkcemi Linuxu, mohou však obsahovat dosud nenalezené a neodstraněné problémy. Třetí číslo se jednoduše inkrementuje s každým novým uvolněním vedlejší verze[26].

Při spuštění příkazu **make menuconfig** se objeví textová nabídka s řadou konfiguračních voleb, například zda chcete v jádře emulovat matematický koprocesor. Jeden z těchto dotazů se bude týkat instalace podpory pro sítě TCP/IP. Aby bylo vaše jádro schopno pracovat se sítěmi, musíte na tento dotaz odpovědět `y` (yes – ano).

Síťové volby v jádře 2.0 a vyšším

Po skončení úvodní obecné části konfigurace budete dále dotazováni, zda si přejete přidat podporu různých funkcí, například ovladačů SCSI nebo zvukových karet. Výzva zároveň nabízí možné odpovědi. Pokud odpovíte ?, objeví se popis toho, co vám daná volba konkrétně nabízí. Vždy budete mít možnost zvolit y pro statické zahrnutí komponenty do jádra nebo n pro její úplné vyloučení. U komponent, které mohou být přeloženy jako za běhu zaváděné moduly, se bude nabízet také volba m. Modulární ovladače musí být před použitím nahrány a jsou vhodné pro zařízení, která nepoužíváte často.

Následující seznam otázek se týká síťové problematiky. Přesná množina voleb se v důsledku probíhajícího vývoje neustále mění. Typický seznam nabízený většinou jader verze 2.0 a 2.1 vypadá takto:

```
*
* Network device support
*
Network device support (CONFIG_NETDEVICES) [Y/n/?]
Na tuto otázku musíte odpovědět y, pokud chcete používat jakékoliv síťové zaří-
zení, ať už Ethernet, SLIP, PPP nebo cokoliv jiného. Odpovíte-li y, automaticky
se zapne podpora ethernetových zařízení. Pokud chcete podporu i jiných zaříze-
ní, musíte kladně odpovědět i na další otázky:
PLIP (parallel port) support (CONFIG_PLIP) [N/y/m/?] y
PPP (point-to-point) support (CONFIG_PPP) [N/y/m/?] y
*
* CCP compressors for PPP are only built as modules.
*
SLIP (serial line) support (CONFIG_SLIP) [N/y/m/?] m
 CSLIP compressed headers (CONFIG_SLIP_COMPRESSED) [N/y/?] (NEW) y
 Keepalive and linefill (CONFIG_SLIP_SMART) [N/y/?] (NEW) y
 Six bit SLIP encapsulation (CONFIG_SLIP_MODE_SLIP6) [N/y/?] (NEW) y
```

Tyto otázky se týkají různých protokolů linkové vrstvy, které Linux podporuje. Protokoly PPP a SLIP umožňují přenos datagramů IP po sériové lince. PPP je vlastně skupina protokolů použí-

26 Je dobré, když se používají vývojová jádra a uživatelé hlásí v nich nalezené chyby. Pokud můžete nějaký počítač použít jako testovací, uděláte tak velmi užitečnou věc. Instrukce pro hlášení chyb jsou uvedeny v souboru `/usr/src/linux/REPORTING-BUGS` ve zdrojovém kódu jádra.

vaných pro přenos síťového provozu sériovou linkou. Některé z protokolů rodiny PPP řeší autentikaci u telefonního serveru, další řeší přenos konkrétních datových protokolů – PPP totiž není omezen na přenos IP datagramů, může přenášet i jiné protokoly, jako například IPX.

Pokud odpovíte y nebo m na dotaz na podporu protokolu SLIP, budete dále muset odpovědět ještě na tři otázky. Volba týkající se komprese hlaviček zapíná protokol CSLIP, v němž mohou být IP hlavičky komprimovány až na tři bajty. Všimněte si, že tato volba automaticky nezapíná použití protokolu CSLIP, pouze dodá do jádra funkce, které jeho použití dovolí. Volba Keepalive and linefill způsobí, že protokol SLIP bude na lince periodicky automaticky generovat nějakou aktivitu, aby nedocházelo k odpojení serverem kvůli neaktivitě. Konečně volba `Six bit SLIP` encapsulation umožňuje provozovat SLIP i na linkách, které nejsou schopny přenášet celou 8bitovou množinu znaků. Trochu se to podobá technice uuencoding nebo binhex, které se používají pro přenos binárních souborů elektronickou poštou.

Protokol PLIP dovoluje posílání IP datagramů přes paralelní port. Nejčastěji se používá pro komunikaci s dosovými stanicemi. Na typickém PC bude PLIP rychlejší než SLIP nebo PPP, má však výrazně vyšší režijní zatížení procesoru, takže zatímco přenos bude rychlý, ostatní úlohy mohou být zpomaleny.

Další otázky se týkají síťových karet různých výrobců. S vývojem dalších a dalších ovladačů se budou otázky v této části rozrůstat. Pokud chcete sestavit jádro pro použití na více počítačích nebo pokud máte v počítači instalováno více síťových karet, můžete povolit více než jeden ovladač:

```
  .
  .
Ethernet (10 or 100Mbit) (CONFIG_NET_ETHERNET) [Y/n/?]
3COM cards (CONFIG_NET_VENDOR_3COM) [Y/n/?]
3c501 support (CONFIG_EL1) [N/y/m/?]
3c503 support (CONFIG_EL2) [N/y/m/?]
3c509/3c579 support (CONFIG_EL3) [Y/m/n/?]
3c590/3c900 series (592/595/597/900/905) "Vortex/Boomerang" support/
    (CONFIG_VORTEX) [N/y/m/?]
AMD LANCE and PCnet (AT1500 and NE2100) support (CONFIG_LANCE) [N/y/?]
AMD PCInet32 (VLB and PCI) support (CONFIG_LANCE32) [N/y/?] (NEW)
Western Digital/SMC cards (CONFIG_NET_VENDOR_SMC) [N/y/?]
WD80*3 support (CONFIG_WD80x3) [N/y/m/?] (NEW)
SMC Ultra support (CONFIG_ULTRA) [N/y/m/?] (NEW)
SMC Ultra32 support (CONFIG_ULTRA32) [N/y/m/?] (NEW)
SMC 9194 support (CONFIG_SMC9194) [N/y/m/?] (NEW)
Other ISA cards (CONFIG_NET_ISA) [N/y/?]
Cabletron E21xx support (CONFIG_E2100) [N/y/m/?] (NEW)
DEPCA, DE10x, DE200, DE201, DE202, DE422 support (CONFIG_DEPCA) [N/y/m/?]/
    (NEW)
EtherWORKS 3 (DE203, DE204, DE205) support (CONFIG_EWRK3) [N/y/m/?] (NEW)
EtherExpress 16 support (CONFIG_EEXPRESS) [N/y/m/?] (NEW)
HP PCLAN+ (27247B and 27252A) support (CONFIG_HPLAN_PLUS) [N/y/m/?] (NEW)
HP PCLAN (27245 and other 27xxx series) support (CONFIG_HPLAN) [N/y/m/?]/
    (NEW)
HP 10/100VG PCLAN (ISA, EISA, PCI) support (CONFIG_HP100) [N/y/m/?] (NEW)
NE2000/NE1000 support (CONFIG_NE2000) [N/y/m/?] (NEW)
SK_G16 support (CONFIG_SK_G16) [N/y/?] (NEW)
EISA, VLB, PCI and on card controllers (CONFIG_NET_EISA) [N/y/?]
Apricot Xen-II on card ethernet (CONFIG_APRICOT) [N/y/m/?] (NEW)
Intel EtherExpress/Pro 100B support (CONFIG_EEXPRESS_PRO100B) [N/y/m/?]/
```

```
    (NEW)
DE425, DE434, DE435, DE450, DE500 support (CONFIG_DE4X5) [N/y/m/?] (NEW)
DECchip Tulip (dc21x4x) PCI support (CONFIG_DEC_ELCP) [N/y/m/?] (NEW)
Digi Intl. RightSwitch SE-X support (CONFIG_DGRS) [N/y/m/?] (NEW)
Pocket and portable adaptors (CONFIG_NET_POCKET) [N/y/?]
AT-LAN-TEC/RealTek pocket adaptor support (CONFIG_ATP) [N/y/?] (NEW)
D-Link DE600 pocket adaptor support (CONFIG_DE600) [N/y/m/?] (NEW)
D-Link DE620 pocket adaptor support (CONFIG_DE620) [N/y/m/?] (NEW)
Token Ring driver support (CONFIG_TR) [N/y/?]
IBM Tropic chipset based adaptor support (CONFIG_IBMTR) [N/y/m/?] (NEW)
FDDI driver support (CONFIG_FDDI) [N/y/?]
Digital DEFEA and DEFPA adapter support (CONFIG_DEFXX) [N/y/?] (NEW)
ARCnet support (CONFIG_ARCNET) [N/y/m/?]
  Enable arc0e (ARCnet "Ether-Encap" packet format) (CONFIG_ARCNET_ETH)/
      [N/y/?] (NEW)
  Enable arc0s (ARCnet RFC1051 packet format) (CONFIG_ARCNET_1051)/
      [N/y/?] (NEW)
 .
 .
```

Konečně v části věnované souborovým systémům se vás konfigurační skript zeptá, zda budete chtít podporovat síťový souborový systém NFS. NFS vám umožňuje exportovat souborové systémy na jiné počítače, což vaše soubory na těchto počítačích zpřístupní stejně, jako kdyby byly jejich lokální:

```
NFS file system support (CONFIG_NFS_FS) [y]
```

O NFS budeme podrobněji hovořit v kapitole 14.

Síťové volby v jádře 2.0.0 a vyšším

Linux 2.0.0 znamenal v oblasti sítí podstatnou změnu. Standardní součástí jádra se stala řada funkcí, například podpora IPX. Kromě toho byla přidána řada dalších voleb. Mnohé z nich se používají jen za velmi specifických okolností a nebudeme se jimi zabývat. O čem nebudeme mluvit zde, naleznete pravděpodobně v dokumentu Networking HOWTO. V dalším textu si popíšeme řadu užitečných voleb a vysvětlíme si, kdy byste měli jednotlivé volby použít.

Základ

Abyste mohli používat síťovou podporu TCP/IP, musíte na následující otázku odpovědět y. I pokud na ni odpovíte n, budete později moci zapnout podporu protokolu IPX.

```
Networking options  --->
    [*] TCP/IP networking
```

Brány

Tuto volbu musíte zapnout, pokud váš systém funguje jako brána mezi dvěma sítěmi nebo mezi lokální sítí a linkou SLIP a podobně. Pokud volbu necháte zapnutou vždy, nic se nestane, nicméně můžete ji vypnout a nakonfigurovat počítač jako *firewall*. Firewally jsou počítače připojené ke dvěma nebo více sítím, nepředávají však provoz mezi nimi. Typicky se používají k zajištění přístupu na Internet s minimalizací rizik pro interní síť. Uživatelé se mohou k firewallu přihlásit a používat internetové služby, nicméně interní počítače jsou chráněny před útoky z Internetu, protože firewall nepropustí dovnitř žádná spojení. (O firewallech budeme mluvit v kapitole 9.)

```
[*] IP: forwarding/gatewaying
```

Virtuální hostitelé

Následující volby vám umožní přiřadit jednomu rozhraní více IP adres. To může být užitečné, pokud provozujete takzvané „virtuální hostitele“ – tedy v případě, kdy jeden počítač vypadá a chová se jako více samostatných počítačů, každý s vlastní síťovou identitou. O problematice IP aliasů budeme více hovořit za chvíli.

```
[*] Network aliasing
    <*> IP: aliasing support
```

Účtování

Tato volba umožňuje shromažďovat data o IP provozu (budeme o ní hovořit v kapitole 10).

```
[*] IP: accounting
```

PC záplata

Tato volba umožňuje obejít nekompatibilitu s některými verzemi PC/TCP, tedy s komerční implementací TCP/IP na dosových počítačích. Při zapnutí této volby budete stále schopni komunikovat i s normálními linuxovými počítači, výkon linky se ale může lehce snížit.

```
--- (it is safe to leave these untouched)
[*] IP: PC/TCP compatibility mode
```

Bezdiskové bootování

Tato volba zapíná protokol RARP (*Reverse Address Resolution Protocol*). Protokol RARP používají bezdiskoví klienti a X terminály k vyžádání své IP adresy v době bootování. Pokud budete s tímto typem klientů pracovat, musíte tuto volbu zapnout. K přidávání záznamů do RARP tabulky jádra slouží krátký programek **rarp**, který je součástí standardních síťových nástrojů:

```
<*> IP: Reverse ARP
```

MTU

Při posílání dat protokolem TCP/IP musí jádro rozbít blok přenášených dat na menší kusy, které zpracovává protokol IP. Velikost těchto bloků se označuje jako *Maximum Transfer Unit*, MTU. Pro počítače přístupné lokální sítí, jako je například Ethernet, se MTU nastavuje na délku odpovídající nejdelší povolené délce paketu v Ethernetu – tedy na 1 500 bajtů. Pokud přenášíte data rozsáhlou sítí jako je Internet, měli byste zvolit menší velikost MTU, abyste zajistili, že vaše datagramy nebudou cestou dále děleny na menší kusy procesem zvaným *IP fragmentace*[27]. Jádro je schopno zjistit nejmenší MTU po trase a automaticky nakonfigurovat TCP spojení tak, aby tuto velikost používalo. Implicitně je toto chování zapnuto. Pokud na následující volbu odpovíte y, tato funkce se vypne.

Pokud budete chtít nastavit menší velikost datových bloků pro přenos na určité počítače (které jsou například připojeny linkou SLIP), můžete to udělat volbou **mss** příkazu **route**, o kterém budeme stručně mluvit na konci této kapitoly:

```
[ ] IP: Disable Path MTU Discovery (normally enabled)
```

Bezpečnostní funkce

Protokol IP podporuje funkci nazvanou *zdrojové směrování*. Tato funkce vám umožňuje definovat trasu pro přenos datagramu a zakódovat ji přímo v přenášeném datagramu. Bývalo to užiteč-

27 Nezapomínejte, že protokol IP může být přenášen celou řadou různých typů sítí, a ne všechny podporují stejně velké pakety jako Ethernet.

né v dobách, než byly běžně rozšířeny směrovací protokoly jako RIP nebo OSPF. V současné době je tato funkce chápána spíše jako bezpečnostní ohrožení, protože umožňuje schopnému útočníkovi obejít některé typy firewallových ochran tím, že se přenos nebude řídit směrovací tabulkou brány. Za normálních okolností proto chcete datagramy se zdrojovým směrováním odfiltrovat, a proto je tato volba standardně zapnuta:

```
[*] IP: Drop source routed frames
```

Podpora Novellu

Tato volba zapíná podporu protokolu IPX, tedy transportního protokolu používaného v sítích Novell. Linux může klidně fungovat jako směrovač protokolu IPX a tato podpora je užitečná v sítích, kde používáte souborové servery Novell. Podpora protokolu NCP rovněž vyžaduje zapnutí podpory IPX, takže pokud chcete přistupovat k souborovým systém novellových serverů, musíte tuto volbu zapnout. (O protokolu IPX a systému NCP budeme hovořit v kapitole 15.)

```
<*> The IPX protocol
```

Podpora rádia

Následující tři volby zapínají podporu tří protokolů podporovaných Linuxem a používaných v amatérských rádiových sítích. Jsou to AX.25, NetRom a Rose (v této příručce se jim nevěnujeme, jsou ale popsány v dokumentu AX25-HOWTO):

```
<*> Amateur Radio AX.25 Level 2
<*> Amateur Radio NET/ROM
<*> Amateur Radio X.25 PLP (Rose)
```

Linux podporuje ještě další ovladač, takzvaný *dummy driver*. Na začátku části věnované volbě síťových ovladačů se objevuje následující otázka:

```
<*> Dummy net driver support
```

Ovladač dummy toho dohromady moc nedělá, je ale velmi užitečný na samostatných počítačích nebo počítačích s PPP/SLIP připojením. Jedná se vlastně o maškarádu lokálního ovladače. Pokud máte počítač s připojením PPP/SLIP a žádným dalším rozhraním, budete zřejmě potřebovat rozhraní, které bude mít trvale přidělenu vaši IP adresu. Podrobněji o tom hovoříme v části *Rozhraní dummy* v kapitole 5. Dnes můžete stejného efektu dosáhnout pomocí IP aliasu s tím, že vám přidělenou IP adresu nastavíte jako alias lokálního rozhraní.

Prohlídka síťových zařízení v Linuxu

Jádro Linuxu podporuje množství ovladačů hardwaru pro různé typy zařízení. Tato stať obsahuje krátký přehled dostupných rodin ovladačů a názvů rozhraní, které ovladače používají.

V Linuxu existuje množství standardních názvů rozhraní, jejichž výčet následuje. Většina ovladačů podporuje více než jedno rozhraní. V takovém případě jsou rozhraní číslována, například *eth0*, *eth1* a podobně.

`lo`

Lokální zpětnovazebné rozhraní. Používá se pro testovací účely a pro dvojice síťových aplikací. Pracuje jako uzavřený obvod, kde všechny datagramy, které jsou na něj poslány, jsou okamžitě vráceny síťové vrstvě počítače. V jádru existuje vždy jedno toto lokální rozhraní a mít více těchto zařízení nemá smysl.

`eth0, eth1, ...`

Rozhraní ethernetových karet. Používá je většina standardních ethernetových karet včetně většiny ethernetových karet na paralelním portu.

`tr0, tr1, ...`

Rozhraní tokenringových karet. Používá je většina tokenringových karet včetně karet jiných výrobců než IBM.

`sl0, sl1, ...`

Rozhraní SLIP. Tato rozhraní se přidělují sériovým linkám v tom pořadí, jak jsou protokolem SLIP alokovány.

`ppp0, ppp1, ...`

Rozhraní PPP. Stejně jako rozhraní SLIP je i rozhraní PPP přiděleno sériové lince v okamžiku, kdy je sériová linka přepnuta do režimu PPP.

`plip0, plip1, ...`

Rozhraní PLIP. Protokol PLIP dopravuje datagramy po paralelních linkách. Tato rozhraní jsou přidělována ovladačem PLIP při procesu zavádění systému a jsou mapována na paralelní porty. V jádrech *2.0.x* je přímý vztah mezi názvem zařízení a číslem paralelního portu, u novějších jader jsou ovladače přidělovány sekvenčně jako u protokolů SLIP a PPP.

`ax0, ax1, ...`

Rozhraní AX.25. AX.25 je primární protokol používaný v radioamatérských sítích. Rozhraní protokolu AX.25 jsou alokována a mapována podobně jako zařízení protokolu SLIP.

Existuje celá řada dalších rozhraní pro jiné síťové ovladače. Zde jsme si uvedli pouze ty nejběžnější.

V následujícím textu se budeme podrobně věnovat použití výše popsaných ovladačů. O konfiguraci ostatních hovoří dokument Networking-HOWTO, o konfiguraci radioamatérských zařízení pak dokument AX25-HOWTO.

Instalace Ethernetu

Aktuální síťový kód Linuxu podporuje širokou skupinu ethernetových karet. Většinu ovladačů napsal Donald Becker, který je autorem rodiny ovladačů pro karty založené na čipu National Semiconductor 8390; tato rodina je známá pod názvem Becker Series Drivers. Celá řada dalších programátorů napsala další ovladače, takže dnes prakticky neexistuje ethernetová karta, kterou by Linux nepodporoval. Seznam podporovaných karet se navíc stále rozrůstá, takže pokud není vámi používaná karta podporována právě teď, může se to brzy změnit.

V minulosti jsme se pokusili o uvedení seznamu všech podporovaných ethernetových karet, dnes by to zabralo příliš mnoho práce a času. Naštěstí Paul Gortmaker udržuje dokument Ethernet-HOWTO, který obsahuje seznam všech podporovaných karet a uvádí u jednotlivých karet užitečné informace o tom, jak je v Linuxu rozběhnout[28]. Tento dokument se každý měsíc objevuje ve skupině *comp.os.linux.answers* a můžete jej také získat z některého zrcadla LDP.

I když jste si jisti že víte, jak konkrétní ethernetovou kartu nainstalovat, i tak je užitečné se do dokumentu Ethernet-HOWTO podívat, co se v něm o dané kartě říká. Dozvíte se informace nad standardní konfigurační údaje. Můžete si například ušetřit spoustu trápení, když budete vědět, že ně-

28 Paula můžete kontaktovat na adrese *gpg109@rsphy1.anu.edu.au*.

které ethernetové karty používající DMA používají standardně stejný DMA kanál jako SCSI řadič Adaptec 1542. Pokud kanál některého z těchto zařízení nezměníte, zjistíte, že vám síťová karta volně zapisuje přijímaná data na náhodná místa disku.

Chcete-li použít kteroukoliv podporovanou kartu, můžete využít předkompilované jádro kterékoliv z hlavních distribucí Linuxu. Obvykle obsahují moduly pro všechna podporovaná zařízení a instalační proces vám umožní zvolit, které ovladače chcete nahrát. Z dlouhodobého hlediska je ale lepší sestavit si vlastní jádro a přeložit pouze ty ovladače, které budete skutečně potřebovat. Ušetříte tak diskový prostor i paměť.

Automatická detekce Ethernetu

Řada ovladačů ethernetových karet je dostatečně chytrých na to, aby dokázaly najít vaši kartu. Díky tomu nemusíte jádru říkat, jak máte kartu nastavenu. V dokumentu Ethernet-HOWTO je uvedeno, zda jednotlivé ovladače provádějí automatickou detekci a v jakém pořadí prohledávají vstupně-výstupní adresy.

Automatická detekce má tři omezení. Za prvé jádro nemusí korektně rozpoznat všechny karty. To platí zejména pro některé levnější klony běžných karet. Druhým problémem je, že jádro nebude automaticky detekovat výskyt více než jedné karty, pokud mu to explicitně nenařídíte. Toto chování je záměrné, protože se předpokládá, že budete osobně chtít nastavit, které rozhraní bude které kartě přiřazeno. Nejspolehlivější metoda, jak toho dosáhnout, je nastavit konfiguraci karet ručně. A konečně, ovladač nemusí testovat tu adresu, na níž máte kartu nastavenu. Obecně platí, že ovladače automaticky zkoušejí všechny adresy, pro něž lze dané zařízení nakonfigurovat, někdy však některé adresy vynechávají, aby se předešlo hardwarovým konfliktům s jinými zařízeními, která mohou pracovat na stejných adresách.

Karty na sběrnici PCI by měly být detekovány spolehlivě. Pokud ale používáte více než jednu kartu nebo pokud se automatická detekce karty nezdaří, je třeba jádru explicitně nastavit bázovou adresu a název karty.

Při zavádění systému můžete uvést parametry a informace, které si může přečíst kterákoliv komponenta jádra. Tento mechanismus vám dovoluje předat jádru informace, které ethernetový ovladač použije k nalezení ethernetové karty, aniž by prováděl automatickou detekci.

Pokud pro zavádění systému používáte program lilo, můžete předat jádru parametry pomocí volby append v souboru lilo.conf. Chcete-li jádro informovat o ethernetovém zařízení, předejte mu následující parametr:

```
ether=irq,base_addr,[param1,][param2,]name
```

První čtyři parametry jsou číselné, zatímco poslední představuje název zařízení. Hodnoty *irq*, *base_addr* a *name* jsou povinné, dvě hodnoty *param* jsou nepovinné. Kterákoliv z číselných hodnot může být nulová, v takovém případě jádro použije autodetekci.

První parametr nastavuje IRQ kanál, který bude zařízení přidělen. Jádro se implicitně pokusí automaticky detekovat IRQ kanál daného zařízení. Ovladač 3c503 například disponuje speciální vlastností, která vybere volný IRQ kanál z kanálů 5, 9, 3 a 4 a nakonfiguruje kartu tak, aby tento kanál používala. Parametr *base_addr* udává bázovou vstupně-výstupní adresu karty. Zadáte-li hodnotu nula, jádro se pokusí hodnotu zjistit automaticky.

Zbývající dva parametry mohou různé typy ovladačů používat odlišným způsobem. U karet sdílejících paměť, jako například WD80x3, určují počáteční a koncovou adresu oblasti sdílené paměti. Ostatní karty používají zpravidla parametr *param1* k nastavení úrovně zobrazení ladicích informací. Hodnoty od 1 do 7 ukazují zvyšující se úroveň rozsahu výpisů, zatímco hodnota 8 ji vypíná;

0 udává implicitní volbu. Ovladač 3c503 používá *param2* k výběru vnitřního transceiveru (implicitně) nebo vnějšího transceiveru (hodnota 1). Nulová hodnota způsobí použití BNC konektoru na kartě; hodnota 1 vyvolá použití portu AUI. Pokud nepotřebujete nic zvláštního nastavit, hodnoty *param* nemusí být vůbec uvedeny.

Pokud máte dvě ethernetové karty, může Linux nechat automaticky detekovat jednu z nich a parametry druhé karty mu pak předáte pomocí `lilo`. Pokud se ale rozhodnete pro automatickou detekci jedné a ruční zadání druhé karty, je třeba zajistit, aby jádro náhodou nenašlo nejprve druhou kartu, protože v takovém případě by první vůbec nefungovala. Lze tak učinit pomocí volby *reserve* v programu `lilo`, která explicitně řekne jádru, aby nedetekovalo ve vstupně-výstupním prostoru obsazeném druhou kartou. Chcete-li například, aby Linux nainstaloval druhou ethernetovou kartu na adrese 0x300 jako eth1, musíte jádru předat následující parametry:

```
reserve=0x300,32 ether=0,0x300,eth1
```

Volba *reserve* zajistí, že žádný ovladač nebude při detekci „svého" zařízení prohlížet vstupně -výstupní oblast této druhé karty. Pomocí parametrů jádra můžete potlačit automatickou detekci i pro první kartu eth0:

```
reserve=0x340,32 ether=0,0x340,eth0
```

Automatickou detekci můžete vypnout úplně. Dělá se to například v případě, kdy nechcete, aby jádro automaticky detekovalo kartu, kterou jste dočasně odstranili. Vypnutí autodetekce se dosáhne tím, že parametru *base_addr* přiřadíte hodnotu -1:

```
ether=0,-1,eth0
```

Chcete-li parametry jádru předat v době spouštění systému, zadáváte je na výzvu „boot:" programu **lilo**. Aby program tuto výzvu zobrazil, musíte v době spouštění systému stisknout některou z kláves Control, Alt nebo Shift. Pokud na tuto výzvu stisknete tabulátor, objeví se seznam jader, která můžete nabootovat. Chcete-li nabootovat nějaké jádro s nějakými parametry, zadejte jméno jádra, mezeru a požadované parametry. Po stisknutí klávesy Enter se nahraje příslušné jádro a předají se mu zadané parametry.

Pokud chcete, aby se nastavené parametry automaticky uplatnily při každém spuštění systému, zadejte je v souboru `/etc/lilo.conf` pomocí klíčového slova `append=`. Příklad může vypadat takto:

```
boot=/dev/hda
root=/dev/hda2
install=/boot/boot.b
map=/boot/map
vga=normal
delay=20
append="ether=10,300,eth0"

image=/boot/vmlinuz-2.2.14
label=2.2.14
read-only
```

Po úpravě souboru `lilo.conf` musíte znovu spustit příkaz `lilo`, aby se změny uplatnily.

Ovladač PLIP

Protokol PLIP znamená *IP-protokol po paralelní lince* (*Parallel Line IP*) a představuje levnou alternativu sítě, pokud chcete propojit pouze dva počítače. Používá paralelní port společně se speciálním kabelem a dosahuje rychlosti od 10 KB/s do 20 KB/s.

Protokol PLIP původně vyvinula firma Crynwr, Inc. Jeho původní návrh byl poměrně prostý: po dlouhou dobu byly u počítačů PC paralelní porty používány pouze jako jednosměrné porty pro tiskárny; to znamená, že osm datových linek tohoto portu mohlo být použito pouze pro posílání informací z počítače do periferního zařízení, přenos opačným směrem nebyl možný. Návrh protokolu PLIP společnosti Crynwr se s tím vypořádal tak, že pět stavových linek používá pro vstup dat, což sice omezuje přenos pouze na poloviny bajtů najednou, nicméně umožňuje to obousměrnou komunikaci. Tento pracovní režim se nazývá režim 0 protokolu PLIP. Dnešní paralelní porty zvládají úplnou obousměrnou osmibitovou komunikaci a protokol PLIP byl rozšířen o režim 1, který toho využívá.

Jádra Linuxu do verze 2.0 podporovala pouze PLIP režim 0, a s rozšířením výskytu obousměrných paralelních portů byl vyvinut patch pro jádro 2.0, které tento port využívá. Od jádra 2.2 je již režim 1 podporován standardně[29]. Na rozdíl od předchozích verzí kódu PLIP je nynější ovladač kompatibilní s implementací společnosti Crynwr i s ovladačem protokolu PLIP v NCSA[30] **telnetu**. K propojení dvou počítačů za pomoci protokolu PLIP potřebujete speciální kabel, který seženete v některých obchodech pod označením kabel „Null Printer“ nebo „Turbo Laplink“. Poměrně jednoduše si ale můžete vyrobit kabel vlastní. Příloha A ukazuje, jak na to.

Linuxový ovladač protokolu PLIP je výsledkem práce obrovského množství lidí. Momentálně ho spravuje Niibe Yutaka[31]. Pokud je ovladač protokolu PLIP vestavěn do jádra, nastaví následující síťové rozhraní pro každý dostupný paralelní port, přičemž rozhraní plip0 bude odpovídat paralelnímu portu lp0, rozhraní plip1 paralelnímu portu lp1 a tak dále. Mapování rozhraní portům se liší v jádrech verze 2.0 a 2.2. V jádrech 2.0 bylo mapování napevno zakódováno v souboru drivers/net/Space.d zdrojového kódu jádra. Standardní mapování v tomto souboru vypadá takto:

Pokud máte porty tiskárny nastaveny jinak, musíte tyto hodnoty v souboru drivers/net/Space.c upravit a znovu sestavit jádro.

V jádrech 2.2 využívá ovladač PLIP sdílení paralelního portu mechanismem „parport“ vyvinutým Philipem Blundellem[32]. Nový ovladač alokuje jména síťových zařízení PLIP postupně stejně jako u ovladačů ethernetových karet nebo PPP. První vytvořené PLIP zařízení tedy bude plip0, druhé bude plip1 a tak dále. Fyzická paralelní rozhraní se rovněž alokují postupně. Standardně se ovladač paralelního portu snaží nalézt fyzické paralelní porty autodetekcí a očísluje je v tom pořadí, v jakém je nalezne. Lepší je jádru explicitně sdělit vstupně-výstupní parametry jednotlivých fyzických portů. Můžete to provést buď předáním parametrů modulu parport_pc.o při jeho nahrání, nebo pokud máte tento ovladač napevno vestavěn v jádře, můžete programem **lilo** předat jádru příslušné parametry při spouštění systému. Nastavení přerušení kteréhokoliv zařízení je možné později změnit zapsáním nové hodnoty přerušení do odpovídajícího souboru `/proc/parport*/irq`.

Konfigurace fyzických vstupně-výstupních parametrů v jádře 2.2 při nahrávání modulu je poměrně jednoduchá. Chcete-li ovladači říct, že máte dva paralelní porty na adresách 0x278 a 0x378 s přerušeními 5 a 7, nahrajete modul s následujícími parametry:

```
modprobe parport_pc io=0x278,0x378 irq=5,7
```

---

[29] Patch podporující obousměrné porty v jádře 2.0 je dostupný na adrese *http://www.cyberelk.demon.co.uk/parport.html*.

[30] NCSA **telnet** je oblíbený program pro DOS, který používá TCP/IP přes Ethernet nebo PLIP a podporuje služby telnet a FTP.

[31] Niibeho můžete kontaktovat na adrese *gniibe@mri.co.jp*.

[32] Philipa můžete kontaktovat na adrese *Philip.Blundell@pobox.com*.

Stejné parametry předávané jádru pro vestavěný ovladač budou vypadat takto:

```
parport=0x278,5 parport=0x378,7
```

Pomocí klíčového slova *append* můžete tyto parametry předávat jádru automaticky při každém spouštění systému.

Jakmile se inicializuje ovladač PLIP, ať už při spouštění systému, je-li vestavěný do jádra, nebo po nahrání modulu plip.o, bude každému paralelnímu rozhraní přiřazeno odpovídající zařízení plip. Tedy plip0 bude přiřazeno prvnímu paralelnímu rozhraní, plip1 druhému a tak dále. Tato automatická nastavení můžete změnit ručně dalšími parametry jádra. Pokud budete chtít například přiřadit parport0 síťovému zařízení plip1 a parport1 síťovému zařízení plip0, použijete následující parametry jádra:

```
plip=parport1 plip=parport0
```

Nicméně toto mapování neznamená, že nemůžete používat paralelní porty obvyklým způsobem. Ovladač protokolu PLIP přistupuje k paralelním portům pouze v případě, že je k němu nakonfigurováno odpovídající rozhraní.

Ovladače SLIP a PPP

Protokoly PPP (Point-to-Point Protocol) a SLIP (Serial Line IP) se hojně využívají k přenášení IP paketů po sériové lince. Množství firem nabízí přístup k Internetu prostřednictvím telefonického připojení protokoly SLIP nebo PPP. Tímto způsobem se obvykle připojují soukromé osoby, pro něž je jiný typ připojení cenově nedostupný.

Abyste mohli používat ovladače SLIP nebo PPP, nejsou nutné žádné hardwarové úpravy; stačí použít libovolný sériový port. Protože konfigurace sériových portů není specifickou problematikou sítí na bázi TCP/IP, bude jim věnována samostatná kapitola. Více informací o PPP tak naleznete v kapitole 8 a o SLIP v kapitole 7.

Další typy sítí

Řada jiných sítí se konfiguruje podobně jako Ethernet. Parametry předávané zaváděným modulům budou jiné a některé ovladače nepodporují více karet než jednu, nicméně všechno ostatní je prakticky stejné. Dokumentaci k různým kartám obecně najdete v adresáři `/usr/src/linux/Documentation/networking/` ve zdrojovém kódu jádra.

Konfigurace sériových zařízení

Internet se rozrůstá neuvěřitelným tempem. Velká část tohoto růstu jde na vrub uživatelům, kteří si nemohou dovolit vysokorychlostní trvalá připojení a kteří používají protokoly jako SLIP, PPP nebo UUCP k telefonickému spojení s poskytovatelem internetového připojení, od kterého si tak stáhnou svou denní dávku konferencí a pošty.

Tato kapitola má pomoci všem lidem, kteří jsou odkázáni pouze na modemy. Nebudeme se pouštět do popisů mechanismů jak nakonfigurovat modem (o tom vám podstatně více než my řekne manuál, který k modemu dostanete), nicméně pokryjeme většinu linuxově zaměřené problematiky týkající se práce se zařízeními, jež komunikují sériovým portem. Budeme hovořit o programech pro sériovou komunikaci, o vytvoření souboru sériového zařízení, o sériovém hardwaru a o konfiguraci sériových zařízení příkazy **setserial** a **stty**. Řada dalších témat je uvedena v dokumentu Serial-HOWTO Davida Lawyera[33].

Komunikační software pro modemové linky

V Linuxu je k dispozici mnoho komunikačních balíků. Spoustu z nich tvoří *terminálové programy*, které umožňují uživateli propojení s jiným počítačem a jež se tváří, jakoby uživatel seděl před jednoduchým terminálem. Tradičním unixovým terminálovým programem je **kermit**. Dnes už však je značně zastaralý a pravděpodobně byste jej považovali za příliš složitý na používání. Dnes jsou dostupné mnohem komfortnější programy, které podporují různé další funkce jako adresář s telefonními čísly, skriptové jazyky pro automatické volání a připojování ke vzdáleným počítačovým systémům a řada různých souborových komunikačních protokolů. Jeden z nich se nazývá **minicom** a byl vyvinut podle některých nejpopulárnějších dosových terminálových programů. Spokojeni budou i uživatelé X11, plně vybaveným komunikačním programem na bázi X11 je například **seyon**.

Terminálové programy nicméně nejsou jediné dostupné programy pro sériovou komunikaci. Existují jiné programy, které vám dovolují připojit se ke vzdálenému počítači a stáhnout zprávy a poštu najednou, přičemž jejich přečtení a zodpovězení provedete později. Tím můžete ušetřit hodně času a bude to pro vás výhodné zejména pokud žijete někde, kde jsou i místní hovory časově tarifikovány. Čtení a zodpovězení provedete offline a poté se znovu připojíte a všechny odpovědi najednou odešlete. Toto řešení samozřejmě vyžaduje trochu více diskového prostoru, protože všechny zprávy musíte mít před přečtením uloženy na disku, při současných cenách pevných disků to ale nepředstavuje zásadní nevýhodu.

33 Davida můžete kontaktovat na adrese *bf347@lafn.org*.

Typickým představitelem tohoto typu softwaru je UUCP. Jedná se o programový balík, který kopíruje data z jednoho hostitele na druhý a spouští programy na vzdáleném hostiteli. Často se používá pro přenos pošty nebo konferencí v soukromých sítích. Balík UUCP od Iana Taylora, který běží také v prostředí Linuxu, bude popsán v kapitole 16. Další neinteraktivní komunikační software se používá například v sítích Fidonet. Jsou k dispozici i aplikace přenesené na platformu Linuxu, například **ifmail**, i když si nemyslíme, že by je používalo hodně lidí.

Protokoly PPP a SLIP stojí někde mezi, protože umožňují jak interaktivní, tak i neinteraktivní použití. Mnoho lidí využívá protokol PPP nebo SLIP pro připojení ke školní síti nebo k nějakému komerčnímu poskytovateli. Mohou tak využívat služby protokolu FTP a jiných. Protokoly PPP a SLIP se ale také často používají k propojení několika lokálních sítí pevnými nebo občasnými linkami. Toto využití je však zajímavé pouze pokud použijeme zařízení ISDN nebo jiné rychlé zařízení.

Úvod k sériovým zařízením

Zařízení, pomocí nichž umožňuje jádro Unixu přístup k sériovým zařízením, se typicky nazývají *tty* (vyslovuje se to stejně jako při hláskování v angličtině, tedy *tí-tí-uaj*). Je to zkratka názvu společnosti *Teletype*, která bývala v počátcích unixové éry jedním z hlavních výrobců terminálů. V současnosti se tento termín používá pro jakékoliv znakově orientované datové terminály. V této kapitole budeme tento termín používat výhradně k označení zařízení jádra, nikoliv fyzického terminálu.

Linux rozlišuje tři třídy tty zařízení: sériová zařízení, virtuální zařízení (k nimž můžete na lokální konzole přistupovat stiskem kláves Alt+F1 až Alt+F*nn*) a pseudoterminály (podobné obousměrnému potrubí, které používají aplikace jako je X11). První zmíněná třída je rovněž považována za tty zařízení, protože původní znakově orientované terminály byly k linuxovému počítači připojovány sériovým kabelem nebo telefonní linkou a modemem. Druhé dvě třídy jsou klasická tty zařízení, protože se z pohledu programátora chovají stejně.

Protokoly SLIP a PPP bývají velmi často implementovány přímo v jádru. Jádro nechápe *tty* zařízení jako síťové zařízení, se kterým byste mohli pracovat stejně jako s ethernetovou kartou například příkazem **ifconfig**. Chápe nicméně *tty* zařízení jako místa, k nimž lze síťová zařízení připojit. Provede se to tak, že jádro změní takzvanou „linkovou disciplínu“ tty zařízení. SLIP i PPP jsou linkové discplíny, které mohou být na tty zařízeních povoleny. Hlavní myšlenka je ta, že sériový ovladač obsluhuje příchozí data různě podle toho, pro jakou discíplinu je linka nakonfigurována. Při použití standardní disciplíny ovladač jednoduše předává každý znak, který obdrží. Když je zvolena disciplína SLIP nebo PPP, ovladač místo toho čte bloky dat, přidá k nim speciální hlavičky, které umožní vzdálenému zařízení identifikovat blok dat v datovém proudu a tento nový blok odešle. Není nijak důležité, abychom tomu nyní rozuměli, o protokolech SLIP a PPP budeme hovořit v dalších kapitolách a stejně se to všechno děje automaticky.

Přístup k sériovým zařízením

Stejně jako ke všem unixovým zařízením je i k sériovým zařízením přistupováno pomocí speciálních souborů zařízení, které se nacházejí v adresáři `/dev`. Existují dvě skupiny souborů zařízení, které se vztahují k ovladačům sériových zařízení, přičemž každé sériové zařízení má přiřazen jeden soubor z každé skupiny. Zařízení se bude chovat různě v závislosti na typu souboru, kterým toto zařízení otevřeme. Hned si tento rozdíl vysvětlíme, protože to může být užitečné pro snazší pochopení konfigurace a některých doporučení, s nimiž se můžete při práci se sériovými porty

potkat. Nicméně v praxi stejně budete používat pouze jeden z těchto typů a v budoucnu možná ten druhý úplně vymizí.

Důležitější ze zmíněných dvou tříd jsou zařízení s hlavním číslem 4 a soubory těchto speciálních zařízení jsou pojmenovány `ttyS0`, `ttyS1` a tak dále. Druhá skupina má hlavní číslo 5 a používá se pro vytáčení směrem z počítače; soubory zařízení se jmenují `cua0`, `cua1` a tak dále. V unixovém světě se obecně počítá od nuly, i když „normální lidé" mají tendenci počítat od jedničky. To přináší lehké zmatky, protože port `COM1:` je reprezentován zařízením `/dev/ttyS0`, port `COM2:` zařízením `/dev/ttyS1` a tak dále. Kdokoliv se trochu orientuje v problematice počítačů IBM PC ví, že porty `COM3:` a vyšší stejně nebyly nikdy pořádně standardizovány.

Zařízení *cua* (callout) byla vyvinuta kvůli odstranění problémů s konflikty na sériovém zařízení při práci s modemy, které musí obsluhovat příchozí i odchozí spojení. Bohužel vytvořily své vlastní nové problémy a dnes se již prakticky nepoužívají. Podívejme se nyní, v čem problém spočívá.

Linux, stejně jako Unix, umožňuje každému zařízení nebo souboru, aby jej otevřelo více procesů současně. U tty zařízení to typicky není dobré, protože oba procesy se budou prakticky trvale rušit. Naštěstí byl vyvinut mechanismus umožňující, aby proces mohl před pokusem o otevření tty zařízení ověřit, zda je nemá otevřeno už jiný proces. Tento mechanismus se označuje jako *zámkové soubory*. Celá myšlenka spočívá v tom, že chce-li proces otevřít nějaké tty zařízení, na určitém místě hledá soubor pojmenovaný stejně jako zařízení, které chce otevřít. Pokud tento soubor neexistuje, proces jej vytvoří a otevře zařízení. Pokud by soubor existoval, proces bude předpokládat, že zařízení již bylo otevřeno jiným procesem a podle toho se zachová. Posledním chytrým trikem, jenž zajišťuje chod mechanismu zamykání prostřednictvím souborů, je to, že proces, který zámkový soubor vytvoří, do něj zapíše svůj vlastní identifikátor procesu (pid). Budeme o tom hovořit za chvíli.

Mechanismus souborových zámků bezvadně funguje v situaci, kdy je přesně definováno umístění těchto souborů a kdy všechny procesy vědí, kde je mají hledat. To ovšem nebyla vždycky pravda. Neplatilo to do doby, dokud standard Linux Filesystem Standard nedefinoval pevné umístění těchto souborů. Jednu dobu existovaly přinejmenším čtyři (ne-li více) míst, kde různé programy tyto soubory ukládaly: `/usr/spool/locks`, `/var/spool/locks`, `/var/lock/` a `/usr/lock`. Tato nejednotnost vedla k chaosu. Programy vytvářely zámkové soubory téhož zařízení na různých místech, což bylo stejné, jako kdyby tento mechanismus nepoužívaly vůbec.

Jako řešení tohoto problému byla vytvořena zařízení `cua`. Namísto aby se při řešení sporů o sériovou linku spoléhalo na zámkové soubory, rozhodlo se, že tyto spory bude velmi jednoduše řešit přímo jádro. Pokud bylo zařízení `ttyS` otevřeno, pokus o otevření zařízení `cua` vedl k chybě, podle níž mohl volající proces poznat, že zařízení již je otevřeno. Pokud bylo otevřeno zařízení `cua` a došlo k pokusu o otevření zařízení `ttyS`, žádost byla zablokována – to znamená, že byla umístěna do fronty a čekala do doby, než bude zařízení `cua` uvolněno. Pracovalo to poměrně dobře, pokud jste používali jeden modem, kterým jste obsluhovali příchozí volání a čas od času jste chtěli tímto zařízením provést odchozí volání. Nefunguje to ale tehdy, pokud máte více programů, které chtějí všechny realizovat odchozí volání jedním zařízením. Jediná možnost řešení tohoto problému – používat zámkové soubory!

Naštěstí standard Linux Filesystem Standard nadefinoval umístění zámkových souborů do adresáře `/var/lock` a stanovil konvenci, že například zámkový soubor zařízení `ttyS1` bude pojmenován `LCK..ttyS1`. Do stejného adresáře se ukládají i zámkové soubory zařízení `cua`, nicméně použití tohoto typu zařízení se dnes nedoporučuje.

Zařízení `cua` budou pravděpodobně ještě nějakou dobu existovat kvůli zpětné kompatibilitě, postupem času však vymizí. Pokud nevíte, které zařízení použít, použijte `ttyS` a ověřte si, že používáte Linux kompatibilní se standardem FSSTND nebo že alespoň všechny programy pracující se sériovým zařízením používají stejné umístění zámkových souborů. Většina programů, které se sériovou linkou pracují, obsahují parametr pro sestavení, kterým můžete umístění zámkových souborů definovat. Vesměs se jedná o proměnnou `LOCKDIR` v souboru `Makefile` nebo v konfiguračním hlavičkovém souboru. Pokud program překládáte, je dobré tento parametr nastavit tak, aby byl kompatibilní s FSSTND. Pokud používáte již přeložený program a nejste si jisti, kam zámky ukládá, zkuste to zjistit následujícím příkazem:

```
strings binaryfile | grep lock
```

Pokud nalezené umístění neodpovídá tomu, co používá zbytek systému, můžete zkusit vytvořit symbolický odkaz z adresáře používaného tímto programem na `/var/lock`. Není to moc pěkné, ale funguje to.

Soubory sériových zařízení

Vedlejší čísla zařízení jsou pro oba typy stejná. Máte-li modem připojen na jednom z portů COM1: až COM 4:, bude vedlejší číslo rovno číslu COM portu plus 63. Pokud používáte speciální sériový hardware, například kartu, která podporuje několik sériových linek, budete mu muset zřejmě vytvořit vlastní soubor zařízení, pravděpodobně nebude pracovat se standardním souborem. Příslušné podrobnosti byste měli najít v dokumentu Serial-HOWTO.

Budeme předpokládat, že váš modem je připojen na port COM2:. Jeho vedlejší číslo tak bude mít hodnotu 65 a hlavní číslo bude při normálním použití rovno 4. Mělo by existovat zařízení `ttyS1`, které bude mít tato čísla. Vypište si sériová tty zařízení z adresáře `/dev`. Pátý a šestý sloupec by měly obsahovat hlavní a vedlejší číslo:

```
$ ls -l /dev/ttyS*
0 crw-rw----   1 uucp     dialout    4,  64 Oct 13  1997 /dev/ttyS0
0 crw-rw----   1 uucp     dialout    4,  65 Jan 26 21:55 /dev/ttyS1
0 crw-rw----   1 uucp     dialout    4,  66 Oct 13  1997 /dev/ttyS2
0 crw-rw----   1 uucp     dialout    4,  67 Oct 13  1997 /dev/ttyS3
```

Jestliže neexistuje zařízení s hlavním číslem 4 a vedlejším číslem 65, musíte je vytvořit: přihlaste se jako superuživatel a napište:

```
# mknod -m 666 /dev/ttyS1 c 4 65
# chown uucp.dialout /dev/ttyS1
```

Různé distribuce Linuxu používají různé strategie pro toho, kdo by měl sériové zařízení vlastnit. Někdy je vlastní uživatel *root*, jindy nějaký jiný uživatel, například *uucp*. Moderní distribuce používají skupinu uživatel s přístupem k sériovým zařízením a kdokoliv je má používat, přidá se do této skupiny.

Někteří lidé doporučují vytvořit soubor `/dev/modem`, který by sloužil jako symbolický odkaz na váš modem, takže si příležitostní uživatelé nemusí pamatovat poněkud neintuitivní název například `ttyS1`. Nemůžete pak ale v jednom programu používat soubor modem a ve druhém skutečný název souboru zařízení. Jejich zámkové soubory by pak měly různá jména a mechanismus zamykání by nefungoval.

Sériový hardware

Nejrozšířenějším standardem pro sériovou komunikaci ve světě PC je dnes RS-232. K přenosu jednotlivých bitů a synchronizaci využívá několik elektronických obvodů. Další linky lze využít k signalizaci výskytu nosné (kterou používají modemy) a k inicializačnímu handshakingu.

Hardwarový handshaking je nepovinný, je ale velice užitečný. Umožňuje oběma stanicím signalizovat, zda jsou schopny přijmout další data, nebo zda by měla druhá strana počkat, až přijímající strana dokončí zpracování přijatých dat. K tomuto účelu se používají linky „Clear to Send" (CTS) a „Ready to Send" (RTS), z jejichž názvů je odvozen hovorový název celého hardwarového handshakingu: „RTS/CTS". Druhým známým typem handshakingu je „XON/XOFF". Tento mechanismus používá dva speciální znaky, typicky Ctrl+-S a Ctrl+Q, kterými se vzdálené straně signalizuje, že má zastavit nebo zahájit odesílání dat. Tato metoda je jednoduchá na implementaci a použitelná v hloupých terminálech; může způsobit problémy, pokud přenášíte binární data, protože tyto znaky můžete odeslat jako součást dat a nebudete je chtít považovat za řídicí znaky. Navíc je toto řešení poněkud pomalejší než hardwarové řešení. Hardwarové řešení je čistší, rychlejší a pokud máte na výběr, doporučuje se je upřednostnit před mechanismem XON/XOFF.

V původních počítačích PC je rozhraní RS-232 obvykle řízeno čipem UART pojmenovaným 8250. Počítače zhruba od doby 486 používají UART pojmenovaný 16450. Ten je poněkud rychlejší než 8250. Většina počítačů Pentium obsahuje ještě novější verzi UART čipu, pojmenovanou 16550. Některá zařízení (typicky interní modemy firmy Rockwell) používají úplně jiný čip, který emuluje chování čipu 16550 a může být obsluhován stejně. Linux ve standardním ovladači sériového portu podporuje všechny tyto čipy[34].

Čip 16550 je oproti čipům 8250 a 16450 zásadně vylepšen, protože obsahuje vyrovnávací paměť FIFO o velikosti 16 bajtů. Čip 16550 vlastně představuje celou rodinu UART zařízení, do které patří 16550, 16550A a 16550AFN (později přejmenovaný na PC16550DN). Rozdíl spočívá v tom, zda vyrovnávací paměť opravdu funguje – což se s jistotou ví pouze o čipu 16550AFN. Existoval i čip NS16550, ale jeho vyrovnávací paměť nikdy nefungovala.

Použití konfiguračních nástrojů

Nyní se podívejme na dva nástroje, které se pro konfiguraci sériových linek nejčastěji používají. Jsou to programy **setserial** a **stty**.

Příkaz setserial

Jádro se velmi snaží o správné zjištění konfigurace sériových zařízení, nicméně různé rozdíly v jejich vlastnostech znemožňují dosažení stoprocentní spolehlivosti. Dobrým příkladem jsou například problémy s interními modemy, o kterých jsme už hovořili. Jimi používaný čip UART má 16bajtový buffer, nicméně tváří se jako čip 16450 – pokud tedy jádru explicitně neřekneme, že jde o čip 16550, nebude tento buffer využívat. Dalším příkladem mohou být hloupé čtyřportové karty, které sdílejí jedno přerušení pro všechna sériová zařízení. Musíme jádru přesně říct, která přerušení se používají a že jsou případně sdílená.

Program **setserial** slouží k nastavení sériových portů za běhu. Tento příkaz se typicky spouští v době spouštění systému skriptem, který se na některých distribucích jmenuje `0setserial`, na

34 Všimněte si, že se vůbec nezmiňujeme o zařízení WinModem(TM)! Tato zařízení jsou velice jednoduchá a k provedení veškeré potřebné práce plně spoléhají na procesor a nikoliv na specializovaný hardware. Pokud si kupujete modem, vřele vám doporučujeme *nekupovat* nic takového, kupte si opravdový modem. Podpora zařízení WinModem existuje i v Linuxu, to ale neznamená, že by je měl někdo používat.

jiných `rc.serial`. Tento skript zodpovídá za nastavení sériových zařízení tak, aby byl správně podchycen všechen nestandardní nebo neobvyklý hardware v počítači.

Obecná syntaxe příkazu **setserial** je:

```
setserial device [parametry],
```

kde parametr `device` představuje některé sériové zařízení, například *ttyS0*.

Program **setserial** používá celou řadu parametrů. Nejběžnější z nich jsou shrnuty v tabulce 4.1. Informace o dalších parametrech najdete na manuálových stránkách programu **setserial**.

Tabulka 4.1 – Řádkové parametry programu setserial

Parametr	Popis
`port` *port_number*	Udává adresu vstupně-výstupního portu sériového zařízení. Hodnota se zadává hexadecimálně, například 0x2f8.
`irq` *num*	Udává číslo použitého přerušení.
`uart` *uart_type*	Specifikuje UART čip zařízení. Možné hodnoty jsou 16450, 16550 a podobně. Nastavení této hodnoty na *none* vypne dané sériové zařízení.
`fourport`	Tento parametr jádru říká, že daný port je jedním z portů karty AST Fourport.
`spd_hi`	Naprogramuje UART tak, aby pracoval rychlostí 57,6 kb/s, pokud proces požaduje rychlost 38,4 kb/s.
`spd_vhi`	Naprogramuje UART tak, aby pracoval rychlostí 115,2 kb/s, pokud proces požaduje rychlost 38,4 kb/s.
`spd_normal`	Naprogramuje UART na použití standardní rychlosti 38,4 kb/s. Tento parametr se používá k odstranění účinku parametrů spd_hi nebo spd_vhi na daném zařízení.
`auto_irq`	Tento parametr způsobí, že se jádro pokusí automaticky detekovat IRQ daného zařízení. Výsledek nemusí být úplně spolehlivý, takže se na tuto operaci dívejme spíše jako na „hádání" přerušení. Pokud znáte číslo přerušení, které zařízení používá, měli byste je zadat pomocí parametru irq.
`autoconfig`	Tento parametr může být zadán jen společně s parametrem port. Způsobí, že jádro bude automaticky detekovat typ UART na daném portu. Pokud je zadán i parametr auto_irq, bude se automaticky detekovat i přerušení.
`skip_test`	Tento parametr jádru říká, aby se nepokoušelo o automatickou detekci UART čipu. Používá se tehdy, pokud jádro stejně čip správně nedetekuje.

Typický a jednoduchý rc soubor pro nastavení sériových portů při spouštění systému vypadá tak, jako na příkladu 4.1. Většina distribucí Linuxu obsahuje něco lepšího, než je jen tento skript:

Příklad 4.1 – Příklad konfiguračního skriptu rc.serial

```
# /etc/rc.serial - konfigurační skript sériové linky.
#
# Konfigurace sériových zařízení
/sbin/setserial /dev/ttyS0 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS1 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS2 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS3 auto_irq skip_test autoconfig
#
# Zobrazení konfigurace sériových zařízení
/sbin/setserial -bg /dev/ttyS*
```

Parametr `-bg /dev/ttyS*` v posledním příkazu způsobí vytištění pěkně formátovaného přehledu hardwarové konfigurace všech aktivních sériových zařízení. Výstup bude vypadat podobně jako v příkladu 4.2:

Příklad 4.2 – Výstup příkazu setserial –bg /dev/ttyS*

```
/dev/ttyS0 at 0x03f8 (irq = 4) is a 16550A
/dev/ttyS1 at 0x02f8 (irq = 3) is a 16550A
```

Příkaz stty

Název příkazu **stty** zřejmě znamená „set tty“, nicméně příkazem **stty** je možné také zobrazit konfiguraci terminálů. Program **stty** umožňuje nastavení ještě rozsáhlejšího souboru vlastností než program **setserial**. Za chvilku si popíšeme nejdůležitější z nich. Zbytek pak najdete na manuálových stránkách programu **stty**.

Příkaz **stty** se nejčastěji používá ke konfiguraci parametrů terminálů, například zda mají být lokálně zobrazovány odesílané znaky nebo která klávesa má generovat přerušovací signál. Říkali jsme si už, že sériová zařízení jsou zařízení typu tty, takže program **stty** je použitelný i pro ně.

Jedno z nejdůležitějších použití programu **stty** pro sériová zařízení spočívá v povolení hardwarového handshakingu. O hardwarovém handshakingu jsme už stručně mluvili. Standardní nastavení sériových zařízení má hardwarový handshaking vypnut. Toto nastavení umožňuje použití třívodičových sériových kabelů, které nepodporují signály potřebné pro hardwarový handshaking, takže pokud bychom jej standardně zapnuli, nebyli bychom schopni odeslat žádný znak a toto nastavení změnit.

Překvapivě hardwarový handshaking nezapínají ani některá sériová komunikační zařízení, takže pokud váš modem hardwarový handshaking podporuje, nakonfigurujte jej tak, aby jej používal (jak to udělat najdete v návodu k modemu) a stejně nakonfigurujte i sériové zařízení. Příkaz **stty** má příznak crtscts, který zapíná hardwarový handshaking daného zařízení, takže jej budete muset uvést. Tento příkaz se nejlépe spouští ze souboru rc.serial (nebo jeho ekvivalentu) v době spouštění systému například tak, jak to ukazuje příklad 4.3.

Příklad 4.3 – Příklad skriptu rc.serial s programem stty

```
#
stty crtscts < /dev/ttyS0
stty crtscts < /dev/ttyS1
stty crtscts < /dev/ttyS2
stty crtscts < /dev/ttyS3
#
```

Příkaz **stty** standardně pracuje s aktuálním terminálem, pokud ale použijeme přesměrování vstupu („<“), můžeme příkazem **stty** nastavit jakékoliv zařízení. Běžnou chybou je zapomenutí, zda se má použít symbol „<“ nebo „>“, moderní verze příkazu **stty** používají jasnější syntaxi. Při použití nové syntaxe bude stejný příklad vypadat tak, jako příklad 4.4:

Příklad 4.4 – Příklad skriptu rc.serial s programem stty a moderní syntaxí

```
#
stty crtscts -F /dev/ttyS0
stty crtscts -F /dev/ttyS1
stty crtscts -F /dev/ttyS2
stty crtscts -F /dev/ttyS3
#
```

Zmínili jsme se už, že příkazem **stty** můžete zobrazit konfigurační parametry tty zařízení. Pokud chcete zobrazit všechna nastavení daného tty zařízení, použijte příkaz:

```
$ stty -a -F /dev/ttyS1
```

Výstup tohoto příkazu, který vidíte v příkladu 4.5, obsahuje hodnoty všech stavových příznaků daného zařízení. Příznaky uvedené symbolem „minus“, například `-crtscts`, udávají, že tento příznak je vypnut.

Příklad 4.5 – Výstup příkazu stty –a

```
speed 19200 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\ ; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
       eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
       werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl -ixon
        -ixoff -iuclc -ixany -imaxbel
-opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0
       bs0 vt0 ff0
-isig -icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop
      -echoprt echoctl echoke
```

Přehled nejdůležitějších příznaků obsahuje tabulka 4.2. Jednotlivé příznaky se zapnou tak, že se příkazu **stty** předají a vypnou tak, že se mu předají s uvádějícím symbolem minus. Pokud tedy budeme chtít na zařízení ttyS0 vypnout hardwarový handshaking, zadáme:

```
$ stty -crtscts -F /dev/ttyS0
```

Tabulka 4.2 – Důležité příznaky příkazu stty pro konfiguraci sériových zařízení

Příznak	Popis
`N`	Nastavuje rychlost linky na N bitů za sekundu.
`crtscts`	Zapíná/vypíná hardwarový handshaking.
`ixon`	Zapíná/vypíná XON/XOFF handshaking.
`clocal`	Zapíná/vypíná signály pro řízení modemu jako DTR/DTS a DCD. Je to nutné při použití třívodičového sériového kabelu, protože ten tyto signály nepřenáší.
`cs5 cs6 cs7 cs8`	Nastavuje počet datových bitů na 5, 6, 7 respektive 8.
`parodd`	Zapíná lichou paritu. Je-li příznak vypnut, používá se sudá parita.
`parenb`	Zapíná paritu. Není-li příznak zapnut, kontrola parity se neprovádí.
`cstopb`	Zapíná použití dvou stop-bitů. Pokud je příznak vypnut, používá se jen jeden stop-bit.
`echo`	Zapíná/vypíná lokální echo odesílaných znaků.

Následující příklad kombinuje některé z těchto příznaků a nastavuje zařízení `ttyS0` pro práci rychlostí 19 200 b/s, s osmi datovými bity, bez parity, s hardwarovým handshakingem a bez lokálního echa:

```
$ stty 19200 cs8 -parenb crtscts -echo -F /dev/ttyS0
```

Sériová zařízení a výzva login:

Bývalo poměrně obvyklé, že Unix byl nainstalován s jedním serverem a řadou „hloupých“ znakových terminálů nebo modemů. Dnes je tento typ instalace vzácnější, což je dobrá zpráva pro všechny, kteří si jej chtějí vyzkoušet, protože „hloupé“ terminály jsou dnes velice levné. Modemové instalace sice vzácnější nejsou, ale obvykle dnes používají přihlášení protokolem SLIP nebo PPP (o kterých budeme hovořit v kapitolách 7 a 8). Nicméně všechny tyto konfigurace mohou využívat jednoduchý program nazvaný **getty**.

Termín **getty** je zřejmě zkratka z „get tty“. Program **getty** otevře sériové zařízení, správně je nakonfiguruje, případně nakonfiguruje modem, a potom čeká na spojení. Aktivní spojení na sériové lince je typicky indikováno signálem DCD (Data Carrier Detect). Jakmile je detekováno spojení, program **getty** vypíše výzvu `login:` a pak spustí program **login**, který zajistí samotné přihlášení. Na každém z virtuálních terminálů (například `/dev/tty1`) v Linuxu tento program běží.

Existuje řada různých implementací programu **getty**, každá z nich obsluhuje některé konfigurace lépe než jiná. Program **getty**, o kterém budeme hovořit my, se jmenuje **mgetty**. Je velice oblíbený, protože obsahuje všechny funkce, pro něž je velmi přívětivý k modemům včetně podpory automatických faxových programů a hlasových modemů. Zaměříme se na konfiguraci programu **mgetty** tak, aby reagoval na klasické datové volání a zbytek už necháme na vás.

Konfigurace démona mgetty

Zdrojový kód démona **mgetty** je k dispozici na adrese `ftp://alpha.greenie.net/pub/mgetty/source/` a bývá součástí prakticky všech distribucí Linuxu. Démon **mgetty** se od ostatních démonů **getty** liší zejména v tom, že je navržen specificky pro podporu modemů kompatibilních se standardem Hayes. Podporuje i přímá terminálová připojení, lépe se však hodí pro telefonické aplikace. K detekci příchozího volání nepoužívá signál DCD, ale čeká na zprávu `RING`, kterou moderní modemy generují, když přijde hovor a nejsou nakonfigurovány na automatickou odpověď.

Hlavní spustitelný program se jmenuje `/usr/sbin/mgetty` a odpovídající hlavní konfigurační soubor je `/etc/mgetty/mgetty.config`. Kromě toho existuje ještě řada dalších binárních a konfiguračních souborů, které pokrývají všechny funkce programu **mgetty**.

U většiny instalací se celá konfigurace programu redukuje na úpravu souboru `/etc/mgetty/mgetty.config` a přidání příslušné položky do souboru `/etc/inittab` tak, aby se program **mgetty** spouštěl automaticky.

Příklad 4.6 znázorňuje velmi jednoduchý konfigurační soubor programu **mgetty**. V tomto příkladu konfigurujeme dvě sériová zařízení. První z nich, `/dev/ttyS0`, podporuje modem kompatibilní s Hayes a rychlostí 38 400 b/s. Druhé, `/dev/ttyS1`, podporuje přímo připojený terminál VT100 s rychlostí 19 200 b/s.

Příklad 4.6 – Příklad konfiguračního souboru /etc/mgetty/mgetty.config

```
#
# konfigurační soubor programu mgetty
#
# toto je příklad konfigurace, podrobnosti viz mgetty.info
#
# komentáře začínají "#", prázdné řádky se ignorují
#
# ----- globální sekce -----
#
```

```
# V této sekci nastavujeme globální vlastnosti, nastavení jednotlivých
# portů budou následovat
#
# přístup k modemům rychlostí 38 400 b/s
speed 38400
#
# nastavení ladicí úrovně na "4" (implicitní dle policy.h)
debug 4
#

# ----- části pro jednotlivé porty -----
#
# Tady se uvádějí věci platné jen pro jednu linku, ne pro všechny
#
#
# Modem Hayes připojený na ttyS0: bez faxu, menší logování
#
port ttyS0
  debug 3
  data-only y
#
# přímé spojení s terminálem VT100, který nepoužívá DTR
#
port ttyS1
  direct y
  speed 19200
  toggle-dtr n
#
```

Konfigurační soubor obsahuje globální část a část specifickou pro jednotlivé porty. V našem příkladu nastavujeme v globální části rychlost na 38 400 b/s. Toto nastavení zdědí port ttyS0. Všechny porty obsluhované programem **mgetty** budou tuto rychlost používat, ledaže by v části konkrétního portu byla nastavena jiná rychlost tak, jak jsme to udělali pro zařízení ttyS1.

Klíčové slovo debug řídí „ukecanost" logovacích záznamů programu **mgetty**. Klíčové slovo data-only nařizuje programu **mgetty** ignorovat všechny faxmodemové funkce a pracovat se zařízením jako s čistým modemem. Klíčové slovo direct v konfiguraci portu ttyS1 programu **mgetty** říká, aby se na tomto portu nepokoušel inicializovat modem. Konečně klíčové slovo toggle-dtr programu nařizuje, aby neukončoval spojení pomocí signálu DTR (Data Terminal Ready) sériového rozhraní, protože některé terminály to nemají rády.

Můžete se také rozhodnout nechat soubor `mgetty.config` prázdný a stejné parametry specifikovat pomocí řádkových voleb příkazu. Dokumentace dodávaná s programem obsahuje úplný popis parametrů konfiguračního souboru i řádkových parametrů. Viz následující příklad.

Abychom konfiguraci aktivovali, musíme do souboru `/etc/inittab` přidat dva záznamy. Soubor inittab je konfiguračním souborem příkazu **init** na Unixu System V. Příkaz **init** provádí inicializaci systému, představuje způsob, jak zajistit automatické spuštění příkazů při spouštění systému i jak je spustit znovu poté, co budou ukončeny. To je ideální řešení pro programy jako je **getty**.

```
T0:23:respawn:/sbin/mgetty ttyS0
T1:23:respawn:/sbin/mgetty ttyS1
```

Každý řádek souboru `/etc/inittab` obsahuje čtyři údaje oddělené dvojtečkou. První údaj je identifikátor, který jednoznačně definuje každý ze záznamů v tomto souboru. Klasicky to bývaly dva

znaky, novější verze umožňují čtyři znaky. Druhá položka představuje seznam běhových úrovní, na nichž má být program spouštěn. Běhové úrovně představují metodu jak nabídnout různé konfigurace stejného počítače a implementují se pomocí stromů spouštěcích skriptů umístěných v adresářích `/etc/rc1.d`, `/etc/rc2.d` a podobně. Tato funkce bývá typicky implementována velmi jednoduše a vlastní záznamy buď vytvářejte podle ostatních, nebo prostudujte dokumentaci k systému, kde se dozvíte více. Třetí údaj říká, kdy se má akce provést. Pro potřeby programů jako je `getty` používáme hodnotu `respawn`, což znamená, že program má být znovu spuštěn v okamžiku, kdy dokončí svou činnost. Lze zde použít i různé další volby, ale pro naše potřeby nám nyní žádná jiná nevyhovuje. Čtvrtým údajem je samotný spouštěný příkaz, což je tedy `mgetty` s případnými parametry, které mu předáváme. V našem jednoduchém případě spouštíme (a znovu spouštíme) program `mgetty` vždy na běhových úrovních 2 nebo 3 a jako parametr uvádíme pouze jméno zařízení, které chceme používat. Příkaz `mgetty` automaticky předpokládá cestu `/dev/`, takže ji nemusíme uvádět.

V tomto textu jsme se snažili o stručné představení programu **mgetty** a toho, jak zajistit přihlašovací výzvu u sériových zařízení. Podstatně podrobnější informace naleznete v dokumentu Serial -HOWTO.

Po provedení úprav v konfiguračních souborech musíte znovu spustit **Proces init**, aby se změny projevily. Stačí poslat **Procesu init** signál HANGUP. Tento proces má vždy identifikátor 1, takže můžete jednoduše použít následující příkaz:

```
# kill -HUP 1
```

KAPITOLA 5

Konfigurace sítí TCP/IP

V této kapitole si projdeme kroky, které jsou nezbytné pro konfiguraci sítí na bázi protokolu TCP/IP. Začneme s přidělením IP adresy, pak si projdeme nastavení rozhraní TCP/IP a představíme si několik užitečných nástrojů pro hledání problémů ve vaší síťové instalaci.

Většinu úkolů probraných v této kapitole budete muset obvykle provést pouze jedenkrát. Pozdější změny v konfiguračních souborech budou nutné jen v případech, kdy budete chtít do sítě přidat nový systém nebo budete-li chtít systém kompletně překonfigurovat. Nicméně některé z příkazů pro konfiguraci protokolu TCP/IP musí být spouštěny pokaždé při zavádění systému. To se zpravidla provádí jejich voláním ze systémových skriptů v adresáři `/etc/rc`.

Síťová část zaváděcí procedury je typicky obsažena v nějakém skriptu. Názvy tohoto skriptu se liší na různých distribucích Linuxu. Ve většině starších distribucí se používají skripty `rc.net` nebo `rc.inet`. Někdy se také můžete setkat se dvěma skripty s názvy `rc.inet1` a `rc.inet2`. První inicializuje síťovou část jádra systému, zatímco druhý spouští základní síťové služby a aplikace. V moderních distribucích jsou soubory `rc` strukturovány mnohem promyšlenějším způsobem, v adresáři `/etc/init.d/` (nebo `/etc/rc.d/init.d/`) najdete skripty, které vytvářejí síťová rozhraní a další soubory `rc`, jež spouštějí síťové aplikační programy. Příklady v této knize vycházejí z tohoto posledního řešení.

V této kapitole budeme hovořit o skriptech konfigurujících síťová rozhraní, o aplikacích budeme hovořit v dalších kapitolách. Po přečtení této kapitoly byste měli dokázat vytvořit posloupnost příkazů, která na vašem počítači správně nakonfiguruje síť s protokolem TCP/IP. Pak byste měli nahradit všechny vzorové příkazy ve vašich konfiguračních skriptech svými vlastními příkazy, zkontrolovat, že je skript při startu systému spouštěn základním `rc` skriptem, a restartovat počítač. Síťové `rc` skripty dodávané společně s vaší oblíbenou verzí Linuxu vám mohou posloužit jako dobré příklady.

Připojování souborového systému /proc

Některé z konfiguračních nástrojů ve verzích NET-2 a NET-3 Linuxu spoléhají při komunikaci s jádrem na souborový systém `/proc`. Toto rozhraní umožňuje přístup k aktuálním informacím jádra za pomoci mechanismu, jenž je podobný souborovému systému. Když je tento systém připojen, můžete stejně jako u ostatních souborových systémů vypisovat jeho soubory nebo zobrazovat jejich obsah. Typickými položkami jsou soubory `loadavg`, které obsahují průměrné zatížení systému nebo soubor `meminfo`, který zobrazuje aktuální obsazení fyzické paměti a využití odkládacích souborů.

Síťový kód k tomu přidává adresář `net`. Obsahuje množství souborů s informacemi, jako například tabulky ARP jádra systému, stav TCP spojení a směrovací tabulky. Většina síťových administrativních nástrojů čerpá informace z těchto souborů.

Souborový systém `proc` (někdy se používá označení `procfs`) je obvykle připojen při zavádění systému jako adresář `/proc`. Nejlépe toho dosáhnete přidáním následujícího řádku do souboru `/etc/fstab`:

```
# připojovací bod procfs
none /proc proc defaults
```

Potom ze svého skriptu `/etc/rc` spusťte příkaz **mount / proc**.

V současné době je souborový systém procfs implicitně začleněn do většiny jader operačního systému. Není-li souborový systém procfs ve vašem jádru, objeví se zpráva jako mount: fs type procfs not supported by kernel. V tom případě budete muset přeložit jádro systému a na dotaz, zda si přejete nainstalovat podporu souborového systému procfs, odpovědět „yes“.

Instalace binárních souborů

Používáte-li jednu z předkompilovaných distribucí Linuxu, bude velmi pravděpodobně obsahovat hlavní síťové aplikace a utility, a také kompletní sadu vzorových souborů. Jediným případem, kdy budete chtít získat a nainstalovat nové utility, bude instalace nové verze jádra operačního systému. Protože to občas vede ke změnám v síťové vrstvě jádra, budete muset aktualizovat i základní konfigurační nástroje. To znamená přinejmenším rekompilaci binárních souborů, ale někdy budete potřebovat i nejnovější sady binárních souborů. Tyto soubory jsou dostupné na oficiální adrese *ftp.inka.de/pub/com/Linux/networking/NetTools/*, kde jsou sbaleny v archivu `net-tools-XXX.tar.gz`, kde `XXX` označuje číslo verze. Linuxu 2.0 odpovídá balík `net-tools-1.45`.

Pokud si chcete zkompilovat a nainstalovat standardní síťové aplikace na bázi protokolu TCP/IP sami, získáte jejich zdrojový kód na většině linuxových FTP serverů. Většina moderních distribucí Linuxu obsahuje poměrně kompletní balík síťových aplikací TCP/IP, například webový prohlížeč, programy pro **telnet** a **ftp** a další síťové programy, jako je například **talk**. Narazíte-li na něco, co musíte přeložit sami, je dobrá šance, že se vám to bez potíží podaří; obvykle stačí dodržet instrukce uvedené ve zdrojovém balíku.

Nastavení jména hostitele

Většina, ne-li všechny, aplikací spoléhá na to, že je název místního hostitele nastaven na nějakou rozumnou hodnotu. To se obvykle provede při zavádění systému pomocí příkazu **hostname**. Chcete-li nastavit název hostitele na *jméno*, zadejte:

```
# hostname jméno
```

U tohoto příkazu je běžné používat nekvalifikované názvy hostitele bez jakéhokoliv názvu domény. Například hostitelé v pivovaru Virtual Brewery (který je popsán v příloze A) se mohou jmenovat `vale.vbrew.com` nebo `vlager.vbrew.com`. Toto jsou jejich oficiální, *plně kvalifikovaná doménová jména* (FQDN). Názvy místních hostitelů budou tvořit pouze první část z plně kvalifikovaného jména, například **vale**. Protože je však název lokálního hostitele často používán pro vyhledání IP adresy hostitele, musíte zajistit, aby knihovna resolveru byla schopná nalézt IP adresu tohoto hostitele. To obvykle znamená, že musíte název hostitele specifikovat v souboru `/etc/hosts`.

Někteří lidé doporučují používat příkaz `domainname`, kterým se jádro operačního systému dozví o doméně ve zbývající části FQDN. U tohoto způsobu můžete zkombinovat hodnoty `hostname` a `domainname` a získat tak plně kvalifikované jméno. Tento způsob je však v pořádku maximálně z poloviny. Příkaz `domainname` se všeobecně používá k nastavení NIS domény hostitele, která může být zcela jiná než DNS doména, ke které patří váš hostitel. Pokud chcete zajistit, aby byla krátká podoba jména vašeho hostitele rozlišitelná všemi staršími verzemi příkazu `hostname`, přidejte je namísto toho jako záznam do místního DNS serveru, né jméno. Pak můžete použít parametr `--fqdn` příkazu `hostname` a získat tak plně kvalifikované doménové jméno.

Přiřazení IP adresy

Chcete-li nakonfigurovat síťový software na hostiteli, který není určen pro práci v síti (aby na něm mohl být například spouštěn software síťových konferencí INN), můžete tuto stať s klidným svědomím přeskočit, protože k tomu budete potřebovat pouze IP adresu svého lokálního rozhraní, a ta je vždy 127.0.0.1.

Mírně složitější je tento problém v reálných sítích typu Ethernet. Chcete-li připojit svého hostitele k existující síti, musíte požádat jejího správce, aby vám v této síti přidělil IP adresu. Nastavujete-li celou síť sami, musíte si přidělit IP adresy sami.

Hostitelé, kteří patří do místní sítě, by měli obvykle sdílet adresy ze stejné logické IP sítě. Z toho důvodu musíte přidělit síti síťovou IP adresu. Máte-li několik fyzických sítí, musíte jim buď přidělit odlišné síťové adresy, anebo musíte použít podsítě, čímž rozdělíte svůj rozsah IP adres do několika podsítí. O podsítích budeme hovořit v další části, *Vytváření podsítí*.

Při výběru IP adres sítí záleží na tom, zda v blízké době plánujete připojení k Internetu. Pokud ano, měli byste *hned* požádat o přidělení oficiální IP adresy. Požádejte o pomoc svého poskytovatele síťových služeb. Pokud chcete získat oficiální síťovou adresu čistě pro případ, že byste se někdy k Internetu připojovali, vyžádejte si na adrese *hostmaster@internic.net* žádost o síťovou adresu, nebo se obraťte na národní NIC, pokud u vás existuje[35].

35 Pozn. překladatele: V České republice zajišťuje přidělování IP adres jednak řada takzvaných LIR (Local Internet Registry) – společností, které mají delegován nějaký rozsah adres a rozdělují je dále. Nadřazenou autoritou je RIR (Regional Internet Registry), kterým je pro celou Evropu organizace RIPE NCC (*www.ripe.net*). Ti se s vámi ovšem bavit nebudou a podstatně jednodušší je domluvit se se svým poskytovatelem internetového připojení, který obvykle bez zbytečného otálení poskytne určitý rozsah IP adres za jednorázový poplatek (a pokud to nedělá, neprodleně přejděte k jinému poskytovateli).

Není-li vaše síť připojena k Internetu a v dohledné době její připojení neplánujete, můžete si vybrat libovolnou platnou síťovou adresu. Musíte ovšem zajistit, aby žádné pakety z vaší interní sítě „neutekly" na Internet. Abyste zaručili, že nedojde k žádné škodě dokonce i když nějaký paket uteče, měli byste použít některou ze síťových adres, rezervovaných pro privátní sítě. Agentura IANA (Internet Assigned Number Agency) rezervovala několik sítí tříd A, B i C, které můžete použít i bez registrace. Tyto adresy platí pouze ve vaší privátní síti a mezi internetovými systémy nedochází k jejich směrování. Rozsahy těchto adres jsou definovány v dokumentu RFC 1597 a jsou uvedeny v tabulce 2.1 v kapitole 2. Všimněte si, že druhý a třetí blok obsahují 16, respektive 256 sítí.

Volba adres z těchto rozsahů není výhodná jen pro zcela nepřipojené sítě, pomocí jednoho počítače používaného jako brána budete pořád schopni zajistit do jisté míry omezený přístup k Internetu. Ve vaší interní síti bude tato brána známa pod její interní adresou, zatímco okolní svět ji uvidí pod její oficiální adresou (kterou vám přidělí poskytovatel). K této problematice se vrátíme v souvislosti s IP maškarádou v kapitole 11.

Ve zbytku knihy budeme předpokládat, že správce sítě v pivovaru používá adresu třídy B, řekněme 172.16.0.0. Samozřejmě potřebám pivovaru i vinařství by postačovala adresa třídy C, nicméně třídu B používáme kvůli jednoduchosti. Příklady rozdělení na podsítě v dalším textu budou díky tomu o něco názornější.

Vytváření podsítí

Abyste mohli provozovat více ethernetových (nebo jiných sítí), musíte svou síť rozdělit na podsítě. Rozdělení na podsítě je nutné pouze v případě, že máte více *vysílaných sítí*, spojení uzel-uzel se nepočítají. Pokud máte například jeden Ethernet a jednu nebo více SLIP linek do světa, nepotřebujete podsítě. Podrobněji je to vysvětleno v kapitole 7.

Kvůli správě dvou Ethernetů se správce pivovaru rozhodl použít 8 bitů hostitelské části adresy jako podsíťovou adresu. Tím zůstává dalších 8 bitů volných pro adresu hostitele, což nám umožní připojit na každou podsíť 254 počítačů. Pak přiřadil podsíť 1 pivovaru a podsíť 2 vinařství. Tomu budou odpovídat síťové adresy 172.16.1.0 a 172.16.2.0. Maska podsítě bude 255.255.255.0.

Počítač **vlager**, což je brána mezi oběma sítěmi, má na obou sítích přidělenu adresu 1, takže jeho IP adresy na jednotlivých sítích budou 172.16.1.1 a 172.16.2.1.

Všimněte si, že v našem příkladu používáme kvůli názornosti adresu třídy B, realističtější přístup by byla adresa třídy C. S novým síťovým kódem není rozdělení na podsítě omezeno na hranice bajtů, takže i síť třídy C můžeme rozdělit na několik podsítí. Můžeme například použít dva bity hostitelské části pro masku sítě, čímž dostáváme čtyři podsítě s 64 počítači na každé z nich[36],[37].

36 První číslo na každé podsíti je adresa samotné podsítě a poslední číslo je rezervováno jako vysílací adresa, takže budeme mít pouze 62 možných počítačů na každé podsíti.

37 Pozn. překladatele: Předpokládejme například, že budeme mít síť třídy C s adresou 192.168.10.0. Použijeme-li síťovou masku 255.255.255.192 (ono 192 dekadicky je 11 000 000 binárně), dostáváme dva bity pro adresaci podsítě a 6 bitů pro adresaci počítačů na podsítích. Pak máme čtyři podsítě: 192.168.10.0, 192.168.10.64, 192.168.10.128 a 192.168.10.192 (do dvojkové soustavy už si to převádějte sami – je to práce, ale jen tak to lze pochopit). Pro podsítě platí omezení známé u hostitelských adres (tedy že nelze použít „samé nulové" a „samé jedničkové" adresy), takže počítače na první (respektive nulté) podsíti budeme číslovat 192.168.10.1, 192.168.10.2... a na druhé (respektive první) pak 192.168.10.65, 162.168.10.66...

Pouze se musíte ujistit, že jste vybrali jednu ze tříd A, B nebo C, protože jinak by pravděpodobně vše nefungovalo zcela správně. Plánujete-li v blízké době připojení k Internetu, měli byste si už *nyní* obstarat oficiální IP adresu. Nejlépe je požádat o pomoc svého poskytovatele síťových služeb. Pokud si chcete obstarat číslo sítě jenom proto, že se někdy v budoucnosti budete chtít připojit na Internet, vyžádejte si na adrese **hostmaster@internic.net** formulář Network Address Application Form.

Budete-li spravovat několik ethernetových sítí (nebo jiných sítí, jakmile pro ně budou dostupné ovladače), musíte rozdělit vaši síť na podsítě. Všimněte si, že vytváření podsítí je požadováno pouze v případě, že vlastníte více než jednu *vysílací síť* (*broadcast network*); nepočítaje v to spojení pomocí protokolu PPP. Máte-li například jeden Ethernet a jedno nebo více spojení s vnějším světem pomocí protokolu SLIP, nepotřebujete ve své síti vytvářet podsítě. Důvod bude objasněn v kapitole 7.

Síťový správce pivovaru požádal například centrum NIC o přidělení čísla sítě třídy B, obdržel adresu **191.72.0.0**. Aby si správce přizpůsobil své dva Ethernety k obrazu svému, rozhodl se použít osm bitů z části hostitele jako doplňující bity podsítě. Tato úprava poskytla části hostitele dalších osm bitů, což dovoluje až 254 hostitelů v každé podsíti. Dále správce přidělil pivovaru číslo podsítě 1 a vinárně číslo podsítě 2. Tedy jejich příslušné síťové adresy jsou **191.72.1.0** a **191.72.2.0**. Maska podsítě je **255.255.255.0**.

Bráně **vlager**, což je brána mezi dvěma sítěmi, bylo u obou podsítí přiděleno číslo hostitele 1, takže její IP adresy jsou **191.72.1.1**, resp. **191.72.2.1**.

Vytvoření souborů hosts a networks

Po vytvoření podsítí byste měli za pomoci souboru `/etc/hosts` nastavit jednoduchou variantu rozlišení názvu hostitele. Pokud nehodláte k rozlišení adres používat DNS nebo NIS, musíte do souboru hosts vložit všechny hostitele.

Dokonce i v případě, kdy budete za normálních okolností používat pro rozlišení názvů služby DNS nebo NIS, budete muset v souboru `/etc/hosts` specifikovat přinejmenším určitou vybranou skupinu hlavních hostitelů. Nějaký druh rozlišení názvů hostitelů budete totiž potřebovat i tehdy, nepoběží-li žádné síťové rozhraní – například při zavádění operačního systému. Není to jen otázka pohodlí, ale umožní vám to také používat v `rc` skriptech symbolické názvy hostitelů. Takže při změně IP adres vám postačí pouze zkopírovat aktualizovaný soubor `hosts` na všechny počítače a nemusíte se zatěžovat samostatnou editací velkého počtu `rc` souborů. Do souboru `hosts` obvykle přidáte všechny názvy a adresy lokálního hostitele, adresy všech používaných bran a eventuálních serverů NIS[38].

V průběhu úvodního testování byste měli také zajistit, že váš resolver používá pouze informace ze souboru `hosts`. Vzorové soubory služeb DNS a NIS mohou při jejich neúmyslném použití vést k velmi zajímavým efektům. Chcete-li, aby všechny aplikace při vyhledání IP adres hostitelů používaly výhradně soubor `/etc/hosts`, musíte upravit soubor `/etc/host.conf`. Všechny řádky,

38 Adresy serverů NIC jsou zapotřebí pouze pokud používáte NIS Petera Erikssona. Jiné implementace služby NIS si své servery nacházejí za běhu prostřednictvím programu **ypbind**.

které začínají klíčovým slovem *order*, označte jako komentáře. Provedete to tak, že na první pozici příslušného řádku vložíte znak # a nakonec vložíte řádek:

```
order hosts
```

Konfigurace knihovny resolveru bude podrobně rozebrána v kapitole 6.

Soubor hosts obsahuje na každém řádku jednu položku, která se skládá z IP adresy, jména hostitele a z nepovinného seznamu aliasů hostitele. Jednotlivá pole jsou vzájemně oddělena mezerami nebo tabulátory a pole s adresou musí začínat v prvním sloupci. Cokoliv, co následuje za znakem # na stejném řádku, je považováno za komentář a ignoruje se.

Názvy hostitelů mohou být buď plně kvalifikované, nebo relativní vzhledem k místní doméně. U serveru **vale** byste typicky zadali jak plně kvalifikované jméno **vale.vbrew.com**, tak i relativní jméno **vale**. V souboru *hosts* tak bude server znám pod oficiálním i pod zkráceným jménem.

Následující příklad ilustruje, jak by mohl vypadat soubor hosts ve společnosti Virtual Brewery. Jsou v něm obsaženy dva speciální názvy, **vlager-if1** a **vlager-if2**, které definují adresy obou rozhraní používaných branou **vlager**.

```
#
# Soubor hosts pro Virtual Brewery/Virtual Winery
#
# IP            FQDN                 aliasy
#
127.0.0.1       localhost
#
172.16.1.1      vlager.vbrew.com      vlager vlager-if1
172.16.1.2      vstout.vbrew.com      vstout
172.16.1.3      vale.vbrew.com        vale
#
172.16.2.1      vlager-if2
172.16.2.2      vbeaujolais.vbrew.com vbeaujolais
172.16.2.3      vbardolino.vbrew.com  vbardolino
172.16.2.4      vchianti.vbrew.com    vchianti
```

Stejně jako u IP adres hostitelů budete také někdy chtít použít symbolický název pro síťové adresy. Proto má soubor `hosts` kamaráda nazvaného `/etc/networks`, který mapuje názvy sítí na jejich adresy a opačně. Ve společnosti Virtual Brewery můžeme nainstalovat následující soubor networks[39]:

```
# Soubor /etc/networks pro Virtual Brewery
brew-net      172.16.1.0
wine-net      172.16.2.0
```

39 Pozor na to, že názvy sítí souboru `networks` nesmějí kolidovat s názvy počítačů v souboru `hosts`, jinak by se některé programy mohly chovat nedefinovaným způsobem.

Konfigurace rozhraní pro protokol IP

Po nastavení hardwaru, které jsme probírali v kapitole 4, musíte o těchto zařízeních říci síťovému softwaru jádra. K nastavení síťových rozhraní a inicializaci směrovací tabulky se používá několik příkazů. Tyto úkoly se obvykle provádějí při každém startu počítače z inicializačního skriptu sítě. Základní konfigurační nástroje se nazývají **ifconfig** (kde „if" znamená rozhraní – interface) a **route**.

Příkaz ifconfig se používá pro zpřístupnění rozhraní síťové vrstvě jádra. Tento proces v sobě zahrnuje přidělení IP adresy a dalších parametrů a aktivaci rozhraní. Výraz „aktivní" znamená, že jádro pomocí takového rozhraní může vysílat a přijímat IP datagramy. Nejjednodušší způsob aktivace rozhraní je následující:

```
ifconfig rozhraní ip_adresa
```

Výše uvedený příkaz přidělí rozhraní IP adresu a aktivuje ho. Všechny ostatní parametry jsou nastaveny na své implicitní hodnoty. Například implicitní maska podsítě je odvozena z IP adresy podle třídy sítě, například adrese třídy B odpovídá maska 255.255.0.0. Příkaz **ifconfig** je detailně popsán později v části *Vše o příkazu ifconfig*.

Příkaz **route** umožňuje přidávat nebo odstraňovat trasy ze směrovací tabulky jádra systému. Může být volán následujícím způsobem:

```
route [add|del] [-net|-host] cíl [rozhraní]
```

Parametry add a del určují, zda se má trasa pro zadaný cíl přidat nebo odebrat. Parametry –net a –host říkají, zda je cílem trasy síť nebo jednotlivý počítač (pokud nic neuvedete, předpokládá se, že je to počítač). Konečně nepovinný parametr rozhraní říká, přes které rozhraní má být trasa směrována. Pokud parametr neuvedete, jádro Linuxu provede obvykle správný odhad. O tomto tématu budeme podrobněji hovořit v následující části.

Lokální rozhraní

Jedním z prvních aktivovaných rozhraní je lokální zpětnovazebné rozhraní:

```
# ifconfig lo 127.0.0.1
```

Občas zjistíte, že se místo IP adresy používá fiktivní název hostitele **localhost**. Příkaz **ifconfig** vyhledá příslušné jméno v souboru hosts, kde by mu měla být přidělena adresa 127.0.0.1:

```
# Příklad záznamu pro localhost v /etc/hosts
localhost      127.0.0.1
```

Chcete-li si prohlédnout konfiguraci nějakého rozhraní, spusťte příkaz **ifconfig** a jako parametr zadejte pouze název tohoto rozhraní:

```
$ ifconfig lo
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          Collisions:0
```

Jak vidíme, lokálnímu rozhraní byla přidělena síťová maska 255.0.0.0, protože adresa 127.0.0.1 je adresou třídy A.

Nyní můžete začít se svou miniaturní sítí experimentovat. Zatím však ve směrovací tabulce stále chybí položka, která řekne protokolu IP, že může toto rozhraní používat jako trasu k cílové adrese 127.0.0.1. Tu přidáte následujícím příkazem:

```
# route add 127.0.0.1
```

Opět můžete použít namísto IP adresy název hostitele **localhost** za předpokladu, že jej máte definován v souboru `/etc/hosts`.

Dále byste měli zkontrolovat, že vše správně funguje, například pomocí použití příkazu **ping**. Příkaz **ping** je síťovým ekvivalentem sonaru[40] a je používán k ověření dostupnosti příslušné adresy a k měření prodlev, které se vyskytují při posílání datagramu tam a zpět. Čas požadovaný na tuto operaci je často uváděn pod označením „round-trip time“.

```
# ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=32 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=32 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=32 time=0.4 ms
^C--- localhost ping statistics ---
3 packets transmitter, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.4 ms
```

Spustíte-li příkaz **ping** výše uvedeným způsobem, bude se tento příkaz snažit posílat pakety tak dlouho, dokud ho uživatel nepřeruší. Symbol ^C označuje místo, kde byla stisknuta kombinace kláves Ctrl+C.

Výše uvedený příklad ukazuje, že pakety pro adresu 127.0.0.1 jsou doručovány správně a odpovědi se vracejí příkazu **ping** zpět téměř okamžitě. To naznačuje, že jste při svém prvním nastavení síťového rozhraní uspěli.

Pokud se výstup příkazu **ping** nepodobá výše uvedenému výpisu, pak je tu problém. Zkontrolujte případná chybová hlášení, zda nenaznačují, že některý ze souborů nebyl korektně nainstalován. Zkontrolujte, zda jsou binární soubory příkazů **ifconfig** a **route** kompatibilní s vámi používanou verzí jádra operačního systému a hlavně se podívejte, zda bylo jádro zkompilováno s povolením síťových služeb (to poznáte podle přítomnosti adresáře `/proc/net`). Pokud obdržíte chybovou zprávu „Network unreachable“, pak bude zřejmě chyba v nastavení příkazu **route**. Ujistěte se, že používáte stejnou adresu, kterou jste předali příkazu **ifconfig**.

Výše popsané kroky stačí k tomu, abyste mohli na samostatném hostiteli používat síťové aplikace. Jakmile přidáte do síťového inicializačního skriptu uvedené řádky a ujistíte se, že se skript při startu systému skutečně spouští, můžete počítač restartovat a vyzkoušet různé aplikace. Například příkaz **telnet localhost** by měl s vaším hostitelem navázat telnetové spojení a nabídnout vám přihlašovací výzvu.

Lokální rozhraní je ale užitečné nejen jako příklad vhodný pro knihy o sítích nebo jako testovací prostředek při vývoji. Ve skutečnosti ho používají některé aplikace v průběhu normálních operací[41]. Proto ho musíte nakonfigurovat vždy, bez ohledu na to, zda váš počítač je či není připojen k síti.

---

40 Vzpomínáte si na „Echoes“ od Pink Floydů?

41 Například aplikace založené na RPC používají rozhraní při startu k registraci sama sebe u démona **portmapper**. Mezi tyto aplikace patří například NIS a NFS.

Ethernetová rozhraní

Konfigurace ethernetového rozhraní má s nastavením lokálního rozhraní mnoho společného, pouze ve spojitosti s podsítěmi vyžaduje několik dalších parametrů.
Ve společnosti Virtual Brewery jsme rozdělili síť IP, která byla původně sítí třídy B, na podsítě které jsou třídy C. Aby se v této změně vaše rozhraní vyznalo, bude příkaz **ifconfig** vypadat následovně:

```
# ifconfig eth0 vstout netmask 255.255.255.0
```

Tento příkaz přiřadí rozhraní eth0 IP adresu počítače **vstout** (172.16.1.2)[42]. Pokud bychom vynechali síťovou masku, pak by si ji příkaz **ifconfig** odvodil z třídy sítě, ze které vyplyne síťová maska 255.255.0.0. Rychlé ověření vypíše následující text:

```
# ifconfig eth0
eth0      Link encap 10Mps Ethernet HWaddr  00:00:C0:90:B3:42
          inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING  MTU 1500  Metric 1
          RX packets 0 errors 0 dropped 0 overrun 0
          TX packets 0 errors 0 dropped 0 overrun 0
```

Můžete si všimnout, že příkaz **ifconfig** automaticky nastavil vysílací adresu (výše uvedené pole Bcast) na obvyklou hodnotu, což je číslo sítě se všemi nastavenými bity v části hostitele. Také velikost přenosové jednotky (maximální velikost IP datagramu, kterou bude generovat jádro pro toto rozhraní) byla nastavena na maximální délku ethernetového paketu – na 1 500 bajtů. Obvykle můžete tyto standardní hodnoty použít, nicméně všechny mohou být změněny speciálními parametry, které jsou popsány v části *Vše o příkazu ifconfig*.

Všechny tyto hodnoty mohou být změněny pomocí speciálních voleb, jež budou popsány dále.

Stejně jako tomu bylo u lokálního rozhraní, musíte nyní nainstalovat směrovací data, která budou jádro informovat o síti, jež je dosažitelná prostřednictvím rozhraní eth0. Pro firmu Virtual Brewery byste volali příkaz **route** následujícím způsobem:

```
# route add -net 191.72.1.0
```

Na první pohled to vypadá trochu magicky, protože není zcela zřejmé, jak příkaz **route** zjistí, přes které rozhraní má směrovat. Náš trik je ale celkem jednoduchý: jádro operačního systému si ověří všechna rozhraní, která již byla nakonfigurována a porovná cílovou adresu (v tomto případě `191.72.1.0`) se síťovou částí adresy rozhraní (to znamená, že bude po bitech porovnávat hodnotu získanou z adresy rozhraní a síťové masky). Jediné vyhovující rozhraní bude eth0.

K čemu tedy slouží volba –net? Používá se z toho důvodu, že příkaz **route** umí obsluhovat jak směrování do sítí, tak i směrování k jednotlivým hostitelům (jak jsme si ukázali u příkladu s lokálním rozhraním a jeho adresou). Je-li mu předána adresa v tečkové notaci, pokusí se příkaz **route** odhadnout, zda se jedná o adresu sítě nebo hostitele tak, že se podívá na hostitelskou část adresy. Je-li hostitelská část adresy rovna nule, bude příkaz **route** předpokládat, že se jedná o síť, v opačném případě ji bude považovat za adresu hostitele. Teď by si ale příkaz **route** myslel, že je adresa `191.72.1.0` adresou hostitele a nikoliv adresou sítě, protože nemůže nic tušit o námi vytvořených podsítích. Tuto informaci mu musíme sdělit explicitně pomocí příznaku `-net`.

Samozřejmě, že vypisování výše uvedeného příkazu **route** je únavné a člověk při něm může udělat chyby. Mnohem pohodlnější je použít názvy sítí, které jsme již nadefinovali v souboru `/etc/networks`. Tento způsob výrazně zlepší čitelnost příkazu **route** a dokonce můžeme vynechat i parametr `-net`, protože příkaz **route** již bude vědět, že adresa `191.72.1.0` označuje síť.

42 Pozn. překladatele.: Definovanou v souboru `/etc/hosts`.

```
# route add brew-net
```

Po skončení základních konfiguračních kroků se budeme chtít ujistit, že ethernetové rozhraní opravdu funguje správně. Vyberte si nějaký počítač na ethernetové síti, například **vlager,** a zadejte příkaz:

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 191.72.1.1: icmp_seq=0. time=11. ms
64 bytes from 191.72.1.1: icmp_seq=1. time=7. ms
64 bytes from 191.72.1.1: icmp_seq=2. time=12. ms
64 bytes from 191.72.1.1: icmp_seq=3. time=3. ms
^C
----vstout.vbrew.com PING Statistics
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms) min/avg/max = 3/8/12
```

Pokud neuvidíte výstup podobný výše uvedenému, bude zřejmě něco v nepořádku. Dochází-li k neobvykle vysokým ztrátám paketů, znamená to pravděpodobně hardwarový problém, například špatné nebo chybějící terminátory a podobně. Pokud nepřijmete vůbec žádné pakety, měli byste zkontrolovat konfiguraci rozhraní pomocí příkazu **netstat**, o kterém budeme hovořit později v části *Příkaz netstat*. Statistika paketů zobrazená příkazem **ifconfig** by vám měla říci, zda vůbec byly nějaké pakety rozhraním odeslány. Máte-li přístup i ke vzdálenému hostiteli, měli byste zkontrolovat statistiku rozhraní i na tomto počítači. Tak můžete přesně určit, kde se pakety ztrácejí. Kromě toho byste si měli pomocí příkazu **route** zobrazit směrovací informace a zkontrolovat, zda mají oba hostitelé správně nastavené směrování. Spustíte-li příkaz **route** bez parametrů, vytiskne kompletní směrovací tabulku jádra (parametr –n způsobí pouze to, že namísto jmen hostitelů budou vypisovány jejich IP adresy):

```
# route -n
Kernel routing table
Destination  Gateway  Genmask         Flags Metric Ref Use    Iface
127.0.0.1    *        255.255.255.255 UH    1      0      112 lo
172.16.1.0   *        255.255.255.0   U     1      0       10 eth0
```

Podrobný popis jednotlivých údajů naleznete dále v části *Příkaz netstat*. Sloupec Flag obsahuje seznam všech příznaků daného rozhraní. Aktivní rozhraní mají vždy nastaven příznak U (od „up“), příznak H znamená, že cílová adresa patří hostiteli a ne síti. Je-li příznak H zobrazen i u trasy, která má být trasou na síť, musíte položku tabulky upravit a spustit příkaz **route** znovu s parametrem –net. To, že se zadaná trasa vůbec používá, zjistíte na základě položky Use ve druhém sloupci zprava. S každým voláním příkazu **ping** by se tato hodnota měla zvyšovat.

Směrování pomocí bran

V předešlém textu jsme hovořili o nastavení hostitele v jediné síti Ethernet. Poměrně často se však setkáváme se sítěmi, které jsou pomocí brány připojeny k jiné síti. Tyto brány mohou jednoduše spojovat dva nebo více Ethernetů, mohou ale také zajišťovat spojení s okolním světem, například s Internetem. Abyste mohli využít služeb bran, musíte síťové vrstvě poskytnout doplňující směrovací informace.

Například Ethernety ve společnostech Virtual Brewery a Virtual Winery jsou propojeny pomocí takovéto brány, konkrétně počítačem **vlager**. Za předpokladu, že brána **vlager** je již nakonfigurována, nám zbývá pouze přidat do směrovací tabulky hostitele **vstout** další položku, která řekne

jádru operačního systému, že všichni hostitelé sítě vinařství jsou dosažitelní přes bránu **vlager**. Příslušné volání příkazu **route** je uvedeno dále, klíčové slovo gw mu sděluje, že následující parametr definuje bránu.

```
# route add wine-net gw vlager
```

Samozřejmě že každý hostitel v síti vinařství musí mít odpovídající směrovací záznam k síti pivovaru. V opačném případě byste mohli posílat data pouze z pivovaru do vinařství, počítače v síti vinařství by však nebyly schopné odpovědět.

Tento příklad popisuje pouze bránu, která přenáší pakety mezi dvěma izolovanými Ethernety. Nyní předpokládejme, že brána **vlager** je také připojena k Internetu (například pomocí samostatné ISDN linky). V tom případě budeme chtít, aby *všechny* datagramy, které nejsou určeny přímo pro síť pivovaru, byly směrovány k bráně **vlager**. To lze provést tak, že označíme bránu **vlager** jako implicitní bránu pro hostitele **vstout**:

```
# route add default gw vlager
```

Název sítě **default** je zkratkou adresy 0.0.0.0, která označuje implicitní trasu. Implicitní trasa vede ke každému cíli a bude použita vždy, pokud se nenajde jiná specifičtější trasa. Název **default** nemusíte přidávat do souboru `/etc/networks`, protože je zabudován přímo v příkazu **route**.

Když při volání příkazu **ping** na hostitele za jednou nebo více branami dochází k velké ztrátě paketů, může to znamenat přetíženou síť. Ztráta paketů není ani tak způsobena technickými nedostatky, jako spíše dočasným špičkovým zatížením předávajících hostitelů, které pak přicházející datagramy obsluhují postupně nebo je dokonce zahazují.

Konfigurace brány

Konfigurace počítače pro předávání paketů mezi dvěma Ethernety je poměrně přehledná. Budeme předpokládat, že jsme u brány **vlager**, která je vybavena dvěma ethernetovými kartami, z nichž každá je připojena k jedné ze sítí pivovaru a vinařství. Pak stačí nakonfigurovat obě rozhraní, přidělit jim odpovídající IP adresy, nastavit trasy a to je vše.

Je rozumné přidat informace o obou rozhraních do souboru `hosts`, protože tak pro ně budeme mít jednoduché názvy. Ukazuje to následující příklad:

```
172.16.1.1      vlager.vbrew.com  vlager vlager-if1
172.16.2.1      vlager-if2
```

Sekvence příkazů pro nastavení obou rozhraní je následující:

```
# ifconfig eth0 vlager-if1
# route add brew-net
# ifconfig eth1 vlager-if2
# route add wine-net
```

Rozhraní PLIP

Propojení počítačů pomocí linky PLIP je poněkud odlišné od propojení Ethernetem. Linky PLIP jsou příkladem spojení *point-to-point*, protože na každém konci spojení je pouze jeden počítač. Sítě jako Ethernet se označují jako *vysílající* sítě. Konfigurace linky point-to-point je odlišná od vysílající sítě, protože samotné point-to-point linky nepodporují vlastní síť.

Protokol PLIP nabízí levné a jednoduché propojení mezi počítači. Jako příklad si vezmeme laptop nějakého zaměstnance pivovaru, který je s branou **vlager** propojen protokolem PLIP. Tento laptop se jmenuje **vlite** a má pouze jediný paralelní port. Při zavádění systému bude tento port registrován jako rozhraní `plip1`. Abyste spojení aktivovali, musíte nakonfigurovat rozhraní `plip1` následujícími příkazy[43]:

```
# ifconfig plip1 vlite pointopoint vlager
# route add default gw vlager
```

První příkaz nastavuje rozhraní a sděluje jádru operačního systému, že se jedná o spojení typu point-to-point, u něhož má vzdálená strana přidělenu adresu **vlager**. Druhý řádek nainstaluje implicitní trasu s využitím hostitele **vlager** jako brány. Na straně brány **vlager** je nutná podobná konfigurace, která spojení zaktivuje (volání příkazu **route** není zapotřebí):

```
# ifconfig plip1 vlager pointopoint vlite
```

Zajímavé je, že rozhraní plip1 brány **vlager** nemusí mít samostatnou IP adresu, ale může mu být také přidělena adresa 172.16.1.1. Sítě typu point-to-point nepředstavují samostatné sítě, takže jejich rozhraní nevyžadují přidělení samostatné adresy. Aby se předešlo nejasnostem, využívá jádro ve směrovací tabulce informace o použitém rozhraní[44].

Nyní máme vyřešeno směrování z laptopu do sítě pivovaru; zatím nám však chybí trasa ze všech počítačů sítě pivovaru na počítač **vlite**. Jeden z poměrně nešikovných způsobů spočívá v přidání této konkrétní trasy do směrovacích tabulek všech hostitelů, přičemž budeme **vlager** definovat jako bránu k počítači **vlite**:

```
# route add vlite gw vlager
```

Lepším řešením dočasných tras je dynamické směrování. Jeden z možných způsobů spočívá v použití směrovacího démona **gated**, který musí být nainstalován na každém hostiteli v síti, a ty si pak budou dynamicky předávat směrovací informace. Nejjednodušší způsob je ale použít *ARP proxy*. Při nainstalovaném ARP proxy bude brána **vlager** odpovídat na libovolné dotazy ohledně hostitele **vlite** zasláním své vlastní ethernetové adresy. Všechny pakety pro **vlite** tak skončí na bráně **vlager**, která je pošle na laptop. K problematice ARP proxy se vrátíme v části *Kontrola tabulek ARP*.

Současné verze balíku net-tools obsahují nástroj **plipconfig**, který umožuje nastavit parametry časování PLIP rozhraní. IRQ paralelního portu lze nastavit příkazem **ifcongif**.

Rozhraní SLIP a PPP

Ačkoliv jsou spojení typu SLIP nebo PPP pouze jednoduchými spojeními typu point-to-point, je třeba k nim říct daleko více. Použití SLIP spojení totiž v sobě obvykle zahrnuje vytočení čísla vzdálené sítě prostřednictvím modemu a přepnutí sériové linky do režimu SLIP. Podobně se používá i protokol PPP. Podrobněji budeme o protokolech SLIP a PPP hovořit v kapitolách 7 a 8.

Fiktivní rozhraní

Fiktivní (*dummy*) rozhraní je trošku exotické, nicméně docela užitečné. Jeho hlavní výhoda vynikne u samostatných počítačů a u počítačů, jejichž jediné síťové spojení je připojení pomocí modemu. Koneckonců, tyto počítače představují po většinu času rovněž samostatné počítače.

---

43 Klíčové slovo **pointopoint** není překlep. Skutečně se zapisuje takto.

44 Kvůli jistotě byste měli linky PLIP nebo SLIP konfigurovat až poté, co budete mít vytvořeny kompletní směrovací tabulky pro celý Ethernet. U některých starších jader by jinak mohly později doplňované trasy skončit na lince point-to-point.

Dilema u samostatných hostitelů spočívá v tom, že mají aktivní pouze jediné síťové zařízení, a to konkrétně lokální zařízení, které má obvykle přidělenu adresu 127.0.0.1. Nicméně v určitých situacích potřebujete posílat data na „oficiální" IP adresu místního hostitele. Vezměme si například laptop **vlite**, který je momentálně odpojen od sítě. Aplikace na laptopu **vlite** může chtít poslat nějaká data jiné aplikaci na tomtéž počítači. Prohledá-li soubor `/etc/hosts/`, získá IP adresu 172.16.1.65 a na ni se pokusí poslat data. Protože jediným aktivním rozhraním je v dané chvíli pouze lokální rozhraní, nemůže jádro vědět, že adresa 172.16.1.65 je jeho vlastní adresa! Jádro tedy datagram zruší a vrátí aplikaci chybové hlášení.

V tomto bodě vstupuje do hry fiktivní rozhraní. Vyřeší dilema tím, že bude sloužit jako „alernativní ego" lokálního rozhraní. V případě laptopu **vlite** bychom mu přidělili adresu 172.16.1.65 a přidali trasu, která na tuto adresu povede. Všechny datagramy na adresu 172.16.1.65 pak budou doručeny lokálně. Správné volání vypadá takto[45]:

```
# ifconfig dummy vlite
# route add vlite
```

IP aliasy

Nová jádra podporují funkci, kterou je možné použití fiktivního rozhraní úplně vyloučit, nemluvě o některých dalších užitečných možnostech. Pomocí *IP aliasů* je možné jednomu fyzickému zařízení přidělit více IP adres. V nejjednodušším případě můžete nahradit fiktivní rozhraní tak, že lokálnímu rozhraní přidělíte jako alias i oficiální IP adresu počítače. Ve složitejších případech můžete počítač nakonfigurovat tak, že se bude chovat jako několik různých počítačů s vlastními IP adresami. Této konfiguraci se někdy říká „virtuální hostitel" i když tento termín označuje i některé jiné techniky[46].

Chcete-li nějakému rozhraní přidělit alias, musíte nejprve ověřit, zda máte jádro přeloženo s podporou IP aliasů (zkontrolujte, zda existuje soubor `/proc/net/ip_alias`; pokud ne, musíte jádro znovu přeložit). Konfigurace IP aliasu je prakticky stejná jako konfigurace skutečného síťového zařízení, pouze použijete speciální název zařízení, aby se vědělo, že jde o požadovaný alias. Například:

```
# ifconfig lo:0 172.16.1.1
```

Tímto příkazem přidělíte lokálnímu zařízení alias 172.16.1.1. Na IP aliasy se odkazuje přidáním sekvence :n k názvu zařízení, kde *n* je celé číslo. V našem příkladu jsme vytvořili nultý alias zařízení lo. Jednomu fyzickému zařízení tak můžeme přiřadit více aliasů.

Každý z aliasů můžeme chápat jako samostatné fyzické zařízení a co se týče síťového sofwaru jádra, tak se tak i chovají – nicméně tato různá rozhraní sdílejí hardware s ostatními.

```
# ifconfig dummy vlite
# route add vlite
```


45 Pokud jste fiktivní zařízení nahráli jako samostatný modul a nemáte je vestavěno přímo v jádře, bude se jmenovat `dummy0`. Je to dáno tím, že těchto modulů můžete nahrát více a budou pak fungovat jako více fiktivních zařízení.

46 V poslední době se IP aliasy označují jako virtuální hostitelé na úrovni síťové vrstvy. U služeb WWW a SMTP se častěji používají virtuální hostitelé na úrovni aplikační vrstvy, kdy jedna IP adresa slouží více virtuálním hostitelům a v požadavách příslušné aplikační vrstvy se předává název konkrétního hostitele. Služby jako FTP tímto způsobem fungovat neumějí a vyžadují virtuální hostitele na úrovni síťové vrstvy.

Vše o příkazu ifconfig

Příkaz **ifconfig** má mnohem více parametrů, než jsme si zatím uvedli. Klasické volání tohoto příkazu vypadá takto:

```
ifconfig rozhraní [adresa [parametry]]
```

Parametr rozhraní představuje název rozhraní a parametr adresa představuje IP adresu přidělovanou tomuto rozhraní. Může to být buď IP adresa v tečkové notaci nebo jméno, které si příkaz **ifconfig** vyhledá v souboru `/etc/hosts`.

Je-li příkaz **ifconfig** vyvolán pouze s názvem rozhraní, zobrazí konfiguraci příslušného rozhraní. Je-li spuštěn bez parametrů, zobrazí všechna již dříve nakonfigurovaná rozhraní; volba `-a` vede k zobrazení i neaktivních rozhraní. Příklad volání pro ethernetové rozhraní `eth0` vypadá takto:

```
# ifconfig eth0
eth0      Link encap 10Mbps Ethernet  HWaddr 00:00:C0:90:B3:42
          inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING  MTU 1500  Metric 0
          RX packets 3136 errors 217 dropped 7 overrun 26
          TX packets 1752 errors 25 dropped 0 overrun 0
```

Údaje MTU a Metrics ukazují nastavené MTU a hodnotu metriky rozhraní. Hodnota metriky se v některých operačních systémech tradičně používá k výpočtu ceny dané trasy. Linux tuto hodnotu prozatím nevyužívá, nicméně z důvodů kompatibility ji definuje.

Řádky RX a TX ukazují, kolik paketů bylo bezchybně přijato nebo vysláno, ke kolika chybám došlo, kolik paketů bylo zahozeno (pravděpodobně z důvodu nedostatku paměti) a kolik paketů se ztratilo kvůli přetížení. K přetížení přijímače obvykle dojde v případě, kdy pakety přicházejí rychleji, než stačí jádro operačního systému obsluhovat přerušení. Hodnoty příznaků zobrazené příkazem **ifconfig** zhruba odpovídají názvům jeho voleb na příkazovém řádku, které si vysvětlíme později.

Následuje seznam parametrů rozeznávaných příkazem **ifconfig** společně s odpovídajícími názvy příznaků. Volby zapínající nějakou funkci umožňují i její vypnutí pomocí znaku minus (-) před názvem volby.

`up` — Označí rozhraní jako přístupné pro IP vrstvu. Tato volba je použita automaticky, je-li na příkazovém řádku definována adresa rozhraní. Lze ji také použít k opětovné aktivaci rozhraní, které bylo dočasně deaktivováno pomocí volby `down`. Tato volba odpovídá příznakům `UP` a `RUNNING`.

`down` — Označí rozhraní jako nepřístupné pro IP vrstvu. Tato volba vypne jakýkoliv IP provoz přes dané rozhraní. Volba rovněž automaticky smaže všechny směrovací záznamy používající dané rozhraní.

`netmask` *`maska`* — Tato volba přidělí masku podsítě, kterou bude rozhraní používat. Může být zadána jako 32bitové hexadecimální číslo s předponou `0x` nebo jako čtveřice dekadických čísel oddělených tečkou. I když je tečkový formát obvyklejší, s hexadecimálním se snáze pracuje. Síťová maska je z podstaty binární a převody mezi dvojkovou a šestnáctkovou soustavou jsou jednodušší než mezi dvojkovou a desítkovou.

`pointopoint` *adresa* — Tato volba se používá u IP spojení typu point-to-point, které zahrnuje pouze dva hostitele. Je nutná například při konfiguraci rozhraní SLIP nebo PLIP. (Byla-li nastavena adresa point-to-point, zobrazí příkaz **ifconfig** příznak `POINTOPOINT`.)

`broadcast` *adresa* — Vysílací adresa je obvykle vytvořena z čísla sítě nastavením všech bitů hostitelské části na jedničku. Některé implementace protokolu IP (například BSD 4.2) používají odlišné schéma, při němž jsou naopak všechny bity hostitelské části nulové. Volba `broadcast` slouží pro přizpůsobení těmto podivným prostředím. (Byla-li nastavena vysílací adresa, zobrazí příkaz **ifconfig** příznak `BROADCAST`.)

`irq` — Tento parametr umožňuje u některých zařízení nastavit číslo používaného přerušení. Je to užitečné zejména pro PLIP rozhraní, hodí se však i pro některé ethernetové karty.

`metric` *číslo* — Tato volba slouží k přiřazení metriky záznamu daného rozhraní ve směrovací tabulce. Hodnotu metriky používá protokol RIP (Route Information Protocol) k vytvoření směrovacích tabulek sítí[47]. Implicitní hodnota metriky používaná příkazem ifconfig je rovna nule. Pokud nepoužíváte démona RIP, je vám tato volba k ničemu; pokud ano, budete jen zřídka potřebovat tuto hodnotu měnit.

`mtu` *bajtů* — Tato volba nastavuje maximální přenosovou jednotku (MTU), která představuje maximální počet bajtů, jež rozhraní dokáže obsloužit v jedné transakci. U sítí typu Ethernet je hodnota MTU implicitně nastavena na 1 500 (největší povolená délka ethernetového paketu); u rozhraní typu SLIP je MTU nastavena na hodnotu 296. (Pro nastavení MTU na linkách SLIP nejsou žádná omezení, hodnota 296 představuje vhodný kompromis.)

`arp` — Toto je volba typická pro sítě, jako je Ethernet nebo rádiová síť. Povoluje použití protokolu ARP k detekci fyzických adres hostitelů, kteří jsou připojeni k síti. U vysílajících sítí je tato volba implicitně povolena. Je-li protokol ARP zakázán, zobrazí příkaz **ifconfig** příznak NOARP.

`-arp` — Zakáže na tomto rozhraní používat protokol ARP.

`promisc` — Přepne rozhraní do promiskuitního režimu. U vysílajících sítí to bude znamenat, že dané rozhraní bude přijímat všechny pakety bez ohledu na to, zda byly určeny pro jiného hostitele či nikoliv. Tato volba umožní analyzovat síťový provoz pomocí paketových filtrů (takzvané *sledování Ethernetu*). Je to dobrý způsob k nalezení síťových problémů, které jsou jinak těžko odhalitelné. Na druhou stranu umožní tato volba útočníkům kromě jiných věcí i zjištění potřebných hesel z vašeho síťového provozu. Jednou z ochran proti tomuto typu útoku je všeobecný zákaz připojení jakéhokoliv počítače do vaší ethernetové sítě. Můžete také použít zabezpečené autentikační protokoly, jako je například Kerberos nebo balík secure shell[48]. Této volbě odpovídá příznak PROMISC.

47 Protokol RIP vybírá optimální trasu k hostiteli na základě „délky“ tras. Délka se počítá jako součet jednotlivých metrik rozhraní, přes která trasa vede. Standardně má cesta přes jednu bránu délku 1, může však mít přiřazenu jakoukoliv celou hodnotu do 16. (Trasa o délce 16 odpovídá nekonečnu. Tyto trasy jsou považovány za nepoužitelné.) Cenu uzlu nastavuje právě parametr `metric`, a ta je pak vysílána směrovacím démonem.

48 ssh můžete získat na adrese **ftp.cs.hut.fi** v adresáři `/pub/ssh`.

`-promisc`	Tato volba vypne promiskuitní režim.
`allmulti`	Multicastové adresy se podobají vysílacím adresám, nepřijímají je však automaticky všechny počítače, ale pouze ty, které jsou tak nastaveny. Je to užitečné pro aplikace jako jsou ethernetové videokonference nebo přenos zvuku, kdy přijímají pouze ti, které to zajímá. Multicastové adresy jsou podporovány většinou ethernetových ovladačů, ne však všemi. Je-li tato volba zapnuta, rozhraní přijímá a předává ke zpracování multicastové pakety. Této volbě odpovídá příznak ALLMULTI.
-allmulti	Tato volba vypne multicastové adresy.

Příkaz netstat

Příkaz **netstat** je užitečný nástroj pro kontrolu konfigurace a aktivity sítě. Jde vlastně o sbírku několika nástrojů spojených dohromady. V následujících částech probereme jeho jednotlivé funkce.

Zobrazení směrovací tabulky

Spustíte-li příkaz **netstat** s parametrem `-r`, zobrazí se směrovací tabulka jádra stejně, jako to dělá příkaz **route**. Na hostiteli **vstout** se zobrazí:

```
# netstat -nr
Kernel IP routing table
Destination   Gateway      Genmask         Flags  MSS Window  irtt Iface
127.0.0.1     *            255.255.255.255 UH       0 0          0 lo
172.16.1.0    *            255.255.255.0   U        0 0          0 eth0
172.16.2.0    172.16.1.1   255.255.255.0   UG       0 0          0 eth0
```

Volba –n způsobí, že příkaz **netstat** zobrazí adresy jako čísla a ne jako symbolické názvy hostitelů a sítí. To je užitečné zejména chcete-li se vyhnout vyhledávání adres po síti (například na DNS nebo NIS serveru).

Druhý sloupec výstupu příkazu **netstat** zobrazuje bránu, na niž trasa směruje. Není-li použita žádná brána, zobrazí se hvězdička. Třetí sloupec ukazuje „obecnost" trasy, tedy síťovou masku trasy. Při hledání odpovídající trasy pro nějakou IP adresu jádro projde všechna data směrovací tabulky, mezi adresou a maskou aplikuje binární AND a výsledek porovnává s cílem trasy v tabulce[49].

Čtvrtý sloupec zobrazuje různé symboly, které popisují trasu:

`G`	Směrování používá bránu.
`U`	Používané rozhraní je aktivní.
`H`	Daná trasa vede pouze k jedinému hostiteli. To je například případ lokálního rozhraní 127.0.0.1.
`D`	Trasa byla vytvořena dynamicky. Tento příznak se nastavuje, pokud byla položka směrovací tabulky vytvořena směrovacím démonem jako je **gated** nebo ICMP zprávou o přesměrování (viz část *Protokol ICMP* v kapitole 2).
`M`	Příznak se nastavuje, byla-li trasa upravena ICMP zprávou o přesměrování.
`!`	Jedná se o zamítnutou trasu a datagramy přes ni posílané budou zahazovány.

49 Pozn. překladatele: A samozřejmě vrátí tu trasu, kde je maskovaná adresa požadovaného cíle stejná jako adresa cíle v tabulce. Vyhovuje-li témuž cíli více tras (což je skoro vždycky), vrátí tu trasu, která má „nejdelší" masku (tedy masku s nejvíce jedničkami) – taková trasa je považována za „nejspecifičtější" a předpokládá se, že vede „nejlépe" k cíli.

Další tři sloupce zobrazují hodnoty MSS, Window a irtt, které budou použity pro TCP spojení přes danou trasu. Hodnota MSS znamená Maximum Segment Size a představuje velikost nejdelšího datagramu, který může jádro pro přenos po této trase vytvořit. Window představuje maximální objem dat, který je systém schopen ze vzdáleného hostitele přijmout „v jednom zátahu". Zkratka irtt znamená „initial round trip time" – počáteční čas pro přenos tam a zpět. Protokol TCP zaručuje, že data budou spolehlivě doručena a v případě ztráty datagramu zajistí jeho opakované odeslání. Protokol TCP si pamatuje jak dlouho trvá, než bude datagram doručen na vzdálený konec spojení a než od něj přijde potvrzení o jeho přijetí – tento údaj je právě round-trip time[50] a pokud se do té doby potvrzení neobjeví, protokol předpokládá ztrátu datagramu a odešle jej znovu. Jeho původní hodnotu používá protokol TCP při prvním navázání spojení. Ve většině sítí je standardně nastavená hodnota vyhovující, nicméně u některých pomalých sítí (například radioamatérských) bývá příliš krátká a vede ke zbytečnému opakovanému odesílání datagramů. Hodnotu irtt je možné nastavit příkazem **route**. Nastavením hodnoty 0 se indikuje, že se má použít standardní hodnota.

Konečně v posledním sloupci je zobrazeno síťové rozhraní, přes něž trasa vede.
Sloupec Ref ve výstupu příkazu netstat zobrazuje počet odkazů na dané směrování, to znamená, kolik dalších směrování (například přes brány) spoléhá na přítomnost daného směrování. Poslední dva sloupce zobrazují, kolikrát byla použita směrovací data a dále rozhraní, kterými při doručování procházejí datagramy.

Zobrazení statistik rozhraní

Spuštěním příkazu **netstat** s parametrem `-i` dojde k vypsání statistik všech nakonfigurovaných rozhraní. Pokud je uveden i parametr `-a`, zobrazí se statistiky *všech* rozhraní v jádře, nejenom těch doposud nakonfigurovaných. Na počítači **vstout** by mohl výstup příkazu **netstat** vypadat takto:

```
# netstat -i
Kernel Interface table
Iface MTU Met  RX-OK RX-ERR RX-DRP RX-OVR  TX-OK TX-ERR TX-DRP TX-OVR Flags
lo      0   0   3185      0      0      0   3185      0      0      0 BLRU
eth0 1500   0 972633     17     20    120 628711    217      0      0 BRU
```

Údaje `MTU` a `Met` uvádějí aktuální hodnoty MTU a metriky daného rozhraní. Sloupce `RX` a `TX` ukazují, kolik paketů bylo přijato a vysláno bezchybně (`RX-OK/TX-OK`), kolik bylo poškozeno (`RX-ERR/TX-ERR`), kolik bylo zahozeno (`RX-DRP/TX-DRP`) a kolik se ztratilo z důvodu přetížení (`RX-OVR/TX-OVR`).

Poslední sloupec zobrazuje příznaky daného zařízení. Jsou to jednoznakové ekvivalenty dlouhých názvů příznaků, které se vypisují při zobrazení konfigurace rozhraní pomocí příkazu **ifconfig**:

`B` Byla nastavena vysílací adresa.

`L` Dané rozhraní je lokální rozhraní.

`M` Jsou přijímány všechny pakety (promiskuitní režim).

`O` Pro dané rozhraní je vypnut protokol ARP.

`P` Toto spojení je typu point-to-point.

`R` Rozhraní právě běží.

`U` Rozhraní je povoleno.

50 Pozn. překladatele: Round-trip asi opravdu nejde přeložit jinak než „tam a zpět". Doslova to znamená „kruhový výlet" a ve spojení „round-trip ticket" to je zpáteční jízdenka. (Která je obvykle levnější než „one-way ticket".)

Zobrazení spojení

Příkaz **netstat** podporuje skupinu voleb pro zobrazení aktivních nebo pasivních soketů. Volby `-t`, `-u`, `-w` a `-x` ukazují aktivní TCP, UDP, RAW nebo unix-soketová spojení. Pokud k nim doplníte i parametr `-a`, budou zobrazeny i sokety čekající na spojení (tedy naslouchající). Tato kombinace parametrů vám poskytne úplný výpis všech serverů, které právě běží ve vašem systému.

Při použití příkazu **netstat –ta** se na hostiteli **vlager** zobrazí následující výpis:

```
$ netstat -ta
Active Internet Connections
Proto Recv-Q Send-Q Local Address    Foreign Address    (State)
tcp        0      0 *:domain         *:*                LISTEN
tcp        0      0 *:time           *:*                LISTEN
tcp        0      0 *:smtp           *:*                LISTEN
tcp        0      0 vlager:smtp      vstout:1040        ESTABLISHED
tcp        0      0 *:telnet         *:*                LISTEN
tcp        0      0 localhost:1046   vbardolino:telnet  ESTABLISHED
tcp        0      0 *:chargen        *:*                LISTEN
tcp        0      0 *:daytime        *:*                LISTEN
tcp        0      0 *:discard        *:*                LISTEN
tcp        0      0 *:echo           *:*                LISTEN
tcp        0      0 *:shell          *:*                LISTEN
tcp        0      0 *:login          *:*                LISTEN
```

Tento výpis ukazuje, že většina serverů čeká na příchozí spojení. Nicméně čtvrtý řádek ukazuje příchozí spojení typu SMTP z hostitele **vstout** a šestý řádek vám sděluje, že existuje odchozí spojení typu telnet s hostitelem **vbardolino**[51].

Pokud bychom použili pouze argument –a, zobrazí se všechny sokety ze všech protokolových rodin.

Kontrola tabulek ARP

V určitých situacích je vhodné si prohlédnout, případně změnit obsah tabulek ARP jádra systému – například máte-li podezření, že příčinou občasných síťových problémů je duplicitní IP adresa. Pro takovéto situace byl vytvořen nástroj **arp**. Jeho volby jsou následující:

```
arp [-v] [-t hwtype] -a [hostname]
arp [-v] [-t hwtype] -s hostname hwaddr
arp [-v] -d hostname [hostname...]
```

Všechny parametry hostname (název hostitele) mohou být zadány jako IP adresy nebo jako symbolické názvy.

První typ volání příkazu **arp** zobrazí pro danou IP adresu nebo hostitele položku v ARP tabulce. Nebyl-li zadán název hostitele, zobrazí se všechny položky tabulky. Například na počítači **vlager** můžeme tímto příkazem dostat:

```
# arp -a
IP address      HW type                 HW address
172.16.1.3      10Mbps Ethernet         00:00:C0:5A:42:C1
```

51 Zda je spojení odchozí poznáte z čísla portu v lokální adrese. Čísla portů u *odchozího* spojení budou vždy nějaká obecná celá čísla. V případě *příchozího* spojení bude použito známé číslo portu dané služby a pro ně program **netstat** zobrazuje symbolické názvy služeb jako třeba `smtp`, které jsou definovány v souboru `/etc/services`.

```
172.16.1.2      10Mbps Ethernet         00:00:C0:90:B3:42
172.16.2.4      10Mbps Ethernet         00:00:C0:04:69:AA
```

Vidíme zde ethernetové adresy hostitelů **vlager**, **vstout** a **vale**.

Pomocí volby `-t` můžete omezit výpis pouze na zadaný typ hardwarových zařízení. Ten může být ether, ax25 nebo pronet, což odpovídá (po řadě) 10 Mb Ethernetu, zařízením AMPR AX.25 a zařízení IEEE 802.5 Token Ring.

Parametr –s slouží k trvalému přidání ethernetové adresy hostitele do tabulky. Parametr hwaddr určuje hardwarovou adresu, u níž se implicitně předpokládá, že jde o ethernetovou adresu určenou šesti hexadecimálními bajty oddělenými dvojtečkami. Pomocí volby –t můžete nastavit hardwarové adresy pro jiné typy hardwaru.

Z různých důvodů mohou ARP dotazy selhávat, například je-li na vzdáleném hostiteli chybný ovladač ARP, nebo když v síti existuje další hostitel, který se chybně identifikuje IP adresou vzdáleného hostitele. V takovém případě musíte IP adresu problematického počítače přidat do tabulky ručně. Ruční zadávání údajů do ARP tabulky je rovněž (velice drastické) opatření, kterým se můžete chránit proti hostitelům, jež se vydávají za někoho jiného.

Použijete-li při spuštění příkazu **arp** parametr –d, smažou se z tabulky všechna data, která se vztahují k danému hostiteli. Pomocí tohoto parametru můžete rozhraní přinutit k tomu, aby se znovu pokusilo pomocí dotazu získat ethernetovou adresu odpovídající dané IP adrese. Je to užitečné v případě, kdy špatně nakonfigurovaný systém vyslal do ARP tabulky špatné informace (samozřejmě předtím musíte chybného hostitele znovu zkonfigurovat).

Volba –s slouží k implementaci techniky *ARP proxy*. Je to speciální technika, kdy se nějaký hostitel, řekněme **gate**, chová jako brána pro jiného hostitele, řekněme **fnord**, a předstírá, že IP adresy obou hostitelů odpovídají hardwarové adrese brány. Provede to tak, že brána zveřejní ARP položku hostitele **fnord**, která bude ukazovat na její vlastní ethernetové rozhraní. Když nyní nějaký hostitel pošle ARP dotaz na hostitele **fnord**, hostitel **gate** vrátí odpověď, která bude obsahovat jeho vlastní ethernetovou adresu. Dotazující se hostitel pak všechny datagramy pro **fnord** posílá na hostitele **gate**, který je samozřejmě musí směrovat na **fnord**.

Tyto záměny jsou nutné například v případě, kdy chcete přistupovat k hostiteli **fnord** z počítače s DOSem, který nemá zcela korektní implementaci TCP a nedokáže správě směrovat. Při použití ARP proxy se bude dosovému počítači **fnord** jevit jako počítač na lokální síti a on proto nemusí umět směrovat přes brány.

Další velice užitečnou aplikací techniky ARP proxy je případ, kdy se jeden z vašich hostitelů chová jako brána vůči nějakému jinému hostiteli jen dočasně, například při připojení pomocí modemu. V předešlém příkladu jsme se již setkali s laptopem **vlite**, který je občas připojen k bráně **vlager** spojením typu PLIP. Samozřejmě že tento způsob bude fungovat pouze v případě, kdy je adresa hostitele, na kterém chcete provozovat techniku ARP proxy, ve stejné podsíti jako vaše brána. Například hostitel **vstout** by mohl používat techniku ARP proxy pro libovolného hostitele sítě pivovaru (172.16.1.0), ale nikdy pro hostitele z podsítě vinařství (172.16.2.0).

Správné volání příkazu **arp**, kdy bude hostiteli **fnord** poskytnuto ARP proxy, je uvedeno dále; zadaná ethernetová adresa musí samozřejmě odpovídat hostiteli **gate**:

```
# arp -s fnord 00:00:c0:a1:42:e0 pub
```

Položku ARP proxy můžete odstranit následujícím způsobem:

```
# arp -d fnord
```

KAPITOLA 6

Jmenné služby a konfigurace resolveru

Ve druhé kapitole jsme si řekli, že sítě na bázi protokolu TCP/IP mohou používat různá schémata konverze názvů na adresy. Nejjednodušším způsobem je tabulka hostitelů v souboru `/etc/hosts`. Tento způsob je vhodný pouze pro malé lokální sítě, které spravuje jediný správce a jež nepoužívají žádnou IP komunikaci s okolním světem. Formát souboru `hosts` jsme popsali v kapitole 5.

Další možností převodu názvů na IP adresy je použití služby BIND (*Berkeley Internet Name Domain*). Konfigurace služby BIND je skutečným oříškem, ale jakmile ji jednou zvládnete, bude zohlednění jakýchkoliv změn v síti velmi jednoduché. V systému Linux, stejně jako u dalších unixových systémů, poskytuje jmenné služby program **named**. Po svém spuštění si program načte do své vyrovnávací paměti obsah několika hlavních souborů a pak bude čekat na dotazy od vzdálených nebo místních uživatelských procesů. Existuje několik způsobů, jak službu BIND nastavit a ne všechny vyžadují spuštěný jmenný server na každém hostiteli.

V této kapitole můžeme nabídnout jen o málo více než jen hrubý náčrt způsobu, jakým lze provozovat jmenný server. Náš popis by vám měl stačit, pokud provozujete malou lokální síť s připojením do Internetu. Nejnovější informace o službě BIND najete v jejím zdrojovém balíku, který obsahuje manuálové stránky, popis verze a Příručku operátora BIND. Nelekejte se názvu, jde o velice užitečný dokument. Podrobnější popis služby DNS a související problematiky najdete například v knize „DNS and BIND“ Paula Albitze a Cricketa Liua (vydalo nakladatelství O'Reilly). Otázkám systému DNS se věnuje i konference nazvaná **comp.protocols.tcp-ip.domains**. Technické podrobnosti o DNS naleznete v definičních dokumentech RFC 1033, 1034 a 1035.

Knihovna resolveru

Hovoříme-li o *resolveru*, nemáme na mysli žádnou speciální aplikaci, ale knihovnu resolveru. Jedná se o skupinu funkcí nacházející se ve standardní knihovně jazyka C. Centrálními funkcemi knihovny jsou procedury `gethostbyname(2)` a `gethostbyaddr(2)`, které dokáží nalézt IP adresu odpovídající danému názvu a naopak. Mohou být nastaveny tak, aby pouze využívaly soubor

`hosts`, aby se dotazovaly různých jmenných serverů nebo aby používaly databázi *hosts* služby NIS (Network Information Service).

Po svém volání si funkce resolveru přečtou své konfigurační soubory. Z těchto souborů poznají, koho se mají dotazovat, v jakém pořadí a dozvědí se i další podrobnosti o prostředí, v němž pracují. Starší standardní knihovna Linuxu libc používala jako hlavní konfigurační soubor `/etc/host.conf`, verze 2 standardní knihovny GNU používá soubor `/etc/nsswitch.conf`. Popíšeme si oba, protože oba se běžně používají.

Soubor host.conf

Soubor `/etc/host.conf` říká starším verzím Linuxu, které služby má resolver používat a v jakém pořadí.

Jednotlivé volby musí být v souboru host.conf uvedeny na samostatných řádcích. Pole mohou být oddělena oddělovači (mezery nebo tabulátory). Symbol # označuje komentář, který končí u prvního znaku nové řádky. K dispozici jsou následující volby:

order — Tato volba určuje pořadí, ve kterém budou použity jednotlivé služby resolveru. Přípustné možnosti jsou *bind* pro použití dotazů na jmenný server, *hosts* pro vyhledávání v souboru `/etc/hosts` a *nis* pro vyhledávání pomocí služby NIS. Může být zadána buď jedna nebo více voleb. Pořadí, ve kterém jsou tyto volby uvedeny, bude určovat pořadí, ve kterém budou volány jim odpovídající služby.

multi — Hodnota této volby může být on nebo off. Určuje, zda může mít hostitel v souboru `/etc/hosts` několik IP adres. Počítače tohoto typu se označují jako *multihomed* systémy. U dotazů systému DNS nebo NIS nemá tento přepínač žádný význam.

`nospoof` — O této volbě budeme hovořit v části *Reverzní hledání*. Systém DNS umí najít název hostitele náležející dané IP adrese prostřednictvím domény **in-addr.arpa**. Snaha jmenného serveru vrátit chybný název hostitele se označuje jako takzvaný *spoofing*. Obrana proti tomuto typu útoku spočívá v tom, že resolver nakonfigurujeme tak, že zpětným dotazem zjistí, že IP adrese vrácené jmenným serverem opravdu odpovídá název, který jsme hledali. Pokud neodpovídá, bude název odmítnut a resolver vrátí chybové hlášení. Toto chování se zapíná pomocí volby *nospoof on*.

`alert` — Tato volba pracuje s parametry `on` nebo `off`. Je-li zapnutá, pak veškeré pokusy o spoofing (viz výše) budou zaznamenány prostřednictvím funkce *syslog*.

`trim` — Tato volba má jako parametr název domény, která bude před vyhledáváním z názvů hostitelů odstraněna. Volba je užitečná u těch položek v souboru `hosts`, u kterých jsme zadali pouze názvy hostitelů bez názvu místní domény. Při vyhledávání lokálního hostitele, u nějž je název místní domény uveden, bude její název odstraněn a vyhledání v souboru `/etc/hosts` tak může uspět. Volbu *trim* lze použít vícekrát, takže počítač se může chovat jako lokální na více doménách.

V příkladu 6.1 je uveden příklad souboru `host.conf` pro počítač **vlager**.

Příklad 6.1

```
# /etc/host.conf
# Používáme named, ale ne NIS
order   bind,hosts
# Povolujeme více adres
multi   on
# Ochrana před spoofingem
nospoof on
# Odříznutí lokální domény (není nutné).
trim    vbrew.com.
```

Proměnné prostředí resolveru

Nastavení v souboru `host.conf` lze přepsat pomocí několika proměnných prostředí resolveru:

`RESOLV_HOST_CONF`	Určuje soubor, který bude načten místo souboru `/etc/host.conf`.
`RESOLV_SERV_ORDER`	Přepíše nastavené volby *order* v souboru `host.conf`. Službami mohou být *hosts*, *bind* nebo *nis*. Odděleny mohou být mezerou, čárkou nebo středníkem.
`RESOLV_SPOOF_CHECK`	Určí opatření, která budou použita proti spoofingu. Při hodnotě *off* je kontrola úplně vypnutá. Hodnoty *warn* a *warn off* kontrolují spoofing, a zapíná se, respektive vypíná se záznam detekovaných pokusů. Hodnota * bude kontrolovat spoofing, ale rozsah zaznamenávání chyb ponechá nastavený podle souboru `host.conf`.
`RESOLV_MULTI`	Přepisuje volbu multi v souboru `host.conf`. Proměnná může mít hodnoty on nebo off.
`RESOLV_OVERRIDE_TRIM_DOMAINS`	Tato proměnná určuje názvy domén a přepisuje nastavení volby *trim* v souboru host.conf.
`RESOLV_ADD_TRIM_DOMAINS`	Tato proměnná doplňuje nastavení volby *trim* o další odřezávané domény.

Soubor nsswitch.conf

Verze 2 standardní knihovny GNU libc obsahuje mocnější a pružnější náhradu za původní mechanismus souboru `host.conf`. Služby jmen byly doplněny o možnost poskytovat i jiné druhy údajů. Konfigurační volby všech těchto různých funkcí prohlížejících různé databáze byly soustředěny v jednom konfiguračním souboru `nsswitch.conf`.

Pomocí souboru `nsswitch.conf` může administrátor nakonfigurovat různé databáze. My se omezíme pouze na ty parametry, které se týkají vzájemného převodu názvů a IP adres. Informace o dalších službách snadno naleznete v dokumentaci ke standardní knihovně GNU.

Volby v souboru `nsswitch.conf` musí být uvedeny na samostatných řádcích a jednotlivé údaje se od sebe oddělují mezerami nebo tabulátory. Znak # označuje začátek komentáře, který pokračuje až do konce daného řádku. Každý řádek popisuje jednu službu, rozlišování názvů je jednou z nich. Prvním údajem na řádku je název prohledávané databáze, ukončený dvojtečkou. Databáze zajišťující převod mezi jmény a adresami má název hosts. Souvisí s ní i databáze networks, která slouží k převodu síťových názvů na adresy sítí. Ve zbytku řádku jsou parametry udávající způsob, jakým se budou hodnoty databáze zjišťovat.

Můžeme použít následující volby:

`dns`	Adresy se zjišťují pomocí služby DNS. Tato volba má smysl pouze pro rozlišování názvů hostitelů, nikoliv názvů sítí. Tento mechanismus se dále nastavuje souborem `/etc/resolv.conf`, o kterém budeme hovořit dále.
`files`	Hledá adresy hostitelů a sítí v lokálních souborech `/etc/hosts` a `/etc/networks`.
`nis` nebo `nisplus`	K rozlišování názvů používá službu NIS. O službách NIS a NIS+ budeme podrobně hovořit v kapitole 13.

Pořadí, v němž jsou volby na řádku uvedeny, udává, v jakém pořadí budou jednotlivé služby použity. Jednotlivé služby se používají postupně zleva a standardně hledání končí, jakmile je některou službou název nalezen.

Jednoduché nastavení pro databáze hostitelů a sítí odpovídající tomu, co jsme nastavovali ve starším souboru `host.conf`, je uvedeno v příkladu 6.2.

Příklad 6.2 – Příklad souboru nsswitch.conf

```
# /etc/nsswitch.conf
#
# Příklad konfigurace služby jmen.
# Informace o tomto souboru jsou uvedeny v balíku 'libc6-doc'.

hosts:          dns files
networks:       files
```

Tento příklad způsobí, že se nejprve bude volat služba DNS a pokud té se nepodaří adresu zjistit, použije se soubor `/etc/hosts`. Názvy a adresy sítí budou hledány pouze v souboru `/etc/networks`. Způsob chování je možné definovat přesněji pomocí „položek akcí“, které udávají, co se má provést v závislosti na výsledku předchozího pokusu o nalezení. Položky akcí se uvádějí mezi názvy používaných služeb a jsou uzavřeny v hranatých závorkách [a]. Obecná syntaxe položky akce je:

`[[!] status = akce]`

Existují dvě možné akce:

`return` Řízení se vrátí programu, který o zjištění názvu žádal. Pokud bylo hledání úspěšné, vrátí resolver požadované informace, v opačném případě vrátí prázdný výstup.

`continue` Resolver přejde na další službu a pokusí se název zjistit jejím prostřednictvím.

Nepovinný symbol „!“ znamená, že následující status má být před testováním invertován, chová se tedy jako podmínka typu „když ne“.

Stavy, na něž můžeme reagovat, jsou:

`succes` Požadovaný údaj byl bez chyby nalezen. Implicitní akcí pro tento status je return.

`notfound` Při vyhledávání nedošlo k chybě, nicméně název hostitele nebo sítě nebylo možné najít. Implicitní akce pro tento status je continue.

`unavail` Dotazovaná služba je nedostupná. Může to znamenat, že služba nemůže číst soubor `hosts` nebo `networks` nebo že vzdálený NIS nebo DNS server na požadavek neodpověděl. Implicitní akce pro tento status je `continue`.

`tryagain` Tento status znamená, že služba je momentálně nedostupná. U souborového rozlišování to typicky znamená, že soubory jsou momentálně zamčeny jiným procesem. U jiných služeb to může znamenat, že jmenný server nemůže dočasně přijímat požadavky. Implicitní akce pro tento status je `continue`.

Jednoduchý příklad použití tohoto mechanismu je uveden v příkladu 6.3.

Příklad 6.3 – Příklad souboru nsswitch.conf s definicí akcí

```
# /etc/nsswitch.conf
#
# Příklad konfigurace služby jmen.
# Informace o tomto souboru jsou uvedeny v balíku 'libc6-doc'.

hosts:          dns [!UNAVAIL=return] files
networks:       files
```

Tento příklad se pokouší zjistit název službou DNS. Pokud je návratový status jiný než „nedostupný“, resolver vrátí, co zjistil. Tehdy a jen tehdy, vrátí-li služba DNS status „nedostupný“, se resolver pokusí o rozlišení pomocí souboru `/etc/hosts`. Znamená to, že soubor `/etc/hosts` budeme používat pouze v případě, že jmenný server není z nějakého důvodu dostupný.

Konfigurace vyhledávání souborem resolv.conf

Nakonfigurujete-li knihovnu resolveru tak, aby k vyhledávání hostitelů používala jmennou službu BIND, musíte jí sdělit, jaké má používat jmenné servery. K tomuto účelu se používá speciální soubor s názvem `resolv.conf`. Pokud tento soubor neexistuje nebo je prázdný, bude resolver předpokládat, že se jmenný server nachází na vašem místním hostiteli.

Provozujete-li jmenný server na svém místním hostiteli, musíte ho nastavit samostatně, což si vysvětlíme v dalším textu. Pokud se nacházíte v lokální síti a máte možnost používat existující jmenný server, měli byste dát této možnosti přednost. Pokud se k Internetu připojujete vytáčenou linkou, budete typicky v souboru `resolv.conf` specifikovat jmenný server vašeho poskytovatele konektivity.

Nejdůležitější volbou v souboru `resolv.conf` je volba *name server*, která udává IP adresu serveru, jenž se má používat. Pokud opakovaným uvedením volby nadefinujete více jmenných serverů, budou se používat v uvedeném pořadí. Z toho důvodu byste měli na prvním místě uvést nejspolehlivější jmenný server. Současná implementace umožňuje zadat až tři jmenné servery. Pokud není žádný server definován, předpokládá resolver, že server běží na lokálním počítači.

Další dvě volby, *domain* a *search*, umožňují používat zkrácená jména hostitelů v lokální doméně. Pokud se chcete například telnetem připojit k počítači v lokální doméně, nebudete typicky chtít vypisovat celý její název a zadáte pouze samotný název počítače, například **gauss** a očekáváte, že resolver si sám doplní **mathematics.groucho.edu**.

To je funkce parametru *domain*. Umožňuje zadat název implicitní domény, která se má připojit k hledanému názvu v případě, že se jej nepodaří nalézt. Pokud například zadáte jméno **gauss**, pokusí se DNS najít adresu počítače **gauss.** a nepovede se jí to, protože nejde o jméno domény nejvyšší úrovně. Poté resolver připojí název domény **mathematics.groucho.edu.**, opakuje dotaz a tentokrát uspěje.

Možná si myslíte, že je to hezké, jenže jakmile jdete mimo doménu katedry matematiky, jste opět odkázáni na použití plně kvalifikovaných jmen. Jenže vy byste třeba chtěli používat zkrácené názvy jako **quark.physics** pro počítače katedry fyziky.

Zde přichází ke slovu *prohledávací seznam*. Tento seznam se definuje volbou *search*, která představuje zobecnění volby *domain*. Zatímco ta slouží k nastavení jediné implicitní domény, volba *search* umožňuje zadat celý seznam domén a postupně se zkoušejí všechny, dokud hledání neuspěje. Položky seznamu se oddělují mezerami nebo tabulátory.

Nastavují implicitní domény, které budou připojeny k názvu hostitele v případě, že se službě BIND nepodaří při prvním dotazu nalézt příslušný název hostitele. Volba *search* určuje seznam zkoušených názvů domén. Jednotlivé položky v seznamu jsou od sebe odděleny mezerami nebo tabulátory.

Příkazy *domain* a *search* se navzájem vylučují a mohou být uvedeny pouze jednou. Pokud není použit žádný z nich, pokusí se resolver zjistit jméno lokální domény z jména lokálního počítače voláním funkce `getdomainname(2)`. Pokud není v názvu lokálního počítače specifikována doména, bude jako implicitní doména použita kořenová doména.

Rozhodnete-li se tyto volby v souboru `resolv.conf` použít, musíte si dát pozor na to, které domény chcete do seznamu přidat. Knihovny resolveru do verze BIND 4.9 si v případě nezadání seznamu implicitních domén tento seznam vytvořily samy tak, že použily název lokální domény a všechny její rodičovské domény až po kořenovou. To způsobovalo určité potíže, protože požadavky se tak dostávaly k DNS serverům, kterým vůbec nepatřily. Řekněme, že jste ve virtuálním pivovaru a chcete se přihlásit k počítači **foot.groucho.edu**. Drobným překlepem zadáte namísto jména **foot** název **foo**, který neexistuje. Jmenný server vám tedy sdělí, že takovýto počítač nezná. Starší verze resolveru se nyní pokoušely požadavek obsloužit připojením domény **vbrew.com** a **com**. Ta druhá varianta je ovšem problematická, protože doména **groucho.edu.com** by klidně mohla existovat. Jejich jmenný server dokonce může znát i jejich počítač **foo** a vrátí vám IP adresu tohoto počítače – což je samozřejmě něco úplně jiného, než co jste chtěli.

U některých aplikací mohou takovéto chybně nalezené adresy představovat vážný bezpečnostní problém. Proto byste měli seznam prohledávaných domén omezit pouze na lokální domény nebo něco podobného. Na katedře matematiky univerzity Groucho Marx by mohl prohledávací seznam obsahovat například domény *mathematics.groucho.edu* a *groucho.edu*.

Pokud vám připadají implicitní domény zmatené, podívejte se na následující příklad souboru `resolv.conf` pro virtuální pivovar:

```
# /etc/resolv.conf
# Naše doména
domain          vbrew.com
#
# Jako centrální nameserver slouží vlager:
nameserver      172.16.1.1
```

Když budeme hledat název **vale**, pokusí se resolver nejprve najít **vale** a když se mu to nepodaří, pak **vale.vbrew.com**.

Robustnost resolveru

Pokud používáte menší lokální síť v rámci rozsáhlé sítě, měli byste rozhodně používat centrální jmenné servery, jsou-li tyto k dispozici. Výhoda tohoto způsobu spočívá v tom, že centrální jmenné servery si vytvoří velkou vyrovnávací paměť, protože na ně budou směrovány veškeré dotazy. Toto schéma má ale i nevýhody: pokud by shořel kabel páteřní sítě Olafovy univerzity, nedalo by se pracovat na žádné jejich lokální síti, protože by resolver nemohl nalézt žádný jmenný server. Tato situace způsobí problémy většině síťových služeb například přihlášení k X terminálům nebo tisku.

I když není příliš běžné, aby začala hořet páteř univerzitní sítě, budete asi chtít proti takovýmto případům učinit určitá opatření.

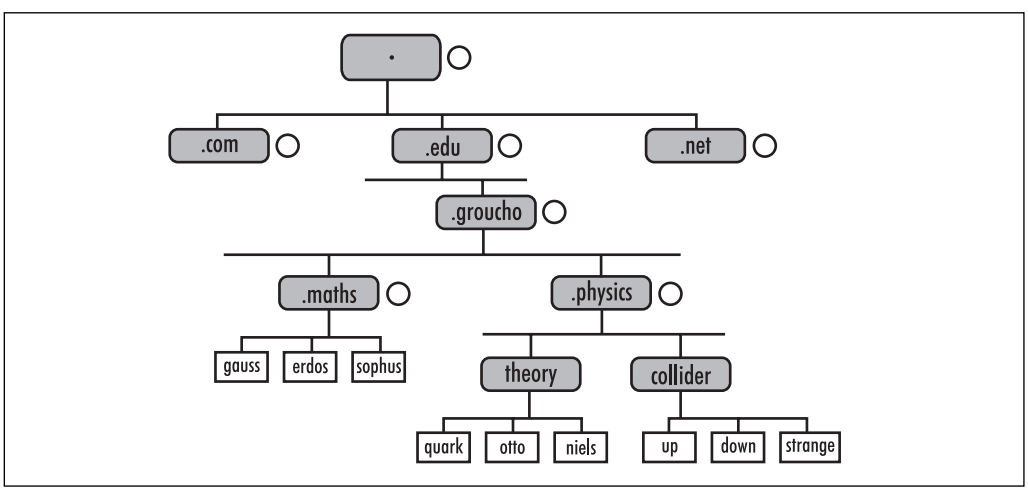
Jednou z možností je vytvořit místní jmenný server tak, aby hledal názvy hostitelů z vaší místní domény a všechny ostatní dotazy na další názvy hostitelů předával na hlavní server. Samozřejmě je to možné pouze v případě, že provozujete svou vlastní doménu.

Máte i druhou možnost – v souboru `/etc/hosts` udržovat záložní tabulku hostitelů vaší domény nebo lokální sítě. To není tak složité. Pak zajistíte, aby resolver nejprve používal DNS a pak tento soubor. V `/etc/host.conf` toho dosáhnete příkazem *order bind, hosts*, v souboru `/etc/nsswitch.conf` příkazem *hosts: dns files*.

Jak DNS funguje

DNS organizuje počítače do hierarchie domén. *Doména* představuje skupinu lokalit, které spolu nějak souvisejí – například tak, že vytvářejí jistou síť (všechny počítače univerzity, všechny počítače sítě BITNET a podobně), že všechny patří jedné instituci (například americké vládě) nebo že jsou geograficky pohromadě. Například univerzity jsou typicky sdruženy v doméně **edu**, přičemž každá univerzita nebo jiná vysoká škola používá vlastní *subdoménu*, v níž jsou sdruženi její hostitelé. Groucho Marx University má přidělenu doménu **groucho.edu** a lokální síť katedry matematiky má přidělenu subdoménu **maths.groucho.edu**. Hostitelé v síti katedry budou tento název domény připojovat ke svým názvům; takže hostitel **erdos** bude znám jako **erdos.maths.groucho.edu**. Tento název se označuje jako *plně kvalifikované doménové jméno* (FQDN) a jednoznačně identifikuje tohoto hostitele po celém světě.

Obrázek 6.1 ukazuje část doménového prostoru jmen. Vstup u kořene tohoto stromu, který je označen tečkou, je poměrně výstižně nazván *kořenovou doménou* a obsahuje v sobě všechny domény. Aby se naznačilo, že název hostitele je zadán plně kvalifikovaným doménovým jménem a nikoliv relativně pomocí nějaké (implicitní) subdomény, píše se někdy s tečkou na konci. Ta udává, že název poslední části představuje kořenovou doménu.



Obrázek 6.1 – Část doménového prostoru jmen

V závislosti na jejím umístění v hierarchii názvů může být doména nazvána doménou nejvyšší úrovně (primární doménou), doménou druhé úrovně nebo třetí úrovně. Více úrovní rozdělení je sice možné, ale vyskytuje se jen zřídka. Následuje přehled několika domén nejvyšší úrovně, s nimiž se můžete často setkat:

edu	Vzdělávací instituce (většinou v USA), jako jsou univerzity.
com	Komerční organizace a společnosti.
org	Nekomerční organizace. Často jsou v této doméně soukromé sítě typu UUCP.
net	Brány a další administrativní hostitelé na síti.
mil	Americké armádní instituce.
gov	Americké vládní instituce.
uucp	Oficiálně byly do této domény přesunuty názvy všech systémů, které byly dříve používány jako názvy UUCP bez domén.

Historicky byly první čtyři domény určeny jen pro USA, nicméně nedávné změny v politice použití domén vedly k tomu, že tyto domény nazvané globální domény nejvyšší úrovně (gTLD) jsou nyní chápány jako globální. V současné době probíhají jednání o rozšíření počtu gTLD domén a v blízké budoucnosti by měl jejich počet vzrůst[52].

Všechny státy mimo USA používají vlastní doménu nejvyšší úrovně[53], která je tvořena dvoupísmenným kódem země podle definice standardu ISO-3166. Například Finsko používá doménu **fi**, doména **fr** je přidělena Francii, doména **de** Německu, doména **au** patří Austrálii. Pod touto doménou nejvyšší úrovně může centrum NIC každé země libovolným způsobem organizovat názvy hostitelů[54]. Například Austrálie má doménu druhé úrovně podobnou mezinárodním doménám nejvyšší úrovně, tedy **com.au**, **edu.au** a podobně. Ostatní země, jako je například Německo nebo Česká republika, tuto speciální úroveň nepoužívají, ale využívají raději delší názvy, které odkazují přímo na organizace, jimž doména patří. Například není nic výjimečného setkat se s hostitelem, jehož název je například **ftp.informatik.uni-erlangen.de**. Zřejmě to nijak neovlivňuje německou výkonnost.

U těchto národních domén samozřejmě nemusí platit, že hostitel pod příslušnou doménou skutečně fyzicky leží v dané zemi; pouze to signalizuje, že hostitel byl registrován centrem NIC dané země. Švédský výrobce může mít filiálku v Austrálii a přesto bude mít všechny své hostitele registrovány pod doménou nejvyšší úrovně **se**.

Organizováním názvů do hierarchie názvů domén se tedy elegantně vyřeší problém jedinečnosti názvů; v systému DNS musí být název hostitele jedinečný pouze uvnitř vlastní domény, protože ta mu dává název odlišný od ostatních hostitelů po celém světě. Kromě toho jsou plně kvalifikované názvy poměrně lehce zapamatovatelné. Je rovněž vhodné rozdělit rozsáhlou doménu na několik subdomén.

Ale systém DNS toho umí ještě mnohem více: umožňuje pověřit správou subdomény její správce. Například správci ve výpočetním středisku univerzity Groucho Marx mohou vytvořit subdoménu pro každou katedru. Již jsme se setkali se subdoménami **math** a **physics**. Pokud se bude zdát správcům síť katedry fyziky pro správu zvenčí příliš rozsáhlá a chaotická (koneckonců fyzici jsou známí jako neovladatelná skupina lidí), mohou jednoduše předat řízení nad doménou **phy-**

52 Pozn. překladatele: K 16. listopadu 2000 byla jednání ukončena a budou zavedeny následující nové globální domény (v závorce je uveden správce příslušné domény): **.aero** (Societe Internationale de Telecommunications Aeronautiques SC, SITA), **.biz** (JVTeam, LLC), **.coop** (National Cooperative Business Association, NCBA), **.info** (Afilias, LLC), **.museum** (Museum Domain Management Association, MDMA), **.name** (Global Name Registry, LTD) a **.pro** (RegistryPro, LTD). Zatím nicméně zůstává nevyjasněno, jaká pravidla budou pro přidělování adres v těchto doménách přesně platit.

53 Pozn. překladatele: I USA mají geografickou primární doménu **.us**, ale líní Američané ji moc nevyužívají a raději používají doménu **.com** a podobné.

54 Pozn. překladatele: NIC (Network Information Center) je instituce, která v dané zemi zajišťuje přidělování a registraci sekundárních domén a udržuje servery primární domény. V ČR je touto institucí CZ-NIC, z.s.p.o., *http://www.nic.cz*.

sics.groucho.edu správcům této sítě. Ti potom mohou používat libovolné názvy hostitelů a přidělovat jim IP adresy své sítě, aniž by do toho zvenčí někdo zasahoval.

Tím se celý prostor jmen rozpadá na *zóny*, které vždy začínají od nějaké domény. Mezi *zónou* a *doménou* je drobný rozdíl: doména **groucho.edu** obsahuje všechny hostitele univerzity Groucho Marx, zatímco *zóna* **groucho.edu** obsahuje pouze hostitele, které přímo spravuje výpočetní centrum, například mimo jiné hostitele na katedře matematiky. Hostitelé z katedry fyziky patří do odlišné zóny, konkrétně **physics.groucho.edu**. Na obrázku 6.1 je začátek zóny označen malým kolečkem napravo od názvu domény.

Vyhledávání názvů s pomocí DNS

Na první pohled se může zdát, že u všech těchto rozsáhlých domén a zón je provedení rozlišení názvu nesmírně komplikované. Koneckonců, neřídí-li názvy, které jsou přidělovány daným hostitelům, žádná centrální správa, jak je má potom aplikace na nižší úrovni uhodnout?

Nyní přichází na řadu bezelstná vlastnost DNS. Chcete-li nalézt IP adresu serveru **erdos**, pak vám DNS odpoví, že se máte jít zeptat těch lidí, kteří ho spravují, a oni vám ji řeknou.

Systém DNS je vlastně obrovskou distribuovanou databází. Je implementován pomocí takzvaných jmenných serverů, které dodávají informace o dané doméně nebo o dané skupině domén. U každé zóny existují nejméně dva a nejvýše několik jmenných serverů, které uchovávají všechny závazné informace o hostitelích v příslušné zóně. Pokud chcete získat IP adresu serveru **erdos**, pak se stačí spojit s jmenným serverem zóny **groucho.edu**, který vám následně vrátí požadovaná data.

Možná si myslíte, že se o tom mnohem snadněji mluví, ale hůře se to provádí. Takže, jak mám vědět, jak se spojit s jmenným serverem na Groucho Marx University? V případě, že váš počítač není vybaven věštírnou pro hádání adres jmenných serverů, zvládne i to systém DNS za něj. Když chce vaše aplikace vyhledat informace o serveru **erdos**, spojí se s místním jmenným serverem, který aplikuje na požadované informace takzvaný iterační dotaz. Začne zasláním dotazu jmennému serveru kořenové domény, kde se zeptá na adresu **erdos.maths.groucho.edu**. Kořenový jmenný server pozná, že tento název nepatří do jeho správní zóny, ale patří do nějaké zóny pod doménou **edu**. Takže vám sdělí, že se máte spojit s jmenným serverem zóny **edu**, kde získáte více informací. Ke své odpovědi dále přibalí i seznam všech jmenných serverů domény **edu** společně s jejich adresami. Potom bude váš místní jmenný server pokračovat a pošle dotaz na některý z nich, například na **a.isi.edu**. Podobným způsobem jako u kořenového jmenného serveru i server **a.isi.edu** ví, že lidé z **groucho.edu** mají svou vlastní zónu a odkáže vás na její vlastní servery. Místní jmenný server bude pokračovat v dotazu na server **erdos** na jednom z těchto serverů, který konečně zjistí, že hledaný název patří do jeho zóny, a vrátí odpovídající IP adresu.

Teď to možná vypadá, že kvůli vyhledání jedné mizerné IP adresy bylo zapotřebí provést spoustu operací, ale ve skutečnosti je to pouhé minimum v porovnání s množstvím dat, která by musela být přenesena, kdyby se stále pracovalo se souborem `HOSTS.TXT`. Stále však existuje prostor, jak toto schéma dále vylepšit.

Aby zkrátil dobu odpovědi při dalších dotazech, uchovává jmenný server získané informace ve své místní *vyrovnávací paměti*. Takže když chce příště někdo z vaší lokální sítě vyhledat adresu hostitele v doméně **groucho.edu**, nemusí jmenný server znovu absolvovat celý výše popsaný proces, ale přímo se spojí s jmenným serverem pro doménu **groucho.edu**[55].

55 Kdyby se informace neukládaly do vyrovnávací paměti, byl by DNS neefektivní stejně jako jakýkoliv jiný systém, protože každý dotaz by bylo nutné řešit pomocí kořenových serverů.

Samozřejmě, že server nebude tyto informace uchovávat donekonečna, ale po určité době je smaže. Tato doba platnosti se nazývá *životnost*, zkráceně TTL. V databázi DNS přiděluje každé položce TTL správce, který je zodpovědný za danou zónu.

Typy jmenných serverů

Jmenné servery, které uchovávají všechny informace o hostitelích příslušné zóny, se nazývají *autoritativními jmennými servery* této zóny a někdy jsou také označovány jako *hlavní jmenné servery*. Jakýkoliv dotaz na hostitele uvnitř této zóny nakonec skončí u některého z hlavních jmenných serverů zóny.

Hlavní servery musí být velmi dobře synchronizovány. Toho dosáhneme tak, že jeden ze serverů pracuje jako *primární*. Ten bude načítat informace o své zóně z datových souborů a ostatní servery budou pracovat jako *sekundární* a budou v pravidelných intervalech přenášet data z primárního serveru.

Vytvořením několika jmenných serverů se rozkládá zatížení a zvyšuje se spolehlivost. Pokud některý ze serverů přestane pracovat, například když spadne nebo ztratí síťové spojení, přejdou všechny dotazy na ostatní servery. Samozřejmě vás toto schéma neochrání před chybami serveru, jejichž důsledkem budou špatné odpovědi na všechny požadavky systému DNS, například z důvodu softwarových chyb ve vlastním programu serveru.

Kromě toho můžete provozovat i jmenný server, který nebude autoritativním serverem žádné domény[56]. Tento typ serveru je důležitý, protože je schopen provádět DNS dotazy pro aplikace, které jsou spuštěny v lokální síti, a tyto informace ukládá do své vyrovnávací paměti. Proto se tento typ serveru nazývá *caching-only* server.

Databáze DNS

Ukázali jsme si, že systém DNS nejenom určuje IP adresy hostitelů, ale také vyměňuje informace mezi jmennými servery. Ve skutečnosti může databáze DNS obsahovat celou řadu různých typů záznamů.

V databázi DNS se jednotlivým informacím říká *zdrojový záznam*, zkráceně RR (resource record). Každý záznam má přidělen typ, který popisuje druh dat, jež obsahuje a třídu, určující typ sítě, k níž se záznam vztahuje. Třídy pokrývají potřeby různých adresových schémat, jako jsou IP adresy (třída IN) nebo adresy sítí Hesiod (používaných systémem Kerberos z MIT) a některé další. Obvyklým typem zdrojového záznamu je záznam A, který přiděluje plně kvalifikovanému doménovému jménu IP adresu.

Hostitel samozřejmě může mít více než jeden název. Můžete mít například server, který poskytuje služby FTP a WWW a máte pro něj dvě jména: **ftp.machine.org** a **www.machine.org**. Nicméně jen jedno z těchto jmen je označeno jako oficiální, neboli *kanonické*, ostatní jsou pak aliasy tohoto kanonického jména. Rozdíl spočívá v tom, že kanonický název hostitele je ten, jenž je definován záznamem typu A, zatímco ostatní jména jsou definována záznamem typu CNAME, který je odkazem na kanonický název.

Nebudeme si zde popisovat všechny typy záznamů, to si necháme do příští kapitoly. Nyní vám raději poskytneme krátký příklad. Příklad 6.4 ukazuje část doménové databáze, kterou používají jmenné servery zóny **physics.groucho.edu**.

56 Tedy skoro žádné. Jmenný server musí vždy obsluhovat alespoň název **localhost** a reverzní převod pro adresu 127.0.0.1.

Příklad 6.4 – Část souboru named.hosts pro katedru fyziky

```
; Autoritativní údaje o doméně physics.groucho.edu.
@  IN  SOA niels.physics.groucho.edu. janet.niels.physics.groucho.edu. (
                  1999090200       ; sériové číslo
                  360000           ; obnovování
                  3600             ; opakování pokusů
                  3600000          ; platnost
                  3600             ; implicitní ttl
                )
;
; Jmenné servery
              IN    NS       niels
              IN    NS       gauss.maths.groucho.edu.
gauss.maths.groucho.edu. IN A 149.76.4.23
;
; Teoretická fyzika (podsíť 12)
niels         IN    A        149.76.12.1
              IN    A        149.76.1.12
nameserver    IN    CNAME    niels
otto          IN    A        149.76.12.2
quark         IN    A        149.76.12.4
down          IN    A        149.76.12.5
strange       IN    A        149.76.12.6
...
; Laboratoř urychlovačů (podsíť 14)
boson         IN    A        149.76.14.1
muon          IN    A        149.76.14.7
bogon         IN    A        149.76.14.12
...
```

Kromě záznamů typu A a CNAME můžete v horní části souboru vidět speciální záznam, který je rozložen na několika řádcích. Je to zdrojový záznam typu SOA – *Start of Authority*, kde jsou umístěny všeobecné informace o zóně spravované daným serverem. Tento záznam například obsahuje implicitní hodnotu životnosti všech údajů.

Všimněte si, že všechny názvy v ukázkovém souboru, které nekončí tečkou, budou interpretovány vzhledem k doméně **physics.groucho.edu**. Speciální název „@“ použitý v záznamu typu SOA odkazuje na vlastní název domény.

Řekli jsme si už, že jmenné servery domény **groucho.edu** musí nějakým způsobem vědět o zóně **physics**, aby mohly odkazovat dotazy na její jmenné servery. Toho se obvykle dosáhne pomocí dvojice záznamů: záznam typu NS, který poskytne FQDN jmenného serveru, a záznam typu A, jenž tomuto názvu přidruží adresu. Protože právě tyto záznamy propojují celý prostor jmen dohromady, označují se často jako takzvané glue records (tmelicí záznamy). Jsou jedinými typy záznamů, kdy rodičovská zóna uchovává informace o hostitelích podřazené zóny. V příkladu 6.5 jsou uvedeny tmelicí záznamy s odkazy na jmenné servery katedry fyziky, které musí být uvedeny v databázi nadřazeného jmenného serveru (tedy serveru domény **groucho.edu**).

Příklad 6.5 – Část souboru named.hosts jmenného serveru univerzity Groucho Marx

```
; Data zóny groucho.edu.
 @  IN  SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                      1999070100       ; sériové číslo
                      360000           ; obnovování
                      3600             ; opakování pokusů
                      3600000          ; platnost
                      3600             ; implicitní ttl
              }
 ....
 ;
 ; Tmelicí záznamy zóny physics.groucho.edu
 physics        IN     NS        niels.physics.groucho.edu.
                IN     NS        gauss.maths.groucho.edu.
 niels.physics  IN     A         149.76.12.1
 gauss.maths    IN     A         149.76.4.23
 ...
```

Reverzní hledání

Kromě vyhledávání IP-adresy, která náleží hostiteli, je někdy nutné vyhledat také název kanonického hostitele, který odpovídá nějaké adrese. Tento proces se označuje jako *reverzní mapování* a některé síťové služby jej používají k ověřování identity klienta. Pokud se používá soubor hosts, reverzní hledání představuje pouze otázku toho, nalézt v tomto souboru hostitele, který odpovídá dané adrese. U DNS samozřejmě nepřipadá v úvahu úplné prohledání jmenného prostoru. Místo toho byla vytvořena speciální doména **in-addr.arpa**, která obsahuje IP adresy všech hostitelů v obrácené tečkové notaci. Například IP adrese 149.76.12.4 odpovídá název **4.12.76.149.in-addr.arpa**. Sdružení těchto názvů s názvy kanonických hostitelů zajišťuje záznam typu PTR.

Vytvoření zóny obvykle znamená, že její správci mají plnou kontrolu nad způsobem, jakým přidělují jednotlivým názvům adresy. Protože většinou obsluhují jednu nebo více sítí nebo podsítí, existuje mezi DNS zónami a IP sítěmi jedno či více mapování. Například katedra fyziky obsahuje podsítě 149.76.8.0, 149.76.12.0 a 149.76.14.0.

V důsledku toho musí být v doméně **in-addr.arpa** společně se zónou **physics** vytvořeny nové zóny a ty musí být zpřístupněny správcům sítě katedry – půjde o zóny **8.76.149.in-addr.arpa**, **12.76.149.in-addr.arpa** a **14.76.149.in-addr.arpa**. Jinak by instalace nového hostitele v laboratoři urychlovačů vyžadovala, aby se správci spojili se svou nadřazenou doménou, kde se nová adresa zapíše do souboru zóny **in-addr.arpa**.

Zónová databáze podsítě 12 je uvedena na příkladu 6.6. Odpovídající tmelicí záznamy v databázi nadřazené zóny jsou uvedeny v příkladu 6.7.

Příklad 6.6 – Část souboru named.rev pro podsíť 12

```
; doména 12.76.149.in-addr.arpa.
 @  IN  SOA  niels.physics.groucho.edu. janet.niels.physics.groucho.edu. {
                      1999090200 360000 3600 3600000 3600
            }
 2         IN     PTR       otto.physics.groucho.edu.
 4         IN     PTR       quark.physics.groucho.edu.
 5         IN     PTR       down.physics.groucho.edu.
 6         IN     PTR       strange.physics.groucho.edu.
```

Příklad 6.7 – Část souboru named.rev pro síť 149.76

```
; doména 76.149.in-addr.arpa
 @  IN  SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                      1999070100 360000 3600 3600000 3600
                  }
 ...
 ; podsíť 4: katedra matematiky
 1.4         IN     PTR      sophus.maths.groucho.edu.
 17.4        IN     PTR      erdos.maths.groucho.edu.
 23.4        IN     PTR      gauss.maths.groucho.edu.
 ...
 ; podsíť 12: katedra fyziky, samostatná zóna
 12          IN     NS       niels.physics.groucho.edu.
             IN     NS       gauss.maths.groucho.edu.
 niels.physics.groucho.edu. IN  A 149.76.12.1
 gauss.maths.groucho.edu. IN  A   149.76.4.23
 ...
```

Zóny mohou být vytvářeny pouze jako nadmnožiny sítí IP. Co je ještě horší je to, že síťové masky podsítí musí být na hranicích celých bajtů. Všechny podsítě v univerzitě Groucho Marx mají síťovou masku 255.255.255.0, takže lze pro každou podsíť vytvořit zónu v doméně **in-addr.arpa**. Pokud by se používala síťová maska 255.255.255.128, nebylo by možné vytvořit zónu pro podsíť 149.76.12.128, protože by neexistoval žádný způsob, jak sdělit systému DNS, že doména **12.76.149.in-addr.arpa** byla rozdělena na dvě zóny se samostatnou správou a s názvy hostitelů od 1 do 127, respektive od 129 do 254.

Provozování programu named

Program, který na většině unixových počítačů poskytuje doménové jmenné služby, se obvykle jmenuje named (čti *nejmdí*, ne *nejmd*). Jedná se o serverový program původně vyvinutý pro operační systém BSD, kde poskytuje služby jmen klientům a případně i dalším jmenným serverům. Nějakou dobu se nejvíce používal BIND verze 4 a byl součástí většiny distribucí Linuxu. Novou verzí používanou v dnešních distribucích je verze 8, která od minulé verze představuje podstatnou změnu[57]. Obsahuje mnoho nových funkcí, například podporu dynamické aktualizace DNS, oznamování změn v DNS, má výrazně vyšší výkon a používá novou syntaxi konfiguračního souboru. Podrobnosti naleznete v dokumentaci dodávané se zdrojovými soubory.

Tato stať vyžaduje určité znalosti toho, jakým způsobem systém doménových jmen funguje. Bude-li pro vás následující text tak trochu španělskou vesnicí, měli byste si znovu přečíst předcházející část *Jak DNS funguje*.

Program **named** je obvykle spuštěn při zavádění systému a běží tak dlouho, dokud počítač nevypnete. Do verze 8 získávaly informace z konfiguračního souboru `/etc/named.boot` a z mnoha dalších souborů, které obsahují data týkající se mapování názvů domén na adresy. Tyto soubory se označují jako *zónové soubory*. Verze BIND 8 používá namísto souboru `/etc/named.boot` soubor `/etc/named.conf`.

Program **named** spustíte z příkazové řádky příkazem:

```
# /usr/sbin/named
```

57 BIND 4.9 vyvinul Paul Vixie, *paul@vix.com*, nyní však BIND udržuje Internet Software Consortium, *bind-bugs@isc.org*.

Program **named** načte konfigurační soubor named.conf a všechny další v něm uvedené zónové soubory. Své identifikační číslo procesu zapíše ve formátu ASCII do souboru `/var/run/named.pid`, je-li to nutné, načte zónové soubory z primárních serverů a na portu 53 začne přijímat DNS dotazy.

Konfigurační soubor named.boot

Konfigurační soubor verzí starších než 8 měl velmi jednoduchou strukturu. Verze 8 používá úplně odlišnou syntaxi konfiguračního souboru, protože musí pracovat s řadou nových funkcí. Název konfiguračního souboru `/etc/named.boot` ve starších verzích se ve verzi 8 změnil na `/etc/named.conf`. Zaměříme se na nastavení souboru `/etc/named.boot`, protože ten doposud používá řada distribucí, nicméně pro ilustraci rozdílů budeme uvádět i adekvátní příkazy souboru `named.conf` a řekneme si, jak starý formát zkonvertovat do nového.

Soubor `named.boot` je zpravidla velmi malý a obsahuje pouze ukazatele na hlavní soubory s informacemi o zónách a ukazateli na další jmenné servery. V tomto zaváděcím souboru začínají komentáře znaky # nebo ; a končí u dalšího znaku nové řádky. Dříve než si probereme formát souboru `named.boot` podrobněji, podíváme se na vzorový soubor pro bránu **vlager**, který vidíte na příkladu 6.8:

Příklad 6.8 – Soubor named.boot brány vlager

```
;
; Soubor /etc/named.boot brány vlager.vbrew.com
;
directory     /var/named
;
;             doména                    soubor
;-----------------
cache         .                         named.ca
primary       vbrew.com                 named.hosts
primary       0.0.127.in-addr.arpa      named.local
primary       16.172.in-addr.arpa       named.rev
```

Podívejme se postupně na jednotlivé příkazy. Klíčové slovo *directory* říká programu **named**, že všechny soubory dále uvedené jsou uloženy v adresáři `/var/named`. Díky tomu si ušetříme něco psaní.

Klíčové slovo *primary* nahrává do démona **named** informace. Tyto informace se získávají z hlavních souborů, uvedených jako parametry. První příkaz říká programu **named**, že má fungovat jako primární server domény **vbrew.com** a data má načíst ze souboru named.hosts.

Klíčové slovo *cache* je velmi zvláštní a mělo by být použito na všech systémech, na nichž jmenný server běží. Říká programu **named**, že má aktivovat vyrovnávací paměť a že má načíst názvy primárních serverů ze zadaného souboru (v našem případě named.ca). K doporučením pro provoz jmenného serveru se vrátíme později.

Následuje seznam nejdůležitějších voleb, které můžete použít v souboru named.boot:

`directory` Tato volba určuje adresář, ve kterém budou umístěny zónové soubory. Názvy souborů mohou být zadávány relativně vůči tomuto adresáři. Několikanásobným použitím volby directory lze zadat i více adresářů. Podle standardů souborového systému Linuxu by tímto adresářem měl být adresář `/var/named`.

`primary` Tato volba používá jako parametry název domény a název souboru a nastavuje server jako autoritativní primární server této domény. Zónová data se nahrají ze zadaného hlavního souboru. Obecně bude v souboru named.boot vždy minimálně jedna položka s volbou primary a tou bude zpětné mapování sítě 127.0.0.0, což je lokální síť.

secondary Tato volba používá jako parametr název domény, seznam adres a název souboru. Nastavuje server jako sekundární server dané domény. Sekundární server také uchovává autoritativní údaje o doméně. Nezískává je však ze souborů, ale snaží se je stáhnout z primárního serveru. Proto musí být v seznamu adres uveden minimálně jeden primární server. Místní server bude kontaktovat jeden po druhém, dokud se mu nepodaří úspěšný přenos zónové databáze, která se pak uloží jako záložní soubor se zadaným názvem. Neodpovídá-li ani jeden z primárních serverů, použijí se data získaná ze záložních souborů. Program **named** se v pravidelných intervalech pokouší obnovovat data zóny. Tato problematika je vysvětlena dále společně se zdrojovými záznamy typu SOA.

cache Tato volba má jako parametry doménu a název souboru. Soubor obsahuje seznam záznamů ukazujících na kořenové jmenné servery. Jsou rozlišovány pouze záznamy typů NS a A. Parametr doména je obvykle název kořenové domény, tedy prostá tečka. Tyto informace jsou pro program **named** nezbytné. Pokud nebude příkaz cache v konfiguračním souboru uveden, program si nevytvoří vlastní lokální vyrovnávací paměť. To způsobí výrazné snížení výkonu a zvýšení zatížení sítě v případě, že se nadřazený server nenachází v místní síti. Kromě toho se nebude moci program **named** spojit s žádným kořenovým jmenným serverem, a tak nebude moci překládat jiné adresy kromě těch, pro něž je autoritativním serverem. (Výjimku z tohoto pravidla představují takzvané forwardery, o nichž hovoříme dále.)

forwarders Tato volba používá jako parametr seznam adres. IP adresy v seznamu určují seznam jmenných serverů, kterých se může program **named** dotazovat v případě, že se mu nepodaří vyřešit dotaz pomocí své místní vyrovnávací paměti. Jmenné servery jsou v příslušném pořadí neustále dotazovány, dokud některý z nich neodpoví na dotaz.

slave Tento příkaz označí jmenný server jako *podřízený* server. To znamená, že nikdy nebude sám provádět rekurzivní dotazy, ale bude je všechny postupovat na servery definované volbou forwarders.

Ještě jsme neuvedli dvě další volby, sortlist a domain. Kromě toho existují ještě dvě direktivy, které lze použít v databázových souborech zón. Jde o direktivy $INCLUDE a $ORIGIN. Protože jsou používány jen velmi zřídka, nebudeme se jimi dále zabývat.

Konfigurační soubor named.conf programu BIND verze 8

V programu BIND verze 8 byla uvedena řada nových funkcí, a proto byla zvolena i nová syntaxe konfiguračního souboru. Soubor named.boot s jednořádkovými konfiguračními příkazy byl nahrazen souborem named.conf, který používá podobnou strukturu jako program **gated**, jež trochu připomíná kód jazyka C.

Nová syntaxe je složitější, existuje však naštěstí nástroj, který umožňuje konverzi starých konfiguračních souborů do nového formátu. Ve zdrojovém balíku BIND 8 existuje skript v Perlu, který se jmenuje **named-bootconf.pl** a jenž přečte stávající soubor named.boot ze standardního vstupu

a konvertuje jej na ekvivalentní soubor named.conf na standardní výstup. Abyste mohli skript použít, musíte mít nainstalován interpret Perlu.

Skript se používá asi následujícím způsobem:

```
# cd /etc
# named-bootconf.pl <named.boot >named.conf
```

Program pak vygeneruje soubor named.conf, který bude vypadat podobně jako v příkladu 6.9. Odstranili jsme různé vysvětlující komentáře, které skript v souboru vytváří, abychom ukázali prakticky přímou souvislost nové a staré syntaxe.

Příklad 6.9 – Konfigurační soubor verze BIND 8

```
//
// Soubor /etc/named.boot brány vlager.vbrew.com
options {
        directory "/var/named";
} ;

zone "." {
        type hint;
        file "named.ca";
} ;

zone "vbrew.com" {
        type master;
        file "named.hosts";
} ;

zone "0.0.127.in-addr.arpa" {
        type master;
        file "named.local";
} ;

zone "16.172.in-addr.arpa" {
        type master;
            file "named.rev";
} ;
```

Když se podíváte pečlivě, zjistíte, že každý jeden řádek souboru named.boot je v souboru named.conf převeden na příkaz uzavřený ve stylu jazyka C ve složených závorkách, { a } .

Komentáře uváděné v souboru named.boot středníkem jsou nyní označeny dvěma lomítky.

Příkaz *directory* byl převeden na blok *options* s klauzulí *directory*.

Příkazy *cache* a *primary* byly převedeny na bloky *zone* s klauzulí *type* o hodnotě *hint*, respektive *master*.

Samotné zónové soubory není třeba upravovat, jejich syntaxe se nezměnila.

Tato nová syntaxe konfiguračního souboru umožňuje použití celé řady různých nových voleb, o kterých jsme nemluvili. Pokud vás tyto nové možnosti zajímají, nejlepším zdrojem informací je dokumentace k balíku BIND, která je dodávána s jeho zdrojovými texty.

Databázové soubory systému DNS

Hlavní soubory, které využívá program **named**, například soubor `named.hosts`, jsou vždy sdruženy s nějakou doménou. Tato doména se nazývá *počátek*. Název domény je zadán parametry příkazů cache a primary. V rámci hlavního souboru můžete zadávat názvy domén a hostitelů relativně vůči této doméně. Název uvedený v konfiguračním souboru je považován za *absolutní*, pokud končí tečkou, v opačném případě je považován za relativní vůči počátku. Vlastní počátek může být odkazován symbolem @.

Všechna data obsažená v hlavním souboru jsou rozdělena do takzvaných *zdrojových záznamů*, RR. Vytvářejí nejmenší jednotky informace dostupné pomocí systému DNS. Každý zdrojový záznam je určitého typu. Například záznamy typu A mapují název hostitele na IP adresu a záznam typu CNAME přiřazuje přezdívky hostitele oficiálnímu názvu hostitele. Zajímá-li vás příklad, podívejte se na výpis 6.11, který představuje hlavní soubor `named.hosts` virtuálního pivovaru.

Položky zdrojových záznamů v hlavních souborech používají společný formát:

```
[doména] [ttl] [třída] typ rdata
```

Pole jsou navzájem oddělena pomocí mezer nebo tabulátorů. Položka může pokračovat na několika řádcích, pokud na konci prvního řádku uvedete levou závorku a pravou závorku umístíte na konec záznamu. Středníky představují začátek komentáře a cokoliv až do konce řádku se ignoruje. Následuje popis jednotlivých částí záznamu:

doména — Název domény, ke které se záznam vztahuje. Není-li uveden žádný název domény, bude se předpokládat, že se záznam vztahuje k doméně uvedené v předchozím záznamu.

ttl — Aby bylo po uplynutí určité doby možno donutit resolver k vyřazení určitých informací, je ke každému záznamu RR přiřazena životnost, *ttl*. Pole ttl udává čas v sekundách, po který budou ještě informace po stažení ze serveru považovány za platné. Čas ttl je desítkové číslo s maximálně osmi číslicemi. Nezadáte-li žádný časový údaj, bude jeho implicitní hodnota rovna hodnotě pole *minimum* předcházejícího záznamu SOA.

třída — Tato volba určuje třídu adresy, například třídu IN pro IP adresy nebo HS pro adresy třídy Hesiod. U sítí založených na protokolu TCP/IP se používá třída IN. Není-li třída zadána, použije se třída z předchozího záznamu typu RR.

typ — Tato volba popisuje typ záznamu RR. Nejběžnějšími typy jsou záznamy A, SOA, PTR a NS. V následující stati budeme popisovat různé typy záznamů RR.

rdata — Tato položka obsahuje data záznamu RR. Formát tohoto pole závisí na typu záznamu. Dále si popíšeme data používaná u jednotlivých typů záznamů.

Následuje neúplný seznam typů zdrojových záznamů, které lze použít v hlavních souborech systému DNS. Existují i další typy, ale těmi se zabývat nebudeme, protože se jedná o experimentální typy s velmi řídkým využitím.

SOA — Tento typ záznamu definuje zónu (SOA znamená Start of Authority). Tento záznam signalizuje, že následující záznamy budou obsahovat administrativní informace o doméně. Každý hlavní soubor definovaný příkazem *primary* musí obsahovat záznam typu SOA této zóny. Zdrojová data obsahují následující pole:

origin — Kanonický název hostitele primárního jmenného serveru této domény. Obvykle je zadán jako absolutní název.

contact — E-mailová adresa osoby odpovědné za správu domény, u které je znak @ nahrazen tečkou. Je-li například ve virtuálním pivovaru odpovědnou osobou **janet**, potom bude toto pole obsahovat adresu *janet.vbrew.com*.

serial — Číslo verze souboru zóny vyjádřené jednou desítkovou číslicí. Kdykoliv v souboru zóny změníte data, měli byste inkrementovat i toto číslo. Klasická konvence obsahuje datum aktualizace společně s číslem verze pro případ více aktualizací v jednom dni, například 2000022600 je první aktualizace dne 26. 2. 2000. Sériová čísla používají sekundární jmenné servery k rozpoznávání změn v informacích o zónách. Aby měly sekundární servery aktuální informace, v pravidelných intervalech si po primárním serveru vyžádají záznam SOA a porovnávají jeho sériové číslo s číslem, které je obsaženo v jejich kopii zónového souboru. Změnilo-li se číslo, potom sekundární servery přenesou z primárního serveru celou databázi zóny.

refresh — Tato volba udává interval v sekundách, po který mají sekundární servery čekat, než provedou opětovnou kontrolu záznamu typu SOA s primárním serverem. Opět se jedná o desítkové číslo s maximálně osmi číslicemi. Síťová topologie se obecně příliš často nemění, takže by toto číslo mělo v rozsáhlejších sítích odpovídat zhruba dnům a v menších sítích by tento interval měl být ještě delší.

retry — Toto číslo určuje interval, po jehož uplynutí by se měl sekundární server znovu spojit s primárním serverem, když se nepodaří nějaký požadavek nebo aktualizace zónových informací. Tento interval by neměl být příliš malý, jinak by dočasný výpadek serveru nebo nějaký síťový problém mohl způsobit obrovské plýtvání se síťovými zdroji ze strany sekundárních serverů. Vhodnou hodnotou pro tento interval je jedna hodina nebo půl hodiny.

expire — Tato volba udává čas v sekundách, po jehož uplynutí by měl server skartovat všechna data o zónách, pokud se mu během tohoto intervalu nepodařilo spojit s primárním serverem. Normálně by měla být tato hodnota hodně velká, alespoň týden (604 800 sekund), nicméně její prodloužení až na měsíc může být také užitečné.

minimum — Toto je implicitní hodnota ttl zdrojových záznamů, které nemají definován vlastní ttl. Volba přikazuje ostatním jmenným serverům, aby po předem dané době zrušily záznam ve vyrovnávací paměti. Tato hodnota nemá nic společného s časem, po kterém se budou snažit sekundární servery o aktualizaci svých informací. Hodnota by měla být dostatečně vysoká, zejména u sítí typu LAN, u nichž se prakticky nemění síťová topologie. Vhodné je použít hodnotu týden nebo dokonce měsíc. V případech, kde se některé záznamy mění často, můžete pro ně nastavit vlastní ttl.

A — Tento typ záznamu přiřazuje IP adresu k názvu hostitele. Pole zdrojových dat obsahuje adresu respektující tečkovou notaci. Každý hostitel musí mít pouze jeden záznam typu A. Název hostitele uvedený v tomto záznamu je považován za oficiální, neboli *kanonický* název hostitele. Všechny další názvy téhož hostitele jsou přezdívky a pomocí záznamu typu CNAME se mapují na kanonický název. Je-li kanonické jméno našeho hostitele **vlager**, uvedeme toto jméno v záznamu A společně s adresou hostitele. Budeme-li chtít téže adrese přiřadit další jména, například **news**, vytvoříme záznam typu CNAME, který bude asociovat alternativní jméno a kanonické jméno. O záznamech CNAME budeme hovořit za chvíli.

NS — Tento typ záznamu definuje primární a všechny sekundární jmenné servery zóny. Záznam ukazuje na hlavní jmenný server zóny, datová část obsahuje kanonické jméno tohoto serveru. Záznamy typu NS potkáváme ve dvou situacích: První z nich je případ, kdy delegujeme pravomoc na podřízenou zónu, druhý je v hlavní databázi samotné podřízené zóny. Skupiny serverů specifikované v nadřízené i podřízené zóně si musí odpovídat. Záznam typu NS definuje názvy primárního a sekundárních serverů zóny. Abychom tyto názvy mohli použít, musíme je být schopni převést na adresy. Často ovšem server patří přímo do té zóny, kterou obsluhuje a dostáváme tak klasický problém „kuřete a vejce“: adresu serveru nezjistíme, dokud se na ni serveru nezeptáme a nemůžeme se ho zeptat, dokud neznáme jeho adresu. K vyřešení tohoto dilematu musíme zavést speciální záznamy typu A přímo v databázi nadřazené zóny. Záznam A umožní serveru rodičovské zóny zjistit adresu jmenného serveru podřízené zóny. Tyto záznamy se často označují jako *tmelicí záznamy*, protože fungují jako „tmel“, který propojuje podřízenou zónu s jejím rodičem.

CNAME — Záznam přiřazuje kanonickému názvu hostitele přezdívku. Definuje alternativní názvy, kterými se můžeme odkazovat na hostitele, jehož kanonické jméno je uvedeno jako parametr. Kanonický název hostitele je ten název, pro který existuje v hlavní databázi záznam typu A; přezdívky jsou s tímto názvem jednoduše spojeny pomocí záznamu typu CNAME, ale jinak se k nim žádné další záznamy nevztahují.

PTR — Tento typ záznamu slouží ke sdružení názvů v doméně **in-addr.arpa** s názvy hostitelů. Záznam se používá ke zpětnému mapování IP adres na názvy hostitelů. Zadaný název hostitele musí být v kanonickém tvaru.

MX — Tento záznam definuje *poštovní server* dané domény. O poštovních serverech budeme hovořit v kapitole 17, *Směrování pošty na Internetu*. Syntaxe záznamu typu MX je: [*doména*] [*ttl*] [*třída*] MX *preference hostitel* Parametr *hostitel* definuje název poštovního serveru domény. Každému serveru odpovídá jeho *preference*. Poštovní agent snažící se o doručení pošty do domény se pokusí použít všechny hostitele definované záznamy MX cílové domény do doby, než u jednoho z nich uspěje. Pořadí, ve kterém je používá, závisí právě na jejich preferenci: první se zkusí hostitel s nejmenší preferencí a v případě neúspěchu postupně další a další podle rostoucí preference.

HINFO — Tento typ záznamu poskytuje informace o hardwaru a softwaru daného systému. Jeho syntaxe je: [*doména*] [*ttl*] [*třída*] HINFO *hardware software* Údaj *hardware* určuje typ hardwaru, který daný hostitel používá. Pro jeho specifikaci existují konvence, seznam platných „názvů hardwarových platforem“ definuje dokument RFC 1700, Assigned Numbers. Pokud údaj obsahuje mezeru, musí být uzavřen v uvozovkách. Údaj *software* obsahuje používaný operační systém. Opět by měl být vybrán korektní název podle dokumentu Assigned Numbers[58]. Záznam HINFO popisující linuxový stroj na platformě Intel by vypadal takto: tao 36500 IN HINFO IBM-PC LINUX2.2 Záznamy HINFO popisující linuxový stroj na platformách Motorola-68000 budou vypadat takto: cevad 36500 IN HINFO ATARI-104ST LINUX2.0 jedd 36500 IN HINFO AMIGA-3000 LINUX2.0

58 Pozn. překladatele: Záznamy typu `HINFO` prakticky nikdo nepoužívá. Jsou to čistě informativní záznamy – jejich údaje ani DNS ani nikdo jiný k ničemu nepotřebuje. Řada lidí se domnívá, že zveřejněním hardwarové a softwarové platformy svých počítačů mohou usnadnit situaci útočníkům, kteří tak budou moci vést cílenější útok, jelikož přesně vědí, na co útočí.

Konfigurace typu „caching-only“

Existuje jeden speciální typ konfigurace programu **named**, o němž jsme se zmínili před tím, než jsme se pustili do podrobnějšího výkladu. Označuje se jako konfigurace *caching-only*. V této konfiguraci server neobsluhuje žádnou doménu, funguje však jako „zpracovatel“ všech DNS dotazů, které vaše počítače generují. Výhodou tohoto řešení je, že si server vybuduje vyrovnávací paměť a konkrétním jmenným serverům na Internetu se tak posílá vždy pouze první dotaz na určitého hostitele. Další stejné dotazy zodpoví přímo lokální server podle své vyrovnávací paměti. V této chvíli to nevypadá moc užitečně, výhody zjistíme při použití telefonického připojení k Internetu, které popisujeme v kapitolách 7 a 8.

Soubor named.boot serveru v režimu caching-only bude vypadat nějak takto:

```
; Soubor named.boot caching-only serveru
directory                          /var/named
primary       0.0.127.in-addr.arpa   named.local ; síť localhost
cache         .                      named.ca    ; kořenové servery
```

Kromě souboru named.boot musíte ještě vytvořit soubor named.ca, který bude obsahovat platný seznam kořenových serverů. Můžete jej vytvořit podle příkladu 6.10. Žádné další soubory nejsou v této konfiguraci zapotřebí.

Sestavení hlavních souborů

Příklady 6.10, 6.11, 6.12 a 6.13 obsahují příklady hlavních souborů jmenného serveru pivovaru na počítači **vlager**. Vzhledem k povaze diskutované sítě (jednoduchá síť typu LAN) je příklad poměrně průhledný.

Soubor named.ca, který vidíte v příkladu 6.10, ukazuje záznamy pro kořenové jmenné servery. Typický takovýto soubor obvykle definuje zhruba deset jmenných serverů. Aktuální seznam jmenných serverů kořenové domény můžete získat pomocí nástroje **nslookup**, který bude popsán v další části kapitoly[59].

Příklad 6.10 – Soubor named.ca

```
;
; /var/named/named.ca
;   Soubor kořenových serverů pro pivovar. Nejsme na
;   Internetu, tak je nepotřebujeme. V případě potřeby
;   stačí záznamy aktivovat odstraněním středníků.
;
;.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
;A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;.                        3600000      NS    B.ROOT-SERVERS.NET.
;B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;.                        3600000      NS    C.ROOT-SERVERS.NET.
;C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;.                        3600000      NS    D.ROOT-SERVERS.NET.
;D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
;.                        3600000      NS    E.ROOT-SERVERS.NET.
;E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
```

59 Uvědomte si, že se nemůžete vašeho jmenného serveru zeptat na kořenové servery, pokud v něm alespoň nějaké nemáte definovány. Problém můžete vyřešit tak, že se programem **nslookup** zeptáte jiného jmenného serveru, nebo použijete jako základ soubor 6.10 a pomocí něj získáte aktuální platný seznam všech kořenových serverů.

```
;.                        3600000      NS    F.ROOT-SERVERS.NET.
;F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
;.                        3600000      NS    G.ROOT-SERVERS.NET.
;G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;.                        3600000      NS    H.ROOT-SERVERS.NET.
;H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;.                        3600000      NS    I.ROOT-SERVERS.NET.
;I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;.                        3600000      NS    J.ROOT-SERVERS.NET.
;J.ROOT-SERVERS.NET.      3600000      A     198.41.0.10
;.                        3600000      NS    K.ROOT-SERVERS.NET.
;K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
;.                        3600000      NS    L.ROOT-SERVERS.NET.
;L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;.                        3600000      NS    M.ROOT-SERVERS.NET.
;M.ROOT-SERVERS.NET.      3600000      A     202.12.27.33
;
```

Příklad 6.11 – Soubor named.hosts

```
;
; /var/named/named.hosts
;   Lokální hostitelé pivovaru, doména vbrew.com
;
@               IN  SOA   vlager.vbrew.com. janet.vbrew.com. (
                          2000012601 ; sériové číslo
                          86400      ; obnovení: denně
                          3600       ; opakování: co hodinu
                          3600000    ; zneplatnění:  42 dní
                          604800     ; minimální ttl: týden
                          )
                IN  NS    vlager.vbrew.com.
;
; poštu obsahuje vlager
                IN  MX    10 vlager
;
; lokální adresa
localhost.      IN  A     127.0.0.1
;
; Ethernet pivovaru
vlager          IN  A     172.16.1.1
vlager-if1      IN  CNAME vlager
; vlager je rovněž server pro konference
news            IN  CNAME vlager
vstout          IN  A     172.16.1.2
vale            IN  A     172.16.1.3
;
; Ethernet vinařství
vlager-if2      IN  A     172.16.2.1
vbardolino      IN  A     172.16.2.2
vchianti        IN  A     172.16.2.3
vbeaujolais     IN  A     172.16.2.4
;
```

```
; (podřízený) Ethernet palírny
vbourbon         IN  A     172.16.3.1
vbourbon-if1     IN  CNAME vbourbon
```

Příklad 6.12 – Soubor named.local

```
;
; /var/named/named.local
;   Reverzní mapování 127.0.0 - doména 0.0.127.in-addr.arpa.
;
;
@             IN  SOA   vlager.vbrew.com. joe.vbrew.com. (
                        1          ; sériové číslo
                        360000     ; obnovení: 100 hodin
                        3600       ; opakování: co hodinu
                        3600000    ; zneplatnění:  42 dní
                        604800     ; minimální ttl: 100 hodin
)
              IN  NS    vlager.vbrew.com.
1             IN  PTR   localhost.
```

Příklad 6.13 – Soubor named.rev

```
;
; /var/named/named.rev
;   Reverzní mapování našich IP adres, doména 16.172.in-addr.arpa.
;
@             IN  SOA   vlager.vbrew.com. joe.vbrew.com. (
                          16         ; sériové číslo
                          86400      ; obnovení: denně
                          3600       ; opakování: co hodinu
                          3600000    ; zneplatnění:  42 dní
                          604800     ; minimální ttl: týden
                          )
              IN  NS    vlager.vbrew.com.
; pivovar
1.1           IN  PTR   vlager.vbrew.com.
2.1           IN  PTR   vstout.vbrew.com.
3.1           IN  PTR   vale.vbrew.com.
; vinařství
1.2           IN  PTR   vlager-if2.vbrew.com.
2.2           IN  PTR   vbardolino.vbrew.com.
3.2           IN  PTR   vchianti.vbrew.com.
4.2           IN  PTR   vbeaujolais.vbrew.com.
```

Kontrola nastavení jmenného serveru

Pro kontrolu činnosti jmenného serveru existuje vynikající nástroj **nslookup**. Lze jej používat jak interaktivně, tak i z příkazové řádky. Ve druhém případě ho spustíte takto:

```
$ nslookup hostname
```

Program se dotáže jmenného serveru zadaného v souboru resolv.conf na adresu hostitele, jehož název jsme zadali. (Je-li v souboru resolv.conf uveden více než jeden server, pak **nslookup** náhodně zvolí jeden z nich.)

Interaktivní režim je však mnohem zajímavější. Kromě vyhledávání jednotlivých hostitelů se můžete dotazovat na libovolný typ záznamu systému DNS a dále můžete přenést celou databázi zóny.

Spustíte-li **nslookup** bez parametrů, zobrazí používaný jmenný server a přepne se do interaktivního režimu. Na výzvu „>“ můžete zadat libovolný název domény, na který by se měl program dotazovat. Implicitně se ptá na záznamy typu A, což jsou ty, které obsahují IP adresy v dané doméně.

Tento typ je možné změnit příkazem:

```
> set type = typ
```

kde parametr *typ* může být jeden z již popsaných typů záznamů, nebo klíčové slovo ANY.

Typická ukázka práce s programem **nslookup** může vypadat takto:

```
$ nslookup
Default Server:  tao.linux.org.au
Address:  203.41.101.121

> metalab.unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

Name:    metalab.unc.edu
Address:  152.2.254.81

>
```

V odpovědích je nejprve uvedeno, kterého serveru jsme se ptali a pak odpověď na náš dotaz.

Pokusíte-li se dotazovat na název, který nemá přidělenu žádnou IP adresu, ale v databázi systému DNS byly nalezeny jiné záznamy, vrátí příkaz **nslookup** chybovou zprávu „No type A records found“. Programem **nslookup** je ale možné dotazovat se i na jiné typy záznamů, stačí typ změnit příkazem **set type**. Záznam SOA domény **unc.edu** získáme například takto:

```
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

*** No address (A) records available for unc.edu
> set type=SOA
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

unc.edu

        origin = ns.unc.edu

        mail addr = host-reg.ns.unc.edu
        serial = 1998111011
        refresh = 14400 (4H)
```

```
        retry   = 3600 (1H)
        expire  = 1209600 (2W)
        minimum ttl = 86400 (1D)
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
unc.edu name server = ns.unc.edu
ns2.unc.edu     internet address = 152.2.253.100
ncnoc.ncren.net internet address = 192.101.21.1
ncnoc.ncren.net internet address = 128.109.193.1
ns.unc.edu      internet address = 152.2.21.1
```

Podobně se můžeme zeptat i na záznamy typu MX:

```
> set type=MX
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

unc.edu preference = 0, mail exchanger = conga.oit.unc.edu
unc.edu preference = 10, mail exchanger = imsety.oit.unc.edu
unc.edu name server = ns.unc.edu
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
conga.oit.unc.edu       internet address = 152.2.22.21
imsety.oit.unc.edu      internet address = 152.2.21.99
ns.unc.edu      internet address = 152.2.21.1
ns2.unc.edu     internet address = 152.2.253.100
ncnoc.ncren.net internet address = 192.101.21.1
ncnoc.ncren.net internet address = 128.109.193.1
```

Zadáme-li typ ANY, budou vráceny všechny záznamy libovolného typu, které se k požadovanému názvu vztahují.

Praktickou aplikací programu **nslookup** je kromě testování jmenného serveru i získání aktuálního seznamu kořenových jmenných serverů. Lze to provést tak, že se zeptáte na záznamy typu NS, odpovídající kořenové doméně:

```
> set type=NS
> .
Server:  tao.linux.org.au
Address:  203.41.101.121

Non-authoritative answer:
(root)  name server = A.ROOT-SERVERS.NET
(root)  name server = H.ROOT-SERVERS.NET
(root)  name server = B.ROOT-SERVERS.NET
(root)  name server = C.ROOT-SERVERS.NET
(root)  name server = D.ROOT-SERVERS.NET
(root)  name server = E.ROOT-SERVERS.NET
(root)  name server = I.ROOT-SERVERS.NET
(root)  name server = F.ROOT-SERVERS.NET
(root)  name server = G.ROOT-SERVERS.NET
(root)  name server = J.ROOT-SERVERS.NET
(root)  name server = K.ROOT-SERVERS.NET
(root)  name server = L.ROOT-SERVERS.NET
(root)  name server = M.ROOT-SERVERS.NET
```

```
Authoritative answers can be found from:
A.ROOT-SERVERS.NET      internet address = 198.41.0.4
H.ROOT-SERVERS.NET      internet address = 128.63.2.53
B.ROOT-SERVERS.NET      internet address = 128.9.0.107
C.ROOT-SERVERS.NET      internet address = 192.33.4.12
D.ROOT-SERVERS.NET      internet address = 128.8.10.90
E.ROOT-SERVERS.NET      internet address = 192.203.230.10
I.ROOT-SERVERS.NET      internet address = 192.36.148.17
F.ROOT-SERVERS.NET      internet address = 192.5.5.241
G.ROOT-SERVERS.NET      internet address = 192.112.36.4
J.ROOT-SERVERS.NET      internet address = 198.41.0.10
K.ROOT-SERVERS.NET      internet address = 193.0.14.129
L.ROOT-SERVERS.NET      internet address = 198.32.64.12
M.ROOT-SERVERS.NET      internet address = 202.12.27.33
```

Kompletní seznam příkazů programu **nslookup** získáte tak, že mu zadáte příkaz **help**.

Další užitečné nástroje

Existuje několik dalších nástrojů, které vám při správě služby BIND mohou pomoci. Krátce si popíšeme některé z nich. Podrobnější informace o jejich použití získáte v nápovědě, která se s nimi dodává.

Nástroj **hostcvt** pomáhá při prvotní konfiguraci služby BIND. Provádí konverzi souboru `/etc/hosts` do hlavních souborů pro program named. Vygeneruje data jak pro přímé mapování (typ záznamu A), tak i pro zpětné mapování (typ záznamu PTR) a postará se i o přezdívky. Samozřejmě, že nemůže udělat vše, protože si sami budete chtít nastavit časovací konstanty záznamu SOA nebo vytvořit záznam MX. Přesto vám však může ušetřit dost starostí. Nástroj **hostcvt** je částí zdrojového kódu služby BIND, ale je možné ho nalézt i jako samostatný balík na některých linuxových FTP serverech.

Jakmile nakonfigurujete vlastní jmenný server, budete si ho určitě chtít otestovat. Existuje několik dobrých nástrojů, které vám testování usnadní: jedním z nich je **dnswalk**, napsaný v Perlu. Dalším je **nslint**. Oba projdou databáze DNS serveru, hledají v nich obvyklé chyby a kontrolují, zda jsou údaje konzistentní. Dalšími užitečnými nástroji jsou **host** a **dig**, což jsou obecné programy pro dotazování se systému DNS. Pomocí nich můžete ručně zkoumat a diagnostikovat záznamy DNS.

Všechny nástroje jsou snadno dostupné v předpřipravených balících: **dnswalk** a **nslint** najdete na adresách **http://www.visi.com/~barr/dnswalk/** a **ftp://ftp.ee.lbl.gov/nslint.tar.Z**. Zdrojové kódy programů **host** a **dig** pak na adresách **ftp://ftp.nikhef.nl/pub/network/** a **ftp://ftp.is.co.za/networking/ip/dns/dig/**.

KAPITOLA 7

Linka SLIP

Paketové protokoly jako IP nebo IPX spolehají na to, že přijímající hostitel pozná v přijímaném datovém proudu začátek a konec každého paketu. Mechanismus používaný k detekci začátků a konců paketů se označuje jako *deliminace*. V lokální síti tento mechanismus implementuje Ethernet, na sériových komunikačních linkách jej implementují protokoly SLIP a PPP.

Relativně nízké ceny pomalých vytáčených linek a rychlejších telefonních okruhů vedly k velkému rozšíření sériové komunikace protokolem IP, zejména pro zajištění konektivity k Internetu koncovým uživatelům. Hardware potřebný pro provoz protokolů SLIP nebo PPP je jednoduchý a široce dostupný. Potřebujeme pouze modem a sériový port vybavený portem FIFO.

Protokol SLIP je implementačně velmi jednoduchý a jistou dobu byl z obou zmíněných protokolů výrazně používanější. Dnes už většina uživatelů preferuje protokol PPP. Tento protokol nabízí celou řadu užitečných funkcí, které se zasloužily o jeho oblíbenost. O nejdůležitějších z nich se později zmíníme.

Jádro Linuxu obsahuje ovladače protokolu SLIP i PPP. Oba ovladače se už nějakou dobu používají a jsou stabilní a spolehlivé. V této a následující kapitole budeme hovořit o obou protokolech a o tom, jak je nakonfigurovat.

Obecné požadavky

Abyste mohli používat protokoly SLIP nebo PPP, bude třeba nastavit základní síťové funkce, které byly popsány v předchozích kapitolách. Přinejmenším musíte nastavit lokální rozhraní a povolit službu pro rozlišení názvů. Když se budete připojovat k Internetu, budete samozřejmě chtít používat systém DNS. Zde máte dvě možnosti: buď používat DNS přes sériovou linku a v souboru `/etc/resolv.conf` nastavit IP adresu DNS serveru vašeho poskytovatele, nebo si postupem popsaným v kapitole 6 nakonfigurovat caching-only DNS server.

Použití protokolu SLIP

Dial-up servery často nabízejí službu SLIP prostřednictvím speciálních uživatelských účtů. Po přihlášení k takovému účtu nejste vpuštěni do obecného uživatelského rozhraní; namísto toho je spuštěn program nebo skript, který povolí na serveru pro danou sériovou linku ovladač SLIP a nakonfiguruje patřičné síťové rozhraní. Totéž se musí provést na vaší straně spojení.

V některých operačních systémech je ovladač SLIP speciálním uživatelským programem; v operačním systému Linux je součástí jádra systému, takže je výrazně rychlejší. Je však nutné, aby byla sériová linka explicitně převedena do režimu SLIP. To se provede pomocí speciálního režimu linky, SLIPDISC. Je-li zařízení tty v normálním režimu (DISC0), probíhá přenos dat uživatelským procesem voláními `read(2)` a `write(2)` a ovladač SLIP nebude schopen zapisovat nebo číst ze za-

řízení tty. V režimu SLIPDISC jsou role obráceny: nyní nebude moci zapisovat nebo číst ze zařízení žádný uživatelský proces, ale všechna data přicházející na sériový port budou přímo předána ovladači SLIP.

Vlastní ovladač protokolu SLIP rozumí množství variací protokolu SLIP. Kromě běžného protokolu SLIP ovládá také protokol CSLIP, který u odcházejících IP paketů provádí takzvanou Van Jacobsonovu kompresi hlaviček (viz RFC 1144). To výrazně zvyšuje propustnost dat při interaktivní práci. Mimoto existují i šestibitové verze obou protokolů.

Jednoduchý způsob, jak přepnout sériovou linku do režimu SLIP, spočívá ve využití nástroje **slattach**. Předpokládejme, že máte modem na zařízení `/dev/ttyS3` a že jste se úspěšně přihlásili k serveru SLIP. Potom spustíte následující příkaz:

```
# slattach /dev/ttyS3 &
```

Tento příkaz přepne režim zařízení ttyS3 na SLIPDISC a připojí je k jednomu ze síťových rozhraní SLIP. Je-li to vaše první aktivní spojení pomocí protokolu SLIP, bude linka připojena k rozhraní sl0; další bude připojena k rozhraní sl1 a tak dále. Současné verze jader operačního systému podporují až 256 současných připojení pomocí protokolu SLIP.

Implicitní režim nastavený příkazem **slattach** bude CSLIP. K volbě některého jiného režimu lze použít parametr –p. Chcete-li používat standardní protokol SLIP (bez komprese), zadejte:

```
# slattach -p slip /dev/ttyS3 &
```

Další možné režimy jsou uvedeny v tabulce 7.1. Kromě toho existuje speciální pseudorežim adaptive, při němž jádro provádí automatickou detekci toho, jaký režim se používá na druhém konci linky.

Tabulka 7.1 – Režimy linky SLIP v Linuxu

Režim	Popis
slip	Klasický protokol SLIP.
cslip	Protokol SLIP s Van Jacobsonovou kompresí hlaviček.
slip6	Protokol SLIP s šestibitovým kódováním. Použitá kódovací metoda se podobá příkazu uuencode a převádí všechny datagramy na tisknutelné znaky. Tato konverze je užitečná, pokud nemáte sériovou linku se správně fungujícím osmým bitem.
cslip6	Protokol SLIP s Van Jacobsonovou kompresí hlaviček a šestibitovým kódováním.
adaptive	Nejedná se o samostatný režim linky, jádro se pokusí detekovat režim používaný vzdáleným počítačem a přejde do něj.

Musíte používat stejný režim jako vzdálená strana. Pokud připojený počítač používá řekněme režim CSLIP, vy jej musíte používat také. Pokud spojení nefunguje, jako první ověřte, že obě strany používají stejný komunikační režim. Nevíte-li, co má vzdálený počítač nastaveno, vyzkoušejte adaptivní režim. Jádru se možná podaří správně detekovat potřebný typ.

Příkaz **slattach** umožňuje nejen aktivovat protokol SLIP, ale i jiné sériové protokoly, například PPP nebo KISS (další z protokolů používaných v rádiových sítích). Takovéto jeho použití nicméně není příliš obvyklé, protože podporu zmíněných protokolů zajišťují lepší nástroje. Podrobnosti naleznete na manuálové stránce příkazu slattach(8).

Když linku přepnete do režimu SLIP, musíte nakonfigurovat její síťové rozhraní. K tomu použijeme standardní příkazy **ifconfig** a **route**. Předpokládejme, že jsme se z brány **vlager** připojili k serveru **cowslip**. Pak je nutné spustit následující sekvenci příkazů:

```
# ifconfig sl0 vlager-slip pointopoint cowslip
# route add cowslip
# route add default gw cowslip
```

První příkaz nastaví rozhraní jako point-to-point spojení s hostitelem **cowslip**, druhý a třetí příkaz přidají směrovací záznamy na hostitele **cowslip** a implicitní trasu přes **cowslip**.

U příkazu **ifconfig** je vhodné všimnout si dvou věcí: Parametr pointopoint specifikuje adresu vzdáleného konce linky, pro místní rozhraní SLIP používáme název vlager-slip.

Už jsme říkali, že rozhraní SLIP může pracovat se stejnou adresou jako ethernetové rozhraní brány **vlager**. Název **vlager-slip** pak bude pouhým aliasem IP adresy 172.16.1.1. Druhá možnost je přiřadit lince SLIP samostatnou adresu. Používá se to zejména v případě, že lokální síť používá neregistrované IP adresy tak, jak je tomu v síti pivovaru. O této problematice budeme podrobněji hovořit v další části.

Ve zbytku kapitoly budeme lokální SLIP rozhraní brány **vlager** označovat názvem **vlager-slip**.

Při rušení spojení pomocí protokolu SLIP musíte nejprve odstranit všechny trasy přes hostitele **cowslip**. K tomu slouží příkaz **route** s parametrem del. Potom je nutné odpojit rozhraní a poslat programu **slattach** signál HUP. Nakonec je třeba zavěsit modem pomocí terminálového programu:

```
# route del default
# route del cowslip
# ifconfig sl0 down
# kill -HUP 516
```

Hodnotu *516* je nutné nahradit příslušným identifikátorem procesu **slattach**, který chcete ukončit. (Identifikátor zjistíte například příkazem **ps ax**.)

Práce s privátními IP sítěmi

V kapitole 5 jsem se zmiňoval, že pivovar používá ethernetovou síť s neregistrovanými IP adresami, které jsou rezervovány pouze pro interní použití. Pakety z nebo na takovou síť se v Internetu nesměrují. Pokud bychom se přes bránu **vlager** připojili na počítač **cowslip** a brána by fungovala jako směrovač sítě pivovaru, počítače na této síti by se nemohly spojit s „opravdovými" internetovými počítači, protože jejich pakety by byly prvním pořádným směrovačem v tichosti zahozeny.

Abychom tento problém vyřešili, nakonfigurujeme bránu **vlager** jako jakousi „startovací rampu" internetových služeb. Vzhledem k vnějšímu světu se bude chovat jako normální počítač připojený protokolem SLIP s registrovanou IP adresou (kterou jí pravděpodobně přidělí poskytovatel přístupu provozující bránu **cowslip**). Kdokoliv se k bráně **vlager** přihlásí, bude moci používat textově orientované programy jako **ftp**, **telnet** nebo dokonce **lynx** a využívat tak Internet. Kdokoliv v síti pivovaru se tedy může telnetnout a přihlásit k bráně **vlager** a na ní pak spouštět internetové programy. Kvůli uživatelům služby WWW bychom například mohli na bráně provozovat takzvaný *proxy server*, který by předával požadavky uživatelů ze všech počítačů na odpovídající internetové hostitele.

Nutnost přihlásit se k bráně **vlager** ovšem práci s Internetem poněkud komplikuje. Kromě toho, že nám ale ušetřila papírování (a náklady) spojené s registrací celé IP sítě, zároveň nám funguje jako firewall. Firewally jsou vyhrazené počítače, které poskytují uživatelům lokální sítě omezený přístup k Internetu, aniž by zároveň vystavovaly hostitele na interní síti možnostem útoků z vnějšího světa. Konfigurace jednoduchého firewallu je popsána v kapitole 9. V kapitole 11 budeme hovořit o funkci zvané „IP maškaráda“, která představuje mocnou alternativu k proxy serverům.

Předpokládejme, že pivovar má pro svou SLIP linku přidělenu IP adresu 192.168.5.74[60]. Jediné co musíte udělat bude zajistit, aby ve výše popsané konfiguraci byla prostřednictvím souboru `/etc/hosts` tato adresa přidělena rozhraní **vlager-slip**. Samotný postup aktivace SLIP spojení se nijak nezmění.

Použití nástroje dip

Až sem to bylo poměrně jednoduché. Přesto však možná budete chtít výše uvedené kroky zautomatizovat natolik, aby bylo možné celý postup vyvolat jediným příkazem, který otevře sériové zařízení, zavolá modemem k poskytovateli, přihlásí se, přepne linku do režimu SLIP a nakonfiguruje síťové rozhraní. Přesně k tomu účelu slouží nástroj **dip**.

dip znamená *Dialup IP*. Původně jej napsal Fred van Kempen a postupně jej modifikovala celá řada lidí. Dnes existuje jedna verze, kterou používají prakticky všichni: Verze `dip337p-uri` je součástí většiny moderních distribucí Linuxu, kromě toho je k dispozici v FTP archivu **metalab.unc.edu**.

Nástroj **dip** je interpretem jednoduchého skriptového jazyka, který se vám stará o modem, nastavuje linku do režimu SLIP a konfiguruje různá rozhraní. Skriptovací jazyk je velmi mocný a bude vyhovovat většině konfigurací.

Aby mohl **dip** nakonfigurovat rozhraní SLIP, potřebuje práva superuživatele. Mohlo by to svádět k tomu povolit programu setuid root, aby jej mohli všichni uživatelé použít bez toho, že byste superuživatelská práva přidělili přímo jim. To je velmi nebezpečné, protože nastavením falešných rozhraní a implicitních tras je možné zcela rozvrátit provoz sítě. A co je ještě horší, vaši uživatelé tím získají možnost spojení s libovolným serverem SLIP a mohou tak síť vystavit nebezpečným útokům. Pokud tedy chcete umožnit svým uživatelům vytvářet spojení SLIP, napište pro každý plánovaný server SLIP malý wrapper, který teprve bude spouštět **dip** se skriptem pro vytvoření příslušného spojení. Dobře napsanému wrapperu je možné bez obav přidělit superuživatelská práva[61]. Alternativní a pružnější řešení je povolit oprávněným uživatelům superuživatelský přístup k programu **dip** pomocí nástroje jako je **sudo**.

Příklad skriptu

Předpokládejme, že se protokolem SLIP připojujeme k hostiteli **cowslip** a že jsme k vytvoření tohoto spojení napsali pro program **dip** skript nazvaný `cowslip.dip`. Program **dip** budeme spouštět s názvem skriptu jako s parametrem:

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
```

60 Pozn. překladatele: Všimněte si, že adresa 192.168.5.74 je rovněž rezervovaná privátní adresa! To už ale není problém náš, ale problém poskytovatele připojení. Někteří poskytovatelé totiž, aby „neplýtvali“ svými reálnými IP adresami, přidělují klientům privátní adresy a nějakou svou hlavní branou pak zajišťují IP maškarádu. Pokud vás pohoršuje, že platíte a nedostali jste „opravdovou“ adresu, nemáte k tomu důvod. Při dobře nastavené maškarádě je pro vás převod adres zcela transparentní a zadarmo dostáváte výhodu dobré ochrany vašeho počítače před zlým okolním světem. Jste omezeni pouze v tom, že ze svého počítače nemůžete poskytovat vnějšímu světu služby WWW a jiných serverů.

61 Program **diplogin** musí být rovněž spouštěn s setuid root. Viz text na konci kapitoly.

```
Written by Fred N. van Kempen, MicroWalt Corporation.
connected to cowslip.moo.com with addr 192.168.5.74
#
```

Samotný skript je uveden v příkladu 7.1.

Příklad 7.1 – Skript pro program dip

```
# Příklad dip skriptu pro připojení ke cowslip-u
# Nastavení lokálního a vzdáleného jména a adresy
        get $local vlager-slip
        get $remote cowslip
        port ttyS3              # volba sériového portu
        speed 38400             # nastavení maximální rychlosti
        modem HAYES             # nastavení typu modemu
        reset                   # reset modemu a tty
        flush                   # smazání odezvy od modemu
# Příprava na vytáčení
        send ATQ0V1E1X1\ r
        wait OK 2
        if $errlvl != 0 goto error
        dial 41988
        if $errlvl != 0 goto error
        wait CONNECT 60
        if $errlvl != 0 goto error
# Teď jsme připojeni
        sleep 3
        send \ r\ n\ r\ n
        wait ogin: 10
        if $errlvl != 0 goto error
        send Svlager\ n
        wait ssword: 5
        if $errlvl != 0 goto error
        send knockknock\ n
        wait running 30
        if $errlvl != 0 goto error
# Jsme přihlášení, vzdálená strana spouští SLIP.
        print Connected to $remote with address $rmtip
        default                 # Tato linka bude implicitní trasou
        mode SLIP               # Taky přejdeme do režimu SLIP
# ošetření případných chyb
error:
        print SLIP to $remote failed.
```

Po spojení s hostitelem **cowslip** a povolení protokolu SLIP se proces **dip** odpojí od terminálu a poběží na pozadí. Potom můžete začít používat normální síťové služby po lince SLIP. Chcete-li spojení ukončit, vyvolejte nástroj **dip** s parametrem -k. Tento příkaz pošle procesu **dip** signál HUP, přičemž použije záznam identifikačního čísla procesu **dip**, který je uložen v souboru `/etc/dip.pid`:

```
# dip -k
```

Ve skriptovém jazyku nástroje **dip** znamenají klíčová slova uvedená symbolem dolaru názvy proměnných. Nástroj má předdefinovanou skupinu proměnných, které uvádíme níže. Například pro-

měnné *$remote* a *$local* obsahují názvy vzdáleného a místního hostitele, kteří se účastní spojení pomocí protokolu SLIP.

První dva příkazy ve vzorovém skriptu jsou příkazy **get**, které programu **dip** umožňují nastavovat hodnoty proměnných. Zde nastavujeme název místního a vzdáleného počítače na **vlager**, respektive **cowslip**.

Dalších pět řádků nastavuje terminálovou linku a modem. Příkaz **reset** pošle modemu inicializační řetězec. Následující příkaz **flush** odstraní odpověď od modemu, aby mohla správně fungovat následující přihlašovací sekvence. Tato přihlašovací sekvence je poměrně přehledná: nejdříve se vytočí číslo 41 98 8, což je telefonní číslo hostitele **cowslip** a přihlásíme se na účet **Svlager** s heslem **knockknock**. Příkaz **wait** způsobí čekání na řetězec uvedený jako první parametr, druhý parametr udává timeout, po kterém se čekání ukončí, pokud řetězec nedorazí. Příkazy **if** na různých místech přihlašovací procedury průběžně kontrolují, zda nedošlo k nějaké chybě.

Posledními příkazy spouštěnými po přihlášení jsou **default**, který nastaví linku SLIP jako implicitní trasu, a **mode**, jenž nastaví režim linky a nakonfiguruje její rozhraní.

Referenční příručka programu dip

V následujícím textu uvádíme popis většiny příkazů programu **dip**. Přehled všech dostupných příkazů získáte, když nástroj **dip** spustíte v testovacím režimu a zadáte příkaz **help**. Chcete-li zjistit syntaxi konkrétního příkazu, zadejte jej bez parametrů. (Nefunguje to samozřejmě pro příkazy, které parametry nepoužívají.) Následující příklad ilustruje použití nápovědy programu **dip**:

```
# dip -t
DIP: Dialup IP Protocol Driver version 3.3.7p-uri (25 Dec 96)
Written by Fred N. van Kempen, MicroWalt Corporation.
Debian version 3.3.7p-2 (debian).

DIP> help
DIP knows about the following commands:

        beep        bootp       break       chatkey     config
        databits    dec         default     dial        echo
        flush       get         goto        help        if
        inc         init        mode        modem       netmask
        onexit      parity      password    proxyarp    print
        psend       port        quit        reset       securidfixed
        securid     send        shell       skey        sleep
        speed       stopbits    term        timeout     wait

DIP> echo
Usage: echo on|off
DIP>
```

V průběhu následujícího textu ukazují příklady začínající výzvou **DIP>**, jak daný příklad zadat v ladicím režimu a jaká bude odezva příkazu. Příklady bez této výzvy se vztahují k obsahu skriptů.

Příkazy modemu

Nástroj **dip** obsahuje značný počet příkazů pro konfiguraci sériové linky a modemu. Význam některých z nich je zřejmý, například příkazu **port** vybírajícího sériový port, nebo příkazů **speed**, **databits**, **stopbits** a **parity**, které nastavují obecné parametry linky. Příkaz **modem** nastavuje typ modemu. V současné době je jediným podporovaným typem modemu typ HAYES (musí být za-

dáno velkými písmeny). Programu **dip** musíte typ modemu zadat, jinak by odmítl provést příkazy **dial** a **reset**. Příkaz **reset** posílá modemu inicializační řetězec; hodnota řetězce přitom závisí na typu modemu. Pro modemy kompatibilní se standardem HAYES je tímto řetězcem příkaz **ATZ**.

Příkaz **flush** slouží ke smazání všech odpovědí, které modem do této chvíle poslal. Bez něj by nemusela přihlašovací sekvence následující po inicializaci modemu fungovat správně, protože by načetla odezvu **OK**, vrácenou modemem po provedení inicializace.

Příkaz **init** nastavuje inicializační řetězec, který bude předán modemu před začátkem vytáčení. Implicitním řetězcem pro modemy kompatibilní se standardem HAYES je „ATE0 Q0 V1 X1“, který zapne zobrazování příkazů, dlouhé kódy výsledků a nastaví volání naslepo (bez detekce vytáčecího tónu). Moderní modemy mají poměrně vhodně zvolená standardní nastavení, takže tato inicializace nebývá nutná, i když na druhé straně ničemu neuškodí.

Příkaz **dial** odešle inicializační řetězec a vytočí číslo vzdáleného systému. Implicitním vytáčecím příkazem pro modemy kompatibilní se standardem HAYES je příkaz **ATD**.

Příkaz echo

Příkaz echo slouží jako ladicí prostředek. Po zadání příkazu **echo on** bude **dip** vypisovat na konzolu všechno, co posílá na sériovou linku. Zobrazování lze vypnout příkazem **echo off**.

Nástroj **dip** také umožňuje dočasně opustit skriptový režim a vstoupit do terminálového režimu. V tomto režimu lze nástroj **dip** používat jako jakýkoliv jiný běžný terminálový program. Můžete v něm zapisovat na sériovou linku nebo z ní číst. Chcete-li tento režim opustit, stiskněte kombinaci kláves **Ctrl+]**.

Příkaz get

Příkazem **get** se v programu **dip** nastavují proměnné. Nejjednodušším způsobem je přiřadit proměnné konstantní hodnotu tak, jak to děláme ve skriptu `cowslip.dip`. Můžete ale také požádat uživatele, aby hodnotu zadal. K tomu slouží klíčové slovo **ask**, které je nutno uvést na místě hodnoty proměnné:

```
DIP> get $local ask
Enter the value for $local: _
```

Třetí metoda spočívá v získání hodnoty ze vzdáleného hostitele. Na první pohled to vypadá zvláštně, ale v některých případech je tento způsob velmi užitečný. Některé servery SLIP vám nedovolí při spojení pomocí protokolu SLIP používat vlastní IP adresu a místo toho vám při každém přihlášení přidělí nějakou adresu ze svého seznamu adres a zobrazí zprávu, která vás bude o přidělené adrese informovat. Pokud tato zpráva bude například „`Your address: 193.174.7.202`“, můžete si následujícím kódem tuto adresu vyzvednout:

```
# konec přihlášení
wait address: 10
get $locip remote
```

Příkaz print

Tento příkaz umožňuje vypsat text na konzolu, ze které byl nástroj **dip** spuštěn. V rámci příkazu print lze použít libovolné definované proměnné, například:

```
DIP> print Using port $port at speed $speed
Using port cua3 at speed 38400
```

Názvy proměnných

Nástroj **dip** rozumí pouze předdefinované skupině proměnných. Název proměnné vždy začíná symbolem dolaru a musí být psán malými písmeny.

Proměnné *$local* a *$locip* obsahují název místního hostitele a jeho IP adresu. Když proměnné *$local* přiřadíte kanonický název hostitele, pokusí se **dip** automaticky tento název převést na IP adresu a přiřadit ji proměnné *$locip*. Obdobně se **dip** chová při nastavení proměnné *$locip*: reverzním dotazem se pokusí zjistit název, který této adrese odpovídá a uloží jej do proměnné *$local*.

Proměnné *$remote* a *$rmtip* se chovají úplně stejně a obsahují název a IP adresu vzdáleného hostitele.

Proměnná *$mtu* obsahuje hodnotu MTU daného spojení.

Tyto proměnné jsou jedinými proměnnými, kterým mohou být přímo přidělovány hodnoty pomocí příkazu **get**. Hodnoty dalších proměnných se nastavují v důsledku volání stejnojmenných konfiguračních příkazů a obsahují použitou hodnotu nastavení – například *$modem*, *$port* nebo *$speed*.

Proměnná *$errlvl* umožňuje přistupovat k výsledku naposledy spuštěného příkazu. Návratová hodnota rovná nule znamená úspěšnou operaci, zatímco nenulová hodnota naznačuje chybnou operaci.

Příkazy if a goto

Příkaz **if** umožňuje podmíněné větvení, nejedná se o úplný příkaz *if* známý z různých programovacích jazyků. Jeho syntaxe je následující:

```
if var op number goto label
```

Celý podmíněný výraz je tvořen jednoduchým porovnáním hodnoty proměnné (specifikované parametrem *var*) s celočíselnou hodnotou (specifikovanou parametrem *number*). Operátor porovnání *op* může být ==, !=, >, <, >= nebo <=.

Příkaz **goto** zajistí, že skript bude pokračovat na řádce za návěštím definovaným hodnotou *label*. Návěští musí začínat na prvním znaku řádku a bezprostředně za ním musí následovat dvojtečka.

Příkazy send, wait a sleep

Tyto příkazy umožňují implementovat ve skriptech jednoduché dialogy. Příkaz **send** zapíše své parametry na sériovou linku. Nepodporuje použití proměnných, zato však rozumí všem kombinacím znaků s převráceným lomítkem, které pocházejí z jazyka C, jako je například \ n pro nový řádek nebo \ b pro backspace. Znak tildy (~) slouží jako zkratka pro sekvenci CR/LF.

Parametrem příkazu **wait** je slovo. Příkaz čte vstup ze sériové linky tak dlouho, dokud nedojde sekvence znaků odpovídající zadanému slovu. Slovo nesmí obsahovat mezeru. K příkazu wait můžete přidat i druhý nepovinný parametr, který bude udávat délku čekání v sekundách. Pokud očekávané slovo v zadané době nedojde, příkaz nastaví hodnotu proměnné *$errlvl* na 1. Příkaz se používá k detekci přihlašovací a jiných výzev.

Příkaz **sleep** slouží k nastavení časové prodlevy, například k vyčkání na dokončení přihlašovací procedury. Interval je zadáván v sekundách.

Příkazy mode a default

Tyto příkazy se používají k přepnutí sériové linky do režimu SLIP a ke konfiguraci rozhraní. Příkaz **mode** je posledním příkazem spuštěným v nástroji **dip** předtím, než se nástroj přepne do režimu démona.

Příkaz **mode** má jako parametr název protokolu. Nástroj **dip** v současné době akceptuje protokoly SLIP, CSLIP, SLIP6, CSLIP6, PPP a TERM. Bohužel, v současné verzi není možné použití adaptivního protokolu SLIP.

Po přepnutí sériové linky do režimu SLIP se spustí příkaz **ifconfig**, který nakonfiguruje rozhraní jako dvoubodový spoj a příkaz **route**, kterým se nastaví trasa ke vzdálenému hostiteli.

Pokud skript navíc spustí před příkazem **mode** i příkaz **default**, nastaví se SLIP linka také jako implicitní trasa.

Spuštění v režimu serveru

Nastavení klienta s protokolem SLIP bylo poměrně obtížné. Nastavení hostitele, aby fungoval jako SLIP server, je mnohem jednodušší.

Existují dva způsoby konfigurace SLIP serveru. V obou případech potřebujete pro každého SLIP klienta jeden přihlašovací účet. Řekněme, že poskytujete SLIP linku Arthuru Dentovi na adrese **dent.beta.com**. Do souboru `passwd` můžete přidat následující řádek, kterým vytvoříte účet **dent**:

```
dent:*:501:60:Arthur Dent's SLIP account:/tmp:/usr/sbin/diplogin
```

Pomocí příkazu **passwd** pak nastavíte heslo tohoto účtu.

Jednou z možných realizací SLIP serveru je použití nástroje **dip** v režimu serveru. V tom případě se spouští příkazem **diplogin**. Jeho hlavním konfiguračním souborem bude soubor `/etc/diphosts`, kde uvádíte IP adresu přiřazenou klientovi poté, co se přihlásí. Alternativně můžete použít nástroj **sliplogin**, který je odvozen z balíku BSD a umožňuje mnohem pružnější konfigurační schéma, na jehož základě lze spouštět skripty kdykoliv se vzdálený hostitel připojí nebo odpojí.

Když se nyní uživatel **dent** přihlásí, spustí se nástroj **dip** jako server. Aby zjistil, zda je přihlášený uživatel oprávněn používat službu SLIP, vyhledá jeho jméno v souboru `/etc/diphosts`. Tento soubor definuje přístupová práva a parametry spojení každého uživatele služby SLIP. Obecný formát záznamu v souboru `/etc/diphosts` vypadá takto:

```
# /etc/diphosts
user:password:rem-addr:loc-addr:netmask:comments:protocol,MTU
#
```

Jednotlivé údaje jsou popsány v tabulce 7.2.

Tabulka 7.2 – Popis údajů v souboru /etc/diphosts

Údaj	Popis
user	Jméno uživatele, k němuž se tento údaj vztahuje.
password	Tento údaj umožňuje dodatečnou ochranu dalším heslem v závislosti na připojení. Můžete zde zadat zakódované heslo (stejně jako v souboru /etc/passwd) a program **diplogin** pak uživatele požádá o zadání tohoto hesla předtím, než mu povolí SLIP přístup. Toto heslo se používá navíc kromě klasického přihlašovacího hesla, které bude muset uživatel zadat také.
rem-addr	Adresa přiřazená vzdálenému počítači. Může být zadána buď názvem, který bude poté přeložen, nebo přímo jako IP adresa v tečkové notaci.
loc-addr	Adresa přiřazená místnímu konci SLIP linky. Rovněž může být zadána názvem nebo číselnou hodnotou.
netmask	Síťová maska používaná pro potřeby směrování. Tento údaj je často chápán chybně. Nejedná se o masku pro samotnou linku SLIP, ale o masku, která bude společně s údajem rem-addr sloužit ke směrování na vzdálený systém. Musí být proto použita taková maska, jakou používá síť vzdáleného systému.
comments	Tento údaj může být libovolný text, sloužící k popisu údajů v souboru. Není k ničemu využíván.
protocol	Tento údaj definuje, jaký protokol nebo režim linky se má použít. Možné hodnoty jsou stejné jako hodnoty parametru –p příkazu **slattach**.
MTU	Povolené MTU dané linky. Tento údaj definuje největší délku datagramu, přenášenou linkou. Jakýkoliv datagram poslaný na linku a delší než MTU bude fragmentován na části kratší než MTU. Typicky se MTU nastavuje na obou koncích linky stejně.

Příklad záznamu pro uživatele **dent** by mohl vypadat takto:

```
dent::dent.beta.com:vbrew.com:255.255.255.0:Arthur Dent:CSLIP,296
```

Uživatel **dent** má povolen přístup bez dodatečné ochrany heslem. Bude mít přiřazenu adresu odpovídající stroji **dent.beta.com** se síťovou maskou 255.255.255.0. Jeho implicitní trasa povede na počítač **vbrew.com** a bude používat protokol CSLIP s MTU 296 bajtů.

Když se uživatel **dent** přihlásí, nástroj **diplogin** si o něm vytáhne informace ze souboru diphosts. Pokud není druhý údaj prázdný, vyzve ho k zadání „externího bezpečnostního hesla“. Řetězec zadaný uživatelem se pak zašifruje a porovná s heslem v souboru diphosts. Nebudou-li hesla souhlasit, bude přihlášení odmítnuto. Obsahuje-li heslo řetězec s/key a nástroj **dip** byl přeložen s podporou S/Key, dojde k autentikaci protokolem S/Key. Tento autentikační mechanismus je popsán v dokumentaci k balíku **dip**.

Po úspěšném přihlášení přepne **diplogin** linku do režimu CSLIP nebo SLIP a nastaví rozhraní a směrování. Spojení trvá až do doby, než se vzdálený uživatel odpojí a modem zavěsí linku. Poté **diplogin** přepne linku zpět do normálního režimu a ukončí se.

Nástroj diplogin vyžaduje práva superuživatele. Pokud program dip nespouštíte jako setuid root, musíte diplogin vytvořit jako samostatnou kopii programu dip a ne pouze jako odkaz. Pak můžete diplogin bezpečně spouštět setuid root, aniž by byl ovlivněn status programu dip.

KAPITOLA 8

Protokol PPP

Protokol PPP slouží stejně jako protokol SLIP k posílání datagramů po sériové lince, avšak odstraňuje spoustu nedostatků, kterými trpí protokol SLIP. V první řadě umožňuje i přenos jiných protokolů, nejen protokolu IP. Dále zajišťuje chybovou detekci na lince, zatímco protokol SLIP bez problémů přenáší i poškozené datagramy, pokud není poškozena přímo jejich hlavička. Navíc umožňuje komunikujícím stranám dohodnout na počátku parametry spojení, například IP adresy nebo maximální velikost datagramu a zajišťuje ověření totožnosti klienta. Vestavěný mechanismus dohody usnadňuje automatické navázání spojení, možnost autentikace klienta odstraňuje nutnost vytvářet speciální uživatelské účty, které potřebuje protokol SLIP. Pro každou z těchto funkcí používá PPP samostatný protokol. Jednotlivé stavební kameny protokolu PPP si nyní popíšeme. Tato kapitola nicméně není úplným popisem protokolu PPP. Pokud potřebujete vědět více, doporučujeme vám přečíst si definiční RFC a další zhruba desítku RFC, které s tímto protokolem souvisejí[62]. Nakladatelství O'Reilly navíc vydalo velmi podrobnou knihu, *Using & Managing PPP* Andrewa Suna.

Na nejnižší vrstvě protokolu PPP se nachází protokol HDLC, *High-Level Data Link Control Protocol*. Ten definuje ohraničení jednotlivých rámců protokolu PPP a poskytuje 16bitový kontrolní součet[63]. Na rozdíl od primitivnějšího zapouzdření v protokolu SLIP může rámec v protokolu PPP obsahovat pakety i jiných protokolů než IP, například protokolu IPX sítě Novell nebo protokolu Appletalk. Tato možnost se implementuje tak, že základní rámec HDLC je rozšířen o údaj definující typ paketu, který je rámcem přenášen.

Protokol LCP, *Link Control Protocol*, se používá nad protokolem HDLC a používá se k dohodnutí parametrů týkajících se datového spojení, jako jsou například *Maximum Receive Unit*, MRU, jež uvádí maximální velikost datagramu, kterou je komunikující strana ochotná přijímat.

Důležitým krokem ve fázi konfigurace spojení protokolem PPP je ověření totožnosti klienta. Ačkoliv není povinné, je u vytáčených linek prakticky nezbytné, aby se znemožnil libovolný přístup komukoliv. Volaný hostitel (server) typicky vyzve klienta k ověření tím, že prokáže znalost nějakého tajného klíče. Pokud klient nebude schopen tuto znalost prokázat, spojení se ukončí. U protokolu PPP funguje ověřování totožnosti oběma směry; to znamená, že i klient může požádat server, aby prokázal svou totožnost. Obě tyto ověřovací procedury jsou vzájemně nezávislé. Autentikace se provádí dvěma různými protokoly, *Password Authentication Protocol* (PAP) a *Challenge Handshake Authentication Protocol* (CHAP).

Každý síťový protokol, který je směrován po datové lince (například IP, AppleTalk a podobné) se konfiguruje dynamicky odpovídajícím protokolem *Network Control Protocol* (NCP)[64]. Například

62 Příslušné RFC dokumenty jsou uvedeny v seznamu literatury na konci knihy.

63 Protokol HDLC je podstatně obecnější, definuje jej International Standards Organisation (ISO) a je základní součástí i jiných protokolů, například X.25.

64 Pozn. překladatele: Neplést s NetWare Core Protocol, což je protokol, jímž v sítích Novell NetWare komunikují stanice a servery.

aby bylo možné po PPP lince poslat IP datagram, musí se obě strany nejprve dohodnout, jakou budou používat IP adresu. K tomu jim poslouží protokol *Internet Protocol Control Protocol* (IPCP) – jeden z protokolů množiny NCP.

Kromě standardního posílání IP datagramů po lince podporuje protokol PPP také Van Jacobsonovu kompresi hlaviček IP datagramů. Tato technika změnšuje velikost hlaviček až na tři bajty. Používá se i v protokolu CSLIP a je všeobecně známá pod názvem VJ komprese hlaviček. Použití této komprese může být rovněž sjednáno při spuštění protokolem IPCP.

Protokol PPP v Linuxu

V Linuxu je činnost protokolu PPP rozdělena na dvě části; na komponentu jádra, která obsluhuje nízkoúrovňové protokoly (HDLC, IPCP, IPXCP a podobně) a na démona **pppd**, který běží v uživatelském prostoru a obsluhuje protokoly vyšší úrovně, například PAP a CHAP. Současná implementace protokolu PPP na Linuxu obsahuje démona **pppd** a program **chat**, který automatizuje připojení ke vzdálenému systému.

Ovladač pro jádro napsal Michael Callahan a přepracoval jej Paul Mackerras. Démon **pppd** byl odvozen z volně šiřitelné implementace protokolu PPP[65] v počítačích Sun a 386BSD, jejímž autorem je Drew Perkins a další a nyní ji spravuje Paul Mackerras. Na platformu Linuxu ho převedl Al Longyear. Program **chat** napsal Karl Fox[66].

Protokol PPP je stejně jako protokol SLIP implementován pomocí speciálního režimu linky. Chcete-li používat nějakou sériovou linku jako linku s protokolem PPP, musíte nejprve obvyklým způsobem vytvořit spojení pomocí modemu a následně převést linku do režimu protokolu PPP. V tomto režimu budou všechna příchozí data předána ovladači protokolu PPP, který ověří platnost rámců protokolu HDLC (každý rámec HDLC je doplněn 16bitovým kontrolním součtem), rozbalí je a předá k dalšímu zpracování. Současná implementace dokáže přenášet protokol IP, s případnou Van Jacobsonovou kompresí hlaviček, a dále protokol IPX.

Ovladači jádra operačního systému pomáhá démon **pppd**, který realizuje inicializační a autentikační fázi, které musejí proběhnout před zahájením samotného datového přenosu. Chování démona **pppd** je možné doladit pomocí mnoha parametrů. Jelikož je ovladač protokolu PPP poměrně složitý, není možné popsat všechny tyto parametry v jediné kapitole. Tato kniha tak nepopisuje všechny vlastnosti démona **pppd**, poskytuje pouze stručný úvod. Chcete-li se o tomto problému dozvědět více, přečtěte si knihu *Using & Managing PPP*, manuálové stránky démona **pppd** a soubor README v distribuci zdrojových kódů démona. Zde najdete odpovědi na většinu otázek, o kterých v této kapitole nehovoříme. Užitečný je rovněž dokument PPP-HOWTO.

Asi nejlepší pomoci s nastavením protokolu PPP se vám dostane od někoho, kdo používá stejnou distribuci Linuxu jako vy. Problémy s konfigurací protokolu PPP jsou velmi běžné, takže můžete vyzkoušet diskusní skupiny uživatelů Linuxu ve vašem okolí, nebo linuxový IRC kanál. Pokud stále něco nebudete vědět, přečtěte si konferenci **comp.protocols.ppp**. Na tomto místě se potkává většina lidí, kteří na vývoji démona **pppd** pracují.

Spuštění démona pppd

Když se chcete připojit na Internet pomocí protokolu PPP, musíte nejdříve nastavit základní síťovou podporu, jako je lokální zpětnovazebné zařízení a resolver. O této problematice jsme mluvili v kapitolách 5 a 6. Jako jmenný server můžete v souboru `/etc/resolv.conf` nastavit server vašeho poskytovatele připojení, znamená to ale, že všechny DNS požadavky budou přenášeny vaší

65 Obecné otázky týkající se protokolu PPP vám zodpoví v konferenci Linux-net na adrese **vger.rutgers.edu**.

66 Karla můžete kontaktovat na adrese **karl@morningstar.com**.

sériovou linkou. Tato situace není optimální, protože čím jste blíže jmennému serveru, tím rychlejší bude vyhledávání. Alternativním řešením je nakonfigurovat na některém počítači na vaší síti caching-only server. Pak to znamená, že první dotaz na určité jméno bude poslán po sériové lince, všechny další dotazy však budou obslouženy přímo vaším lokálním serverem a odezva bude mnohem rychlejší. O této konfiguraci jsme mluvili v kapitole 6.

V úvodním příkladu navázání PPP spojení pomocí démona **pppd** budeme předpokládat, že jsme opět na hostiteli **vlager**. Nejprve zavoláme na server **c3po** a přihlásíme se na účet **ppp**. Server **c3po** aktivuje svůj ovladač protokolu PPP. Po ukončení komunikačního programu, kterým navážeme spojení, spustíme následující příkaz, přičemž místo zařízení `ttyS3` použijete to zařízení, jímž jsme se připojili:

```
# pppd /dev/ttyS3 38400 crtscts defaultroute
```

Tento příkaz přepne sériovou linku ttyS3 do režimu PPP a naváže IP spojení se serverem **c3po**. Linka bude pracovat s přenosovou rychlostí 38 400 b/s. Parametr **crtscts** zapíná hardwarový handshake na sériovém portu, což je pro rychlosti nad 9 600 b/s nezbytné.

Démon **pppd** si ihned po spuštění dohodne se vzdáleným protějškem některé parametry linky prostřednictvím protokolu LCP. Při sjednávání parametrů budou obvykle fungovat implicitní hodnoty, takže se jimi nebudeme nyní zabývat. V rámci této dohody si oba konce linky vymění nebo vyžádají své IP adresy.

Pro tuto chvíli budeme také předpokládat, že server **c3po** od nás nebude vyžadovat žádný typ autentikace, čímž máme konfigurační fázi úspěšně za sebou.

Následně dohodne démon **pppd** se svým protějškem parametry IP spojení pomocí protokolu IPCP. Protože jsme démonu nezadali žádnou IP adresu, pokusí se prostřednictvím resolveru zjistit adresu lokálního počítače. Oba hostitelé si navzájem sdělí své IP adresy.

Tato implicitní nastavení jsou zpravidla dostačující. Dokonce i v případě, že je váš počítač připojen do sítě Ethernet, můžete použít stejnou IP adresu jak pro Ethernet, tak i pro rozhraní PPP. Nicméně démon nám umožňuje nastavit jinou IP adresu, popřípadě si adresu vyžádat od vzdáleného počítače. Tyto volby jsou popsány později v části *Nastavení IP konfigurace*.

Po provedení fáze nastavení protokolem IPCP připraví démon **pppd** síťovou vrstvu pro použití linky PPP. Nejprve nastaví síťové rozhraní PPP jako dvoubodové spojení s tím, že první PPP linka bude pojmenována ppp0, druhá ppp1 a tak dále. Dále do směrovací tabulky přidá položku ukazující na hostitele na druhém konci spojení. Ve výše zmíněném příkladu nastaví démon **pppd** trasu na server **c3po** jako implicitní, protože jsme použili parametr defaultroute[67]. Tato volba zjednoduší směrování, protože způsobí, že všechny datagramy poslané jiným než lokálním hostitelům budou poslány na server **c3po** – což je jediná cesta, jak se s takovými hostiteli spojit. Démon **pppd** podporuje různá další směrovací schémata, budeme o nich podrobněji hovořit později.

Konfigurační soubory

Dříve než démon **pppd** analyzuje parametry na příkazovém řádku, projde několik souborů, zda neobsahují implicitní volby. Tyto soubory mohou obsahovat libovolné parametry příkazové řádky, které mohou být rozepsány na několika řádcích. Komentáře začínají symbolem #.

Prvním prohlíženým souborem je soubor `/etc/ppp/options`. Je rozumné pomocí tohoto souboru nastavit všeobecně platné volby, protože tak zabráníte uživatelům v nechtěném narušení bezpečnosti celého systému. Chcete-li například, aby démon **pppd** vyžadoval od svého protějšku nějaký typ ověření totožnosti (buď pomocí protokolu PAP nebo pomocí protokolu CHAP), uveďte

67 Implicitní trasa se nastaví pouze v případě, že ještě žádná není nastavena.

v tomto souboru volbu `auth`. Tuto volbu nemůže uživatel potlačit, takže nemůže navázat spojení pomocí protokolu PPP s žádným systémem, který není v ověřovací databázi. Některé volby nicméně lze potlačit, například nastavení připojovacího řetězce.

Druhým souborem čteným po souboru `/etc/ppp/options` je soubor `.ppprc`. Nachází se v domovském adresáři uživatele a umožňuje tak uživatelům nastavit vlastní standardní volby.

Příklad souboru `/etc/ppp/options` by mohl vypadat takto:

```
# Globální volby pro pppd běžící na vlager.vbrew.com
lock                          # používej zamykání zařízení dle UUCP
auth                          # požaduj autentikaci
usehostname                   # pro CHAP použij lokální jméno
domain vbrew.com              # naše doména
```

Klíčové slovo lock nastaví démona **pppd**, aby používal standardní metodu zamykání zařízení používanou v UUCP. Podle této konvence vytvoří každý proces, který přistupuje k sériovému zařízení (řekněme k zařízení `/dev/ttyS3`), soubor `LCK..ttyS3` ve speciálním adresáři zámkových souborů. Tím se dalším programům, jako je například **minicom** nebo **uucico**, signalizuje, že zařízení nemají používat, protože je používá protokol PPP.

Další tři volby se vztahují k autentikaci a tedy k bezpečnosti systému. Nastavení autentikace je ideální uvést v globálním konfiguračním souboru, protože jde o „privilegovaná“ nastavení a nelze je přepsat hodnotami v uživatelských souborech `~/.ppprc`.

Automatické vytáčení programem chat

Jednou z věcí, která se vám mohla zdát v předchozím příkladu nepohodlná, je nutnost manuálně navázat spojení dříve, než se spustí démon **pppd**. Na rozdíl od nástroje **dip** nemá démon **pppd** svůj vlastní skriptový jazyk, který by umožňoval vytočení a přihlášení ke vzdálenému systému. Místo toho spoléhá na nějaký externí program nebo skript příkazového interpretu, který za něj tento úkol provede. Příkaz, který to zajistí, lze démonu **pppd** předat pomocí volby connect na příkazovém řádku. Démon **pppd** přesměruje standardní vstup a výstup tohoto příkazu na sériovou linku.

Balík **pppd** obsahuje velmi jednoduchý program nazvaný **chat**, který je určen k takovéto automatizaci jednoduchých přihlašovacích sekvencí. Budeme o něm hovořit podrobněji.

Pokud používáte složitější přihlašovací postup, potřebujete něco výkonnějšího, než je **chat**. Užitečnou alternativou je například program **expect** Dona Libese. Obsahuje výkonný jazyk vycházející z jazyka Tcl, a byl navržen přesně pro tento druh aplikace. Pokud používáte přihlašovací mechanismus, který v sobě zahrnuje řekněme autentikaci s použitím nějakého „kalkulátorového“ generátoru klíče, program **expect** to dokáže. Protože existuje velké množství různých variant přihlašování, nebudeme se zde zabývat tím, jak vytvořit příslušné skripty. Poznamenáme jenom, že vytvořený skript můžete volat tak, že jeho název předáte programu **pppd** pomocí volby connect. Je také důležité upozornit vás, že po dobu běhu skriptu jsou standardní vstup a výstup přesměrovány na modem a nepracují na terminálu, na němž byl **pppd** spuštěn. Pokud potřebujete nějakou interakci s uživatelem, musíte ji ošetřit vytvořením samostatného virtuálního terminálu nebo nějakým jiným mechanismem.

Program **chat** umožňuje používat komunikační skripty ve stylu UUCP. Komunikační skript je v podstatě složen ze střídající se sekvence očekávaných řetězců, které přijdou ze vzdáleného systému a z odpovědí, jež na ně posíláme. Následuje typický výpis části komunikačního skriptu:

```
ogin: b1ff ssword: s3k|<r1t
```

Tento skript říká programu **chat**, aby vyčkal, až vzdálený systém pošle výzvu k přihlášení a potom mu poslal přihlašovací jméno **b1ff**. Čekáme pouze na řetězec ogin:, takže nebude vadit, když bude přihlašovací výzva začínat malým nebo velkým písmenem, nebo když dorazí ve zkomolené formě. Následující řetězec je opět očekávaný řetězec, který pozdrží program **chat**, dokud nepřijde výzva k zadání hesla, potom odešle naše heslo jako odpověď.

To je v podstatě vše o tom, jak komunikační skripty pracují. Kompletní skript sloužící k vytáčení serveru by musel samozřejmě obsahovat patřičné příkazy pro modem. Předpokládejme, že váš modem rozumí množině příkazů standardu Hayes a že telefonní číslo vytáčeného serveru je 31 87 14. Kompletní volání programu **chat**, které by provedlo spojení se serverem **c3po,** by potom vypadalo následovně:

```
$ chat -v '' ATZ OK ATDT318714 CONNECT '' ogin: ppp word: GaGariN
```

Podle definice musí být první řetězec očekávaný řetězec, ale protože nám modem nic neřekne, dokud ho nepobídneme, necháme program **chat** „přeskočit" první řetězec tím, že mu zadáme jen prázný řetězec. Pak pokračujeme odesláním příkazu ATZ, což je inicializační řetězec pro modemy kompatibilní se standardem Hayes a následně budeme čekat na odpověď (OK). Následující řetězec pošle příkaz pro vytočení i s vytáčeným číslem a čeká na odezvu CONNECT. Dále následuje opět prázdný řetězec, protože teď nechceme nic posílat a čekáme na výzvu k přihlášení. Zbytek komunikačního skriptu funguje naprosto shodně s výše popsaným postupem. Celý popis je možná trochu matoucí, za chvíli ale uvidíme, že existuje způsob, jak vytvořit skript pro program **chat** podstatně jednodušeji.

Parametr –v zajistí, že program **chat** bude veškeré aktivity zaznamenávat prostřednictvím služby **syslog**[68].

Zadávání komunikačního skriptu na příkazové řádce s sebou nese jistá rizika, protože si uživatelé mohou prohlédnout příkazovou řádku daného procesu pomocí příkazu **ps**. Tomuto problému se vyhnete, když umístíte komunikační skript do souboru jako dial-c3po. Pomocí parametru –f pak donutíte program **chat**, aby četl skript ze souboru a ne z příkazové řádky. Tato operace s sebou přináší další výhodu, protože sekvence dialogu nyní budou podstatně čitelnější. Budeme-li převádět náš příklad, soubor dial-c3po bude vypadat takto:

```
''      ATZ
OK      ATDT318714
CONNECT ''
ogin:   ppp
word:   GaGariN
```

Když vytváříme pro program **chat** skript tímto způsobem, zapisujeme sekvenci, na jejíž přijetí čekáme, na levou stranu řádku, sekvenci, jíž odpovíme, pak na pravou stranu řádku. V této podobě je vytáčecí skript podstatně čitelnější.

Celé spuštění démona **pppd** by mohlo nyní vypadat následovně:

```
# pppd connect "chat -f dial-c3po" /dev/ttyS3 38400 -detach \
        crtscts modem defaultroute
```

Kromě volby *connect* specifikující skript pro vytáčení jsme do příkazové řádky přidali ještě další dvě volby: *–detach* sdělí démonu **pppd**, aby se neodpojoval od konzoly a zůstal jako proces v po-

68 Pokud v souboru `syslog.conf` přesměrujete tyto zaznamenávané zprávy do souboru, ověřte, že soubor není veřejně čitelný. Program **chat** totiž zaznamenává celý komunikační skript včetně hesla.

zadí. Klíčové slovo *modem* sděluje démonu **pppd**, že na sériovém zařízení je připojen modem a aby proto provedl některé akce pro modem specifické, jako je například zavěšení linky před a po volání. Pokud toto klíčové slovo nepoužijete, nebude démon **pppd** sledovat signál DCD sériového portu a nepozná tak, pokud by vzdálená strana neočekávaně zavěsila.

Výše uvedené příklady byly poměrně jednoduché; program **chat** však umožňuje psát mnohem složitější komunikační skripty. Dovoluje například specifikovat řetězce, při jejichž přijetí se program ukončí s chybou. Typickými řetězci pro zastavení běhu skriptu jsou zprávy jako BUSY nebo NO CARRIER, které modem generuje v případě, že je volané číslo obsazené nebo vzdálená strana nezvedá telefon. Aby program rozpoznal tyto zprávy okamžitě a nečekal na vypršení časových intervalů, můžete je zadat na začátku skriptu pomocí klíčového slova ABORT:

```
$ chat -v ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ...
```

Podobným způsobem můžete ve skriptu změnit hodnotu čekací doby pomocí volby TIMEOUT. V některých případech je nutné podmíněné provedení určitých částí komunikačního skriptu: pokud například od vzdálené strany nepřijmete výzvu k přihlášení, budete chtít poslat příkaz BREAK nebo znak CR. Toho docílíte přidáním podskriptu k očekávanému řetězci. Ten se bude skládat ze sekvence posílaných a očekávaných řetězců, stejně jako tomu bylo v celém skriptu. Sekvence podskriptu je oddělena pomlčkami. Podskript se spouští tehdy, pokud očekávaný řetězec (k němuž je podskript připojen) nebude přijat v nastaveném časovém intervalu. V našem příkladu bychom mohli komunikační skript upravit takto:

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

Když **chat** nepřijme v nastaveném intervalu výzvu k přihlášení, spustí podskript, který odešle příkaz BREAK a znovu čeká na přihlašovací výzvu. Když bude tentokrát výzva přijata, bude se pokračovat v přihlášení; pokud se v nastaveném intervalu nepřijme, ohlásí skript chybu.

Nastavení konfigurace protokolu IP

K nastavení parametrů protokolu IP v době navazování spojení slouží protokol IPCP. Typicky každá ze stran posílá paket IPCP Configuration Request, v němž specifikuje, které volby chce nastavit nestandardním způsobem a jak. Po přijetí požadavku vzdálená strana jednotlivé nároky posoudí a buď je schválí, nebo zamítne.

Démon **pppd** vám dává k dispozici velký prostor pro nastavování parametrů, které se budou při připojování domlouvat. Jednotlivé parametry se nastavují různými volbami příkazové řádky, o kterých budeme hovořit později.

Volba IP adres

Všechna IP rozhraní musí mít přidělenu IP adresu, proto i PPP zařízení má vždy nějakou IP adresu. Rodina protokolů PPP poskytuje mechanismus, který umožňuje automatické přiřazení IP adres PPP rozhraním. Tak je možné, že program PPP na jednom konci dvoubodového spoje přidělí vzdálenému rozhraní adresu, kterou bude používat, nebo mohou obě strany používat vlastní adresy.

Některé PPP servery obsluhující velký počet klientů přidělují IP adresy dynamicky: připojenému systému se adresa přidělí v okamžiku, kdy zavolá a po odpojení se mu adresa odebere. Tím se počet potřebných IP adres omezí na počet současně možných připojení. Toto řešení je sice výhodné pro správce PPP serveru, obvykle však bývá méně výhodné pro připojující se klienty. V ka-

pitole 6 jsme hovořili o způsobu, jakým se IP adresy mapují na názvy hostitelů pomocí databází. Aby se klient mohl k vašemu systému připojit, musí znát jeho IP adresu nebo název. Pokud jste ale uživateli PPP služby, která vám přiděluje IP adresu dynamicky, těžko ji mohou vaši klienti znát, pokud nebudete mít implementován nějaký mechanismus, který změní záznamy DNS serveru poté, co je vám adresa přidělena. Takové systémy existují, nebudeme se jimi ale podrobněji zabývat, soustředíme se na příjemnější řešení, kdy používáte stejnou IP adresu pokaždé, když se k serveru připojíte[69].

V předchozím příkladu nám démon **pppd** vytočil server **c3po** a navázal s ním spojení. Nebyla provedena žádná opatření, která by některému z obou konců přidělila konkrétní IP adresu. Místo toho jsme využili standardního chování démona **pppd**, kdy se pokusí z názvu lokálního systému (v našem případě **vlager**) zjistit lokální IP adresu a tu použije na svém konci linky, adresu systému **c3po** si nechá sdělit od něj. Kromě tohoto řešení podporuje protokol PPP ještě další alternativy.

Chcete-li používat konkrétní adresy, zadáte démonu **pppd** následující parametr:

```
local_addr:remote_addr
```

Hodnoty *local_addr* a *remote_addr* mohou být zapsány buď ve formě tečkové notace, nebo jako názvy hostitelů[70]. Tento parametr způsobí, že se démon **pppd** pokusí použít první adresu jako svou vlastní a druhou adresu „vnutit“ protější straně. Pokud by vzdálená strana některou z adres odmítla, nedojde k navázání spojení[71].

Pokud se připojujete k nějakému serveru a očekáváte, že vám adresu přidělí, měli byste zajistit, ať se váš démon **pppd** nepokouší „prosadit“ vlastní adresu. Dosáhnete toho pomocí volby noipdefault a parametr *local_addr* necháte prázdný. Uvedením parametr noipdefault zabráníte démonu v použití lokální adresy počítače pro linku PPP.

Pokud si chcete nastavit lokální adresu a je vám jedno, jakou adresu chce používat vzdálená strana, nevyplňujte údaj *remote_addr*. Pokud například chcete, aby **vlager** používal místo své adresy pro linku PPP adresu 130.83.4.27, uveďte na příkazovém řádku 130.83.4.27:. Podobně chcete-li nastavit pouze vzdálenou adresu, neuveďte lokální adresu. Implicitně pak démon **pppd** použije adresu přidělenou vašemu hostiteli trvale.

Směrování přes linku PPP

Po nastavení síťového rozhraní démon **pppd** nastaví trasu ke vzdálenému systému pouze jako trasu k jedinému hostiteli. Představuje-li vzdálený hostitel celou lokální síť, budete se určitě chtít spojit i se systémy „za“ vaším partnerem na lince, takže musíte nastavit trasu na síť.

Už jsme si ukázali, že démona **pppd** je možné za pomoci volby defaultroute požádat, aby trasu přes vzdálený systém nastavil jako implicitní. Tato volba je velmi užitečná v případě, že se PPP server chová jako brána do Internetu.

Realizace obráceného případu, kdy se váš systém chová jako brána pro vzdálený systém, je také poměrně jednoduchá. Představme si například zaměstnance pivovaru, jehož domácí počítač se

69 Podrobnější informace o systémech dynamického mapování názvů na IP adresy můžete najít na *http://www.dynip.com/* a *http://www.justlinux.com/dynamic_dns.html*.

70 Použití názvů má v tomto případě dopad na autentikace protokolem CHAP. Podívejte se, prosím, do části *Autentikace protokolem PPP*, která následuje dále.

71 Parametry `ipcp-accept-local` a `ipcp-accept-remote` nařídí démonu **pppd** akceptovat lokální a vzdálenou adresu nabízené vzdáleným koncem i v případě, že v lokální konfiguraci nastavujeme vlastní adresy. Pokud tyto parametry nejsou uvedeny, odmítne démon **pppd** přijmout jakoukoliv adresu, kterou mu vzdálená strana nabízí.

jmenuje **oneshot**. Předpokládejme dále, že jsme bránu **vlager** nastavili jako volaný PPP server. Pokud **vlager** nakonfigurujeme tak, aby zaměstnanci dynamicky přidělil IP adresu podsítě pivovaru, můžeme démonu **pppd** zadat parametr proxyarp, kterým pro systém **oneshot** nastavíme funkci ARP proxy. Tím zajistíme, že **oneshot** bude přímo dostupný všem počítačům ze sítí pivovaru i vinařství.

Ne vždy to ale jde tak jednoduše, jako v tomto případě. Například spojení dvou lokálních sítí typicky vyžaduje přidání konkrétního síťového směrování, protože obě sítě již mohou mít své vlastní implicitní trasy. Navíc pokud byste v obou sítích nastavili PPP linku jako implicitní trasu, vznikla by smyčka, ve které budou mezi oběma protějšky obíhat pakety určené pro neznámé lokace tak dlouho, dokud neskončí jejich životnost.

Jako příklad předpokládejme, že si pivovar otevírá pobočku v nějakém jiném městě. Tato pobočka bude mít svůj vlastní Ethernet, který bude používat síťovou IP adresu 172.16.3.0, což je podsíť 3 v síti pivovaru třídy B. Pobočka se chce spojit s hlavní sítí pivovaru protokolem PPP, aby si mohli aktualizovat databáze zákazníků. Hostitel **vlager** se opět chová jako brána, bude podporovat protokol PPP. Jeho partner na síti pobočky se jmenuje **vbourbon** a má IP adresu 172.16.3.1. Síť je znázorněna na obrázku A2 v příloze A.

Když se hostitel **vbourbon** spojí s hostitelem **vlager**, nastaví jako obvykle implicitní trasu na hostitele **vlager**. Nicméně na bráně **vlager** máme implicitně nastavenou dvoubodovou trasu pouze na hostitele **vbourbon** a musíme explicitně nastavit trasu na podsíť 3 přes bránu **vbourbon**. Můžeme to udělat ručně příkazem **route** po navázání PPP spojení, což ovšem není příliš praktické řešení. Naštěstí můžeme zajistit i automatické nastavení této trasy pomocí další funkce démona **pppd**, o níž jsme zatím nehovořili. Jde o příkaz **ip-up**. Je to skript nebo program umístěný v adresáři `/etc/ppp` a démon **pppd** jej automaticky spustí po nastavení PPP rozhraní. Pokud tuto funkci používáme, skript se spouští s následujícími parametry:

```
ip-up iface device speed local_addr remote_addr
```

Následující tabulka obsahuje význam jednotlivých parametrů. (V prvním sloupci uvádíme číslo, které příkazový interpret používá pro přístup k dané hodnotě.)

Parametr	Název	Funkce
`$1`	*`iface`*	Používané síťové rozhraní, například ppp0.
`$2`	*`device`*	Cesta k použitému sériovému zařízení (pokud se používá stdin/stdout, pak /dev/tty).
`$3`	*`speed`*	Rychlost sériové linky v bitech za sekundu.
`$4`	*`local_addr`*	IP adresa našeho konce linky v tečkové notaci.
`$5`	*`remote_addr`*	IP adresa vzdáleného konce linky v tečkové notaci.

V našem případě by mohl skript **ip-up** obsahovat následující kód[72]:

```
#!/bin/sh
case $5 in
172.16.3.1)            # je to vbourbon
        route add -net 172.16.3.0 gw 172.16.3.1;;
...
esac
exit 0
```

72 Pokud bychom chtěli po připojení vytvořit trasy i na další sítě, ošetřili bychom to na místě sekvence „...“ ve skriptu pomocí dalších příkazů `case`.

Analogicky se po ukončení spojení volá skript `/etc/ppp/ip-down`, kterým můžeme zrušit nastavení provedená skriptem **ip-up**. Ve skriptu **ip-down** bychom tedy zadali příkaz **route**, kterým bychom odstranili trasu vytvořenou skriptem **ip-up**.

Nicméně směrovací schéma ještě není kompletní. Na obou hostitelích s protokolem PPP jsme nastavili příslušné položky směrovací tabulky, avšak žádný z ostatních hostitelů na obou sítích o nově vzniklém PPP spojení nic neví. Nevadí to v případě, že hostitelé na síti pivovaru budou mít nastavenu implicitní trasu na `vlager` a hostitelé na síti pobočky implicitní trasu na `vbourbon`. Pokud by to neplatilo, jedinou možností by bylo použití směrovacího démona jako je `gated`. Po vytvoření nového spojení na bráně `vlager` by démon všechny počítače na své síti o této trase vyrozuměl.

Nastavení parametrů linky

V předchozím textu jsme hovořili o protokolu LCP (Link Control Protocol), který se používá ke sjednání vlastností linky a k jejímu otestování.

Dvě nejdůležitější volby, jež mohou být sjednávány pomocí protokolu LCP, jsou *Asynchronous Control Character Map* a *Maximum Receive Unit*. Existuje i řada dalších voleb, které umožňuje protokol nastavit, jsou však příliš specializované na to, abychom se zde jimi mohli zabývat.

Nastavení Asynchronous Control Character Map, takzvaná *asynchronní mapa* nebo *async map*, se používá na asynchronních linkách, jako jsou například telefonní linky, k identifikaci řídicích znaků, které musí být nahrazeny escape sekvencí (tedy konkrétní sekvencí dvou jiných znaků). Měli byste se například vyvarovat použití znaků XON a XOFF, které jsou používány k softwarovému handshakingu, protože špatně nastavený modem by se mohl po přijetí znaku XOFF zablokovat. Dalším kandidátem jsou Ctrl-] (escape znak **telnetu**). Protokol PPP umožňuje nahrazovat libovolné znaky s ASCII kódy od 0 do 31 tím, že je specifikujete v asynchronní mapě.

Asynchronní mapa je bitová mapa o šířce 32 bitů zapsaná v šestnáctkové podobě. Nejméně významný bit odpovídá znaku s ASCII kódem 00 (NULL), nejvýznamnější bit znaku s kódem 31. Těchto 32 znaků představuje řídicí znaky ASCII kódování. Pokud je bit odpovídající konkrétnímu znaku v mapě nastaven, znamená to, že znak má být před odesláním na linku nahrazován escape sekvencí.

Pokud chcete svému protějšku specifikovat, které znaky musí a nemusí nahrazovat, můžete mapu definovat parametrem asyncmap démona **pppd**. Pokud chcete například nahrazovat pouze znaky ^S a ^Q (ASCII kódy 17 a 19, používané často jako znaky XON a XOFF), použijete následující nastavení:

```
asyncmap 0x000A0000
```

Převod je velmi jednoduchý, pokud dokážete převádět dvojková čísla na šestnáctková. Představte si před sebou 32 bitů. Nejpravější odpovídá ASCII znaku 00 (NULL), nejlevější znaku ASCII 31. Všechny bity nahrazovaných znaků nastavte jako jedničkové, všechny bity nenahrazovaných znaků jako nulové. Pro převod tohoto řetězce na parametr pro program **pppd** prostě vezměte vždy čtveřici bitů a převeďte ji na šestnáctkovou hodnotu. Měli byste dostat osm šestnáctkových číslic. Spojte je dohromady, přidejte před ně „0x“ abyste naznačili, že jde o šestnáctkovou hodnotu, a máte to hotovo.

Implicitně je asynchronní mapa nastavena na hodnotu *0xffffffff*, to znamená, že se budou nahrazovat všechny řídicí znaky. Je to bezpečná implicitní volba, obvykle je to ale více, než potřebujete. Každý znak uvedený v asynchronní mapě se převádí na dva znaky přenášené po lince, takže mapování jde na úkor zvýšeného zatížení linky a odpovídajícího poklesu výkonu[73].

Ve většině případů vyhovuje asynchronní mapa 0x0 – neprovádí se žádné mapování.

Hodnota Maximum Receive Unit (MRU) říká protější straně, jakou maximální velikost HDLC rámců chceme přijímat. Trochu to připomíná hodnotu Maximum Transfer Unit (MTU), nicméně tato dvě čísla nemají téměř nic společného. Hodnota MTU je parametrem síťového zařízení jádra a definuje maximální velikost rámce, který umí příslušné rozhraní přenést. Hodnota MRU představuje více méně pouze doporučení pro vzdálenou stranu, aby nevytvářela rámce delší než tato hodnota, nicméně rozhraní stejně musí být schopno zpracovat rámce až do délky 1 500 bajtů.

Volba hodnoty MRU proto není ani tak otázkou toho, co je daná linka schopna přenést, jako spíše otázkou nastavení, se kterým dosáhnete nejvyššího výkonu. Pokud hodláte po lince provozovat interaktivní aplikace, pak je dobré nastavit hodnotu MRU na nízkou hodnotu 296 bajtů, aby nějaký větší paket (například paket z relace FTP) nezpůsobil „zaseknutí" interaktivního přenosu. Chcete-li démonu **pppd** sdělit, že hodláte používat MRU o velikosti 296 bajtů, předáte mu parametr mru 296. Malé hodnoty MRU mají ovšem smysl pouze v případě, že používáte VJ kompresi hlaviček (která je standardně zapnuta). V opačném případě byste totiž značnou část přenosového pásma vyplýtvali tím, že byste přenášeli celé hlavičky IP datagramů.

Démon **pppd** rozumí i dalším volbám protokolu LCP, které nastavují celkové chování procesu sjednávání parametrů, jako je například maximální počet konfiguračních požadavků, které si mohou obě strany vyměnit, než se spojení ukončí. Dokud si nebudete absolutně jisti tím, co děláte, neměli byste tyto parametry měnit.

Kromě toho existují ještě dvě volby týkající se echa protokolu LCP. Protokol PPP definuje dvě zprávy, *Echo Request* a *Echo Response*. Tyto zprávy používá démon **pppd** ke kontrole, zda je linka stále aktivní. Funkci můžete povolit za parametrem lcp-echo-interval, za kterým následuje časový údaj v sekundách. Pokud nebudou v tomto intervalu ze vzdáleného hostitele přijaty žádné rámce, vygeneruje démon Echo Request a bude očekávat, že mu protějšek vrátí zprávu Echo Response. Jestliže se tak nestane, pak se spojení po určitém počtu odeslaných výzev ukončí. Tento počet je možné nastavit pomocí volby lcp-echo-failure. Implicitně je celá tato funkce vypnuta.

Obecné bezpečnostní úvahy

Špatně nakonfigurovaný démon protokolu PPP může mít z hlediska bezpečnosti zhoubný vliv na celý systém. V nejhorším případě to vypadá tak, že má každý uživatel možnost připojení do vaší sítě Ethernet (a to je velice špatné). V této stati si povíme o několika málo opatřeních, která by měla zabezpečit vaši konfiguraci protokolu PPP.

Ke konfiguraci síťových zařízení a směrovací tabulky jsou zapotřebí práva superuživatele. Typicky se tento problém řeší tak, že se **pppd** spouští jako setuid root. Démon **pppd** však umožňuje nastavit řadu vlastností významných z hlediska bezpečnosti.

Abyste se chránili před útoky provedenými změnou parametrů démona **pppd**, měli byste v souboru `/etc/ppp/options` specifikovat nejdůležitější nastavení, například ta, uvedená v dřívější části *Konfigurační soubory*. Některá z nich, například nastavení autentikace, uživatel nemůže potla-

73 Pozn. překladatele: Není to úplně zanedbatelné snížení – při rovnoměrném rozložení přenášených znaků může být přenosová rychlost snížena v nejhorším případě o více než 11 %.

čit, a proto poskytují rozumnou ochranu před zneužitím. Důležitým nastavením, které je nutné chránit, je parametr `connect`. Pokud máte v plánu povolit spuštění démona **pppd** pro připojení k Internetu i jiným uživatelům než je superuživatel, vždy byste měli v souboru `/etc/ppp/options` specifikovat volby `connect` a `noauth`. Pokud to neuděláte, uživatelé budou moci prostřednictvím démona **pppd** spustit v režimu superuživatele jakékoliv příkazy tím, že volbu `connect` uvedou na příkazovém řádku nebo ve svém osobním konfiguračním souboru.

Další rozumná věc je omezit použití démona **pppd** pouze na ty uživatele, kteří toto právo opravdu potřebují tak, že v souboru `/etc/group` založíte novou skupinu a do ní začleníte pouze takto privilegované uživatele. Skupinového vlastníka souboru **pppd** byste pak měli změnit na tuto skupinu a měli byste souboru odstranit právo všeobecného spuštění. Předpokládejme, že jste vytvořili skupinu *dialout*, pak stačí provést následující příkazy:

```
# chown root /usr/sbin/pppd
# chgrp dialout /usr/sbin/pppd
# chmod 4750 /usr/sbin/pppd
```

Samozřejmě se musíte chránit i vůči systémům, k nimž přistupujete. Abyste se chránili před hostiteli, kteří se vydávají za někoho jiného, měli byste při komunikaci s protějškem vždy používat nějaký druh ověření totožnosti. Kromě toho byste měli cizím hostitelům zakázat používání IP adres podle vlastního výběru a omezit jejich výběr jen na několik málo adres. V následující stati se budeme zabývat právě těmito tématy.

Autentikace protokolem PPP

Mechanismus PPP umožňuje, aby si kterákoliv z komunikujících stran vyžádala ověření totožnosti svého protějšku jedním ze dvou mechanismů – protokoly *Password Authentication Protocol* (PAP) nebo *Challenge Handshake Authentication Protocol* (CHAP). Po navázání spojení může kterýkoliv účastník požádat druhého o autentikaci, ať už jde o volaného nebo volajícího. V následujícím popisu budeme v případě potřeby používat obecné termíny „klient“ a „server“ k rozlišení systému, který si autentikaci vyžádal, a systému, který se autentikuje. Démon protokolu PPP si může autentikaci vyžádat tak, že odešle speciální požadavek protokolem LCP, ve kterém uvede požadovanou metodu autentikace.

Protokol PAP versus CHAP

Protokol PAP, používaný většinou internetových poskytovatelů, pracuje v podstatě stejně jako klasická přihlašovací procedura. Klient ověří svou totožnost tak, že serveru pošle své uživatelské jméno a (volitelně zašifrované) heslo. Server tyto údaje porovná se svou databází „tajemství“[74]. Tato technika je však zranitelná odposlechem, kdy útočník sleduje data přenášená sériovou linkou, ale lze ji také obejít metodou pokus-omyl.

Protokol CHAP takové nedostatky nemá. Server pošle klientovi náhodně vygenerovaný řetězec s „výzvou“ a spolu s ním i svůj název hostitele. Klient na základě názvu hostitele vyhledá příslušnou tajnou informaci, zkombinuje ji s přijatou výzvou a zašifruje tento řetězec pomocí jednosměrné šifrovací funkce. Výsledek pak společně s názvem hostitele klienta pošle zpět serveru. Server pak provede stejné výpočty a dojde-li k témuž výsledku, povolí klientovi přístup.

Další vlastností protokolu CHAP je, že nevyžaduje po klientovi ověření jeho totožnosti jen při spuštění, ale posílá výzvy v pravidelných intervalech, aby se ujistil, že klienta nenahradil nějaký

74 Slovíčko „tajemství“ (secret) je termín, kterým protokol PPP označuje hesla. Pro „tajemství“ protokolu PPP neplatí stejná délková omezení jako pro přihlašovací hesla Linuxu.

vetřelec, například přepnutím telefonních linek nebo tím, že chybně nakonfigurovaný modem neinformoval démona o ukončení původního volání a umožnil přijetí dalšího volání.

Démon **pppd** udržuje tajné klíče protokolů PAP a CHAP ve dvou samostatných souborech, pojmenovaných `/etc/ppp/pap-secrets` a `/etc/ppp/chap-secrets`. Uvedením vzdáleného hostitele v jednom z těchto souborů máte možnost určit, který protokol bude použit pro jeho i vaši autentikaci.

Démon **pppd** implicitně nevyžaduje po svém protějšku ověření totožnosti, ale je-li o to vzdálenou stranou požádán, svou totožnost prokáže. Protože je protokol CHAP mnohem silnější než protokol PAP, snaží se ho démon **pppd** používat, kdykoliv jen je to možné. Pokud ho protějšek nepodporuje nebo pokud démon **pppd** nemůže pro vzdálený systém nalézt v souboru `chap-secrets` tajný klíč, použije protokol PAP. Nenajde-li klíč potřebný pro autentikaci ani v souboru `pap-secrets`, odmítne se autentikovat a v důsledku toho se spojení ukončí.

Toto chování lze upravit několika způsoby. Použijete-li klíčové slovo `auth`, bude démon **pppd** požadovat po svém protějšku autentikaci. Bude souhlasit s použitím protokolu CHAP i PAP samozřejmě za předpokladu, že v příslušné databázi má uloženy potřebné informace. Existují i další možnosti jak zapnout nebo vypnout konkrétní autentifikační protokol, ty však nebudeme popisovat.

Pokud budou všechny systémy, se kterými máte spojení pomocí protokolu PPP, souhlasit s ověřením své totožnosti, měli byste vložit volbu `auth` do globálního souboru `/etc/ppp/options` a pro každý systém definovat v souboru `chap-secrets` příslušná hesla. Pokud některý systém nepodporuje protokol CHAP, vložte záznam o tomto systému do souboru `pap-secrets`. Tím zajistíte, že se k vám nebude moci připojit žádný cizí systém.

V dalším textu popíšeme soubory s tajnými klíči protokolu PPP, tedy soubory `pap-secrets` a `chap-secrets`. Nacházejí se v adresáři `/etc/ppp` a obsahují trojice klient, server a heslo, případně i seznam IP adres. Interpretace polí klienta a serveru je u protokolů CHAP a PAP odlišná a závisí také na tom, zda dokazujeme svou totožnost našemu protějšku, nebo zda požadujeme, aby on prokázal svou totožnost nám.

Soubor hesel protokolu CHAP

Když musíte nějakému serveru dokázat svou totožnost pomocí protokolu CHAP, vyhledá démon **pppd** v souboru chap-secrets položku, kde pole klienta odpovídá názvu lokálního systému a pole serveru názvu vzdáleného systému, který autentikaci požaduje. Když požadujete po protějšku, aby dokázal svou totožnost, budou role obrácené: démon **pppd** vyhledá položku, u které se pole klienta shoduje s názvem vzdáleného hostitele (zasílá se v odpovědi protokolu CHAP) a pole serveru je shodné s názvem místního hostitele.

Následuje příklad souboru `chap-secrets` pro bránu **vlager**[75]:

```
# Autentikační informace protokolu CHAP pro bránu vlager
#
# klient          server          heslo                adresa
#-------------------------------------------------------------------------
vlager.vbrew.com  c3po.lucas.com   "Use The Source Luke" vlager.vbrew.com
c3po.lucas.com    vlager.vbrew.com "arttoo! arttoo!"     c3po.lucas.com
*                 vlager.vbrew.com "TuXdrinksVicBitter"  pub.vbrew.com
```

75 Uvozovky nejsou součástí tajné informace, slouží pouze k zachování mezer, které v ní mohou být použity.

Když se brána **vlager** protokolem PPP spojí se systémem **c3po**, vyžadá si hostitel **c3po** autentikaci tím, že pošle výzvu protokolu CHAP. **pppd** na **vlageru** pak v souboru chap-secrets najde údaj, kde je klient **vlager.vbrew.com** a server **c3po.lucas.com**, takže najde první řádek, který vidíme v předchozím příkladu. Pak vytvoří odpověď z výzvy a hesla (Use The Source Luke) a pošle ji hostiteli **c3po**.

Démon **pppd** však také vytvoří autentikační výzvu pro **c3po**, která bude obsahovat jedinečný náhodně generovaný řetězec a plně kvalifikované jméno hostitele **vlager.vbrew.com**. Hostitel **c3po** vytvoří analogickým postupem odpověď a odešle ji zpět bráně **vlager**. Nyní démon **pppd** vyjme z odpovědi název klienta (**c3po.vbrew.com**) a vyhledá v souboru chap-secrets řádek, v němž je klientem **c3po** a serverem **vlager**. Této podmínce vyhovuje druhý řádek, takže démon **pppd** zkombinuje svou výzvu s tajnou informací arttoo! arttoo! a výsledek porovná s odpovědí, která přišla od hostitele **c3po**.

Volitelné čtvrté pole obsahuje seznam IP adres, které jsou přijatelné pro klienty uvedené v prvním poli. Adresy mohou být zadány v tečkové notaci nebo jako názvy hostitelů, které vyhledá resolver. Pokud by hostitel **c3po** během sjednávání spojení protokolem IPCP požadoval použití IP adresy, která není uvedena v tomto seznamu, bude požadavek zamítnut a spojení bude ukončeno. Hostitel **c3po** je tak omezen pouze na použití své vlastní IP adresy. Je-li pole s adresami prázdné, budou povoleny libovolné adresy; pokud je zde znak „-“, nemůže daný klient použít žádnou IP adresu.

Třetí řádek ve vzorovém souboru chap-secrets povoluje navázat spojení libovolnému hostiteli, protože hodnota * v poli klienta nebo serveru odpovídá libovolnému názvu hostitele. Jediným požadavkem je, že klient musí znát příslušnou tajnou informaci a musí použít adresu **pub.vbrew.com**. Položky se zástupnými znaky představující názvy hostitelů se mohou v souboru s tajnými informacemi objevit na kterémkoliv místě, protože démon **pppd** bude vždy používat ten řádek, který aktuální dvojici klient/server odpovídá nejlépe.

Za určitých okolností může démon **pppd** potřebovat pomoc při vytváření jmen. Jak jsme si již říkali, název vzdáleného hostitele dodá vždy protějšek ve výzvě nebo v odpovědi protokolu CHAP. Název lokálního hostitele bude implicitně zjištěn voláním funkce gethostname(2). Pokud jste nastavili název systému jako nekvalifikovaný název hostitele, pak musíte démonu **pppd** poskytnout název domény pomocí volby domain:

```
# pppd ... domain vbrew.com
```

Tato volba připojí k názvu **vlager** i název domény pivovaru u všech aktivit, které se vztahují k ověřování totožnosti. Další volbami, které mění představu démona **pppd** o názvu místního hostitele, jsou volby usehostname a name. Když na příkazovou řádku zadáte místní IP adresu pomocí dvojice *local:remote* a parametr *local* bude obsahovat název a nikoliv IP adresu, bude použit tento název.

Soubor hesel protokolu PAP

Soubor s tajnými informacemi protokolu PAP je velmi podobný souboru, který využívá protokol CHAP. První dvě pole vždy obsahují jméno uživatele a název serveru; třetí pole obsahuje tajnou informaci protokolu PAP. Když vzdálený počítač prokazuje svou totožnost, použije démon **pppd** ten záznam, kde server odpovídá místnímu hostiteli a uživatel odpovídá uživatelskému jménu poslanému vzdáleným systémem. Pokud prokazujeme svou totožnost my, použije démon **pppd** ten řádek, kde název serveru odpovídá názvu vzdáleného systému a odešle na tomto řádku uvedené uživatelské jméno a heslo.

Příklad souboru hesel protokolu PAP může vypadat takto:

```
# /etc/ppp/pap-secrets
#
# uživatel     server      heslo          adresa
vlager-pap     c3po        cresspahl      vlager.vbrew.com
c3po           vlager      DonaldGNUth    c3po.lucas.com
```

První řádek používáme při své autentikaci hostiteli **c3po**. Druhý řádek udává, jak se hostitel **c3po** autentikuje nám.

Název **vlager-pap** v prvním sloupci představuje jméno uživatele, které pošleme hostiteli **c3po**. Démon **pppd** implicitně odesílá jako uživatelské jméno název místního systému, parametrem user s uvedením jiného jména to však můžeme změnit.

Při výběru položky ze souboru pap-secrets musí démon **pppd** znát název vzdáleného hostitele. Protože neexistuje žádný způsob, jak by ho mohl zjistit, musíte mu ho předat na příkazovém řádku pomocí volby remotename, za níž následuje název hostitele vašeho protějšku. Abychom se pomocí výše uvedeného souboru mohli autentikovat hostiteli **c3po**, musíme na příkazovém řádku démona **pppd** doplnit následující volbu:

```
# pppd ... remotename c3po user vlager-pap
```

Ve čtvrtém poli (a ve všech následujících polích) můžete zadat, jaké adresy může daný hostitel používat, je to stejné jako při použití protokolu CHAP. Protějšek pak může používat pouze adresy z tohoto seznamu. Ve vzorovém příkladu požadujeme, aby hostitel **c3po** používal svou skutečnou IP adresu a žádnou jinou.

Protokol PAP představuje poměrně slabou metodu na ověření totožnosti, a proto se doporučuje, kdykoliv je to možné, používat místo něj protokol CHAP. Z toho důvodu zde nebudeme protokol PAP popisovat podrobněji. Další informace o jeho použití najdete na manuálové stránce `pppd(8)`.

Ladění nastavení protokolu PPP

Démon **pppd** bude implicitně zapisovat veškeré varování a chybové zprávy prostřednictvím démona **syslog**. Do souboru syslog.conf musíte přidat údaj, který tyto zprávy přesměruje do souboru nebo přímo na konzolu, jinak by je démon **syslog** jednoduše zrušil. Následující položka způsobí, že budou všechny zprávy posílány do souboru `/var/log/ppp-log`:

```
daemon.*                /var/log/ppp-log
```

Jestliže nastavení démona PPP nefunguje správně, nahlédněte do tohoto souboru. Pokud vám záznamy v tomto souboru nenapoví, můžete parametrem debug zapnout výpis podrobnějších ladicích informací. Tato volba nařídí démonu **pppd**, aby prostřednictvím démona **syslog** zaznamenával obsah všech přijatých a odeslaných řídicích paketů.

Konečně nejdrastičtějším způsobem je povolení ladicích informací na úrovni jádra operačního systému. To je možné provést spuštěním démona **pppd** s volbou kdebug. Za touto volbou následuje číselný údaj, který je vytvořen součtem následujících hodnot: 1 pro obecné ladicí informace, 2 pro vytištění obsahu všech příchozích rámců protokolu HDLC a 4 pro výpis všech odchozích rámců protokolu HDLC. Abyste mohli zachytávat ladicí zprávy jádra operačního systému, musíte buď spustit démona **syslogd**, který čte soubor `/proc/kmsg`, nebo démona **klogd**. Oba démoni směrují ladicí informace jádra démonu **syslog**.

Složitější konfigurace protokolu PPP

Nejběžnější aplikací protokolu PPP je jeho využití k připojení telefonní linkou k nějaké síti, jako je například Internet. Ne všem však toto základní použití stačí. V této části budeme hovořit o některých složitějších konfiguracích protokolu PPP v Linuxu.

PPP server

Aby se démon **pppd** choval jako server, stačí nastavit sériové tty zařízení tak, aby po přijetí příchozích dat správně volalo démona **pppd**. Jednou z možností jak to udělat je vytvořit speciální účet, řekněme **ppp**, a jako přihlašovací skript mu přidělit skript nebo program, který spustí démona **pppd** se správnými parametry. Alternativou, chcete-li použít protokoly PAP nebo CHAP, je použít program **mgetty** pro obsluhu modemu a využít jeho funkce „/AutoPPP/“.

Chcete-li server vytvořit přihlašovací metodou, stačí do souboru `/etc/passwd` přidat následující řádek[76]:

```
ppp:x:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin
```

Pokud váš systém podporuje stínová hesla, musíte dále do souboru `/etc/shadow` přidat:

```
ppp:!:10913:0:99999:7:::
```

Samozřejmě, použité hodnoty UID a GID závisí na tom, kdo chcete aby spojení vlastnil a jak je vytváříte. Pomocí příkazu **passwd** dále musíte pro tento účet vytvořit heslo.

Skript **ppplogin** by mohl vypadat takto:

```
#!/bin/sh
# ppplogin - skript spouštějící pppd po přihlášení
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

Příkaz **mesg** zabrání ostatním uživatelům na tento terminál zapisovat například příkazem **write**. Příkaz **stty** vypíná echo. Tento příkaz je nezbytný, jinak by totiž bylo vzdálenému systému zpět odesláno vše, co poslal on. Nejdůležitějším parametrem démona **pppd** je -detach, protože tím démonu zabráníme odpojit se od řízeného terminálu. Pokud bychom jej neuvedli, démon by se přepnul na pozadí a skript by se ukončil. To by vedlo k zavěšení linky a ukončení spojení. Parametr silent způsobí, že démon nejprve počká, co mu pošle vzdálený systém, a pak na to bude reagovat. Tato volba zabrání ukončení spojení kvůli vypršení časového limitu v případě, že spuštění klientského démona protokolu PPP trvá dlouho. Parametr modem zajistí, že démon bude řídit stavové linky modemu na sériovém portu. Tento parametr byste měli uvést vždy, když démon **pppd** komunikuje pomocí modemu. Parametr crtscts zapíná hardwarový handshaking.

Kromě této možnosti budete možná chtít použít i nějakou autentikační metodu, typicky uvedením parametru auth na příkazovém řádku démona **pppd** nebo v globálním konfiguračním souboru. Na manuálových stránkách naleznete rovněž popis toho, jak zapínat a vypínat konkrétní autentikační protokoly.

Pokud budete chtít použít program **mgetty**, stačí vám nastavit jej tak, aby podporoval to sériové zařízení, na němž máte připojen modem (viz část *Konfigurace démona mgetty* v kapitole 4). Dále v konfiguračním souboru nastavíte démona **pppd**, aby používal protokoly PAP a/nebo CHAP a konečně do souboru `/etc/mgetty/login.config` přidáte něco takového, jako:

76 Tuto operaci vám usnadní programy **useradd** nebo **adduser**, pokud je ovšem máte.

```
# Konfigurace mgetty tak, aby detekoval příchozí PPP volání
# a spouštěl démona pppd pro obsluhu spojení
/AutoPPP/ -      ppp   /usr/sbin/pppd auth -chap +pap login
```

První údaj představuje malé kouzlo umožňující detekovat, že příchozí volání je vedeno protokolem PPP. Velikost písmen v tomto řetězci nesmíte měnit, velká a malá písmena se rozlišují. Třetí sloupec udává uživatelské jméno, které se bude objevovat v příkazu **who**, když se někdo přihlásí. Zbytek řádku představuje příkaz pro spuštění démona. V našem případě nařizujeme autentikaci protokolem PAP, zakazujeme protokol CHAP a říkáme, že pro autentikaci uživatelů má být použit systémový soubor passwd. To je zřejmě nastavení, které bude ve většině případů vyhovovat. Jednotlivé parametry můžete uvést buď na příkazovém řádku, nebo v konfiguračním souboru démona **pppd**.

Následuje malý přehled operací, které byste měli provést, pokud chcete k vašemu počítači povolit vytáčený přístup protokolem PPP. Vždy si ověřte, že daný krok funguje a pak teprve přejděte k dalšímu:

1. Nakonfigurujte modem do režimu automatické odpovědi. U modemů kompatibilních se standardem Hayes to provedete příkazem jako ATS0=3. Pokud budete používat démona **mgetty**, není tento krok nutný.
2. Nastavte sériové zařízení nějakým **getty** příkazem tak, aby odpovídalo na příchozí volání. Typicky se používá démon **mgetty**.
3. Zvažte použitou autentikaci. Budete volajícího identifikovat protokolem PAP, CHAP, nebo systémovým přihlášením?
4. Výše popsaným postupem nastavte démona **pppd** jako server.
5. Rozmyslete si směrování. Musíte volajícímu poskytovat trasu k síti? Směrování můžete nastavit skriptem `ip-up`.

Vyžádané vytáčení

Jakmile se objeví nějaký IP provoz, který je zapotřebí přenést po telefonní lince, funkce *vyžádaného vytáčení* způsobí, že modem zavolá vzdálenému hostiteli a naváže s ním spojení. Vyžádané vytáčení je užitečné zejména v případech, kdy si nemůžete dovolit zůstat telefonní linkou trvale připojeni k poskytovateli. Například pokud platíte místní hovory časovým tarifem, může být levnější připojovat se pouze tehdy, když to potřebujete a odpojit se, jakmile Internet nepotřebujete[77].

Tradiční řešení používalo příkaz **diald**, který sice fungoval dobře, jeho nastavení však bylo velmi komplikované. Verze 2.3.0 a vyšší démona **pppd** mají vestavěnou podporu vyžádaného vytáčení, která se konfiguruje velmi snadno. Aby tato funkce fungovala, musíte rovněž používat moderní jádro. Bude vyhovovat kterékoliv jádro od 2.0 výše.

Nastavení démona **pppd** pro podporu vyžádaného vytáčení obnáší pouze uvedení příslušných parametrů v jeho konfiguračním souboru nebo na příkazovém řádku. Následující tabulka shrnuje parametry, které se k vyžádanému vytáčení vztahují.

77 Pozn. překladatele: Což u nás platí vždy.... V komunikačně nemonopolních Spojených státech existují telefonní společnosti, které například místní hovory neúčtují vůbec, nebo je zpoplatňují jednorázovým poplatkem. Škoda mluvit...

Parametr	Popis
`demand`	Tento parametr říká, že PPP linka má být použita v režimu vyžádaného vytáčení. Bude vytvořeno síťové zařízení, příkaz connect se však provede až když místní hostitel vygeneruje nějaký datagram. Tato volba je k nastavení vyžádaného vytáčení nezbytná.
`active-filter` *výraz*	Tato volba definuje, které pakety mají být chápány jako aktivní provoz. Jakýkoliv paket odpovídající zadanému výrazu vynuluje počitadlo neaktivity a způsobí tak, že démon **pppd** bude znovu čekat s ukončením spojení. Syntaxe nastavení filtru je odvozena od příkazu **tcpdump**. Implicitnímu nastavení vyhovují všechny datagramy.
`holdoff n`	Tento parametr nastavuje minimální čas v sekundách pro opětovné obnovení spojení od chvíle jeho neplánovaného ukončení. Když spojení vypadne a démon **pppd** je stále považuje za aktivní, bude obnoveno až po takto nastaveném čase. Toto nastavení nemá vliv na opětovné navázání spojení po jeho ukončení z důvodu neaktivity.
`idle n`	Je-li tento parametr nastaven, démon **pppd** ukončí spojení po uplynutí nastaveného počtu sekund neaktivity. Počitadlo se nuluje vždy, když se přenese aktivní paket.

Jednoduchá konfigurace pro vyžádané vytáčení by tedy mohla vypadat takto:

```
demand
holdoff 60
idle 180
```

Toto nastavení povolí vyžádané vytáčení, po výpadku spojení je bude obnovovat za 60 sekund a v případě neaktivity ukončí spojení po 180 sekundách.

Trvalé vytáčení

Trvalé vytáčení je funkce, kterou budou používat ti, již jsou telefonní linkou připojeni k síti trvale. Mezi vyžádaným vytáčením a trvalým vytáčením je drobný rozdíl. Při použití trvalého vytáčení je spojení navázáno ihned po spuštění démona **pppd** a „trvalost" se projeví v okamžiku, kdy se telefonní spojení neočekávaně přeruší. Tato funkce zajistí, že linka bude trvale funkční, protože v případě výpadku spojení bude spojení automaticky navázáno znovu.

Možná patříte mezi šťastlivce, kteří za telefonní připojení neplatí – možná voláte pouze místní hovory a váš operátor je poskytuje zdarma, nebo vám telefon platí zaměstnavatel. V těchto situacích je trvalé vytáčení velmi užitečné. Pokud za hovory platíte, musíte být trochu opatrnější. Platíte-li za hovory časovým tarifem, zřejmě vám trvalé vytáčení vyhovovat nebude, ledaže byste opravdu potřebovali být připojeni 24 hodin denně. Pokud platíte za hovory jednorázovou částkou a ne časovým tarifem, je třeba dávat pozor na to, aby modem trvale vytáčení neopakoval. Démon **pppd** nabízí parametry, které mohou pomoci řešení tohoto problému.

Pro aktivaci trvalého vytáčení musíte v některém konfiguračním souboru uvést parametr `persist`. Pouhým uvedením tohoto parametru zajistíte, že démon **pppd** bude automaticky navazovat spojení příkazem `connect` znovu vždy, když linka vypadne. Pokud by vám vadilo okamžité opakované vytáčení (například pokud vzdálený server nebo modem potřebují nějaký čas na zotavení z výpadku), můžete použít parametr `holdoff` a specifikovat tak dobu, po kterou démon počká, než se pokusí o nové navázání spojení. Tento parametr sice nedokáže úplně vyřešit problém plateb za neustálé telefonování v případě závažnější chyby, může však jeho dopad poněkud redukovat.

Typická konfigurace trvalého vytáčení bude vypadat takto:

```
persist
holdoff 300
```

Čas pozdržení se udává v sekundách. V našem případě tak démon po výpadku spojení počká pět minut a pak se pokusí spojení obnovit.

Je dokonce možné kombinovat trvalé vytáčení s vyžádaným vytáčením a parametrem idle definovat dobu na ukončení spojení v případě neaktivity, nicméně tato kombinace není úplně typická. Na manuálových stránkách démona pppd je stručně popsána i tato konfigurace.

KAPITOLA 9

TCP/IP Firewall

Bezpečnost je pro společnosti i jednotlivce stále důležitější. Internet poskytuje mocné nástroje k šíření informací o sobě a k získávání informací o jiných, zároveň však své uživatele vystavuje nebezpečím, kterých by jinak byli ušetřeni. Mezi možná nebezpečí patří počítačová kriminalita, krádeže informací a poškozování dat.

Neautorizovaná a bezzásadová osoba, která získá přístup k nějakému systému, může uhodnout systémové heslo nebo využít chyb nebo nestandardního chování některých programů k tomu, aby získala normální uživatelský účet. Jakmile má možnost se k počítači přihlásit, může mít přístup k informacím, které lze zneužít – například obchodně citlivé informace jako marketingové plány, podrobnosti o nových projektech nebo databáze zákazníků. Poškození nebo modifikace takovýchto údajů může firmě způsobit značné škody.

Nejspolehlivější metoda jak těmto nehodám zabránit spočívá v tom, že neautorizovaným osobám nebude přístup k počítači umožněn. Zde vstupují do hry firewally.

> **UPOZORNĚNÍ:** Nastavit bezpečný firewall je umění. Předpokládá to dobré pochopení technologie, zároveň to ale vyžaduje stejně dobré pochopení filozofie, na níž firewally stojí. V tomto textu nebudeme hovořit o všem, co potřebujete, rozhodně vám doporučujeme podniknout vlastní pečlivý průzkum předtím, než se rozhodnete pro nějaké řešení firewallu – včetně řešení, která uvádíme zde.

Informací potřebných pro návrh a nastavení dobrého firewallu je tolik, že by vydaly na samostatnou knihu. Existuje proto logicky celá řada knih, ve kterých si můžete své vědomosti o firewallech rozšířit. Dvě z nich jsou:

Building Internet Firewalls D. Chapmana a E. Zwickyho (O'Reilly). Tato kniha popisuje, jak navrhnout a nainstalovat firewall na systémech Unix, Linux a Windows NT a jak nakonfigurovat internetové služby, aby přes firewall fungovaly.

Firewalls and Internet Security W. Cheswicka a S. Bellovina (Addison Wesley). Tato kniha popisuje filozofie návrhu a implementace firewallů.

V této kapitole se zaměříme na technické podrobnosti specifické pro Linux. Později si uvedeme příklad konfigurace firewallu, která vám může posloužit jako základ pro vaši vlastní konfiguraci, nicméně – jako vždy, když je ve hře bezpečnost – nevěřte nikomu! Dvakrát zkontrolujte návrh, ujistěte se, že všemu rozumíte, a pak jej upravte podle svých potřeb. Abyste byli v bezpečí, musíte mít jistotu.

Metody útoků

Pro síťového administrátora je nutné, aby chápal podstatu možných útoků na počítačovou bezpečnost. Stručně si popíšeme jednotlivé typy útoků, abyste mohli přesně pochopit, před čím vás může firewall na Linuxu chránit. Kromě toho byste si měli nastudovat i další materiály, abyste se mohli chránit i před dalšími typy útoků. Dále shrnujeme některé nejvýznamnější metody útoků a metody, jak se proti nim chránit.

Neautorizovaný přístup

Znamená to jednoduše tolik, že osoby, které by neměly být schopny používat váš systém, se k němu mohou připojit a mohou jej použít. Například osoby mimo vaši firmu se mohou pokoušet o připojení k vašemu účetnímu počítači nebo internímu NFS serveru.

Existuje řada metod, jak se proti těmto útokům bránit, nutné je však přesně definovat, kdo má k čemu mít přístup. Není pak problém zakázat síťový přístup všem kromě těch, kteří jej mít mají.

Využití známých slabin programů

Některé programy a síťové služby nebyly původně navrženy s důrazem na bezpečnost a jsou tak zákonitě zranitelné. Příkladem jsou vzdálené služby BSD (rlogin, rexec a další).

Nejlepší metodou ochrany proti těmto útokům je vypnutí všech zranitelných služeb nebo jejich náhrada jinými alternativami. Při použití Open-Source programů je někdy možné některé slabiny odstranit úpravou zdrojového kódu.

Zablokování služby

Útok zablokování služby způsobí, že program nebo služba nebude fungovat nebo jej uživatelé nebudou moci použít. Tento typ útoku je možné provést na úrovni síťové vrstvy odesláním vhodně upravených vadných datagramů, které způsobí výpadek síťového spojení. Útok je rovněž možné provést na úrovni aplikační vrstvy, kdy pomocí vhodně zvolených příkazů dojde k přetížení služby nebo k jejímu selhání.

Útoky zablokováním služby je možné minimalizovat odfiltrováním podezřelého síťového provozu a podezřelých aplikačních příkazů a požadavků. Je dobré znát podrobnosti o používaných metodách a sledovat zprávy, které o nových typech útoků informují.

Spoofing

Tento typ útoků znamená, že počítač nebo aplikace se vydávají za někoho jiného. Typicky se útočník vydává za „přátelský“ systém tím, že falšuje IP adresy v datagramech. Například známá chyba v BSD službě rlogin umožňuje touto metodou předstírat připojení z jiného systému pomocí uhodnutí sekvenčních čísel v TCP paketech.

Jako ochranu proti těmto útokům ověřujte autenticitu datagramů a příkazů. Neprovádějte směrování datagramů s neplatnými zdrojovými adresami. Do mechanismů řízení spojení zaveďte nepředvídatelné prvky, využijte například náhodné volby sekvenčních čísel a přiřazovaných portů.

Odposlouchávání

Nejjednodušší typ útoku. Útočící systém je nastaven tak, aby přijímal data, která mu nepatří. Dobře napsaný odposlouchávací program může ze síťového provozu získat přihlašovací jména a hesla uživatelů. Na tento typ útoků jsou zejména náchylné vysílací sítě jako je Ethernet.

Jako obranu proti těmto útokům se vyhněte použití vysílaných technologií a přenášejte citlivá data v zašifrované podobě.

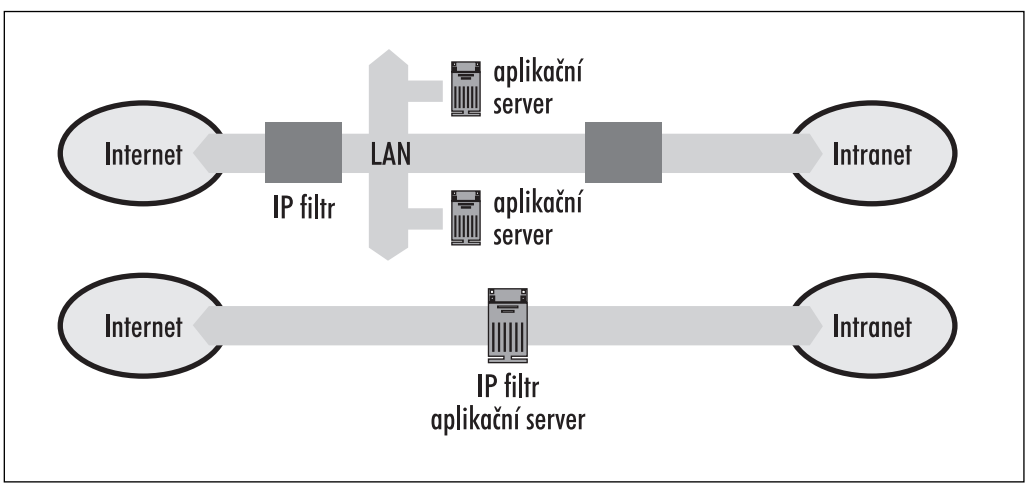
IP firewally jsou účinné jako ochrana před neautorizovaným přístupem, zablokováním služby na úrovni síťové vrstvy a spoofingem. Nejsou příliš účinné jako ochrana před využitím slabých míst v programech a proti odposlechu.

Co je to firewall?

Firewall je bezpečný a důvěryhodný počítač zapojený mezi privátní a veřejnou sítí[78]. Firewallový systém má nastavena pravidla udávající, jaký síťový provoz může propouštět a jaký má být zablokován nebo odmítnut. V některých velkých organizacích se firewally používají i uvnitř sítě jako ochrana citlivých oddělení organizace od ostatních zaměstnanců. Většina případů počítačové kriminality totiž pochází *zevnitř* organizace a nikoliv *zvenčí*.

Firewally je možné konstruovat různými metodami. Nejpokročilejší metoda používá několik samostatných systémů a rozděluje síť na různě zabezpečené úrovně. Dva počítače fungující jako filtry umožňují průchod pouze přesně definovaným typům provozu a mezi nimi jsou umístěny síťové servery, jako například poštovní brána, server WWW a podobně. Takováto konfigurace může být velice bezpečná a umožňuje snadno nastavit kdo se může připojit zvenčí dovnitř a zevnitř ven. Tento typ ochrany se obvykle používá ve velkých společnostech.

Typičtěji je firewall tvořen jediným počítačem, který zajišťuje vše. Jedná se o poněkud méně bezpečné řešení, protože pokud bude v samotném firewallu chyba, která umožní neautorizovaný přístup k němu, může být narušena celá bezpečnost sítě. Nicméně tento typ firewallu je levnější a snáze spravovatelný než složitější výše popsané uspořádání. Obrázek 9.1 znázorňuje oba typy konfigurace.



Obrázek 9.1 – Hlavní metody návrhu firewallu

Jádro Linuxu obsahuje řadu vestavěných funkcí, které mu umožňují pracovat jako účinný IP firewall. Implementace síťových služeb obsahuje funkce, které umožňují řadou různých způsobů konfigurovat filtrování paketů, a zajišťují mechanismy, jež umožní přesně nastavit, jaká pravidla chce-

78 Termín *firewall* pochází z oblasti protipožární ochrany. Jedná se o bariéru z nehořlavého materiálu, umístěnou mezi potenciálním zdrojem požáru a jeho okolím.

te použít. Firewall založený na Linuxu je natolik flexibilní, že může být použit v obou konfiguracích podle obrázku 9.1. Síťový software Linuxu obsahuje i další funkce, které se k firewallům vztahují – IP účtování (viz kapitola 10) a IP maškaráda (viz kapitola 11).

Co je to filtrování?

Filtrování je jednoduše mechanismus, který rozhoduje o tom, které IP datagramy mají být zpracovány normálně a které mají být zrušeny. *Zrušením* rozumíme, že datagram bude smazán a úplně ignorován, jako kdyby vůbec nebyl přijat. K rozhodování o tom, které datagramy zpracovávat a které zahazovat je možné použít celou řadu kritérií, například:

- Typ protokolu: TCP, UDP, ICMP a podobně
- Číslo portu (pro TCP/UDP)
- Typ datagramu: SYN/ACK, data, ICMP Echo Request a podobně
- Zdrojovou adresu datagramu – odkud pochází
- Cílovou adresu datagramu – kam má dojít

Důležité je chápat, že filtrování se odehrává na úrovni síťové vrstvy. Filtrování neví nic o aplikacích, které síťové spojení používají, stará se pouze o samotné spojení. Můžete například uživatelům zakázat přístup do vaší interní sítě na standardním portu telnetu, pokud ovšem používáte pouze filtrování, nemůžete jim zabránit použít telnet pro připojení k jinému portu, ke kterému toto připojení povolujete. Tomuto typu problémů můžete předejít pomocí proxy serverů pro všechny služby, které firewallem povolujete. Proxy server rozumí aplikaci, kterou obsluhuje a může proto zabránit zneužitím, například použít telnet pro průchod přes firewall prostřednitvím WWW-portu. Pokud bude váš firewall podporovat funkci WWW-proxy, veškerá připojení na port WWW bude obsluhovat proxy server a povolí průchod pouze legitimním HTTP požadavkům. Existuje celá řada proxy serverů. Některé z nich jsou dostupné zdarma, k dispozici je i řada komerčních produktů. Některé oblíbené popisuje dokument Firewall-HOWTO, my se jimi v tomto textu nebudeme zabývat.

Filtrační pravidla jsou sestavena z různých kombinací výše uvedených kritérií. Můžete například chtít, aby uživatelé na síti pivovaru neměli na Internetu přístup k ničemu jinému než ke službě WWW. Nakonfigurujete tedy firewall tak, aby povoloval průchod:

- datagramům se zdrojovou adresou sítě pivovaru, libovolnou cílovou adresou a cílovým portem 80 (služba WWW),
- datagramům s libovolnou zdrojovou adresou, zdrojovým portem 80 (služba WWW) a cílovou adresou kdekoliv v síti pivovaru

Všimněte si, že používáme dvě pravidla. Potřebujeme totiž, aby naše data prošla ven, ale zároveň aby data zvnějšku mohla projít dovnitř. Jak za chvíli uvidíme, Linux konstrukci takovýchto pravidel usnadňuje a umožňuje obě dvě zadat jediným příkazem.

Vytvoření firewallu na Linuxu

K vytvoření firewallu na Linuxu potřebujete sestavit jádro s podporou firewallu a dále potřebujete odpovídající konfigurační nástroj. V jádrech před verzí 2.2 je tímto nástrojem **ipfwadm**. V jádrech 2.2.*x* byla uvedena třetí generace IP firewallů, nazvaná *IP Chains*. Administrují se nástrojem **ipchains**, který je podobný nástroji **ipfwadm**. Jádra 2.3.15 a pozdější podporují čtvrtou generaci

firewallů, pojmenovanou *netfilter*. Systém *netfilter* je systém mnoha tváří, poskytuje zpětnou kompatibilitu s nástroji **ipfwadm** i **ipchains** a nabízí i nový nástroj **iptables**. O rozdílech mezi těmito třemi systémy budeme hovořit za chvíli.

Konfigurace jádra pro firewall

Aby mohl Linux fungovat jako firewall, musí tuto funkci podporovat jádro. Nastavení této podpory je velmi jednoduché, po spuštění make menuconfig stačí zadat potřebné volby[79]. Jak se jádro konfiguruje jsme popisovali v kapitole 3. V jádrech 2.2 musíte zadat následující volby:

```
Networking options  --->
       [*] Network firewalls
       [*] TCP/IP networking
       [*] IP: firewalling
       [*] IP: firewall packet logging
V jádrech 2.4.0 a novějších volíte namísto toho následující nastavení:
Networking options  --->
     [*] Network packet filtering (replaces ipchains)
         IP: Netfilter Configuration  --->
             .
            <M> Userspace queueing via NETLINK (EXPERIMENTAL)
            <M> IP tables support (required for filtering/masq/NAT)
            <M>   limit match support
            <M>   MAC address match support
            <M>   netfilter MARK match support
            <M>   Multiple port match support
            <M>   TOS match support
            <M>   Connection state match support
            <M>   Unclean match support (EXPERIMENTAL)
            <M>   Owner match support (EXPERIMENTAL)
            <M>   Packet filtering
            <M>     REJECT target support
            <M>     MIRROR target support (EXPERIMENTAL)
             .
            <M>   Packet mangling
            <M>     TOS target support
            <M>     MARK target support
            <M>   LOG target support
            <M> ipchains (2.2-style) support
            <M> ipfwadm (2.0-style) support
```

Nástroj ipfwadm

Nástroj **ipfwadm** (IP Firewall Administration) slouží k vytváření pravidel firewallu v jádrech před verzí 2.2.0. Syntaxe příkazů je poněkud matoucí, protože příkazy dokáží provádět velmi komplikované operace, ukážeme si nicméně některé typické příklady ilustrující nejobvyklejší způsoby použití tohoto nástroje.

Nástroj **ipfwadm** bývá součástí většiny moderních distribucí Linuxu, i když se obvykle neinstaluje implicitně. Zřejmě budete muset nainstalovat nějaký balík, který tento nástroj obsahuje. Pokud

79 Funkce *firewall packet logging* je speciální funkce, která na zvolené zařízení zapisuje informace o datagramech, jež vyhovují pravidlům firewallu, takže můžete sledovat provoz, jenž firewallem prochází.

ve vaší distribuci tento nástroj není, zdrojový balík můžete získat ze serveru *ftp.xos.nl* v `/pub/linux/ipfwadm` a můžete si program přeložit sami.

Nástroj ipchains

Stejně jako **ipfwadm** i nástroj **ipchains** může při prvním setkání působit zmateně. Je stejně pružný jako nástroj **ipfwadm**, navíc má zjednodušenou syntaxi příkazů a nabízí mechanismus „zřetězení“, který umožňuje spravovat více sad pravidel a spojovat je dohromady. Mechanismus zřetězování pravidel popíšeme v samostatné části ke konci kapitoly, protože pro většinu situací není tato funkce nutná.

Příkaz **ipchains** se objevuje ve většině distribucí Linuxu založených na jádře 2.2 a novějších. Pokud si jej chcete přeložit sami, najdete zdrojový kód na adrese *http://www.rustcorp.com/linux/ipchains/*. Součástí zdrojového balíku je i „maskovací“ skript **ipfwadm-wrapper**, který se tváří jako program **ipfwadm**, provádí však převod parametrů a volá nástroj **ipchains**. Díky tomu je migrace ze starší verze firewallu poměrně pohodlná.

Nástroj iptables

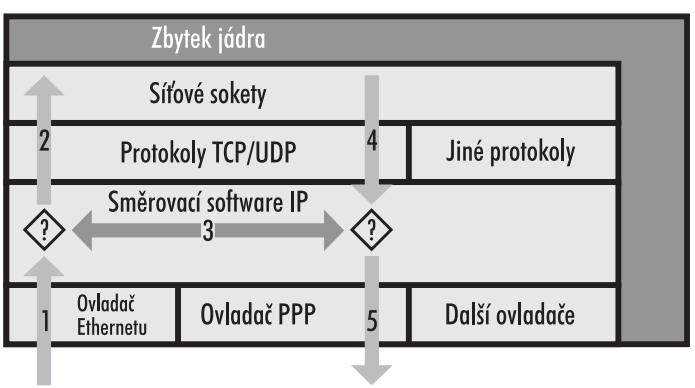
Syntaxe nástroje **iptables** se podobá syntaxi nástroje **ipchains**. Rozdíly vyplývají z různých vylepšení a z nového návrhu nástroje, který umožňuje jeho použití prostřednictvím sdílených knihoven. Stejně jako u nástroje **ipchains**, i pro nástroj **iptables** uvedeme stejné příklady, takže si budete moci porovnat syntaxi jednotlivých příkazů.

Nástroj **iptables** je součástí zdrojového balíku *netfilter* na adrese *http://www.samba.org/netfilter/*. Pravděpodobně se bude nacházet i v distribucích Linuxu založených na jádře 2.4.

O systému *netfilter* budeme hovořit v samostatné části úplně na konci této kapitoly.

Tři způsoby realizace filtrace

Uvědomme si, jakým způsobem linuxový počítač, respektive jakýkoliv počítač schopný směrovat IP pakety, zpracovává docházející datagramy. Základní kroky, znázorněné na obrázku 9.2, jsou:



Obrázek 9.2 – Fáze zpracování IP datagramu

- Přijetí IP datagramu (1).
- Datagram se zkontroluje, zda je určen pro proces na tomto počítači.
- Pokud datagram patří tomuto počítači, zpracuje se lokálně (2).
- Pokud datagram není určen tomuto počítači, prohlédne se směrovací tabulka a hledá se vhodná trasa. Datagram se pak předá správnému rozhraní nebo se zahodí, pokud pro něj není známá trasa (3).
- Datagramy od lokálních procesů se posílají směrovacímu systému, který je předá správnému rozhraní (4).
- Odcházející datagram se zkontroluje, zda je pro něj k dispozici trasa a pokud ne, zahodí se.
- Odeslání IP datagramu (5).

V našem diagramu reprezentuje trasa 1-3-5 směrování datagramu od počítače z lokální ethernetové sítě na počítač dosažitelný linkou PPP. Trasy 1-2 a 4-5 představují příjem a odeslání datagramů síťovým procesem na našem počítači. Trasa 4-3-2 představuje tok dat lokálním zpětnovazebním rozhraním. Data typicky přicházejí a odcházejí různými síťovými rozhraními. Otazníky ve schématu označují místa, kdy musí IP vrstva provést rozhodnutí o směrování.

Firewall v jádře Linuxu dokáže filtrovat na různých místech tohoto procesu. Znamená to, že můžete filtrovat datagramy přicházející na váš počítač, datagramy, které váš počítač směruje, i datagramy, které jsou odesílány.

V nástrojích **ipfwadm** a **ipchains** odpovídají pravidla třídy Input trase 1 v diagramu, třída Forwarding odpovídá trase 3 a třída Output trase 5. Až budeme později hovořit o programu *netfilter*, uvidíme, že v něm se místa zpracování změnila a třída Input odpovídá trase 2 a třída Output trase 4. Má to zásadní dopad na způsob sestavování pravidel, nicméně základní principy zůstávají ve všech verzích firewallů stejné.

Celé uspořádání může na první pohled vypadat komplikovaně, zajišťuje však flexibilitu, která umožňuje vytvářet velmi složité a mocné konfigurace.

Původní IP firewall (jádra 2.0)

První generace podpory firewallu v Linuxu se objevila v jádrech série 1.1. Jednalo se o přenos BSD firewallu ipfw na platformu Linux, který provedl Alan Cox. Podpora v jádrech 2.0 pak byla druhá generace podpory a na jejím vylepšení se podíleli Jos Vos, Pauline Middelink a další.

Použití programu ipfwadm

Příkaz **ipfwadm** sloužil jako konfigurační nástroj druhé generace linuxových firewallů. Asi nejlepší způsob, jak si tento nástroj popsat, bude uvedení příkladu. Začneme tím, že zkusíme nastavit příklad, o kterém jsme se už zmínili.

Základní příklad

Řekněme, že máme nějakou organizaci se sítí a pro přístup k Internetu používáme firewall založený na Linuxu. Dále předpokládejme, že chceme našim uživatelům povolit přístup k webovým serverům na Internetu, nechceme však propouštět jakýkoliv jiný provoz.

Zavedeme tedy pravidla, která povolí předávání datagramů se zdrojovou adresou pocházející z naší sítě na cílový port 80, a dále předávání odpovídajících datagramů s odpovědí.

Předpokládejme, že naše síť používá 24bitovou síťovou masku (síť třída C) a adresu 172.16.1.0. Můžeme pak použít následující pravidla:

```
# ipfwadm -F -f
# ipfwadm -F -p deny
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80
# ipfwadm -F -a accept -P tcp -S 0/0 80 -D 172.16.1.0/24
```

Parametr -F říká programu **ipfwadm**, že se jedná o pravidlo třídy Forwarding. První příkaz říká programu **ipfwadm**, aby zrušil všechna pravidla této třídy. Tím si zajistíme, že začneme přidávat naše pravidla za přesně definovaného stavu firewallu.

Druhé pravidlo definuje implicitní předávací politiku. Říkáme jádru, že nepovolujeme předávání IP datagramů. Nastavení implicitní politiky je velice důležité, protože říká jádru co má dělat, pokud přijme datagram, který není popsán žádným z pravidel. U většiny konfigurací firewallu nastavujeme implicitní politiku na zákaz předávání, čímž zajistíme, že firewall nepředá nic, co jsme mu explicitně nepřikázali předávat.

Třetí a čtvrtý řádek představují implementaci našich pravidel. Třetí pravidlo umožňuje definovaným datagramům opouštět naši síť, čtvrté pravidlo umožňuje příjem definovaných datagramů.

Podívejme se na jednotlivé parametry:

-F	Jedná se o předávací pravidlo.
-a accept	Pravidlo přidáváme s politikou „accept“, tedy povolujeme předávání datagramů, které pravidlu vyhovují.
-P tcp	Pravidlo platí pro datagramy protokolu TCP (nikoliv pro protokoly UDP a ICMP).
-S 172.16.1.0/24	Prvních 24 bitů zdrojové adresy musí odpovídat adrese 172.16.1.0.
-D 0/0 80	Nula bitů cílové adresy musí odpovídat adrese 0.0.0.0. Což je jenom „stručnější“ výraz, jak říct „cokoliv“. Číslo 80 představuje cílový port, tedy službu WWW. Místo čísla portu můžete použít i název služby definovaný v souboru `/etc/services`, takže parametr `-D 0/0 www` bude znamenat to samé.

Program **ipfwadm** pracuje se síťovou maskou ve tvaru, který není úplně obvyklý. Zápis */nn* znamená, kolik bitů adresy je významných, tedy velikost masky. Významné bity se počítají zleva doprava, některé typické příklady uvádí tabulka 9.1.

Tabulka 9.1 – Typické hodnoty síťové masky

Maska	Bitů
255.0.0.0	8
255.255.0.0	16
255.255.255.0	24
255.255.255.128	25
255.255.255.192	26
255.255.255.224	27
255.255.255.240	28
255.255.255.248	29
255.255.255.252	30

Už jsme se zmínili o tom, že program **ipfwadm** nabízí malý trik, který usnadňuje zadávání tohoto typu pravidel. Tímto trikem je parametr -b, který znamená obousměrné pravidlo.

Příznak obousměrného pravidla nám dovoluje sloučit dvě pravidla do jediného takto:

```
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80 -b
```

Důležité vylepšení

Podívejme se na naše pravidla pečlivěji. Všimli jste si, že jsme stále ponechali jednu možnost, jak může někdo zvenku náš firewall překonat?

Naše pravidla povolují propustit zvenčí všechny datagramy ze zdrojového portu 80. Sem ovšem spadají i datagramy s nastaveným bitem SYN! Bit SYN definuje, že TCP datagram je žádost o spojení. Pokud má útočník privilegovaný přístup k nějakému počítači, může se přes náš firewall připojit k jakémukoliv počítači, stačí jenom, aby odesílal své datagramy z portu 80. To jsme ovšem rozhodně nechtěli.

Naštěstí je náš problém řešitelný. Příkaz **ipfwadm** obsahuje další příznak, který umožňuje vytvářet pravidla vztahující se na datagramy s nastaveným bitem SYN. Změníme proto náš příklad a nastavíme pravidla takto:

```
# ipfwadm -F -a deny -P tcp -S 0/0 80 -D 172.16.10.0/24 -y
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80 -b
```

Parametr -y říká, že pravidlo platí pro datagramy s nastaveným příznakem SYN. Naše nové pravidlo tedy říká „zakaž všechny datagramy určené pro naši síť pocházející odkudkoliv z portu 80 s nastaveným příznakem SYN“ – jinak řečeno „zakaž všechny žádosti o otevření spojení přicházející z portu 80“.

Proč jsme toto nové pravidlo umístili *před* naše původní pravidlo? IP firewall funguje tak, že použije první vyhovující pravidlo. Datagramy, které chceme zakázat, vyhovují *oběma* pravidlům, proto musí být zakazující pravidlo uvedeno *před* pravidlem povolujícím.

Výpis pravidel

Po zadání pravidel můžeme následujícím příkazem požádat o jejich vypsání:

```
# ipfwadm -F -l
```

Toto pravidlo vypíše všechna nastavená předávací pravidla. Výstup bude vypadat takto nějak:

```
# ipfwadm -F -l
IP firewall forward rules, default policy: deny
type  prot source              destination         ports
deny  tcp  anywhere            172.16.10.0/24      www -> any
acc   tcp  172.16.1.0/24       anywhere            any -> www
```

Příkaz **ipfwadm** se pokusí přeložit čísla portů na názvy služeb pomocí souboru `/etc/services`.

Na výpisu ovšem nevidíme některé důležité podrobnosti. Nevidíme zde například efekt parametru -y. Příkaz **ipfwadm** dokáže generovat i podrobnější výstup po zadání parametru -e. Neukážeme si jej celý, protože je příliš dlouhý, než aby se vešel na stránku, nicméně tento výpis obsahuje i sloupec opt (options), který uvádí nastavení příznaku -y pro potlačení paketů SYN:

```
# ipfwadm -F -l -e
IP firewall forward rules, default policy: deny
 pkts bytes type  prot opt  tosa tosx ifname  ifaddress   source        ...
    0     0 deny  tcp  --y- 0xFF 0x00 any     any         anywhere      ...
    0     0 acc   tcp  b--- 0xFF 0x00 any     any         172.16.1.0/24 ...
```

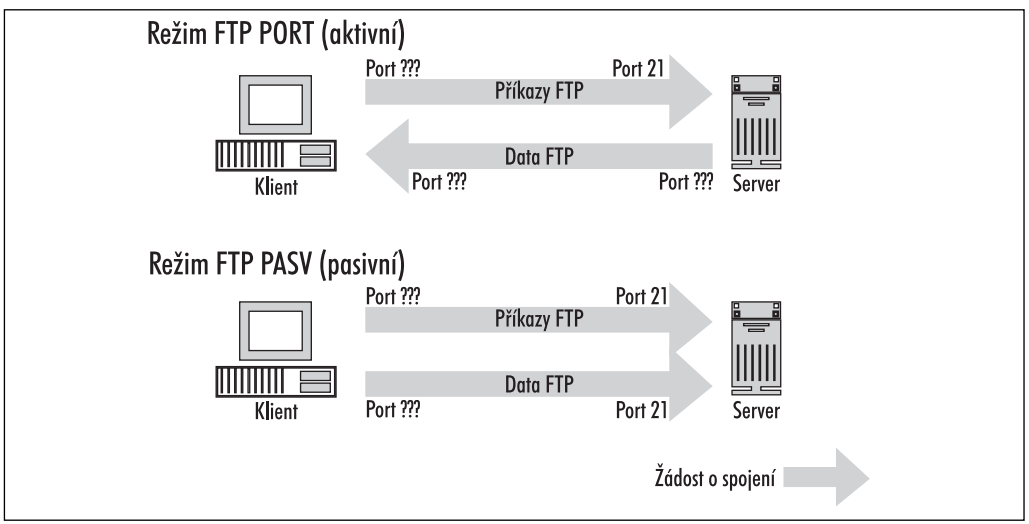
Složitější příklad

Předchozí příklad byl velmi jednoduchý. Ne všechny síťové služby se konfigurují tak snadno jako služba WWW, ve skutečnosti bývá konfigurace typického firewallu podstatně složitější. Podívejme se nyní na další typický příklad, na službu FTP. Chceme, aby se uživatelé naší interní sítě mohli přihlásit k FTP serverům na Internetu a číst a zapisovat soubory. Nechceme ale, aby se někdo z Internetu mohl přihlásit k našim interním FTP serverům.

Víme, že služba FTP používá dva TCP porty: port 20 (ftp-data) a port 21 (ftp), takže:

```
# ipfwadm -a deny -P tcp -S 0/0 20 -D 172.16.1.0/24 -y
# ipfwadm -a accept -P tcp -S 172.16.1.0/24 -D 0/0 20 -b
#
# ipfwadm -a deny -P tcp -S 0/0 21 -D 172.16.1.0/24 -y
# ipfwadm -a accept -P tcp -S 172.16.1.0/24 -D 0/0 21 -b
```

V pořádku? Ne nutně! Servery FTP mohou pracovat ve dvou různých režimech: pasivním a aktivním[80]. V pasivním režimu FTP server čeká na připojení od klienta. V aktivním režimu sám navazuje spojení s klientem. Standardně se obvykle používá aktivní režim. Rozdíl mezi oběma režimy ilustruje obrázek 9.3.



Obrázek 9.3 – Režimy práce FTP serveru

Většina FTP serverů při práci v aktivním režimu navazuje spojení z portu 20, což nám poněkud usnadňuje situaci, bohužel se tak nechovají všechny[81].

Co to pro nás znamená? Podívejme se na pravidlo týkající se portu 20, tedy datového portu FTP. Tak jak máme toto pravidlo definováno předpokládá, že náš klient bude navazovat spojení se serverem. To bude platit v pasivním režimu. Bude ovšem velmi obtížné nakonfigurovat firewall pro podporu aktivního režimu, protože nemůžeme spolehlivě vědět, které porty budou použity. Pokud bychom povolili příchozí spojení na všech portech, vystavili bychom tak útokům všechny služby, které spojení přijímají.

80 Aktivní režim se (poněkud neintuitivně) zapíná příkazem **PORT**. Pasivní režim se zapíná příkazem **PASV**.

81 Nechová se tak například démon ProFTPd, přinejmenším ve starších verzích.

Problém nejbezpečněji vyřešíme tak, že povolíme použití služby FTP pouze v pasivním režimu. Většina FTP serverů a řada FTP klientů tento režim podporují. I oblíbený klient **ncftp** podporuje pasivní režim, nicméně bude nutné jej pro tento režim nakonfigurovat. I řada webových prohlížečů, například Netscape, podporuje pasivní režim, takže by neměl být problém najít software, který bude této podmínce vyhovovat. Celý problém můžete také sprovodit ze světa tím, že na firewall nainstalujete FTP proxy server, který bude přijímat spojení z interní sítě a bude navazovat spojení na Internet.

Při nastavování firewallu obvykle narazíte na celou řadu podobných problémů. Vždy musíte dávat velký pozor na to, jak služba přesně pracuje, abyste mohli zavést všechna potřebná pravidla. Úplná konfigurace firewallu může být velice složitá.

Přehled parametrů programu ipfwadm

Příkaz **ipfwadm** pracuje s celou řadou parametrů, které se vztahují ke konfiguraci IP firewallu. Obecná syntaxe je:

`ipfwadm` *`kategorie příkaz parametr [volby]`*

Podívejme se podrobněji na jednotlivé části.

Kategorie

Musí být uveden právě jeden z následujících parametrů. Tímto parametrem firewallu říkáme, který typ pravidel modifikujeme.

`-I`	Vstupní pravidlo (třída Input).
`-O`	Výstupní pravidlo (třída Output).
`-F`	Předávací pravidlo (třída Forward).

Příkaz

Musí být uveden alespoň jeden z následujících parametrů. Bude se vztahovat pouze k pravidlům zvolené kategorie. Příkazy firewallu říkají, co má udělat.

`-a [`*`politika`*`]`	Přidá nové pravidlo na konec seznamu pravidel.
`-i [`*`politika`*`]`	Vloží pravidlo do seznamu pravidel.
`-d [`*`politika`*`]`	Vymaže pravidlo ze seznamu pravidel.
`-p` *`politika`*	Nastaví implicitní politiku.
`-l`	Vypíše všechna pravidla.
`-f`	Smaže všechna pravidla.

Možná nastavení politiky jsou:

`accept`	Umožní příjem, odeslání nebo předání vyhovujícího datagramu.
`deny`	Zablokuje příjem, odeslání nebo předání vyhovujícího datagramu.
`reject`	Zablokuje příjem, odeslání nebo předání vyhovujícího datagramu a zároveň odesílajícímu hostiteli odešle ICMP zprávu o chybě.

Parametr

Musí být uveden alespoň jeden z následujících parametrů. Pomocí parametrů se definuje, pro které datagramy má pravidlo platit.

`-P` *protokol*	Může specifikovat protokoly TCP, UDP nebo ICMP. Například `-P tcp`
`-S` *adresa*`[/`*maska*`]` `[`*port*`]`	Zdrojová adresa, pro niž pravidlo platí. Pokud neuvedete masku, předpokládá se maska „/32“. Můžete rovněž zadat port, jehož se pravidlo týká. Aby pravidlo fungovalo, musíte parametrem `-P` definovat protokol. Pokud neuvedete port nebo rozsah portů, vyhovují pravidlu všechny porty. Porty mohou být definovány svými názvy podle souboru `/etc/services`. V případě protokolu ICMP slouží údaj o portu ke specifikaci typu ICMP datagramu. Je možné definovat i rozsahy portů zadáním *od_portu*`:`*po_port*. Například: `-S 172.29.16.1/24 ftp:ftp_data`
`-D` *adresa*`[/`*maska*`]` `[`*port*`]`	Cílová adresa, pro niž pravidlo platí. Pro zadání cílové adresy platí stejná pravidla jako pro zadání zdrojové adresy. Například: `-D 172.29.16.1/24 smtp`
`-V` *adresa*	Udává adresu síťového rozhraní, na němž je paket přijímán (třída `-I`) nebo odesílán (třída `-O`). Tím můžeme vytvářet pravidla vztahující se k určitým síťovým rozhraním, například: `V 172.29.16.1`
`-W` *název*	Udává název síťového rozhraní. Parametr se chová stejně jako `-V`, pouze namísto adresy rozhraní zadáváte jeho název. Například: `-W ppp0`

Nepovinné volby

Následující volby mohou být občas velmi užitečné:

`-b` Obousměrné pravidlo. Tento příznak zahrnuje provoz oběma směry mezi zadaným cílem a zdrojem. Ušetří nám zadávání dvou pravidel – jednoho pro příchozí a druhého pro odchozí provoz.

`-o` Zapíná záznam vyhovujících datagramů do logu jádra. Všechny datagramy vyhovující nějakému pravidlu budou zaznamenány jako zpráva jádra. Tento parametr je užitečný k detekci pokusů o neautorizovaný přístup.

`-y` Používá se k zachycení žádostí o navázání spojení. Použití této volby způsobí, že pravidlu budou vyhovovat pouze TCP datagramy s žádostí o navázání spojení. Vyhovují mu tedy datagramy s nastaveným příznakem SYN a nenastaveným příznakem ACK. Je užitečná k odfiltrování požadavků o navázání TCP spojení, pro jiné protokoly se ignoruje.

`-k` Používá se k zachycení potvrzujících datagramů. Pravidlu vyhovují pouze TCP datagramy potvrzující přijetí žádosti o navázání spojení. Vyhovují mu pouze TCP datagramy s nastaveným příznakem ACK. Používá se k odfiltrování požadavků o navázání TCP spojení, pro jiné protokoly se ignoruje.

Typy ICMP datagramů

Konfigurační příkazy firewallu umožňují specifikovat typy ICMP datagramů. Na rozdíl od portů TCP a UDP neexistuje konfigurační soubor, který by definoval typy ICMP datagramů a jejich popis. Typy ICMP datagramů jsou definovány v dokumentu RFC-1700 Assigned Numbers. Kromě toho jsou typy ICMP datagramů uvedeny v hlavičkových souborech standardní knihovny C. Definuje je soubor `/usr/include/netinet/ip_icmp.h`, který je součástí standardního balíku GNU

knihovny a používají jej programátoři při vytváření programů pracujících s protokolem ICMP. Uvádíme je v tabulce 9.2. Rozhraní programu **iptables** umožňuje definovat typy ICMP datagramů i názvem, proto ve druhém sloupci uvádíme i tyto názvy.

Tabulka 9.2 – Typy ICMP datagramů

Typ číslo	Název v iptables	Název typu
0	echo-reply	Echo Reply
3	destination-unreachable	Destination Unreachable
4	source-quench	Source Quench
5	redirect	Redirect
8	echo-request	Echo Request
11	time-exceeded	Time Exceeded
12	parameter-problem	Parameter Problem
13	timestamp-request	Timestamp Request
14	timestamp-reply	Timestamp Reply
15	nemá	Information Request
16	nemá	Information Reply
17	address-mask-request	Address Mask Request
18	address-mask-reply	Address Mask Reply

IP Firewall Chains (jádra 2.2)

Většina vlastností v Linuxu se vyvíjí tak, aby odpovídaly požadavkům uživatelů. Ani firewally nejsou výjimkou. Klasická implementace IP firewallu ve většině případů vyhovuje, nicméně při konfiguraci složitějších prostředí může být komplikovaná a neefektivní. Z toho důvodu byla uvedena nová metoda konfigurace IP firewallu a byly uvedeny nové funkce. Tato nová metoda se označuje jako „IP Firewall Chains“ a poprvé byla obecně uvolněna v jádře 2.2.0.

Systém IP Firewall Chains vytvořili Paul Russel a Michael Neuling[82]. Dokumentaci k programu IP Firewall Chains vytvořil Paul v dokumentu IPCHAINS-HOWTO.

IP Firewall Chains vám umožňuje vytvářet třídy firewallových pravidel, do kterých pak můžete přidávat nebo odebírat počítače nebo sítě. Výhodou takového řetězení pravidel je, že mohou zvýšit výkon firewallu v případě, že obsahuje velké množství pravidel.

IP Firewall Chains je podporováno v jádrech 2.2 a existuje i jako patch pro jádra 2.0.*. Zmíněný dokument HOWTO popisuje, kde patch získat a uvádí také řadu užitečných informací o tom, jak efektivně pracovat s konfiguračním nástrojem **ipchains**.

Použití nástroje ipchains

Existují dvě možnosti, jak nástroj **ipchains** použít. První způsob používá skript **ipfwadm-wrapper**, což je v podstatě náhrada programu **ipfwadm**, která transparentně spouští program **ipchains**. Pokud jej budete používat, nemusíte číst dále. Místo toho si znovu přečtěte předcházející popis programu **ipfwadm** a všude jej nahraďte názvem **ipfwadm-wrapper**. Toto řešení je pl-

82 Paula můžete kontaktovat na adrese *Paul.Russel@rustcorp.com.au*.

ně funkční, není ale zaručeno, že skript bude dále vyvíjen a nemůžete také využít výhody, které technologie Firewall Chains nabízí.

Druhá možnost, jak program **ipchains** použít, spočívá v jeho volání s tím, že se naučíte novou syntaxi a upravíte pravidla tak, aby namísto staré syntaxe používala novou. Při troše opatrnosti můžete také zjistit, že při převodu starých pravidel na nová je můžete částečně optimalizovat. Syntaxe programu **ipchains** je jednodušší než u programu **ipfwadm**, takže toto řešení doporučujeme.

Program **ipfwadm** konfiguroval firewall pomocí tří tříd pravidel. Pomocí Firewall Chains můžete vytvářet libovolný počet tříd, které budou navzájem propojeny, nicméně vždy budou přítomny tři základní třídy. Tyto standardní třídy přesně odpovídají třídám programu **ipfwadm**, jsou však pojmenovány `input`, `forward` a `output`.

Podívejme se nejprve na obecnou syntaxi příkazu **ipchains** a pak si ukážeme, jak **ipchains** použít namísto **ipfwadm**, aniž bychom se zabývali pokročilejšími možnostmi tohoto programu. Budeme se držet našich původních příkladů a přepíšeme je pro novou syntaxi.

Syntaxe příkazu ipchains

Syntaxe příkazu **ipchains** je velmi jednoduchá. Podívejme se na její nejdůležitější rysy. Obecně se **ipchains** spouští takto:

`ipchains` *`příkaz specifikace_pravidla volby`*

Příkazy

Existuje celá řada způsobů, jak programem **ipchains** manipulovat s pravidly a s třídami pravidel. Podívejme se na ty, které se vztahují k IP firewallům.

`-A` *`třída`*	Přidává na konec třídy jedno nebo více pravidel. Pokud je jako zdroj nebo cíl uveden název počítače a ten má přiřazeno více IP adres, budou přidána pravidla pro všechny adresy[83].
`-I` *`třída č_prav`*	Přidá jedno nebo více pravidel na začátek třídy. Opět, je-li zadán název a tomu odpovídá více adres, budou přidána pravidla pro všechny adresy.
`-D` *`třída`*	Vymaže ze třídy jedno nebo více pravidel, která odpovídají následující specifikaci.
`-D` *`třída č_prav`*	Vymaže ze třídy pravidlo na pozici *č_prav*. Pravidla jsou číslována od jedničky.
`-R` *`třída č_prav`*	Nahradí pravidlo na pozici *č_prav* novým pravidlem.
`-C` *`třída`*	Porovná datagram popsaný následující specifikací s pravidly dané třídy. Příkaz vrátí zprávu o tom, jak bude datagram pravidly této třídy zpracován. Tato volba je velmi užitečná k testování konfigurace firewallu a budeme se jí později věnovat podrobněji.
`-L [`*`třída`*`]`	Vypíše pravidla dané třídy, případně všech tříd, pokud není třída zadána.

83 Pozn. překladatele: Zatím jsme žili v představě, že jedné IP adrese může sice odpovídat více názvů (jeden kanonický a libovolný počet aliasů), nicméně že určitý název se vždy převede pouze na jednu jedinou adresu. Jak vidíme, neplatí to vždy. DNS povoluje přiřadit jednomu názvu více IP adres a toto uspořádání se používá k distribuci zátěže na více počítačů.

Například názvu *www.seznam.cz* odpovídá *** IP adres (a tento název je tedy obsluhován *** počítači). Vždy když se bavíte se serverem Seznam, budete náhodně připojeni k jednomu z počítačů, na nichž server běží. Tento mechanismus distribuce zátěže se označuje jako *DNS round-robin*.

`-F [třída]` Vymaže pravidla dané třídy, případně všech tříd, není-li třída zadána.

`-Z [třída]` Vynuluje počitadla datagramů a bajtů pro všechna pravidla dané třídy nebo všech tříd, není-li třída specifikována.

`-N třída` Vytvoří novou třídu se zadaným názvem. Třída zadaného jména nesmí existovat. Tímto příkazem se vytvářejí uživatelem definované třídy.

`-X [třída]` Vymaže třídu definovanou uživatelem, případně všechny uživatelem definované třídy, není-li název třídy uveden. Aby příkaz uspěl, na třídu nesmějí existovat odkazy z žádných jiných tříd.

`-P třída politika` Nastavuje implicitní politiku dané třídy. Platné politiky jsou `ACCEPT`, `DENY`, `REJECT`, `REDIR` a `RETURN`. `ACCEPT`, `DENY` a `REJECT` mají stejný význam jako v klasické implementaci firewallu. Pravidlo `REDIR` říká, že datagram má být transparentně přesměrován na nějaký port firewallu. Hodnota `RETURN` způsobí návrat do třídy, z níž byla volána třída s tímto pravidlem a zpracování pokračuje pravidlem následujícím za volajícím pravidlem.

Specifikace pravidel

Pomocí celé řady parametrů programu **ipchains** je možné vytvářet pravidla udávající, které typy paketů jsou pravidlem zpracovány. Pokud ve specifikaci pravidla není některý z parametrů uveden, předpokládá se jeho standardní hodnota.

`-p [!]protokol` Udává protokol, který pravidlu vyhovuje. Platné názvy protokolů jsou `tcp`, `udp`, `icmp` a `all`. Pokud chcete definovat jiný protokol, můžete uvést jeho číslo. Hodnotou `4` můžete například definovat zapouzdřující protokol `ipip`. Je-li uveden `!`, pravidlo je negováno a budou mu vyhovovat všechny protokoly kromě zadaného. Není-li parametr uveden, předpokládá se hodnota `all`.

`-s [!]adresa[/maska] [[!]port]`

Udává zdrojovou adresu a port datagramu, který pravidlu vyhovuje. Adresa může být zadána názvem počítače, názvem sítě nebo IP adresou. Nepovinný údaj *maska* představuje síťovou masku a může být zadán buď v tradiční podobě (například `/255.255.255.0`), nebo v moderní podobě (například `/24`). Nepovinný údaj *port* definuje TCP nebo UDP port nebo typ ICMP datagramu. Port můžete specifikovat pouze v případě, že jste parametrem `-p` definovali jeden z protokolů `tcp`, `udp` nebo `icmp`. Může být zadán i rozsah portů zadáním spodní a horní meze oddělených dvojtečkou. Například hodnota `20:25` znamená porty 20 (včetně) až 25 (včetně). Opět je možné pomocí `!` pravidlo negovat.

`-d [!]adresa[/maska] [[!]port]`

Udává cílovou adresu a port datagramu, který pravidlu vyhovuje. Parametr se používá stejně jako `-s`.

`-j cíl` Definuje akci, která se má provést, jestliže datagram pravidlu vyhovuje. Tento parametr můžete chápat jako příkaz „běž na“. Platné cílové hodnoty jsou `ACCEPT`, `DENY`, `REJECT`, `REDIR` a `RETURN`. Jejich význam jsme vysvětlili dříve. Kromě toho můžete uvést název uživatelské třídy pravidel a zpracování datagramu bude pokračovat v této třídě. Pokud parametr není uveden, nestane se s datagramem vůbec nic, a dojde pouze k inkrementaci počitadel bajtů a datagramů.

`-i [!]`*`název_rozhraní`*

Definuje rozhraní na němž je datagram přijat nebo odesílán. Symbol ! opět neguje význam pravidla. Pokud název rozhraní končí symbolem +, vyhovují všechna rozhraní začínající zadaným řetězcem. Například - ppp+ definuje všechna rozhraní PPP, -i ! eth+ definuje všechna rozhraní kromě ethernetových rozhraní.

`[!]-f` Udává, že pravidlo platí pro všechno kromě prvního fragmentu fragmentovaného datagramu.

Volby

Následující volby příkazu **ipchains** jsou obecné. Některé z nich slouží k nastavení specifických nuancí programu **ipchains**.

`-b` Příkaz vygeneruje dvě pravidla. Jedno pravidlo bude odpovídat zadaným parametrům, druhé bude odpovídat těmto parametrům v opačném pořadí.

`-v` Zapne „výřečný“ výstup programu **ipchains**. Program bude sdělovat více informací.

`-n` Způsobí, že **ipchains** bude vypisovat IP adresy a porty jako čísla a nebude se pokoušet o jejich převod na názvy.

`-l` Zapne logování datagramů. Každý datagram vyhovující pravidlu bude jádrem zaznamenán prostřednictvím funkce printk(), kterou obvykle obsluhuje program **sysklogd** a zapisuje do souboru. Tímto způsobem můžete zachytit netypické datagramy.

`-o[`*`max_vel`*`]` Datagram vyhovující danému pravidlu se zkopíruje na zařízení „síťové linky“ v uživatelském prostoru. Parametr *max_vel* definuje maximální počet bajtů, které se z každého datagramu kopírují. Tato volba je užitečná zejména pro programátory, v budoucnosti ji však možná využijí i různé programy.

`-m` *`značka`* Datagramy vyhovující pravidlu budou *označeny* hodnotou. Značka je 32bitové číslo bez znaménka. V současných implementacích pro označení není použití, v budoucích verzích však může značka rozhodovat o tom, jak má být datagram zpracován dalšími programy – například směrovacím systémem. Pokud hodnota značky začíná symboly + nebo -, přičte se nebo odečte ke stávající hodnotě značky.

`-t` *`andmaska xormaska`* Umožňuje manipulovat s bity typu služby v hlavičce IP datagramů, které pravidlu vyhovují. Bity typu služby se používají v inteligentních směrovačích ke zvýšení priority datagramů. Směrovací software Linuxu podporuje tyto prioritní operace. Hodnoty *andmaska* a *xormaska* definují bitové masky, kterými budou bity typu služby v datagramu logicky ANDovány a XORovány. Jedná se o pokročilou funkci, která je popsána v dokumentu IPCHAINS-HOWTO.

`-x` Všechna čísla ve výstupu programu **ipchains** budou uvedena přesně, bez zaokrouhlování.

`-y` Pravidlo bude platit pro TCP datagramy s nastaveným bitem SYN a nenastavenými bity ACK a FIN. Slouží k filtrování požadavků na navázání spojení.

Zpět k základnímu příkladu

Opět předpokládejme, že máme nějakou organizaci se sítí a pro přístup k Internetu používáme firewall založený na Linuxu. Našim uživatelům chceme povolit přístup k webovým serverům na Internetu, nechceme však propouštět jakýkoliv jiný provoz.

Síť používá 24bitovou síťovou masku (síť třída C) a adresu 172.16.1.0. Můžeme pak použít následující pravidla:

```
# ipchains -F forward
# ipchains -P forward DENY
# ipchains -A forward -s 0/0 80 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 80 -p tcp -b -j ACCEPT
```

První příkaz vymaže všechna pravidla třídy forward, druhý příkaz nastaví implicitní politiku této třídy na deny. Třetí a čtvrté pravidlo definují požadované filtrování. Čtvrté pravidlo propouští tam a zpět datagramy mezi námi a webovými servery, třetí pravidlo zakazuje navázat spojení z venkovního portu 80.

Pokud budeme chtít přidat pravidla povolující našim uživatelům použití služby FTP v pasivním režimu, bude to vypadat takto:

```
# ipchains -A forward -s 0/0 20 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 20 -p tcp -b -j ACCEPT
# ipchains -A forward -s 0/0 21 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 21 -p tcp -b -j ACCEPT
```

Výpis pravidel programu ipchains

K vypsání pravidel definovaných programem **ipchains** použijeme parametr `-L`. Stejně jako u programu **ipfwadm** můžeme pomocí dalších parametrů definovat podrobnost výpisu. V nejjednodušším případě vytvoří **ipchains** asi takovýto výstup:

```
# ipchains -L -n
Chain input (policy ACCEPT):
Chain forward (policy DENY):
target     prot opt     source               destination          ports
DENY       tcp  -y----  0.0.0.0/0            172.16.1.0/24        80 ->   *
ACCEPT     tcp  ------  172.16.1.0/24        0.0.0.0/0            * ->   80
ACCEPT     tcp  ------  0.0.0.0/0            172.16.1.0/24        80 ->   *
ACCEPT     tcp  ------  172.16.1.0/24        0.0.0.0/0            * ->   20
ACCEPT     tcp  ------  0.0.0.0/0            172.16.1.0/24        20 ->   *
ACCEPT     tcp  ------  172.16.1.0/24        0.0.0.0/0            * ->   21
ACCEPT     tcp  ------  0.0.0.0/0            172.16.1.0/24        21 ->   *

Chain output (policy ACCEPT):
```

Pokud neuvedeme název třídy, vypíše program pravidla ve všech třídách. Parametr -n programu říká, aby se nepokoušel o převod adres a čísel portů na názvy. Z výpisu by mělo být jasné, co nám program říká.

Podrobnější výpis vynucený parametrem -v uvádí větší množství podrobností. Ve výstupu jsou uvedeny hodnoty počitadel datagramů a paketů, masky pro typ služby, názvy rozhraní, značky a maximální velikost.

Každé pravidlo programu **ipchains** obsahuje počitadlo datagramů a počitadlo bajtů. Tímto mechanismem se implementuje IP účtování, o kterém hovoříme v kapitole 10. Implicitně se hodnoty těchto počitadel vypisují v zaokrouhleném tvaru a pomocí symbolů K a M se uvádějí tisíce a miliony[84]. Pokud uvedeme parametr -x, budou hodnoty počitadel vypsány v úplném tvaru bez zaokrouhlení.

Použití tříd

Teď už víme, jak pomocí programu **ipchains** nahradit program **ipfwadm** s tím, že získáváme poněkud jednodušší syntaxi a některé funkce navíc. Je načase seznámit se s tím, kdy a proč používat uživatelem definované třídy pravidel. Kromě toho si povíme něco o skriptech, které se s balíkem **ipchains** distribuují.

Uživatelem definované třídy

Tři třídy pravidel v klasickém firewallu představovaly mechanismus pro vytváření konfigurací, které byly snadno pochopitelné a udržovatelné na menších sítích s jednoduchými požadavky na firewall. Jakmile potřebujeme složitější konfiguraci, vynoří se celá řada problémů. Větší sítě typicky vyžadují použití většího množství pravidel než jsme doposud viděli, zároveň narůstají nároky na komplikovanější pravidla, která pokrývají různé specifické situace. S rostoucím počtem pravidel klesá výkon firewallu, protože každý datagram je nutné testovat proti velkému množství podmínek, navíc se komplikuje správa firewallu. Není totiž možné atomicky zapnout nebo vypnout celou skupinu pravidel a v době změny konfigurace firewallu tak síť může být vystavena útokům.

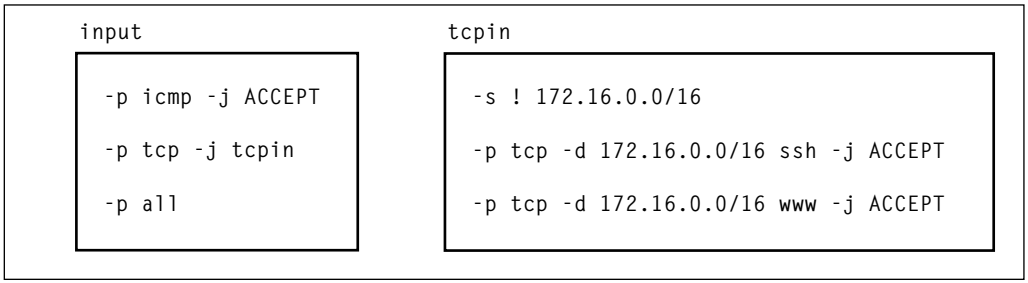
Návrh systému Firewall Chains řeší oba tyto problémy, protože umožňuje správci firewallu vytvářet libovolný počet tříd pravidel, která se pak propojují do tří vestavěných tříd. Pomocí parametru `-N` programu **ipchains** můžeme vytvořit novou třídu pravidel, jejíž název bude dlouhý maximálně osm znaků (je rozumné v názvech tříd používat pouze malá písmena). Pomocí parametru `-j` definujeme akci, která se má provést, když datagram pravidlu vyhovuje. Tímto parametrem můžeme definovat, že pokud datagram vyhovuje zadanému pravidlu, může být podroben dalšímu testování proti pravidlům uživatelské třídy. Budeme to ilustrovat na příkladu.

Podívejme se na následující příkazy **ipchains**:

```
ipchains -P input DENY
ipchains -N tcpin
ipchains -A tcpin -s ! 172.16.0.0/16
ipchains -A tcpin -p tcp -d 172.16.0.0/16 ssh -j ACCEPT
ipchains -A tcpin -p tcp -d 172.16.0.0/16 www -j ACCEPT
ipchains -A input -p tcp -j tcpin
ipchains -A input -p all
```

Implicitní politiku třídy input jsme nastavili na deny. Druhým příkazem vytváříme uživatelskou třídu nazvanou tcpin. Třetí příkaz přidává do třídy tcpin pravidlo, kterému vyhovuje jakýkoliv datagram, jenž byl zvenku předán naší síti tomuto pravidlu není přidělena žádná akce. Toto pravidlo je zavedeno pro potřeby účtování a bude podrobněji vysvětleno v kapitole 10. Dalším dvěma pravidlům vyhovují datagramy určené naší lokální síti z portů ssh nebo WWW, datagramy vyhovující těmto pravidlům budou přijaty. Další pravidlo je to místo, kde začínají kouzla s třídami. Způsobí, že firewall bude všechny datagramy protokolu TCP testovat proti uživatelské třídě tcpip. Konečně jsme do třídy input přidali další pravidlo, kterému budou vyhovovat všechny datagramy. Jedná se o další účtovací pravidlo. Skupiny takto vzniklých pravidel znázorňuje obrázek 9.4.

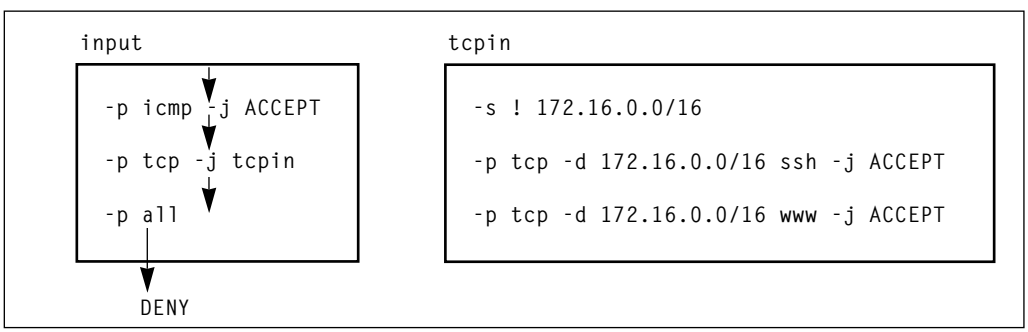
84 Pozn. překladatele: K je samozřejmě 1 024 a M samozřejmě 1 048 576.



Obrázek 9.4 – Jednoduché skupiny pravidel

Třídy input a tcpin obsahují námi vestavěná pravidla. Zpracování datagramu začíná vždy v některé z vestavěných tříd. Ukážeme si, jakým způsobem třídy fungují na příkladech zpracování různých typů datagramů.

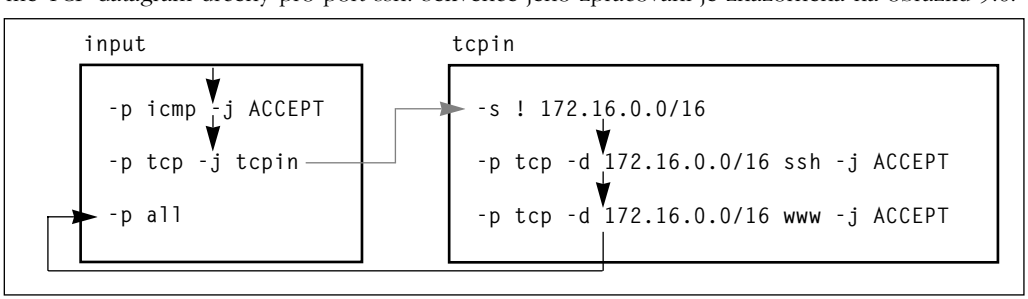
Podívejme se nejprve co se bude dít, když bude přijat UDP datagram určený pro některého našeho hostitele. Jeho zpracování jednotlivými pravidly demonstruje obrázek 9.5.



Obrázek 9.5 – Sekvence testovaných pravidel při zpracování UDP datagramu

Datagram je přijat třídou input a projde oběma prvními pravidly, protože ta se vztahují k protokolům ICMP a TCP. Vyhovuje třetímu pravidlu třídy, to však nespecifikuje žádnou akci, takže sice dojde k inkrementaci počitadel bajtů a datagramů, nic dalšího se ale nestane. Datagram dojde na konec třídy input, kde narazí na její implicitní pravidlo a je zrušen.

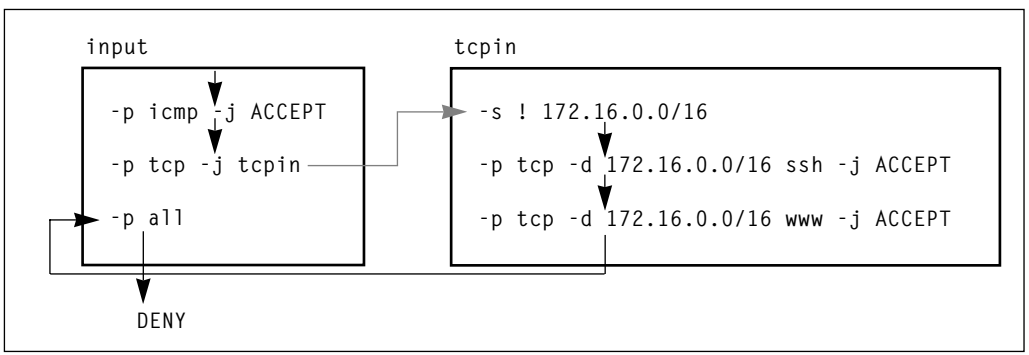
Abychom viděli chování uživatelem definované třídy, podívejme se nyní co se stane, když přijmeme TCP datagram určený pro port ssh. Sekvence jeho zpracování je znázorněna na obrázku 9.6.



Obrázek 9.6 – Sekvence zpracování TCP datagramu pro port ssh

V tomto případě datagram vyhovuje druhému pravidlu třídy input, které specifikuje cíl tcpin. Je-li jako cíl datagramu specifikována uživatelem definovaná třída, dojde k tomu, že datagram bude testován pravidly této třídy, takže zpracování pokračuje prvním pravidlem třídy tcpin. Prvnímu pravidlu vyhovuje jakýkoliv datagram pocházející z vnější sítě a pravidlo nespecifikuje žádnou akci. Jedná se opět o účtovací pravidlo a zpracování datagramu pokračuje druhým pravidlem ve třídě. Tomuto pravidlu datagram vyhovuje a pravidlo specifikuje akci ACCEPT. K dalšímu zpracování datagramu nedochází a datagram je přijat.

Nakonec se podívejme na to, co se stane, když datagram dojde na konec uživatelem definované třídy. Ukážeme si to na zpracování TCP datagramu určeného pro jiný než jeden ze dvou povolených portů. Zpracování takového datagramu demonstruje obrázek 9.7.



Obrázek 9.7 – Sekvence zpracování TCP datagramu pro port telnet

Uživatelem definované třídy nemají implicitní politiku. Jakmile jsou otestována všechna pravidla a datagram žádnému nevyhovuje, chová se třída tak, jako kdyby obsahovala implicitní pravidlo RETURN. Pokud vám takové chování nevyhovuje, musíte na konci třídy definovat pravidlo, kterému budou vyhovovat všechny datagramy a jež provede vámi požadovanou implicitní akci. V našem příkladu dojde při zpracování datagramu třídou tcpin k pokračování zpracování v nadřazené třídě pravidlem následujícím za tím, kterým byla uživatelská třída volána. Datagram nakonec dorazí na konec třídy input, která má definovánu implicitní politiku a datagram bude zrušen.

Tento příklad byl velmi jednoduchý, demonstroval však smysl použití tříd. Praktické použití tříd bude mnohem složitější. Poněkud komplikovanější příklad je vytvořen následující sekvencí příkazů.

```
#
# Implicitní předávací politika REJECT
ipchains -P forward REJECT
#
# vytvoření uživatelských tříd
ipchains -N sshin
ipchains -N sshout
ipchains -N wwwin
ipchains -N wwwout
#
# Odmítáme spojení špatným směrem
ipchains -A wwwin -p tcp -s 172.16.0.0/16 -y -j REJECT
ipchains -A wwwout -p tcp -d 172.16.0.0/16 -y -j REJECT
ipchains -A sshin -p tcp -s 172.16.0.0/16 -y -j REJECT
ipchains -A sshout -p tcp -d 172.16.0.0/16 -y -j REJECT
```

```
#
# Odmítáme vše na konci uživatelské třídy
ipchains -A sshin -j REJECT
ipchains -A sshout -j REJECT
ipchains -A wwwin -j REJECT
ipchains -A wwwout -j REJECT
#
# předáváme služby www a ssh odpovídajícím třídám
ipchains -A forward -p tcp -d 172.16.0.0/16 ssh -b -j sshin
ipchains -A forward -p tcp -s 172.16.0.0/16 -d 0/0 ssh -b -j sshout
ipchains -A forward -p tcp -d 172.16.0.0/16 www -b -j wwwin
ipchains -A forward -p tcp -s 172.16.0.0/16 -d 0/0 www -b -j wwwout
#
# Pravidla povolující hostitele vkládáme na 2. pozici našich tříd
ipchains -I wwwin 2 -d 172.16.1.2 -b -j ACCEPT
ipchains -I wwwout 2 -s 172.16.1.0/24 -b -j ACCEPT
ipchains -I sshin 2 -d 172.16.1.4 -b -j ACCEPT
ipchains -I sshout 2 -s 172.16.1.4 -b -j ACCEPT
ipchains -I sshout 2 -s 172.16.1.6 -b -j ACCEPT
#
```

V tomto příkladu jsme pomocí uživatelsky definovaných tříd zjednodušili správu konfigurace firewallu a zároveň zlepšili efektivitu zpracování datagramů v porovnání s využitím pouze vestavěných tříd.

V příkladu vytváříme uživatelské třídy pro oba směry toku datagramů služeb WWW a ssh. Do třídy wwwout umístíme pravidla definující počítače, které mají povoleno vytvářet odchozí připojení na webové servery, ve třídě sshin definujeme počítače, kterým povolujeme připojit se k nám službou **ssh**. Předpokládali jsme, že potřebujeme individuálně nastavovat, které počítače v naší síti mohou vytvářet a přijímat WWW a ssh spojení. Zjednodušení je zřejmé, protože uživatelem definovaná třída nám umožňuje sdružovat logicky spolu související pravidla a nemusíme mít všechna pravidla smíchaná dohromady. Dochází i ke zlepšení výkonu firewallu, protože pro jednotlivé datagramy se v průměru zkrátil počet pravidel, proti kterým jsou testovány. Účinnost tohoto mechanismu se zvýší s rostoucím počtem pravidel. Pokud bychom nepoužili uživatelské třídy, v nejhorším případě bude přijatý datagram testován proti všem pravidlům v seznamu. Budeme-li předpokládat, že každému pravidlu vyhovuje stejný počet datagramů z balíku všech přijímaných, i pak v průměru porovnáváme každý datagram proti polovině pravidel. Pomocí uživatelských tříd se vyhneme testování proti skupinám podrobných pravidel jednoduše tím, že datagram nevyhovuje jednoduchému pravidlu v základní třídě, které by jej do podřízené třídy „poslalo“.

Podpůrné skripty programu ipchains

Softwarový balík **ipchains** je dodáván se třemi podpůrnými skripty. O prvním z nich jsme už v krátkosti hovořili, další dva slouží k jednoduchému uložení a obnovení konfigurace firewallu.

Skript **ipfwadm-wrapper** emuluje řádkovou syntaxi programu **ipfwadm**, pravidla firewallu však vytváří voláním programu **ipchains**. Tento skript představuje možnost, jak firewall snadno nakonfigurovat pomocí původních pravidel a mohou jej využít ti, kteří se nechtějí učit novou syntaxi programu **ipchains**. Skript **ipfwadm-wrapper** se ve dvou věcech chová odlišně od programu **ipfwadm**: Příkaz **ipchains** nepodporuje definici rozhraní jeho IP adresou, proto skript **ipfwadm-wrapper** sice akceptuje parametr `-v`, pokouší se jej ale převést na parametr `-w` programu **ipchains** tím, že hledá název rozhraní, kterému je daná adresa přiřazena. Vždy když použijete pa-

rametr `-v`, skript **ipfwadm-wrapper** vás na tuto skutečnost upozorní. Druhým rozdílem je to, že nejsou korektně převedena pravidla pro účtování fragmentů.

Skripty **ipchains-save** a **ipchains-restore** výrazně usnadňují vytváření a úpravy konfigurace firewallu. Příkaz **ipchains-save** přečte stávající konfiguraci firewallu a ve zjednodušeném tvaru ji zapíše na standardní výstup. Příkaz **ipchains-restore** pak čte data v tomto formátu a danými pravidly nastaví firewall. Výhoda použití těchto skriptů proti přímé modifikaci konfiguračního skriptu firewallu a jeho testování spočívá v tom, že konfiguraci můžete jednou dynamicky vytvořit a pak ji uložit. Kdykoliv v budoucnu ji můžete obnovit, změnit a případně znovu uložit.

Skripty se používají asi takovýmto způsobem: Příkaz

```
ipchains-save >/var/state/ipchains/firewall.state
```

uloží stávající konfiguraci firewallu. K jejímu obnovení typicky v době startu systému zadáte:

```
ipchains-restore </var/state/ipchains/firewall.state
```

Skript **ipchains-restore** testuje, zda už neexistuje některá z uživatelských tříd definovaných ve vstupních datech. Spustíte-li skript s parametrem -f, nejprve smaže pravidla již existující třídy a pak ji naplní pravidly podle vstupních dat. Bez uvedení tohoto parametru se vás skript zeptá, zda má konfiguraci dané třídy přeskočit, nebo zda má původní pravidla smazat a nahradit je novými.

Netfilter a IP Tables (jádra 2.4)

Při vývoji programu Firewall Chains došel Paul Russel k závěru, že firewally by měly být méně komplikované, takže se soustředil na zjednodušení způsobů, jimiž jsou datagramy ve firewallovém kódu jádra zpracovávány a navrhl nový mechanismus filtrování, který je jednak jednodušší a jednak podstatně pružnější. Tento mechanismus nazval *netfilter*.

V době vzniku tohoto textu nebyl návrh mechanismu *netfilter* ještě stabilizován. Doufáme, že omluvíte jakékoliv nesrovnalosti v popisu tohoto mechanismu a jeho konfiguračních nástrojů, ke kterým dojde v důsledku změn provedených v programu po vzniku tohoto textu. Považujeme *netfilter* za natolik významnou techniku, že jej zde chceme stručně popsat, i když jsme si vědomi, že některé z našich popisů budou spekulativní. Pokud narazíte na nějaké nesrovnalosti, přesný a aktuální popis všech vlastnostní systému *netfilter* by měl být k nalezení v příslušném dokumentu HOWTO.

Co je špatné na technice IP Chains? Významně zjednodušila efektivitu a správu firewallových pravidel. Nicméně mechanismus zpracování datagramů je stále příliš složitý zejména ve spojení s dalšími technologiemi souvisejícími s firewally, například s technologií IP maškarády (viz kapitola 11) nebo s jinými technikami překladu adres. Jedním z důvodů těchto komplikací je skutečnost, že IP maškaráda a technologie NAT (Network Address Translation) byly vyvinuty nezávisle na firewallu a až později byly do jeho kódu integrovány. Výhodnější řešení je tyto techniky od počátku zahrnout do celkového návrhu firewallu. Pokud chce někdo v současném uspořádání zasáhnout do mechanismu zpracování datagramů v jádru, obtížně se hledá místo, kam vlastní kód zařadit a budou nutné přímé modifikace kódu jádra.

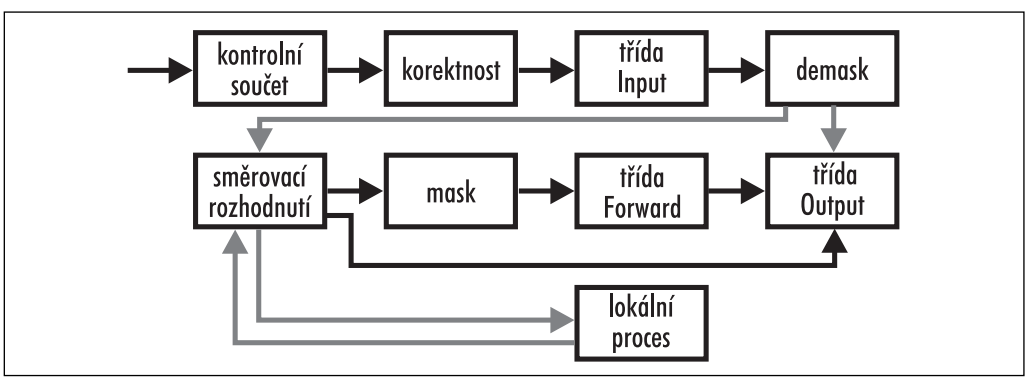
Kromě toho existují i další problémy. Konkrétně třída „input“ popisuje vstup síťové IP vrstvy jako celku. Tato třída zahrnuje jak datagramy, které jsou určeny danému počítači, tak i datagramy, které jím mají být směrovány. Je to trošku proti zdravému rozumu, protože se tím částečně překrývají funkce třídy input a forward, která zpracovává pouze směrované datagramy, nicméně nastupuje vždy až po třídě input. Pokud chcete jiným způsobem ošetřovat datagramy určené lokálními

hostiteli a jinak datagramy jím směrované, musíte vytvářet složitá pravidla, která budou zpracovávat jeden nebo druhý typ datagramů. Stejný problém se týká i třídy output.

Tyto komplikace se nevyhnutelně přenesly i na práci správce systému, protože se projevují v tom, jak je třeba vytvářet pravidla pro firewall. Navíc jakékoliv rozšíření filtrovacích funkcí vyžaduje přímé zásahy do jádra, protože v něm jsou filtrační politiky implementovány a neexistuje k nim žádné transparentní rozhraní. Systém *netfilter* řeší jak složitost, tak neobratnost předchozích řešení tím, že v jádře implementuje obecnou platformu, která zjednodušuje proces zpracování datagramů a zároveň umožňuje modifikovat filtrační politiky bez nutnosti modifikovat jádro.

Podívejme se nyní na dvě klíčové změny. Obrázek 9.8 ilustruje, jakým způsobem jsou datagramy zpracovávány mechanismem IP Chains, na obrázku 9.9 vidíme, jak je zpracovává implementace *netfilter*. Hlavní rozdíly spočívají v odstranění maškarády z hlavního kódu a ve změně umístění tříd input a output. Zároveň s těmito změnami byl vyvinut nový konfigurační nástroj, nazvaný **iptables**.

V implementaci IP Chains se třída input vztahuje na všechny datagramy přijaté počítačem, bez ohledu na to, zda jsou určeny pro něj, nebo zda je má pouze směrovat. V implementaci netfilter se třída input vztahuje pouze na datagramy určené lokálnímu hostiteli, třída forward pak pouze na datagramy určené jinému hostiteli. Analogicky se v implementaci IP Chains vztahuje třída output na všechny odesílané datagramy bez ohledu na to, zda je odesílá lokální systém, nebo zda je směruje jinému hostiteli. V implementaci netfilter se třída output vztahuje pouze na datagramy generované lokálním systémem a netýká se datagramů směrovaných jinému hostiteli. Jen samotná tato změna přináší výrazné zjednodušení řady firewallových konfigurací.

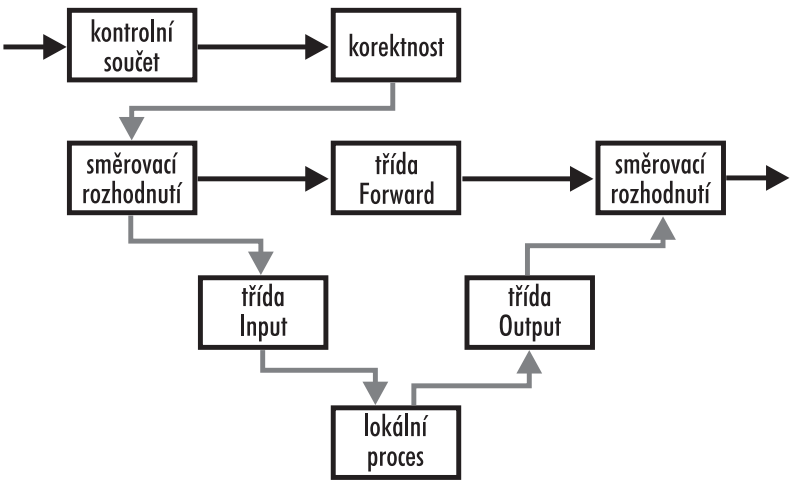


Obrázek 9.8 – Zpracování datagramů v implementaci IP Chains

Komponenty „demask“ a „mask“ na obrázku 9.8 jsou samostatné komponenty jádra, které odpovídají za zpracování příchozích a odchozích datagramů maškarády. V modulech implementace netfilter byly navrženy znovu.

Představme si případ, kdy jsou implicitní politiky tříd input, output i forward nastaveny na deny. V implementaci IP Chains pak budeme potřebovat šest pravidel, aby mohl nějaký typ datagramu projít firewallem: vždy dvě do tříd input, output a forward (jedno z pravidel bude pokrývat odchozí směr, druhé pak příchozí směr). Jistě si dokážete představit, jak se to může komplikovat, když budeme chtít rozlišovat služby, které mohou být směrovány a služby, které se mohou připojit k lokálnímu hostiteli, směrovat se však nebudou. Implementace IP Chains umožňuje tyto konfigurace poněkud zjednodušit pomocí uživatelských tříd, nicméně návrh firewallu stále nebude zřejmý a bude vyžadovat značnou míru zkušeností.

Implementace netfilter s programem **iptables** úplně odstraňuje tyto komplikace. Propouštíme-li nějakou službu přes firewall, nicméně jí nedovolujeme dosáhnout lokálního počítače, budou nám stačit dvě pravidla: obě ve třídě forward, jedno pro směr „tam“ a druhé pro směr „zpět“. Jedná se o intuitivní řešení firewallu a návrh konfigurace firewallu se tak výrazně usnadní.



Obrázek 9.9 – Zpracování datagramu v implementaci netfilter

Podrobný popis provedených změn obsahuje dokument PACKET-FILTERING-HOWTO, takže my se zde zaměříme na praktičtější aspekty.

Zpětná kompatibilita s ipfwadm a s ipchains

Významnou vlastností implementace *netfilter* je to, že dokáže emulovat rozhraní programů **ipfwadm** a **ipchains**. Tato emulace tak poněkud usnadňuje přechod na novou generaci firewallu.

Dva moduly jádra, `ipfwadm.o` a `ipchains.o` zajišťují zpětnou kompatibilitu s programy **ipfwadm** a **ipchains**. Můžete mít nahrán pouze jeden z těchto modulů, a to jen tehdy, není-li nahrán modul `ip_tables.o`. Po nahrání příslušného modulu se *netfilter* chová úplně stejně jako předchozí implementace firewallu.

Emulace rozhraní programu **ipchains** se zapne následujícími příkazy:

```
rmmod ip_tables
modprobe ipchains
ipchains ...
```

Použití programu iptables

Program **iptables** slouží k nastavení filtračních pravidel mechanismu *netfilter*. Jeho syntaxe je přímo odvozena od programu **ipchains**, liší se však v jednom podstatném rysu: je *rozšiřitelná*. Znamená to, že funkce programu je možné doplnit bez nutnosti jeho nového přeložení. Tento trik je realizován pomocí sdílených knihoven. Pro program existuje několik standardních rozšíření, o kterých budeme za chvíli mluvit.

Než budete moci program **iptables** použít, musíte nahrát modul jádra *netfilter*, který zajišťuje podporu tohoto programu. Nejjednodušší způsob spočívá v následujícím volání programu **modprobe**:

```
modprobe ip_tables
```

Příkaz `iptables` slouží jak k nastavení filtrace, tak i k nastavení překladu adres. Zajišťují to dvě tabulky pravidel, nazvané *filter* a *nat*. Tabulka *filter* se nahrává automaticky, pokud neuvedete parametr `-t`. Kromě toho systém obsahuje pět vestavěných tříd. Třídy `INPUT` a `FORWARD` se týkají tabulky *filter*, třídy `PREROUTING a POSTROUTING` tabulky *nat* a třída `OUTPUT` obou tabulek. V této kapitole budeme hovořit pouze o tabulce *filter*. Tabulku *nat* popisujeme v kapitole 11.

Obecná syntaxe většiny příkazů **iptables** je:

```
iptables příkaz specifikace_pravidel rozšíření
```

Nejprve se seznámíme s některými parametry a pak si ukážeme příklady.

Příkazy

Existuje řada způsobů, jak v programu **iptables** manipulovat s pravidly a třídami pravidel. K nastavení firewallů se vztahují následující příkazy:

`-A` *třída*	Přidává na konec třídy jedno nebo více pravidel. Pokud je jako zdroj nebo cíl uveden název počítače a ten má přiřazeno více IP adres, budou přidána pravidla pro všechny adresy.
`-I` *třída č_prav*	Přidá jedno nebo více pravidel na začátek třídy. Opět je-li zadán název a tomu odpovídá více adres, budou přidána pravidla pro všechny adresy.
`-D` *třída*	Vymaže ze třídy jedno nebo více pravidel, která odpovídají následující specifikaci.
`-D` *třída č_prav*	Vymaže ze třídy pravidlo na pozici *č_prav*. Pravidla jsou číslována od jedničky.
`-R` *třída č_prav*	Nahradí pravidlo na pozici *č_prav* novým pravidlem.
`-C` *třída*	Porovná datagram popsaný následující specifikací s pravidly dané třídy. Příkaz vrátí zprávu o tom, jak bude datagram pravidly této třídy zpracován. Tato volba je velmi užitečná k testování konfigurace firewallu a budeme se jí později věnovat podrobněji.
`-L [`*třída*`]`	Vypíše pravidla dané třídy, případně všech tříd, pokud není třída zadána.
`-F [`*třída*`]`	Vymaže pravidla dané třídy, případně všech tříd, není-li třída zadána.
`-Z [`*třída*`]`	Vynuluje počitadla datagramů a bajtů pro všechna pravidla dané třídy nebo všech tříd, není-li třída specifikována.
`-N` *třída*	Vytvoří novou třídu se zadaným názvem. Třída zadaného jména nesmí existovat. Tímto příkazem se vytvářejí uživatelem definované třídy.
`-X [`*třída*`]`	Vymaže třídu definovanou uživatelem, případně všechny uživatelem definované třídy, není-li název třídy uveden. Aby příkaz uspěl, na třídu nesmějí existovat odkazy z žádných jiných tříd.

`-P` *třída politika* Nastavuje implicitní politiku dané třídy. Platné politiky jsou `ACCEPT`, `DROP`, `QUEUE` a `RETURN`. `ACCEPT` povolí průchod datagramu. `DROP` způsobí zrušení datagramu. `QUEUE` způsobí předání datagramu do uživatelského prostoru k dalšímu zpracování. `RETURN` způsobí návrat do nadřízené třídy pravidel.

Specifikace pravidel

Program **iptables** má celou řadu parametrů sloužících k definici pravidel. Kdykoliv se zadává pravidlo, používají se všechny následující parametry. Pokud některý z parametrů není zadán, použije se jeho implicitní hodnota.

`-p [!]`*protokol* Udává protokol, který pravidlu vyhovuje. Platné názvy protokolů jsou `tcp`, `udp`, `icmp` nebo číselná hodnota protokolu, pokud ji znáte[85]. Hodnotou `4` můžete například definovat zapouzdřující protokol `ipip`. Je-li uveden `!`, pravidlo je negováno a budou mu vyhovovat všechny protokoly kromě zadaného. Není-li parametr uveden, budou pravidlu vyhovovat všechny protokoly.

`-s [!]`*adresa*`[/`*maska*`]` Udává zdrojovou adresu datagramu, který pravidlu vyhovuje. Adresa může být zadána názvem počítače, názvem sítě nebo IP adresou. Nepovinný údaj *maska* představuje síťovou masku a může být zadán buď v tradiční podobě (například `/255.255.255.0`) nebo v moderní podobě (například `/24`).

`-d [!]`*adresa*`[/`*maska*`]` Udává cílovou adresu datagramu, který pravidlu vyhovuje. Parametr se používá stejně jako `-s`.

`-j` *cíl* Definuje akci, která se má provést, jestliže datagram pravidlu vyhovuje. Tento parametr můžete chápat jako příkaz „běž na“. Platné cílové hodnoty jsou `ACCEPT`, `DROP`, `QUEUE` a `RETURN`. Jejich význam jsme vysvětlili dříve. Kromě toho můžete uvést název uživatelské třídy pravidel a zpracování datagramu bude pokračovat v této třídě. Pokud parametr není uveden, nestane se s datagramem vůbec nic, a dojde pouze k inkrementaci počitadel bajtů a datagramů.

`-i [!]`*název_rozhraní* Definuje rozhraní, na němž je datagram přijat. Symbol `!` opět neguje význam pravidla. Pokud název rozhraní končí symbolem +, vyhovují všechna rozhraní začínající zadaným řetězcem. Například `-i ppp+` definuje všechna rozhraní PPP, `-i ! eth+` definuje všechna rozhraní kromě ethernetových rozhraní.

`-o [!]`*název_rozhraní* Definuje rozhraní, na němž je datagram odesílán. Použití je stejné jako u parametru `-i`.

`[!]-f` Udává, že pravidlo platí pro všechno kromě prvního fragmentu fragmentovaného datagramu.

Volby

Následující volby příkazu **iptables** jsou obecné. Některé z nich slouží k nastavení specifických nuancí systému *netfilter*.

`-v` Zapne „výřečný“ výstup programu **iptables**. Program bude sdělovat více informací.

85 Názvy a čísla protokolů naleznete v souboru `/etc/protocols`.

`-n` Způsobí, že **iptables** bude vypisovat IP adresy a porty jako čísla a nebude se pokoušet o jejich převod na názvy.

`-x` Všechna čísla ve výstupu programu **iptaibles** budou uvedena přesně, bez zaokrouhlování.

`--line-numbers` Při výpisu pravidel jednotlivých tříd budou řádky číslovány. Čísla řádků odpovídají pořadí pravidla ve třídě.

Rozšíření

Už jsme se zmínili, že nástroj **iptables** je rozšiřitelný pomocí modulů sdílených knihoven. Existuje několik standardních rozšíření, která implementují funkce programu **ipchains**. Abyste mohli rozšíření použít, musíte jeho název sdělit programu **iptables** parametrem `-m` *název*. Následující seznam obsahuje parametry `-m` a `-p`, jež nastavují kontext rozšíření, společně s parametry, které rozšíření poskytuje.

Rozšíření TCP: `-m tcp -p tcp`

`-sport [!] [port[:port]]`

Definuje zdrojový port datagramu, z nějž musí datagram pocházet, aby pravidlu vyhovoval. Je možné specifikovat i rozsahy portů zadáním spodní a horní meze oddělených dvojtečkou. Například hodnota `20:25` znamená porty od 20 (včetně) po 25 (včetně). Znakem ! je možné význam pravidla negovat.

`-dport [!] [port[:port]]`

Definuje cílový port datagramu, na nějž musí být datagram určen, aby pravidlu vyhovoval. Použití parametru je stejné jako u –sport.

`-tcp-flags [!] maska comp`

Pravidlu budou vyhovovat ty TCP datagramy, jejichž příznaky odpovídají podmínkám specifikovaným parametry maska a comp. maska je čárkami oddělený seznam příznaků, které mají být do testování zahrnuty, comp je čárkami oddělený seznam testovaných příznaků, jež musí být nastaveny. Platnými příznaky jsou *SYN*, *ACK*, *FIN*, *RST*, *URG*, *PSH*, *ALL* a *NONE*. Jedná se o složitější nastavení, popis významu jednotlivých příznaků najdete v nějakém kvalitním popisu protokolu TCP, například v dokumentu RFC 793. Symbolem ! je možné význam pravidla negovat.

`[!] -syn` Udává, že pravidlu budou vyhovovat pouze datagramy s nastaveným příznakem *SYN* a nenastavenými příznaky *ACK* a *FIN*. Tyto datagramy slouží k navázání TCP spojení a pravidlo tak slouží k ošetření otevírání spojení. Tento příznak je zkratkou zápisu: -tcp-flags SYN,RST,ACK SYN Při použití symbolu negace budou pravidlu vyhovovat datagramy s nenastavenými příznaky *SYN* a *ACK*.

Rozšíření UDP: `-m udp -p udp`

`-sport [!] [port[:port]]`

Definuje zdrojový port datagramu, z nějž musí datagram pocházet, aby pravidlu vyhovoval. Je možné specifikovat i rozsahy portů zadáním spodní a horní meze oddělených dvojtečkou. Například hodnota 20:25

znamená porty od 20 (včetně) po 25 (včetně). Znakem ! je možné význam pravidla negovat.

`-dport [!] [port[:port]]`

Definuje cílový port datagramu, na nějž musí být datagram určen, aby pravidlu vyhovoval. Použití parametru je stejné jako u –sport.

Rozšíření ICMP: `-m icmp -p icmp`

`-icmp-type [!] typ` Udává typ ICMP zprávy, která bude pravidlu vyhovovat. Typ je možné zadat číslem nebo názvem. Některé platné názvy jsou: `echo-request`, `echo-reply`, `source-quench`, `time-exceeded`, `destination-unreachable`, `network-unreachable`, `host-unreachable`, `protocol-unreachable` a `port-unreachable`.

Rozšíření MAC: `-m mac`

`-mac-source [!] adresa`

Udává ethernetovou adresu hostitele, který datagram vyslal. Tento parametr má význam pouze ve třídách input a forward, protože u všech datagramů ve třídě output jsme odesilatelem my.

Znovu přepracovaný jednoduchý příklad

Pokud budeme chtít náš známý příklad implementovat pomocí *netfilter*, stačí nahrát modul `ipchains.o` a tvářit se, že máme nainstalován program **ipchains**. Nicméně si ukážeme jeho přímou implementaci programem **iptables**.

Znovu předpokládáme, že máme síť a chceme uživatelům povolit WWW přístup na Internet a nechceme povolit jakýkoliv jiný provoz.

Máme 24bitovou síťovou masku (třída C) a síťovou adresu 172.16.1.0. Programem **iptables** zavedeme následující pravidla:

```
# modprobe ip_tables
# iptables -F FORWARD
# iptables -P FORWARD DROP
# iptables -A FORWARD -m tcp -p tcp -s 0/0 --sport 80 -d 172.16.1.0/24 /
    --syn -j DROP
# iptables -A FORWARD -m tcp -p tcp -s 172.16.1.0/24 --sport /
    80 -d 0/0 -j ACCEPT
# iptables -A FORWARD -m tcp -p tcp -d 172.16.1.0/24 --dport 80 -s 0/0 -j /
    ACCEPT
```

V tomto příkladu budou příkazy **iptables** interpretovány úplně stejně jako příkazy **ipchains**. Musíme mít nahrán modul ip_tables.o. Všimněte si, že program **iptables** nepodporuje parametr -b, takže musíme zadávat samostatná pravidla pro oba směry.

Manipulace s bity typu služby

Bity typu služby (TOS) jsou čtyři bitové příznaky v hlavičce IP datagramu. Pokud je některý z těchto bitů nastaven, mohou směrovače zpracovávat datagramy odlišně od zpracování datagramů bez těchto příznaků. Každý z bitů má svůj vlastní význam a v jednom datagramu může být nastaven

pouze jeden z nich, kombinace příznaků nejsou povoleny. Bity se označují jako bity typu služby, protože umožňují, aby odesílající aplikace síti sdělila typ služby, kterou vyžaduje.

Existují následující třídy síťových služeb:

Minimum delay

Používá se, pokud nejdůležitějším kritériem je čas přenosu datagramu od zdroje k cíli, tedy pokud se požaduje co nejmenší zpoždění. Poskytovatel připojení může současně používat řekněme satelitní a optickou linku. Data přenášená satelitní linkou procházejí obecně delší trasou a jejich zpoždění je větší než pozemní spojení mezi dvěma stejnými místy. Směrovače mohou tedy zajistit, že datagramy s tímto příznakem budou přenášeny optickou linkou.

Maximum throughput

Používá se, pokud je důležitý objem dat přenášený v jednotlivých časových intervalech. Existuje celá řada síťových aplikací, pro něž není kritické zpoždění přenosu, je však pro ně důležitá dostatečná propustnost sítě – například přenosy online audia nebo videa. Poskytovatel pak bude tato data směrovat přes vysoce propustnou linku, byť s větším zpožděním, například přes satelitní linku.

Maximum reliability

Používá se, pokud je důležité, aby data v mezích možností dorazila k cíli bez nutnosti jejich opakovaného odesílání. Protokol IP může být přenášen přes řadu přenosových médií. Protokoly SLIP a PPP jsou například typickými protokoly datové linky, nejsou nicméně tak spolehlivé jako přenos přes nějakou síť, například síť X.25. Poskytovatel může mít k dispozici záložní spoj s vysokou spolehlivostí a při uvedení tohoto příznaku bude data posílat tímto spojem.

Minimum cost

Používá se, je-li nutné minimalizovat přenosové náklady. Pronájem pásma na satelitní transoceánské lince je obecně levnější než pronájem pásma na optickém kabelu se stejným cílem a pokud poskytovatel používá oba spoje, může přenášená data zpoplatňovat různě podle toho, který spoj byl použit. V našem případě může nastavení tohoto bitu způsobit směrování datagramu přes levnější satelitní linku.

Nastavení TOS bitů pomocí ipfwadm a ipchains

Programy **ipfwadm** a **ipchains** pracují s TOS bity prakticky stejně. V obou můžete nastavit pravidla, kterým budou vyhovovat datagramy s určitými příznaky. Pomocí parametru `-t` pak můžete příznaky měnit.

Změny se definují dvěma bitovými maskami. První maska se logicky ANDuje s příznaky v datagramu, druhá se s nimi logicky XORuje. Pokud vám to připadá komplikované, hned si ukážeme, jak jednotlivé příznaky nastavovat.

Bitové masky se definují osmibitovou šestnáctkovou hodnotou. Jak **ipfwadm**, tak **ipchains** používají stejnou syntaxi:

```
-t andmaska xormaska
```

Pokud chcete určitý typ služby nastavit, můžete naštěstí vždy použít stejné masky, aniž by bylo nutné nad tím příliš přemýšlet. Doporučené použití a odpovídající masky jsou uvedeny v tabulce 9.3.

Tabulka 9.3 – Doporučené nastavení TOS bitů

TOS	maska AND	maska XOR	doporučené použití
Minimum Delay	0x01	0x10	ftp, telnet, ssh
Maximum Throughput	0x01	0x08	ftp-data, www
Maximum Reliability	0x01	0x04	snmp, dns
Minimum Cost	0x01	0x02	nntp, smtp

Nastavení TOS bitů pomocí iptables

Nástroj **iptables** umožňuje definovat pravidla, kterým budou vyhovovat pouze datagramy s určitým nastavením TOS bitů pomocí parametru `-m tos` a nastavovat TOS bity datagramům vyhovujícím danému pravidlu pomocí parametru `-j akce`. TOS bity je možné nastavovat pouze ve třídách output a forward. Testování a nastavování probíhá nezávisle. Můžete vytvářet libovolné skupiny pravidel. Můžete například vytvořit pravidlo, které zruší všechny datagramy s určitým nastavením TOS, nebo můžete vytvořit pravidlo, které nastaví TOS u datagramů pocházejících od určitého počítače. Ve většině případů budete používat pravidla, která kombinují jak testování, tak nastavení TOS bitů stejně, jako to bylo u programů **ipfwadm** a **ipchains**.

Namísto složité dvoumaskové konfigurace programů **ipfwadm** a **ipchains** používá program **iptables** jednodušší řešení, které přímo říká, které TOS bity musí vyhovovat nebo které TOS bity mají být nastaveny. Navíc je nemusíte specifikovat jejich šestnáctkovými hodnotami a můžete použít názvy uvedené v následující tabulce.

Obecná syntaxe pravidla, kterému vyhovují datagramy s určitým nastavením TOS je:

```
-m tos --tos název [další_parametry] -j akce
```

Obecná syntaxe pravidla pro nastavení TOS bitů je:

```
[další_parametry] -j TOS --set název
```

Tyto příkazy se obvykle používají současně, pokud to však potřebujete, můžete je použít i nezávisle.

Název	Šestnáctkově
Normal-Service	0x00
Minimize-Cost	0x02
Maximize-Reliability	0x04
Maximize-Throughput	0x08
Minimize-Delay	0x10

Testování konfigurace firewallu

Po navržení požadované konfigurace firewallu je nutné ověřit, že se konfigurace opravdu chová tak, jak jste chtěli. Jedna možnost je použít testovací počítač vně vaší sítě a vyzkoušet, jestli se vám nepodaří přes firewall proniknout. Tato metoda je ovšem velmi pracná a zdlouhavá a je omezena na testování pouze z těch adres, které jste schopni použít.

Implementace firewallu na Linuxu nabízí rychlejší a snadnější metodu. Umožňuje vám ručně navrhnout testy a spustit je proti konfiguraci firewallu stejně, jako byste posílali skutečné datagramy.

Tato metoda testování je podporována všemi generacemi firewallů na Linuxu, tedy **ipfwadm**, **ipchains** a **iptables**. Samotný test se provádí pomocí příkazu *cheсk*.

Obecný postup testu je následující:

1. Navrhněte a vytvořte firewall pomocí programu **ipfwadm**, **ipchains** nebo **iptables**.
2. Navrhněte sérii testů, které prověří, zda firewall funguje tak, jak potřebujete. U těchto testů můžete používat libovolné zdrojové a cílové adresy, takže zvolte různé kombinace adres tak, aby některé byly povoleny a jiné zakázány. Pokud povolujete nebo zakazujete adresy z nějakého rozsahu, je rozumné otestovat adresy na hranicích rozsahu – jednu adresu právě uvnitř a jednu adresu právě vně. Tím si ověříte, že máte hranice nastaveny správně, protože běžná chyba spočívá ve špatném nastavení síťové masky. Pokud filtrujete na základě protokolů nebo čísel portů, měli byste ověřit všechny významné kombinace všech parametrů. Pokud například chcete, aby za určitých okolností byly propouštěny pouze TCP datagramy, ověřte si, že UDP datagramy firewallem neprojdou.
3. Vytvořte pravidla pro programy **ipfwadm**, **ipchains** nebo **iptables**, která implementují jednotlivé testy. Obvykle se vyplatí zapsat všechna pravidla ve skriptu, takže budete moci testy spouštět opakovaně, pokud objevíte nějaké chyby v konfiguraci. Testy používají prakticky stejnou syntaxi jako pravidla firewallu, jednotlivé parametry však mají poněkud odlišný význam. Například parametr definující zdrojovou adresu v pravidle určuje, jaká musí být zdrojová adresa datagramu, aby pravidlu vyhovoval. Zdrojová adresa v testovacím pravidle naproti tomu určuje, jakou zdrojovou adresu bude mít vygenerovaný testovací datagram. V programu **ipfwadm** musíte použít přepínač `-c` udávající, že pravidlo je testovacím pravidlem. V programech **ipchains** a **iptables** slouží ke stejnému účelu parametr `-C`. Ve všech pravidlech musíte definovat zdrojovou adresu, cílovou adresu, protokol a rozhraní. Další parametry, například číslo portu nebo nastavení TOS bitů, jsou nepovinné.
4. Proveďte každý testovací příkaz a sledujte výstup. Výsledkem každého testovacího příkazu je jediné slovo sdělující, jaký je konečný cíl datagramu poté, co jej firewall zpracuje – tedy, co se s tímto datagramem stane. V programech **ipchains** a **iptables** jsou samozřejmě testovány i uživatelem definované třídy pravidel.
5. Porovnejte výsledky testů s požadovanými výsledky. Pokud narazíte na nějaké nesrovnalosti, budete muset analyzovat navržená pravidla a zjistit, kde jste udělali chybu. Pokud jste jednotlivé testovací příkazy zadali do skriptu, budete moci po opravě konfigurace firewallu snadno spustit test znovu. Bývá rozumné úplně vymazat nastavená pravidla a vytvářet je od počátku, nikoliv se pokoušet o jejich dynamické úpravy. Tím máte zajištěno, že testovaná konfigurace firewallu opravdu odpovídá té, kterou nastavujete v jeho konfiguračních skriptech.

Podívejme se v krátkosti na to, jak by vypadalo testování naší základní firewallové konfigurace pomocí programu **ipchains**. Připomeňme si, že lokální síť měla adresu 172.16.1.0 se síťovou maskou 255.255.255.0 a povolovali jsme pouze připojení z naší sítě na webové servery. Žádná jiná data by neměla firewallem projít. Začněme datagramem, o němž víme, že by měl projít – spojení z naší sítě na nějaký webový server:

```
# ipchains -C forward -p tcp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
accepted
```

Všimněte si, které parametry a jak jsme zadávali. Výstup příkazu nám říká, že datagram byl přijat k předání, což je to, co jsme očekávali.

Teď vyzkoušíme další test, tentokrát se zdrojovou adresou, která nepatří do naší sítě. Tento datagram by neměl projít:

```
# ipchains -C forward -p tcp -s 172.16.2.0 1025 -d 44.136.8.2 80 -i eth0
denied
```

Teď vyzkoušíme další testy, sice se stejnými vlastnostmi jako v prvním případě, ale s jinými protokoly. Tyto datagramy by projít neměly:

```
# ipchains -C forward -p udp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
denied
# ipchains -C forward -p icmp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
denied
```

Ještě vyzkoušíme jiný cílový port a opět předpokládáme, že datagram neprojde:

```
# ipchains -C forward -p tcp -s 172.16.1.0 1025 -d 44.136.8.2 23 -i eth0
denied
```

Návrh série vyčerpávajících testů je poměrně zdlouhavá záležitost. I když se může jednat o téměř stejně náročnou operaci jako byl samotný návrh konfigurace firewallu, je to jediná metoda jak ověřit, že návrh funguje tak, jak jste očekávali.

Příklad konfigurace firewallu

Představili jsme si základy konfigurace firewallu. Nyní se podíváme na to, jak by mohla vypadat skutečná konfigurace firewallu.

Konfigurace v následujícím příkladu byla navržena tak, aby ji bylo možné snadno rozšířit a upravit. Celý příklad předkládáme ve třech verzích. První z nich je implementována pomocí příkazu **ipfwadm** (nebo skriptem **ipfwadm-wrapper**), druhá používá **ipchains** a třetí **iptables**. V příkladu nevyužíváme uživatelem definované třídy, nicméně můžete si porovnat rozdíly a podobnosti mezi novou a starou syntaxí nástrojů pro konfiguraci firewallu.

```
#!/bin/bash
##############################################################################
# VERZE PRO IPFWADM
# Příklad konfigurace firewallu na jednom počítači, který sám
# neposkytuje žádné služby.
##############################################################################

# UŽIVATELEM KONFIGUROVATELNÁ ČÁST

# Název a umístění nástroje ipfwadm. Pro jádra 2.2.* zadejte ipfwadm-wrapper
IPFWADM=ipfwadm

# Cesta k souboru ipfwadm
PATH="/sbin"

# Interní adresový prostor naší sítě a jemu příslušné síťové zařízení
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Vnější adresy a odpovídající síťové zařízení
```

```
ANYADDR="0/0"
ANYDEV="eth1"

# TCP služby, které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"

# UDP služby, které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
UDPIN="domain"
UDPOUT="domain"

# ICMP služby, které chceme povolit.
# "" znamená všechny typy služeb
# Čísla typů viz /usr/include/netinet/ip_icmp.
# Mezerami oddělovaný seznam.
ICMPIN="0 3 11"
ICMPOUT="8 3 11"

# Logování. Odstraněním komentáře se zapne logování datagramů,
# které firewall nepropustil.
# LOGGING=1

# KONEC UŽIVATELEM KONFIGUROVATELNÉ ČÁSTI
################################################################################
# Vyprázdnění pravidel třídy Incoming
$IPFWADM -I -f

# Implicitně zakazujeme přístup třídou Incoming
$IPFWADM -I -p deny

# SPOOFING
# Nechceme zvenčí přijímat žádné datagramy se zdrojovou adresou
# z naší sítě, takže je blokujeme
$IPFWADM -I -a deny -S $OURNET -W $ANYDEV

# SMURF
# Zakazujeme ICMP na naši vysílací adresu, ochrana před "Smurf" útoky
$IPFWADM -I -a deny -P icmp -W $ANYDEV -D $OURBCAST

# TCP
# Přijímáme všechny TCP datagramy patřící existujícím spojením (tedy
# s nastaveným bitem ACK) pro všechny povolené TCP porty.
# To by mělo obsáhnout více než 95 % TCP paketů.
$IPFWADM -I -a accept -P tcp -D $OURNET $TCPIN -k -b

# TCP - PŘÍCHOZÍ SPOJENÍ
# Požadavky na TCP spojení zvenčí povolíme pouze na vybraných portech.
$IPFWADM -I -a accept -P tcp -W $ANYDEV -D $OURNET $TCPIN -y
```

```
# TCP - ODCHOZÍ SPOJENÍ
# Povolíme všechna odchozí spojení na povolených portech.
$IPFWADM -I -a accept -P tcp -W $OURDEV -D $ANYADDR $TCPOUT -y

# UDP - PŘÍCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPFWADM -I -a accept -P udp -W $ANYDEV -D $OURNET $UDPIN

# UDP - ODCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPFWADM -I -a accept -P udp -W $OURDEV -D $ANYADDR $UDPOUT

# ICMP - PŘÍCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPFWADM -I -a accept -P icmp -W $ANYDEV -D $OURNET $UDPIN

# ICMP - ODCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPFWADM -I -a accept -P icmp -W $OURDEV -D $ANYADDR $UDPOUT

# IMPLICITNÍ a LOGOVÁNÍ
# Všechny zbývající datagramy projdou k implicitnímu pravidlu a
# budou zahozeny. Pokud byla nastavena proměnná LOGGING, budou
# zaznamenány.
#
if [ "$LOGGING" ]
then
        # Logovat zahozené TCP
        $IPFWADM -I -a reject -P tcp -o

        # Logovat zahozené UDP
        $IPFWADM -I -a reject -P udp -o

        # Logovat zahozené ICMP
        $IPFWADM -I -a reject -P icmp -o
fi
#
# konec
```

Nyní budeme stejný příklad implementovat pomocí programu **ipchains**.

```
#!/bin/bash
################################################################################
# VERZE PRO IPCHAINS
# Příklad konfigurace firewallu na jednom počítači, který sám
# neposkytuje žádné služby.
################################################################################

# UŽIVATELEM KONFIGUROVATELNÁ ČÁST

# Název a umístění nástroje iptables.
IPCHAINS=ipchains

# Cesta k souboru ipchains
```

```
PATH="/sbin"

# Interní adresový prostor naší sítě a jemu příslušné síťové zařízení
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Vnější adresy a odpovídající síťové zařízení
ANYADDR="0/0"
ANYDEV="eth1"

# TCP služby, které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"

# UDP služby, které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
UDPIN="domain"
UDPOUT="domain"

# ICMP služby, které chceme povolit.
# "" znamená všechny typy služeb
# Čísla typů viz /usr/include/netinet/ip_icmp.
# Mezerami oddělovaný seznam.
ICMPIN="0 3 11"
ICMPOUT="8 3 11"

# Logování. Odstraněním komentáře se zapne logování datagramů,
# které firewall nepropustil.
# LOGGING=1

# KONEC UŽIVATELEM KONFIGUROVATELNÉ ČÁSTI
################################################################################
# Vyprázdnění pravidel třídy Input
$IPCHAINS -F input

# Implicitně zakazujeme přístup třídou Input
$IPCHAINS -P input deny

# SPOOFING
# Nechceme zvenčí přijímat žádné datagramy se zdrojovou adresou
# z naší sítě, takže je blokujeme
$IPCHAINS -A input -s $OURNET -i $ANYDEV -j deny

# SMURF
# Zakazujeme ICMP na naši vysílací adresu, ochrana před "Smurf" útoky
$IPCHAINS -A input -p icmp -w $ANYDEV -d $OURBCAST -j deny

# Chceme přijímat fragmenty, v ipchains to musíme explicitně povolit.
$IPCHAINS -A input -f -j accept
```

```
# TCP
# Přijímáme všechny TCP datagramy patřící existujícím spojením (tedy
# s nastaveným bitem ACK) pro všechny povolené TCP porty.
# To by mělo obsáhnout více než 95 % TCP paketů.
$IPCHAINS -A input -p tcp -d $OURNET $TCPIN ! -y -b -j accept

# TCP - PŘÍCHOZÍ SPOJENÍ
# Požadavky na TCP spojení zvenčí povolíme pouze na povolených portech.
$IPCHAINS -A input -p tcp -i $ANYDEV -d $OURNET $TCPIN -y -j accept

# TCP - ODCHOZÍ SPOJENÍ
# Povolíme všechna odchozí spojení na povolených portech.
$IPCHAINS -A input -p tcp -i $OURDEV -d $ANYADDR $TCPOUT -y -j accept

# UDP - PŘÍCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPCHAINS -A input -p udp -i $ANYDEV -d $OURNET $UDPIN -j accept

# UDP - ODCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPCHAINS -A input -p udp -i $OURDEV -d $ANYADDR $UDPOUT -j accept

# ICMP - PŘÍCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPCHAINS -A input -p icmp -w $ANYDEV -d $OURNET $UDPIN -j accept

# ICMP - ODCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPCHAINS -A input -p icmp -i $OURDEV -d $ANYADDR $UDPOUT -j accept

# IMPLICITNÍ a LOGOVÁNÍ
# Všechny zbývající datagramy projdou k implicitnímu pravidlu a
# budou zahozeny. Pokud byla nastavena proměnná LOGGING, budou
# zaznamenány.
#
if [ "$LOGGING" ]
then
      # Logovat zahozené TCP
        $IPCHAINS -A input -p tcp -l -j reject

# Logovat zahozené UDP
        $IPCHAINS -A input -p udp -l -j reject

        # Logovat zahozené ICMP
        $IPCHAINS -A input -p icmp -l -j reject
fi
#
# konec.
```

V příkladu používajícím program **iptables** pracujeme s pravidly třídy FORWARD, protože třída INPUT má v této implementaci jiný význam. Vedlejším efektem je, že žádná z pravidel nechrání samotný firewall. Pokud bychom chtěli přesně replikovat příklad s programem **ipchains**, museli

bychom všechna pravidla zkopírovat i do třídy INPUT. Kvůli jednoduchosti místo toho zahazujeme všechny datagramy přijaté z vnější sítě.

```
#!/bin/bash
###############################################################################
# VERZE PRO IPTABLES
# Příklad konfigurace firewallu na jednom počítači, který sám
# neposkytuje žádné služby.
###############################################################################

# UŽIVATELEM KONFIGUROVATELNÁ ČÁST

# Název a umístění nástroje ipchains.
IPTABLES=iptables

# Cesta k souboru ipchains.
PATH="/sbin"

# Interní adresový prostor naší sítě a jemu příslušné síťové zařízení
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Vnější adresy a odpovídající síťové zařízení
ANYADDR="0/0"
ANYDEV="eth1"

# TCP služby, které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
TCPIN="smtp,www"
TCPOUT="smtp,www,ftp,ftp-data,irc"

# UDP služby, které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
UDPIN="domain"
UDPOUT="domain"

# ICMP služby, které chceme povolit.
# "" znamená všechny typy služeb
# Čísla typů viz /usr/include/netinet/ip_icmp.
# Mezerami oddělovaný seznam.
ICMPIN="0,3,11"
ICMPOUT="8,3,11"

# Logování. Odstraněním komentáře se zapne logování datagramů,
# které firewall nepropustil.
# LOGGING=1

# KONEC UŽIVATELEM KONFIGUROVATELNÉ ČÁSTI
###############################################################################
# Vyprázdnění pravidel třídy Forward
$IPTABLES -F FORWARD
```

```
# Implicitně zakazujeme přístup třídou Forward
$IPTABLES -P FORWARD deny

# Zahození všech datagramů pro tento počítač zvenčí.
$IPTABLES -A INPUT -i $ANYDEV -j DROP

# SPOOFING
# Nechceme zvenčí přijímat žádné datagramy se zdrojovou adresou
# z naší sítě, takže je blokujeme
$IPTABLES -A FORWARD -s $OURNET -i $ANYDEV -j DROP

# SMURF
# Zakazujeme ICMP na naši vysílací adresu, ochrana před "Smurf" útoky
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET -j DENY

# Chceme přijímat fragmenty, v iptables to musíme explicitně povolit.
$IPTABLES -A FORWARD -f -j ACCEPT

# TCP
# Přijímáme všechny TCP datagramy patřící existujícím spojením (tedy
# s nastaveným bitem ACK) pro všechny povolené TCP porty.
# To by mělo obsáhnout více než 95 % TCP paketů.
$IPTABLES -A FORWARD -m multiport -p tcp -d $OURNET --dports $TCPIN /
    ! --tcp-flags SYN,ACK ACK -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p tcp -s $OURNET --sports $TCPIN /
    ! --tcp-flags SYN,ACK ACK -j ACCEPT


# TCP - PŘÍCHOZÍ SPOJENÍ
# Požadavky na TCP spojení zvenčí povolíme pouze na povolených portech.
$IPTABLES -A FORWARD -m multiport -p tcp -i $ANYDEV -d $OURNET $TCPIN /
    --syn -j ACCEPT

# TCP - ODCHOZÍ SPOJENÍ
# Povolíme všechna odchozí spojení na povolených portech.
$IPTABLES -A FORWARD -m multiport -p tcp -i $OURDEV -d $ANYADDR /
    --dports $TCPOUT --syn -j ACCEPT

# UDP - PŘÍCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -d $OURNET /
    --dports $UDPIN -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -s $OURNET /
    --sports $UDPIN -j ACCEPT

# UDP - ODCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -d $ANYADDR /
    --dports $UDPOUT -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -s $ANYADDR /
            --sports $UDPOUT -j ACCEPT
```

```
# ICMP - PŘÍCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET /
    --dports $ICMPIN -j ACCEPT

# ICMP - ODCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPTABLES -A FORWARD -m multiport -p icmp -i $OURDEV -d $ANYADDR /
    --dports $ICMPOUT -j ACCEPT

# IMPLICITNÍ a LOGOVÁNÍ
# Všechny zbývající datagramy projdou k implicitnímu pravidlu a
# budou zahozeny. Pokud byla nastavena proměnná LOGGING, budou
# zaznamenány.
#
if [ "$LOGGING" ]
then
# Logovat zahozené TCP
        $IPTABLES -A FORWARD -m tcp -p tcp -j LOG
      # Logovat zahozené UDP
        $IPTABLES -A FORWARD -m udp -p udp -j LOG
        # Logovat zahozené ICMP
        $IPTABLES -A FORWARD -m udp -p icmp -j LOG
fi
#
# konec.
```

V řadě jednoduchých situací budete moci tento příklad použít s tím, že pouze upravíte údaje v části označené jako „uživatelem konfigurovatelná“, kde specifikujete, které protokoly a typy datagramů chcete propouštět dovnitř a ven. Ve složitějších případech budete muset modifikovat i spodní část příkladu. Nezapomeňte, že se jedná pouze o příklad, takže jej pečlivě prostudujte, zda opravdu dělá to, co vy potřebujete.

IP účtování

V současném světě komerčních internetových služeb je stále důležitější vědět, kolik dat přijímáte a odesíláte svým síťovým připojením. Pokud jste poskytovatel internetového připojení a účtujete podle objemu přenesených dat, je pro vás tato znalost nezbytná. Pokud jste zákazník poskytovatele, který podle objemu dat účtuje, určitě vás bude zajímat, nakolik jsou údaje uváděné poskytovatelem správné.

Kromě toho má účtování i jiné využití, nemusí jít jen o peníze a poplatky. Pokud provozujete nějaký server, který nabízí řadu různých síťových služeb, jistě vás bude zajímat, nakolik jsou jednotlivé služby zatěžovány. Pomocí těchto informací se usnadňují rozhodnutí jako zda aktualizovat hardware, zda rozdělit služby na více serverů a podobně.

Jádro Linuxu nabízí funkci, která umožňuje shromažďovat všechny typy užitečných informací o síťovém provozu, který jádro vidí. Tato funkce se označuje jako *IP účtování*.

Konfigurace jádra pro účtování

Účtovací služby jsou úzce spjaty s firewally. Místa, kde se účtovací data shromažďují, odpovídají těm místům, kde probíhá filtrace firewallem: směr do a z vašeho počítače a software zajišťující směrování. Pokud jste nečetli kapitolu věnovanou firewallům, bude dobré to napravit, protože zde budeme vycházet z některých znalostí, které jsme se dozvěděli v kapitole 9.

Než budete účtování aktivovat, musíte si ověřit, zda je jádro pro tuto službu nakonfigurováno. Podívejte se, zda existuje soubor `/proc/net/ip_acct`. Pokud ano, jádro účtování podporuje. Pokud ne, musíte sestavit nové jádro a v jádrech 2.0 a 2.2 odpovědět „Y“ na následující dotazy:

```
Networking options  --->
        [*] Network firewalls
        [*] TCP/IP networking
         ...
        [*] IP: accounting
V jádrech 2.4 pak na tyto otázky:
Networking options  --->
    [*] Network packet filtering (replaces ipchains)
```

Konfigurace IP účtování

Protože je účtování blízce spjato s funkcemi firewallu, konfiguruje se i stejným nástrojem, tedy programy **ipfwadm**, **ipchains** nebo **iptables** podle verze jádra. Syntaxe příkazu se velmi podobá klasickým pravidlům firewallu, takže o ní nebudeme příliš hovořit a místo toho se zaměříme na to, co můžete pomocí této funkce o síťovém provozu zjistit.

Obecná syntaxe příkazu **ipfwadm** pro konfiguraci IP účování je:

```
# ipfwadm -A [směr] [příkaz] [parametry]
```

Nový je parametr *směr*. Může mít hodnoty in, out nebo both. Směry jsou chápány z pohledu účtovacího počítače, tedy in znamená data přicházející ze sítě do tohoto počítače, out data odcházející z tohoto počítače na síť a both je jednoduše součet obou těchto směrů.

Obecná syntaxe příkazů **ipchains** a **iptables** je:

```
# ipchains -A třída specifikace_pravidla
# iptables -A třída specifikace_pravidla
```

Příkazy **ipchains** a **iptables** umožňují definovat směr způsobem, který je konzistentnější se samotnými pravidly firewallu. V těchto programech sice nemůžete specifikovat pravidla provádějící účtování pro oba směry současně, na druhé straně můžete specifikovat pravidla pro směrovací třídu, což ve starších verzích nebylo možné. Rozdíly uvidíme v příkladech o něco později.

Příkazy jsou stejné jako pravidla pro konfiguraci firewallu, pouze se zde nespecifikuje akce, která se má s datagramem provést. Účtovací pravidla je možné známými způsoby přidávat, vkládat, mazat a vypisovat. V programech **ipchains** a **iptables** jsou účtovacími pravidly všechna platná pravidla, pravidla, u nichž není specifikována akce parametrem –j pak fungují jako čistě účtovací.

Parametry specifikující pravidla jsou pro účtování stejná jako pro samotný firewall. Pomocí nich přesně definujeme typy síťového provozu, které chceme účtovat.

Účtování podle adresy

Podívejme se nyní na příklad ilustrující, jak IP účtování používat.

Představme si směrovač, který spojuje dvě oddělení virtuálního pivovaru. Směrovač má dvě ethernetová rozhraní, eth0 a eth1, která jsou připojena ke zmíněným dvěma odděleními, a rozhraní ppp0, jež nás vysokorychlostní sériovou linkou připojuje k síti univerzity Groucho Marx.

Dále si představme, že pro potřeby proplácení potřebujeme znát objemy dat generované jednotlivými odděleními na sériové lince a pro potřeby správy potřebujeme znát objem dat přenášený mezi odděleními.

Následující tabulka shrnuje adresy rozhraní, které budeme v příkladu používat:

Rozhraní	Adresa	Maska
eth0	172.16.3.0	255.255.255.0
eth1	172.16.4.0	255.255.255.0

Abychom mohli odpovědět na otázku „Kolik provozu generují jednotlivá rozdělení na lince PPP?“, zavedeme následující pravidla:

```
# ipfwadm -A both -a -W ppp0 -S 172.16.3.0/24 -b
# ipfwadm -A both -a -W ppp0 -S 172.16.4.0/24 -b
```

nebo

```
# ipchains -A input -i ppp0 -d 172.16.3.0/24
# ipchains -A output -i ppp0 -s 172.16.3.0/24
# ipchains -A input -i ppp0 -d 172.16.4.0/24
# ipchains -A output -i ppp0 -s 172.16.4.0/24
```

a s programem **iptables**

```
# iptables -A FORWARD -i ppp0 -d 172.16.3.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.3.0/24
# iptables -A FORWARD -i ppp0 -d 172.16.4.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.4.0/24
```

První polovina pravidel v každé skupině říká „počítej data přenášená oběma směry přes rozhraní ppp0 se zdrojem nebo cílem (připomeňme si funkci příznaku –b v programech **ipfwadm** a **ipchains**) 172.16.3.0./24“. Druhá polovina pravidel dělá to samé, ovšem pro druhou ethernetovou síť.

Abychom mohli odpovědět na druhou otázku, „Kolik dat se přenáší mezi odděleními?“, potřebujeme následující pravidla:

```
# ipfwadm -A both -a -S 172.16.3.0/24 -D 172.16.4.0/24 -b
```

nebo

```
# ipchains -A forward -s 172.16.3.0/24 -d 172.16.4.0/24 -b
```

nebo

```
# iptables -A FORWARD -s 172.16.3.0/24 -d 172.16.4.0/24
# iptables -A FORWARD -s 172.16.4.0/24 -d 172.16.3.0/24
```

Tato pravidla počítají datagramy se zdrojovou adresou v jednom oddělení a s cílovou adresou v druhém a naopak.

Účtování podle portu služby

Předpokládejme nyní, že chceme získat přesnější představu o tom, jaký typ provozu se přenáší přes PPP linku. Může nás například zajímat, nakolik je linka vytížena službami FTP, SMTP a WWW.

Skript s pravidly, která tyto informace získají, může vypadat takto:

```
#!/bin/sh
# Statistiky FTP, SMTP a WWW přenosů přes PPP linku pro ipfwadm
#
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 ftp ftp-data
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 smtp
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 www
nebo
#!/bin/sh
# Statistiky FTP, SMTP a WWW přenosů přes PPP linku pro ipchains
#
ipchains -A input -i ppp0 -p tcp -s 0/0 ftp-data:ftp
ipchains -A output -i ppp0 -p tcp -d 0/0 ftp-data:ftp
ipchains -A input -i ppp0 -p tcp -s 0/0 smtp
ipchains -A output -i ppp0 -p tcp -d 0/0 smtp
ipchains -A input -i ppp0 -p tcp -s 0/0 www
ipchains -A output -i ppp0 -p tcp -d 0/0 www
```

nebo

```
#!/bin/sh
# Statistiky FTP, SMTP a WWW přenosů přes PPP linku pro iptables
```

```
#
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport ftp-data:ftp
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport ftp-data:ftp
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport smtp
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport smtp
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport www
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport www
```

V této konfiguraci najdeme řadu zajímavých momentů. Nejprve si všimněte, že jsme specifikovali protokol. Protože v pravidlech definujeme čísla portů, musíme uvést i protokol, jelikož protokoly TCP a UDP používají samostatné porty. Všechny zmíněné služby používají protokol TCP, proto jsme specifikovali jej. Dále jsme v jednom příkazu definovali dvě služby – ftp a ftp-data. Program **ipfwadm** dovoluje definovat jeden port, rozsah portů nebo seznam portů. Příkaz **ipchains** pak dovoluje definovat jeden port nebo rozsah portů a této možnosti jsme také využili. Zápis „ftp-data:ftp" znamená „porty od ftp-data (20) po ftp (21)" a tímto zápisem v programech **ipchains** i **iptables** specifikujeme skupinu portů. Pokud je v účtovacím pravidle specifikováno více portů, budou se do celkových statistik započítávat data přijatá nebo odeslaná kterýmkoliv z těchto portů. Protože víme, že služba FTP používá dva porty, jeden pro přenos příkazů a druhý pro přenos dat, uvedli jsme je oba dva, abychom získali celkové statistiky FTP přenosů. Dále jsme definovali zdrojovou adresu „0/0", což je speciální notace označující všechny adresy – v programech **ipfwadm** a **ipchains** je totiž při zadání portů nutné zadat i adresu.

Podívejme se nyní na tento příklad blíže a zkusme získat o využití rozhraní přesnější představu. Řekněme, že služby WWW, FTP a SMTP představují „základní" provoz, všechno ostatní je „nezákladní" provoz. Pokud nás bude zajímat poměr mezi základním a nezákladním provozem, můžeme definovat například takováto pravidla:

```
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 ftp ftp-data smtp www
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 1:19 22:24 26:79 81:32767
```

Podíváte-li se do souboru `/etc/services`, zjistíte, že druhé pravidlo pokrývá všechny porty kromě portů služeb WWW, FTP a SMTP.

Jak to budeme implementovat pomocí programů **ipchains** a **iptables**, které dovolují definovat port pouze jediným parametrem? V takovém případě můžeme s výhodou využít uživatelsky definované třídy stejně, jako jsme to dělali při konfiguraci samotného firewallu. Podívejte se na následující řešení:

```
# ipchains -N a-essent
# ipchains -N a-noness
# ipchains -A a-essent -j ACCEPT
# ipchains -A a-noness -j ACCEPT
# ipchains -A forward -i ppp0 -p tcp -s 0/0 ftp-data:ftp -j a-essent
# ipchains -A forward -i ppp0 -p tcp -s 0/0 smtp -j a-essent
# ipchains -A forward -i ppp0 -p tcp -s 0/0 www -j a-essent
# ipchains -A forward -j a-noness
```

Vytvořili jsme dvě uživatelské třídy, a-essent, kde shromažďujeme účtovací data základních služeb, a a-noness, kde shromažďujeme data ostatních služeb. Pak jsme do směrovací třídy přidali pravidla, která pokrývají základní služby a přeskakují do třídy a-essent, v níž se nachází jediné pravidlo povolující všechny datagramy a zároveň shromažďující jejich statistiky. Poslední pravidlo skáče do třídy a-noness, kde máme opět jediné pravidlo, povolující a počítající všechny datagramy. Pravidlo směřující do třídy a-noness nebude uplatněno na žádnou ze základních služeb, protože ty už byly povoleny ve své samostatné třídě. Statistiky základních a ostatních služeb tak budeme

mít k dispozici v pravidlech v námi definovaných třídách. Toto řešení je pouze jedno z možných, jsou i jiná. Implementace stejného chování programem **iptables** bude vypadat takto:

```
# iptables -N a-essent
# iptables -N a-noness
# iptables -A a-essent -j ACCEPT
# iptables -A a-noness -j ACCEPT
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport ftp-data:ftp -j a-essent
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport smtp -j a-essent
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport www -j a-essent
# iptables -A FORWARD -j a-noness
```

Všechno vypadá poměrně jednoduše. Bohužel, při účtování podle typů služeb narážíme na jeden malý, nicméně nezanedbatelný problém. Jistě si vzpomenete, když jsme v souvislosti se sítěmi TCP/IP hovořili o hodnotě MTU. Hodnota MTU definuje velikost největšího možného datagramu, který bude síťovým zařízením přenesen. Pokud směrovač přijme datagram delší než je MTU rozhraní, přes něž má být odeslán, provede směrovač trik zvaný *fragmentace*. Směrovač rozdělí velký datagram na části kratší než MTU daného rozhraní a odešle tyto části. Pro jednotlivé části vygeneruje směrovač nové hlavičky a pomocí nich bude přijímající počítač schopen data zrekonstruovat. Bohužel, procesem fragmentace se u všech kromě prvního fragmentu ztrácí číslo portu. Znamená to, že IP účtování bude schopné správně zaznamenávat pouze první fragmenty a nefragmentované datagramy. Program **ipfwadm** používá malý trik, takže i když nebudeme vědět, které službě jednotlivé fragmenty patří, stále je budeme schopni počítat. První verze účtovacích programů na Linuxu přiřazovaly fragmentům falešné číslo portu 0xFFFF, a to jsme mohli počítat. K zachycení druhého a dalších fragmentů proto použijeme následující pravidlo:

```
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 0xFFFF
```

Implementace použitá v IP Chains je sice trochu propracovanější, nicméně výsledek je prakticky stejný. S programem **ipchains** použijeme následující příkaz:

```
# ipchains -A forward -i ppp0 -p tcp -f
A s iptables:
# iptables -A FORWARD -i ppp0 -m tcp -p tcp -f
```

Ani zde se nedozvíme původní port přenášených dat, nicméně budeme alespoň schopni říct, kolik dat je fragmentovaných a budeme schopni určit, kolik provozu tato data tvoří.

V jádrech 2.2 můžete při sestavování jádra použít parametr, který celý problém vyřeší za předpokladu, že daný počítač představuje jediný přístupový uzel do sítě. Pokud při sestavování jádra zadáte volbu IP: always defragment, linuxový směrovač všechny fragmentované datagramy před dalším směrováním a odesláním zrekonstruuje. Tato operace se provádí předtím, než datagramy zpracovává firewall a účtovací systém, takže se k nim nikdy žádné fragmenty nedostanou. V jádrech 2.2 musíte přeložit a nahrát modul forward-fragment z balíku *netfilter*.[86]

Účtování ICMP datagramů

Protokol ICMP nepoužívá čísla portů, a proto se podrobnosti o statistikách tohoto protokolu zjišťují obtížněji. Protokol ICMP používá různé typy datagramů. Řada z nich je neškodných a běžných, jiné se objevují jen za speciálních okolností. Občas se znudění útočníci pokoušejí zablokovat přístup k nějakému systému tím, že na něj posílají obrovské množství ICMP datagramů. Tento po-

86 Pozn. překladatele: Jak bylo řečeno, tento přístup lze použít pouze v případě, že jste k vnějšímu světu připojeni jediným uzlem. Pokud máte přípojných bodů více, není zaručeno, že všechny fragmenty jednoho datagramu projdou stejným uzlem – bude je sestavovat až cílový počítač a směrovač tuto funkci nemůže plnit, protože prostě nemusí všechny fragmenty obdržet.

stup se běžně označuje jako *ping flooding*. I když IP účtování s tím nemůže nic udělat (ovšem firewall do jisté míry může[87]), můžeme přinejmenším vytvořit účtovací pravidla z nichž se dozvíme, že se někdo o takovýto útok pokouší.

Protokol ICMP nepoužívá porty tak jako protokoly TCP a UDP. Namísto toho pracuje s různými typy ICMP zpráv. Můžeme vytvářet pravidla pro účtování jednotlivých typů těchto zpráv. V programu **ipfwadm** to uděláme tak, že místo čísla portu specifikujeme číslo typu ICMP zprávy. Typy ICMP jsme uvedli v části *Typy ICMP datagramů* v kapitole 9.

Účtovací pravidla zjišťující statistiky dat příkazu **ping** budou vypadat takto:

```
# ipfwadm -A both -a -P icmp -S 0/0 8
# ipfwadm -A both -a -P icmp -S 0/0 0
# ipfwadm -A both -a -P icmp -S 0/0 0xff
```

nebo pro **ipchains**

```
# ipchains -A forward -p icmp -s 0/0 8
# ipchains -A forward -p icmp -s 0/0 0
# ipchains -A forward -p icmp -s 0/0 -f
```

nebo pro **iptables**

```
# iptables -A FORWARD -m icmp -p icmp --sports echo-request
# iptables -A FORWARD -m icmp -p icmp --sports echo-reply
# iptables -A FORWARD -m icmp -p icmp -f
```

První pravidlo počítá zprávy „ICMP Echo Request“ (žádosti programu **ping**), druhé pravidlo zprávy „ICMP Echo Reply“ (odpovědi na tyto žádosti). Třetí pravidlo počítá statistiky ICMP fragmentů. Jedná se o podobný trik, o kterém jsme hovořili v souvislosti s fragmenty TCP a UDP.

Pokud v pravidlech specifikujete zdrojové a/nebo cílové adresy, můžete získávat i informace o tom, odkud pakety pocházejí – například zda zevnitř nebo vně vaší sítě. Jakmile zjistíte odkud datagram pochází, můžete na firewallu zavést pravidla, která je zablokují, nebo můžete podniknout jiné akce, například kontaktovat správce vzdálené sítě s žádostí o řešení problému, případně můžete zvolit nějaký radikálnější postup v případě, že se jedná o napadení vaší sítě.

Účtování podle protokolu

Nyní si představme, že nás zajímá, jaký provoz připadá na protokoly TCP, UDP a ICMP. Můžeme to zjistit následujícími pravidly:

```
# ipfwadm -A both -a -W ppp0 -P tcp -D 0/0
# ipfwadm -A both -a -W ppp0 -P udp -D 0/0
# ipfwadm -A both -a -W ppp0 -P icmp -D 0/0
```

nebo

```
# ipchains -A forward -i ppp0 -p tcp -d 0/0
# ipchains -A forward -i ppp0 -p udp -d 0/0
# ipchains -A forward -i ppp0 -p icmp -d 0/0
```

nebo

87 Pozn. překladatele: Většinou nepomůže ani firewall. Předpokládejme typickou konfiguraci, kdy jste k Internetu připojeni relativně pomalou linkou přes nějaký směrovač či firewall a další rozvod v interní síti je řešen relativně rychlými linkami. Pokud vás někdo zahltí ICMP datagramy, firewall je sice může všechny zahodit, takže neproniknou dále do interní sítě, nicméně internetová linka zůstává plně vytížena a nebudete ji moci použít. Pak vám musí pomoci někdo na druhém konci vaší linky (tedy zřejmě poskytovatel) a zablokovat datagramy na *svém* firewallu, aby se na vaši linku vůbec nedostaly.

```
# iptables -A FORWARD -i ppp0 -m tcp -p tcp
# iptables -A FORWARD -o ppp0 -m tcp -p tcp
# iptables -A FORWARD -i ppp0 -m udp -p udp
# iptables -A FORWARD -o ppp0 -m udp -p udp
# iptables -A FORWARD -i ppp0 -m icmp -p icmp
# iptables -A FORWARD -o ppp0 -m icmp -p icmp
```

Při zavedení těchto pravidel budou vyhodnocovány všechny datagramy přicházející PPP linkou a budou se zvlášť počítat statistiky pro TCP, UDP a ICMP datagramy. V programu **iptables** sledujeme příchozí a odchozí provoz samostatně, protože program neumí pracovat s obousměrnými pravidly.

Vyhodnocení výsledků účtování

Je sice hezké, že dokážeme statistiky přenosů zjišťovat, jak se na ně ale podíváme? Zajímají-li nás nashromážděná data a odpovídající pravidla, použijeme příkaz pro vypsání pravidel firewallu. Ve výstupu u každého pravidla jsou uvedeny hodnoty počitadel paketů a bajtů.

V příkazech **ipfwadm**, **ipchains** a **iptables** se práce s účtovacími daty liší, takže o nich budeme hovořit samostatně.

Výpis dat programem ipfwadm

Nejjednodušší způsob zjištění účtovacích dat programem **ipfwadm** bude vypadat takto:

```
# ipfwadm -A -l
IP accounting rules
pkts  bytes dir prot source              destination          ports
 9833 2345K i/o all  172.16.3.0/24       anywhere             n/a
56527   33M i/o all  172.16.4.0/24       anywhere             n/a
```

Takto se dozvíme počet paketů odeslaných v jednotlivých směrech. Pokud použijeme rozšířený výpis pomocí parametru –e (neuvádíme jej zde, protože se nevejde na šířku stránky), uvidíme také zadané parametry a názvy rozhraní. Význam většiny údajů ve výstupu by měl být zřejmý, nicméně následující údaje si možná zaslouží vysvětlení:

dir	Směr platnosti pravidla. Možné hodnoty jsou in, out a both.
prot	Protokol, pro nějž pravidlo platí.
opt	Parametry, které jsme zadali v pravidlu.
ifname	Název rozhraní, pro nějž pravidlo platí.
ifaddress	Adresa rozhraní, pro niž pravidlo platí.

Program **ipfwadm** vypisuje standardně počitadla ve zkráceném tvaru, zaokrouhleně na nejbližší tisíce (K) nebo miliony (M). Přesné hodnoty statistik můžeme zjistit pomocí následujících parametrů:

```
# ipfwadm -A -l -e -x
```

Výpis dat programem ipchains

Program **ipchains** nevypisuje účtovací data, pokud si o ně neřekneme parametrem –v. Nejjednodušší způsob vypsání těchto dat proto vypadá:

```
# ipchains -L -v
```

Stejně jako v programu **ipfwadm** i zde můžeme přepínačem –x požádat o rozšířený výpis, kdy budou hodnoty uvedeny v příslušných jednotkách:

```
# ipchains -L -v -x
```

Výpis dat programem iptables

Program **iptables** se chová velmi podobně jako program **ipchains**. I zde musíme uvést parametr `-v`, aby došlo k zobrazení účtovacích dat:

```
# iptables -L -v
```

Rovněž zde můžeme použít přepínač –x pro zobrazení rozšířeného výpisu.

Nulování počitadel

Pokud necháte účtování běžet příliš dlouho, počitadla přetečou. Dojde-li k přetečení, těžko zjistíte, jaké hodnoty vlastně byly napočítány. Abyste těmto problémům předešli, měli byste data číst pravidelně, zaznamenávat je a nulovat počitadla, takže budete získávat data periodicky pro nastavené intervaly.

V programech **ipfwadm** a **ipchains** můžete nulování zajistit velmi jednoduše:

```
# ipfwadm -A -z
```

nebo

```
# ipchains -Z
```

nebo

```
# iptables -Z
```

Můžete dokonce zkombinovat vypsání dat a vynulování počitadel do jednoho příkazu, takže budete mít zajištěno, že nedojde k žádné ztrátě informací mezi těmito dvěma operacemi:

```
# ipfwadm -A -l -z
```

nebo

```
# ipchains -L -Z
```

nebo

```
# iptables -L -Z -v
```

Tyto příkazy vypíší statistiky počitadel a ihned počitadla vynulují. Pokud budete chtít data shromažďovat pravidelně, uvedete zřejmě tyto příkazy ve skriptu, který výstup zaznamená a někam uloží, přičemž skript budete spouštět periodicky pomocí příkazu **cron**.

Vymazání pravidel

Poslední možný užitečný příkaz umožňuje zrušit všechna nastavená účtovací pravidla. Je to užitečné zejména pokud budete chtít zásadně změnit způsob účtování, aniž byste museli restartovat účtovací počítač.

Parametr `-f` příkazu **ipfwadm** vymaže všechna pravidla zadaného typu. Program **ipchains** má přepínač `-F`, který dělá to samé:

```
# ipfwadm -A -f
```

nebo

```
# ipchains -F
```

nebo

```
# iptables -F
```

Těmito příkazy se odstraní všechna nadefinovaná účtovací pravidla, takže je nemusíte odstraňovat jedno po druhém. Vymazání pravidel v programu **ipchains** nezpůsobí odstranění uživatelem definovaných tříd, budou pouze vymazána pravidla v těchto třídách.

Pasivní shromažďování účtovacích dat

Poslední zajímavý trik: Pokud máte počítač připojený k Ethernetu, můžete pomocí účtovacích pravidel zaznamenávat všechna data na daném segmentu, ne jenom data vysílaná a přijímaná účtovacím počítačem. Počítač bude pasivně přijímat všechna data na segmentu a zaznamená je.

Nejprve musíte na počítači vypnout předávání IP datagramů, aby se nepokoušel směrovat všechna data, která přijme[88]. V jádrech 2.0.36 a 2.2 se to provede příkazem:

```
# echo 0 >/proc/sys/net/ipv4/ip_forward
```

Pak příkazem ifconfig přepnete ethernetové rozhraní do promiskuitního režimu. Nyní můžete zavést účtovací pravidla zaznamenávající provoz na celém segmentu, aniž by se přímo týkala účtovacího počítače.

88 Nemůžete to samozřejmě udělat, pokud počítač pracuje jako směrovač. Vypnete-li v takovém případě předávání datagramů, počítač přestane směrovat. Uvedený postup se hodí pouze pro systém s jediným fyzickým rozhraním.

KAPITOLA 11

IP maškaráda a překlad síťových adres

Nemusíte mít nijak dobrou paměť, abyste si pamatovali doby, kdy si propojení více počítačů v síti mohly dovolit pouze velké organizace. Současné síťové technologie jsou natolik levné, že došlo hned ke dvěma věcem. Za prvé jsou lokální sítě dnes zcela běžné, dokonce i u domácích uživatelů. Řada uživatelů Linuxu používá dva nebo více počítačů propojených nějakým Ethernetem. Za druhé jsou síťové prostředky, zejména IP adresy, velice vzácným artiklem a zatímco kdysi se přidělovaly zdarma, dnes se běžně kupují a prodávají.

Většina uživatelů s lokální sítí a připojením k Internetu bude zřejmě chtít, aby byl Internet přístupný ze všech počítačů sítě. Směrovací pravidla protokolu IP jsou ovšem velice přísná. Tradiční řešení by vyžadovalo získat celou síť IP adres, pravděpodobně síť třídy C, pak jednotlivým počítačům na síti přidělit adresy z této oblasti a konečně celou síť nějakým směrovačem připojit k Internetu.

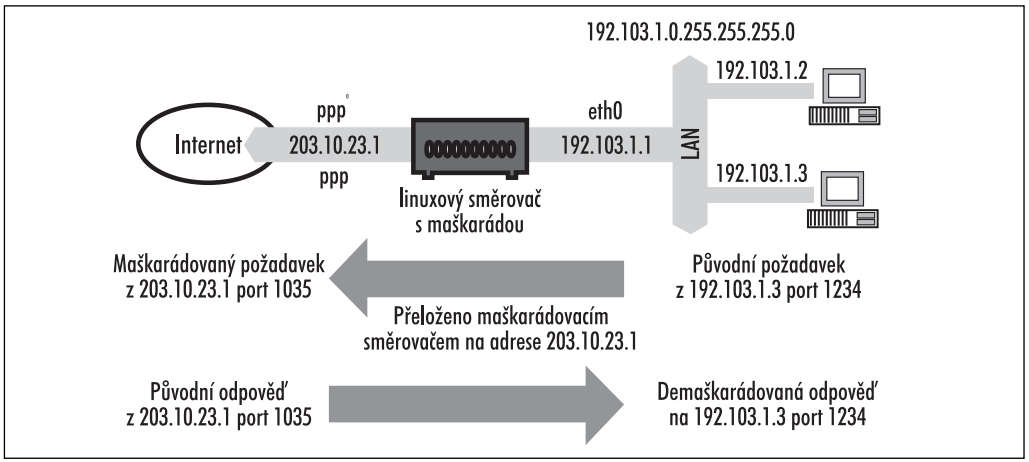
V komerčním prostředí Internetu je ale takové řešení nesmírně drahé. Budete muset platit za přidělení sítě adres. Dále budete zřejmě platit poskytovateli připojení za nastavení tras na vaši síť tak, aby byla z Internetu dostupná. Pro velké společnosti může být toto řešení dostupné, u domácích instalací se však zřejmě finančně nevyplatí.

Naštěstí Linux nabízí řešení tohoto problému. Toto řešení je tvořeno skupinou síťových funkcí, označovaných jako *Překlad síťových adres* (Network Address Translation, NAT). NAT je proces, který mění síťové adresy v hlavičkách datagramů v době jejich přenosu. Na první pohled to zní zvláštně, za chvíli však uvidíme, že tím vyřešíme před chvílí zmiňovaný problém. IP maškaráda je jedna z technologií NAT, která umožňuje všem hostitelům na síti přistupovat k Internetu prostřednictvím jediné IP adresy.

IP maškaráda dovoluje, aby všechny počítače na síti používaly privátní (rezervované) IP adresy a linuxový směrovač bude provádět inteligentní překlady adres a portů v čase přenosu. Když přijme datagram z počítače v lokální síti, zjistí, o jaký typ datagramu jde (tedy TCP, UDP, IMCP a podobně) a změní datagram tak, aby to vypadalo, že jej odeslal samotný směrovač (a zároveň si zapamatuje, že to udělal). Pak pošle datagram na Internet prostřednictvím své jediné platné IP adresy. Když konečný adresát tento datagram přijme, domnívá se, že pochází přímo od směrovače a pošle mu odpověď. Když směrovač implementující maškarádu takovýto datagram přijme, podí-

vá se do tabulek maškarádovaných spojení a zjistí, zda daný datagram patří nějakému počítači v lokální síti, a pokud ano, opět provede úpravu hlaviček datagramu na původní hodnoty a odešle datagram na lokální síť.

Jednoduchý příklad je znázorněn na obrázku 11.1.



Obrázek 11.1 – Typická konfigurace maškarády

Máme malou ethernetovou síť používající jednu z rezervovaných síťových adres. Síť používá k přístupu na Internet linuxový směrovač s maškarádou. Jedna ze stanic na naší síti (192.168.1.3) chce navázat spojení se vzdáleným hostitelem 209.1.106.178 na portu 8888. Stanice směruje svůj požadavek na směrovač s maškarádou, který jej vyhodnotí jako požadavek, jejž je nutné ošetřit maškarádou. Přijme datagram, alokuje nějaký volný port (1035), uvede svou vlastní IP adresu a port z původního požadavku a pošle datagram cílovému hostiteli. Ten se domnívá, že přijal požadavek přímo od maškarádovacího počítače a vygeneruje odpověď. Maškarádovací počítač po přijetí odpovědi najde v tabulce maškarádových spojení příslušnou asociaci a provede opačnou substituci než u původního požadavku. Odpověď pak pošle původnímu odesilateli.

Lokální počítač se domnívá, že se baví přímo se vzdáleným hostitelem. Vzdálený hostitel pro změnu o lokálním hostiteli vůbec nic neví a domnívá se, že se baví přímo s maškarádovacím hostitelem. Pouze maškarádovací hostitel ví, kdo se s kým doopravdy baví a na jakých portech a zajišťuje překlady adres a portů potřebné pro průběh spojení.

Může to sice vypadat trošku komplikovaně a možná to i tak je, nicméně to funguje a celá služba se velmi snadno konfiguruje. Nelamte si proto hlavu s tím, pokud vám nějaké detaily unikají.

Vedlejší efekty, výhody a nevýhody

IP maškaráda s sebou přináší různé vedlejší efekty, z nichž některé jsou užitečné a jiné mohou být naopak nepříjemné.

Žádný z hostitelů na interní síti za maškarádovacím počítačem není přímo viditelný, tím pádem vám stačí pouze jediná platná a směrovatelná IP adresa a pomocí ní mohou s Internetem komunikovat všechny počítače na interní síti. Má to i nevýhodu – žádný z hostitelů na lokální síti není z Internetu viditelný a není tak možné se s nimi z Internetu spojit. Jediným viditelným počítačem

na maškarádované síti je samotný počítač maškarády. To je významné při provozu služeb jako je pošta nebo FTP. Musíte rozhodnout, jaké služby bude zajišťovat přímo maškarádovací počítač a které služby bude ošetřovat prostřednictvím proxy serveru nebo jiným speciálním způsobem.

Dále, protože žádný z maškarádovaných hostitelů není viditelný, jsou relativně chráněny před útoky zvenčí – tím se zjednoduší, případně úplně vyloučí nutnost konfigurace firewallu. Nicméně na to nelze příliš spoléhat. Celá síť bude zabezpečena jenom tak, jak je zabezpečen počítač maškarády, takže pokud vám na bezpečnosti záleží, i tak byste měli použít firewall.

Dále může mít maškaráda určitý dopad na výkon sítě. U typické maškarády to bude pravděpodobně stěží postřehnutelné. Pokud pracujete s velkým počtem aktivních maškarádovaných spojení, můžete zjistit, že zpracování datagramů na maškarádovacím počítači má dopad na propustnost sítě. V porovnání s klasickým směrováním musí maškaráda provést s každým jedním datagramem poměrně velký počet operací. Počítač 386SX16 pro maškarádu na telefonní lince bude zřejmě stačit, ovšem pokud jej budete chtít použít jako hlavní směrovač na firemní síti připojené vysokorychlostní linkou, zřejmě bude jeho výkon nedostatečný.

A konečně, některé síťové služby prostě nebudou s maškarádou fungovat, alespoň ne bez určité pomoci. Typicky se jedná o služby, které závisejí na navazování příchozích spojení, například některé typy přímých komunikačních kanálů, některé funkce v IRC nebo jisté typy video a audio služeb. Pro některé z těchto služeb existují speciální moduly jádra, které jejich provoz umožňují – budeme o nich za chvíli mluvit. U jiných zjistíte, že žádná podpora neexistuje, takže musíte být opatrní. Maškarádu nelze použít úplně vždy.

Konfigurace jádra pro maškarádu

Abyste mohli maškarádu použít, musíte mít jádro přeloženo s její podporou. Při konfiguraci jádra 2.2 musíte zvolit následující volby:

```
Networking options  --->
        [*] Network firewalls
        [*] TCP/IP networking
        [*] IP: firewalling
        [*] IP: masquerading
        --- Protocol-specific masquerading support will be built as modules.
        [*] IP: ipautofw masq support
        [*] IP: ICMP masquerading
```

Některé typy maškarády jsou dostupné pouze jako moduly jádra. Znamená to, že při sestavování jádra musíte kromě klasického „make zImage“ zadat i příkaz „make modules“.

Jádra série 2.4 neposkytují podporu maškarády jako parametr při sestavování jádra. Místo toho musíte zvolit podporu filtrace paketů:

```
Networking options  --->
    [M] Network packet filtering (replaces ipchains)
```

V jádrech série 2.2 existuje řada pomocných modulů, které se vytvářejí při sestavování jádra. Některé protokoly začínají komunikaci odchozím spojením na určitém portu a pak očekávají příchozí spojení na jiném. Za normálních okolností nelze takové protokoly pod maškarádou použít, protože neexistuje metoda, jak bez podrobné analýzy samotného obsahu datagramů asociovat druhý požadavek s prvním. Přesně to dělají podpůrné moduly – prohlédnou si tělo datagramu a umožňují maškarádu i pro protokoly, u nichž by jinak nebyla možná. Podporovány jsou následující protokoly:

Modul	Protokol
ip_masq_ftp	FTP
ip_masq_irc	IRC
ip_masq_raudio	RealAudio
ip_masq_cuseeme	CU-See-Me
ip_masq_vdolive	VDO Live
ip_masq_quake	IdSoftware Quake

K zavedení podpory zmíněných protokolů musíte uvedené moduly nahrát ručně příkazem **insmod**. Tyto moduly nelze nahrát démonem **kerneld**. Každý z modulů používá jako parametr číslo portu, na kterém má poslouchat. Pro modul protokolu RealAudio™ můžete zadat[89]:

```
# insmod ip_masq_raudio.o ports=7070,7071,7072
```

Čísla zadávaných portů závisejí na konkrétním protokolu. Další podrobnosti o modulech maškarády a o jejich použití nabízí dokument IP masquerade mini-HOWTO Ambrose Au[90].

Balík *netfilter* obsahuje moduly, které plní podobnou funkci. Pokud budete například chtít zajistit podporu spojení FTP relací, použijete moduly `ip_conntrack_ftp` a `ip_nat_ftp.o`.

Konfigurace maškarády

Pokud jste četli kapitoly o firewallech a účtování, zřejmě vás nepřekvapí, že i maškaráda se nastavuje příkazy **ipfwadm**, **ipchains** a **iptables**.

Pravidla maškarády představují velmi speciální třídu filtračních pravidel. Maškarádě mohou podléhat pouze datagramy přicházející na jednom rozhraní a odcházející jiným rozhraním. Pravidla maškarády se velmi podobají předávacím pravidlům firewallu, obsahují však speciální parametr, který jádru říká, že datagram má projít maškarádou. Příkaz **ipfwadm** používá parametr –m, **ipchains** pak –j MASQ a **iptables** –j MASQUERADE, které definují, že na datagram má být uplatněna maškaráda.

Podívejme se na příklad. Student katedry počítačů na univerzitě Groucho Marx má doma několik počítačů propojených ethernetovou sítí. Pro tuto síť používá IP adresy z jedné z rezervovaných oblastí. Bydlí společně s dalšími studenty a všichni chtějí používat Internet. Protože studenti nikdy nebyli při penězích, nemohou si dovolit pevné připojení k Internetu a používají místo toho vytáčenou PPP linku. Všichni chtějí na svých počítačích používat služby jako IRC, WWW a FTP – řešením je maškaráda.

Nejprve musí nakonfigurovat linuxový počítač pro podporu vytáčené linky a jako směrovač pro lokální síť. IP adresa, kterou počítač dostane v okamžiku připojení, není důležitá. Směrovač je nastaven s maškarádou a pro lokální síť používá IP adresy z rezervované oblasti 192.168.1.0. Dále je nutné zajistit, aby všechny počítače na lokální síti měly nastavenu implicitní trasu směřující na tento směrovač.

Následující příkazy **ipfwadm** stačí k zavedení maškarády v této konfiguraci sítě:

89 RealAudio je ochranná známka společnosti Progressive Networks Corporation.

90 Ambrose můžete kontaktovat na adrese *ambrose@writeme.com*.

```
# ipfwadm -F -p deny
# ipfwadm -F -a accept -m -S 192.168.1.0/24 -D 0/0
```

nebo s **ipchains**

```
# ipchains -P forward -j deny
# ipchains -A forward -s 192.168.1.0/24 -d 0/0 -j MASQ
```

anebo s **iptables**

```
# iptables -t nat -P POSTROUTING DROP
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Kdykoliv se nyní některý z počítačů na lokální síti pokusí o spojení se službou na nějakém vzdáleném systému, odesílané datagramy budou zpracovány maškarádou. První pravidlo vždy zakazuje přímé směrování jakýchkoliv jiných datagramů a poněkud tak zvyšuje bezpečnost konfigurace.

K vypsání vytvořených pravidel maškarády můžete použít parametr `-l` příkazu **ipfwadm**, o kterém jsme hovořili v kapitole věnované firewallům.

K vypsání výše vytvořených pravidel bychom mohli použít příkaz:

```
# ipfwadm -F -l -e
```

Měli bychom dostat asi takovýto výsledek:

```
# ipfwadm -F -l -e
IP firewall forward rules, default policy: deny
 pkts bytes type  prot opt  tosa tosx ifname  ifaddress  …
    0     0 acc/m all  ---- 0xFF 0x00 any     any        …
```

Parametr „`/m`“ ve výpisu říká, že se jedná o pravidlo maškarády.

K výpisu pravidel maškarády v programu **ipchains** použijete parametr `-L`. Při zadání výše vytvořených pravidel dostaneme něco takového:

```
# ipchains -L
Chain input (policy ACCEPT):
Chain forward (policy DENY):
target     prot opt     source               destination          ports
MASQ       all  ------  192.168.1.0/24       anywhere             n/a

Chain output (policy ACCEPT):
```

Jakákoliv pravidla vztahující se k maškarádě jsou uvedena s operací MASQ.

Konečně s příkazem iptables použijeme:

```
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy DROP)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere           MASQUERADE

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

I zde jsou pravidla vztahující se k maškarádě uvedena operací `MASQUERADE`.

Nastavení časovacích parametrů IP maškarády

Při navázání každého spojení si software implementující maškarádu vytvoří v paměti asociaci mezi počítači, pro něž bylo spojení vytvořeno. Tyto asociace můžete v kterémkoliv okamžiku vidět v souboru `/proc/net/ip_masquerade`. Po určité době nečinnosti budou asociace zrušeny.

Časový limit asociace je možné nastavit příkazem **ipfwadm**. Obecná syntaxe je:

```
ipfwadm -M -s <tcp> <tcpfin> <udp>
```

S příkazem **ipchains** je syntaxe:

```
ipchains -M -S <tcp> <tcpfin> <udp>
```

Implementace v programu **iptables** používá podstatně delší časové limity a nedovoluje je změnit.

Jednotlivé hodnoty jsou časovače s nastavením v sekundách. Jejich význam uvádí následující tabulka:

Parametr	Popis
`tcp`	Timeout TCP relace. Jak dlouho může být TCP spojení neaktivní, než dojde k odstranění asociace.
`tcpfin`	Timeout TCP po příznaku FIN. Jak dlouho zůstane asociace zachována po ukončení TCP spojení.
`udp`	Timeout UDP relace. Jak dlouho může být UDP spojení neaktivní, než dojde k odstranění asociace.

Obsluha dotazů na jmenné servery

Obsluha dotazů na jmenné servery od počítačů na síti s maškarádou je vždy problematická. V tomto prostředí přicházejí v úvahu dvě řešení. Jedna možnost je nakonfigurovat všechny počítače tak, aby používaly stejný DNS server jako počítač s maškarádou a pak nechat maškarádu, ať se postará o zbytek. Druhá možnost je na nějakém počítači rozběhnout jmenný server v konfiguraci caching-only a nechat všechny stanice používat tento jmenný server. I když je to poněkud složitější řešení, bude pravděpodobně výhodnější, protože se tak redukuje DNS provoz na internetové lince a pro většinu dotazů přijde odpověď rychleji, protože budou obslouženy z vyrovnávací paměti lokálního serveru. Nevýhodou je pouze to, že celá konfigurace je poněkud složitější. Konfiguraci jmenného serveru popisujeme v kapitole 6, v části *Konfigurace typu „caching-only“*.

Další informace o překladu síťových adres

Software *netfilter* umožňuje řadu různých řešení překladu síťových adres, IP maškaráda je pouze jedním z nich.

Je například možné vytvořit taková pravidla překladu, že se budou překládat pouze určité adresy nebo rozsahy adres a ostatní zůstanou beze změny, nebo překládat všechny adresy na skupinu adres a ne pouze na jedinou tak, jak to dělá maškaráda. Pomocí příkazu **iptables** můžete vytvořit překladová pravidla, která budou mapovat cokoliv na cokoliv, s využitím kombinací všech standardních atributů, například zdrojových adres, cílových adres, typů protokolů, čísel portů a podobně.

O překladu zdrojové adresy datagramu se v dokumentaci k balíku netfilter hovoří jako o „Source NAT“ nebo `SNAT`. Překlad cílové adresy se označuje jako „Destination NAT“, `DNAT`. Překlady TCP a UDP portů se označují termínem `REDIRECT`. Výrazy `SNAT`, `DNAT` a `REDIRECT` představují operace, které můžete v pravidlech programu **iptables** definovat příznakem `-j` a vytvářet tak různá složitá a kombinovaná pravidla.

Téma překladu adres a jeho použití by dokázalo zabrat celou samostatnou kapitolu. Bohužel zde nemáme prostor, abychom se tomuto tématu mohli věnovat podrobněji. Další informace naleznete v dokumentu IPTABLES-HOWTO, kde se mimo jiné hovoří i o problematice překladu síťových adres.

91 Ne-li dokonce celou knihu.

Důležité síťové aplikace

Po úspěšném nastavení protokolu IP a resolveru musíte začít věnovat pozornost službám, které hodláte na síti poskytovat. Tato kapitola se zabývá konfiguracemi několika jednoduchých síťových aplikací, včetně serveru **inetd** a programů z rodiny **rlogin**. Krátce se také zmíníme o rozhraní Remote Procedure Call, na kterém jsou založeny služby Network File System (NFS) a Network Information System (NIS). Konfigurace systémů NFS a NIS je nicméně o dost složitější, takže o ní budeme hovořit v samostatných kapitolách, stejně jako o elektronické poště a o síťových news.

Samozřejmě v této knize nemůžeme probrat všechny síťové aplikace. Budete-li chtít nainstalovat nějakou z aplikací, která zde není popsána, například programy **talk**, **gopher** nebo **http**, můžete příslušné informace nalézt na odpovídajících manuálových stránkách.

Superserver inetd

Programy, které poskytují služby po síti, se označují jako *démony*. Démon je program, který otevře určitý port, typicky známý port určité služby, a čeká na příchozí spojení. Pokud k takovému spojení dojde, vytvoří proces potomka, jenž toto spojení obslouží a rodičovský proces bude stále pokračovat v odposlouchávání dalších požadavků. Toto řešení sice funguje dobře, má ale několik nevýhod. V paměti musí být trvale přítomna alespoň jedna instance každého démona pro každou provozovanou službu. Kromě toho se v každém démonu opakuje ta část kódu, která zajišťuje obsluhu a poslouchání na portu.

Řešením této neefektivity je na většině unixových systémů speciální síťový démon, který můžeme chápat jako superserver. Tento démon otevírá sokety pro řadu síťových služeb a na všech poslouchá. Když se na některém z portů objeví příchozí spojení, superserver je přijme a spustí server pro konkrétní port, kterému pak předá soket k obsluze. Superserver pak pokračuje v poslouchání na portech[92].

Nejběžnější superserver se jmenuje **inetd**, Internet Daemon. Je spouštěn při zavádění systému a seznam služeb, které má spravovat, získává z konfiguračního souboru `/etc/inetd.conf`. Kromě výše zmíněných volaných serverů existuje i řada triviálních služeb, které provádí démon **inetd** sám. Tyto služby se nazývají *interní služby*. Jednou z nich jsou služba **chargen**, která generuje řetězec znaků nebo služba **daytime**, jež vrací systémový čas.

92 Pozn. překladatele: Toto řešení má však nevýhodu v režii spojené se spuštěním serveru služby démonem **inted**. Proto se u hodně zatížených služeb, kde se požaduje co nejrychlejší odezva, dává přednost tomu, že běží přímo server dané služby a nepoužívá se pro ni nepřímé spouštění superserverem. Typicky tak trvale běží démon **httpd**, server služby WWW, jehož spouštění prostřednictvím superdémona by bylo pomalé.

Záznamy v konfiguračním souboru jsou uloženy na samostatných řádcích ve formátu:

```
služba typ protokol čekání uživatel server příkazový_řádek
```

Jednotlivá pole mají následující význam:

služba — Určuje název služby. Název služby musí být převoditelný na číslo portu prostřednictvím souboru `/etc/services`. Tento soubor bude popsán později v části *Soubory services a protocols*.

typ — Určuje typ soketu. Může mít hodnotu stream (u protokolů s navazováním spojení) nebo dgram (pro datagramové služby). Služby založené na protokolu TCP proto vždy používají hodnotu stream, služby založené na protokolu UDP dgram.

protokol — Určuje název přenosového protokolu, který bude daná služba používat. Musí to být korektní název protokolu, jenž se nachází v souboru protocols, o kterém budeme hovořit později.

čekání — Tato volba se vztahuje pouze na sokety typu dgram. Může mít hodnotu wait nebo nowait. Je-li zadána volba wait, spustí démon **inetd** pro daný port v jednom okamžiku vždy pouze jeden server. V opačném případě bude po spuštění serveru okamžitě pokračovat v odposlouchávání portu. Tato nastavení se liší podle typu serveru. Jednovláknové servery, které přečtou příchozí datagramy, odpoví na ně a ukončí se, by měly být spouštěny jako wait. Patří sem většina RPC služeb. Opačným typem jsou vícevláknové servery, které umožňují současný běh více instancí serveru. U těchto serverů se používá hodnota nowait. Sokety typu stream by měly vždy používat volbu nowait.

uživatel — Tato volba určuje identifikační číslo uživatele, pod kterým bude daný proces spouštěn. Tímto uživatelem je často uživatel **root**, avšak některé služby mohou používat i jiné účty. Zde je velmi vhodné uplatňovat princip nejnižších práv, což znamená, že byste neměli příkaz spouštět na účtu s vyššími právy, pokud je spouštěný program pro svou správnou funkci nevyžaduje. Například server news NNTP běží s právy uživatele **news**, zatímco služby, které by mohly představovat bezpečnostní rizika (jako je služba **tftp** nebo **finger**), jsou často spouštěny s právy uživatele **nobody**.

server — Určuje název plné cesty ke spouštěnému programu serveru. Vnitřní služby jsou označeny klíčovým slovem internal.

příkazový_řádek — Tato volba určuje příkazovou řádku, která bude předána danému serveru. Začíná názvem spouštěného serveru a za ním mohou následovat předávané parametry. Pokud chcete použít TCP wrapper, musíte zde specifikovat úplnou cestu k serveru. Jinak zadáváte pouze název serveru tak, jak se objevuje v seznamu procesů. O TCP wrapperech budeme zanedlouho hovořit. Toto pole je u interních služeb prázdné.

Vzorový soubor `/etc/inetd.conf` je uveden v příkladu 12.1. Služba **finger** je zakomentována a není tedy dostupná. Tato služba se často z bezpečnostních důvodů vypíná, protože útočníkům umožňuje zjistit uživatelská jména a další informace o uživatelích vašeho systému.

Příklad 12.1 – Příklad souboru /etc/inetd.conf

```
#
# služby démona inetd
ftp      stream tcp nowait root  /usr/sbin/ftpd    in.ftpd -l
telnet   stream tcp nowait root  /usr/sbin/telnetd in.telnetd -b/etc/issue
#finger  stream tcp nowait bin   /usr/sbin/fingerd in.fingerd
#tftp    dgram  udp wait  nobody /usr/sbin/tftpd   in.tftpd
#tftp    dgram  udp wait  nobody /usr/sbin/tftpd   in.tftpd /boot/diskless
#login   stream tcp nowait root  /usr/sbin/rlogind in.rlogind
#shell   stream tcp nowait root  /usr/sbin/rshd    in.rshd
#exec    stream tcp nowait root  /usr/sbin/rexecd  in.rexecd
#
# interní služby
#
daytime  stream tcp nowait root internal
daytime  dgram  udp nowait root internal
time     stream tcp nowait root internal
time     dgram  udp nowait root internal
echo     stream tcp nowait root internal
echo     dgram  udp nowait root internal
discard  stream tcp nowait root internal
discard  dgram  udp nowait root internal
chargen  stream tcp nowait root internal
chargen  dgram  udp nowait root internal
```

Služba **tftp** je rovněž zakomentována. Tato služba implementuje protokol *Trivial File Transfer Protocol*, který umožňuje z vašeho systému přenést libovolný veřejně čitelný soubor, aniž by byla prováděna jakákoliv kontrola pomocí hesla a podobně. To je nepříjemné zejména u souboru `/etc/passwd`, zvlášť v případě, že nepoužíváte stínový soubor hesel.

Protokol TFTP obecně používají bezdiskoví klienti a X terminály ke stahování startovacího kódu ze zaváděcích serverů. Pokud potřebujete spouštět službu **tftpd** z tohoto důvodu, ujistěte se, že jste omezili její rozsah pouze na ty adresáře, ze kterých budou klienti získávat soubory. To provedete tak, že na příkazovém řádku démona **tftpd** uvedete dostupné adresáře. Vidíme to na druhém řádku výpisu.

Řízení přístupu wrapperem tcpd

Protože přístup k počítačům prostřednictvím sítě přináší mnoho bezpečnostních rizik, síťové aplikace jsou proti různým typům útoků odolné. Některé aplikace ovšem mohou obsahovat chyby (nejdrastičtěji to demonstroval internetový červ RTM) nebo nemusí umět rozlišovat mezi bezpečnými hostiteli, od kterých budou přijímány požadavky na konkrétní službu, a nebezpečnými hostiteli, jejichž požadavky by měly zamítnout. Již jsme se krátce zmínili o službách **finger** a **tftp**. Tyto služby budete typicky chtít povolit pouze „důvěryhodným hostitelům“, což ale nelze pomocí standardního nastavení, při kterém superserver **inetd** poskytuje konkrétní služby buď všem klientům, nebo nikomu.

Pro řízení přístupu podle hostitele se často používá démon **tcpd**, takzvaný *wrapper* (česky asi „obal“)[93]. Tento démon se volá u chráněných nebo sledovaných služeb namísto příslušného serveru služby. Démon zkontroluje, zda požadavek přichází od oprávněného hostitele a pouze v ta-

93 Napsal jej Wietse Venema, *wietse@wzv.win.tue.nl*.

kovém případě pak spustí skutečný server služby. Tento postup nelze použít u služeb založených na protokolu UDP.

Chcete-li například „obalit" démona **finger**, musíte v souboru `inetd.conf` změnit odpovídající řádek následujícím způsobem z:

```
# démon finger bez ochrany
finger    stream tcp nowait bin    /usr/sbin/fingerd in.fingerd
```

na:

```
# démon finger s ochranou
finger  stream  tcp     nowait  root    /usr/sbin/tcpd   in.fingerd
```

Pokud nepřidáte žádné řízení přístupu, bude se tato úprava klientovi jevit stejně, jako obvyklé nastavení služby **finger**, s tou výjimkou, že veškeré požadavky budou zapisovány službou **syslog**.

Řízení přístupu je implementováno za pomoci souborů `/etc/hosts.allow` a `/etc/hosts.deny`. Tyto soubory obsahují položky, které povolují a zakazují přístup k určitým službám podle žádajícího hostitele. Když nástroj **tcpd** vyřizuje požadavek na službu **finger** od klienta **biff.foobar.com**, hledá v souborech hosts.allow a hosts.deny (v tomto pořadí) položku odpovídající jak požadované službě, tak i hostiteli klienta. Pokud je odpovídající položka nalezena v souboru hosts.allow, bude přístup povolen a soubor hosts.deny se už neprohlíží. Pokud bude odpovídající položka nalezena v souboru hosts.deny, bude požadavek odmítnut a spojení se ukončí. Jestliže se ani v jednom souboru odpovídající položka nenajde, bude požadavek přijat.

Položky v souborech pro řízení přístupu vypadají následovně:

`seznam_služeb``: `*`seznam_hostitelů`* `[:`*`příkazy_shellu`*`]`

Pole *seznam_služeb* obsahuje seznam názvů služeb podle souboru `/etc/services` nebo klíčové slovo ALL. Chcete-li zadat všechny služby kromě služeb **finger** a **tftp**, můžete zapsat ALL EXCEPT finger, tftp.

Pole *seznam_hostitelů* obsahuje seznam názvů hostitelů nebo jejich IP adres, případně klíčová slova ALL, LOCAL, UNKNOWN nebo PARANOID. Klíčovému slovu ALL vyhovují všichni hostitelé, klíčovému slovu LOCAL vyhovují pouze ty názvy hostitelů, které neobsahují tečku[94], UNKNOWN znamená hostitele, u nichž se nepodařil převod názvu nebo adresy a klíčovému slovu PARANOID hostitelé, u nichž nesouhlasí zpětný převod IP adresy na název[95]. Název začínající tečkou znamená všechny hostitele, jejichž doména je shodná s uvedeným názvem. Například názvu **.foobar.com** bude vyhovovat třeba hostitel **biff.foobar.com**, ne však hostitel **nurks.fredsville.com**. Zadání končící tečkou znamená všechy hostitele, kteří začínají danou IP adresou, takže například zadání **172.16.** bude vyhovovat hostitel `172.16.32.1`, ne však hostitel `172.15.9.1`. Hodnota ve tvaru *n.n.n.n/m.m.m.m* je chápána jako IP adresa se síťovou maskou, takže předchozí příklad bychom mohli definovat také jako `172.16.0.0/255.255.0.0`. Konečně údaj začínající lomítkem specifikuje název souboru, v němž jsou uvedena jména nebo adresy vyhovujících hostitelů. Takže údaj */var/access/trustedhosts* způsobí, že program **tcpd** načte daný soubor a prohlédne jej, zda některý ze záznamů v souboru nevyhovuje právě připojenému hostiteli.

---

94 Přičemž název bez tečky mají typicky pouze lokální hostitelé uvedení v souboru `/etc/hosts`.

95 I když samo klíčové slovo implikuje poněkud extrémní význam, volba PARANOID je vhodná standardní volba, protože chrání před zákeřnými hostiteli, kteří se vydávají za někoho jiného. Ne všechny verze programu **tcpd** tuto volbu podporují, pokud ji vaše verze neumí, musíte si přeložit novější verzi.

Chcete-li zakázat přístup ke službám **finger** a **tftp** všem hostitelům s výjimkou lokálních hostitelů, vložte do souboru `/etc/hosts.deny` následující řádek a soubor `/etc/hosts.allow` nechte prázdný:

```
in.tftpd, in.fingerd:  ALL EXCEPT LOCAL, .vaše.doména
```

Nepovinný údaj *příkazy_shellu* může obsahovat příkaz, jenž bude vykonán při splnění dané položky. Je to užitečné k nastavení „pastí“, které mohou odhalit potenciální útočníky:

```
in.ftpd: ALL EXCEPT LOCAL, .vbrew.com : \
      echo "request from %d@%h: >> /var/log/finger.log; \
      if [ %h != "vlager.vbrew.com:" ]; then \
          finger -l @%h >> /var/log/finger.log \
      fi
```

Nástroj tcpd nahradí parametry %h a %d skutečným názvem klienta a skutečným názvem služby. Další podrobnosti naleznete na manuálových stránkách hosts_access(5).

Soubory services a protocols

Čísla portů, na kterých jsou nabízeny „standardní“ služby, jsou definována v RFC Assigned Numbers. Aby mohly servery nebo klienti převádět názvy služeb na tato čísla, musí být alespoň část z tohoto seznamu uložena na každém hostiteli; ukládá se v souboru `/etc/services`. Položky v souboru mají následující syntaxi:

služba port/protokol [*aliasy*]

Pole *služba* definuje název služby, *port* definuje port, na kterém je služba nabízena, a pole *protokol* definuje typ používaného transportního protokolu. Tento údaj je obecně buď tcp nebo udp. Stejnou službu je možné nabízet oběma protokoly, naopak lze na stejném portu různých protokolů nabízet různé služby. Pole *aliasy* umožňuje zadat alternativní názvy stejné služby.

Soubor services, který je dodáván společně se síťovým softwarem, nebudete muset obvykle měnit. Přesto zde uvádíme malý výpis z tohoto souboru:

Příklad 12.2 – Příklad souboru /etc/services

```
# Soubor služeb:
#
# známé služby
echo           7/tcp                # Echo
echo           7/udp                #
discard        9/tcp  sink null      # Discard
discard        9/udp  sink null      #
daytime       13/tcp                # Daytime
daytime       13/udp                #
chargen       19/tcp  ttytst source  # Character Generator
chargen       19/udp  ttytst source  #
ftp-data      20/tcp                # File Transfer Protocol (Data)
ftp           21/tcp                # File Transfer Protocol (Control)
telnet        23/tcp                # Virtual Terminal Protocol
smtp          25/tcp                # Simple Mail Transfer Protocol
nntp         119/tcp  readnews       # Network News Transfer Protocol
#
# unixové služby
```

```
exec          512/tcp                  # BSD rexecd
biff          512/udp   comsat          # notifikace pošty
login         513/tcp                  # vzdálený login
who           513/udp   whod           # vzdálené who a uptime
shell         514/tcp   cmd            # vzdálený příkaz, bez hesla
syslog        514/udp                  # vzdálené logování
printer       515/tcp   spooler        # vzdálený tiskový spooling
route         520/udp   router routed  # směrovací protokol RIP
```

Všimněte si, že například služba **echo** je poskytována na portu 7 jak protokolem TCP, tak i protokolem UDP, zatímco port 512 používají dvě odlišné služby: vzdálené spouštění (**rexec**) pomocí protokolu TCP a démon **COMSAT** (zajišťující oznamování došlé pošty, viz *xbiff(1x)*) protokolem UDP.

Podobně jako u názvů služeb potřebuje síťová knihovna i nějaký způsob, jakým by mohla přeložit názvy protokolů – například těch protokolů, které jsou použity v souboru services – na čísla protokolů, jimž by rozuměly IP vrstvy ostatních hostitelů. To se provede vyhledáním daného názvu v souboru `/etc/protocols`. Ten obsahuje na každém řádku jednu položku obsahující název protokolu a přidělené číslo. Provádění změn v tomto souboru je ještě méně pravděpodobné, než zasahování do souboru `/etc/services`. Příklad tohoto souboru vidíte na výpisu 12.3.

Příklad 12.3 – Příklad souboru /etc/protocols

```
#
# Internetové protokoly (IP)
#
ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # internet group multicast protocol
tcp     6       TCP             # transmission control protocol
udp     17      UDP             # user datagram protocol
raw     255     RAW             # RAW IP interface
```

Vzdálené volání procedur

Balík RPC (*Remote Procedure Call*) poskytuje obecný mechanismus pro aplikace typu klient-server. Balík RPC vyvinula firma Sun Microsystems a jde o sbírku nástrojů a knihoven funkcí. Důležitou aplikací postavenou na balíku RPC je systém NIS (Network Information System, viz kapitola 13) a NFS (Network File System, viz kapitola 14).

Server RPC se skládá ze sady procedur, které si může klient volat tím způsobem, že pošle serveru RPC požadavek společně s parametry procedury. Server spustí místo klienta požadovanou proceduru a existuje-li návratová hodnota, pošle ji zpět klientovi. Aby mohl být balík RPC strojově nezávislý, všechna data předávaná mezi klientem a serverem se konvertují do formátu *External Data Representation* (XDR). K přenosu dat ve formátu XDR používá knihovna RPC standardní TCP a UDP sokety. Firma Sun uvolnila balík RPC jako Public Domain, celý balík je popsán v kolekci několika RFC dokumentů.

Někdy způsobí zdokonalení RPC aplikace nekompatibilní změny v rozhraní volání procedur. Samozřejmě že prostá výměna serveru může způsobit havárii všech aplikací, které stále očekávají původní chování. Proto mají programy RPC přiděleny číslo své verze, obvykle se začíná hodnotou 1 a při každé nové verzi rozhraní RPC je tato hodnota zvýšena. Server může často současně nabízet několik verzí RPC; klient ve svém požadavku specifikuje číslo verze, kterou chce použít.

Vlastní síťová komunikace probíhající mezi servery RPC a jejich klienty je zvláštní. Server RPC nabízí jednu nebo více skupin procedur; každá množina procedur se nazývá *program* a je jednoznačně určena takzvaným *číslem programu*. Seznam definující přiřazení názvů služeb k číslům programů je obvykle uložen v souboru `/etc/rpc`, jehož část ukazuje následující výpis.

Příklad 12.4 – Příklad souboru /etc/rpc

```
#
# /etc/rpc - různé služby založené na RPC
#
portmapper  100000    portmap sunrpc
rstatd      100001    rstat rstat_svc rup perfmeter
rusersd     100002    rusers
nfs         100003    nfsprog
ypserv      100004    ypprog
mountd      100005    mount showmount
ypbind      100007
walld       100008    rwall shutdown
yppasswss   100009    yppasswd
bootparam   100026
ypupdated   100028    ypupdate
```

U sítí na bázi protokolu TCP/IP čelili autoři balíku RPC problému, jak sdružit čísla programů s obecnými síťovými službami. Zvolili takovou variantu, kdy každý server poskytuje pro každý program a pro každou verzi programu jak port pro protokol TCP, tak i port pro protokol UDP. Obecně budou aplikace při posílání dat používat protokol UDP a protokol TCP budou používat pouze v případě, že se posílaná data nevejdou do jediného datagramu protokolu UDP.

Samozřejmě že klienti musí mít možnost zjistit port, na který je namapováno dané číslo programu. V tomto případě by bylo použití konfiguračního souboru příliš nepružné, jelikož aplikace RPC nepoužívají vyhrazené porty; nemůžeme mít tedy žádnou jistotu, že port původně určený pro použití naší databázovou aplikací nebude obsazen nějakým jiným procesem. Proto aplikace RPC vyberou některý z dostupných portů a zaregistrují ho pomocí speciálního démona, takzvaného *mapovače portů* (*portmapper*). Tento démon se chová jako zprostředkovatel mezi všemi servery RPC, které jsou spuštěny na daném počítači: klient, který chce kontaktovat službu s daným číslem programu, se nejdříve bude dotazovat mapovače portů na hostiteli serveru, ten mu pak vrátí čísla portů TCP a UDP, na kterých je daná služba dosažitelná.

Tato metoda má konkrétní nevýhodu v tom, že zavádí jedno slabé místo celého systému, podobně jako to činí démon **inetd** u standardních Berkeley služeb. Avšak tento případ je ještě horší, protože když selže démon mapující porty, ztratí se veškeré informace o portech RPC; to obvykle způsobí, že budete muset manuálně znovu nastartovat všechny servery RPC nebo dokonce celý počítač.

V Linuxu se program na mapování portů nazývá `/sbin/portmap` nebo `/usr/sbin/ rpc.portmap`. Kromě toho, že je třeba zajistit, aby se mapovač automaticky spouštěl při startu systému, není třeba žádná další konfigurace.

Konfigurace vzdáleného přihlašování a spouštění

Často je velmi užitečné spustit příkaz na vzdáleném počítači a vstupy a výstupy tohoto programu zapisovat nebo číst prostřednictvím síťového spojení.

Tradičně se pro spouštění programů na vzdáleném hostiteli používaly příkazy **rlogin**, **rsh** a **rcp**. Příklad příkazu **rlogin** jsme viděli v kapitole 1 v části pojmenované *Úvod do sítí TCP/IP*. O bezpečnostních problémech souvisejících s tímto příkazem jsme hovořili v téže kapitole v části *Bezpečnost systému*, kde jsme rovněž doporučili nahradit tento příkaz příkazem **ssh**. Balík **ssh** obsahuje náhrady zmíněných programů, programy **slogin**, **ssh** a **scp**.

Všechny tyto příkazy vyvolají na vzdáleném hostiteli příkazový interpret a umožní uživateli spouštět programy. Samozřejmě že klient musí mít na hostiteli, na kterém hodlá spouštět příkazy, založený účet. Všechny tyto příkazy totiž provádějí ověření totožnosti. *r*-příkazy ověřují totožnost prostou výměnou jména a hesla bez šifrování, takže kdokoliv, kdo odposlouchává síťový provoz, může heslo zjistit. Rodina příkazů **ssh** používá vyšší úroveň zabezpečení: využívá šifrování s veřejným klíčem, při němž je možné provést autentikaci, aniž by po síti byla přenášena jakákoliv snadno zjistitelná data.

V některých případech je možné autentikaci ještě více potlačit. Pokud se například často přihlašujete k jiným počítačům na vaší lokální síti, asi nebudete chtít při každém přihlášení znovu zapisovat heslo. Takové řešení bylo možné už i u *r*-příkazů, balík **ssh** je ještě více zjednodušuje. Nicméně z bezpečnostního hlediska se stále nejedná o příliš vhodné řešení, protože jakmile dojde k narušení jednoho počítače na síti, bude útočník moci získat přístup i k účtům na všech ostatních počítačích. Jedná se však o řešení oblíbené a uživateli hojně používané.

Zaměříme se nyní na to, jak odstranit příkazy rodiny *r* a jak pracovat s balíkem **ssh**.

Vypnutí r-příkazů

Začneme tím, že příkazy rodiny *r*, pokud jsou nainstalovány, odstraníme. Nejjednodušší způsob jak tyto příkazy vypnout spočívá v zakomentování (nebo smazání) příslušných řádků v souboru `/etc/inetd.conf`. Odpovídající řádky budou vypadat asi takto:

```
# Shell, login, exec a talk jsou BSD protokoly.
shell    stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rshd
login    stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rlogind
exec     stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rexecd
```

Řádky můžete zakomentovat tak, že na jejich začátku zapíšete #, nebo je můžete úplně smazat. Aby se změny projevily, musíte restartovat démona **inetd**. Ideálně bude rozumné odpovídající programy také smazat.

Instalace a konfigurace ssh

OpenSSH je volně distribuovaná verze rodiny ssh programů, mutace pro Linux je dostupná na adrese *http://violet.ibs.com.au/openssb* a je součástí většiny moderních distribucí Linuxu[96]. Nebudeme zde vysvětlovat, jak balík přeložit; dobrý popis je přímo součástí distribuce. Pokud se vám podaří získat již přeloženou verzi, s výhodou ji můžete nainstalovat.

Relace programu **ssh** zahrnuje dvě strany. Jednou z nich je klient **ssh**, který musí být nakonfigurován a spuštěn na lokálním počítači, druhým je démon **ssh**, který běží na vzdáleném hostiteli.

Démon ssh

Démon **sshd** je program, který naslouchá síťovým spojením od klientů **ssh**, zajišťuje autentikaci a spouští požadované příkazy. Má jeden hlavní konfigurační soubor `/etc/ssh/sshd_config`

96 Balík OpenSSH byl vyvinut v rámci projektu OpenBSD a je pěkným příkladem výborného zdarma dostupného softwaru.

a dále speciální soubor obsahující klíč používaný pro autentikaci a šifrování přenášených dat. Každý server a každý klient má svůj vlastní klíč.

Náhodný klíč vygeneruje nástroj **ssh-keygen**. Typicky se použije jednou v době instalace k vygenerování klíče, který se pak uloží do souboru `/etc/ssh/ssh_host_key`. Klíče mohou mít délku 512 bitů a větší. Standardně generuje program **ssh-keygen** klíč o délce 1 024 bitů a toto nastavení většina uživatelů používá. Náhodný klíč vygenerujete následujícím příkazem:

```
# ssh-keygen -f /etc/ssh/ssh_host_key
```

Budete požádáni o zadání hesla. Klíč na serveru však nesmí být chráněn heslem, takže jen stiskněte Enter a heslo nezadávejte. Výstup programu bude vypadat nějak takto:

```
Generating RSA keys:  ......ooooo0.............................ooooo0
Key generation complete.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_key
Your public key has been saved in /etc/ssh/ssh_host_key.pub
The key fingerprint is:
1024 3a:14:78:8e:5a:a3:6b:bc:b0:69:10:23:b7:d8:56:82 root@moria
```

Zjistíte, že byly vygenerovány dva soubory. Jeden z nich obsahuje privátní klíč, který musíte udržovat v tajnosti a jenž je uložen v souboru `/etc/ssh/ssh_host_key`. Druhý je takzvaný veřejný klíč, který sdílíte s ostatními. Ten se nachází v souboru `/etc/ssh/ssh_host_key.pub`.

Jakmile máte vytvořeny klíče, můžete se pustit do nastavení konfiguračního souboru. Balík **ssh** je velmi mocný a konfigurační soubor proto obsahuje spoustu voleb. Uvedeme si jednoduchý příklad, od kterého můžete začít, o dalších možnostech a podrobnostech se dozvíte v dokumentaci k balíku **ssh**. Následující výpis představuje bezpečný a minimální konfigurační soubor démona **sshd**. Další konfigurační možnosti jsou podrobně popsány na manuálové stránce **sshd(8)**.

```
# /etc/ssh/sshd_config
#

# IP adresa, na níž posloucháme na spojení. 0.0.0.0, znamená všechny
# lokální adresy.
ListenAddress 0.0.0.0

# TCP port, na kterém posloucháme. Standard je 22.
Port 22

# Název souboru s privátním klíčem.
HostKey /etc/ssh/ssh_host_key

# Délka klíče v bitech.
ServerKeyBits 1024

# Může se přes ssh přihlásit root?
PermitRootLogin no

# Má démon ssh před povolením přihlášení ověřit, zda jsou domovský
# adresář uživatele a jeho přístupová práva bezpečná?
StrictModes yes
```

```
# Povolujeme starou autentikační metodu souborů ~/.rhosts a
# /etc/hosts.equiv?
RhostsAuthentication no
# Povolujeme autentikaci mechanismem RSA?
RSAAuthentication yes
# Povolujeme autentikaci heslem?
PasswordAuthentication yes

# Povolujeme /etc/hosts.equiv v kombinaci s RSA autentikací
# hostitele?
RhostsRSAAuthentication no
# Ignorujeme soubory ~/.rhosts?
IgnoreRhosts yes
# Povolujeme přihlášení k účtům s prázdnými hesly?
PermitEmptyPasswords no
```

K zajištění bezpečnosti systému je nutné správně nastavit přístupová práva ke konfiguračnímu souboru. Použijeme k tomu následující příkazy:

```
# chown -R root:root /etc/ssh
# chmod 755 /etc/ssh
# chmod 600 /etc/ssh/ssh_host_key
# chmod 644 /etc/ssh/ssh_host_key.pub
# chmod 644 /etc/ssh/sshd_config
```

Posledním krokem je spuštění démona **sshd**. Obvykle pro něj vytvoříte rc soubor nebo jej přidáte do nějakého existujícího, takže se bude spouštět automaticky při startu systému. Démon běží samostatně a neuvádí se v souboru `/etc/inetd.conf`. Démon musí být spuštěn pod právy uživatele root. Syntaxe spuštění je velmi prostá:

```
/usr/sbin/sshd
```

Po svém spuštění se démon automaticky přepne do pozadí. Nyní může váš počítač přijímat **ssh** spojení.

Klient ssh

Klientských programů **ssh** je několik: **slogin**, **scp** a **ssh**. Všechny čtou stejný konfigurační soubor, pojmenovaný obvykle `/etc/ssh/ssh_config`. Kromě toho mohou číst konfigurační soubory z adresáře `.ssh` v domovském adresáři uživatele, který je spouští. Nejdůležitějším z těchto souborů je .ssh/config, který může obsahovat nastavení přepisující standardní hodnoty v souboru `/etc/ssh/ssh_config` a soubory `.ssh/identity` a `.ssh/identity.pub`, jež obsahují privátní a veřejný klíč uživatele. Dalšími důležitými jsou `.ssh/known_hosts` a `.ssh/authorized_keys`, budeme o nich hovořit později v části *Práce s ssh*. Nejprve se podívejme na globální konfigurační soubor a soubory s uživatelskými klíči.

Soubor `/etc/ssh/ssh_config` se velmi podobá konfiguračnímu souboru serveru. I v něm je možné použít celou řadu voleb, jeho minimální podobu uvádí příklad 12.5. Podrobnosti o zbylých konfiguračních volbách jsou popsány na manuálové stránce **sshd(8)**. Můžete přidávat sekce odpovídající určitým hostitelům nebo skupinám hostitelů. Parametrem volby `Host` může být buď přesná specifikace názvu hostitele, nebo můžete použít zástupné znaky a specifikovat více hostitelů tak, jako to děláme i my v našem příkladu. Například zápisem `Host *.vbrew.com` můžete vytvořit údaj vztahující se ke všem hostitelům v doméně `vbrew.com`.

Příklad 12.5 – Příklad konfiguračního souboru klienta ssh

```
# /etc/ssh/ssh_config

# Standardní nastavení při připojení ke vzdálenému hostiteli
Host *
  # Komprimovat data relace?
  Compression yes
  # .. s jakou úrovní? (1 - rychlé/slabé, 9 - pomalé/účinné)
  CompressionLevel 6

  # Přepnutí na rsh pokud selže bezpečné spojení?
  FallBackToRsh no

  # Posílat keep-alive zprávy? Užitečné při použití maškarády
  KeepAlive yes

  # Zkoušet autentikaci RSA?
  RSAAuthentication yes
  # Zkoušet autentikaci RSA v kombinaci s .rhosts?
  RhostsRSAAuthentication yes
```

Když jsme hovořili o konfiguraci serveru, říkali jsme, že každý hostitel a každý uživatel mají svůj klíč. Klíč uživatele je uložen v souboru `~/.ssh/identity`. Klíč vygenerujete příkazem `ssh-keygen` stejně jako jsme generovali klíč pro server, v tomto případě však nemusíte zadávat název souboru. Program `ssh-keygen` standardně ukládá klíč na správné místo, nicméně se vás stejně zeptá, pokud byste chtěli toto umístění změnit. Někdy je užitečné mít více identifikačních souborů a tímto způsobem je právě můžete vytvořit. Stejně jako předtím vás program `ssh-keygen` požádá o zadání hesla. Hesla představují další úroveň zabezpečení a rozhodně je doporučujeme. Při zadávání hesla se toto heslo nebude zobrazovat na obrazovce.

> **UPOZORNĚNÍ:** Pokud heslo zapomenete, neexistuje metoda jak je zjistit. Volte heslo tak, abyste si je mohli snadno zapamatovat, na druhé straně ovšem tak, aby se nedalo jednoduše uhodnout – vaše jméno není dobrá volba. Aby bylo heslo účinné, mělo by být dlouhé 10 až 30 znaků a nemělo by být tvořeno obyčejným textem. Použijte v něm i různé speciální znaky. Pokud heslo zapomenete, budete muset vygenerovat nový klíč.

Všichni uživatelé by měli jednou spustit program **ssh-keygen**, aby došlo k vygenerování jejich klíčů. Program **ssh-keygen** vytvoří uživatelům adresáře `~/.ssh/`, nastaví jim správná přístupová práva a vygeneruje privátní a veřejný klíč v souborech `.ssh/identity` a `.ssh/identity.pub`. Příklad použití programu vypadá takto:

```
$ ssh-keygen
Generating RSA keys:  .......ooooo0.............................
Key generation complete.
Enter file in which to save the key (/home/maggie/.ssh/identity):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/maggie/.ssh/identity.
Your public key has been saved in /home/maggie/.ssh/identity.pub.
The key fingerprint is:
1024 85:49:53:f4:8a:d6:d9:05:d0:1f:23:c4:d7:2a:11:67 maggie@moria
$
```

Nyní můžete **ssh** používat.

Práce s ssh

Nyní bychom měli mít příkaz **ssh** a související programy nainstalovány a připraveny ke spuštění. Podívejme se nyní krátce na to, jak je spustit.

Nejprve se zkusíme přihlásit ke vzdálenému hostiteli. Použijeme program **slogin** prakticky stejně, jako jsme v příkladu kdysi dříve popisovali použití programu **rlogin**. Při prvním připojení k hostiteli si program **ssh** zjistí veřejný klíč hostitele a požádá vás o potvrzení identity hostitele tím, že vám nabídne zkrácenou verzi jeho veřejného klíče, takzvaný **fingerprint** (otisk).

Administrátor vzdáleného systému by vám měl poskytnout otisk veřejného klíče svého hostitele, který si můžete uložit do souboru `.ssh/known_hosts`. Pokud vám administrátor tento otisk neposkytl, můžete se sice k hostiteli připojit, budete však upozorněni na to, že hostitel má vám neznámý veřejný klíč a budete dotázáni, zda si přejete tento klíč akceptovat. Pokud jste si jisti, že nikdo nezmanipuloval záznamy v DNS databázi a že se opravdu bavíte s tím hostitelem, se kterým chcete, pak klíč akceptujte. Klíč se automaticky uloží do souboru `.ssh/known_hosts` a při příštím připojení se už vás systém nebude na nic ptát. Pokud by se kdykoliv v budoucnu změnil veřejný klíč, kterým se vám vzdálený hostitel prokazuje, budete na to upozorněni, protože se může jednat o bezpečnostní ohrožení.

První přihlášení ke vzdálenému hostiteli bude vypadat nějak takto:

```
$ slogin vchianti.vbrew.com
The authenticity of host 'vchianti.vbrew.com' can't be established.
Key fingerprint is 1024 7b:d4:a8:28:c5:19:52:53:3a:fe:8d:95:dd:14:93:f5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vchianti.vbrew.com,172.16.2.3' to the list of/
    known hosts.
maggie@vchianti.vbrew.com's password:
Last login: Tue Feb  1 23:28:58 2000 from vstout.vbrew.com
$
```

Budete vyzváni k zadání hesla, kdy musíte zadat vaše heslo pro vzdálený systém, nikoliv lokální heslo. Heslo se při zadávání nezobrazuje.

Pokud neuvedete žádné další parametry, příkaz **slogin** vás na vzdálený systém přihlásí pod stejným uživatelským jménem, jaké používáte na lokálním systému. Toto chování můžete změnit parametrem –l, za kterým zadáte přihlašovací jméno, jež se má použít na vzdáleném systému. Přesně to jsme dělali v příkladu dříve v této knize.

Soubory z a na vzdáleného hostitele můžete kopírovat příkazem **scp**. Jeho syntaxe je podobná běžnému příkazu **cp** s tím rozdílem, že před názvem souboru můžete zadat i jméno hostitele. Takovéto zadání bude interpretováno jako cesta k souboru na vzdáleném hostiteli. Následující příklad ilustruje použití příkazu **scp** ke zkopírování lokálního souboru `/tmp/fred` do `/home/maggie` na hostiteli *chianti.vbrew.com*:

```
$ scp /tmp/fred vchianti.vbrew.com:/home/maggie/
maggie@vchianti.vbrew.com's password:
fred                100% |*****************************| 50165   00:01 ETA
```

I zde budete požádáni o zadání hesla. Příkaz **scp** zobrazuje užitečný ukazatel průběhu operace. Soubory ze vzdáleného hostitele na lokální systém můžete kopírovat stejně snadno, stačí prostě zadat název vzdáleného hostitele a cestu na tomto hostiteli jako zdrojový parametr a jako cílový parametr cestu na místním systému. Je dokonce možné kopírovat soubory z jednoho vzdáleného hostitele na jiný vzdálený systém, tato operace se ale obvykle nepoužívá, protože veškerá data budou přenášena přes váš počítač.

Příkazy na vzdáleném hostiteli můžete spouštět pomocí příkazu **ssh**. Jeho syntaxe je opět velice jednoduchá. Následující příkaz ukazuje, jak může uživatel maggie získat výpis kořenového adresáře na hostiteli *vchianti.vbrew.com*:

```
$ ssh vchianti.vbrew.com ls -CF /
maggie@vchianti.vbrew.com's password:
bin/    console@  dos/     home/    lost+found/  pub@   tmp/  vmlinuz@
boot/   dev/      etc/     initrd/  mnt/         root/  usr/  vmlinuz.old@
cdrom/  disk/     floppy/  lib/     proc/        sbin/  var/
```

Příkaz **ssh** můžete uvést v rouře a přesměrovat na něj vstupy nebo výstupy stejně jako u jakéhokoliv jiného příkazu, pouze s tím rozdílem, že vstupy a výstupy budou směrovány na vzdáleného hostitele prostřednictvím **ssh** spojení. Následující příklad ukazuje, jak můžeme tuto funkci použít společně s programem **tar** ke zkopírování celé adresářové struktury ze vzdáleného systému na lokální:

```
$ ssh vchianti.vbrew.com "tar cf - /etc/" | tar xvf -
maggie@vchianti.vbrew.com's password:
etc/GNUstep
etc/Muttrc
etc/Net
etc/X11
etc/adduser.conf
..
..
```

Spouštěný příkaz jsme uzavřeli do uvozovek, aby bylo zřejmé, co jsou parametry příkazu **ssh** a co platí pro lokální příkazový interpret. Tento příklad spustí na vzdáleném hostiteli příkaz **tar** a archivuje jím strukturu celého adresáře `/etc/`, výstup příkazu se zapisuje na standardní výstup. Odtud jej rourou předáváme další instanci programu **tar**, která běží na lokálním systému a rozbaluje data přijatá ze standardního vstupu.

I zde budeme požádáni o zadání hesla. Teď už by mělo být zřejmé, proč jsme se zmiňovali o možnosti nakonfigurovat **ssh** tak, aby při každém volání heslo nepožadoval. Ukážeme si nyní, jak nakonfigurovat **ssh** tak, aby při spojení se vzdáleným systémem *vchianti.vbrew.com* nežádal o zadání hesla. Už jsme se zmínili o souboru `.ssh/authorized_keys`, nyní jej použijeme. Soubor .ssh/authorized_keys obsahuje *veřejné* klíče všech vzdálených uživatelských účtů, ze kterých budeme chtít přihlašovat bez hesla. Automatické přihlášení můžete nastavit tak, že zkopírujete obsah souboru .ssh/identity.pub ze *vzdáleného* účtu do lokálního souboru `.ssh/authorized_keys`. Je nezbytně nutné, aby přístupová práva souboru authorized_keys dovolovala čtení a zápis do tohoto souboru jenom vám, v opačném případě by mohl kdokoliv použít jeho obsah a přihlašovat se ze vzdálených systémů na váš účet. Abyste zajistili správné nastavení práv, spusťte:

```
$ chmod 600 ~/.ssh/authorized_keys
```

Veřejný klíč je *jeden* dlouhý řádek textu. Pokud jej do lokálního souboru duplikujete operací kopírování a vložení, odstraňte eventuální znaky konce řádku, které se touto operací mohly vytvořit. Soubor `.ssh/authorized_keys` může obsahovat řadu klíčů, každý na samostatném řádku.

Nástroje balíku **ssh** jsou velmi mocné a nabízí celou řadu dalších užitečných funkcí a voleb, které by vás mohly zajímat. Další informace naleznete na manuálových stránkách a v dokumentaci, která se s tímto programem dodává.

KAPITOLA 13

Elektronická pošta

Elektronická pošta představuje jedno z nejvýznamnějších využití síťových služeb od doby, kdy byla síť vynalezena. Původně to byla jednoduchá služba, která kopírovala soubor z jednoho počítače do druhého, kde ho připojila k souboru *poštovní schránky* příjemce. V zásadě tento mechanismus zůstal stejný, i když stále rostoucí síť vedla se svými složitými směrovacími požadavky a se svým stále se zvyšujícím počtem zpráv k nutnosti vytvoření propracovanějšího schématu.

Byly vyvinuty různé standardy výměny elektronické pošty. Systémy v síti Internet se drží standardu RFC 822 doplněného některými dalšími RFC standardy, které umožňují prostřednictvím elektronické pošty strojově nezávislý přenos prakticky *čehokoliv* včetně grafiky, zvuku a nestandardních znaků[97]. Společnost CCITT definovala další standard, nazvaný X.400. Stále se používá v některých velkých společnostech a ve státních sítích, postupně se však vytrácí.

Na platformě Unix byl implementován velký počet programů pro přenos pošty. Jedním z nejznámějších programů je **sendmail**, vyvinutý Ericem Allmanem na University of California v Berkeley. Eric Allman dnes nabízí **sendmail** na komerční bázi, samotný program však zůstává k dispozici zdarma. **sendmail** se v některých distribucích Linuxu používá jako standardní poštovní program. O jeho konfiguraci budeme hovořit v kapitole 14.

Dále se na Linuxu používá **Exim** napsaný Philipem Hazelem na University of Cambridge. Konfiguraci programu **Exim** popisujeme v kapitole 15.

V porovnání se **sendmailem** je **Exim** docela mladý. Pro většinu aplikací budou vyhovovat oba tyto programy.

Jak **Exim**, tak **sendmail** vycházejí z řady konfiguračních souborů, které si musíte upravit podle potřeb vašeho systému. Kromě nastavení, která jsou pro činnost elektronické pošty nezbytná (například název systému), existuje celá řada dalších parametrů, jež je možné dolaďovat. Hlavní konfigurační soubor programu **sendmail** je na první pohled velmi těžko čitelný. Vypadá skoro jako něco, co napsala kočka pobíhající po klávesnici se zapnutým Shiftem. Konfigurační soubory programu **Exim** jsou strukturovanější a snáze čitelné. **Exim** však nenabízí přímou podporu UUCP a zpracovává pouze doménové adresy. Dnes už to není tak zásadní omezení jako v minulosti a v řadě případů možnosti programu **Exim** plně dostačují. Ať už použijete kterýkoliv z těchto programů, nároky na jejich nastavení jsou přibližně stejné.

V této kapitole si vysvětlíme, co to elektronická pošta je a s jakými problémy se musí její administrátor vyrovnat. V kapitolách 14 a 15 uvádíme návod pro počáteční nastavení programů **sendmail** a **Exim**. Uvedené informace by měly postačit ke zprovoznění pošty na typickém systému, kromě toho je však k dispozici i celá řada dalších voleb a dolaďováním těch nejjemnějších nuancí můžete strávit spoustu šťastných hodin.

[97] Pokud tomuto tvrzení nevěříte, přečtěte si dokument RFC 1437.

Ke konci této kapitoly se stručně zmíníme o programu **elm**, což je na unixových systémech velmi často používaný poštovní klient.

Další informace týkající se elektronické pošty na Linuxu najdete v dokumentu Electronic Mail HOWTO Guylhema Aznara[98], který se pravidelně objevuje na comp.os.linux. Zdrojové distribuce programů **elm**, **Exim** a **sendmail** rovněž obsahují rozsáhlou dokumentaci, která by vám měla zodpovědět většinu otázek týkajících se jejich instalace. V příslušných místech textu se na tyto dokumentace odkazujeme. Pokud potřebujete nějaké obecné informace o elektronické poště, existuje na toto téma řada RFC dokumentů. Ty jsou uvedeny na konci knihy.

Co je to poštovní zpráva?

Poštovní zpráva se zpravidla skládá z těla zprávy, což je samotný text zprávy, a ze speciálních administrativních dat, která označují příjemce, přenosové médium a podobně – trochu připomínají informace na dopisní obálce.

Tato administrativní data spadají do dvou kategorií; v první kategorii jsou zastoupena všechna data vztahující se k transportnímu médiu, jako je adresa odesilatele a adresa příjemce. Z tohoto důvodu se tato kategorie nazývá *obálka*. V závislosti na tom, kudy prochází daná zpráva, mohou být data z této kategorie transportním softwarem transformována.

Druhou kategorii představují všechna data, která jsou nutná pro manipulaci s poštovní zprávou a jež se nevztahují k žádnému transportnímu mechanismu. Příkladem může být řádka s předmětem zprávy, seznam všech příjemců nebo datum zaslání dané zprávy. V mnoha sítích se stalo standardem, že tato kategorie dat je uvedena před vlastní poštovní zprávou a tvoří takzvanou *poštovní hlavičku*. Od *textu zprávy* je hlavička oddělena prázdným řádkem[99].

Ve světě Unixu používá většina transportních programů formát hlavičky, který byl definován v dokumentu RFC 822. Jeho původním záměrem bylo vytvořit standard pro použití v sítích ARPANET, ale protože byl navržen jako nezávislý na prostředí, byl jednoduše upraven pro použití v dalších sítích, včetně mnoha sítí založených na protokolu UUCP.

Nicméně dokument RFC 822 je pouze největším obecným standardem. Vznikly i novější standardy, a to z důvodu zvyšujících se potřeb, jako je šifrování dat, podpora mezinárodních znakových sad a podpora multimediálního rozšíření pošty (MIME, viz RFC 1341).

Ve všech těchto standardech se hlavička zprávy skládá z několika řádků, za nimiž následuje prázdný řádek. Řádek obsahuje název pole, které začíná ve sloupci jedna, a vlastní pole, které je odděleno dvojtečkou a mezerou. Formát a sémantika každého pole jsou odlišné a závisí na názvu daného pole. Pole hlavičky může pokračovat i na novém řádku v případě, že následující řádek začíná mezerou nebo tabulátorem. Pole mohou být uspořádána v libovolném pořadí.

Typická hlavička poštovní zprávy vypadá asi takto:

```
Return-Path: <ph10@cus.cam.ac.uk>
Received: ursa.cus.cam.ac.uk (cusexim@ursa.cus.cam.ac.uk [131.111.8.6])
    by al.animats.net (8.9.3/8.9.3/Debian 8.9.3-6) with ESMTP id WAA04654
    for <terry@animats.net>; Sun, 30 Jan 2000 22:30:01 +1100
Received: from ph10 (helo=localhost) by ursa.cus.cam.ac.uk with local-smtp
    (Exim 3.13 #1) id 12EsYC-0001eF-00; Sun, 30 Jan 2000 11:29:52 +0000
Date: Sun, 30 Jan 2000 11:29:52 +0000 (GMT)
From: Philip Hazel <ph10@cus.cam.ac.uk>
Reply-To: Philip Hazel <ph10@cus.cam.ac.uk>
```

98 Guylhema můžete kontaktovat na adrese *guylhem@danmark.linux.eu.org*.

99 Kromě toho bývá zvykem připojovat ke zprávě *signaturu* (`.sig`), která obsahuje informace o odesilateli a případně nějaký vtip nebo citát. Ta bývá od těla zprávy oddělena řádkem obsahujícím znaky „--“ a mezeru.

```
To: Terry Dawson <terry@animats.net>, Andy Oram <andyo@oreilly.com>
Subject: Electronic mail chapter
In-Reply-To: <38921283.A58948F2@animats.net>
Message-ID: <Pine.SOL.3.96.1000130111515.5800A-200000@ursa.cus.cam.ac.uk>
```

Všechna potřebná pole hlavičky jsou obvykle vygenerována používaným poštovním klientem, což může být například **elm**, **pine**, **mush** nebo **mailx**. Některá pole jsou volitelná a může je přidat sám uživatel. Například program **elm** umožní upravit část hlavičky zprávy. Další pole jsou přidána poštovním transportním programem. Pokud se podíváte do své poštovní schránky, zjistíte, že každá zpráva začíná řádkem „From“ (bez mezery!) Nejedná se o hlavičku podle standardu RFC 822, tento řádek doplnil poštovní klient kvůli pohodlí programů, kterými poštu prohlížíte. Aby se předešlo případným problémům s řádky textu, které začínají textem „From“, stalo se standardem začínat takovéto řádky znakem „>“.

Následuje seznam běžně používaných polí hlavičky a jejich význam:

`From:` Toto pole obsahuje adresu odesilatele a může eventuálně obsahovat i jeho „skutečné jméno“. Pro toto pole se používá obrovské množství formátů.

`To:` Toto pole obsahuje adresu příjemce.

`Cc:` Seznam adresátů, kteří obdrží kopii zprávy. Jednotlivé adresy se od sebe oddělují čárkami.

`Bcc:` Seznam adresátů, kteří obdrží kopii zprávy. Rozdíl mezi poli „Cc:“ a „Bcc:“ spočívá v tom, že adresy uvedené v poli „Bcc:“ se neobjeví ve zprávách doručených jednotlivým adresátům. Tímto způsobem můžete poslat kopii zprávy více lidem, aniž by se jednotliví adresáti o sobě navzájem dozvěděli. Jednotlivé adresy se od sebe oddělují čárkami.

`Subject:` Toto pole obsahuje popis obsahu zprávy vyjádřený několika slovy.

`Date:` Toto pole obsahuje datum, kdy byla zpráva odeslána.

`Reply-To:` Toto pole určuje adresu, na kterou chce odesilatel směrovat odpověď od příjemce. Pole může být užitečné v případě, že máte několik účtů, ale přitom chcete dostávat poštu pouze na adresu, kterou používáte nejčastěji. Toto pole je volitelné.

`Organization:` Toto pole obsahuje společnost, která vlastní počítač a z něhož pochází daná pošta. Máte-li svůj počítač v soukromém vlastnictví, nechte toto pole buď volné, nebo sem zadejte řetězec „private“, případně nějaký nesmysl. Toto pole není definováno žádným RFC dokumentem a je úplně nepovinné. Některé poštovní programy je podporují, jiné ne.

`Message-ID:` Toto pole obsahuje řetězec, který vygeneroval transportní systém původního odesilatele. Představuje jednoznačný identifikátor zprávy.

`Received:` Toto pole vloží do hlavičky každý systém, který zpracoval vaši poštu (včetně počítačů odesilatele a příjemce). V něm je uveden název daného systému, identifikátor zprávy, čas a datum, kdy daný systém obdržel tuto zprávu, dále od kterého systému tato zpráva pochází a který transportní program byl použit k jejímu doručení. Tyto informace se uvádějí z toho důvodu, abyste mohli sledovat směrování pošty a případně si stěžovat příslušné zodpovědné osobě.

`X-cokoliv`: Žádný z poštovních programů by neměl mít potíže s jakoukoliv hlavičkou začínající řetězcem X-. Tento řetězec se používá kvůli implementaci doplňkových vlastností, jež zatím nebyly vloženy do standardu RFC, nebo které do něj ani vloženy nebudou. Tento řetězec například používala poštovní konference Linux Activists, kde se pomocí hlavičky `X-Mn-Key:` vybíral požadovaný kanál konference.

Jak se pošta doručuje?

Typicky vytvoříte poštu pomocí nějakého poštovního rozhraní, například pomocí programu **mail** nebo **mailx**; nebo pomocí dokonalejších programů, jako je **elm**, **mush** nebo **pine**. Takový program se nazývá *poštovní uživatelský agent* (*mail user agent*), zkráceně MUA. Když se posílá nějaká poštovní zpráva, ve většině případů předá uživatelský agent poštu k doručení dalšímu programu. Tento program se nazývá *poštovní přenosový agent* (*mail transport agent*), zkráceně MTA. Ve většině systémů se pro lokální i vzdálené doručování používá stejný přenosový agent, kterým typicky bývá `/usr/sbin/sendmail` (nebo na systémech nekompatibilních s FSSTN `/usr/lib/sendmail`). U systémů založených na UUCP se obvykle používají dva přenosoví agenti, **rmail** pro vzdálené doručení a **lmail** pro lokální doručení.

Místní doručování znamená samozřejmě více, než jen pouhé přidání příchozí zprávy do poštovní schránky příjemce. Místní MTA bude obvykle rozumět aliasům (což je nastavení, kdy místní adresy příjemce odkazují na další adresy) a předávání (což je přesměrování pošty uživatele na nějakou jinou destinaci). Také zprávy, které nemohou být doručeny, musí být *odmítnuty*, což znamená, že je systém musí vrátit odesilateli společně s nějakou chybovou zprávou.

U vzdáleného doručování závisí použitý transportní program na povaze daného spojení. Při doručování pošty po sítích TCP/IP se nejčastěji používá protokol SMTP (*Simple Mail Transfer Protocol*), popsaný v dokumentu RFC 821. Protokol SMTP byl navržen pro přímé doručení pošty na počítač adresáta, přičemž přenos pošty se sjedná s démonem SMTP, který je spuštěn na vzdálené straně. Dnes bývá obvyklé, že větší společnosti používají pro doručování pošty všem příjemcům jediný počítač, který je pak správně předá tam, kam patří.

V sítích na bázi protokolu UUCP nebývá obvykle pošta doručována přímo, ale je doručena požadovanému hostiteli přes několik zprostředkujících systémů. Posílá-li se zpráva pomocí spojení na bázi protokolu UUCP, spustí MTA odesilatele obvykle pomocí příkazu **uux** na doručujících systémech program **rmail** a pošle mu na standardní vstup danou zprávu.

Protože se tento proces provádí samostatně pro každou zprávu, může způsobit jak značné pracovní zatížení na hlavních poštovních uzlech, tak i přeplnění dočasné fronty protokolu UUCP stovkami malých souborů, které zabírají neúměrné množství místa na disku[100]. Proto někteří MTA umožní seskupit několik zpráv pro vzdálený systém do jednoho dávkového souboru. Dávkový soubor obsahuje příkazy protokolu SMTP, které by za normálních okolností spustil místní hostitel, kdyby se použilo přímé spojení. Tento postup se označuje jako protokol BSMTP neboli *dávkový* protokol SMTP. Dávka je potom předána programům **rsmtp** nebo **bsmtp** na vzdáleném systému, které zpracují vstup stejným způsobem, jako kdyby došlo k normálnímu SMTP spojení.

[100] Diskový prostor se totiž typicky alokuje v blocích o velikosti 1 024 bajtů, takže i pár desítek bajtů dlouhá zpráva zabere celý kilobajt.

Adresy elektronické pošty

U elektronické pošty se adresa skládá přinejmenším ze dvou částí. Jednou částí je název *poštovní domény*, který se finálně přeloží buď na adresu počítače příjemce nebo na adresu jiného počítače, jenž poštu pro příjemce přijme. Druhou částí je nějaký způsob jednoznačné identifikace uživatele, což může být přihlašovací jméno, skutečné jméno ve tvaru „Křestní.Příjmení“ nebo jakákoliv přezdívka, která se přeloží na uživatele nebo skupinu uživatelů. Jiná poštovní adresní schémata, jako je X.400, používají obecnější množinu „atributů“, které slouží k vyhledání hostitele příjemce na adresářovém serveru X.500.

Způsob interpretace poštovní adresy značně závisí na používaném typu sítě. Zaměříme se na to, jak poštovní adresy interpretují sítě TCP/IP a UUCP.

RFC 822

Systémy v síti Internet dodržují standard popsaný v RFC 822, který vyžaduje notaci *user@host.domain*, kde *host.domain* je plně kvalifikované doménové jméno daného hostitele. Spojovacím členem je znak „zavináč“ (@), který se v tomto kontextu často vyslovuje jako „at“[101].

Tato notace nespecifikuje cestu k cílovému počítači. Směrování zprávy je ponecháno na jiných mechanismech, o kterých se ještě krátce zmíníme.

Pokud používáte systém připojený k Internetu, potkáte se se standardem RFC 822 poměrně často. Jeho použití se nevztahuje jen na poštu, ale i na jiné služby, jako jsou například news.

Starší formáty adres

V původním prostředí protokolu UUCP se používaly adresy *path!host!user*, kde cesta *path* definuje sekvenci hostitelů, jimiž musí zpráva projít, než dorazí k cílovému hostiteli *host*. Tato forma zápisu se nazývá *vykřičníková notace*[102], podle znaku vykřičníku, který se používá v jejím zápisu. Dnes již mnoho sítí založených na protokolu UUCP adoptovalo standard z dokumentu RFC 822, a tudíž bude rozumět i doménové adrese.

Na jiných sítích se stále používají i jiné formáty adres. Například sítě založené na DECnet používají jako oddělovač dvě dvojtečky a pracují s adresami ve tvaru *host::user*[103]. Standard X.400 používá úplně odlišné adresační schéma, které příjemce popisuje dvojicemi příznak-hodnota, jako například stát nebo organizace.

Konečně FidoNet identifikuje každého uživatele zápisem jako 2:320/204.9, který obsahuje čtyři číselné údaje definující zónu (2 je Evropa), síť (320 je Paříž a okolí), uzel (lokální rozbočovač) a bod (počítač koncového uživatele). Adresy sítě FidoNet je možné mapovat na adresy formátu RFC 822, předchozí zápis by se zadal takto: Thomas.Quinot@p9.f204.n320.z2.fidonet.org. Je celkem zřejmé, že doménové adresy z toho všeho vycházejí nejlépe...

Kombinace různých formátů

Je celkem jasné, že když se dá dohromady spousta různých systémů a spousta chytrých lidí, budou hledat způsoby jak odlišné systémy propojit tak, aby spolu mohly komunikovat. Existuje pro-

101 Pozn. překladatele: Vyslov *et* – tedy „uživatel XY *na* počítači ABC“.

102 Pozn. překladatele: V angličtině *bang path*, v telegrafní angličtině se totiž vykřičník čte „bang“ (podobně jako u nás tečka „stop“.

103 Pokud byste chtěli doručit poštu uživateli sítě DECnet z prostředí RFC 822, můžete použít adresu *"host::user"@relay*, kde *relay* je název nějakého předávacího stroje mezi Internetem a DECnetem.

to řada poštovních bran, které jsou schopny propojovat různé poštovní systémy a zajistit tak předávání pošty z jednoho systému na jiný. Těmito branami se nebudeme nijak podrobně zabývat, zaměříme se spíš na některé komplikace, k nimž může při použití takovýchto systémů dojít.

Představme si situaci, kdy kombinujeme vykřičníkové adresy a RFC 822. Tyto dva typy adres se příliš dobře neslučují. Předpokládejme adresu ve tvaru *domainA!user@domainB*. Není zcela jasné, zda znak „@“ bude mít přednost před cestou nebo tomu bude naopak. Pošleme zprávu na doménu B, která ji předá na *domainA!user*, nebo ji pošleme na doménu A, která ji předá na *user@domainB*?

Adresy, v nichž jsou zastoupeny různé typy operátorů adres, se nazývají *hybridní adresy*. Nejznámější z nich vidíte ve výše uvedeném příkladu. Tato adresa je obvykle vyřešena tak, že operátor „@“ dostane přednost před cestou. Ve výše uvedeném příkladu to znamená, že zpráva bude odeslána nejprve na doménu B.

Existuje však způsob, jak ve formátu odpovídajícím RFC 822 zadat přesné směrování zprávy. Adresa *<@domainA,@domainB:user@domainC>* definuje adresu uživatele domény C, ke které se dostaneme přes domény A a B (v tomto pořadí). Tento typ adresy se často označuje jako *směrovaná adresa*. Není nicméně rozumné s tímto chováním počítat, protože revize dokumentu RFC popisující směrování pošty doporučují ignorovat směrované adresy a pokusit se o přímé doručení na vzdálený systém.

Dále existuje ještě adresový operátor „%“: U adresy *user%domainB@domainA* bude zpráva nejprve poslána na doménu A, kde se nejpravější (a v našem případě jediný) znak „%“ nahradí znakem „@“.Takže nyní budeme mít adresu *user@domainB* a zpráva bude spokojeně doručena danému uživateli na doméně B. Tento typ adresy se někdy označuje jako „Ye Olde ARPAnet Kludge“ a jeho používání se nedoporučuje.

Použití odlišných typů adres má určité důsledky, které si popíšeme dále. V prostředí, kde se používá standard RFC 822, byste neměli používat žádné jiné adresy než absolutní doménové adresy tvaru *user@host.domain*.

Jak pracuje směrování pošty?

Proces doručování zprávy hostiteli příjemce se nazývá *směrování*. Kromě nalezení cesty ze systému odesilatele do cílového systému v sobě tento proces zahrnuje i kontrolu chyb a optimalizaci rychlosti a nákladů.

Existuje obrovský rozdíl ve způsobu, jakým se starají o směrování pošty UUCP systémy a internetové systémy. V Internetu provede hlavní práci související se směrováním dat na hostitele příjemce (který je známý prostřednictvím své IP adresy) síťová úroveň IP, zatímco v zóně protokolu UUCP musí být směrování provedeno uživatelem nebo je musí vygenerovat poštovní přenosový agent.

Směrování pošty v Internetu

V Internetu záleží pouze na konfiguraci cílového hostitele, zda se vůbec uskuteční nějaké konkrétní směrování pošty. Implicitně je nastaveno přímé doručení zprávy cílovému hostiteli, tak, že se nejprve zjistí, kterému hostiteli má být zpráva doručena, a pak se mu přímo doručí. Většina systémů obvykle preferuje doručení veškeré příchozí pošty na jeden spolehlivý poštovní server, který je schopen postarat se o veškerou poštu a jenž ji potom předá místním hostitelům. Tato služba je oznamována v takzvaném MX záznamu v DNS databázi domény. Zkratka MX znamená *Mail Exchanger* a v zásadě říká, že uvedený server slouží jako předávací systém pro veškerou poštu

v doméně. Pomocí MX záznamů lze také obsluhovat poštu pro hostitele, kteří nejsou přímo připojeni k Internetu, například sítě UUCP nebo FidoNet musejí poštu přijímat přes nějakou bránu.

MX záznamy mají vždy přiřazenu takzvanou *preferenci*, což je celočíselná hodnota. Existuje-li pro jednu doménu více poštovních serverů, bude se pošta doručovat tomu s nejnižším preferenčním číslem a pouze pokud by se to nepovedlo, bude použit další s vyšším preferenčním číslem. Je-li nějaký hostitel zároveň poštovním serverem pro určitou cílovou adresu, smí poštu na tuto adresu doručovat pouze prostřednictvím systémů, které mají preferenční číslo nižší než on sám; což je bezpečný způsob, jak zabránit vytváření poštovních smyček. Pokud pro nějakou doménu neexistuje MX záznam, transportní agent zjistí, zda samotné doméně není přiřazena IP adresa, a pokud je, doručuje poštu přímo na tuto adresu.

Předpokládejme, že například organizace Foobar Inc. chce veškerou poštu spravovat na svém počítači s názvem **mailhub**. Potom bude mít v databázi DNS přibližně následující záznam MX:

```
green.foobar.com          IN   MX     5   mailhub.foobar.com.
```

Tento záznam říká, že na adrese **mailhub.foobar.com** je poštovní server pro doménu **green.foobar.com** s preferencí 5. Hostitel, který chce doručit zprávu na adresu **joe@green.foobar.com**, nahlédne do DNS a nalezne MX záznam, který směřuje na počítač **mailhub**. Pokud neexistuje žádný jiný MX záznam s prioritou menší než 5, bude zpráva doručena poštovnímu serveru **mailhub**, který ji posléze odešle na hostitele **green**.

Výše uvedený příklad je jen zjednodušeným znázorněním toho, jak pracují záznamy MX. Podrobnější informace o směrování pošty na Internetu naleznete v dokumentech RFC 821, RFC 974 a RFC 1123.

Směrování pošty v sítích UUCP

Směrování pošty v sítích na bázi protokolu UUCP je mnohem komplikovanější než v síti Internet, protože vlastní transportní software neprovádí žádné směrování. Dříve musela být veškerá pošta adresována pomocí vykřičníkové notace, která uváděla seznam hostitelů, přes něž se doručovala pošta. Jednotliví hostitelé byli odděleni vykřičníkem, za kterým následovalo jméno uživatele. Pokud jste chtěli adresovat dopis uživateli Janet na počítač s názvem **moria**, museli jste zadat cestu jako **eek!swim!moria!janet**. Tato formule poslala poštu z vašeho hostitele na hostitele **eek**, z něj pak na hostitele **swim** a nakonec dorazila pošta z hostitele **swim** na hostitele **moria**.

Zcela zřejmá nevýhoda této techniky spočívá v tom, že musíte znát podrobnosti o síťové topologii, rychlosti linek a podobně. Ještě horší věcí je, že změna v uspořádání síťové topologie – například odstranění některých spojení nebo odstranění nějakého hostitele – může způsobit nedoručení zprávy, protože jste nebyli obeznámeni s provedenými změnami. A konečně v případě, že změníte místo svého působení, budete pravděpodobně muset aktualizovat veškerá směrování.

Použití tohoto mechanismu směrování bylo dáno jedním důvodem – totiž neexistencí centrálně přidělovaných názvů. Předpokládejme, že existují dva systémy s názvem **moria**, jeden ve Spojených státech a druhý ve Francii. Na který systém ale adresa **moria!janet** odkazuje? To bude jasné až tehdy, uvedete-li cestu, pomocí které se lze spojit s hostitelem **moria**.

Prvním krokem, který vedl k odstraňování nejednoznačných názvů hostitelů, byl vznik The UUCP Mapping Project. Tento projekt je umístěn na Rutgers University a registruje všechny oficiální názvy UUCP počítačů společně s informacemi o jejich sousedech a geografickém umístění. Projekt zajišťuje, aby nebyl žádný název hostitele použit dvakrát. Informace získané projektem mapování názvů jsou zveřejňovány jako takzvané *Usenetové mapy*, které jsou pravidelně distribuovány po-

mocí Usenetu. Typická položka systému v mapě vypadá (po odstranění komentářů) následovně[104]:

```
moria
        bert(DAILY/2),
        swim(WEEKLY)
```

Tato položka uvádí, že hostitel **moria** má spojení s hostitelem **bert**, se kterým se spojuje dvakrát denně. Dále má spojení s hostitelem **swim**, s nímž se spojuje každý týden. K formátu souboru s mapami se ještě vrátíme.

Pomocí informací o jednotlivých spojeních, které jsou uvedeny v souborech s mapami, je možné automaticky vygenerovat celou cestu z vašeho hostitele k libovolnému cílovému systému. Tyto informace se obvykle ukládají v souboru paths, někdy se tento soubor také nazývá *databáze cest*. Předpokládejme, že v souboru map stojí, že s hostitelem **bert** se můžete spojit přes hostitele **ernie**. V tom případě by položka pro hostitele **moria** v souboru paths, který byl vygenerován z výše uvedené části mapy, mohla vypadat asi takto:

```
moria           ernie!bert!moria!%s
```

Pokud nyní zadáte cílovou adresu janet@moria.uucp, vybere MTA agent výše uvedené směrování a pošle zprávu na hostitele **ernie** a jako adresu uvede **bert!moria!janet**.

Není ovšem vhodné vytvářet soubor paths ze všech usenetových map. Informace v těchto mapách jsou často poměrně zkreslené a mnohdy i zastaralé. Z tohoto důvodu se vytváří soubor paths ze všech světových map protokolu UUCP pouze na několika hlavních hostitelích. Většina systémů spravuje pouze informace o systémech, které jsou v jejich okolí, a poštu pro hostitele, které nenajdou ve své databázi, pošlou chytřejším hostitelům, jež mají kompletnější směrovací informace. Toto schéma se nazývá *smart-host routing*. Hostitelé, kteří udržují pouze jediné poštovní spojení pomocí protokolu UUCP (takzvané *listové systémy*), sami neprovádí žádné směrování; plně se spoléhají na svého chytřejšího hostitele.

Kombinace UUCP a RFC 822

Zatím je nejlepším lékem na problémy se směrováním pošty v sítích na bázi protokolu UUCP převzetí systému DNS do sítí UUCP. Samozřejmě, že pomocí protokolu UUCP se nemůžete dotazovat jmenného serveru. Některé systémy UUCP však vytvořily malé domény, které vnitřně koordinují směrování pošty. V mapách tyto domény zveřejní pouze jednoho nebo dva hostitele, kteří budou označení jako poštovní brány. Tím pádem nemusí v mapách existovat položka pro každého hostitele, který se nachází v příslušné doméně. Brány se starají o veškerou poštu, která přichází do domény nebo která z dané domény odchází. Směrovací schéma uvnitř domény je pro okolní svět neviditelné.

Tento princip funguje velmi dobře se směrováním na chytřejší hostitele. O globální směrovací informace se starají pouze brány; vedlejší hostitelé příslušné domény vystačí pouze s malým ručně vytvořeným souborem paths, který uvádí směrování uvnitř jejich domény a směrování na bránu. Dokonce ani poštovní brány nemusí mít informace o směrování na každého UUCP hostitele na světě. Kromě kompletních informací o směrování uvnitř jimi obsluhované domény potřebují mít ve svých databázích pouze směrování na všechny ostatní domény. Například níže uvedená položka cesty bude směrovat veškerou poštu určenou pro systémy v doméně **sub.org** na hostitele **smurf**:

---

[104] Mapy systémů registrovaných pod UUCP Mapping Project jsou distribuovány prostřednictvím skupiny comp.mail.maps, různé organizace mohou dále zveřejňovat vlastní mapy svých sítí.

```
.sub.org          swim!smurf!%s
```

Pošta směřující na adresu **claire@jones.sub.org** bude poslána na hostitele **swim** s adresou **smurf!jones!claire**.

Hierarchické uspořádání prostoru s názvy domén umožňuje poštovním serverům kombinovat přesnější směrování s méně přesným směrováním. Například systém ve Francii může mít přesné směrování pro subdomény v rámci domény **fr**, ale veškerou poštu určenou pro hostitele v doméně **us** bude směrovat na nějaký systém ve Spojených státech. Takovéto doménové směrování značně redukuje velikost směrovacích databází a stejně tak i potřebnou administrativní režii.

Hlavním přínosem použití názvu domén v prostředí sítí UUCP je to, že shoda se standardem RFC 822 umožňuje mezi sítěmi na bázi protokolu UUCP a sítí Internet jednoduchou průchodnost dat. V dnešní době má spousta domén UUCP spojení s internetovou bránou, která se chová jako jejich chytřejší hostitel. Posílání zpráv po Internetu je rychlejší a směrování informací je mnohem spolehlivější, protože hostitelé v Internetu mohou používat místo usenetových map systém DNS.

Aby mohl být systém na bázi protokolu UUCP dosažitelný z Internetu, uvádí obvykle internetové brány záznam MX pro domény na bázi protokolu UUCP (záznamy MX byly popsány výše v části *Směrování pošty v Internetu*). Předpokládejme třeba, že hostitel **moria** patří do domény **orcnet.org**. Hostitel **gcc2.groucho.edu** se chová vůči této doméně jako internetová brána. Z toho důvodu použije hostitel **moria** jako svého chytřejšího hostitele **gcc2**, takže veškeré zprávy pro cizí domény budou doručovány pomocí Internetu. Na druhou stranu hostitel **gcc2** uvede záznam MX pro doménu ***.orcnet.org** a tím pádem může doručit veškerou příchozí poštu určenou pro systémy v doméně **orcnet** na hostitele **moria**. Znak * v ***.ornet.org** znamená všechny hostitele v dané doméně, takže pro ně nemusíte mít samostatné záznamy. Takovéto nastavení je typické pro čistě UUCP domény.

Nyní zůstal už jen poslední problém, totiž že transportní programy protokolu UUCP neumí obsloužit plně kvalifikovaná doménová jména. Většina balíků UUCP byla navržena tak, aby zvládly názvy systémů do délky osmi znaků, některé dokonce i méně, a použití nealfanumerických znaků, jako jsou například tečky, je u většiny z nich absolutně nepřípustné.

Proto je nutná existence určitého převodu mezi názvy odpovídajícími standardu RFC 822 a názvy hostitelů protokolu UUCP. Způsob, jakým je tento převod prováděn, zcela závisí na typu implementace. Obecný způsob, jak namapovat názvy FQDN na názvy protokolu UUCP, spočívá v použití mapování přímo v souboru cest:

```
moria.orcnet.org   ernie!bert!moria!%s
```

Tato formule vytvoří z adresy, která udává plně kvalifikovaný název domény, čistokrevnou vykřičníkovou notaci pro použití v protokolu UUCP. Některé mailery k tomuto účelu poskytují speciální soubory; například program **sendmail** používá soubor uucpxtable.

Někdy se při posílání pošty ze sítě na bázi protokolu UUCP do Internetu vyžaduje zpětná transformace (hovorově zvaná *doménizace*). Pokud odesilatel pošty použije jako cílovou adresu plně kvalifikované doménové jméno, lze se tomuto problému vyhnout tak, že se při doručování zprávy chytřejšímu hostiteli neodstraní název domény z adresy na obálce. Nicméně stále ještě existují UUCP systémy, které nejsou součástí žádné domény. Jejich doménizace se typicky provede jejich přiřazením do fiktivní domény **uucp**.

Databáze cest poskytuje v sítích na bázi protokolu UUCP hlavní informace o směrování. Typická položka vypadá asi takto (název systému je od cesty oddělen tabulátory):

```
moria.orcnet.org  ernie!bert!moria!%s
moria             ernie!bert!moria!%s
```

Tento záznam uvádí, že každá zpráva určená pro hostitele **moria** bude doručena přes hostitele **ernie** a **bert**. Je zde zadán jak plně kvalifikovaný název, tak i název pro protokol UUCP. Oba názvy jsou zde uvedeny pro případ, že by mailer neměl samostatný způsob pro mapování mezi těmito dvěma jmennými prostory.

Pokud chcete směrovat všechny zprávy určené pro hostitele v rámci nějaké domény na jejich poštovní brány, můžete v databázi cest zadat také cestu, která bude mít jako cíl název domény, před nímž bude uvedena tečka. Pokud mohou být například hostitelé v doméně **sub.org** dosažitelní přes poštovní bránu **swim!smurf**, mohla by položka v databázi cest vypadat následovně:

```
.sub.org        swim!smurf!%s
```

Vytvoření souboru cest je přijatelné pouze v případě, že provozujete systém, který neobsahuje příliš mnoho informací o směrování. Provádíte-li směrování pro velké množství hostitelů, bude lepší k tomuto účelu použít příkaz **pathalias**, který umí vytvořit soubor cest ze souborů map. Mapy lze spravovat daleko snáze, protože konkrétní systém lze přidat nebo odstranit tak, že v mapě upravíte jeho položku a necháte znovu vytvořit soubor map. I když se mapy zveřejňované usenetovým projektem mapování názvů už ke směrování moc nepoužívají, mohou menší sítě na bázi protokolu UUCP poskytovat informace o směrování pomocí svých vlastních skupin map.

Soubor map se skládá především ze seznamu systémů, ve kterém má každý systém uveden seznam systémů, s nimiž se může spojit nebo jež se mohou spojit s ním. Název systému začíná ve sloupci jedna a za ním následuje seznam jednotlivých spojení, která jsou vzájemně oddělena čárkami. Seznam může pokračovat i na dalších řádcích v případě, že následující řádek začíná znakem tabulátor. Každý záznam o spojení je tvořen názvem systému, za kterým je v hranatých závorkách uvedena jeho režie. Režie představuje aritmetický výraz složený z čísel a symbolických výrazů jako DAILY nebo WEEKLY. Řádky začínající znakem # jsou ignorovány.

Jako příklad si vezměme hostitele **moria**, který se spojuje dvakrát denně s hostitelem **swim.twobirds.com** a jedenkrát týdně s hostitelem **bert.sesame.com**. Pro spojení s hostitelem **bert** používá pouze pomalý modem s rychlostí 2 400 b/s. Hostitel **moria** by měl v souboru map zveřejnit následující položku:

```
moria.orcnet.org
        bert.sesame.com(DAILY/2),
        swim.twobirds.com(WEEKLY+LOW)
moria.orcnet.org = moria
```

Poslední řádek říká, že hostitel **moria** může být rozpoznán i na základě svého názvu pro protokol UUCP. Všimněte si, že musí být uvedena režie *DAILY/2*, protože volání dvakrát denně ve skutečnosti snižuje režii tohoto spojení na polovinu.

Při použití takovýchto souborů map je schopen příkaz **pathalias** vypočítat optimální směrování na libovolný cílový systém, který je uveden v souboru cest. Dále je schopen z těchto souborů vytvořit databázi cest, která může být poté používána pro směrování do těchto systémů.

Příkaz **pathalias** poskytuje množství dalších vlastností, například skrytí systému (tedy to, že systémy mohou být přístupné pouze pomocí brány). Podrobnosti i úplný seznam příkazů pro ohodnocování spojů najdete na manuálových stránkách příkazu **pathalias**.

Komentáře v souboru map obvykle obsahují doplňkové informace o systémech, které jsou v tomto souboru popsány. Tyto komentáře musí odpovídat pevně stanovenému formátu, takže je lze získat z map. Například program **uuwho** používá k zobrazení těchto informací, které jsou mimochodem naformátovány moc hezky, databázi vytvořenou ze souborů map. Pokud svůj systém zaregistrujete u organizace, která distribuuje svým členům soubory map, budete obvykle muset v mapě vyplnit přibližně takovýto nějaký záznam. Následuje vzorová položka mapy (je to záznam popisující Olafův systém):

```
#N      monad, monad.swb.de, monad.swb.sub.org
#S      AT 486DX50; Linux 0.99
#O      private
#C      Olaf Kirch
#E      okir@monad.swb.de
#P      Kattreinstr. 38, D-64295 Darmstadt, FRG
#L      49 52 03 N / 08 38 40 E
#U      brewhq
#W      okir@monad.swb.de (Olaf Kirch); Sun Jul 25 16:59:32 MET DST
#
monad   brewhq(DAILY/2)
# Domains
monad = monad.swb.de
monad = monad.swb.sub.org
```

Za prvními dvěma znaky je vždy zapsán tabulátor. Význam většiny polí je zřejmý; detailní popis obdržíte od jakékoliv domény, u které se zaregistrujete. Největší zábavou je rozluštit význam pole *L*: to uvádí vaše geografické umístění ve formátu zeměpisná šířka/zeměpisná délka a používá se při vykreslování postscriptových map, které ukazují všechny systémy v rámci dané země nebo všechny systémy na světě[105].

Konfigurace programu elm

Název **elm** znamená „electronic mail“ a tento program je tak jedním z nejvýstižněji pojmenovaných unixových nástrojů. Program **elm** poskytuje celoobrazovkové rozhraní s propracovanou nápovědou. Nebudeme se zde zabývat způsobem jeho použití, ale zdržíme se pouze u jeho konfiguračních voleb.

Teoreticky můžete provozovat program **elm** bez jakékoliv konfigurace a přitom bude vše pracovat správně – pokud budete mít štěstí. Existuje však několik voleb, které je potřeba nastavit, i když budou potřeba jen při určitých událostech.

Při startu si program **elm** přečte několik konfiguračních proměnných ze souboru elm.rc, který se nachází v adresáři `/etc/elm`. Potom se pokusí přečíst z vašeho domovského adresáře soubor .elm/elmrc. Tento soubor si obvykle nebudete vytvářet sami. Program ho vytvoří za vás, když z nabídky options vyberete volbu „Save new options“.

Sada voleb použitá v privátním souboru `elmrc` je také dostupná v globálním souboru `elm.rc`. Většina nastavení uvedená v soukromém souboru elmrc potlačí nastavení uvedená v globálním souboru.

[105] Pravidelně jsou zveřejňovány na news.lists.ps–maps. Pozor – jsou OBROVSKÉ!

Globální nastavení programu elm

V globálním souboru `elm.rc` musíte nastavit volby, které se týkají názvu vašeho hostitele. Například ve společnosti Virtual Brewery by mohl soubor pro hostitele **vlager** obsahovat následující informace:

```
#
# Jméno hostitele
hostname = vlager
#
# Jméno domény
hostdomain = .vbrew.com
#
# Plně kvalifikované doménové jméno
hostfullname = vlager.vbrew.com
```

Tyto volby poskytují programu **elm** informace o názvu místního hostitele. I když se tyto informace používají jen zřídka, měli byste je mít nastaveny. Tyto informace mají význam pouze v případě, že je uvedete v globálním konfiguračním souboru; nacházejí-li se ve vašem soukromém souboru elmrc, budou ignorovány.

Mezinárodní znakové sady

V minulosti byly navrženy doplňky standardu definovaného v dokumentu RFC 822, které by umožnily podporu různých typů zpráv, například prostý text, binární soubory, postscriptové soubory a podobně. Tato skupina standardů se běžně označuje jako MIME (Multipurpose Internet Mail Extensions). Kromě jiného tyto standardy umožňují příjemci zjistit, zda byla při psaní zprávy použita jiná znaková sada než standardní znaková sada ASCII, například francouzské akcenty nebo německé přehlásky. V určitých mezích tyto standardy podporuje i program **elm**.

Znaková sada, kterou vnitřně používá operační systém Linux, se obvykle označuje jako ISO-88591, což je název standardu, jejž tato znaková sada splňuje. Tento standard je také známý pod označením Latin-1. Všechny zprávy používající znaky z této znakové sady by měly mít ve své hlavičce následující řádek:

```
Content-Type:  text/plain; charset=iso-8859-1
```

Přijímací systém by měl toto pole rozeznat a při zobrazování zprávy by měl provést příslušná opatření. Pro zprávy typu *text/plain* (prostý text) je implicitně nastaveno pole znakové sady *charset* na hodnotu *us-ascii*.

Aby bylo možné zobrazit zprávy napsané v jiné znakové sadě, než je znaková sada ASCII, musí program **elm** vědět, jak má tyto znaky zobrazit. Pokud program **elm** obdrží zprávu, ve které má pole *charset* jinou hodnotu než *us-ascii* (nebo jiného typu než je *text/plain*), pokusí se implicitně zobrazit zprávu pomocí příkazu **metamail**. Zprávy vyžadující pro zobrazení program **metamail** mají na obrazovce v přehledu zpráv zobrazeno v prvním sloupci písmeno **M.**

Protože je implicitní znakovou sadou operačního systému Linux znaková sada ISO-8859-1, není nutné pro zobrazení zpráv vytvořených pomocí této znakové sady používat program **metamail**. Je-li programu **elm** sděleno, že zobrazovací systém odpovídá standardu ISO-8859-1, pak se program **metamail** nepoužije, ale zpráva bude zobrazena přímo. To zajistíte nastavením následující volby v globálním souboru elm.rc:

```
displaycharset = iso-8859-1
```

Pamatujte, že tuto volbu byste měli nastavit i v případě, že neplánujete posílání nebo přijímání zpráv, které obsahují jiné znaky než definuje standard ASCII. Tato volba se uvádí proto, že lidé posílající takovýto typ zpráv obvykle nakonfigurují svůj mailer tak, aby vložil do hlavičky pošty patřičné pole Content-Type:, a to bez ohledu na skutečnost, zda je či není daná zpráva napsána ve znakové sadě ASCII.

Ale pouze nastavení této volby v souboru elm.rc nestačí. Problém spočívá v tom, že při zobrazování zprávy pomocí vestavěného prohlížeče zavolá program **elm** pro každý zobrazovaný znak příslušnou funkci knihovny, aby zjistil, zda daný znak je či není zobrazitelný. Implicitně bude tato funkce považovat za zobrazitelné pouze znaky splňující standard ASCII a všechny ostatní znaky zobrazí jako ^?. Tento problém lze vyřešit tak, že nastavíme proměnnou prostředí LC_CTYPE na hodnotu ISO-8859-1. Tato proměnná sdělí knihovně, aby považovala za zobrazitelné všechny znaky ze znakové sady Latin-1. Podpora této i dalších vlastností je dostupná od verze knihovny 4.5.8.

Při posílání zpráv, které obsahují speciální znaky ze znakové sady ISO-8859-1, byste se měli ujistit, že jste v souboru elm.rc nastavili ještě další dvě proměnné:

```
charset = iso-8859-1
textencoding = 8bit
```

Tyto volby způsobí, že program **elm** uvede v hlavičce pošty, že příslušná zpráva byla vytvořena pomocí znakové sady ISO-8859-1 a že bude poslána ve formě 8bitových hodnot (implicitně se veškeré znaky překládají na 7bitové hodnoty).

Samozřejmě že jakoukoliv z těchto voleb lze místo v globálním konfiguračním souboru uvést v soukromém souboru elmrc, takže uživatelé mohou používat odlišné volby v případě, že jim globální nastavení nevyhovuje.

Program sendmail

Říká se, že *skutečným* správcem Unixu nebudete, dokud neupravíte soubor sendmail.cf. Ale také se říká, že musíte být blázen, když to chcete zkoušet i podruhé.

Program **sendmail** je neuvěřitelně výkonný nástroj. Většina lidí ho chápe jen velmi obtížně a ještě hůře se ho učí. Jakýkoliv program, jehož kompletní manuál (manuál k programu **sendmail** Bryana Costalese a Erica Allmana vydalo nakladatelství O'Reilly and Associates) má 1 050 stran, zcela pochopitelně vyděsí většinu lidí. Na konci této knihy uvádíme odkazy na dokumentaci k programu **sendmail**.

Nové verze programu **sendmail** jsou naštěstí odlišné. Odstraňují nutnost úprav záhadného souboru sendmail.cf a obsahují nástroje, které jej vytvoří automaticky z daleko jednodušších makrosouborů. Nemusíte proto znát celou syntaxi souboru sendmail.cf, protože v makrosouborech ji nepotřebujete. Namísto toho pouze uvedete názvy položek, například názvy funkcí, které má vaše konfigurace podporovat, a uvedete nějaké parametry, jak se má funkce chovat. Tradiční unixový nástroj **m4** pak vezme vámi vytvořené makrosoubory, zkombinuje je se šablonami definujícími syntaxi souboru sendmail.cf a vytvoří tento soubor automaticky.

V této kapitole se s programem **sendmail** seznámíme, ukážeme si, jak jej nainstalovat, nakonfigurovat a otestovat, přičemž jako příklad použijeme síť virtuálního pivovaru. Věříme, že vám zde uvedené příklady pomohou pochopit konfiguraci programu **sendmail** a že na jejich základě budete schopni vytvářet i podstatně složitější konfigurace.

Instalace programu sendmail

Transportní agent **sendmail** bývá obvykle součástí většiny linuxových distribucí. V takovém případě je instalace velmi jednoduchá. I přesto ale bývá rozumné instalovat **sendmail** přímo ze zdrojových kódů, zejména pokud vám záleží na bezpečnosti. Program **sendmail** je velmi složitý a v průběhu let získal pověst programu s řadou bezpečnostních slabin. Jedním z nejznámějších příkladů je červ RTM, který využíval chyby přetečení bufferu ve starších verzích programu **sendmail**. Stručně jsme o něm hovořili v kapitole 9. Většina takovýchto ohrožení staví na tom, že kopie programu **sendmail** na různých systémech jsou identické a ukládají data vždy na určitá místa. Přesně k tomu dochází, pokud **sendmail** nainstalujete přímo z distribuce. Pokud si jej sami přeložíte, riziko se zmenšuje. Novější verze programu **sendmail** jsou podstatně odolnější, protože s nárůstem významu bezpečnosti prodělaly velmi přísné testování.

Zdrojový kód programu **sendmail** je k dispozici na anonymním FTP na adrese *ftp.sendmail.org*.

Překlad je velice jednoduchý, protože zdrojový balík přímo podporuje Linux. **sendmail** přeložíte následujícím postupem:

```
# cd /usr/local/src
# tar xvfz sendmail.8.9.3.tar.gz
# cd src
# ./Build
K instalaci vzniklých binárních souborů (kterou provedete následujícími příka-
zy) potřebujete práva superuživatele:
# cd obj.Linux.2.0.36.i586
# make install
```

Tím nainstalujete program **sendmail** do adresáře `/usr/sbin`. Kromě toho se v adresáři `/usr/bin` vytvoří několik symbolických odkazů. Zmíníme se o nich, až se budeme zabývat některými příklady spouštění programu **sendmail**.

Konfigurační soubory – přehled

Tradiční program **sendmail** se nastavuje pomocí systémového konfiguračního souboru (obvykle je to soubor `/etc/mail/sendmail.cf` nebo na starších distribucích `/etc/sendmail.cf` nebo dokonce `/usr/lib/sendmail.cf`), který se nepodobá žádnému z jazyků, s nimiž jste se mohli až doposud setkat. Editace souboru sendmail.cf tak, aby se program choval požadovaným způsobem, může být náročnou zkušeností.

V současné době jsou všechna konfigurační nastavení řízena pomocí maker, která používají snadno pochopitelnou syntaxi. Konfigurace generovaná pomocí maker pokrývá většinu potřeb, stále však ještě můžete vzniklý soubor sendmail.cf upravit ručně tak, aby pracoval i ve velmi složitých prostředích.

Soubory sendmail.cf a sendmail.mc

Makroprocesor **m4** vytvoří soubor sendmail.cf z makrosouboru, který vytváří správce systému. Ve zbytku textu budeme tento konfigurační soubor označovat jako sendmail.mc.

Proces konfigurace v zásadě spočívá ve vytvoření správného souboru sendmail.mc, který obsahuje makra popisující požadovanou konfiguraci. Makra představují výrazy, kterým rozumí procesor **m4**, jenž je pak převede do složité syntaxe souboru sendmail.cf. Makro se skládá z názvu makra (text psaný velkými písmeny na začátku), který může být propojen s nějakou funkcí nějakého programovacího jazyka, a z nějakých parametrů (textu v závorkách), jež se použijí při rozkladu makra. Tyto parametry se mohou buď přímo promítnout do souboru sendmail.cf, nebo mohou ovlivnit způsob, jakým bude makro zpracováno.

Soubor sendmail.mc pro minimální konfiguraci (UUCP nebo SMTP s předáváním veškeré pošty jedinému přímo připojenému hostiteli) bude dlouhý 10 až 15 řádků, nepočítaje v to komentáře.

Dva příklady souboru sendmail.mc

Pokud spravujete více poštovních systémů, nemusíte konfigurační soubor pojmenovat sendmail.mc. Namísto toho se běžně pojmenovává názvem hostitele, pro nějž je určen – v našem případě to bude vstout.m4. Na názvu příliš nezáleží, hlavně že se výstup bude jmenovat sendmail.cf. Pojmenovávání jednotlivých souborů názvy hostitelů umožňuje ukládat je všechny v jednom adresáři, což je otázka pohodlí administrátora. Podívejme se na dva příklady konfiguračních souborů, čistě abychom viděli, o co jde.

Většina konfigurací programu **sendmail** dnes používá čistě SMTP. Konfigurace **sendmailu** pro podporu SMTP je velice jednoduchá. Konfigurace v příkladu 14.1 předpokládá existenci DNS ser-

veru, který zajistí překlad názvů na adresy, a veškerou poštu pro všechny systémy bude doručovat čistě protokolem SMTP.

Příklad 14.1 – Konfigurační soubor vstout.smtp.m4

```
divert(-1)
#
# Příklad konfigurace pro vstout - jen smtp
#
divert(0)
VERSIONID('@(#)sendmail.mc    8.7 (Linux) 3/5/96')
OSTYPE('linux')
#
# Podpora lokálního a smtp doručování.
MAILER('local')
MAILER('smtp')
#
FEATURE(rbl)
FEATURE(access_db)
# end
```

Soubor sendmail.mc pro počítač vstout ve virtuálním pivovaru vidíme na příkladu 14.2. Pro komunikaci se všemi hostiteli na síti pivovaru používáme protokol SMTP a můžete si všimnout podobnosti s právě uvedeným příkladem. Kromě toho se veškerá pošta pro ostatní počítače na Internetu odesílá protokolem UUCP přes počítač moria.

Příklad 14.2 – Konfigurační soubor vstout.uucpsmtp.m4

```
divert(-1)
#
# Příklad konfigurace pro vstout
#
divert(0)
VERSIONID('@(#)sendmail.mc 8.7 (Linux) 3/5/96')
OSTYPE('linux')
dnl
# moria je předávací hostitel, používáme přenos "uucp-new".
define('SMART_HOST', 'uucp-new:moria')
dnl
# Podpora lokálního, smtp a uucp doručování.
MAILER('local')
MAILER('smtp')
MAILER('uucp')
LOCAL_NET_CONFIG
# Toto pravidlo zajišťuje, že všechna lokální pošta se posílá přes
# smtp, ostatní pošta jde přes předávacího hostitele.
R$* < @ $* .$m. > $*   $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
dnl
#
FEATURE(rbl)
FEATURE(access_db)
# end
```

Pokud si obě konfigurace porovnáte, pravděpodobně se vám podaří odhadnout, co jednotlivé konfigurační parametry dělají. Postupně si je všechny podrobně popíšeme.

Typicky používané parametry souboru sendmail.mc

Několik položek souboru sendmail.mc je zapotřebí vždycky, jiné je možné vynechat v případě, že vám budou vyhovovat implicitní nastavení. Obecné konfigurační příkazy souboru sendmail.mc jsou:

1. VERSIONID
2. OSTYPE
3. DOMAIN
4. FEATURE
5. Definice lokálních maker
6. MAILER
7. Pravidla LOCAL_*

V dalším textu budeme o jednotlivých parametrech hovořit a jejich použití si vysvětlíme na příkladech 14.1 a 14.2.

Komentáře

Řádky souboru `sendmail.mc`, které začínají znakem #, program **m4** nezpracovává a standardně je přímo přepisuje do výsledného souboru `sendmail.cf`. Je to užitečné, abyste okomentovali jednotlivá nastavení jak ve vstupním, tak i ve výstupním souboru.

Pokud chcete v souboru `sendmail.mc` použít komentáře, které se *nemají* přepsat do výstupního souboru, můžete použít tokeny *divert* a *dnl* programu **m4**. Nastavení *divert(-1)* potlačí veškerý výstup, *divert(0)* obnoví původní chování výstupu. Jakýkoliv výstup generovaný mezi těmito dvěma řádky bude potlačen. V našich příkladech tato nastavení používáme k vytvoření komentářů, které zůstanou pouze v souboru `sendmail.mc`. Pokud bychom chtěli dosáhnout stejného efektu na jediném řádku, můžeme použít token *dnl*, který doslova znamená „smaž všechny znaky až po začátek dalšího řádku a přidej na výstup znak nového řádku“. I tento token v našich příkladech používáme.

Jedná se o standardní příkazy programu **m4** a můžete se o nich dozvědět více na manuálových stránkách.

```
VERSIONID a OSTYPE
VERSIONID('@(#)sendmail.mc 8.9 (Linux) 01/10/98')
```

Makro VERSIONID je nepovinné a často se používá k zaznamenání verze konfigurace do souboru sendmail.cf. Proto jej zpravidla v příkladech najdete a i my vám doporučujeme jej používat. V každém případě však nezapomeňte na makro:

```
OSTYPE('linux')
```

Jedná se o pravděpodobně nejdůležitější definici. Makro OSTYPE způsobí přidání konfiguračních nastavení, která představují správné standardní hodnoty pro daný cílový systém. Většina definicí v makru OSTYPE zahrnuje nastavení cest k různým konfiguračním souborům, cesty k poštovním programům a jejich parametry, a umístění adresářů, v nichž **sendmail** ukládá zprávy. Standardní distribuce programu **sendmail** obsahuje nastavení potřebných hodnot i pro Linux a tato nastavení zahrne právě uvedené makro. Některé distribuce Linuxu, zejména Debian, obsahují vlastní de-

finiční soubor, který je plně kompatibilní s Linux-FHS. Pokud vaše distribuce takovýto soubor obsahuje, měli byste jej asi použít namísto standardního nastavení.

Definice OSTYPE by měla být uvedena hned z počátku konfiguračního souboru, protože na ní závisí řada dalších definic.

DOMAIN

Makro DOMAIN je užitečné v případě, že chcete na stejné síti nakonfigurovat větší počet počítačů stejným způsobem. Pokud nastavujete jen několik počítačů, pravděpodobně jeho použití nestojí za to. Typicky se tímto makrem konfigurují nastavení platná pro všechny počítače v síti, například název předávacího hostitele a podobně.

Standardní instalace obsahuje adresář šablon programu **m4**, které slouží k řízení konfiguračního procesu. Tento adresář se obvykle jmenuje `/usr/share/sendmail.cf` nebo nějak podobně. Naleznete zde podadresář domain, který obsahuje konfigurační šablony platné pro celou doménu. Abyste mohli použít makro DOMAIN, musíte si vytvořit vlastní makrosoubor obsahující standardní nastavení platná pro vaše systémy a uložit jej do podadresáře domain. Typicky se zde uvádějí definice specifické pro vaši doménu, například definice předávacího hostitele nebo poštovního uzlu, můžete však nastavit i další hodnoty.

Distribuce programu **sendmail** obsahuje celou řadu doménových konfiguračních souborů, pomocí nichž si můžete vytvořit svůj vlastní.

Pokud si vlastní doménový soubor uložíte jako `/usr/share/sendmail.cf/domain/vbrew.m4`, můžete jej pak v souboru `sendmail.mc` použít takto:

```
DOMAIN('vbrew')
```

FEATURE

Makro FEATURE umožňuje zahrnout do konfigurace předdefinované funkce programu **sendmail**. Tyto funkce umožňují velmi jednoduchou konfiguraci. Je jich velké množství a v této kapitole se zmíníme jen o některých nejdůležitějších. Podrobný popis všech předdefinovaných funkcí naleznete v `CF` souboru dodávaném s programem **sendmail**.

K zahrnutí kterékoliv funkce stačí v souboru `sendmail.mc` uvést takovýto řádek:

```
FEATURE(název),
```

kde jako *název* zapíšete název požadované funkce. Některé funkce mají nepovinné parametry. Pokud byste chtěli použít jiné než standardní nastavení, zadáte položku takto:

```
FEATURE(název, parametr),
```

kde *parametr* je hodnota požadovaného parametru.

Definice lokálních maker

Standardní konfigurační makrosoubory programu **sendmail** obsahují řadu propojení a proměnných, které lze použít k úpravě konfigurace. Obecně se označují jako *definice lokálních maker*. Řada z nich je uvedena v souboru CF v distribuci **sendmailu**. Definice lokálních maker se obvykle používají uvedením názvu makra s parametrem udávajícím hodnotu předávanou proměnné, již makro udržuje. V dalších příkladech v této kapitole si popíšeme některá základní lokální makra.

Definice transportních protokolů pošty

Pokud chcete, aby **sendmail** doručoval poštu i jinak než jenom lokálně, musíte mu říct, jaký způsob přenosu má použít. Velmi to zjednodušuje makro `MAILER`. Současné verze programu **sendmail** podporují řadu transportních protokolů, některé jsou experimentální, jiné se používají jen zřídka.

V naší síti potřebujeme přenos protokolem SMTP pro příjem a odesílání pošty mezi počítači v lokální síti a přenos protokolem UUCP pro příjem a odesílání pošty na předávacího hostitele. Dosáhneme toho jednoduše tak, že nadefinujeme přenosové protokoly `smtp` a `uucp`. Přenos typu `local` je implicitně zapnut, pro větší názornost jej však můžete zapnout i explicitně. Pokud používáte protokoly `smtp` i `uucp`, je nutné protokol `smtp` uvést jako první. Následující seznam představuje některé běžně používané přenosové protokoly:

`local` Tento transportní protokol zahrnuje jednak lokálního agenta pro doručování zpráv do schránek uživatelů místního systému a jednak doručovač prog používaný pro odesílání zpráv lokálním programům. Tento protokol se aktivuje standardně.

`smtp` Tento protokol implementuje přenos protokolem SMTP (Simple Mail Transport Protocol), což je nejběžnější poštovní přenosový protokol v Internetu. Při jeho aktivaci se zapínají čtyři doručovací systémy: `smtp` (základní SMTP), `esmtp` (rozšířené SMTP), `smtp8` (osmibitové binární SMTP) a `relay` (pro předávání zpráv mezi poštovními branami).

`uucp` Přenos protokolem UUCP zahrnuje dva programy: `uucp-old` pro tradiční UUCP přenosy a `uucp-new`, který umožňuje přenos zpráv více adresátům v jedné zásilce.

`usenet` Tento systém umožňuje přímé odesílání zpráv do usenetových news. Lokální zprávy určené na adresu *news.group.usenet* budou předány do sítě news a skupiny *news.group*.

`fax` Pokud máte nainstalován program HylaFAX, můžete tímto transportním systémem směrovat poštu na něj, takže získáte vestavěnou faxovou bránu. Tato funkce je v experimentálním stadiu a další informace o ní můžete zjistit na adrese *http://www.vix.com/hylafax*.

Existují i další transportní systémy, například pop, procmail, mail11, phquery a cyrus, které jsou sice užitečné, ale méně používané. Pokud vás zajímají, můžete se o nich dozvědět více buď v již zmíněné knize věnované sendmailu, nebo přímo v dokumentaci k tomuto programu.

Konfigurace směrování pošty pro lokální hostitele

Konfigurace pošty ve virtuálním pivovaru je pravděpodobně složitější, než tomu bude ve většině reálných aplikací. Většina dnešních sítí používá pouze protokol SMTP a vůbec se nezabývá UUCP přenosy. V našem příkladu jsme nastavili „chytrého hostitele“, kterého jsme použili pro obsluhu veškeré odchozí pošty. Protože ale na lokální síti používáme SMTP, musíme programu **sendmail** říct, aby poštu pro lokální systémy neposílal přes poštovní bránu. Makro `LOCAL_NET_CONFIG` umožňuje vložit přímo do souboru `sendmail.cf` pravidla, která definují způsob obsluhy lokální pošty. O přepisovacích pravidlech budeme podrobněji hovořit později, pro tuto chvíli nám stačí vědět, že jsme nadefinovali pravidlo, které nařizuje veškerou poštu pro doménu *vbrew.com* doručovat přímo na cílové hostitele protokolem SMTP.

Vygenerování souboru sendmail.cf

Po dokončení úprav konfiguračních souborů pro program **m4** je nyní musíte zpracovat a vytvořit z nich soubor `/etc/mail/sendmail.cf`, kterému rozumí **sendmail**. Tento krok je velmi jednoduchý, jak vidíte z následujícího příkladu:

```
# cd /etc/mail
# m4 /usr/share/sendmail.cf/m4/cf.m4 vstout.uucpsmtp.mc >sendmail.cf
```

Tento příkaz spustí makroprocesor **m4** a předá mu názvy dvou definičních makrosouborů, aby je v zadaném pořadí zpracoval. Prvním z nich je standardní makrošablona dodávaná přímo se **sendmailem**, druhým je samozřejmě námi vytvořený soubor. Výstup příkazu směrujeme do souboru `/etc/mail/sendmail.cf`, což je požadovaný cílový soubor.

Nyní můžete spustit **sendmail** s novou konfigurací.

Interpretace a vytváření přepisovacích pravidel

Bezesporu nejmocnější funkcí programu **sendmail** jsou přepisovací pravidla. Přepisovací pravidla používá **sendmail** ke zjištění, jak zpracovat přijaté zprávy. **Sendmail** předává adresy z hlaviček zprávy přes soubory přepisovacích pravidel, takzvané *sady pravidel*. Přepisovací pravidlo transformuje poštovní adresu z jednoho tvaru do jiného a můžete si je představit jako příkaz editoru, který nahrazuje text vyhovující nějaké podmínce jiným textem.

Každé pravidlo má levou a pravou stranu, které jsou odděleny alespoň jedním tabulátorem. Když **sendmail** zpracovává poštu, prohlíží přepisovací pravidla a hledá shodu s levou stranou. Pokud adrese vyhovuje levá straně pravidla, nahradí se pravou stranou a zpracovává se znovu.

Příkazy R a S

V souboru `sendmail.cf` se sady pravidel definují příkazem `S`*n*, kde *n* nastavuje číslo aktuální sady pravidel.

Samotná pravidla se definují příkazem `R`. Každý příkaz `R` se po svém přečtení přidá do aktuální sady pravidel.

Pokud pracujete čistě se souborem `sendmail.mc`, nemusíte se o příkazy `S` starat, makra je vytvoří za vás. Ručně musíte vytvořit pouze `R` příkazy.

Sada pravidel programu **sendmail** tedy vypadá takto:

```
Sn
Rlhs rhs
Rlhs2 rhs2
```

Některá užitečná makra

Sendmail interně používá některá standardní makra. V rámci přepisovacích pravidel jsou nejužitečnější tato:

$j	Plně kvalifikované doménové jméno tohoto hostitele.
$w	Hostitelská část plně kvalifikovaného doménového jména hostitele.
$m	Doménová část plně kvalifikovaného doménového jména hostitele.

Tato makra můžeme použít v přepisovacích pravidlech. Konkrétně v pravidlech pro virtuální pivovar jsme použili makro $m.

Levá strana

Na levé straně přepisovacího pravidla definujete vzor, kterému má vyhovovat adresa, jíž chcete transformovat. Většina znaků se musí shodovat přesně, existují však i znaky se speciálním významem, které popisuje následující přehled. Na levé straně přepisovacích pravidel je možné použít následující symboly:

`$@` Vyhovuje právě žádný token.

`$*` Vyhovuje žádný nebo více tokenů.

`$+` Vyhovuje jeden nebo více tokenů.

`$-` Vyhovuje právě jeden token

`$=`*x* Vyhovuje jakákoliv fráze třídy *x*.

`$~`*x* Vyhovuje jakékoliv slovo mimo třídu *x*.

Token je řetězec znaků oddělených mezerou. Neexistuje žádná možnost, aby mezera byla součástí tokenu a není to ani nutné, protože výrazové šablony jsou natolik pružné, že umožňují tuto potřebu obejít. Když adresa pravidlu vyhovuje, text vyhovující jednotlivým částem šablony bude přiřazen do speciálních proměnných, které je možné použít na pravé straně pravidla. Jedinou výjimkou je výraz `$@`, kterému nevyhovuje žádný token, a proto nikdy negeneruje text, který by bylo možné použít na pravé straně pravidla.

Pravá strana

Když adresa vyhovuje levé straně pravidla, původní text se vymaže a nahradí se pravou stranou pravidla. Všechny tokeny na pravé straně se zkopírují přesně, výjimkou jsou ty začínající symbolem dolaru. Stejně jako na levé straně je možné i na pravé straně použít celou řadu metasymbolů. Popisuje je následující seznam. Pravá strana pravidel může obsahovat symboly:

`$`*n* Tento metasymbol se nahradí *n*tým výrazem z levé strany.

`$[`*název*`$]` Tento metasymbol převede název hostitele na kanonické jméno. Nahrazuje se kanonickým názvem zadaného názvu hostitele.

`$[`*mapa klíč* `$@`*parametry* `$:`*implicitní* `$]`

Obecnější tvar vyhledání. Výstupem je výsledek hledání *klíče* v *mapě* s použitím *parametrů*. *Mapa* může být jakákoliv mapa podporovaná **sendmailem**, například virtusertable, o které budeme hovořit za chvíli. Pokud vyhledání není úspěšné, výstup bude *implicitní*. Pokud vyhledání není úspěšné a není nastavena implicitní hodnota, vstup se nezmění a výstupem bude *klíč*.

`$>`*n* Tento symbol způsobí zpracování zbytku řádku pravidly *n*té sady. Výstup volané sady pravidel bude použit jako výstup tohoto pravidla. Tento mechanismus umožňuje pravidlům volat další sady pravidel.

`$#mailer` Tento metasymbol způsobí ukončení vyhodnocování sady pravidel a definuje mailer, který má být použit k doručení zprávy v dalším kroku. Tento metasymbol by měl být použit pouze v sadě *0* nebo v nějaké její subrutině. Jedná se o závěrečnou fázi zpracování adresy a měly by za ním být uvedeny následující dva metasymboly.

`$@hostitel` Tento metasymbol specifikuje hostitele, na něhož bude zpráva předána. Pokud je cílem místní hostitel, nemusí být tento symbol uveden. Hodnota *hostitel* může být dvojtečkami oddělený seznam cílových hostitelů, které budou v zapsaném pořadí vyzkoušeny k odeslání zprávy.

`&:uživatel` Tento metasymbol specifikuje cílového adresáta zprávy.
Pokud adresa pravidlu vyhovuje, za normálních okolností se vzniklý výstup znovu testuje proti stejnému pravidlu a v případě shody se transformace opakuje tak dlouho, dokud výstup pravidlu nevyhovuje. Teprve pak se provádí zpracování následujícím pravidlem v sadě. Toto chování je možné změnit tak, že na začátku pravé strany uvedeme jeden ze dvou speciálních metasymbolů pravé strany, které jsou popsány v následující tabulce. Metasymboly pro řízení smyček jsou:

$@ Tento symbol způsobí, že zbytek pravé strany bude předán jako výstup celé sady pravidel. Nezpracovává se už žádné další pravidlo v sadě.

$: Tento metasymbol způsobí okamžité ukončení pravidla, vyhodnotí se však následující pravidla v sadě.

Příklad jednoduchého pravidla

Abychom lépe pochopili jak fungují substituce v adresách, podívejme se nejprve na příklad levé strany pravidla:

```
$* < $+ >
```

Tomuto pravidlu vyhovuje „žádný nebo více tokenů následovaný symbolem <, následovaný jedním nebo více tokeny a ukončený znakem >“.

Pokud bychom pravidlo použili na adresy brewer@vbrew.com nebo Head Brewer < >, nebudou pravidlu vyhovovat. První adresa nevyhovuje, protože neobsahuje znak <, druhá nevyhovuje, protože symbol $+ představuje *jeden nebo více* tokenů a mezi znaky < a > v adrese není žádný token. Když adresa pravidlu nevyhovuje, pravá strana pravidla se neuplatní.

Pokud bychom pravidlo použili na adresu Head Brewer < brewer@vbrew.com >, adresa vyhovuje a na pravé straně bude symbol $1 obsahovat text Head Brewer a symbol $2 bude obsahovat brewer@vbrew.com.

Pokud bychom pravidlo použili na adresu < brewer@vbrew.com >, bude vyhovovat rovněž, protože symbolu $* vyhovuje *žádný nebo více tokenů* a symbol $1 na pravé straně bude prázdný řetězec.

Sémantika sad pravidel

Každá ze sad pravidel se volá k provedení jiné fáze zpracování pošty. Při vytváření pravidel je důležité chápat, co má která sada pravidel dělat. Podíváme se na jednotlivé sady pravidel, které nám konfigurační skripty programu **m4** umožňují měnit.

`LOCAL_RULE_3` Sada 3 zodpovídá za konverzi adresy z libovolného formátu do formátu, který **sendmail** dále zpracovává. Výstupní formát sady má známý tvar *lokální_část@hostitel_a_doména*. Sada 3 by měla umístit hostitelskou část konvertované adresy mezi symboly < a >, čímž se usnadní zpracování dalšími sadami. Sada 3 se používá jako první v sekvenci zpracování adres, takže pokud **sendmail** používáte k předávání pošty pro nějaký systém, který používá atypický formát

adres, měli byste pomocí makra LOCAL_RULE_3 doplnit pravidlo, které tento formát zkonvertuje do normálního formátu.

`LOCAL_RULE_0 a LOCAL_NET_CONFIG`

Sada 0 se uplatní na adresu příjemce po zpracování adresy sadou 3. Makro LOCAL_NET_CONFIG způsobí vložení pravidel do *spodní poloviny* sady 0. Sada 0 zodpovídá za doručení zprávy příjemce, takže musí vygenerovat trojici mailer, hostitel a uživatel. Pravidla se uplatní před případnou definicí „chytrého" hostitele, takže pokud přidáte pravidla zajišťující správný převod adres, adresy vyhovující těmto pravidlům nebudou obslouženy tímto hostitelem. Tímto mechanismem jsme zajistili přímé doručení protokolem smtp pro hostitele v naší lokální síti.

`LOCAL_RULE_1 a LOCAL_RULE_2`

Sada 1 se uplatní na všechny adresy odesilatele, sada 2 na všechny adresy příjemce. Typicky jsou obě prázdné.

Interpretace pravidla v našem příkladu

V příkladu 14.3 jsme použili makro `LOCAL_NET_CONFIG` k vytvoření pravidla, které zajistí, že jakákoliv pošta pro naši doménu bude doručena přímo prostřednictvím maileru **smtp**. Nyní víme, jak se pravidla vytvářejí, takže si můžeme vysvětlit, jak pravidlo funguje. Podívejme se na ně znovu.

Příklad 14.3 – Přepisovací pravidlo z vstout.uucpsmtp.m4

```
LOCAL_NET_CONFIG
# Toto pravidlo zajišťuje, že všechna lokální pošta se posílá přes
# smtp, ostatní pošta jde přes předávacího hostitele.
R$* < @ $* .$m. > $*  $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
```

Víme už, že makro LOCAL_NET_CONFIG způsobí přidání pravidla někam na konec sady 0, nicméně před definici předávacího hostitele. Víme také, že sada 0 je poslední prováděnou sadou a že jejím výsledkem by měla být trojice údajů definující mailer, uživatele a hostitele.

Můžeme samozřejmě ignorovat oba komentářové řádky, ty nemají na nic vliv. Samotné pravidlo je řádek začínající znakem R. Víme už, že R je příkaz **sendmailu** a přidává pravidlo do aktuální sady pravidel, což je v našem případě sada 0. Podívejme se nejprve na levou stranu a pak na pravou stranu pravidla.

Levá strana vypadá takto: `$* < @ $* .$m. > $*`

Sada 0 předpokládá existenci znaků `< a >`, které doplní sada 3. Sada 3 konvertuje adresy do běžného formátu a aby usnadnila další zpracování, umístí hostitelskou část adresy mezi znaky `< a >`.

Tomuto pravidlu bude vyhovovat jakákoliv adresa ve formátu typu `CílovýUživatel < @ nějakýhostitel.našedoména. > nějaký text`. Vyhovují mu tedy zprávy pro jakéhokoliv uživatele v naší doméně.

Připomeňme si, že text vyhovující metasymbolům na levé straně pravidla je přiřazen metasymbolům pro použití na pravé straně. V našem případě bude prvnímu symbolu `$*` vyhovovat všechno po znak <. Veškerý tento text bude přiřazen symbolu `$1` na pravé straně. Další část textu (mezi `< a >`) vyhovuje druhému symbolu `$*` a na pravé straně bude uložena v `$2`. Podobně poslední část adresy bude uložena v `$3`.

Levá strana pravidla by měla být jasná. Tomuto pravidlu vyhovuje pošta pro jakéhokoliv uživatele na jakémkoliv počítači v naší doméně. Uživatel bude uložen v `$1`, hostitel v `$2` a případný další text v `$3`. Pak se volá pravá strana pravidla, která adresu zpracuje.

Nejprve se podívejme na to, co bychom měli na výstupu dostat. Pravá strana pravidla vypadá takto: `$#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3`.

Při zpracování pravé strany se interpretují jednotlivé metasymboly a provádějí se příslušné substituce.

Symbol `$#` definuje mailer, v našem případě *smtp*.

Symbol `$@` definuje cílového hostitele. V našem případě je tento hostitel definován jako `$2.$m.`, což je plně kvalifikované doménové jméno hostitele v naší doméně. Toto FQDN vytvoříme z hostitelské části jména přiřazené symbolu `$2`, ke které připojíme doménové jméno (`$m`).

Symbol `$:` definuje uživatele, kterého rovněž známe z levé strany a máme jej zapsaného v symbolu `$1`. Nakonec zachováme text části <> a případný ukončovací text, přičemž použijeme hodnoty zjištěné v levé části pravidla.

Protože výsledkem pravidla je mailer, předá se mu zpráva k doručení. V našem případě bude zpráva předána cílovému hostiteli k doručení protokolem SMTP.

Konfigurace nastavení programu sendmail

Program **sendmail** má řadu voleb, které umožní upravit provádění různých operací. Je jich velké množství, proto se v následujícím seznamu omezíme pouze na některé nejčastěji používané.

Jednotlivé volby můžete nastavit buď v souboru **m4** (což je vhodnější postup), nebo je můžete uvést přímo v souboru `sendmail.cf`. Pokud budeme například chtít, aby **sendmail** pro každou zprávu spustil novou úlohu, můžeme do konfiguračního souboru `sendmail.mc` přidat následující řádek:

```
define('confSEPARATE_PROC','true')
```

V souboru `sendmail.cf` se vytvoří odpovídající řádek:

```
O ForkEachJob=true
```

Následující seznam představuje běžné volby v souboru sendmail.mc (a jejich ekvivalenty v souboru `sendmail.cf`).

`confMIN_FREE_BLOCKS (MinFreeBlocks)` — Mohou nastat situace, kdy nějaké problémy znemožní okamžité doručení zprávy a ta pak musí být uložena v poštovní frontě. Pokud váš hostitel zpracovává velké objemy pošty, může se fronta rozrůst natolik, že zaplní celý souborový systém, na němž je uložena. Aby se tomu zabránilo, umožňuje **sendmail** nastavit tímto parametrem minimální počet volných diskových bloků, které musí být k dispozici před přijetím zprávy. Tím zajistíte, že **sendmail** nikdy nezaplní celý diskový prostor. (Implicitní hodnota je 100.)

`confME_TOO (MeToo)` — Když se expanduje cílová adresa, například při překladu aliasu, může se stát, že odesilatel se objeví také jako adresát zprávy. Tato volba řídí, zda bude autorovi zprávy poslána její kopie v případě, že se objeví na seznamu příjemců. Platné hodnoty jsou „true“ a „false“. (Implicitní hodnota je false.)

`confMAX_DAEMON_CHILDREN (MaxDaemonChildren)`

Kdykoliv **sendmail** přijme SMTP spojení ze vzdáleného hostitele, vytvoří svou kopii, která příchozí zprávu obslouží. Díky tomu je **sendmail** schopen současně obsluhovat více příchozích spojení. Je to sice užitečné chování, nicméně každá nová kopie zabírá paměť hostitelského systému. Pokud by vzniklo velké množství současných spojení (například zotavení po výpadku nebo kvůli útoku na systém), mohly by démony **sendmail** zabrat veškerou dostupnou paměť. Tato volba umožní nastavit, kolik démonů může být současně spuštěno. Po dosažení limitu budou další spojení odmítána, dokud neskončí nějaké předchozí spojení. (Implicitní hodnota není definována.)

`confSEPARATE_PROC (ForkEachJob)`

Při zpracovávání fronty a odesílání zpráv zpracovává **sendmail** jednu zprávu po druhé. Při zapnutí této volby bude **sendmail** vytvářet svou kopii pro každou doručovanou zprávu. Toto chování je užitečné zejména v případě, že ve frontě se nachází větší počet zpráv například kvůli problémům cílového systému. (Implicitní hodnota je false.)

`confSMTP_LOGIN_MSG (SmtpGreetingMessage)`

Kdykoliv **sendmail** přijme připojení, vypíše uvítací hlášení. Implicitně toto hlášení obsahuje název hostitele, název přenosového agenta, verzi sendmailu, verzi lokální konfigurace a datum. Standard RFC 821 říká, že první část uvítacího hlášení musí být plně kvalifikované doménové jméno hostitele, zbytek může být libovolný. Můžete použít makra programu **sendmail**, která budou při použití expandována. Jediný, kdo toto hlášení uvidí, bude zvědavý správce systému, který se snaží diagnostikovat problémy poštovního systému, nebo příliš zvědaví uživatelé, jež zajímá, jak váš poštovní systém funguje. Jejich námahu můžete odměnit nějakou vtipnou poznámkou, nicméně buďte příjemní. **sendmail** automaticky za první část hlášení doplní slovo ESMTP, které vzdálenému hostiteli signalizuje, že nás systém podporuje protokol ESMTP. (Implicitní hodnota je `$j Sendmail $v/$Z; $b`).

Některá užitečná nastavení sendmailu

Možných konfigurací **sendmailu** je obrovské množství. V dalším textu si popíšeme jenom několik důležitých druhů konfigurace, které mohou být užitečné pro řadu různých instalací.

Uživatelé mohou nastavovat údaj From:

Občas je užitečné přepsat pole From: odchozí zprávy. Řekněme, že máte webový systém pro odesílání pošty. Normálně by odchozí pošta pocházela od uživatele, který vlastní proces webového serveru. Můžeme chtít adresu odesilatele přepsat, aby se pošta tvářila, že pochází od jiného uži-

vatele či adresy našeho systému. **Sendmail** umožňuje definovat, kteří uživatelé systému mají právo tuto hodnotu měnit.

Volba `use_ct_file` zapíná tuto možnost a aktivuje soubor se seznamem jmen, která mají právo měnit adresu odesilatele. Implicitně má toto právo jen malý počet uživatelů (například `root`). Implicitně se pro tyto uživatele používá soubor `/etc/mail/trusted-users` na systémech, které používají adresář `/etc/mail` nebo soubor `/etc/sendmail.ct` na systémech, jež tento adresář nepoužívají. Název a umístění souboru je možné změnit parametrem `confCT_FILE`.

Funkci povolíte tím, že v souboru `sendmail.mc` uvedete `FEATURE(use_ct_file)`.

Poštovní aliasy

Poštovní aliasy jsou mocná funkce, která umožňuje přesměrování pošty do schránek s alternativními názvy uživatelů nebo procesů na cílovém hostiteli. Bývá například obvyklé, že odezva a komentáře týkající se webových stránek se odesílají uživateli *webmaster*. Velmi často na cílovém počítači neexistuje žádný uživatel *webmaster* a jedná se o pouhý alias jiného uživatele systému. Aliasy běžně využívají servery obsluhující poštovní konference, kdy alias směruje poštu serveru konference.

Aliasy se definují v souboru `/etc/aliases`. Při přijetí příchozí zprávy se **sendmail** v tomto souboru dívá, jak se zprávou naložit. Pokud v souboru nalezne záznam odpovídající adresátovi zprávy, přesměruje zprávu tak, jak mu to záznam nařizuje.

Pomocí aliasu je možné provést tři operace:

- Definují odkazy nebo známá jména adresátů, u nichž se pošta směruje na konkrétního uživatele nebo více uživatelů.
- Dokáží spustit program a předat mu zprávu jako vstup.
- Mohou poslat zprávu do souboru.

Aby systém odpovídal RFC standardu, musí mít definovány aliasy Postmaster a MAILER-DAEMON.

Při vytváření aliasů, které spouštějí programy nebo zapisují do souborů, postupujte vždy velice opatrně, protože **sendmail** běží s právy superuživatele.

Podrobnosti týkající se poštovních aliasů najdete na manuálových stránkách `alias(5)`. Příklad souboru `aliases` vidíte na následujícím výpisu:

Příklad 14.4 – Soubor aliases

```
#
# Následující aliasy musí existovat podle RFC.
# Je nutné je směrovat někomu, kdo čte poštu pravidelně
#
postmaster:    root                              # povinný údaj
MAILER-DAEMON: postmaster                        # povinný údaj
#
#
# příklady běžných aliasů
#
usenet:        janet                             # alias osoby
admin:         joe,janet                         # alias pro více lidí
newspak-users: :include:/usr/lib/lists/newspak # čte adresáty ze souboru
changefeed:    |/usr/local/lib/gup              # alias spouštějící program
complaints:    /var/log/complaints               # alias zapisující do souboru
#
```

Kdykoliv upravíte soubor `aliases`, nezapomeňte spustit příkaz:

```
# /usr/bin/newaliases
```

Tento příkaz vygeneruje nové tabulky aliasů, které **sendmail** interně používá. Příkaz `/usr/bin/newaliases` je symbolický odkaz přímo na **sendmail** a takto spuštěný volá **sendmail** následujícím způsobem:

```
# /usr/lib/sendmail -bi
```

Příkaz **newaliases** je jednodušší a pohodlnější způsob, jak toto volání provést.

Použití předávacího hostitele

Občas narazíte na zprávu, kterou nedokážete přímo poslat cílovému systému. Bývá pohodlné mít na síti jeden systém, který se stará o doručování pošty na obtížně dosažitelné cíle namísto řešení, kdy budou tyto operace provádět všechny systémy.

Existuje několik dobrých důvodů, proč ke správě pošty používat jediného hostitele. Usnadníte si administraci tím, že budete mít pouze jediného hostitele s podrobnou konfigurací, který bude umět přenášet poštu na všechny typy systémů včetně UUCP, Usenet a podobně. Všechny ostatní počítače potřebují pouhý jeden poštovní protokol k poslání pošty na tento centrální systém. Hostitelé plnící úlohy takovýchto centrálních směrovacích a předávacích uzlů se označují jako *předávací hostitelé*. Pokud máte předávacího hostitele, který od vás přijímá poštu, můžete mu poslat cokoliv a nestaráte se o to, jak konkrétní zprávy doručit.

Další výhodné použití předávacího hostitele spočívá v řízení přenosu pošty přes firewall. Organizace může mít instalovánu privátní IP síť s neregistrovanými IP adresami. Tato síť může být k Internetu připojena přes firewall. Odeslání a příjem pošty protokolem SMTP u hostitelů v privátní síti není možné, protože tito hostitelé nemohou vytvářet a přijímat přímá spojení s Internetem. Můžeme však mít firewall, který bude zároveň plnit funkci předávacího hostitele[106]. Předávací hostitel na firewallu může navazovat přímá spojení s ostatními hostiteli v privátní síti a s hostiteli na Internetu. Tento hostitel tak může přijmout poštu od kohokoliv a zajistit její odeslání přímo na správného adresáta.

Předávací hostitel se obvykle použije pokud selžou všechny ostatní metody transportu. V případě organizace s privátní sítí je rozumné řešení nastavit počítače tak, aby se nejprve pokusily doručit poštu přímo, a pokud se jim to nepovede, aby použily předávací systém. Tím se předávacímu systému dost odlehčí, protože veškerou interní poštu si budou počítače na privátní síti posílat přímo.

Sendmail umožňuje jednoduché nastavení předávacího hostitele volbou `SMART_HOST`; v konfiguraci pro virtuální pivovar jsme ji použili. Odpovídající část konfiguračního souboru vypadala takto:

```
define('SMART_HOST', 'uucp-new:moria')
LOCAL_NET_CONFIG
# Toto pravidlo zajišťuje, že všechna lokální pošta se posílá přes
# smtp, ostatní pošta jde přes předávacího hostitele.
R$* < @ $* .$m. > $*   $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
```

106 Pozn. překladatele: Toto řešení je sice *možné*, ale je neskutečně *nevýhodné*. Firewall (má-li za něco stát) by měl být systém, na kterém neběží vůbec *nic*, který vůbec *nic* nedělá a jenom si třikrát rozmyslí, jestli přijatý paket taky někam pošle. Spustit na stroji, který ma zajišťovat bezpečnost celé sítě, něco tak nebezpečného jako je sendmail, to už je zbytečné firewall vůbec vytvářet.

Makro SMART_HOST umožňuje definovat hostitele pro předávání veškeré odchozí pošty, kterou nelze doručit přímo, a zároveň umožňuje definovat protokol, který se má pro komunikaci s tímto hostitelem použít.

V našem případě jsme použili přenos uucp-new přes UUCP hostitele *moria*. Pokud bychom chtěli **sendmail** nastavit pro použití SMTP předávacího hostitele, mohli bychom to udělat například takto:

```
define('SMART_HOST', 'mail.isp.net')
```

Nemusíme specifikovat typ přenosu, protože SMTP je implicitní.

Tušíte už, co může dělat makro LOCAL_NET_CONFIG a přepisovací pravidla?

Makro LOCAL_NET_CONFIG umožňuje doplnit přepisovací pravidla programu sendmail, která budou definovat poštu doručovanou přímo lokálním systémem. V našem příkladu jsme použili pravidlo, jemuž vyhovovaly všechny adresy naší domény (.$m.) a přepsali jsme adresu pro přímé doručení SMTP systémem. Tím zajistíme, že veškerá pošta pro hostitele v naší doméně bude předána SMTP maileru a odeslána cílovému hostiteli, aniž bychom použili předávacího hostitele.

Manipulace s nevyžádanou poštou (spam)

Pokud se přihlásíte na nějakou poštovní konferenci, zveřejníte svou adresu na webových stránkách nebo pošlete článek na UseNet, pravděpodobně vám začne docházet nevyžádaná reklamní pošta. Je celkem běžné, že různá individua hledají na Internetu poštovní adresy a zařazují je do seznamů, které pak prodávají společnostem, které hledají různé metody inzerce svého zboží. Tento typ hromadného rozesílání pošty se běžně označuje jako spam.

Free On-line Dictionary of Computing[107] definuje spam (týkající se elektronické pošty) jako:

Nerozlišeně rozesílat velké objemy nevyžádané pošty k inzerci výrobku nebo služby. Spam v tomto smyslu představuje elektronický ekvivalent reklamních poštovních zásilek rozesílaných obyvatelům.

V 90. letech s nárůstem komerčního potenciálu sítě vznikají společnosti, které nabízejí spamming jako „službu“ firmám, které chtějí inzerovat na síti. Provádějí to odesíláním pošty na seznamy adres, usenetové news a poštovní konference. Tyto praktiky vedou spoustu uživatelů k rozhořčeným a agresivním akcím proti společnostem, které je provozují.

Naštěstí **sendmail** obsahuje určité mechanismy, které mohou usnadnit manipulaci s nevyžádanou poštou.

Real-time Blackhole List

Služba Real-time Blackhole List je veřejná služba, jejímž cílem je omezit objemy nevyžádané reklamní pošty. V databázi, kterou je možné z Internetu prohledávat, jsou umístěny adresy a systémy známé rozesíláním spamu. Databázi doplňují uživatelé, kteří nevyžádanou poštu z různých adres přijímají. Někdy se na tomto seznamu ocitnou i dobře známé domény, které zanedbají kontrolu spamových aktivit. I když občas někdo protestuje proti konkrétním záznamům v tomto seznamu, jedná se o velmi populární službu a případné stížnosti bývají řešeny velmi rychle. Podrobnosti o tom, jak služba funguje, je možné najít na domovských stránkách Mail Abuse Protection System na adrese *http://maps.vix.com/rbl/*.

[107] Free On-Line Dictionary of Computing bývá součástí řady distribucí Linuxu a je k dispozici na adrese *http://wombat.doc.ic.ac.uk/foldoc/*.

Pokud v **sendmailu** zapnete příslušnou funkci, bude veškerou příchozí poštu testovat proti tomuto seznamu a rozhodne, zda zprávu přijmout. Pokud provozujete velký systém s řadou uživatelů, můžete tím ušetřit značný diskový prostor. Parametrem služby je název serveru, který má být dotazován. Implicitně se používá hlavní server na adrese *rbl.maps.vix.com*.

Službu Real-time Blackhole List nastavíte následujícím makrem v souboru sendmail.mc:

```
FEATURE(rbl)
```

Pokud budete chtít použít jiný RBL server, můžete zadat makro takto:

```
FEATURE(rbl,'rbl.host.net')
```

Přístupová databáze

Alternativní řešení, které nabízí větší míru kontroly za cenu větších konfiguračních nároků, je funkce access_db programu **sendmail**. Přístupová databáze umožňuje nastavit, od kterých uživatelů nebo hostitelů budete přijímat poštu a pro které budete poštu předávat.

Správné nastavení hostitelů, pro něž předáváte poštu, je velmi důležité, protože se jedná o další techniku běžně používanou spammery k obejití systémů jako jsou Real-time Blackhole List. Namísto přímého odeslání pošty na váš systém ji spammer pošle na nějakého nic netušícího hostitele, který vám ji předá. Příchozí spojení pak nepochází od spammera ale od předávajícího hostitele. Aby nemohl být váš vlastní poštovní systém tímto způsobem zneužit, měli byste poštu předávat pouze pro známé systémy. Verze **sendmail** 8.9.0 a novější mají předávání pošty standardně vypnuto, takže pokud chcete předávání povolit, musíte to nastavit v přístupové databázi.

Základní myšlenka je jednoduchá. Když **sendmail** přijme příchozí spojení, zjistí si informace z hlavičky a pak se podívá do přístupové databáze, zda se má zprávou dále zabývat.

Přístupová databáze je soustava pravidel popisujících, co se má provést pro zprávy odeslané z uvedených hostitelů. Implicitně je přístupová databáze uložena v souboru `/etc/mail/access`. Tato tabulka má jednoduchý formát. Každý řádek obsahuje jedno pravidlo. Může to být úplná poštovní adresa, název hostitele nebo IP adresa. Za tím je uvedeno, co se má provést. Existuje pět typů možných akcí:

`OK`	Přijme zprávu.
`RELAY`	Přijme zprávu od tohoto hostitele nebo uživatele dokonce i v případě, že není určena pro místní systém, umožní tedy předání zprávy na další systém.
`REJECT`	Odmítne zprávu s obecným hlášením.
`DISCARD`	Zruší zprávu mailerem $#discard.
`### text`	Vrátí chybovou zprávu kde *###* je kód chyby (podle RFC 821) a *text* je chybové hlášení.

Příklad souboru `/etc/mail/access` může vypadat takto:

```
friends@cybermail.com   REJECT
aol.com                 REJECT
207.46.131.30           REJECT
postmaster@aol.com      OK
linux.org.au            RELAY
```

Tento příklad odmítá zprávu z adresy *friends@cybermail.com*, od jakéhokoliv systému v doméně *aol.com* a od systému *207.46.131.30*. Další pravidlo povoluje příjem zpráv z adresy *postmas-*

ter@aol.com, i když zbytek domény je zakázán. Poslední pravidlo povoluje předávání pošty všem hostitelům v doméně *linux.org.au*.

Řízení přístupu databází zapnete následujícím příkazem v souboru sendmail.mc:

```
FEATURE(access_db)
```

Při implicitním nastavení se databáze vytváří příkazem hash `-o /etc/mail/access`, který z prostého textového souboru udělá jednoduchou hashovanou databázi. Ve většině případů to naprosto postačuje. Existují i další možnosti, které možná budete chtít použít v případě, že máte velikou přístupovou databázi. Podrobnosti naleznete v knize o **sendmailu** nebo přímo v jeho dokumentaci.

Zabránění uživatelům v příjmu pošty

Pokud máte uživatele nebo automatické procesy, které poštu odesílají, ale nepotřebují ji přijímat, bývá užitečné pro takovéto adresáty zakázat příjem pošty. Tím se šetří diskový prostor od ukládání zpráv, které nikdo nebude číst. Funkce `blacklist_recipients` společně s funkcí `access_db` umožňuje zablokovat příjem pošty pro zvolené lokální uživatele.

Pokud chcete tuto funkci zapnout, musíte v souboru sendmail.mc uvést:

```
FEATURE(access_db)
FEATURE(blacklist_recipients)
```

Pokud chcete pro nějakého uživatele zakázat příjem pošty, uveďte jej v přístupové databázi. Obvykle použijete akci typu `###` s nějakým rozumným chybovým hlášením, aby odesilatel věděl, že pošta nebyla doručena. Tato funkce funguje i pro uživatele ve virtuálních poštovních doménách, v takovém případě uvedete kromě jména uživatele i název virtuální domény. Příklad souboru `/etc/mail/access` může vypadat takto:

```
daemon          550 Daemon does not accept or read mail.
flacco          550 Mail for this user has been administratively disabled.
grump@dairy.org 550 Mail disabled for this recipient.
```

Konfigurace virtuálních poštovních hostitelů

Virtuální poštovní hostitelé umožňují aktivovat funkci, kdy počítač přijímá a doručuje poštu pro řadu různých domén tak, jako kdyby se jednalo o řadu samostatných systémů. Typicky virtuální hostitele nabízejí poskytovatelé internetového připojení v kombinaci s virtuálními webovými servery. Tato funkce se nastavuje poměrně jednoduše a protože nikdy nevíte, kdy budete chtít svého hostitele použít jako virtuálního hostitele pro konferenci týkající se nějakého pěkného linuxového projektu, popíšeme si, jak to udělat.

Příjem pošty pro jiné domény

Když **sendmail** přijme zprávu, porovnává cílového hostitele v hlavičce zprávy s názvem lokálního systému. Pokud si odpovídají, **sendmail** zprávu přijme k lokálnímu doručení. Pokud si neodpovídají, **sendmail** se rozhodne, zda zprávu přijmout a předat ji uvedenému cílovému hostiteli (viz předchozí text *Přístupová databáze*).

Pokud chceme poskytovat virtuální poštovní hostitele, musíme **sendmailu** v první řadě říct, aby přijímal poštu i pro domény, jejichž jsme hostitelem. To se naštěstí provede velmi jednoduše.

Funkce `use_cw_file` umožňuje definovat název souboru, v němž jsou uvedeny domény, pro něž má **sendmail** přijímat poštu. Funkci aktivujete následujícím příkazem v souboru sendmail.mc:

```
FEATURE(use_cw_file)
```

Soubor s hostěnými doménami se standardně jmenuje `/etc/mail/local-host-names` na systémech používajících adresář `/etc/mail` nebo `/etc/sendmail.cw` na ostatních. Název a umístění souboru můžete změnit makrem confCW_FILE takto:

```
define('confCW_FILE', '/etc/virtualnames')
```

Přidržíme-li se standardního souboru a budeme-li chtít obsluhovat poštu domén *bovine.net*, *dairy.org* a *artist.org*, vyvoříme soubor `/etc/mail/local-host-names`, který bude vypadat takto:

```
bovine.net
dairy.org
artist.org
```

Když to uděláme a za předpokladu, že pro dané domény existují DNS záznamy které směrují poštu na náš systém, bude **sendmail** přijímat poštu těchto domén stejně jako by byla určena přímo pro naši doménu.

Předávání pošty virtuálních domén na jejich adresáty

Funkce virtusertable konfiguruje podporu tabulek virtuálních uživatelů, které používáme k obsluze pošty virtuálních domén. Tabulka virtuálních uživatelů mapuje příchozí poštu určenou pro *user@host* na poštu pro *otheruser@otherhost*. Můžete to chápat jako trochu chytřejší alias, který se netýká jen uživatelského jména, ale i doménového jména.

Tabulky virtuálních uživatelů zapnete následujícím příkazem v souboru sendmail.mc:

```
FEATURE(virtusertable)
```

Implicitně se soubor s překladovou tabulkou nachází v `/etc/mail/virtusertable`. Můžete nastavit jiné umístění pomocí parametru makra, podrobnosti o dostupných možnostech najdete v dokumentaci k **sendmailu**.

Formát tabulky virtuálních uživatelů je velmi jednoduchý. Levá strana každého řádku obsahuje pravidlo definující původní adresu, pravá strana pak adresu na níž má být virtuální adresa mapována.

Následující příklad ukazuje tři možné typy položek:

```
samiam@bovine.net      colin
sunny@bovine.net       darkhorse@mystery.net
@dairy.org             mail@jhm.org
@artist.org            $1@red.firefly.com
```

V tomto případě sloužíme jako virtuální hostitel pro domény *bovine.net*, *dairy.org* a *artist.org*.

První záznam směruje poštu uživatele v doméně *bovine.net* na lokálního uživatele našeho systému. Druhý záznam směruje poštu jiného uživatele této domény na úplně jiného uživatele v úplně jiné doméně. Třetí příklad směruje poštu adresovanou komukoliv v doméně *dairy.org* na jednu konkrétní adresu. Konečně poslední příklad směruje poštu jakéhokoliv uživatele v doméně *artist.org* na stejného uživatele v jiné doméně, tedy pošta pro *julie@artist.org* bude poslána uživateli *julie@red.firefly.com*.

Testování konfigurace

Program **m4** zpracuje definiční makrosoubor podle svých vlastních syntaktických pravidel, aniž by cokoliv věděl o syntaxi používané programem **sendmail**. Pokud tedy v makrosouboru něco zadáte špatně, neobjeví se žádné chybové hlášení. Proto je nesmírně důležité, abyste celou konfiguraci pečlivě otestovali. Naštěstí **sendmail** nabízí poměrně jednoduchý způsob, jak to udělat.

Sendmail podporuje režim „testování adres“, který umožňuje otestovat konfiguraci a odhalit chyby. V tomto režimu spouštíme **sendmail** z příkazové řádky a on se ptá na sadu pravidel a na cílovou adresu. Pak zpracuje zadanou adresu podle zvolených pravidel a postupně vypíše výstup jednotlivých použitých pravidel. V tomto režimu **sendmail** spustíte s parametrem `-bt`:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
>
```

Implicitní konfigurační soubor se nazývá **sendmail.cf**, alternativní soubor můžete otestovat pomocí volby `-C`. Abychom otestovali naši konfiguraci, potřebujeme zadat řadu různých adres, na kterých uvidíme, že jsou zpracovány tak, jak potřebujeme. Budeme si to demonstrovat na složitější konfiguraci používající UUCP podle příkladu 14.2.

Nejprve otestujme, zda je **sendmail** schopen doručit poštu lokálnímu uživateli. V následujících testech předpokládáme, že všechny adresy budou přepsány na mailer *local* na lokálním systému:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac
rewrite: ruleset   3   input: isaac
rewrite: ruleset  96   input: isaac
rewrite: ruleset  96 returns: isaac
rewrite: ruleset   3 returns: isaac
rewrite: ruleset   0   input: isaac
rewrite: ruleset 199   input: isaac
rewrite: ruleset 199 returns: isaac
rewrite: ruleset  98   input: isaac
rewrite: ruleset  98 returns: isaac
rewrite: ruleset 198   input: isaac
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Výstup ukazuje, jak program **sendmail** vnitřně zpracovává adresu *isaac*. Každý řádek ukazuje, co se předává sadě pravidel a výsledek zpracování hodnoty sadou. Říkáme programu, že chceme adresu zpracovat sadami 3 a 0. Sada 0 se testuje standardně, sadu 3 jsme si vynutili, protože ta se standardně netestuje. Na posledním řádku vidíme, že výsledkem sady 0 je doručení zprávy na adresu *isaac* mailerem *local*.

Dál otestujeme poštu adresovanou na naši SMTP adresu *isacc@vstout.vbrew.com*. Měli bychom dostat stejný výsledek jako v předchozím případě:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vstout.vbrew.com
rewrite: ruleset   3   input: isaac @ vstout . vbrew . com
```

```
rewrite: ruleset  96   input: isaac < @ vstout . vbrew . com >
rewrite: ruleset  96 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Test dopadl úspěšně. Teď vyzkoušíme naši UUCP adresu *vstout!isacc*.

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 vstout!isaac
rewrite: ruleset   3   input: vstout ! isaac
rewrite: ruleset  96   input: isaac < @ vstout . UUCP >
rewrite: ruleset  96 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Test rovněž dopadl správně. Těmito testy jsme ověřili, že pošta určená uživateli lokálního systému bude doručena bez ohledu na to, jak mu byla adresována. Pokud jste pro svůj systém definovali jakékoliv aliasy, virtuální hostitele a podobně, měli byste testy vyzkoušet pro všechna možná jména, pod nimiž je hostitel znám, abyste ověřili, že vše funguje správně.

Dále si ověříme, že pošta adresovaná jiným systémům v doméně *vbrew.com* bude doručena přímo danému systému protokolem SMTP:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vale.vbrew.com
rewrite: ruleset   3   input: isaac @ vale . vbrew . com
rewrite: ruleset  96   input: isaac < @ vale . vbrew . com >
rewrite: ruleset  96 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 198 returns: $# smtp $@ vale . vbrew . com . /
```

```
    $: isaac < @ vale . vbrew . com . >
rewrite: ruleset   0 returns: $# smtp $@ vale . vbrew . com . /
    $: isaac < @ vale . vbrew . com . >
```

Vidíme, že test nařizuje předání pošty mailerem SMTP na hostitele *vale.vbrew.com* jeho uživateli *isacc*. Potvrdili jsme si, že definice v makru LOCAL_NET_CONFIG funguje správně. Aby test dopadl úspěšně, musí být možné provést překlad cílového jména, tedy buď musíme mít příslušný záznam v `/etc/hosts` nebo v DNS. Co se stane, pokud jméno nepůjde převést, si vyzkoušíme tak, že záměrně zadáme neplatné jméno hostitele:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vXXXX.vbrew.com
rewrite: ruleset   3   input: isaac @ vXXXX . vbrew . com
rewrite: ruleset  96   input: isaac < @ vXXXX . vbrew . com >
vXXXX.vbrew.com: Name server timeout
rewrite: ruleset  96 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset   3 returns: isaac < @ vXXXX . vbrew . com >
== Ruleset 3,0 (3) status 75
rewrite: ruleset   0   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 199   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 199 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  98   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  98 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 198   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  95   input: < uucp-new : moria > isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset  95 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset 198 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset   0 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
```

Výsledek je dost odlišný. Nejprve sada 3 vrací chybu udávající, že název nelze převést. S touto situací se vyrovnáme díky dalšímu klíčovému prvku naší konfigurace, kterým je předávací hostitel. Smyslem předávacího hostitele je doručit cokoliv, co neumíme doručit jinak. Jméno hostitele, které jsme v příkladu zadali, nebylo možno převést a pravidla rozhodla, že pošta bude předána našemu předávacímu hostiteli *moria* mailerem *uucp-new*. Ten může být připojen lépe a třeba bude vědět, co s poštou udělat.

Poslední test ověří, že pošta adresovaná mimo naši doménu bude předána předávacímu hostiteli. Výsledek by měl být podobný předchozímu testu:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@linux.org.au
rewrite: ruleset   3   input: isaac @ linux . org . au
rewrite: ruleset  96   input: isaac < @ linux . org . au >
rewrite: ruleset  96 returns: isaac < @ linux . org . au . >
rewrite: ruleset   3 returns: isaac < @ linux . org . au . >
rewrite: ruleset   0   input: isaac < @ linux . org . au . >
rewrite: ruleset 199   input: isaac < @ linux . org . au . >
```

```
rewrite: ruleset 199 returns: isaac < @ linux . org . au . >
rewrite: ruleset  98   input: isaac < @ linux . org . au . >
rewrite: ruleset  98 returns: isaac < @ linux . org . au . >
rewrite: ruleset 198   input: isaac < @ linux . org . au . >
rewrite: ruleset  95   input: < uucp-new : moria > isaac </
    @ linux . org . au . >
rewrite: ruleset  95 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
rewrite: ruleset 198 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
rewrite: ruleset   0 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
```

Z výsledku testu vidíme, že název hostitele byl převeden a že zpráva bude předána předávacímu hostiteli. Definice LOCAL_NET_CONFIG tedy funguje správně a v obou případech se chová jak má. Celý test proběhl dobře, takže věříme, že konfigurace je správná a můžeme ji použít.

Spuštění programu sendmail

Démon **sendmail** může být spuštěn dvěma způsoby. Jedna možnost je spouštět jej prostřednictvím démona **inetd**, druhá a obvyklejší je spouštět jej samostatně. Je také obvyklé, že poštovní programy spouštějí **sendmail** jako uživatelský příkaz k doručení nově napsané pošty.

Při samostatném spouštění programu **sendmail** jej uveďte v souboru `rc`, aby se spouštěl automaticky při startu systému. Běžně se používá následující syntaxe:

```
/usr/sbin/sendmail -bd -q10m
```

Parametr `-bd` říká programu, aby se spustil jako démon. Vytvoří svou kopii a poběží v pozadí. Parametr `-q10m` říká, aby testoval frontu každých 10 minut. Můžete nastavit jiný testovací interval.

Pokud chcete **sendmail** spouštět démonem **inetd**, použijete takovýto záznam:

```
smtp  stream  tcp nowait  nobody  /usr/sbin/sendmail -bs
```

Parametr –bs říká programu, aby na standardním vstupu a výstupu používal protokol SMTP, což je podmínka pro použití s démonem **inetd**.

Příkaz **runq** je obvykle symbolický odkaz na soubor **sendmail** a představuje jednodušší variantu příkazu:

```
# sendmail -q
```

Po spuštění tímto způsobem zpracuje **sendmail** poštu ve frontě. Pokud **sendmail** spouštíte démonem **inetd**, musíte vytvořit také úlohu pro démona **cron**, která bude pravidelně spouštět příkaz **runq**, aby bylo zajištěno pravidelné odesílání pošty z fronty.

Záznam v tabulce démona **cron** může vypadat takto:

```
# Výběr poštovní fronty každých 15 minut
0,15,30,45     *     *     *     *     /usr/bin/runq
```

Na většině instalací se **sendmail** každých 15 minut spouští a odešle všechnu poštu z fronty přesně tak, jako by to dělal při výše uvedeném nastavení.

Tipy a triky

Existuje celá řada postupů jak zefektivnit práci s programem **sendmail**. Balík tohoto programu obsahuje řadu administrativních nástrojů, některé z nich si krátce představíme.

Správa poštovní fronty

Pošta k odeslání se ukládá v adresáři `/var/spool/mqueue`. Tento adresář představuje poštovní frontu. Program **sendmail** umožňuje zobrazit seznam úloh ve frontě a jejich status.

Příkaz **/usr/bin/mailq** je symbolický odkaz na **sendmail** a chová se stejně jako

```
# sendmail -bp
```

Výstup zobrazí identifikátor zprávy, její velikost, čas kdy byla zařazena do fronty, kdo ji odesílá a status zprávy. Příklad ukazuje zprávu, která uvízla ve frontě kvůli problémům:

```
$ mailq
                Mail Queue (1 request)
--Q-ID-- --Size-- -----Q-Time----- ------------Sender/Recipient------------
RAA00275      124 Wed Dec  9 17:47 root
                 (host map: lookup (tao.linux.org.au): deferred)
                                   terry@tao.linux.org.au
```

Zpráva je stále ve frontě, protože nelze zjistit IP adresu cílového hostitele.

Okamžité zpracování fronty můžeme nařídit příkazem `/usr/bin/runq`.

Příkaz `runq` nemá žádný výstup, `sendmail` začne na pozadí zpracovávat poštovní frontu.

Okamžité zpracování fronty vzdáleným hostitelem

Pokud používáte občasné připojení k Internetu s *pevnou* IP adresou a používáte nějakého hostitele k ukládání vaší pošty po dobu, kdy nejste připojeni, určitě uvítáte možnost přinutit tohoto hostitele, aby vám vaši poštu předal hned poté, co se k Internetu připojíte.

Distribuce programu **sendmail** obsahuje krátký skript v Perlu, který tuto operaci usnadňuje u hostitelů, jež ji podporují. Skript **etrn** má na vzdáleném hostiteli podobný efekt jako příkaz **runq** na lokálním hostiteli. Pokud zadáme následující příkaz:

```
# etrn vstout.vbrew.com
```

donutíme tak hostitele *`vstout.vbrew.com`*, aby zpracoval zprávy ve frontě určené pro náš systém.

Typicky tento příkaz přidáte do PPP skriptu `ip-up`, takže se provede hned poté, co se naváže síťové spojení.

Analýza poštovních statistik

Sendmail shromažďuje informace o přenesené poště a o hostitelích, jimž poštu doručoval. Tyto informace se zobrazí pomocí dvou příkazů: **mailstats** a **hoststat**.

```
mailstats
```

Příkaz **mailstats** zobrazí statistiky týkající se objemu přenesené pošty. Nejprve se vytiskne čas, v němž byla statistika vytvořena a pak následuje tabulka s jedním řádkem pro každý používaný mailer a s řádkem celkové statistiky. Každý řádek obsahuje osm údajů:

Údaj	Význam
M	Číslo maileru (transportního protokolu).
msgsfr	Počet zpráv přijatých mailerem.
bytes_from	Počet KB přijatých mailerem.
msgto	Počet zpráv odeslaných mailerem.
bytes_to	Počet KB odeslaných mailerem.
msgsrej	Počet odmítnutých zpráv.
msgsdis	Počet zrušených zpráv.
Mailer	Název maileru.

Příklad výstupu příkazu **mailstats** ukazuje výpis 14.5.

Příklad 14.5 – Výstup programu mailstats

```
# /usr/sbin/mailstats
Statistics from Sun Dec 20 22:47:02 1998
 M   msgsfr  bytes_from   msgsto    bytes_to  msgsrej msgsdis  Mailer
 0        0          0K       19        515K        0       0  prog
 3       33        545K        0          0K        0       0  local
 5       88        972K      139       1018K        0       0  esmtp
=============================================================
 T      121       1517K      158       1533K        0       0
```

Tato data se shromažďují, pokud je v souboru sendmail.cf povolena volba StatusFile a pokud stavový soubor existuje. Typicky uvedete v souboru sendmail.cf:

```
# stavový soubor
O StatusFile=/var/log/sendmail.st
K vynulování statistik musíte stavovému souboru nastavit nulovou délku:
> /var/log/sendmail.st
a restartovat sendmail.
```

hoststat

Příkaz **hoststat** vypisuje informace o hostitelích, jimž se **sendmail** pokusil doručit poštu. Příkaz **hoststat** je ekvivalentní následujícímu příkazu:

```
sendmail -bh
```

Výstup obsahuje na každém řádku jednoho hostitele a u každého pak čas, kdy se spojení uskutečnilo a status zprávy.

Následující příklad ukazuje, jak může výstup programu **hoststat** vypadat. Většina položek obsahuje úspěšně doručené zprávy. Výsledek u hostitele *earthlink.net* naopak upozorňuje na neúspěch. Stavový text občas napomůže ke zjištění, proč k chybě došlo. V tomto případě došlo k timeoutu spojení, pravděpodobně protože hostitel byl nedostupný.

Příklad 14.6 – Výstup programu hoststat

```
# hoststat
 -------------- Hostname ---------- How long ago ---------Results---------
 mail.telstra.com.au                   04:05:41 250 Message accepted for
 scooter.eye-net.com.au              81+08:32:42 250 OK id=0zTGai-0008S9-0
```

```
yarrina.connect.com.a              53+10:46:03 250 LAA09163 Message acce
happy.optus.com.au                 55+03:34:40 250 Mail accepted
mail.zip.com.au                       04:05:33 250 RAA23904 Message acce
kwanon.research.canon.com.au       44+04:39:10 250 ok 911542267 qp 21186
linux.org.au                       83+10:04:11 250 IAA31139 Message acce
albert.aapra.org.au                   00:00:12 250 VAA21968 Message acce
field.medicine.adelaide.edu.au     53+10:46:03 250 ok 910742814 qp 721
copper.fuller.net                  65+12:38:00 250 OAA14470 Message acce
amsat.org                           5+06:49:21 250 UAA07526 Message acce
mail.acm.org                       53+10:46:17 250 TAA25012 Message acce
extmail.bigpond.com                11+04:06:20 250 ok
earthlink.net                      45+05:41:09 Deferred: Connection time
```

Příkazem **purgestat** se vymažou schromážděná data a jeho ekvivalentem je příkaz:

```
# sendmail -bH
```

Statistiky se rozrůstají dokud je nesmažete. Můžete je mazat periodickým spouštěním příkazu purgestat, abyste mohli snáze najít novější záznamy, zejména pokud je váš systém hodně zatížený. Tento příkaz můžete uvést v tabulce démona cron, nebo jej můžete čas od času spouštět ručně.

Nastavení a spuštění programu Exim

V této kapitole uvádíme stručný přehled jak nastavit a provozovat program Exim. I když po stránce chování je Exim dobře kompatibilní se **sendmailem**, jejich konfigurační soubory se značně liší.

Konfigurační soubor se na většině distribucí Linuxu typicky jmenuje `/etc/exim.conf` nebo `/etc/exim/config`, popřípadě `/usr/lib/exim/config` na starších distribucích. Kde je konfigurační soubor uložen, zjistíte příkazem:

```
$ exim -bP configure_file
```

Konfigurační soubor bude potřeba upravit, aby obsahoval hodnoty specifické pro váš systém. U většiny typických konfigurací nejsou změny obtížné a jednou nastavená funkční konfigurace se obvykle nemění.

Implicitně Exim veškerou došlou poštu zpracovává a doručuje okamžitě. Pokud máte velký provoz, můžete Exim nastavit tak, aby všechny zprávy řadil do fronty a zpracovával je pouze v pravidelných intervalech.

Při obsluze pošty na síti TCP/IP běží Exim často jako démon, při spuštění systému se spouští příkazem `/etc/init.d/exim`[108]. Po spuštění se přepne na pozadí, kde čeká na příchozí spojení na portu SMTP (obvykle port 25). Je to výhodné zejména pokud očekáváte větší provoz, protože Exim nemusíte spouštět pro každé příchozí spojení. Alternativně je možné svěřit správu portu SMTP programu **inetd**, který bude spouštět Exim vždy při přijetí spojení na tomto portu. Tato konfigurace je výhodná, pokud máte málo paměti a malý poštovní provoz.

Exim má komplikovanou soustavu řádkových parametrů, z nichž se řada shoduje se sendmailem. Program ovšem nemusíte spouštět se soustavou parametrů tak, aby vyhovoval tomu, co potřebujete udělat, nejběžnější operace můžete implementovat pomocí tradičních příkazů jako jsou **rmail** a **rsmtp**. Jedná se o symbolické odkazy na Exim (a pokud ne, můžete je tak nastavit). Když spustíte některý z těchto příkazů, Exim si zjistí, pod jakým názvem byl spuštěn a sám si nastaví správné parametry.

Dva odkazy na Exim by měly existovat ve všech případech: `/usr/bin/rmail` a `/usr/sbin/sendmail`[109]. Když napíšete a odešlete zprávu pomocí uživatelského agenta jako je například **elm**, pře-

108 Další možná umístění jsou `/etc/rc.d/init.d` a `rc.inet2`. Poslední metoda je obvyklá na systémech používajících strukturu souborů v adresáři `/etc` ve stylu BSD systémů.

109 Jedná se o nové umístění programu **sendmail** podle standardu Linux File System Standard. Dalším obvyklým umístěním je `/usr/lib/sendmail`, které obvykle používají poštovní programy, jež nebyly nastaveny speciálně pro Linux. Oba soubory můžete definovat jako symbolické odkazy na Exim, takže programy a skripty spouštějící sendmail spustí se stejným efektem Exim.

dává se zpráva k doručení rourou programům **sendmail** nebo **rmail**, proto by oba měly ukazovat na Exim. Seznam adresátů zprávy se Eximu předává na příkazovém řádku[110]. To samé platí pro poštu přicházející protokolem UUCP. Potřebné příkazy můžete nastavit tak, aby ukazovaly na Exim následujícím postupem:

```
$ ln -s /usr/sbin/exim /usr/bin/rmail
$ ln -s /usr/sbin/exim /usr/sbin/sendmail
```

Pokud se budete chtít zabývat konfigurací programu Exim podrobněji, měli byste si prostudovat jeho popis. Pokud není součástí vaší distribuce Linuxu, najdete jej ve zdrojových kódech programu Exim, nebo si jej můžete přečíst na webových stránkách *http://www.exim.org*.

Spuštění programu Exim

Před spuštěním programu Exim se musíte rozhodnout, zda má obsluhovat příchozí SMTP spojení samostatným démonem, nebo zda má být port SMTP spravován démonem **inetd**, který spustí Exim pouze tehdy, když si nějaký klient SMTP spojení vyžádá. Obvykle se dává přednost spuštění samostatného démona, protože se tím poštovní server zatěžuje daleko méně, než opakovaným vytvářením nových procesů pro každé příchozí spojení. Protože poštovní server navíc doručuje většinu příchozí pošty přímo uživatelům, na ostatních počítačích byste měli zvolit spouštění démonem **inetd**.

Ať už zvolíte kterýkoliv z režimů operace, měli byste zkontrolovat, že soubor `/etc/services` obsahuje následující řádek:

```
smtp            25/tcp          # Simple Mail Transfer Protocol
```

Tím je definováno, který TCP port se používá pro SMTP konverzaci. Port s číslem 25 je standard definovaný RFC dokumentem „Assigned Numbers“ (RFC 1700).

Když Exim běží v režimu démona, přepne se na pozadí a čeká na spojení na portu SMTP. Když nastane spojení, rozdvojí se a vzniklý synovský proces ošetří komunikaci s partnerským procesem na klientském počítači. Démon Exim se obvykle spouští z rc skriptu při startu systému takto:

```
/usr/sbin/exim -bd -q15m
```

Parametr `-bd` jej spouští v režimu démona, parametrem `-q15m` se programu nařizuje zpracovat eventuální poštu ve frontě vždy každých 15 minut.

Pokud chcete raději použít démona **inetd**, musí soubor `/etc/inetd.conf` obsahovat řádek jako

```
smtp    stream  tcp nowait  root  /usr/sbin/exim  in.exim -bs
```

Nezapomeňte, že po provedení změn musíte démona **inetd** donutit znovu načíst konfiguraci tím, že mu pošlete signál HUP[111].

Režim démona a spouštění pomocí **inetd** se navzájem vylučují. Pokud spouštíte Exim jako démona, měli byste zkontrolovat, že v souboru inetd.conf není definován žádný řádek pro službu SMTP. Naopak pokud spouštíte Exim prostřednictvím démona **inetd**, zkontrolujte, že se nespouští přímo v žádném rc skriptu.

Že je Exim správně nastaven k příjmu příchozích SMTP spojení si můžete ověřit telnetem na port SMTP. Takto nějak bude vypadat úspěšné připojení k Eximu:

---

[110] Někteří uživatelští agenti nicméně předávají zprávu transportnímu agentu protokolem SMTP, který pak volají s parametry `-bs`.

[111] Použijte příkaz `kill HUP` *`pid`*, kde *pid* je číslo procesu **inetd**, které zjistíte příkazem `ps`.

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 richard.vbrew.com ESMTP Exim 3.13 #1 Sun, 30 Jan 2000 16:23:55 +0600
quit
221 richard.brew.com closing connection
Connection closed by foreign host.
```

Pokud se v tomto textu neobjeví vítací hlášení protokolu SMTP (řádek začínající číslem 220), zkontrolujte, zda existuje proces démona Exim, nebo zda je **inetd** správně nakonfigurován. Pokud bude všechno v pořádku, podívejte se do logovacích souborů programu Exim (budeme o nich hovořit za chvíli), abyste zjistili, zda chyba není v konfiguraci Eximu.

Když pošta nefunguje

K řešení instalačních potíží existuje celá řada možností. První z nich je kontrola logovacích souborů programu. Na linuxových systémech se typicky nacházejí v adresáři `/var/log/exim/log` a jmenují se `exim_mainlog`, `exim_rejectlog` a `exim_paniclog`. Na jiných operačních systémech bývají často v adresáři `/var/spool/exim/log`. Kde se logovací soubory nacházejí, můžete zjistit příkazem:

```
exim -bP log_file_path
```

Hlavní log obsahuje všechny transakce, log zamítných zpráv obsahuje podrobnosti o zprávách, které byly odmítnuty z důvodů nastavených politik. Záznamy paniky se vztahují k chybám v konfiguraci a k podobným událostem.

Typická část hlavního logu je uvedena níže. Každý záznam je tvořen jedním řádkem, který obsahuje datum a čas. Rozdělili jsme jej na několik řádků, aby se vešel na šířku stránky:

```
2000-01-30 15:46:37 12EwYe-0004WO-00 <= jack@vstout.vbrew.com
  H=vstout.vbrew.com [192.168.131.111] U=exim P=esmtp S=32100
  id=38690D72.286F@vstout.vbrew.com
2000-01-30 15:46:37 12EwYe-0004WO-00 => jill <jill@vbrew.com>
  D=localuser T=local_delivery
2000-01-30 15:46:37 12EwYe-0004WO-00 Completed
```

Tento záznam ukazuje, že pošta od *jack@vstout.vbrew.com* pro *jill@vbrew.com* byla úspěšně doručena do schránky na lokálním počítači. Přijetí zprávy ukazuje symbol **<=**, odchod zprávy symbol **=>**.

Existují dva typy chyb doručení: trvalé a dočasné. Trvalá chyba je zaznamenána v následujícím záznamu a označuje se symboly **:

```
2000-01-30 14:48:28 12EvcH-0003rC-00 ** bill@lager.vbrew.com
  R=lookuphost T=smtp: SMTP error from remote mailer after RCPT TO:
  <bill@lager.vbrew.com>: host lager.vbrew.com [192.168.157.2]:
  550 <bill@lager.vbrew.com>... User unknown
```

Po podobné chybě odešle Exim odesilateli zprávu o chybě.

Dočasné chyby jsou označeny symboly **==**.

```
2000-01-30 12:50:50 12E9Un-0004Wq-00 == jim@bitter.vbrew.com
  T=smtp defer (145): Connection timed out
```

Takovéto chyby jsou typické v situaci, kdy Exim správně zjistí, že zpráva má být doručena na vzdáleného hostitele, nepodaří se mu ale spojit s jeho SMTP službou. Hostitel může být vypnutý, nebo mohou být problémy se síťovým spojením. Kdykoliv je zpráva tímto způsobem *odložena*, zůstává ve frontě Eximu a pokusy o doručení se pravidelně opakují. Pokud se ji ovšem v nějaké poměrně dlouhé době nepodaří doručit (typicky několik dnů), dojde k trvalé chybě a zpráva bude zahozena.

Pokud se vám příčinu chyby nepodaří zjistit z hlášení, která Exim generuje, můžete zapnout ladicí režim. Provedete to pomocí parametru `-d`, za kterým může následovat úroveň podrobností (hodnota 9 je nejobšírnější režim). Pak Exim zobrazuje hlášení o činnosti na obrazovce a můžete tak získat podrobnější údaje o tom, co není v pořádku.

Překlad programu Exim

Exim stále prochází aktivním vývojem. Verze, které jsou součástí distribucí, pravděpodobně nejsou posledními dostupnými verzemi. Pokud potřebujete nějakou funkci nebo opravu, která je dostupná až v novější verzi, musíte získat zdrojový kód a přeložit si program sami. Poslední verzi získáte prostřednictvím odkazů na adrese *http://www.exim.org*.

Linux je jeden z mnoha operačních systémů, které zdrojový kód Eximu podporují. Když chcete Exim přeložit pro Linux, musíte upravit soubor `src/EDITME` a výsledek umístit do souboru `Local/Makefile`. V souboru `src/EDITME` jsou komentáře, které říkají, k čemu se různá nastavení používají. Pak spustíte program **make**. Podrobnosti o sestavení programu najdete v dodávané dokumentaci.

Režimy doručování pošty

Jak už bylo řečeno, Exim je schopen buď doručovat poštu okamžitě, nebo ji řadit do fronty a zpracovat později. Všechna příchozí pošta je uložena v adresáři `input` pod `/var/spool/exim`. Pokud se nepoužívá řazení do fronty, provádí se doručení každé zprávy ihned poté, co dorazí. V opačném případě zůstává ve frontě tak dlouho, dokud ji nevyzvedne proces zvaný *queue-runner*. Zpracování fronty může být nastaveno napevno parametrem `queue_only` v konfiguračním souboru, nebo může být zapínáno podmíněně na základě zatížení systému v poslední minutě takto:

```
queue_only_load = 4
```

Tento příkaz způsobí zpracování fronty, pokud zatížení systému přesáhne 4[112].

Pokud nemáte počítač trvale připojen k Internetu, můžete zapnout řazení do fronty pouze u zpráv na vzdálené adresy, doručování lokálních zpráv však může probíhat okamžitě. Dosáhnete toho nastavením:

```
queue_remote_domains = *
```

Pokud zapnete jakýkoliv typ řazení zpráv do fronty, musíte zajitit, aby fronta byla pravidelně vybírána, typicky každých 10 až 15 minut. I pokud nemáte zapnuto řazení zpráv do fronty, stále je nutné frontu vybírat, protože se do ní řadí i zprávy odložené kvůli dočasné chybě. Spouštíte-li Exim v režimu démona, parametrem –q15m na příkazovém řádku mu nařídíte výběr fronty každých 15 minut. Kromě toho můžete Exim pravidelně spouštět prostřednictvím démona **cron**.

---

[112] Zatížení systému je standardní unixová metrika založená na tom, kolik procesů je uloženo ve frontě a čeká na spuštění. Příkaz **uptime** zobrazí průměrné zatížení v poslední minutě, 5 a 15 minutách.

Zprávy zařazené ve frontě můžete vypsat pomocí parametru `-bp`. Můžete také vytvořit **mailq** jako odkaz na Exim a spouštět **mailq**:

```
$ mailq
 2h    52K 12EwGE-0005jD-00 <sam@vbrew.com>
         D bob@vbrew.com
           harry@example.net
```

Tento výstup říká, že ve frontě čeká jedna zpráva bob@vbrew.com, nicméně se ji zatím nepodařilo doručit na adresu *harry@example.net*, i když je ve frontě už dvě hodiny. Velikost zprávy je 52K a Exim pro ni používá identifikátor `12EwGE-0005jD-00`. Příčinu, proč nebyla zpráva doručena, můžete zjistit z logovacího souboru této zprávy, který se nachází v adresáři `msglog`. Jednoduše to provedete parametrem `-Mvl`:

```
$ exim -Mvl 12EwGE-0005jD-00
2000-01-30 17:28:13 example.net [192.168.8.2]: Connection timed out
2000-01-30 17:28:13 harry@example.net: remote_smtp transport deferred:
  Connection timed out
```

Logovací soubory jednotlivých zpráv obsahují záznamy o konkrétních zprávách, takže je můžete snadno ověřit. Některé informace je možné zjistit z hlavního logovacího souboru pomocí nástroje **exigrep**:

```
$ exigrep 12EwGE-0005jD-00 /var/log/exim/exim_mainlog
```

Tento příkaz může trvat déle, zejména na zatíženém systému, kde může být logovací soubor dlouhý. Nástroj **exigrep** je užitečný zejména pokud hledáme informace o více zprávách. Prvním parametrem je regulární výraz a vypíše všechny logovací záznamy o všech zprávách, u nichž alespoň jeden ze záznamů obsahuje řetězec odpovídající danému regulárnímu výrazu. Díky tomu je možné snadno zjistit všechny zprávy určené na nějakou adresu nebo všechny přicházející nebo odcházející na nějakého hostitele.

Celkový přehled o činnosti programu Exim můžete získat tak, že na jeho hlavní logovací soubor spustíte **tail**. Další možnost je spustit nástroj **eximon**, dodávaný přímo s Eximem. Jedná se o X11 aplikaci, která obsahuje výpis hlavního logovacího souboru, a kromě toho zobrazuje seznam zpráv čekajících na doručení a nějaké statistiky o aktivitě programu.

Různé konfigurační volby

Uveďme si několik nejužitečnějších voleb, které můžete nastavit v konfiguračním souboru.

message_size_limit	Tento parametr omezuje velikost zpráv, které bude Exim přijímat.
return_size_limit	Tento parametr omezuje velikost původní zprávy, kterou Exim vrací jako součást chybové zprávy.
deliver_load_max	Pokud zatížení systému dosáhne nastavené hodnoty, doručování pošty se pozastaví, nicméně zprávy budou i nadále přijímány.
smtp_accept_max	Maximální počet současných SMTP spojení, které Exim přijímá.
log_level	Množství podrobností zapisovaných v logovacích souborech. Kromě toho existují i další parametry začínající *log_*, které nařizují záznam konkrétních údajů.

Směrování a doručování zpráv

Exim rozděluje doručení zpráv na tři samostatné úlohy: směrování, doručení a přenos. Pro každou operaci existuje řada různých modulů, každý z nich se konfiguruje samostatně. V konfiguračním souboru tak bývá typicky nastaveno více směrovacích, doručovacích a přenosových modulů.

Směrovače analyzují vzdálené adresy, určují, kterému hostiteli má být zpráva poslána a jaký doručovací modul má být použit. U hostitelů připojených k Internetu bývá často nastaven jediný směrovač, který provádí analýzu jmen pomocí DNS. Alternativně může být samostatný směrovač pro adresy hostitelů v lokální síti a druhý, který bude všechny ostatní zprávy směrovat na *předávacího hostitele*, například server poskytovatele internetového připojení.

Lokální adresy se předávají doručovacím modulům, kterých je typicky několik, a které obsluhují aliasy a předávání a také identifikaci lokálních schránek. Doručovací seznamy mohou být obsluhovány aliasovým nebo předávacím doručovačem. Pokud je pro nějakou adresu použit alias nebo předání, nová adresa bude nezávisle zpracována směrovačem nebo doručovačem. Nejběžnějším doručením je doručení do schránky, zprávy však mohou být předávány příkazům nebo připojovány k jiným souborům než jsou poštovní schránky.

Transportní moduly implementují způsob přenosu, například přenesení zprávy SMTP spojením nebo uložení do lokální schránky. Jaký transportní modul pro danou adresu použít rozhodují směrovače a doručovače. Pokud se transport nezdaří, Exim buď zprávu zahodí a vygeneruje chybovou zprávu, nebo zprávu odloží pro pozdější doručení.

V konfiguraci jednotlivých služeb máte velkou volnost. U každé z nich je k dispozici množství ovladačů, ze kterých si volíte ty, které potřebujete. Pak je Eximu popíšete v samostatných částech konfiguračního souboru. Nejprve se definují transportní moduly, pak doručovače a nakonec směrovače. Neexistují žádné implicitně používané moduly, i když se Exim dodává se standardním konfiguračním souborem, který vyhovuje ve většině jednoduchých nasazení. Pokud chcete změnit směrovací politiky nebo metody transportu, je jednodušší vyjít ze standardní konfigurace a tu upravit, než se pokoušet o vytvoření celé nové konfigurace úplně od začátku.

Směrování zpráv

Když má Exim adresu k doručení, podívá se nejprve, zda nejde o doménu, kterou obsluhuje lokální hostitel – to zjistí porovnáním se seznamem domén `local_domains` v konfiguračním souboru. Pokud není tato volba nastavena, za lokální doménu se považuje pouze doména odpovídající názvu lokálního hostitele. Pokud je doména lokální, předá se adresa doručovacímu modulu. Pokud není, obsluhuje ji směrovač a zjistí, kterému stroji zásilku předat[113].

Doručování zpráv na lokální adresy

Ve většině případů je lokální adresa čistě přihlašovací jméno uživatele, v takovém případě se zásilka doručí do poštovní schránky uživatele, tedy do `/var/spool/mail/`*`uživatel`*. Dalšími případy mohou být aliasy, názvy poštovních konferencí a pošta předávaná lokálním uživatelem. V těchto případech se lokální adresa přeloží na nový seznam adres, které mohou být místní nebo vzdálené.

Kromě těchto „normálních“ adres umí Exim obsluhovat další typy lokálních cílů, jako jsou soubory a roury. Při doručování do souboru připojí Exim zprávu na konec zadaného souboru, který

[113] Tento popis je zjednodušený. Je možné, že direktor předá transportnímu modulu adresu k doručení na vzdáleného hostitele a naopak směrovač může předat transportnímu modulu adresu k lokálnímu doručení, která vede k zapsání zprávy do souboru nebo roury. Směrovače mohou také za určitých okolností předávat adresy doručovacím modulům.

v případě potřeby vytvoří. Soubory a roury nejsou adresovatelné přímo, nemůžete tedy poslat poštu na adresu řekněme `/etc/passwd@vbrew.com` a předpokládat, že dojde k přepsání souboru s hesly – doručování do souboru je možné pouze tehdy, je-li soubor specifikován aliasem nebo forwardingem. Adresa `/etc/passwd@vbrew.com` je nicméně ze syntaktického hlediska platnou adresou a pokud pro ni Exim přijme zprávu, bude (typicky) hledat uživatele s přihlašovacím jménem `/etc/passwd`, nenajde jej a zprávu zahodí.

V seznamu aliasů a v předávacím souboru je za název souboru považováno cokoliv, co začíná lomítkem a nepředstavuje to plně kvalifikovanou poštovní adresu. Takže `/tmp/junk` v předávacím nebo aliasovém souboru znamená soubor, `/tmp/junk@vbrew.com` bude chápáno jako poštovní adresa, i když zřejmě nepříliš užitečná. Nicméně platné adresy tohoto typu můžete potkat při odesílání pošty přes X.400 brány, protože adresy podle standardu X.400 začínají lomítkem.

Podobně je rourovým příkazem cokoliv začínající symbolem roury (`|`), pokud to nepředstavuje platnou adresu. Pokud nezměníte konfiguraci programu, Exim nespouští příkazy v příkazovém interpretu, ale rozdělí je na sám název příkazu a na parametry a spustí jej přímo. Zpráva se pak příkazu předává na standardní vstup.

Například pokud chcete poštovní konferenci směrovat na nějakou lokální skupinu, můžete použít skript pojmenovaný **gateit** a vytvořit lokální alias, který všechny pro danou konfrenci doručí do skriptu příkazem `|gateit`. Pokud příkazový řádek obsahuje čárku, musí být celý i s úvodní rourou uzavřen v uvozovkách.

Lokální uživatelé

Lokální adresa nejčastěji znamená schránku uživatele. Ty se normálně nacházejí v adresáři `/var/spool/mail` a jde o soubor pojmenovaný jménem uživatele, který jej také vlastní. Pokud tento soubor neexistuje, Exim jej automaticky vytvoří.

V některých konfiguracích je skupina souboru nastavena na skupinu daného uživatele a práva jsou 0600. V těchto případech běží doručovací proces jako uživatel a ten může celou schránku smazat. V jiných konfiguracích je skupina nastavena na mail a soubor má režim 0660, doručovací proces běží pod uid systému a skupinou mail, a uživatel nemůže smazat soubor schránky, může jej nicméně vyprázdnit.

Momentálně standardním místem pro ukládání schránek je adresář `/var/spool/mail`, nicméně některé poštovní programy mohou být přeloženy tak, aby používaly jiné cesty, například `/usr/spool/mail`. Pokud systematicky dochází k chybám doručování pošty lokálním uživatelům, mohlo by pomoci, pokud `/usr/spool/mail` vytvoříte jako symbolický odkaz na `/var/spool/mail`.

V souboru aliasů by měly být uvedeny adresy MAILER-DAEMON a postmaster, které by měly reprezentovat adresu správce systému. Adresu MAILER-DAEMON Exim používá při odesílání chybových zpráv. Doporučuje se také, aby jako alias pro administrátora systému byla nastavena adresa root, zejména pokud se doručování provádí s právy uživatele, aby se tak předešlo situaci, kdy bude jakákoliv pošta doručována v superuživatelském režimu.

Předávání

Uživatel může přesměrovat svou poštu na alternativní adresu vytvořením souboru `.forward` v domovském adresáři. Tento soubor obsahuje seznam adres oddělených čárkami a/nebo oddělovači řádků. Čtou se a interpretují všechny řádky souboru. Typickým příkladem souboru .forward pro dobu dovolené může být

```
janet, "vacation"
```

V popisech starších souborů `.forward` můžete vidět, že jména uživatelů jsou uvedena obráceným lomítkem. Tento mechanismus byl nutný u některých starších transportních agentů, a zakazoval jim hledat v souboru `.forward` v domovském adresáři nového uživatele, protože by to mohlo vést ke vzniku smyček. U Eximu to není nutné, protože vzniku podobných smyček zabraňuje automaticky[143]. Nicméně použití obráceného lomítka je povoleno a dává smysl v případě, kdy je najednou obsluhováno více domén. Bez uvedení obráceného lomítka se nekvalifikované uživatelské jméno doplní implicitní doménou, při použití obráceného lomítka zůstane doména z původní adresy.

První adresa v souboru přeposílá příchozí zprávu do poštovní schránky uživatele *janet*, příkaz **vacation** vrátí odesilateli krátké upozornění[144].

Kromě klasických předávacích souborů umí Exim také jejich složitější variantu, takzvané *filtry*. Namísto prostého předání může filtr nejprve otestovat obsah a předat ji například pouze tehdy, pokud předmět obsahuje slovo „urgent“. Tuto možnost musí uživatelům povolit správce systému.

Aliasy

Exim umí pracovat se soubory aliasů kompatibilními s formátem programu **sendmail**. Položky v souboru mají následující tvar:

alias: *adresáti*

Adresáti jsou tvořeni čárkami odděleným seznamem adres, jimiž bude alias nahrazen. Seznam adresátů může pokračovat i na následujícím řádku za předpokladu, že je tento řádek odsazen od okraje.

Další speciální funkce umožňuje, aby Exim obsluhoval poštovní seznamy uložené odděleně od souboru aliasů. Pokud místo adresátů zadáte údaj `:include:`*soubor*, Exim přečte zadaný soubor a jeho obsah bude chápat jako seznam adresátů. Další možnost správy poštovních konferencí je popsána dále v části *Poštovní konference*.

Hlavní soubor aliasů se jmenuje `/etc/aliases`. Pokud bude nastaven jako volně zapisovatelný nebo zapisovatelný skupinou, Exim jej odmítne použít a doručování lokálních zpráv pozastaví. Test používaný pro kontrolu režimu souboru můžete nastavit změnou hodnoty `modemask` v direktoru `system_aliases`.

Soubor aliasů může vypadat takto:

```
# vbrew.com, soubor /etc/aliases
hostmaster: janet
postmaster: janet
usenet: phil
# Poštovní konference vývojářů
development: joe, sue, mark, biff,
        /var/mail/log/development
owner-development: joe
# Obecná oznámení se zasílají všem uživatelům
announce: :include: /etc/Exim/staff,
        /var/mail/log/announce
```

[114] Pokud má být zpracována adresa, kterou direktor vygeneroval zpracováním původní adresy, tak už se direktor znovu nevolá.

[115] Pokud používáte nějaký takový program, zajistěte, aby neodpovídal na zprávy posílané z poštovních seznamů. Je docela nepříjemné, pokud někdo odjede na dovolenou a ke každé zprávě došlé do konference se přidá zpráva oznamující, že on je na dovolené. Pro správce konferencí: právě proto není rozumné nastavovat hodnotu `Reply-To:` ve zprávách posílaných konferencí na adresu, kterou se do konference přispívá.

```
owner-announce: root
# poštovní konference ppp se směruje na lokální skupinu
ppp-list: "|/usr/local/bin/gateit local.lists.ppp"
```

Pokud jsou v souboru aliasů tak jako zde uvedeny soubory a roury, je nutné Eximu říct, pod jakým uživatelským ID má doručování probíhat. V konfiguračním souboru musí být nastavena volba user (a případně group), a to v části direktoru aliasů nebo v transportním modulu, který doručování provádí.

Pokud dojde k chybě při doručování pošty na adresu uvedenou v souboru aliasů, Exim jako obvykle pošle chybové hlášení odesilateli zprávy, což nemusí stačit. Pomocí volby errors_to je možné nastavit doručování chybových zpráv na jinou adresu, například na správce poštovního systému.

Poštovní konference

Kromě aliasů je možné poštovní konference obsluhovat také direktorem `forwardfile`. Konference jsou ukládány v jednom adresáři, například `/etc/exim/lists` a konference pojmenovaná například *nag-bugs* pak bude uložena v souboru `lists/nag-bugs`. Tento soubor obsahuje adresy příjemců oddělené buď čárkami nebo oddělovači řádků. Řádky začínající symbolem # představují komentáře. Takto nastavená data je možné používat následujícím direktorem:

```
lists:
  driver = forwardfile
  file = /etc/exim/lists/${ local_part}
  no_check_local_user
  errors_to = ${ local_part} -request
```

Při spuštění direktoru dojde k expandování hodnot v údajích file a errors_to. Expanze znamená, že řetězce začínající symbolem $ budou při každém použití řetězce nahrazeny. Nejjednodušším způsobem expanze je použití jedné z proměnných programu Exim tak, jak to děláme tady. Řetězec ${ local_part} bude nahrazen hodnotou $local_part, což je lokální část právě zpracovávané adresy.

Pro každou poštovní konferenci by měl existovat uživatel (nebo alias), který bude pojmenován *názevkonference*-request. Chyby při doručování nebo expanzi adres budou oznamovány na tuto adresu.

Ochrana před spammingem

Spam, nebo také nevyžádaná elektronická pošta, je nepříjemný problém řady uživatelů. Byl založen projekt, který má napomoci řešení tohoto problému. Jmenuje se Mail Abuse Protection System (MAPS) a v jeho rámci byl vytvořen mechanismus, který redukuje problém spammingu. Tento mechanismus se jmenuje Real Time Blackhole List (RBL). O tom, jak MAPS RBL funguje, se můžete informovat na adrese *http://maps.vix.com/rbl/*. Myšlenka je jednoduchá. Systémy odesílající spam jsou zařazeny do databáze a programy jako Exim si mohou v databázi ověřit, že právě příchozí pošta nepochází z nějaké takové adresy.

Se vznikem projektu RBL vznikla i řada dalších seznamů. Jedním z nich je Dial-Up List (DUL), který obsahuje IP adresy telefonicky připojovaných hostitelů. Ti by měli veškerou odchozí poštu směrovat zásadně přes poštovní server svého poskytovatele. Řada systémů blokuje příjem pošty od telefonicky připojených hostitelů, protože když takový hostitel obchází poštovní server svého poskytovatele, neznamená to obvykle nic dobrého.

Exim obsahuje podporu tohoto typu seznamů. Konfiguruje se velmi snadno. Do souboru `/etc/exim.conf` přidáte následující řádek:

```
# Vixie / MAPS RBL (http://maps.vix.com/rbl)
rbl_domains = rbl.maps.vix.com : dul.maps.vix.com
```

Tento příklad testuje seznamy RBL a DUL a odmítá veškerou poštu od hostitelů, kteří jsou v některém ze seznamů uvedeni. Volba rbl_hosts umožňuje nastavit, pro jaké skupiny hostitelů se má nebo nemá kontrola proti RBL seznamu provádět. Implicitní nastavení je:

```
rbl_hosts = *
```

Znamená to, že kontrola se bude provádět pro všechny hostitele. Pokud chcete přepsat nastavení seznamu a od určitých hostitelů přijímat poštu vždy, můžete zadat například:

```
bl_hosts = ! nocheck.example.com : *
```

Vykřičník před prvním údajem znamená negaci údaje: pokud se k vám připojí hostitel nocheck.example.com, vyhovuje tomuto údaji. Protože se jedná o negovaný údaj, nebude kontrola proti RBL provedena. U všech ostatních hostitelů se kontrola provádět bude.

Nastavení pro UUCP

Exim neobsahuje podporu přenosu pošty přes UUCP a nepodporuje vykřičníkové adresy formátu UUCP. Pokud se ale používá doménové adresování, je možné poměrně jednoduše Exim pro práci s UUCP nastavit. Následující příklad představuje konfiguraci, v níž se určité domény posílají na UUCP:

```
# Transport
uucp:
  driver = pipe
  user = nobody
  command = "/usr/local/bin/uux -r - \
    ${ substr_-5:$host} !rmail ${ local_part} "
  return_fail_output = true

# Router
uucphost:
  transport = uucp
  driver = domainlist
  route_file = /usr/exim/uucphosts
  search_type = lsearch
```

V celém konfiguračním souboru by se sekce Transport nacházela mezi ostatními doručovači a sekce Router mezi ostatními směrovači. Soubor `/usr/exim/uucphosts` obsahuje údaje jako:

```
darksite.example.com:            darksite.UUCP
```

který bude interpretován jako „Pošli poštu adresovanou doméně *darksite.example.com* na UUCP hostitele *darksite*. Konfiguraci je možné zjednodušit tak, že směrovač nebude k názvu *darksite* přidávat příponu *UUCP*, kterou musí transportní modul zase odstranit, nicméně použité nastavení má výhodu v tom, že je jasný rozdíl mezi doménovým jménem *darksite.example.com* a UUCP jménem *darksite*.

Kdykoliv směrovač narazí na adresu uvedenou v souboru tras, pošle ji transportnímu modulu UUCP, který ji následně předá příkazu **uux** (popsanému v kapitole 16). Pokud dojde k chybě, **uux** vygeneruje nějaký výstup a skončí s nenulovým výstupním kódem. Nastavení return_fail_output zajistí, že chybové hlášení bude předáno zpět odesilateli.

Pokud jsou příchozí UUCP zprávy sdružovány v souborech ve formátu dávkového SMTP, je možné je Eximu přímo předat příkazem:

```
exim -bS </var/uucp/incoming/001
```

Je zde nicméně jeden problém. Když Exim obdrží zprávu lokálně, předpokládá, že odesilatelem je lokální uživatel, který Exim volal, nicméně u UUCP dávek chceme, aby byl odesilatel převzat z příchozí zprávy. Exim to provede, pokud byl zavolán procesem označeným jako *prověřený uživatel*. Pokud budete příchozí UUCP poštu obsluhovat uživatelem uucp, musíte v konfiguračním souboru zadat nastavení:

```
trusted_users = uucp
```

Tím se zajistí, že adresy budou správně převzaty ze zpráv.

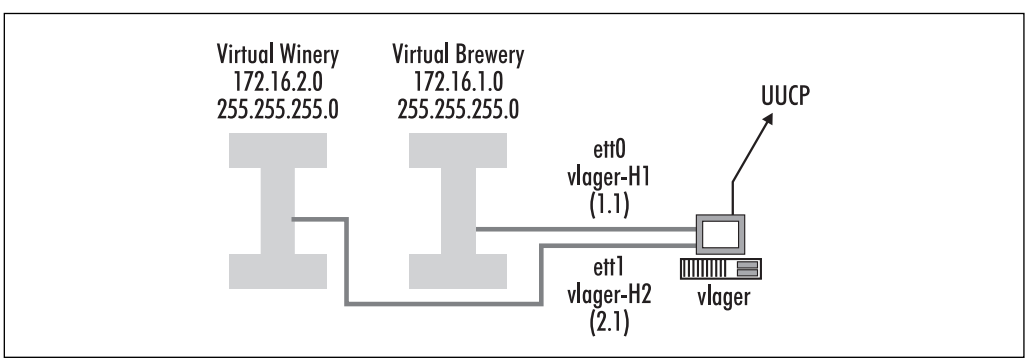
Příklad sítě virtuálního pivovaru

V knize jsme používali příklad sítě, která byla poněkud jednodušší než síť Groucho Marx University a zřejmě se bude více blížit vašim praktickým potřebám.

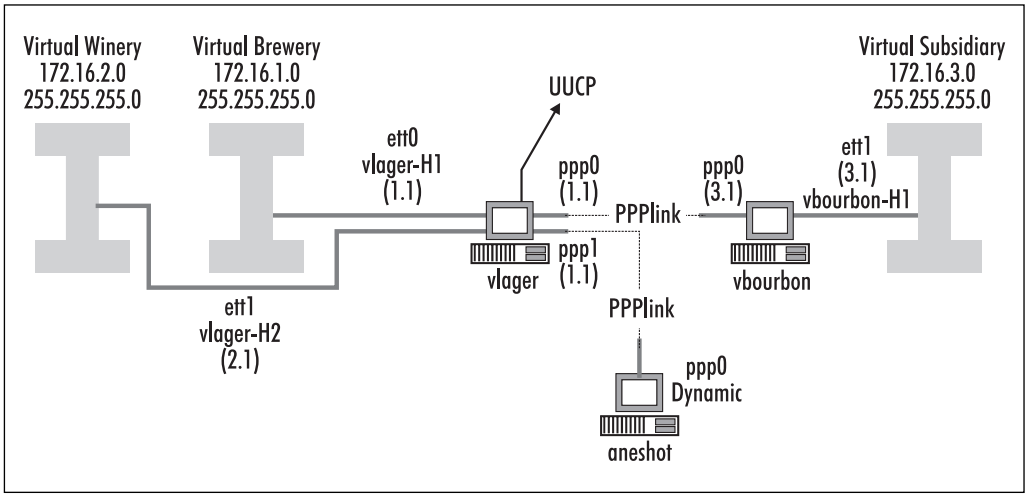
Virtuální pivovar je malá společnost, která, jak název napovídá, vaří virtuální pivo. Aby mohli vařit lépe, mají virtuální pivovarníci počítače propojeny v síti, přičemž všechny počítače používají to nejnovější a nejstabilnější linuxové jádro. Konfiguraci sítě ukazuje obrázek A.1.

Na stejném patře, jenom na druhé straně chodby, sídlí virtuální vinařství, které s virtuálním pivovarem úzce spolupracuje. Vinaři mají svou vlastní síť. Obě společnosti si své sítě propojily. Nejprve vytvořili bránu, která směruje datagramy mezi oběma podsítěmi. Pak si pořídili UUCP linku do okolního světa, kterou přenášejí poštu a news. A nakonec si obstarali i PPP spojení pro přístup na Internet.

Pivovar i vinařství používají podsítě třídy C v rámci třídy B pivovaru, a propojují je branou **vlager**, která zároveň slouží jako UUCP spoj. Tuto konfiguraci znázorňuje obrázek A.2.



Obrázek A.1 – Podsítě virtuálního pivovaru a virtuálního vinařství



Obrázek A.2 – Síť virtuálního pivovaru

Připojení sítě virtuální pobočky

Postupem času se virtuální pivovar rozrostl a otevřel si pobočku v jiném městě. V pobočce mají vlastní síť s IP adresou 172.16.3.0, což je třetí podsíť třídy B pivovaru. Počítač vlager funguje jako brána sítě pivovaru a podporuje PPP spojení. Její kolega na síti pobočky se jmenuje vbourbon a má IP adresu 172.16.3.1. Strukturu rovněž vidíte na obrázku A.2.

Užitečná zapojení kabelů

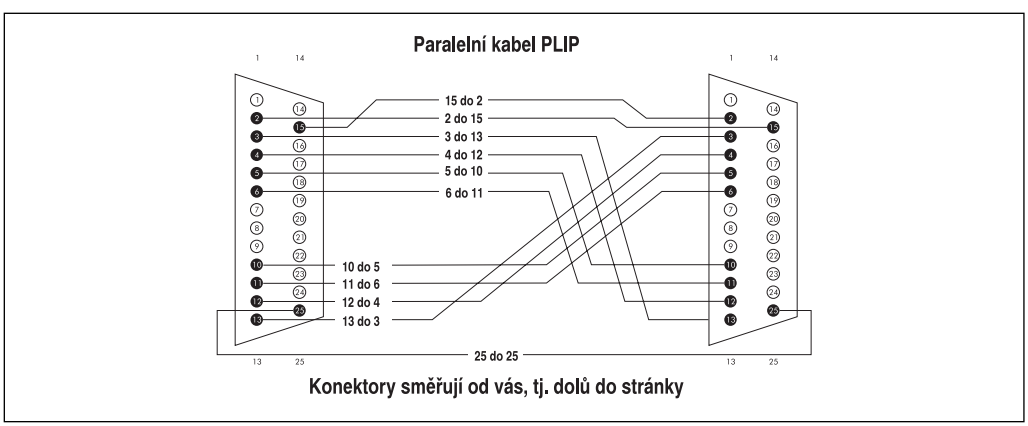
Pokud chcete propojit dva počítače a nemáte Ethernet, můžete použít buď null-modemový sériový kabel nebo paralelní kabel PLIP.

Kabely si můžete samozřejmě koupit, nicméně jejich výroba je velmi jednoduchá a vyjde vás podstatně levněji.

Paralelní kabel PLIP

K sestavení paralelního PLIP kabelu potřebujete dva 25pinové konektory (označení DB-25) a nějaký 11žilový drát. Ten může být dlouhý maximálně 15 metrů. Může a nemusí být stíněný, pokud se ale blížíte k limitu 15 metrů, bude lépe použít stíněný drát.

Podíváte-li se na konektor, měli byste vedle každého pinu přečíst drobná čísla, přičemž jedničku má pin vlevo nahoře (držíte-li konektor širší stranou nahoru) a pin v pravém dolním rohu má číslo 25. PLIP kabel vyrobíte tak, že propojíte vodiče podle následujícího obrázku.



Obrázek B.1 – Paralelní kabel PLIP

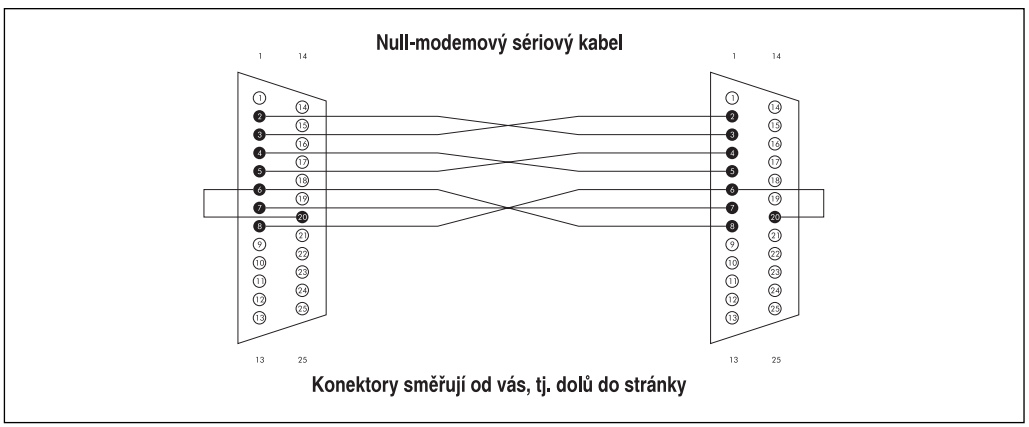
Všechny ostatní vodiče jsou nepropojeny. Pokud používáte stíněný kabel, stínění připojte ke kovovému masivu konektoru pouze na jednom konci kabelu.

Null-modemový sériový kabel

Sériový null-modemový kabel funguje pro SLIP i PPP. Opět budete potřebovat dva DB-25 konektory, tentokrát vám ale postačí osmižilový vodič.

Existují i jednodušší (třívodičové) null-modemové kabely, zde uvedené zapojení ale umožní používat hardwarové řízení toku – které je lepší než řízení protokolem XON/XOFF – nebo dokonce se úplně obejít bez řízení toku. Zapojení vidíte na obrázku B.2.

Pokud používáte stíněný kabel, stínění připojíte na první pin pouze jednoho konektoru.



Obrázek B.2 – Sériový null-modemový kabel

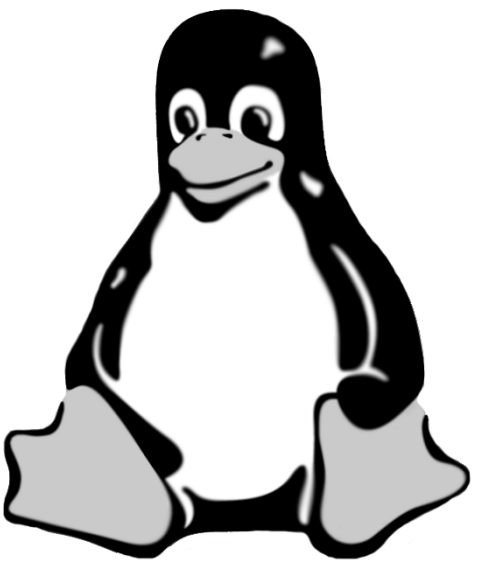
Projekt SAGE

Pokud se vám nepodaří dozvědět se všechno potřebné z příspěvků na *comp.os.linux.** ani v dokumentaci, možná stojí za to připojit se k projektu SAGE, System Administrators Guild, sponzorovanému USENIXem. Hlavním cílem projektu SAGE je vzdělávat profesionální správce systémů. Propojuje správce systémů a síti, vývojáře hardwaru i softwaru, slouží ke sdílení problémů a řešení, komunikaci s uživateli, dodavateli a managementem.

Mezi hlavní iniciativy projektu SAGE patří:

- Společně s USENIXem pořádá ročně velmi oblíbenou konferenci System Administration Conference (LISA).
- Vydává knihu **Job Descriptions for System Administrators**, první ze série praktických příruček pokrývajících problémy a postupy správců systému.
- Vytváří archivy z přednášek na konferencích správců a z dalších zajímavých pramenů. Archiv najdete na adrese *ftp.sage.usenix.org*.
- Vytváří skupiny působící v různých důležitých oblastech, například publikování, strategie, distribuce informací, vzdělávání, dodavatelé nebo standardy.

Další informace o USENIX Association a o skupině projektu SAGE získáte na americkém telefonním čísle (510) 528-8649, nebo elektronickou poštou na adrese *office@usenix.org*. V elektronické podobě si můžete informace vyžádat na adrese *info@usenix.org*. Roční členství v SAGE stojí 25 dolarů (zároveň musíte být členem USENIXu). Členství zahrnuje odběr časopisů *login:* a *Computing Systems*, slevy účastnických poplatků na konferencích, slevy na odebírané publikace a další.



ČÁST IV

Praktické návody (HOWTO)

Konfigurace systému

Originál: http://tldp.org/HOWTO/Config-HOWTO/

Tento dokument se zaměřuje na snadnější a jednodušší vyladění vašeho nově nainstalovaného systému Linux. Najdete zde řadu tipů pro konfiguraci nejběžnějších aplikací a služeb.

Úvod

Proč toto HOWTO

Aktuální distribuce se blíží dokonalosti, ale stále je zapotřebí nějaké jemné doladění. Mnoho nových uživatelů má strach ze zdánlivé složitosti systému Linux, a díky tomu jsem si všiml, že se některé otázky objevují na c.o.l.setup stále znovu. Z důvodu pokusu o vyřešení této situace a vlastní pohodlnosti, jsem napsal seznam co – dělat (to –do), který se nakonec stal tímto HOWTO. Najdete zde tipy pro konfiguraci a příklady nejčastěji používaných aplikací, programů a služeb, které by vám měly ušetřit značné množství času a práce.

Tento dokument vytvářím hlavně pro RedHat. V současné době mám k dispozici počítače se systémy Red Hat a Mandrake a jádry v rozsahu od 2.0.36 do 2.2.15, takže nepovažujte některé z mých tipů za dogma, pokud máte jiné distribuce. Předchozí verze tohoto dokumentu obsahovaly některé informace o systémech SuSE, Debian a Caldera, ale počítače s těmito systémy už nemám k dispozici, takže nemohu detaily aktualizovat. Žádná informace je lepší než nepřesná informace, takže adaptace mých tipů na vaše distribuce je pouze na vás.

Tento praktický návod, a není to ani jeho záměrem, nemá nahradit jiné.Čtení dokumentace a praktických návodů se vždy vyplatí, takže vám to důrazně doporučuji, pokud se chcete dozvědět více. Také to není pro začátečníky: pokud zjistíte, že něco nechápete, podívejte se prosím do příslušného HOWTO. Dovolte mi připomenout, že správným místem pro hledání pomoci s konfigurací systému Linux je Usenet, tj. news:comp.os.linux.setup. *Prosím*, nehledejte pomoc u mě, protože jsem zcela zaneprázdněn.

Oficiální místo pro tento dokument, na kterém se nachází také ostatní HOWTO, na která se odkazuji, a některé překlady, je http://www.linuxdoc.org.

Co budeme konfigurovat

Mohlo by zde být nekonečně mnoho konfigurací hardwaru PC, ale podle mých zkušeností je toto společné: PC s velkým diskem rozděleným do tří oddílů (jeden pro DOS/Windows, jeden pro Linux, jeden odkládací), zvuková karta, modem, jednotka CD-ROM, tiskárna, myš. Velmi častá je také externí jednotka Zip připojená k paralelnímu portu. Tento počítač je možná připojen do kombinované sítě Windows-Linux, kde pracuje jako server.

To je hardware, který si myslím, že chcete nastavit, a následující tipy je snadné adaptovat na různé konfigurace. Implicitně předpokládám, že se budete při úpravách a opravách hlásit jako uživatel root.

A nyní, vážení, pojďme na to.

Obecné nastavení systému

Několik slov o zabezpečení

Ještě před spuštěním vašeho systému byste se měli rozhodnout, jakou úroveň zabezpečení chcete implementovat. Než se rozhodnete, co chcete dělat, nepřipojujte váš počítač do sítě.

Zabezpečení je obrovským tématem, které přesahuje tento dokument. Dvěma dobrými výchozími body jsou příručka správce zabezpečení systému Linux (Linux Security Administrator's Guide) na adrese http://www.securityportal.com/lasg a příručka zabezpečení systému Linux (Linux Security Guide) na adrese http://nic.com/~dave/SecurityAdminGuide/index.html. Měli byste zvážit alespoň následující věci: používání stínových hesel (Shadow Password HOWTO), omezení přístupu k tomuto počítači ze sítě (část Omezování přístupu ze sítě), používání Secure Shell (http://www.openssh.org) nebo zabezpečené vzdálené heslo (Secure Remote Password; http://srp.stanford.edu/srp/). Hodně štěstí.

Začněte poznámkami!

Pro udržování vaší instalace je *nezbytné*, abyste přesně věděli, co se stalo ve vašem počítači, které balíčky jste kdy nainstalovali, co jste odstranili nebo modifikovali atd. Takže první věc, co uděláte, než si začnete hrát s vaším počítačem, bude vytvoření „deníku". Do něj si budete poznamenávat *všechny* změny, které provedete jako root; já mám také svůj deník, do kterého si poznamenávám všechny úpravy systémových souborů, doinstalované balíčky .rpm a soubory .tar.gz, které jsem nainstaloval. Optimálně, díky možnosti zpětného procházení vašich změn, byste měli být schopni obnovit původní instalaci.

Dělejte si záložní kopie systémových souborů, se kterými jste něco dělali. Přes to všechno používejte RCS. Pak budete schopni vrátit všechny změny. Nikdy nepracujte jako root bez toho, abyste si poznamenávali změny!

Klávesnice

Pokud jste tento krok vynechali při instalaci nebo jste vyměnili vaši klávesnici, budete muset:

- najít příslušnou tabulku kláves nacházející se v /usr/lib/kbd/keymaps/i386; například qwuerty/it-latin1.kmap.gz podporuje italskou klávesnici;
- upravit soubor /etc/sysconfig/keyboard tak, aby obsahoval: `KEYTABLE="it-latin1"`;
- nastavit rychlost opakování a prodlevy klávesnice přidáním tohoto řádku do /etc/rc.d/rc.sysinit:

```
/sbin/kbdrate -s -r 16 -d 500  # nebo jak budete chtít
```

Pro načtení tabulky kláves spusťte

```
/etc/rc.d/init.d/keytable start
```

Dalšími speciálními klávesami se budeme zabývat v následujících částech. Pro implicitní zapnutí NumLock přidejte tyto řádky do /etc/rc.d/rc.sysinit:

```
for tty in /dev/tty[1-9]*; do
  setleds -D +num < $tty
done
```

Normálně konzola systému Linux nerozlišuje mezi šipkami a šipkami s klávesou Shift, ale některé aplikace (jmenovitě editor Jed) ano. Normálně jsou tyto vazby kláves k dispozici pouze v programu xterm. Následující mapa kláves, kterou můžete načíst při spuštění, je velmi užitečná:

```
# načtěte tuto mapu kláves pomocí: loadkey shift.map
# Shift + Up
shift keycode 103 = F100
string F100 = "\033[a"
# Shift + Left
shift keycode 106 = F101
string F101 = "\033[c"
# Shift + Right
shift keycode 105 = F102
string F102 = "\033[d"
# Shift + Down
shift keycode 108 = F103
string F103 = "\033[b"
# Ctrl + Ins
control keycode 110 = F104
string F104="\033[2^"
# Shift + Ins
shift keycode 110 = F105
string F105="\033[2$"
# Shift + PgUp
shift keycode 104 = F106
string F106 = "\033[5$"
# Shift + PgDn
shift keycode 109 = F107
string F107 = "\033[6$"
# Shift + Home
shift keycode 102 = F108
string F108 = "\033[1$"
# Shift + End
shift keycode 107 = F109
string F109 = "\033[4$"
# Shift + Del
shift keycode 111 = F110
string F110 = "\033[3$"
# Ctrl + Del
control keycode 111 = F111
string F111 = "\033[3^"
```

Spouštěcí a záchranná disketa

Vytvořte si spouštěcí diskety pro váš nově nainstalovaný systém. Vaše distribuce může obsahovat příkaz pro vytváření těchto disket (řekněme, `mkbootdisk` nebo něco podobného); pokud ne, tyto příkazy to udělají:

```
#~ dd if=/boot/vmlinuz-2.0.36-0.7 of=/dev/fd0  # použít obraz jádra
#~ rdev /dev/fd0 /dev/hda2                     # váš kořenový oddíl
```

Také mějte připravené záchranné diskety. Na ftp://ibiblio.org/pub/Linux/system/recovery jsou široké možnosti záchranných disket; pokud nevíte, kterou zvolit, doporučuji vyzkoušet projekt Tomsrtbt, jehož domovská stránka je na adrese http://www.toms.net/rb. Je velmi povedený, ale na první pohled to vypadá, že některé utility chybí; například zde není `ftp`, ale místo něj je tu `nc` (netcat). Podívejte se do jeho dokumentace.

Jádro

Podle mého názoru první věcí, kterou je dále potřeba udělat, je sestavit jádro, které se nejlépe hodí k vašemu systému. Je to velmi jednoduché, ale v každém případě se podívejte do souboru `README` v /usr/src/linux/ nebo do Kernel HOWTO. Poznámky:

- zvažte pečlivě vaše potřeby. Volba konfigurace jádra, použití záplat a jeho zkompilování jednou pro vždy je produktivnější, než překonfigurovávání a kompilace každý měsíc; zejména, když váš Linux pracuje jako server. Nezapomeňte vložit podporu pro všechen hardware, který byste v budoucnu pravděpodobně přidávali (např. SCSI, Zip, síťové karty atd.); použití modulů je obvykle nejlepší volbou;
- uživatelé notebooků: pokud plánujete používat PCMCIA modem/fax, nezapomeňte zkompilovat podporu sériových portů *do jádra*. Nekompilujte jej jako modul nebo váš modem PCMCIA nebude pracovat;
- zakompilujte všechno, co budete potřebovat; tj. nezapomeňte na moduly pcmcia nebo ovladače zvuku ALSA;
- pro ušetření času potřebného pro další překonfigurování a nové zkompilování jádra je dobré ukládat vaši konfiguraci do souboru a uchovávat ji na bezpečném místě. Pokud inovujete jádro a použijete „make oldconfig“, bude vzat váš starý konfigurační soubor a na nezahrnuté funkce budete dotázáni, zda mají být zahrnuty, což se projeví v novém, aktualizovaném konfiguračním souboru.

Výkon pevného disku

Výkon vašeho disku (E)IDE může být hodně vylepšen *opatrným* použitím `hdparm`(8). Pokud jej vaše distribuce Linuxu neobsahuje, najdete jej na adrese ftp://ibiblio.org/pub/Linux/system/hardware; hledejte soubor s názvem hdparm-X.Y.tar.gz.

Jelikož mnoho detailů závisí na vašem pevném disku a jeho řadiči, nemohu vám dát obecný recept. Riskujete, že zničíte váš systém souborů, takže si *přečtěte pozorně manuálové stránky*, než některé z těchto voleb použijete. V tom nejjednodušším případě můžete přidat následující řádek do /etc/rc.d/rc.sysinit:

```
/sbin/hdparm -c1 /dev/hda  # je předpokládána první jednotka IDE
```

což zapíná podporu 32bitového vstupu a výstupu (E)IDE. O volbě „-m“ mi psal Mark Lord, autor `hdparm`:

*(...) pokud váš systém používá starší komponenty [< 1997], bude to v pořádku. Se staršími *by mohl* být problém (je to nepravděpodobné). Skutečně špatné čipy byly CMD0646 a RZ1000, používané *hodně* na 486 a (starších) základních deskách 586 před 2-3 lety.*

V novějších počítačích by toto nastavení mělo fungovat správně:

```
/sbin/hdparm -c1 -A1 -m16 -d1 /dev/hda
```

Jednotka Zip pro paralelní port

Jádra obsahují ovladač pro starší (ppa) i novější (imm) jednoty Zip. Pokud překompilujete jádro, ujistěte se, že je zapnuta podpora pro SCSI a disky SCSI. Pamatujte si, že se mohou vyskytnout konflikty mezi tiskárnou a jednotkou Zip na stejném paralelním portu, takže bude lepší použít moduly jádra.

Disky Zip jsou prodávány předformátované na oddíl /dev/sda4. Pro zapnutí jednotky Zip přidejte toto do /etc/rc.d/rc.sysinit:

```
# Zapnutí jednotky Zip
/sbin/modprobe ppa  # imm pro novější modely
```

Disky Zip mohou být připojeny pomocí /etc/fstab jak je uvedeno níže, nebo pomocí Mtools přidáním tohoto řádku do vašeho /etc/mtools.conf:

```
drive z: file="/dev/sda4" exclusive
```

kromě toho, vám příkaz `mzip` umožňuje vysunutí, dotaz na stav, zápis a ochranu disků Zip heslem; podrobnosti najdete v manuálových stránkách pomocí `man mzip`. Domovská stránka Mtools je na adrese http://linux.wauug.org/pub/knaff/mtools.

Ovladače zařízení

Zařízení v `/dev` (nebo spíše odkazy na aktuální ovladače zařízení) mohou chybět. Zkontrolujte, čemu odpovídají vaše zařízení pro myš, modem a jednotku CD-ROM, pak udělejte následující:

```
~# cd /dev
/dev# ln -s ttyS0 mouse; ln -s ttyS1 modem; ln -s hdb cdrom; ln -s sda4 zip
```

Ve většině notebooků je ovladač myši /dev/psaux: vezměte to v úvahu při konfigurování X11. Pokud budete chtít, proveďte `chmod 666` pro tato zařízení, abyste je zpřístupnili všem uživatelům.

Zvuková karta

Mé stolní PC je osazeno starým Sound Blaster 16; dokonce i když máte něco jiného, můžete použít toto jako vodítko.

Zkompiloval jsem podporu pro zvukovou kartu jako modul (`sb.o`). Pak to dám do /etc/modules.conf:

```
options sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330
alias sound sb
```

Pro zapnutí zvuku se ujistěte, že je spouštěno `modprobe sound` v /etc/rc.d/rc.sysinit. Eventuálně si můžete stáhnout nástroj `sndconfig` ze serveru RedHat.

Kromě standardních ovladačů jádra pro zvuk jsou svělou volbou ovladače Alsa (http://www.alsa-project.org). Překvapivě, jsou zvukové kanály implicitně vypnuté. Budete muset použít `aumix` a tento /etc/aumixrc pro nastavení hlasitosti na 100 %:

```
vol:100:100:P
synth:100:100:P
pcm:100:100:P
line:100:100:P
mic:100:100:R
cd:100:100:P
```

Zprávy při přihlášení

Pokud chcete upravit zprávy při přihlášení, zkontrolujte, zda váš /etc/rc.d/rc.local přepisuje /etc/issue a /etc/motd (RedHat to dělá). Pokud ano, chopte se vašeho editoru.

Pokud byste chtěli barevnou zprávu při přihlášení, můžete upravit váš `rc.local` vložením řádků jako jsou tyto:

```
# místo ^[ vložte skutečný znak escape. To uděláte:
# emacs: ^Q ESC   vi: ^V ESC   joe:  ' O 2 7   jed: ' ESC
ESC="^["  # skutečný znak escape
BLUE="$ESC[44;37m"
NORMAL="$ESC[40;37m"
CLEAR="$ESC[H$ESC[J"

> /etc/issue
echo "$CLEAR" >> /etc/issue
echo "$BLUE   Welcome to MyServer (192.168.1.1)   " >> /etc/issue
echo "$NORMAL " >> /etc/issue
echo "" >> /etc/issue
```

Název hostitele

Spuštění příkazu `hostname new_host_name` nemusí stačit. Pro obejití hrozného uzamčení `sendmail` proveďte následující kroky (platné pouze pro samostatné počítače):

- upravte /etc/sysconfig/network a zde změňte název hostitele (např. `new_host_name.your_domain`);
- stejně upravte /etc/HOSTNAME;
- přidejte nový název hostitele do řádku /etc/hosts:

```
127.0.0.1       localhost  new_host_name.your_domain
```

Myš

Služby pro myš `gpm` jsou užitečné pro provádění vyjmutí a vložení v režimu tty a pro použití myši v některých aplikacích. Ujistěte se, že máte soubor /etc/sysconfig/mouse a že obsahuje:

```
MOUSETYPE="Microsoft"
XEMU3=yes
```

Kromě toho musíte mít soubor /etc/rc.d/init.d/gpm, do kterého přidáte parametry pro příkazový řádek. Můj obsahuje:

```
...
       daemon gpm -t $MOUSETYPE -d 2 -a 5 -B 132 # dvoutlačítková myš
...
```

Samozřejmě zajistěte, aby šlo o správnou konfiguraci pro váš druh myši. Ve většině notebooků je `MOUSETYPE „PS/2“`.

Pokud byste chtěli používat nabídky v konzole pomocí klávesy Ctrl, nastavte `gpm-root`. Upravte výchozí konfiguraci v /etc/gpm-root.conf, pak spusťte `gpm-root` v /etc/rc.d/rc.local.

Připojovací body

Je šikovné mít připojovací body pro disketovou jednotku, další zařízení a adresáře NFS. Například můžete provést toto:

```
~# cd /mnt; mkdir floppy cdrom win zip server
```

Toto vytváří připojovací body pro disketovou jednotku DOS/Win, jednotku CD-ROM, oddíl pro Windows, jednotku Zip pro paralelní port a adresář NFS.

Nyní upravte soubor /etc/fstab a přidejte následující položky:

```
/dev/fd0        /mnt/floppy     auto            user,noauto 0 1
/dev/cdrom      /mnt/cdrom      iso9660         ro,user,noauto 0 1
/dev/zip        /mnt/zip        vfat            user,noauto,exec 0 1
/dev/hda1       /mnt/win        vfat            user,noauto 0 1
server:/export  /mnt/server     nfs             defaults
```

Samozřejmě musíte použít správná zařízení v prvním poli.

Všimněte si druhu souborového systému „auto“ v prvním řádku; to vám umožňuje připojovat diskety ext2 i vfat (DOS/Windows), ale potřebujete poslední verzi `mount`. Možná vám bude více vyhovovat `mtools`.

Automatické připojování

Pokud nemáte rádi připojování a odpojování věcí, zvažte použití `autofs`(5). Řeknete démonu autofs, co chcete automaticky připojovat a kde začít, souborem /etc/auto.master. Jeho struktura je jednoduchá:

```
/misc   /etc/auto.misc
/mnt    /etc/auto.mnt
```

V tomto příkladu řeknete autofs, že chcete automaticky připojovat média v /misc a /mnt, přičemž jsou připojovací body udány v /etc/auto.misc a /etc/auto.mnt. Příklad /etc/auto.misc:

```
# export NFS
server          -ro                     my.buddy.net:/pub/export
# vyměnitelná média
cdrom           -fstype=iso9660,ro      :/dev/hdb
floppy          -fstype=auto            :/dev/fd0
```

Spusťte automounter. Od této chvíle, kdykoliv se pokusíte o přístup k neexistujícímu připojovacímu bodu /misc/cdrom, bude vytvořen a jednotka CD-ROM bude připojena.

lilo(8) a LOADLIN.EXE

Mnoho uživatelů používá na svém PC systémy Linux i DOS/Windows a chtějí mít možnost volby operačního systému při spouštění počítače; to by mělo být provedeno při instalaci, ale pokud ne, proveďte následující. Předpokládejme, že /dev/hda1 obsahuje DOS/Windows a že /dev/hda2 obsahuje Linux.

```
~# fdisk
Using /dev/hda as default device!

Command (m for help):a
Partition number (1-4): 2

Command (m for help):w
~#
```

Toto udělá oddíl Linux spustitelným. Pak zapište do souboru /etc/lilo.conf toto:

```
boot = /dev/hda2
compact                # může být v konfliktu s "linear"
delay = 100            # 10 sekund
linear                 # zbaví nás problému "1024. cylindru"
message = /boot/bootmesg.txt  # můžete zadat vlastní
root = current
image = /boot/vmlinuz  # spouštět linux implicitně tím, že je tato položka prv-
ní
  label = linux
  read-only
#  append="mem=128M"   # aby bylo možno použít více než 64M paměti
other = /dev/hda1
  table = /dev/hda
  label = win
```

Nyní spusťte /sbin/lilo a je hotovo. Protože je `lilo` rozhodující částí vaší instalace, velmi doporučuji, abyste si k němu přečetli dokumentaci.

Pro spuštění systému Linux z DOS/Windows bez resetování umístěte `LOADLIN.EXE` do adresáře (v oddílu DOS!) umístěného v cestě DOS; pak zkopírujte vaše jádro do, řekněme, `C:\TEMP\VMLINUZ`. Následující jednoduchý soubor `.BAT` spustí Linux:

```
rem   linux.bat
smartdrv /C
loadlin c:\temp\vmlinuz root=/dev/hda2 ro
```

Pokud používáte Windows 9x, nastavte vlastnosti tohoto `.BAT` tak, aby se spouštěl v režimu MS-DOS.

> **Tip pro zabezpečení:** Před nainstalováním systému Linux je dobré vytvořit záložní kopii vašeho MBR. Připravte si záchrannou disketu Windows a ujistěte se, že obsahuje `FDISK.EXE`. Pro obnovu MBR musíte provést jen
>
> ```
> A:\> fdisk /mbr
> ```

Konfigurace tiskárny

Všechny distribuce, které znám, mají konfigurační nástroj pro nastavování tiskáren (`printtool`, `yast` nebo `magicfilter`); pokud jej nemáte, zde je základní ruční konfigurace.

Předpokládejme, že máte ne-postscriptovou (také „ne pouze pro Windows“!) tiskárnu, kterou chcete používat pro tisk holého textu (např. zdrojových souborů C) a postscriptových souborů pomocí Ghostscript, o kterém předpokládejme, že je již nainstalován.

Nastavování tiskárny se provádí v několika krocích:

- zjistěte které paralelní tiskové zařízení to je: zkuste

  ```
  ~# echo "hello, world" > /dev/lp0
  ~# echo "hello, world" > /dev/lp1
  ```

  a podívejte se, které pracuje.

- Vytvořte dva adresáře pro řazení:

  ```
  ~# cd /var/spool/lpd
  /var/spool/lpd/# mkdir raw; mkdir postscript
  ```

- pokud vaše tiskárna předvádí „efekt schodiště“ (většina inkoustových tiskáren to dělá), budete potřebovat filtr. Zkuste vytisknout dva řádky pomocí

  ```
  ~# echo "první řádek" > /dev/lp1 ; echo "druhý řádek" > /dev/lp1
  ```

  pokud je výstup podobný tomuto:

  ```
  první řádek
            druhý řádek
  ```

  pak uložte tento skript jako /var/spool/lpd/raw/filter:

  ```
  #!/bin/sh
  # Tento filtr eliminuje "efekt schodiště"
  awk '{print $0, "\r"}'
  ```

  a změňte jej na spustitelný pomocí `chmod 755 /var/spool/lpd/raw/filter`.

- vytvořte filtr pro emulaci PostScriptu. Vytvořte následující filtr jako /var/spool/lpd/postscript/filter:

  ```
  #!/bin/sh

  DEVICE=djet500
  RESOLUTION=300x300
  PAPERSIZE=a4
  SENDEOF=

  nenscript -TUS -ZB -p- |
  if [ "$DEVICE" = "PostScript" ]; then
    cat -
  else
    gs -q -sDEVICE=$DEVICE \
      -r$RESOLUTION \
      -sPAPERSIZE=$PAPERSIZE \
      -dNOPAUSE \
      -dSAFER \
  ```

```
    -sOutputFile=- -
fi

if [ "$SENDEOF" != "" ]; then
  printf "\004"
fi
```

(v tomto příkladu je předpokládána tiskárna HP DeskJet. Upravte si jej podle vaší tiskárny).

- nakonec přidejte následující položky do /etc/printcap:

```
# /etc/printcap
lp|ps|PS|PostScript|djps:\
       :sd=/var/spool/lpd/postscript:\
       :mx#0:\
       :lp=/dev/lp1:\
       :if=/var/spool/lpd/postscript/filter:\
          :sh:
raw:\
       :sd=/var/spool/lpd/raw:\
       :mx#0:\
       :lp=/dev/lp1:\
       :if=/var/spool/lpd/raw/filter:\
       :sh:
```

Pro složitější nebo exotičtější konfigurace tisku si přečtěte Praktický návod „Tisk v Linuxu“ (kapitola 8).

Pokud používáte `printtool`, uvědomte si, že GSDEVICE zvolený nástrojem Printtool bude pracovat, ale nemusí být tím nejlepším pro vaši tiskárnu. Můžete zkusit malý podvod se souborem `postscript.cfg`; například jsem změnil GSDEVICE z `cdj500` na `djet500` a nyní je můj tisk rychlejší.

SVGATextMode

Tento nástroj, který je k dispozici na ftp://tsx-11.mit.edu/pub/linux/sources/sbin, je užitečný pro změny rozlišení obrazovky konzoly, písma a tvaru kurzoru. Uživatelé, jejichž jazyk obsahuje znaky s diakritikou, je budou moci používat v aplikacích konzoly, zatímco uživatelé notebooků mohou změnit tvar kurzoru, aby byl viditelnější.

Upravte /etc/TextConfig nebo /etc/TextMode, kde začneme výchozí definicí VGA. Evropané by měli být šťastni za sekci „LoadFont“:

```
Option "LoadFont"
FontProg "/usr/bin/setfont"
FontPath "/usr/lib/kbd/consolefonts"
FontSelect "lat1u-16.psf"   8x16 9x16 8x15 9x15
FontSelect "lat1u-14.psf"   8x14 9x14 8x13 9x13
FontSelect "lat1u-12.psf"   8x12 9x12 8x11 9x11
FontSelect "lat1u-08.psf"   8x8  9x8  8x7  9x7
```

Jakmile je to hotovo, zkuste vaši konfiguraci s příkazem jako např. `SVGATextMode "80x34x9"`, a pokud to vypadá, že všechno funguje správně, odstraňte varování z /etc/TextMode a vložte tento řádek do etc/rc.d/rc.sysinit:

```
# SVGATextMode
```

```
/usr/sbin/SVGATextMode "80x34x9"
```

Pamatujte si, že obdélníkový kurzor pracuje pouze v některých režimech; na mém notebooku „80x30x9“.

Obvyklé úlohy správy

Tady začíná legrace. Tato část je zaměřená na síť, i když na vás čeká mnoho jiných úloh.

Sítě jsou široké téma, které zde nelze zcela pokrýt. Podívejte se na NET-HOWTO. Většina distribucí obsahuje dokumentaci k nastavování síťových služeb. Zde se budeme zabývat pouze několika body.

Krátký seznam co-dělat (to-do) pro služby, které byste mohli chtít nainstalovat: cron a načasované úlohy jako kalendář nebo připomínač, Http, Samba, telnet/ssh přístup, anonymní ftp, POP/IMAP server, NFS...

Konfigurace sítě

Pokud vaše síťová karta nebyla rozpoznána při instalaci, nebojte se: ve většině případů je kompatibilní s NE2000 nebo 3c59x. Spusťte příkaz `modprobe ne` nebo `modprobe 3c59x` a podívejte se, zda je načten relevantní modul, a pak přidejte tento řádek do /etc/modules.conf:

```
alias eth0 ne  # or 3c59x
```

Nyní jste připraveni použít `netcfg` nebo podobný nástroj pro konfiguraci sítě. Relevantní soubory jsou /etc/HOSTNAME, /etc/hosts, /etc/resolv.conf, /etc/sysconfig/network a /etc/sysconfig/network-scripts/ifcfg-eth0; služby by měly být spuštěny skriptem v /etc/rc.d/init.d.

Toto je příklad /etc/hosts:

```
127.0.0.1               localhost
192.168.1.1             paleo.eocene.net        paleo
192.168.1.2             nautilus.eocene.net     nautilus
```

Toto je /etc/resolv.conf:

```
search df.unibo.it,eocene.net
nameserver 195.210.91.100
```

Toto je /etc/sysconfig/network (Red Hat):

```
NETWORKING=false
FORWARD_IPV4=true
HOSTNAME=nautilus.eocene.net
DOMAINNAME=eocene.net
```

A konečně, /etc/sysconfig/network-scripts/ifcfg-eth0. Tento je také pouze v distribuci Red Hat; musí být spustitelný.

```
DEVICE=eth0
IPADDR=192.168.1.2
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
```

```
ONBOOT=no
```

Ačkoliv může být skutečná metoda spouštění síťových služeb vaší distribuce složitější, následující skript by měl být z počátku dostačující:

```
#!/bin/sh

# net-up.sh: set up network access

DEVICE=eth0
IPADDR=192.168.1.100
NETMASK=255.255.255.0
NETWORK=192.168.1.0
GATEWAY=192.168.1.1

ifconfig $DEVICE $IPADDR netmask $NETMASK up
route add -net $NETWORK netmask $NETMASK $DEVICE
route add default gw $GATEWAY
```

Tento skript je užitečný pro zapínání přístupu k síti, když používáte záchrannou disketu. Samozřejmě vám umožňuje pouze ping, ftp a telnet; nebude spouštět žádného daemona.

Síť pro notebooky

Když zasunete vaši síťovou PCMCIA kartu, spustí se skript /etc/pcmcia/network. Všechno, co potřebujete, je správně nastavený /etc/sysconfig/network-scripts/ifcfg-eth0.

Nastavování sítě ale může být trochu záludnější. Ve skutečnosti musíte zadat správná nastavení pro každou síť, ke které se připojujete, stejně jako nastavení notebooku, když není připojený.

Vytvořil jsem hrubé, ale funkční řešení. Používám můj notebook jako samostatný počítač a k síti se připojuji přes PPP; doma mám adresu IP 192.168.1.2 a na univerzitě adresu IP 137.204.x.y. Takže jsem vytvořil sadu konfiguračních souborů pro každou síť; všechny jsou uloženy v /etc/mobnet. Pro volbu pracovního prostředí používám skript. Toto je například /etc/mobnet/home.cfg:

```
# /etc/mobnet/home.conf

HOSTNAME=nautilus.eocene.net    # kompletní název hostitele
DOMAINNAME=eocene.net           # vaše doména
IPADDR=192.168.1.2
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
GATEWAY=192.168.1.1
FORWARD_IPV4=true
NAMESERVER=195.210.91.100       # vyžadováno
SEARCH=df.unibo.it,eocene.net   # volitelně
SERVICES="inet httpd smb sshd"
```

Toto je `mnet` – skript, který používám pro zvolení síťového profilu:

```
#!/bin/sh
# mnet: skript pro nastavení konfigurace "mobilní síť".
# Poslední úprava: 15. července 2000
```

```
# spuštění a zastavení služeb
activate_services()
{
  for service in $(echo $SERVICES) ; do
    [ -x /etc/rc.d/init.d/$service ] && /etc/rc.d/init.d/$service $1
  done
}

# využití
if [ $# = 0 ] ; then
  echo "Použití: mnet <config name>"
  echo "Příklad: mnet office"
  exit 1
fi

# kontrola zda konfigurace existuje
if [ ! -e /etc/mobnet/$1.conf ]; then
  echo "Tato konfigurace neexistuje."
  exit 1
fi

# načtení konfigurace
. /etc/mobnet/$1.conf

# nastavení názvu hostitele
echo $HOSTNAME > /etc/HOSTNAME
/bin/hostname $HOSTNAME

# nastavení názvového serveru(ů)
cat <<EOF > /etc/resolv.conf
# /etc/resolv.conf
search $SEARCH
nameserver $NAMESERVER
EOF

# zastavení předchozích služeb, pokud byly spuštěné
if [ -f /etc/mobnet/services.prev ]; then
  NEWSERVICES=$SERVICES
  . /etc/mobnet/services.prev
  activate_services stop
  SERVICES=$NEWSERVICES
fi

if [ $1 != "none" ]; then
# nastavení parametrů sítě
  cat <<EOF > /etc/sysconfig/network
  NETWORKING=yes
  FORWARD_IPV4=true
  HOSTNAME=$HOSTNAME
  DOMAINNAME=$DOMAINNAME
  GATEWAY=$GATEWAY
  GATEWAYDEV=eth0
EOF
```

```
  cat <<EOF > /etc/sysconfig/network-scripts/ifcfg-eth0
  DEVICE=eth0
  IPADDR=$IPADDR
  NETMASK=$NETMASK
  NETWORK=$NETWORK
  BROADCAST=$BROADCAST
  ONBOOT=no
EOF
  /bin/chmod +x /etc/sysconfig/network-scripts/ifcfg-eth0

  # zkopírování dalších konfiguračních souborů
  /bin/cp -f /etc/mobnet/hosts.$1       /etc/hosts
  /bin/cp -f /etc/mobnet/smb.conf.$1    /etc/smb.conf

  echo -n "Vložte síťovou PCMCIA kartu a stiskněte <enter>: "
  read

  # OK, nyní spustit služby
  activate_services start
  echo "SERVICES=\"$SERVICES\"" > /etc/mobnet/services.prev

else # it's not "none"

  cat <<EOF > /etc/sysconfig/network
  NETWORKING=false
  FORWARD_IPV4=false
  HOSTNAME=$HOSTNAME
  DOMAINNAME=$DOMAINNAME
EOF
  /bin/rm -f /etc/sysconfig/network-scripts/ifcfg-eth0*
  /sbin/ifconfig eth0 down
  echo "SERVICES=$SERVICES" > /etc/mobnet/services.prev
  echo "Nyní můžete vyjmout kartu PC."
  exit 0

fi

# konec mnet.
```

Jak jsem řekl, je neučesaný a nedodělaný: na síti mohou záviset další soubory, jako např. /etc/fstab, /etc/exports a /etc/printcap. Přemýšlejte o síťových tiskárnách a sdílených adresářích NFS. Volně si toto řešení přizpůsobte vašim potřebám.

Sdílení Internetu

Je to jedna z nejužitečnějších úloh serveru se systémem Linux. V tuto chvíli má většina jader firewall, maškarádování a předávání implicitně vypnuté; pokud jste na pochybách, podívejte se IP-Masquerade mini-HOWTO, kde se dozvíte, jak je zapnout. Pak nainstalujte ipfwadm (jádra 2.0.x; http://www.xos.nl/linux/ipfwadm/) nebo ipchains (jádra 2.2.x; http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html). Nezapomeňte zapnout moduly jádra pro služby, které potřebujete, např. pro ftp přidáte tento řádek do /etc/rc.d/rc.sysconfig:

```
/sbin/modprobe ip_masq_ftp
```

Další moduly jsou obvykle v /lib/modules/KERNEL-VERSION/ipv4.

Zapnutí maskování IP pro další počítače ve vaší místní síti je velmi jednoduché. Nejprve zkontrolujte skripty pro inicializaci sítě (měly by být v /etc/sysconfig/network) a podívejte se, zda obsahují řádek `FORWARD_IPV4=true`. Používá se pro nastavení /proc/sys/net/ipv4/ip_forward na 1 při startu síťového podsystému.

Přidejte tyto řádky do /etc/rc.d/rc.sysinit:

```
# implicitně: pakety se nemohou přeposílat ven
/sbin/ipfwadm -F -p deny
# umožní všem počítačům v místní síti přístup k internetu
/sbin/ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
# eventuálně mohou být povoleny pouze tyto dva počítače
# /sbin/ipfwadm -F -a m -S 192.168.1.100/24 -D 0.0.0.0/0
# /sbin/ipfwadm -F -a m -S 192.168.1.101/24 -D 0.0.0.0/0
```

Pokud používáte jádro série 2.2.x, použijte `ipfwadm-wrapper` namísto `ipfwadm` pro rychlé spuštění. Více informací najdete na adrese http://ipmasq.cjb.net.

(Pozn. odb. korektora: u novějších jader použijte spíše ipehains.)

Nyní budete chtít udělat něco, abyste počítačům klientů povolili vytáčení ISP; já používám Mserver (http://cpwright.villagenet.com/mserver/). V etc/mserver.conf upravujte pouze položky „checkhost“, „shadow“ a „cname“. Pak nadefinujte vaše připojení. Samozřejmě nainstalujte na klientské počítače odpovídajícího klienta.

Omezování přístupu ze sítě

Předpokládejme, že se připojujete k Internetu přes PPP. Jakmile jste připojeni, může být váš počítač zranitelný útoky. Vložte do /etc/hosts.allow:

```
# povolit přístup pouze pro localhost
ALL: 127.
```

a do /etc/hosts.deny:

```
# odepření přístupu všem
ALL: ALL
```

Pokud jste v síti s přímým přístupem k Internetu, měli byste z bezpečnostních důvodů vypnout finger, telnet a možná další služby; místo služby telnet použijte `ssh`. Je potřeba upravit /etc/inetd.conf. Eventuálně můžete omezit přístup ze sítě přidáním do /etc/hosts.allow:

```
in.telnetd: 192.168.1., .another.trusted.network
in.ftpd: 192.168.1., .another.trusted.network
```

a tohoto do /etc/hosts.deny:

```
in.telnetd: ALL
in.ftpd: ALL
```

Exporty NFS

Časté je exportování domovských adresářů na server; problém nastává, pokud UID a GID uživatele nejsou pro různé počítače konzistentní. Pokud uživatel `guido' má UID/GID = 500 na počítači `server` a UID/GID = 512 na počítači `client`, je vhodnou konfigurací toto:

```
# /etc/exports
/tmp            my.client.machine(rw)
/home/guido     my.client.machine(rw,all_squash,anonuid=512,anongid=512)
```

Samba

Je to skoro triviální, ale vždycky je tu potřeba něco malého udělat. Pokud se chcete připojovat ke klientům Windows 98/NT, přečetli jste si dokumentaci a, případně, zapnuli čistě textová hesla? Distribuce obsahuje soubory .reg pro Win9x/NT/2000; pokud se vaši klienti nemohou připojit k serveru se systémem Linux, načtěte je na každém klientovi.

Samba obsahuje docela kompletní příklad /etc/smb.conf, ale kupodivu postrádá část, která by vám ukázala, jak připojit nebo odpojit vyměnitelná média. Musíte použít klauzule `preexec` a `postexec`:

```
[cdrom]
  comment = CD-ROM
  path = /mnt/cdrom
  public = yes
  read only = yes
; možná budete muset použít "root preexec/postexec"
  preexec = mount /mnt/cdrom
  postexec = umount /mnt/cdrom
```

Takže: víte, co je to Swat, že ano? Zapněte jej přidáním tohoto řádku do vašeho /etc/inetd.conf:

```
swat      stream  tcp     nowait.400      root /usr/sbin/swat swat
```

a tohoto do `/etc/services`:

```
swat            901/tcp
```

Restartujte `inetd` s SIGHUP a nasměrujte váš prohlížeč na `http://localhost:901`.

Konfigurace softwaru

Zde jsou konfigurační soubory, které budeme upravovat: /etc/profile /etc/bashrc `.bashrc .bash_profile .bash_logout .inputrc .less .lessrc .xinitrc .fvwmrc .fvwm2rc95 .Xmodmap .Xmodmap.num .Xdefaults .jedrc .abbrevs.sl .joerc .emacs`. Nepřidávejte uživatele, dokud nedokončíte tuto konfiguraci vašeho systému; soubory umístíte do /etc/skel.

bash(1)

Je to nejdůležitější část softwaru po jádře. Pro přizpůsobení chování `bash` jsou zde tyto hlavní soubory:

- /etc/bashrc obsahuje systémové aliasy a funkce;
- /etc/profile obsahuje prostředí systému a programy po spuštění;
- `$HOME/.bashrc` obsahuje aliasy uživatelů a funkce;
- `$HOME/.bash_profile` obsahuje uživatelské prostředí a programy po spuštění;
- `$HOME/.inputrc` obsahuje vazby kláves a další věci.

Níže jsou uvedeny příklady těchto souborů. První je nejdůležitější: /etc/profile. Používá se pro konfiguraci mnoha funkcí systému Linux, jak uvidíte v následujících částech. Hledejte prosím obrácené uvozovky!

```
# /etc/profile

# System wide environment and startup programs
# Functions and aliases go in /etc/bashrc

# This file sets up the following features and programs:
# path, prompts, a few environment variables, colour ls, less,
# rxvt, Backspace key behaviour, xterm title.
#
# Users can override these settings and/or add others in their
# $HOME/.bash_profile

# first: root or normal user? Set PATH and umask accordingly. Note that the
# PATH is normally set by login(1), but what if you access the machine
# via ssh?

if [ $(id -gn) = $(id -un) -a $(id -u) -gt 14 ]; then
  umask 002  # normal user
  PATH="/usr/local/bin:/bin:/usr/bin:."
else
  umask 022  # root
  PATH="/sbin:/bin:/usr/sbin:/usr/bin"
fi

# Now extend the PATH.
PATH="$PATH:/usr/X11R6/bin:$HOME/bin:." # !!! Beware of ./ !!!

# notify the user: login or non-login shell. If login, the prompt is
# blue; otherwise, magenta. Root's prompt is red.
# See the Colour-ls mini HOWTO for an explanation of the escape codes.
USER=$(whoami)
if [ $LOGNAME = $USER ] ; then
  COLOUR=44  # blue
else
  COLOUR=45  # magenta
fi

if [ $USER = 'root' ] ; then
  COLOUR=41  # red
  PATH="$PATH:/usr/local/bin" # my choice
fi
```

```
ESC="\033"
PROMPT='\h'    # hostname
STYLE='m'      # plain
# PROMPT='\u'  # username
# STYLE=';1m'  # bold
PS1="\[$ESC[$COLOUR;37$STYLE\]$PROMPT:\[$ESC[37;40$STYLE\]\w\\$ "
PS2="> "

# Ulimits: no core dumps, max file size 200 Mb.
ulimit -c 0 -f 200000

# a few variables
USER=$(id -un)
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"  # sendmail, postfix, smail
# MAIL="$HOME/Mailbox"        # qmail
NNTPSERVER=news.myisp.it      # put your own here
VISUAL=jed
EDITOR=jed
HOSTNAME=$(/bin/hostname)
HISTSIZE=1000
HISTFILESIZE=1000
export PATH PS1 PS2 USER LOGNAME MAIL NNTPSERVER
export VISUAL EDITOR HOSTNAME HISTSIZE HISTFILESIZE

# enable colour ls
eval $(dircolors /etc/DIR_COLORS -b)
export LS_OPTIONS='-s -F -T 0 --color=yes'

# customize less
LESS='-M-Q'
LESSEDIT="%E ?lt+%lt. %f"
LESSOPEN="| lesspipe.sh %s"
LESSCHARDEF=8bcccbcc13b.4b95.33b. # show colours in ls -l | less
# LESSCHARSET=latin1
PAGER=less
export LESS LESSEDIT LESSOPEN VISUAL LESSCHARDEF PAGER

# you might need this to fix the backspace key in rxvt/xterm
stty erase ^H  # alternative: ^?

# set xterm title: full path
case $TERM in
  xterm*|rxvt)
    PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME}: ${PWD}\007"'
    ;;
esac

for i in /etc/profile.d/*.sh ; do
  if [ -x $i ]; then
    . $i # beware - variables and aliases might get overridden!
  fi
done
```

```
# call fortune, if available
if [ -x /usr/games/fortune ] ; then
echo ; /usr/games/fortune ; echo
fi
```

Toto je příklad /etc/bashrc:

```
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile
# Insert PS1 definitions here if you experience problems.

export CDPATH="$CDPATH:~"

# common aliases
alias cp='cp -i'
alias l=less
alias ls="ls $LS_OPTIONS"
alias mv='mv -i'
alias rm='rm -i'
alias rmbk='/bin/rm -f .*~ *~ *aux *bak *log *tmp 2> /dev/null'
alias u='cd ..'
alias which="type -path"
alias x=startx

# A few useful functions
c ()    # cd to the new directory and list its contents
{
  cd $1 ; ls
}

inst()  # Install a .tar.gz archive in current directory
{
  if [ $# != 0 ]; then tar zxvf $1; fi
}

cz()    # List the contents of a .zip archive
{
  if [ $# != 0 ]; then unzip -l $*; fi
}

ctgz()  # List the contents of a .tar.gz archive
{
  for file in $* ; do
    tar ztf ${file}
  done
}

tgz()   # Create a .tgz archive a la zip.
{
  if [ $# != 0 ]; then
    name=$1.tar; shift; tar -rvf ${name} $* ; gzip -9 ${name}
```

```
  fi
}

crpm()  # list information on an .rpm file
{
  if [ $# != 0 ]; then rpm -qil $1 | less; fi
}
```

Toto je příklad `.bashrc`:

```
# $HOME/.bashrc
# Source global definitions

if [ -f /etc/bashrc ]; then
  . /etc/bashrc
fi

# this is needed to notify the user that they are in non-login shell
if [ "$GET_PS1" = "" ] ; then
  COLOUR=45; ESC="\033"; STYLE=';1m';  # STYLE='m'
  USER=$(whoami)
  export PS1="\[$ESC[$COLOUR;37$STYLE\]$USER:\[$ESC[37;40$STYLE\]\w\\$ "
fi

# personal aliases
alias backup='tar -Mcvf /dev/fd0'
alias dial='eznet up myisp'
alias f='cd ~/fortran'
alias hangup='eznet down'
alias lyx='lyx -width 580 -height 450'
alias restore='tar -M -xpvf /dev/fd0'

# personal functions
xj()    # Launch xjed and a file in background
{
  xjed $1 &
}
```

Toto je příklad `.bash_profile`:

```
# $HOME/.bash_profile

# User specific environment and startup programs
# This file contains user-defined settings that override
# those in /etc/profile

# Get user aliases and functions
if [ -f ~/.bashrc ]; then
  GET_PS1="NO"  # don't change the prompt colour
  . ~/.bashrc
fi

# set a few 'default' directories
export CDPATH="$CDPATH:$HOME:$HOME/text:$HOME/text/geology"
```

Toto je příklad .inputrc:

```
# $HOME/.inputrc

# key bindings
"\e[1~": beginning-of-line
"\e[3~": delete-char
"\e[4~": end-of-line
# (F1 .. F5) are "\e[[A" ... "\e[[E"
"\e[[A": "info \C-m"

set bell-style visible          # please don't beep
set meta-flag On                # allow 8-bit input (i.e, accented letters)
set convert-meta Off            # don't strip 8-bit characters
set output-meta On              # display 8-bit characters correctly
set horizontal-scroll-mode On   # scroll long command lines
set show-all-if-ambiguous On    # after TAB is pressed
```

Aby klávesy backspace a delete pracovaly správně v xterm a dalších aplikacích X11, je potřeba také následující:

- vložte toto do vašeho .xinitrc:

  ```
  usermodmap=$HOME/.Xmodmap
  xmodmap $usermodmap
  ```

- pak bude váš .Xmodmap obsahovat:

  ```
  keycode 22 = BackSpace
  keycode 107 = Delete
  ```

  to opraví konzolu. Pro opravu xterm:

- vložte toto do vašeho .Xdefaults:

  ```
  xterm*VT100.Translations: #override <Key>BackSpace: string(0x7F)\n\
          <Key>Delete:        string(0x1b) string("[3~")\n\
          <Key>Home:          string(0x1b) string("[1~")\n\
          <Key>End:           string(0x1b) string("[4~")\n\
          Ctrl<Key>Prior:     string(0x1b) string("[40~")\n\
          Ctrl<Key>Next:      string(0x1b) string("[41~")

  nxterm*VT100.Translations: #override <Key>BackSpace: string(0x7F)\n\
          <Key>Delete:        string(0x1b) string("[3~")\n\
          <Key>Home:          string(0x1b) string("[1~")\n\
      <Key>End:           string(0x1b) string("[4~")\n\
          Ctrl<Key>Prior:     string(0x1b) string("[40~")\n\
          Ctrl<Key>Next:      string(0x1b) string("[41~")
  ```

rxvt je trochu komplikovanější, jelikož některé volby zadané při kompilaci ovlivňují jeho chování. Podívejte se na /etc/profile nahoře.

Více informací o bash(1) a readline(3) najdete v manuálových stránkách.

Nečekejte, že bude každá aplikace pracovat správně! Pokud spustíte joe v xterm, některé klávesy například nebudou pracovat; to samé platí pro některé verze rxvt.

I18n

(Tato část se netýká anglicky mluvících uživatelů)

Je to známo také jako „lokalizace“. Uf! Toto slovo znamená „přizpůsobení systému Linux vašim místním konvencím: jazyku, formátu data, měně atd“'.

Ačkoliv má Red Hat svou vlastní metodu nastavování i18n (/etc/sysconfig/i18n), možná budete chtít zapnout váš jazyk pouze v některých případech. Třeba budu chtít zapnout i18n v `kdm` (pomocí `kdmconfig`) a `xfce`, ale když pracuji v konzole nebo xterm, chci číst zprávy v angličtině.

Podívejte se na tyto řádky:

```
LANG=it # zvolte svůj jazyk: fr, de, es, ...
LANGUAGE=it
LC_ALL=it
export LANG LANGUAGE LC_ALL
```

Pokud je přidáte do vašeho `.xinitrc` nebo `.xsession` před řádek, kde se spouští správce oken, obdržíte lokalizované zprávy – včetně těch v xterm spuštěném ze správce oken. Ale pokud byste raději dostávali zprávy v angličtině, nastavte jazyk na „en“ a vložte ty samé řádky do `.bash_profile`.

ls(1)

`ls` může zobrazovat výpisy adresářů pomocí barev pro odlišení různých druhů souborů. Pro zapnutí této funkce potřebujete právě řádky v /etc/profile uvedené výše. Toto samozřejmě nebude pracovat se staršími verzemi `rxvt`; místo toho použijte některou verzi `xterm`. Vypadá to, jako by některé staré `rxvt` měly chybu, která jim brání v některých případech v odvozování prostředí.

less(1)

S tímto excelentním pagerem můžete procházet nejen čistě textové soubory, ale také komprimované archívy gzip, tar a zip, manuálové stránky a další. Jeho konfigurace zahrnuje několik kroků:

- pro jeho použití s klávesami pro pohyb mějte tento čistě v ASCII soubor `.lesskey` ve vašem domovském adresáři:

  ```
  ^[[A   back-line
  ^[[B   forw-line
  ^[[C   right-scroll
  ^[[D   left-scroll
  ^[OA   back-line
  ^[OB   forw-line
  ^[OC   right-scroll
  ^[OD   left-scroll
  ^[[6~  forw-scroll
  ^[[5~  back-scroll
  ^[[1~  goto-line
  ^[[4~  goto-end
  ^[[7~  goto-line
  ^[[8~  goto-end
  ```

  pak spusťte příkaz `lesskey` (Jsou to escape sekvence pro terminály vt100). Toto vytvoří binární soubor `.less` obsahující vazby kláves.

- uložte následující soubor jako /usr/bin/lesspipe.sh:

```
#!/bin/sh
# This is a preprocessor for 'less'.  It is used when this environment
# variable is set:   LESSOPEN="|lesspipe.sh %s"

lesspipe() {
  case "$1" in
  *.tar) tar tf $1 2>/dev/null ;; # View contents of .tar and .tgz files
  *.tgz|*.tar.gz|*.tar.Z|*.tar.z) tar ztf $1 2>/dev/null ;;
  *.Z|*.z|*.gz) gzip -dc $1  2>/dev/null ;; # View compressed files co-
rrectly
  *.bz2) bzip2 -dc $1  2>/dev/null ;;
  *.zip) unzip -l $1 2>/dev/null ;; # View archives
  *.arj) unarj -l $1 2>/dev/null ;;
  *.rpm) rpm -qpil $1 2>/dev/null ;;
  *.cpio) cpio --list -F $1 2>/dev/null ;;
  *.1|*.2|*.3|*.4|*.5|*.6|*.7|*.8|*.9|*.n|*.l|*.man) FILE='file -L $1'
    FILE='echo $FILE | cut -d ' ' -f 2'
    if [ "$FILE" = "troff" ]; then
      groff -s -p -t -e -Tascii -mandoc $1
    fi ;;
  *) file $1 | grep text > /dev/null ;
    if [ $? = 1 ] ; then # it's not some kind of text
      strings $1
    fi ;;
  esac
}

lesspipe $1
```

pak jej učiňte spustitelným nastavením `chmod 755 lesspipe.sh`.

- umístěte proměnné, které ovlivňují `less` do /etc/profile, jak jste viděli výše.

Editor

Budou zde zmíněny pouze nejpopulárnější.

emacs(1)

Málokdy používám `emacs`, takže mám pro vás jen pár tipů. Některé distribuce `emacs` nemají předkonfigurované barvy a zvýrazňování syntaxe. Umístěte toto do vašeho `.emacs`:

```
(global-font-lock-mode t)
(setq font-lock-maximum-decoration t)
```

Toto pracuje pouze v X11. Kromě toho pro zapnutí znaků s diakritikou přidáte tento řádek:

```
(standard-display-european 1)
```

Nechám na vás pročtení veškeré dokumentace k `emacs` a nalezení toho, jak si jej přizpůsobit vlastním potřebám---také to může zabrat měsíce hackování. Dobrým pomocníkem je generátor Dotfile (viz. Konfigurační souftware).

joe(1)

Některé verze `joe` nepracují v konzole barevně a nefungují v nich některé speciální klávesy. Rychlým a nečistým (a neelegantním) řešením předchozího problému je toto:

```
~$ export TERM=vt100
~$ joe myfile
   (edituje váš soubor)
~$ export TERM=linux
```

Aby fungovaly speciální klávesy, všechno, co potřebujete udělat, je upravit `.joerc`, `.jstarrc` nebo vaši oblíbenou emulaci; můžete začít systémovými konfiguračními soubory v /usr/lib/joe. Hledejte čtvrtou sekci (vazby kláves). Následující umožní Home a End:

```
bol ^[ [ 1 ~    Go to beginning of line
eol ^[ [ 4 ~    Go to end of line
```

Najděte požadované escape sekvence pomocí `cat` následovaným speciálními znaky.

jed(1)

To je můj oblíbený editor: dělá to, co potřebuju, je malý a jednodušeji nastavitelný než `emacs` a celkem dobře emuluje jiné editory. Mnoho uživatelů na mé univerzitě používá `jed` pro emulaci `EDT`, systémového editoru VMS.

Konfigurační soubory editoru `jed` jsou `.jedrc` a /usr/lib/jed/lib/*; dříve uvedený může být přizpůsoben z `jed.rc` v uvedeném adresáři.

- pokud `xjed` zřejmě nerozpoznává klávesu DEL, přidejte nebo zakomentujte tyto řádky ve vašem `.jedrc`:

  ```
  #ifdef XWINDOWS
    x_set_keysym (0xFFFF, 0, "\e[3~");
    setkey (''delete_char_cmd'', "\e[3~");
  #endif
  ```

- aby `jed` emuloval `EDT` (nebo jiné editory), všechno, co musíte udělat, je upravit pár řádek v `.jedrc`. Pokud chcete, aby „+“ na numerické klávesnici odstraňovalo slova místo jednotlivých znaků, přidejte do `.jedrc`:

  ```
  unsetkey("\eOl");
  unsetkey("\eOP\eOl");
  setkey("edt_wdel", "\eOl");
  setkey("edt_uwdel", "\eOP\eOl");
  ```

  za řádek, který obsahuje `() = evalfile("edt")` (nebo něco podobného);

- aby `xjed` používal numerickou klávesnici pro emulaci `EDT`, vložte do `.Xmodmap`:

  ```
  keycode 77  = KP_F1
  keycode 112 = KP_F2
  keycode 63  = KP_F3
  keycode 82  = KP_F4
  keycode 86  = KP_Separator
  ```

- přizpůsobení barev `xjed` se dělá přidáním takovýchto řádků do `.Xdefaults`:

```
xjed*Geometry: 80x32+150+50
xjed*font: 10x20
xjed*background: midnight blue
# and so on...
```

- zkratky funkcí neocenitelně šetří čas. Vytvořte soubor podobný tomu následujícímu jako `$HOME/.abbrevs.sl` (tento název můžete změnit přidáním `variable Abbrev_File = "/usr/lib/jed/abbrev.sl";` do `.jedrc`):

```
create_abbrev_table ("Global", "0-9A-Za-z");
define_abbrev ("Global", "GG", "Guido onzato");
create_abbrev_table ("TeX", "\\A-Za-z0-9");
define_abbrev ("TeX", "\\beq", "\\begin{equation}");
define_abbrev ("TeX", "\\eeq", "\\end{equation}");
% a tak dále...
```

  a napište `ESC x abbrev_mode` pro jeho zapnutí. Pro implicitní zapnutí zkratek přidejte položky jako jsou tyto do vašeho `.jedrc`:

```
define text_mode_hook ()
{
  set_abbrev_mode (1);
}
%
define fortran_hook ()
{
  set_abbrev_mode (1);
  use_abbrev_table ("Fortran");
}
% a tak dále...
```

pine(1)

Upravte globální konfiguraci v /usr/lib/pine.conf a dávejte pozor alespoň na následující pole: `user-domain`, `smtp-server` a `nntp-server`. Všimněte si, že `inbox-path` závisí na vašem MTA: pokud používáte `sendmail` nebo `postfix`, bude zde `/var/spool/mail/$USER`; v Qmail /home/$USER/Mailbox (ale root bude používat /var/qmail/alias/Mailbox.

minicom(1)

Uživatelé nemohou `minicom` používat, dokud root nevytvoří globální konfiguraci. Nezapomeňte ji vytvořit.

efax(1)

Tento balíček je pravděpodobně nejvhodnější pro jednoduché odesílání a přijímání faxů. Budete muset upravit skript /usr/bin/fax nebo (v mandrake) /etc/fax.config; je to jednoduché, ale z několika věcí mě docela rozbolela hlava:

- pro zjištění toho, zda je váš modem třídy 1, 2, nebo 2.0 použijte minicom nebo podobný program pro spuštění příkazu `at+fclass=?`. Odpověď může být `0,1` nebo `2`; 1 a 2 jsou třídy podporované vaším modemem;

- DIALPREFIX: je možné, že jednoduché zadání `T' nebo `P' nebude fungovat v některých zemích – přinejmenším v Itálii. Místo toho zadejte `ATDT' nebo `ATDP';
- INIT a RESET: tyto řetězce obsahují inicializátory `-i' a `-k', které potřebuje efax. Pokud chcete přidat příkaz AT, přidejte jej do příslušného řetězce vynecháním `AT' a přeřazením `-i' nebo `-k' před zbytek. Příklad: pro přidání příkazu `ATX3' do INIT připojíte `-iX3'.

Když je to hotovo, je potřeba opravit několik oprávnění, aby mohli i jiní uživatelé než root odesílat a přijímat faxy. Do adresářů /var/lock a /var/spool/fax musí být možno zapisovat. K tomu vytvořte skupinu faxusers, přidejte do ní uživatele a pak napište:

```
~# chown root.faxusers /var/lock
~# mkdir /var/spool/fax # if it doesn't exist yet
~# chown root.faxusers /var/spool/fax; chmod g+w /var/spool/fax
```

Jako normální uživatel spustíte před odesíláním faxu newgrp faxusers.

Ghostscript

Tento dokonalý nástroj trpí malým problémem. Vzhledem k dobře známé regulaci exportu z USA utilita pdf2ps nepracuje se zašifrovanými soubory .pdf. Nevadí: nasměrujte váš prohlížeč na http://www.ozemail.com.au/~geoffk/pdfencrypt, stáhněte soubor pdf_sec.ps a nahraďte soubor se stejným názvem, který pochází z distribuce Ghostscript.

TeX a spol.

„Kořenem" systému TeX je adresář $TEXMF, který je /usr/share/texmf v teTeX; ostatní distribuce se mohou lišit (hledejte ve vašem systému „texmf").

Rozšiřování $TEXINPUTS

Pro vložení postscriptových prvků nebo souborů TeX, které jsou uloženy v podadresářích, je vhodné rozšířit cestu hledání TeXu tak, aby obsahovala podadresáře. Vložte tento příkaz do vašeho .bash_profile:

```
export TEXINPUTS="$HOME/mylib::./figures"
```

což umožní programu TeX hledání v $HOME/mylib *před* implicitními adresáři a v adresáři ./figures *později*.

Vzorky dělení

Pro nastavení vzorků dělení pro váš jazyk upravte soubor $TEXMF/tex/generic/config/language.dat a pak proveďte:

```
~# texconfig init ; texconfig hyphen
```

I když nepíšete v angličtině, neodstraňujte položku „english"; TeX se bez ní neobejde.

Dvips

Pro úpravu dvips je potřeba upravit soubor $TEXMF/dvips/config/config.ps. Buďte si vědomi toho, že pole výchozího rozlišení také ovlivňují chování xdvi; pokud se setkáte s otravnými pokusy o vytváření písem pokaždé, když jej spustíte, vložte řádek

```
XDvi*mfmode:
```

Do vašeho `.Xdefaults`. Mělo by to pomoci.

Přidávání balíčků LaTeX

Další balíčky LaTeX jsou k dispozici na nejbližším zrcadle CTAN (Comprehensive TeX Archive Network), např. ftp://ftp.dante.de/pub/tex. Rozbalte balíček do $TEXMF/tex/latex.

Pokud neexistuje soubor .sty, spusťte příkaz `latex newstyle.ins` nebo `latex newstyle.dtx`, abyste jej vytvořili, pak spusťte příkaz `texhash`, aby teTeX rozpoznal nový balíček.

Vyhněte se PPProblémům!

Budu považovat za samozřejmé, že má vaše jádro zakompilovanou podporu PPP a TCP/IP, že loopback je zapnutý a že již máte správně nainstalovaný balíček `pppd` a, pokud chcete, nastavené uid root. Váš ISP samozřejmě musí podporovat PPP.

Existují dva způsoby uvedení PPP do chodu: a) ruční konfigurace, a b) konfigurační program, který to dělá automaticky. Ať už zvolíte cokoliv, připravte si následující informace:

- telefonní číslo vašeho ISP;
- název vašeho ISP, adresu poštovního serveru a adresu serveru s diskusními skupinami;
- doménu vašeho ISP;
- vaše uživatelské jméno a heslo.

Ruční konfigurace je dřina. Jsou to úpravy souborů a vytváření skriptů; není to moc práce, ale je snadné se splést a začátečníci se často bojí. Existuje PPP HOWTO. Eventuálně existují nástroje, které se vás zeptají na výše uvedené údaje a udělají všechnu práci.

Gnome a KDE obsahují, podle pořadí, `gnome-ppp` a `kppp`, pomocí kterých je nastavování jednoduché. Eventuálně vám doporučuji se podívat na pár nástrojů založených na tty, `wvdial` a `eznet`. Zadáte jim telefonní číslo vašeho ISP, vaše jméno a vaše heslo a je to hotovo. Jejich domovské stránky jsou na adresách http://www.worldvisions.ca/wvdial a http://www.hwaci.com/sw/eznet. Oba jsou skvělé, ale doporučuji ten druhý.

Rychlý start pomocí eznet

Nejdříve vytvořte /etc/resolv.conf s tímto:

```
nameserver w.x.y.z
```

kam zadáte adresu jmenného serveru ISP. Pro vytvoření účtu pomocí `eznet` spusťte následující příkaz:

```
#~ eznet add service=YOUR_ISP user=NAME password=PASSWORD phone=PHONE
```

který vytvoří soubor /var/eznet/eznet.conf, jehož vlastníkem je root.root s oprávněním 600; změňte je pomocí chmod na 666 pokud chcete, aby k nim byl přístup. Nyní vytočte číslo vašeho ISP pomocí `eznet up YOUR_ISP`. Pokud modem stále čeká na oznamovací tón a nepřipojí se, pak zkuste tento příkaz:

```
#~ eznet change YOUR_ISP init0=atx3
```

Příkaz pro zavěšení je `eznet down`. To je vše!

Rychlý start pomocí wvdial

Nastavení `wvdial`'s je ještě kratší. Napište `wvdialconf /etc/wvdial.conf`, pak upravte výsledný soubor, aby obsahoval vaše uživatelské jméno, heslo a telefonní číslo. Zkuste `wvdial` a mějte překřížené prsty. Pro zavěšení jej zastavte pomocí Ctrl-C.

Klient POP

Pro načtení vaší pošty ze serveru POP3 potřebujete klienta POP. Většina těchto klientů vyžaduje, abyste měli spuštěný MTA jako `sendmail`, `qmail` nebo `postfix`; to je trochu náročné na počítačích nižší úrovně. Samozřejmě existují klienti, kteří pracují bez MTA. První dobře reprezentuje `fetchmail`; druhé `fetchpop` nebo `frenchie`. Servery: ftp://www.ibiblio.org/pub/Linux/system/mail/pop, http://www.lowcountry.com/~jscottb/tcltk.shtml.

Pro nastavení těchto klientů:

- `fetchpop`: když jej spustíte poprvé, požádá vás o nějaké informace. Odpovězte na otázky a bude nastavený. `fetchpop` musí být použit s přepínačem `-r`, pokud server POP3 vašeho ISP nepodporuje příkaz LAST správně.
- `frenchie`: stejný, upravte /.frenchie/frenchierc;
- `fetchmail`: upravte tento vzorový `.fetchmailrc`:

```
# $HOME/.fetchmailrc
poll mbox.myisp.com with protocol pop3;
  user john there with password _Loo%ny is john here
```

Jeden uživatel říkal, že přidání „smtphost localhost" na druhý řádek dramaticky zvýšilo výkon. Musíte nastavit oprávnění pro tento soubor pomocí příkazu `chmod 600 .fetchmailrc`, jinak se `fetchmail` odmítne spustit. Tento příklad je jen základní; existují nekonečné možnosti konfigurace. Podívejte se na http://www.ccil.org/~esr/fetchmail.

Základní filtrování pošty

Budete se chtít chránit před nevyžádanou poštou nebo velkými zprávami. Existují dva případy: 1) trvalé připojení k síti a 2) připojení POP. V prvním případě můžete vytvořit soubor `.procmailrc`, zatímco v druhém případě existují nástroje pro kontrolu pošty ještě před jejím stažením.

Velmi jednoduchý `.procmailrc`, který definuje nová pravidla:

```
# $HOME/.procmailrc

MAILDIR=$HOME/mail # make sure it exists

# Store messages directed to the "foo" mailing list to $HOME/mail/foo
:0
* ^To:.*foo
foo

# Discard messages that are not explicitly sent to me or to one of the
# mailling lists I subscribed to.
:0
* !^TO(guido|jed|lugvr|ldp|nobody)
/dev/null

# ditto, for messages larger than 50k.
```

```
:0
* > 50000
/dev/null
```

Pro více příkladů spusťte `man procmailex`.

Uživatelé POP budou chtít použít `poppy`, užitečný skript v jazyce Perl pro kontrolu pošty před jejím stažením. Najdete si jej na http://www.freshmeat. .net/.

Window System (XFree86)

Nastavování X Server

Pojďme na to, už to není tak obtížné, jako to bývalo... Všechny velké distribuce obsahují nástroj pro nastavování X11 (např. `XConfigurator`, `sax`, `XF86Setup` nebo alespoň `xf86config`). Konfigurace X je dnes vlastně automatická, ale z některých videokaret vás může rozbolet hlava.

Nejprve zkontrolujte na serveru XFree86 (http://www.xfree86.org), zda je vaše videokarta podporována. Pokud ano, zkuste tuto proceduru:

- nainstalujte server s normální VGA;
- jděte na ftp://ftp.XFree86.org/pub/XFree86/current/binaries, přejděte do správného podadresáře a stáhněte archívy `X_version_bin.tgz`, `X_version_set.tgz` a všechny servery. Kromě jiných programů první archív obsahuje nejnovější `SuperProbe`;
- rozbalte `X_version_bin.tgz` do dočasného adresáře, přejděte do něj a spusťte `./SuperProbe`. Pokud je vaše videokarta rozpoznána, je šance, že ji budete moci nastavit. Jinak hodně štěstí;
- nainstalujte servery a `X_version_set.tgz` z /usr/X11R6/, pak spusťte `XF86Setup`.

Vždycky mi to fungovalo, ale nemusí to být jednoduché. Pamatujte si prosím, že se většinou X11 nespustí, protože jste zvolili špatné parametry vašeho monitoru! Začněte s konzervativním nastavením, tj. 800x600 a 256 barev, pak ho zvedněte. *Varování*: tyto operace jsou nebezpečné a váš monitor se může poškodit!

Pokud vaše karta není podporována, můžete: 1) čekat na další verzi XFree86, 2) zakoupit komerční server X, 3) zakoupit podporovanou videokartu. *Quartum non datur.*

Numerická klávesnice

Výše jste viděli, jak rozchodit některé speciální klávesy. Vzorový soubor `.Xmodmap` pracuje dobře, pokud chcete použít Xjed, ale dělá numerickou klávesnici nefunkční. Pak budete potřebovat jiný konfigurační soubor, kterému budeme říkat `.Xmodmap.num`:

```
! Definitions can be found in <X11/keysymdef.h>

keycode 77  = Num_Lock
keycode 112 = KP_Divide
keycode 63  = KP_Multiply
keycode 82  = KP_Subtract
keycode 86  = KP_Add
keycode 79  = KP_7
keycode 80  = KP_8
keycode 81  = KP_9
keycode 83  = KP_4
keycode 84  = KP_5
```

```
keycode 85  = KP_6
keycode 87  = KP_1
keycode 88  = KP_2
keycode 89  = KP_3
keycode 90  = KP_0
keycode 91  = KP_Decimal
```

Ujistěte se, že váš soubor /etc/X11/XF86Config neobsahuje tyto tři řádky:

```
ServerNumLock
Xleds
XkbDisable
```

V případě, že ano, tak je zakomentujte. Pro znovuzapnutí numerické klávesnice spustíte příkaz `xmodmap .Xmodmap.num`.

Grafické přihlášení s xdm

Abyste byli přivítáni grafickým přihlášením, upravte soubor /etc/inittab, který by měl zahrnovat řádek podobný tomuto:

```
x:5:respawn:/usr/bin/X11/xdm -nodaemon # též kdm nebo gdm
```

kde 5 je runlevel odpovídající X11. Upravte řádek, který definuje výchozí runlevel (obvykle 2 nebo 3), a změňte jej:

```
id:5:initdefault:
```

Počet barev je specifikován v /etc/X11/xdm/Xserver:

```
:0 local /usr/X11R6/bin/X :0 -bpp 16 vt07  # první server X, 65k barev
:1 local /usr/X11R6/bin/X :1 -bpp 32 vt08  # druhý server X, true colour
```

Pokud již máte `.xinitrc`, zkopírujte jej do `.xsession` a učiňte jej spustitelným pomocí `chmod +x .xsession`. Nyní spusťte příkaz `telinit 5` a je to hotovo.

Správce oken - Window Manager

Jakmile X pracuje, jsou zde nekonečné možnosti konfigurace; závisí to na správci oken, který používáte, jsou jich na výběr desítky. Většinou se to dělá úpravami v jednom nebo více souborech textovýcg ve vašem domovském adresáři; v jiných případech nemusíte nic upravovat a použijete applet nebo dokonce nabídku.

Některé příklady:

- **rodina fvwm**: zkopírujte /etc/X11/fvwm/system.fvwmrc (nebo podobný) do vašeho domovského adresáře, přičemž použijte příslušný název, prohlédněte si jej a začněte experimentovat. Může trvat hodně času, než to bude vypadat přesně tak, jak chcete;
- **WindowMaker**: obsahuje více konfiguračních souborů, které jsou uloženy v `$HOME/GNUstep`, a skvělý konfigurační applet;
- **KDE**, **Gnome**, **xfce** a další: nic se zde neupravuje ručně, všechno lze udělat skrze nabídky.

Krátce: pokud nechcete upravovat konfigurační soubory, zvolte něco jako `icewm`, `fvwm*`, `blackbox` atd.; pokud ano, je volba aktuálně omezena na KDE, Gnome, WindowMaker, a Xfce. Opravte mě, pokud se mýlím.

Je důležité mít dobrý `.xinitrc`. Příklad:

```
#!/bin/sh
# $HOME/.xinitrc

usermodmap=$HOME/.Xmodmap
xmodmap $usermodmap

xset s noblank  # vypnutí spořiče
xset s 300 2    # spořič se spustí za 5 min.
xset m 10 5     # nastavení rychlosti myši

rxvt -cr green -ls -bg black -fg white -fn 7x14 \
  -geometry 80x30+57+0 &

if [ "$1" = "" ] ; then  # implicitně
  WINMGR=wmaker
else
  WINMGR=$1
fi

$WINMGR
```

Ačkoliv to nevypadá, že by to bylo striktně vyžadováno, udělejte jej spustitelným pomocí `chmod +x .xinitrc`.

Výše uvedený `.xinitrc` vám umožní vybrat správce oken: vyzkoušejte

```
$ startx startkde # nebo jiný správce oken
```

Implicitní hodnoty pro aplikace X11

Najděte adresář implicitních nastavení (měl by to být /usr/X11R6/lib/X11/app-defaults). Některé aplikace sem ukládají konfigurační soubory.

Přidávání písem

Poslední verze XFree86 (řekněme, > 3.3.4) používají X Font Server, který sám podporuje písma PostScript Type 1 a True Type, takže můžete použít spousty písem dostupných na webu. Existuje pro to jednoduchá procedura.

Předpokládejme, že jste stáhli sadu písem Type 1, např. Freefont (ftp://ftp.gimp.org/pub/gimp/fonts/freefonts-0.10.tar.gz). Aby je viděl server písem, rozbalte archív do /usr/X11R6/lib/X11/fonts/. Pak upravte /etc/X11/fs/config, přidejte položku pro nový adresář a restartujte server písem.

Pokud si vytváříte svou vlastní sadu písem, budete potřebovat doplnit soubory `fonts.dir` a `fonts.scale`; to se dělá pomocí nástroje `typelinst`, který je k dispozici na adrese http://http://goblet.anu.edu.au/~m9305357/type1inst.html.

Co se týká písem True Type, seskupte je do adresáře, který si vyberete a vytvořte `fonts.dir` pomocí `ttmkfdir > fonts.dir`, zahrnutého v archívu Freetype; http://www.freetype.org. Pak proveďte to samé. Například, pokud chcete používat stejná písma jako ve Windows, řekněme, /mnt/win/windows/fonts, přejděte do tohoto adresáře, spusťte `ttmkfdir`, upravte /etc/X11/fs/config a restartujte server písem.

To všechno začalo na původním serveru písem X True Type: /http://www.dcs.ed.ac.uk/home/jec/programs/xfsft/.

Uživatelské konfigurace

Když máte upravené konfigurační soubory, zkopírujte je do /etc/skel jak jste viděli v části Konfigurace software.

Vytváření balíčků .rpm

`rpm` je tak skvělá metoda pro udržení balíčků pod kontrolou, že nejsem ochoten instalovat z archívů .tar.gz. Pouze ve velmi málo speciálních případech ano (např. zabezpečení). Kdykoliv instalujete tar archív, zvažte jeho převod na archív .rpm a pak jej znovu naistalujte; podívejte se do RPM HOWTO. Také pokud používáte nejnovější verze `gcc`, může být vhodné vložit toto do vašeho `/etc/rpmrc`:

```
optflags: i386 -O2 -mpentiumpro
```

Inovace

Pokud inovujete počítač, proveďte zálohu jako obvykle a uložte několik dalších souborů. To mohou být /etc/X11/XF86Config, /usr/bin/fax, veškerý obsah /usr/local, konfigurace jádra, celý adresář /etc a všechnu poštu v /var/spool/mail.

Nyní můžete inovovat (ve vyjímečných případech kazit!) aplikace, se kterými se dodává vaše distribuce, a přidat další balíčky. Udržujte si jejich seznam.

Konfigurační software a dokumentace

Existuje více programů, které zjednodušují nastavení a konfiguraci systému Linux. Některé se staly standardem: Red Hat, Caldera a další distribuce obsahují aplikace jako `setup`, `printtool`, `netcfg`, `usertool`, atd., zatímco SuSE dodává všeobecný konfigurační program YaST. Další užitečné programy jsou:

- **The Dotfile Generator**: pěkná aplikace X s moduly pro konfiguraci balíčků jako `emacs`, `bash`, `procmail` a dalších. Jeho stránky jsou na adrese http://www.imada.ou.dk/~blackie/dotfile;
- **Linuxconf**: poslední konfigurační nástroj. Umí dělat všechno; jak v konzole, tak pod X. Podívejte se na http://www.solucorp.qc.ca/linuxconf.

Dokumenty o konfiguraci systému Linux se objevují všude. Jedním z nejlepších je TrinityOS, http://www.ecst.csuchico.edu/~dranch/LINUX/index-linux.html. Otravujte autora, aby udělal svůj dokument v hezčích formátech.

Opravdu dobrá stránka je http://dotfiles.com. Přesně, co říkají – sada konfiguračních souborů.

Na konec

Copyright

Zpětná vazba

Možná více než jiná HOWTO, toto potřebuje a vítá vaše doporučení, kritiku a příspěvky. Zpětná vazba není jen vítána – je nutná. Pokud si myslíte, že něco chybí nebo je špatně, napište mi prosím. Pokud máte distribuci jinou než Red Hat/Mandrake a vaše konfigurační soubory jsou odlišné nebo umístěné v jiných adresářích, dejte mi prosím vědět a já vaše tipy zařadím. Mým záměrem je zjednodušení života se systémem Linux jak jen to bude možné.

Linux obsahuje mnoho balíčků, takže není možné zahnout pokyny pro všechny. Držte se při vašich požadavcích a doporučeních „nejpřijatelnějších“ programů---Nechám to na vašem uvážení.

Záruka

Tento dokument je poskytován „tak jak je“. Opravdu se ho snažím psát co nejpřesněji, ale informace zde obsažené používáte na své riziko. V žádném případě neodpovídám za případná poškození vyplývající z použití tohoto dokumentu.

Chtěl bych poděkovat autorům všech ostatních HOWTO a autorům a správcům manuálových stránek, které jsem nestydatě „vykrádal“, a všem ostatním lidem, kteří mi provádí zpětnou vazbu.

Doufám, že vám tento dokument bude užitečný. Vždycky, když provádím novou instalaci systému Linux, skutečně dělám...

Užijte si jej,

Guido =8-)

Jádro Linuxu

Originál: http://tldp.org/HOWTO/Kernel-HOWTO.html

Úvod

Patříte mezi ty, kteří by si měli tento dokument přečíst? Ano, pokud jste se setkali s některou z následujících situací:

- „Tento balíček wizzo-46.5.6 tvrdí, že potřebuje jádro 2.8.193 a já mám pořád jádro 1.0.9!"
- V jednom z těch nových jader je ovladač zařízení, které prostě musíte mít.
- Nemáte ani trochu představu o tom, jak přeložit jádro.
- „Jsou ty nesmysly v souboru README *opravdu* pravdivé?"
- Přišli jste, zkusili jste, a ono to nefunguje.
- Potřebujete nějak odbýt lidi, kteří po vás chtějí nainstalovat nové jádro.

Toto si opravdu přečtěte!!!

Některé příklady v tomto dokumentu předpokládají, že máte nainstalovány nástroje `tar`, `find` a `xargs`. Jedná se o naprostý standard – s tímto požadavkem by neměly být žádné problémy. Předpokládáme také, že se vyznáte ve struktuře souborového systému – pokud ne, je velmi důležité, abyste si uschovali výstup příkazu mount (či obsah souboru /etc/fstab, pokud k němu máte přístup). Tato informace je důležitá a nezmění se, pokud nebudete přerozdělovat pevný disk, přidávat či přeinstalovávat operační systém či něco podobného.

Poslední produkční verze jádra je v okamžiku vzniku tohoto dokumentu 2.2.9, takže veškeré odkazy a příklady vychází z tohoto jádra. I přesto jsme se v mezích možností snažili tento dokument vytvořit tak, aby nebyl na konkrétní verzi jádra závislý. Jádro je neustále ve vývoji a každá nová verze vyvolá několik změn. Tato skutečnost by neměla způsobit příliš vážné problémy, ale drobné zmatky možná ano.

Existují dvě základní verze linuxového jádra – produkční a vývojová. Produkční verze jsou označovány sudým číslem na druhém místě v čísle verze. Produkční verze jsou tak 1.2.x, 2.0.x, 2.2.x či 2.4.x. Tyto verze jsou stabilní a bezchybné[1]. Vývojové verze (například 2.1.x, 2.3.x atd.) jsou jádra určená pro testování a odhalování chyb.

Typografické konvence

Text v tomto formátu reprezentuje výstupy na obrazovce, jména souborů a cokoli, co zadáváte z klávesnice, jako jsou příkazy či volby příkazů. Příkazy a jejich výstupy jsou někdy uvozeny apostrofy, což způsobuje následující klasický problém: pokud se tečka objeví na konci věty, často ji uživatelé zadají společně s příkazem. Americká typografická konvence totiž přikazuje umisťovat

[1] Pozn. překladatele: Ale aspoň se o to snaží...

tečku v rámci citace. Příklady v tomto dokumentu vychází z toho, že tečka není součástí citace – pokud tedy bude nutno zadat příkaz „make config", bude v tomto dokumentu uvedeno 'make config' , případně (make config) a ne 'make config.'

Stručný přehled překladu jádra

Tuto kapitolu napsal Al Dev (alavoor@yahoo.com). Nejnovější verzi najdete na adrese http://www.milkywaygalaxy.freeservers.com/. Zrcadlené stránky jsou na http://www.angelfire.com/country/aldev0 a http://www.geocities.com/alavoor/index.html. Na těchto adresách najdete spoustu užitečných informací.

Překlad jádra zajišťuje, že jádro není rozsáhlé a ve výsledku je tak Linux rychlejším operačním systémem. Pomocí překladu jádra je také umožněna podpora nových zařízení.

Předběžná opatření

Před překladem jádra je vhodné zálohovat systém. Pokud jste jej ještě nezálohovali, tak to udělejte teď. Můžete použít komerční zálohovací programy, například BRS Backup-Recovery-Software (http://24.221.230.253/) – na těchto stránkách naleznete i odkazy na open-source / freeware zálohovací nástroje. Zálohování systému je pouze návrh a před překladem jádra není nezbytné.

Pro netrpělivé

1. Rozbalte zdrojové kódy
2. make clean
3. make xconfig
4. make dep
5. make
6. make bzImage
7. make modules
8. make install
9. make modules_install
10. Nakonfigurujte LILO nebo GRUB

Podrobnosti viz následující text.

Překlad jádra – krok za krokem

Podrobnosti o výše uvedeném postupu.

> **POZNÁMKA:** Zápis 'bash#' označuje příkazový řádek příkazového interpretu bash, měli byste zadávat až příkazy uvedené za tímto zápisem. Následující příkazy byly testovány na distribuci RedHat Linux s jádrem 2.4.7-10, ale měly by být bez větších úprav funkční také na jiných distribucích. Měly by fungovat i pro starší jádra, jako 2.2, 2.0 a 1.3.

1. Uvědomte si, že můžete mít ve vašem systému více než jedno jádro. Následujícím postupem si nepřepíšete ani si nezničíte vaše současné jádro. Tento postup je **zcela bezpečný** a současné jádro nebude nijak dotčeno.

2. Přihlaste se jako uživatel root. Připojte jednotku CD-ROM a nainstalujte zdrojové balíčky s jádrem:

```
bash$ su - root
bash# cd /mnt/cdrom/RedHat/RPMS
bash# rpm -i kernel-headers*.rpm
bash# rpm -i kernel-source*.rpm
bash# rpm -i dev86*.rpm
bash# rpm -i bin86*.rpm
```

   Balíčky bin86*.rpm a as86 jsou nutné pouze pro starší verze Linuxu, jako je například RedHat 5.x. Intel assembler as86 můžete získat z balíčku dev86*.rpm nebo z adres http://rpmfind.net/linux/RPM/mandrake/7.1/Mandrake/RPMS/bin86-0.4-12mdk.i586.html a http://rpmfind.net/linux/RPM/kondara/jirai/i586/bin86-0.4-8k.i586.html.

3. Spusťte X-Window systém příkazem *startx*. Pokud nemůžete X-Window systém spustit, přejděte na následující krok:

```
bash# man startx
bash# startx
bash# cd /usr/src/linux
bash# make xconfig
```

   Nemůžete-li spustit X-Window systém, zkuste

```
bash# export TERM=xterm
bash# make menuconfig
```

   Dojde-li k chybnému zobrazení na obrazovce, zkuste emulovat jiný terminál, například vt100, vt102, vt220 nebo ansi. Pokud se pomocí telnetu přihlašujete ke vzdálenému systému, rozhodně musíte použít emulátory jako vt100 nebo vt220.

   Například:

```
bash# export TERM=vt220
bash# export TERM=ansi
```

   Ještě primitivnější varianta je

```
bash# export TERM=vt100
bash# make menuconfig
```

   Pokud by volba menuconfig nefungovala, zkuste

```
bash# make config
```

   Příkazy make xconfig nebo make menuconfig spustí konfigurační program s uživatelsky přívětivým rozhraním. Příkaz make config má řádkové rozhraní. Konfiguraci jádra můžete nahrát ze souboru */usr/src/linux/.config*.

4. V rámci příkazu *make xconfig* potřebujete provést následující kroky, jinak se pravděpodobně nevyhnete problémům:
   - Zvolte správný typ procesoru – Pentium 3, AMD K6, Cyrix, Pentium 4, Intel 386, DEC Alpha, PowerPC a podobně, jinak se vám jádro ani nepodaří spustit.
   - Správně nastavte podporu SMP – máte jeden procesor, nebo více procesorů?
   - Souborové systémy – jako součást jádra (ne jako moduly) zvolte Windows 95 VFAT, MSDOS a NTFS (to je jen mé osobní doporučení).

- Povolte podporu modulů! S touto volbou budete moci za běhu systému dynamicky zavádět potřebné ovladače zařízení. Viz manuálové stránky:

```
bash# rpm -i /mnt/cdrom/Redhat/RPMS/modutils*.rpm
bash# man lsmod
bash# man insmod
bash# man rmmod
bash# man depmod
bash# man modprobe
```

5. Uložte změny a ukončete práci s konfiguračním programem. Všechny provedené změny jsou nyní uloženy do konfiguračního souboru /usr/src/linux/.config. Nyní zadejte:

```
bash# make dep
bash# make clean
```

6. Prostudujte následující soubor (obsahuje mnoho informací o sestavování jádra).

   TIP: Pro lepší čitelnost můžete použít editor *gvim*, který podporuje barevné zobrazení.

```
bash# gvim -R /usr/src/linux/arch/i386/config.in
bash# man less
bash# less /usr/src/linux/arch/i386/config.in
```

   Pro nápovědu stiskněte h a pro navigaci používejte písmena i, j, k, l, h nebo šipky a klávesy PAGE UP/PAGE DOWN.

7. Nyní proveďte vlastní sestavení jádra pomocí příkazu make:

```
bash# cd /usr/src/linux
bash# man nohup
bash# nohup make bzImage &
bash# tail -f nohup.out (.... pro sledování průběhu)
```

   Tento příkaz vytvoří jádro do souboru /usr/src/linux/arch/i386/boot/bzImage.

8. Poté, co je obraz jádra bzImage úspěšně vytvořen, zkopírujte jej do adresáře /boot. Tato podmínka je nutná, protože v opačném případě by nebylo možné jádro pro spuštění systému použít. V systému RedHat můžete použít dva způsoby zavedení systému – Lilo nebo Grub. Poté si prostudujte nápovědu k příkazu lilo (viz dokument http://www.linux-doc.org/HOWTO/LILO-crash-rescue-HOWTO.htm) a ukázkový konfigurační soubor lilo.conf. Vždy do názvu souboru s jádrem přidávejte informace o tom, kdy bylo jádro sestaveno.

```
bash# cp /usr/src/linux/arch/i386/boot/bzImage /boot/bzImage.myker.26mar2001
bash# man lilo
bash# man lilo.conf
```

   Do souboru `lilo.conf` přidejte následující řádky:

```
image=/boot/bzImage.myker.26mar2001
label=myker
root=/dev/hda1
read-only
```

   Název startovacího zařízení pro položku root můžete zjistit takto:

```
bash# df /boot
```

9. Nyní zadejte:

```
bash# lilo
bash# lilo -q
```

   Program lilo musíte spustit vždy, když je opětovně přeloženo jádro bzImage.

10. Restartujte počítač a po stisku tabulátoru v úvodní nabídce programu lilo napište myker. Pokud se operační systém spustí, bylo vše provedeno správně. V opačném případě vyberte původní jádro a vše proveďte znovu. Vaše původní jádro zůstalo nedotčené v souboru /boot/vmlinuz-2.0.34-0.6.

11. Pokud se jádro zavede a funguje správně, můžete si vytvořit bootovací disketu. Vložte do mechaniky prázdnou disketu a zadejte:

```
bash# cd /usr/src/linux
bash# make bzdisk
```

   Viz též mkbootdisk:

```
bash# rpm -i mkbootdisk*.rpm
bash# man mkbootdisk
```

12. Moduly – tento krok platí pouze v případě, že jste povolili podporu modulů podle kroku 3. Moduly jsou umístěny v adresáři /lib/modules. Tento krok MUSÍTE provést, pokud jste povolili nebo zakázali moduly, jinak při nebo po zavedení jádra obdržíte hlášení „unresolved symbols“. Zkontrolujte existenci příkazu insmod, který se pro nahrávání modulů používá nejčastěji:

```
bash# cd /usr/src/linux
bash# make modules
bash# make modules_install
```

   Tím zkopírujete moduly do adresáře `/lib/modules`. Pokud budete chtít například nahrát modul `/lib/modules/2.4.2-2/kernel/drivers/block/loop.o`, zadáte:

```
bash# man insmod
bash# modprobe loop
bash# insmod loop
bash# lsmod
```

Adresáře, které program insmod prohledává, můžete nastavit v souboru `/etc/modules.conf`.

Řešení běžných chyb

Systém se zasekne v LILO

Symptom: Po přeložení jádra a restartu systému se počítač zasekne ještě před spuštěním LILO.

Důvod: Pravděpodobně nejsou v BIOSu správně nastaveny konfigurace primárních a sekundárních pevných disků.

Řešení: Znovu zapněte počítač a stiskem klávesy Del vyvolejte SETUP. Zvolte nastavení IDE a správně nastavte parametry primárního a sekundárních disků. Když systém startuje, hledá primární IDE disk a na něm pak tzv. Master Boot Record. Ten načte a zahájí načítání jádra Linuxu z pevného disku.

No init found

Následující chyba se přihodí mnoha novým uživatelům. Přeložené jádro se nespustí a objeví se následující hlášení:

```
Warning: unable to open an initial console
Kernel panic: no init found. Try passing init= option to kernel
```

Tento problém je nejčastěji způsoben špatně nastavenou hodnotou parametru root = v souboru `/etc/lilo.conf`. Ve výše uvedeném příkladě jsme použili nastavení `root=/dev/hda1`, ale ve vašem případě tento parametr může nabývat jiných hodnot, jako je `/dev/hdb2` či `/dev/hda7`.

Jádro se pokouší spustit příkaz init v adresáři `/sbin`, který je umístěn právě na kořenovém oddílu. Další informace získáte z příslušné manové stránky:

```
bash# man init
```

„depmod" vrací chybu „Unresolved symbol"

Po spuštění příkazu depmod se objevuje chyba *Unresolved symbol*. Může to vypadat například takto:

```
bash$ su - root
bash# man depmod
bash# depmod
depmod: *** Unresolved symbols in /lib/modules/version/kernel/drivers/md/linear.o
depmod: *** Unresolved symbols in /lib/modules/version/kernel/drivers/md/multipath.o
depmod: *** Unresolved symbols in /lib/modules/version/kernel/drivers/md/raid0.o
depmod: *** Unresolved symbols in /lib/modules/version/kernel/drivers/md/raid1.o
depmod: *** Unresolved symbols in /lib/modules/version/kernel/drivers/md/raid5.o
```

Důvod: Po přeložení nového jádra příkazem *make bzImage* jste nepřeložili moduly příkazem *make modules* a nenainstalovali jste je.

Řešení: Po přeložení jádra musíte zadat:

```
bash$ su - root
bash# cd /usr/src/linux
bash# make modules
bash# make modules_install
```

Jádro nenahraje modul – dojde k chybě „Unresolved symbol"

Po zavedení jádra při pokusu o nahrání modulů se objevuje chybové hlášení „Unresolved symbol: _název_nějaké_funkce". Znamená to, že nedošlo k úplnému překladu jádra a modulů. Je nutné před samotným překladem jádra zadat příkaz **make clean** a po něm přeložit moduly. Postup je následující:

```
bash# cd /usr/src/linux
        bash# make dep
        bash# make clean
        bash# nohup make bzImage &
        bash# tail -f nohup.out     (.... pro sledování průběhu)
        bash# make modules
        bash# make modules_install
```

Jádro nemůže nahrát modul

Pokud jádro nemůže nahrát nějaký modul (například ovladač síťové karty nebo jiného zařízení), můžete vyzkoušet přeložit ovladač tohoto zařízení přímo jako součást jádra. V některých případech *nefungují moduly* a ovladač musí být přeložen jako součást jádra. Například ovladače některých síťových karet nepodporují možnost být zavedeny jako moduly a MUSÍ být přeloženy pří-

mo v jádře. V rámci konfigurace příkazem *make xconfig* u těchto zařízení nesmíte zvolit podporu modulem.

Po přeložení jádra

Po úspěšném překladu a spuštění nového jádra budete možná muset provést některé z následujících kroků, jinak vám příslušná zařízení nemusí fungovat správně. (Postupy byly testovány na distribuci Red Hat Linux, měly by ale fungovat i na jiných distribucích.)

Konfigurace videokarty/monitoru

- Podívejte se do manuálu videokarty a hledejte část „Technické specifikace".
- Podívejte se do manuálu monitoru a hledejte část „Technické specifikace".

Videokartu a monitor můžete nakonfigurovat následujícími příkazy:

```
bash$ su - root
bash# man Xconfigurator
bash# /usr/bin/X11/Xconfigurator --help
bash# /usr/bin/X11/Xconfigurator
bash# /usr/bin/X11/Xconfigurator --expert
```

Viz též:

```
bash# man xf86config
bash# /usr/bin/X11/xf86config
```

Pokud systém nedokáže vaši grafickou kartu detekovat automaticky, použijte parametr – expert a zvolte „Unlisted card". Pokud váš monitor není v seznamu známých monitorů, zvolte obecný SVGA 1024x768.

Konfigurace zvukové karty

- Připojte reproduktory k výstupu zvukové karty.
- Připojte audiokabel z CD mechaniky na zvukovou kartu (jinak nebudete moci přehrávat audio CD).
- Přečtěte si HOWTO dokumenty věnované zvuku.

```
bash$ su - root
bash# man sndconfig
bash# /usr/sbin/sndconfig
```

Pokud používáte KDE, příkazem *K Start->ControlCenter->SoundServer->General->Test Sound* přehrajete testovací zvuk. Příkazem *K Start->MultiMedia->SoundMixer->SoundVolumeSlider* můžete upravit hlasitost.

Konfigurace síťové karty

- Použijte /sbin/linuxconf
- nebo použijte ovládací panely KDE
- přečtěte si HOWTO dokumenty věnované problematice sítí

Konfigurace firewallu a maškarády

U jader 2.4 a vyšších jsou firewall a IP maškaráda implementovány balíkem NetFilter. V konfiguraci jádra tedy musíte povolit NetFilter a pak spustit příslušné konfigurační skripty, které najdete na adrese http://www.boingworld.com/workshops/linux/iptables-tutorial. Hlavní stránky balíku NetFilter najdete na http://netfilter.samba.org/. Další informace najdete například na adresách http://www.linuxsecurity.com/feature_stories/kernel-netfilter.html a http://netfilter.filewatcher.org/netfilter-faq.html.

U jader nižších než 2.4 byste měli zvolit instalaci firewallu přes RPM balíky, které najdete na http://rpmfind.net/linux/rpm2html/search.php?query=firewall nebo http://rpmfind.net/linux/RPM/contrib/noarch//SRPMS//firewall-2.2-3.src.html.

Konfigurace dalších zařízení

Přečtěte si příslušné dokumenty HOWTO.

Ukázkový soubor lilo.conf

Pro soubor /etc/lilo.conf byste měli dodržovat konvenci názvosloví – pro jádro 2.2.17 ker2217, pro jádro 2.2.14 ker2214 atd. Na jednom systému můžete mít více obrazů jádra:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=firewall

image=/boot/vmlinuz-2.2.14-5.0
         label=ker2214
         read-only
         root=/dev/hda9

image=/boot/vmlinuz-2.2.17-14
         label=ker2217
         read-only
         root=/dev/hda9

#image=/usr/src/linux/arch/i386/boot/bzImage
#        label=myker
#        root=/dev/hda7
#        read-only

image=/boot/bzImage.myker.11feb2001
         label=myker11feb
         root=/dev/hda9
         read-only

image=/boot/bzImage.myker.01jan2001
         label=myker01jan
         root=/dev/hda9
         read-only
```

```
image=/boot/bzImage.myker-firewall.16mar2001
        label=firewall
        root=/dev/hda9
        read-only
```

Důležité otázky a odpovědi

O co se jádro stará

Unixové jádro je prostředníkem mezi vašimi programy a vaším hardwarem. Jednak zajišťuje správu paměti pro všechny spuštěné programy (procesy) a dále se stará o férové (či neférové, máte -li ten pocit) sdílení strojového času procesoru. Navíc poskytuje přenositelné rozhraní pro přístup vašich programů k hardwaru.

Pochopitelně se jádro stará o mnohem více věci, ale tyto základní funkce jsou nejdůležitější.

Kdy je nutné provést aktualizaci jádra

Nové verze jádra nabízejí podporu většího množství hardwaru (v Unixu se k hardwaru přistupuje přes tzv. ovladače zařízení), mohou mít vylepšenou správu procesů, mohou být rychlejší oproti předchozím verzím, měly by být stabilnější než předchozí verze a snad odstraňují chyby předchozích verzí. Mnoho uživatelů aktualizuje jádro právě z důvodů podpory nových zařízení či odstranění chyb.

Jaký druh hardwaru podporuje nové jádro

Podívejte se do souboru Hardware-HOWTO. Nebo se také můžete podívat do souboru config.in, případně na výstup příkazu make config. Zde zjistíte veškerý standardní distribucí podporovaný hardware, ale ne veškerý hardware, který Linux podporuje. Mnoho běžných ovladačů zařízení (jako jsou ovladače pro PCMCIA a páskové jednotky) je řešeno pomocí modulů a jsou distribuovány odděleně.

Jakou musíme mít verzi překladače gcc a knihovny libc

Linus Torvalds informuje o požadavcích na překladač a knihovny v souboru README. Pokud nemáte tuto verzi, dokumentace k požadované verzi gcc by vám měla poskytnout informace o tom, zda je nutné inovovat také knihovnu libc. Toto není obtížná procedura, ale je velmi důležité dodržet přesný postup.

Co jsou to moduly

Existuje kód jádra, který ale není přilinkován (vložen) přímo do vlastního jádra. Tyto moduly jsou překládány samostatně a mohou být připojovány a odpojovány za běhu operačního systému. Vzhledem k této pružnosti je dnes tento postup pro rozšiřování funkčnosti jádra doporučován. Mnoho populárních ovladačů zařízení, jako jsou ovladače karet PCMIA či ovladač páskové jednotky QIC-80/40, je realizováno právě pomocí modulů.

Kolik diskového prostoru jádro vyžaduje

Požadavek na diskový prostor velmi závisí na konkrétní konfiguraci Linuxu. Komprimovaný zdrojový kód pro verzi 2.2.9 je veliký 14 MB, ovšem na mnoha serverech je uložena nekomprimovaná podoba. Nekomprimovaný a přeložený tvar pro běžné nastavení zabírá okolo 67 MB.

Kolik času překlad jádra vyžaduje

Na novějších počítačích je oproti minulosti překlad jádra mnohem rychlejší. Na konfiguraci s procesorem AMD K6-2/300 s rychlým diskem trvá překlad jádra 2.2.x přibližně čtyři minuty. Na počítačích s procesory Pentium, 486 či 386 se připravte na čekání, možná v hodinách, možná ve dnech...

Pokud tolik času nemáte a máte přístup k rychlejšímu počítači, můžete jádro přeložit na něm a následně je přenést na pomalejší počítač (pochopitelně musíte při překladu použít správné parametry a ověřit si, že jednotlivé utility nejsou příliš zastaralé).

Jak tedy jádro vytvořit

Jak získat zdrojový kód

Zdrojový kód jádra můžete získat ze serveru ftp.kernel.org. Jednotlivé verze jádra jsou k dispozici v adresáři /pub/linux/kernel/vx.y, kde x.y je verze (například 2.2), a jak bylo řečeno v kapitole 1, lichou poslední číslicí jsou označovány vývojové verze a ty mohou být nestabilní. Typický název souboru se zdrojovým kódem jádra je linux-x.y.z.tar.gz, kde x.y.z je číslo verze. Na serverech s distribucemi velmi často bývají také k dispozici soubory s příponou .bz2, které jsou komprimovány pomocí programu bzip2 (tyto soubory bývají menší, a tak vyžadují méně času na stažení).

Nejlepší je použít adresu serveru ftp.xx.kernel.org, kde xx je kód vašeho státu, například ftp.cz.kernel.org pro Českou republiku či ftp.us.kernel.org pro Spojené státy americké.

Rozbalení zdrojového kódu

Přihlaste se jako uživatel root a přejděte do adresáře /usr/src (pomocí příkazu cd /usr/src). Pokud jste spolu s první instalací Linuxu instalovali také zdrojový kód jádra (tak tomu je ve většině běžných případů), měl by v tomto adresáři existovat podadresář linux, který obsahuje zdrojový kód nainstalovaného jádra. Pokud máte k dispozici dostatek místa na disku a chcete mít jistotu možnosti návratu a překladu této starší verze, uchovejte si obsah tohoto podadresáře. Rozumný přístup je přejmenovat tento podadresář podle verze jádra, které je aktuálně nainstalováno a spuštěno. Verzi aktuálně spuštěného jádra zjistíte pomocí příkazu uname –r. Tedy, pokud tento příkaz zobrazí 1.0.9, můžete původní podadresář linux přejmenovat na linux-1.0.9 (pomocí příkazu mv). Pokud si myslíte, že zálohu nebudete potřebovat, můžete celý podadresář linux odstranit. Před rozbalením nového jádra si ověřte, že v adresáři /usr/src žádný podadresář linux neexistuje.

Nyní můžete v adresáři /usr/src dekomprimovat zdrojový kód jádra pomocí příkazu tar zxpvf linux-x.y.z.tar.gz (pokud máte soubor s příponou .tar a ne .gz, použijte příkaz tar xpvf linux-x.y.z.tar). Obsah archivu bude rozbalen do správné adresářové struktury. Po dokončení se zobrazí opět příkazová řádka – přejděte do adresáře linux a prostudujte si soubor README. V tomto souboru je sekce nazvaná INSTALLING the kernel, ve které jsou uvedeny důležité informace – jaké mají být vytvořeny symbolické odkazy, informace o odstranění zastaralých souborů *.o atd.

Pokud máte k dispozici soubor s příponou .bz2 a program bzip2 (http://www.muraroa.deamon.co.uk), použijte příkaz:

```
bzcat linux-x.y.z.tar.bz2 | tar xvf -
```

Nastavení jádra

POZNÁMKA: Některé informace uvedené v této kapitole doplňují či opakují informace uvedené v příslušném souboru README od Linuse Torvaldse.

Příkaz make config zadaný v adresáři `/usr/src/linux` spustí konfigurační skript, který vám položí několik otázek. Pokud tento skript vyžaduje bash, tak zkontrolujte, že současný příkazový interpret je uložen v `/bin/bash`, `/bin/sh` či `$BASH`.

Samozřejmě existuje několik příjemnějších alternativ k příkazu `make config` a vy je můžete oprávněně považovat za komfortnější a jednodušší. Pravděpodobně světově nejpoužívanější možností je příkaz `make menuconfig`. Bez ohledu na to, jakou možnost si zvolíte, je velmi vhodné seznámit se s daným rozhraním, protože se k němu můžete vrátit dříve, než jste předpokládali. Pokud máte spuštěn systém X-window, můžete zkusit příkaz `make xconfig`. Příkaz `make menuconfig` je pro ty, co preferují textově orientované nabídky. Tato rozhraní mají jednu základní výhodu – pokud zvolíte špatnou volbu během konfigurace, není problém svůj omyl jednoduše napravit. Jednotlivé volby jsou v obou těchto případech zobrazovány v podobě hierarchicky orientované nabídky.

Na většinu otázek budete odpovídat ano (y-es) či ne (n-o), ovladače zařízení zpravidla navíc nabízí možnost volby `m`, tedy možnost použití modulu místo přímého zařazení k vlastnímu jádru. Takový ovladač je tedy sice přeložen, ale jako samostatný modul. Některé zlé jazyky tuto volbu překládají jako možná (maybe). Některé zřejmé a nepříliš důležité volby zde popsány nejsou – podívejte se do části věnované dalším volbám, ve které jsou popsány některé další volby. Pokud jste zvolili cestu s `make menuconfig`, pro označení voleb můžete používat mezerník.

Ve verzích 2.0.x a novějších se objevila volba `?`, která poskytuje krátký popis daného konfiguračního parametru. Tyto popisy bývají velmi aktuální.

Emulace matematických operací v jádru (Typ procesoru a některé vlastnosti) { Kernel math emulation (Processor type and features)}

Pokud nemáte k dispozici matematický koprocesor (máte tedy samotný procesor 386 či 486SX), musíte na tuto otázku odpovědět kladně (y). Pokud koprocesor k dispozici máte a přesto odpovíte kladně, nic se neděje – je používán koprocesor a emulace je ignorována. Pro většinu současných počítačů by měla být odpověď záporná (n), ale jak již bylo řečeno, pokud odpovíte kladně, nic se nestane.

Rozšířená podpora disků MFM/RLL a IDE disků/CD-ROM (bloková zařízení) { Enhanced (MFM/RLL) disk and IDE disk/cdrom support (Block Devices)}

Pravděpodobně budete tuto podporu potřebovat. Tato volba říká, že jádro bude podporovat standardní pevné disky pro osobní počítače (a ty má většina uživatelů). Tato volba nezahrnuje podporu zařízení SCSI (o nich více později).

Budete dotázáni na podporu pouze starých disků (`old disk only`) a nových IDE (`new IDE`) zařízení. Vyberte jednu volbu – rozdíl je v tom, že starší ovladač podporuje pouze dva disky na jednom rozhraní, kdežto nový ovladač podporuje druhé rozhraní a zařízení IDE/ATAPI CD-ROM. Nový ovladač je o 4 KB větší a obsahuje jiné množství chyb, může zvýšit výkon vašeho disku, především tehdy, pokud máte k dispozici nejnovější hardware (EIDE).

Podpora sítí
{ Networking support (General Setup)}

V zásadě byste měli odpovídat kladně v případě, že váš počítač je připojen k Internetu či se k Internetu připojujete pomocí vytáčené linky a protokolů SLIP, PPP, TERM apod. Mnoho balíčků (jako je X-window) vyžaduje podporu sítí i když nejste připojení ke skutečné síti. V tomto případě byste také měli odpovídat kladně. Pokud si nejste absolutně jisti svým rozhodnutím, odpovězte kladně i na otázku (bude položena později) týkající se podpory sítí založených na sadě protokolů TCP/IP.

Systém V IPC
{ System V IPC (General Setup)}

Jedna z nejlepších definic IPC (Interprocess Communication) je uvedena ve slovníčku knihy o Perlu. To není nijak překvapující, někteří programátoři v Perlu využívají IPC k vzájemné komunikaci mezi procesy, stejně jako mnoho dalších balíčků (nejvíce proslulý je například DOOM), není tedy dobrý nápad odpovědět záporně, pokud zcela přesně nevíte, co děláte.

Typ procesoru
{ Processor family (Processor type and features)}

(U starších jader použijte volbu –m486 k optimalizaci pro 486.)

Tradičně je překlad optimalizován pro konkrétní procesor. Jádra mohou správně fungovat také na dalších procesorech, ale pravděpodobně pak bude jádro o něco větší. V novějších jádrech toto již není pravdivé tvrzení, měli byste tedy určit, pro jaký procesor bude jádro překládáno. Jádro pro 386 bude správně fungovat na všech počítačích.

Podpora zařízení SCSI
{ SCSI support}

Pokud máte v počítači nějaká zařízení SCSI, odpovězte na tento dotaz kladně. Budete dotázáni na další doplňující informace, jako je podpora jednotek CD-ROM, pevných disků a jaký typ adaptéru SCSI máte k dispozici. Více informací najdete v dokumentu SCSI-HOWTO.

Podpora síťových zařízení
{ Network device support}

Pokud máte ve svém počítači síťovou kartu či budete chtít používat protokoly SLIP, PPP či paralelní adaptéry pro připojení k Internetu, odpovězte kladně. Následně budete dotázáni na konkrétní typ vaší síťové karty a jaký protokol budete používat.

Souborové systémy
{ Filesystems}

Konfiguračním skriptem budete dotázáni na typy podporovaných souborových systémů:

Standard (minix) – novější distribuce nevytvářejí tento souborový systém a mnoho uživatelů jej nepoužívá, ale může být pořád důvod na výběr této volby. Používají jej totiž některé záchranné programy a pořád ještě mnoho disket může být naformátováno pod tímto systémem, protože je jeho použití na disketách relativně bezproblémové.

Second extended – toto je standardní souborový systém pro Linux. Téměř vždy jej používáte a proto odpovězte kladně.

msdos – pokud budete chtít používat diskové oddíly či diskety vytvořené pod MS-DOS, odpovězte kladně.

K dispozici máte také několik dalších podporovaných souborových systémů.

/proc – Tento souborový systém (původem z Bell Labs) nevytváří klasický souborový systém na disku. Jedná se o rozhraní mezi jádrem a procesy. Mnoho programů (jako je například `ps`) tento systém používá. Zkuste někdy zadat příkaz `cat /prcoc/meminfo` či `cat /proc/devices`. Některé příkazové interprety (například `rc`) využívají pro vstupy a výstupy `/proc/self/fd` (na jiných systémech známé jako `/dev/fd`). Na tuto otázku byste měli téměř vždy odpovědět kladně, mnoho důležitých nástrojů je na tomto systému závislých.

NFS – pokud je váš počítač připojený k síti a chcete využívat souborové systémy umístěné na jiných počítačích, odpovězte na tuto otázku kladně.

ISO9660 – na tomto souborovém systému je založeno mnoho disků CD-ROM. Pokud chcete využívat pod Linuxem jednotku CD-ROM, odpovězte na tuto otázku kladně.

Co když ale nevíte, jaký souborový systém potřebujete? V takovém případě zadejte příkaz `mount`. Výstup by měl vypadat nějak takto:

```
/dev/hda1 on / type ext2 (defaults)
/dev/hda3 on /usr type ext2 (defaults)
none on /proc type proc (defaults)
/dev/fd0 on /mnt type msdos (defaults)
```

Podívejte se na každý jednotlivý řádek. Slovo následující za `type` určuje typ souborového systému. V tomto příkladě souborové systémy `/ a /usr jsou typu second extended, používá se /proc` a je připojena disketová mechanika pomocí souborového systému typu msdos.

Pokud máte povolenou podporu /proc, můžete zkusit zadat příkaz `cat /proc/filesystems` a získáte tak seznam aktuálně podporovaných souborových systémů.

Konfigurace využívající zřídkakdy používané souborové systémy mohou způsobit nárůst velikosti jádra. Podívejte se do kapitoly 10 věnované modulům, jak tomuto nepříjemnému důsledku zabránit a do kapitoly 8 věnované některým léčkám, proč takováto `jádra nejsou žádoucí`.

Znaková zařízení
{ Character devices}

Zde můžete povolit podporu zařízení pro tisk (paralelní tiskárny), myší (i typu PS/2 – mnoho notebooků využívá pro vestavěná polohovací zařízení právě protokol PS/2), některých páskových jednotek a dalších znakových zařízení. Pokud je používáte, odpovězte kladně.

> **POZNÁMKA:** Program `gpm` umožňuje používat myši i mimo X-window (například pro kopírování mezi virtuálními konzolami). Nenastávají žádné problémy, pokud máte myš pro sériové rozhraní, protože ty naprosto správně spolupracují s X-window. V opačném případě ale budete potřebovat některé speciální triky.

Zvuk
{ Sound}

Pokud máte neodolatelnou touhu pracovat se zvukem, odpovězte na tuto otázku kladně. Následně budete dotázani na doplňující informace o vaší zvukové kartě (na dotaz, zda instalovat plnou verzi ovladače, můžete odpovědět záporně – snížíte tak paměťové nároky pro jádro a budete využívat pouze vlastnosti, které opravdu potřebujete).

Pokud se o podporu zvukových karet vážně zajímáte, podívejte se jak na ovladače dostupné zdarma na adrese http://www.linux.org.uk/OSS/, tak na komerční ovladače Open Sound System (http://www.opensound.com) a také na projekt ALSA (http://www.alsa-project.org).

Další volby
{ Other configuration options}

Zde nejsou uvedeny všechny dostupné volby – jednak se jejich hodnoty nedají zobecnit, jednak je jejich hodnota snadno odvoditelná (například volba 3Com 3C509 zajistí podporu tohoto typu síťové karty). Úplný seznam všech voleb (spolu s popisem, jak je umístit do skriptu `Configure`) je spravován Axelem Boldtem (boltd@math.ucsb.edu). Je dostupný jako nápověda i jako jeden velký soubor `Documentation/Configure.help` ve zdrojovém kódu jádra.

Kernel hacking

Citace ze souboru `README` od Linuse Torvaldse:

Tyto volby obvykle vedou k nárůstu nebo zpomalení jádra (nebo obojímu) a mohou snížit stabilitu jádra. Je to způsobeno tím, že některé rutiny se pokouší přerušit špatný kód k nalezení problémů jádra (kmalloc()). Takže byste měli na tuto otázku pro produkční verzi jádra odpovědět záporně.

Co dál? (Makefile)

Poté, co dokončíte nastavení, budete vyzváni ke kontrole nadřazených souborů `Makefile` pro případné dodatečné nastavení. Podívejte se tedy na soubor `Makefile` – s největší pravděpodobností jej nebudete muset měnit, ale neuškodí si jej prohlédnout. Můžete také změnit volby příkazem `rdev` dříve, než umístíte nové jádro. Pokud se v souboru `Makefile` ztratíte, nic si z toho nedělejte.

Překlad jádra

Úklid a závislosti

Po ukončení konfiguračního skriptu jste také vyzváni k příkazu `make dep` a (možná) `make clean`. Zadejte tedy příkaz `make dep`, který zajistí, že všechny závislosti, jako jsou hlavičkové soubory, jsou správně nastaveny a vyžadované soubory jsou na svých místech. Tato kontrola netrvá nijak dlouho, pokud tedy váš počítač není příliš pomalý. Pro starší verze jádra byste měli po ukončení zadat příkaz `make clean`. Tento příkaz odstraní všechny soubory `*.o` a některé další věci, které v systému zůstaly po předchozích verzích. V žádném případě nezapomeňte provést tento krok před samotným překladem jádra.

Vlastní překlad

Po provedení kontroly závislostí a úklidu můžete zadat příkaz `make bzImage` či `make bzdisk` (tato část vyžaduje nejvíce času). První příkaz `make bzImage` přeloží jádro a umístí jej pod název `bzImage` do adresáře `arch/i386/boot` (včetně dalších věcí). Toto je nové komprimované jádro. Druhý příkaz `make bzdisk` vykonává totéž, ale současně umístí nové jádro `bzImage` na disketu, kterou jste zasunuli do jednotky A: Tuto disketu můžete použít pro testování nového jádra – pokud se toto nové jádro nechová správně či se nespustí vůbec, můžete jednoduše vyndat disketu z mechaniky a nastartovat počítač s původním jádrem. Tuto disketu můžete také použít v případě, kdy jste omylem z pevného disku toto nové jádro odstranili (či provedli něco podobně chybného). Můžete ji také použít pro instalaci nového systému ve chvíli, kdy přenášíte obsah jednoho disku na druhý.

Takřka všechna dnešní jádra jsou komprimována, odtud umístění písmenek bz před vlastní název. Komprimované jádro je automaticky dekomprimováno před vlastním spuštěním.

Ve starších jádrech nemáte možnost volby pro vytvoření komprimovaného jádra. Tato volba je dnes dostupná i z toho důvodu, že velikost nových verzí jádra se neustále zvyšuje a starší metody neumožňují obsluhovat tato velká jádra.

Další možnosti příkazu make

Příkaz `make mrproper` je rozsáhlejší obdobou příkazu `make clean`. Někdy je provedení tohoto příkazu potřebné, můžete jej chtít provést například při každé nové záplatě. Tento příkaz také vymaže váš konfigurační soubor s nastavením jádra. Proto si soubor `.config` můžete chtít zálohovat, pokud tedy vlastní obsah uznáte za důležitý.

Příkaz `make oldconfig` se pokusí nastavit nové jádro podle původního konfiguračního souboru. Tento příkaz je spouštěný v rámci příkazu `make config`. Pokud nemáte žádné dříve překládané jádro či žádný starší konfigurační soubor, tak byste neměli tento příkaz spouštět.

Podrobnosti o příkazu make modulesInstalace jádra

Poté, co máte nové jádro odpovídající vašim požadavkům, nastává vhodný okamžik pro jeho instalaci. Mnoho uživatelů pro tuto činnost preferuje LILO (Linux Loader). Příkaz `make bzlilo` nainstaluje jádro, spustí LILO a připraví vše pro spuštění systému. To vše ale pouze za předpokladu, že LILO je na vašem systému nastaveno takto: jádro je `/vmlinuz`, program `lilo` je umístěn v adresáři `/sbin` a konfigurační soubor pro LILO je `/etc/lilo.conf`.

V opačném případě musíte použít LILO přímo – je jednoduché jej nainstalovat a používat, ale vykazuje občasné tendence zmást uživatele právě svým konfiguračním souborem. Pro starší verze je konfigurační soubor `/etc/lilo/config` a pro nové verze `/etc/lilo.conf`. Podívejte se na jeho obsah platný pro současné nastavení, který bude vypadat přibližně takto:

```
image = /vmlinuz
label = Linux
root = /dev/hda1
...
```

`Parametr image =` je nastaven na aktuálně nainstalované jádro. Mnoho uživatelů používá právě `/vmlinuz. Parametr label =` je používán pro určení, jaké jádro či operační systém bude spuštěn a parametr `root =` je kořenový adresář konkrétního operačního systému. Udělejte si záložní kopii původního souboru s jádrem a nahraďte jej novým jádrem `bzImage (pokud používáte /vmlinuz`, tak pak například pomocí příkazu cp bzImage /vmlinuz). Následně spusťte program `lilo (na` novějších systémech stačí napsat příkaz `lilo`, na starších možná budete muset zadat příkaz `/etc/lilo/install` či `/etc/lilo/lilo -C /etc/lilo/config)`.

Pokud se chcete o nastavení pro LILO dozvědět více či LILO vůbec nemáte nainstalováno, stáhněte si nejnovější verzi z vašeho oblíbeného serveru ftp a řiďte se instalačními pokyny.

Pro spuštění některé ze starších verzí jádra (další z možností, jak se můžete ochránit před důsledky plynoucími z problémů s novým jádrem), nakopírujte řádky následující za `image = xxx` (pochopitelně včetně tohoto řádku) na konec konfiguračního souboru a změňte parametr `image = xxx na image = yyy`, kde yyy je název původního jádra. Tento název je včetně cesty a tuto zálohu máte vytvořenou z předchozího kroku. Poté změňte parametr `label = zzz na label = linux-backup` a spusťte `lilo`. Můžete také chtít vložit do konfiguračního souboru řádek s parametrem `delay = x`, kde x je počet desetin sekundy, během kterých LILO čeká před automatic-

kým spuštěním aktuálního jádra. Tento proces lze před uplynutím definované doby zastavit stiskem klávesy (například Shift) a například v případě problémů s novým jádrem zadat název původního jádra (určený hodnotou parametru `label`).

Opravy jádra

Uplatnění záplat

Přírůstkové inovace jádra jsou distribuovány jako záplaty. Například, pokud máte verzi 1.1.45 a máte k dispozici záplatu `patch46.gz`, znamená to, že můžete provést upgrade na verzi 1.1.46 pouhou aplikací této záplaty. Můžete také nejprve chtít vytvořit si záložní kopii zdrojových kódů jádra verze 1.1.45 (nejprve zadejte příkaz `make clean` a pak příkazy `cd /usr/src` a `tar zcvf old-tree.tar.gz linux`, čímž získáte komprimovanou zálohu zdrojového kódu).

Pokračujme ve výše uvedeném příkladě – budeme předpokládat, že máte soubor `patch46.gz` nakopírován do adresáře `/usr/src`. Přejděte do tohoto adresáře a zadejte příkaz `zcat patch46.gz | patch-p0` (případně `patch -p0 < patch46`, není li soubor se záplatou komprimován).Výstup vás bude informovat o pokusech aplikovat změny a zda tyto pokusy dopadly úspěšně či nikoli. Ve většině případů tato akce proběhne příliš rychle na to, abyste si mohli vše pečlivě pročíst, proto můžete pro program patch použít volbu `-s`. Ta určuje, že ve výstupu budou obsažena pouze chybová hlášení (neuvidíte tedy hlášení typu „Ahoj, můj počítač je připraven na aplikování záplaty“, ale proti gustu ...). Pro seznámení se s částmi, které neproběhly hladce, přejděte do adresáře `/usr/src/linux` a najděte soubory s příponou `.rej`. Některé verze záplat (starší verze, které mohou být přeloženy na horším souborovém systému) ponechávají tyto soubory s příponou #. Pro nalezení těchto souborů můžete použít příkaz `find`:

```
find . -name '*.rej' -print
```

Tento příkaz vypíše na standardní výstup všechny soubory v aktuálním adresáři a všech podadresářích, které mají příponu `.rej`.

Pokud vše proběhlo v pořádku, zadejte příkaz `make clean`, `make config a make dep`, jak bylo popsáno v kapitolách 3 a 4.

Příkaz `patch` má také několik voleb. Jak bylo zmíněno výše, příkaz `patch -s` potlačí zobrazení všech hlášení kromě chybových. Pokud jste ponechali zdrojový kód vašeho jádra v původním adresáři (`/usr/src/linux`), příkaz `patch -p1` (v tomto adresáři) provede opravu jádra korektně. Další volby příkazu `patch` jsou popsány v nápovědě.

Pokud něco nefunguje správně?

POZNÁMKA: Tato kapitola se většinou odkazuje na starší verze jádra.

Častým problémem je, že při aplikování záplaty dochází ke změně souboru `config.in` a ty se neprovedou vždy správně, protože jste změnili volby na vašem počítači. Toto je problém, se kterým se můžete setkat zejména ve starších verzích. Pro nápravu se podívejte na obsah souboru `config.in.rej` a zjistíte, co zůstalo z původní záplaty. Změny bývají nejčastěji označeny + a – na začátku každého řádku. Podívejte se také na okolní řádky a zapamatujte si, zda jsou volby nastaveny na y či n. Nyní změňte v souboru `config.in` odpovídající parametry na stejné hodnoty. Pak zadejte příkaz:

```
patch -p0 < config.in.rej
```

a pokud se nezobrazí žádná chyba, můžete pokračovat s dalším nastavováním a překladem. Soubor `config.in.rej` sice zůstane, ale můžete jej smazat.

Pokud se setkáte s dalšími problémy, mohli jste instalovat záplaty ve špatném pořadí. Pokud se zobrazí chybové hlášení „`previously applied patch detected: Assume -R?`", možná se pokoušíte aplikovat záplatu, která je nižní verze, než verze aktuálního jádra. Pokud odpovíte kladně, dojde k pokusu o „snížení" zdrojového kódu jádra, ale takřka vždy dojde k chybě. Následně budete muset získat kompletní zdrojový kód jádra (což by nebyl špatný nápad hned jako první krok).

`Pro zrušení záplaty` můžete použít příkaz `patch -R`.

Pokud se objeví nějaké problémy se záplatami, je nejlepší začít vše od začátku nad nově rozbaleným zdrojovým kódem jádra.

Odstranění souborů s příponou .orig

Po aplikování několika záplat se na vašem disku začnou hromadit soubory s příponou `.orig`, po třech záplatách (u nás například u verze 1.1.51 z 1.1.48) zabíraly tyto soubory okolo půl megabajtu.

Následující příkaz vás zbaví starostí s těmito soubory:

```
find . -name '*.orig' -exec rm -f { } ';'
```

Verze programu patch, které používají pro odmítnuté změny značku #, používají místo přípony .orig znak tilda (~).

V závislosti na použití utility *xargs* můžete tyto soubory odstranit také takto:

```
find . -name '*.orig' | xargs rm
```

Další z možností je:

```
find . -name '*.orig' -print0 | xargs --null rm -
```

Další záplaty

Mimo standardních originálních záplat existují také další nestandardní záplaty. Pokud tyto záplaty použijete, nebudete možná schopni originální záplaty správně aplikovat a budete muset instalovat celý původní zdrojový kód a aplikovat jednotlivé originální záplaty. To může být velmi vyčerpávající, zejména tehdy, pokud nechcete měnit zdrojové kódy (s pravděpodobně špatným výsledkem). Po instalaci původního zdrojového kódu jádra máte na výběr spokojit se s tímto starším jádrem, pokusit se přinutit ke spolupráci originální a nestandardní záplaty či počkat na novou verzi záplaty.

Jaké jsou známé nestandardní záplaty? Pravděpodobně o nich uslyšíte. My používáme „neblikající" záplatu pro virtuální konzoly, protože nemáme rádi blikající kurzory (tato záplata je, či alespoň dříve byla, pravidelně obnovovaná v souvislosti s novými verzemi jádra). S příchodem ovladačů zařízení ve tvaru modulů dochází k poklesu významu těchto nestandardních ovladačů.

Další balíčky

Linuxové jádro nabízí mnoho vlastností, které nejsou obsaženy přímo ve zdrojovém kódu jádra. Tyto vlastnosti jsou nejčastěji zajištěny pomocí externích balíčků. V této kapitole jsou uvedeny některé z těch nejznámějších.

Balíček kbd

Linuxová konzola nabízí pravděpodobně více vlastností, než si zaslouží. Mezi tyto vlastnosti patří přepínání písem, definice rozložení kláves, přepínání grafických režimů (v novějších verzích jádra) atd. Balíček kbd obsahuje programy, které umožňují uživatelům využívat všechny tyto vlastnosti a navíc nabízí mnoho dalších písem, přednastavené definice mnoha typů klávesnic a je dostupný na stejných místech jako vlastní zdrojové kódy jádra.

Balíček util-linux

Tento balíček, původně od Ricka Faitha (faith@cs.unc.edu), obsahuje velkou kolekci utilit. Nyní jej spravuje Andries Brouwer (util-linux@math.uio.no) a je dostupný v rámci anonymního serveru ftp www.ibiblio.org v adresáři `/pub/Linux/system/misc`. Součástí balíčku jsou například programy `setterm`, `rdev` či `ctrlaltdel`. Podle slov autora byste ale tento balíček neměli instalovat bezmyšlenkovitě – nebudete potřebovat vše, co je v balíčku obsaženo. V opačném případě se můžete setkat s vážnými problémy.

Balíček hdparm

Stejně jako mnoho balíčků, byl i tento záplatou jádra a podpůrnými programy. Záplaty se staly součástí oficiálního jádra a programy vylepšující a rozšiřující možnosti využití pevných disků jsou distribuovány odděleně.

Balíček gpm

Tento balíček je určen pro podporu práce s myší a umožňuje používat kopírovací funkce mezi dvěma konzolami a některé další vlastnosti s mnoha typy myší.

Některé léčky

make clean

Pokud se vaše nové jádro chová opravdu podivně, je možné, že jste zapomněli před překladem nového jádra provést příkaz `make clean`. Příznaky mohou být libovolné – od úplného pádu operačního systému přes vážné V/V problémy až po snížení výkonu. Ujistěte se také, že nezapomenete na příkaz `make dep`.

Velké nebo pomalé jádro

Pokud nové jádro potřebuje mnoho operační paměti, je příliš velké a/nebo jeho překlad trvá neúměrně dlouho, pravděpodobně jste nakonfigurovali příliš mnoho vlastností jádra (ovladačů zařízení, souborových systémů apod.). Pokud je nepoužíváte, nenastavujte je, protože zabírají příliš paměti. Nejčastějším příznakem je extrémní práce se souborem virtuální paměti – pokud tento případ nastane, podívejte se ještě jednou na nastavení jádra.

Celkovou velikost paměti obsazenou jádrem můžete zjistit odečtem dostupné paměti (zjistíte ji pomocí příkazu `free` či informací v `/proc/meminfo`) od velikosti operační paměti daného počítače.

Paralelní port nepracuje, tiskárna netiskne

V kategorii General setup zkontrolujte nastavení voleb Parallel port support a PC-style hardware. Poté v kategorii Character devices zkontrolujte nastavení volby Parallel printer support.

Dalším možným problémem může být změna v názvosloví pro paralelní zařízení, podle kterého je od verze jádra 2.2 první paralelní port označován jako `lp0` a ne `lp1`, jak tomu bylo v dřívějších verzích jádra. V odhalení tohoto problému vám může pomoci program `dmesg` či protokolové soubory v adresáři `/var/log`.

Jádro nelze přeložit

Pokud nelze jádro přeložit, mohou být špatně nastavené cesty či zdrojové kódy mohou být poškozeny. Také nemusíte mít správnou verzi překladače `gcc` nebo může být poškozený také (chyba může být například v hlavičkových souborech). Ujistěte se, že symbolické odkazy popsané v souboru `README` jsou správně nastavené. Obecně platí, že pokud nelze standardní jádro přeložit, jsou nějaké problémy v celém systému – dost možná bude nutné přeinstalovat některé nástroje.

V některých případech může být pád překladače `gcc` způsoben hardwarovými problémy. Chybové hlášení v takových případech vypadá přibližně takto: `!xxx exited with signal 15!` a vlastní překlad se chová velmi podivně. My jsme se s uvedeným problémem setkali pouze jednou, když jsme měli problémy s vyrovnávací pamětí a překladač padal zcela náhodně. Nejprve zkuste reinstalaci překladače `gcc`. Podezření mějte pouze v případě, kdy překlad jádra proběhne správně při vypnuté externí vyrovnávací paměti či snížení velikosti operační paměti apod.

Většinu uživatelů vyděsí, pokud zjistí, že mohou mít problémy s hardwarem – pomoc hledejte v odpovídajícím souboru často kladených dotazů a odpovědí, který najdete na adrese http://www.bitwizard.nl/sig11/.

Nelze spustit novou verzi jádra

Pokud se zdá, že nemůžete spustit novou verzi jádra, je možné, že jste zapomněli spustit LILO nebo nebylo správně nastaveno. Problém může být například v konfiguračním souboru, kdy parametr `boot` máte nastavený na hodnotu `/dev/hda` místo `/dev/hda1`.

Zapomněli jste spustit LILO nebo systém není možné spustit

Tak to je opravdu problém. Nejlepší řešení tohoto problému je spuštění systému z diskety nebo CD-ROM a připravení jiné startovací diskety (například pomocí příkazu `make zdisk`). Musíte vědět, kde je umístěn hlavní souborový systém (kořenový, `/`) a jakého typu tento souborový systém je (například second extended, minix). V následujícím příkladě musíte také znát, na jakém souborovém systému máte uložený zdrojový kód jádra a jak je tento souborový systém za normálních okolností připojen.

V následujícím příkladě je kořenovým adresářem zařízení `/dev/hda1` a souborový systém se zdrojovými kódy jádra je na zařízení `/dev/hda3`, které bývá za normálních okolností připojeno jako `/usr`. Oba souborové systémy jsou typu second extended. Funkční jádro je uloženo v adresáři `/usr/src/linux/arch/i386/boot` a soubor se jmenuje `bzImage`.

Základní myšlenka záchrany je, že pokud máte k dispozici funkční jádro, bude možné toto jádro použít pro novou startovací disketu. Další možnost, která se ovšem nemusí vždy podařit (úspěšnost závisí na konkrétním způsobu poškození systému), je popsána za tímto příkladem.

Nejprve nastartujte systém ze záchranné diskety či startovacích disket pro instalaci a připojte souborový systém s funkčním jádrem:

```
mkdir /mnt
mount -t ext2 /dev/hda3 /mnt
```

Pokud se vytvoření adresáře `/mnt` nepovede z toho důvodu, že již existuje, netrapte se tím a hlášení o chybě ignorujte. Nyní přejděte do adresáře s funkčním jádrem. Uvědomte si, že platí:

```
/mnt + /usr/src/linux/arch/i386/boot - /usr = /mnt/src/linux/arch/i386/boot
```

Vložte do disketové mechaniky (A:) prázdnou naformátovanou disketu, přeneste na ni obraz jádra a nastavte ji pro kořenový souborový systém.

```
cd /mnt/src/linux/arch/i386/boot
dd if=bzImage of=/dev/fd0
rdev /dev/fd0 /dev/hda1
```

Přejděte do kořenového adresáře a odpojte souborový systém `/usr`.

```
cd /
umount /mnt
```

Nyní byste měli být bez problémů schopni nastartovat systém přímo z výše vytvořené diskety. Nezapomeňte po startu systému spustit LILO (či cokoli jiného, co jste předtím udělali chybně).

Jak již bylo uvedeno výše, máte k dispozici ještě jednu možnost. Máte-li k dispozici funkční jádro v kořenovém adresáři (například `/vmlinuz`), můžete je použít pro startovací disk. Předpokládáme-li všechny dříve vyřčené požadavky, můžete tuto možnost realizovat například takto – změňte `/dev/hda3` na `/dev/hda1` (kořenový adresář), `/mnt/src/linux` na `/mnt a if=bzImage` na `if=vmlinuz`.

Problémy také může způsobit použití LILO s velkými disky (více než 1 024 válců), více informací najdete v nápovědě a dokumentaci.

Jednotka IDE/ATAPI CD-ROM nepracuje

Mnoha uživatelům se nedaří zprovoznit jejich mechaniky CD-ROM, pravděpodobně proto, že problém může být způsoben mnoha důvody.

Pokud je jednotka CD-ROM jediným zařízením na konkrétním rozhraní IDE, musí být nastavena jako master či single. Toto je jedna z nejčastějších příčin.

Někteří výrobci, například Creative Labs, nyní umisťují rozhraní IDE na zvukové karty. Tato skutečnost může způsobit problémy – dříve mnoho uživatelů mělo pouze jeden kanál IDE, poté dva integrované na základní desce (nejčastěji přerušení IRQ 15) a nyní třetí kanál IDE (pravděpodobně přerušení IRQ 11). Třetí kanál IDE způsobuje problémy s jádrem 1.2.x (podpora třetího kanálu IDE začíná až u verze 1.3.x a nepodporuje autodetekci). Pokud se setkáte s tímto problémem, máte na výběr z několika možností.

Máte-li druhý kanál IDE, je velmi pravděpodobné, že jej nepoužíváte či jej nepoužíváte pro obě možná zařízení. Vypněte tedy zařízení na zvukové kartě a přesměrujte je na druhý kanál IDE. Následně můžete zakázat rozhraní zvukové karty, čímž ušetříte jedno přerušení.

Pokud druhé rozhraní IDE nemáte, nastavte kanál IDE na zvukové kartě tak, aby využíval přerušení IRQ 15. Vše by poté mělo fungovat korektně.

Objevují se podivné problémy při požadavcích na směrování

Stáhněte si novou verzi programu `route` a dalších programů, které se při směrování využívají, protože soubor `/usr/include/linux/route.h` (fyzicky je soubor uložen v adresáři `/usr/src/linux`) byl změněn.

Objeví se hlášení „Not a compressed kernel Image file“

Nepoužívejte jako startovací obraz jádra soubor `vmlinuz` vytvořený v adresáři `/usr/src/linux`, správný soubor je `arch/i386/boot/bzImage`.

Po inovaci jádra není možné překládat další programy

Součástí zdrojového kódu jádra je mnoho hlavičkových souborů (těch, které končí příponou .h), jež jsou standardně odkazovány z adresáře `/usr/include`. Nejčastěji se na tyto hlavičkové soubory odkazují programy takto (`xyzzy.h` je libovolný hlavičkový soubor v adresáři `/usr/include/linux`):

```
#include <linux/xyzzy.h>
```

Za normálních okolností existuje odkaz `/usr/include/linux` zastupující adresář `include/linux` ve zdrojovém kódu vašeho jádra (tedy na typickém systému adresář `/usr/src/linux/include/linux`). Pokud tento odkaz neexistuje či odkazuje na špatný adresář, mnoho dalších programů nepůjde přeložit. Velmi často tento problém vzniká tak, že se rozhodnete pročistit disk a vymažete zdrojové kódy jádra. Dalším častým problémem bývají špatně nastavená práva – pokud jste přihlášen jako uživatel `root`, máte předdefinované takové masky pro nově vytvářené soubory, které neumožní ostatním uživatelům k těmto souborům přistupovat (pokud jste rozbalili zdrojový kód jádra bez volby `-p (preserve modification)) a ostatní uživatelé` tak nebudou moci používat překladač jazyka C. Přestože lze pro vyřešení tohoto problému použít příkaz `chmod`, jednodušší cesta je rozbalit zdrojový kód jádra znovu. Toto rozbalení proběhne stejně jako předtím, ovšem s přidáním dalšího parametru:

```
blah# tar zxvpf linux.x.y.z.tar.gz linux/include
```

POZNÁMKA: Pokud odkaz `/usr/src/linux` neexistuje, příkaz `make config` jej vytvoří.

Zvýšení limitů

Následující ukázkové příkazy mohou být užitečné pro zvýšení limitů určovaných jádrem:

```
echo 4096 > /proc/sys/kernel/file-max
echo 12288 > /proc/sys/kernel/inode-max
echo 300 400 500 > /proc/sys/vm/freepages
```

Poznámky k inovaci jádra na verze 2.0.x a 2.2.x

Verze jádra 2.0.x a 2.2.x zavedly některé změny v instalaci jádra – podrobnosti o těchto změnách a nové postupy jsou uvedeny v souboru `Documentation/Changes`. Budete muset inovovat několik klíčových balíčků, jako je `gcc`, `libc` či `SysVInit` a možná také několik systémových souborů, takže s tím počítejte a nepanikařte.

Moduly

Moduly především šetří paměť a usnadňují nastavení jádra a používají se zejména v oblasti souborových systémů, ovladačů síťových karet a páskových jednotek, tiskáren apod.

Instalace utilit pro podporu modulů

Utility pro podporu práce s moduly jsou pod názvem `modutils-x.y.z.tar.gz` dostupné na stejných zdrojích, kde jste získali zdrojový kód jádra. Zvolte nejvyšší možnou verzi, která je shodná či nižší s verzí vámi používaného jádra. Rozbalte tento soubor pomocí příkazu `tar zxvf modutils-x.y.z.tar.gz`, přejděte do vytvořeného adresáře (`modutils-x.y.z`), přečtěte si soubor `README` a dodržte instalační instrukce (ty bývají velmi snadné, zpravidla stačí zadat příkaz `make install`). Nyní byste měli mít v adresáři `/sbin` k dispozici programy `insmod`, `rmmod`, `ksyms`, `lsmod`, `genksyms`, `modprobe` a `depmod`. Správnost instalace si můžete ověřit pomocí ukázkového příkladu hw – podrobnosti najdete v souboru `INSTALL` v příslušném podadresáři.

Program `insmod` jednotlivé moduly vkládá do běžícího jádra. Moduly mají nejčastěji příponu `.o`, ukázkový ovladač zmíněný výše má název `drv_hello.o`, do běžícího jádra jej tedy můžete vložit pomocí příkazu `insmod drv_hello.o`. Přehled modulů, které jsou aktuálně aktivní, získáte pomocí příkazu `lsmod`. Výstup bude vypadat například takto:

```
blah# lsmod
Module:       #pages:  Used by:
drv_hello          1
```

Údaj `drv_hello` je název modulu, který používá jednu stránku paměti (4 KB) a na tomto modulu v tomto okamžiku žádné další moduly nezávisí. Pro odstranění tohoto modulu z jádra stačí zadat příkaz `rmmod drv_hello`. Uvědomte si, že příkaz `rmmod` vyžaduje jako parametr název modulu (tak jak jej vidíte ve výstupu příkazu `lsmod`), nikoli název souboru s modulem. Ostatní utility pro práci s moduly jsou popsány v příslušných manuálových stránkách.

Moduly distribuované spolu s jádrem

Od verze jádra 2.0.30 jsou mnohé vlastnosti dostupné prostřednictvím modulů. Předtím, než začnete moduly používat, si rozmyslete, zda pro vás nebude lepší použít klasické vlastnosti jádra. Pokud ne, neodpovídejte kladně během průběhu příkazu `make config`. Přeložte nové jádro a nastartujte systém s tímto jádrem. Poté přejděte do adresáře `/usr/src/linux` a zadejte příkaz `make modules`, který přeloží všechny požadované moduly a vytvoří symbolické odkazy do adresáře `/usr/src/linux/modules`. Moduly pak můžete používat přímo z tohoto adresáře nebo zadejte příkaz `make modules_install`, který nainstaluje jednotlivé moduly do adresáře `/lib/modules/x.y.z`, kde x.y.z je číslo verze jádra.

Tento přístup může být výhodný pro práci se souborovými systémy. Nebudete často používat souborové systémy typu minix a msdos? Setkáte se s disketou ve formátu msdos? Zadáte příkaz `insmod /usr/src/linux/modules/msdos.o` a po ukončení práce s disketou modul odstraníte pomocí příkazu `rmmod msdos`. Tento přístup vám během normální práce ušetří okolo 50 KB operační paměti. Drobná poznámka na závěr – souborový systém minix byste z důvodu případné obnovy systému měli přeložit přímo do vlastního jádra.

Tipy a triky

Přesměrování výstupu příkazů make a patch

Pokud chcete získat výstupy programů `make` a `patch`, můžete úspěšně přesměrovat jejich výstupy do souboru. Nejprve zjistěte, jaký příkazový interpret používáte (například pomocí příkazu `grep root /etc/passwd`).

Pro příkazové interprety `bash` a `sh` provedete přesměrování výstupu takto:

```
(command) 2>&1 | tee (výstupní soubor)
```

Pro příkazové interprety `csh` a `tcsh` provedete přesměrování výstupu takto:

```
(command) |& tee (výstupní soubor)
```

Pro příkazový interpret `rc` (pravděpodobně jej nepoužíváte) provedete přesměrování výstupu takto:

```
(command) >[2=1] | tee (výstupní soubor)
```

Instalace více jader

Pro testování nového jádra existují také další metody, nejen instalace jádra na disketu. Na rozdíl od mnoha dalších unixů umožňuje LILO spustit jádro umístěné kdekoli na pevném disku (pokud máte velký disk – 500 MB a více – podívejte se do dokumentace k LILO a seznamte se s možnými problémy). Pokud přidáte na konec konfiguračního souboru pro LILO kód:

```
image = /usr/src/linux/arch/i386/boot/bzImage
label = new_kernel
```

budete moci spouštět nově přeložená jádra bez nutnosti přepisu původního jádra `/vmlinuz` (pochopitelně musíte spustit `lilo`). Nejjednodušší způsob, jak vybrat jádro, které se má použít, je stisknout klávesu Shift v okamžiku startu systému (při zobrazení hlášky `LILO`) a následně zadat `new_kernel`.

Pokud chcete na svém disku uchovávat zdrojové kódy pro více verzí jádra (ale pozor na místo na disku), můžete používat oblíbený způsob pojmenování adresářů s jednotlivými verzemi jádra podle vzoru `/usr/src/linux-x.y.z`, kde x.y.z je číslo verze jádra. Poté můžete nastavit platný zdrojový kód pomocí symbolického odkazu (například příkazem `ln -sf linux-1.2.2 /usr/src/linux` nastavíte jako aktuální zdrojový kód jádra verze 1.2.2). Předtím, než vytvoříte tento symbolický odkaz, si ověřte, že poslední argument neodpovídá skutečnému adresáři – v opačném případě nebude výsledek takový, jaký předpokládáte.

Aktualizace jádra

Změny mezi jednotlivými verzemi jádra sleduje a zaznamenává Russell Nelson (nelson@crynwr.com). Pokud se chcete na tyto změny před inovací jádra podívat, je tento dokument k dispozici na adrese http://www.crynwr.com/kchanges nebo na anonymním serveru ftp ftp.emlist.com v adresáři `pub/kchanges`.

Další užitečné dokumenty

- Sound-HOWTO: zvukové karty a utility (Praktické návody, kapitola 18)
- SCSI-HOWTO: vše o SCSI řadičích a zařízeních
- NET-2-HOWTO: sítě
- PPP-HOWTO: sítě pomocí protokolu PPP
- PCMCIA-HOWTO: o ovladačích pro notebook
- Hardware-HOWTO: přehled podporovaného hardwaru
- Module mini-HOWTO: podrobnosti o modulech
- Kerneld mini-HOWTO: informace o kerneld
- BogoMips mini-HOWTO: pro případ, že budete udiveni

Apache

Originál: http://tldp.org/HOWTO/Apache-Overview-HOWTO.html

Tento dokument obsahuje informace o webovém serveru Apache a souvisejících projektech. Obsahuje odkazy na další zdroje informací a detaily o implementaci.

Úvod

Tento dokument obsahuje přehled webového serveru Apache a souvisejících projektů. **Apache je nejpopulárnějším webovým serverem na Internetu**. Noví uživatelé Apache, zejména ti, kteří přecházejí z platformy Windows, často neznají možnosti serveru Apache, užitečné přídavné moduly a, obecněji, jak to všechno spolu funguje. Tento dokument se zaměřuje na obecný obrázek takových možností společně s jejich krátkým popisem a odkazy na další zdroje informací. Informace byly získány z mnoha zdrojů, včetně webových stránek projektů, konferencí, poštovních konferencí, webových sídel Apache a mých vlastních zkušeností. Hlavní zásluhy patří autorům. Bez nich a jejich práce by tento dokument nebyl možný nebo by nebyl zapotřebí.

Apache Software Foundation

Apache Software Foundation poskytuje podporu komunitě open-source projektů kolem Apache. Tyto projekty jsou charakteristické kooperativním a konsensuálním vývojovým procesem, otevřenou a pragmatickou licencí a snahou vytvořit vysoce kvalitní software, který bude představovat špičku ve svém oboru. Nepovažujeme se pouze za skupinu projektů sdílející společný server, ale spíše za komunitu vývojářů a uživatelů.

ASF hostí spoustu úspěšných open-source projektů, například Servlet/JSP engine Tomcat, nebo vývojářský nástroj ANT.

Více informací naleznete na adrese *http://www.apache.org/foundation/*.

Struktura tohoto dokumentu

V první části dokumentu hovoříme o webovém serveru Apache a o souvisejících modulech. Zabýváme se historií, architekturou a možnostmi serveru a popisujeme způsoby, jak je můžete rozšířit a upravit.

Ve druhé části hovoříme o dalších projektech ASF, například o projektech Jakarta a Java XML. Projekty nečleníme podle programovacího jazyka nebo technologie, ale podle nabízených funkcí.

Apache

Apache je přední webový server, s více než 60 % podílu na trhu podle přehledů společnosti Netcraft. Některé klíčové faktory, které se podílely na úspěchu Apache:

- Licence Apache. Je to open – source projekt, s licencí podobnou BSD, která umožňuje komerční i nekomerční využití Apache.
- Talentovaná komunita vývojářů s různým zázemím a otevřený proces vývoje.
- Modulární architektura. Uživatelé Apache mohou jednoduše přidávat funkce nebo rozvíjet Apache pro své specifické prostředí.
- Přenositelnost: Apache běží na skoro všech unixových (a linuxových) systémech, Windows, BeOS, sálových počítačích...
- Robustnost a bezpečnost.

Mnozí komerční dodavatelé vytvořili řešení založená na Apache, včetně Oracle, Red Hat a IBM. Kromě toho Covalent poskytuje přídavné moduly a podporu 24x7 pro Apache.

Následující webové servery používají server Apache nebo servery z něj odvozené. Když je Apache dost dobrý pro ně, pak je také dost dobrý pro vás :)

- Amazon.com
- Yahoo!
- W3 Consortium
- Financial Times
- Network solutions
- MP3.com
- Stanford

Z domovské stránky webového serveru Apache:

Projekt HTTP serveru Apache usiluje o vývoj a údržbu open-source HTTP serveru pro moderní operační systémy včetně Unixu a Windows NT. Cílem projektu je nabídnout bezpečný, výkonný a rozšiřitelný server poskytující HTTP služby ve shodě s platnými standardy.

Apache vznikl jako modifikace webového serveru NCSA, jednoho z vůbec prvních webových serverů. Více o historii serveru Apache se můžete dočíst na adrese *http://httpd.apache.org/ABOUT_APACHE.html*.

Projekt Apache se rozšířil z vytváření pouze webového serveru na další důležité technologie na straně serveru, jakými jsou např. Java nebo XML. Již zmíněná Apache Software Foundation tyto projekty zastřešuje.

Architektura

Existují dvě hlavní verze serveru Apache, řada 1.3 a řada 2.0. Obě řady jsou vhodné pro produkční nasazení, liší se však v architektuře a v možnostech.

Apache 1.3

Apache 1.3 byl přenesen na řadu různých unixových platforem a jde o nejrozšířenější webový server na Internetu.

Procesově orientovaný server

Apache 1.3 na Unixu je procesově orientovaný server. Při startu spustí několik potomků – rodičovský proces vytvoří několik svých identických kopií. Každý z potomků obsluhuje požadavky nezávisle na ostatních. Toto řešení je výhodné kvůli zvýšené stabilitě. Pokud některý z procesů nefunguje dobře (nelze jej ovládat nebo spotřebovává mnoho paměti), lze jej ukončit bez vlivu na

další procesy. Zvýšení stability s sebou přináší snížení výkonu. Na většině unixových systémů je vytvoření procesu a přepnutí kontextu náročná operace. Protože procesy jsou vzájemně nezávislé, nemohou jednoduše sdílet kód a data, což vede ke zvýšeným nárokům na systémové prostředky.

Podpora Windows

Apache 1.3 je první verze, která pracuje i ve Windows, i když na této platformě není považován za tak stabilní jako na unixových platformách. Je to dáno tím, že server byl navrhován pro unixové platformy a přenositelnost na Windows byla doplněna později a není dokonale integrována.

Modularita

Apache 1.3 pracuje s modulární architekturou. Zapínáním a vypínáním modulů můžete do serveru přidávat a odebírat funkce. Můžete jej upravit tak, aby byla zvýšena jeho bezpečnost a výkon. Kromě modulů dodávaných přímo se serverem je k dispozici celá řada modulů třetích stran, které doplňují spoustu různých funkcí.

Apache 2.0

Apache 2.0 je poslední a nejlepší verzí serveru. Jeho architektura byla oproti řadě 1.3 výrazně vylepšena. Některá vylepšení popisujeme dále.

Multi Processing Moduly

V řadě 2.0 je architektura pro zpracování požadavků oddělena do samostatných modulů, tzv. Multi Processing modulů (MPM). Apache tak lze nastavit jako čistě procesově orientovaný server, jako čistě vláknový server, nebo jako kombinaci obojího. Vlákna jsou součástí jednoho procesu a běží paralelně. Na rozdíl od procesů mohou vlákna sdílet kód a data. Vlákna jsou tak „odlehčenější“ než procesy a v případě extrémních požadavků na výkon tak vláknový server obvykle funguje lépe než čistě procesový server. Nevýhodou je menší spolehlivost serveru, protože v případě chyby ve vlákně může dojít k ovlivnění kódu a dat jiných vláken.

Protokolové moduly

Obsluha protokolů byla v Apache 2.0 rovněž oddělena do vlastní vrstvy. Znamená to, že je možné napsat moduly pro obsluhu jiných protokolů, než HTTP, například POP3 pro výběr pošty nebo FTP pro přenos souborů. Tyto moduly mohou využívat výhod základních funkcí serveru a dalších modulů, například autentizace a dynamického generování obsahu. Znamená to například, že můžete provádět autentizaci uživatelů POP3 proti stejné databázi, jakou Apache používá pro ověření přístupu k webovým stránkám, nebo že obsah FTP adresářů můžete dynamicky generovat pomocí PHP, CGI nebo jiných dále popsaných technologií.

Architektura modulů a filtrů

Apache 2.0 používá stejně jako 1.3 modulární architekturu, kterou navíc rozšiřuje o další mechanismus – o filtry. Filtry modulům umožňují modifikovat data generovaná jinými moduly. Mohou tak šifrovat, provádět virovou kontrolu nebo komprimovat nejen statické soubory, ale i dynamicky generovaný obsah.

Problémy s kompatibilitou

Bohužel, i když je modulární API obou verzí podobné, moduly pro verzi 1.3 je nutné pro použití na nové architektuře upravit. Řada hlavních modulů, například PHP nebo mod_perl, existují i pro verzi 2.0 a některé jiné, například mod_dav a mod_ssl, jsou už přímo součástí distribuce serveru. Spouštění modulů v architektuře založené na vláknech si vyžaduje jejich specifickou úpra-

vu. Moduly dodávané se serverem Apache byly takto upraveny a jejich provoz na vláknové architektuře je považován za bezpečný. U modulů a knihoven třetích stran to nemusí platit. Pokud byste potřebovali nějaké takové použít, budete moci Apache provozovat výhradně v režimu procesově orientovaného serveru.

Přenositelnost

Díky knihovně Apache Portable Runtime (APR) nyní Apache funguje stejně dobře na Unixu i ve Windows. Tato knihovna zajišťuje abstrakční vrstvu pro funkce, které se na obou platformách liší, například pro souborové nebo síťové funkce. Přenos Apache na jinou platformu obnáší pouze přenesení této knihovny. Knihovna navíc implementuje platformně závislé optimalizace.

Bezpečnost

Apache obsahuje několik bezpečnostních modulů k zabezpečení a omezení přístupu k serveru.

Autentizace

Autentizační moduly umožňují ověřit identitu klienta, obvykle kontrolou jména a hesla proti nějaké databázi. Apache obsahuje moduly pro autentizaci proti textovým souborům nebo databázím. Existují i další autentizační moduly, které Apache napojí na jiné zabezpečovací platformy nebo databáze včetně NT domény, Oracle, mySQL, PostgresSQL a dalších.

Velmi zajímavé jsou moduly LDAP, protože umožňují integraci s existujícími adresářovými službami. Tyto moduly naleznete na adrese *http://modules.apache.org*. LDAP modul pro Apache 2.0 najdete na stránkách *http://httpd.apache.org/docs-2.0/mod/mod_auth_ldap.html*.

Řízení přístupu

Apache obsahuje modul mod_access, který umožňuje omezit přístup k prostředkům na základě obsahu požadavku, například na základě existence určité hlavičky, adresy IP nebo názvu počítače. Další moduly umožňují omezit přístup klientů s nevhodným chováním, budeme o nich hovořit dále v části věnované výkonu a omezení přenosu.

SSL/TLS

Protokoly Secure Sockets Layer / Transport Layer Security umožňují šifrovat data přenášená mezi serverem a klientem. Ve verzi 1.3 tyto protokoly implementuje modul mod_ssl, dodávaný samostatně na stránkách *http://www.modssl.org/*, přičemž jeho použití si vyžaduje zásah do zdrojového kódu serveru. Toto řešení bylo nezbytné kvůli vývozním omezením na silnou kryptografii. Většina těchto omezení už dnes neplatí a ve verzi 2.0 je mod_ssl standardní součástí distribuce.

Proxy – server

Proxy – server je program, který zpracovává požadavky pro někoho jiného. Existuje celá řada webových proxy – serverů. Klasický proxy – server přijímá požadavky od klientů (typicky webových prohlížečů), kontaktuje vzdálený server a vrací odpověď.

Reverzní proxy je webový server umístěný před jinými servery a poskytuje jednotné rozhraní a odlehčuje koncovým serverům od určitých úkonů (například přenosů SSL).

Apache podporuje oba typy proxy – serverů.

Výkon a škálovatelnost

Samotný výkon je pouze jedním z faktorů, které je potřeba u webového serveru brát v úvahu (prvními jsou obvykle flexibilita a stabilita).

Jak už bylo řečeno, existují řešení pro zvýšení výkonu na velmi zatížených webových serverech, které poskytují statický obsah. Pokud provádíte hosting, server Apache také poskytuje způsoby, kterými můžete zjišťovat a řídit využití šířky pásma. Omezení v této souvislosti obvykle znamená zpomalení poskytnutí obsahu založeného na požadovaném souboru, konkrétní adrese IP klienta atd. To se dělá kvůli znemožnění zneužití.

- **mod_mmap**: Zahrnutý v aktuální verzi Apache 1.3, mapuje do paměti staticky konfigurovaný seznam často požadovaných a málo modifikovaných souborů. Ve verzi 2.0 tuto funkci zajišťuje modul mod_file_cache.
- **Mod_bandwidth** (*http://www.cohprog.com/mod_bandwidth.html*): Module pro Apache 1.3, umožňuje nastavit omezení šířky pásma pro servery nebo připojení, založené na konkrétním adresáři, velikosti souborů a vzdálené adrese IP nebo doméně.
- **Bandwidth share module** (*http://www.topology.org/src/bwshare/README.html*): umožňuje omezování a vyrovnávání šířky pásma podle adresy IP klienta. Funguje pro řadu 1.3 a starší verze řady 2.0.
- **Mod_throttle** (*http://www.snert.com/Software/Throttle/index.shtml*): Omezení šířky pásma pro virtuálního hostitele nebo uživatele. Funguje v řadě 1.3.

Vyrovnávání zátěže

Pomocí reverzního proxy a modulu mod_rewrite může Apache distribuovat požadavky na další servery. Podrobnosti jsou popsány na adrese *http://www.apache.org/docs/misc/rewriteguide.html*.

Modul mod_backhand pro Apache 1.3 umožňuje transparentní přesměrování požadavků HTTP z jednoho webového serveru na jiný. Toto přesměrování může být prováděno na cílové stroje s méně využitými prostředky, čímž se provádí jemné vyrovnávání zátěže pro webové požadavky. Více informací najdete na adrese *http://www.backhand.org/*.

Komprese

Apache 2.0 obsahuje modul mod_deflate, filtr, který zajišťuje komprimaci dat před přenosem klientům. Šetří se tím přenosové pásmo, může však dojít k poklesu výkonu. Pro Apache 1.3 tuto funkci zajišťuje modul mod_gzip (*http://www.remotecommunications.com/apache/mod_gzip/*).

CGI skripty

CGI znamená Common Gateway Interface. Skripty CGI jsou externí programy, které jsou volány, když uživatel požaduje konkrétní stránku. CGI obdrží informace od webového serveru (hodnoty proměnných z formulářů, druh prohlížeče, IP adresa klienta atd.) a použije tyto informace pro vytvoření webové stránky pro klienta.

Apache podporuje CGI a pro řadu 1.3 existuje modul podporující protokol FastCGI. Tím se snižuje režie spojená se spouštěním a ukončováním CGI programů pro každý požadavek. Více viz *http://fastcgi.com/*.

Integrace vývojových platforem

Webové aplikace se obvykle píší ve vyšších jazycích, jako jsou Java, Perl, C# a podobně. Apache nabízí několik modulů pro integraci těchto jazyků do serveru. V řadě případů tyto moduly poskytují kompletní API Apache, takže v příslušném jazyce je možné vytvářet vlastní moduly.

Perl

Mod_perl (*http://perl.apache.org/*) je jedním z nejstarších a nejúspěšnějších projektů Apache. Vkládá interpret jazyka Perl do Apache a umožňuje přístup z jazyka Perl k interním prvkům webového serveru. Umožňuje vytváření celých modulů v jazyce Perl nebo kombinaci kódu jazyků Perl a C. V serverech Apache verze 1.3 musí být každý interpret vložen do jednoho podřízeného procesu, jelikož je server víceprocesový. V hodně zatížených dynamických serverech může nárůst velikosti serveru vadit. Apache 2.0 je vícevláknový, stejně jako poslední verze jazyka Perl. Další generace modulu mod_perl toho využívá a umožňuje sdílení kódu, dat a stavů relací mezi interprety. To jej dělá rychlejším a spolehlivějším řešením.

Mod_perl sám o sobě představuje vlastní platformu s podporou řady dalších modulů, například Mason (*http://www.masonhq.com*) a Embperl (*http://perl.apache.org/embperl/*) pro vkládání Perlu přímo do stránek HTML, nebo AxKit (*http://axkit.org*) podporující šablony v XML.

PHP

Z webového serveru PHP (*http://www.php.net*): *PHP je skriptovací jazyk vkládaný do HTML a nezávislý na platformě.* Je to nejpopulárnější modul pro Apache, a to z různých důvodů:

- Jeho studium netrvá dlouho
- Je k dispozici skvělá dokumentace
- Rozsáhlá podpora databází
- Modularita

PHP je modulární. Existují moduly pro podporu:

- Konektivita k databázím Oracle, MS-SQL, ODBC, MySQL, mSQL, PostgreSQL a mnoha dalším
- Podpora XML
- Přenos souborů: FTP
- HTTP
- Podpora adresářových služeb: LDAP
- Podpora pošty a news: IMAP, POP3, NNTP
- Generování dokumentů PDF
- CORBA
- SNMP

Stačí pouze zkompilovat nebo použít moduly, které potřebujete. PHP může být použito přímo v Apache, jako externí CGI nebo v jiných webových serverech. Je nezávislé na platformě a běží na většině unixových systémů i systémů Windows. Pokud pracujete na platformě Windows, pravděpodobně používáte Internet Information Server s ASP (Active Server Pages) a serverem MS-SQL. Podobnou náhradou ve světě Unixu pro tuto trojici je Apache s PHP a mySQL. Jelikož PHP pracuje:

- s Apache i Microsoft IIS
- s mySQL i MS-SQL
- na Unixu i ve Windows

je zde krásná možnost k přechodu z řešení od Microsoftu k bezpečnějším, stabilnějším a výkonným řešením pro Unix.

Python

Python je skriptovací jazyk podobný jazykům Perl nebo Tcl. Mod_python (*http://www.modpython.org/*), dnes už oficiální součást projektu Apache, zajišťuje integraci Pythonu. Můžete jej využít k vývoji složitých aplikací nebo ke zrychlení stávajících skriptů. Poslední verze podporují i Apache 2.0.

Tcl

Projekt Tcl Apache (*http://tcl.apache.org/*) integruje Tcl do webového serveru Apache. Tcl je malý, rozšiřitelný skriptovací jazyk. Více se o Tcl můžete dozvědět na adrese *http://dev.ajubasolutions.com/default.htm*. Pod záštitou Apache Tcl existuje aktuálně více modulů:

- Mod_dtcl (*http://tcl.apache.org/mod_dtcl/*) a Neowebscript (*http://tcl.apache.org/neowebscript/*) umožňují vkládání Tcl do stránek HTML. To nejlepší z obou modulů kombinuje Rivet (*http://tcl.apache.org/rivet/*).
- Mod_tcl (*http://tcl.apache.org/mod_tcl/mod_tcl.html*) má podobný přístup jako mod_perl a zpřístupňuje API Apache.
- WebSH (*http://tcl.apache.org/websh/*) poskytuje prostředí pro webové aplikace.

Technologie Microsoft

Existuje několik modulů umožňujících integraci s jazyky a technologiemi společnosti Microsoft, jako jsou například platforma .Net nebo Active Server Pages.

.Net

Mod_haydn (*http://haydn.sourceforge.net/*) zajišťuje integraci Mono (*http://www.go-mono.com/*) a Apache a zpřístupňuje API Apache platformě .Net, takže vám například umožňuje psát moduly v jazyce C#. Společnost Covalent (*http://www.covalent.net/*) nabízí modul mod_asp.net, komerční modul pro Windows, který umožňuje v Apache používat aplikace ASP.Net a nahradit tak server Microsoft IIS.

ASP

ASP znamená Active Server Pages a jde o technologii společnosti Microsoft, která umožňuje do stránek HTML vkládat kód, typicky v jazyce Visual Basic. Spouštění ASP na unixových platformách umožňují produkty různých firem, například ChilliSoft (*http://www.chillisoft.com/*) nebo Stryon (*http://www.stryon.com/*).

ISAPI

ISAPI je API rozšiřující server Microsoft IIS podobně, jako lze použít API Apache. Apache obsahuje modul mod_isapi, který tuto funkci implementuje a umožňuje tak spouštět moduly určené pro ISAPI.

Java

Většina aplikačních serverů společností jako Oracle, IBM a BEA obsahuje i moduly pro integraci se serverem Apache. Existují rovněž moduly jako mod_jk a mod_webapp, které umožňují propojení na Tomcat, což je podpora servletů vyvíjená rovněž v Apache Software Foundation.

Moduly pro další jazyky

Hovořili jsme o modulech podporujících populární jazyky jako Perl, Python nebo PHP. Moduly pro podporu dalších jazyků (například JavaScript, Haskell, Ruby a jiných) najdete na adrese *http://modules.apache.org/*.

Správa serveru

Důležitou součástí správy webového serveru je vytváření, konfigurace a monitorování různých služeb.

Vývojové nástroje

Apache je možno rozšiřovat a upravovat různými způsoby. Integrace různých modulů do serveru může být někdy obtížné. Usnadňují to interaktivní nástroje jako Apache Toolbox (*http://www.apachetoolbox.com/*).

Grafická rozhraní pro Apache

Server Apache je konfigurován pomocí textových konfiguračních souborů, což může být komplikované zejména pro správce zvyklé na platformu Windows. Existují open – source grafické nástroje, které tuto úlohu zjednodušují:

- Comanche (*http://www.comanche.org/*) existuje pro více platforem – Unix/Linux, Windows a Mac.
- gui.apache.org (*http://gui.apache.org/*): Grafická uživatelská rozhraní pro projekt Apache. Programy s různým stupněm vývoje.
- Webmin (*http://www.webmin.com/webmin/*): Hezké webové rozhraní.

SNMP

SNMP znamená Simple Network Management Protocol. Umožňuje sledování a správu síťových serverů, vybavení atd. Moduly SNMP pro Apache pomáhají spravovat rozsáhlé nasazení webových serverů, zjišťovat kvalitu nabízených služeb a integraci Apache ve stávajících strukturách správy.

- Open source Mod_SNMP pro Apache 1.3 (*http://www.simpleweb.org/software/packages/mod-snmp/*).
- Covalent SNMP (*http://www.covalent.net/*) je komerční modul SNMP, podporující nejnovější standard SNMPv3, integraci s HP-Openview, Tivoli atd.

Publikování

Vytváření webových stránek vyžaduje nástroje pro správu obsahu a jeho nahrávání na server. Jedním z protokolů, které k tomuto účelu slouží, je DAV (Distributed Authoring and Versioning). DAV je rozšířením protokolu HTTP, které umožňuje uživatelům a aplikacím publikovat a modifikovat webové stránky. Technologie DAV je široce používaná, Microsoft ji implementuje přímo na úrovni operačního systému (WebFolders) a v balíku Office. Najdete ji i v Apple OS-X a řadě dalších produktů od Adobe, Oracle a dalších. Pro Apache 1.3 můžete modul mod_dav získat na adrese *http://www.webdav.org/mod_dav/*. V řadě 2.0 je mod_dav přímo součástí distribuce.

Před nástupem protokolu DAV používal Microsoft vlastní publikační technologii, integrovanou v nástroji FrontPage. Podporu této technologie nabízejí moduly na adrese *http://www.rtr.com/Ready-to-Run_Software/*, nicméně jejich způsob integrace do serveru není považován za bezpečný.

Protokolové moduly

Apache 2.0 zavádí techniku protokolových modulů. Znamená to, že vývojáři mohou použít platformu Apache k implementaci nových protokolů, podporujících například poštu nebo přenosy souborů. mod_ftp je komerční modul pro podporu protokolu FTP společnosti Covalent

(*http://www.covalent.net/*). Modul mod_pop3 je open-source modul implementující protokol POP3, používaný pro přístup k poště na poštovních serverech.

Virtuální hostitelé

Apache poskytuje rozsáhlou podporu virtuálních hostitelů, tedy technologie, kdy jeden fyzický server funguje jako více webových serverů. U Apache 2.0 s podporou MPM můžete vytvořit více potomků, každý bude obsluhovat vlastní doménu pod jiným UID. Je to velmi výhodné z bezpečnostního hlediska v prostředí webového hostingu, protože tak lze vzájemně izolovat jednotlivé zákazníky. Následující přehled uvádí alternativní moduly podporující virtuální servery:

- mod_dynvhost (*http://funkcity.com/0101/*)
- mod_pweb (*http://www.joytec.de/mod_pweb.html*)
- mod_v2h (*http://www.fractal.net/mod_v2h.tm*)

Komerční podpora

Apache se používá v řadě komerčních firem, včetně velkých společností. Tyto společnosti mají obvykle přísné nároky na používané technologie, zejména pokud se jedná o takové jádro internetových služeb, jakým je webový server. Mezi tyto nároky patří výkon, stabilita, možnosti správy, podpora, profesionální služby a integrace se stávajícími systémy. Produkty a služby odpovídající těmto požadavkům nabízí řada komerčních společností, například IBM (*http://www.ibm.com/*), Red Hat (*http://www.redbat.com/*) nebo Covalent (*http://www.covalent.net/*).

Kromě toho je Apache dodáván jako server OEM s řadou jiných produktů.

Další projekty ASF

I když je Apache asi nejpopulárnějším projektem ASF, zastřešuje ASF i řadu dalších projektů. V této kapitole uvádíme přehled těch nejdůležitějších. Většina z nich je součástí projektů Jakarta a XML. Projekt Jakarta zastřešuje projekty věnované jazyku Java, projekt XML pak překvapivě projekty věnované XML.

Aplikace a platformy

Následuje přehled aplikací a vývojových platforem, které jsou součástí ASF.

Servery

Toto jsou některé ze serverů vyvíjených v ASF:

Tomcat

Tomcat je stěžejním produktem projektu Jakarta. Je to oficiální referenční implementace pro technologie Java Servlet a JavaServer Pages.

Více se dozvíte na adrese *http://jakarta.apache.org/tomcat/*.

JAMES (Java Apache Mail Enterprise Server)

Jako doplněk k jiným technologiím Apache na straně serveru JAMES poskytuje *server 100% v jazyce Java navržený tak, aby byl kompletním a přenositelným podnikovým řešením poštovního systému založeného na aktuálně dostupných otevřených protokolech (SMTP, POP3, IMAP, HTTP).*

Více informací viz *http://jakarta.apache.org/james/*.

Lucene

Jakarta Lucene je vysoce výkonný, plně funkční textový vyhledávač napsaný v jazyce Java, který je součástí projektu Jakarta. Více informací naleznete na adrese *http://jakarta.apache.org/lucene/*.

Jetspeed

Jetspeed je webový portál vytvořený v jazyce Java. Má modulární API, které umožňuje seskupení různých zdrojů dat (XML, SMTP, iCalendar).

Více informací najdete na adrese *http://jakarta.apache.org/jetspeed/*.

Správa obsahu

Slide

Slide je systém správy obsahu vysoké úrovně. Koncepčně poskytuje hierarchickou organizaci binárního obsahu, který může být uložen do libovolných, heterogenních, distribuovaných datových úložišť. Kromě toho Slide integruje služby pro zabezpečení, uzamknutí a informace o verzích. Obsahuje také implementace klienta a serveru WebDAV (*http://www.webdav.org/*).

Více na domovské stránce projektu na adrese *http://jakarta.apache.org/slide/index.html*.

Alexandria

Alexandria je integrovaný systém správy dokumentace. Spojuje technologie společné pro mnoho open-source projektů, jako jsou CVS a JavaDoc. Cílem je integrace zdrojového kódu a dokumentace, což podpoří dokumentování a sdílení kódu. Více informací najdete na adrese *http://jakarta.apache.org/alexandria/index.html*.

Vývojové platformy

Následuje přehled platforem pro vývoj aplikací.

Turbine

Turbine je systém založený na servletech, který umožňuje zkušeným vývojářům v jazyce Java rychle sestavovat zabezpečené webové aplikace. Turbine spojuje platformu pro spouštění kódu Java a znovupoužitelné komponenty. Některé z jeho funkcí jsou integrace se systémy šablon, vývoj typu MVC, seznamy řízení přístupu, podpora lokalizace a další. Více informací naleznete na adrese *http://java.apache.org/turbine*.

Avalon

Pokud znáte Perl nebo systémy BSD, Avalon je zhruba ekvivalentem technologií CPAN nebo portů BSD pro Apache. Neposkytuje pouze pomůcku pro společné uložení kódu, je o krok napřed: *jde o program pro vytváření, návrh, vývoj a správu společného systému serverových aplikací vytvořených v jazyce Java*. Poskytuje prostředky k tomu, aby projekty v jazyce Java na straně serveru mohly být jednoduše integrovány a stavěny na sobě navzájem. Více viz *http://java.apache.org/avalon/*.

Prezentace

Následují systémy pro práci se šablonami, transformační mechanismy a další prezentační projekty.

Cocoon

Cocoon pomocí jiných technologií Apache XML, jako např. Xerces, Xalan a FOP, poskytuje všeobecný publikační systém. Dokáže se domluvit s řadou různých datových zdrojů a transformovat

obsah do různých formátů, jako je PDF, HTML, XML a RTF. Může běžet jako servlet nebo jako samostatný program. Více viz *http://xml.apache.org/cocoon/*.

Velocity

Velocity je systém šablon založených na jazyce Java. Může být použit jako samostatná utilita pro generování zdrojového kódu, HTML, zpráv, nebo může být kombinován s jinými systémy pro poskytování služeb šablon. Velocity má paradigma Model View Controller, které vyžaduje oddělování kódu Java od šablony HTML. Více viz *http://jakarta.apache.org/velocity/index.html*.

AxKit

AxKit (*http://axkit.org/*) je populární aplikační server XML určený pro mod_perl a Apache. Umožňuje oddělení obsahu a prezentace a provádí on line konverzi XML na libovolný formát.

Xalan

Xalan je procesor XSLT dostupný pro jazyky Java a C++. XSL je jazyk šablon pro XML. T znamená Transformace. XML je dobrý pro ukládání strukturovaných dat (informací). Někdy potřebujeme tato data zobrazit uživateli nebo aplikovat nějakou transformaci. Xalan bere původní dokument XML, načítá konfiguraci transformace a jeho výstupem je HTML, holý text nebo jiný dokument XML. Více se dozvíte na stránkách *http://xml.apache.org/xalan-j/index.html* a *http://xml.apache.org/xalan-c/index.html*.

FOP

Podle svých webových stránek je *FOP aplikace v jazyce Java, která načítá strom formátovacích objektů a pak jej uloží do dokumentu PDF*. Takže FOP bere dokument XML a provádí výstup PDF podobným způsobem, jakým to Xalan dělá s HTML nebo textem. Více se o FOP dozvíte na adrese *http://xml.apache.org/fop*.

Parsery a knihovny pro přístup k dokumentům

Následuje přehled knihoven, které lze použít ke zpracování a manipulaci s dokumenty v řadě různých formátů.

Xerces

Projekt Xerces poskytuje parsery XML pro různé jazyky, včetně Java, C++ a Perl. Vazby Perl jsou založeny na zdrojích C++. Parser XML je nástroj používaný pro programový přístup k dokumentům XML. Toto je popis standardů podporovaných projektem Xerces:

- DOM (*http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/level-one-core.html*): DOM znamená Document Object Model. Dokumenty XML jsou přirozeně hierarchické (vnořené značky). Dokumenty XML jsou přístupné skrze tři podobná rozhraní. Proces je následující:
  - Analýza dokumentu
  - Vytvoření stromu
  - Přidání/odstranění/úpravy uzlů
  - Serializace stromu
- SAX (*http://www.saxproject.org/*): Jednoduché API pro XML. Je to API založené na proudech. To znamená, že obdržíme zpětná volání tak, jak se elementy vyskytly. Tato zpětná volání mohou být použita např. pro konstrukci stromu DOM.
- XML Namespaces (*http://www.w3.org/TR/REC-xml-names/*).

- XML Schema: Standard XML poskytuje syntaxi pro vytváření dokumentů. XML Schema poskytuje nástroj pro definování *obsahu* dokumentu XML (sémantiku). Umožňuje definovat, že určitý element v dokumentu musí být celé číslo mezi 10 a 20 atd.

Původní základna kódu projektu Xerces XML byla věnována společností IBM. Více informací najdete na adresách *http://xml.apache.org/xerces-j/index.html*, *http://xml.apache.org/xerces-c/index.html* a *http://xml.apache.org/xerces-p/index.html*.

Batik

Batik je sada nástrojů založená na jazyce Java pro aplikace, které chtějí obrázky ve formátu SVG (Scalable Vector Graphics, http://www.w3.org/TR/SVG/) pro řadu účelů, jako např. zobrazování, generování nebo manipulaci.

Je založený na XML a kompatibilní se specifikací W3C. Je trochu netypický pro projekty Apache, v tom, že obsahuje grafický komponent. Batik obsahuje propojení pro rozšíření systému pomocí vlastních značek a to umožňuje konverzi z SVG do jiných formátů, jako např. JPEG nebo PNG.

Domovská stránka projektu Batik je na adrese *http://xml.apache.org/batik/*.

POI

POI je projekt obsahující API pro manipulaci s různými souborovými formáty založenými na Microsoft OLE 2 Compound Document. Patří sem například dokumenty ve formátu Word nebo Excel. Více viz *http://jakarta.apache.org/poi/*.

Interoperatibilita

Následující knihovny slouží ke vzdálené komunikaci a spolupráci mezi servery.

SOAP

Apache SOAP („Simple Object Access Protocol“) a Axis jsou implementace protokolu SOAP (*http://www.w3.org/TR/SOAP*).

SOAP je protokol pro výměnu informací v decentralizovaném, distribuovaném prostředí. Je to protokol založený na XML, který se skládá ze tří částí:

- Obálka, která definuje systém popisu toho, co je ve zprávě a jak se má zpracovat,
- sada pravidel kódování pro vyjadřování instancí aplikací definovaných datových typů,
- a konvence pro reprezentaci vzdálených volání procedur a odpovědí.

Považujte SOAP za systém vzdálených volání procedur založený na HTTP a XML. Na jednu stranu to znamená, že je ve srovnání s jinými systémy užvaněný a pomalý. Na druhou stranu zjednodušuje univerzálnost, ladění a vývoj klientů a serverů pro řadu jazyků (C, Java, Perl, Python, Tcl atd.), jelikož má většina moderních jazyků moduly HTTP a XML. Více se dozvíte na adrese *http://xml.apache.org/soap/*.

XML-RPC

Projekt XML-RPC (*http://xml.apache.org/xmlrpc/*) v Javě implementuje protokol XML-RPC, obdobu protokolu SOAP.

XML security

Projekt XML security (*http://xml.apache.org/security/*) zajišťuje ověřování podpisu XML dokumentů při zabezpečené výměně dokumentů.

Vývoj

Apache Portable Runtime

Projekt APR (*http://apr.apache.org/*) představuje přenositelnou vrstvu, která abstrahuje od řady systémových volání pro práci se soubory, sítí a podobně. Je napsán v jazyce C a pracuje na většině Unixů, ve Windows a v dalších systémech. Je základem Apache 2.0.

Ant

Ant (*http://jakarta.apache.org/ant/*) je v Javě vytvořený nástroj pro sestavování projektů. Má modulární API a lze jej rozšířit o různé schopnosti. Konfigurační soubory jsou zapsány v XML.

Byte Code Library

Byte Code Engineering Library (BCEL, *http://jakarta.apache.org/bcel/*) je knihovna pro analýzu, vytváření a manipulaci s binárním kódem jazyka Java.

Log4j

Tento balíček poskytuje logovací systém, který mohou aplikace v jazyce Java používat. Může být zapnut za běhu bez úprav binárního kódu a byl navržen s ohledem na výkon. Více viz *http://jakarta.apache.org/log4j/*.

ORO a Regexp

ORO je kompletní balíček, který umožňuje podporu regulárních výrazů pro jazyk Java. Zahrnuje podporu regulárních výrazů jazyka Perl5 apod. Všechno pod licencí Apache. Více informací o ORO najdete na adrese *http://jakarta.apache.org/oro/index.html*. Tam je k dispozici také další, jednodušší, balík pro práci s regulárními výrazy, Regexp, *http://jakarta.apache.org/regexp/*.

Struts

Struts je projekt Apache, který se pokouší přenést paradigma návrhu Model-View-Controller (MVC) do vývoje pro web. Staví na technologiích Servlet (*http://java.sun.com/products/servlet*) a JavaServer Pages (*http://java.sun.com/products/jsp*). Modelová část jsou objekty serveru Java, které reprezentují interní stav aplikací. Enterprise Java Beans jsou zde všeobecně používané. Zobrazovací část je konstruována skrze JavaServer Pages (JSP), které jsou kombinací statického HTML/XML a jazyka Java. JSP také vývojáři umožňují definovat vlastní značky. Řídicí část jsou servlety, které dostávají požadavky (GET/POST) od klienta, provádí akce na modelu a aktualizují zobrazení poskytnutím příslušného JSP. Více se můžete dozvědět na adrese *http://jakarta.apache.org/struts/index.html*.

Taglibs

Technologie JavaServer Pages umožňuje vývojářům poskytovat funkce přidáním vlastních značek. Projekt Taglibs má být společným úložištěm pro tato rozšíření. Zahrnuje značky pro společné utility (například date), přístup k databázi SQL atd.

Další informace naleznete na *http://jakarta.apache.org/taglibs/*. Více dokumentace je zahnuto v balíčku.

Databáze

OJB (*http://jakarta.apache.org/ojb/*) je nástroj pro mapování databází, který umožňuje trvalé ukládání objektů jazyka Java v relačních databázích. Xindice (*http://xml.apache.org/xindice/*) je nativní XML databáze určená k ukládání a vyhledávání dokumentů XML.

Commons

Projekt Commons (*http://jakarta.apache.org/commons/*) nabízí celou řadu univerzálních komponent jazyka Java s minimálními závislostmi.

Testování

Následující projekty se zaměřují na testování a analýzu výkonu.

httpd-test

Projekt httpd-test (*http://httpd.apache.org/test/*) představuje platformu pro testování webového serveru Apache a obsahuje například nástroje jako flood (*http://httpd.apache.org/test/flood/*) pro zátěžové testy.

Cactus

Cactus (*http://jakarta.apache.org/cactus/*) je platforma pro testování serverového kódu Java, jako jsou servlety a EJB.

JMeter

Testovací nástroj napsaný v Javě s grafickým rozhraním. Najdete jej na adrese *http://jakarta.apache.org/jmeter/*.

Lakta

Lakta (*http://jakarta.apache.org/lakta/*) je nástroj pro testování HTTP.

Watchdog

Projekt Watchdog (*http://jakarta.apache.org/watchdog/*) poskytuje ověřovací testy pro specifikace Servlet a JavaServer Pages.

Kde nalézt další informace

Další informační prameny věnované serveru Apache.

Webové stránky

http://www.apache.org/

http://www.apacheweek.com/

http://modules.apache.org/

http://www.apachetoday.com/

http://www.apacheworld.org/

http://slashdot.org/index.pl?section=apache

Knihy

Na adrese *http://www.apacheworld.org/apache_overview/books/* naleznete seznam knih věnovaných serveru Apache. Nejde o úplný seznam, jsou v něm pouze knihy, které považuji za dobře napsané a užitečné.

Fóra

Diskusní seznam věnovaný serveru Apache najdete na adrese *http://httpd.apache.org/lists.html*. Obdobné seznamy existují i pro většinu výše zmíněných projektů. Nejprve si přečtěte seznam často kladených otázek, teprve pak se ptejte.

Pokud hledáte komerční podporu, můžete zkusit Covalent (*http://www.covalent.net/*), který nabízí velmi kvalitní podporu serveru Apache. Pokud používáte Apache na Linuxu, možná zajišťuje podporu Apache přímo tvůrce distribuce.

Bezpečnost Linuxu

Originál: http://tldp.org/HOWTO/Security-HOWTO/

Tento dokument představuje obecný přehled bezpečnostní problematiky vztahující se ke správě linuxových systémů. Hovoří o obecných bezpečnostních postupech a uvádí řadu konkrétních příkladů, jak zabezpečit linuxový systém před útoky. Dále obsahuje odkazy na materiály a programy vztahující se k bezpečnosti.

Úvod

Tento dokument hovoří o hlavních bodech týkajících se bezpečnosti linuxových systémů. Popisuje obecné postupy a odkazuje se na další informační prameny.

S problematikou bezpečnosti souvisí i řada dalších praktických návodů, na které se v případě potřeby odkazujeme. Tento dokument nemá za cíl představovat aktuální rizika. Nová rizika se objevují velmi často. Zde se dočtete, kde takovéto aktuální informace získat a dále se dozvíte o obecných postupech, jak řadu rizik eliminovat.

Přehled

Tento dokument se pokouší objasnit některé postupy a běžně používané programy, jejichž smyslem je zvýšit bezpečnost linuxových systémů. Nejprve je ale nutné seznámit se s některými základními informacemi o celé problematice.

Proč bezpečnost potřebujeme?

Ve stále se měnícím světě globálních datových komunikací, levných internetových linek a rychlého vývoje programů se bezpečnost stává stále větším a větším problémem. Bezpečnost představuje základní požadavek, protože globální komunikace ze své podstaty bezpečná není. Při přenosu dat mezi místy A a B na Internetu mohou data procházet celou řadou uzlů, kde mají jiní uživatelé možnost data vidět, případně i modifikovat. I ostatní uživatelé vašeho systému mohou záměrně modifikovat vaše data způsobem, který si nepřejete. Neautorizovaný přístup k vašemu systému může získat útočník (označovaný také jako „cracker"), který se pak může za vás vydávat, odcizit vaše data nebo vám dokonce zabránit v přístupu k vlastnímu systému. Pokud vás zajímá, jaké jsou rozdíly mezi „hackerem" a „crackerem", doporučujeme vám dokument Erika Raymonda „How to Become a Hacker", který je dostupný na adrese http://www.tuxedo.org/~esr/faqs/hacker-howto.html.

Jak moc bezpečné je bezpečné?

Je nutné mít na paměti, že žádný počítačový systém nemůže být úplně bezpečný. Jediné, čeho můžete dosáhnout, je to, aby bylo narušení systému různě obtížné. U běžného počítačového uživatele nejsou zapotřebí žádná speciální opatření k zabezpečení systému. U náročných uživatelů (banky, telekomunikační firmy a podobně) je zapotřebí vyšší míra zabezpečení.

Dále je třeba vzít v úvahu, že čím je systém bezpečnější, tím nepříjemnější jsou příslušná bezpečnostní opatření. Je nutné se rozhodnout, kde je ona hranice, při níž je systém dostatečně bezpečný a zároveň stále rozumně použitelný. Můžete například požadovat, aby u všech uživatelů, kteří se k vašemu systému připojují telefonicky, bylo provedeno zpětné volání na jejich telefonní číslo. Je to bezpečnější, než přímé přihlašování z jakéhokoliv čísla, ale komplikuje to život uživatelům, kteří se zrovna nepřipojují z domova. Můžete počítač nainstalovat bez síťového rozhraní a bez připojení k Internetu, tím se ale limituje jeho použitelnost.

U středních a větších sítí je rozumné vytvořit bezpečnostní politiku, která definuje, jaká míra bezpečnosti je požadována a jakým způsobem se kontroluje. Známý příklad bezpečnostní politiky můžete najít na http://www.faqs.org/rfcs/rfc2196.html. Tento dokument byl nedávno aktualizován a představuje vynikající základ pro vytvoření vaší vlastní bezpečnostní politiky.

Co se snažíte chránit?

Než začnete svůj systém zabezpečovat, měli byste stanovit, proti jakému ohrožení jej chcete chránit, která rizika jste a nejste ochotni nést a jaká bude výsledná zranitelnost systému. Měli byste provést analýzu systému, abyste věděli co chráníte, proč to chráníte, jakou to má hodnotu a kdo je zodpovědný za data a další hodnoty.

- *Riziko* představuje možnost, že by útočník mohl úspěšně získat přístup k vašemu systému. Může pak takový útočník číst a zapisovat soubory? Může spouštět programy, které by mohly vyvolat nějakou škodu? Může smazat kritická data? Může vám nebo vaší firmě zabránit v plnění důležitých úkolů? Nezapomínejte také, že pokud někdo získá přístup k vašemu účtu a k vašemu systému, může se také vydávat za vás. Navíc, jakmile dojde k napadení jednoho účtu na jednom počítači, může následně dojít k napadení celé sítě. Pokud například povolujete uživatelům přihlašovat se pomocí souborů .rhosts, nebo používáte nezabezpečené služby jako tftp, pootevíráte tím útočníkovi dveře. Jakmile se mu podaří získat přístup k jednomu účtu na některém systému, může jej následně použít k získání přístupu k jiným účtům a k jiným systémům.
- *Ohrožení* zpravidla pochází od někoho, kdo má nějakou motivaci získat neautorizovaný přístup k vaší síti nebo počítači. Musíte uvážit, komu chcete přístup umožnit a jaké ohrožení to může představovat. Existuje několik typů útočníků a při zabezpečování systému je rozumné mít na paměti jejich charakteristiky:
  - Zvědavec – Tento typ útočníka se typicky zajímá, co je váš systém zač a jaká data obsahuje.
  - Záškodník – Tento typ útočníka se snaží buď váš systém vyřadit z činnosti, nebo poškodit vaše webové stránky, nebo vás prostě nějakým jiným způsobem donutit vynaložit čas a prostředky na nápravu jím způsobené škody.
  - Ambiciózní útočník – Tento typ se snaží napadením vašeho systému získat popularitu. Napadá exponované systémy, aby dokázal své schopnosti.
  - Konkurent – Tento typ útočníka se zajímá o data, která v systému máte. Může se domnívat, že máte něco, co mu může přinést prospěch, ať finanční nebo jiný.

- Vypůjčovatel – Tento typ útočníka má zájem využít váš systém a jeho prostředky ke svým účelům. Typicky chce provozovat chatové nebo servery irc, archivy pornostránek, nebo dokonce servery DNS.
- Skokan – Tento útočník se váš systém snaží použít pouze jako prostředek k napadení dalších systémů. Pokud má váš systém dobré připojení, nebo slouží jako brána k dalším systémům, můžete očekávat tento typ útoků.

- *Zranitelnost* udává, jak dobře je váš počítač chráněn před jinými systémy a jaká je možnost získat k němu neautorizovaný přístup. Co je v sázce, jestliže se někdo do systému naboural? Samozřejmě se to liší u domácího uživatele připojeného modemem, a u společnosti s počítači připojenými k Internetu a k jiným sítím.

Kolik času vám zabere obnovení ztracených dat? Počáteční časová investice na zabezpečení vám může ušetřit deseti i vícenásobek času později při obnovování dat. Pořizujete pravidelně zálohy a kontrolujete jejich funkčnost?

Vytvoření bezpečnostní politiky

Vytvořte jednoduchou obecnou bezpečnostní politiku, kterou budou vaši uživatelé znát a budou se jí řídit. Měla by chránit jak data, tak i soukromí uživatelů. Další možnosti, které stojí za zvážení, jsou: kdo má mít přístup k systému (může můj účet používat můj kamarád?), kdo je oprávněn instalovat programy, kdo vlastní která data, jak provádět obnovu v případě poškození, a jaké je akceptovatelné užití systému.

Obecně přijímané bezpečnostní politiky začínají větou:

Co není povoleno, je zakázáno.

Znamená to, že pokud uživateli nepovolíte přístup ke službě, uživatel ji nesmí použít. Zajistěte, aby politika fungovala i pro běžné uživatelské účty. Přístup „toto nejsem schopen jednoduše udělat, udělám to jako *root*“ může vést ke vzniku zjevných bezpečnostních děr, které ani nemusí být v daném okamžiku využitelné.

Dokument RFC1244 popisuje, jak vytvořit vlastní síťovou bezpečnostní politiku.

Dokument RFC1281 představuje příklad bezpečnostní politiky s podrobným vysvětlením.

Konečně se můžete podívat do archivu politik na adrese ftp://coast.cs.purdue.edu/pub/doc/policy, kde najdete příklady skutečných bezpečnostních politik.

Opatření k zabezpečení systému

Tento dokument popisuje různá opatření k zabezpečení hodnot, na nichž jste tvrdě pracovali: zabezpečení počítače, dat, uživatelů, sítě, dokonce i reputace. Jak dopadne vaše pověst, pokud útočník smaže data nějakého uživatele? Nebo pokud změní vaše webové stránky? Nebo zveřejní plány vaší společnosti na příští čtvrtletí? Pokud plánujete instalaci sítě, musíte vzít v potaz celou řadu faktorů ještě předtím, než k síti připojíte jediný počítač.

I pokud používáte pouze modemové připojení, nebo provozujete malinkou síť, neznamená to, že se o vás nějaký útočník nebude zajímat. Cílem útoků nejsou pouze populární a velké systémy – řada útočníků se prostě snaží napadnout co největší počet systémů bez ohledu na jejich velikost. Navíc mohou bezpečnostní díru ve vašem systému použít pro přístup k dalším systémům. Útočník má dostatek času a může si dovolit hádat, jak jste systém zabezpečili, prostým zkoušením všech možností. Existuje i řada jiných důvodů, proč může být váš systém cílem útoku, budeme o nich hovořit později.

Zabezpečení počítače

Jde o oblast zabezpečení, na kterou se správce většinou soustřeďuje. Typicky to obnáší zajistit bezpečnost svého systému a předpokládat, že ostatní dělají totéž. Mezi činnosti, za které je správce systému zodpovědný, patří volba dobrých hesel, zabezpečení síťových služeb, údržba záznamů o činnosti počítače a aktualizace programů se známými bezpečnostními chybami. I když jde o naprosto nezbytné činnosti, mohou být velmi únavné, jakmile se síť rozroste na více než jen několik počítačů.

Zabezpečení sítě

Bezpečnost sítě je stejně nutná jako bezpečnosti počítače. Pokud máte v síti stovky nebo tisíce počítačů, těžko můžete očekávat, že každý jeden bude dostatečně zabezpečen. Zajistíte-li přístup pouze autorizovaným uživatelům, nainstalujete firewally, použijete silné šifrování a zajistíte, že v síti nejsou žádné nezabezpečené počítače, zvyšujete bezpečnost celé sítě.

V tomto dokumentu popisujeme některé techniky k zabezpečení sítě a snad vám ukážeme některé způsoby, jak útočníkům zabránit v přístupu k tomu, co se snažíte chránit.

Zabezpečení utajením

Jedním ze způsobů zabezpečení, o kterém se musíme zmínit, je „zabezpečení utajením“. Znamená to například, že službu se známou bezpečnostní chybou spustíme na nějakém nestandardním portu, a budeme doufat, že útočník si její existence nevšimne a nevyužije ji. Buďte si jisti, že útočníci budou schopni službu najít a využít. Zabezpečení utajením nepředstavuje žádné zabezpečení. Jen proto, že se staráte o malou nebo neznámou síť, neznamená to, že se vám útoky vyhnou. V dalších částech si popíšeme, co se snažíme chránit.

Organizace tohoto dokumentu

Tento dokument je rozdělen do více částí. Popisují některé běžné problémy z oblasti zabezpečení. V první části hovoříme o tom, jak počítač chránit před fyzickým nebezpečím. Ve druhé hovoříme o zabezpečení před autorizovanými uživateli. Ve třetí popisujeme, jak nastavit souborové systémy a přístupová práva k souborům. V další, čtvrté, části hovoříme o tom, jak pomocí šifrování zlepšit bezpečnost počítače a sítě. V páté části si řekneme, které parametry jádra mohou ovlivnit bezpečnost. V šesté části hovoříme o zabezpečení před útoky po síti. V sedmé části popisujeme, jak počítač připravit před připojením k síti. V osmé části uvádíme co dělat, jakmile detekujete probíhající nebo provedený útok. V deváté části uvádíme přehled důležitých informačních zdrojů. Desátá část je souhrn obvyklých otázek a odpovědí a konečně jedenáctá část představuje závěrečné shrnutí.

Dvě hlavní zásady, které byste si měli z tohoto dokumentu odnést, jsou:

- Hlídejte svůj systém. Sledujte logy, jako například */var/log/messages*.
- Udržujte systém aktuální, instalujte nové verze programů, a sledujte bezpečnostní upozornění. Už jenom tímto výrazně zvýšíte bezpečnost systému.

Fyzická bezpečnost

První úroveň zabezpečení systému je jeho fyzická bezpečnost. Kdo má k počítači přímý fyzický přístup? A má jej mít? Jste schopni počítač ochránit před jejich zásahy?

Míra fyzického zabezpečení je silně závislá na typu systému a/nebo na prostředcích, které jsou k dispozici.

Pokud jde o domácí počítač, pravděpodobně jej nepotřebujete příliš chránit (i když by měl být chráněn před zásahy dětí a otravných příbuzných). Pokud jde o počítač například v učebně, musíte jej zabezpečit lépe, nicméně stále musíte nechat uživatelům možnost jej použít. U počítače v kanceláři můžete uvažovat o jeho zabezpečení v mimopracovní době, nebo jste-li pryč. U některých společností může být opuštění a nezabezpečení terminálu důvodem k výpovědi.

Zjevná opatření pro fyzické zabezpečení jsou zámky na dveřích a kabelech, zamykatelné skříně, instalace kamerového systému a podobně. To už je ale mimo záběr tohoto dokumentu.

Zámky počítačů

Většina moderních počítačů umožňuje nějakou míru uzamčení. Obvykle je na čelním panelu zámek, kterým můžete zamknout skříň počítače a nikdo tak nebude moci přímo manipulovat s hardwarem počítače. V některých případech se tak dá zabránit nabootování počítače z vlastní diskety nebo disku.

Zámky na skříních podle své konstrukce mohou plnit různé funkce. V některých případech pouze neumožňují otevřít skříň počítače bez násilí. V jiných případech mohou zabránit dokonce i v připojení vlastní klávesnice nebo myši. Podrobnosti naleznete v manuálu k počítači. V některých případech mohou být tyto zámky velmi užitečné, i když jejich kvalita je mnohdy mizerná a k jejich „odemčení“ stačí šroubovák.

Některé počítače (hlavně SPARC a MAC) mají zezadu skříně očko, které je možné použít k zaplombování nebo zamčení skříně.

Bezpečnost BIOSu

BIOS je nízkoúrovňový software, který slouží ke konfiguraci hardwarových prvků počítačů na platformě x86. Služby BIOSu využívá LILO i jiné zavaděče Linuxu ke zjištění, jak systém zavést. I na jiných platformách, na nichž Linux běží, existuje podobná vrstva (Open Firmware na Macu a nových Sunech, bootovací PROM na Sunech a podobně). Pomocí služeb BIOSu můžete útočníkovi zamezit v restartu počítače a manipulaci se systémem.

Řada BIOSů umožňuje nastavit heslo pro spuštění počítače. To sice moc neřeší (BIOS je možné vymazat nebo změnit, pokud má útočník přístup do skříně počítače), může však fungovat jako dobré odstrašující opatření (zpomaluje to a zanechává stopy po napadení). Podobně i na počítačích platformy Sparc je možné nastavit EEPROM tak, aby při spuštění systému bylo vyžadováno heslo. To může útočníka zpomalit.

Dalším rizikem při použití hesla do BIOSu jsou implicitní hesla. Většina výrobců BIOSů nepředpokládá, že uživatel v případě zapomenutí hesla bude otevírat počítač a odpojovat baterii, a proto jejich BIOSy obsahují implicitní hesla, která fungují vždy. Mezi známá implicitní hesla patří:

j262 AWARD_SW AWARD_PW lkwpeter Biostar AMI Award bios BIOS setup cmos AMI!SW1 AMI?SW1 password hewittrand shift + s y x z

Zkoušel jsem Award BIOS a fungovalo heslo AWARD_PW. Tato hesla je možné snadno zjistit na stránkách výrobců BIOSu nebo na adrese http://astalavista.box.sk. Proto je nelze považovat za ochranu před znalým útočníkem.

Řada BIOSů umožňuje nastavit i další rozumná bezpečnostní opatření. Podívejte se do manuálu k počítači, nebo přímo do BIOSu při startu počítače. Existují například možnosti zakázat bootování počítače z diskety, nebo ochrana některých funkcí heslem.

Poznámka: Pokud nastavíte heslo do BIOSu na serveru, nebudete jej schopni spustit bez fyzického zásahu. Budete k němu muset zajet a zadat heslo například i po výpadku proudu.

Zabezpečení zavaděče systému

Většina zavaděčů Linuxu umožňuje také zadat heslo. LILO například obsahuje volby *password* a *restricted*. Volba *password* vyžaduje zadání hesla při každém spuštění systému, volba *restricted* pouze v případě spouštění s parametry (například *single*).

Uvádíme výtah z manuálové stránky *lilo.conf*:

Globální parametry

password = heslo

Stejně jako volba *password* jednoho obrazu (viz níže), platí pro všechny spouštěné obrazy.

restricted

Stejně jako volba *restricted* jednoho obrazu (viz níže), platí pro všechny spouštěné obrazy.

Parametry obrazů

password = heslo

Spuštění daného obrazu je chráněno heslem.

restricted

Heslo se vyžaduje pouze při pokusu spustit obraz s explicitně zadanými parametry (např. single).

Při nastavování všech těchto hesel nezapomínejte, že si je budete muset pamatovat :-). Rovněž nezapomínejte, že tato hesla mohou zkušeného útočníka pouze zpomalit. Nezabrání mu totiž v nabootování systému z diskety a následném připojení disku. Pokud chcete zavaděč chránit heslem, měli byste zakázat bootování z diskety a chránit heslem i BIOS.

Nezapomeňte, že soubor */etc/lilo.conf* musí mít nastavena práva „600“ (tedy čitelný pouze pro superuživatele), jinak z něj kdokoliv bude moci hesla přečíst. Uvítáme informace o zabezpečení jiných zavaděčů (grub, silo, milo, linload a podobně).

Poznámka: Pokud nastavíte heslo zavaděče na serveru, nebudete jej schopni spustit bez fyzického zásahu. Budete k němu muset zajet a zadat heslo například i po výpadku proudu.

xlock a vlock

Pokud se občas od počítače na chvíli vzdálíte, budete možná chtít „zamknout“ konzolu, aby nikdo nemohl zasahovat do vaší práce, nebo si ji prohlížet. K tomuto účelu můžete použít dva programy: *xlock* a *vlock*.

xlock zamyká grafickou konzolu a měl by být součástí každé distribuce s podporou systému X Window. Podrobnosti o tomto programu zjistíte na jeho manuálové stránce, obecně jej však můžete spustit z kteréhokoliv grafického terminálu, a program zamkne displej a k jeho odemknutí je nutné zadat heslo.

vlock je jednoduchý malý program, který umožňuje zamknout některé nebo všechny virtuální konzoly. Můžete zamknout buď tu, na níž právě pracujete, nebo všechny. Pokud zamknete jen jednu konzolu, kdokoliv bude moci k počítači přijít a použít jej, nebude však mít přístup k vaší virtuální konzole.

Zamknutím konzoly sice zabráníte tomu, aby někdo manipuloval s vaší prací, nezabráníte mu však počítač restartovat. Nezabráníte tím také v přístupu k počítači po síti.

Důležité také je, že tím nezabráníte nikomu v přepnutí se z grafického prostředí a v práci na virtuální textové konzole, nebo na konzole, z níž byl systém X Window spuštěn – pak je může zastavit a získat vaše práva. Z toho důvodu byste měli zamykání grafické konzoly používat pouze v případě, že je pod správou *xdm*.

Bezpečnost lokálních zařízení

Pokud máte k počítači připojenu webkameru nebo mikrofon, měli byste zvážit, zda neexistuje nebezpečí, že útočník k těmto zařízením získá přístup. Pokud je zrovna nepoužíváte, je nejrozumnější je odpojit. V opačném případě byste měli pozorně prověřit programy, které umožňují k těmto zařízením přístup.

Detekce fyzického narušení počítače

První věc, které byste si měli vždy všimnout, je reboot počítače. Protože Linux je robustní a stabilní operační systém, mělo by k jeho restartu docházet pouze v případě, že jste prováděli aktualizaci systému, hardwaru a podobně. Pokud došlo k restartu počítače bez vašeho zásahu, může to signalizovat pokus o jeho napadení. Řada metod napadení počítače vyžaduje jeho restart nebo vypnutí.

Hledejte známky zásahu do skříně počítače. I když řada útočníků maže záznamy o své činnosti ze systémových logů, je dobré je zkontrolovat a hledat něco podezřelého.

Rozumné je rovněž ukládat logovací soubory na bezpečném místě, například na vyhrazeném logovacím serveru na chráněné síti. Jakmile dojde k narušení počítače, jsou mnohdy k ničemu i logovací záznamy, protože byly s velkou pravděpodobností také změněny.

Démon *syslog* je možné nakonfigurovat tak, aby automaticky odesílal data na centrální logovací server. Data však při přenosu nejsou šifrována a případný útočník by je mohl vidět a dozvědět se tak citlivá data. Existují verze démona, které umožňují data při přenosu šifrovat.

Nezapomínejte také, že záznamy v logu je možné snadno zfalšovat. Syslog dokonce přijme po síti data tvářící se jako z lokálního systému, aniž by to dal najevo.

Některé věci, na něž byste si měli dávat pozor:

- Krátké a neúplné záznamy
- Záznamy s divným časem
- Záznamy s nesprávnými právy nebo vlastnictvím
- Záznamy o restartech služeb
- Chybějící záznamy
- Přihlášení a *su* z neobvyklých míst

O datech v logovacích souborech budeme mluvit v části 9.5 tohoto dokumentu.

Lokální bezpečnost

Další věcí, kterou je nutné k zabezpečení systému sledovat, je zabezpečení před útoky lokálních uživatelů. Že jsme řekli *lokální* uživatelé? Ano!

Získání přístupu k lokálnímu uživatelskému účtu je první věc, kterou obvykle útočník dělá, aby získal práva superuživatele. Při nedostatečném lokálním zabezpečení je možné normální uživatelský účet „povýšit" na superuživatele s využitím různých chyb a nevhodně nastavených služeb. Pokud bude systém dobře lokálně zabezpečen, bude mít útočník obtížnější pozici.

Dokonce i lokální uživatel může v systému napáchat dost škody, dokonce i když je opravdu tím, za koho se vydává. Vytvářet účty lidem, které neznáte, nebo u nichž nemáte kontaktní informace, není příliš rozumné.

Vytváření nových účtů

Měli byste zajistit, že uživatelské účty poskytují pouze ta minimální oprávnění, která uživatelé ke své práci potřebují. Pokud pro svého desetiletého syna vytvoříte účet, budete chtít pouze aby mohl psát a malovat, ale nemohl mazat cizí data.

Několik základních pravidel, pokud umožňujete přístup ke svému systému i dalším uživatelům:

- Poskytněte jim minimální potřebná oprávnění.
- Sledujte kdy a odkud se přihlašují, nebo by se měli přihlašovat.
- Nezapomínejte mazat nepoužívané uživatelské účty, což můžete zjistit příkazem last nebo ze systémových logů.
- Doporučuje se na všech počítačích v síti používat stejná uživatelská jména, protože se tím zjednodušuje správa účtů a analýza logů.
- Absolutně se nedoporučuje vytváření skupinových uživatelských účtů. Individuální účty je možné snáze monitorovat, což u skupinových nelze.

Účty používané k útokům jsou často dlouhodobě nepoužívané. Protože je nikdo nepoužívá, jsou optimální jako nástroje k útoku.

Bezpečnost superuživatele

Nejdůležitějším uživatelským účtem je superuživatelský účet. Tento uživatel má plnou vládu nad počítačem a případně dalšími počítači v síti. Superuživatelský účet byste měli vždy používat jen krátce, pro konkrétní úkony, a jinak pracovat jako běžný uživatel. Dokonce i drobná chyba provedená superuživatelem může mít vážné důsledky. Čím méně superuživatelský účet používáte, tím jste bezpečnější. Uveďme si několik triků, jak coby superuživatel nezlikvidovat vlastní počítač:

- Při provádění složitějších operací je zkuste nejprve provést nedestruktivním způsobem. Například pokud chcete provést rm *foo*.bak*, zkuste nejprve *ls foo*.bak*, abyste viděli, zda mažete skutečně ty soubory, které chcete. Vhodné je také namísto destruktivních příkazů použít příkaz echo.
- Implicitně používejte alias k příkazu *rm*, který bude žádat potvrzení při mazání souborů.
- Jako superuživatel provádějte pouze jednotlivé konkrétní úkony. Pokud se přistihnete, že něco zkoumáte a zkoušíte, vraťte se do normálního režimu do doby, než budete přesně vědět, co jako superuživatel udělat.
- Nesmírně důležitá je příkazová cesta superuživatele. Příkazová cesta (tedy hodnota proměnné PATH) udává, ve kterých adresářích má příkazový interpret hledat příkazy. Pro su-

peruživatele by měla být cesta co nejkratší a nikdy by neměla obsahovat . (tedy aktuální adresář). Kromě toho by cesta neměla obsahovat adresáře, do nichž je možné volně zapisovat, protože tím by mohl případný útočník modifikovat soubory, které superuživatel spouští.

- Jako superuživatel nikdy nepoužívejte nástroje *rlogin/rsh/rexec*. Jsou cílem celé řady útoků a pro superuživatele jsou obzvlášť nebezpečné. Superuživatel by neměl mít vytvořen soubor *.rhosts*.
- Soubor */etc/securitty* obsahuje seznam terminálů, z nichž se může superuživatel přihlásit. Implicitně (RedHat Linux) je povoleno přihlášení pouze z lokálních virtuálních terminálů. Při rozšiřování tohoto souboru buďte velmi ostražití. Vždy byste měli být schopni se vzdáleně přihlásit jako normální uživatel a pak teprve přejít do režimu superuživatele (nejlépe s použitím nějaké bezpečné metody přihlášení, viz část 6.4). Není tedy důvod povolovat přímé přihlášení superuživatele.
- Jako superuživatel postupujte vždy pomalu a rozvážně. Vaše akce mohou ovlivnit řadu věcí. Přemýšlejte, než něco napíšete!

Pokud absolutně nezbytně potřebujete někomu poskytnout přístup superuživatele k vašemu systému, existuje několik nástrojů, které vám mohou pomoci. Příkaz *sudo* umožňuje běžným uživatelům spouštět některé příkazy, používané superuživatelem. Můžete tak například připojovat a odpojovat vyjímatelná média, ale už nic víc. Kromě toho příkaz *sudo* eviduje veškerá svá úspěšná i neúspěšná použití, takže vidíte, co který uživatel dělal. Z těchto důvodů se dá příkaz *sudo* použít i tam, kde má superuživatelská práva více lidí, neboť umožňuje sledovat provedené zásahy.

Přestože příkaz *sudo* umožňuje poskytnout konkrétním uživatelům konkrétní práva ke konkrétním operacím, má svá nebezpečí. Měl by se používat pouze pro omezený seznam úkonů, jako jsou například restart serveru nebo vytvoření nového uživatele. Programy, které umožňují vstup do příkazového interpretu, dávají přístup k superuživatelskému účtu každému, kdo je přes *sudo* spustí. Typicky jde o většinu editorů. Dokonce i tak nevinné programy jako například *cat* lze použít k přepsání souborů a získat tak privilegia superuživatele. Považujte program *sudo* spíše za nástroj pro sledování provedených operací, neočekávejte od něj, že nahradí superuživatelský účet a stále bude bezpečný.

Bezpečnost souborů a souborového systému

Několik minut příprav a plánování před aktivací systému vám může pomoci ochránit systém a data.

- Neměl by být důvod umožňovat z uživatelských adresářů SUID/SGID programy. Oddíly, na něž může zapisovat i jiný uživatel než root, by měly být v souboru */etc/fstab* připojeny s volbou *nosuid*. Na oddílech s uživatelskými daty můžete také použít volby *nodev* a *noexec*, stejně jako na svazku */var*, čímž zabráníte spouštění programů a vytváření znakových a blokových zařízení, které by stejně neměly být k ničemu zapotřebí.
- Pokud nabízíte souborové systémy pomocí NFS, nastavte */etc/exports* tím nejstriktnějším možným způsobem. Zakažte použití zástupných znaků, zápis jako superuživatel, a je-li to možné, vždy nabídněte přístup pouze pro čtení.
- Nastavte *umask* uživatelů co nejpřísněji, více viz část 5.1.
- Pokud připojujete síťové souborové systémy jako je NFS, nastavte */etc/exports* co nejpřísněji. Vhodné jsou typicky volby *nodev*, *nosuid* a případně *noexec*.

- Nastavte limity souborového systému, nepoužívejte implicitní nastavení bez limitů. Uživatelské limity můžete nastavovat pomocí modulu PAM pro řízení prostředků a souboru */etc/pam.d/limits.conf*. Například limity pro skupinu uživatelů mohou vypadat takto:

```
@users      hard  core    0
@users      hard  nproc   50
@users      hard  rss     5000
```

  Tímto se zakazuje vytvoření výpisu *core*, omezuje se počet procesů na 50 a omezuje se využití paměti na 5 MB.

  Stejná omezení můžete nastavit v konfiguračním souboru */etc/login.defs*.

- Soubory */var/log/wtmp* a */var/run/utmp* obsahují záznamy o přihlášení všech uživatelů. Je nutné udržovat jejich integritu, protože z nich lze zjistit, kdy a odkud se uživatel (nebo potenciální útočník) k systému přihlásil. Tyto soubory by měly mít nastavena práva 644.
- Pomocí příznaku immutable je možné chránit před neúmyslným přepsáním nebo smazáním důležité soubory. Zabraňuje zároveň ve vytvoření pevného odkazu na souboru. Podrobnosti o tomto příznaku naleznete na manuálové stránce příkazu *chattr*.
- SUID a SGID soubory představují potenciální bezpečnostní riziko a je třeba je pečlivě sledovat. Protože tyto programy poskytují speciální práva uživateli, který je spustí, je nutné zajistit, aby takto nebyly instalovány nebezpečné programy. Oblíbený trik útočníků je využít programy SUID-root a ponechat si je jako zadní vrátka do systému pro dobu, kdy bude původní díra odstraněna.

  Zjistěte si všechny SUID a SGID soubory v systému a pravidelně sledujte, co se s nimi děje, abyste zaznamenali změny provedené případným útočníkem. Všechny SUID/SGID programy naleznete příkazem:

```
root#  find / -type f \( -perm -04000 -o -perm -02000 \)
```

  Distribuce Debian spouštějí každou noc úlohu, která zjišťuje všechny existující SUID soubory a porovnává je se soubory nalezenými minulou noc. Záznamy o zjištěných výsledcích naleznete ve */var/log/setuid**.

  U podezřelých programů můžete příkazem *chmod* odstranit příznak SUID/SGID, a obnovit jej pouze v případě nutnosti.

- Další bezpečnostní díru mohou představovat soubory s právem zápisu, zejména systémové soubory, pokud útočník získá přístup do systému a provede jejich modifikaci. Nebezpečné jsou i zapisovatelné adresáře, protože útočník může podle libosti vytvářet a mazat soubory. Všechny soubory s globálním právem zápisu najdete příkazem:

```
root# find / -perm -2 ! -type l -ls
```

  Vždy byste měli přesně vědět, proč zrovna tyto soubory mají právo zápisu. Za normálních okolností jsou některé soubory globálně zapisovatelné, například některé soubory v */dev* a některé symbolické odkazy, proto parametr *! –type l*, kterým jsme je z hledání vyloučili.

- Dalším příznakem přítomnosti útočníka v systému mohou být soubory bez vlastníka. Soubory, které nemají vlastníka nebo jež nepatří žádné skupině, najdete příkazem:

```
root# find / \( -nouser -o -nogroup \) -print
```

- Pravidelně byste měli vyhledávat soubory *.rhosts* a nepovolovat jejich existenci. Nezapomínejte, že útočníkovi pak stačí získat přístup k jedinému účtu a může mít přístup k celé síti. Soubory *.rhosts* naleznete příkazem:

```
root# find /home -name .rhosts -print
```

- A konečně, než změníte přístupová práva jakéhokoliv systémového souboru, ujistěte se, že přesně rozumíte tomu, co děláte. Nikdy neměňte práva souborů jen proto, že je to jednoduchá náprava toho, že něco nefunguje. Před změnou práv si vždy zjistěte, proč soubor zrovna takováto práva má.

Nastavení hodnoty *umask*

Příkazem *umask* můžete zjistit implicitní práva vytvářených souborů. Jedná se o osmičkový doplněk požadovaných práv. Pokud se soubory vytvářejí bez ohledu na přístupová práva, může uživatel omylem poskytnout právo ke čtení nebo zápisu někomu, komu nechtěl. Typická nastavení hodnoty *umask* jsou 022, 027 a 077 (které je nejpřísnější). Normálně se tato hodnota nastavuje v */etc/profile* a platí tak pro všechny uživatele systému. Masku pro vytváření souborů lze vypočítat odečtením požadovaných práv od hodnoty 777. Jinak řečeno, hodnota masky 777 způsobí, že nově vytvořené soubory nebudou mít nastavena práva čtení, zápisu a spuštění vůbec pro nikoho. Maska 666 způsobí, že nové soubory budou vytvářeny s právy 111. Můžete provést nastavení například:

```
# Nastavení implicitní hodnoty umask
umask 033
```

Uživatel *root* by měl mít nastavenu masku 077, což vyřadí práva čtení, zápisu a spuštění všem ostatním uživatelům, pokud nebudou práva explicitně změněna příkazem *chmod*. V tomto případě budou mít nově vytvářené adresáře práva 744, získaná odečtením hodnoty 033 od 777. Nově vytvořené soubory s maskou 033 budou mít práva 644.

Pokud používáte RedHat a jejich mechanismus vytváření uživatelů a skupin, stačí použít hodnotu masky 002. Je to dáno tím, že standardně je v každé skupině pouze jeden uživatel.

Práva souborů

Je důležité zajistit, že systémové soubory nemohou otevřít a upravovat uživatelé a skupiny, kteří nemají správu systému provádět.

Unix řídí přístupová práva k souborům a adresářům pro tři kategorie: vlastníka, skupinu a ostatní. Vždy existuje právě jeden vlastník, libovolný počet členů skupiny a pak všichni ostatní.

Rychlé vysvětlení přístupových práv v Unixu:

Vlastnictví – který uživatel a skupina mají právo nastavovat přístupová práva i-uzlu a rodičovského i-uzlu.

Oprávnění – bitové příznaky, které je možno nastavovat a rušit, čímž se umožňují určité typy přístupu. Oprávnění pro adresáře mohou mít jiný význam, než stejná oprávnění pro soubory.

Čtení:

Právo prohlížet si obsah souboru.

Právo číst adresář.

Zápis:

Právo přidat nebo změnit soubor.

Právo mazat nebo přesouvat soubory v adresáři.

Spuštění:

Právo spustit binární soubor nebo skript.

Právo prohledávat adresář, v kombinaci s právem čtení.

Příznak Save Text (pro adresáře)

Takzvaný „sticky bit“ má jiný význam pro adresáře a jiný pro soubory. Je-li nastaven u adresáře, může uživatel mazat pouze soubory, jichž je vlastníkem nebo k nimž má explicitně právo zápisu, a to i v případě, že má právo zápisu do adresáře. Tento příznak je určen pro adresáře jako */tmp*, do nějž sice mohou zapisovat všichni uživatelé, ale není žádoucí, aby kdokoliv mohl cokoliv mazat. Nastavení tohoto příznaku se ve výpisu adresáře indikuje symbolem *t*.

Příznak SUID (pro soubory)

Tento příznak umožňuje souboru provést operaci *set-user-id*. Jestliže je u souboru nastaven příznak SUID a soubor je spustitelný, pak spuštěné procesy budou mít práva vlastníka souboru a nikoliv uživatele, který proces spustil. Toto nastavení je zodpovědné za řadu průniků založených na chybách typu přetečení bufferu.

Příznak SGID (pro soubory)

Tento příznak povoluje operaci *set-group-id*. Jde o podobné chování jako u příznaku SUID, v tomto případě se však nastavení týká skupiny, pod níž proces běží. Aby měl příznak efekt, musí být soubor spustitelný.

Příznak SGID (pro adresáře)

Jestliže je příznak SGID nastaven pro adresář (příkazem *chmod g+s adresář*), soubory vytvářené v adresáři budou mít skupinu stejnou, jako je skupina adresáře.

Vy – vlastník souboru.

Skupina – skupina, do níž patříte.

Kdokoliv – kdokoliv v systému, kdo není vlastníkem ani členem skupiny.

Příklad souboru:

```
-rw-r--r--  1 kevin  users         114 Aug 28  1997 .zlogin
1. bit - adresář?             (ne)
 2. bit - čtení vlastníkem?    (ano, kevinem)
  3. bit - zápis vlastníkem?    (ano, kevinem)
   4. bit - spuštění vlastníkem? (ne)
    5. bit - čtení skupinou?      (ano, users)
     6. bit - zápis skupinou?      (ne)
      7. bit - spuštění skupinou?   (ne)
       8. bit - čtení kýmkoliv?      (ano)
        9. bit - zápis kýmkoliv?      (ne)
         10. bit - spuštění kýmkoliv?  (ne)
```

Následující řádky představují příklady minimálních přístupových práv, která jsou potřebná k provedení popsané akce. Můžete samozřejmě nastavit větší oprávnění, tento přehled ukazuje, jaká mají jednotlivá oprávnění efekt:

`-r--------` Umožňuje čtení vlastníkem .

`--w-------` Umožňuje vlastníkovi soubor změnit nebo smazat. (Kdokoliv, kdo má právo zápisu pro adresář, v němž se soubor nachází, jej může přepsat a tedy smazat.)

`---x------` Vlastník může soubor spustit, ne však, jde-li o skript, u nějž je nutné i právo čtení.

`---s------` Soubor bude spuštěn s efektivním uživatelským ID odpovídajícím vlastníkovi.

`--------s-` Soubor bude spuštěn s efektivním skupinovým ID odpovídajícím skupině.

`-rw------T` Neprovádět aktualizaci posledního času modifikace. Používá se typicky pro swapovací soubory.

`---t------` Nemá význam (dříve tzv. sticky bit).

Příklad adresáře:

```
drwxr-xr-x  3 kevin  users         512 Sep 19 13:47 .public_html/
1. bit - adresář?             (ano, obsahuje spoustu souborů)
 2. bit - čtení vlastníkem?    (ano, kevinem)
  3. bit - zápis vlastníkem?    (ano, kevinem)
   4. bit - prohledávání vlastníkem? (ano, kevinem)
    5. bit - čtení skupinou?      (ano, users)
     6. bit - zápis skupinou?      (ne)
      7. bit - prohledávání skupinou?   (ano, users)
       8. bit - čtení kýmkoliv?      (ano, kýmkoliv)
        9. bit - zápis kýmkoliv?      (ne)
         10. bit - prohledávání kýmkoliv?  (ano, kýmkoliv)
```

Následující řádky představují příklady minimálních přístupových práv, která jsou potřebná k provedení popsané akce. Můžete samozřejmě nastavit větší oprávnění, tento přehled ukazuje, jaká mají jednotlivá oprávnění efekt:

`dr--------` Obsah je možné vypsat, ale nelze číst atributy souborů.

`d--x------` Do adresáře je možno vstoupit a lze jej použít ve spouštěcí cestě.

`dr-x------` Vlastník může číst atributy souborů.

`d-wx------` Soubory je možné vytvářet/mazat, i když adresář není nastaven jako aktuální.

`d------x-t` Zabraňuje ve smazání souborů někým jiným než vlastníkem. Používá se u adresáře */tmp*.

`d---s--s--` Bez efektu.

Systémové konfigurační soubory (obvykle v adresáři */etc*) mají typicky práva 640 (`-rw-r-----`) a vlastní je *root*. V závislosti na požadované míře zabezpečení to můžete změnit. Žádné systémové soubory by neměly být zapisovatelné skupinou a kýmkoliv. Některé konfigurační soubory, například */etc/shadow*, může číst jenom *root*. Adresáře v */etc* by minimálně neměly být přístupné komukoliv.

SUID skripty

SUID skripty představují závažné bezpečnostní riziko, a proto jádro v tomto případě dané nastavení nerespektuje. Bez ohledu na to, za jak bezpečný skript považujete, útočník jej může využít k získání superuživatelského příkazového interpretu.

Kontrola integrity

Další vhodný způsob detekce lokálních i síťových útoků je použití nějakého systému pro kontrolu integrity, jako je například Tripwire, Aide nebo Osiris. Tyto programy vypočtou řadu kontrolních součtů důležitých binárních a konfiguračních souborů a kontrolují je proti databázi původních, správných hodnot. Tím se prozradí jakékoliv změny v souborech.

Rozumné je tento typ programů nainstalovat na disketu a pak na disketě fyzicky zakázat zápis. Tímto způsobem zamezíte útočníkovi modifikovat samotný kontrolní program nebo jeho databázi. Jakmile něco podobného nastavíte, je rozumné kontrolu pravidelně spouštět, abyste poznali, zda nedošlo ke změně.

Můžete například přidat záznam do tabulky démona cron, kterým spustíte kontrolu každou noc a výsledek si necháte odeslat e-mailem. Nastavení typu

```
# nastavení adresáta
MAILTO=kevin
# spuštění Tripwire
15 05 * * * root /usr/local/adm/tcheck/tripwire
```

provede kontrolu každý den v 5:15 ráno.

Kontrola integrity je vynikající nástroj pro detekci útoků dříve, než byste si jich všimli jinak. Na normálním systému se ovšem řada souborů mění, takže musíte pečlivě dávat pozor na to, co jsou normální změny a co může být projevem útoku.

Program Tripwire můžete zdarma získat na adrese http://www.tripwire.org. Můžete si doplatit za manuály a podporu.

Aide najdete na adrese http://www.cs.tut.fi/~rammer/aide.html.

Osiris najdete na adrese http://www.shmoo.com/osiris/.

Trójské koně

Trójské koně jsou pojmenovány po bájné léčce z Homérovy Iliady. Myšlenka je taková, že útočník distribuuje program, který vypadá skvěle, a přesvědčuje uživatele, aby si program nahráli a spustili jako root. Pak program nepozorovaně naruší bezpečnost systému. Zatímco si tedy uživatelé myslí, že program dělá nějakou jednu věc (kterou může klidně dělat velmi dobře), představuje současně i útok na systém.

Měli byste si dávat pozor na to, jaké programy na systém instalujete. RedHat používá ve svých souborech RPM kontrolní součty MD5 a GPG podpisy, takže si můžete ověřit, zda opravdu instalujete originál. Ostatní distribuce používají podobné metody. Nikdy nespouštějte žádné neznámé binární programy, od nichž nemáte zdrojové kódy. Útočníci obvykle nedistribuují své programy společně se zdrojovými kódy, aby je mohl kdokoliv prozkoumat.

I když to může být komplikované, vždy si ověřte, zda zdrojový kód programu opravdu pochází z distribučního serveru programu. Pokud budete program spouštět jako superuživatel, vždy si prohlédněte a zkontrolujte jeho zdrojový kód, ať už sami, nebo požádejte někoho důvěryhodného.

Hesla a šifrování

Jedním z nejvýznamnějších prvků k zajištění bezpečnosti jsou dnes hesla. Pro vás i vaše uživatele je důležité používat bezpečná, neuhodnutelná hesla. Většina moderních linuxových distribucí obsahuje program *passwd*, který vám nedovolí nastavit snadno uhodnutelné heslo. Zkontrolujte, zda program *passwd* ve vašem systému je aktuální a obsahuje tuto funkci.

Hlubší debata o šifrování je mimo záběr tohoto dokumentu, nicméně jistý úvod je na místě. Šifrování je velice užitečné, dnes možná nezbytné. Existuje velké množství šifrovacích metod, každá má své význačné vlastnosti.

Většina Unixů (Linux nevyjímaje) primárně používá k uložení hesel jednosměrné šifrování, založené na algoritmu DES (Data Encryption Standard). Takto zašifrovaná hesla se pak typicky ukládají v souboru */etc/passwd*, nebo (méně typicky) v souboru */etc/shadow*. Když se přihlašujete, provede se nové zašifrování zadaného hesla a to se porovná s údajem v souboru, kde jsou hesla uložena. Pokud se výsledky shodují, musí jít o stejné heslo a povolí se přihlášení do systému. I když DES je sám o sobě obousměrný algoritmus (umožňuje data zašifrovat a pak i odšifrovat), modifi-

kace používaná ve většině Unixů je jednosměrná. Znamená to, že by nemělo být možné obrátit směr šifrování a ze souboru */etc/passwd* (nebo */etc/shadow*) zjistit původní podobu hesla.

Útoky hrubou silou (například programy Crack nebo John the Ripper, viz část 6.9) často umožní uhodnout heslo, pokud nebylo zvoleno dostatečně náhodně. Moduly PAM (viz níže) umožňují šifrovat hesla jinými algoritmy (například MD5 a podobně). I zde můžete použít program Crack.

Doporučujeme tento program pravidelně spouštět proti vaší vlastní databázi hesel a hledat tak slabá hesla. Pak kontaktujte příslušného uživatele a doporučte mu změnu hesla.

Návod na volbu dobrých hesel naleznete na adrese http://consult.cern.ch/writeup/security/security_3.html.

PGP a kryptografie s veřejným klíčem

Kryptografie s veřejným klíčem, například algoritmy používané programem PGP, používá jeden klíč pro zašifrování a jiný pro rozšifrování. Klasická kryptografie naproti tomu používá stejný klíč pro zašifrování i rozšifrování. Tento klíč musí znát obě komunikující strany a je tedy nutné nějakým bezpečným způsobem zajistit jeho přenos od jedné strany ke druhé.

Aby se vyloučila potřeba bezpečného přenosu šifrovacího klíče, používá kryptografie s veřejným klíčem dva klíče: veřejný klíč a privátní klíč. Veřejný klíč každého je všem k dispozici pro účely zašifrování, privátní klíč udržuje každý účastník v tajnosti a používá jej k rozšifrování zpráv zašifrovaných příslušným veřejným klíčem.

Klasická kryptografie i kryptografie s veřejným klíčem mají své výhody a o jejich vlastnostech se můžete dočíst v dokumentu *RSA Cryptography FAQ*, uvedeném na konci této části.

Algoritmus PGP (Pretty Good Privacy) je v Linuxu široce podporován. Verze 2.6.2 a 5.0 jsou považovány za správně fungující. Dobrý úvod do PGP a návod k jeho použití naleznete v dokumentu *PGP FAQ* na adrese http://www.pgp.com/service/export/faq/55faq.cgi.

Ujistěte se, že používáte verzi vhodnou pro vaši zemi. Vzhledem k exportním omezením americké vlády se nepovoluje elektronický přenos silných šifrovacích algoritmů mimo území USA.

Exportní pravidla v USA nyní určuje EAR (Export Administration Regulations), nikoliv ITAR.

Podrobný návod ke konfiguraci PGP v Linuxu naleznete na adrese http://mercury.chem.pitt.edu/~angel/LinuxFocus/English/November1997/article7.html.

Tento návod byl psán pro mezinárodní verzi PGP, lze jej však použít i pro americkou verzi. Pro některé z nejnovějších verzí Linuxu budete možná potřebovat některé doplňky, které najdete na adrese ftp://metalab.unc.edu/pub/Linux/apps/crypto.

Existuje projekt na reimplementaci PGP s otevřeným kódem. GnuPG představuje úplnou a zdarma dostupnou náhradu programu PGP. Protože nepoužívá algoritmy IDEA ani RSA, lze jej použít bez omezení. GnuPG odpovídá standardu OpenPGP. Další informace naleznete na stránkách *GNU Privacy Guard* na adrese http://www.gnupg.org/.

Řadu informací o kryptografii naleznete v dokumentu *RSA Cryptography FAQ*, který můžete najít na adrese http://www.rsa.com/rsalabs/newfaq/. Naleznete v něm informace o pojmech jako jsou „Diffie-Hellmanův algoritmus“, „kryptografie s veřejným klíčem“, „digitální certifikáty“ a podobně.

SSL, S-HTTP a S/MIME

Uživatelé se často ptají na rozdíly mezi různými zabezpečovacími a šifrovacími protokoly, a na to, jak je použít. I když nečtete text o šifrování, bude rozumné si jednotlivé protokoly stručně představit a ukázat, kde nalézt další informace.

- SSL, Secure Sockets Layer, je šifrovací metoda vyvinutá společností Netscape k zajištění bezpečné komunikace po Internetu. Podporuje několik různých šifrovacích protokolů a poskytuje autentikaci klientů a serverů. SSL funguje na transportní vrstvě, vytváří bezpečný šifrovaný datový kanál, a může tak transparentně šifrovat různé typy dat. Nejčastěji se s touto metodou potkáme při návštěvě zabezpečených stránek a prohlížení zabezpečených dokumentů, kde zajišťuje bezpečnou komunikaci mezi prohlížečem a serverem. Další informace můžete najít na adrese http://www.consensus.com/security/ssl-talk-faq.html. Informace o dalších bezpečných protokolech společnosti Netscape naleznete na adrese http://home.netscape.com/info/security-doc.html. Stojí také za zmínku, že protokol SSL lze použít k přenosu řady jiných běžných protokolů, k jejich bezpečnému „obalení". Viz http://www.quiltaholic.com/rickk/sslwrap/.
- S-HTTP je další protokol zajišťující bezpečnostní služby na Internetu. Byl navržen k zajištění utajení, autentikace, integrity a neodpiratelnosti[1], přičemž podporuje více mechanismů správy klíčů a šifrovacích algoritmů na základě dohody mezi účastníky každé transakce. S-HTTP je omezen na specifické programy, které jej implementují, a každou zprávu šifruje samostatně.
- S/MIME, Secure Multipurpose Internet Mail Extension, je šifrovací standard používaný k šifrování elektronické pošty a dalších typů zpráv na Internetu. Jedná se o otevřený standard vyvíjený společností RSA, takže se dá očekávat, že v brzké době bude jeho implementace i v Linuxu. Podrobnější informace o protokolu S/MIME můžete najít na adrese http://home.netscape.com/assist/security/smime/overview.html.

Implementace IPSEC v Linuxu

Kromě CIPE a dalších metod šifrování dat existuje v Linuxu i několik implementací IPSEC. IPSEC je standard navržený IETF, který vytváří kryptograficky zabezpečenou komunikaci na síťové vrstvě IP, a zajišťuje autentikaci, integritu, řízení přístupu a utajení. Informace o IPSEC naleznete na adrese http://www.ietf.org/html.charters/ipsec-charter.html. Tam najdete také odkazy na další protokoly zajišťující správu klíčů, poštovní konferenci o IPSEC a archivy.

Linuxová implementace *x-kernel*, vyvíjená na arizonské univerzitě, používá k implementaci síťových protokolů objektovou základnu nazvanou *x-kernel*. Naleznete ji na adrese http://www.cs.arizona.edu/xkernel/hpcc-blue/linux.html. Jednoduše řečeno, *x-kernel* je metoda předávání zpráv na úrovni jádra, což usnadňuje další implementaci.

Další zdarma dostupná linuxová implementace IPSEC je Linux FreeS/WAN IPSEC. Na webových stránkách tohoto projektu se dočtete, že: „Tato služba umožňuje vytvářet bezpečné tunely přes nezabezpečené sítě. Cokoliv přenášeného po nezajištěné síti je šifrováno branou IPSEC a dešifruje se branou na druhém konci tunelu. Výsledkem je virtuální privátní síť, VPN. Jedná se o síť, která je „privátní", přestože zahrnuje počítače a systémy na „neprivátním" Internetu."

Tento program naleznete na adrese http://www.xs4all.nl/~freeswan/ a v době vzniku tohoto textu se nachází ve verzi 1.0.

Stejně jako u jiných kryptografických produktů není distribuován jako součást jádra vzhledem k platným exportním omezením.

ssh (Secure Shell) a stelnet

ssh a stelnet jsou z rodiny programů, které umožňují připojení ke vzdáleným počítačům zašifrovaným kanálem.

---

[1] Pozn. překladatele: V tomto kontextu je autentifikace = jednoznačné prokázání totožnosti, integrita = nemožnost narušení (modifikace) přenášených dat třetí stranou, neodpiratelnost = nemožnost popřít zprávu, jejímž jsem autorem.

Openssh je rodina programů, které slouží jako bezpečná náhrada programů *rlogin*, *rsh* a *rcp*. K zašifrování komunikace mezi účastníky a k autentikaci uživatelů používá kryptografii s veřejným klíčem. Umožňuje bezpečné přihlášení ke vzdálenému počítači nebo kopírování dat mezi počítači s vyloučením útoku typu „man-in-the-middle" a s vyloučením falšování záznamů DNS. Zajišťuje rovněž kompresi dat a zabezpečenou X11 komunikaci.

V současnosti existuje několik implementací ssh. Původní komerční implementaci společnosti Data Fellows naleznete na adrese http://www.datafellows.com.

Vynikající implementace Openssh je založena na původní verzi ssh společnosti Data Fellows, a je úplně přepracována, takže neobsahuje žádné patentované nebo proprietární části. Je k dispozici zdarma pod licencí BSD. Naleznete ji na adrese http://www.openssh.com.

Dále existuje projekt úplné reimplementace ssh od začátku, pojmenovaný „psst…". Naleznete jej na adrese http://www.net.lut.ac.uk/psst/.

Ssh můžete použít k připojení k linuxovému serveru i ze stanic s Windows. Existuje několik zdarma dostupných implementací klienta pro Windows, například http://guardian.htu.tuwien.ac.at/therapy/ssh/, a také komerční implementace od Data Fellows na adrese http://www.datafellows.com.

SRP je další bezpečná implementace protokolů telnet a ftp. Z jejich webových stránek uvádíme: „Projekt SRP vyvíjí bezpečné internetové programy k použití zdarma. Počínaje plně zabezpečenými distribucemi protokolů Telnet a FTP doufáme, že nahradíme slabé síťové autentikační systémy silnými bez újmy na snadnosti použití. Bezpečnost by měla být standard, ne volba!"

Další informace najdete na adrese http://www-cs-students.stanford.edu/~tjw/srp/.

PAM – Pluggable Authentication Modules

Nové distribuce systémů RedHat a Debian se dodávají s unifikovaným autentikačním mechanismem, nazvaným „PAM". PAM dovoluje změnit používané autentikační metody a požadavky za běhu, a zapouzdření všech lokálních autentikačních metod bez nutnosti znovu překládat jednotlivé programy. Konfigurace mechanismu PAM je mimo rámec tohoto dokumentu. Podrobnosti se můžete dozvědět na webových stránkách PAM na adrese http://www.kernel.org/pub/linux/libs/pam/index.html.

Uveďme si jen několik věcí, které je možné pomocí PAM dosáhnout:

- Šifrování hesel jiným algoritmem než DES. (Čímž se komplikuje jejich odhalení útokem hrubou silou.)
- Nastavení limitů na jednotlivé prostředky, takže nelze provést útoky typu DoS. (Jde o limity na počty procesů, spotřebu paměti a podobně.)
- Jednoduchá aktivace stínových hesel (viz dále).
- Povolit jednotlivým uživatelům přihlášení jen v určitý čas a z určitého místa.

Po několika hodinách instalace a konfigurace můžete zabránit spouště útoků ještě dříve, než k nim dojde. Například následujícími řádky v souboru /etc/pam.d/rlogin můžete všem uživatelům zakázat použití souboru *.rhosts*:

```
# Zákaz rsh/rlogin/rexec
login auth required pam_rhosts_auth.so no_rhosts
```

Cryptographic IP Encapsulation (CIPE)

Primárním smyslem tohoto programu je poskytnout možnost vytvoření bezpečného (ve smyslu odolného proti odposlechu, analýze přenosu a podstrčení falešných dat) propojení sítí prostřednictvím nezabezpečených sítí, jako je Internet.

CIPE šifruje data na síťové úrovni. Pakety cestující mezi počítači v síti jsou šifrovány. Šifrovací modul je umístěn vedle ovladače, který odesílá a přijímá pakety.

To je rozdíl oproti SSH, které šifruje data nad spojením, na úrovni síťového soketu. Šifruje se logické spojení mezi programy běžícími na různých počítačích.

CIPE je možno použít při tunelování, k vytvoření virtuální privátní sítě. Šifrování na nízké úrovni má výhodu v tom, že může mezi dvěma počítači ve virtuální privátní síti fungovat transparentně, bez nutnosti zásahů do aplikačních programů.

Citujeme z dokumentace k CIPE:

> „Standard IPSEC definuje skupinu protokolů, které je možno použít (mimo jiné) i k vytváření šifrovaných virtuálních privátních sítí. Nicméně IPSEC je poměrně těžkopádná a komplikovaná skupina protokolů s řadou možností, a úplná implementace této skupiny je stále poměrně zřídka používaná a některé problémy (například správa klíčů) stále nejsou plně vyřešeny. CIPE volí jednodušší přístup, kdy se většina volitelných věcí (například šifrovací algoritmus) pevně stanovuje při instalaci. Tím se sice omezuje flexibilita, ale vzniká jednodušší (a tedy efektivní a snáze testovatelná implementace."

Další informace můžete najít na adrese http://www.inka.de/~bigred/devel/cipe.html. Stejně jako u jiných kryptografických produktů ani tento není součástí jádra kvůli exportním omezením.

Kerberos

Kerberos je autentikační systém vyvinutý v projektu Athena na MIT. Když se uživatel přihlásí, Kerberos jej autentikuje (pomocí hesla) a pak uživateli umožní prokázat svou totožnost dalším serverům a systémům v síti.

Tato autentikace se používá v programech jako *rlogin*, které umožňují uživateli přihlašovat se k dalším počítačům bez hesla (a bez použití souboru *.rhosts*). Tuto metodu lze použít i k zajištění, aby pošta došla správnému příjemci, a k zajištění, že odesilatel pošty je opravdu ten, za koho se vydává.

Kerberos a další programy této skupiny zabraňují uživateli podvést systém a vydávat se za někoho jiného. Bohužel, instalace systému Kerberos je velmi náročná, vyžaduje modifikace a náhrady řady standardních programů.

Další informace o systému Kerberos naleznete v dokumentu *Kerberos FAQ* na adrese http://nii.isi.edu/info/kerberos/.

Systém Kerberos by rozhodně neměl být prvním krokem ve zvyšování bezpečnosti systému. Jeho instalace je velmi náročná a není zdaleka tak rozšířený, jako například SSH.

Stínová hesla

Stínová hesla představují mechanismus, jak informace o zašifrovaných heslech udržet skryté před běžnými uživateli. Nové verze distribucí RedHat a Debian používají stínová hesla implicitně, na jiných systémech jsou však hesla uložena v souboru */etc/passwd*, kde je mohou uživatelé volně číst. Kdokoliv pak může spustit nějaký program pro luštění hesel a snažit se je uhodnout. Stínová hesla jsou naproti tomu uložena v souboru */etc/shadow*, který je normálním uživatelům nepřístupný. Abyste mohli stínová hesla použít, musíte zajistit u všech nástrojů, které hesla používají, jejich pře-

ložení s podporou stínových hesel. Výše zmíněný systém PAM umožňuje jednoduché vložení stínového modulu, překlad binárních souborů není nutný. Podrobnější informace můžete najít v dokumentu *Shadow-Password HOWTO* na adrese http://metalab.unc.edu/LDP/HOWTO/Shadow-Password-HOWTO.html. Tento dokument je už poměrně starý a u distribucí používajících PAM není nutný.

„Crack" a „John the Ripper"

Pokud váš program *passwd* z nějakého důvodu nevynucuje zadání obtížně uhodnutelných hesel, můžete vyzkoušet nějaký program pro luštění hesel a ověřit si, zda vaši uživatelé používají bezpečná hesla.

Programy pro luštění hesel jsou založeny na jednoduchém principu: zkoušejí jednotlivá slova a jejich variace ze slovníku, každé zašifrují a porovnají s uloženým heslem. Pokud dojde ke shodě, podařilo se jim uhodnout heslo.

Existuje celá řada programů tohoto typu, nejznámější jsou Crack a John the Ripper (http://www.openwall.com/john/). Jejich spuštění zabere hodně procesorového času, můžete si však ověřit, zda by je útočník nemohl využít, tak, že je nejprve vyzkoušíte sami a uvědomíte uživatele se slabými hesly. Útočník sice bude potřebovat jinou bezpečnostní díru k přečtení souboru */etc/passwd*, ale takové díry jsou běžnější, než byste si mysleli.

Protože bezpečnost je dobrá jen tak, jako nejslabší počítač v síti, stojí za zmínku, že pokud v síti máte počítače s Windows, měli byste vyzkoušet program L0pthCrack, implementaci programu Crack pro Windows. Najdete ji na adrese http://www.l0pht.com.

CFS – Cryptographic File System a TCFS – Transparent Cryptographic File System

CFS je metoda umožňující šifrovat celé adresářové stromy a umožňující uživatelům ukládat zašifrované soubory. Využívá služeb NFS serveru na lokálním počítači. RPM balíčky jsou dostupné na adrese http://www.zedz.net/redhat/, podrobnější informace o fungování systému pak na adrese ftp://ftp.research.att.com/dist/mab/.

TCFS je vylepšení CFS o lepší integraci se souborovým systémem, takže celý mechanismus šifrování je pro uživatele transparentní. Další informace naleznete na adrese http://www.tcfs.it/.

Tento systém rovněž nemusíte použít na celý souborový systém, funguje i na část adresářového stromu.

Další možnosti jak používat šifrovaný souborový systém naleznete na http://www.kerneli.org/.

X11, SVGA a bezpečnost displeje

X11

Je důležité zabezpečit grafický displej, aby se útočníkům zabránilo zmocnit se hesel, která zapisujete, číst dokumenty nebo informace, jež si čtete na obrazovce, nebo dokonce využít nějaké díry k získání práv superuživatele. U vzdáleného spouštění grafických aplikací přes síť rovněž existuje riziko odposlechu, kdy útočník může zaznamenat veškerou vaši interakci se vzdáleným systémem.

Systém X Window obsahuje řadu mechanismů pro řízení přístupu. Jedním z nejjednodušších je mechanismus založený na počítači. Pomocí *xhost* můžete specifikovat, ze kterých počítačů je možný přístup k vašemu displeji. To ovšem není příliš bezpečné, protože jakmile někdo získá přístup k vašemu počítači, může doplnit i svůj počítač a snadno se tak dostat dovnitř. Pokud navíc povo-

lujete přístup z nedůvěryhodných počítačů, kdokoliv na nich může získat přístup k vašemu displeji.

Pokud používáte k přihlášení *xdm* (X Display Manager), máte možnost použít bezpečnější přístupovou metodu: MIT-MAGIC-COOKIE-1. Ve vašem souboru *.Xauthority* bude uložen 128bitový vygenerovaný „cookie". Pokud potřebujete povolit přístup k vašemu displeji vzdálenému počítači, můžete použít příkaz *xauth* a informace v souboru *.Xauthority* a povolit přístup pouze pro toto jedno připojení. Viz dokument *Remote-X-Apps mini-howto* na adrese http://metalab.unc.edu/LDP/HOWTO/mini/Remote-X-Apps.html.

Můžete také použít *ssh* (viz část 6.4) a provozovat zabezpečená X spojení. Výhoda je také v tom, že mechanismus je pro koncového uživatele transparentní a po síti se nepřenášejí žádná nešifrovaná data.

Jakákoliv vzdálená připojení k X serveru můžete zakázat parametrem *–nolisten tcp* X serveru. Tím se zabrání jakýmkoliv síťovým spojením na server.

Podívejte se na manuálovou stránku *Xsecurity*, kde naleznete více informací o zabezpečení systému X Window. Nejbezpečnější řešení je použít *xdm* k přihlášení se na vaši konzolu a pak *ssh* k přihlášení na vzdálené systémy, kde chcete spouštět grafické aplikace.

SVGA

Programy SVGAlib jsou typicky SUID-root, aby mohly přímo přistupovat k videozařízení počítače. Tím jsou velmi nebezpečné. Pokud dojde k jejich havárii, typicky budete muset restartovat počítač. Ujistěte se, že všechny spouštěné SVGA programy jsou autentické a důvěryhodné. Ještě lepší je vůbec je nespouštět.

GGI (Generic Graphics Interface project)

Projekt GGI se snaží řešit některé problémy s videorozhraními v Linuxu. GGI přesouvá některé malé části videokódu přímo do jádra a pak řídí přístup k videosystému. Znamená to, že GGI může v kterémkoliv okamžiku obnovit konzolu do známého dobrého stavu. Také nabízí bezpečnostní mechanismy zabraňující ve spuštění trójských koňů, snažících se odposlechnout heslo. Více viz http://synergy.caltech.edu/~ggi/.

Zařízení jádra

V Linuxu existuje několik blokových a znakových zařízení, která také souvisejí s bezpečností.

Jádro obsahuje zařízení */dev/random* a */dev/urandom*, která slouží jako generátor náhodných čísel.

Obě zařízení by měla být dostatečně bezpečná pro použití při generování PGP klíčů, ssh výměny a dalších aplikací, kde se vyžadují bezpečně náhodná čísla. Útočník nesmí být schopen ze znalosti části náhodné sekvence odhadnout její další pokračování. Bylo vynaloženo značné úsilí k zajištění, že čísla generovaná těmito zařízeními jsou opravdu náhodná v pravém slova smyslu.

Rozdíl mezi těmito zařízeními spočívá v tom, že pokud generátoru */dev/random* dojdou náhodná čísla, čeká, než nashromáždí další. Na některých systémech to může vést k zablokování na delší dobu, než se nasbírá dostatek uživatelem generované „entropie". Zařízení */dev/random* je tedy nutné používat s rozvahou. (Pravděpodobně nejlepší je použít toto zařízení při generování velmi citlivých klíčů, kdy uživateli řeknete, ať ťuká do klávesnice, hýbe myší, dokud neřeknete Dost!)

Zařízení */dev/random* používá kvalitní zdroj entropie založený na měření intervalů mezi přerušeními a dalších zdrojích. Blokuje se, dokud není shromážděn dostatečný počet náhodných bitů.

Zařízení */dev/urandom* je podobné, pokud ale nemá dostatek entropie, vrátí silný kryptografický hash toho, co je k dispozici. Není to tak bezpečné, ale pro většinu aplikací to stačí.

Z těchto zařízení můžete číst například příkazem:

```
root#  head -c 6 /dev/urandom | mimencode
```

Tím získáte šest náhodných znaků, které můžete použít jako heslo. Program *mimencode* naleznete v balíčku *metamail*.

Popis algoritmu naleznete v souboru */usr/src/linux/drivers/char/random.c*.

Bezpečnost sítě

Význam zabezpečení sítě roste s tím, že uživatelé tráví stále více a více času připojeni. Narušení síťové bezpečnosti bývá často mnohem snadnější, než narušení bezpečnosti fyzické nebo lokální, a je také mnohem běžnější.

K zajištění síťové bezpečnosti existuje řada dobrých nástrojů, přičemž stále větší množství se stává standardní součástí linuxových distribucí.

Odposlech paketů

Jeden z nejobvyklejších způsobů, jak útočník získá přístup k většímu množství systémů, je instalace programu pro odposlech paketů na počítači, do nějž se mu už podařilo proniknout. Tyto takzvané *sniffery* nedělají nic jiného, než že na síťovém rozhraní poslouchají a snaží se v paketech najít věci jako *passwd*, *login* nebo *su* a pak zaznamenají následující komunikaci. Tímto způsobem útočník získá hesla k systémům, které ani nemusí napadat. Velmi zranitelná jsou tímto způsobem hesla posílaná v přímém tvaru.

Příklad: Počítač A byl napaden. Útočník nainstaloval sniffer. Ten zachytil přihlášení administrátora na stroj B ze stroje C. Tím útočník získal heslo administrátora ke stroji B. Následně administrátor zadal příkaz *su*, aby mohl provést nějakou správu systému. Tím má útočník heslo superuživatele stroje B. Později administrátor nechá někoho ze svého účtu přihlásit se ke stroji Z na úplně jiné síti. Tím má útočník uživatelské jméno a heslo na stroj Z.

V době strukturované kabeláže už ani není nutné, aby útočník nejprve musel nějaký systém napadnout. Stačí mu přinést si laptop a někde v budově se napojit do sítě.

Spolehlivá obrana proti těmto útokům je použití *ssh* a dalších metod šifrování hesla. Další obranou je například protokol APOP pro výběr poštovních schránek. (Klasický protokol POP3 je velmi zranitelný, protože posílá po síti přímo nešifrovaná hesla.)

Systémové služby a tcp_wrapper

Než linuxový systém připojíte k *jakékoliv* síti, nejprve se podívejte, které služby musíte poskytovat. Služby, které poskytovat nemusíte, by měly být vždy vypnuté – máte tak o starost méně a útočník má o možnost méně, jak nalézt nějakou díru.

V Linuxu existuje celá řada způsobů, jak vypínat služby. Podívejte se do konfiguračního souboru */etc/inetd.conf* a uvidíte, které síťové služby nabízí démon *inetd*. Všechny nepotřebné zakomentujte (na začátek příslušného řádku zadejte znak #) a poté pošlete démonu *inetd* signál SIGHUP.

Dále můžete odstranit (nebo zakomentovat) služby v souboru */etc/services*. Povede to k tomu, že danou službu nebudou moci použít ani lokální uživatelé (pokud například odstraníte službu *ftp* a uživatel se z daného počítače pokusí o FTP připojení na vzdálený počítač, nezdaří se mu to a ob-

drží chybové hlášení „unknown service“). Obvykle ale nestojí za to služby z */etc/services* odstraňovat, protože to bezpečnost nijak nezvyšuje. Pokud by lokální uživatel chtěl použít službu FTP i přesto, že je zakomentována, stačí mu použít vlastního klienta FTP a vše bude fungovat. Služby, které obvykle budete nechávat zapnuty, jsou:

- *ftp*
- *telnet* (nebo *ssh*)
- pošta, například *pop-3* nebo *imap*
- *identd*

Pokud víte, že nějaký balíček vůbec nebudete používat, můžete jej odstranit celý – v distribucích RedHat to provedete příkazem *rpm –e balíček*. V distribucích Debian udělá to samé příkaz *dpkg –remove*.

V každém případě byste měli v souboru */etc/inetd.conf* vypnout nástroje *rsh/rlogin/rcp*, tedy služby *login, shell* a *exec*. Tyto protokoly jsou extrémně nebezpečné a v minulosti byly příčinou celé řady úspěšných útoků.

Dále byste měli zkontrolovat adresáře */etc/rc.d/rc[0-9].d* (na RedHatu, v Debianu je to */etc/rc[0-9].d*) a podívat se, zda se zde nespouštějí nějaké nepotřebné servery. Soubory v tomto adresáři jsou fakticky symbolické odkazy na soubory v */etc/rc.d/init.d* (na RedHatu, v Debianu */etc/init.d*). Přejmenováním souboru v adresáři init.d zrušíte všechny symbolické odkazy na něj. Pokud chcete nějakou služby vypnout pouze na určité úrovni běhu, přejmenujte příslušný odkaz nahrazením velkého *S* malým *s*, takto:

```
root#  cd /etc/rc6.d
root#  mv S45dhcpd s45dhcpd
```

Používáte-li *rc* soubory v uspořádání BSD, hledejte nepotřebné programy v */etc/rc**.

Většina linuxových distribucí se dodává s tzv. tcp_wrappery, které „obalují“ všechny služby TCP. Tcp_wrapper (*tcpd*) se spouští ze souboru *inetd* namísto skutečného serveru. Pak provede kontrolu, zda konkrétní počítač má právo požadovat konkrétní službu, a buď spustí server, anebo připojení odmítne. Pomocí programu *tcpd* tak můžete omezit přístup ke službám TCP. Měli byste vytvořit soubor */etc/hosts.allow* a přidat do něj pouze ty počítače, kterým chcete přístup ke službám povolit.

Jste-li klasický domácí uživatel, doporučujeme vám zakázat všechny služby. Navíc *tcpd* zaznamenává neúspěšné pokusy o přístup ke službám, takže poznáte, když se někdo snaží o útok. Přidáváte-li nové služby založené na protokolu TCP, měli byste je nakonfigurovat tak, aby používaly tcp_wrapper. Domácí uživatelé mohou zabránit ostatním v připojení se k jejich počítači, a přitom stále budou mít možnost přijímat poštu a navazovat spojení na Internet. Stačí v souboru */etc/hosts.allow* nastavit:

```
ALL: 127.
```

A v souboru */etc/hosts.deny* samozřejmě nastavit:

```
ALL: ALL
```

Tím se zabrání v připojení k vašemu počítači zvenčí, ale stále máte umožněno připojovat se na Internet.

Nezapomeňte, že tcp_wrapper chrání pouze služby spouštěné démonem *inetd* a několik málo dalších. Stále však mohou na vašem počítači běžet i jiné služby. Příkazem *netstat –ta* můžete zjistit, jaké služby váš počítač nabízí.

Kontrola informací v DNS

Udržování aktuálních informací DNS o všech počítačích ve vaší síti vede ke zvýšení bezpečnosti. Pokud se do vaší sítě připojí neautorizovaný počítač, můžete jej poznat podle toho, že nemá platný záznam DNS. Řadu služeb je možné nakonfigurovat tak, aby nepřijímaly spojení od počítačů, které nemají platné záznamy DNS.

Identd

Identd je malý program typicky spouštěný serverem *inetd*. Sleduje, který uživatel má spuštěnu jakou TCP službu a hlásí to tomu, kdo o tyto údaje požádá.

Řada lidí nechápe užitečnost služby *identd* a proto ji vypínají anebo blokují všechny dotazy. Služba *identd* nepomáhá vzdáleným systémům. Neexistuje způsob jak zjistit, zda údaje poskytnuté touto službou jsou pravdivé. Dotazy také neumožňují žádnou autentikaci.

K čemu tedy taková služba je? Pomáhá vám a představuje další monitorovací nástroj. Pokud služba *identd* funguje správně, pak víte, že vzdáleným systémům odesílá uživatelské jméno nebo *uid* uživatelů, kteří používají TCP služby. Obrátí-li se na vás správce vzdáleného systému a řekne vám, že uživatel ten/a/ten chtěl napadnout jejich systém, můžete proti němu snadno zakročit. Pokud službu *identd* nepoužíváte, budete muset projít spousty a spousty logů, zjistit, kdo byl zrovna přihlášen a obecně vám to bude trvat mnohem déle, pokud vůbec budete úspěšní.

Program *identd* dodávaný s většinou distribucí se dá nastavovat mnohem více, než většina lidí tuší. Pro určité uživatele jej můžete vypnout (prostřednictvím souboru *.noident*), můžete zaznamenávat všechny příchozí dotazy (doporučujeme), můžete dokonce říct, že namísto uživatelského jména má posílat *uid* nebo text NO-USER.

Konfigurace a zabezpečení MTA Postfix

Postfix je poštovní server napsaný Wietsem Venemou, autorem řady bezpečnostních produktů, jako pokus poskytnout alternativu ke všeobecně rozšířenému poštovnímu programu *sendmail*. Postfix usiluje o to „být rychlý, snadno spravovatelný, a snad i bezpečný, a zároveň dostatečně kompatibilní se sendmailem, aby uživatelům nevadil“.

Další informace o programu Postfix můžete najít na jeho domovské stránce na adrese http://www.postfix.org/, a v dokumentu *Configuring and Securing Postfix* na adrese http://www.linuxsecurity.com/feature_stories/feature_story-91.html.

SATAN, ISS a další síťové skenery

Existuje celá řada různých softwarových balíků, které slouží ke skenování portů a služeb u počítačů v síti. Mezi ty nejznámější patří SATAN, ISS, SAINT a Nessus. Tyto programy se připojují k cílovému počítači (nebo k cílovým počítačům) na všech portech, na kterých to jde, a snaží se zjistit, jaké služby zde běží. Na základě těchto informací pak můžete říct, zda je počítač napadnutelný určitým typem útoku.

SATAN (Security Administrator's Tools for Analyzing Networks) je port skener s webovým rozhraním. Lze jej nastavit na provádění jednoduché, střední nebo silné kontroly počítače nebo počítačů v síti. Je rozumné si tento program opatřit a zkontrolovat jím počítače ve vaší síti, a pak odstranit nalezené problémy. Ujistěte se, že vaše kopie programu pochází přímo z metalabs (http://metalab.unc.edu/pub/packages/security/Satan-for-Linux/) nebo z důvěryhodného serveru. Na Internetu se objevil i stejnojmenný trójský kůň (http://www.trouble.org/~zen/satan/satan.html). Navíc SATAN se už delší dobu nevyvíjí, takže jiné nástroje mohou posloužit lépe.

Dalším port skenerem je ISS (Internet Security Scanner). Je rychlejší než SATAN a tedy vhodnější pro velké sítě. SATAN nicméně poskytuje více informací.

Abacus je sada nástrojů k zajištění bezpečnosti systému a detekci průniků. Další informace zjistíte na domovské stránce programu na adrese http://www.psionic.com/abacus.

Dalším skenerem je Nessus. Má grafické rozhraní a umožňuje přidávat doplňky pro provádění nových typů testů. Další informace najdete na adrese http://www.nessus.org/.

Detekce skenování portů

Existují nástroje, které vás upozorní, pokud se někdo pokouší na váš počítač použít SATAN, ISS nebo jiné skenovací nástroje. Pokud ale používáte tcp_wrapper a pravidelně kontrolujete logovací soubory, měli byste takové pokusy sami zaznamenat. I při nejšetrnějším nastavení SATAN stále na běžném RedHat systému zanechá v logovacích souborech záznamy o své činnosti.

Existují také „neviditelné" skenery. Paket s nastaveným bitem TCP ACK (který se používá u navázaného spojení) pravděpodobně projde firewallem. Navrácený paket RST z portu, *na němž žádné spojení není navázáno,* se pak dá považovat za důkaz „života" na tomto portu. Takovou aktivitu tcp_wrapper nezachytí.

Mohl by vás zajímat program SNORT, který detekuje různé typy síťových útoků, viz http://www.snort.org/.

Sendmail, qmail a další MTA

Jednou z nejdůležitějších poskytovaných služeb je poštovní server. Bohužel je tento server zároveň nejzranitelnější vzhledem k množství úkonů, které musí vykonávat a k privilegiím, jež k tomu potřebuje. Pokud používáte *sendmail*, je nesmírně důležité používat aktuální verzi. Tento program má velice bohatou historii různých odhalených chyb. Vždy se ujistěte, že používáte nejnovější verzi z http://www.sendmail.org.

Nezapomeňte, že pokud chcete jenom *odesílat* poštu, pak *sendmail* nepotřebujete. Jako běžný domácí uživatel můžete *sendmail* klidně vypnout a poštu odesílat přímo poštovním klientem. Můžete také odstranit parametry *–bd* ze spouštěcího souboru programu *sendmail*, takže nebude přijímat připojení ze sítě. Jinak řečeno, stačí v příslušném spouštěcím skriptu spouštět *sendmail* pouze takto:

```
# /usr/lib/sendmail -q15m
```

Tím zajistíte, že *sendmail* bude každých 15 minut kontrolovat lokální frontu zpráv a odesílat zprávy v ní uložené.

Řada administrátorů *sendmail* vůbec nepoužívá a místo toho volí jiné programy pro zpracování pošty. Doporučujeme například *qmail*. Ten byl od počátku navrhován s ohledem na bezpečnost. Je rychlý, stabilní a bezpečný. Naleznete jej na adrese http://www.qmail.org.

Přímým konkurentem *qmailu* je *postfix* Wietse Venemy, autora programu *tcp_wrapper* a dalších bezpečnostních produktů. Původně se jmenoval *vmailer* a jeho vývoj sponzorovala společnost IBM. Byl rovněž od počátku navrhován s ohledem na bezpečnost. Další informace o tomto programu najdete na adrese http://www.postfix.org.

Útoky typu odepření služeb

Útok typu *odepření služeb* (DoS) je typ útoku, kdy se útočník snaží natolik zatížit nějaký prostředek, že nebude schopen reagovat na oprávněné požadavky, nebo kdy se snaží oprávněným uživatelům úplně znemožnit přístup k počítači.

DoS útoky se v posledních letech hodně rozšiřují. Dále uvádíme některé nejznámější a nejnovější. Kromě toho se stále objevují nové typy těchto útoků. Aktuální informace naleznete v konferencích věnovaných bezpečnosti a v konferenci bugtraq.

- SYN Flooding – jedná se o síťový útok DoS. Využívá mechanismu, jakým je implementováno navazování spojení TCP. Nová linuxová jádra (2.0.30 a vyšší) obsahují různé volby, které umožní zabránit tomuto útoku v zablokování přístupu k počítači a službám. Příslušné parametry jádra jsou popsány v sedmé části.
- Pentium „F00F" bug – ukázalo se, že jistá sekvence strojových instrukcí způsobí restart klasických procesorů Pentium, bez ohledu na provozovaný operační systém. Týká se to pouze klasických Pentií, nikoliv modernějších verzí (Pentium Pro, Pentium II a vyšší). Jádra 2.0.32 a vyšší obsahují speciální ochranu proti tomuto útoku, která navíc byla v jádře 2.0.33 vylepšena.
- Ping Flooding – jde o jednoduchý útok „hrubou silou". Útočník posílá záplavu paketů ICMP. Pokud tento útok pochází z počítače s lepší konektivitou, než máte vy, váš počítač nebude schopen normální komunikace po síti. Varianta tohoto útoků, tzv. „smurfing", posílá pakety ICMP jinému počítači a jako odesilatele uvádí váš počítač, takže původce útoku je hůře detekovatelný. Další informace o tomto typu útoku můžete najít na adrese http://www.quadrunner.com/~chuegen/smurf.txt.

  Pokud takovýto útok zjistíte, pomocí programu *tcpdump* nebo podobného zjistěte, odkud útok pochází (nebo odkud se *tváří*, že pochází), a sdělte to svému poskytovateli připojení. Tento typ útoku se dá velmi snadno zablokovat na připojovacím routeru nebo firewallem.
- Ping of Death – tento útok posílá pakety ICMP ECHO REQUEST příliš dlouhé, než aby se vešly do datových struktur, které jsou pro ně určeny. Protože posláním jediného velkého (65 510 bajtů) paketu je možné řadu systémů zablokovat nebo dokonce zhroutit, byl útok záhy pojmenován „Ping of Death". Jedná se o známou a dávno opravenou chybu, takže v současné době se jej už nemusíte obávat.
- Teardrop / New Tear – poměrně nový útok založený na chybě v implementaci fragmentace protokolu IP v Linuxu i ve Windows. Chyba byla opravena v jádře 2.0.33.

Programy pro jednotlivé útoky a podrobnější popis jejich činnosti můžete najít pomocí vyhledávače na adrese http://www.rootshell.com/.

Zabezpečení NFS (Network File System)

NFS je velmi rozšířený protokol pro sdílení souborů. Umožňuje, aby servery prostřednictvím démonů *nfsd* a *mountd* „exportovaly" celé souborové systémy na jiné počítače, kde je tento protokol podporován přímo v jádře, nebo nějakým jiným způsobem (nejde-li o linuxové stroje). Démon *mountd* udržuje informace o připojených souborových systémech v souboru */etc/mtab* a zobrazí je příkazem *showmount*.

V řadě prostředí se NFS používá k tomu, aby uživatelé měli přístupné své domovské adresáře bez ohledu na to, ke kterému počítači se přihlásí. Zajištění bezpečnosti při exportu je poměrně slabé. Můžete démonu *nfsd* říct, aby vzdáleného uživatele *root* (*uid=0*) mapoval na uživatele *nobody*, čímž mu zabráníte v úplném přístupu k exportovanému souborovému systému. Protože však jednotliví uživatelé mají přístup ke svým souborům (nebo přesněji k souborům s jejich *uid*), může vzdálený *root* provést *su* na účet libovolného uživatele a přistupovat tak k jeho souborům. Pro útočníka to představuje pouze malou nepříjemnost.

Pokud musíte používat NFS, povolte export pouze na ty počítače, kde je to opravdu nutné. Nikdy neexportujte celý kořenový svazek, exportujte pouze ty adresáře, které musíte.

Další informace o NFS naleznete v dokumentu *NFS HOWTO* na adrese http://metalab.unc.edu/mdw/HOWTO/NFS-HOWTO.html.

NIS (Network Information Service), dříve YP

Network Information Service je metoda distribuce informací skupinám počítačů. master NIS udržuje informační tabulky a konvertuje je na takzvané mapy NIS. Tyto mapy pak poskytuje po síti a umožňuje klientským počítačům získat přihlašovací jména, hesla, domovské adresáře a další informace (všechny informace ze standardního souboru */etc/passwd*). Díky tomu může uživatel změnit heslo jen jednou a změna se projeví v celé doméně NIS.

NIS není bezpečná služba. Nikdy ani tak nebyla navrhována. Byla navržena jako jednoduchá a užitečná. Pokud kdokoliv uhodne název vaší domény NIS, může získat kopii souboru *passwd* a pomocí dalších programů pak luštit hesla uživatelů. Kromě toho je možné NIS i obelstít a provádět spoustu nehezkých triků. Pokud musíte NIS používat, berte na vědomí všechna s tím spojená rizika.

Existuje i bezpečnější varianta služby NIS, pojmenovaná NIS+. Podrobnější informace najdete v dokumentu *NIS HOWTO* na adrese http://metalab.unc.edu/mdw/HOWTO/NIS-HOWTO.html.

Firewally

Firewall představuje prostředek k řízení, jaké informace mohou cestovat dovnitř a vně vaší sítě. Typicky je firewall připojen jednak k Internetu a jednak k lokální síti, a jakákoliv komunikace s Internetem je možná pouze přes něj. Díky tomu může firewall určovat, co může mezi vaší sítí a Internetem putovat.

Existuje řada typů firewallů a způsobů, jak je nastavit. Velmi dobrý firewall lze vytvořit z linuxového počítače. Funkce firewallu může být přímo součástí jádra 2.0 a vyšších. Uživatelské nástroje, jako *ipfwadm* pro jádra 2.0, *ipchains* pro jádra 2.2 a iptables pro jádra 2.4, umožňují kdykoliv nastavovat povolené typy síťového provozu. Různé typy provozu je také možné logovat.

Firewally představují velmi užitečnou a důležitou techniku v zabezpečení sítě. Nikdy však nepodlehněte dojmu, že jste-li za firewallem, nemusíte se starat o bezpečnost lokálních počítačů. To je fatální chyba. Podrobnější informace o firewallech a Linuxu naleznete ve velmi dobrém dokumentu *Firewall-HOWTO* na adrese http://metalab.unc.edu/mdw/HOWTO/Firewall-HOWTO.html.

Další informace najdete také v dokumentu *IP-Masquerade mini-howto* na adrese http://metalab.unc.edu/mdw/HOWTO/mini/IP-Masquerade.html.

Podrobnosti o programu *ipfwadm* (programu, který umožňuje nastavovat firewall) najdete na jeho domovské stránce na adrese http://www.xos.nl/linux/ipfwadm/.

Pokud nemáte s firewally žádné zkušenosti a hodláte nastavovat více než jen jednoduché zabezpečení, nutně si přečtěte knihu Firewall nakladatelství O'Reilly and Associates (http://www.ora.com), nebo nějakou jinou literaturu věnovanou této problematice. Vynikající dokumenty o firewallech zveřejnil National Institute of Standards and Technology. I když pocházejí z roku 1995, stále jsou velmi aktuální. Najdete je na adrese http://csrc.nist.gov/nistpubs/800-10/main.html. Další zajímavé odkazy jsou:

- The Freefire Project – seznam zdarma dostupných firewallů, http://sites.inka.de/sites/lina/freefire-l/index_en.html.

- SunWorld Firewall Design – dokument stejných autorů jako výše zmíněná kniha nakladatelství O'Reilly. Popisuje různé typy firewallů. Najdete jej na adrese http://www.sunworld.com/swol-01-1996/swol-01-firewall.html.
- Mason – nástroj pro automatické nastavení firewallu v Linuxu. Tento skript se postupně učí, co potřebujete na síti dělat. Další informace najdete na adrese http://www.pobox.com/~wstearns/mason/.

IP Chains – firewall v jádře 2.2

Linux IP Firewalling Chains je aktualizace firewallového kódu z jádra 2.0 v jádrech 2.2. Oproti předchozím implementacím obsahuje řadu vylepšení, například:

- Pružnější manipulace s pakety.
- Lepší možnosti účtování.
- Atomické změny nastavení.
- Možnost explicitního zpracování fragmentů.
- Logování podezřelých paketů.
- Obsluha i jiných protokolů než ICMP/TCP/UDP.

Pokud používáte program ipfwadm na jádře 2.0, existují skripty, které provedou konverzi konfiguračních souborů do formátu *ipchains*. Další informace naleznete v dokumentu *IP Chains HOWTO* na adrese http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html.

Netfilter – firewall v jádře 2.4

Jedná se o další vylepšení jaderného firewallu pro jádra 2.4.

Subsystém netfilter představuje kompletně přepracované předchozí implementace *ipfwadm* a *ipchains*. Netfilter nabízí celou řadu vylepšení a představuje tak velmi robustní a spolehlivý nástroj pro zabezpečení sítí.

Nastavování pravidel firewallu se provádí příkazem *iptables*.

Netfilter umožňuje manipulovat s pakety při jejich průchodu různými částmi jádra. Jednotlivé části obsahují podporu maškarády, klasický paketový filtr a složitějších funkcí, jako je NAT. Systém obsahuje dokonce podporu pro distribuci zátěže určitého typu požadavků na více serverů.

Velmi mocné jsou funkce stavové inspekce. Stavová inspekce umožňuje sledovat a řídit komunikaci procházející filtrem. Díky možnosti udržovat přehled stavových a kontextových informací o jednotlivých spojeních se usnadňuje vytváření pravidel pro interpretaci protokolů vyšší úrovně.

Navíc je možné vytvářet samostatné moduly plnící další funkce, například předání paketů uživatelskému programu a jejich následné vrácení do jádra. Díky tomu se zjednodušují operace, které bylo dříve nutné řešit přímými zásahy do jádra.

Další informace o IP Tables naleznete v dokumentech:

- Oskar Andreasson IP Tables Tutorial (http://www.linuxsecurity.com/feature_stories/feature_story-94.html) – rozhovor serveru *LinuxSecurity.com* s Oskarem Andreassonem o jeho „IP Tables tutorialu" a o použití tohoto dokumentu při vytváření robustních firewallů.
- Linux Security Quick-Start (http://www.linuxsecurity.com/feature_stories/feature_story-93.html) – Hal Burgiss je autorem dvou základních příruček věnovaných zabezpečení Linuxu, kde se mimo jiné popisuje i správa firewallu.
- Netfiltering Homepage (http://netfilter.samba.org/) – Domovská stránka produktu.

- Linux Kernel 2.4 Firewalling Matures: netfilter (http://www.linuxsecurity.com/feature_stories/kernel-netfilter.html) – Tento článek na serveru *LinuxSecurity.com* obsahuje základní informace o filtraci paketů, jak používat iptables, a nové funkce poslední generace linuxových firewallů.

VPN – Virtuální privátní sítě

VPN je metoda umožňující vytvořit „virtuální" síť nad nějakou stávající sítí. Tato virtuální síť je typicky šifrovaná a předává data pouze mezi známými entitami připojenými k síti. VPN se často používají k připojení do interní firemní sítě zvnějšku přes Internet.

Pokud používáte maškarádovací firewall v Linuxu a potřebujete povolit průchod paketů MS PPTP (VPN produkt společnosti Microsoft), existuje úprava jádra, která to umožňuje, viz ftp://ftp.rubyriver.com/pub/jhardin/masquerade/ip_masq_vpn.html.

Pro Linux existuje několik produktů VPN:

- vpnd, http://sunsite.dk/vpnd/
- Free S/Wan, http://www.xs4all.nl/~freeswan/
- Implementace VPN pomocí ssh, více viz VPN mini-howto
- vps (virtual private server), http://www.strongcrypto.com/
- yawipin, mailto:http://yavipin.sourceforge.net

Další odkazy najdete také v části věnované IPSEC.

Bezpečnostní příprava (než se připojíte k síti)

Máte tedy počítač zkontrolován, věříte, že je tak bezpečný, jak je jen možné, a chystáte se jej připojit k síti. Existuje několik věcí, které byste měli v tomto okamžiku udělat jako ochranu před útokem, abyste byli schopni útok rychle detekovat, zastavit jej a počítač znovu zprovoznit.

Pořiďte si úplnou zálohu

Debata o metodách a způsobech zálohování je mimo rozsah tohoto dokumentu, uveďme si však několik informací vztahujících se k zálohování a bezpečnosti.

Pokud máte na disku uloženo méně než 650 MB dat, dobrá metoda je vytvořit zálohu na CD disk. Takovou zálohu nelze později modifikovat a při dobrém uložení má dlouhou životnost. Samozřejmě budete potřebovat alespoň 650 MB volného diskového prostoru k vytvoření obrazu zálohovaných dat. Pásky a další přepisovatelná média byste měli ihned po dokončení zálohování chránit proti zápisu. Nezapomeňte zálohy uložit na bezpečné místo. Dobrá záloha zajistí, že máte bod, od nějž můžete systém obnovovat.

Zvolte vhodný zálohovací plán

Velmi snadno se používá cyklus šesti pásek. Pracuje se čtyřmi páskami pro každý pracovní den, pátou pro sudé pátky a šestou pro liché pátky. Každý den se provádí inkrementální záloha, každý pátek pak úplná záloha na příslušnou pásku. Pokud provedete nějaké zásadní změny konfigurace nebo doplníte důležitá data, je rozumné pořídit úplnou zálohu mimo běžný páskový cyklus.

Testujte zálohy

Pravidelně zálohy testujte, abyste měli jistotu, že fungují tak, jak čekáte. Pravidelně byste měli soubory obnovit a porovnat s ostrými daty, dále byste měli pravidelně testovat čitelnost starších záloh.

Zálohujte databáze RPM

V případě napadení počítače můžete databázi RPM použít podobně jako program *tripwire*, ovšem pouze máte-li jistotu, že nebyla modifikována. Měli byste mít zálohu databáze RPM uloženu na disketě mimo počítač. Podobné funkce lze použít i na distribucích Debian.

Soubory */var/lib/rpm/fileindes.rpm* a */var/lib/rpm/packages.rpm* se velmi pravděpodobně nevejdou na jednu disketu. Pokud je ale zkomprimujete, můžete každý nahrát na samostatnou disketu.

Při napadení systému pak můžete použít příkaz

```
root#  rpm -Va
```

a provést kontrolu jednotlivých souborů v počítači. Podívejte se na manuálové stránky programu *rpm*, protože program nabízí ještě další volby. Musíte také mít zajištěno, že nedošlo přímo k modifikaci programu *rpm*.

Při přidání každého balíčku RPM do systému musíte provést zálohu databáze RPM. Sami se rozhodněte, zda to za to stojí.

Sledujte logy systému

Důležité je, aby nedošlo k porušení systémových logů. Dobrý začátek je nastavit soubory v adresáři */var/log* tak, aby je mohla číst a zapisovat pouze omezená skupina uživatelů.

Nezapomínejte pravidelně sledovat, co se v těchto záznamech objevuje. Důležité jsou typicky záznamy třídy „auth" – například vícenásobné neúspěšné přihlášení může indikovat pokus o útok.

Umístění logovacích souborů závisí na distribuci. V systémech, které odpovídají standardu „Linux Filesystem Standard", jako je například RedHat, hledejte v adresáři */var/log* soubory *messages*, *maillog* a další.

Kam ukládá logovací soubory ta která distribuce můžete zjistit v souboru */etc/syslog.conf*. Tento soubor říká logovacímu démonu *syslogd*, kam záznamy ukládat.

Kromě toho můžete nakonfigurovat skript nebo démona zajišťující rotaci logů tak, abyste měli více času na jejich prozkoumání. V nových distribucích RedHat to zajišťuje balíček *logrotate*, jiné distribuce mají jistě něco podobného.

Jestliže došlo k poškození logů, zkuste, zda se vám podaří zjistit, kdy k poškození došlo a co bylo změněno. Existují delší časová období, pro která nejsou žádné záznamy? Dobrý nápad je vyhledat nepoškozené logy v zálohách (pokud je máte).

Typicky útočník modifikuje logovací soubory, aby zakryl stopy po své činnosti, a právě proto byste neměli zapomínat je pravidelně kontrolovat. Můžete zaznamenat pokus útočníka o přístup, nebo objevit program, který se snaží o přístup na účet superuživatele. Možná se vám podaří odhalit podezřelé záznamy dříve, než je útočník zlikviduje. Rozhodně byste měli záznamy třídy *auth* oddělit od ostatních záznamů – týká se to především záznamů o použití příkazu *su*, záznamů o přihlášení uživatelů a dalších informací o uživatelských účtech.

Pokud je to možné, nakonfigurujte *syslogd* tak, aby kopii důležitých dat posílal na bezpečný systém. Tím se útočníkovi zabrání zakrýt důkazy o své přítomnosti. Podrobnosti viz manuálová stránka *syslogd.conf*, volba @.

Existují i propracovanější logovací programy. Podívejte se na program Secure Syslog na adrese http://www.core-sdi.com/ssyslog/. Tento program umožňuje šifrovat záznamy v logu a zabránit tak jejich modifikaci.

Dalším takovým programem je *syslog-ng* (http://www.balabit.hu/products/syslog-ng.html). Nabízí podstatně větší možnosti logování a podporuje záznam na vzdálený systém, aby byly logy chráněny před modifikací.

A konečně – logovací záznamy jsou k ničemu, pokud je nikdo nečte. Udělejte si občas chvíli času a podívejte se na ně, abyste získali představu, jak vypadají za normálních okolností. Daleko snáze pak odhalíte cokoliv neobvyklého.

Provádějte aktualizace systému

Většina uživatelů instaluje Linux z CD disků. Vzhledem k průběžnému odhalování bezpečnostních problémů dochází k častému zveřejňování nových (opravených) programů. Než počítač připojíte k síti, je rozumné navštívit stránky distributora a stáhnout si aktualizované verze programů, které jste nainstalovali z CD. Často tyto nové verze obsahují důležité bezpečnostní opravy a je tedy velmi rozumné je nainstalovat.

Co dělat během a po útoku

Řekněme, že jste se drželi zde (nebo jinde) popsaných doporučení, a zaznamenali jste útok na váš systém. Co teď? Nejdůležitější je zůstat klidný. Neuváženým postupem můžete napáchat více škody, než samotný útočník.

Útok probíhá

Odhalení právě probíhajícího útoku může být docela vzrušující. Vaše reakce může mít dalekosáhlé následky.

Pokud jde o fyzické narušení, zjevně jste přistihli někoho; může jít přímo o fyzické vloupání k vám domů, do kanceláře nebo do učebny. Pak je vhodné obrátit se na příslušné instituce. V případě učebny může jít také o to, že se někdo snaží otevřít nebo restartovat počítač. V závislosti na vaší pozici a zavedených postupech jej můžete požádat o ukončení jeho činnosti, nebo se můžete obrátit na ostrahu.

Odhalíte-li útok ze strany lokálního uživatele, nejprve se ujistěte, že útočí opravdu ten, koho myslíte. Podívejte se, odkud je uživatel přihlášen. Přihlašuje se takto běžně? Ne? Zkuste jej kontaktovat nějakým nepočítačovým způsobem – například telefonicky, nebo zajděte do jeho kanceláře. Pokud je opravdu přihlášen, požádejte jej o vysvětlení, co že to dělá, nebo jej rovnou požádejte, ať s danou činností přestane. Pokud není přihlášen a nemá potuchy, o čem mluvíte, pak je nutné situaci podrobněji prověřit. Než vznesete nějaké obvinění, shromážděte dostatečné množství spolehlivých informací.

Detekujete-li síťový útok, pak v první řadě (je-li to možné) odpojte počítač od sítě. Je-li připojen modemem, odpojte telefonní kabel, je-li připojen přes Ethernet, odpojte ethernetový kabel. Tím zabráníte vzniku dalších škod a útočník se bude domnívat, že jde nejspíš o síťový problém, a ne o odhalení útoku.

Nemůžete-li počítač odpojit od sítě (jedná se o důležitý systém, nebo k němu nemáte fyzický přístup), použijte nástroje jako tcp_wrapper nebo ipfwadm a zablokujte přístup útočícího systému. Pokud nemůžete zablokovat přístup celému vzdálenému systému, zablokujte uživatelský účet, přes nějž útok probíhá. Nezapomeňte, že zablokování účtu není úplně triviální. Nezapomínejte na soubory *.rhosts*, přístup přes FTP a případná možná zadní vrátka.

Jakmile provedete výše popsané (odpojení systému, zablokování přístupu, zablokování účtu), zabijte všechny procesy daného uživatele a odhlašte jej.

Několik dalších minut byste měli systém pozorně sledovat, protože útočník se bude pravděpodobně chtít vrátit – možná použije jiný účet, možná se připojí z jiné adresy.

K útoku už došlo

Odhalili jste útok, nebo se vám jej podařilo přímo zastavit? Takže co teď?

Zavření přístupu

Pokud se vám podaří odhalit metodu, kterou se útočník do počítače dostal, měli byste tuto přístupovou cestu odříznout. Můžete si například všimnout neobvyklé aktivity přes FTP bezprostředně před přihlášením útočníka. Vypněte tedy službu FTP a zkontrolujte, zda neexistuje aktualizovaná verze vámi používaného programu nebo zda nebyla hlášena nějaká chyba. Zkontrolujte logy a podívejte se na stránky s bezpečnostní problematikou, zda nejsou k dispozici nějaké nové opravy. Bezpečnostní opravy pro systém Caldera najdete na stránce http://www.caldera.com/tech-ref/security/. RedHat zatím nemá odděleny bezpečnostní opravy od ostatních oprav, všechny najdete na adrese http://www.redhat.com/errata.

Debian má jednak poštovní konferenci věnovanou bezpečnosti, a jednak samostatné stránky na adrese http://www.debian.org/security/.

Je velmi pravděpodobné, že jakmile se objeví nějaká oprava v jedné distribuci, záhy se objeví i v dalších.

Existuje také projekt bezpečnostního auditu Linuxu. Metodicky prochází jednotlivé uživatelské programy a hledá možné bezpečnostní díry. Z prohlášení projektu:

„Snažíme se o systematický audit zdrojových kódů Linuxu s cílem dosáhnout stejné bezpečnosti jako u OpenBSD. Doposud jsme odhalili (a opravili) několik problémů, uvítáme však další pomoc. Konference není moderovaná a může sloužit jako dobrý zdroj obecných informací o bezpečnosti. Adresa je security-audit@ferret.lmh.ox.ac.uk. Chcete-li se do konference přihlásit, pošlete zprávu na adresu security-audit-subscribe@ferret.lmh.ox.ac.uk."

Pokud se vám nepodaří útočníkovi zablokovat přístup, velmi pravděpodobně se vrátí – a to nejen přímo do napadnutého počítače, ale možná i jinam do sítě. Pokud se útočníkovi podařilo nějakou dobu odposlouchávat provoz na síti, je velmi pravděpodobné, že získal přístup i k jiným systémům.

Odhad škody

Prvním krokem je odhad škody. Co bylo poškozeno? Pokud používáte nějakou metodu kontroly integrity, například Tripwire, můžete zkontrolovat, které soubory byly poškozeny. Pokud nic takového nemáte, budete muset zkontrolovat všechna důležitá data.

Protože instalace linuxových systémů je poměrně jednoduchá, rozumné řešení může být záloha konfiguračních souborů, smazání disků, reinstalace systému a obnova uživatelských dat a konfiguračních souborů ze záloh. Tím máte zajištěno, že máte nový, čistý systém. Pokud musíte obno-

vovat nějaká data z nabouraného systému, dávejte si obzvláštní pozor na binární soubory, protože může jít o trójské koně zanechané útočníkem.

Reinstalace by měla být chápána jako povinná, pokud útočník získal práva superuživatele. Navíc možná budete chtít uchovat důkazy o napadení, takže možná celý disk napadeného systému uložíte na bezpečné místo.

Pak je načase se začít starat o to, jak dlouho napadení trvalo a zda nebyla poškozena i data na zálohách. O problematice zálohování budeme ještě hovořit.

Zálohovat, zálohovat, zálohovat!

Pravidelné zálohy jsou z bezpečnostního pohledu nezbytné. Pokud dojde k nabourání systému, můžete data obnovit ze záloh. Samozřejmě některá data mohou být hodnotná i pro útočníka, takže nejen že vaše data smaže, ale také je ukradne a pořídí si vlastní kopii – tak či tak vám alespoň zbudou zálohy.

Před obnovením poškozených souborů byste měli projít několik záloh zpátky. Útočník mohl soubory poškodit už před delší dobou a vy jste mohli úspěšně zálohovat poškozená data! Samozřejmě i pro zálohy musí platit nějaká bezpečnostní pravidla. Rozhodně je ukládejte na bezpečném místě tak, abyste věděli, kdo k nim má přístup. (Pokud má útočník fyzický přístup k zálohám, může si pořídit kopii dat, aniž byste se to kdy dozvěděli.)

Stopování útočníka

Takže jste útok zastavili a systém obnovili. Tím ale práce nekončí. I když pravděpodobnost vypátrání útočníka není velká, měli byste útok nahlásit.

Útok byste měli ohlásit administrativnímu kontaktu systému, z nějž k útoku došlo. Kontakt můžete zjistit pomocí databáze *whois* nebo *Internic*. Můžete jim poslat zprávu s relevantními logovacími záznamy, daty a časy. Pokud zjistíte o útoku cokoliv dalšího, napište to také. Po odeslání zprávy můžete pokračovat telefonátem správci vzdáleného systému. Pokud správce útočníka vypátral, bude s ním možná hovořit, a tak dále.

Dobrý útočník často používá mnoho mezilehlých systémů, přičemž některé z nich ani netuší, že jsou takto zneužívány. Pokus o vysledování útočníka může být velmi obtížný. Při rozhovorech se správci jiných systémů buďte zdvořilí, daleko ochotněji vám pak pomohou.

Dále byste měli informovat případnou bezpečnostní organizaci, jíž jste členem (CERT nebo podobně) a dodavatele distribuce.

Informace o bezpečnosti

Existuje velké množství serverů věnovaných obecně bezpečnostním otázkám a speciálně bezpečnosti Linuxu. Je rozumné se přihlásit do jedné nebo více bezpečnostních konferencí a sledovat zprávy o nově odhalených chybách. Většina těchto konferencí nemá velký provoz, a jejich zprávy jsou velmi cenné.

LinuxSecurity.com

Server LinuxSecurity.com obsahuje velké množství materiálů napsaných pracovníky tohoto serveru a řadou dalších autorů.

- Linux Advisory Watch (http://www.linuxsecurity.com/vuln-newsletter.html) – vyčerpávající týdeník zaměřený na nově odhalené bezpečnostní chyby. Obsahuje odkazy na aktualizované balíky a popisy jednotlivých chyb.

- Linux Security Week (http://www.linuxsecurity.com/newsletter.html) – cílem tohoto týdeníku je poskytnout čtenářům nejdůležitější zprávy z oblasti bezpečnosti Linuxu.
- Linux Security Discussion List (http://www.linuxsecurity.com/general/mailinglists.html) – poštovní konference zaměřená na obecné otázky bezpečnosti.
- Linux Security Newsletters (http://www.linuxsecurity.com/general/mailinglists.html) – informace o objednání jednotlivých zpravodajů.
- comp.os.linux.security FAQ (http://www.linuxsecurity.com/docs/colsfaq.html) – časté otázky a odpovědi z diskusní skupiny comp.os.linux.security.
- Linux Security Documentation (http://www.linuxsecurity.com/docs/) – vynikající místo s informacemi o bezpečnosti Linuxu a Open-Source produktů.

FTP servery

CERT je Computer Emergency Response Team. Často zveřejňují upozornění na nové útoky a opravy. Další informace viz ftp://ftp.cert.org/.

ZEDZ (dříve Replay, http://www.zedz.net/) nabízí archiv řady bezpečnostních produktů. Protože nejde o americký server, nemusí se řídit americkými omezeními silné kryptografie.

Matt Blaze je autor CFS a odborník na bezpečnost. Jeho archiv najdete na adrese ftp://ftp.research.att.com/pub/mab.

tue.nl je vynikající nizozemský server věnovaný bezpečnosti, ftp://ftp.win.tue.nl/pub/security/.

WWW servery

- Hacker FAQ – otázky a odpovědi o hackerech, http://www.solon.com/~seebs/faqs/hacker.html.
- Archive COAST obsahuje řadu bezpečnostních programů a informací, http://www.cs.purdue.edu/coast/.
- Stránky společnosti SuSE věnované bezpečnosti, http://www.suse.de/security/.
- Server Rootshell.com ukazuje, jaké díry útočníci momentálně používají, http://www.rootshell.com/.
- BUGTRAQ zveřejňuje bezpečnostní problémy, http://www.netspace.org/lsv-archive/bugtraq.html.
- CERT, Computer Emergency Response Team, oznamuje běžné útoky na unixové systémy, http://www.cert.org/.
- Dan Farmer je autor programu SATAN a dalších bezpečnostních nástrojů. Jeho stránky obsahují zajímavý průzkum o bezpečnosti a řadu bezpečnostních nástrojů, http://www.trouble.org/.
- The Linux Security WWW je server s dobrými informacemi o bezpečnosti Linuxu, http://www.aoy.com/Linux/Security/.
- Infilsec je databáze chyb, z níž se dozvíte, jak jsou které systémy zranitelné, http://www.infilsec.com/vulnerabilities/.
- CIAC vydává pravidelné bulletiny o běžných útocích, http://ciac.llnl.gov/cgi-bin/index/bulletins.
- Úvod do technologie PAM najdete na adrese http://www.kernel.org/pub/linux/libs/pam/.

- Debian nabízí stránky s bezpečnostními opravami a informacemi, http://www.debian.com/security/.
- WWW security FAQ Lincolna Steina je vynikající dokument o bezpečnosti webových serverů, http://www.w3.org/Security/Faq/www-security-faq.html.

Poštovní konference

- Bugtraq – pro přihlášení se pošlete e-mail na adresu listserv@netspace.org, v těle zprávy uveďte text *subscribe bugtraq*. (Archiv konference naleznete na adrese uvedené v předchozí části.)
- CIAC – pošlete e-mail na adresu majordomo@tholia.llnl.gov, v těle zprávy uveďte *subscribe ciac-bulletin*.
- RedHat provozuje řadu konferencí, nejdůležitější je *redhat-announce*, kde se můžete dočíst o bezpečnostních (a jiných) opravách bezprostředně po jejich zveřejnění. Pošlete e-mail na adresu redhat-announce-list-request@redhat.com, jako subject uveďte *Subscribe*. Další informace a archiv najdete na adrese https://listman.redhat.com/mailman/listinfo/.
- Debian nabízí konferenci věnovanou bezpečnostním opravám, další informace najdete na adrese http://www.debian.com/security/.

Tištěné materiály

Byla vydána celá řada knih věnovaná bezpečnostní problematice. Následující seznam uvádí jen vybrané z nich. Kromě specializovaných knih je otázka bezpečnosti zmiňována i v řadě knih o správě systému.

- D. Brent Chapman, Elizabeth D. Zwicky: Building Internet Firewalls, září 1995, ISBN 1-56592-124-0
- Simson Garfinkel, Gene Spafford: Practical UNIX & Internet Security, duben 1996, ISBN 1-56592-148-8
- Deborah Russel, G. T. Gangemi sr.: Computer Security Basics, červenec 1991, ISBN 0-937175-71-4
- Olaf Kirch: Linux Network Administrator's Guide, leden 1995, ISBN 1-56592-087-2
- Simson Garfinkel: PGP: Pretty Good Privacy, prosinec 1994, ISBN 1-56592-098-8
- David Icove, Karl Seger, William von Storch: Computer Crime A Crimefighter's Handbook, srpen 1995, ISBN 1-56592-086-4
- John S. Flowers: Linux Security, březen 1999, ISBN 0735700354
- Maximum Linux Security : A Hacker's Guide to Protecting Your Linux Server and Network, červenec 1999, ISBN 0471290009
- Donn Parker: Fighting Computer Crime, září 1998, ISBN 0471163783

Slovníček

Dále uvádíme některé časté pojmy z oboru počítačové bezpečnosti. Vyčerpávající slovník termínů z této oblasti naleznete na serveru LinuxSecurity.com.

- *Authentication – Proces zajištění, že přijatá data jsou stejná jako odeslaná, a že odesilatel dat je opravdu tím, za koho se vydává.*

- *Bastion host* – Počítačový systém, který musí být silně zabezpečen, protože je vystaven útokům, obvykle proto, že je vystaven na Internetu a představuje hlavní kontaktní systém pro uživatele interní sítě.
- *Buffer overflow* – Běžná programátorská chyba je nerezervovat si dostatečný prostor pro data a nekontrolovat jejich velikost. Když dojde k přetečení dat, může být spuštěný program donucen dělat něco jiného. Typicky se přetečení použije k přepsání návratové adresy funkce na zásobníku na jinou adresu.
- *Denial of service* – Útok, který spotřebuje prostředky systému na něco, na co nejsou určeny, takže znemožní použití systému k legitimním účelům.
- *Dual-homed host* – Obecný počítačový systém, který má alespoň dvě síťová rozhraní.
- *Firewall* – Komponenta nebo skupina komponent, které omezují přístup mezi chráněnou síti a Internetem, nebo mezi různými sítěmi.
- *Host* – Počítač připojený k síti.
- *IP spoofing* – Složitá technika útoků, sestávající z několika složek. Jedná se o útok, kdy se útočící systém vydává za někoho jiného. Podrobně je tento mechanismus popsán daemonem9, routem a infinitym ve 48. vydání magazínu Phrack.
- *Non-repudiation* – Schopnost příjemce prokázat, že odesilatel nějaká data odeslal, i když ten by to popíral.
- *Packet* – Základní komunikační jednotka na Internetu.
- *Packet filtering* – Operace, při níž zařízení selektivně řídí tok dat mezi sítěmi. Filtr umožňuje propouštět nebo blokovat pakety typicky při jejich směrování z jedné sítě do druhé. Filtrování se nastavuje skupinou pravidel, která říkají, jaké pakety (odkud a kam, na jakých portech) mají být propuštěny a které mají být zahozeny.
- *Perimeter network* – Síť mezi chráněnou sítí a externí sítí, představující dodatečný bezpečnostní prvek. Někdy se označuje jako demilitarizovaná zóna, DMZ.
- *Proxy-erver* – Program, který komunikuje s externími servery jménem svých klientů. Klienti se baví s proxy-serverem, ten jejich požadavky tlumočí externím serverům a vrací klientům odpovědi.
- *Superuser* – Označení uživatele root.

Časté otázky

- Je bezpečnější překládat ovladače přímo do jádra, nebo je překládat jako moduly?

  Někdo tvrdí, že je lepší zakázat nahrávání ovladačů zařízení jako modulů, protože útočník by mohl namísto požadovaného ovladače nahrát trójského koně nebo modul jinak ovlivňující bezpečnost systému.

  Abyste ale mohli nahrát modul, musíte být superuživatel. Soubory s modulárními ovladači může rovněž přepsat pouze superuživatel. Útočník by tedy k takovému útoku nejprve musel získat práva superuživatele – a pokud se mu to podaří, může udělat podstatně horší věci, než nahrát nějaký modul.

  Moduly slouží k dynamické podpoře zařízení, která nemusí být používána často. Na serverech nebo firewallech k takovým situacím obvykle nedochází. Proto je logičtější na serveru přeložit potřebné ovladače přímo jako součást jádra. Modulární ovladače jsou navíc pomalejší.

- Proč se nejde ze vzdáleného systému přihlásit jako root?

  Viz část 4.2. Toto omezení je záměrné, aby nebylo možné se aplikací jako *telnet* přihlásit jako superuživatel. Jednalo by se o závažný bezpečnostní problém, protože heslo se po síti přenáší nešifrovaně. Nezapomínejte: útočník má na své straně čas a pomocí automatizovaných programů může číhat na zadávaná hesla.

- Jak zapnu podporu stínových hesel?

  Jako *root* spusťte *pwconv*, tím by měl vzniknout soubor */etc/shadow*, se kterým budou aplikace pracovat. Pokud používáte RedHat 4.2 a vyšší, moduly PAM zajistí automatický přechod od souboru */etc/passwd* ke stínovým heslům bez nutnosti dalších úprav.

  Trocha doplňujících informací: stínová hesla jsou mechanismus uložení hesel v jiném souboru než v běžném */etc/passwd*. Má to několik výhod. První je ta, že stínový soubor hesel, */etc/shadow*, může číst pouze root, zatímco soubor */etc/passwd* musí být čitelný všemi uživateli. Další výhoda je, že administrátor může zapínat a vypínat účty, aniž by status konkrétních účtů byl znám ostatním uživatelům systému.

  Soubor */etc/passwd* v tomto případě slouží k uložení uživatelských jmen a skupin, a používají jej programy jako */bin/ls* k mapování uživatelského ID na uživatelské jméno ve výpisech souborů. Soubor */etc/shadow* pak obsahuje pouze jméno uživatele a jeho heslo, a případně některé další informace, například o aktivitě účtu.

  Pokud vás zajímá problematika zabezpečení hesel, bude vás nejspíš zajímat i problematika volby bezpečných hesel. Můžete k tomu použít modul *pam_cracklib*, který je součástí PAM. Zadaná hesla porovná proti databázi programu Crack a zjistí, jestli je heslo snadno uhodnutelné.

- Jak můžu zapnout rozšíření SSL pro server Apache?

  Můžete také vyzkoušet server ZEDZ (http://www.zedz.net/), na kterém je k dispozici řada připravených balíků, a nachází se mimo USA.

- Jak spravovat více uživatelských účtů a stále zachovat bezpečnost?

  Většina distribucí obsahuje kvalitní nástroje pro nastavování vlastností uživatelských účtů.

  - Programy *pwconv* a *unpwconv* slouží k přechodu na stínová a na normální hesla.
  - Programy *pwck* a *grpck* kontrolují správnost organizace souborů *passwd* a *group*.
  - Programy *useradd*, *usermod* a *userdel* slouží k přidávání, modifikaci a rušení uživatelských účtů. Pro práci se skupinami slouží programy *groupadd*, *groupmod* a *groupdel*.
  - Hesla skupin se nastavují programem *gpasswd*.

  Všechny tyto programy umějí pracovat jak se stínovými, tak i s normálními hesly. Další informace najdete na manuálových stránkách příslušných programů.

- Jak můžu při použití serveru Apache chránit přístup k některým dokumentům heslem?

  Určitě neznáte stránky http://www.apacheweek.org, že?

  Informace o autentikaci uživatelů můžete najít na adrese http://www.apacheweek.com/-features/userauth, další informace o bezpečnosti serveru na adrese http://www.apache.org/docs/misc/security_tips.html.

Závěr

Budete-li sledovat bezpečnostní konference a pravidelně aktualizovat svůj systém, uděláte pro jeho bezpečnost hodně. Pokud navíc budete sledovat logovací soubory a pravidelně spouštět nástroj typu Tripwire, uděláte ještě více.

Zajištění rozumné úrovně bezpečnosti u domácího počítače není příliš obtížné. Pracnější je to u produkčních strojů, ale i zde Linux poslouží jako bezpečná platforma. Vzhledem k povaze vývoje Linuxu se bezpečnostní opravy typicky objevují podstatně dříve než u komerčních systémů, což z Linuxu činí optimální platformu pro aplikace, kde se klade důraz na bezpečnost.

Linux Intranet Server

Originál: http://tldp.org/HOWTO/Installation-HOWTO/

Tento dokument obsahuje základní informace o firewallových systémech a poskytuje některé podrobnější informace o instalaci filtrovacích a proxy firewallů na linuxových systémech. HTML verzi tohoto dokumentu můžete najít na adrese *http://www.grennan.com/Firewall-HOWTO.html*.

Úvod

Původní verzi dokumentu Firewall-HOWTO napsal před delším časem David Rudder a rád bych mu poděkoval za to, že mi umožnil jeho práci aktualizovat.

Dále bych chtěl poděkovat Ianu Goughovi za ochotnou pomoc dyslektickému pisateli.

Firewally získaly velkou oblibu jako základní kámen bezpečnosti Internetu. Stejně jako u celé řady jiných populárních témat i kolem nich panuje velké množství nepochopení. Tento dokument uvádí základní informace o tom, co to firewally jsou a jak je konfigurovat.

Při vytváření tohoto dokumentu bylo použito jádro 2.2.13 a distribuce RedHat 6.1, všechny příklady se tedy vztahují k této distribuci. Pokud byste narazili na nějaké odlišnosti oproti své distribuci, kontaktujte mne a já dokument upravím.

Odezva

Jakákoliv odezva je vítána. **Oznamte mi prosím jakékoliv nepřesnosti, které v tomto textu naleznete.** Jsem jenom člověk a přiznávám se, že dělám chyby. Pokud budete chtít něco opravit, kontaktujte mne. Pokusím se odpovědět na všechnu došlou poštu, jsem však dost zaneprázdněn, takže se nezlobte, pokud se mi to nepodaří.

Moje poštovní adresa je *mark@grennan.com*.

Vysvětlení činnosti firewallů

Firewall v původním slova smyslu představuje zařízení, které má zabránit šíření požáru. V domech jsou firewally tvořeny zdmi, které zcela oddělují jednotlivé části budovy. Firewall v autě je kovová přepážka oddělující motor od prostoru pro cestující.

Internetové firewally mají udržet plameny internetového pekla mimo vaši privátní síť. Mohou také zachovat uživatele vaší privátní sítě čisté a neposkvrněné tím, že je odstaví od všeho pokušení, které Internet nabízí.

První počítačový firewall byl nesměrující unixový počítač s připojením na dvě různé sítě. Jedna síťová karta je připojena na Internet, druhá na privátní síť. Pokud se chcete z privátní sítě dostat na Internet, musíte se nejprve přihlásit na firewall – unixový server. K přístupu na Internet pak využíváte prostředků tohoto systému. Můžete například použít X Windows System, na firewallu

spustit prohlížeč Netscape a zobrazit jej na své stanici. Prohlížeč spuštěný na firewallu bude mít přístup k oběma sítím.

Takovýto duální systém (systém se dvěma síťovými připojeními) je vynikající, pokud můžete důvěřovat **všem** jeho uživatelům. Můžete jednoduše vytvořit linuxový systém a na něm účty pro všechny, kdo potřebují přístup k Internetu. Při takovém uspořádání je jediný počítač s přístupem k Internetu na celé privátní síti právě firewall. Nikdo nemůže nic nahrát na vlastní pracovní stanici. Nejprve musí soubor nahrát na firewall a odtud pak na svou stanici.

Důležitá poznámka: 99 % všech průniků do systému začíná získáním přístupu k napadenému systému. Vzhledem k tomu nemůžu tento typ firewallu doporučit. Navíc je také velice omezující.

Politiky firewallu

Nemůžete věřit tomu, že firewall je všechno, co potřebujete. *Nejprve musíte vytvořit jeho politiku.*

Firewally se používají ke dvěma účelům:

1. Udržet lidi (červy, útočníky) venku
2. Udržet lidi (zaměstnance, děti) uvnitř

Když jsem začal firewall instalovat, překvapilo mě, že společnosti pro niž jsem pracoval šlo více o to „špehovat“ své zaměstnance než o to, zabránit útokům proti jejich síti.

Přinejmenším v zemi kde pracuji (v Oklahomě) má zaměstnavatel právo sledovat telefonní hovory a internetové aktivity zaměstnanců. Stačí, když to oznámí.

Nechápejte mě špatně. Souhlasím s tím, že lidi mají v práci pracovat a ne si hrát. A mám pocit, že pracovní morálka docela upadá. Všiml jsem si ale také toho, že management obvykle nejčastěji porušuje pravidla, která sám zavedl. Zažil jsem, jak byl obyčejný zaměstnanec postihován za to, že si na Internetu zjistil, jaké má nejlepší spojení do práce, zatímco manažer stejného podniku trávil hodiny pracovní doby tím, že na Internetu hledal luxusní restaurace a noční kluby, kam by mohl pozvat prominentní zákazníky.

Osobně se domnívám, že nejlepší způsob jak předcházet takovýmto způsobům zneužívání Internetu spočívá v tom, že záznamy firewallu budou zveřejněny na webu tak, aby je mohl vidět kdokoliv.

Bezpečnostní problematika je nebezpečná věc. Pokud spravujete firewall, hlídejte si záda.

Jak vytvořit bezpečnostní politiku

Četl jsem řadu dlouhých dokumentů o tom, jak bezpečnostní politiku vytvořit. Po letech zkušeností mohu říct, nevěřte jim ani slovo. Vytvoření bezpečnostní politiky je nesmírně jednoduché.

1. Definujte, jaké služby chcete nabízet.
2. Definujte, komu tyto služby budete nabízet.
3. Definujte, jaké služby potřebují jaké skupiny uživatelů.
4. Pro každou skupinu služeb definujte, jak má být zajištěna bezpečnost služby.
5. Všechny ostatní metody přístupu prohlašte za nežádoucí.

Postupem času se politika může komplikovat, nicméně pro začátek se nesnažte zacházet do velkých detailů. Snažte se, ať je jednoduchá a jasná.

Typy firewallů

Existují dva typy firewallů:

1. Filtrovací firewally – které blokují určené síťové pakety.
2. Proxy servery (někdy také označované jako firewally) – které přistupují k síti za vás.

Filtrovací firewally

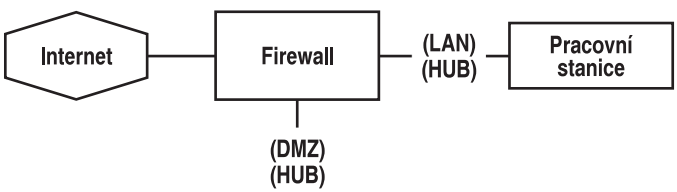
Filtrování paketů je metoda činnosti firewallu, která je vestavěna přímo v jádře Linuxu.

Filtrovací firewall funguje na síťové úrovni. Data mohou přes firewall projít pouze tehdy, pokud to nastavení firewallu povoluje. Veškeré přicházející pakety se filtrují na základě jejich typu, zdrojové adresy, cílové adresy a portu.

Řada směrovačů umožňuje částečně fungovat jako firewall. Filtrovací firewally je naopak možné chápat jako jistý druh směrovače. Vzhledem k tomu musíte pro správné nastavení firewallu přesně rozumět problematice struktury IP paketů.

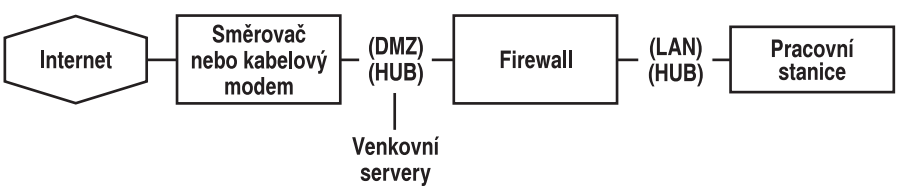
Protože firewall analyzuje a zaznamenává jenom malý objem dat, filtrovací firewally jsou nenáročné na procesor a vyvolávají jen malé síťové zpoždění.

Filtrovací firewally nezajišťují ochranu hesly. Uživatelé se jim nijak neidentifikují. Jedinou identifikací je IP adresa, přiřazená jejich stanici. To může způsobovat problémy v případě, že používáte protokol DHCP (dynamické přidělování IP adres). Je to dáno tím, že pravidla jsou založena na IP



adresách a budete je muset upravovat pokaždé, když dojde ke změně přiřazení IP adres. Nevím, jak tento proces automatizovat.

Filtrovací firewally jsou pro uživatele transparentní. Uživatel nemusí v síťových aplikacích nastavovat žádné speciální volby. U proxy serverů to většinou neplatí.



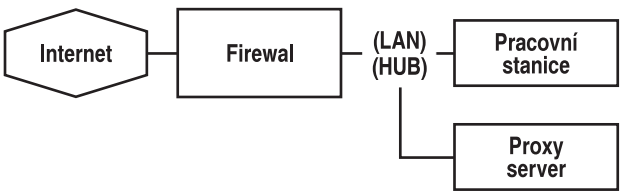
Proxy servery

Proxy servery se často používají k řízení a sledování odchozího provozu. Některé aplikační proxy servery ukládají přenášená data. Tím se snižuje objem přenášených dat a zrychluje se přístup uživatelům, kteří budou chtít stejná data příště. Představují také neoddiskutovatelný doklad toho, co bylo přeneseno.

Existují dva typy proxy serverů:

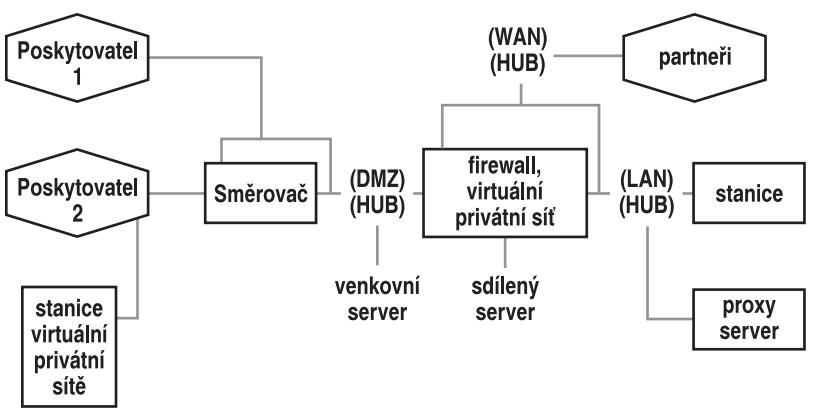
1. Aplikační proxy servery – ty pracují za vás

2. SOCKS proxy servery – ty propojují porty



Aplikační proxy servery

Nejlepším příkladem je osoba, která se telnetem připojí na jiný počítač a z tohoto počítače se telnetem připojuje k nějakému vzdálenému počítači. Při použití aplikačního proxy serveru se tento proces automatizuje. Když se chcete telnetem připojit k nějakému vzdálenému počítači, klient tel-



netu vás nejprve připojí k proxy serveru. Proxy server se pak připojí k tomu počítači, k němuž jste se chtěli připojit, a vrací vám příslušná data.

Protože proxy server zajišťuje veškerou komunikaci, může zaznamenávat vše co dělá(te). U HTTP proxy serverů to například může znamenat zaznamenání každé navštívené adresy. U FTP proxy serverů mohou být zaznamenány všechny přenesené soubory. Tyto proxy servery mohou dokonce z přenášených dokumentů odfiltrovat „neslušná“ slova nebo je mohou kontrolovat na výskyt virů.

Aplikační proxy servery mohou zajišťovat autentikaci uživatelů. Než server provede připojení ke vzdálenému počítači, může uživatele nejprve požádat o přihlášení. Při prohlížení webových stránek to může vypadat tak, že každá navštívená stránka je chráněna heslem.

SOCKS proxy

SOCKS server funguje jako stará telefonní ústředna. Jednoduše propojuje vaše připojení k SOCKS serveru s propojením na vnější počítač.

Většina SOCKS serverů funguje pouze pro TCP spojení. Stejně jako filtrovací firewally nezajišťují autentikaci uživatelů. Mohou však zaznamenávat, kam se uživatelé připojují.

Architektura firewallu

Existuje řada možností jak strukturovat síť při zavedení ochrany firewallem.

Pokud máte pevné připojení k Internetu prostřednictvím směrovače, můžete směrovač přímo připojit k firewallu. Můžete také použít rozbočovač a umožnit tak plný přístup k vašim serverům, které jsou umístěny vně firewallu.

Vytáčené připojení

Můžete být připojeni nějakým vytáčeným připojením jako je ISDN. V takovém případě můžete pomocí třetí síťové karty zajistit filtrované DMZ[1]. Tím umožníte plné internetové služby a přitom je máte odděleny od interní sítě.

Architektura s jedním směrovačem

V případě, že je mezi vámi a Internetem směrovač nebo kabelový modem a pokud směrovač spravujete sami, můžete na něm nastavit přísná filtrační pravidla. Pokud směrovač spravuje poskytovatel internetového připojení, nemusíte mít k nastavení pravidel dostatečná oprávnění. Můžete však požádat poskytovatele, ať vám potřebnou filtraci zavede.

Firewall s proxy serverem

Pokud potřebujete sledovat, co uživatelé vaši sítě navštěvují a síť je poměrně malá, můžete do firewallu integrovat proxy server. Dělají to někteří poskytovatelé internetového připojení a vytvářejí si tak přehledy o zájmech svých klientů, které pak prodávají marketingovým agenturám.

Pokud chcete, můžete proxy server umístit i na lokální síti. V takovém případě je nutné na firewallu nastavit taková pravidla, aby umožnil připojení ke službám, které proxy server poskytuje, právě jenom proxy serveru. Pak se uživatelé mohou dostat na Internet pouze prostřednictvím proxy serveru.

Redundantní internetové připojení

Pokud provozujete službu jako je Yahoo nebo SlashDot, pravděpodobně budete chtít zvýšit spolehlivost vašeho systému použitím redundantních směrovačů a firewallů. (Viz dokument High Availability HOWTO.)

Použitím round-robin DNS kolotoče můžete zajistit přístup k více serverům z jedné URL adresy a budete-li mít více poskytovatelů připojení, směrovačů a firewallů s využitím technologie vysoké dostupnosti, můžete vytvořit službu, která bude téměř 100% spolehlivá.

Konfigurace sítě vám může snadno přerůst přes hlavu. Nezapomínejte prověřovat jakákoliv připojení. K narušení celé privátní sítě stačí uživatel s modemem.

1 Pozn. překladatele: DMZ znamená *demilitarizovanou zónu*. Tímto termínem se označuje část sítě, která je částečně chráněna, není však chráněna úplně. Typicky máte u velké sítě první („hodný“) firewall, který odděluje Internet od demilitarizované zóny. V té se nacházejí veřejně přístupné servery, například webový server a DNS server. Pak následuje druhý („zlý“) firewall a za ním už je vaše kompletně zabezpečená interní síť.

Vytvoření filtrujícího firewallu

Hardwarové požadavky

Filtrující firewall nemá vysoké hardwarové nároky. Nejde téměř o nic víc než o jednoduchý směrovač.

Vše co potřebujete je:

- 486 DX66 s 32 MB paměti
- pevný disk 250 MB (lépe 500 MB)
- síťové připojení (síťovou kartu, sériový port, bezdrátovou přípojku, ...)
- monitor a klávesnice

U některých systémů s využitím konzoly na sériovém portu se můžete obejít i bez monitoru a klávesnice.

Pokud vytváříte proxy server s velkým provozem, použijte co nejlepší vybavení. Jde o to, že pro každého uživatele připojeného k systému se vytváří nový proces. Pokud bude systém současně používat 50 nebo více uživatelů, doporučuji

- Pentium II s 64 MB paměti
- dva gigové disky k ukládání přístupových logů
- dvě síťová připojení
- monitor a klávesnici

Síťová připojení mohou být libovolného typu (síťové karty, ISDN i modemy).

Požadavky na software

Volba jádra

Chcete-li vytvořit filtrující firewall, nepotřebujete žádný speciální software. Linux vše zařídí. V době vzniku tohoto textu používám RedHat 6.1.

Firewall vestavěný v Linuxu se několikrát měnil. Pokud používáte staré jádro Linuxu (verzi 1.0.x nebo starší), přejděte na nové. Tato stará jádra používala software ipfwadm z *http://www.xos.nl/linux/ipfwadm/*, který již není podporován.

Používáte-li jádro 2.2.13 nebo novější, budete používat software ipchaims z adresy *http://www.rustcorp.com/linux/ipchains/*.

Používáte-li nové jádro 2.4, obsahuje nový firewallový nástroj s lepšími funkcemi. Budu o něm brzy hovořit.

Volba proxy serveru

Pokud chcete provozovat proxy server, musíte si nainstalovat některý z následujících balíků:

1. Squid
2. The TIS Firewall Toolkit (FWTK)
3. SOCKS

Squid je vynikající software a využívá linuxovou funkci Transparent Proxy. Popíšeme si instalaci právě tohoto programu.

V době vzniku tohoto textu došlo ke sloučení společností Network Associates a Trusted Information System (TIS). Nové informace o změnách naleznete na jejich webových stránkách. V současné době je možné FWTK stále získat na adrese *http://www.tis.com/research/software/*.

Společnost Trusted Information System nabízí kolekci programů k vybudování firewallu. Pomocí těchto programů můžete vytvořit samostatné démony pro jednotlivé používané služby (WWW, Telnet a podobně).

Příprava linuxového systému

Nainstalujte co nejméně softwaru. Já jsem instalaci zahájil v konfiguraci serveru a pak jsem v souboru /etc/inetd.conf vypnul všechny nepotřebné služby. Pro vyšší bezpečnost byste měli nepotřebné služby odinstalovat.

Protože většina distribucí neobsahuje jádro vhodné pro naše účely, budete si muset přeložit vlastní jádro. Nejlepší je překládat je na jiném počítači než na firewallu. Pokud na firewall nainstalujete překladač C a další utility, po dokončení konfigurace jádra je odstraňte.

Překlad jádra

Začněte s čistou minimální distribucí Linuxu. Čím méně programů je nahraných, tím méně děr, zadních vrátek a/nebo chyb může bezpečnost vašeho systému ohrozit.

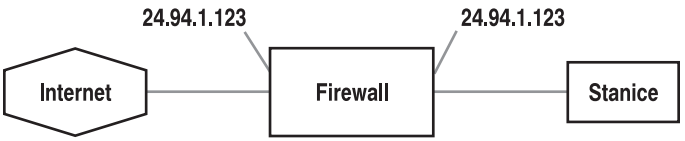
Zvolte si stabilní jádro. Já používám jádro 2.2.13. Další dokumentace je založena na tomto jádře.

Budete muset jádro Linuxu přeložit s potřebnými parametry. Pokud jste ještě nikdy jádro nevytvářeli, přečtěte si dokumenty Kernel HOWTO, Ethernet HOWTO a NET-2 HOWTO.

Následuje seznam síťových nastavení o nichž vím, že fungují. Některá z nich jsou označena otazníkem. Pokud je budete používat, zapněte je také.

K editaci nastavení jádra používám `make menuconfig`.

```
<*> Packet socket
[ ] Kernel/User netlink socket
[*] Network firewalls
[ ] Socket Filtering
<*> Unix domain sockets
[*] TCP/IP networking
[ ] IP: multicasting
[*] IP: advanced router
[ ] IP: kernel level autoconfiguration
[*] IP: firewalling
[?] IP: always defragment (required for masquerading)
[?] IP: transparent proxy support
[?] IP: masquerading
--- Protocol-specific masquerading support will be built as modules.
[?] IP: ICMP masquerading
--- Protocol-specific masquerading support will be built as modules.
[ ] IP: masquerading special modules support
[*] IP: optimize as router not host
< > IP: tunneling
< > IP: GRE tunnels over IP
[?] IP: aliasing support
[*] IP: TCP syncookie support (not enabled per default)
--- (it is safe to leave these untouched)
```




```
< > IP: Reverse ARP
[*] IP: Allow large windows (not recommended if <16Mb of memory)
< > The IPv6 protocol (EXPERIMENTAL)
---
< > The IPX protocol
< > Appletalk DDP
< > CCITT X.25 Packet Layer (EXPERIMENTAL)
< > LAPB Data Link Driver (EXPERIMENTAL)
[ ] Bridging (EXPERIMENTAL)
[ ] 802.2 LLC (EXPERIMENTAL)
< > Acorn Econet/AUN protocols (EXPERIMENTAL)
< > WAN router
[ ] Fast switching (read help!)
[ ] Forwarding between high speed interfaces
[ ] PU is too slow to handle full bandwidth
QoS and/or fair queueing --->
```

Po zavedení všech potřebných nastavení byste měli jádro přeložit, znovu nainstalovat a restartovat systém.

K provedení všech těchto operací najednou používám příkaz

```
make dep;make clean;make bzlilo;make modules;make modules_install;init 6
```

Nastavení dvou síťových karet

Pokud máte v počítači dvě síťové karty, budete možná muset do souboru /etc/lilo.conf přidat příkaz append, kterým specifikujete přerušení a porty obou karet. Já používám následující příkaz append:

```
append="ether=12,0x300,eth0 ether=15,0x340,eth1"
```

Nastavení síťových adres

Nyní se dostáváme k zábavné části konfigurace. Nebudu se zabývat podrobnostmi o vytvoření lokální sítě. K tomu si přečtěte dokument Networking HOWTO.

Naším cílem je na filtrující firewall zavést dvě síťová připojení. Jedno vede na Internet (nezabezpečená strana), druhé na lokální síť (zabezpečená strana).

Musíte rozhodnout následující body:

1. Budete v lokální síti používat skutečné IP adresy, nebo si zavedete vlastní?
2. Poskytuje vám IP adresy poskytovatel, nebo používáte statické?

Protože nechcete aby byl z Internetu přístup na vaši privátní síť, nemusíte v ní používat „opravdové“ IP adresy. Můžete si v ní přidělit jakékoliv adresy chcete. Toto řešení se ovšem nedoporučuje. Pokud by totiž data „utekla“ mimo vaši síť, byla by směrována na nějaký úplně jiný systém.

Určité rozsahy IP adres jsou určeny pro přiřazení v privátních sítích. Jednou z rezervovaných oblastí je 192.168.1.*xxx*, kterou budeme používat v našich příkladech.

Aby celý systém fungoval, musíte zavést IP maškarádu. Tento proces zajistí, že firewall bude pakety předávat a překládat jejich adresy na „opravdové" IP adresy, které mohou po Internetu cestovat.

Použitím rezervovaných nesměrovatelných adres zvyšujete bezpečnost vaší sítě. Internetové směrovače totiž pakety s těmito adresami nepropouštějí.

V tomto okamžiku byste si měli přečíst dokument IP Masquerading HOWTO.

Síťová karta na straně Internetu musí mít přiřazenu „opravdovou" IP adresu. Tuto adresu můžete mít trvale přidělenu (statická adresa), nebo ji můžete dostávat přidělovánu vždy při připojení k síti protokolem PPP.

Síťové kartě na straně lokální sítě přiřadíte nějakou interní adresu, například 192.168.1.1. Tuto adresu budete používat jako adresu brány. Všem ostatním počítačům na chráněné lokální síti přiřadíte další adresy z oblasti 192.168.1.*xxx* (tedy 192.168.1.2 až 192.168.1.254).

Já používám RedHat Linux. Pro nastavení sítě v době startu systému jsem v adresáři /etc/sysconfig/network-scripts přidal soubor ifcfg-eth1. V tomto adresáři můžete najít také soubory ifcfg-ppp0 nebo ifcfg-tr0. Jsou pojmenovány podle používaného typu připojení.

Jako příklad uvádím soubor ifcfg-eth1 konfigurující druhou ethernetovou kartu:

```
DEVICE=eth1
IPADDR=192.168.1.1
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
GATEWAY=24.94.1.123
ONBOOT=yes
```

Pokud používáte vytáčené připojení, hledejte soubory ifcfg-ppp0 a chat-ppp0. Ty nastavují připojení protokolem PPP.

Tento soubor může vypadat například takto:

```
DEVICE="ppp0"
ONBOOT="yes"
USERCTL="no"
MODEMPORT="/dev/modem"
LINESPEED="115200"
PERSIST="yes"
DEFABORT="yes"
DEBUG="yes"
INITSTRING="ATZ"
DEFROUTE="yes"
HARDFLOWCTL="yes"
ESCAPECHARS="no"
PPPOPTIONS=""
PAPNAME="LoginID"
REMIP=""
NETMASK=""
IPADDR=""
MRU=""
MTU=""
```

```
DISCONNECTTIMEOUT=""
RETRYTIMEOUT="5"
BOOTPROTO="none"
```

Testování sítě

Začněte příkazy ifconfig a route. Pokud používáte dvě síťové karty, měl by ifconfig vypadat přibližně takto:

```
#ifconfig
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:3924 Metric:1
        RX packets:1620 errors:0 dropped:0 overruns:0
        TX packets:1620 errors:0 dropped:0 overruns:0
        collisions:0 txqueuelan:0

eth0    Link encap:10Mbps Ethernet HWaddr 00:00:09:85:AC:55
        inet addr:24.94.1.123 Bcast:24.94.1.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:1000 errors:0 dropped:0 overruns:0
        TX packets:1100 errors:0 dropped:0 overruns:0
        collisions:0 txqueuelan:0
        Interrupt:12 Base address:0x310

eth1    Link encap:10Mbps Ethernet HWaddr 00:00:09:80:1E:D7
        inet addr:192.168.1.1 Bcast:192.168.1.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:1110 errors:0 dropped:0 overruns:0
        TX packets:1111 errors:0 dropped:0 overruns:0
        collisions:0 txqueuelan:0
        Interrupt:15 Base address:0x350
```

Směrovací tabulka by měla vypadat asi takto.

```
#route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
24.94.1.0       *               255.255.255.0   U     1500   0       15 eth0
192.168.1.0     *               255.255.255.0   U     1500   0        0 eth1
127.0.0.0       *               255.0.0.0       U     3584   0        2 lo
default         24.94.1.123     *               UG    1500   0       72 eth0
```

Poznámka: 24.94.1.0 je internetová strana tohoto firewallu, 192.168.1.0 strana privátní sítě.

Nejprve byste měli ověřit, zda lze ze všech stanic na lokální síti pingnout na interní adresu firewallu (v našem případě 192.168.1.1). Pokud ne, přečtěte si dokument NET HOWTO a dejte síť do pořádku.

Pak zkuste z firewallu pingnout na nějaký internetový systém. Já používám k testování adresu *www.internic.net*. Pokud to nefunguje, zkuste server vašeho poskytovatele. Pokud nefunguje ani to, nemáte v pořádku připojení k Internetu. Z firewallu byste měli být schopni připojit se kamko-

liv na Internetu. Zkontrolujte nastavení výchozí brány. Pokud používáte vytáčené připojení, zkontrolujte, zda máte správné ID a heslo. Znovu si přečtěte dokument NET HOWTO a zkuste to znovu.

Dále zkuste z vnitřního počítače pingnout na vnější adresu firewallu (24.94.1.123). To by nemělo jít. Pokud to funguje, máte nastavenou maškarádou nebo IP forwarding nebo nějaká filtrační pravidla. Všechno vypněte a zkuste to znovu. Musíte mít ověřeno, že filtrace funguje.

Pro jádra novější než 2.1.102 můžete použít příkaz

```
echo "0" > /proc/sys/net/ipv4/ip_forward
```

Pokud používáte starší jádro (**proč?**), musíte je znovu přeložit s vypnutím funkce IP forwarding. (Lepší je aktualizovat jádro.)

Znovu zkuste ping na vnější adresu firewallu (24.94.1.123). Nemělo by to jít.

Nyní zapněte IP forwarding a/nebo maskování. Z kteréhokoliv systému na lokální síti byste měli dopingnout kamkoliv na Internet.

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Poznámka: Pokud na lokální síti používáte „opravdové“ IP adresy (ne adresy ze skupiny 192.168.1.* apod.) a nemůžete se dopingnout na Internet, ale *můžete* se dopingnout na vnější stranu firewallu, ověřte si, zda váš poskytovatel správně směruje pakety pro vaše privátní adresy.

Tento test můžete provést s využitím někoho jiného na Internetu (například kamaráda s jiným poskytovatelem), který vyzkouší traceroute na vaši síť. Pokud se trasování zastaví na směrovači poskytovatele, pak nezajišťuje směrování vašeho provozu.

Funguje to? Výborně. Ta složitá část je hotova.

Zabezpečení firewallu

Firewall není k ničemu, pokud běží na napadnutelném systému. „Zlí hoši“ mohou získat přístup nějakou nefirewallovou službou a změní konfiguraci firewallu podle svých potřeb. Musíte vypnout všechny nepotřebné služby.

Podívejte se do souboru /etc/inetd.conf. Tento soubor slouží ke konfiguraci superserveru inetd. Řídí celou řadu serverových démonů, které spouští po přijetí požadavku na různé známé porty.

Pokud je máte zapnuty, měli byste vypnout služby echo, discard, daytime, chargen, ftp, gopher, shell, login, exec, talk, ntalk, pop-2, pop-3, netstat, systat, tftp, bootp, finger, cfinger, time, swat a linuxconfig.

Službu vypnete tak, že na první pozici příslušného řádku přidáte znak #. Až změníte vše potřebné, pošlete procesu inetd signál HUP zadáním příkazu `kill -HUP <pid>`, kde <pid> je číslo procesu inetd. Tím proces znovu načte svůj konfigurační soubor (inetd.conf) a restartuje se, aniž by bylo nutné restartovat celý systém.

Otestujte nové nastavení telnetem na port 15 (netstat) firewallu. Pokud dostanete nějakou odezvu, nepodařilo se vám služby vypnout:

```
telnet localhost 15
```

Dále můžete vytvořit soubor /etc/nologin. Napište do něj nějaký text jako třeba *Běž pryč*. Pokud takový soubor existuje, nedovolí proces login přihlášení uživatele. Zobrazí pouze obsah tohoto souboru a odmítne přihlášení. Přihlásit se může jenom root.

Dále můžete upravit soubor /etc/securetty. Pokud se má přihlásit uživatel root, musí se přihlásit na některém z tty uvedených v tomto souboru. Jiné zamítnuté pokusy zaznamenává služba syslog. Zavedete-li obě tyto ochrany, jediná možnost přihlášení k firewallu bude z konzoly jako uživatel root.

Nikdy se službou telnet nikam nepřihlašujte jako root. Pokud potřebujete vzdálený přístup, použijte službu SSH (Secure Shell). Nejlepší bude telnet úplně vypnout.

Pokud jste skutečně paranoidní, použijte lids (Linux Intrusion Detect System). Jedná se o doplněk jádra sloužící k detekci průniků, který dokáže chránit důležité systémové soubory. Pokud pracuje, nemůže nikdo (ani root) změnit chráněné soubory a adresáře (a jejich podadresáře). Chcete-li chráněné soubory změnit, musíte systém restartovat s nastavením LILO security=1 (v tomto případě doporučuji zároveň systém startovat v jednouživatelském režimu).

Nastavení IP filtrování (IPFWADM)

Pokud používáte jádro 2.1.102 nebo novější, přejděte na další část IPCHAINS.

Ve starších jádrech je IP forwarding standardně zapnutý. Proto byste měli začít zakázáním všeho a zrušením všech pravidel, která mohou platit od minula. Následující fragment kódu patří do spouštěcího síťového skriptu (/etc/rc.d/init.d/network)

```
#
# setup IP packet Accounting and Forwarding
#
# Forwarding
#
# By default DENY all services
ipfwadm -F -p deny
# Flush all commands
ipfwadm -F -f
ipfwadm -I -f
ipfwadm -O -f
```

Teď máme dokonalý firewall. Nepropustí vůbec nic.

Nyní vytvoříme soubor /etc/rc.d/rc.firewall. Tento skript povolí poštu, web a DNS provoz.

```
#! /bin/sh
#
# rc.firewall
#
# Source function library.
. /etc/rc.d/init.d/functions
# Get config.
. /etc/sysconfig/network
# Check that networking is up.
if [ ${NETWORKING} = "no" ]
then
exit 0
fi
case "$1" in
start)
echo -n "Starting Firewall Services: "
# Allow email to got to the server
```

```
/sbin/ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/0 1024:65535 -D 192.1.2.10 25
# Allow email connections to outside email servers
/sbin/ipfwadm -F -a accept -b -P tcp -S 192.1.2.10 25 -D 0.0.0.0/0 1024:65535
# Allow Web connections to your Web Server
/sbin/ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/0 1024:65535 -D 192.1.2.11 80
# Allow Web connections to outside Web Server
/sbin/ipfwadm -F -a accept -b -P tcp -S 192.1.2.* 80 -D 0.0.0.0/0 1024:65535
# Allow DNS traffic
/sbin/ipfwadm -F -a accept -b -P udp -S 0.0.0.0/0 53 -D 192.1.2.0/24
;;
stop)
echo -n "Stopping Firewall Services: "
ipfwadm -F -p deny
;;
status)
echo -n "Now do you show firewall stats?"
;;
restart|reload)
$0 stop
$0 start
;;
*)
echo "Usage: firewall {start|stop|status|restart|reload}"
exit 1
esac
```

Poznámka: V tomto skriptu povolujeme poštovní (SMTP) server na počítači 192.1.2.10, který musí umět přijímat a odesílat na portu 25. Webový server běží na počítači 192.1.2.11. Kdokoliv na lokální síti má přístup k externím webovým a DNS serverům.

Toto řešení není úplně bezpečné. Port 80 nemusí být používán jako port webového serveru, takže šikovný hacker jej může využít k tomu, že si jeho prostřednictvím vytvoří přes náš firewall virtuální privátní síť. Ochrana spočívá v instalaci webového proxy serveru a na firewallu povolíme průchod pouze tomuto serveru. Uživatelé na lokální síti budou muset k externím webovým serverům přistupovat přes proxy server.

Možná budete chtít provoz přes firewall zaznamenávat. Následující skript bude zaznamenávat všechny pakety. Můžete jej lehce upravit a sledovat pouze pakety z určitého systému.

```
# Flush the current accounting rules
ipfwadm -A -f
# Accounting
/sbin/ipfwadm -A -f
/sbin/ipfwadm -A out -i -S 192.1.2.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -A out -i -S 0.0.0.0/0 -D 192.1.2.0/24
/sbin/ipfwadm -A in -i -S 192.1.2.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -A in -i -S 0.0.0.0/0 -D 192.1.2.0/24
```

Pokud nepotřebujete nic více než filtrující firewall, můžete tímto skončit. Otestujte jej a užijte si ho.

Nastavení IP filtrování (IPCHAINS)

Linux ipchains je přepsaná verze linuxového firewalového kódu IPv4 a kódu ipfwadm, který je sám přepsanou verzí BSD kódu ipfw – aspoň myslím. Je potřebný k administraci paketových IP filtrů v linuxovém jádře 2.1.102 a vyšších.

Původní kód nedokázal pracovat s fragmenty, používal 32bitová počitadla (přinejmenším na platformě Intel), neumožňoval specifikovat jiné protokoly než TCP, UDP a ICMP, nedovoloval atomické provedení větších změn, neumožňoval specifikaci inverzních pravidel, nebyl úplně spolehlivý a obtížně se spravoval (takže byl náchylný na chyby uživatele). Aspoň to tvrdí autoři.

Nehodlám se pouštět do podrobnějšího popisu toho, jak nastavit firewall na bázi IPChains, protože k tomu slouží **vynikající** HOWTO dokument *http://www.rustcorp/linux/ipchains/HOWTO.html*. Skončilo by to tak, že bych ho musel opsat. Proto uvádím jenom základy.

Už podle názvu pracujete s řetězci. Na začátku máte tři vestavěné řetězce – vstupní, výstupní a forwardovací, které nemůžete zrušit. Kromě toho můžete vytvářet vlastní řetězce. Řetězec představuje skupinu pravidel, která můžete doplňovat a rušit.

S celými řetězci můžete provádět následující operace:

1. Vytvořit nový řetězec (-N).
2. Vymazat prázdný řetězec (-X).
3. Změnit politiku vestavěného řetězce (-P).
4. Vypsat pravidla v řetězci (-L).
5. Vymazat pravidla v řetězci (-F).
6. Vynulovat paketová a bajtová počitadla všech pravidel v řetězci (-Z).

S jednotlivými pravidly v řetězci můžete provádět další operace:

1. Připojit k řetězci nové pravidlo (-A).
2. Vložit nové pravidlo na určité místo řetězce (-I).
3. Nahradit pravidlo na určitém místě řetězce (-R).
4. Smazat pravidlo na určitém místě řetězce (-D).
5. Vymazat první vyhovující pravidlo v řetězci (-D).

Kromě toho existuje i několik operací týkajících se maškarádování:

1. Vypsání momentálně maškarádovaných spojení (-M -L)
2. Nastavení timeoutu maškarádovaných (-M -S)

Při změně pravidel firewallu mohou vzniknout problémy s načasováním. Pokud byste nebyli opatrní, mohlo by se stát, že někdy uprostřed provádění změn umožníte průchod nechtěným paketům. Velmi jednoduché řešení vypadá takto:

```
# ipchains -I input 1 -j DENY
# ipchains -I output 1 -j DENY
# ipchains -I forward 1 -j DENY
```

... teď provedete změny ...

```
# ipchains -D input 1
# ipchains -D output 1
```

```
# ipchains -D forward 1
#
```

Tím dojde k odmítnutí všech paketů, které přijdou v době provádění změn.

A takto bude vypadat nastavení výše zavedených pravidel v programu IPChains.

```
#!/bin/sh
#
# rc.firewall
#
## Flush everything, start from scratch
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
## Redirect for HTTP Transparent Proxy
#$IPCHAINS -A input -p tcp -s 192.1.2.0/24 -d 0.0.0.0/0 80 -j REDIRECT 8080
## Create your own chain
/sbin/ipchains -N my-chain
# Allow email to got to the server
/sbin/ipchains -A my-chain -s 0.0.0.0/0 smtp -d 192.1.2.10 1024:-j ACCEPT
# Allow email connections to outside email servers
/sbin/ipchains -A my-chain -s 192.1.2.10 -d 0.0.0.0/0 smtp -j ACCEPT
# Allow Web connections to your Web Server
/sbin/ipchains -A my-chain -s 0.0.0.0/0 www -d 192.1.2.11 1024: -j ACCEPT
# Allow Web connections to outside Web Server
/sbin/ipchains -A my-chain -s 192.1.2.0/24 1024: -d 0.0.0.0/0 www -j ACCEPT
# Allow DNS traffic
/sbin/ipchains -A my-chain -p UDP -s 0.0.0.0/0 dns -d 192.1.2.0/24 -j ACCEPT
## If you are using masquerading
# don't masq internal-internal traffic
/sbin/ipchains -A forward -s 192.1.2.0/24 -d 192.1.2.0/24 -j ACCEPT
# don't masq external interface direct
/sbin/ipchains -A forward -s 24.94.1.0/24 -d 0.0.0.0/0 -j ACCEPT
# masquerade all internal IP's going outside
/sbin/ipchains -A forward -s 192.1.2.0/24 -d 0.0.0.0/0 -j MASQ
## Deny everything else
/sbin/ipchains -P my-chain input DENY
```

Tím jsme ovšem neskončili. Zatím nemáte nastavený dokonalý firewall a jsem si jistý, že budete chtít povolit i jiné služby. Opět doporučuji přečíst si dokument IPCHAINS-HOWTO.

Instalace transparentního SQUID proxy serveru

Proxy server SQUID je dostupný na adrese *http://squid.nlanr.net/*.

Program je dostupný pro distribuce RedHat a Debian. Pokud to jde, použijte jednu z nich.

Instalace proxy serveru TIS

Získání programu

Program TIS FWTK je dostupný na adrese *http://ww.tis.com/research/software*.

Neudělejte stejnou chybu, jakou jsem udělal já. Když si soubory z TIS stáhnete, **přečtěte si dokument README**. Program TIS FWTK je schován ve skrytém adresáři jejich serveru.

TIS požaduje, abyste si přečetli licenční smlouvu na adrese *http://www.tis.com/research/software/fwtk_readme.html* a poté poslali mail na adresu *fwtk-request@tislabs.com* s jediným slovem **accepted** v těle zprávy. Předmět zprávy nemusí být vyplněn. Jejich systém vám pak pošle zpátky název skrytého adresáře (platný 12 hodin), odkud si budete moci program stáhnout.

V době vzniku tohoto textu je poslední verzí programu FWTK verze 2.1.

Instalace TIS FWTK

Spusťte `make install`.

Standardní instalační adresář je /usr/local/etc. Můžete jej změnit na nějaký bezpečnější (já jsem to neudělal). Rozhodl jsem se nastavit práva na tento adresář na 'chmod 700'.

Poslední co zbývá je konfigurace firewallu.

Konfigurace TIS FWTK

Teď začíná ta správná legrace. Musíme systém naučit volat nově nainstalované služby a musíme vytvořit tabulky, které je řídí.

Nehodlám zde opisovat manuál k programu TIS FWTK. Ukážu vám nastavení, které mi funguje a upozorním vás na problémy, na které jsem narazil společně s popisem toho, jak jsem je vyřešil.

Řízení firewallu obstarávají tři soubory:

- **/etc/services** – říká systému, na kterých portech služby pracují
- **/etc/inetd.conf** – říká programu inetd, jakou službu má volat, když někdo požaduje určitou službu
- **/usr/local/etc/netperm-table** – říká službám FWTK, kdo má jaké služby povoleny

Při konfiguraci FWTK budete tyto soubory měnit postupně odspodu nahoru. Změna souboru services bez předchozího správného nastavení v souborech inetd.conf a netperm-table může způsobit nedostupnost systému.

Soubor netperm-table

Tento soubor určuje, kdo má přístup ke službám TIS FWTK. Na provoz procházející firewallem musíte pohlížet z obou stran. Uživatelé vně chráněné sítě by se měli před získáním přístupu identifikovat, uživatelé uvnitř sítě mohou mít povolen průchod přímo.

Aby byla možná identifikace uživatelů, používá firewall program nazvaný **authsrv**, který udržuje databázi uživatelských ID a hesel. Autentikační část souboru netperm-table udává, kde je tato databáze uložena a kdo k ní má povolen přístup.

Měl jsem problémy zakázat přístup k této službě. Všimněte si, že na řádku permit-hosts mám nastavenu hodnotu '*', takže přístup má kdokoliv. Správné nastavení (pokud se vám podaří je zprovoznit) by mělo být `authsrv: permit-hosts localhost`.

```
#
# Proxy configuration table
#
# Authentication server and client rules
authsrv: database /usr/local/etc/fw-authdb
authsrv: permit-hosts *
```

```
authsrv: badsleep 1200
authsrv: nobogus true
# Client Applications using the Authentication server
*: authserver 127.0.0.1 114
```

K inicializaci databáze musíte být root a spustit program **./authsrv** v adresáři /var/local/etc, čímž se vytvoří záznam o administrátorovi. Takto vypadá příklad vytvoření záznamu.

V dokumentaci k FWTK zjistíte, jak vytvářet uživatele a skupiny.

```
#
# authsrv
authsrv# list
authsrv# adduser admin "Auth DB admin"
ok - user added initially disabled
authsrv# ena admin
enabled
authsrv# proto admin pass
changed
authsrv# pass admin "plugh"
Password changed.
authsrv# superwiz admin
set wizard
authsrv# list
Report for users in database
user group longname ok? proto last
------ ------ ------------------ ----- ------ -----
admin Auth DB admin ena passw never
authsrv# display admin
Report for user admin (Auth DB admin)
Authentication protocol: password
Flags: WIZARD
authsrv# ^D
EOT
#
```

Nastavení telnetové brány (tn-gw) je nejjednodušší a je to první věc, kterou byste měli začít.

V uvedeném příkladu povoluji počítačům z interní sítě přístup ven bez nutnosti autentikace (`permit-hosts 192.1.2.* -passok`). Ostatní uživatelé se však musejí autentikovat (`permit-hosts * -auth`).

Kromě toho jsem jednomu systému (192.1.2.202) povolil přímý přístup k firewallu bez nutnosti procházet přes firewall. Zajišťují to dva řádky inetacl-in.telnetd. Později si uvedeme, jak se těmto řádkům říká.

Timeout pro službu telnet by měl být krátký.

```
# pravidla telnetové brány:
tn-gw: denial-msg /usr/local/etc/tn-deny.txt
tn-gw: welcome-msg /usr/local/etc/tn-welcome.txt
tn-gw: help-msg /usr/local/etc/tn-help.txt
tn-gw: timeout 90
tn-gw: permit-hosts 192.1.2.* -passok -xok
tn-gw: permit-hosts * -auth
# pouze administrátor má na firewall přímý přístup telnetem na portu 24
```

```
netacl-in.telnetd: permit-hosts 192.1.2.202 -exec /usr/sbin/in.telnetd
```

r-příkazy fungují stejně jako telnet.

```
# pravidla rlogin brány:
rlogin-gw: denial-msg /usr/local/etc/rlogin-deny.txt
rlogin-gw: welcome-msg /usr/local/etc/rlogin-welcome.txt
rlogin-gw: help-msg /usr/local/etc/rlogin-help.txt
rlogin-gw: timeout 90
rlogin-gw: permit-hosts 192.1.2.* -passok -xok
rlogin-gw: permit-hosts * -auth -xok
# pouze administrátor má na firewall přímý přístup rloginem
netacl-rlogind: permit-hosts 192.1.2.202 -exec /usr/libexec/rlogind -a
```

Nikomu byste neměli umožnit přímý přístup k firewallu, což platí i pro službu FTP – proto na firewall neinstalujte FTP server.

I zde řádek permit-hosts umožňuje komukoliv z chráněné sítě přímý přístup na Internet, všichni ostatní se musejí autentikovat. Kromě toho je nastaveno zaznamenávání všech přijatých a odeslaných souborů (`-log {retr stor}`).

Nastavení timeoutu říká, za jak dlouho budou uzavřena chybná připojení i to, jak dlouho bude spojení udržováno bez aktivity.

```
# pravidla ftp brány:
ftp-gw: denial-msg /usr/local/etc/ftp-deny.txt
ftp-gw: welcome-msg /usr/local/etc/ftp-welcome.txt
ftp-gw: help-msg /usr/local/etc/ftp-help.txt
ftp-gw: timeout 300
ftp-gw: permit-hosts 192.1.2.* -log { retr stor }
ftp-gw: permit-hosts * -authall -log { retr stor }
```

FTP přenosy zprostředkované přes web, gopher a prohlížeče jsou řízeny nastavením http-gw. První dva řádky nastavují adresář pro ukládání dokumentů, které firewallem přicházejí. Tyto soubory vlastní uživatel root a jsou uloženy v adresáři, kam má přístup pouze root.

Webová připojení by měla být krátká. Délka spojení určuje, jak dlouho bude neplatné spojení trvat.

```
# pravidla www a gopher brány:
http-gw: userid root
http-gw: directory /jail
http-gw: timeout 90
http-gw: default-httpd www.afs.net
http-gw: hosts 192.1.2.* -log { read write ftp }
http-gw: deny-hosts *
```

ssl-gw je jednoduchá brána propouštějící cokoliv. Pozor na ni. V tomto případě povoluji komukoliv z chráněné sítě připojit se na jakýkoliv server vně sítě s výjimkou adres 127.0.0.* a 192.1.1.* a to pouze na portech 443 až 563. Tyto porty jsou používány pro službu SSL.

```
# pravidla ssl brány:
ssl-gw: timeout 300
ssl-gw: hosts 192.1.2.* -dest { !127.0.0.* !192.1.1.* *:443:563 }
ssl-gw: deny-hosts *
```

Další příklad ukazuje, jak pomocí plug-gw povolit přístup na server news. Povolujeme komukoliv z interní sítě připojení pouze na jediný server a na jeho konkrétní port.

Následující řádek povoluje přenos dat zpět na chráněnou síť.

Protože většina klientů počítá s tím, že zůstávají připojeni v době, kdy si uživatel zprávy čte, timeout této služby by měl být nastavený na dlouhou dobu.

```
# brána NetNews
plug-gw: timeout 3600
plug-gw: port nntp 192.1.2.* -plug-to 24.94.1.22 -port nntp
plug-gw: port nntp 24.94.1.22 -plug-to 192.1.2.* -port nntp
```

Brána služby finger je jednoduchá. Kdokoliv uvnitř chráněné sítě se nejprve musí autentikovat a potom mu povolíme použít program finger na firewallu. Kdokoliv jiný jenom obdrží nastavené hlášení.

```
# povolení služby finger
netacl-fingerd: permit-hosts 192.1.2.* -exec /usr/libexec/fingerd
netacl-fingerd: permit-hosts * -exec /bin/cat /usr/local/etc/finger.txt
```

Nenastavoval jsem poštovní služby a službu X-windows, proto neuvádím příklady konfigurace. Pokud jste někdo tyto služby konfigurovali, pošlete mi prosím mail.

Soubor /etc/services

V tomto souboru všechno začíná. Když se klient připojí k firewallu, připojí se na nějaký známý port (menší než 1024). Například služba telnet se připojuje na port 23. Démon inetd toto spojení zaregistruje a v souboru /etc/services zjistí název příslušné služby. Pak volá program, který je této službě přiřazen v souboru /etc/inetd.conf.

Některé služby které vytváříme normálně v souboru /etc/services nejsou. Můžete je přiřadit kterémukoliv portu chcete. Například administrátorskou službu telnet (telnet-a) jsem přiřadil na port 24. Můžete ji přiřadit klidně třeba na port 2323. Když se administrátor (tedy *vy*) bude chtít připojit k firewallu, musí provést telnet na port 24 a ne na 23 a pokud máte soubor netperm-table nastaven tak jako já, budete se moci připojit pouze z jednoho konkrétního počítače uvnitř chráněné sítě.

```
telnet-a  24/tcp
ftp-gw    21/tcp            # this named changed
auth      113/tcp   ident   # User Verification
ssl-gw    443/tcp
```

SOCKS proxy server

Instalace proxy serveru

Proxy server SOCKS můžete získat na adrese *http://www.socks.nec.com/*.

Tento soubor dekomprimujte a rozbalte do nějakého adresáře na vašem systému a držte se pokynů pro jeho sestavení. Já jsem s tím měl různé problémy. Zkontrolujte si, zda máte v pořádku soubory Makefile.

Důležitá věc je to, že tento proxy server je nutné přidat do souboru /etc/inetd.conf. Musíte přidat následující řádek:

```
socks  stream tcp nowait nobody /usr/local/etc/sockd sockd
```

Pak se bude server spouštět, když bude potřeba.

Konfigurace proxy serveru

Program SOCKS potřebuje dva samostatné konfigurační soubory. Jeden říká, jaké přístupy jsou povoleny, druhý říká, kam jednotlivé požadavky směrovat. Přístupový soubor je umístěn na serveru. Směrovací soubor musí být na každé unixové stanici. Dosovské počítače a zřejmě i Macy si zajišťují vlastní směrování.

Přístupový soubor

V programu socks4.2Beta se tento soubor jmenuje „sockd.conf“. Měl by obsahovat dva řádky, jeden typu permit, druhý typu deny. Každý řádek obsahuje tři údaje:

- Identifikátor (permit nebo deny)
- IP adresa
- Modifikátor adresy

Identifikátor je buď permit nebo deny. Měli byste mít oba dva typy řádku.

IP adresa obsahuje čtyřbajtovou adresu v klasické IP notaci, například 192.168.1.0.

Modifikátor adresy je opět klasická čtyřbajtová IP adresa. Funguje podobně jako maska sítě. Chápejte uvedené číslo jako 32bitové binární, skládající se z jedniček a nul. Pokud je nějaký bit jedničkový, pak musí odpovídající bit v kontrolované adrese odpovídat hodnotě stejného bitu v IP adrese v řádku. Řekněme, že máme řádek.

```
permit 192.168.1.23 255.255.255.255
```

Tento řádek povolí pouze adresy, v nichž všechny bity odpovídají adrese 192.168.1.23, tedy pouze právě adresu 192.168.1.23. Řádek

```
permit 192.168.1.0 255.255.255.0
```

povolí všechny adresy v rozsahu 192.168.1.0 až 192.168.1.255, tedy celou jednu třídu C adres. Rozhodně byste neměli zadat řádek

```
permit 192.168.1.0 0.0.0.0
```

který povolí jakoukoliv adresu.

Nejprve tedy povolte všechny potřebné adresy a pak zakažte zbytek. Pokud chcete povolit všechny počítače v doméně 192.168.1.*, bude fungovat nastavení

```
permit 192.168.1.0 255.255.255.0
deny 0.0.0.0 0.0.0.0
```

Všimněte si prvního údaje 0.0.0.0 na řádku deny. Máme-li nastaven modifikátor 0.0.0.0, na adrese vůbec nezáleží a pravidlu vyhovuje vše. Obvykle se zadává právě adresa 0.0.0.0, protože ta se snadno zapisuje.

V souboru samozřejmě můžete mít více řádků jednotlivých typů.

Přístup je možné povolovat nebo zakazovat i určitým uživatelům. Dělá se to pomocí autentikace ident. Ne všechny systémy však ident podporují, například Trumpet Winsock, proto o tomto typu autentikace nebudeme mluvit. Potřebné informace získáte v dokumentaci k programu SOCKS.

Směrovací soubor

Směrovací soubor v programu SOCKS se nešťastně jmenuje „socks.conf". Říkám nešťastně, protože jeho název je natolik blízký přístupovému souboru, že je jednoduché si tyto soubory splést.

Směrovací soubor říká klientům programu SOCKS kdy tuto službu používat a kdy ne. V našem příkladu například nebudeme potřebovat službu SOCKS pro komunikaci s počítačem 192.168.1.1 – firewallem. S tím máme přímé propojení Ethernetem. Samozřejmě SOCKS nepotřebujeme pro přístup na adresu 127.0.0.1, protože sami se sebou se můžete bavit jak chcete. Soubor obsahuje tři typy položek:

- deny
- direct
- sockd

Údaj deny říká, které požadavky se mají zamítnout. Obsahuje stejné údaje jako v souboru sockd.conf, tedy identifikátor, adresu a modifikátor. Protože je řízení přístupu obsluhováno i souborem sockd.conf, nastavuje se modifikátor na 0.0.0.0. Pokud si chcete zakázat přístup kamkoliv, můžete to udělat zde.

Údaj direct říká, pro které adresy se nemá služba SOCKS používat. Zde by měly být uvedeny všechny adresy, které je možné používat bez proxy serveru. I zde jsou tři údaje, identifikátor, adresa a modifikátor. V našem příkladu použijeme

```
direct 192.168.1.0 255.255.255.0
```

Tím povolujeme přímý přístup k čemukoliv na chráněné síti.

Údaj sockd definuje počítač, na kterém běží server SOCKS. Syntaxe je následující:

```
sockd @=<seznamserverů> <IP adresa> <modifikátor>
```

Všimněte si údaje @=. Ten nám umožňuje definovat seznam proxy serverů. V našem příkladu používáme pouze jediný proxy server. Můžete jich však mít více například pro zvýšení přenosové kapacity nebo jako zálohu pro případ výpadku.

IP adresa a modifikátor fungují stejně jako v ostatních příkladech. Nastavujete zde, kam mohou které adresy jít.

DNS za firewallem

Nastavení služby Domain Name Service za firewallem je poměrně jednoduché. Stačí pouze nainstalovat DNS server na firewallu. Pak každému počítači za firewallem řeknete, aby používal tento server.

Práce s proxy serverem

Unix

Aby mohla aplikace pracovat s proxy serverem, musí být „sockifikovaná". Budete potřebovat dva různé telnetové klienty, jednoho pro přímou komunikaci, druhého pro komunikaci přes proxy server. Program SOCKS se dodává s popisem, jak programy sockifikovat i s některými připravenými běžný-

mi klienty. Pokud použijete sockifikovanou verzi pro přímý přístup, SOCKS vás automaticky přepne na přímou verzi programu. Proto je nutné všechny programy na chráněné síti přejmenovat a nahradit je sockifikovanými verzemi. Z programu „finger" tak dostaneme „finger.orig", z „telnet" bude „telnet.orig" a podobně. O těchto souborech musíte program SOCKS informovat v souboru include/socks.h.

Některé programy zvládají směrování a komunikaci se SOCKS serverem samy. Umí to například Netscape. Chcete-li používat proxy server v programu Netscape, stačí zadat adresu proxy serveru (v našem případě 192.168.1.1) do údaje SOCK v konfiguračním poli Proxies. U každé aplikace bude nutný nějaký typ nastavení bez ohledu na to, jak s proxy serverem funguje.

MS Windows s Trumpet Winsockem

Trumpet Winsock má vestavěnou podporu proxy serverů. V nabídce „setup" zadáte adresu proxy serveru a adresy všech počítačů, které jsou přístupné přímo. Trumpet bude všechen odcházející provoz správně obsluhovat.

Proxy server s podporou UDP paketů

Program SOCKS pracuje pouze s TCP pakety, nikoliv s UDP pakety. Tím jeho použitelnost poněkud klesá. Protokol UDP používá řada užitečných programů, například Archie. Existuje program, který má fungovat jako proxy server pro UDP pakety, jmenuje se UDPrelay a vytvořil jej Tom Fitzgerald (*fitz@wang.com*). V době vzniku tohoto textu však nebyl k dispozici pro Linux.

Nevýhody proxy serverů

Proxy server je v první řadě *bezpečnostní zařízení*. Pokud jej používáte k zajištění přístupu na Internetu pro počítače, pro něž nemáte řádné IP adresy, přináší to s sebou nevýhody. Proxy server sice umožňuje poměrně rozsáhlý přístup zevnitř sítě ven, neumožňuje však jakkoliv přistupovat zvenku dovnitř. Znamená to, že na interních počítačích nemohou pracovat žádné servery, nelze s nimi navázat žádné přímé poštovní, chatovací nebo archie spojení. Tyto nevýhody vypadají nepodstatně, ale:

- Důležitý dokument, na němž pracujete, jste si nechali na počítači v chráněné síti. Jste doma a chtěli byste na něm dále pracovat. Nemůžete. Nemůžete se ke svému počítači dostat, protože je za firewallem. Mohli byste se sice nejprve přihlásit na firewall, jenomže protože veškerý přístup zevnitř ven zajišťuje proxy server, nemáte na firewallu účet.
- Dcera je na vysoké a chcete jí poslat poštu. Potřebujete jí říct něco důvěrného a nejraději byste jí poštu poslali přímo na její počítač. Ne že byste nevěřili správci vašeho systému, ale přece jenom, jde o důvěrné věci.
- Zásadní nevýhodou proxy serverů je nemožnost použití UDP paketů. Předpokládám však, že podpora protokolu UDP bude brzy realizována.

Další problémy s proxy servery má služba FTP. Když stahujete nějaká data nebo jen vypisujete obsah adresáře na FTP serveru, server otevře na klientském počítači soket a přes něj vám posílá informace. Proxy server to nedovolí, takže FTP nebude fungovat.

Navíc jsou proxy servery pomalé. Vzhledem k velkému množství operací, které musejí provádět, je prakticky jakékoliv řešení rychlejší.

V zásadě platí, že pokud máte dost IP adres a nezáleží vám příliš na bezpečnosti, nepoužívejte ani firewall, ani proxy server. Pokud nemáte dost IP adres, nicméně na bezpečnosti vám pořád příliš nezáleží, můžete použít nějaký IP emulátor, například Term, Slirp nebo TIA. Term získáte na adrese *ftp://sunsite.unc.edu*, Slirp na adrese *ftp://blitzen.canbera.edu.au/pub/slirp*, TIA je k dispozici na *marketplace.com*. Tyto programy jsou rychlé, umožňují lepší spojení a zajišťují větší míru pří-

stupu zvenčí dovnitř. Proxy servery jsou výhodné pro sítě, kde je hodně počítačů, které všechny potřebují častý přístup na Internet. V takovém případě se vyplatí jednorázově nakonfigurovat proxy server.

Složitější konfigurace

Existuje ještě jeden příklad konfigurace, o kterém bych se chtěl zmínit dříve, než tento dokument uzavřu. Zatím popsaná konfigurace bude pravděpodobně většině uživatelů vyhovovat. Následující text však popisuje složitější konfiguraci, na níž si můžeme vyjasnit některá problémová místa. Pokud vás zajímá více než to, o čem jsme doposud hovořili, nebo pokud se prostě jenom zajímáte o možnosti proxy serverů a firewallů, čtěte dále.

Velká síť s vysokým zabezpečením

Řekněme, že vedete teroristickou skupinu a chcete si vytvořit počítačovou síť. Máte 50 počítačů a podsítě 32 (5bitových) IP adres. V rámci sítě potřebujete různé úrovně přístupu, protože různým lidem říkáte různé věci. Proto musí být jednotlivé části sítě chráněny od ostatních.

Úrovně ochrany jsou:

1. Externí úroveň. Tato úroveň je přístupná všem. Zde inzerujete a získáváte nové dobrovolníky.
2. Dobrovolníci. Na této úrovni se pohybují různí nevýznamní dobrovolníci. Vysvětlujete jim, jak jsou vlády nepřátelské a jak vyrábět bomby.
3. Žoldnéři. Tady máte *opravdové* plány. Na této úrovni jsou uloženy informace o převzetí moci nad zbytkem světa, o plánech likvidace nepřátelských vlád, seznamy členů a tak.

Nastavení sítě

IP adresy jsou rozděleny takto:

- jedna adresa 192.168.1.255 je vysílací a tedy nepoužitelná
- 23 z 32 adres je přiřazeno 23 počítačům, které budou přístupné z Internetu
- jedna další IP adresa je přiřazena jednomu linuxovému stroji
- druhá další IP adresa je přiřazena druhému linuxovému stroji
- dvě IP adresy potřebuje směrovač
- čtyři adresy jsou nepoužité, ale jako kamufláž mají přiřazena jména paul, ringo, john a george
- obě chráněné sítě používají adresy 192.168.1.xxx

Dále vytvoříme dvě samostatné sítě, každou v jedné místnosti. Jsou propojeny infračerveným spojem, takže zvnějšku nejsou odposlouchávatelné. Infračervené propojení se naštěstí chová jako klasický Ethernet.

Obě samostatné sítě jsou připojeny každá k jednomu vyhrazenému linuxovému stroji.

K oběma chráněným sítím je připojen souborový server. Musí to tak být, protože v plánech na ovládnutí světa figurují i někteří výše postavení dobrovolníci. Tento server má adresu 192.168.1.17 na síti dobrovolníků a 192.168.1.23 na síti žoldnéřů. IP forwarding je vypnut.

I na obou linuxových strojích je vypnut IP forwarding. Směrovač nebude předávat pakety pro síť 192.168.1.xxx, pokud na to nebude explicitně nastaven, takže obě sítě budou z Internetu dostup-

né. Důvodem vypnutí forwardingu je to, aby se pakety ze sítě dobrovolníků nemohly dostat na síť žoldnéřů a naopak.

NFS server je možné nastavit tak, aby na různé sítě nabízel různé soubory. To může být velmi užitečné a pomocí pár triků se symbolickými odkazy jde zařídit i to, aby společné soubory byly sdíleny na všech sítích. Pomocí tohoto nastavení a další síťové karty můžeme zajistit, že server bude sloužit všem třem sítím.

Protože na všech třech úrovních potřebujeme získávat různé informace, všechny musejí mít přístup na Internet. Externí síť je na Internet připojena přímo, takže zde se nemusíme s proxy serverem zatěžovat. Sítě dobrovolníků a žoldnéřů jsou za firewallem, a tak pro ně musíme proxy server nastavit.

Obě sítě budou nastaveny velmi podobně. Obě mají přiřazeny stejné IP adresy. Definujme si nyní pár pravidel, ať se nám nastavení trochu zkomplikuje.

- Souborový server nemůže mít přístup na Internet – tím by byl vystaven virům a dalším odpornostem, a protože je pro nás velmi důležitý, tak to raději vůbec nepovolíme.
- Dobrovolníkům nepovolíme přístup k WWW. Procházejí právě výcvikem a možnost získávání různých informací by jim v této fázi mohla uškodit.

Soubor sockd.conf na linuxovém stroji sítě dobrovolníků tedy bude obsahovat následující řádek:

```
deny 192.168.1.17 255.255.255.255
```

A v síti žoldnéřů

```
deny 192.168.1.123 255.255.255.255
```

Kromě toho na stroji dobrovolníků nastavíme

```
deny 0.0.0.0 0.0.0.0 eq 80
```

Tím říkáme, že zakazujeme přístup všem počítačům na port rovný (equal) 80, tedy na port HTTP. Tím budou možné všechny ostatní služby, WWW přístup však bude zakázán.

Kromě toho v obou souborech nastavíme

```
permit 192.168.1.0 255.255.255.0
```

čímž všem počítačům na síti 192.168.1.xxx povolíme použití proxy serveru (kromě těch, které už ho mají zakázán – tedy kromě souborového serveru a kromě HTTP přístupu ze sítě dobrovolníků).

Soubor sockd.conf na síti dobrovolníků bude nastaven takto:

```
deny 192.168.1.17 255.255.255.255
deny 0.0.0.0 0.0.0.0 eq 80
permit 192.168.1.0 255.255.255.0
```

a na síti žoldnéřů takto:

```
deny 192.168.1.23 255.255.255.255
permit 192.168.1.0 255.255.255.0
```

Tím by mělo být všechno správně nastaveno. Jednotlivé sítě jsou izolovány s nastavením potřebné míry propojení. Všichni budou spokojení.

Usnadnění správy

Nástroje firewallů

Existuje několik softwarových balíků, které usnadňují správu firewallu.

Buďte opatrní a nepoužívejte je, pokud s nimi neumíte pracovat. Tak jak vám mohou usnadnit práci, stejně jednoduše vám dovolí i udělat chybu.

Pro nastavení filtračních pravidel existují grafická i webová rozhraní. Některé společnosti dokonce dodávají vlastní komerční firewally založené na Linuxu, nainstalovaném ve speciálním počítači se speciálním obslužným programem. (Milé.)

Nemám příliš rád grafická rozhraní, nicméně jsem nějakou dobu používal firewally s grafickým rozhraním. Takové rozhraní je užitečné, protože nabídne přehledný obrázek nastavení všech filtračních pravidel.

gfcc (GTK+ Firewall Control Center) je aplikace umožňující nastavovat firewallové politiky a pravidla, založená na balíku ipchains. Můžete ji získat na adrese *http://icarus.autostock.co.kr*. Jedná se o velmi pěkný nástroj.

V příloze A uvádím konfigurační RC skripty. Ty fungují s i bez gfcc.

Existuje řada skriptů pro nastavení firewallu. Jeden velmi dokonalý můžete najít na adrese *http://www.jasmine.org.uk/~simon/bookshelf/papers/instant-firewall/instant-firewall.html*. Další dobrý skript je na adrese *http://www.pointman.org/*.

Kfirewall je grafické rozhraní k balíkům ipchains nebo ipfwadm (v závislosti na verzi jádra). Najdete je na adrese *http://megaman.ypsilonia.net/kfirewall/*.

FCT je HTML nástroj pro konfiguraci firewallu. Umožňuje automatické generování skriptů obsahujících filtrovací příkazy (ipfwadm) pro firewall s více rozhraními a libovolnými službami. Nachází se na adrese *http://www.fen.baynet.de/~ft114/FCT/firewall.htm*.

Obecné nástroje

WebMin je nástroj pro obecnou správu systému. Neumožňuje sice nastavovat pravidla firewallů, umí však zapínat a vypínat démony a procesy. Jde o velmi dobrý program a věřím, že jeho autor J. Cameron jej doplní i o modul administrace IPCHAINS. Nachází se na adrese *http://www.webmin.com/*.

Pokud poskytujete přístup k Internetu, bude vás zajímat program IPFA (IP Firewall Accounting) na adrese *http://www.soaring-bird.com/ipfa/*. Umožňuje vytváření měsíčních, denních nebo minutových záznamů a obsahuje webové grafické rozhraní pro správu.

httptunel vytváří obousměrnou virtuální datovou cestu tunelem přes HTTP požadavky. HTTP požadavky je pak možné odesílat prostřednictvím firewallu. Umožňuje tak vytvoření virtuální privátní sítě. Adresa programu je *http://sunsite.auc.dk/vpnd/*.

Příklady skriptů

RC skript používající GFCC

```
#!/bin/bash
#
# Firewall Script - Version 0.9.1
#
```

```
# chkconfig: 2345 09 99
# popis: firewallový skript pro jádro 2.2.x
# Nastavit pro testování:
# set -x
#
# POZNÁMKY:
#
# Tento skript je napsán pro RedHat 6.1 a vyšší.
#
# Opatrně při poskytování veřejných služeb jako web nebo ftp.
#
# INSTALACE:
# 1. nahrejte tento soubor do /etc/rc.d/init.d (budete muset být root..)
# pojmenujte jej tak nějak jako "firewall" :-)
# ať jej vlastní root --> "chown root.root (název)"
# ať je spustitelný --> "chmod 755 (název)"
#
# 2. použijte GFCC k vytvoření pravidel firewallu a exportujte je do
# souboru /etc/gfcc/rules/firewall.rule.sh.
#
# 3. přidejte firewall do inicializačních struktur RH --> "chkconfig --add (ná-
zev)"
# při příštím startu systému by to mělo naběhnout automaticky!
# spěte sladce, protože máte systém O NĚCO MÉNĚ zranitelný...
#
# Poznámky k verzím:
# 30 Jan, 2000 - Změněno na GFCC skript
# 11 Dec, 1999 - aktualizoval Mark Grennan <mark@grennan.com>
# 20 July, 1999 - původní verze od Anthony Ball <tony@LinuxSIG.org>
#
########################################################
# Knihovna zdrojových funkcí.
. /etc/rc.d/init.d/functions
# Konfigurace zdrojové sítě.
. /etc/sysconfig/network
# Kontrola zda je síť nastavena.
[ ${NETWORKING} = "no" ] && exit 0
# Podíváme se, jak nás volali
case "$1" in
start)
# Začínáme poskytovat přístup
action "Spouštím firewall: " /bin/true
/etc/gfcc/rules/firewall.rule.sh
echo
;;
stop)
action "Zastavuji firewall: " /bin/true
echo 0 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
echo
;;
```

```
restart)
action "Restartuji firewall: " /bin/true
$0 stop
$0 start
echo
;;
status)
# Výpis nastavení
/sbin/ipchains -L
;;
test)
action "Testovací režim: " /bin/true
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
echo 1 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -A input -j ACCEPT
/sbin/ipchains -A output -j ACCEPT
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -i $PUBLIC -j MASQ
echo
;;
*)
echo "Spuštění: $0 {start|stop|restart|status|test}"
exit 1
esac
```

GFCC skript

Tento skript byl vygenerován programem Graphical Firewall (GFCC). Nejedná se o funkční sadu pravidel, jde pouze o exportovanou sadu.

```
#!/bin/sh
# Vygenerováno programem Gtk+ firewall control center
IPCHAINS=/sbin/ipchains
localnet="192.168.1.0/24"
firewallhost="192.168.1.1/32"
localhost="172.0.0.0/8"
DNS1="24.94.163.119/32"
DNS2="24.94.163.124/32"
Broadcast="255.255.255.255/32"
Multicast="224.0.0.0/8"
Any="0.0.0.0/0"
mail_grennan_com="192.168.1.1/32"
mark_grennan_com="192.168.1.3/32"
$IPCHAINS -P input DENY
$IPCHAINS -P forward ACCEPT
$IPCHAINS -P output ACCEPT
$IPCHAINS -F
$IPCHAINS -X
# vstupní pravidla
$IPCHAINS -A input -s $Any -d $Broadcast -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any netbios-ns -j DENY
```

```
$IPCHAINS -A input -p tcp -s $Any -d $Any netbios-ns -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any netbios-dgm -j DENY
$IPCHAINS -A input -p tcp -s $Any -d $Any netbios-dgm -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any bootps -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any bootpc -j DENY
$IPCHAINS -A input -s $Multicast -d $Any -j DENY
$IPCHAINS -A input -s $localhost -d $Any -i lo -j ACCEPT
$IPCHAINS -A input -s $localnet -d $Any -i eth1 -j ACCEPT
$IPCHAINS -A input -s $localnet -d $Broadcast -i eth1 -j ACCEPT
$IPCHAINS -A input -p icmp -s $Any -d $Any -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any -j ACCEPT ! -y
$IPCHAINS -A input -p udp -s $DNS1 domain -d $Any 1023:65535 -j ACCEPT
$IPCHAINS -A input -p udp -s $DNS2 domain -d $Any 1023:65535 -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any ssh -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any telnet -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any smtp -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any pop-3 -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any auth -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any www -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any ftp -j ACCEPT
$IPCHAINS -A input -s $Any -d $Any -j DENY -l
# předávací pravidla
$IPCHAINS -A forward -s $localnet -d $Any -j MASQ
# výstupní pravidla
```

RC skript bez GFCC

Toto nastavení pravidel jsem vytvořil sám. Nepoužívá GFCC.

```
#!/bin/bash
#
# Firewall Script - Version 0.9.0
# chkconfig: 2345 09 99
# popis: firewallový skript pro jádro 2.2.x
# Nastavit pro testování:
# set -x
#
# POZNÁMKY:
#
# Tento skript je určen pro RedHat 6.0 a vyšší.
#
# Nastavení by mělo vyhovovat většině směrovačů, vytáčených připojení a kabelo-
vých
# modemů. Je určeno pro distribuci RedHat.
#
# Pozor při nabízení veřejných služeb jako web nebo ftp.
#
# INSTALACE:
# 1. Tento soubor je určen pro systém RedHat. Měl by pracovat
# i na jiných distribucích, ale...
# Kdo ví?!!? Pokyny platí pro RedHat.
#
# 2. nahrejte tento soubor do /etc/rc.d/init.d (budete muset být root..)
```

```
# pojmenujte jej tak nějak jako "firewall" :-)
# ať jej vlastní root --> "chown root.root (název)"
# ať je spustitelný --> "chmod 755 (název)"
#
# 3. nastavte hodnoty pro vaši síť, rozhraní a DNS servery
# odkomentujte řádky povolující potřebné volitelné služby
# zajistěte, aby karta v počítači byla "eth0" (nebo změňte tuto hodnotu ve
skriptu)
# otestujte skript --> "/etc/rc.d/init.d/<název> start"
# pravidla si můžete vypsat --> "ipchains -L -n"
# opravte co nefunguje... :-)
#
# 4. přidejte firewall do inicializačních struktur RH --> "chkconfig --add (ná-
zev)"
# při příštím startu systému by to mělo naběhnout automaticky!
# spěte sladce, protože máte systém O NĚCO MÉNĚ zranitelný...
#
# Poznámky k verzím:
# 20 July, 1999 - původní verze - Anthony Ball <tony@LinuxSIG.org>
# 11 Dec, 1999 - aktualizoval Mark Grennan <mark@grennan.com>
#
########################################################
# Doplňte následující hodnoty aby platily pro vaši síť
PRIVATENET=xxx.xxx.xxx.xxx/xx
PUBLIC=ppp0
PRIVATE=eth0
# vaše DNS servery
DNS1=xxx.xxx.xxx.xxx
DNS2=xxx.xxx.xxx.xxx
########################################################
# některé obecné užitečné hodnoty
ANY=0.0.0.0/0
ALLONES=255.255.255.255
# Knihvona zdrojových funkcí
. /etc/rc.d/init.d/functions
# Konfigurace zdrojové sítě
. /etc/sysconfig/network
# Kontrola zda je síť nastavena
[ ${NETWORKING} = "no" ] && exit 0
# Podíváme se, jak nás spustili
case "$1" in
start)
# Povolujeme přístup
action "Spouštím firewall: " /bin/true
##
## Nastavení prostředí
##
# Smazání všech pravidel
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
# Všechno zakázat
/sbin/ipchains -I input 1 -j DENY
```

```
# nastavíme politiku na deny (implicitně je ACCEPT)
/sbin/ipchains -P input DENY
/sbin/ipchains -P output ACCEPT
/sbin/ipchains -P forward ACCEPT
# Zapneme předávání paketů
echo 1 > /proc/sys/net/ipv4/ip_forward
##
## Instalace modulů
##
# Přidáme modul aktivního ftp. Tím se povolí aktivní ftp na počítače v lokální
# síti (ale ne na směrovač, protože nemá maškarádu)
if ! ( /sbin/lsmod | /bin/grep masq_ftp > /dev/null ); then
/sbin/insmod ip_masq_ftp
fi
##
## Nějaká bezpečnostní nastavení
##
# zapneme ověřování zdrojových adres a ochranu před spoofingem na všech
# aktivních i budoucích rozhraních.
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
echo 1 > $f
done
else
echo
echo "POTÍŽE PŘI AKTIVACI OCHRANY PŘED SPOOFINGEM! POZOR! "
echo
fi
# vypneme broadcast na zbývajících rozhraních
/sbin/ipchains -A input -d 0.0.0.0 -j DENY
/sbin/ipchains -A input -d 255.255.255.255 -j DENY
# toto vypínáme bez logování, protože toho bývá hodně...
/sbin/ipchains -A input -p udp -d $ANY 137 -j DENY # NetBIOS přes IP
/sbin/ipchains -A input -p tcp -d $ANY 137 -j DENY # ""
/sbin/ipchains -A input -p udp -d $ANY 138 -j DENY # ""
/sbin/ipchains -A input -p tcp -d $ANY 138 -j DENY # ""
/sbin/ipchains -A input -p udp -d $ANY 67 -j DENY # bootp
/sbin/ipchains -A input -p udp -d $ANY 68 -j DENY # ""
/sbin/ipchains -A input -s 224.0.0.0/8 -j DENY # Multicastové adresy
##
## Povolíme privátní síti ven
##
# povolíme pakety na zpětném rozhraní
/sbin/ipchains -A input -i lo -j ACCEPT
# povolíme pakety z interního "prověřeného" rozhraní
/sbin/ipchains -A input -i $PRIVATE -s $PRIVATENET -d $ANY -j ACCEPT
/sbin/ipchains -A input -i $PRIVATE -d $ALLONES -j ACCEPT
##
## Povolíme vnější služby na firewall (jestli si trouváte)
##
# povolíme ICMP
/sbin/ipchains -A input -p icmp -j ACCEPT
# povolíme TCP
```

```
/sbin/ipchains -A input -p tcp ! -y -j ACCEPT
# povolíme DNS dotazy (na firewall)
/sbin/ipchains -A input -p udp -s $DNS1 domain -d $ANY 1023: -j ACCEPT
/sbin/ipchains -A input -p udp -s $DNS2 domain -d $ANY 1023: -j ACCEPT
# nebo (LEPŠÍ ŘEŠENÍ) spustíme na směrovači caching DNS server a místo před-
chozích
# dáme následující dva řádky...
# /sbin/ipchains -A input -p udp -s $DNS1 domain -d $ANY domain -j ACCEPT
# /sbin/ipchains -A input -p udp -s $DNS2 domain -d $ANY domain -j ACCEPT
# odkomentováním následujícího se povolí příchozí ssh
/sbin/ipchains -A input -p tcp -d $ANY 22 -j ACCEPT
# odkomentováním následujícího se povolí telnet (ŠPATNÝ NÁPAD!!)
/sbin/ipchains -A input -p tcp -d $ANY telnet -j ACCEPT
# odkomentováním následujícího se povolí (network time protocol) na směrovač
# /sbin/ipchains -A input -p udp -d $ANY ntp -j ACCEPT
# odkomentováním následujícího se povolí SMTP (ne pro klienty, pouze server)
/sbin/ipchains -A input -p tcp -d $ANY smtp -j ACCEPT
# odkomentováním následujícího se povolí POP3 (pro klienty)
/sbin/ipchains -A input -p tcp -d $ANY 110 -j ACCEPT
# povolíme auth pro posílání pošty a ftp
/sbin/ipchains -A input -p tcp -d $ANY auth -j ACCEPT
# povolíme HTTP (pouze pokud na směrovači běží www server)
/sbin/ipchains -A input -p tcp -d $ANY http -j ACCEPT
# povolíme FTP
/sbin/ipchains -A input -p tcp -d $ANY ftp -j ACCEPT
##
## Nastavení maškarády
##
# maškaráda paketů z interní sítě
/sbin/ipchains -A forward -s $PRIVATENET -d $ANY -j MASQ
##
## zakážeme COKOLIV jiného a zalogujeme to do /var/log/messages
##
/sbin/ipchains -A input -l -j DENY
# Zrušení plošného zákazu
/sbin/ipchains -D input 1
;;
stop)
action "Zastavuji firewall: " /bin/true
echo 0 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
echo
;;
restart)
action "Restartuji firewall: " /bin/true
$0 stop
$0 start
echo
;;
status)
# Výpis nastavení
```

```
/sbin/ipchains -L
;;
test)
##
## Tak jednoduché, jak to jenom jde
## (Není to VŮBEC bezpečné)
action "POZOR! Testování firewallu: " /bin/true
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
echo 1 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -A input -j ACCEPT
/sbin/ipchains -A output -j ACCEPT
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -i $PUBLIC -j MASQ
echo
;;
*)
echo "Spuštění: $0 {start|stop|restart|status|test}"
exit 1
esac
```

VPN RC skript pro RedHat

```
#!/bin/sh
#
# vpnd
# Tento skript zapíná a vypíná vpnd (Virtual Privage Network connections).
#
# chkconfig: - 96 96
# popis: vpnd
#
# Knihovna zdrojových funkcí.
. /etc/rc.d/init.d/functions
# Konfigurace zdrojové sítě.
. /etc/sysconfig/network
# Kontrola zda je síť nastavena.
[ ${NETWORKING} = "no" ] && exit 0
[ -f /usr/sbin/vpnd ] || exit 0
[ -f /etc/vpnd.conf ] || exit 0
RETVAL=0
# Podíváme se, jak nás volali.
case "$1" in
start)
# Start daemons.
echo -n "Spouštím vpnd: "
daemon vpnd
RETVAL=$?
[ $RETVAL -eq 0 ] && touch /var/lock/subsys/vpnd
echo
;;
stop)
```

```
# Stop daemons.
echo -n "Zastavuji vpnd: "
killproc vpnd
RETVAL=$?
[ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/vpnd
echo
;;
restart)
$0 stop
$0 start
;;
*)
echo "Spuštění: vpnd {start|stop|restart}"
exit 1
esac
exit $RETVAL
```

Úvod do programování v BASH

Originál: http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html

Tento text vám má pomoci začít programovat jednoduché a složitější skripty příkazového interpretu. Není určen jako dokument pro pokročilé. Nejsem ani expert ani zkušený programátor v příkazovém interpretu. Rozhodl jsem se tento text napsat, protože mi to pomůže naučit se spoustu věcí a může to být užitečné i ostatním. Je vítána jakákoliv odezva, zejména ve formě doplňků.

Úvod

Kde získat nejnovější verzi

http://www.linuxdoc.org/HOWTO/Bash-Prog-Intro-HOWTO.html

Předpoklady

Vhodná je obeznámenost s příkazovou řádkou Linuxu a se základními programátorskými dovednostmi. I když tento text nepředstavuje úvod do programování, objasňuje (nebo se alespoň snaží) řadu základních postupů.

Použití dokumentu

Tento dokument vám bude užitečný v následujících případech:

- Umíte programovat a chcete začít psát nějaké skripty příkazového interpretu.
- Máte jakési ponětí o programování v příkazovém interpretu a potřebujete nějakou referenční příručku.
- Chcete se podívat na nějaké skripty a vysvětlení, abyste mohli začít psát vlastní.
- Přecházíte nebo jste přešli z prostředí DOS/Windows a chcete psát „dávkové soubory".
- Jste úplný blázen a čtete všechna HOWTO.

Velmi jednoduché skripty

Tento dokument by vám měl poskytnout základní představu o programování příkazového interpretu na základě různých příkladů.

V této části najdete několik krátkých skriptů, které by vám měly pomoci pochopit některé techniky.

Klasický skript „Hello World“

```
#!/bin/bash
echo Hello World
```

Tento skript má pouhé dva řádky. První řádek říká systému, jaký program má ke spuštění souboru použít.

Druhý řádek definuje jedinou akci prováděnou skriptem, totiž vypsání textu „Hello World“ na terminál.

Pokud uvidíte hlášení typu *./hello.sh: Command not found*, pak je zřejmě chyba v prvním řádku `#!/bin/bash`. Zkuste příkaz `whereis bash` nebo se podívejte do části *Jak najít bash* abyste zjistili, jak má řádek správně vypadat.

Velmi jednoduchý zálohovací skript

```
#!/bin/bash
tar -cZf /var/my-backup.tgz /home/me/
```

V tomto skriptu nevypisujeme nic na terminál, místo toho vytváříme tarový archiv obsahující domovský adresář uživatele. Tento skript NENÍ určen k běžnému použití, později si ukážeme vhodnější zálohovací skript.

Vše o přesměrování

Teorie a stručná reference

Existují tři souborové deskriptory, *stdin*, *stdout* a *stderr* (kde *std* znamená *standard*).

V zásadě můžete:

- přesměrovat *stdout* do souboru
- přesměrovat *stderr* do souboru
- přesměrovat *stdout* na *stderr*
- přesměrovat *stderr* na *stdout*
- přesměrovat *stderr* a *stdout* do souboru
- přesměrovat *stderr* a *stdout* na *stdout*
- přesměrovat *stderr* a *stdout* na *stderr*

Symbol *1* „reprezentuje“ *stdout*, *2* pak *stderr*.

Malá poznámka k této problematice: příkazem `less` můžete prohlížet jak *stdout* (který zůstává v bufferu), tak i *stderr*, který se tiskne na obrazovce, ale vymaže se, když „procházíte“ bufferem.

Příklad: stdout do souboru

Tímto způsobem se výstup programu zapíše do souboru.

```
ls -l > ls-l.txt
```

V tomto příkladu se vytvoří souboru *ls-l.txt*, který bude obsahovat to, co byste viděli na obrazovce po zadání a spuštění příkazu `ls -l`.

Příklad: stderr do souboru

Tímto způsobem se chybový výstup programu zapíše do souboru.

```
grep da * 2> grep-errors.txt
```

Vytvoří se soubor *grep-errors.txt*, který bude obsahovat chybový výstup příkazu `grep da *`.

Příklad: stdout na stderr

Tímto způsobem se bude výstup programu vypisovat do stejného souborového deskriptoru jako standardní chybový výstup.

```
grep da * 1>&2
```

Standardní výstup příkazu se posílá na standardní chybový výstup, což můžete ověřit různými způsoby.

Příklad: stderr na stdout

Tímto způsobem se bude chybový výstup programu vypisovat do stejného souborového deskriptoru jako standardní výstup.

```
grep * 2>&1
```

Standardní chybový výstup programu se nyní posílá na *stdout*, takže když si výstup prohlédnete programem `less`, zjistíte, že řádky, které normálně „mizí“ (protože jsou zapisovány na *stderr*) tentokrát zůstávají (protože se vypisují na *stdout*).

Příklad: stderr a stdout do souboru

Tímto způsobem se celý výstup programu přesměruje do souboru. Je to užitečné například u programů spouštěných démonem *cron*, pokud chcete, aby program proběhl v úplné tichosti.

```
rm -f $(find / -name core) &> /dev/null
```

Tento příkaz (zamýšlený ke spouštění démonem *cron*) vymaže všechny soubory *core* ve všech adresářích. Pokud úplně potlačíte výstup nějakého programu, měli byste si být velmi přesně jisti tím, co program dělá.

Roury

V této části se jednoduše a prakticky vysvětluje, jak používat roury a k čemu se dají použít.

Co to je a k čemu je to dobré

Roury umožňují velice jednoduše předat výstup jednoho programu jako vstup dalšímu programu.

Příklad: jednoduchá roura s programem sed

Takto vypadá jednoduchý příklad použití roury:

```
ls -l | sed -e "s/[aeio]/u/g"
```

Stane se toto: spustí se první příkaz (`ls -l`) a jeho výstup se místo vypsání předá programu `sed`, který pak vypíše to, co s výpisem provede.

Příklad: alternativa k ls-l *.txt

Následující příklad je složitější alternativou k příkazu `ls -l *.txt`, ovšem my jej tady uvádíme jako příklad použití roury, ne jako návod pro vypisování souborů:

```
ls -l | grep "\.txt$"
```

Výpis příkazu `ls -l` se předává programu `grep`, který pak vypíše všechny řádky odpovídající regulárnímu výrazu `"\.txt$"`.

Proměnné

Stejně jako v jiných programovacích jazycích můžete i ve skriptech používat proměnné. Proměnná může obsahovat číslo, znak nebo řetězec znaků.

Proměnné nemusíte deklarovat, automaticky se vytvoří, když jim přiřadíte hodnotu.

Příklad: Hello World s proměnnými

```
#!/bin/bash
STR="Hello World!"
echo $STR
```

Ve druhém řádku se vytvoří proměnná pojmenovaná STR a bude mít hodnotu „Hello World!“. Hodnota proměnné se dá přečíst tak, že se před název proměnné uvede symbol $. Všimněte si (a vyzkoušejte si), že pokud nepoužijete symbol $, bude výstup programu jiný a pravděpodobně ne takový, jaký jste chtěli.

Příklad: Velmi jednoduchý zálohovací skript *(vylepšený)*

```
#!/bin/bash
OF=/var/my-backup-$(date +%Y%m%d).tgz
tar -cZf $OF /home/me/
```

Tento skript obsahuje několik novinek. V první řadě si musíme vyjasnit vytvoření a inicializaci proměnné na druhém řádku. Všimněte si výrazu *$(date +%Y%m%d)*. Pokud jej spustíte ve skriptu, zjistíte, že provede příkaz v závorkách a převezme jeho výstup.

Všimněte si také, že v našem skriptu bude název výstupních souborů každý den jiný díky přepínači *+%Y%m%d*. Můžete jej změnit a nastavit si jiný formát.

Některé další příklady:

```
echo ls
echo $(ls)
```

Lokální proměnné

Lokální proměnné se vytvářejí pomocí klíčového slova *local*.

```
#!/bin/bash
HELLO=Hello
function hello {
local HELLO=World
echo $HELLO
}
echo $HELLO
hello
echo $HELLO
```

Tento příklad by měl stačit k pochopení, jak pracovat s lokálními proměnnými.

Podmínky

Podmínky umožňují rozhodnout, zda provést nebo neprovést nějakou operaci. Rozhodnutí se provádí na základě vyhodnocení nějakého výrazu.

Suchá teorie

Podmínky mají mnoho formátů. Nejjednodušší typ je **if** *výraz* **then** *příkaz*, kdy se *příkaz* provede jenom tehdy, pokud bude hodnota *výrazu* pravdivá. Například *2<1* je výraz, který je nepravdivý, zatímco *2>1* je pravdivý výraz.

Jiná forma podmínky může být **if** *výraz* **then** *příkaz1* **else** *příkaz2*. V tomto případě se provede *příkaz1* tehdy, je-li *výraz* pravdivý, v opačném případě se provede *příkaz2*.

Další forma podmínky může být **if** *výraz1* **then** *příkaz1* **else if** *výraz2* **then** *příkaz2* **else** *příkaz3*. V tomto příkladu jsme doplnili část **else if ...**, která vede k tomu, že *příkaz* se provede pouze pokud je pravdivý *výraz2*. Zbylé chování je takové, jak asi očekáváte (viz předchozí příklady).

Poznámka k syntaxi:

Základní syntaxe konstrukce *if* vypadá v BASH takto:

```
if [výraz];
then
kód je-li výraz pravdivý
fi
```

Příklad: jednoduchá podmínka if ... then

```
#!/bin/bash
if [ "foo" = "foo" ]; then
    echo expression evaluated as true
fi
```

Kód, který se provede v případě, že je výraz v lomených závorkách pravdivý, se nachází mezi slovy `then` a `fi`, která označují konec podmíněně prováděného kódu.

Příklad: jednoduchá podmínka if ... then ... else

```
#!/bin/bash
if [ "foo" = "foo" ]; then
    echo expression evaluated as true
else
    echo expression evaluated as false
fi
```

Příklad: podmínka s proměnnými

```
#!/bin/bash
T1="foo"
T2="bar"
if [ "$T1" = "$T2" ]; then
    echo expression evaluated as true
else
    echo expression evaluated as false
fi
```

Smyčky for, while a until

V této části se seznámíme se smyčkami **for**, **while** a **until**.

Smyčka **for** se poněkud liší od jiných programovacích jazyků. V zásadě se používá k iteraci přes posloupnost „slov“ v řetězci.

Smyčka **while** opakovaně provádí část kódu, pokud je řídící výraz pravdivý, a zastaví se teprve až výraz bude nepravdivý (nebo pokud se v těle smyčky explicitně provede příkaz *break*).

Smyčka **until** je prakticky totožná se smyčkou **while** s tím rozdílem, že tělo se provádí tak dlouho, dokud je řídící výraz nepravdivý.

Pokud máte pocit, že smyčky **while** a **until** jsou velice podobné, máte pravdu.

Příklad smyčky for

```
#!/bin/bash
for i in $( ls ); do
    echo item: $i
done
```

Ve druhém řádku deklarujeme *i* jako proměnnou, která bude nabývat jednotlivé hodnoty obsažené v *$(ls)*.

Třetí řádek může být klidně delší, případně mohou před příkazem *done* následovat i další řádky.

Slovíčko *done* na čtvrtém řádku označuje konec těla smyčky, proměnná *i* nabude další hodnotu ze vstupní množiny a tělo se provede znovu.

Tento příklad nemá valného praktického použití, rozumnější příklad by mohl používat smyčku **for** k nalezení pouze souborů vyhovujících nějaké požadované vlastnosti.

Smyčka for jako v C

Fiesh navrhl doplnit následující typ smyčky. Jedná se o smyčku **for** podobnou té, jak ji známe z jazyka C, Perlu a podobně.

```
#!/bin/bash
for i in `seq 1 10`;
do
    echo $i
done
```

Příklad smyčky while

```
#!/bin/bash
COUNTER=0
while [ $COUNTER -lt 10 ]; do
    echo The counter is $COUNTER
    let COUNTER=COUNTER+1
done
```

Tento skript „emuluje“ chování struktury *for* známé z Pascalu, C a podobně.

Příklad smyčky until

```
#!/bin/bash
COUNTER=20
until [ $COUNTER -lt 10 ]; do
    echo COUNTER $COUNTER
    let COUNTER-=1
done
```

Funkce

Ve většině programovacích jazyků můžete používat funkce, které dovolují členit kód na logické celky nebo provozovat tajemné umění rekurze.

Deklarace funkce obnáší pouhé zapsání `function moje_funkce { můj_kód }`.

Volání funkce je stejné jako volání jakéhokoliv programu – stačí zapsat její název.

Příklad funkce

```
#!/bin/bash
function quit {
    exit
}
function hello {
    echo Hello!
}
hello
quit
echo foo
```

Řádky 2 až 4 obsahují funkci `quit`. Řádky 5 až 7 obsahují funkci `hello`. Pokud si nejste úplně jisti, co tento skript udělá, vyzkoušejte si jej.

Funkce nemusejí být deklarovány v žádném specifickém pořadí.

Když skript spustíte, zjistíte že nejprve zavolá funkci `hello`, pak funkci `quit` a na řádek 10 se už vůbec nedostane.

Příklad funkce s parametry

```
#!/bin/bash
function quit {
    exit
}
function e {
    echo $1
}
e Hello
e World
quit
echo foo
```

Skript je prakticky shodný s předchozím. Hlavní rozdíl je ve funkci `e`. Tato funkce vytiskne první předaný parametr. Parametry se uvnitř funkcí obsluhují stejně jako parametry předávané celému skriptu.

Uživatelské rozhraní

Jednoduchá nabídka příkazem select

```
#!/bin/bash
OPTIONS="Hello Quit"
select opt in $OPTIONS; do
    if [ "$opt" = "Quit" ]; then
      echo done
      exit
    elif [ "$opt" = "Hello" ]; then
      echo Hello World
    else
```

```
      clear
      echo bad option
    fi
done
```

Když si tento skript spustíte, zjistíte, že jde o sen každého programátora, co se týče textových nabídek. Jistě si také všimnete, že celá konstrukce je velmi podobná konstrukci **for** s tím rozdílem, že smyčka neprobíhá přes všechna „slova“ v $OPTIONS, ale dotáže se uživatele, kterou větev provést.

Použití příkazového řádku

```
#!/bin/bash
if [ -z "$1" ]; then
    echo usage: $0 directory
    exit
fi
SRCD=$1
TGTD="/var/backups/"
OF=home-$(date +%Y%m%d).tgz
tar -cZf $TGTD$OF $SRCD
```

Mělo by být jasné, co tento skript dělá. Výraz v podmínce testuje, zda program obdržel parametr ($1) a pokud ne, ukončí jej a zobrazí malou nápovědu. Zbytek skriptu už by měl být v této chvíli jasný.

Různé

Při řadě různých příležitostí potřebujete uživatele požádat o zadání nějakého vstupu a existuje několik způsobů, jak to udělat. Toto je jeden z nich:

```
#!/bin/bash
echo Please, enter your name
read NAME
echo "Hi $NAME!"
```

Další možnost je získat příkazem `READ` více vstupních hodnot. Demonstruje to následující příklad:

```
#!/bin/bash
echo Please, enter your firstname and lastname
read FN LN
echo "Hi! $LN, $FN !"
```

Aritmetické operace

V příkazovém řádku nebo v shellu zkuste toto:

```
echo 1 + 1
```

Pokud jste očekávali, že uvidíte `2`, budete zklamáni. Co když ale budete potřebovat, aby BASH vyhodnotil nějaký aritmetický výraz? Řešení vypadá takto:

```
echo $((1+1))
```

Získáme „logičtější“ výsledek. Tímto způsobem se vyhodnocují aritmetické výrazy. Alternativou je zápis

```
echo $[1+1]
```

Pokud chcete pracovat se zlomky nebo potřebujete složitější matematické operace, můžete počítat pomocí programu `bc`.

Pokud byste v příkazovém řádku zadali `echo $[3/4]`, výsledek by byl `0`, protože BASH pracuje pouze s celočíselnou aritmetikou. Pokud zadáte `echo 3/4|bc -l`, pak pravděpodobně dostanete `0.75`.

Jak najít bash

Ze zprávy od Mikeho (viz *Poděkování*).

Ve všech skriptech se uvádí `#!/bin/bash`... možná by byl užitečný příklad jak zjistit, kde je *bash* uložen.

Nejlepší metoda je `locate bash`, program *locate* však není na všech systémech.

Obvykle bude z kořenového adresáře fungovat `find ./ -name bash`.

Doporučená místa k prohlédnutí:

ls -l /bin/bash

ls -l /sbin/bash

ls -l /usr/local/bin/bash

ls -l /usr/bin/bash

ls -l /usr/sbin/bash

ls -l /usr/local/sbin/bash

(Nedokážu si představit žádný jiný rozumný adresář, na různých systémech byl *bash* vždy v jednom z nich.)

Můžete také zkusit `which bash`.

Získání návratové hodnoty programu

Návratová hodnota spuštěného programu se ukládá ve speciální proměnné `$?`.

Následující skript demonstruje, jak získat návratovou hodnotu. Předpokládáme, že adresář *dada* neexistuje. (Příklad rovněž navrhl Mike.)

```
#!/bin/bash
cd /dada &> /dev/null
echo rv: $?
cd $(pwd) &> /dev/null
echo rv: $?
```

Zachycení výstupu příkazu

Následující skript vypíše všechny tabulky ve všech databázích (za předpokladu, že máte nainstalováno MySQL). Zkuste změnit příkaz *mysql* tak, aby používal platné uživatelské jméno a heslo.

```
#!/bin/bash
DBS=`mysql -uroot -e"show databases"`
for b in $DBS ;
do
    mysql -uroot -e"show tables from $b"
done
```

Více zdrojových souborů

Skript může být rozdělen do více souborů...

Tabulky

Operátory porovnávání řetězců

Příklady porovnávání řetězců

Porovnání dvou řetězců:

```
#!/bin/bash
S1='string'
S2='String'
if [ $S1=$S2 ];
then
    echo "S1('$S1') is not equal to S2('$S2')"
fi
if [ $S1=$S1 ];
then
    echo "S1('$S1') is equal to S1('$S1')"
fi
```

Doplňuji zde poznámku, kterou poslal Andreas Beck a která se vztahuje k zápisu *if [$S1 = $S2]*.

Není to příliš vhodné, protože pokud je *$S1* nebo *$S2* prázdný, dojde k chybě. Rozumnější je zápis *x$1=x$2* nebo *"$1"="$2"*.

Aritmetické operátory

Aritmetické relační operátory

Užitečné příkazy

Tuto část přepsal Kees (viz *Poděkování*).

Některé z následujících příkazů obsahují prakticky úplné vlastní programovací jazyky. Uvedeme si zde pouze základní popis vybraných příkazů. Podrobnější informace získáte na příslušných manuálových stránkách.

sed (dávkový editor)

Sed je neinteraktivní editor. Namísto modifikace souboru prostřednictvím pohybu kurzoru po obrazovce zadáte programu **sed** název modifikovaného souboru a skript s editačními příkazy. Sed můžete chápat také jako filtr. Podívejme se na některé příklady:

```
$sed 's/to_be_replaced/replaced/g' /tmp/dummy
```

Tento příkaz nahradí řetězec *to_be_replaced* řetězcem *replaced*, přičemž vstup čte ze souboru */tmp/dummy*. Výsledek bude poslán na standardní výstup (typicky tedy na konzolu), pomocí přesměrování symbolem > však můžete výstup zapsat do dalšího souboru.

```
$sed 12,18d /tmp/dummy
```

Dojde k vypsání všech řádků kromě řádků 12 až 18. Původní soubor přitom nebude změněn.

awk
(manipulace s datovými soubory, čtení a zpracování textu)

Existuje řada implementací jazyka AWK (nejznámější jsou GNU *gawk* a „new awk" *mawk*). Princip je jednoduchý: AWK hledá zadaný vzor a pro každou shodu se vzorem provede určitou akci.

Vytvořili jsme si soubor */tmp/dummy*, který obsahuje následující řádky:

test123

test

tteesstt

```
$awk '/test/ {print}' /tmp/dummy
vypíše
test123
test
```

Vyhledávaným vzorem je *test* a operace, která se má provést s daným řetězcem je vytisknout jej.

```
$awk '/test/ {i=i+1} END {print i}' /tmp/dummy
2
```

Pokud chcete najít více vzorů, je rozumné nahradit text v apostrofech výrazem *–f file.awk* a pak všechny vzory a operace definovat v souboru *file.awk*.

grep (tisk řádků odpovídajících hledanému vzoru)

Několik příkladů příkazu *grep* jsme už viděli, jedná se o příkaz, který vypíše řádky odpovídající zadanému vzoru. Dokáže však ještě víc.

```
$grep "look for this" /var/log/messages -c
12
```

V souboru /var/log/messages byl řetězec look for this nalezen dvanáctkrát. (Dobře, tenhle příklad je podvrh, museli jsme soubor /var/log/messages modifikovat.)

wc (počítání řádků, slov a bajtů)

V následujícím příkladu uvidíme, že výstup nebude odpovídat našim očekáváním. Soubor *dummy*, s nímž pracujeme, obsahuje text *bash introduction howto test file*.

```
$wc --words --lines --bytes /tmp/dummy
2 5 34 /tmp/dummy
```

Příkaz wc se nestará o pořadí parametrů. Spočítané hodnoty vypisuje vždy v neměnném pořadí, tedy řádky, slova a bajty.

sort (seřazení řádků textu)

Tentokrát obsahuje náš soubor text

b
c
a

```
$sort /tmp/dummy
```

Výstup bude

a
b
c

s1=s2	s1 se shoduje s s2
s1!=s2	s1 se neshoduje s s2
s1<s2	...
s1>s2	...
-n s1	s1 není prázdný (obsahuje jeden nebo více znaků)
-z s1	s1 je prázdný

Žádný příkaz by neměl být tak jednoduchý :-).

bc (kalkulačka)

Příkaz *bc* přijímá vstup z příkazového řádku (nebo ze souboru, ne z přesměrování nebo roury), případně i z uživatelského rozhraní. Následující demonstrace předvede některé příklady. Všimněte si, že *bc* spustíme s parametrem `-q`, aby nevypisoval uvítací hlášení. (Kurzívou jsou uváděny texty vytisknuté programem.)

```
$bc -q

1 == 5
0
0.05 == 0.05
1
5 != 5
0
2 ^ 8
256
sqrt(9)
3
while (i != 9) {
```

+
-
*
/
% (zbytek po dělení)

```
i = i + 1;
print i
```

-lt	<
-gt	>
-le	<=
-ge	>=
-eq	==
-ne	!=

```
}
123456789
quit
```

tput (inicializace terminálu nebo dotaz na databázi terminfo)

Malá demonstrace možností programu *tput*:

```
$tput cup 10 4
```

Prompt se objeví na souřadnicích (y10, x4).

```
$tput reset
```

Vymaže obrazovku a zobrazí prompt na (y1, x1). Levý horní roh je (y0, x0).

```
$tput cols
80
```

Vypíše počet znaků na řádku.

Vřele doporučujeme se (přinejmenším) s tímto programem seznámit. Existuje spousta malých programků, které vám dovolí dělat na příkazové řádce hotové divy.

Některé příklady jsme převzali z manuálu nebo z dokumentů FAQ.

Další skripty

Použití příkazu na všechny soubory v adresáři

Příklad: Velmi jednoduchý zálohovací skript (*vylepšený*)

```
#!/bin/bash
SRCD="/home/"
TGTD="/var/backups/"
OF=home-$(date +%Y%m%d).tgz
tar -cZf $TGTD$OF $SRCD
```

Přejmenování souborů

```
#!/bin/sh
# renna: přejmenuje více souborů podle zadaných pravidel
# napsal felix hudson Jan - 2000
```

```
# nejprve ověříme různé "režimy" tohoto programu
# pokud první parametr ($1) vyhovuje podmínce, provedeme příslušnou
# část programu a skončíme

# budeme přejmenovávat prefix?
if [ $1 = p ]; then

# nyní máme v ($1) režim a ve ($2) prefix
  prefix=$2 ; shift ; shift

# rychlá kontrola, jestli byly zadány názvy souborů
# pokud ne, raději neuděláme nic, než abychom přejmenovali neexistující soubo-
ry!!
  if [$1 = ]; then
    echo "no files given"
    exit 0
  fi

# následující smyčka prochází přes všechny zadané soubory
# pro každý soubor se provede přejmenování
  for file in $*
  do
    mv ${file} $prefix$file
  done

# a ukončíme program
  exit 0
fi

# budeme přejmenovávat suffix?
# zbytek je prakticky shodný s předchozím kódem, komentáře viz výše
if [ $1 = s ]; then
  suffix=$2 ; shift ; shift

  if [$1 = ]; then
    echo "no files given"
    exit 0
  fi

  for file in $*
  do
    mv ${file} $file$suffix
  done

  exit 0
fi

# bude přejmenovávat celé soubory?
if [ $1 = r ]; then
  shift

# následující kousek slouží čistě jako ochrana, abychom nepoškodili
# žádné soubory pokud uživatel nespecifikuje, co se má udělat
```

```
  if [ $# -lt 3 ] ; then
    echo "usage: renna r [expression] [replacement] files... "
    exit 0
  fi

# smazání ostatních informací
  OLD=$1 ; NEW=$2 ; shift ; shift

# následující smyčka prochází přes všechny zadané soubory a přejmenovává
# je pomocí programu sed
# jde o jednoduchý příkaz, který čte standardní vstup a nahrazuje
# jeden řetězec jiným
# my mu předáváme název souboru (na standardní vstup) a nahrazujeme
# požadovaný text
  for file in $*
  do
    new=`echo ${file} | sed s/${OLD}/${NEW}/g`
    mv ${file} $new
  done
  exit 0
fi

# pokud jsme se dostali až sem, nebyly programu předány žádné
# rozumné parametry, takže řekneme uživateli, co má dělat
echo "usage;"
echo " renna p [prefix] files.."
echo " renna s [suffix] files.."
echo " renna r [expression] [replacement] files.."
exit 0

# konec
```

Přejmenování souborů (jednoduché)

```
#!/bin/bash
# renames.sh
# jednoduchý přejmenovávací filtr

criteria=$1
re_match=$2
replace=$3

for i in $( ls *$criteria* );
do
  src=$i
  tgt=$(echo $i | sed -e "s/$re_match/$replace/")
  mv $src $tgt
done
```

Když něco nefunguje (ladění)

Jak spustit BASH

Rozumné je zadat na první řádek

```
#!/bin/bash -x
```

Díky tomu uvidíme spoustu zajímavých informací navíc.

O tomto dokumentu

Uvítáme všechny opravy a doplňky, případně další náměty na to, co by se mělo v dokumentu objevit. Budu jej aktualizovat, jak budu moci.

Bez záruky

Dokument je poskytován bez jakýchkoliv záruk.

Překlady

italština: William Ghelfi (*wizzy@tiscalinet.it*), *http://web.tiscalinet.it/penguin_rules/*

francouzština: Laurent Martelli

korejština: Minseok Park, *http://kldp.org/*

korejština: Chun Hye Jin

španělština: anonym, *http://www.insflug.org/*

Určitě existují i další překlady, nemám však o nich informace. Pokud o nějakém víte, laskavě mě informujte.

Poděkování

- Překladatelům do jiných jazyků (viz předchozí odstavec).
- Nathanu Hurstovi za řadu oprav.
- Jonu Abbottovi za komentáře k vyhodnocování aritmetických výrazů.
- Felixi Hudsonovi za napsání skriptu *renna*.
- Keesi van der Broekovi za řadu oprav a přepsání části věnované užitečným programům.
- Mikeovi za části věnované nalezení *bash*e a za testovací soubory.
- Fieshovi za pěkný nápad do části o smyčkách.
- Lionovi za zmínku a časté chybě (`./hello.sh: Command not found`).
- Andreasi Beckovi za opravy a komentáře.

Historie

Doplnění nových překladů a drobné opravy.

Doplněna Keesova část věnovaná užitečným příkazům.

Další opravy a doplnění.

[1] Pozn. překladatele: V tomto případě není archiv komprimován programem *gzip*, ale programem *compress*.

Dodány příklady porovnávání řetězců.

v0.8 zrušeno číslování verzí, datum by snad mělo stačit

v0.7 Další opravy a dopsání některých TO-DO částí

v0.6 Drobné opravy

v0.5 Přidána část o přesměrování

v0.4 Díky bývalému šéfovi byl dokument odstraněn z původního umístění a byl přesunut na správné místo na *www.linuxdoc.org*.

předchozí: nepamatuji si a nepoužívám rcs ani cvs :-(

Další informace

Introduction to bash (under BE), *http://org.laol.net/lamug/beforever/bashtut.htm*

Bourne Shell Programming, *http://207.213.123.70/book/*

Sestavení a instalace softwarových balíků pro Linux

Originál: http://tldp.org/HOWTO/Software-Building-HOWTO.html

Tato vyčerpávající příručka popisuje postup pro sestavení a instalaci „obecných" programů v Linuxu. Kromě toho částečně hovoří i o binárních distribucích programů v balíčcích *rpm* a *deb*.

Úvod

Řada programů pro různé Unixy a Linuxy se distribuuje jako komprimovaný archiv zdrojových souborů. Stejný program tak lze sestavit na různých cílových platformách, což autorovi šetří práci s vytvářením různých cílových verzí. Jedna a táž distribuce programu tak může posléze běžet na procesorech Intel, Alpha nebo RISC. Na druhé straně se tak proces sestavení a instalace přenáší na koncového uživatele – faktického správce systému, toho, který sedí u klávesnice, tedy na vás. Nic se ale nebojte, celý proces není ani tak hrozný ani záhadný jak vypadá, a tento dokument to jasně ukáže.

Rozbalení souborů

Stáhli jste si nebo jinak získali balík s programem. S největší pravděpodobností je to archiv (vytvořený programem *tar*) a komprimovaný program *gzip* s příponou *.tar.gz* nebo *.tgz* (často se označuje jako „tarball"). Nejprve jej zkopírujte do nějakého pracovního adresáře. Pak jej dekomprimujte a rozbalte. Vhodný příkaz pro tento účel je *tar xvzf soubor*, kde *soubor* je samozřejmě jméno rozbalovaného souboru. Při tomto procesu se obvykle všechny soubory rozbalí do různých podadresářů, které budou automaticky vytvořeny. Stejný postup funguje i pokud soubor s programem končí příponou *.Z*[1]. Průběh celého postupu si můžete zobrazit příkazem *tar tvzf soubor*, kdy uvidíte názvy všech rozbalovaných souborů, aniž by k jejich rozbalení skutečně došlo.

Výše uvedená metoda rozbalení archivu je ekvivalentní následujícím příkazům:

```
gzip -cd filename | tar xvf -
gunzip -c filename | tar xvf -
```

(Parametr „-“ říká programu *tar*, aby vstup četl ze *stdin*.)

Zdrojové soubory dodávané v novém formátu *bzip2* (*.bz2*) je možné rozbalit příkazem *bzip2 -cd soubor | tar xvf -*. Jednodušeji to jde příkazem *tar xyvf soubor* za předpokladu, že máte program *tar* správně opatchován (viz dokument *Bzip2 HOWTO*). Distribuce Debian používá poněkud jinou úpravu programu *tar*, a pak fungují parametry *-I*, *--bzip2* a *--bunzip2*.

Někdy je nutné archivované soubory rozbalit a instalovat z domovského adresáře uživatele, nebo z nějakého jiného konkrétního adresáře, například */*, */usr/src* nebo */opt* – mělo by to být uvedeno v konfiguračních informacích balíku. Pokud při pokusu o rozbalení archivu dojde k nějakým chybám, může to být z tohoto důvodu. Přečtěte si dokumentaci k programu, zejména soubory *README* a/nebo *Install*, a v případě potřeby upravte konfigurační soubory a/nebo *Makefile* podle pokynů v dokumentaci. Za normálních okolností neupravujte soubor *Imake*, protože to může vést k nečekaným efektům. Většina programů umožňuje celý proces automatizovat příkazem *make install*, který zkopíruje jednotlivé binární soubory tam, kam patří.

- Někdy se můžete setkat se soubory shar (shell archives). Používají se z toho důvodu, že jsou normálně čitelné a moderátoři diskusních skupin je mohou procházet a nevhodné vyřadit. Rozbalují se příkazem unshar soubor.shar. Jinak se s nimi pracuje stejně jako s tgz archivy.
- Některé programy se dodávají komprimované pomocí nestandardních nástrojů z DOSu, Maců nebo dokonce Amigy – například zip, arc, lha, arj, zoo, rar a shk. Například na http://metalab.unc.edu/ i na mnoha jiných místech můžete najít nástroje pro rozbalení těchto typů archivů.

V některých případech bude nutné, abyste v rozbalené zdrojové distribuci provedli nějaké úpravy prostřednictvím souborů patch nebo diff, které obsahují požadované změny. Je-li to nutné, byste se měli dozvědět v dokumentaci. Za normálních okolností k tomu použijete program patch, a to příkazem patch < *soubor_s_úpravou*.

Teď můžete začít se sestavováním programu.

Použití programu make

Klíčovým v procesu sestavení je soubor *Makefile*. Ve své nejjednodušší podobě je *Makefile* skript pro překlad a sestavení „binárek“ – spustitelných programů. Dále může *Makefile* poskytovat možnost aktualizace programu bez nutnosti nového překladu všech zdrojových souborů.

V určité fázi sestavovaní spustí *Makefile* program *cc* nebo *gcc*. Je to vlastně preprocesor, překladač a linker. Tento proces vytvoří ze zdrojových souborů binární, spustitelné soubory.

Spuštění programu *make* se obvykle provede pouze příkazem *make*. Obecně by tak mělo dojít k sestavení všech spustitelných souborů daného programového balíku. Příkaz *make* ale může umožňovat i jiné operace, například instalaci příslušných souborů do správných adresářů (*make install*) nebo odstranění nepotřebných objektových souborů (*make clean*). Spuštění příkazu *make* s parametrem *-p* vám umožní prohlédnout si zamýšlený proces sestavení, protože se vypíší jednotlivé operace, které *make* provede, aniž by je skutečně vykonal.

Obecně použitelný soubor *Makefile* používají jen nejjednodušší programy. Složitější programy vyžadují jeho úpravy podle umístění knihoven, hlavičkových souborů a dalších prostředků na daném systému. Dochází k tomu zejména pokud jsou k překladu zapotřebí knihovny X11. Tento úkon provedou příkazy *imake* a *xmkmf*.

Soubor *Imakefile* je, podle manuálových stránek, šablona pro soubor *Makefile*. Program *imake* vytvoří podle souboru *Imakefile* soubor *Makefile* vhodný pro váš systém. Ve většině případů ale bu-

dete asi spouštět skript *xmkmf*, který představuje rozhraní k programu *imake*. Konkrétní pokyny byste měli najít v souborech *README* nebo *INSTALL*, které jsou součástí distribuce programu. (Pokud po rozbalení archivu najdete souboru *Imake*, je to jasná stopa, že máte spustit *xmkmf*.) Podrobnější informace najdete v manuálových stránkách programů *imake* a *xmkmf*.

Příkazy *xmkmf* a *make* možná budete muset spustit jako *root*, zejména pokud spouštíte *make install*, který kopíruje soubory do adresářů */usr/bin* nebo */usr/local/bin*. Spuštění programu *make* jako běžný uživatel v těchto případech povede k chybám, protože nemáte potřebná práva pro zápis do systémových adresářů. Zkontrolujte také, že vytvořené binární soubory mají nastavena práva spuštění pro vás a další uživatele.

Spuštěním programu *xmkmf* dojde k vytvoření souboru *Makefile* vhodného pro váš systém podle pokynů v souboru *Imake*. Typicky budete spouštět *xmkmf* s parametrem *-a*, kdy dojde k automatickému vytvoření *Makefile* a spuštění *make includes* a *make depend*. Tím se nastaví proměnné a nadefinuje se umístění knihoven pro překladač a linker. V některých případech distribuce neobsahuje soubor *Imake*, ale skript *INSTALL* nebo *configure*, které plní stejnou úlohu. Pokud spouštíte skript *configure*, spouštějte jej jako *./configure*, aby opravdu došlo ke spuštění správného skriptu v aktuálním adresáři. Ve většině případů je postup instalace popsán v souboru *README*, který je součástí distribuce programu.

Obvykle je rozumné prohlédnout si soubor *Makefile* vytvořený pomocí *xmkmf* nebo nějakým instalačním skriptem. Soubor by měl obsahovat hodnoty vyhovující vašemu systému, v některých případech však budete možná muset ručně doladit některé detaily.

Instalace nově vzniklých binárních souborů do příslušných systémových adresářů se obvykle provede příkazem *make install* spuštěným jako *root*. Typicky se binární programy na moderních linuxových distribucích ukládají do adresářů */usr/bin*, */usr/X11R6* a */usr/local/bin*. Doporučené umístění je adresář */usr/local/bin*, protože se tak nově instalované programy nemíchají s programy, které jsou součástí vlastní distribuce.

Balíky určené primárně pro různé komerční verze Unixů mohou instalovat programy do adresáře */opt*, nebo jiného neobvyklého adresáře. To může vést k chybám, pokud adresář neexistuje. Nejjednodušší metoda je jako *root* vytvořit adresář */opt*, nechat program nainstalovat a pak příslušný adresář přidat do proměnné PATH prostředí. Alternativně můžete adresář */opt* vytvořit jako symbolický odkaz na adresář */usr/local/bin*.

Obecný postup instalace tedy vypadá takto:

- Přečíst si soubor *README* nebo jinou příslušnou dokumentaci
- Spustit *xmkmf -a*, nebo skript *INSTALL* nebo *configure*
- Je-li to nutné, spustit *make clean, make Makefiles, make includes* a *make depend*
- Spustit *make*
- Ověřit práva souborů
- Je-li to nutné, spustit *make install*

Poznámky:

- Samotné sestavení programů se obvykle neprovádí jako *root*. Práva roota potřebujete pouze k instalaci nově vytvořených programů do systémových adresářů.
- Až se seznámíte s programem *make* a jeho použitím, budete možná chtít překladači gcc prostřednictvím souboru *Makefile* předat vylepšené optimalizační parametry. Mezi běžné optimalizační parametry patří *-O2, -fomit-frame-pointer, -funroll-loops* a *-mpentium* (máte

-li procesor Pentium). Při modifikacích souboru *Makefile* buďte opatrní a používejte zdravý rozum!

- Po vytvoření binárních programů je možná budete chtít „stripnout“. Příkaz *strip* odstraní z binárních souborů symbolické ladicí informace a často dramaticky sníží jejich velikost. Znemožní se tím také samozřejmě ladění programu.
- Projekt *Pack Distribution Project* (http://sunsite.auc.dk/pack/) představuje jinou metodu vytváření archivů softwarových balíků, založenou na skriptovacím jazyce Python, která používá nástroje pro správu symbolických odkazů na soubory instalované v různých adresářích. Tyto archivy jsou normální tgz archivy, ale instalují se do adresáře */coll* a */pack*. Pokud se s takto distribuovanými programy setkáte, budete asi potřebovat balík *Pack-Collection* z výše uvedené adresy.

Předpřipravené binární distribuce

Co je špatného na rpm?

Ruční přeložení a instalace programu ze zdrojové distribuce je pro některé uživatele zjevně natolik strašidelný úkol, že raději používají binární balíčky ve formátu *rpm*, *deb*, případně *slp*. I když instalace *rpm* balíčku může být stejně jednoduchá a rychlá jako instalace programů v některých jiných známých operačních systémech, rozhodně je dobré mít na paměti některé nevýhody automatické instalace předpřipravených binárních balíků.

Nejprve je dobré vědět, že programy bývají distribuovány nejprve jako zdrojové balíky, a vydání binárních balíků bývá zpožděno o několik dní, týdnů, někdy i měsíců. Aktuální *rpm* verze je typicky o několik subverzí pozadu za aktuální zdrojovou verzí. Pokud tedy chcete používat aktuální verze, asi nebudete chtít na uvolnění *rpm* či *deb* balíků čekat. U některých méně populárních programů někdy vůbec nedojde k uvolnění *rpm* verze.

Za druhé zdrojová distribuce je obvykle kompletnější, nabízí více možností a umožňuje lepší úpravy a doladění parametrů. Binární distribuce nemusí obsahovat funkce možnosti zdrojové distribuce. Zdrojový *rpm* balík obsahuje kompletní zdrojový kód a je ekvivalentní *tgz* distribuci. Jeho instalace se typicky provede příkazem *rpm --recompile balík.rpm* nebo *rpm --rebuild balík.rpm*.

Za třetí se binární distribuce nemusí správně nainstalovat, nebo i pokud se nainstaluje, nemusí správně fungovat. Může to být způsobeno různými knihovnami použitými ve vašem systému, nebo může být balík prostě chybně sestaven nebo poškozen. V každém případě při instalaci *rpm* či *deb* balíku se musíte plně svěřit do rukou tomu, kdo balík vytvořil.

A konečně, mít po ruce zdrojový kód programu je užitečné, protože si s ním můžete hrát a něco nového se naučit. Je mnohem rozumnější mít zdrojový kód v tom archivu, z nějž jste program sestavili, než jej mít jako samostatný zdrojový *rpm* balík.

Ani instalace *rpm* balíku nemusí být nutně bezproblémová. Pokud dojde ke konfliktu v závislostech, balík se nenainstaluje. Balík navíc může vyžadovat jiné verze knihoven, než které máte nainstalovány, a instalace nemusí fungovat ani v případě, že chybějící knihovny „vytvoříte“ jako symbolické odkazy na existující knihovny. Instalace z *rpm* balíků i navzdory své jednoduchosti velmi často selže ze stejných důvodů, jak instalace přímo ze zdrojového balíku.

Balíky *rpm* a *deb* musíte instalovat jako *root*, abyste měli potřebná zápisová práva, čímž se otevírají potenciální bezpečnostní díry, protože můžete neúmyslně poškodit systémové binární soubory a knihovny, nebo dokonce nainstalovat trójského koně. Je proto nezbytné stahovat *rpm* a *deb* balíky pouze z „důvěryhodných zdrojů“. V každém případě byste měli před instalací zkontrolovat

digitální podpis balíku a jeho kontrolní součet (příkazem *rpm --checksig balík.rpm*). Rovněž doporučujeme spustit *rpm --K -nopgp balík.rpm*. Pro *deb* balíky použijete příkazy *dpkg -I | --info balík.deb* a *dpkg -e | --control balík.deb*. Příklad:

```
rpm --checksig gnucash-1.1.23-4.i386.rpm

gnucash-1.1.23-4.i386.rpm: size md5 OK

rpm -K --nopgp gnucash-1.1.23-4.i386.rpm

gnucash-1.1.23-4.i386.rpm: size md5 OK
```

Opravdový paranoik (a v tomto případě je paranoia velice zdravá věc) použije nástroje *unrpm* a *rpmunpack* (najdete je v adresáři *utils/package* na ftp://metalab.unc.edu/pub/Linux/utils/package), které slouží k rozbalení a kontrole jednotlivých komponent balíčků.

Klee Diene (klee@debian.org) napsal experimentální program *dpkgcert*, který ověřuje integritu nainstalovaných *.deb* souborů proti MD5 podpisům. Najdete jej na ftp://ftp.debian.org/pub/debian/project/experimental. Na stránkách *Jim Pick Software* najdete databázi *dpkgcert* certifikátů pro balíčky v typické distribuci Debianu.

Ve své nejjednodušší podobě příkazy *rpm -i balík.rpm* a *dpkg --install balík.deb* automaticky rozbalí a nainstalují program. Ovšem pozor – slepé používání těchto příkazů může být pro váš systém krajně nezdravé!

Stejné, i když trochu méně důrazné varování platí i pro instalační nástroj *pkgtool* v distribuci Slackware. Obecně veškeré „automatické“ instalace vyžadují zvýšenou opatrnost.

Programy *martian* a *alien* (http://www.people.cornell.edu/pages/rc42/program/martian.html a http://kitenet.net/programs/alien/) umožňují konverzi mezi formáty *rpm*, *deb*, *slp* a *tgz*. Tak můžete různé balíčky používat na všech distribucích Linuxu.

Pozorně si přečtěte manuálové stránky programu *rpm* a *dpkg*, dokument *RPM HOWTO*, a dále *Quick Guide to Red Hat's Package Manager* (http://www.tfug.org/helpdesk/linux/rpm.html) a *The Debian Package Management Tools* (http://www.debian.org/doc/FAQ/debian-faq-7.html).

Problémy s rpm – příklad

Jan Hubička (hubicka@paru.cas.cz) napsal velmi pěkný program pro práci s fraktály, *xaos*. Na domovské stránce programu (http://www.paru.cas.cz/~hubicka/XaoS) nabízí jak *tgz* archiv, tak i *rpm* verzi. Budeme líní a zkusíme nainstalovat *rpm*, abychom se nemuseli „trápit“ se zdrojovými kódy.

Bohužel *rpm* verze se nepovede nainstalovat. Zkoušeli jsme dvě různé verze, a ani jedna nefungovala:

```
rpm -i --test XaoS-3.0-1.i386.rpm
error: failed dependencies:
        libslang.so.0 is needed by XaoS-3.0-1
        libpng.so.0 is needed by XaoS-3.0-1
        libaa.so.1 is needed by XaoS-3.0-1

rpm -i --test xaos-3.0-8.i386.rpm
error: failed dependencies:
        libaa.so.1 is needed by xaos-3.0-8
```

Zvláštní ovšem je, že knihovny *libslang.so.0*, *libpng.so.0* a *libaa.so.1* všechny v adresáři */usr/lib* najdeme. RPM balíček asi musel být sestaven s poněkud jinými verzemi těchto knihoven, i když distribuční čísla knihoven souhlasí.

Jako test tedy zkusíme instalovat balík *xaos-3.0-8.i386.rpm* s parametrem *--nodeps*, čímž si vynutíme instalaci bez ohledu na „chybějící" knihovny. Po spuštění program nefunguje:

xaos: error in loading shared libraries: xaos: undefined symbol: __fabsl

Zkusme se tedy podrobněji podívat, v čem je problém. Spustíme na program *xaos* nástroj *ldd* a zjistíme, že všechny knihovny, které potřebuje, v systému jsou. Spustíme tedy nástroj *nm* na knihovnu */usr/lib/libaa.so.1* a zjistíme, že opravdu neexportuje symbol *__fabsl*. Samozřejmě mohou chybět i jiné symboly z jiných knihoven... S tím, bohužel, neuděláme vůbec nic, pokud nechceme knihovny měnit (a riskovat zásah do funkčnosti systému).

A dost! Stáhneme si zdrojovou distribuci *XaoS-3.0.tar.gz*, spustíme *./configure*, *make* a jako root pak *make install*. Všechno funguje v pořádku.

To byl jen jeden z mnoha příkladů, kdy předpřipravené binární balíčky způsobí mnohem více problémů, než stojí za to.

Problémy s Termcap a Terminfo

Podle příslušné manuálové stránky je *terminfo* databáze popisující terminály, používaná textovými programy. Definuje obecnou množinu řídicích příkazů, používaných pro zobrazování na terminálech a umožňuje podporovat různé typy terminálů bez potřeby speciálních ovladačů. Knihovny databáze *terminfo* najdete v moderních distribucích Linuxu v adresáři */usr/share/terminfo*.

Databáze *terminfo* prakticky nahradila starší *termcap* a úplně archaický *termlib*. Na instalaci programů to obvykle nemá žádný vliv, s výjimkou programů, které vyžadují přítomnost *termcap*.

Většina distribucí Linuxu dnes používá *terminfo*, stále však obsahuje i starší knihovny *termcap* kvůli kompatibilitě se starými aplikacemi (viz */etc/termcap*). Někdy je nutné kvůli instalaci programů linkovaných s knihovnami *termcap* použít speciální balík pro zajištění kompatibility. V některých velmi zřídkavých případech může být nutné zakomentovat ve zdrojovém kódu programu řádek *#define termcap*. Rozhodující informace na toto téma naleznete v dokumentaci vaší distribuce.

Zpětná kompatibilita s binárními formáty a.out

Velmi zřídka může být nutné použít binární soubory ve formátu *a.out*, buď proto, že není k dispozici zdrojový kód, nebo proto, že z nějakých důvodů nejde program přeložit do formátu ELF.

Protože se to občas stává, instalace založené na ELF prakticky vždy obsahují i kompletní knihovny pro formát *a.out*, uložené v adresáři */usr/i486-linuxaout/lib*. Číslovací schéma těchto knihoven je jiné než u knihoven formátu ELF, čímž se chytře předchází konfliktům. Binární programy ve formátu *a.out* by tak měly být schopny po spuštění bez potíží nalézt potřebné knihovny. Ne vždy tomu ale tak je.

Chcete-li spouštět programy ve formátu *a.out*, musí jádro obsahovat jejich podporu – ať už vestavěnou, nebo jako modul. Navíc některé distribuce Linuxu vyžadují instalaci speciálního balíčku pro zajištění kompatibility – například balíček *xcompat* pro Debian kvůli spouštění X aplikací ve formátu *a.out*.

Příklad

Jerry Smith napsal před několika lety velmi šikovný program *pro správu adres*. Používá sice knihovny Motif, ale existuje i jako staticky linkovaný binární program ve formátu *a.out*. Bohužel pokud byste chtěli zdrojovou verzi přeložit s knihovnami *lesstif*, bude nutná celá řada zásahů do zdrojového kódu. A ještě jednou bohužel, binární verze ve formátu *a.out* skončí na systému používajícím formát ELF s následující chybou:

```
xrolodex: can't load library '//lib/libX11.so.3'
No such library
```

Taková knihovna sice v systému je, v adresáři */usr/i486-linuxaout/lib*, ale *xrolodex* ji z nějakých důvodů nenajde. Jednoduché řešení spočívá ve vytvoření symbolického odkazy v adresáři */lib*:

```
ln -s /usr/i486-linuxaout/lib/X11.so.3.1.0 libX11.so.3
```

Možná bude nutné vytvořit podobné odkazy i pro knihovny *libXt.so.3* a *libc.so.4*. Musíte si dávat velký pozor na to, abyste nepřepsali existující knihovny nebo nevyrobili nějaký konflikt v číslech verzí. Naštěstí nové knihovny formátu ELF používají vyšší čísla verzí než starší knihovny formátu *a.out*, takže by k takovým problémům nemělo docházet.

Po vytvoření všech tří odkazů funguje program *xrolodex* správně.

Program *xrolodex* byl původně distribuován na http://www.spectro.com/, ale zdá se, že už tam není. Můžete jej najít na http://metalab.unc.edu/pub/Linux/apps/reminder/xrolodex.tar.z.

Problémy

Pokud příkazy *xmkmf* a/nebo *make* proběhnou bez chyb, můžete přejít k další části. Bohužel v opravdovém světě jen máloco funguje napoprvé správně.

Chyby při linkování

- Řekněme, že příkaz make skončí chybou *Link error: -lX11: No such file or directory*, i když předtím správně proběhl příkaz xmkmf. Může to znamenat, že soubor *Imake* nebyl správně nastaven. Podívejte se na začátek souboru *Makefile* a hledejte řádky jako:

  ```
  LIB=            -L/usr/X11/lib
  INCLUDE=        -I/usr/X11/include/X11
  LIBS=           -lX11 -lc -lm
  ```

  Přepínače *-L* a *-I* říkají překladači a linkeru, kde mají hledat knihovny a hlavičkové soubory. V tomto příkladu by měly být knihovny X11 v adresáři */usr/X11/lib* a hlavičkové soubory v */usr/X11/include/X11*. Pokud to na vašem počítači neplatí, upravte soubor *Makefile* a zkuste překlad znovu.

- Nedefinované odkazy do matematické knihovny, například:

  ```
  /tmp/cca011551.o(.text+0x11): undefined reference to 'cos'
  ```

  Řešením je explicitně linkovat knihovnu *math* přidáním parametru *-lm* do příznaků LIB nebo LIBS v souboru *Makefile* (viz předchozí příklad).

- Další možnost, pokud *xmkmf* skončí chybou, je zkusit následující skript:

  ```
  make -DUseInstalled -I/usr/X386/lib/X11/config
  ```

  Jedná se o operaci, která by měla být ekvivalentní spuštění *xmkmf*.

- Někdy může problémy vyřešit spuštění příkazu *ldconfig* jako *root*.
- Někdy *Makefile* používá neznámé aliasy knihoven, které jsou v systému přítomny. Například může být požadována knihovna *libX11.so.6*, ovšem takový soubor ani odkaz v adresáři */usr/X11R6/lib* nenajdete. Ovšem je tam knihovna *libX11.so.6.1*. Řešením je příkaz *ln -s /usr/X11R6/lib/libX11.so.6.1 /usr/X11R6/lib/libX11.so.6*. Následně může být nutné spustit *ldconfig*.
- Někdy mohou být k sestavení vyžadovány starší verze knihoven X11R5. Pokud tyto R5 verze v adresáři */usr/X11R6/lib* máte (při instalaci jste si to mohli zvolit), pak stačí jen zkontrolovat, že máte vytvořeny všechny odkazy, které program potřebuje. Knihovny R5 se jmenují *libX11.so.3.1.0*, *libXaw.so.3.1.0* a *libXt.so.3.1.0*. Obecně budete potřebovat odkazy jako *libX11.so.3 -> libX11.so.3.1.0*. Možná bude program potřebovat i odkazy jako *libX11.so -> libX11.so.3.1.0*. Chybějící odkazy samozřejmě vytvoříte příkazem *ln -s libX11.so.3.1.0 libX11.so*.
- Některé programy mohou vyžadovat instalaci novějších verzí jedné nebo více knihoven. Například verze *4.x* balíku StarOffice byly známé tím, že vyžadovaly *libc* verze 5.4.4 nebo vyšší. Dokonce ani novější verze StarOffice 5.0 po nainstalování nefungovala, pokud jste používali novější knihovny *glibc 2.1*. Nové StarOffice 5.1 už naštěstí mají tyto problémy vyřešeny. Používáte-li starší verze StarOffice, musíte jako *root* zkopírovat jednu nebo více potřebných knihoven do příslušných adresářů, smazat staré knihovny a předělat symbolické odkazy (informace najdete v dokumentu *StarOffice mini HOWTO*). Pozor! Postupujte mimořádně opatrně, v případě chyby můžete pokazit celý systém. Nejnovější verze knihoven obvykle najdete na ftp://metalab.unc.edu/pub/Linux/libs.

Další problémy

- Nainstalovaný perlový nebo shellový skript vypisuje hlášení *No such file or directory*. Zkontrolujte přístupová práva, zda je skript spustitelný, a zkontrolujte jeho hlavičku, zda skript spouští správný interpret. Skript může například začínat řádkem:

  ```
  #!/usr/local/bin/perl
  ```

  Pokud máte Perl nainstalován v adresáři */usr/bin* a ne v */usr/local/bin*, skript nebude fungovat. Máte dvě možnosti nápravy – buď upravit hlavičku skriptu na *#!/usr/bin/perl* nebo vytvořit symbolický odkaz, *ln -s /usr/bin/perl /usr/local/bin/perl*.
- Některé X11 programy vyžadují k sestavení knihovny Motif. Standardní distribuce Linuxu tyto knihovny neobsahují a v současné době knihovny stojí kolem 100 až 200 dolarů. (Ve většině případů je ale jde úspěšně nahradit volně šířenými knihovnami Lesstif, http://www.lesstif.org/.) Pokud pro sestavení nějakého programu Motif potřebujete a nemáte jej, možná se vám podaří získat staticky linkované binární verze. Staticky linkovaný program v sobě přímo obsahuje knihovní funkce. Výsledkem je sice větší binární soubor, ale funguje i na systémech, kde potřebné knihovny nejsou.

  Pokud program k sestavení potřebuje knihovny, které v systému nejsou, vzniknou chyby při linkování (undefined reference). Může jít o drahé proprietární knihovny, nebo knihovny nemusejí být z různých důvodů k dispozici. V takových případech bývá nejjednodušší řešení získat staticky linkovanou binárku buď od autora nebo z vašeho LUGu.
- Spuštěním skriptu *configure* vznikne nesmyslný *Makefile*, zjevně se nevztahující k programu, který se snažíte přeložit. Stává se to, pokud spustíte špatný skript *configure* někde jinde ve spouštěcí cestě. Vždy spouštějte *./configure*.

- Většina distribucí Linuxu přešla od starší knihovny *libc 5* ke knihovnám *libc 6 / glibc 2*. Přeložené binární soubory pracující se staršími knihovnami nemusejí po aktualizaci knihoven fungovat. Řešení je buď aplikaci znovu přeložit, nebo získat přeloženou verzi určenou pro nové knihovny. Pokud si aktualizujete systém na *libc 6* a objevují se problémy, podívejte se na dokument *Glibc 2 HOWTO*.

  Mezi různými verzemi *glibs* existují drobné nekompatibility, takže binární soubor sestavený pro *glibc 2.1* nemusí fungovat s *glibc 2.0* a naopak.
- Někdy může být nutné v souboru *Makefile* odstranit parametr *-ansi* překladače. Tím se povolují rozšířené funkce *gcc*, nekompatibilní se standardem ANSI, a umožní se sestavení programů, které je vyžadují.
- Některé programy musí být instalovány jako *setuid root*, aby mohly pracovat s právy superuživatele. Provedete to (jako *root*) příkazem *chmod u+s soubor*, přičemž vlastníkem programu musí být *root*. Tyto problémy se mohou objevovat u programů, které přistupují k hardware (například k modemu, CD-ROM a podobně), nebo v režimu konzoly používají knihovny SVGA. Pokud program funguje spuštěný uživatelem *root* a normálním uživatelům vypisuje chybu *access denied*, jde zřejmě o tento problém.

Upozornění: Program s nastavením bitem *setuid* může představovat bezpečnostní riziko. Program běží s právy superuživatele a může tedy potenciálně napáchat značné škody. Ujistěte se, že víte, co program dělá!

Doladění

Můžete upravit soubor *Makefile* a využít nejvhodnější optimalizace pro váš systém. Například příznakem *-O2* zapnete nejvyšší úroveň optimalizace, příznak *-fomit-frame-pointer* vede ke vzniku menších binárních souborů (které ale nebude možné ladit). Nezkoušejte tyto parametry měnit pokud nevíte, co znamenají, a v každém případě s nimi manipulujte až poté, co se vám podaří přeložit fungující základní verzi programu.

Kde hledat pomoc

Podle mých zkušeností se asi 25 % programů podaří přeložit na první pokus. Zhruba 50 % se podaří přesvědčit, aby se daly přeložit s vynaložením úsilí od kategorie „triviální“ po kategorii „herkulovská“. Stále tedy zbývá početná skupina programů, které se nepovede přeložit vůbec. V těchto případech se můžete pokusit najít binární verze těchto programů v archivech ftp://metalab.unc.edu/ nebo ftp://tsx-11.mit.edu/. Rovněž RedHat (http://redhat.com/) a Debian (http://www.debian.org/) nabízejí rozsáhlé archivy binárních verzí většiny oblíbených programů. Možná vám bude autor programu ochoten poskytnout binární verzi vhodnou pro váš systém.

Pokud používáte předpřipravené binární verze, musíte zkontrolovat, že:

- program je kompatibilní s vaším systémem (např. Intel),
- program je kompatibilní s vaším jádrem,
- máte aktuální knihovny,
- máte příslušné instalační nástroje (*rpm* nebo *deb*).

Pokud všechno selže, zkuste diskusní skupiny jako news://comp.os.linux.x/ nebo news://comp.os.linux.development/.

A pokud nic z toho nepomůže, minimálně jste se ze všech sil snažili a naučili jste se spoustu nových věcí.

Závěrečné kroky

Přečtěte si dokumentaci k programu a zjistěte, zda není nutné nastavit nějaké proměnné prostředí (v souborech *.bashrc* nebo *.cshrc*) a zda není nutné upravit soubory *.Xdefaults* a *.Xresources*.

Aplikace může být dodávána se vzorovým konfiguračním souborem, obvykle se jmenuje *Xfoo.ad*. Upravte jej podle potřeb vašeho systému, přejmenujte jej (*mv*) na *Xfoo* a jako *root* nahrajte do adresáře */usr/lib/X11/app-defaults*. Pokud to neuděláte, program může fungovat divně, nebo vůbec.

Většina distribucí programů obsahuje i manuálové stránky. Jako *root* zkopírujte soubory do příslušných adresářů */usr/man*, */usr/local/man* nebo*/usr/X11R6/man* (*man1* – *man9*) a případně je přejmenujte. Pokud například soubor *Xfoo.man* skončí v adresáři */usr/man/man4*, měl by se jmenovat *Xfoo.4*. Podle konvence patří manuály k uživatelským programům do sekce 1, ke hrám do sekce 6, k administrativním programům do sekce 8. Pokud chcete, můžete si samozřejmě svůj systém nakonfigurovat úplně jinak...

Některé programy se neumějí nainstalovat do příslušných systémových adresářů – soubor *Makefile* neobsahuje volbu *install*. Pokud se s takovým programem potkáte, můžete programy nainstalovat ručně zkopírováním do vhodného systémového adresáře – */usr/bin*, */usr/local/bin* nebo */usr/X11R6/bin*, samozřejmě jako *root*. Pro programy, které nejsou standardní součástí distribuce, je doporučeným adresářem */usr/local/bin*.

Většinu z výše uvedeného ve většině případů automaticky řeší příkaz *make install*, případně *make install.man* nebo *make install_man*. Měli byste to zjistit v souborech *README* nebo *INSTALL* daného programu.

První příklad: Xscrabble

Přišlo mi, že program *Xscrabble* Matta Chapmana bych měl rozhodně mít, protože jsem vášnivý hráč této hry. Stáhl jsem si jej, rozbalil a přeložil postupem uvedeným v souboru *README*:

```
xmkmf
make Makefiles
make includes
make
```

Samozřejmě to nefungovalo...

```
gcc -o xscrab -O2 -O -L/usr/X11R6/lib
init.o xinit.o misc.o moves.o cmove.o main.o xutils.o mess.o popup.o
widgets.o display.o user.o CircPerc.o
-lXaw -lXmu -lXExExt -lXext -lX11 -lXt -lSM -lICE -lXExExt -lXext -lX11
-lXpm -L../Xc -lXc

BarGraf.o(.text+0xe7): undefined reference to `XtAddConverter'
BarGraf.o(.text+0x29a): undefined reference to `XSetClipMask'
BarGraf.o(.text+0x2ff): undefined reference to `XSetClipRectangles'
BarGraf.o(.text+0x375): undefined reference to `XDrawString'
BarGraf.o(.text+0x3e7): undefined reference to `XDrawLine'
atd.
atd.
```

```
atd...
```

Zmínil jsem se o tom v konferenci news://comp.os.linux.x/, a někdo mě upozornil, že při linkování nebyly zjevně nalezeny knihovny Xt, Xaw, Xmu a X11. Hmmm...

Program používal dva hlavní soubory *Makefile*, a ten v adresáři *src* mě zaujal. Jeden řádek definoval *LOCAL_LIBS* jako *LOCAL_LIBS = $(XAWLIB) $(XMULIB) $(XTOOLLIB) $(XLIB)*. To tedy byly odkazy na knihovny, které linker nenašel.

Hledal jsem další výskyt *LOCAL_LIBS* a na řádku 495 jsem našel

```
$(CCLINK) -o $@ $(LDOPTIONS) $(OBJS) $(LOCAL_LIBS) $(LDLIBS)
$(EXTRA_LOAD_FLAGS)
```

Co jsou zač ty *LDLIBS*?

```
LDLIBS = $(LDPOSTLIB) $(THREADS_LIBS) $(SYS_LIBRARIES) $(EXTRA_LIBRARIES)
```

Hodnota *SYS_LIBRARIES* byla:

```
SYS_LIBRARIES = -lXpm -L../Xc -lXc
```

Aha! Tady máme ty chybějící knihovny.

Linker asi potřebuje vidět *LDLIBS* dříve než *LOCAL_LIBS*. Zkusil jsem tedy upravit *Makefile* tak, že jsem prohodil pořadí proměnných *LOCAL_LIBS* a *LDLIBS* na řádku 495. Ten nyní vypadal takto:

```
$(CCLINK) -o $@ $(LDOPTIONS) $(OBJS) $(LDLIBS) $(LOCAL_LIBS)
$(EXTRA_LOAD_FLAGS)
```

Znovu jsem spustil překlad a – ono to fungovalo. Samozřejmě byly nutné ještě některé další úpravy, například přejmenování slovníku, ale od té doby mi program přinesl spoustu zábavy.

(Nová verze programu *Xscrabble* je k dispozici i ve formátu RPM a instalace proběhne bezproblémově.)

Xscrabble si můžete stáhnout na adrese http://www.belgarath.demon.co.uk/programs/index.html.

Druhý příklad: Xloadimage

Tento příklad představuje jednodušší problém. Program *xloadimage* vypadal jako užitečný přírůstek do mé sbírky grafických nástrojů. Soubor *xloadi41.gz* jsem si zkopíroval přímo z CD přiloženého k vynikající knize *X User Tools*. Rozbalil jsem jej samozřejmě příkazem *tar xvzf*, ovšem příkaz *make* vyhlásil jakési škaredé chyby a skončil:

```
gcc -c -O -fstrength-reduce -finline-functions -fforce-mem
-fforce-addr -DSYSV  -I/usr/X11R6/include
-DSYSPATHFILE=\"/usr/lib/X11/Xloadimage\" mcidas.c

In file included from /usr/include/stdlib.h:32,
                 from image.h:23,
                 from xloadimage.h:15,
                 from mcidas.c:7:
/usr/lib/gcc-lib/i486-linux/2.6.3/include/stddef.h:215:
conflicting types for `wchar_t'
/usr/X11R6/include/X11/Xlib.h:74: previous declaration of
`wchar_t'
make[1]: *** [mcidas.o] Error 1
make[1]: Leaving directory
```

```
`/home/thegrendel/tst/xloadimage.4.1'
make: *** [default] Error 2
```

Chybové hlášení obsahovalo hlavní stopu. Podívejme se na soubor *image.h*, řádek 23...

```
#include <stdlib.h>
```

Aha, někde ve zdrojovém kódu programu *xloadimage* došlo k předefinování *wchar_t* na něco jiného, než je uvedeno ve standardním hlavičkovém souboru *stdlib.h*. Jako první jsem zkusil zakomentovat řádek 23 souboru *image.h*, co kdyby *stdlib.h* vůbec nebylo zapotřebí.

Tentokrát překlad proběhl bez chyb a program správně funguje.

Třetí příklad: Fortune

Tento příklad vyžaduje jisté znalosti jazyka C. Většina programů pro UNIX/Linux je psána v jazyce C a alespoň základní znalosti tohoto jazyka jsou nesmírně užitečné, pokud to s Linuxem myslíte vážně. Známý program *fortune* vypisuje vždy při startu systému nějaké vtipné pořekadlo. Bohužel, pokus o překlad programu na distribuci RedHat s jádrem 2.0.30 skončil chybou:

```
~/fortune# make all

gcc -O2 -Wall -fomit-frame-pointer -pipe   -c fortune.c -o
fortune.o
fortune.c: In function `add_dir':
fortune.c:551: structure has no member named `d_namlen'
fortune.c:553: structure has no member named `d_namlen'
make[1]: *** [fortune.o] Error 1
make[1]: Leaving directory `/home/thegrendel/for/fortune/fortune'
make: *** [fortune-bin] Error 2
```

Podíváme se na *fortune.c*, inkriminované řádky vypadají takto:

```
if (dirent->d_namlen == 0)
    continue;
name = copy(dirent->d_name, dirent->d_namlen);
```

Musíme najít strukturu *dirent*, ta však není v souboru *fortune.c* deklarována, a (jak nám řekne *grep*), není deklarována ani v žádném jiném ze zdrojových souborů. Na začátku souboru *fortune.c* ale najdeme následující řádek:

```
#include <dirent.h>
```

To vypadá na systémový hlavičkový soubor, který budeme logicky hledat v adresáři */usr/include*. V adresáři */usr/include* sice najdeme soubor *dirent.h*, ani v něm však není struktura *dirent* deklarována. Obsahuje však odkaz na další soubor:

```
#include <linux/dirent.h>
```

Nakonec se tedy dostáváme k souboru /usr/include/linux/dirent.h, kde najdeme kýženou deklaraci:

```
struct dirent {
    long            d_ino;
    __kernel_off_t  d_off;
    unsigned short  d_reclen;
    char            d_name[256]; /* We must not include limits.h! */
};
```

Deklarace struktury zjevně neobsahuje položku *d_namelen*, jsou zde ale „kandidáti“ na její ekvivalent. S největší pravděpodobností to bude *d_reclen*, kterážto položka pravděpodobně reprezentuje délku čehosi a je typu *short int*. Druhý kandidát, *d_ino*, bude asi číslo inode, můžeme odhadnout z jejího názvu a typu. Vypadá to, že se bavíme se strukturou „položka adresáře“ a jednotlivé prvky představují atributy souboru – název, inode a délka.

Zkusme upravit soubor *fortune.c* a změnit dva odkazy na *d_namelen* na *d_reclen*. Zkusíme znovu překlad. Úspěch.

Čtvrtý příklad: Hearts

Bavíme se o staříčkých herckách, které pro Unix napsal někdy v 80. letech Bob Ankeney, v roce 1992 je vylepšil Mike Yang a momentálně je udržuje Jonathan Badger. Předchůdcem této hry byl ještě starší program v Pascalu Dona Backuse, později upravený Jeffem Hemmerlingem. Tato hra byla původně určena pro více hráčů, funguje však i s jedním hráčem proti počítačovým protivníkům. Grafika je pěkná, i když hře chybí některé sofistikované funkce a inteligence počítače taky není nejlepší. Tak či tak, zdá se, že jde pořád o jedinou existující implementaci hercek na Linuxu.

Vzhledem ke stáří programu jde očekávat, že jeho překlad nebude jednoduchý. Vyžadoval zdlouhavé a složité řešení různých úskalí. Je to příklad trpělivosti.

Než začnete, zkontrolujte si, že máte nainstalovány knihovny *motif* nebo *lesstif*.

```
xmkmf
make
client.c: In function `read_card':
client.c:430: `_tty' undeclared (first use in this function)
client.c:430: (Each undeclared identifier is reported only once
client.c:430: for each function it appears in.)
client.c: In function `scan':
client.c:685: `_tty' undeclared (first use in this function)
make: *** [client.o] Error 1
```

A toto je viník v souboru *client.c*:

```
#ifndef SYSV
    (buf[2] != _tty.sg_erase) && (buf[2] != _tty.sg_kill)) {
#else
    (buf[2] != CERASE) && (buf[2] != CKILL)) {
#endif
```

Přidejte v souboru *client.c* na řádek 39 příkaz

```
#define SYSV
```

Tím obejdeme odkaz na *_tty*.

```
make
client.c:41: sys/termio.h: No such file or directory
make: *** [client.o] Error 1
```

Hlavičkový soubor *termio.h* je na Linuxu v adresáři */usr/include* a ne v */usr/include/sys*, jak to bylo na starších Unixech. Změníme tedy řádek 41 souboru *client.c* z

```
#include <sys/termio.h>
```

na

```
#include <termio.h>
make
gcc -o hearts -g      -L/usr/X11R6/lib client.o hearts.o select.o connect.o
sockio.o start_dist.o  -lcurses -ltermlib
/usr/bin/ld: cannot open -ltermlib: No such file or directory
collect2: ld returned 1 exit status
make: *** [hearts] Error 1
```

Moderní distribuce Linuxu používají databáze *terminfo* a/nebo *termcap* namísto archaické *termlib*. Upravíme *Makefile*.

Řádek 655:

```
CURSES_LIBRARIES = -lcurses -ltermlib
```

změníme na

```
CURSES_LIBRARIES = -lcurses -ltermcap
```

```
make
gcc -o xmhearts -g      -L/usr/X11R6/lib xmclient.o hearts.o select.o
connect.o sockio.o start_dist.o gfx.o  -lXm_s -lXt -lSM -lICE -lXext -lX11
-lPW
/usr/bin/ld: cannot open -lXm_s: No such file or directory
collect2: ld returned 1 exit status
```

Hlavní knihovna *lesstif* je *libXm* a ne *libXm_s*. Upravíme tedy *Makefile*.

Řádek 653:

```
XMLIB = -lXm_s $(XTOOLLIB) $(XLIB) -lPW
```

změníme na

```
XMLIB = -lXm $(XTOOLLIB) $(XLIB) -lPW
```

```
make
gcc -o xmhearts -g      -L/usr/X11R6/lib xmclient.o hearts.o select.o
connect.o sockio.o start_dist.o gfx.o  -lXm -lXt -lSM -lICE -lXext -lX11 -lPW
/usr/bin/ld: cannot open -lPW: No such file or directory
collect2: ld returned 1 exit status
make: *** [xmhearts] Error 1
```

Prověříme klasické podezřelé a zjistíme, že nemáme knihovnu PW. Upravíme *Makefile*.

Řádek 653

```
XMLIB = -lXm $(XTOOLLIB) $(XLIB) -lPW
```

změníme na

```
XMLIB = -lXm $(XTOOLLIB) $(XLIB) -lPEX5
(Knihovna PEX5 se tak asi nejvíc podobá knihovně PW.)
make
rm -f xmhearts
gcc -o xmhearts -g      -L/usr/X11R6/lib xmclient.o hearts.o select.o
connect.o sockio.o start_dist.o gfx.o  -lXm -lXt -lSM -lICE -lXext -lX11 -lPEX5
```

Překlad se konečně podařil!!!

```
Instalace. Jako root:
```

```
[root@localhost hearts]# make install
install -c -s  hearts /usr/X11R6/bin/hearts
install -c -s  xmhearts /usr/X11R6/bin/xmhearts
install -c -s  xawhearts /usr/X11R6/bin/xawhearts
install in . done
Zkusíme program spustit
localhost:~/% xmhearts
Can't invoke distributor!
```

Podíváme se do souboru README: „Uložte soubory *heartsd*, *hearts_dist* a *hearts.instr* do adresáře *HEARTSLIB* definovaného v *local.h* a nastavte jim práva pro čtení."

Ze souboru *local.h*:

```
/* where the distributor, dealer and instructions live */

#define HEARTSLIB "/usr/local/lib/hearts"
```

Klasická chyba mezi židlí a klávesnicí. Jako *root*:

```
cd /usr/local/lib
mkdir hearts
cd !$
cp /home/username/hearts/heartsd .
cp /home/username/hearts/hearts_dist .
cp /home/username/hearts/hearts.instr .
```

Opět zkusíme program spustit.

```
xmhearts
```

Někdy funguje dobře, ale většinou zhavaruje s hlášením „dealer died". „Distributor" a „dealer" hledají na hardwarových portech. Mohlo by to znamenat, že tyto programy potřebují práva superuživatele. Jako *root* zkusíme

```
chmod u+s /usr/local/lib/heartsd
chmod u+s /usr/local/lib/hearts_dist
```

(Jak už jsme říkali, *suid root* programy představují bezpečnostní riziko.)

```
xmhearts
```

A všechno funguje! Hercky si můžete stáhnout na adrese ftp://metalab.unc.edu/pub/Linux/games/multiplayer/hearts.tgz.

Pátý příklad: XmDipmon

Program XmDipmon je šikovná malá aplikace zobrazující tlačítko, které znázorňuje status připojení k Internetu. Bliká a pípá, když se připojení přeruší, což na bídných telefonních linkách bývá dost často. Bohužel, funguje pouze s programem *dip* a je tedy k ničemu většině lidí, kteří se k Internetu připojují programem *chat*.

Překlad programu *XmDipmon* není problém. Používá knihovny *Motif*, stejně dobře ale funguje s knihovnami *Lesstif*. Cílem je upravit program tak, aby pracoval společně s programem *chat*. To samozřejmě vyžaduje jisté zásahy do zdrojového kódu a neobejde se to bez znalostí programování.

Citujeme ze souboru *README* programu *XmDipmon*: „Po spuštění hledá program soubor */etc/dip.pid* (může hledat jiný soubor nastavený přepínačem *-pidfile*). Tento soubor obsahuje PID démona *dip* (*dip* se po navázání spojení přepíná do režimu démona).“

Pomocí přepínače *-pidfile* můžeme program donutit hledat jiný soubor, takový, který existuje po navázání připojení programem *chat*. Zjevným kandidátem je zamykací soubor modemu. Můžete tedy zkusit program spustit příkazem *xmdipmon -pidfile /var/lock/LCK..ttyS3* (za předpokladu, že modem je na portu *ttyS3*). Tím ale vyřešíme pouze část problému. Program na základě znalosti PID monitoroval démona *dip* a my jej potřebujeme upravit tak, aby periodicky testoval stav procesu programu *chat* nebo *ppp*.

Program je celý uložen v jediném zdrojovém souboru, který je naštěstí dobře komentovaný. Při prohlížení souboru *xmdipmon.c* narazíme na funkci *getProcFile*, u které se říká:

```
/*****
* Name:                 getProcFile
* Return Type:  Boolean
* Description:  tries to open the /proc entry as read from the dip pid file.
<snip>
*****/
```

Jsme na horké stopě! Podívejme se na tělo funkce:

```
/* we watch the status of the real dip deamon */
sprintf(buf, "/proc/%i/status", pid);
procfile = (String)XtMalloc(strlen(buf)*sizeof(char)+1);
strcpy(procfile, buf);
procfile[strlen(buf)] = '\0';
```

Klíčový je řádek 2 383:

```
sprintf(buf, "/proc/%i/status", pid);
```

Ten testuje, zda proces démona *dip* běží. Jak jej tedy změníme, aby namísto toho monitoroval démona *pppd*?

Podívejme se na manuálovou stránku programu *pppd*:

```
FILES
/var/run/pppn.pid (BSD or Linux), /etc/ppp/pppn.pid (others)
Process-ID for pppd process on ppp interface unit n.
```

Změníme tedy řádek 2 383 takto:

```
sprintf(buf, "/var/run/ppp0.pid" );
```

Zkusíme upravený program přeložit – bez problémů. A teď jej otestujeme. Funguje nádherně. Malé modré tlačítko indikuje navázání připojení a začne blikat a pípat, když se připojení přeruší. Máme tedy plně funkční monitor programu *chat*.

Program můžete najít na adrese http://www.xs4all.nl/~ripley/RSD/linux.html.

Odkud brát programy

Teď jste jistě celí žhaví nově nabyté vědomosti použít k doplnění vašeho systému o spoustu různých programků a dalších vylepšení. Můžete je najít na adresách http://www.redhat.com/linux-info/linux-app-list/linapps.html, nebo na některém z CD-ROM archivů společností http://www.red-

hat.com/, http://www.infomagic.com/, http://www.lsl.com/, http://www.cheapbytes.com/ a dalších.

Vyčerpávajícím zdrojem je archiv ftp://ftp.vix.com/pub/usenet/comp.sources.unix/.

Řada zdrojových kódů se objevuje na konferenci news://alt.sources/. Pokud hledáte konkrétní program, zkuste se zeptat na news://alt.sources.wanted/. Dalším dobrým zdrojem je konference news://comp.os.linux.announce/. Můžete se také přihlásit do poštovní konference unix-sources@pa.dec.com – stačí poslat e-mail s textem *subscribe*.

Archivy konference news://alt.sources/ najdete na adresách ftp://ftp.sterling.com/usenet/alt.sources/, ftp://wuarchive.wustl.edu/usenet/alt.sources/articles a ftp://src.doc.ic.ac.uk/usenet/alt.sources/articles.

KAPITOLA 8

Tisk v Linuxu

Originál: http://www.tldp.org/HOWTO/Printing-HOWTO/

Tento dokument představuje souhrn informací o tom, jak v Linuxu (a obecně i jiných Unixech) cokoliv generovat, prohlížet, faxovat a tisknout.

Úvod

Tento dokument by měl obsahovat všechno, co potřebujete vědět k nastavení tiskových služeb na linuxovém stroji. Život tomu chtěl, že je to o něco složitější než v klikacím světě Microsoftu a Apple, na druhé straně je to ale o něco pružnější a určitě jednodušší na správu ve velkých sítích.

Dokument je strukturován tak, že většině lidí bude stačit přečíst jenom první polovinu, nebo tak nějak. Většina obskurních a na situaci závislých informací je uvedena ve druhé polovině a lze je snadno najít v obsahu. Informace do kapitoly 8 nebo 9 jsou pravděpodobně užitečné téměř pro každého.

Rychlý začátek

Nejrychlejší metoda jak začít pravděpodobně bude použít konfigurační nástroje dodávané s vaším systémem. Za předpokladu, že obsahují podporu vašeho ovladače a ovladač vaší tiskárny, mělo by být docela snadné takto rychle provést základní nastavení. Informace o konfiguračních nástrojích různých dodavatelů najdete v kapitole 9.

Pokud konfigurační nástroje dodavatele nefungují, měli byste zjistit, zda vaše tiskárna vůbec fungovat *může*. V části 5.3.1 najdete odkaz na seznam kompatibilních tiskáren.

Pokud tiskárna má s určitým ovladačem fungovat, zjistěte, zda jej máte nainstalován, a pokud ne, nainstalujte jej. Typicky byste měli být schopni najít upravený balík Ghostscript obsahující novější verzi tohoto programu a doplňující ovladače. Pokud jej nenajdete, můžete si jej přeložit sami – tento proces není triviální, ale je dobře dokumentovaný. Podrobnosti o systému Ghostscript najdete v kapitole 10.

Po instalaci správného ovladače zkuste tiskárnu znovu nakonfigurovat pomocí nástrojů dodavatele. Nepovede-li se to, zkuste jiný nástroj podle doporučení v kapitole 8. Pokud nepomůže ani to, budete muset provést konfiguraci sami, opět podle návodu v kapitole 8.

Pokud tiskárna stále nefunguje, bude potřeba něco diagnostiky. V takovém případě bude nejlepší si nejprve přečíst větší část tohoto dokumentu, abyste zjistili, jak věci mají fungovat. Pak budete mít výrazně lepší pozici na hledání potíží.

Jak tisknout

Tisknout budete různými příkazy podle toho, jaký tiskový spooler máte nainstalován.

PDQ

Většina dnešních systémů používá *lpd*, takže pro ně tato část neplatí. Nicméně ve většině případů doporučuji PDQ nainstalovat a používat namísto (nebo vedle) *lpd*. PDQ má lepší podporu různých parametrů tiskáren a podobně.

Při použití PDQ používáte místo příkazu *lpr* příkaz *pdq* nebo *xpdq*. Oba fungují podobně jako klasický *lpd* tak, že vytisknou zadané soubory, nebo, pokud žádný soubor nezadáte, vytisknou *stdin*.

Xpdq

Xpdq je aplikace pro systém X Window, která zobrazí seznam dostupných tiskáren a přehled tiskových front (včetně aktuálně zpracovávaných a již zpracovaných úloh). V nabídce *File* nabízí dvě možnosti – jednu pro tisk zadaného souboru, druhou pro tisk *stdin*. V dialogu *Driver Options* můžete zvolit všechny parametry, které podporuje ovladač tiskárny – typicky zde volíte duplex, rozlišení, typ a velikost papíru a podobně.

Pdq

Systém pro řádkový tisk se jmenuje jednoduše *pdq*. Ve většině případů jej lze použít stejně jako *lpr*, zná i přepínač *-P*. Stejně jako *lpr* vytiskne zadaný soubor (soubory), nebo *stdin*. Parametry tiskárny je možné nastavovat přepínači *-o* a *-a*.

LPD a příkaz lpr

Pokud už máte nastaven systém *lpd*, nebo jej nastavil administrátor či dodavatel, jediné, co se musíte naučit, je používat příkaz *lpr*. O jeho použití a o některých užitečných technikách pro manipulaci s tiskovou frontou hovoří dokument *Printing Usage HOWTO*. Anebo si jednoduše přečtěte manuálovou stránku *lpr(1)*.

Stručně řečeno, parametrem *-P* definujete tiskovou frontu a pak uvedete název souboru, který se má vytisknout, popřípadě nic, pokud chcete tisknout z *stdin*. Tradičně *lpr* neumožňuje nastavovat parametry tiskárny, ale různé systémy podporují určité parametry pomocí přepínačů *-o*, *-Z* a *-J*.

Pokud ale máte nový systém nebo novou tiskárnu, budete si muset nejprve systém jedním nebo druhým způsobem nastavit sami.

Grafické tiskové nástroje

Většina tiskových systémů sama o sobě nabízí jen velmi jednoduché řádkové rozhraní. Namísto přímého použití příkazu *lpr* dáte možná přednost nějakému grafickému rozhraní. Ta obecně umožňují nastavit různé možnosti tisku (tiskárnu, typ papíru, řazení, počet kopií a podobně) snadno použitelnou grafickou metodou. Některá z nich nabízejí i další funkce.

GPR

GPR Thomase Hubella používá kód CUPSu[1] k filtraci postskriptových úloh a nabízí jednoduché řízení těchto úloh. Některé parametry (například tisk více kopií, výběr stránek a podobně) implementuje přímo GPR, většina ostatních je implementována tiskárnou nebo systémem spooleru.

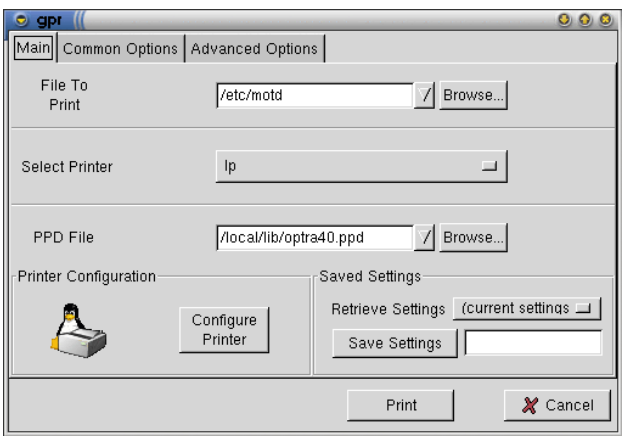
GPR funguje s LPD nebo s LPRng; případně se dá specificky přeložit pro práci s modifikovaným VL LPD pro Linux. Při normálním přeložení používá přímo knihovnu *libppd* a vytváří postscripto-

[1] Pozn. překladatele: CUPS = Common UNIX Printing System, viz kapitola 6.5.

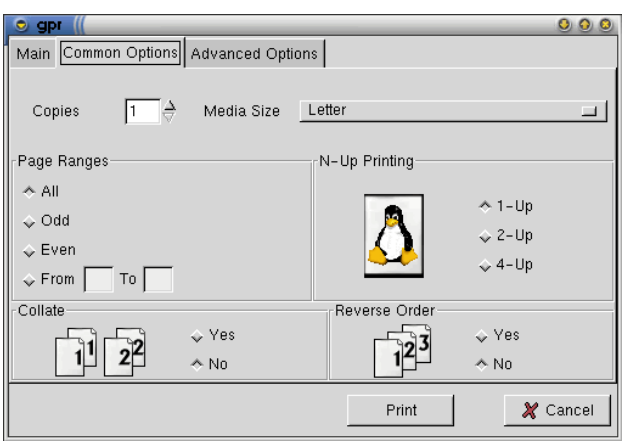
vý soubor přímo pro tiskárnu, který pak na ni příkazem *lpr* posílá. Je-li přeložen s podporou VA LPD, posílá příkazu *lpr* nemodifikovaný vstupní postscriptový soubor a parametry, které pro tisk úlohy zadáte. Je to bezesporu lepší řešení, protože umožňuje podle potřeby přesměrovat úlohu na jinou tiskárnu. Bohužel však vyžaduje speciální VA LPD, které není příliš používané (i když jeho instalace je samozřejmě triviální).

Při práci s GPR musíte nejprve vybrat tiskárnu (zadáním tiskové fronty) a ověřit, že má GPR nahrán správný soubor PPD. Pokud ne, musíte soubor PPD specifikovat a nastavit parametry tiskárny v dialogu *Printer Configuration* (který vyvoláte stiskem tlačítka *Printer Configuration*, a jenž obsahuje parametry tiskárny definované v PPD).

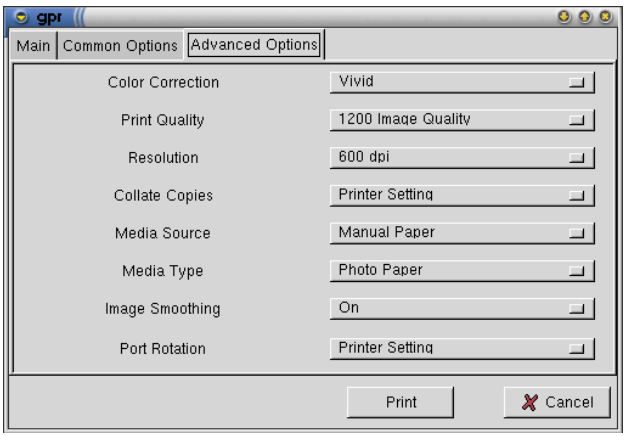
Po nastavení tiskárny můžete tisknout vybrané soubory a nastavovat parametry tisku na záložkách *Common* a *Advanced*. Volby *Common* implementuje přímo GPR shodně pro všechny tiskárny, volby *Advanced* jsou definovány v souboru PPD konkrétně pro používanou tiskárnu. Přehled jednotlivých možností můžete vidět na obrázcích 2 a 3.



Obr. 1 – Hlavní nabídka programu GPR



Obr. 2 – Volby Common

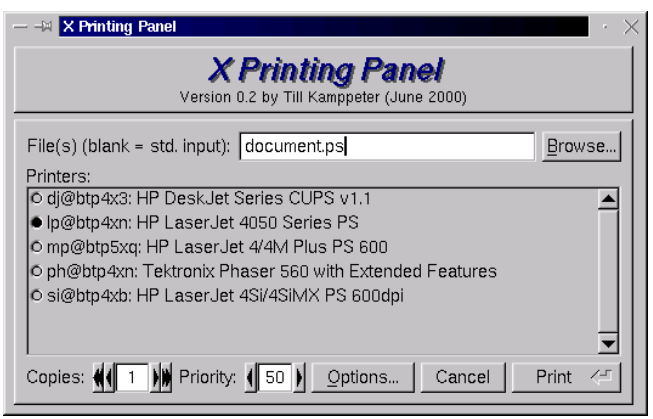


Obr. 3 – Volby Advanced

XPP

Pokud jako tiskový spooler používáte CUPS, můžete použít program XPP (viz obrázek 4). Chcete-li tímto programem tisknout, spusťte *xpp* a vyberte soubor (nebo jej nevybírejte, pokud chcete *xpp* použít namísto *lpr* pro tisk z *stdin*). Pak ze seznamu nakonfigurovaných tiskáren zvolte tiskárnu a případné parametry, které chcete pro tisk použít. Na obrázku 5 vidíte příklad panelu s parametry se zvýrazněním standardních parametrů CUPS.

Vybranou tiskárnu a všechna nastavení můžete uložit stiskem tlačítka *Save Settings*.

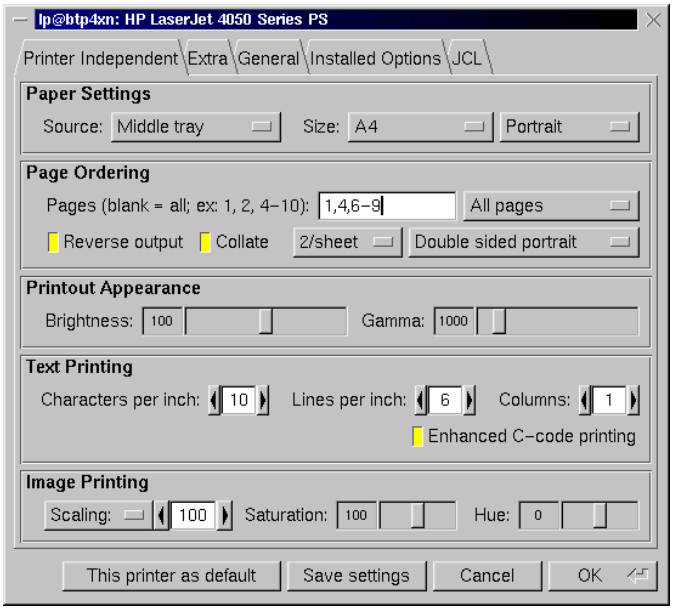


Obr. 4 – Hlavní okno programu XPP

XPDQ

PDQ se dá snadno nastavit k tisku do front ovládaných většinou spoolerů, konfigurační syntaxe PDQ nabízí jednoduchý způsob definice vlastních filtrů a uživatelských parametrů. Můžete tedy s úspěchem použít *xpdq* jako rozhraní k tiskovému systému LPD.

Další informace naleznete v kapitole 6.2.



Obr. 5 – Parametry CUPS/XPP

Tisková zařízení jádra

Pro obsluhu paralelního portu existují dvě úplně odlišná zařízení – které z nich používáte závisí na verzi vašeho jádra (a to můžete zjistit příkazem *uname -a*). Ovladače se změnily v jádře 2.1.33; prakticky všechny dnešní systémy používají jádra 2.2 nebo vyšší; takže klidně můžete přeskočit na část popisující zařízení *parport*.

Pouze poznámka platná pro oba typy ovladačů. Nejdůležitější asi je, že řada uživatelů zjistila, že Linux nedetekuje paralelní port, pokud v BIOSu nevypnou podporu „Plug-and-Play“. (Není to velké překvapení, PnP záznamy pro jiná než PCI zařízení, používané ve Windows a obdobných systémech, představují takovou malinkou katastrofu.)

Zařízení lp (jádra <=2.1.32)

Za předpokladu, že máte přeloženu nebo nahránu podporu zařízení *lp* (pokud je nahrána, bude výstup příkazu *cat /proc/devices* obsahovat zařízení *lp*), pak jádro (<=2.1.32) poskytuje jedno nebo více zařízení */dev/lp0*, */dev/lp1*, */dev/lp2*. Tato zařízení se *nepřiřazují* dynamicky, každé odpovídá konkrétní hardwarové vstupně-výstupní adrese. Znamená to, že první a jediná tiskárna může být klidně *lp0* nebo *lp1* v závislosti na hardwaru. Prostě zkuste obojí.

Někteří uživatelé si stěžují, že obousměrný port není detekován, pokud použijí starší jednosměrný kabel[2]. V takovém případě zkontrolujte, zda používáte správný kabel.

Na jednom portu nemůžete současně spustit ovladače *plip* a *lp* (s jádrem 2.0). Můžete nicméně kdykoliv jeden nebo druhý ovladač nahrát ručně, případně démonem *kerneld* u jader 2.x (a novějších 1.3.x).

[2] Pozn. překladatele: Neuvěřitelné!!!

Při správném nastavení přerušení a dalších parametrů ovšem můžete na jednom portu používat *plip* a na druhém *lp*. Úspěšně se to podařilo úpravami ovladačů; zajímalo by mě, zda se to někomu povedlo pouze vhodnými parametry příkazů. Existuje nástroj *tunelp*, který s jádrem 2.0 umožňuje nastavovat přerušení a další parametry paralelních portů.

Pokud je ovladač *lp* součástí jádra, přijímá jádro parametr *lp=* pro nastavení přerušení a vstupně -výstupní adresy. Můžete použít příkazový řádek programů LILO/LOADLIN a nastavit adresy a přerušení, které bude ovladač používat.

Syntaxe: *lp=port0[,irq0[,port1[,irq1[,port2[,irq2]]]]]*

Například: *lp=0x378,0* nebo *lp=0x278,5,0x378,7*[3]

Pokud chcete tuto funkci použít, musíte definovat *všechny* používané porty, neexistují žádné implicitní hodnoty. Vestavěný ovladač můžete vypnout parametrem *lp=0*.

Pokud ovladač nahráváte jako modul, můžete vstupně-výstupní adresu a přerušení definovat na příkazovém řádku *insmod* (nebo v souboru */etc/conf.modules*, pokud používáte *kerneld*) s obvyklou syntaxí parametrů. Parametry jsou *io=port0,port1, port2* a *irq=irq0,irq1,irq2*. Další informace viz manuálová stránka příkazu *insmod*.

Zdrojový kód ovladače paralelního portu v jádře 2.0 najdete v souboru */usr/src/linux/drivers/char/lp.c*.

Zařízení parport (jádra >= 2.1.33)

Počínaje jádrem 2.1.33 (s patchem až pro 2.0.30) je zařízení *lp* pouhým klientem nového zařízení *parport*. Přidáním zařízení *parport* se odstranila řada problémů s původním ovladačem zařízení *lp* – bylo možné sdílet port s jinými ovladači, bylo zavedeno dynamické přiřazování dostupných portů číslům zařízení namísto pevného vztahu mezi vstupně-výstupní adresou a zařízením, a další.

Díky novému ovladači *parport* se mohla objevit celá řada nových paralelních ovladačů pro zařízení jako jsou mechaniky Zip, CD-ROM a disky Backpack a další. Některé z nich existují i v jádrech 2.0; zkuste je najít na webu.

Hlavním rozdílem, kterého si můžete všimnout ve vztahu k tisku, je dynamické přiřazení zařízení *lp* paralelním portům. Tedy to, co v Linuxu 2.0 bylo *lp1,* může být v Linuxu 2.2 klidně *lp0*. Pokud přecházíte z jádra s ovladačem *lp* na jádro s ovladačem *parport*, nezapomeňte to zkontrolovat.

Většina oblíbených problémů s tímto zařízením vzniká právě díky chybné konfiguraci.

> Některé distribuce Linuxu nemají správně nastaven soubor */etc/modules.conf* (nebo */etc/conf.modules*), takže se ovladač v okamžiku potřeby nenahraje správně. S posledními verzemi nástroje *modutils* vypadá správný řádek v souboru *modules.conf* takto:
>
> ```
> alias /dev/printers lp             # jenom pro devfs?
> alias /dev/lp*      lp             # jenom pro devfs?
> alias parport_lowlevel parport_pc  # chybí v Red Hat 6.0-6.1
> ```
>
> **BIOS**
>
> Řada BIOSů obsluhuje paralelní port jako zařízení Plug-and-Play. Tím se ovšem perfektně jednoduché prakticky vždy přítomné zařízení rozšiřuje o naprosto zbytečné složitosti. Pokud Linux nedetekuje paralelní port správně, vypněte u něj podporu PnP (ve většině BIOSů nastavujete *LPT1*). Správné nastavení bude obvykle *legacy*, *ISA* nebo *0x378* – určitě ale ne *disabled*.

[3] Pokud si stejně jako já nemůžete vzpomenout na standardní čísla portů, vidíte je právě v tomto druhém příkladu. Zbývající paralelní port používá adresu *0x3bc*, ovšem netuším, jaké přerušení.

Doporučujeme také dokumentaci ve zdrojových kódech jádra (http://people.redhat.com-/twaugh/parport/html/parportguide.html) nebo stránky http://people.redhat.com/twaugh/parport/.

Sériová zařízení

Sériová zařízení se v Linuxu obvykle jmenují nějak jako */dev/ttyS1*. Nástrojem *stty* můžete interaktivně zobrazovat nebo měnit nastavení sériových portů, nástrojem *setserial* pak můžete nastavovat některé další parametry, přerušení a vstupně-výstupní adresy nestandardních portů. Podrobnější debatu na téma sériových portů v Linuxu najdete v dokumentu *Serial-HOWTO*.

Pokud používáte pomalou sériovou tiskárnu s řízením toku, může dojít k tomu, že se ztrácejí části tiskových úloh. Může to způsobovat sériový port, který se standardně chová tak, že všechny nepřenesené znaky ze svého bufferu odstraní 30 sekund po zavření zařízení. Buffer má délku 4 096 znaků a pokud tiskárna používá softwarové řízení toku a je dost pomalá na to, aby nestihla zpracovat celý buffer do 30 sekund od chvíle, kdy tiskový program zavře sériový port, konec bufferu se ztratí. Pokud příkazem *cat soubor > /dev/ttyS2* správně vytisknete krátké soubory, ale u dlouhých se konec souboru ztratí, může jít o tento případ.

Interval 30 sekund je možné změnit parametrem *closing_wait* příkazu *setserial* (verze 2.12 a vyšší). Sériové porty se typicky inicializují voláním *setserial* v souboru *rc.serial*. Můžete toto volání upravit taky, aby kromě ostatních parametrů rovnou nastavilo i delší interval *closing_wait*.

USB zařízení

Nemám k dispozici žádné USB zařízení k otestování, takže mohu nabídnout pouze odkazy. Po správném nastavení byste měli získat zařízení */dev/usb/lp0* analogické tomu, co dostáváte pro paralelní port, které by mělo správně fungovat v souboru *princap* nebo jako lokální zařízení PDQ. Práce s USB je popsána na webovém serveru *Linux USB* (http://www.linux-usb.org/).

Podporované tiskárny

Jádro Linuxu vám umožní bavit se s jakoukoliv tiskárnou, kterou připojíte k sériovému, paralelnímu nebo USB portu, plus s jakoukoliv síťovou tiskárnou. To ale samo o sobě nestačí – musíte být také schopni generovat data, kterým bude tiskárna rozumět. Typicky mezi nepoužitelné tiskárny patří tiskárny označované jako „Windows“ nebo „GDI“. Jazyk pro ovládání těchto tiskáren je úplně nebo částečně nedokumentován, stejně jako podrobnosti o návrhu tiskového mechanismu. Typicky výrobce dodává s tiskárnou ovladač pro Windows a vesele je prodává pouze uživatelům Windows; proto se tyto tiskárny označují také jako *Winprinters*. V některých případech dodává výrobce ovladače i pro NT, OS/2 a jiné operační systémy.

Řada těchto tiskáren v Linuxu nefunguje. Některé z nich fungují, některé fungují pouze částečně (obvykle díky tomu, že někdo pomocí reverzního překladu ovladače zjistil, jak s tiskárnou zacházet). Viz seznam použitelných tiskáren dále v této kapitole.

Některé tiskárny jsou něco mezi. Například některé modely tiskáren NEC implementují zjednodušenou verzi ovládacího jazyka PCL, díky čemuž mohou v rozlišení 300 DPI na tiskárně tisknout programy ovládající PCL. Ovšem pouze NEC ví, jak z tiskárny dostat plné možné rozlišení 600 DPI.

Pokud už některou z těchto tiskáren máte, existují různé prapodivné cestičky, jak je využít i v Linuxu, jde ale o poměrně nehezké postupy. Další podrobnosti o využití těchto tiskáren najdete v kapitole 12.

PostScript

Co se týče tiskáren v Linuxu, nejlepší volba je koupit tiskárnu, jejíž firmware nativně podporuje PostScript. Prakticky všechny unixové programy produkující tiskový výstup jej vytvářejí jako PostScript, takže je zjevně příjemné mít tiskárnu, která jej přímo podporuje. Bohužel, podpora PostScriptu bývá kromě laserových tiskáren jen zřídkavá, a někdy se prodává pouze jako drahý doplněk.

Unixové programy a obecně publikační průmysl se ustálily na používání PostScriptu coby standardního jazyka pro práci s tiskárnami. Má to několik důvodů:

- *Načasování* – PostScript se objevil jako součást Apple Laserwriter, perfektní tiskárny k Macintoshům, které jsou zodpovědné za revoluci v publikování v 80. letech.
- *Nezávislost na zařízení* – Pomocí PostScriptu je možné generovat výstup na rastrové obrazovce, vektorové obrazovce, faxu nebo prakticky jakémkoliv tiskovém zařízení bez nutnosti úprav původního programu. Výstup PostScriptu bude na jakémkoliv zařízení vypadat stejně, samozřejmě v rámci možností daného zařízení. Před vznikem PDF se k výměně složitějších dokumentů používal právě PostScript. Jediným důvodem, proč tento standard nezůstal jediným, je to, že Windows typicky neobsahují prohlížeč PostScriptu, takže Adobe doplnil PostScript o hypertextové odkazy a kompresi, výsledek pojmenoval PDF, začal distribuovat prohlížeče tohoto formátu a vytvořil trh pro své „destilační“ nástroje (jejichž funkce plní i programy ps2pdf a pdf2ps balíku ghostscript).
- *Jde o kompletní programovací jazyk* – můžete pomocí něj vytvářet programy, které budou dělat prakticky cokoliv. Většinou je to užitečné k definici procedur, které pak v rámci celého dokumentu stále dokola opakují stejné akce, například vykreslení loga nebo pozadí stránek. Neexistuje ale žádný důvod, proč byste pomocí PostScriptu nemohli třeba počítat π.
- *Jde o otevřený formát* – PostScript je kompletně dokumentován ve volně dostupných publikacích (které najdete v každém slušnějším obchodě ve světě). I když tvůrcem formátu a dominantní komerční implementace je Adobe, nezávislé implementace nabízejí i jiní výrobci, například Aladdin.

Tiskárny bez PostScriptu

Pokud nemáte finance na nákup (obvykle drahé) PostScriptové tiskárny, můžete použít jakoukoliv tiskárnu, kterou podporuje Ghostscript, volně šířený interpret PostScriptu, používaný namísto přímé podpory PostScriptu tiskárnou. Většina linuxových distribucí obsahuje z licenčních důvodů pouze poněkud zastaralou verzi Ghostscriptu. Naštěstí obvykle existuje možnost získat i aktuální verzi Ghostscriptu.

Společnost Adobe vytvořila nový jazyk pro práci s tiskárnami, pojmenovaný *PrintGear*. Pokud vím, jde o výrazně zjednodušený binární jazyk s jistými rysy PostScriptu, nicméně nekompatibilní s PostScriptem. Neslyšel jsem, že by jej Ghostscript podporoval. Některé tiskárny s jazykem PrintGear však podporují i jiné jazyky, jako například PCL, a tyto tiskárny pak v Linuxu bez potíží fungují (pokud je ovšem PCL implementován přímo v tiskárně, a ne v ovladači pro Windows).

Adobe dále nabízí implementaci PostSciptu, pojmenovanou *PressReady*. Chová se podobně jako Ghostscript a nabízí podporu PostScriptu na tiskárnách, které ji nativně neobsahují, nevýhodou ovšem je, že tento program funguje pouze ve Windows.

Které tiskárny fungují?

Chystáte-li se kupovat tiskárnu, máte několik možností, kde zjistit, zda bude fungovat. Kolektivně udržovaná databáze tiskáren se snaží o vyčerpávající seznam stavu podpory různých tiskáren v Linuxu. Podívejte se na její on line verzi na adrese http://www.linuxprinting.org/database.html, kde najdete i informace o tom, jaký ovladač s kterou tiskárnou použít.

Další seznam použitelných tiskáren naleznete na adrese http://www.cs.wisc.edu/~ghost/printer.html, kde se nachází *Ghostscript printer compatibility list*.

Stovky jednoduchých zhodnocení funguje-nefunguje najdete na adrese http://www.deja.com/usenet/.

Chcete-li mít jistotu, zkontrolujte všechny tři. Pokud jste líní, krátký seznam doporučených tiskáren najdete na mé webové stránce, http://www.linuxprinting.org/suggested.html. Tato stránka se soustřeďuje na barevné inkoustové tiskárny a levné laserové tiskárny, najít kompatibilní zařízení střední a vyšší třídy je poměrně jednoduché.

Jak kupovat tiskárnu

Volba tiskárny není dnes nic jednoduchého, existuje celá řada modelů, z nichž je možno vybírat. Uveďme si několik tipů, podle nichž se při koupi můžete orientovat:

Cena

Dostanete to, za co zaplatíte. Většina tiskáren v ceně mezi 200 – 300 dolary tiskne v rozumné kvalitě, nicméně má vysoké náklady na stránku. U některých tiskáren stojí jedna či dvě nové náplně stejně, jako celá tiskárna! Navíc nejlevnější tiskárny nevydrží příliš dlouho. Typicky mají nejlevnější tiskárny střední dobu mezi poruchami zhruba tři měsíce – což je docela málo, pokud má být tiskárna hodně používána.

Inkoustové tiskárny

Tiskové hlavy inkoustových tiskáren se po jisté době nenávratně ucpou, čili nutná je možnost časem hlavu vyměnit. Tiskové hlavy těchto tiskáren jsou drahé, přičemž kombinovaná hlava s náplní stojí zhruba 10krát (!!!) tolik, co samotná náplň. Proto je rozumné volit takové tiskárny, kde hlava není integrovaná s náplní. Tento typ tiskáren vyrábí Epson, HP u řady DeskJet používá kombinované hlavy. Canon používá tří nezávisle na sobě vyměnitelné náplně – což je asi nejlepší řešení. Na druhé straně, náplně do tiskáren HP nejsou až tak hrozně drahé a kvalita tisku těchto tiskáren je asi nejlepší. Tiskárny Canon stojí ohledně kvality až na třetím místě – a tiskárny Epson jsou zatím v Linuxu nejlépe podporované. Prostě nemůžete vyhrát.

Laserové tiskárny

U laserových tiskáren je nutné měnit tiskový válec a toner, plus zásobník nespotřebovaného toneru. Nejlevnější řešení kombinuje válec s tonerem v jedné velké náplni, tyto tiskárny mají největší provozní náklady. Nejlepší velkoobjemové tiskárny používají samostatný toner, nebo alespoň samostatnou tonerovou kazetu a samostatný zásobník na odpad.

Fotografické tiskárny

Nejlepší barevný výstup ve fotografické kvalitě poskytují tiskárny s plynulým tónováním, které kombinují halogenidy stříbra a laserové osvícení a vyrábějí – překvapivě – opravdové fotografie! Tyto tiskárny ovšem stojí desítky tisíc dolarů, takže například *Ofoto.com* nabízí jednorázové tisky fotografií za rozumné ceny[4]. Výsledek je vynikající, ani nejlepší inkoustové tiskárny nemají šanci na srovnání.

[4] Pozn. překladatele: V ČR tuto službu nabízí např. http://www.fotolab.cz.

Nejvyšší za rozumnou cenu dosažitelnou kvalitu nabízejí sublimační tiskárny, například série Alp (používají tepelný přenos pevného barviva nebo přímo sublimaci barviva). Podpora těchto tiskáren v Linuxu je, bohužel, slabá (v jedné referenci, kterou jsem získal, se hovoří o pruzích a zrnitém obrazu), a ani pak není jisté, zda bude možné použít nejvyšší kvalitu sublimačního přenosu.

Běžnější inkoustové tiskárny pro tisk fotografií používají šesti nebo sedmibarevný tisk (CMYKcm nebo CMYKcmy). Provoz těchto speciálních tiskáren je drahý. Buď vám dojde modrá a musíte vyměnit celou barevnou kazetu, nebo je možné doplňovat barviva samostatně, ale náplň pak stojí majlant. Drahý je také speciální papír – nemůžete očekávat vysokou kvalitu při tisku na běžný kancelářský papír. Viz dále část věnovaná tisku fotografií a popis nastavení barev v Ghostscriptu.

Rychlost

Rychlost je úměrná možnostem procesoru tiskárny, kapacitě připojení a ceně tiskárny. Nejrychlejší tiskárny jsou síťové PostScriptové tiskárny s výkonným interním procesorem. Rychlost tisku běžných tiskáren bude zčásti ovlivněna rychlostí vykreslování Ghostcriptu, proto je vhodné použít rozumně výkonný počítač – zejména stránky se spoustou barev mají velké nároky na paměť tiskového počítače. Pokud máte dostatek paměti, mělo by všechno fungovat bez potíží.

Formuláře

Pokud chcete tisknout formuláře s více kopiemi, potřebujete úhozovou tiskárnu. Řada společností stále vyrábí jehličkové tiskárny, které většinou emulují tradiční modely firmy Epson a fungují tak bez potíží.

Samolepky

Existují dvě podporované řady speciálních tiskáren na samolepky – Dymo-Costar a Seiko SLP. Ostatní tiskárny mohou a nemusejí pracovat. Avery vyrábí samolepicí štítky různých velikostí ve formátu A4, takže můžete štítky tisknout i na normálních tiskárnách.

Plottery

K tisku na velké formáty se dnes obvykle používají obří inkoustové tiskárny, oblíbené jsou modely firmy HP. Existují i menší inkoustové tiskárny vhodné pro tisk na formát A3. Většina těchto tiskáren používá jazyky RTL, HP-GL nebo HP-GL/2, což jsou všechno proprietární vektorové jazyky společnosti HP, výstup je typicky generován přímo aplikačními programy.

Co mám já?

Mám tiskárny HP Deskjet 500, Lexmark Optra 40 a Canon BJC-4100. Všechny fungují perfektně, HP a Canon jsou starší modely dobře podporované Ghostscriptem, Optra je modernější barevná inkoustová tiskárna s plnou podporou PostScriptu a PCL 5 (!).

Kromě toho používám tiskový server Hawking Technology 7117 (vyráběný ve skutečnosti společností Zero One Technologies na Tchaj-wanu), díky tomu může být tiskárna kdekoliv, kde je napájení a Ethernet, nemusí být poblíž počítače. Je to malá krabička připojená přímo na paralelní port tiskárny, na druhém konci má ethernetovou zásuvku. Jedinou nevýhodou je, že neumí obousměrnou komunikaci, takže si nemůžu nechat e-mailem posílat upozornění, že dochází toner.

Spoolovací programy

Až donedávna měli uživatelé Linuxu jednoduchou volbu – všichni používali stejný starý *lpd*, převzatý prakticky doslova z kódu Net-2 BSD. Dokonce i dnes většina dodavatelů tento program distribuuje. Nicméně situace se mění. Systémy SVR4 včetně Sun Solaris používají úplně jiné řešení spooleru, postavené kolem programu *lpsched*.

V současné době lze vybírat z více dobrých systémů. Postupně je všechny popíšu, přečtěte si popisy a rozhodněte se sami. Nejjednodušší moderní systém s grafickým rozhraním je PDQ, hodí se jak pro normální domácí uživatele, tak i (v hybridním uspořádání *pdq/lprng*) i pro větší prostředí. V komerčním nasazení, kde se vesměs používají síťové postscriptové tiskárny, je dobrou alternativou rozhraní jako GPR s LPRng; přímo pracuje s možnostmi PPD a má o něco pěknější rozhraní. V jiných případech je dobrou volbou CUPS, má vynikající podporu postscriptových tiskáren, a nabízí dále podporu IPP, webové rozhraní a další funkce.

LPD

LPD, původní Line Printer Daemon z BSD Unixu, je v unixovém světě standardem již řadu let. Je k dispozici na všech unixových systémech a nabízí minimální množinu funkcí pokrývajících potřeby z doby sdílení počítačů. Navzdory specifické historii je i dnes užitečný jako základní tiskový spooler. K rozumnému použití s moderními tiskárnami vyžaduje specifická nastavení, jako jsou různé filtrační skripty a uživatelská rozhraní. To ale všechno existuje a funguje.

LPD je rovněž název síťového tiskového protokolu, definovaného dokumentem RFC 1179 (http://www.ietf.org/rfc/rfc1179.txt). Tento protokol umí nejen démon LPD, ale v zásadě každý tiskový server, síťové tiskárny i jiné spoolovací systémy. LPD je nejmenší společná množina funkcí standardního síťového tiskového prostředí.

LPRng (viz kapitola 6.3) je výrazně lepší implementace standardní LPD návrhu. Pokud musíte používat LPD, zvolte raději LPRng. Vyžaduje o hodně méně magie k tomu, abyste jej donutili dělat přesně to, co potřebujete, a všechny čáry jsou dobře dokumentovány.

Po světě se toulá řada různých zdrojových podob LPD. Pravděpodobným oficiálním vlastníkem je nějaký derivát BSD Unixu, nicméně každý vesele implementuje různé změny, které se neznámými způsoby prolínají a tak lze jen velmi obtížně určit, jakou konkrétní verzi LPD kdo používá. Z těch veřejně dostupných LPD používá VA Linux jeden s drobnými úpravami, které vylepšují uživatelské rozhraní. LPD na SourceForge (http://sourceforge.net/project/?group_id=3800) podporuje zadávání parametrů přepínačem *-o*, ty se pak předávají filtrům. Je to podobné funkci, nabízené i řadou tradičních unixových dodavatelů a analogické (i když nekompatibilní) s mechanismem přepínače *-z* programu LPRng.

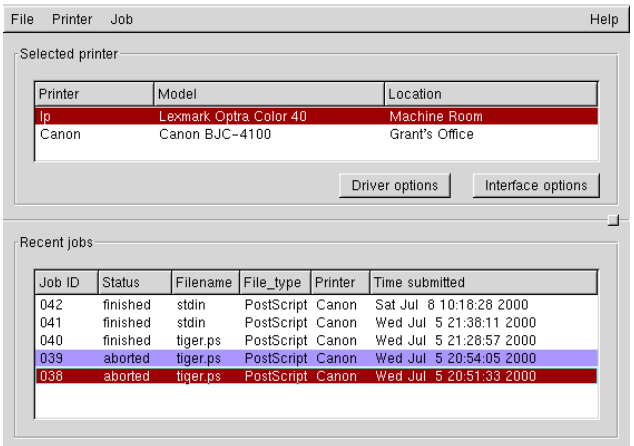
Rozhraní k programu LPD

Pokud musíte používat LPD, je rozumné použít nějaké samostatné rozhraní. Možností je několik, nejlepší jsou asi GPR (kapitola 3.3) a XPDQ (kapitola 6.2). Kromě toho existují i další.

PDQ

PDQ (http://feynman.tam.uiuc.edu/pdq/) je tiskový systém s vestavěnou výkonnou syntaxí pro konfiguraci ovladačů, který nepoužívá tiskového démona. Zahrnuje možnost nastavovat parametry tiskárny a používat grafické nebo řádkové nástroje, jimiž mohou uživatelé tyto parametry nastavit. Uživateli se zobrazí hezký dialog, ve kterém může nastavit rozlišení, duplexní režim, typ papíru a další (viz obrázek 7).

Spouštění všech filtrů v uživatelském režimu má řadu výhod – prakticky se eliminují bezpečnostní problémy vznikající v PostScriptu, efektivně lze tisknout vícesouborové úlohy LaTeXu jako soubory *dvi* a další.

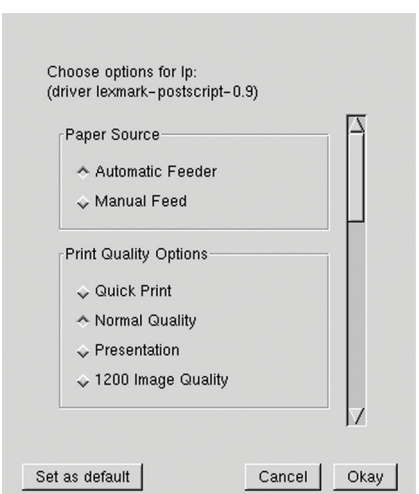


Obrázek 6 – Hlavní okno programu XPDQ

Právě toto řešení nyní používám, napsal jsem si specifikační soubory ovladače pro své tiskárny – distribuce obsahuje řadu už hotových, takže máte k dispozici spoustu příkladů, z nichž můžete vyjít. Kromě toho jsem napsal i několik nástrojů, které automatizují generování specifikačních souborů.

PDQ má i své chyby. Zásadní problém je, že před odesláním úlohy na tiskárnu ji nejprve celou zpracuje. Znamená to, že u velkých úloh může být PDQ nepraktický – skončíte stovkami megabajtů dat, které se budou kopírovat tam a zpátky na disku. Ještě horší je, že u pomalých ovladačů, jako jsou například ovladače kvalitních inkoustových tiskáren, nezačne tisk do doby, než Ghostscript a ovladač skončí zpracování úlohy. To může být klidně i několik minut od odeslání úlohy.

Pokud máte hodně uživatelů, hodně tiskáren, nebo cokoliv jiného složitého, doporučuji použít PDQ pouze jako rozhraní k síťovému tisku založenému na protokolu LPD (protokolem *lpd* můžete tisknout i na lokálním počítači). Ve většině případů pak doporučuji namísto tradičního LPD použít raději LPRng.



Obrázek 7 – Okno s parametry ovladače programu XPDQ

LPRng

Někteří dodavatelé Linuxu (například Caldera), nabízejí LPRng, modernější tiskový mechanismus, než starička LPD. LPRng se daleko snáze spravuje ve větších instalacích (rozuměj – s více než jednou tiskárnou, při použití sériových tiskáren, nebo jakýchkoliv podivných síťových tiskáren, nepoužívajících *lpd*). Kromě toho je celý kód tohoto systému průhlednější, než historický a komplikovaný LPD. Hrdě o sobě může tvrdit, že je bezpečný – nepoužívá žádné SUID programu a podporuje autentikaci přes PGP nebo Kerberos.

LPRng obsahuje i příklady nastavení běžných síťových tiskáren, zejména tiskáren HP LaserJet, které zahrnují i funkce účtování. Pokud vás LPRng zaujalo, podívejte se na webové stránky LPRng, http://www.astart.com/lprng/LPRng.html. LPRng navíc používá víceméně stejný filtrační mechanismus jako LPD, takže podpora LPD na mých stránkách http://www.linuxprinting.org/lpd-doc.html platí i pro LPRng. Díky tomu můžete používat ovladače pro většinu tiskáren.

LPRng se distribuuje pod licencemi GPL nebo Artistic.

PPR

PPR (http://ppr.trincoll.edu/) je spooler se zaměřením na PostScript, obsahující základní schopnost parsovat PostScript, z níž je odvozeno několik pěkných funkcí. Zahrnují možnosti účtování, dobrou podporu klientů Appletalk, SMB a LPD a mnohem lepší obsluhu chyb, než *lpd*. PPR, stejně jako ostatní spoolery, dokáže prostřednictvím Ghostscriptu tisknout i na tiskárny, které neumějí Postscript.

Na PPR jsem narazil poměrně nedávno, nevím o nikom, kdo jej používá. Program vznikl a používá se na Trinity College. Distribuuje se pod licencí typu BSD.

Podle dokumentace jde o částečně experimentální řešení. Poškozené soubory nebudou vytištěny, ale odmítnuty, a je problémem uživatele, aby dokument opravil. V některých prostředích to může být nevyhovující, ovšem většina uživatelů Postscript generuje prostřednictvím několika málo známých generátorů, takže by to neměl být až takový problém.

CUPS

CUPS (http://www.cups.org/) je zajímavou novinkou na scéně. Jde o implementaci protokolu Internet Printing Protocol, což je RFC standard podobného typu jako HTTP, který nahrazuje staříčký (a hloupoučký) protokol LDP. Implementaci CUPS vytvořil Michael Sweet ze společnosti Easy Software Products; program je distribuován pod licencí GPL.

Postupně jsem CUPS začal používat a funguje opravdu tak, jak autor tvrdí. Nabízí řadu pěkných funkcí včetně manipulace s parametry tisku; nabízí webové, grafické a řádkové rozhraní a dále nabízí filtrovací systém založený na MIME se silnou podporou PostScriptu. Nicméně jde o poměrně nový produkt a tak má své mouchy a jen stěží jej můžeme doporučit pro velká prostředí nebo prostředí s nároky na bezpečnost. Je to nicméně příjemné řešení, zejména u menších instalací a ve skupinách vzájemně si důvěřujících uživatelů.

Stejně jako ostatní systémy lze i CUPS použít s většinou stávajících ovladačů. Bohužel je dost obtížné nakonfigurovat libovolný ovladač pro práci s CUPSem (zejména pokud chcete využít všech možností), a proto je nejlepší nalézt existující soubory PPD a filtrační skripty. Existují přinejmenším čtyři skupiny ovladačů, které můžete s CUPSem použít:

Foomatic (http://www.linuxprinting.org/foomatic.html)

Můj webový systém CUPS-O-Matic dokáže vygenerovat PPD pro jakýkoliv ovladač tiskárny, který je plně popsán v databázi LinuxPrinting.org. Tyto PPD soubory se pak používají společně se skriptem *cupsomatic*. CUPS-O-Matic pracuje pouze s volně šířenými ovladači. V současné době je podporována řada tiskáren a jak XPP tak QtCUPS (grafická rozhraní ke CUPSu) obsahují podporu různých úprav používaných s ovladači systému Foomatic, které CUPS přímo nepodporuje. Foomatic představuje základ pro podporu tiskáren bez PostScriptu v řadě distribucí Linuxu.

Ovladače CUPS a KUPS (http://cups.sourceforge.net/)

Projekt CUPS Drivers shromažďuje PPD soubory použitelné s postscriptovými tiskárnami nebo s filtrem *ps2gs2raw*. Tyto soubory pracují s volně šířenými ovladači. KUPS je přiložený konfigurační program. Celý projekt se víceméně zastavil po nástupu Foomaticu, nicméně KUPS představuje pěkný konfigurační nástroj, dodávaný s řadou distribucí.

PPD k postscriptovým tiskárnám

CUPS může přímo používat výrobcem dodávané PPD soubory pro postscriptové tiskárny. Ty jsou často dodávány společně s ovladačem pro Windows, případně jsou k nalezení na stránkách výrobce. Adobe (http://www.adobe.com/products/printerdrivers/winppd.html) distribuuje PPD soubory řady postscriptových tiskáren.

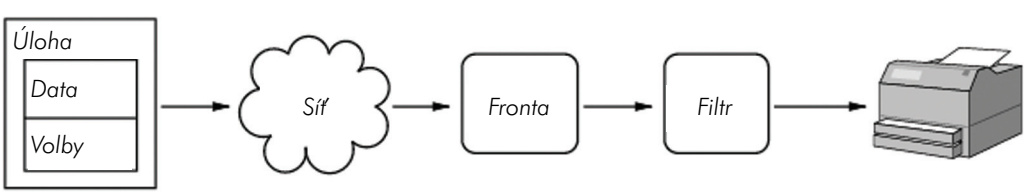
ESP Print Pro

Společnost Easy Software Products (http://www.easysw.com/) prodává CUPS společně s řadou proprietárních ovladačů. I když nejde o volně šířený software, umí pracovat s řadou běžných tiskáren. Celý balík je poměrně drahý v porovnání s cenou za podporu jedné tiskárny, nicméně nemusí být k zahození. Ovladače nejsou vždy špičkové, nicméně podporují větší množinu tiskáren, než ovladače volně šířené, a v některých případech je i méně kvalitní tisk lepší, než žádný tisk. Balík dále obsahuje grafické rozhraní, které ale rozhodně není tak dobré, jako XPP nebo QtCUPS.

Velmi pěkné grafické rozhraní pro CUPS nabízí program XPP (http://cups.sourceforge.net/xpp/) (viz obrázek 5), jehož součástí je vynikající rozhraní pro nastavení voleb tisku (viz obrázek 6). Informace o programu XPP najdete v kapitole 3.3.3.

Jak to celé funguje

Abyste si dokázali poradit s různými záludnostmi tisku, je nutné chápat, jak celý tiskový systém funguje. Všechny systémy fungují v zásadě stejně, i když se může poněkud lišit pořadí některých kroků, případně mohou být některé vynechány.



Obrázek 8 – Činnost spooleru

1. Uživatel odesílá tiskovou úlohu společně s nastavenými parametry. Data úlohy jsou obvykle, byť ne vždycky, PostScript.
2. Spooler zkopíruje úlohu a parametry přes síť směrem k tiskárně.
3. Spooler čeká na dostupnost tiskárny.
4. Spooler uplatní na úlohu uživatelem zvolené parametry a provede překlad dat úlohy do nativního jazyka tiskárny, což typicky není PostScript. Tomuto kroku se říká filtrování a většina problémů s tiskem spočívá právě v dobře nastaveném filtrování.
5. Úloha je hotova. V této fázi spooler obvykle provede potřebný úklid. Pokud došlo při zpracování úlohy k nějaké chybě, spooler typicky uživatele nějakým způsobem (například e-mailem) upozorní.

PDQ

PDQ znamená „Print, Don't Queue" a tento název charakterizuje i jeho návrh. Když tisknete pomocí PDQ, odehrává se následující sekvence kroků:

- Spustíte pdq nebo xpdq a zvolíte tisknutý soubor.
- Zvolíte tiskárnu.
- Nastavíte různé parametry a volby nabízené definičním souborem tiskárny (duplexní režim, počet kopií, kvalita tisku a podobně).
- PDQ provede analýzu tisknutých dat a podle instrukcí v ovladači se dozví, jak vaše data a zvolené parametry předat tiskárně.
- PDQ pošle zpracovaná data tiskárně na port definovaný pro tuto tiskárnu (například přímo na /dev/lp0, nebo LPD démonu v síti nebo dokonce přes síť systémům Apple či Microsoft, nebo třeba na fax).
- Pokud PDQ nemůže okamžitě odeslat data na tiskárnu, spustí proces v pozadí, který se o to bude opakovaně pokoušet dokud neuspěje, nebo dokud nevyprší časový limit.

Po celou dobu tohoto procesu i po něm je možné stav každé tiskové úlohy zjistit pomocí xpdq. Nevytisknuté úlohy jsou zobrazeny červeně a je možné jejich tisk opakovat.

LPD

LPD znamená Line Printer Daemon a v různých kontextech se touto zkratkou označuje jak samotný démon, tak celá skupina programů, které zajišťují tiskový spooling. Jsou to:

lpd Spoolingový démon. Jedna instance běží a řídí celý tisk na počítači, další instance běží pro každou tiskárnu v době, kdy tiskne.

lpr Uživatelský příkaz pro tisk. Kontaktuje *lpd* a přidá úlohu do fronty.

lpq Vypíše seznam úloh ve frontě.

lpc Příkaz pro řízení tiskového systému. Umožňuje zastavovat, spouštět, měnit pořadí a podobně úloh ve frontách.

lprm Odstraňuje úlohu z fronty.

Jak to zapadá do sebe? Odehrávají se následující kroky:

1. Při startu systému se spouští *lpd*. Čeká na spojení a spravuje tiskové fronty.
2. Uživatel zadává úlohy k tisku příkazem *lpr,* popřípadě grafickým rozhraním jako GPR, PDQ a podobně; *lpr* kontaktuje po síti *lpd* a předá mu datový soubor (obsahující tištěná data) a řídicí soubor (obsahující parametry úlohy).
3. Když se uvolní tiskárna, spustí hlavní *lpd* svou další instanci, která obslouží tisk úlohy.
4. Synovský *lpd* spustí příslušný filtr (definovaný příznakem *if* v souboru */etc/printcap*) a výsledná data pošle na tiskárnu.

Tento systém byl původně navržen v době, kdy většina tiskáren byly řádkové tiskárny – tedy v době, kdy se převážně tiskly čistě textové soubory. Umístěním všech potřebných kroků do příslušných filtrů je možné pomocí *lpd* vyhovět i nárokům moderního tisku (tedy, alespoň jakžtakž – řada jiných systémů funguje lépe).

Při tvorbě filtrů pro LPD je možné použít řadu užitečných programů. Jsou to mimo jiné:

gs — Ghostscript je softwarový interpret PostScriptu (označovaný též jako Raster Image Processor nebo RIP). Jako vstup přijímá PostScript a výstup vytváří v různých ovládacích jazycích tiskáren nebo v různých grafických formátech. O tomto programu hovoříme v kapitole 10.

ppdfilt — Samostatná verze jedné z komponent CUPSu. Filtruje PostScript a provádí některé základní transformace (jako například řazení více kopií) a doplňuje uživatelem nastavené parametry na základě PPD souboru (PostScript Printer Definition), který bývá obvykle dodáván s tiskárnou.

ps2ps — Jde o skript, dodávaný s Ghostscriptem. Filtruje PostScript na jednodušší PostScript na nižší úrovni jazyka. Je to užitečné, pokud máte starší postscriptovou tiskárnu, protože většina moderních programů produkuje i moderní PostScript.

mpage — Tento nástroj přijímá vstup v PostScriptu a generuje výstup s více stránkami tištěného textu na jedné tiskové stránce. Tuto funkci nabízí i řada dalších programů, například enscript, nenscript a a2ps.

a2ps — Program a2ps, any-to-ps, přijímá vstup v různých formátech a konvertuje je na PostScript.

Jak všechno nastavit

V případě běžných konfigurací můžete tuto kapitolu pravděpodobně ignorovat a přejít rovnou ke kapitole 9, nebo ještě lépe, nahlédnout do dokumentace dodavatele. Většina distribucí Linuxu nabízí jeden či více „blbuvzdorných“ nástrojů, které umožňují za běžných podmínek a při běžných tiskárnách vše dále popsané jednoduše nastavit.

Pokud nástroje dodavatele nefungují, nebo pokud byste chtěli interaktivně řídit parametry tisku při tisku, měli byste použít jiný systém. Dobrou volbou je PDQ, nabízí velmi dobrou funkčnost a jednoduchou konfiguraci. Dalším dobrým systémem je APS Filter, který konfiguruje fronty a filtry LPD na většině unixových systémů.

Dále můžete vyzkoušet rozhraní tiskového systému na stránkách http://www.linuxprinting.org/, která umožňují propojit řadu volně šířených ovladačů s různými spoolovacími systémy. Po završení celého projektu budou tato rozhraní nabízet nejlepší funkčnost – podporují všechny typy volně šířených ovladačů, podporují uživatelská nastavení a všechny běžné spoolery.

Konfigurace PDQ

PDQ může nastavit superuživatel nebo i normální uživatel. Superuživatel mění celosystémová nastavení v */etc/printrc*, běžný uživatel mění pouze svá nastavení v *.printrc*. Konfigurace je však v obou případech stejná.

Pokud PDQ není součástí vaší distribuce, můžete získat zdrojové kódy z domovské stránky PDQ (http://feynman.tam.uiuc.edu/pdq/) a přeložit si je sami. Překlad je velmi jednoduchý, ale musíte mít nainstalovány vývojové verze různých knihoven GTK, vývojové verze knihoven C, překladač a další vývojové nástroje.

Ovladače a rozhraní

PDQ umožňuje uživateli volit, na které tiskárně tisknout. Tiskárna se v PDQ definuje kombinací „ovladače“ a „rozhraní“. Jak ovladače, tak rozhraní nejsou v zásadě nic jiného, než kusy textu v konfiguračním souboru PDQ.

Rozhraní definuje, jak dostat data k tiskárně. Nejobvyklejší rozhraní, definovaná ve vzorovém souboru *printrc*, který je součástí distribuce PDQ jsou:

local-port Lokální port je rozhraní, které komunikuje se sériovým nebo paralelním portem počítače, na němž PDQ běží. Pomocí tohoto rozhraní může PDQ tisknout přímo na paralelní port. Pokud pracujete na víceuživatelském systému, mohou tím vznikat zmatky a pokud pracujete v síti, funguje lokální port jenom na jediném systému. V těchto případech můžete v *lpd* definovat pro daný port nefiltrovanou frontu a tisknout prostřednictvím démona *lpd* ze všech systémů a účtů bez jakýchkoliv potíží. Parametrem tohoto rozhraní je název zařízení, typická hodnota je například */dev/lp0*.

bsd-lpd Toto rozhraní po síti komunikuje s démonem LPD nebo se síťovou tiskárnou, která umí protokol LPD. PDQ podporuje odesílání, rušení a řazení úloh na rozhraní LPD. Parametry tohoto rozhraní jsou název hostitele a název fronty.

appletalk Toto rozhraní umožňuje tisknout na tiskárny v síti Appletalk; pokud máte tiskárnu připojenu k počítači Mac, je to váš případ. Pro práci s tímto rozhraním musíte mít instalován balík Netatalk.

Ovladač pak říká, jak musí být data zpracována do formátu, kterému rozumí konkrétní tiskárna. U postscriptových tiskáren to bude obnášet konverzi ASCII na PostScript, u tiskáren bez podpory PostScriptu pak konverzi z PostScriptu do nativního jazyka tiskárny prostřednictvím Ghostscriptu.

Pokud žádný z ovladačů dodávaných s PDQ vaší tiskárně nevyhovuje, čtěte dál a dozvíte se, jak si napsat vlastní.

Definice tiskárny

Abyste v PDQ definovali tiskárnu, musíte:

- Ověřit, že máte v systému nebo v osobním souboru .printrc definovány správné rozhraní a ovladač.
- Pokud chcete tiskárnu definovat v /etc/printrc (tedy pro všechny uživatele), musíte být superuživatel.
- Spusťte xpdq a zvolte Printer -> Add printer. Objeví se „Průvodce“, ve kterém vyberete rozhraní a ovladač.

A to je opravdu všechno, hlavní práce je totiž nalézt nebo vytvořit správnou specifikaci ovladače v případě, že nenaleznete žádný vhodný.

Vytvoření deklarace ovladače

Nyní si ukážeme příklad toho, jak v PDQ deklarovat ovladač. Než se do toho ale pustíte, existuje několik míst, kde byste měli zkusit najít existující deklaraci:

- Samotný PDQ obsahuje řadu připravených ovladačů.
- Databáze na LinuxPrinting.org (http://www.linuxprinting.org/database.html) obsahuje program PDQ-O-Matic (http://www.linuxprinting.org/pdq-doc.html), který vytvoří PDQ specifikaci podle informací v databázi. Za předpokladu, že databáze obsahuje správné údaje o tiskárně, je to nejlepší metoda pro oživení tiskárny bez podpory PostScriptu.
- Vytvořil jsem nástroj ppdtopdq (http://www.picante.com/~gtaylor/download/printing/), který vezme PPD soubor (PostScript Printer Definition) a se zhruba 75% úspěšností z něj vytvoří PDQ specifikaci. Tento postup použijte, pokud máte postscriptovou tiskárnu.

Pokud budete vytvářet PDQ ovladač sami, podívejte se na několik míst:

- Syntaxe pro specifikaci ovladače je poměrně bohatá a je plně dokumentovaná na manuálové stránce printrc(5).
- Distribuce PDQ obsahuje několik příkladů. Podívejte se na soubor pro Epson Stylus, který demonstruje strukturu pro definici u tiskáren ovládaných přes Ghostscript.
- Databáze tiskáren (http://www.linuxprinting.org/database.html) obsahuje ovladače pro více než 600 tiskáren. Z nich zjistíte, jaké parametry předat Ghostscriptu, případně jakým programem výstup pro Ghostscript zpracovat.

Pokud vytvoříte vlastní ovladač nebo pokud vylepšíte některý z distribuce PDQ nebo nějaký vytvořený pomocí výše uvedených generátorů, podělte se o něj se zbytkem světa. Pošlete mi jej (gtaylor+pht@picante.com) a já už zajistím, že další uživatelé stejného typu tiskárny jej najdou.

A nyní se podívejme na postup vytvoření ovladače pro tiskárnu, která je v databázi Printing HOWTO uvedena jako funkční, ale nenalezli jste pro nic PDQ specifikaci. Příklad se bude týkat tiskárny Canon BJC-210.

Nejprve se podívejme do databáze na údaje o této tiskárně (http://www.linuxprinting.org/show_printer.cgi?recnum=58752). Všimněte si, že je její podpora „perfektní“, takže můžete očekávat srovnatelné (nebo lepší) výsledky jako uživatelé Windows. Důležité informace jsou v posledních dvou údajích:

Notes — Tyto poznámky často obsahují užitečné informace. U řady tiskáren zde najdete odkaz na „More Info“, který obvykle vede na stránky uživatelů této tiskárny nebo na domovské stránky ovladače.

Driver List — U řady tiskáren je uveden seznam ovladačů, o nichž se ví, že fungují. To je nejdůležitější část. Můžete zvolit odkaz na ovladač, který vede na stránky konkrétního ovladače, kde se často dovíte podrobnější informace o tom, jak ovladač provozovat a případně i další odkazy na různé jiné informace.

Ovladač PDQ plní dvě logické funkce: interakci s uživatelem a zpracování tiskové úlohy. Ty jsou v souboru reprezentovány na třech místech:

Option Declarations — Definuje parametry, které může uživatel nastavit a deklaruje proměnné PDQ pro použití v dalších částech ovladače.

Language Filters — Zpracovávají tiskovou úlohu ze vstupního formátu (typicky PostScript nebo ASCII) do jazyka, kterému tiskárna rozumí (například PCL). V této i v následující části je možné použít proměnné, nastavené v předchozí části.

Output Filter — Tento finální filtr zpracovává data pro tiskárnu bez ohledu na typ vstupu, velmi často se zde nastavují parametry tisku.

Podívejme se nyní na jednotlivé části pro tiskárnu Canon BJC-210:

Options

V seznamu ovladačů pro tuto tiskárnu najdeme ovladače bj200 a bjc600, což jsou oba ovladače pro Ghostscript. V poznámkách se dozvíme, že pro černobílý tisk se doporučuje ovladač bj200.

Tedy, co se týče uživatele, nabízí tiskárna BJC-210 minimálně jednu užitečnou volbu: uživatel si může zvolit černobílý nebo barevný tisk. Nadeklarujme tuto volbu jako „MODE“:

```
option {
  var = "MODE"
  desc = "Režim tisku"
  # default_choice "Color"    # odkomentujte -> implicitně barva
  choice "BW" {
    # Hodnota "value" se přiřadí proměnné MODE. My přiřazujeme
    # text odpovídající různým ovladačům pro Ghostscript-
    value = "bj200"
    help = "Rychlý černobílý tisk pouze s černou náplní."
    desc = "Černobílý"
  }
  choice "Color" {
    value = "bjc600"
    help = "Plně barevný tisk"
    desc = "Barevný"
  }
}
```

S použitím výše uvedené deklarace uvidí uživatel v parametrech tiskárny v *xpdq* možnosti Color a BW. V řádkovém nástroji *pdq* může volit parametry *-oBW* a *-oColor*. Implicitní hodnotu je možné nastavit v *xpdq*, nebo deklarovat klíčovým slovem *default_choice*.

Language Filtering

PDQ zjišťuje typ vstupu příkazem *file*. Pro každý typ, který příkaz *file* identifikuje, a který budete chtít obsluhovat, musíte definovat oddíl *language_driver*. V příslušném oddílu se pak nachází vesměs jen skript zpracovávající jazyk tiskové úlohy, vytvořený v libovolném (!) jazyce (implicitně se používá Bourne shell).

V našem případě budeme chtít tisknout Postscript a ASCII. Budeme potřebovat dva ovladače – jeden spouštějící Ghostscript pro tisk úlohy v PostScriptu a druhý, který do ASCII souborů doplní řídící znaky CR:

```
# Použije se první language_driver odpovídající výstupu příkazu file(1)
language_driver ps {
  # file(1) vrací "PostScript document text conforming at..."
  filetype_regx = "postscript"
  convert_exec = {
    gs -sDEVICE=$MODE -r360x360 \     # parametry gs z databáze
       -q -dNOPAUSE -dBATCH -dSAFER \ # "obvyklé" parametry Ghostscriptu
       -sOutputFile=$OUTPUT $INPUT    # zpracování INPUT na OUTPUT

    # Poslední dva uvedené řádky budou stejné pro většinu tiskáren
    # podporovaných přes Ghostscript. První řádek však bude pro každou
```

```
    # tiskárnu jiný.
  }
}

# Text deklarujeme až za PostScriptem, protože file(1) velmi často identifikuje
# PostScript jako text (což je ostatně pravda).
language_driver text {
  # Netestujeme filetype_regx; máme shodu s názvem ovladače - "text"
  convert_exec = {#!/usr/bin/perl
     # Perl, čistě jenom proto, že můžeme!
     my ($in, $out) = ($ENV{'INPUT'}, $ENV{'OUTPUT'});
     open INPUT, "$in";
     open OUTPUT, ">$out";
     while(<INPUT>) {
        chomp;
        print OUTPUT, "$_\r\n";
     }
  }
}
```

A je to! Jiné tiskárny mohou vyžadovat i výstupní filtr (o kterém budeme hovořit dále), ale pro BJC-210 stačí výše uvedené. Teď už to jenom všechno zabalíme dohromady:

```
driver canon-bjc210-0.1 {
  option {
    var = "MODE"

    ...

  }

  # Použije se první language_driver odpovídající výstupu příkazu file(1)
  language_driver ps {

    ...

  }

  # Text deklarujeme až za PostScriptem, protože file(1) velmi často
  # identifikuje PostScript jako text (což je ostatně pravda).
  language_driver text {

    ...

  }
}
```

Output Filtering

Pokud chceme nějakým mechanismem zasáhnout všechny tiskové úlohy, nebo provést nějakou transformaci nad daty všech typů, pak to provedeme v části *filter_exec*. U předchozího příkladu jsme nic takového nepotřebovali, ale čistě kvůli příkladu si nyní ukážeme volbu duplexního režimu a rozlišení u tiskáren LaserJet a tiskáren, které podporují PJL:

```
river generic-ljet4-with-duplex-0.1 {
  # Nejprve dvě uživatelem nastavitelné volby:
  option {
    var = "DUPLEX_MODE"
    desc = "Duplex Mode"
    default_choice = "SIMPLEX"
    choice "SIMPLEX" {
      value = "OFF"
      desc = "Jednostranný tisk"
    }
    choice "DUPLEX" {
      value = "ON"
      desc = "Oboustranný tisk"
    }
  }

  option {
    var = "GS_RES"
    desc = "Rozlišení"
    default_choice = "DPI600"
    choice "DPI300" {
      value = "-r300x300"
      desc = "300 dpi"
    }
    choice "DPI600" {
      value = "-r600x600"
      desc = "600 dpi"
    }
  }

  # Vstup v PostScriptu předáváme ovladači /jet4et4 Ghostsciptu:
  language_driver ps {
    filetype_regx = "postscript"
    convert_exec = {
       gs -sDEVICE=ljet4 $GS_RES \
          -q -dNOPAUSE -dBATCH -dSAFER \
          -sOutputFile=$OUTPUT $INPUT
    }
  }

  # Nakonec obalíme úlohu příkazy PJL:
  filter_exec {
    # potřebujeme echo, které umí escape sekvence...
    echo -ne '\33%-12345X' > $OUTPUT

    echo "@PJL SET DUPLEX=$DUPLEX_MODE"    >> $OUTPUT
    # Můžete přidávat i další @PJL příkazy.
    # Nezapomeňte výstup přidat (>>) do výstupního souboru!

    cat $INPUT >> $OUTPUT
    echo -ne '\33%-12345X' >> $OUTPUT
  }
}
```

Konfigurace LPD

Většina linuxových systémů obsahuje LPD. V této kapitole popisujeme základní nastavení LPD, podrobnostem o vytváření složitějších filtrů a o síťovém tisku se věnujeme samostatně.

Základní konfigurace LPD

Základní nastavení LPD vede k systému, který umí řadit úlohy do fronty a tisknout je. Nestará se o to, zda tiskárna bude úlohám rozumět a výsledný vzhled asi nebude dokonalý. Někde ale musíme začít.

Chcete-li do LPD přidat fontu, musíte upravit soubor */etc/printcap* a vytvořit nový adresář pod */var/spool/lpd*.

Záznam v */etc/printcap* vypadá takto:

```
# LOCAL djet500
lp|dj|deskjet:\
        :sd=/var/spool/lpd/dj:\
        :mx#0:\
        :lp=/dev/lp0:\
        :sh:
```

Tímto definujeme frontu pojmenovanou *lp*, *dj* a *deskjet*, umístěnou adresáři */var/spool/lpd/dj*, bez limitu velikosti úlohy, ze které se tiskne na zařízení */dev/lp0* a která k úloze nepřidává hlavičkovou stránku (například s údajem o tom, kdo úlohu vytvořil).

Teď si přečtěte manuálovou stránku *printcap*.

Výše uvedený příklad vypadá velmi jednoduše, je v něm ale zrada – pokud na tiskárnu DeskJet 500 nebudeme posílat data, kterým bude tiskárna rozumět, budou se tisknout divné věci. Například pokud vytisknete normální unixový text, bude tiskárna doslovně interpretovat řídicí znaky LF a výsledkem bude:

```
This is line one.
                This is line two.
                                This is line three.
```

a tak dále. Pokud tímto způsobem budeme tisknout PostScript, dostaneme krásný výpis postscriptových příkazů zobrazený v tomto „schodišťovém“ efektu, výsledek ale k ničemu moc nebude.

Zjevně je potřeba něco navíc, a to je právě úkolem filtrace. Pozorný čtenář, který nepřeskočil manuálovou stránku *printcap,* jistě postřehl parametry *if* a *of*. Nuže, *if*, neboli *input filter*, je to, co teď potřebujeme.

Pokud si napíšeme krátký skript, který ke znakům LF doplní i znaky CR, můžeme zrušit schodišťový efekt. Přidáme tedy řádek s parametrem *if*:

```
lp|dj|deskjet:\
        :sd=/var/spool/lpd/dj:\
        :mx#0:\
        :lp=/dev/lp0:\
        :if=/var/spool/lpd/dj/filter:\
        :sh:
```

Filtrační skript může vypadat například takto:

```
#!perl
# Předchozí řádek musí obsahovat celou cestu k perlu
# Skript musí být spustitelný, chmod 755 filter
```

```
while(<STDIN>){chomp $_; print "$_\r\n";};
# Pokud chcete na konci úlohy odstránkovat, doplňte print "\f";
```

Pokud bychom to všechno udělali, dostaneme frontu, jejímž prostřednictvím můžeme tisknout normální textové soubory a dostaneme použitelné výsledky. (Ano, existují asi čtyři miliony lepších způsobů, jak filtr napsat, ale nejsou tak názorné. Klidně to udělejte efektivněji.)

Posledním zbývajícím problémem je fakt, že tisk prostého textu dneska málokoho nadchne – rozhodně bychom chtěli tisknout i PostScript, grafiku a podobně. Jistě to jde a je to poměrně jednoduché. Celá finta spočívá v rozšíření schopností výše uvedeného filtru.

Takovému filtru říkáme „magický" filtr. Plní stejnou úlohu jako jazykové filtry v PDQ. Nenamáhejte se s tím psát si jej sami, pokud netisknete vyloženě avantgardní formáty – řada už existuje a mají snadno použitelná interaktivní konfigurační rozhraní. Stačí prostě zvolit vhodný existující filtr.

LPD-O-Matic

Ldpomatic (http://www.linuxprinting.org/lpd-doc.html) je filtr, používající data z databáze LinuxPrinting.org. Podporuje prakticky všechny volně distribuované ovladače tiskáren včetně ghostscriptových ovladačů, uniprintových ovladačů a dalších různých filtračních programů. Funguje s různými verzemi LPD včetně základní BDS verze i nové verze VA Linux LPD a podporuje nastavení parametrů. (Uživatelé LPRng by měli použít balíky magicfilter nebo apsfilter; speciální databází ovládaná verze magicfilter je ve vývoji.)

APS Filter

apsfilter (http://www.apsfilter.org/) je filtr určený pro použití v různých Unixech. Podporuje prakticky všechny ghostscriptové ovladače. I on pracuje s různými typy LPD, včetně základního BSD i LPRng. V současné době jde o asi nejlepší systém pro podporu tiskáren, které neumějí PostScript.

Tiskové filtry RHS

RHS je filtrační systém vytvořený společností Red Hat. Jeho distribuce začala, pokud vím, ve verzi Red Hat 4, kde sloužil jako základ pro grafickou nástavbu printtool pro snadnou konfiguraci tiskáren. Jiné distribuce včetně Debianu dnes distribuují kombinaci rhs-printfilter/printtool. Jedná se asi o nejrozšířenější filtrační systém.

Systém rhs je založen na ASCII databázi, která je jeho součástí. Ta obsahuje údaje o podpoře řady ovladačů Ghostscript a Uniprint, nepodporuje však ovladače filtrového typu. Filtry navíc příliš nepodporují uživatelskou volbu parametrů tisku.

Printtool ukládá v adresáři fronty konfigurační soubor *postscript.cfg*. V tomto souboru podobnému shell skriptům představuje každé nastavení jednu proměnnou. V neobvyklých případech můžete užitečné zásahy provést přímo modifikací konfiguračního souboru, kterou nemusí umožňovat printtool – typicky půjde o specifikaci neobvyklého ghostscriptového ovladače nebo PPD souboru u VA verze rhs-printfilters.

VA Linux v rámci kontraktu s HP obsahuje některá vylepšení systému rhs-printfilters. Správné verze umožňují volbu parametrů postscriptových tiskáren, které jsou řízeny PPD soubory. O tomto systému hovoříme v kapitole 8.2.2.

Použití těchto filtrů má jednu nevýhodu: starší verze *lpd* nespouštějí filtr *if* pro vzdálené tiskárny, zatímco většina novějších verzí ano (i když často jen bez parametrů). Verze LPD obsažené v moderních distribucích Linuxu a FreeBSD jej spouštějí, verze LPD ve většině komerčních Unixů ne. Více informací na toto téma uvádíme dále v kapitole věnované tisku v síti. Pokud pracujete jen s lokálně připojenými tiskárnami, tyto problémy se vás netýkají.

LPD pro postscriptové tiskárny

I když většina verzí LPD nezpracovává PostScript dobře (bez ohledu na uživatelská nastavení), VA Linux nedávno modifikoval LPD a filtrační software Red Hat, takže nyní podporuje postscriptové tiskárny poměrně dobře. V současné době tento systém funguje jen pro Red Hat 6.2 a novější, i když lze balík snadno upravit i pro jiné distribuce.

Jak to funguje

Nový systém VA používá soubory PPD – Postscript Printer Definition. PPD soubory poskytuje výrobce tiskárny a obsahují informace o funkcích, které tiskárna umí, společně s postscriptovým kódem, jejž tyto funkce aktivuje. V systému VA funguje klasické schéma LPD poněkud odlišně.

1. Uživatel může nastavovat parametry příznakem *-o*. Můžete například zadat *-o MediaType:Transparency* při tisku na fólii. Je možné také použít nástavby jako GPR, které umožňují volit parametry v dialogovém rozhraní. Příslušné obrázky můžete vidět v kapitole 3.3.1.
2. LPR předá parametry LPD jako rozšířené atributy v řídicím souboru LPD.
3. Modifikovaná verze rhs-printfilters obdrží rozšířené parametry v proměnných prostředí a pomocí ppdfilt je přidá k tiskovým datům.

Získání a instalace

RPM balíky nebo zdrojové kódy můžete získat na domovské stránce projektu na SourceForge, http://printing.sourceforge.net/. Informace o instalaci najdete v instalačním dokumentu micro-HOWTO, http://printing.sourceforge.net/gpr-libppd-uhowto.html. V zásadě musíte odinstalovat původní Red Hat verzi balíků printtool, lpd a rhs-printfilters, a nainstalovat jejich VA verze společně s balíky ppdfilt, gpr a s dalšími nástroji.

Dále potřebujete PPD soubor pro svou postscriptovou tiskárnu. Tyto soubory se obvykle dají snadno najít. VA Linux a HP distribuují PPD soubory pro řadu modelů Laserjet. Další výrobci poskytují PPD soubory ke svým tiskárnám, Adobe distribuuje PPD soubory pro řadu tiskáren na adrese http://www.adobe.com/products/printerdrivers/winppd.html.

V současné době je instalace většiny nástrojů poměrně obtížná. Budoucí instalační nástroje mají být postaveny nad konfigurační knihovnou tiskáren, libprinterconf, která umožňuje automatickou detekci a konfiguraci rhs-printfilter pro síťové i lokální tiskárny.

> Je možné použít GPR i samostatně, bez modifikovaného LPD a dokonce i bez rhs-printfilters. GPR je možné přeložit s podporou kompletního zpracování postscriptových úloh. Tato možnost může být z hlediska instalace jednodušší pro uživatele, kteří nepotřebují tisknout přímo přes *lpr*.

Nastavení parametrů PostScriptu

Po nastavení LPD systému s podporou PostScriptu můžete parametry tiskáren nastavovat dvěma způsoby:

Pomocí grafického rozhraní

Abyste mohli použít GPR, musíte nejprve vybrat správný soubor PPD. Pak budou na záložce Advanced k dispozici parametry tiskárny. Základní parametry ppdfilt najdete na záložce Common.

Z příkazového řádku

Tato verze *lpr* podporuje přepínač *-o*. S jeho pomocí můžete zadat jakoukoliv dvojici parametr/hodnota podle PPD souboru vaší tiskárny. Předpokládejme například, že soubor PPD obsahuje následující:

```
*OpenUI *PrintQuality/Print Quality: PickOne
*DefaultPrintQuality: None
*OrderDependency: 150 AnySetup *PrintQuality
*PrintQuality None/Printer Setting: ""
*PrintQuality Quick/QuickPrint:  "<< /DeviceRenderingInfo ...
*PrintQuality Normal/Normal: "<< /DeviceRenderingInfo << /...
*PrintQuality Pres/Presentation: "<< /DeviceRenderingInfo ...
*PrintQuality Image/1200 Image Quality: "<< /DeviceRenderi...
*CloseUI: *PrintQuality
```

Parametr PrintQuality povoluje hodnoty Quick, Normal, Pres a Image. Na příkazovém řádku můžete zadat:

```
% lpr -o PrintQuality:Image file.ps
```

Kromě toho existuje řada voleb společných pro všechny tiskárny, ty budou fungovat stejně jako volby definované v souboru PPD. Jsou to například:

page-ranges Rozsah stránek pro tisk, například *page-ranges:2-3*

page-set Umožňuje tisknout pouze liché (odd) nebo sudé (even) stránky. Například *page-set:odd*

number-up Na každý list můžete vytisknout více stránek, například *number-up:2_*

Další parametry najdete na manuálové stránce ppdfilt.

Přístupová práva k souborům

Na základě spousty žádostí uvádím výpis přístupových práv u důležitých souborů na svém systému. Nastavení lze provést i lepšími způsoby, například pomocí SGID binárních programů, nemusí být vše nastaveno SUID root. Jde o implicitní nastavení systému a funguje dobře. (Upřímně řečeno, pokud by implicitní nastavení nefungovalo, je čas na změnu dodavatele...)

```
-r-sr-sr-x   1 root     lp    /usr/bin/lpr*
-r-sr-sr-x   1 root     lp    /usr/bin/lprm*
-rwxr--r--   1 root     root  /usr/sbin/lpd*
-r-xr-sr-x   1 root     lp    /usr/sbin/lpc*
drwxrwxr-x   4 root     lp    /var/spool/lpd/
drwxr-xr-x   2 root     lp    /var/spool/lpd/lp/
```

Lpd musí běžet jako root, aby se mohl připojit na nízký port, určený pro službu lpd. Po připojení na port by měl přejít na UID lp.lp nebo tak nějak, ale nemyslím, že to dělá. I to je jeden z důvodů, proč nepoužívat standardní BSD LPD.

PDQ používá jiný mechanismus, který není založen na démonovi, takže používá jiné programy. Jediné SUID root programy jsou rozhraní k *lpd – lpd_cancel*, *lpd_print* a *lpd_status*. Ty jsou SUID, protože vlastní unixový tiskový server požaduje, aby tiskové požadavky pocházely z privilegovaného portu. Pokud jako tiskárny ovládané přes PDQ používáte pouze síťové tiskové servery (například HP JetDirect nebo MarkNet od Lexmarku), pak ani tyto programy nemusí být SUID.

Velké instalace

Velké instalace, čímž myslím sítě s více než dvěma tiskárnami nebo počítači, mají speciální požadavky. Dále uvádíme několik doporučení. V opravdu velkých prostředích je problémem už samotná distribuce nastavení *printcap* a filtrů. Tyto problémy řeší Cisco Enterprise Print System (http://ceps.sourceforge.net/) a představuje tak dobrý začátek, případně téměř kompletní řešení,

v závislosti na skutečných potřebách. Střední a velká prostředí si mohou vystačit s nativními funkcemi LPRng.

- Každá tiskárna by měla mít jedno řídicí místo, odkud může administrátor pozastavovat, řadit a přesměrovávat frontu. Lze to implementovat tak, že vše budete tisknout na lokální server, který pak úlohy řadí a směruje je na správné tiskárny. Rozsáhlejší prostředí nebo distribuované sítě mohou používat samostatný server pro každou budovu nebo jinou jednotku.
- Používejte LPRng, přinejmenším na serverech – BSD LPD má příliš mnoho chyb na „opravdové“ použití. Platí to i pro CUPS – alespoň za stavu v polovině roku 2000. Nemusíte mi ale věřit – vyzkoušejte si různé spoolery a zjistíte, který vám vyhovuje nejvíce.
- Klientské systémy by neměly používat vlastní tiskové konfigurace. Tuto funkci lze implementovat pomocí rozšířené syntaxe *printcap* v LPRng, takže můžete mít všude stejný *printcap*. CUPS to řeší pomocí distribuované databáze.
- Tiskové fronty by se neměly jmenovat podle typu tiskárny, ale nějak rozumně, například podle umístění tiskárny nebo podle možností tiskárny. Za tři roky, až se tiskárna pokazí, ji budete moci nahradit jinou a nedojde ke zmatkům.
- Vytvořte webové stránky s podrobnými informacemi o tiskárnách, včetně jejich umístění, možností a podobně. Mohly by případně vypisovat i obsah tiskové fronty a umožnit smazání úloh z fronty. Ve složitých síťových prostředích není možné pracovat bez dostatečné dokumentace.
- Na unixových systémech používejte PDQ nebo něco podobného, co umožňuje nastavovat parametry tiskové úlohy a zároveň nutí provádět veškerá ghostscriptová zpracování pod správným UID. Pokud používáte pouze postscriptové tiskárny (což je ideální), můžete použít také GPR nebo XPP, které jsou pěknější.
- Na systémech Windows a Apple používejte všude buď konkrétní ovladače konkrétních tiskáren (Samba podporuje mechanismus automatické aktualizace ovladačů), anebo ještě lépe, používejte všude obecný postscriptový ovladač. Nikdy nemíchejte různá nastavení, jednoduché textové editory často vytvářejí na různých ovladačích různé výstupy a uživatelé se těžko vyrovnávají s tím, když se výsledná podoba výtisku mění pro každou dvojici klient/tiskárna.
- Pokud je to možné, tisknete-li velké objemy dokumentů, kupte si velkokapacitní tiskárnu. Pokud na to nemáte, využijte funkci více tiskáren na jedné frontě podporovanou LPRng, a obstarejte si chůvu – tiskárny jsou složitá mechanická zařízení a v intenzivním provozu mají tendenci papír zmuchlat nebo spotřebovat...
- Nevěřte tomu, že tiskárna musí být připojena k počítači, ethernetové „tiskové servery“ stojí méně než 100 dolarů. Možnost umístit tiskárnu kdekoliv, kde je síť, je velká výhoda oproti vynucenému umístění poblíž počítače. Snadno pak tiskárny soustředíte na jedno centrální místo.
- Používejte SNMP nebo jiné monitorovací mechanismy, které jsou k dispozici – snadno pak bude moci pověřená osoba doplňovat papír, toner a podobně. Správu tiskáren s podporou SNMP umožňuje například *npadmin* (viz kapitola 11.10.1).

Účtování

Standardní LPD nabízí jen malou pomoc při účtování tisku. V parametru *af* souboru printcap můžete definovat jméno účtovacího souboru, to se však pouze předává jako parametr filtru *if*. Sami

pak musíte zajistit, aby filtr provedl zápis do účtovacího souboru a sami jej pak musíte zpracovávat (tradiční formát je vhodný zejména pro řádkový tiskárny a špatně se analyzuje v Perlu, takže není důvod se jej držet). Pokud jako filtr použijete program lpdomatic, musíte jej také upravit, protože vyžaduje specifikaci účtovacího souboru v konfiguračním souboru.

Ghostscript nabízí operátor PageCount, který lze použít k určení počtu stránek v každé úloze – typicky pouze na konec každé postscriptové úlohy doplníte pár řádků pro zápis údajů do účtovacího souboru. Pěkný příklad najdete v souboru *unix-lpr.sh* v distribuci Ghostscriptu.

Standardní implementace účtování provádí zápis do účtovacího souboru přímo z interpretu Ghostscriptu a není tak použitelná s doporučeným parametrem *-dSAFER*. Lepší řešení by bylo po skončení úlohy zjistit počet stránek PJL dotazem na tiskárnu, nebo napsat v postscriptu kód, který vypíše počet stránek na stdout – zde jej lze snadno odchytit a zpracovat bez nutnosti zápisu do souboru.

Spooler LPRng obsahuje příklad implementace účtování pro tiskárny HP – předpokládám, že používá PJL dotazy na tiskárny. Tento způsob by měl fungovat u většiny PJL, postscriptových a SNMP tiskáren, které podporují obousměrnou komunikaci.

Pokud máte síťovou tiskárnu s podporou SNMP, můžete pomocí programu *npadmin* zjistit počet stránek po skončení každé úlohy. Tato metoda by měla spolehlivě fungovat pro všechny typy úloh. Další informace o programu *npadmin* najdete v kapitole 11.10.1.

Řešení jednotlivých distribucí

Tato kapitola je ze své podstaty neúplná. Klidně mi pošlete informace o své oblíbené distribuci. Momentálně nevím o žádné distribuci, která by podporovala nebo přímo poskytovala software, jejž doporučuji malým uživatelům: PDQ. Několik distribucí podporuje CUPS, což je momentálně systém s nejvíce funkcemi, některé distribuce dokonce používají databázově řízené konfigurační nástroje.

Kromě toho existuje řada samostatných balíků, které usnadňují konfiguraci tiskáren v Unixu. Hovoříme o nich v kapitole 8, najděte si část, odpovídající vámi používanému tiskovému spooleru.

Red Hat

Red Hat obsahuje grafický nástroj pro administraci tiskáren, *printtool*, který umožňuje přidávat vzdálené i síťové tiskárny. Umožňuje zvolit typ ghostscriptem podporovaných tiskáren a dále zařízení, na něž se bude tisknout. Pak v */etc/printcap* definuje tiskovou frontu a používá filtrační program z balíku *rhs-printfilters* k podpoře PostScriptu a dalších běžných vstupních typů. Toto řešení funguje poměrně dobře a v běžných situacích se velmi snadno nastavuje.

Red Hat 6.x obsahuje BSD verzi LPD, Red Hat 7.x používá standardně LPRng.

S Red Hatem narazíte v okamžiku, kdy máte tiskárnu, kterou jejich standardní Ghostscript nepodporuje (používají GNU Ghostscript namísto Aladdin Ghostscriptu, takže je podporováno méně tiskáren). Podívejte se do seznamu podporovaných tiskáren (http://www.linuxprinting.org/printer_list.cgi), kde zjistíte, zda standardní Red Hat vaši tiskárnu podporuje. Pokud ne, můžete si nainstalovat Aladdin Ghostscript a balíky *lpdomatic* nebo *apsfilter*, které vědí všechno o tiskárnách podporovaných nejnovějším Ghostscriptem i o některých dalších.

V budoucích verzích Red Hatu má být *printtool* přepracován, aby podporoval mnohem více tiskáren a zamýšlí se i podpora případných náhrad balíku *rhs-printfilters* (současná verze filtru má problémy s řadou běžných tiskáren, jako jsou různé DeskJety bez PCL a většina Lexmarků). Měly by být zahrnuty také některé funkce PPD pocházející z VA Linuxu.

Debian

Debian nabízí volbu mezi standardním LPD, LPRng a CUPSem. LPRng a CUPS jsou zjevně lepší volby. PDQ najdete v nestabilní verzi. Dále Debian nabízí různé konfigurační nástroje, nejlepší volbou je asi *apsfilter* verze 5 nebo vyšší, protože obsahuje podporu pro ovladače uniprint LPRng a Ghostscriptu. Podporován je i *printtools* od Red Hatu pro ty uživatele, kteří mají rádi grafické nástroje.

SuSE

Tiskový systém v SuSE Linuxu je založen na *apsfilter* s některými vylepšeními. *apsfilter* od SuSE rozeznává všechny běžné formáty souborů (včetně HTML, pokud je nainstalován program *html2ps*). V systémech SuSE jsou dvě možnosti nastavení tiskáren:

- YaST umožňuje nastavit tiskárny „PostScript“, „DeskJet“ a „Other“, podporované ovladači Ghostscript, kromě toho je možné nastavit tiskárny HP GDI (DeskJet 710/720, 820, 1000, pomocí balíku ppa). YaST zapíše do /etc/printcap údaje pro každou tiskárnu („raw“, „ascii“, „auto“ a „color“, pokud jde o barevnou tiskárnu). Dále YaST vytvoří adresář fronty a nastaví soubory apsfilterrc, kde můžete doladit některá nastavení (preload Ghostscriptu, velikost a orientace papíru, rozlišení, escape sekvence tiskárny a podobně). Pomocí YaST je možné nastavovat i síťové tiskárny (TCP/IP, Samba a Novell NetWare).
- Kromě toho SuSE obsahuje i standardní program SETUP z původního balíku apsfilter (s některými vylepšeními). Tento konfigurační skript spustíte příkazem lprsetup. Jakmile si zvyknete na jeho rozhraní, budete moci konfigurovat místní i síťové tiskárny.

Instalační manuál SuSE popisuje oba postupy instalace.

Wolf Rogner ohlásil se SuSE některé problémy. Vadit mohou následující věci:

- Klasický skript SETUP z apsfilter není v pořádku, stejně jako konfigurační nástroj v KDE. Použijte YaST.
- U síťových tiskáren ovládaných přes Ghostscript budete muset nejprve odkomentovat řádek REMOTE_PRINTER=“remote“ v /etc/apsfilterrc. Pak pomocí YaST nastavte tiskárnu a pomocí konfigurace sítě nastavte vzdálenou tiskovou frontu.
- YaST nepodporuje barevné laserové tiskárny, takže nastavte černobílou tiskárnu, a pak všude v printcap změňte mono na color. Možná budete muset přejmenovat i adresář fronty.

Caldera

Caldera obsahuje LPRng. Nemám tušení, jaké používají konfigurační nástroje.

Jeden ze zaměstnanců Caldery se nedávno stal údržbářem databáze LinuxPrinting.org, zjevně chtějí v budoucích verzích nabízet tiskové systémy založené na CUPS a Foomatic.

Corel

Corel je založen na Debianu, takže pro něj platí to samé, co pro Debian. Navíc používají vlastní konfigurační nástroj, založený na knihovně sysAPS, který pak používá mou databázi. Takto je to vyřešeno ve WordPerfectu.

Corel provozuje diskusní skupinu news://cnews.corel.com/corelsupport.linux.printing, ve které se objevuje hodně příspěvků ohledně WordPerfectu a Corel Linuxu.

Mandrake

Ve verzi 7.2b1 používá Mandrake standardně CUPS. Administrativní rozhraní poskytuje program QtCUPS. Snažili se poskytnout co možná nejvíce ovladačů, a používají PPD soubory pro CUPS, generované rozhraním foomatic (http://www.linuxprinting.org/foomatic.html).

Starší verze Mandrake pokud vím používaly stejné nástroje jako Red Hat.

Slackware

Slackware obsahuje *apsfilter*. Skript SETUP je nainstalován jako program *apsfilterconfig*. Rozumné nastavení byste měli být schopni vytvořit prostým spuštěním tohoto programu.

Možná, že Slackware obsahuje i jiné systémy, ale nemám možnost to zjistit.

Další distribuce

Napište mi, jak to řeší jiné distribuce!

Ghostscript

Ghostscript (http://www.cs.wisc.edu/~ghost/) je nesmírně významný program pro softwarově řízený tisk. Většina programů v Unixu generuje výstup v PostScriptu, jehož podpora je u většiny tiskáren placený doplněk. Ghostscript je naproti tomu zadarmo a z postscriptového vstupu vygeneruje výstup v jazyce, kterému rozumí vaše tiskárna. Při propojení s deklaracemi tiskáren v PDQ nebo se vstupním filtrem *lpd* vám nabízí virtuální postscriptovou tiskárnu a výrazně usnadňuje život.

Ghostscript existuje v několika podobách. Komerční verzi Ghostscriptu, Aladdin, lze zadarmo používat pro osobní potřebu, nesmí však být distribuována komerčními entitami. Tato verze je zhruba rok napřed před free verzí. Momentálně například podporuje řadu barevných inkoustových tiskáren, které starší verze neznají, a má také výrazně lepší podporu PDF.

Hlavní free verzí Ghostscriptu je GNU Ghostscript, což je jednoduše stará verze Aladdin Ghostscriptu. Díky tomuto poněkud podivnému uspořádání je Aladdin samofinancující se free projekt. Vývoj vede L. Peter s několika zaměstnanci, a nové verze licencuje dodavatelům hardwaru a softwaru k použití v komerčních produktech. I když toto uspořádání umožňuje L. Peterovi už řadu let pracovat na Ghostscriptu, bohužel znemožňuje spolupráci s širší vývojářskou komunitou. Toto řešení zejména nevyhovuje autorům ovladačů. Plány L. Petera na odchod do důchodu předpokládají zapojení širší komunity, takže zvažuje změnu licence a vytvořil projekt na SourceForge.

Třetí verzí Ghostscriptu je ESP Ghostcript, udržovaný společností Easy Software Products (autoři CUPSu) na základě kontraktu se společností Epson. ESP Ghostscript je kombinací ovladačů gimp-print projektu a GNU Ghostscriptu, plus vybraných doplňků. Tato verze ještě není plně funkční, ale brzy by měla být vylepšena a měla by usnadnit život majitelům tiskáren s ovladači gimp-print.

Ať už *gs* používáte jakkoliv, rozhodně jej vždy spouštějte se zákazem přístupu k souborům (*-dSAFER*). PostScript je plnohodnotný programovací jazyk a škaredý program v PostScriptu vám může způsobit řadu problémů.

Abychom nezapomněli, PDF, Portable Document Format od Adobe, je (přinejmenším od verze 1.3) pouze o trochu více než komprimovaný PostScript. Ghostscript umí zpracovávat PDF stejně jako PostScript. Možná tak budete první v okolí, kdo umí přímo tisknout PDF.

Spuštění Ghostscriptu

Typicky se Ghostscript spouští filtrem, který používáte (doporučuji *apsfilter* nebo *lpdomatic*), pro účely ladění je však často užitečné spouštět jej přímo.

gs -help vypíše stručný seznam parametrů a dostupných ovladačů (jsou uvedeny ovladače, s nimiž je Ghostscript přeložen, nejde o úplný seznam všech podporovaných ovladačů).

Pro účely testování budete *gs* spouštět například takto: „*gs <parametry> -q -dSAFER -sOutputFile=/dev/lp1 test.ps*“.

Doladění výstupu Ghostscriptu

Existuje řada věcí, které můžete podniknout, pokud není výstup Ghostscriptu uspokojivý (můžete vlastně provést naprosto *cokoliv*, protože máte k dispozici zdrojový kód).

Některé z těchto možností jsou popsány v *Ghostscript User Guide* (soubor *Use.htm*, http://www.cs.wisc.edu/~ghost/aladdin/doc/Use.htm, v distribuci Ghostscriptu, měli byste jej najít v */usr/doc* nebo */usr/share/doc*). Vybrané možnosti můžete použít jako parametry ovladače ve filtračním systému.

Velikost a umístění výstupu

Umístění, velikost a poměr stran obrázku na stránce se v ghostscriptu řídí ovladači specifickými pro danou tiskárny. Pokud zjistíte, že tištěné stránky jsou příliš krátké, příliš dlouhé nebo dvojnásobně veliké, podívejte se do zdrojového kódu ovladače a upravte parametry, které uznáte za vhodné. Bohužel, každý ovladač je jiný, takže neumím říct, co přesně kde nastavit, nicméně většina ovladačů je rozumně dokumentovaná.

Gamma, velikost bodu atd.

Většina tiskáren jiných než laserových trpí tím, že mají velké tiskové body. Výsledkem jsou příliš tmavé obrázky. Pokud máte tento problém s jinak neupravitelným ovladačem, můžete zkusit použít vlastní transformační funkci. Vytvořte následující soubor v adresáři *lib* ghostscriptu a jeho název přidejte při spouštění *gs* před název tisknutého souboru. Možná budete muset upravit konkrétní hodnoty tak, aby vyhovovaly vaší tiskárně. Menší hodnoty vedou k jasnějším bodům. Zejména pokud ovladač používá k rasterizaci barev Floyd-Steinbergův algoritmus, bude rozumné použít menší hodnoty (0.2-0.15).

```
%!
%transformační funkce pro CMYK
{0.3 exp} {0.3 exp} {0.3 exp} {0.3 exp} setcolortransfer
```

Nastavením těchto hodnot můžete „opravit“ tiskárnu, které netiskne určitá barva. Pokud se budete o něco takového snažit, doporučuji jako test soubor *colorcir.ps*, dodávaný s ghostscriptem (v adresáři *examples*).

U řady novějších barevných ovladačů existují řádkové parametry nebo konfigurační soubory, které umožňují pomocí gamma korekce a dalších nastavení uzpůsobit tiskárnu různým typům papíru. Než začnete nevyhovující tisky opravovat přímo zásahy do PostScriptu, zkuste nejprve tato nastavení.

Barevný tisk v Ghostscriptu

Barevný dithering je v Ghostscriptu implicitně optimalizován pro zařízení s nízkým rozlišením. Dithering je poměrně bídný díky snaze vytvářet výstup 60 PPI (ne DPI, ale PPI – počet „zdánlivých“ barevných bodů na palec po provedení ditheringu). Na moderních barevných tiskárnách je výstup

nehezký, zejména inkoustové tiskárny s kvalitním papírem jsou schopny tisknout s mnohem lepším PPI.

Můžete to upravit parametrem *-dDITHERPPI=x*, kde *x* je požadovaná hodnota. Nemusí to fungovat pro všechny ovladače, řada novějších ovladačů (například ovladač *stp* pro Epson Stylus) používá vlastní dithering a tento parametr nebere na vědomí. Jiné ovladače mohou používat buď standardní ghostscriptový dithering, nebo svůj vlastní (například *bjc600* pro tiskárny Canon Bubblejet).

Dithering v Ghostscriptu je poměrně hloupý. Jádro Ghostscriptu jednoduše nepodporuje řadu věcí, potřebných pro kvalitní tisk na moderních tiskárnách. Řada projektů, které by tuto situaci řešily (a svět otevřeného softwaru je schopen tato řešení poskytnout) je brzděna licenční politikou Ghostscriptu a jeho uzavřeným vývojem. Na setkání *Open Source Printing Summit 2000* (http://www.linuxprinting.org/summit.html) tuto situaci řešili všichni, jichž se to týká, a dá se čekat, že v brzké době dojde ke zlepšení.

Sítě

Jednou z funkcí většiny spoolerů je, že podporují přes síť tisk na tiskárny fyzicky připojené k jiným počítačům, nebo přímo do sítě. Při vhodné kombinaci filtračních skriptů a vybraných nástrojů můžete transparentně tisknout na tiskárny ve všech typech sítí.

Tisk na unixové lpd stroje

Aby mohly vzdálené počítače tisknout na vaší tiskárně protokolem LPD, musíte je uvést v souboru */etc/hosts.equiv* nebo */etc/hosts.lpd*. (Pozor na to, že soubor *host.equiv* způsobuje celou řadu dalších efektů, než sem nějaký počítač uvedete, měli byste dobře vědět, co děláte.) Pomocí příznaku *rs* můžete povolit tisk jenom určitým uživatelům vzdáleného systému – přečtěte si manuálové stránky *lpd*.

Když používáte pdq

V PDQ definujete rozhraní tiskárny jako v *bsd-lpd*. Parametry tohoto rozhraní je název vzdáleného počítače a název fronty, průvodce konfigurací vás o tyto údaje požádá.

Když používáte lpd

Abyste mohli tisknout na vzdáleném počítači, záznam v */etc/printcap* by měl vypadat takto nějak:

```
# REMOTE djet500
lp|dj|deskjet:\
        :sd=/var/spool/lpd/dj:\
        :rm=vzdálený.počítač.com:\
        :rp=vzdálená_fronta:\
        :sh:
```

V tomto uspořádání stále existuje i lokální fronta spravovaná démonem *lpd*. Pokud by vzdálený stroj nebyl dostupný, budou úlohy čekat v lokální frontě, dokud je nebude možné odeslat.

Když používáte rlpr

Pomocí *rlpr* můžete odeslat tiskovou úlohu přímo do vzdálené fronty, aniž byste museli konfigurovat *lpd*. Je to užitečné zejména v situacích, kdy pouze čas od času tisknete na různé tiskárny. Z dokumentace k *rlpr* uvádíme:

„Rlpr používá TCP/IP k odeslání tiskových úloh na *lpd* servery kdekoliv v síti."

Na rozdíl od *lpr nevyžaduje*, aby lokální počítač „znal" vzdálenou tiskárnu, na níž chcete tisknout (například prostřednictvím */etc/printcap*), a je tedy považován za pružnější řešení s menšími nároky na administraci.

rlpr je možné použít kdekoliv, kde se používá klasické *lpr* a je s ním zpětně kompatibilní.

Hlavní výhodou *rlpr* je možnost vzdáleně tisknout *odkudkoliv kdekoliv* bez ohledu na to, jak je nakonfigurován systém, z nějž tisknete. Rlpr může fungovat jako filtr stejně jako tradiční *lpr*, takže vzdálení klienti jako netscape, xemacs atd. mohou velmi jednoduše tisknout na lokální tiskárně.

Rlpr je k dispozici na Metalabu, ftp://metalab.unc.edu/pub/Linux/system/printing/.

Tisk na Windows nebo Sambu

V praktickém návodu *Tisk na Windows* (kapitola 16) najdete více informací, než zde.

Když používáte pdq

Pokud vím, neexistuje předdefinované rozhraní *smb*, nemělo by ale být obtížné vyjít z rozhraní appletalk a upravit je. Zkuste to někdo udělat a pošlete mi výsledek!

Podrobnější rady, jak to udělat, najdete dále v části Windows/LPD.

Když používáte lpd

Je možné směrovat tiskovou frontu přes program *smbclient* (součást balíku samba) na tiskovou službu SMB na protokolu TPC/IP. Samba obsahuje skript *smbprint*, který to zajišťuje. Stručně řečeno, v adresáři fronty příslušné tiskárny vytvoříte konfigurační soubor a jako filtr *if* nastavíte skript *smbprint*.

Záznam v */etc/printcap* bude vypadat takto:

```
lp|remote-smbprinter:\
    :sh:\
    :lp=/dev/null:\
    :sd=/var/spool/lpd/lp:\
    :if=/usr/local/sbin/smbprint:
```

Přečtěte si dokumentaci uvedenou ve skriptu *smbprint*, kde naleznete více informací o potřebných nastaveních.

Kromě toho můžete použít *smbclient* a směrovat soubor přímo na tiskovou službu SMB, bez použití *lpd*. Viz manuálové stránky.

Tisk na NetWare

Balík *ncpfs* obsahuje program *nprint*, který poskytuje stejné funkce jako *smbprint*, ovšem v sítích NetWare. Balík *ncpfs* můžete získat na Metalabu, ftp://metalab.unc.edu/pub/Linux/system/filesystems/ncpfs/. Z dokumentace uvádíme:

S pomocí *ncpfw* můžete v Linuxu připojovat svazku na serveru NetWare. Můžete také tisknout do tiskových front NetWare, nebo fronty NetWare obsluhovat spoolerem v Unixu. Potřebujete jádro 1.2.x nebo 1.3.54 nebo vyšší. *ncpfs* nefunguje s jádry 1.3.x nižšími než 1.3.54.

Když používáte lpd

Abyste mohli *nprint* použít s *lpd*, napište si malý skript, který bude tisknout *stdin* na tiskárnu NetWare, a ten nainstalujte jako filtr *if* pro *lpd* frontu. Dostanete něco jako:

```
sub2|remote-Nwprinter:\
        :sh:\
        :lp=/dev/null:\
        :sd=/var/spool/lpd/sub2:\
        :if=/var/spool/lpd/nprint-script:
Skript nprint-script# Nejprve zkuste účet guest bez hesla!
/usr/local/bin/nprint -S síť -U jméno -P heslo -q fronta -
```

Tisk na tiskárny EtherTalk (Apple)

Balík *netatalk* obsahuje něco podobného jako *nprint* a *smbclient*. Postup tisku na a ze sítí Apple už byl popsán daleko lépe, než bych to dokázal já. Podívejte se na dokument *Linux Netatalk – HOWTO* (http://thehamptons.com/anders/netatalk/)

Když používáte pdq

PDQ má deklarováno rozhraní *appletalk*. To používá balík Netatalk k tisku a síťovou tiskárnu Apple. V průvodci přidáním nové tiskárny v programu *xpdq* stačí jednoduše vybrat toto rozhraní.

Tisk na síťovou tiskárnu

Řada tiskáren obsahuje přímo ethernetové rozhraní a je možné na ně přímo tisknout, typicky protokolem LPD. Držte se instrukcí, které jsou přiloženy u tiskárny nebo jejího síťového adaptéru, nicméně obecně platí, že na těchto tiskárnách „běží“ *lpd* a nabízí jednu nebo více front, na které můžete tisknout. Tiskárny HP mohou fungovat například s takovouto konfigurací:

```
lj-5|remote-hplj:\
        :sh:\
        :sd=/var/spool/lpd/lj-5:\
        :rm=název.tiskárny.com:\
        :rp=raw:
```

nebo, pokud použijete PDQ a rozhraní *bsd-lpd*, pak budou parametry *REMOTE_HOST=název.tiskárny.com* a *QUEUE=raw*.

Tiskárny HP LaserJet s rozhraním JetDirect obecně nabízejí dvě tiskové fronty – *raw*, která umí tisknout PCL (a případně PostScript), a *text*, jež zpracovává čisté ASCII (a umí se automaticky vyrovnat se „schodišťovým“ efektem). Pokud máte tříportový JetDirect Plus3, fronty se jmenují *raw1*, *text2* a tak dále.

Společnost ISS identifikovala nebezpečí DoS útoku, kterým je možno zablokovat rozhraní tiskáren HP JetDirect. Většina z nich byla popsána na podzim 98. Tento typ problémů je u podobných zařízení poměrně běžný, obecně by neměla být tato přístupná z běžného Internetu.

Ve velkých prostředích, zejména tam, kde některé tiskárny nepodporují přímo PostScript, je rozumné zřídit vyhrazený tiskový server, na nějž budou všechny počítače tisknout a na kterém poběží všechny ghostscriptové úlohy. Díky tomu bude možné s frontami manipulovat příkazy *topq* a *lprm*.

Navíc bude takovýto server udržovat fronty jednotlivých tiskáren, takže uživatelé „vytisknou“ úlohy okamžitě a budou moci pokračovat v práci, aniž by museli čekat, než skončí tisk jiných úloh jiných uživatelů. Doporučuje se to také, pokud máte starší JetDirect, snižuje se zaseknutí tiskárny.

Provedete to tak, že na tiskovém serveru vytvoříte frontu, která bude tisknout přímo na HP LJ se síťovým rozhraním, a všechny klienty v síti nastavíte tak, aby tiskli na LPD frontu na tomto serveru.

Některé síťové tiskárny HP zjevně ignorují nastavení úvodní stránky, které posílá klient. Tisk interně generované úvodní stránky můžete vypnout tak, že se na tiskárnu připojíte telnetem, dvakrát zmáčknete Enter, napíšete „banner: 0“ a „quit“. Tímto způsobem můžete měnit i jiná nastavení, jejich seznam získáte příkazem „?“.

Všechna nastavení je možné měnit programem *webJetAdmin* od HP, http://www.hp.com/go/webjetadmin. Tento balík běží jako démon a na zvoleném portu reaguje na HTTP požadavky. Nabízí formuláře a applety, kterými je možné řídit HP tiskárny v síti. Teoreticky může ovládat i unixové tiskové fronty, provádí to ale přes službu *rexec*, která není nijak zabezpečena. Použití této funkce vám rozhodně nedoporučuji.

Tisk na zařízení AppSocket

Některé tiskárny (a některé „tiskové servery“) podporují pouze ubohý pseudoprotokol, komunikující přímými TCP spojeními, někdy se označuje jako protokol AppSocket. Do této kategorie spadají zejména starší modely karet JetDirect (a některé JetDirectEx). V zásadě se tiskne tak, že musíte navázat TCP spojení s tiskárnou na určitém portu (typicky 9100, nebo 9100, 9101 a 9102 u tříportových karet) a po tomto spojení odeslat tiskovou úlohu. LPRng obsahuje podporu odeslání úlohy na libovolný port, u BSD LPD to není tak jednoduché. Asi nejlepší řešení je použít nástroj *netcat*.

PDQ rozhraní s použitím nástroje *netcat* bude vypadat takto nějak:

```
interface tcp-port-0.1 {

   help "Toto je jedno z prvních rozhraní podporovaných síťovými
         tiskárnami a tiskovými servery. Zařízení pouze očekává TCP
         spojení na jistý port a jím přijatá data tiskne.\n
         Toto rozhraní vyžaduje program netcat (\"nc\")."

   required_args "REMOTE_HOST"

   argument {
      var = "REMOTE_HOST"
      desc = "Vzdálené zařízení"
      help = "IP adresa nebo název tiskového serveru."
   }

   argument {
      var = "REMOTE_PORT"
      def_value = "9100"
      desc = "Vzdálený port"
      help = "Číslo TCP portu, na nějž se mají posílat úlohy. Většina
              karet JetDirect a jejich klonů používají port 9100
              (nebo 9101 pro druhý port, atd.)"
   }

   requires "nc"

   send_exec { cat $OUTPUT | nc $REMOTE_HOST $REMOTE_PORT }

}
```

Pokud by to nefungovalo, je možné tisk implementovat mimo jiné i pomocí Perlu programem, který uvádíme dále. Lepší výkon dosáhnete s programem *netcat* (nc), který dělá téměř to samé obecnějším postupem. Ve většině distribucí by měl být tento program k dispozici.

```
#!/usr/bin/perl
# Díky Danu McLaughlinovi za původní verzi tohoto skriptu a
# Jimu W. Jonesovi za to, že seděl vedle něj ;)

$fileName = @ARGV[0];

open(IN,"$fileName") || die "Nelze otevřít soubor $fileName";

$dpi300     = "\x1B*t300R";
$dosCr      = "\x1B&k3G";
$ends = "\x0A";

$port =  9100 unless $port;
$them = "bach.sr.hp.com" unless $them;

$AF_INET = 2;
$SOCK_STREAM = 1;
$SIG{'INT'} = 'dokill';
$sockaddr = 'S n a4 x8';

chop($hostname = `hostname`);
($name,$aliases,$proto) = getprotobyname('tcp');
($name,$aliases,$port) = getservbyname($port,'tcp')
    unless $port =~ /^\d+$/;;
($name,$aliases,$type,$len,$thisaddr) =
        gethostbyname($hostname);
($name,$aliases,$type,$len,$thataddr) = gethostbyname($them);
$this = pack($sockaddr, $AF_INET, 0, $thisaddr);
$that = pack($sockaddr, $AF_INET, $port, $thataddr);

if (socket(S, $AF_INET, $SOCK_STREAM, $proto)) {
#    print "socket ok\n";
}
else {
    die $!;
}
# Přiřazení adresy soketu.
if (bind(S, $this)) {
#    print "bind ok\n";
}
else {
    die $!;
}

# Volání serveru.

if (connect(S,$that)) {
#    print "connect ok\n";
}
else {
```

```
    die $!;
}

# Nastavení soketu na buffering.

select(S); $| = 1; select(STDOUT);

#    print S "@PJL ECHO Hi $hostname! $ends";
#    print S "@PJL OPMSG DISPLAY=\"Job $whoami\" $ends";
#    print S $dpi300;

# Forkneme se, aby nedošlo k deadlocku.

if($child = fork) {
    print S $dosCr;
    print S $TimesNewR;

    while (<IN>) {
        print S;
    }
    sleep 3;
    do dokill();
} else {
    while(<S>) {
        print;
    }
}

sub dokill {
    kill 9,$child if $child;
}
```

Spuštění if pro vzdálené tiskárny ve starém LPD

Jedna zvláštnost starších verzí *lpd* je ta, že pro vzdálené tiskárny nespouští rozhraní *if*. (Verze od 0.43 nebo tak nějak jsou upraveny na základě změn ve FreeBSD a *if* už se spouští vždy.) Pokud narazíte na to, že potřebujete pro vzdálenou tiskárnu spustit *if* a vaše verze *lpr* to neumí, můžete to vyřešit tak, že vytvoříte dvě fronty a budete úlohy předávat mezi nimi. Podívejme se na následující konfiguraci *printcap*:

```
lj-5:\
        :lp=/dev/null:sh:\
        :sd=/var/spool/lpd/lj-5:\
        :if=/usr/lib/lpd/filter-lj-5:
lj-5-remote:sh:rm=printer.name.com:\
        :rp=raw:sd=/var/spool/lpd/lj-5-raw:
```

A na skript *filter-lj-5*:

```
#!/bin/sh
gs <options> -q -dSAFER -sOutputFile=- - | \
        lpr -Plj-5-remote -U$5
```

Volba *-U* u *lpr* funguje pouze tehdy, je-li *lpr* spuštěn jako démon, a slouží k nastavení správného jména vlastníka úlohy v sekundární frontě. Možná byste měli použít nějakou spolehlivější metodu

získání uživatelského jména, protože ne ve všech případech je předáváno jako pátý parametr. Viz manuálové stránky *printcap*.

Tisk z Windows

Tisk z klientů Windows (a případně OS/2) je podporován přímo protokolem SMB z balíku Samba, který podporuje rovněž sdílení souborových systémů Unixu ve Windows.

Samba obsahuje vyčerpávající dokumentaci a rovněž dokument Samba FAQ, který pokrývá nejčastější způsoby použití. Můžete buď nakonfigurovat filtr na Unixu a tisknout na něj PostScript, nebo můžete na všech stanicích Windows nainstalovat ovladače příslušné tiskárny a tisknout přímo bez filtru. Použití nativních ovladačů Windows může v některých případech produkovat lepší výsledky, je to ale administrativně náročnější, zejména pokud je stanic mnoho. Zkuste tedy nejprve PostScript. Nové verze Samby podporují mechanismus automatického stažení ovladače, poskytovaný servery Windows NT.

Používáte-li PDQ, nakonfigurujte Sambu tak, aby spouštěla příkaz *pdq* se správnými parametry namísto příkazu *lpr*, který spouští standardně. Předpokládám, že Samba spustí *pdq* jako správný uživatel, takže by tak mělo všechno správně fungovat. Samba nabízí několik parametrů, kterými můžete tisk pomocí PDQ upravit:

printcap
: Tento parametr by měl ukazovat na „falešný“ *printcap*, obsahující seznam nabízených tiskáren. Potřebujete pouze uvést krátké a dlouhé jméno každé tiskárny, vždy jednu tiskárnu na řádek:

```
lp1|Printer One
lp2|Printer Two
lp3|Printer Three
```

Krátké jméno se použije jako název tiskárny v příkazu *print*.

příkaz *print*
: Musíte jej nastavit, něco jako *pdq -P %p %s ; rm %s*.

příkaz *lprm*
: Momentálně není žádná rozumná alternativa, kterou zde nastavíte. PDQ úlohy ve frontě po čase vyřazuje, takže pokud tiskárnu úplně zrušíte, není to problém. Pokud se prostě jen rozhodnete a nechcete úlohu vytisknout, můžete ji zrušit pomocí *xpdq*, z Windows to ale jde obtížně. Prostě zde zadejte nějaký příkaz, který nebude dělat nic, například *true*. Pokud jako back-end používáte *lpd* nebo LPRng, měl by příslušný příkaz *lprm* fungovat. Nevím ale, jak Samba identifikuje číslo úlohy ve frontě, když úlohu zadává *pdq*.

příkaz *lpq*
: Ani zde nenabízí PDQ rozumnou variantu. Distribuované systémy nenabízejí rozumný způsob prohlížení fronty, nicméně centralizované systémy založené na Sambě mohou zajistit prohlížení fronty. Prostě zde zadejte nějaký příkaz, který nebude dělat nic, například *true*. Pokud jako back-end používáte *lpd* nebo LPRng, měl by příslušný příkaz *lpq* fungovat, pouze úlohu neuvidíte do té doby, než skončí její zpracování v PDQ.

Tisk z Apple

Netatalk podporuje tisk klientů Apple protokolem EtherTalk. Další informace najdete v dokumentu Netatalk HOWTO, http://thehamptons.com/anders/netatalk/.

Modernější verze Macu dokáží tisknout i přes TCP/IP protokolem LPD. UV nabízí velmi pěknou stránku http://www.itc.virginia.edu/desktop/mac/ip_printing/ip_printing.html, kde se dozvíte podrobnosti o nastavení.

Tisk z NetWare

Balík *ncpfs* obsahuje démona *pserver*, který umí poskytovat službu tiskového serveru frontám NetWare. Pokud vím, tento systém vyžaduje NetWare založený na Bindery, tedy NetWare 2.x, 3.x, nebo novější s emulací Bindery.

Další informace o *ncpfs* a programu *pserver* najdete na stránkách ftp://ftp.gwdg.de/pub/linux/misc/ncpfs/.

Administrace síťových tiskáren

Většina síťových tiskáren podporuje nějakou metodu vzdálené správy. Často jde o snadno použitelné webové rozhraní. Ještě užitečnější je podpora správy přes SNMP. Typicky tak můžete zjistit zajímavé informace o stavu tiskárny, jako jsou množství toneru a papíru, o zatížení tiskárny, a můžete také měnit některá nastavení. Ovládání tiskáren přes SNMP a řada dalších věcí týkajících se tisku je standardizována skupinou Printer Working Group pod IEEE, http://www.pwg.org/.

npadmin

npadmin je řádkový program poskytující rozhraní k běžným SNMP funkcím síťových tiskáren. Implementuje standard Printer MIB (http://www.ietf.org/rfc/rfc1759.txt) a některá proprietární řešení výrobců, která se týkají zejména starších zařízení. Podporovány jsou jak funkce printer-discovery, tak i různé stavové dotazy na tiskárny.

npadmin má vynikající manuálovou stránku a existuje i v podobách RPM a dpkg pro distribuce, které tento typ balíčků používají.

Další SNMP nástroje

Kromě *npadmin* existuje i několik dalších užitečných SNMP nástrojů. *snmptraplogd* může logovat události SNMP trapů. Je to užitečné pro sledování zaseknutí papíru, vyčerpání papíru a podobně, jednoduché je také přesměrování konkrétních událostí na e-mail a podobně.

I když *npadmin* nabízí jednoduchou podporu SNMP rozhraní většiny síťových tiskáren, některé tiskárny používají i různá proprietární rozšíření, která *npadmin* nezná. V takovém případě můžete použít nástroje CMU SNMP, které podporují libovolné operace SNMP GET a SET. S trochou práce tak budete moci využít všechny funkce, které SNMP vaší tiskárny nabízí. Budete potřebovat specifikaci výrobce, abyste zjistili, co všechny proměnné znamenají – výrobci často předpokládají, že uživatelé používají výhradně jimi dodávané proprietární nástroje (určené typicky jen pro Windows).

Knihovna *libprinterconf* VA Linuxu nabízí funkci detekce síťových tiskáren. Identifikace tiskáren probíhá podle vestavěné databáze signatur tiskáren. Tato databáze není momentálně příliš rozsáhlá, pokrývá ale většinu běžných modelů.

Winprinters

Jak už bylo řečeno dříve, některé tiskárny jsou ze své podstaty nepodporované, protože neznají žádný „normální“ komunikační jazyk, namísto toho využívají procesoru počítače k vykreslení tiskového rastru, který se pak fixní rychlostí posílá na tiskárny. V některých případech tyto tiskárny umějí i něco normálního jako PCL, ale často ani to ne. U některých (opravdu „low-end“ modelů) tyto tiskárny dokonce ani nepoužívají normální paralelní komunikaci, ale spoléhají na to, že ovladač emuluje to, co má řešit hardware (například řízení komunikace).

Každopádně, pokud na nějakou takovou hrůzu narazíte, existuje několik možných řešení.

Redirektor Ghostscriptu

V současné době existuje tiskový ovladač pro Ghostscript (jmenuje se *mswinpr2*), který tiskne pomocí GDI volání Windows. Existuje také nástroj pro přesměrování portu, *redmon*, který umožňuje úlohu před konečným tiskem prohnat přes Ghostscript (podobně jako to dělá filtr *if* v LPD v Unixu). Díky tomu mohou Windows tisknout PostScript s použitím originálního ovladače.

Pokud na tiskárně nemůžete tisknout přímo, můžete ji exportovat jako postscriptovou tiskárnu pomocí programů *redmon*, *Ghostscript* a *mswinpr2* ve Windows a tisknout pomocí ovladače od výrobce.

HP Winprinters

Některé tiskárny HP používají „Printing Performance Architecture“ (což je marketingový výraz pro „ta tiskárna je moc levná, než aby uměla PCL“). Tato metoda je podporována překladačem *pbm2ppa* Tima Normana. V zásadě použijete Ghostscript k vykreslení PostScriptu do rastrového obrázku ve formátu *pbm*, a pak použijete *pbm2ppa* ke konverzi tohoto rastru do interního rastru tiskárny, který je možné vytisknout. V současné době už by měl být tento program k dispozici i jako ovladač pro Ghostscript.

Tento program můžete získat na adrese http://www.rpi.edu/~normat/technical/ppa/, *pbm2ppa* podporuje některé modely HP 720, 820 a 1000. Podrobnosti se dozvíte v dokumentaci k tomuto balíku.

Lexmark Winprinters

Většina levných inkoustových tiskáren Lexmark používá proprietární jazyk a jsou to tedy klasické Winprinters. Nicméně Henryk Paluch napsal program, který umí tisknout na tiskárně Lexmark 7000. Možná bude schopen rozšířit podporu i o barevný tisk a další tiskárny Lexmark. Další informace najdete na adrese http://bimbo.fjfi.cvut.cz/~paluch/l7kdriver/.

Podobně existují i ovladače pro modely 5700, 1000, 1100, 2070, 3200 a další. Podívejte se do databáze podporovaných tiskáren, kde najdete i odkazy na příslušné ovladače.

Jak tisknout na fax

Na fax můžete tisknout s pomocí nebo bez pomoci modemu.

Použití faxmodemu

Existuje řada faxovacích programů, které umožňují faxovat a přijímat dokumenty. Jedním z nejlepších je HylaFAX Sama Lefflera, http://www.hylafax.org/. Podporuje všechno možné, počínaje více modemy až po broadcasting.

SuSE obsahuje klienta HylaFAX napsaného v Javě, který údajně pracuje na jakékoliv platformě s podporou Javy (včetně Windows a Linuxu). Kromě toho pro většinu platforem existují i faxoví klienti nepoužívající Javu, v Linuxu budete téměř jisté nastavit faxování podle vašich potřeb.

Další možností, vhodnější pro malé instalace, je *efax*, http://casas.ee.ubc.ca/efax/, program, který odesílá a přijímá faxy. Program *mgetty* umí spolupracovat s *efax*em a přijímat faxy (a také hlasovou poštu a interaktivní přihlášení).

Faxování z PDQ

PDQ neobsahuje rozhraní pro fax, jednoduchou a pouze částečně otestovanou variantu uvádím zde:

```
interface efax-0.1 {
   help "Toto rozhraní používá program fax z balíku efax k odeslání
         faxu. Nejprve upravte /etc/efax.rc a zkuste, zda funguje
         příkaz \"fax send\". Toto rozhraní můžete připojit k obecnému
         postscriptovému ovladači a definujte fax jako tiskárnu."

   requires { "efax" "fax" }

   # Protože potřebujeme telefonní číslo, průvodce přidáním tiskárny
   # se na ně bude ptát při instalaci. To není praktické, i když logicky
   # je číslo vyžadováno, a proto není jeho zadání povinné. Skript
   # send_exec testuje zadání čísla. Průvodce můžete přeskočit přidáním
   # tiskárny ručně do .printrc, označte číslo jako povinné, a skript
   # se pak na ně bude ptát až při tisku.
   argument {
      var = "PHONE_NUMBER"
      desc = "Phone Number"
      help = "Volané číslo. Prefixy jako "9" je vhodné definovat v
              souboru /etc/efax.rc."
   }

   option {
      var = "RESOLUTION"
      desc = "Fax resolution"
      default_choice = "high"
      choice "low" {
         value = "-l"
         desc = "Low"
         help = "Nízké rozlišení u faxu je 96lpi."
      }
      choice "high" {
         value = ""
         desc = "High"
         help = "Vysoké rozlišení u faxu je 192lpi."
      }
   }

   # Pokud číslo nezadáte, úloha se nezpracuje a zjistíte to jedině tak
   # že se podíváte na chybové hlášení na konci podrobných informací
   # o úloze. Hmm.
   send_exec {
     if [ "x$PHONE_NUMBER" != "x" ]
     then
          fax send $RESOLUTION $PHONE_NUMBER $INPUT
     else
          echo 'Musíte zadat číslo!'
          false
     fi
   }
}
```

Služba Remote Printing Service

Existuje experimentální služba, kam můžete poslat e-mail, obsahující to, co chcete někomu poslat na fax. Podporován je i PostScript, takže i přesto, že služba není dostupná všude, může být velice užitečná. Další informace o této službě najdete na adrese http://www.tpc.int/.

Komerční faxové služby

Řada společností nabízí faxové služby přes Internet. Například EFax, http://www.efax.com/, nabízí zdarma příjem faxů přes e-mail a odesílání faxů za poplatek. Podobné služby nabízejí i další společnosti.

Jak vytvořit něco, co stojí za vytisknutí

Teď už se dostáváme k poslednímu kroku v softwarovém řetězci. V zásadě Linux umí spustit řadu typů binárních souborů s různým stupněm úspěšnosti: Linux/x86, Linux/Alpha, Linux/Sparc, Linux/foo, iBCS, Win16/Win32s (pomocí dosemu a Wine), Mac/68k (pomocí Executor) a Java. Budeme hovořit pouze o nativním GNU/Linuxu a běžném unixovém softwaru.

Mark-up jazyky

Většina mark-up jazyků se hodí zejména pro větší nebo opakující se projekty, kdy počítač řídí vzhled textu tak, aby vypadal jednotně.

nroff Jde o jeden z prvních mark-up jazyků na původní verzi Unixu. Nejznámějším příkladem věcí, formátovaných pomocí maker *roff jsou manuálové stránky. Řada lidí na ně nedá dopustit, nicméně alespoň podle mě má nroff příliš záhadnou syntaxi (viz obrázek 11) a v současné době nepředstavuje nejlepší volbu. Je ale dobré vědět, že manuálovou stránku můžete vysázet přímo v PostScriptu pomocí programu *groff*. Většina příkazů *man* to umí sama příkazem *man -t foo | lpr*.

Obrázek 9 – Příklad vstupu pro roff

```
.B man
is the system's manual pager. Each
.I page
argument given to
.B man
is normally the name of a program, utility or function.
The
.I manual page
associated with each of these arguments is then found and
displayed. A
.IR section ,
if provided, will direct
.B man
to look
only in that
.I section
of the manual.
```

TeX TeX a makrobalík LaTeX představují jeden z nejrozšířenějších mark-up jazyků na unixových systémech, i když TeX nepochází původně z Unixu a je k dispozici na řadě různých systémů. V LaTeXu se velmi často vytvářejí technické dokumenty, protože je velmi zjednodušena problematika sazby a LaTeX je navíc jeden z mála systémů, který podporuje sazbu matematických vzorců jak kompletně, tak i dobře. Výstupem TeXu je formát *dvi*, který je možné programy *dvips* a *dvilj* konvertovat na PostScript nebo PCL. Pokud budete chtít nainstalovat TeX nebo LaTeX, nainstalujte celou skupinu balíků teTeX, která obsahuje všechno. Nové verze TeXu obsahují i pdfTeX a pdfLaTeX, které vytvářejí přímo soubory *pdf*. K dispozici jsou i příkazy pro vkládání odkazů a navigačních funkcí do PDF dokumentů.

Obrázek 10 – Příklad vstupu pro LaTeX

```
\subsubsection{NAT}

  Each real server is assigned a different IP address, and the NA
  implements address translation for all inbound and outbound
  packets.

  \begin{description}
  \item[Advantage] Implementation simplicity, especially if we
        already implement other NAT capabilities.

  \item[Disadvantage] Return traffic from the server goes through
        address translation, which may incur a speed penalty.  This
        probably isn't too bad if we design for it from the
        beginning.

  \item[Disadvantage] NAT breaks the end-to-end semantics of normal
        internet traffic.  Protocols like ftp, H.323, etc would
        require special support involving snooping and in-stream
        rewriting, or complete protocol proxying; neither is likely
        to be practical.
  \end{description}
```

SGML Pro unixové systémy existuje přinejmenším jeden volně šířený parser SGML, díky němuž vznikl dokumentační systém Linuxdoc, založený na SGML. Může podporovat i jiné DTD, zejména DocBook. Tento dokument byl napsán v DocBook-DTD SGML, příklad vidíte na obrázku 11.

Obrázek 11 – Příklad DocBook SGML

```
<VarListEntry>
 <Term>SGML</Term>
 <ListItem>
  <Para>
   There is at least one free SGML parser available for Un*x
   systems; it forms the basis of Linuxdoc-SGML's homegrown
   document system.  It can support other DTD's, as well, most
   notably DocBook.  This document is written in DocBook-DTD
   SGML.
  </Para>
 </ListItem>
</VarListEntry>
```

Textové procesory WYSIWYG

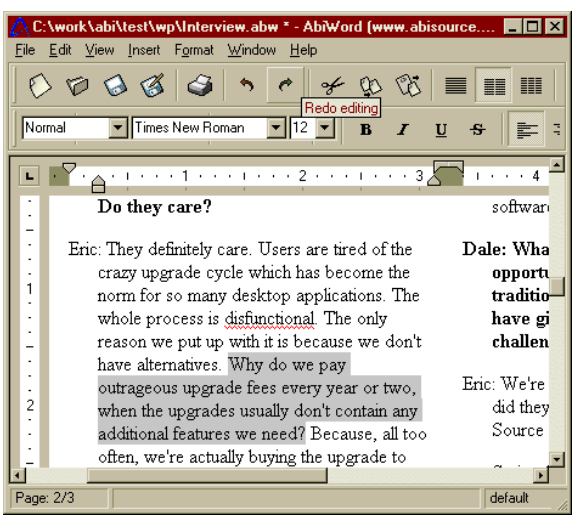
Programů pro WYSIWYG práci s textem je dostatek. Existují úplné kancelářské balíky, jeden z nich je pro osobní potřebu zdarma (StarOffice).

StarOffice — Sun Microsystems distribuoval StarOffice pro Linux zdarma (nyní již pouze Open Office.org). Tento kompletní kancelářský balík obsahuje všechny funkce, které byste očekávali, včetně importu a exportu dokumentů ve formátu Microsoft Office (tedy Word a podobně). Existuje mini-HOWTO dokument popisující získání a instalaci tohoto balíku. Tiskovým výstupem je PostScript, takže by měl fungovat s jakoukoliv tiskárnou, která v Linuxu funguje.

WordPerfect — Corel distribuuje základní verzi WordPerfectu 8 pro Linux zdarma a prodává různé balíky Word Perfect Office 2000 (obsahující WordPerfect, Corel Draw a Quatro Pro verze 9). Stránka *Linux WordPerfect Fonts and Printers*, http://www.rodsbooks.com/wpfonts/, obsahuje informace o konfiguraci WordPerfectu pro použití s Ghostscriptem nebo s originálními ovladači WordPerfectu (které jsou stejné jako ovladače pro DOS pro případ, že by ovladač vaší tiskárny v distribuci nebyl).

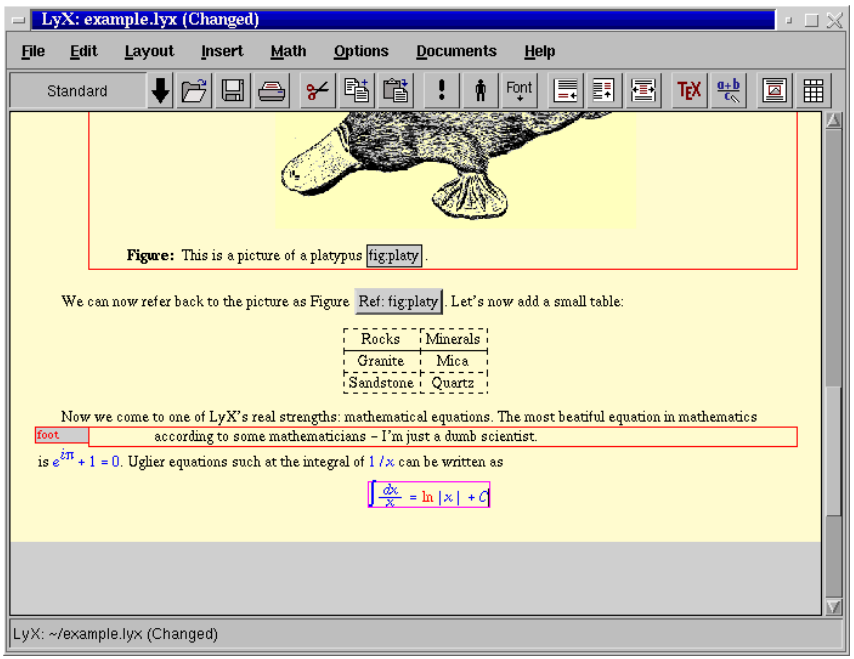
Applix — Applix je multiplatformní (Unixy, Windows a další) kancelářský balík prodávaný společností Applix. Red Hat a SuSE jej prodávaly rovněž v době, kdy neexistovala jiná varianta, nyní jej prodává už zase jen Applix. Jedná se o jediný aplikační balík v nativním unixovém stylu, pravděpodobně se vám bude líbit jeho unixový přístup k věci.

AbiWord — AbiWord, http://www.abisource.com/, je jeden z několika GPL WYSIWYG editorů. Jde o velmi pěkný editor založený na formátu XML. Podporuje také import souborů z Wordu. Vývoj AbiWordu stále pokračuje, už dnes je ale použitelný.



Obrázek 12 – AbiWord

LyX — LyX je rozhraní k LaTeXu, které vypadá velmi slibně. Další informace najdete na adrese http://www.lyx.org/. Existuje i KDE verze LyXu, pojmenovaná Klyx.



Obrázek 13 – LyX

Maxwell Maxwell je jednoduchý textový editor založený na formátu MS RTF, který začal jako komerční produkt a nyní se distribuuje pod GPL.

Uvítám informace o dalších produktech.

Řešení jednotlivých distribucí

Abyste z normálních tiskáren získali slušné fotografie, je nutné uhlídat celou řadu detailů. Pokud ještě nemáte fototiskárnu, podívejte se na tipy v kapitole 5.4.

Ghostscript a fotografie

Ghostscript má potíže s tiskem barevných fotografií u většiny ovladačů. Problémy spočívají v několika věcech:

- Řada ovladačů má špatně vyladěnou podporu barev. Barvy často neodpovídají podobě na obrazovce a výtisku ve Windows. Na druhé straně, všechny ovladače i samotný Ghostscript mají nastavitelnou barevnou podporu. Jednou z možností je nastavení gama korekce (viz kapitola 10.2.2), další jsou popsány v dokumentačním souboru Use.htm Ghostscriptu.
- Vím pouze o jediném ovladači pro Ghostscript, který podporuje šesti a sedmibarevný tisk, momentálně je v beta-verzi a podporuje modely Epson Stylus Photo. Tvrdí se, že barevný výstup je lepší než u originálního ovladače pro Windows. Jádro Ghostscriptu nepodporuje jiné barevné modely než CMYK a RGB, je nutné tuto podporu dopracovat.
- Ghostscript často generuje nehezký dithering a vytváří různé chyby, například proužky. Dithering lze obvykle opravit, viz kapitola 10.2.3 a dokumentace k příslušnému ovladači.

Některé z těchto problémů lze upravit doladěním Ghostscriptu. Podrobnosti najdete v kapitole 10. Nastavování parametrů Ghostscriptu se výrazně zjednoduší, pokud je deklarujete jako parametry tiskového systému.

Momentálně je tedy lepší fotografie tisknout jiným systémem než Ghostscriptem, takové programy samozřejmě existují. Nejvýznamnějším je tiskový doplněk pro GIMP, který podporuje tisk na Epson Stylus a na postscriptových tiskárnách (s podporou PPD). Část podporující Epson Stylus existuje i pro Ghostscript jako ovladač stp. Další variantou jsou externí programy pnm-to-něco, podporující tisk na tiskárnách jako je Lexmark – tyto programy provádějí tisk mapování pixel-na-pixel.

Nejlepší řešení je samozřejmě koupě PostScriptové tiskárny, ty lze obvykle plně ovládat dostupnými programy a při tisku využít všech možností tiskárny.

Papír

U barevných inkoustových tiskáren závisí kvalita tisku významně na použitém papíru. Drahé lesklé papíry umožňují tisknout v téměř fotografické kvalitě, výstup na klasický kancelářský papír dává obvykle ušmudlané barvy a rozplizlé detaily. Speciální matné papíry do inkoustových tiskáren dávají výsledek někde uprostřed a hodí se zejména pro tisk finálních podob dokumentů. Tvrdé lesklé fotografické papíry dávají stejný výsledek jako „normální“ lesklé fotopapíry, výsledek však více připomíná klasickou fotografii.

Nastavení tiskárny

Při tisku fotografií byste na většině barevných inkoustových tiskáren měli použít nejkvalitnější (a nejpomalejší) režim tisku, jinak budou větší plochy pruhované nebo barevně nejasné. V Ghostscriptu obvykle stačí zvolit nejvyšší rozlišení. U postscriptových tiskáren budete možná muset před samotnou úlohou tiskárnu nastavit nějakým příkazem podle PPD souboru. PPD podpora v GIMPu neobsahuje nastavení kvality tisku (rozuměj specifické nastavení konkrétní tiskárny), nicméně pro vlastní potřebu jsem si tuto podporu doplnil – pokud chcete, kontaktujte mne. Pokud používáte PDQ nebo CUPS, budete moci snadno nastavovat všechny parametry tiskárny. U postscriptových tiskáren nabízí tato nastavení i *libppd* VA Linuxu a GPR.

Trvanlivost tisku

Barevné výtisky z inkoustových tiskáren obvykle za pár let vyblednou, zejména pokud jsou vystaveny slunečnímu světlu – to je vlastnost inkoustu. Tiskárny s výměnnými inkoustovými náplněmi, jako Epson nebo Canon, mohou pracovat i s takzvaným archivním inkoustem, který je na blednutí méně náchylný. Novější tiskárny obvykle používají pigmentové inkousty, které neblednou tak rychle jako starší barvené inkousty. Nicméně pro delší skladování se inkoustové výtisky obecně nehodí – vypalte si obrázky na CD a uskladněte je takto.

Komerční programy

Existuje program *xwtools*, http://home.t-online.de/home/jj.sarton/startE.htm, který umožňuje tisk fotografií se všemi parádičkami na vybraných tiskárnách Epson, HP a Canon. Bohužel byl napsán pod NDA, takže nejsou k dispozici jeho zdrojové kódy. Pokud nepoužíváte Epson Stylus Color 300 na Linuxu, stojí pro osobní použití 15 dolarů, ceny pro komerční použití nejsou známé.

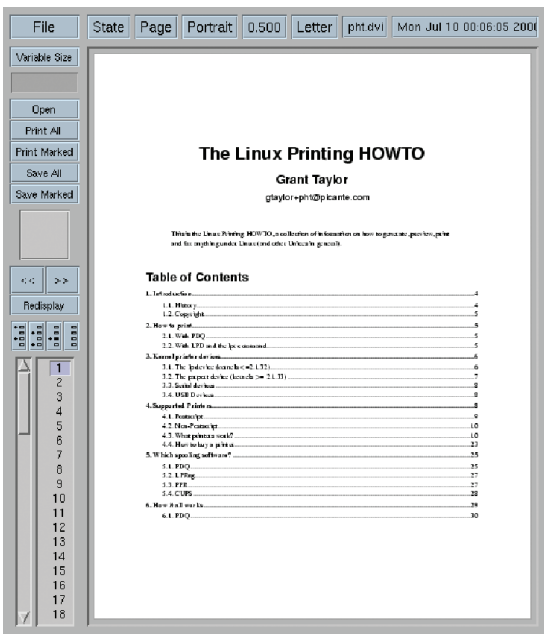
Balík ESP Print Pro společnosti Easy Software podporuje některé jinak nepodporované tiskárny. Tyto ovladače nejsou ideální pro tisk fotografií, ale aspoň fungují.

Náhled před tiskem

Téměř vše, co jde vytisknout, si můžete také prohlédnout na obrazovce.

PostScript

Ghostscript obsahuje ovladač X11, který se nejlépe používá s prohlížečem PostScriptu *gv*. Nové verze tohoto programu umí zobrazit i soubory PDF. Program *gv* je náhradou staršího prohlížeče Ghostview, nové rozhraní je mnohem pěknější a chytřejší než původní.



Obrázek 14 – Gv

Tex dvi

Soubory DeVice Independent TeXu si můžete v X11 prohlížet pomocí *xdvi*, http://www.linuxprinting.org/man/xdvi.1.html. Moderní verze *xdvi* volají pro vykreslení specialit PostScriptu Ghostscript.

Existuje také ovladač pro VT100, jmenuje se *dgvt*. Prohlížeč *tmview* používá knihovnu *svgalib*, pokud nemáte nic lepšího k dispozici.

Adobe PDF

Adobe Acrobat Reader existuje i pro Linux, můžete si jej stáhnout na http://www.adobe.com/.

Kromě toho můžete použít *xpdf* a také *gv* podporuje PDF soubory.

Sériové tiskárny a LPD

Použití sériových tiskáren s *lpd* je trochu složité.

Nastavení v *printcap*

LPD nabízí pět parametrů, kterými můžete v */etc/printcap* nastavit všechny vlastnosti sériového portu a připojení tiskárny. Přečtěte si manuálovou stránku programu *printcap* a popis parametrů *br#*, *fc#*, *xc#*, *fs#* a *xs#*. Poslední čtyři představují bitové mapy sloužící k nastavení portu, parametr *br#* nastavuje přenosovou rychlost, například *br#9600*.

Velmi jednoduchý je převod nastavení *stty* na příznaky pro *printcap*. Podívejte se na manuálovou stránku *stty*.

Pomocí *stty* nastavte port tiskárny tak, abyste byli schopni příkazem *cat* na tiskárnu poslat a vytisknout soubor. Takto vypadá příkaz *stty -a* pro můj port tiskárny:

```
dina:/usr/users/andy/work/lpd/lpd# stty -a < /dev/ttyS2
speed 9600 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr
-igncr -icrnl ixon -ixoff -iuclc -ixany -imaxbel
-opost -olcuc -ocrnl -onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0
bs0 vt0 ff0
-isig -icanon -iexten -echo -echoe -echok -echonl -noflsh -xcase
-tostop -echoprt -echoctl -echoke
```

Jediné rozdíly tohoto nastavení proti nastavení při startu systému jsou příznaky *-clocal*, *-crtscts* a *ixon*. Možná budete potřebovat úplně jiná nastavení v závislosti na tom, jaké používá tiskárna řízení přenosu.

Ve skutečnosti musíte *stty* použít poněkud zvláštním způsobem. Protože *ssty* pracuje s terminálem připojeným na standardní vstup, manipulujete se sériovým portem pomocí znaku „<“ tak, jak to vidíte výše.

Jakmile máte *stty* nastaveno správně, takže příkaz *cat file > /dev/ttyS2* (v mém případě) pošle soubor na tiskárnu, podívejte se do souboru */usr/src/linux/include/asm-i386/termbits.h*. Tento soubor obsahuje řadu příkazu *#define* a několik struktur. (Případně můžete tento soubor poslat na tiskárnu (to vám funguje, že?) a použít jej jako referenci.) Najděte si část, která začíná:

```
/* c_cflag bit meaning */
#define CBAUD   0000017
```

V této části najdete význam bitů pro parametry *fc#* a *fs#*. Všimněte si, že názvy v tomto souboru odpovídají názvům, které se objevují ve výstupu příkazu *stty*. Neříkal jsem, že to bude jednoduché?

Všimněte si, která nastavení jsou ve výstupu příkazu *stty* uvedena znakem -. Sečtěte hodnoty u všech těchto parametrů (hodnoty jsou v osmičkové soustavě). Tím dostáváte bity, které musíte vynulovat a tato hodnota slouží jako argument parametru *fc#*. Nicméně hned vzápětí budeme nastavovat potřebné bity, takže nejjednodušší je prostě nejprve vynulovat všechno, což můžete provést volbou *fc#0177777*.

Teď sečtěte hodnoty u parametrů, u nichž není uveden na začátku výpisu příkazu *stty* znak -. V mém případě jde o parametry CS8 (0000060), HUPCL (0002000) a CREAD (0000200). Dále nezapomeňte na příznaky nastavující přenosovou rychlost (u mne je to 0000015). Všechno sečtěte, v mém případě dostanete hodnotu 0002275. To je argument pro parametr *fs#* (v mém případě stačí zadat *fs#02275*).

Stejný postup s nulováním a nastavováním bitů proveďte i s další částí souboru, *c_lflag*. V mém případě se nenastavuje nic, takže používám parametry *xc#0157777* a *xs#0*.

Starší sériové tiskárny vynechávající znaky

Jon Luckey upozornil, že některé starší sériové tiskárny s levným sériovým rozhraním a malou vyrovnávací pamětí opravdu myslí STOP, když to v řízení přenosu signalizují. Zjistil, že pokud příkazem *setserial* vypnul frontu u sériového portu s čipem 16 550, vyřešil se problém s vynechávání znaků. (Stačí pouze označit typ *uart* jako *8 250.*)

Co chybí?

Ještě stále chybí řada částí do úplného tiskového systému. Existují projekty, které se je snaží doplnit, ovšem většina z nich doposud nic rozumného nevyřešila a snahy o standardizaci potřebných protokolů a API jsou stále v plenkách.

Propojení

Existuje obecný problém donutit všechny části, aby spolu navzájem komunikovaly, zejména mechanismem nezávislým na použitém spooleru. Tento problém se nejzřetelněji projevuje u dojemné snahy aplikací řídit všechny „obvyklé“ tiskové funkce. Autor aplikace prostě nemá možnost zjistit informace o tiskárně, úlohách a podobně, neexistuje standardizovaná metoda vytvoření úlohy, žádný rozumný způsob, jak se dozvědět stav úlohy, dokonce ani žádný standardní způsob generování tiskových dat (i když většina moderních systémů nabízí nějaká specifická řešení).

Snahy o vytvoření rozumného tiskového aplikačního rozhraní se soustřeďují kolem knihovny *sysAPS* společnosti Corel, která poskytuje základní implementace některých funkcí pro práci s frontami a tiskárnami.

Fonty

Obsluha fontů je hrozně nešikovná. Obrazovka, tiskárna, aplikace a datové soubory by měly mít v ideálním případě přístup ke stejným fontům. To však, bohužel, není pravda. Existují plány na odstranění manipulace s fonty z X serveru, čímž by se vyřešila část problémů, stále však zůstává problém s dobrým mapováním tiskových fontů na aplikační fonty. Zatím není žádný projekt, který by se to snažil řešil, v současné době aplikace prostě do tiskových dat vkládají své vlastní fonty.

Metadata

Aplikace a spoolery se potřebují nějakým způsobem dozvědět vlastnosti tiskárny a ovladače. Současné standardizované schéma, implementované ve Windows, Macu a CUPSu používá soubory Postscript Printer Definiton, PPD, na jejichž základě buduje programové i uživatelské rozhraní. Problém ovšem logicky vzniká s tiskárnami, které PostScript nepodporují, proto IEEE Printer Working Group pracuje na projektu vytvoření souborů Universal Printer Driver Format, UPDF. Zatím se jim podařilo vytvořit příklad ve formátu XML. Tento příklad hodně připomíná PPD soubory a chybí v něm jakékoliv údaje závislé na ovladači nebo platformě, takže formát UPDF doposud není použitelný. IBM používá v OS/2 plně parametrizovanou architekturu ovladačů, která je volně šiřitelná. Může být použita jako užitečný zdroj nápadů a kódu, případně jako dostatečně dobrý systém pro okamžité použití. Ani tento systém však nenabízí žádný definovaný mechanismus pro předání důležitých parametrů od ovladače k aplikaci. Očekává se, že by měl být vyvinut XML formát a/nebo API, které umožní zjistit nastavitelné parametry.

Ovladače

Stav na poli volně šířených ovladačů je poměrně bídný. Existuje naštěstí několik projektů, které se to snaží řešit, a u tiskáren využívajících jejich kód lze dosáhnout velmi dobrých výsledků. Snahou je nabídnout jak dobré ovladače, tak i dobrou základnu nabízející ke sdílení často duplikované (a obtížné) části ovladačů – například dithering.

V úsilí o dosažení tohoto cíle je nezbytná spolupráce výrobců tiskáren. V současné době výrobci neposkytují ani ty základní informace potřebné k ovládání jejich zařízení. Na setkání Printing Summit 2000 byla přítomna i řada výrobců a bylo dosaženo jakési malé dohody. Výrobci se snaží utajit algoritmy ditheringu a podobné, protože tyto softwarové komponenty mají na svědomí vysoce kvalitní tiskové výstupy, a jednotliví výrobci si navzájem samozřejmě konkurují. Výrobci přítomní na setkání mají nyní jasnější představu o fungování free softwarové komunity a o tom, co se po nich chce. Není to mnoho, ale je to základ pro budoucí pokrok.

Fonty v Linuxu

Originál: http://tldp.org/HOWTO/Font-HOWTO/

Souhrnný zdroj informací, sloužící jako počáteční místo řešení všech otázek ohledně fontů v Linuxu.

Úvod

Stručně řečeno – smyslem tohoto dokumentu je pokrýt díru v dokumentaci týkající se fontů. Existuje několik různých dokumentů ohledně fontů v Linuxu, žádný z nich však není souhrnný – všechny se soustřeďují vždy na konkrétní problém. Cílem tohoto dokumentu tedy není přinést radikální pohled na problematiku práce s fonty (i když zde najdete některé informace, jež jinde nejsou), cílem je hlavně poskytnout souhrnný zdroj, sloužící jako počáteční místo řešení všech otázek ohledně fontů v Linuxu.

Rychlý úvod do problematiky fontů

Typy fontů

Rastrové fonty

Rastr (bitmapa) je matice bodů. Rastrové fonty jsou reprezentovány přesně tímto způsobem – jako matice bodů. Proto jsou *závislé na zařízení* – správně fungují jen v určitém rozlišení. Font určený pro obrazovku s rozlišením 75 DPI bude mít stále 75 DPI i na tiskárně s rozlišením 1 200 DPI.

Existují dva typy rastrových fontů – tiskové fonty, například fonty *pk* generované programem *dvips*, a obrazovkové fonty, používané v systému X Window a na konzole. Soubory s obrazovkovými fonty mají typicky přípony *bdf.gz* nebo *pcf.gz*. Obrazovkové rastrové fonty mají největší význam pro terminálová okna, konzoly a textové editory, kde nevadí nemožnost změnit jejich velikost a nemožnost je rozumně vytisknout.

Fonty Type 1

Standard Type 1 byl navržen společností Adobe a podporuje je standard PostScript společnosti Adobe. Díky tomu jsou dobře podporovány i v Linuxu. Podporují je systémy X Window a ghostscript. Postscriptové fonty tradičně představují v UNIXech správnou volbu ohledně čehokoliv, co zahrnuje tisk.

Typicky se fonty Type 1 v UNIXu distribuují jako soubory *afm* (adobe font metric) a obrysové (outline) soubory, obvykle *pfb* (printer font binary) nebo *pfa* (printer font ascii). Obrysový soubor obsahuje definice tvarů, metrikové soubory definují metriky.

Fonty Type 1 pro jiné platformy mohou být distribuovány v jiných formátech. Například postscriptové fonty pro Windows často používají pro definice metrik odlišný souborový formát *pfm*.

Fonty Type 3

Tyto fonty se distribuují obdobně jako fonty Type 1 – jako dvojice *afm* souboru s metrikami a *pfa* souboru. I když jsou podporovány standardem PostScript, nepodporuje je systém X Window a jejich použití je tedy omezené.

Fonty TrueType

Fonty TrueType vyvinula společnost Apple. Svůj formát zpřístupnila společnosti Microsoft a úspěšně tak zaútočila na nadvládu společnosti Adobe na trhu s fonty. Fonty TrueType ukládají metriky i definice tvarů v jediném souboru (typicky s příponou *ttf*). Nedávno byly vyvinuty font servery, které umožňují v systému X Window práci s fonty TrueType. Již delší čas tyto fonty podporuje ghostscript a PostScript. Díky tomu popularita TrueType fontů v Linuxu roste.

Fonty Type 42

Fonty Type 42 jsou ve skutečnosti normální fonty TrueType s hlavičkami, které umožňují jejich vykreslení interpretem PostScriptu. Řada aplikací, (například ghostscript) s těmito fonty transparentně pracuje. Pokud ale máte postscriptovou tiskárnu, budete možná muset tyto fonty explicitně vytvořit.

Type 1 versus TrueType – srovnání

Navzdory historickým sporům mezi producenty fontů Type 1 a TrueType mají oba formáty hodně společného. Obojí jsou škálovatelné obrysové fonty. Fonty Type 1 definují tvary znaků pomocí kubických křivek, na rozdíl od fontů TrueType, které používají kvadratické křivky. Teoreticky je to výhoda, protože křivky fontů Type 1 tak představují nadmnožinu křivek fontů TrueType. V praxi je rozdíl zanedbatelný.

Fonty TrueType mají jednoznačnou výhodu v lepší podpoře hintingu[1] (i fonty Type 1 nabízejí možnost hintingu, ale není tak dobrá jako u fontů TrueType). Tato funkce má ale význam pouze u zařízení s malým rozlišením, jako jsou obrazovky (lepší hinting není už při rozlišení 600 DPI patrné, a to ani při malé velikosti písma). Dalším bodem, díky němuž je tato zjevná výhoda poněkud problematická, je to, že dobře hintované fonty TrueType jsou velice vzácné. Softwarové balíky s podporou hintingu jsou totiž pro většinu „malonávrhářů“ nesmírně drahé. Jejich použití si tedy mohou dovolit pouze společnosti, jako je například Monotype.

Hlavní rozdíl mezi fonty TrueType a Type 1 tak spočívá v jejich dostupnosti a podpoře aplikacemi. Široká dostupnost TrueType fontů pro Windows vedla k tomu, že řada webových stránek je navrhována s předpokladem, že konkrétní fonty existují. Řada uživatelů Windows má navíc i mnoho dalších TrueType fontů, protože jsou součástí různých aplikací. V Linuxu naopak většina aplikací podporuje fonty Type 1 a úroveň podpory fontů TrueType není tak vysoká. Navíc řada dodavatelů fontů stále poskytuje fonty ve formátu Type 1. Například Adobe nabízí jen velmi málo fontů TrueType. Doporučuji používat to, co pro daný účel vyhovuje a snažit se vyhnout konverzi z jednoho formátu do druhého (protože konverze vždy vede ke ztrátě kvality).

Metafont

Metafont vyvinul Donald E. Knuth jako součást sázecího systému TeX. Metafont je grafický programovací jazyk (podobně jako PostScript), jehož aplikace jsou širší než jen tvorba fontů. Metafonty nabízejí řadu výborných vlastností. Jednou z nejlepších funkcí je vynikající podpora škálování. Metafont Computer Modern má jiné tvary při velikost 20 bodů a 10 bodů[2]. Tvar se s velikos-

[1] Pozn. překladatele: *Hinting*, v typografii redukce tloušťky písma.

[2] Pozn. překladatele: *Point*, typografická jednotka velikosti, obvykle, ale ne vždy a všude, 1/72 palce.

tí mění, protože je vhodné, aby menší fonty byly proporcionálně širší než větší fonty (díky tomu větší fonty vypadají elegantněji, a menší jsou lépe čitelné).

Metafonty mají typicky příponu *mf*. Vykreslují se do rastru závislého na konkrétním zařízení. Vykreslení je pomalé, takže kvalita je vynikající, ale nehodí se pro použití ve WYSIWYG aplikacích.

Řezy písma

Typicky se fonty dodávají v několika variantách, tzv. *řezech*. Například většina fontů obsahuje varianty tučného písma, kurzivy a tučné kurzivy. Některé fonty navíc obsahují kapitálky nebo polotučné varianty. Skupina řezů jednoho fontu se označuje jako *rodina*. Například rodina Garamond obsahuje řezy Garamond-italic, Garamond-bold, Garamond-bold-italic, Garamond-demibold a Garamond-demibold-italic. Garamond ve verzi Adobe expert navíc obsahuje řezy Garamond-smallcaps a Garamond-titling-capitals.

Typografie

V této kapitole budeme hovořit o základech typografie. I když následující informace nejsou životně nutné, pro řadu lidí mohou být zajímavé.

Klasifikace rodin písma

Pevná versus proměnná šířka

Rodiny písem se dají rozlišovat podle řady kritérií. Prvním z nich jsou fonty s pevnou šířkou a fonty s proměnnou šířkou. Fonty s pevnou šířkou vypadají jako psané na stroji, protože všechny znaky jsou stejně široké. Tento typ písma je vhodný např. pro textové editory nebo počítačovou konzolu, nehodí se ale pro delší bloky textu. Druhým typem jsou fonty s proměnnou šířkou. Většinou používáme právě fonty s proměnnou šířkou, i když užitečné jsou i ty s pevnou šířkou (v tomto dokumentu je používáme v příkladech příkazů). Nejznámějším fontem s pevnou šířkou je Courier.

Patkové nebo bezpatkové

Patky[3] jsou malé plošky na koncích znaků. Například písmeno *i* ve fontech jako je Times Roman má patky vedoucí ze základny *i* a z vrcholku *i*. Patková písma jsou *obvykle* považována za lépe čitelná než bezpatková písma. Existuje jich celá řada typů.

Bezpatková písma nemají ony plošky navíc a vypadají tak „ostřeji“. Obvykle se moc nehodí pro sazbu dlouhých dokumentů. Dobře však poslouží v dokumentech, které slouží jen k rychlému prohlédnutí (například webové stránky, katalogy, reklamní brožury). Další vhodné použití mají jako obrazovkové fonty, zejména u malého písma. Díky menší míře detailů jsou za těchto okolností lépe čitelné. Microsoft například prohlašuje font Verdana za dobře čitelný i ve velmi malé velikosti.

Mezi významná bezpatková písma patří Lucida Sans, MS Comic Sans, Verdana, Myriad, Avant Garde, Arial, Century Gothic a Helvetica. Mimochodem, řada typografů považuje písmo Helvetica za nepříjemné. Používá se příliš často a v řadě knih o typografii se doporučuje tomuto písmu vyhnout.

[3] Pozn. překladatele: Patka je anglicky *serif*, francouzské *sans* pak znamená bez. Čili už víme, co to jsou písma *serif* a *sans-serif* :-)

Nové a staré – různé typy patkových písem

Medievalové písmo

Medievalové fonty[4] jsou založeny na velmi tradičních stylech pocházejících až z 15. století. Tato písma mají velmi konzervativní vzhled a jsou dobře čitelná. Hodí se k sazbě delších dokumentů. Anglickým termínem „starý styl" se míní styl, nikoliv stáří. Existují klasické medievalové fonty, například Goudy Old Style, navržené ve 20. století. Tato skupina písem má následující vlastnosti:

- Dobře definované výrazné patky.
- Diagonální zvýraznění. Představte si písmo psané plnicím perem, v němž čáry pod úhlem 45 stupňů od vertikály proti směru hodinových ručiček jsou silnější, a čáry pod úhlem 45 stupňů od vertikály po směru hodinových ručiček jsou tenčí. Medievalové fonty tento efekt často používají.
- Čitelnost. Medievalové fonty jsou prakticky vždy velmi dobře čitelné.
- Jemnost a nevelký kontrast. Fonty používají širší a tenčí čáry, rozdíl mezi nimi je ale nepříliš velký.

Mezi známé medievalové fonty patří Garamond, Goudy Old Style, Jenson a Caslon (ten poslední je sporný, někdy bývá považován za tranzitivní).

Moderní písmo

Moderní písma jsou protějškem medievalových. Mají typicky výraznější charakter a vzhled a používají se spíše ke zvýraznění částí dokumentu, než k sazbě dlouhého textu. Nic ale není jen černé nebo bílé – některá moderní písma, například Computer Modern, Monotype Modern nebo New Century Schoolbook, jsou velmi dobře čitelná (kontrast mezi silnějšími a slabšími čarami je snížený a tak mají lepší čitelnost). Tato písma jsou založena na návrzích oblíbených v 19. století a později. Mezi jejich hlavní rysy patří:

- Jemnější patky, často jen tenké vodorovné linky.
- Svislé zvýraznění. Svislé čáry jsou silnější, vodorovné slabší.
- Často výrazný kontrast mezi silnějšími a slabšími čarami.
- Písma s větším kontrastem mezi silnějšími a slabšími čarami jsou hůře čitelná.

Asi nejznámějším písmem této kategorie je Bodoni. Dalšími jsou Computer Modern a Monotype Modern (z nějž předchozí zmíněné vychází).

Tranzitivní písma

Tranzitivní písma patří někam mezi moderní a medievalová. Řada z nich je stejně dobře čitelná jako medievalová, vycházejí však z poněkud jiného designu. I když je na nich vidět přechod k modernímu stylu, jsou výrazně jemnější, než klasická moderní písma. Do této kategorie patří například Times Roman, Utopia, Bulmer nebo Baskerville. Z nich Times je blíže k medievalové skupině, Bulmer pak blíže k moderní.

Písma s plochou patkou

Písma s plochou patkou[5] odvozují svůj název od tlusté, jakoby blokové patky, na rozdíl od jemných oblouků medievalových písem a tenkých linek moderních písem. Tato písma vypadají robustně a jsou poměrně dobře čitelná. Řada z nich má egyptské názvy – například Nile nebo Egyptienne (i když s egyptštinou nemají nic společného). Tyto fonty jsou vynikající pro psaní textů, u kterých může utrpět kvalita (například texty určené ke kopírování, text v novinách). Nejznámější jsou Claredon, Memphis a Egyptienne, společně s několika písmy používanými v psacích stro-

[4] Pozn. překladatele: Tento specifický typografický termín známe z anglického jazyka jako *Old Style fonts*...

[5] Pozn. překladatele: Anglicky *slub-serif*, česky též písmo se čtyřhrannou patkou nebo egyptienka.

jích. Řada písem s plochou patkou má pevnou šířku a obráceně – prakticky všechna písma s pevnou šířkou mají plochou patku.

Revoluce bezpatkových písem

Nástup bezpatkových písem je poměrně nový fenomén. První známější bezpatková písma byla navržena koncem 19. a začátkem 20. století. Mezi první patří Futura, Grotesque a Gill Sans. Tato písma představují reprezentanty „geometrické“, „groteskní“ a „humanistické“ skupiny bezpatkových písem.

Groteskní

Tato písma si vysloužila svůj název tím, že veřejnost byla zprvu poněkud šokována jejich poměrně výrazným vzhledem. Tato písma vypadají velmi prostě, protože nemají patky a jejich design je jednoduchý a jasný. Hodí se dobře pro sazbu titulků. Lépe čitelné varianty se používají v komiksech a v marketingových brožurách, kde se text vyskytuje v menších celcích. Nevypadají tak umělecky jako geometrická písma. V porovnání s nimi mají větší variace v šířce, více tahů a jsou hranatější (nepoužívají tolik kruhových oblouků). Významný rozdíl proti geometrickým písmům vykazují velké *G* a malé *a*. I zde jsou tato písmena jednoduchá, ale nezacházejí do takového extrému jako „brutálně“ avantgardní geometrická písma.

Mezi známé představitele této kategorie patří Helvetica, Grotesque, Arial, Franklin Gothic a Univers.

Geometrická

Písmo Futura bylo uvedeno sloganem *form follows function* – tedy, forma slouží obsahu. Geometrická skupina písem má výrazně minimalistický vzhled. Významným rysem je konstantní tloušťka linky. Je to zřetelné zejména u tučného řezu. Groteskní a humanistická písma v tučném řezu vykazují variace v tloušťce linky, u geometrických písem to nastává jen výjimečně. Další významný rys je minimalistický návrh. Znaky jsou téměř vždy tvořeny svislými a vodorovnými čarami v kombinaci s kruhovými oblouky (vypadají jako kreslená pravítkem a kružítkem). Znaky obsahují minimum tahů. Nejvýrazněji se geometrická písma poznají podle velkého *G*, které se skládá jen ze dvou křivek – dlouhého kruhového oblouku a vodorovné čárky. Podobně výrazné je malé *a* – svislá čára a kružnice. Klasická podoba písmene „a“ se nepoužívá – je příliš složitá. Známá písma této kategorie jsou Avant Garde, Futura a Century Gothic.

Humanistická

Jak název možná napovídá, návrh těchto písem se snaží omezit mechaničnost jejich vzhledu. V řadě ohledů se podobají spíše patkovým písmům. Vypadají jako „rukou psaná“. Vykazují lehké variace tloušťky linky, což vyniká zejména v tučné variantě. Oblé části nejsou tak rigidní jako u geometrických písem. Mnohá používají dvoudílné malé *g*, podobného tvaru jako u klasických patkových písem. Dají se snáze používat aniž by vzniklé dokumenty byly příliš ošklivé, protože jsou částečně příbuzná s klasickými patkovými písmy.

Kompatibilní písma

Kombinování různých písem není jednoduché, proto se doporučuje to s kombinacemi nepřehánět. Logická volba dvou druhů písem padá na patkové a bezpatkové. Lze říci, že dobře vypadá kombinace medievalových a humanistických písem, anebo moderních a geometrických. Tranzitivní písma se rovněž kombinují s humanistickými. Písma s plochou patkou se obvykle kombinují s groteskními písmy, i když některá z nich lze kombinovat i s geometrickými a humanistickými.

Vyplývá z toho tedy, že rozumná filozofie je kombinovat konzervativní patková písma a mírnější bezpatková, a divočejší moderní patková písma s avantgardněji vyhlížejícími geometrickými.

Ligatury, kapitálky a speciální písma

Ligatury

Volba vhodných rozestupů mezi písmeny s sebou přináší řadu problémů. Například při správné sazbě písmen *fi* musí být *i* velmi blízko k *f*. Problém je, že tečka nad *i* koliduje s vrcholem *f*, a horní patka *i* koliduje se střední linkou *f*. Proto se v písmech používají *ligatury* – slitky. Ligatura *fi* je jediný znak, kterým se nahrazuje dvojice znaků *f* a *i*. Většina písem obsahuje slitky *fi* a *fl*. Speciální písma obsahují často i další slitky, například *ffl*, *ffi* a beztečkové *i*.

Kapitálky

Kapitálky jsou písma, v nichž je velikost velkých písmen zmenšena na velikost malých. Používají se často ke psaní výrazných nadpisů (a často je používá LaTeX). Při psaní nadpisu pomocí kapitálek se typicky první písmeno napíše normálně velké, zbytek pak kapitálkami. Výhodou oproti použití všech písmen velkých je mnohem lepší čitelnost (psaní jen velkými písmeny je v typografii veliký prohřešek).

Speciální písma

Speciální písma obsahují různá vylepšení – slitky, ornamenty, kapitálky a ozdobné kapitálky (okrasná, kaligrafická písmena).

Metriky a tvary písem

Metriky fontů definují rozestupy mezi znaky u fontů s proměnnou šířkou. Obsahují informace o velikosti písma a informace o párování písmen (tzv. *kerning*), které definují dvojice písmen, jež budou používat speciální rozestup. Například písmena *To* typicky představují stlačovaný pár, protože při správném rozestupu je *o* poněkud podsazeno pod *T*. Tyto informace jsou potřebné pro sázecí programy jako LaTeX kvůli zalamování řádků a stránek. Podobně jsou nutné pro publikační WYSIWYG programy.

Další důležitou komponentou písma jsou obrysy, nebo tvary. Jednotlivé znaky ve fontech se označují jako *glyphy*.

Zpřístupnění fontů v systému X Window

Existuje řada způsobů, jak lze do systému X Window přidat fonty. XFree86 používají k hledání fontů *font path* – což je prostý seznam adresářů, nebo *font server*. Font server je proces běžící na pozadí, který XFree86 poskytuje fonty. Výhodou font serveru je, že dokáže posílat fonty na vzdálený displej.

Nedávno byl *xfs* (X font server) doplněn o podporu fontů TrueType a běží jako samostatný program. Tato upravená verze se používá v distribucích RedHat a derivátech, a je součástí XFree86 verze 4.0 a novějších. *Xfs* je prostě standardní font server používaný s XFree86, jeho zdrojový kód je součástí zdrojového stromu XFree86. V novějších distribucích se používá verze, která běží jako samostatný program. Samostatný font server doplněný o podporu fontů TrueType (tuto podporu zajišťuje font server *xfstt*) je momentálně asi nejlepší dostupné řešení pro správu fontů. Mezi výhody patří:

- Podpora různých typů fontů včetně Type 1, TrueType a rastrových.

- Podpora zobrazení na vzdálených displejích.
- Zjednodušená správa fontů – provádí se nástrojem chkfontpath namísto editace konfiguračních souborů. Tím se jednak zjednodušuje práce a zároveň se usnadňuje možnost tvorby skriptových balíků.

Protože různé distribuce používají různé konfigurace, neexistuje univerzální řešení. Uživatele lze rozdělit do tří skupin:

- Distribuce používající samostatný xfs doplněný o podporu TrueType. Sem patří RedHat a distribuce na něm založené – Mandrake a TurboLinux. Debian 3.0 by měl rovněž obsahovat doplněný xfs. Uživatelé těchto systémů mohou fonty TrueType i Type 1 instalovat prostřednictvím xfs.
- Některé distribuce obsahují samostatný xfs, ale bez podpory TrueType. XFree86 podporují TrueType od verze 4.0. Patří sem stabilní („potato“) Debian. Uživatelé těchto systémů by měli fonty Type 1 instalovat pomocí xfs, fonty TrueType pomocí xfstt. Uživatelé systému Debian si mohou přečíst dokument *TT-Debian mini-HOWTO* (http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/other-formats/html_single/TT-Debian.html), kde se popisuje instalace fontů TrueType v Debianu.
- Pokud nepoužíváte xfs, musíte fonty Type 1 instalovat přidáním do font path XFree86 a nástrojem xset. Uživatelé XFree86 3.x mohou fonty TrueType instalovat pomocí xfstt, uživatelé verze XFree86 4.x je mohou přidávat do font path.

Font path

XFree86 hledá fonty prohledáváním tzv. *font path*, seznamu adresářů (nebo serverů, viz dále), které obsahují fonty. Když aplikace požaduje nějaký font, postupně se prohledávají adresáře ve font path, dokud font není nalezen. Chcete-li fonty zpřístupnit, musíte správně nastavit font path. Další adresář můžete do font path přidat příkazem *xset fp+ adresář*. Poté musíte X serveru říct, aby znovu nalezl dostupné fonty příkazem:

```
xset fp rehash
```

Protože tyto příkazy budete chtít provést automatizovaně, měli byste je umístit do souboru *.xinitrc* (nebo do souborů *.Xclients* nebo *.xsession* – to záleží na tom, jak spouštíte systém X Window). (Je rozumné dva z těchto souborů vytvořit jako symbolický odkaz na třetí, aby se předešlo zmatkům.) Další možnost, jak příslušné příkazy automaticky provést, je editace souboru *XF86Config*. Pokud například chcete do font path při spuštění X automaticky přidat adresář */usr/share/fonts/myfonts*, upravte soubor *XF86Config* takto:

```
...
Section "Files"
...
FontPath /usr/share/fonts/myfonts
...
EndSection
...
```

Výhodou editace souboru *XF86Config* je to, že změny platí v celém systému.

Instalace fontů Type 1

Spuštění Type1inst

Nejjednodušší metoda instalace fontů Type 1 je spuštěním nástroje *type1inst*. Jedná se o skript, který automaticky vytvoří soubory *fonts.dir* a *fonts.scale*, které systém X Window potřebuje. Stačí se přepnout do adresáře s fonty a zadat *type1inst*.

Pokud používáte xfs

Nyní musíte fonty přidat do font path. Pokud používáte samostatně běžící verzi *xfs*, provedete to úpravou konfiguračního souboru. Uživatelé distribuce RedHat pouze spustí *chkfontpath*, formát je *chkfontpath --add adresář*.

Fonty budou přístupné po restartu *xfs*, musíte jej tedy znovu zavést signálem SIGHUP.

Nyní by fonty měly být přístupné. Spusťte *xset fp rehash*, aby si je X našly.

Pokud nepoužíváte xfs

V tom případě musíte adresář s fonty přidat do font path postupem, který byl popsán dříve.

Fonty TrueType

Přidání fontů TrueType je o něco obtížnější, protože potřebujete font server, který je podporuje. To umí dva font servery – *xfstt* a *xfs*.

Xfstt je font server pro TrueType fonty. I když se snadno konfiguruje a je velmi šikovný, *xfs* je zřejmě oblíbenější. Hlavní výhodou *xfs* je, že podporuje fonty Type 1 i TrueType.

Xfstt

Server *xfstt* si můžete stáhnout a nainstalovat. Po nainstalování provedete následující:

1. Nainstalovat fonty do správného adresáře (viz dokumentace font serveru).
2. Přepnout se do tohoto adresáře a spustit *xfstt --sync*. Font server prohledá fonty a vytvoří soubor *fonts.dir*.
3. Do font path přidejte *unix/:7100*.

Nyní by se fonty měly v aplikacích jako GIMP a Netscape objevovat a zobrazovat. Tyto operace můžete nakonfigurovat tak, aby se provedly vždy při startu systému. Podívejte se, zda existuje inicializační soubor (pokud používáte RPM, zkuste *rpm -ql xfstt | grep init* a hledejte něco jako */etc/rc.d/init.d/xfstt*). Pokud inicializační soubor neexistuje, stačí do */etc/rc.local* přidat následující řádky:

```
/usr/X11R6/bin/xfstt --sync
/usr/X11R6/bin/xfstt &
```

xfs

Některé novější distribuce Linuxu obsahují font server *xfs* spouštěný jako samostatný program. Konkrétně RedHat a distribuce na něm založené používají modulární *xfs* s podporou TrueType. Spouštění *xfs* jako samostatného programu má několik výhod, zejména pokud obsahuje rovněž podporu fontů TrueType. Hlavní výhodou je to, že jelikož font server není součástí X serveru, je možné fonty poskytovat na vzdálené displeje. Navíc je také jednodušší modifikovat font path.

Xfs font path

Jako font server používá *xfs* svou vlastní font path. Jak to zapadá do celého systému? Funguje to takto: *xfs* přidáte do font path XFree86 jako *unix/:port*. Od této chvíle jsou všechny fonty dostupné font serveru dostupné také v XFree86.

Font path *xfs* se definuje v konfiguračním souboru *xfs*, což je */etc/X11/fs/config* na RedHatu, a */etc/X11/xfs/config* na Debianu. Uživatelé distribuce RedHat nemusí tento soubor explicitně editovat, mohou použít nástroj *chkfontpath*. Jeho syntaxe je jednoduchá:

```
chkfontpath --add directory
```

Uživatelé ostatních distribucí mohou konfigurační soubor editovat takto:

```
catalogue = /usr/X11R6/lib/X11/fonts/misc:unscaled,
...
/usr/share/fonts/my_new_fonts/,
...
/usr/share/fonts/some_other_directory
# in 12 points, decipoints
default-point-size = 120
...
```

Výše uvedený příklad přidá do font path adresář */usr/share/fonts/my_new_fonts*. Pozor na to, že poslední řádek v seznamu adresářů *nekončí čárkou*. Aby se modifikace font path projevily, musíte *xfs* znovu nahrát příkazem */etc/rc.d/init.d/xfs reload* nebo signálem SIGHUP příkazem *kill -HUP [pid]* nebo *killall -HUP xfs*. Alternativně je možné *xfs* pouze restartovat, ale v takovém případě je rozumné restartovat celé X sezení.

Instalace fontů v xfs

Příprava fontů pro *xfs* obnáší následující kroky:

- Pokud nemáte xfs nainstalován, nainstalujte jej.
- Nahrajte fonty do nového adresáře.
- Pokud instalujete fonty Type 1, připravte nový adresář tak, že v něm spustíte příkaz *type1inst*.
- Pokud instalujete fonty TrueType (nezapomeňte, že ne ve všech distribucích xfs fonty TrueType podporuje), připravte adresář tím, že v něm spustíte příkazy *ttmkfdir -o fonts.scale* a *mkfontdir*. Pokud jste fonty nahráli do nového adresáře, budete muset *fonts.scale* zkopírovat do fonts.dir nebo vytvořit odkaz. Příkaz *ttmkfdir* je součástí balíku *freetype*.
- Teď můžete nový adresář přidat do font path *xfs*. Uživatelé distribuce RedHat to provedou příkazem *chkfontpath*, uživatelé jiných distribucí editací konfiguračního souboru *xfs*.
- Pokud je xfs už nainstalován, zjistěte si, na kterém portu běží. Můžete to provést příkazem:

  ```
  ps ax | grep xfs
  ```

- Nyní příkazem *xset -q* zkontrolujte font path XFree86.
- Pokud font path obsahuje něco jako *unix:/port*, kde *port* odpovídá hodnotě, na níž server běží, pak je *xfs* nastaven správně. Pokud ne, musíte jej přidat do font path XFree86. *xset fp+ unix:/port* a *xset fp rehash*. Trvalé přidání můžete provést editací souboru .xinirc. Přidání platné pro celý systém… provedete úpravou souboru XF86Config (pravděpodobně */etc/X11/XF86Config* nebo */usr/X11R6/lib/X11/XF86Config*) přidáním řádku *FontPath „unix:/port“* v sekci *Files*: Například:

```
...
Section "Files"
...
FontPath "unix/:-1"
...
EndSection
...
```

- Pokud je xfs správně nainstalován, můžete jej znovu nahrát výše popsaným postupem, nebo restartovat příkazem:

```
/etc/rc.d/init.d/xfs restart
```

- Po restartu *xfs* je rozumné restartovat i X sezení.

Zpřístupnění fontů v programu Ghostscript

Abyste fonty zpřístupnili programu *ghostscript*, stačí mu říct, kde se nacházejí příslušné soubory. Je nutné editovat soubor */usr/share/ghostscript/version/Fontmap*. Formát tohoto souboru je velmi jednoduchý a samovysvětlující.

Type 1

Přidání fontů Type 1 je velice prostě. V adresáři s přidávanými fonty spusťte *type1inst*. Tím vznikne soubor *Fontmap*, který připojíte k souboru *Fontmap* programu *ghostscript*.

TrueType

Přidání fontu TrueType je poněkud složitější, protože musíme zjistit jeho jméno. Jedna možnost (brutální) je použití nástroje *ttf2pt1* pro konverzi TrueType na Type 1 a pak jméno fontu zjistit ze souboru *afm*. (Určitě musí existovat i jednodušší cesta, ale tato funguje.) Stačí provést:

```
ttf2pt1 -A fontname - 2 > /dev/null  |grep FontName
```

Pak do souboru *Fontmap* přidáte záznam ve správném formátu, například *some-font (/usr/share/fonts/subdirectory/somefont.pfb);* Tento postup sice funguje, ale zkuste jej provést pro 500 nebo tak nějak fontů… To je situace, která si žádá o krátký skript:

```
#!/usr/bin/perl
# ttfontmap - generuj soubor Fontmap pro TrueType fonty
my $directory=shift || print STDERR "Usage: ttfontmap {directory}\n";

$directory=~s/\/$//;

for my $fontname ( glob ( "$directory/*.ttf" ) )
{
    open ( R, "sh -c \"ttf2pt1 -A $fontname - 2>/dev/null\" |" );
    while ( <R> )
    {
        if ( $_ =~ /^FontName/ )
        {
            s/^FontName\s*//;
            chomp;
            print "/" .  $_ . "    ($fontname);\n" ;
        }
```

```
    }
    close R;
}
```

Skript si můžete stáhnout na adrese http://pegasus.rutgers.edu/~elflord/font_howto/ttfontmap.

Stačí výše uvedený skript zkopírovat do souboru *ttfontmap* a ten pak nahrát někam do cesty (například do */usr/bin*). Skript pak spustíte příkazem:

```
ttfontmap adresář > soubor,
```

kde jako *adresář* zadáte adresář s fonty. Dostanete *soubor*, který stačí připojit k souboru *Fontmap*. Jistě vás napadá použití *ttfontmap adresář >> /usr/share/ghostscript/version/Fontmap*. Důrazně toto použití *nedoporučuji* (co se stane, když se přepíšete a místo „>>“ dáte jen „>“?)

Použití Ghostscriptu k prohlížení fontů

Jakmile fonty v programu *ghostscript* zpřístupníte, můžete si je také prohlédnout. Spusťte interprer *ghostscript* na soubor *prfont.ps*, který je součástí instalace, a po jeho spuštění zadejte */NázevFontu DoFont* (kde *NázevFontu* je jméno fontu, který si chcete zobrazit). Jsou i jiné způsoby jak spustit *gs*. Pokud například budete chtít vytvořit postscriptový soubor, který si budete moci prohlédnout v nějakém příjemnějším prohlížeči, jako je například *gv*, můžete zadat *gs -sDEVICE=pswrite -sOutputFile=somefile.ps prfont.ps*. Vzniklý soubor si pak můžete třeba i vytisknout.

Konverze TrueType na Type1

Proč?

Lepší otázka možná je – *proč ne?* Typický uživatel Linuxu má za sebou přechod z Windows a obvykle má nashromážděnu celou řadu fontů TrueType. Řada těchto fontů (například ty, dodávané s MS Wordem nebo s Corelem) má velmi dobrou kvalitu. Některé linuxové aplikace, například StarOffice nebo LaTeX však fonty TrueType nepodporují, i když podporují fonty Type1[6]. Je to škoda, protože vzhledem k podpoře TrueType fontů v ghostscriptu a ve font serverech má Linux potřebnou infrastrukturu pro práci s těmito fonty.

Jak?

Navštivte stránky http://quadrant.netspace.net.au/ttf2pt1/ a stáhněte si program *ttf2pt1*. Konverzi fontu TrueType na Type 1 pak provedete následujícím příkazem:

```
ttf2pt1 -b soubor.ttf název,
```

kde *název* bude jméno souboru nově vytvářeného fontu Type 1. (Název může být libovolný, je ale rozumné pojmenovat jej stejně jako původní font TrueType, tedy například *ttf2pt1 -b foo.ttf foo*.)

Tento postup funguje rozumně pro jeden font. Pokud jich ale máte hodně, potřebujeme nějakou šikovnější metodu. Nejjednodušší je použít následující smyčku:

```
for X in *.ttf; do ttf2pt1 -b $X ${X%%.ttf}; done
```

Další možnost je stáhnout si balík *ttfutils* (http://pegasus.rutgers.edu/~elflord/font_howto/ttfutils-0.2.tar.gz) a provést konverzi příkazem *ttf2type1*:

```
ttf2type1 *.ttf
```

[6] Zdá se, že StarOffice nyní fonty TrueType podporují, nicméně postup je značně netriviální.

WYSIWYG publikování a fonty

Úvod a přehled

Instalace fontů pro publikování metodou WYSIWYG je v Linuxu poměrně složitá úloha. Typicky obnáší tři kroky:

- Zpřístupnění fontů pro X server.
- Zpřístupnění fontů pro ghostscript.
- Zpřístupnění fontů v samotné aplikaci.

Hlavním důvodem této složitosti je to, že *tiskový systém* (ghostscript) je nezávislý na *zobrazovacím systému*. Díky tomu v Linuxu levá ruka neví, co dělá pravá. Řešení problému je netriviální, protože je možné, že tiskové fonty a obrazovkové fonty budou umístěny na různých počítačích, takže není záruka, že všechny fonty používané X klientem budou také tisknutelné.

Dobrá zpráva je, že většina WYSIWYG aplikací tento problém rozumně řeší. Řešení spočívá v mechanismu, který nějakým způsobem mapuje obrazovkové fonty na tiskové fonty. (Což je hlavní problém. Další problém představuje například seskupení normální, tučné a kurzivové varianty jednoho písma do správné rodiny.) Bohužel neexistuje žádný standardní způsob, jak to udělat. Celý proces by se velmi zjednodušil zavedením standardu pro správu fontů, protože pak by všechny aplikace mohly používat celosystémově platnou konfiguraci.

Applixware

Existují dvě možnosti, jak do Applixware instalovat fonty. Jedna spočívá v použití FontTastic, což je „privátní" font server Applixware. Druhá možnost je editace mapy fontů Applixware tak, aby se používaly v systému nainstalované fonty. Instalace pomocí font serveru je mnohem pohodlnější, takto nainstalované fonty je však možné tisknout pouze v rozlišení 300 DPI.

FontTastic

Použití programu FontTastic je ta jednodušší metoda. Instalaci nových fontů provedete následujícím postupem:

1. Spusťte Applixware jako root.
2. Vyberte nabídku *Tools*.
3. Zvolte *Font Installer*.
4. Potvrďte dialog tlačítkem *OK*.
5. Zvolte nabídku *Catalog* a příkaz *Create*.
6. Vyplňte údaj *Catalog Name*. Není důležité, co tam zadáte. Pro zbytek příkladu budeme předpokládat, že jste zadali *foobar*.
7. Ze seznamu katalogů vyberte *foobar*.
8. Z nabídky *Services* zvolte *install fonts into -> FontTastic font server*.
9. Zkontrolujte, že v seznamu katalogů je vybrán *foobar* a klepněte na tlačítko *Select Files*.
10. V dialogu pro výběr souborů zvolte fonty, které chcete nainstalovat. Po vybrání klepněte na *OK*. Pokud budete chtít například nainstalovat font *arial.ttf* z adresáře */usr/share/fonts/ttfonts*, zadáte v dialogu *Current Directory* hodnotu */usr/share/fonts/ttfonts*, vyberete soubor *arial.ttf* a klepnete na *OK*. Můžete najednou vybrat více souborů, ale musí se nacházet ve stejném adresáři.

11. Seznam fontů můžete upravit, fonty je možné odstranit nebo přejmenovat.
12. V nabídce *Services* zvolte *Update* a *OK*. V nabídce *Services* zvolte *Exit* a ukončete Applixware.
13. Hotovo! Při příštím spuštění Applixware budete moci nové fonty použít!

Použití systémových fontů v Applixware

Tato metoda je obtížnější, ale poskytuje lepší výsledky. Doporučuji ji použít pro důležité fonty, které často používáte. Jednotlivé kroky vypadají takto:

Instalace fontů v X Window

Viz kapitola 4.

Instalace fontů v Ghostscriptu

Viz kapitola 5.1.

Editace fontmap.dir

Toto je poslední krok ve zpřístupnění fontů pro Applixware, a je to také krok nejsložitější. Soubor *fontmap.dir* se nachází v instalaci Applixware pod *axdate/fontmetrics*. Důvodem tohoto kroku v zásadě je říct Applixware, které obrazovkové fonty se mapují na které definiční soubory. Obecně je to netriviální problém, protože obrazovkové fonty nemusí být vždy na stejném počítači, kde je nainstalována aplikace.

Popíšeme si, jak fonty do *fontmap.dir* přidat. V následujícím příkladu přidáváme font Baskerville Italic.

1. Nejprve přidáme řádek, který říká *FontRecord = Baskerville-Normal-Italic*. V zásadě může být jméno použité v tomto záznamu naprosto libovolné. Název však musí být jedinečný. Proto je rozumné používat stejné názvy, které pro daný font používá Ghostscript.
2. Dále přidáme řádek *Family = Baskerville*. Název rodiny se bude objevovat v nabídce pro výběr fontů. Typicky toto jméno není jedinečné, protože varianty tučná, normální, kurziva a tučná kurziva většinou spadají do stejné rodiny.
3. Pokud je písmo tučné, kurziva nebo obojí, musíme přidat řádku *Slant = 1* pro kurzivu a *Weight = 1* pro tučné písmo. Jde-li o tučnou kurzivu, přidáme oba řádky. V našem příkladu přidáváme pouze *Slant = 1*.
4. Dále přidáme řádek *ScreenName = „-paradise-baskerville-medium-i-normal--0-0-0-0-p-0-iso8859-1“*. *ScreenName* je název, pod kterým daný font zná X server. Fonty, jejichž názvy obsahují řetězec *bask*, můžeme vypsat příkazem *xlsfonts|grep -i bask*.
5. Dále přidáme řádek definující název tiskového fontu: *PostScriptPrinterName = Baskerville-Normal-Italic*.
6. Konečně musíme specifikovat umístění definičních souborů fontu: *MetricsFile = /usr/share/fonts/misc/baskvli.afm* a *Type1FontFileName = /usr/share/fonts/misc/baskvli.pfb*. Pokud přidáváte font TrueType, můžete nástrojem *ttf2pt1* vytvořit soubor *afm*: *ttf2pt1 -A foo.ttf –> foo.afm* (nebo můžete použít balík *ttfutils* a použít příkaz *ttf2afm*). Pak přidáte pouze řádek *MetricsFile = /usr/share/fonts/misc/foo.afm*. Nepřidáváte řádek *Type1FontFileName* – o to ať se postará Ghostscript.

A to je všechno. Po přidání celé rodiny byste měli dostat něco jako:

```
FontRecord = Baskerville-Normal
Family = Baskerville
```

```
ScreenName = "-paradise-baskerville-medium-r-normal--0-0-0-0-p-0-iso8859-1"
PostScriptPrintName = Baskerville-Normal
MetricsFile = /usr/share/fonts/misc/baskvl.afm
Type1FontFileName = /usr/share/fonts/misc/baskvl.pfb

FontRecord = Baskerville-Normal-Italic
Family = Baskerville
Slant = 1
ScreenName = "-paradise-baskerville-medium-i-normal--0-0-0-0-p-0-iso8859-1"
PostScriptPrintName = Baskerville-Normal-Italic
MetricsFile = /usr/share/fonts/misc/baskvli.afm
Type1FontFileName = /usr/share/fonts/misc/baskvli.pfb

FontRecord = Baskerville-Bold
Family = Baskerville
Weight = 1
ScreenName = "-paradise-baskerville-bold-r-normal--0-0-0-0-p-0-iso8859-1"
PostScriptPrintName = Baskerville-Bold
MetricsFile = /usr/share/fonts/misc/baskvlb.afm
Type1FontFileName = /usr/share/fonts/misc/baskvlb.pfb

FontRecord = Baskerville-Bold-Italic
Family = Baskerville
Weight = 1
Slant = 1
ScreenName = "-paradise-baskerville-bold-i-normal--0-0-0-0-p-0-iso8859-1"
PostScriptPrintName = Baskerville-Bold-Italic
MetricsFile = /usr/share/fonts/misc/baskvlbi.afm
Type1FontFileName = /usr/share/fonts/misc/baskvlbi.pfb
```

Pomocí tohoto konfiguračního souboru jde udělat i jiné věci. Samotný soubor obsahuje vysvětlení formátu jednotlivých příkazů.

StarOffice

Popíšeme si postup platný pro Star Office 5.0. Postup v novějších verzích Star Office je podobný, nástroj se ale jmenuje *spadmin* a nikoliv *psetup*. Bude vhodné uvést, že vynikajícím zdrojem informací na toto téma je stránka Johna McLaughlina (http://www.mindspring.com/~john_mcl/adding_fonts.html), ze které jsme pro následující výklad čerpali.

Vyzkoušel jsem Star Office 5.0 i 5.1 a zdá se, že Star Office 5.1 jsou při přidávání fontů o něco chytřejší. Do Star Office 5.0 se mi nepodařilo přidat fonty TrueType, do 5.1 a novější verze ano.

Napřed si zálohujte konfiguraci!

Je rozumné si pořídit zálohu pro případ, že byste konfiguraci kompletně rozházeli. Manipulace s fonty má dopad na různé soubory v adresáři *xp3*. Rozhodně byste měli zálohovat soubor *xp3/psstd.fonts*. Doporučuji ale zálohovat celý adresář *xp3*. Stačí se přepnout do adresáře Star Office a zadat:

```
tar cvzf xp3.tgz xp3
```

Zálohu obnovíte tak, že smažete adresář *xp3* a rozbalíte archiv:

```
rm -rf xp3
```

```
tar xvzf xp3.tgz
```

Přidání fontů Type 1

Přidání fontů Type 1 je poměrně jednoduché. Pokud byste chtěli ve Star Office 5.0 použít fonty TrueType, nejlepší je provést jejich konverzi na fonty Type 1 a pak pokračovat následujícím postupem. Používáte-li Star Office 5.1, můžete fonty True Type nainstalovat přímo (i když postup je o něco obtížnější). Nejprve začněte obvyklými kroky – zpřístupněte fonty v X a v Ghostscriptu. Poté je možné fonty do Star Office nainstalovat nástrojem *psetup*. Postup je následující:

1. Jako root spusťte *psetup* (nebo *spadmin* ve Star Office 5.1).
2. Klepněte na tlačítko *Add fonts*.
3. Nejjednodušší metoda je nyní klepnout na *Initialize font paths*. Tím se v nabídce vytvoří seznam všech fontů, které zná systém X Window.
4. Zvolte adresář obsahující fonty, které chcete přidat (měl by být v nabídce) a klepněte na *OK*.
5. Klepněte na tlačítko *Convert all font metrics*.

A je to. Po restartu Star Office by měly být nové fonty přístupné.

Přidání fontů TrueType

Přidání fontů TrueType sice není jednoduché, ale je možné. Po delším experimentování a dlouhém čtení stránek Johna McLaughlina (http://www.mindspring.com/~john_mcl/adding_fonts.html) se mi konečně podařilo ve Star Office 5.1 fonty přidat. Tento postup nefunguje ve Star Office 5.0. Pokud tisknete přes ghostscript, použijte následující postup:

- Zpřístupněte fonty v X Windows.
- Zpřístupněte fonty v Ghostscriptu.
- Pro přidávané fonty potřebujete vytvořit soubory *afm*:

  ```
  ttf2pt1 -A foo.ttf - > foo.afm
  ```

- Alternativně použijte balík *ttfutils* (http://pegasus.rutgers.edu/~elflord/font_howto/ttfutils-0.2.tar.gz) a příkaz *ttf2afm*. Jeho výhodou je, že můžete zpracovat více fontů najednou:

  ```
  ttf2afm *.ttf
  ```

- Star Office potřebují pro každý soubor *ttf* odpovídající soubor *pfb*. Můžete je vytvořit příkazem:

  ```
  touch foo.pfb
  ```

- Star Office tyto soubory používají pouze k tisku. Po přidání fontu do PPD si Star Office myslí, že font je přímo součástí tiskárny (ve skutečnosti je součástí ghostscriptu) a soubory *pfb* se k ničemu nepoužívají, nicméně se zdá, že Star Office trvají na jejich přítomnosti, aby font nainstalovaly.
- Pomocí *spadmin* můžete font(y) nainstalovat.
- Nyní fonty přidejte do souboru PPD odpovídajícího konfiguraci vaší tiskárny. Název musí být stejný jako název používaný ve Star Office, ne jako název používaný v Ghostscriptu. Pokud se například font jmenuje *foobar.ttf* a odpovídající afm soubor *foobar.afm*, použijte jako název fontu *foobar*. Celý řádek bude vypadat nějak takto:

  ```
  *Font cloistrk: Standard "(001.002)" Standard ROM
  ```

Pokud netisknete přes ghostscript, musíte se vyrovnat s jinými problémy. V takovémto případě by nebylo rozumné Star Office přesvědčovat, že tiskárna umí dané fonty tisknout, protože ona to *neumí*. Zatímco *gv* by postscriptové soubory zobrazil správně, nepodařilo by se je vytisknout. Pokud používáte postscriptovou tiskárnu, rozdíl je následující:

- Needitujte soubor PPD.
- Namísto vytvoření prázdného *pfb* souboru příkazem touch potřebujete, aby soubory *pfb* byly skutečné postscriptové fonty Type 42. Font Type 42 je ve skutečnosti „tiskárnový TrueType font". I pokud tyto soubory používáte, většinou o nich nevíte, protože většina aplikací s nimi pracuje transparentně. Fonty Type 42 vytvoříte příkazem *ttfps* (ftp://ftp.dcs.ed.ac.uk/pub/jek/programs/ttfps.tar.gz):

```
ttfps foo.ttf foo.pfb
```

Můžete narazit na různé problémy. Star Office nemusí k zobrazování používat správné fonty. Je rozumné zkontrolovat soubor *xp3/psstd.fonts* a případně jej upravit tak, aby Star Office zobrazovaly opravdu tím fontem, kterým chcete. Navíc Star Office špatně zvládají konfigurační problémy. Pokud nebude konfigurace v pořádku, je možné, že se editor vůbec nespustí. Proto byste si měli konfiguraci zálohovat.

Pod pokličkou

Chcete-li ve Star Office instalovat fonty TrueType, měli byste vědět, jak Star Office s fonty pracují. Když spustíte *spadmin* nebo *psetup*, stane se následující:

- Star Office vytvoří symbolické odkazy na *pfb* soubory v adresáři *xp3/pssoftfonts*.
- Soubory *afm* ze zkopírují do adresáře *xp3/fontmetrics/afm*.
- Do souboru *xp3/psstd.fonts* se přidá záznam. Tento soubor obsahuje názvy všech fontů, používaných ve Star Office a mapuje je na příslušné definiční soubory.

Proto je rozumné zálohovat celý adresář *xp3* – je to jediný spolehlivý způsob, jak obnovit výchozí konfiguraci Star Office.

Word Perfect

Tady nic nenajdete. Dokonalou příručkou pro instalaci fontů do Word Perfectu jsou stránky Roda Smithe, http://www.rodsbooks.com/wpfonts/.

Netscape

Asi nejobávanější aplikací, co se týče, fontů je Netscape. Existuje ale poměrně jednoduchá metoda, jak na jeho ohavné fonty zaútočit. Hlavní problém spočívá v tom, že Netscape chce používat fonty v rozlišení 75 DPI, což je typicky dost málo. Můžete to napravit v souboru *.Xdefaults*:

```
Netscape*documentFonts.sizeIncrement: 20
Netscape*documentFonts.xResolution*iso-8859-1: 100
Netscape*documentFonts.yResolution*iso-8859-1: 100
```

Namísto čísla 100 můžete zvolit cokoliv jiného. Pokud máte rádi hodně velké písmo, můžete klidně zadat 150.

Další dobrou věcí při řešení ošklivých fontů v Netscape je pořídit si balík fontů Microsoft. Tyto fonty se používají velice často a rozdíl ve zobrazení, pokud je máte nebo nemáte, je značný.

TeX / LaTeX

Stručný úvod do fontů v TeXu / LaTeXu

Přidávat fonty v programech TeX a LaTeX je poměrně složité. Stejně jako u řady jiných věcí je to ale snadné, pokud víte, jak na to. Některé fonty se distribuují ve formátu metafont, jiné jako Type 1. Obvykle se Type 1 dá snáze sehnat. Nicméně formát metafont má tu výhodu, že při různých velikostech se mění tvary písmen, zatímco fonty Type 1 a TrueType jsou ve všech velikostech pouze zvětšenina nebo zmenšenina stejného tvaru. Hlavní výhodou tohoto chování je, že fonty by měly být při menších velikostech relativně širší a při větších velikostech relativně užší.

V následujícím popisu se zaměříme na fonty Type 1, které jsou snáze dostupné a jejich instalace je problematičtější.

Dále uvádíme základní úvod do fontů v LaTeXu. Při práci s fonty Type 1 používá LaTeX následující soubory:

- *.pl* – property list. Textová verze souboru metriky fontu.
- *.vpl* – virtual property list. Textová verze virtuálního souboru fontu.
- *.fd* – font definition. Používá se k definici rodiny fontů.
- *.tfm* – text font metric. Soubor s metrikou, analogický souborům .afm u fontů Type 1. Metriky jsou nutné ke správnému formátování stránky.
- *.vf* – virtuální font. Tyto soubory obsahují podrobnosti o kódování a fungují jako interprety. TeX je považuje za normální fonty. Představte si, že máme nějaký podivný font *foobarexp.pfb*, který obsahuje pár (řekněme 20) alternativních znaků, a že máme virtuální font, který některé z těchto alternativních znaků používá (zbytek znaků používá z fontu *foobar.pfb*). Dvips může říct „chci znak 65 virtuálního fontu *foo.vf*“. Dvips ví, že 65 je ve schématy TeXu vždy „a“. Pak virtuální font mapuje požadavek TeXu na požadavek znaku 14 fontu *foobar.pfb* (což může být alternativa znaku „a“). Mechanismus virtuálních fontů je velmi pružný a umožňuje sestavovat fonty z řady různých souborů. Je to užitečné při práci s fonty jako jsou „expertní“ fonty Adobe.
- *.pk* – rastrový font závislý na zařízení. Obvykle se vytváří podle potřeby (vykreslováním fontů Type 1 a metafontů). Typicky mají velmi vysoké rozlišení (300-1 200 DPI) a slouží k vykreslení na tiskárně. Vzhledem k vysokému rozlišení a vzhledem k tomu, že pro každou velikost každého fontu potřebujeme samostatný soubor *.pk*, mohou zabírat velký objem disku a proto se uchovávají pouze dočasně.
- *.mf* – metafonty. Metafont je grafický programovací jazyk široce používaný k definici fontů (i když jej lze použít i pro jinou grafiku). Oproti fontům Type 1 a TrueType má řadu výhod. Zásadní nevýhodou je, že nejsou tak rozšířené jako fonty TrueType a Type 1. (A také to, že se příliš nehodí k WYSIWYG publikování – což ovšem není problém, pokud k publikování používáte TeX.)

Dále je dobré orientovat se v adresářové struktuře TeXu. Uvádíme hlavní adresáře, o kterých byste měli mít povědomí:

- *$TEXMF/fonts* – hlavní adresář fontů.
- *$TEXMF/fonts/type1* – adresář s fonty Type 1.
- *$TEXMF/fonts/type1/foundry* – adresář s definičními soubory fontů z dané skupiny.
- *$TEXMF/fonts/type1/foundry/fontname* – obsahuje font daného názvu. Název je obvykle v normální angličtině a nepoužívá kryptografické názvy TeXu.
- *$TEXMF/fonts/afm/foundry/fontname* – obsahuje *afm* soubory fontu z dané skupiny.
- *$TEXMF/fonts/tfm/foundry/fontame* – analogicky k adresáři *afm*, obsahuje ale soubory *tfm*.
- *$TEXMF/fonts/vf/foundry/fontname* – podobně jako výše, obsahuje ale virtuální fonty.
- *$TEXMF/fonts/source/foundry/fontname* – podobně jako výše, obsahuje ale metafonty.
- *$TEXMF/dvips/config/psfonts.map* – mapa fontů pro dvips. Funkcí i formátem se tento soubor podobá souboru *Fontmap* v ghostscriptu.
- *$TEXMF/tex/latex/psnfss* – tady jsou umístěny definiční soubory všech fontů.

Přidání fontů Type 1

Pojmenování fontů

Nejprve je nutné fonty správně pojmenovat. Návod k pojmenovávání fontů naleznete v dokumentaci programu *fontinst* (měl by být v podadresáři *fontinst* adresáře s dokumentací TeXu). Z dlouhého výkladu uděláme stručný výtah a zjistíme, že název je konstruován jako FNW{V}E{n}, kde:

- F je jednoznaková zkratka typu písma (m = monotype, p = adobe, b = bitstream, f = free)
- N je dvouznaková zkratka názvu písma (například ag = „Avant Garde“)
- W je šířka fontu (r = normální, b = tučné, l = zúžené, d = polotučné)
- V je nepovinný sklon (i = kurziva, o = šikmé)
- E je zkratka kódování (téměř vždy 8a, což je standard kódování adobe)
- n je nepovinná varianta šířky (n =narrow)

Písmo Adobe Garamond demibold je tedy pojmenováno jako *pgad8a*.

Tvorba virtuálních fontů a metrik

Nyní můžete spustit *fontinst* takto: *latex `kpsewhich fontinst.sty`*, po výzvě zadáte *\latinfamily{font_name}{}\bye*, kde *font_name* jsou první tři znaku názvu souboru s fontem (tedy například *pad* pro Adobe Garamond). Nyní *fontinst* vygeneruje řadu souborů – font description, property list a virtual property list. Dále vygeneruje řadu *.mtx* souborů. *Fontinst* sice tyto soubory vytvoří, nemusíte je ale použít. Potřebujete zkonvertovat property list a virtual property list na metriky a virtuální fonty. K tomu slouží nástroje *vptovf* a *pltotf*:

```
for X in *.pl; do pltotf $X; done
for X in *.vpl; do vptovf $X; done
```

Pak smažte staré soubory *vpl*, *pl* a *mtx*.

Konfigurace dvips

Dále musíte upravit konfigurační soubor *dvips*, *psfonts.map*. Nejlepší způsob, jak si popsat jeho formát, bude příklad:

```
marr8r          ArialMT <8r.enc <farr8a.pfa
marbi8r         Arial_BoldItalicMT <8r.enc <farbi8a.pfa
```

```
marb8r          Arial_BoldMT <8r.enc <farb8a.pfa
marri8r         Arial_ItalicMT <8r.enc <farri8a.pfa
marr8rn         Arial_Narrow <8r.enc <farr8an.pfa
```

Údaj *8r.enc* informuje *dvips* o používaném schématu kódování (ve všech příkladech je to *8r.enc*, protože takto *fontinst* vytváří virtuální fonty). První sloupec udává název fontu v TeXu. Druhý sloupec je skutečný název fontu, který je zapsán v souboru s fontem (tento název je možné zjistit otevřením souboru *afm* v textovém editoru a hledáním direktivy *FontName*). Poslední sloupec je pak definiční soubor fontu. Není nutné uvádět cesty, TeX ví, kde má hledat.

Testování fontu

Spusťte LaTeX na dokument typu *\documentclass{article} \begin{document} \usefont{T1}{pga}{m}{n}\selectfont \huge Testing a new font \dots the quick red fox jumped over the lazy brown dogs \end{document}*, kde místo *pga* uvedete název vašeho fontu. Pokud to funguje, jste téměř hotovi. Nyní zbývá umístit všechny soubory do správných adresářů (viz výše) a spustit *texconfig rehash*.

Vytvoření souboru .sty

Aby se vám fonty snáze používaly, můžete chtít vytvořit soubor *.sty*. Jako vzor použijte soubory v *$TEXMF/tex/latex/psnfss*.

Kde získat fonty pro Linux

TrueType

Komerční programy

Fonty TrueType se dají snadno získat, celou řadu typicky obsahují programy jako Microsoft Word nebo Word Perfect. Koupě programu Word Perfect je jednoduchá metoda, jak získat spoustu fontů. (A pokud chcete ušetřit, stačí koupit nějakou starou verzi Word Perfectu pro Windows. Přečíst fonty z instalačního CD není žádný problém.)

Lucovy stránky

Stránky Luca Devroye (http://cgm.cs.mcgill.ca/~luc/originalfonts.html) obsahují odkazy na různé servery se zdarma distribuovanými fonty. Důležité je, že jde skutečně o zdarma distribuované fonty, nejde o „warez“.

Webové stránky s fonty

Existuje několik stránek, které nabízejí fonty ke stažení zdarma. Odkazy na řadu archivů najdete například na adrese http://www.freewareconnection.com/fonts.html.

Výrobci

Řada výrobců prodává fonty TrueType. Většina z nich je ale poměrně drahá a za stejné peníze rozumněji pořídíte fonty Type 1. Více v části věnované fontům Type 1. Jedna možnost, jak koupit levné fonty TrueType, je adresa http://www.buyfonts.com/. Před koupí levných fontů si přečtěte kapitolku o etice.

Fonty Type 1 a Metafont

Práce s formáty Mac a Windows

Řada výrobců dodává fonty pro uživatele Windows a Maců. Občas tak může docházet k problémům. Typicky se s „fonty pro Windows" pracuje velmi snadno, protože jsou zabaleny jako soubor *zip*. Jediné, co je nutné udělat, je konverze souboru *pfm* na *afm* (pomocí *pfm2afm*).

Fonty pro Macintosh jsou problematičtější, protože jsou typicky ve formátu *.sit.bin* – tedy v archivu *stuffit*. Bohužel, pro Linux neexistuje žádný nástroj, který by uměl rozbalit novější verze archivu *stuffit*. Jediná možnost je spustit *Executor* (emulátor Macu), nebo zkusit *stuffit* v *dosemu* nebo ve *Wine*. Po rozbalení souboru *sit.bin* je možné fonty zkonvertovat nástrojem *t1unmac*, který je součástí balíku *t1utils*.

Někteří dodavatelé bohužel nabízejí fonty pouze ve formátu Macintosh (archivy *stuffit*). Podle Luca Devroye naštěstí většina z nich nabízí fonty Type 1 i ve formátu pro Windows.

Zdarma dostupné

Ctan (http://www.ctan.org/) nabízí řadu dobrých fontů, mnoho z nich je zdarma. Většina je ve formátu Metafont, některé jsou ale Type 1. Bluesky (http://www.bluesky.com/) nabízí zdarma Type 1 verzi fontů Computer Modern. (Jejich kvalita je vynikající, cokoliv ve srovnatelné kvalitě a úplnosti by stálo kolem 500 dolarů.)

Stránky Luca Devroye (http://cgm.cs.mcgill.ca/~luc/originalfonts.html) obsahují odkazy na různé servery se zdarma distribuovanými fonty. Důležité je, že jde skutečně o zdarma distribuované fonty, nejde o „warez".

URW uvolnil standardní postscriptové fonty používané ve většině tiskáren. Jde o docela kvalitní fonty.

Další zdarma dostupné a sharewarové fonty nabízí Walnut Creek Archive (ftp://ftp.cdrom.com/pub/os2/fonts/). Některé z nich jsou zjevně vykradené (a v nevysoké kvalitě). Pokud součástí fontu není i nějaká distribuční licence, jde obvykle o vykradený font. Různé fonty Type 1 nabízí i Winsite (http://www.winsite.com/win3/fonts/atm/). Bohužel, u některých fontů jsou chyby v souborech *afm* a chybí kerning. (Soubory *afm* je možné opravit editací části *FontName* v souboru, která musí odpovídat názvu fontu v definičním souboru fontu. Vytvoření kerningových dvojic je pak docela mimo možnosti tohoto dokumentu.)

Na stránkách Luca Devroye (http://cgm.cs.mcgill.ca/~luc/) najdete několik fontů, které sám navrhl, dále řadu odkazů a nesmírně zajímavé informace na typografická témata. Jde o stránky, které byste rozhodně neměli minout.

Komerční fonty

Cena versus kvalita: Proč kupovat drahé fonty?

Ptáte se – proč jsou některé fonty hrozně drahé, a jiné levné? Jde většinou o „standardní postscriptové fonty", které jsou součástí většiny postscriptových tiskáren. Další známá otázka – proč kupovat dražší fonty? Myslím si, že pro příležitostného uživatele stačí levnější fonty, které bývají často součástí různých CD. Pokud ale používáte fonty pro „opravdovou práci", nebo jste prostě blázen do typografie, pak se bez vysoce kvalitních fontů neobejdete. Některé velmi kvalitní fonty jsou zdarma (například Computer Modern), jiné pak stojí obrovské peníze.

Výhoda levnějších fontů je zjevná – jsou levné. Kvalitní fonty mají ale také své výhody.

- *Etické hledisko*: Levnější fonty jsou často vykradené. Návrh kvalitního fontu vyžaduje dlouhou dobu a zkušeného návrháře. Fonty prodávané pod jeden dolar nejsou téměř jistě ori-

ginální. CD se spoustou různých fontů jsou téměř vždy různě vykradené (výjimkou jsou CD některých výrobců, které ale stojí pár stovek dolarů). Vykradené fonty mají typicky výrazně nižší kvalitu než originály.

- *Úplnost*: Velmi kvalitní fonty (například od Adobe) obsahují různé varianty s řadou pěkných doplňků, rozšiřujících rodinu písma. Často obsahují tučné a polotučné varianty, kurzivu, různé typy kapitálek a slitků.
- *Kvalita*: Řada zdarma šířených fontů a různě vykradených fontů postrádá základní funkce jako párování (kerning) nebo slitky. Jde typicky o levné kopie. Naproti tomu renomovaní návrháři pečlivě studují původní návrh a zpracovávají jej podle svých nejlepších možností.
- *Autenticita*: Autor písma Adobe Garamond (konkrétně Robert Slimbach) pečlivě studoval původní návrhy Clauda Garamonda. Renomovaní návrháři vždy stavějí své návrhy na pečlivém průzkumu, nestačí jim pouze něco stáhnout na Internetu a upravit to Fontographerem.

Cena

- Dobrý zdroj CD s různými fonty Type 1 v rozumné kvalitě je Bitstream (http://www.bitstream.com/). Nejznámějšími produkty jsou 250 font CD (http://www.bitstream.com/products/world/font_cd/bits_collection.html) a 500 font CD (http://www.bitstream.com/products/world/font_cd/500_cd.html). To druhé stálo v době vzniku tohoto textu 50 dolarů. Většina fontů v produktech Corel je licencována právě od Bitstreamu.
- Matchfonts (http://www.matchfonts.com/) nabízí poněkud dražší fonty, distribuované v „balících“ po osmi za 30 dolarů. Nabízejí některé pěkné kaligrafické fonty. Všechny jsou dodávány ve snadno použitelném formátu (ATM fonty pro Windows se dodávají v .exe souborech, ale nenechejte se příponou zmást – jde o běžné zip archivy).
- EFF (http://www.buyfonts.com/) prodává fonty TrueType po dvou dolarech. Nabízejí také „profesionální“ fonty PostScript aTrueType po 16 dolarech.

Kvalita

- Adobe nabízí několik velmi kvalitních fontů na http://www.adobe.com/type/. Některé jsou velmi drahé, ale nabízejí i cenově dostupnější – viz http://www.adobe.com/type/collections.html. Adobe nabízí jedny z nejúplnějších rodin fontů na trhu, například Garamond (http://www.adobe.com/type/browser/P/P_912.html), Caslon (http://www.adobe.com/type/browser/P/P_180.html) a různé jejich mutace.
- Berthold Types Limited (http://www.bertholdtypes.com/) nabízejí různé kvalitní fonty. Některé z nich prodává i Adobe, za stejných cenových podmínek je však nabízí přímo Berthold.
- ITC (http://www.itcfonts.com/) nabízí několik kvalitních fontů (některé z nich nabízí ve svých produktech i Corel). Celé rodiny fontů stojí kolem 100-180 dolarů. Dodávají se ve formátech Type 1 i TrueType, rozumnější je zvolit distribuci pro Windows, protože formát pro Macy se v Linuxu zpracovává hůře.
- Známým dodavatelem fontů je Linotype (http://www.linotypelibrary.com/), nabízí fonty legendárních návrhářů, jako byl Herman Zapf (to je ten, po němž se jmenuje písmo „Zapf Chancery“, navrhl například i Palatino).
- Monotype (http://www.monotype.com/) je autorem většiny fontů používaných v produktech Microsoft.

- Tiro Typeworks (http://www.portal.ca/~tiro/) prodávají kvalitní i když poněkud drahé fonty. Jejich fonty jsou velmi úplné, obsahují slitky, kapitálky, nadpisové fonty a podobně. Nabízejí dokonce formát přímo pro UNIX – což je příjemné překvapení poté, co všude vidíme volbu „Windows nebo Mac“.

Další odkazy

Odkazy na spoustu dalších stránek naleznete na stránkách Luca Devroye, http://cgm.cs.mcgill.ca/~luc/.

Užitečné programy pro Linux

Pro práci s fonty v Linuxu existuje řada programů. Některé z nich jsou téměř nezbytné.

- *chkfontpath* je nástroj pro práci s konfiguračními soubory *xfs*.
- DTM, Definitive Type Manager, http://www.debian.org/~fog/dtm/, je nástroj pro globální správu fontů. Jedná se o vývojovou verzi.
- *fontinst*, http://www.tug.org/applications/fontinst/index.html, je nástroj usnadňující instalaci fontů Type 1 do LaTeXu.
- *freetype*, http://www.freetype.org/, je TrueType knihovna dodávaná s většinou distribucí Linuxu.
- *Ghostscript*, http://www.cs.wisc.edu/~ghost/, je program pro tisk v Linuxu. Verze dodávaná s Linuxem je GNU ghostscript. Jde o starší verzi Alladin ghostscript (jejichž předchozí verze byly licencovány pod GPL).
- *pfm2afm*, http://pegasus.rutgers.edu/~elflord/font_howto/pfm2afm.tgz, je nástroj pro konverzi metrikových souborů *pfm* používaných ve Windows na soubory *afm* používané v Linuxu. Je založen na původní verzi z archivu CTAN s úpravami Roda Smithe, které umožňují překlad nástroje pod Linuxem.
- *mminstance* a *t1utils*, http://www.lcdf.org/~eddietwo/type/, jsou dva balíky pro práci s fonty Type 1. *mminstance* slouží k práci s fonty *multiple master* od Adobe. *t1utils* je sada nástrojů pro konverzi mezi různými formáty fontů Type 1.
- *ttf2pt1*, http://quadrant.netspace.net.au/ttf2pt1/, je nástroj pro konverzi fontů TrueType na Type 1. Je užitečný, pokud používáte aplikace, které pracují pouze s fonty Type 1.
- *ttfps*, ftp://ftp.dcs.ed.ac.uk/pub/jek/programs/ttfps.tar.gz, konvertuje fonty TrueType na Type 42.
- *ttfutils*, http://pegasus.rutgers.edu/~elflord/font_howto/ttfutils-0.2.tar.gz, je balík nástrojů pro práci s fonty TrueType. Vyžaduje přítomnost programu *ttf2pt1*. Velmi užitečný, ne-li nezbytný.
- *type1inst*, ftp://ftp.metalab.unc.edu/pub/Linux/X11/xutils/, základní balík pro instalaci fontů Type 1. Významně usnadňuje instalaci.
- *xfstt*, ftp://ftp.metalab.unc.edu/pub/Linux/X11/fonts/, font server s podporou TrueType. Velmi šikovný, lepší volba je ale *xfs*.
- *xfsft*, http://www.dcs.ed.ac.uk/home/jec/programs/xfsft/, font server, je součástí *xfs*.
- *x-tt*, http://hawk.ise.chuo-u.ac.jp/student/person/tshiozak/x-tt/, font server navržený pro práci s korejskými a japonskými fonty.

Etické a licenční problémy

Licencování fontů je velmi sporný problém. Je pravda, že existuje spousta *zdarma šířených* fontů, avšak pokud neobsahují vlastní licenční podmínky, je velmi vysoká pravděpodobnost, že se jedná o fonty nějakým způsobem „vykradené". Celý problém se komplikuje díky právní úpravě ohledně intelektuálního vlastnictví ve vztahu k fontům. V USA v zásadě platí, že *soubory* s fonty jsou chráněny autorským právem, zatímco samotný *vzhled* fontu nikoliv. Jinak řečeno – je nelegální font distribuovat v binární podobě, avšak je naprosto legální provést „reverzní překlad" fontu jeho vytištěním na papír a návrhem křivek, které budou výtisku odpovídat. Takto vytvořené fonty jsou typicky levné nebo zdarma, jejich kvalita ale bývá nízká. Tyto fonty bývají společně s pirátskými verzemi fontů distribuovány na laciných CD se spoustou fontů. Nelze jednoduše poznat, zda se jedná o pirátský font, nebo o „reverzní překlad". Díky této situaci je nesmírně komplikované jakékoliv (legální) šíření domněle zdarma distribuovaných fontů.

Jednou z nejnepříjemnějších věcí na tomto fontovém pirátství je, že se tím znehodnocuje práce návrhářů originálních fontů. Pirátské fonty jsou typicky distribuovány na CD po tisících, bez zmínky o původních autorech. Naproti tomu u legálně šířených fontů bývají uvedeni původní návrháři.

Na tento problém existuje celá řada rozdílných názorů. Na stránkách http://www.typeright.org/ najdete vysvětlení problematiky intelektuálního vlastnictví. Opačný názor prezentuje Southern Software, http://www.ssifonts.com/ – jejich fonty si ale nekupujte. Fonty Type 1 nabízené touto společností (nekvalitní kopie fontů Adobe) neobsahují AFM a jsou tedy nepoužitelné.

Další názory na téma intelektuálního vlastnictví ve vztahu k fontům najdete na http://www.faqs.org/faqs/fonts-faq/part2/ a http://cgm.cs.mcgill.ca/~luc/. Tyto odkazy představují méně extrémní skupinu názorů.

Odkazy

Informace o fontech

- Stránky Roda Smithe, http://www.rodsbooks.com/, obsahují řadu informací o fontech a tisku v Applixware a Word Perfectu.
- Stránka Johna McLaughlina, http://www.mindspring.com/~john_mcl/adding_fonts.html, popisuje nastavení fontů ve Star Office.
- Stránka Jima Landa, http://www.geocities.com/SiliconValley/5682/postscript.html, obsahuje odkazy na informace o postscriptu a fontech.
- Spoustu otázek ohledně fontů řeší http://www.faqs.org/faqs/fonts-faq/.
- Stránky Luca Devroye, http://cgm.cs.mcgill.ca/~luc/, obsahují tolik informací o fontech a dalších věcech, že by to potopilo loď. Autor stránek navrhl řadu zdarma distribuovaných fontů a nabízí spoustu zajímavých odkazů, informací a komentářů.
- Dokument Font Deuglification HOWTO, http://www.linuxdoc.org/HOWTO/mini/-FDU.html, hovoří o TrueType fontech v Linuxu. Nejlepší z „TrueType HOWTO" dokumentů.
- TT-Debian mini-HOWTO, http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/otherformats/html_single/TT-Debian.html, hovoří o instalaci fontů TrueType v Debianu. Nezbytné čtení pro uživatele distribuce Debian, vhodné také pro všechny, kteří používají xfs bez podpory TrueType fontů.

- The (preliminary) TrueType HOWTO, http://www.moisty.org/~brion/linux/TrueType-HOWTO.html, neúplný dokument z června 1998. Uvádíme jej zde pouze pro úplnost.
- TT-Xfree86 mini-HOWTO, http://www.sfu.ca/~yzhang/linux/truetype/, poněkud zastaralý dokument, platný pro RedHat 5.x.

Informace o postscriptu a tisku

- Standard PostScript popisuje http://www.adobe.com/print/postscript/main.html.
- Domovská stránka Ghostscriptu, http://www.cs.wisc.edu/~ghost/, obsahuje řadu informací a nejnovější tiskové ovladače.
- Stránky Jima Landa, http://www.geocities.com/SiliconValley/5682/postscript.html, obsahují řadu odkazů na stránky věnované PostScriptu a fontům.
- Printing FAQ Christophera Brownea, http://www.hex.net/~cbbrowne/printing.html.

Slovníček

afm	Adobe Font Metric. Tento soubor obsahuje informace o šířkách a mezerách vztahujících se k fontu. Nedefinuje tvary znaků.
anti-aliasing	Technika pro vykreslování fontů na zařízeních s malým rozlišením (např. monitorech). Problém při vykreslování spočívá v tom, že znak je tvořen křivkami, ty se ale vykreslují po bodech. Jednoduché řešení spočívá v tom vykreslit černě všechny body uvnitř znaku, ostatní nechat bílé. Tím se ale neřeší problém bodů *na hranici* znaku. Chytřejší algoritmy vykreslují hraniční body různými odstíny šedi – a to je právě anti-aliasing.
bdf fonty	Rastrové fonty používané v X Window.
bezpatkové písmo (sans-serif)	Písma bez patek (*sans* je francouzsky *bez*). Tato písma mají výraznější vzhled a používají se k sazbě titulků. I když učebnice typografie udávají jejich použití jen pro titulky, dají se použít i jinak. Některá bezpatková písma jsou navržena s ohledem na čitelnost a ne na výraznost. Používají se například k sazbě katalogů nebo marketingových materiálů. Písmo Verdana používá Microsoft kvůli dobré čitelnosti při malé velikosti na zařízeních s nízkým rozlišením. Mezi známá bezpatková písma patří Lucida Sans, MS Comic Sans, Avant Garde, Arial, Verdana a Century Gothic.
bitmap fonts	viz *rastrové fonty*
didone	viz *modern*
DPI	Dots Per Inch, bodů na palec. Jednotka rozlišovací schopnosti zařízení. Monitory typicky zobrazují 75 – 100 DPI, moderní tiskárny 300 – 1 200 DPI.
expert fonts	Kolekce doplňujících znaků, které rozšiřují font. Zahrnují obvykle kapitálky, ornamenty, speciální ligatury a číslice s různou šířkou. Jsou součástí většiny fontů Adobe.
font server	Program, který zpřístupňuje fonty v Xfree86.
glyph	Zajímavé slovo označující tvar. Jedná se o komponenty, z nichž je tvořen obrys znaku. Například tečka nad *i* je jeden glyph, dalším je svislá čára, dalšími jsou patky.

kerning	viz *párování*
ligatura	viz *slitek*
medievalová písma	Tradiční skupina písem. Jsou založeny na návrzích až z 15. století. Jsou vynikající pro sazbu dlouhých textů, např. knih. I když jejich návrh vychází z velmi starých tradic, některá z nich vznikla poměrně nedávno. Například písmo Goudy Old Style navrhl Goudy počátkem 20. století. Klasickými představiteli jsou Goudy Old Style, Garamond a Caslon.
metafont	Grafický jazyk používaný k definici fontů. Metafont má řadu sympatických funkcí, hlavní z nich je ta, že změna velikosti fontů nemusí být lineární. Znamená to, že písmo o velikosti 17 bodů není prostou zvětšeninou písma o velikosti 10 bodů. Před příchodem technologie *multiple master* od Adobe byla tato funkce jedinečná právě pro metafont. Výhodou tohoto jazyka je vytváření velmi kvalitních fontů, nevýhodou je pomalé generování rastrů, takže nejsou vhodné pro WYSIWYG publikování.
metrika	Obsahuje informace o mezerách mezi znaky. Metrika je něco jako rámeček, do nějž lze znak vykreslit. Metrika je nezbytná pro potřeby sazby znaků na stránce, definice tvarů sama o sobě k sazbě není zapotřebí. Typicky proto fonty s proměnnou šířkou obsahují jak definice tvarů, tak definice metrik. Metrika navíc obsahuje informace o kerningu.
modern fonts	Fonty založené na návrzích z 19. století a novější. Mají výrazný vzhled díky svislému zvýraznění. Mají výraznější charakter a vzhled než písma medievalová a tranzitivní, stále si však zachovávají jistou formálnost. Nejsou vhodná pro delší dokumenty, používají se pro výraznější kratší texty. Typickým písmem této kategorie je Bodoni.
old style fonts	viz *medievalová písma*
patkové písmo	Písmo s krátkými čárkami (patkami) na koncích znaků, které typicky zvyšují čitelnost písma. Tento typ písma se velmi obtížně zobrazuje na zařízeních s malou rozlišovací schopností, zejména při malé velikosti písma. Proto bývají na těchto zařízeních mnohdy čitelnější bezpatková písma. Některá patková písma (tzv. moderní) nejsou vhodná pro sazbu dlouhých dokumentů.
pcf	Rastrové fonty používané v systému X Window.
písmo s plochou patkou	Též *slabserif* nebo *egyptienka*. Skupina písem, jejichž patky vypadají jako bloky. Jsou obvykle, ale ne vždy, velmi dobře čitelná. Působí výrazným vzhledem. Známými příklady jsou písma Clarendon, New Century Schoolbook a Memphis.
PostScript	Programovací jazyk určený k tvorbě stránek. Jde o ochrannou známku autora, společnosti Adobe, ale zároveň o standard ISO. Ke zobrazení PostScriptu je nutný interpret. Tím může být počítačový program, například Ghostscript, nebo to přímo umějí některé tiskárny.
rastrové fonty	Tyto fonty jsou jednoduše sbírka bodů. Každý znak je reprezentován maticí bodů. Díky tomu jsou rastrové fonty závislé na rozlišení zobrazovacího zařízení a týž rastrový font tedy nejde použít na obrazovce i na tiskárně. Příkladem rastrových obrazovkových fontů jsou *pcf* a *bdf* fon-

	ty v systému X Window. Tiskové rastrové fonty jsou například PK fonty v TeXu.
sans serif	viz *bezpatkové písmo*
serif	viz *patkové písmo*
slab serif	viz *písmo s plochou patkou*
slitek	Speciální znak sloužící ke zobrazení dvojice znaků. Nejlépe se vysvětluje na příkladu. Při vykreslení dvojice znaků *fi* tečka nad *i* koliduje s vrcholem *f*, a horní patka *i* koliduje se střední linkou *f*. Slitek *fi* je jediný znak, používaný jako náhražka dvojice za sebou následujících znaků *f* a *i*. Dalšími slitky jsou třeba *fl*, *ffi* a *ffl*. Většina fontů obsahuje pouze slitky *fi* a *fl*. Další slitky bývají součástí expertních fontů.
párování	Též *kerning*. U fontů s proměnnou šířkou jsou mezi různými znaky různé mezery. Metrika fontu definuje mezery mezi různými dvojicemi znaků, takzvané *kerningové páry*.
tranzitivní písma	Písma s modernější podobou než klasická medievalová písma. Většina z nich je velmi dobře čitelná. Příkladem jsou písma Baskerville a Times Roman.
Type 1	Typ fontů navržený společností Adobe. Je podporován většinou linuxových aplikací, protože jde o typ již dlouho podporovaný standardem PostScript a X servery. Distribuují se v řadě různých formátů. Na Unixech typicky jako *afm* soubor s metrikou a *pfb* (printer font binary) nebo *pfa* (printer font ascii) soubor definující tvary.
Type 3	Formát podobný formátu Type 1. Přípony jsou obdobné jako u Type 1, není však podporován v X Window a proto s ním pracuje jen velmi málo linuxových aplikací.

Fonty TrueType v XFree86 4.0.x

Originál: http://tldp.org/HOWTO/mini/TT-XFree86.html

Tento dokument popisuje, jak použít fonty TrueType s XFree86 verze 4.0.x.

Postup

Sežeňte si potřebné TrueType fonty (například je stáhněte ze serveru, který je nabízí), a jako *root* zadejte následující příkazy:

```
mkdir /usr/X11R6/lib/X11/fonts/Truetype
cp /home/uživatel/fonty/*.ttf /usr/X11R6/lib/X11/fonts/Truetype
cd /usr/X11R6/lib/X11/fonts/Truetype
ttmkfdir > fonts.scale
mkfontdir
```

Pokud máte na některém diskovém oddílu funkční instalaci Windows a chcete je použít, *musíte* je ponechat na daném umístění – jejich zkopírování do oddílu s Linuxem *není legální*. Nejprve si zjistěte, jak oddíl s Windows najít a zpřístupnit (většina distribucí to sama udělá při instalaci), a pak jako *root* zadejte:

```
cd /etc/X11
oblíbený_editor XF86Config
```

V některých distribucích systému X Window se používá konfigurační soubor s přidaným číslem v názvu (např. *XF86Config.4*). Musíte zjistit, jaký soubor používá váš systém. Zkuste příkaz *less /var/log/Xfree86** a editujte ten soubor, který je uveden v logu.

V editovaném souboru najděte sekci *Files* a na řádku *FontPath* doplňte na konec seznamu cestu */cesta_k_oddílu_s_windows/windows/fonts*.

```
cd /cesta_k_oddílu_s_windows/windows/fonts
ttmkfdir > fonts.scale
mkfontdir
```

Tím byste měli v systému X Window získat možnost používat všechny fonty z Windows. Pokud budete chtít kdykoliv přidat další fonty, prostě je zkopírujte do výše uvedených adresářů a celý postup zopakujte.

Netscape má s těmito fonty potíže, zkontrolujte nastavení „Allow Scaling“ v dialogu „Fonts“ a nenechejte se zmást tím, že nabízí velikosti fontů pouze 0 a 12. Mozilla ani Opera podobným problémem netrpí.

Program *mkfontdir* by měl být součástí vaší distribuce XFree 4.0.*x*.

Pokud nemáte program *ttmkfdir*, můžete jej získat na adrese http://www.joerg-pommnitz.de/-TrueType/ttmkfdir.tar.gz.

Aktuální verzi XFree86 najdete na adrese http://www.xfree86.org/.

KAPITOLA 11

Modemy

Originál: http://tldp.org/HOWTO/Modem-HOWTO.html

Úvod

DSL, kabelové a ISDN modemy najdete v jiném dokumentu

Tento dokument hovoří o klasických analogových modemech připojovaných na sběrnice ISA, PCI nebo USB (u těch jsme velmi struční). Pokud vás zajímají jiné typy modemů, zkuste dokumenty:

- *DSL-HOWTO*
- *ADSL-Bandwidh-Management-HOWTO*
- *Cable-Modems-HOWTO*
- *SuSE ISDN Howto (nejde o dokument Dokumentačního projektu, viz http://brenner.chemie-technik.uni-dortmund.de/doc/sdb/en/html/isdn.html)*
- *úvod do ISDN, http://public.swbell.net/ISDN/overview.html*
- *dokumentaci k ISDN v adresáři isdn dokumentace jádra*
- *http://public.swbell.net/ISDN/overview.html*
- *Příloha B – Digitální modemy*

Dále nemluvíme o PCMCIA a PPP

Informace o modemech připojovaných k PCMCIA sběrnici najdete v dokumentu PCMCIA-HOWTO. Dál v tomto dokumentu nehovoříme o PPP (které se používá k připojení k Internetu přes modem) ani o komunikačních programech. Pouze si řekneme, jak lze komunikační programy použít k otestování modemu. Pokud se chcete modemem připojit k Internetu, musíte si nastavit také PPP. K tomu existuje řada dokumentace, včetně PPP-HOWTO. Další dokumentaci najdete v */usr/doc/ppp*, */usr/share/doc/ppp*, nebo tak nějak.

Co je to modem?

Modem je zařízení sloužící k přenosu digitálních signálů přes normální telefonní linky, které nejsou pro digitální signály určeny. Pokud by byly všechny telefonní linky digitální, nepotřebovali bychom modemy[1]. Umožňuje vašemu počítači připojit se a komunikovat se zbytkem světa. Když používáte modem, typicky pomocí komunikačního programu nebo prohlížeče modem aktivujete a připojíte se k telefonní lince. Zkušenější uživatelé mohou modem nastavit tak, aby se k nim mohli dovolat jiní uživatelé a používat jejich počítač. Tomuto procesu se říká *dial-in*[2].

[1] Pozn. překladatele: Pozor – digitální linkou je v tomto kontextu myšlena zejména ISDN linka. Výroky Českého Telecomu o „úplné digitalizaci sítě“ a „přechodu na digitální ústředny“ s tím nemají nic společného – telefonní linka vedoucí k vám domů je (skoro určitě) pořád analogová.

[2] Pozn. překladatele: Kvůli omezené kapacitě knihy bylo toto téma z českého překladu vypuštěno, originální anglická verze dokumentu na adrese o něm hovoří zejména v kapitolách 3, 12 a 13.

Existují čtyři základní typy modemů: externí, USB, interní a vestavěné. Externí a USB modemy najdete na stole vedle počítače, zbývající dva typy nenajdete, protože jsou uvnitř počítače. Externí modemy se připojují k sériovému portu počítače, USB modemy pak k USB portu. Interní modem je karta, která se vkládá do počítače. Vestavěný modem je nedílnou součástí základní desky počítače. Chová se podobně jako interní modem, nejde jej ale vymontovat nebo nahradit. V současné době jsou interní modemy zejména součástí laptopů. To, co zde budeme říkat o interních modemech, platí vesměs i pro vestavěné modemy.

Podrobnější popis rozdílů naleznete v části *Externí versus Interní*. Když používáte interní, vestavěný nebo USB modem, získáváte zároveň vyhrazený sériový port (který lze použít pouze pro tento modem a pro nic jiného). V Linuxu se sériové porty označují *ttyS0*, *ttyS1* atd. (což obvykle odpovídá označení COM1, COM2, atd. v DOSu a Windows). S novým systémem *devfs* jsou všechny sériové porty v adresáři */dev/ttys* a jmenují se *0*, *1* atd. Další informace viz část *Základní informace o modemech a sériových portech*.

Je v mém počítači modem?

Interní modem obvykle poznáte podle dvojice modulárních telefonních zásuvek. Měly by být hned vedle sebe a vypadají stejně jako zásuvky, do nichž doma připojujete telefon. Síťové karty používají podobné zásuvky, ale obvykle nemají dvě a jsou o něco širší, protože mají typicky 8 kontaktů. Existují i interní DSL „modemy“ a rovněž mají telefonní zásuvky, nejsou ale (alespoň v roce 2002) příliš běžné, většinou jsou externí.

Rychlá instalace

Instalace externího modemu

Pomocí příslušného kabelu připojte modem k volnému sériovému portu počítače. Ujistěte se, že víte, který je to port – většinou COM1 odpovídá *ttyS0*, COM2 *ttyS1* atd. Kontrolu můžete provést v konfiguraci BIOSu. Připojte napájení modemu. Další informace najdete v části *Společné pro všechny*.

Interní modemy (ISA, PCI a AMR)

Nejprve si ověřte, zda bude modem fungovat v Linuxu, protože momentálně (2002) řada těchto modemů nefunguje. Viz seznam modemů na adrese http://www.idir.net/~gromitkc/winmodem.html. Pokud je modem PnP a přímo jej podporuje sériový ovladač (jádra 2.4+), nemusíte konfigurovat nic, protože jej nakonfiguruje sériový ovladač Linuxu. Fyzickou instalaci provedete po odejmutí krytu počítače. Najděte volný slot, odstraňte záslepku na zadní straně počítače, opatrně vsuňte kartu do slotu a přišroubujte ji. Pokud nemáte winmodem (viz *Softwarové modemy*), možná jej nakonfiguruje přímo sériový ovladač a nebudete muset dělat nic. Měli byste to poznat podle hlášení při spouštění počítače (později je můžete zobrazit příkazem *dmesg*).

Interní modemy – ruční nastavení

Pokud modemu nebyl automaticky přidělen nějaký *ttySx* a přerušení (nebo pokud tyto hodnoty potřebujete změnit), musíte je nastavit ručně. Nejprve se musíte rozhodnout, které zařízení *ttySx* pro něj použijete. Zvolte takové, které není přiděleno sériovým portům. Pak nastane problém s přidělením přerušení a vstupně/výstupní adresy. Pro PnP modemy platí: Pokud BIOS už potřebné hodnoty fyzickému zařízení nastavil (což PnP BIOS udělá, pokud se domnívá, že nemáte PnP operační systém), pak musíte nastavené hodnoty přerušení a adresy zjistit a říct je *setserial*.

V jiných případech budete mít určitou možnost volby přerušení a adres (včetně případů, kdy lze změnit hodnoty nastavené BIOSem). Viz *Volba přerušení a adresy*. Pro ISA modemy je k dispozici standardní rozsah adres (odpovídající příslušným *ttySx*). Například můžete chtít použít zařízení

/dev/ttyS2 na adrese *0x3e8* s přerušením 11. PCI modemy používají jiné rozsahy vstupně-výstupních adres, které nekolidují s ISA zařízeními.

ISA modemy – jaké přerušení a adresy lze použít?

Máte-li starý modem s přepínači, podívejte se do návodu (případně přímo na přepínače, pokud jsou popsané). Pokud už BIOS provedl konfiguraci modemu, měli byste ji zjistit příkazem *pnpdump --dumpregs*. Pokud chcete hodnoty nastavit nebo změnit, použijte *isapnp*. Pomocí *pnpdump* zjistíte, jaké změny jsou možné.

PCI i ISA modemy – setserial a nastavení sériového ovladače

Musíte zjistit, odkud se při startu systému spouští *setserial* a přidat řádek jako *setserial /dev/ttyS2 irq 5 port 0x0b8*. U *setserial* 2.15 a vyšších může a nemusí při jeho spuštění z příkazového řádku dojít k uložení parametrů do */etc/setserial.conf*, takže se pro příště zapamatují. Podrobnosti viz *Co to je setserial*.

Nastavení BIOSu pomocí Windows (poslední možnost)

Pokud je modem konfigurován BIOSem, můžete zkusit pomocí Windows 9x „donutit" BIOS nastavit určité adresy a přerušení. Mohou je zapsat do PnP paměti BIOSu, kde je pak budou používat jak Windows, tak Linux. Viz dokument *Plug-and-Play-HOWTO* a hledejte slovo „forced". Ve Windows 3.x můžete provést to samé pomocí ICU. Některé modemy mají možnost softwarově vypnout PnP (pomocí programu, dodávaného s modemem, který běží jen pod Windows).

Společné pro všechny

Připojte modem k telefonní lince. Nastavte komunikační program jako *minicom* nebo *ppp* program (jako *wvdial*). Nastavte rychlost sériového portu na několikrát vyšší, než je přenosová rychlost modemu. Viz *Tabulka rychlostí*, kde zjistíte, jaké jsou „nejlepší" rychlosti. Nastavte plné jméno sériového zařízení (např. */dev/ttyS1* nebo */dev/ttys/1*). Nastavte hardwarové řízení přenosu (RTS/CTS).

Minicom se snáze nastavuje a snáze se jím modem testuje. Pokud ale máte štěstí a *ppp* vám bude fungovat napoprvé, nemusíte se s *minicom*em zatěžovat. Pomocí *minicom*u se můžete přesvědčit, zda váš modem reaguje – po spuštění zadejte příkaz *AT*, stiskněte Enter a měli byste dostat odpověď *OK*, kterou poslal modem. Viz *Vytáčení s minicomem*.

Modemy a Linux

Externí versus interní

Modem může být interní nebo externí. Interní je nainstalován uvnitř počítače (kvůli jeho instalaci musíte odšroubovávat šroubky a podobně), externí se pouze připojí ke konektoru sériového portu. Interní modemy jsou levnější, méně náchylné na ztráty dat kvůli přeplnění bufferů, obvykle mají menší spotřebu a nezabírají místo na stole. Externí modemy se obvykle snáze instalují a mají jednodušší konfiguraci. Mají světýlka, která napovídají, co se zrovna děje a mohou posloužit při diagnostice potíží. Diagnostice rovněž napomáhá to, že sériový port a modem jsou fyzicky dvě samostatné komponenty. Externí modem lze jednoduše přenést k jinému počítači. Pokud je nutné modem resetovat vypnutím napájení (což bývá nutné jen zřídkakdy), pak kvůli vypnutí externího modemu nemusíte vypínat celý počítač.

Většina externích modemů neumí vypnout napájecí zdroj a tím pádem stále odebírají elektřinu, i když jsou vypnuté (pokud zdroj nevypnete ze zásuvky). Každý spotřebovaný watt vás stojí přes dolar za rok. Další nevýhodou externích modemů je to, že budete nuceni použít existující sério-

vý port, který neumí vyšší rychlost než 115 200 b/s (i když – konec roku 2000 – většina interních modemů to neumí taky, ale některé ano). Podrobnosti viz *Nelze nastavit dostatečnou rychlost*.

Interní modemy představují v Linuxu speciální problém, ale jinak pracují stejně dobře jako externí za předpokladu, že se vyhnete těm, které pracují pouze ve Windows. Jejich konfigurace může být velmi snadná (automatická), až hodně obtížná, v závislosti na modemu, znalostech uživatele a na možnosti získat o modemu informace – v tomto dokumentu nenajdete všechno. Některé (softwarové) modemy pracují jen ve Windows, protože v Linuxu pro ně neexistují ovladače. Pokud kupujete nový modem a nejste si jisti, že bude v Linuxu fungovat, zkuste se dohodnout, abyste jej mohli případně vrátit.

Většina moderních modemů je plug-and-play a při jejich konfiguraci mohou nastat různé situace:

- Vše nastaví sériový ovladač (pravděpodobné u PCI modemů)
- Nastavení provedete programem *isapnp*
- Necháte konfiguraci na BIOSu

Poslední dva případy mají své nevýhody. Dokumentace programu isapnp je velmi náročná, i když ji lze snáze pochopit po přečtení (dlouhého) dokumentu *Plug-and-Play-HOWTO*. Pokud chcete konfiguraci nechat na BIOSu, stačí mu říct, že nemáte operační systém s podporou PnP. BIOS ale nemusí provést nastavení správně a vy budete potřebovat zjistit, co vlastně nastavil. Viz *Jak je nastavený sériový port*.

Řada uživatelů Linuxu stále tvrdí, že je mnohem jednodušší si koupit a připojit externí modem. Pokud ale máte vhodný interní modem, nemusí to být o nic těžší.

Je zapotřebí ovladač?

Hardwarové modemy (včetně všech externích modemů) nepotřebují žádný speciální ovladač. Ovšem softwarové modemy (winmodemy, linmodemy) musí mít svůj ovladač (pokud ovšem pro Linux existuje). Sériový port, k němuž je modem připojen, ovladač potřebuje. Ten je k dispozici buď jako modul jádra, nebo je přímo součástí jádra. Sériové ovladače PCI modemů by se měly nainstalovat automaticky, protože je systém detekuje.

Softwarové modemy potřebují ke své činnosti příslušný program a samozřejmě ovladač. Ovladače pro Windows jsou programy **.exe* a v Linuxu nefungují. Musíte tedy použít linuxový ovladač. Viz *Softwarové modemy*.

Externí modemy

Externí PnP modemy

Řada externích modemů je označena jako „Plug-and-Play“, nicméně všechny by měly bez potíží fungovat i bez PnP. Zatímco sériový port je nutné nakonfigurovat (tedy nastavit přerušení a adresu, pokud standardní hodnoty nevyhovují), externí modem žádná taková nastavení nepoužívá. Prostě stačí modem připojit k sériovému portu.

Proč se externí modemy označují jako PnP, když je PnP nijak nenastavuje? Protože mají vestavěnou PnP identifikaci, kterou si může PnP operační systém (prostřednictvím sériového portu) přečíst. Takový operační systém pak ví, že k danému portu máte připojen modem a zná jeho identifikaci. Pokud jde o softwarový modem, může se systém pokusit najít odpovídající ovladač. Může také aplikaci říct, k jakému portu je modem připojen (například */dev/ttyS2* nebo *COM3*). Pokud ale PnP operační systém nemáte, budete muset aplikaci sami říct, k jakému sériovému zařízení je modem připojen. Některé aplikace se pokoušejí modem detekovat na různých portech.

Co znamenají světýlka na některých externích modemech

TM	Test Modem	
AA	Auto Answer	modem bude přijímat příchozí volání
RD	Receive Data, (RxD)	příjem dat
TR	Data Terminal Ready, (DTR)	počítač modemu říká, že je připraven
RI	Ring Indicator	signalizuje zvonění příchozího volání
OH	Off Hook	vyvěšeno (pokud nesvítí, modem zavěsil)
MR	Modem Ready, (DSR)	mode je připraven
EC	Error Correction	korekce chyb
DC	Data Compression	komprese dat
HS	High Speed	vysoká rychlost

Interní modemy

Interní modem je nainstalován přímo v příslušném slotu počítače. Existují modemy pro PCI sloty, modemy pro starší ISA sloty a softwarové AMR „modemy“, připojované do nových malých AMR slotů. Některé novější počítače vůbec nemají ISA sloty. Naopak AMR sloty mají jenom některé novější počítače. Zatímco externí modemy se (pomocí kabelu) připojují k existujícímu sériovému portu, interní modemy mají sériový port vestavěn. Jinak řečeno, modemová karta je zároveň sériový port i modem. Nastavení vstupně-výstupní adresy a přerušení pro sériový port se dříve provádělo pomocí přepínačů na kartě. PnP modemy (respektive sériový port takového modemu) se nenastavují pomocí přepínačů, ale pomocí konfiguračních příkazů, které obdrží po sběrnici počítače. Tyto příkazy může poslat PnP BIOS, program *isapnp* (jen pro ISA sběrnici), program *setpci* (jen pro PCI sběrnici) nebo u některých modemů novějším ovladačem sériového portu. Pokud konfiguraci neprovede sériový ovladač, můžete si vybrat, jak ji provedete:

- ISA – použijte *isapnp*, můžete jej automaticky spustit při startu systému.
- Nechejte nastavení na BIOSu a případně pak řekněte programu *setserial*, co je nastaveno.
- PCI – pomocí *lspci -vv* se podívejte na modem a pak jej nastavte programem *setpci*.

Zejména pokud máte PCI modem, podívejte se na část *Rychlá instalace*.

Softwarové modemy (winmodemy, linmodemy)

Úvodní informace

Softwarové modemy přenášejí část (nebo dokonce většinu) své práce na procesor počítače. K tomu je zapotřebí speciální program (ovladač modemu). Až do konce roku 1999 existovaly tyto ovladače pouze pro Windows a v Linuxu nefungovaly. Ještě horší ale bylo, že výrobci utajovali informace o tom, jak s modemem pracovat, takže nikdo nemohl ovladač pro Linux napsat (i když existovali dobrovolníci ochotní se toho ujmout).

Od té doby se situace trochu vylepšila a v současnosti (konec roku 2001) pro řadu těchto modemů existují i linuxové ovladače. Na tomto poli neexistují žádné standardy, takže různé typy modemů různých výrobců používají různé ovladače (pokud ovšem pod různými značkami není interně úplně stejné zařízení).

Dalším názvem pro softwarové modemy (používá jej Microsoft) je *driver-based modem*. Klasický hardwarový modem (který pracuje pod Linuxem) nepotřebuje žádný ovladač (využívá ale sériový ovladač Linuxu). Zhruba od poloviny roku 1998 jsou téměř všechny interní modemy softwarové.

Softwarové modemy se dělí na dvě skupiny: linmodemy a winmodemy. Winmodemy fungují pouze ve Windows. Linmodemy fungují i v Linuxu (i když původně to byly také winmodemy). Navíc „Winmodem" je dokonce obchodní značka jednoho z winmodemů, ale to nás nemusí v tomto dokumentu zajímat.

Linmodemy

Koncem roku 1999 došlo k tomu, že dva softwarové modemy dokázaly fungovat pod Linuxem, a tak vznikly linmodemy. Společnost Lucent Technologies (LT) neoficiálně uvolnila binární kód pro Linux, podporující většinu jejich PCI modemů. Společnost PC-TEL (Zoltrix) uvedla nový softwarový modem pro Linux. Tím začal zájem o zprovoznění winmodemů i v Linuxu. Existuje GPL ovladač pro čipset MD 563x společnosti Intel. V polovině roku 2001 se objevily ovladače pro Conexant HSF a HCF, Motorola SM56, ESS (pouze ISA) a IBM Mwate pro Thinkpad 600+. Kolik procent softwarových modemů funguje nyní (2001) v Linuxu? Nuže, existuje řada nepodporovaných čipů – Lucent/Agere AMR (Scorpio), 3COM/US Robotics, některé SmartLinky, Ambient HSP a asi i další. Právě teď už ale mohou být některé z nich podporované. Koncem roku 2001 to vypadalo tak, že byla podporována zhruba polovina softwarových modemů.

Upozorňujeme, že zjistit, zda váš modem je „linmodem", může a nemusí být jednoduché. Nejprve musíte zjistit čipset a jeho výrobce. Pouhá znalost typu a značky modemu stačit nemusí. Existují složité metody, jak to zjistit, například pomocí *lspci* a hledání výrobce čipu na základě identifikace modemu. K tomu je nutné prohledávání databází a hledání na Internetu, což nemusí být vždy jednoduché. Může se také stát, že po velkém úsilí zjistíte, že váš modem prostě podporován není. Podrobnosti viz dokument *Linmodem-HOWTO*.

Dokumentace a informace o Linmodemech

- *Linmodem-HOWTO*
- *Winmodems and Linux HOWTO*
- *http://linmodems.org/ je projekt na přetvoření winmodemů na linmodemy*
- *Conexant+Rockwell modem HOWTO*
- *http://www.idir.net/~gromitkc/winmodem.html je seznam modemů s informacemi o linmodemech*
- *PCTel HSP MicroModem Configuration mini-HOWTO*

Typy softwarových modemů

Existují dva základní typy softwarových modemů. V jednom z nich zajišťuje software prakticky veškerou práci. Ve druhém zajišťuje software pouze „řídicí" operace (tedy všechno kromě zpracování digitalizovaného signálu – viz dále). Protože řízení není zajištěno hardwarově, jde o takzvané „bezřadičové" modemy. První typ jsou plně softwarové modemy (někdy prostě jen „softwarové modemy").

V obou typech musí být analogový hardware generující elektrický signál posílaný na telefonní linku. Ten se generuje z digitalizovaného signálu – jakoby digitální obvody vytvořily řadu diskrétních bodů na papíře a modem jimi proložil plynulou křivku. Kromě toho je zapotřebí hardware konvertující příchozí analogový signál na digitální. To se provádí prostým A/D převodem, který provádí kodek (kodér-dekodér).

Pak je nutné digitalizovaný signál konvertovat na posloupnost bajtů. Tomu se říká demodulace, zatímco konverze bajtů na digitalizovaný signál je modulace. Modem ovšem nemůže přicházející bajty jednoduše posílat do počítače, musí provádět dekompresi, chybovou korekci a konverzi sériových dat na paralelní data přenášená sběrnicí. Analogicky to platí i pro odchozí směr.

Rozdíl mezi oběma typy softwarových modemů je v tom, kde se odehrává digitální modulace. V plně softwarovém modemu ji zajišťuje procesor počítače prostřednictvím tzv. HSP (Host Signal Processor). V bezřadičovém modemu modulaci provádí modem, ostatní činnosti pak procesor počítače. Mezi tyto ostatní činnosti spadá zpracování AT příkazů, komprese dat, chybová korekce a simulace sériového portu. I v plně softwarovém modemu stále zůstávají dvě operace, které musí řešit hardware – A/D převod příchozích dat a potlačení echa.

Je můj modem softwarový?

Jak zjistíte, že je určitý interní modem softwarový? Nejprve se podívejte, zda jeho „softwarovost" neprozrazuje přímo jeho název, popis nebo název ovladače pro Windows – hledejte termíny jako HSP (Host Signal Processor), HCF (Host Controlled Family), HSF (Host Signal Family), controllerless, host-controlled, host-based nebo soft-... modem. Pokud je to takový nějaký modem, bude fungovat pouze v případě, že pro něj existuje ovladač pro Linux. Dalším vodítkem může být cena, protože softwarové modemy jsou obvykle levnější.

Pokud neznáte model modemu, ale máte na počítači zároveň nainstalovány Windows, zvolte ikonu „Modem" v Ovládacích panelech. Pak zkuste najít modem v seznamu modemů (viz *Odkazy*). Pokud tento postup nefunguje (nebo jej nelze použít), zkuste se podívat na krabici, v níž byl modem zabalen, případně na návod k němu. Přečtěte si část jako „Minimální požadavky" nebo podobně.

Hardwarový modem funguje dobře i se starými procesory (například 386). Pokud tedy modem požaduje moderní procesor (tedy Pentium nebo jiný „rychlý" procesor, řekněme nad 150 MHz), pak jde asi o plně softwarový modem. Pokud vyžaduje pouze procesor 486 (a lepší), pak to bude asi bezřadičový modem. Špatné také je, pokud jsou požadovány Windows. Nicméně v takovém případě není vyloučeno, že pro modem existuje i linuxový ovladač.

Pokud u modemu není explicitně uvedeno, že vyžaduje Windows, mělo by jít o hardwarový modem. Pokud se uvádí, že modem je „designed for Windows", může to znamenat pouze to, že plně podporuje technologii Microsoft Plug-and-Play, což je v pořádku, protože Linux používá stejné specifikace PnP (i když v Linuxu je obtížnější konfigurace). Dále můžete hledat na webových stránkách výrobce, případně se na něj obrátit e-mailem. Někteří výrobci přímo uvádějí, že určité modemy fungují v Linuxu. V takových případech možná budete muset nainstalovat speciální ovladač.

Mám si koupit softwarový modem?

Pouze v případě, že víte, že pro daný typ existuje dobře fungující linuxový ovladač. Odmyslíme -li si problémy se sháněním ovladače, jaké jsou ostatní plusy a minusy softwarových modemů? Protože softwarový modem nechává část (nebo většinu) své práce dělat procesor počítače, je hardwarově jednodušší a tedy levnější. Na druhé straně se zvyšuje zátěž procesoru, což může mít za následek zpomalení počítače.

Úroveň zatížení procesoru závisí jednak na samotném procesoru a jednak na tom, zda je modem plně softwarový. U moderních procesorů a modemů, které používají procesor pouze jako řadič, je pokles výkonu minimální. Dokonce i v případě plně softwarového modemu nepostřehnete pokles výkonu, pokud zároveň neprovádíte další procesorově náročné úkoly. Samozřejmě v době kdy modem nepoužíváte není výkon počítače ovlivněn.

Stojí úspora na ceně za to? Ve většině případů ano, zejména pokud modem nepoužíváte příliš a/nebo neprovozujete procesorově náročné aplikace v době, kdy modem používáte. Peníze ušetřené za modem můžete použít ke koupi rychlejšího procesoru. Nicméně specializovaná elektronika modemu plní své úkoly rozhodně lépe než univerzální procesor (když nebereme v úvahu,

že není využita, pokud modem nepracuje). Pokud tedy používáte modem hodně, měli byste se softwarovým modelům vyhnout (a vystačíte si s pomalejším procesorem :-).

PCI modemy

PCI modem je karta, která se zasouvá do PCI slotu na základní desce počítače. Řada PCI winmodemů pod Linuxem nefunguje (nejsou pro ně ovladače), jiné ale fungují. Sériový ovladač Linuxu podporuje hardwarové PCI modemy (ne však winmodemy nebo linmodemy). Pokud máte linmodem, bude fungovat jenom po instalaci speciálního ovladače. Pokud jde o hardwarový modem podporovaný sériovým ovladačem, nastaví ovladač automaticky veškerou PnP konfiguraci. Viz *Podpora PCI sběrnice*. Pokud modem není podporován přímo, pravděpodobně bude i tak fungovat dobře, ale jeho konfigurace bude trochu pracnější.

AMR modemy

Jsou výhradně winmodemy, které se zasouvají do speciální AMR (Audio Modem Riser) slotu. Někdy se tento slot používá i pro zvukové karty. Nicméně hlavním účelem slotu je připojení HSF modemů, u kterých prakticky veškerou práci zajišťuje procesor. Tyto „modemy" nejsou prakticky nic jiného než A/D a D/A převodníky. Minimálně několik typů je v Linuxu podporováno.

USB modemy

Některé USB modemy v Linuxu fungují, jiné ne. Linux podporuje modemy odpovídající standardu USB Communication Device Class Abstract Control Model (USB CDC ACM). ACM je podporováno modulem *acm.o*, viz */usb/acm* v dokumentaci zdrojových kódů jádra. Sériový port prvního (nultého) ACM modemu je (máte-li *devfs*) */dev/usb/acm/0*. Jinak to bude zřejmě */dev/usb/ttyACM0*.

Které interní modemy nemusí v Linuxu fungovat

- *Softwarové modemy (winmodemy). V Linuxu je podporována asi jen polovina.*
- *Modemy MWave a DSP mohou fungovat, ale při každém zapnutí počítače je budete muset nejprve nastavit přes Windows/DOS.*
- *Modemy s ovladačem RPI (Rockwell) fungují, ale se sníženým výkonem.*

Modemy MWave a některé DSP

Existuje linuxový ovladač pro modem ACP (MWave) v IBM Thinkpad 600+. Viz dokument *ACP Modem mini-HOWTO*.

Hardwarové modemy používají DSP čip (Digital Signal Processor), ovšem některé tyto procesory se programují ovladačem, který se musí před použitím modemu přehrát z disku do paměti DSP. Bohužel to obvykle řeší programy pro DOS/Windows (které v Linuxu nefungují). Nicméně podařilo se některé tyto modemy zprovoznit i v Linuxu. Existuje například linuxový ovladač pro modemy Lucent. Běžné modemy, které v Linuxu fungují (aniž by bylo zapotřebí ovladač) rovněž mají DSP čip, ovšem jeho naprogramování je vyřešeno přímo v modemu. Jako příklad problémových modemů můžeme uvést starší model od IBM Aptiva MWave.

Jedna možnost, jak některé DSP modemy v Linuxu provozovat, je nabootovat DOS (pokud jej na počítači máte), pak v DOSu nainstalovat ovladač pro modem a pak jej nechat naprogramovat DSP. Bez vypnutí počítače pak můžete spustit Linux. Můžete si napsat dávku, která to provede. Takto může vypadat příklad:

```
rem mwave je program programující modem, dodaný výrobcem
```

```
call c:\mww\dll\mwave start
rem loadlin.exe je program pro spuštění Linuxu z DOSu
rem (viz dokument Config-HOWTO)
c:\linux\loadlin f:\vmlinuz root=/dev/hda3 ro
```

Můžete si například ve Windows vytvořit zástupce spouštějící tento soubor v režimu MS-DOS. Modem se bude hlásit na stejném sériovém portu jako v DOSu.

Modemy Newcom ifx potřebují drobný zásah do jádra, protože simulují sériový port nestandardně. Příslušný patch a další informace najdete na adrese http://quinine.pharmacy.ohio-state.edu/~ejolson/linux/newcom.html.

Ovladače Rockwell (RPI)

Některé starší čipy Rockwell potřebují ovladače Rockwell RPI (Rockwele Protocol Interface), které zajišťují kompresi a chybovou korekci. Lze je použít i v Linuxu, i když příslušné ovladače fungují jen ve Windows. V Linuxu bude fungovat bez komprese a bez chybové korekce. Budete muset inicializačním řetězcem modemu poslat příkaz pro vypnutí RPI – u mého modemu to bylo *+H0*. S vypnutím komprese se přenos poněkud zpomalí.

Základní informace o modemech a sériových portech

Abyste modem nainstalovali a používali, nemusíte těmto věcem rozumět. Pokud ale dojde k nějakým potížím, snáze s dále uvedenými informacemi zjistíte, co je špatně. Podrobnější informace o sériových portech najdete v dokumentu *Serial-HOWTO*.

Modem konvertuje digitální signál na analogový a naopak

Většina páteřních telefonních vedení je sice digitálních, ovšem přípojka k vám domů je typicky analogová, takže umí přenášet pouze analogový signál – tedy proměnné elektrické napětí, jehož hodnoty odpovídají zvukovým vlnám, které říkáte do sluchátka. Při zobrazení na osciloskopu tento signál vypadá jako sinusoida s proměnnou frekvencí a amplitudou. Digitální signál vypadá jako obdélníková vlna – například napětí 3 V může odpovídat binární jedničce, napětí 0 V digitální nule. U většiny sériových portů (přes něž se modemy připojují) odpovídá jedničce -12 V a nule +12 V (případně - a + 5 V).

Abyste mohli odeslat data z počítače přes telefonní linku, musí modem číst digitální signál z počítače a konvertovat jej na analogový. Tento proces obnáší vygenerování sinusové vlny a její *mod*ulaci. Na druhém konci telefonní linky modem tento signál *dem*oduluje a obnovuje původní digitální signál. Když si spojíte „mod“ a „dem“ ze začátku výše uvedených slov, dostáváte slovo *modem* (až na to jedno *d* navíc).

Co je to sériový port?

Sériový port je vstupně-výstupní zařízení. Protože je počítač s modemem propojen právě sériovým portem, je nutné rozumět nejen modemu, ale i sériovému portu.

Většina počítačů má jeden nebo dva sériové porty. Ty poznáte podle 9pinového (případně 25pinového) konektoru na zadní straně počítače. Počítačový program na jeden z těchto pinů posílá data, na jiném je čte. Další piny slouží k řízení přenosu a jako signálová zem.

Sériový port je samozřejmě mnohem víc než jen konektor. Konvertuje data z paralelní formy na sériovou a mění elektrickou reprezentaci dat. Uvnitř počítače se data přenášejí paralelně (více bitů po více vodičích současně). Sériový přenos představuje přenos jednoho bitu po druhém po jediném vodiči. Aby mohl sériový port fungovat, musí tedy konvertovat paralelní data z počítače do sériového tvaru na vysílací pin a analogicky z přijímacího pinu zpět do počítače.

Většinu elektroniky sériového portu řeší čip označovaný jako UART. Podrobnosti o tomto čipu najdete v dokumentu *Serial HOWTO*.

Nejprve si ale dočtěte zbytek této kapitoly, abyste viděli, kam UART zapadá v celém schématu.

Piny a vodiče

Starší počítače používaly 25pinový konektor, většinou se ale využívalo pouze 9 pinů, proto jsou dnešní konektory 9pinové. Ke každému pinu je obvykle připojen jeden vodič. Kromě dvou vodičů používaných pro vysílání a příjem dat slouží další vodič jako signálová zem. Napětí na všech ostatních vodičích se měří právě vůči této zemi. Minimální počet vodičů potřebný pro obousměrnou komunikaci je tedy 3.

Další vodiče slouží k řídicím účelům (signalizaci) a nepřenášejí data. Jeden z nich se například používá k tomu, aby se počítači řeklo, že má přestat posílat data. Další z nich analogicky slouží k tomu, aby počítač mohl říct připojenému zařízení, aby přestalo posílat data do počítače. V případě, že je připojen modem, mohou další vodiče například říkat modemu, aby zavěsil linku, nebo říkat počítači, že bylo navázáno spojení nebo že po lince přichází hovor. Podrobnosti viz dokument *Serial-HOWTO*, část *Pinout and Signals*.

Interní modemy obsahují sériový port

U interních modemů žádný 9pinový konektor nenajdete, jejich chování je ale prakticky stejné, jako kdyby takové propojení existovalo. Namísto signalizace napětím na jednotlivých vodičích může modem například používat stavové bity ve své paměti, které nahrazují neexistující vodiče. Počítači se sériový port interního modemu jeví stejně jako každý jiný sériový port. Platí to i pro limit přenosové rychlosti, který je na běžných sériových portech 115 200 bitů za sekundu.

Vstupně-výstupní adresy a přerušení

Aby mohl počítač s jednotlivými sériovými porty komunikovat, musí operační systém vědět, které porty existují a kde jsou (jaká je jejich vstupně-výstupní adresa). Kromě toho musí vědět, kterým signálem (přerušením) se bude port dožadovat obsluhy. Každý sériový port tedy musí mít nastavenu svou vstupně-výstupní adresu a své přerušení (IRQ). U PCI sběrnice to funguje trochu jinak, protože ta používá vlastní mechanismus řízení přerušení. PCI BIOS ovšem tato PCI přerušení mapuje na klasická IRQ, takže všechno vypadá jako normálně, až na to, že přerušení mohou být sdílena (jedno přerušení může být využíváno více zařízeními).

Vstupně-výstupní adresy jsou něco jiného než paměťové adresy. Když je na adresovou sběrnici počítače poslána vstupně-výstupní adresa, nastaví se rovněž signál, který říká paměti počítače, ať adresu ignoruje, a zároveň říká všem zařízením (tedy i sériovým portům), aby poslouchaly, zda adresa na sběrnici není jejich. Pokud zařízení „zaslechne“ svou adresu, přečte si data na datové sběrnici.

Názvy: ttyS0, ttyS1 atd.

Sériové porty jsou pojmenovány *ttyS0*, *ttyS1* atd. (a obvykle odpovídají portům COM1, COM2 atd. v DOSu/Windows). V adresáři */dev* je pro každý sériový port samostatný soubor. Vypíšete je pří-

kazem *ls /dev/ttyS**. Ovšem pouhá existence souboru například *ttyS3* neznamená, že fyzicky existuje i takový port.

Který název odpovídá kterému fyzickému portu je dáno následujícím procesem. Sériový ovladač (program) udržuje tabulku s informacemi o tom, která vstupně-výstupní adresa odpovídá kterému *ttyS*. Toto mapování názvů na adresy (a přerušení) lze zobrazit i nastavit příkazem *setserial*. Nenastavuje se tím adresa a přerušení samotného hardwaru (to se nastavuje přepínači nebo PnP softwarem). Mapování konkrétního fyzického zařízení na konkrétní *ttyS* tedy závisí jednak na tom, co si (prostřednictvím *setserial*) myslí ovladač a jednak na nastavení samotného zařízení. Pokud dojde k chybě, nemusí fyzický sériový port odpovídat žádnému názvu a nejde jej tedy použít.

Přerušení

Na cestě od telefonní linky do počítače přicházejí data do modemu, zde se převedou do digitální podoby a předají se na sériový port. Jakmile sériový port přijme a ve své frontě uloží nějaký počet bajtů (lze nastavit na 1, 4, 8 nebo 14), signalizuje procesoru, že si má data vyzvednout. Provede to posláním signálu (tzv. přerušení) na vodič, který je typicky vyhrazen pouze danému portu. Port tedy přijme určitý počet bajtů a pak vyvolá přerušení.

Přerušení se generuje také v případě, že delší dobu nepřicházejí žádná další data. Pokud tedy data přicházejí pomalu (například když někdo píše na klávesnici terminálu), může být přerušení vyvoláno po každém přijatém bajtu.

Jednotlivé vodiče pro vyvolání přerušení mají svá čísla a sériový port musí vědět, který z vodičů má použít. Například port *ttyS0* typicky používá přerušovací signál označený jako IRQ4. Kdykoliv port vyžaduje obsluhu od počítače, vyvolá na tomto vodiči přerušení. Je nutné, aby přerušení bylo obslouženo včas, protože buffer sériového portu pojme pouze 16 bajtů dat. Pokud procesor nestihne přerušení včas obsloužit a data přečíst, může dojít k přeplnění bufferu a následné ztrátě dat.

U externích modemů neexistuje metoda, jak přerušit přísun dat od modemu a ztrátě tak zabránit. U interních modemů je zmíněný 16bajtový buffer na stejné kartě jako samotný modem a inteligentní modem nebude zapisovat, pokud je buffer plný. Nedojde tedy k přeplnění, nicméně modem sám musí použít nějaký mechanismus řízení přenosu s modemem na protější straně, aby nezaplnil i svou vyrovnávací paměť. To je jedna výhoda interních modemů.

Přerušení se dále generuje, jakmile sériový port odešle 16 bajtů ze svého vysílacího bufferu na externí zařízení. Tím se uvolní místo pro dalších 16 bajtů a procesor je může dodat. Přerušení také vzniká, pokud se mění některé řídicí signály.

Buffery, o nichž jsme právě mluvili, jsou softwarové buffery. Kromě toho má sériový port velké buffery v hlavní paměti počítače. O nich budeme mluvit později.

Přerušení zprostředkovávají celou řadu informací, ovšem pouze nepřímo. Samotné přerušení jen říká tzv. řadiči přerušení, že konkrétní zařízení si vyžaduje pozornost. Řadič přerušení to pak sdělí procesoru. Procesor následně spustí speciální program, který přerušení obslouží. Tento program je součástí ovladače zařízení – tedy např. sériového portu. Zkusí zjistit, co se se sériovým portem děje a pak problém vyřeší například přečtením dat z portu nebo zápisem dat na port. Co se děje, může obslužný program zjistit velmi snadno, protože port má na svých vstupně-výstupních adresách, které ovladač portu zná, stavové registry. Ovladač si přečte obsah registrů, prozkoumá jej, zjistí, co se děje a zareaguje.

Komprese dat (modemem)

Než budeme pokračovat povídáním o sériovém portu, musíte se zmínit o další věci, kterou modem dělá – o kompresi dat. V některých případech (winmodemy) se komprese provádí procesorem počítače (pomocí ovladače běžícího ve Windows), naše diskuse se ale bude týkat případů, kdy kompresi fyzicky provádí samotný modem – jen v takovém případě bude modem pod Linuxem fungovat.

Aby bylo možné data přes telefonní linku odeslat rychleji, je možné je zakódovat nějakým postupem, který závisí na samotných datech. Zakódovaná data jsou kratší (mají méně bajtů) než původní data a lze je tedy poslat rychleji. Této operaci se říká komprese dat.

Stahujete-li soubory z Internetu, mohou být už komprimovány a není rozumné, aby se je modem pokoušel komprimovat znovu. Modem může poznat, že přenáší již komprimovaná data a nebude se je snažit zkomprimovat více. Pokud přijímáte data komprimovaná odesílajícím modemem, váš modem je dekomprimuje a dodá vám více bajtů, než kolik bylo přeneseno telefonní linkou. Objem toku dat z modemu do počítače tedy bude větší než objem toku z telefonní linky do modemu. Poměr těchto objemů se označuje jako kompresní poměr. Ten může být někdy až 4, ale není to příliš pravděpodobné.

Korekce chyb

Analogicky ke kompresi dat může modem provádět i korekci chyb. Chybová korekce vyžaduje přenos dodatečných informací a sníží tedy přenosovou rychlost, protože ale korekce odstraňuje start a stop bity, celková přenosová rychlost typicky vzroste.

Na rozhraní mezi sériovým portem a okolním světem je každý 8bitový bajt dat vybaven ještě dvěma bity navíc – tzv. start a stop bitem. Bez použití chybové korekce tyto bity typicky projdou modemem a posílají se po telefonní lince. Při zapnutí korekce se start a stop bity odříznou a zbývajících 8 datových bitů se umístí do paketu. Je to efektivnější a vede to k rychlejšímu přenosu i přesto, že paket se prodlužuje o hlavičku a informace potřebné pro korekci chyb.

Rychlost

Data (tedy bajty reprezentující písmena, obrázky a podobně) tečou z počítače do modemu a pak ven telefonní linkou. Objem přenesených dat za sekundu, například 56k (56 000) bitů za sekundu se (nesprávně) označuje jako „rychlost“. Pokud by modem neprováděl kompresi, pak by byla rychlost přenosu mezi počítačem a modemem přibližně stejná jako rychlost přenosu po telefonní lince.

Ve skutečnosti ale vstupují do hry dvě rychlosti, které je třeba rozlišovat:

- *Rychlost přenosu po telefonní lince (tzv. DCE rychlost).*
- *Rychlost přenosu po sériovém portu do modemu (tzv. DTE rychlost).*

Když vytočíte číslo a modem se připojí ke druhému modemu na druhé straně telefonní linky, často vám pošle zprávu jako „CONNECT 28800“ nebo „CONNECT 115200“. Co to znamená? No, je to buď DCE nebo DTE rychlost. Pokud je rychlost vyšší než výrobcem uváděná rychlost modemu, musí jít o DTE rychlost – tedy rychlost přenosu mezi počítačem a modemem. To je případ rychlosti 115 200 v našem příkladu. Naopak rychlost 28 800 musí být DCE rychlost (mezi modemy), protože sériový port takovou rychlost nepoužívá. Modem lze nastavit tak, aby oznamoval buď tu, nebo onu rychlost. Některé modemy oznamují obě rychlosti, rychlost mezi modemy je pak ohlašována jako (například) CARRIER 28800.

Máte-li interní modem, možná čekáte, že DTE rychlost nebude omezena, protože modem je uvnitř počítače a je prakticky jeho součástí. Nicméně rychlostní omezení tam stále je, protože modem obsahuje svůj vlastní sériový port.

Je důležité vědět, že průměrná rychlost je obvykle menší než avizovaná, zejména na lince mezi modemem a počítačem. Pokles průměrné rychlosti je způsoben čekacími stavy – což může být čekání až sekundu, způsobené řízením přenosu. Navíc modem se může přepnout na nižší rychlost, pokud detekuje zhoršení kvality telefonní linky.

Doporučené DTE rychlosti najdete v části *Jakou rychlost použít*.

Řízení přenosu

Řízení přenosu poskytuje možnost zpomalit přenos bajtů vedením. U sériového portu to znamená možnost zastavit a znovu obnovit přenos bez ztráty dat. Řízení přenosu je nezbytné kvůli výkyvům v okamžité přenosové rychlosti.

Příklad řízení přenosu

Předpokládejme situaci, kdy máte externí modem s rychlostí 33,6k, který je připojen k sériovém portu. Modem přijímá a vysílá data po telefonní lince rychlostí 33,6 kilobitů za sekundu. Předpokládejme, že neprovádí kompresi ani chybovou korekci. Sériový port pracuje s rychlostí 115 kb/s a posíláte data z počítače ven. Rychlost přenosu od počítače k modemu je tedy 115,2 kb/s. Ovšem přenos z modemu na telefonní linku probíhá rychlostí 33,6 kb/s. Data do modemu tedy „přitékají" rychleji, než z něj „odtékají". Rozdíl v rychlostech (81,6 kb/s) musí modem vyrovnávat pomocí nějaké vyrovnávací paměti. Ta by se ovšem velmi rychle zaplnila.

A nyní vstupuje do hry řízení přenosu. Když je vyrovnávací paměť modemu téměř plná, modem pošle po sériovém portu signál pro zastavení přenosu. Sériový port předá tento signál ovladači zařízení a přenos rychlostí 115,2 kb/s bude zastaven. Modem stále odesílá data nashromážděná ve své paměti rychlostí 33,6 kb/s. Jakmile mu už téměř žádná data nezbývají, pošle sériovému portu signál pro zahájení přenosu a přenos od počítače k modemu rychlostí 115,2 kb/s se obnoví. Díky řízení přenosu tak na „rychlém" drátě poklesne průměrná přenosová rychlost na potřebných 33,6 kb/s, což je podstatně méně, než špičková rychlost 115,2 kb/s.

Výše uvedený popis se týkal externích modemů, u většiny interních modemů je ale situace podobná. Rychlost přenosu mezi počítačem a modemem je stále omezena, i když se neodehrává po samostatném kabelu. Díky tomu jsou interní modemy kompatibilní s externími.

Příklad se týkal řízení toku od počítače k modemu. Máme zde ale i opačný směr – od modemu (nebo jiného zařízení) do počítače. V každém směru vstupují do hry tři buffery: 1. v modemu, 2. v UART čipu, 3. v paměti počítače (obsluhovaný ovladačem). Řízení přenosu chrání buffery před přeplněním. Buffer UART čipu není tímto mechanismem chráněn, zde se spoléhá na dostatečně rychlou obsluhu přerušení, které čip generuje.

Ve směru od modemu k počítači není řízení přenosu obvykle zapotřebí. Pokud byste ale mezi počítačem a modemem neměli dostatečně rychlou linku, museli byste zpomalit rychlost dat přitékajících po telefonní lince. Provede se to tak, že modem prostě řekne modemu na druhé straně, aby přestal vysílat.

Hardwarové a softwarové řízení přenosu

Pokud je to možné, vždy se preferuje hardwarové řízení přenosu, které využívá vyhrazené vodiče k zasílání signálů „start" a „stop". Moderní modemy používají téměř vždy hardwarové řízení přenosu mezi modemem a sériovým portem. Softwarové řízení přenosu využívá k signalizaci datové vodiče. Používá řídicí ASCII znaky DC1 (start) a DC3 (stop), které přímo vkládá do přenášených dat. Softwarové řízení má jednak pomalejší odezvu a jednak bez dalších opatření neumožňuje pře-

nos binárních dat. Binární data totiž s velkou pravděpodobností budou obsahovat i znaky DC1 a DC3 a je třeba nějak odlišit, které z těchto znaků jsou součástí dat a které řídí přenos. Aby bylo možné softwarové řízení přenosu binárních dat, musí tuto funkci podporovat jak modem, tak ovladač sériového portu.

Když řízení přenosu nefunguje

Pochopení teorie řízení přenosu může mít i praktický užitek. Používal jsem modem k připojení na Internet a všechno fungovalo dobře. Po nějaké době jsem se ale pokoušel odeslat dlouhý soubor a neustále docházelo k chybám. Příjem souborů ale fungoval dobře. Problém byl způsoben vypnutým řízením přenosu. Při odesílání dlouhých souborů došlo k zahlcení modemu, protože modem nikdy neřekl počítači „stop". V opačném směru problém nebyl, protože rychlost přenosu mezi počítačem a modemem byla vždy vyšší než rychlost na telefonní lince. Problém jsem vyřešil tím, že jsem v rámci inicializační sekvence modemu zapnul řízení přenosu.

Řízení přenosu mezi modemy

K tomu dochází mezi dvěma modemy na telefonní lince. Z praktického pohledu se provádí pouze tehdy, je-li zapnuta chybová korekce. I bez chybové korekce je sice možné řízení zapnout, ale může to vést k problémům při přenosu binárních dat a obvykle se to nepoužívá.

Přenosová cesta a vyrovnávací paměti

O tomto tématu jsme si už řekli dost – hovořili jsme o řízení přenosu, o dvojici 16bajtových bufferů v UART čipu a o dvojici větších bufferů v zařízení, připojeném k sériovému portu (například v modemu). Ještě nám ale zbývá jedna dvojice bufferů. Jde o velké buffery (například 4 kB) v paměti, takzvané buffery sériového portu. Když aplikace pošle data na sériový port (a modem), jsou nejprve uložena v odchozím bufferu sériového portu v paměti. Tyto buffery jsou dva, jeden pro odchozí a druhý pro příchozí směr. Následující diagram představuje příklad činnosti bufferů při připojení na Internet. Odesílaná data postupují zleva doprava, přijímaná data zprava doleva.

aplikace (prohlížeč)	4kB buffer v paměti	16B buffer v UART čipu	1kB buffer v modemu	telefonní linka

Ovladač sériového portu vezme řekněme 16 bajtů z odchozího bufferu a jeden bajt po druhém odešle do 16bajtové fronty v UART čipu. Jakmile se data jednou ocitnou v UART čipu, už jejich odeslání nelze zastavit. Čip je odešle do modemu, který má rovněž docela velký buffer (například 1 kB). Když ovladač (na základě požadavku řízení přenosu) zastaví odesílání dat z počítače, znamená to, že přestane přenášet data z velkého bufferu v paměti na UART. I poté však samotná aplikace může odesílat data do bufferu sériového portu, dokud nedojde k jeho zaplnění. Zaplní-li se buffer, aplikace do něj nemůže zapsat další data (dojde k zablokování funkce *write*) a program se zastaví do doby, než se buffer uvolní. Příkaz „stop" řízení přenosu je tedy schopen zastavit činnost celé aplikace. To však neznamená, že se zastaví počítač – ten může pokračovat v provádění jiných procesů. (Tento popis je poněkud zjednodušený, protože existují mechanismy, umožňující aplikaci pokračovat v jiné činnosti, zatímco se čeká na odblokování zápisu.)

Příkazy modemu

Příkazy se modemu posílají stejným vodičem jako data. Příkazy jsou krátké ASCII řetězce. Například text „AT&K3" zapne hardwarové řízení přenosu (RTC/CTS) mezi počítačem a modemem, příkaz „ATDT5393401" začne vytáčet číslo 5393401. Všimněte si, že všechny příkazy začínají znaky „AT". Některé příkazy (například zapnutí hardwarového řízení přenosu) konfigurují modem. Jiné

(například vytáčení čísla) provádějí nějakou operaci. Když spustíte komunikační program, nejprve pošle modemu inicializační řetězec příkazů, kterými modem nastaví. Příkazy se posílají stejným způsobem jako data, ovšem jen dokud modem nezačne vytáčet (nebo dokud nepřijme hovor).

Jakmile se modem spojí s druhým modemem, všechno, co se odešle z počítače na modem, je předáno přímo druhému modemu a není to interpretováno jako příkaz. Existuje jedna možnost, jak „utéct" z tohoto režimu a vrátit se zpět do příkazového režimu, kdy budou zasílané znaky interpretovány jako příkazy. Počítač musí pouze poslat znaky „+++" s definovanou prodlevou před a po. Je-li časování správné, modem se přepne zpět do příkazového režimu. Další možnost, jak vyvolat příkazový režim, je signálem na příslušném řídicím vodiči.

Na Internetu najdete spoustu seznamů modemových příkazů. Odkazy na ně jsou uvedeny v kapitole *Odkazy*. Různé typy a modely modemů bohužel nepoužívají stejné množiny příkazů. To, co funguje pro jeden modem, nemusí fungovat pro druhý. Některé společné příkazy (které ovšem nemusí fungovat u všech modemů) najdete v kapitole *Konfigurace modemu*.

Modul sériového ovladače

Ovladač sériového portu je program, který slouží k obsluze sériového portu. V současnosti jde o jeden z modulů jádra. Počínaje jádry 2.2 se tento modul automaticky nahraje v případě potřeby. Ve starších jádrech musíte mít spuštěn *kerneld*, aby se modul nahrál. V době, kdy se ještě příliš nepoužívala modulární architektura ovladačů, býval ovladač sériového portu obvykle součástí jádra (a v některých jádrech je jím doposud). Pokud je ovladač přeložen jako součást jádra, nepokoušejte se jej zavést jako modul.

Jakmile se modul nahraje, zobrazí hlášení o existujících sériových portech (a často ukáže špatná přerušení). Když však programem *setserial* sdělíte ovladači (snad) správná přerušení, mělo by se stejné hlášení objevit znovu, tentokrát už se správnými hodnotami. Podrobnosti viz kapitola *Serial Module* v dokumentu *Serial-HOWTO*.

Přehled konfigurace

Protože každý modem je připojen k nějakému sériovému portu a protože port má svůj hardware i software, rozpadá se konfigurace na tři části:

- Zjištění hardwarové konfigurace portu (vstupně-výstupní adresa, přerušení) – buď pomocí PnP nebo podle nastavení přepínačů, pak programem setserial. Viz *Nalezení sériového portu*.
- Konfigurace sériového portu na vyšší úrovni komunikačním programem (jako stty). Nastavuje rychlost, řízení přenosu a podobně. Viz *Konfigurace sériového portu: vyšší úroveň*.
- Konfigurace samotného modemu komunikačním programem. Viz *Konfigurace modemu*.

V tomto přehledu jsme vynechali některé věci, které může program *setserial* provést kromě pouhého nalezení sériových portů. Za normálních okolností je nepoužijete, ale v budoucnu tak bude možné například nastavit velmi vysoké rychlosti přenosu.

Mezi komunikační programy patří minicom, seyon nebo wvdial (pro PPP) a *mgetty* (pro příjem hovorů). Tyto komunikační programy potřebují nastavit, i když jejich standardní konfigurace bude možná vyhovovat jen s malými úpravami.

Komunikační program bohužel neumí najít sériový port. Jeho „nalezení" je otázkou nízkoúrovňové PnP konfigurace portu – nastavení jeho adresy a přerušení jak v hardwaru, tak v ovladači. Pokud máte štěstí, provede se to při startu systému automaticky. V minulosti se nastavení hardwaru

provádělo pomocí přepínačů a ovladač se nastavoval programem *setserial*, dneska by všechno měl zvládnout PnP software. Možná ale budete program *setserial* také potřebovat. Pokud by ovladač sériového portu nebyl schopen najít port, k němuž je modem připojen, můžete jej zkusit najít sami postupem popsaným v následující kapitole. Nemusí to ale být jednoduché.

Nalezení sériového portu: adresy a přerušení

Přerušení a adresy – přehled

Aby mohl sériový port správně fungovat, potřebuje mít vstupně-výstupní adresu a přerušení. Bez adresy s ním nebude možné komunikovat a nebude tedy funkční. Bez přerušení by se musel obsluhovat neefektivní dotazovací metodou (v takovém případě se ovladači nastavuje přerušení 0). Port tedy potřebuje adresu a přerušení. Ve starých zlatých časech se obojí nastavovalo přepínači na kartě sériového portu. Dnes se to nastavuje pomocí technologie Plug-and-Play.

Ovladač potřebuje znát adresu a přerušení portu, aby byl schopen se s ním bavit. Moderní ovladač (jádra 2.4) se pokouší nastavení zjistit pomocí PnP a nemusíte mu je tedy říkat programem *setserial*. Ovladač umí i nastavit adresu nebo aktivovat hardware. Existují bohužel některé PCI sériové porty, které ovladač nenajde a budete je muset aktivovat sami. Viz *PCI: zapnutí vypnutého portu*.

Kromě toho ovladač testuje obvyklé adresy ISA portů a zjišťuje, zda takové porty existují. Tento postup se vztahuje na porty nastavované přepínači, někdy i na PnP porty, pokud je ovladač pomocí PnP nenajde. Nastavení sériového portu tím, že mu přidělíte adresu a přerušení, představuje jeho nízkoúrovňovou konfiguraci. Často ji provede ovladač automaticky, někdy ji ale musíte provést ručně. V dalším textu si celý proces popíšeme podrobněji.

Nízkoúrovňová konfigurace obnáší přiřazení vstupně-výstupní adresy, přerušení a názvu (například *ttyS2*). Dvojice adresa-přerušení musí být nastavena jednak v hardwaru portu a jednak ji musí znát ovladač portu. Název portu musí znát jen ovladač. Jedna možnost, jak ovladači všechny hodnoty říct, je program *setserial*. Další možnost je použití PnP, kdy ovladač zjistí/nastaví adresy a přerušení a zapamatuje si je. Pokud nastavujete port přepínači, musíte vždy použít *setserial*. Potřebujete-li port nakonfigurovat, ale nerozumíte podrobnostem, snadno se dostanete do potíží.

Když se Linux spouští, pokouší se detekovat a (nízkoúrovňově) nakonfigurovat některé sériové porty. Přesný průběh závisí na BIOSu, hardwaru, distribuci Linuxu a dalších věcech. Pokud porty pracují správně, nebudete muset do jejich nízkoúrovňové konfigurace nijak zasahovat.

Máte-li se sériovými porty problémy, budete se muset na jejich konfiguraci zaměřit. Používáte-li jádro 2.2 nebo starší, nevyhnete se nízkoúrovňové konfiguraci, pokud:

- chcete použít více než 2 ISA sériové porty,
- instalujete nový sériový port (například interní modem).

U jader 2.2 a vyšších se vám možná podaří používat více než dva sériové porty bez nutnosti nízkoúrovňové konfigurace s využitím sdílení přerušení. Tuto možnost by měly podporovat všechny PCI porty, u ISA portů to podporují jen některé karty. Jednodušší bude přiřadit každému portu vlastní přerušení – pokud jich ale tolik máte k dispozici. Viz *Sdílení přerušení a jádra 2.2+*.

Dokud ovladač nezná správnou adresu a přerušení, port většinou nebude vůbec fungovat. U PnP portu může dojít k tomu, že je port vypnutý a nedá se nalézt (jinak než pomocí PnP nástrojů jako *lspci*). Aplikace a nástroje jako setserial a scanport nepoužívají PnP nástroje a vypnuté porty tedy nenaleznou. I pokud ovladač automaticky nalezne sériový port, může port fungovat velice

pomalu v případě, že má špatné přerušení. Viz *Hodně pomalý port*. U PCI portů je riziko přidělení špatného přerušení velmi malé.

Ve světě Windows se vstupně-výstupní adresy a přerušení označují jako „prostředky“ a nás tedy čeká konfigurace určitých prostředků. Ovšem „prostředků“ existuje mnohem více a tento termín může mít různé významy. Stručně řečeno, nízkoúrovňová konfigurace obnáší zapnutí zařízení, přiřazení jména a nastavení dvou hodnot (přerušení a adresy) na dvou místech:

- v ovladači portu (často programem *setserial* při startu systému),
- v konfiguračních registrech samotného hardwaru portu.

Sledujte hlášení při startu počítače, ta jsou obvykle správná. Pokud ale něco není v pořádku, nemusí se sériový port vůbec objevit, případně hlášení programu setserial nemusí ukázat pravdivou konfiguraci hardwaru portu (ani se to od něj nečeká). Viz *Adresy a přerušení: hlášení při startu systému*.

Podpora PCI sběrnice

Úvod

Některé PCI modemy jsou „winmodemy“ bez linuxového ovladače (a pod Linuxem tedy nefungují), jiné PCI modemy však fungují dobře. Pokud je modem softwarový, budete pro něj potřebovat ovladač – viz dokument *Linmodem-HOWTO*.

Máte-li jádro 2.4, mělo by obsahovat podporu PnP (ať už vestavěnou, nebo modulární). Některé PCI sériové porty umí automaticky detekovat a nízkoúrovňově nastavit ovladač sériového portu, jiné však nenalezne. Samotný ovladač sériového portu nijak nepodporuje softwarové modemy, pro řadu z nich ale existují samostatné ovladače. Jádro 2.2 neobsahuje podporu PCI sériových portů (i když je možné ji do něj přidat). Sériový ovladač v jádře 2.4 si přečte identifikační číslo uložené v hardwaru portu a podle něj pozná, jak port obsluhovat (pokud to umí). Měl by mu přiřadit adresu, zjistit jeho přerušení a podobně. Nebudete tedy muset použít program *setserial*.

Žádost o podporu vašeho portu

Pokud máte interní PCI modem, o němž víte, že to není winmodem, ale který nepracuje, protože jej sériový ovladač nepodporuje, můžete spolupracovat na vytvoření podpory. Musíte kontaktovat správce kódu sériového ovladače, jímž je Ted Y. Ts'o. Nejprve se ale podívejte na seznam modemů na adrese http://www.idir.net/~gromitkc/winmodem.html, kde najdete nejnovější informace o PCI modemech a příbuzných tématech.

Dále se podívejte na Tedovy stránky (http://serial.sourceforge.net/), kde zjistíte, co dělat dál. Takto vypadá zkrácený postup: Musíte Tedovi e-mailem poslat kopii výstupu příkazu *lspci -vv* se všemi informacemi o výrobci a modelu PCI modemu (nebo sériového portu). Pak vám Ted pošle k otestování upravený ovladač, který by mohl fungovat. Musíte si jej stáhnout, přeložit a případně přeložit i jádro. Pak ovladač otestujete, zda funguje, a pošlete Tedovi výsledek. Pokud by se vám do toho všeho chtělo, Tedova adresa je mailto:tytso@mit.edu.

Další informace o PCI

PCI porty nejsou dobře standardizované. Některé s počítačem komunikují prostřednictvím jeho hlavní paměti. Jiné vyžadují speciální zapnutí přerušení. Z výstupu příkazu *lspci -vv* je možné usoudit, zda lze daný port podporovat. Pokud ve výstupu najdete čtyřmístnou vstupně-výstupní adresu, může se port zprovoznit čistě tím, že programu *setserial* řeknete jeho adresu a přerušení. Tímto způsobem se několika lidem podařilo zprovoznit modem 3COM 3CP5610. Pokud například

ve výstupu příkazu *lspci* najdete přerušení 10, adresu 0xecb8 a budete chtít port pojmenovat *ttyS2*, pak příslušný příkaz vypadá takto:

```
setserial /dev/ttyS2 irq 10 port 0xecb8 autoconfig
```

Hlášení „Probing PCI hardware“ vypisované při spouštění systému znamená, že se čtou konfigurační PnP registry na PCI kartách, které obsahují vstupně-výstupní adresy a přerušení jednotlivých karet. Je to jiné „probing“, než které provádí sériový ovladač – ten se snaží číst určité adresy a zjišťuje, jestli na nich není připojen sériový port.

Běžné chyby v nízkoúrovňové konfiguraci

Mezi obvyklé chyby patří:

- Příkaz *setserial*: Uživatel jej spustí (bez parametrů autoconfig a auto_irq) a domnívá se, že program otestoval hardware, zda jsou zadané hodnoty správné. To ale program nedělá.
- Hlášení příkazu *setserial*: Uživatel si je přečte při startu systému (nebo později zadáním příkazu setserial) a domnívá se, že zobrazené hodnoty ukazují skutečnou konfiguraci jeho hardwaru.
- */proc/interrupts*: Pokud se sériové zařízení nepoužívá, není v tomto souboru uvedeno jeho přerušení a uživatel si mylně myslí, že sériové zařízení nebylo nalezeno (nebo nemá nastaveno přerušení).
- */proc/ioports* a */proc/tty/driver/serial*: Uživatel si myslí, že zde nalezne údaje o skutečné konfiguraci hardwaru, ve skutečnosti jsou zde stejné (a možná chybné) informace, jako vypisuje setserial.

Přerušení a adresy musí souhlasit

Na otázku „jaká je adresa a přerušení portu“ existují dvě odpovědi: 1. To, co si myslí ovladač sériového portu (a co nastavuje a vypisuje *setserial*). 2. To, co je opravdu nastaveno v hardwaru portu. Obě nastavení musí být stejná. Pokud nejsou, vznikají problémy, protože ovladač má nesprávné informace o fyzickém portu. V některých případech může být port vypnut a nemá tedy adresu ani přerušení.

Pokud má ovladač nastavenu špatnou adresu, bude posílat data na neexistující sériový port – anebo, což je ještě horší, na nějaké úplně jiné zařízení. Pokud bude mít ovladač špatně nastaveno přerušení, nebude reagovat na žádosti portu o obsluhu, což vede k velmi pomalé, případně žádné, odezvě. Viz *Hodně pomalý port*. Abyste mohli ověřit, že nastavení adresy a přerušení je na obou místech stejné, musíte nejprve zjistit, jaká ta nastavení na obou místech jsou.

Jaká je adresa a přerušení v ovladači?

Úvod

To, co si myslí ovladač, není nezbytně to, jak je nastaven hardware portu. Pokud všechno funguje správně, tak nastavení ovladače budou zřejmě správná (tedy odpovídající nastavením hardwaru) a nebudete muset nic zkoumat (pokud vás k tomu nežene prostá zvědavost). Možností, jak zjistit nastavení ovladače, je několik: hlášení adresy a přerušení při startu systému, „soubory“ v adresáři */proc* a konečně příkaz *setserial*.

Hlášení adres a přerušení při startu systému

V řadě případů dojde k automatické nízkoúrovňové konfigurace portu při startu systému (ta ale nemusí být správná). Abyste zjistili, co se děje, sledujte hlášení vypisovaná při startu. Nezapomeňte se podívat i na hodnoty, které vypisuje BIOS předtím, než se začne spouštět Linux. Hlášení BIOSu můžete pozastavit klávesou Pause a pomocí kláves Shift+PageUp a Shift+PageDown se můžete ve výpisu posouvat. Příkazem *dmesg* (nebo v adresáři */var/log*) znovu zobrazíte hlášení vypisovaná při startu, nemusíte však najít všechna. Takto vypadá příklad výpisu při startu. (Název *ttyS00* odpovídá zařízení */dev/ttyS0*).

Nejprve vidíme, co bylo detekováno (hodnoty přerušení jsou ovšem jenom odhad):

```
Serial driver version 4.27 with no serial options enabled
ttyS00 at 0x03f8 (irq = 4) is a 16550A
ttyS01 at 0x02f8 (irq = 3) is a 16550A
ttyS02 at 0x03e8 (irq = 4) is a 16550A
```

Pak nám *setserial* ukáže, co si nastavil, ale ani to nemusí být v pořádku:

```
Loading the saved-state of the serial devices...
/dev/ttyS0 at 0x03f8 (irq = 4) is a 16550A
/dev/ttyS1 at 0x02f8 (irq = 3) is a 16550A
/dev/ttyS2 at 0x03e8 (irq = 5) is a 16550A
```

Všimněte si malého rozporu: V prvním výpisu vidíme *ttyS2* s přerušením 4, ve druhém pak s přerušením 5. Příkazem *dmesg* druhý výpis neuvidíte, nicméně ve většině případů je právě on ten správný. Než budete pokračovat ve čtení, vyzkoušejte, jestli vám prostě porty nefungují správně – pak byste se nemuseli o všechny následující záludnosti starat...

Druhý výpis pochází z příkazu *setserial* spuštěného při startu systému. Zobrazuje hodnoty, které ovladač považuje za správné. Ani ty ale nemusí být správné. Ve skutečnosti může být přerušení nastaveno třeba na 8 a pravdu nemá ani jeden výpis. Hodnota 5 na druhém výpisu může pocházet z toho, že někdo nebo něco ji zapsal do konfiguračního souboru. Důvodem, proč Linux nemusí přerušení správně zjistit, je to, že se standardně nepokouší přerušení otestovat. Pouze předpokládá „standardní" nastavení (v prvním výpisu), anebo se smíří s nastaveními, které mu někdo řekl (ve druhém výpisu). Ani jedno z nich nemusí být nutně správné. Pokud je přerušení v ovladači nastaveno špatně, funguje sériový port velmi pomalu, nebo vůbec.

První výpis vznikl v okamžiku, kdy Linux testoval adresy sériových portů, netestoval ale jejich přerušení. Najdete-li na tomto výpisu adresu, pak takový port existuje, ale jeho přerušení může být jiné. Linux se nepokouší přerušení zjistit, protože to nemusí být jednoduše možné. Pouze předpokládá, že přerušení je nastaveno na „obvyklé" hodnoty. Můžete zkusit přerušení zjistit příkazem *setserial* s parametry *autoconfig* a *auto_irq*, ani zde ale nemáte garantovánu správnost výsledku.

Hodnoty vypisované BIOSem (které vidíte ještě před startem Linuxu) jsou hodnoty, na něž je hardware původně nastaven. Pokud je port PnP, je možné, že program *isapnp* nebo *setpci* tyto hodnoty změní. Zkuste hledat hlášení těchto programů. Hodnoty přerušení ve druhém výpisu by měly odpovídat tomu, co zobrazil BIOS. Pokud neodpovídají, budete muset buď změnit nastavení hardwaru, nebo programem *setserial* říct, jak je hardware ve skutečnosti nastaven.

Pokud máte PnP port, pak jej PnP ovladač nalezne jen v případě, že hodnoty adres a přerušení nebyly v zařízení natvrdo přepsány jeho konfiguračním (obvykle dosovým) programem. U jader před 2.4 šlo o běžnou příčinu, proč jádro nedetekovalo sériové porty, které fyzicky existovaly. PnP BIOS může tyto porty automaticky nakonfigurovat. O nastaveních PnP budeme hovořit později.

Adresář /proc a příkaz setserial

Zadejte příkaz *setserial -g /dev/ttyS**. Stejné informace můžete nalézt také v „souborech" v adresáři */proc*. Tyto hodnoty ovšem nemusí odpovídat nastavení hardwaru.

V souboru */proc/ioports* najdete adresy, které ovladač používá. V souboru */proc/interrupts* jsou uvedena přerušení, používaná ovladači zařízení, která jsou momentálně používána. Ukazuje tedy, která přerušení se momentálně obsluhují. V souboru */proc/tty/driver/serial* najdete většinu z výše uvedeného, plus počet odeslaných a přijatých bajtů (a to i tehdy, pokud zařízení není zrovna otevřeno).

Znovu upozorňujeme, že co se týče hodnot adres a přerušení, vidíte pouze to, co si ovladač myslí – a to nemusí nutně odpovídat hardwaru. Data o počtu obsloužených přerušení a o počtech přenesených bajtů jsou nicméně pravdivá. Pokud zde vidíte nějaké vyšší hodnoty, zařízení pravděpodobně pracuje (nebo pracovalo). Může jít ale o přerušení generovaná jiným zařízením. Pokud nebyly přijaty žádné bajty (rx:0), ale nějaké byly odeslány (např. tx:3749), pak funguje jen jeden směr přenosu (a asi bude problém s přerušením). Nějaký nízký, avšak nenulový, počet přerušení může znamenat, že tato přerušení nebyla fyzicky generována sériovým portem.

Jak je hardwarově nastaven sériový port?

Úvod

Pokud jde o PCI port nebo ISA PnP port, pak nastavení hardwaru byla provedena pomocí PnP. I pokud nebylo nastaveno vůbec nic nebo pokud je port vypnutý, můžete je pořád najít příkazem *lspci -v* nebo *isapnp --dumpregs*. Port vypnutý prostřednictvím hardwarového přepínače je samozřejmě úplně nedostupný.

PnP porty si své nastavení po vypnutí napájení nepamatují. To je rozdíl proti klasicky (pomocí přepínačů) nastavovaným portům, jejich nastavení je trvalé. Právě proto se mohou PnP porty nacházet ve vypnutém stavu, u klasických portů je to dost nepravděpodobné.

PCI: Jaké byly nastaveny adresy a přerušení?

U PCI zařízení BIOS téměř vždy provede nastavení přerušení a případně i adresy. Nastavení zjistíte příkazem *lspci -vv* (což je lepší metoda) nebo v souboru */proc/bus/pci* (případně */proc/pci* jader < 2.2). Sériový port modemu bývá často označen jako „Communication controller". Pokud *lspci* vypíše „I/O ports at ... [disabled]", znamená to, že port je vypnutý, nemá nastavenu adresu a nelze jej použít. Viz *PCI: Zapnutí vypnutého portu*.

Pokud je uvedeno více adres, správná je s největší pravděpodobností ta první. Přerušení nemůžete (alespoň pomocí *setpci*) změnit. Pokud totiž zapíšete do hardwarového registru zařízení novou hodnotu přerušení, nic se nestane. Nastavení přerušení totiž provádí BIOS a ten pak nastavené hodnoty do registrů zapíše, aby si je program *lspci* mohl přečíst. Pokud potřebujete, můžete programem *setpci* změnit vstupně-výstupní adresu prostřednictvím parametru BASE_ADDRESS_0 nebo podobného.

PCI: Zapnutí vypnutého portu

Pokud port přes nějakou adresu komunikuje, měli byste ve výpisu příkazu *lspci -vv* vidět něco jako „Control: I/O+ ...", kde + znamená, že vstupně-výstupní adresy jsou zapnuté. Pokud uvidíte „I/O-" (a I/O ports at ... [disabled]"), musíte příkazem *setpci* port zapnout. Příkaz vypadá tak nějak jako *setpci -d 151f:000 command=101*. Hodnota *151f* je kód výrobce, *000* je identifikátor zařízení – obojí zjistíte příkazem *lspci -n -v*, nebo v souboru */proc/bus/pci* nebo příkazem *scanpci -v*. Parametr *command=101* znamená, že se do řídicího registru zapíše hodnota 101 (šestnáctkově). Jde o stejný registr, který příkaz *lspci* označuje jako „Control". Hodnota 101 nastaví dva pří-

nakové bity: 1 nastaví I/O na +, 100 nastaví SERR# na +. V našem případě jsme ve výpisu *lspci* viděli zapnutý pouze bit SERR# řídicího registru. Nechali jsme jej tedy zapnutý nastavením 8. bitu a kromě toho jsme zapnuli 0. bit, odpovídající příznaku I/O. Některé sériové karty příznak SERR# nepoužívají, takže pokud ve výpisu vidíte SERR#-, je zbytečné příznak zapínat a parametr bude *command=1*. Nakonec musíte spustit příkaz *setserial* a říct ovladači přerušení a adresu portu.

Další možnost, jak nastavení portu zajistit, je říct BIOSu, že nemáte operační systém s podporou PnP. Pak by měl BIOS všechna zařízení při startu počítače aktivovat sám. Pokud ale na stejném počítači používáte i Windows, může jim toto nastavení BIOSu vadit (viz dokument *Plug-and-Play -HOWTO*).

ISA PnP porty

U ISA PnP portů můžete zkusit program *pnpdump* (součást balíku *isapnptools*). S parametrem *--dumpregs* by měl zobrazit platná nastavení adresy a přerušení portu. Měl by dokonce najít vypnuté ISA PnP porty. Adresa, kterou zkouší, není adresa zařízení, ale speciální adresa používaná pro komunikaci s PnP kartami.

Nalezení portu, který není vypnutý (ISA, PCI, PnP, ne-PnP)

Možná najdete nějaké informace v hlášení BIOSu při zapnutí počítače. Pomocí kláves Shift+PageUp se posuňte na začátek výpisu při startu systému a hledejte první hlášení, která pocházejí z BIOSu. Tak byl port nastaven před spuštěním Linuxu. Nastavení nemůžete změnit programem *setserial*, může to ale jít programy *isapnp* nebo *setpci*. Počínaje jádry 2.4 může tyto hodnoty u většiny (ale ne u všech) portů změnit i ovladač sériového portu.

Příkazem *scanport* se vyzkoušejí všechny vstupně-výstupní adresy a vypíše se, co by mohl být sériový port. Pak můžete vyzkoušet příkaz *setserial* s parametrem *autoconf*. Budete muset (na základě výpisu programu *scanport*) uhodnout adresu, na níž má *setserial* port hledat.

U portů nastavovaných pomocí přepínačů je nutné zjistit, jaký je význam jejich nastavení. Pokud port nemá přepínače, není PnP a byl nastaven nějakým dosovým programem, je třeba sehnat příslušný program a zjistit, jaké hodnoty byly nastaveny.

Zjištění pomocí Windows (poslední možnost)

U PnP portu může (ale nemusí) jejich nastavení v DOSu/Windows odpovídat i nastavení v Linuxu. Pokud necháte port nastavit PnP BIOSem (kterému řeknete, že máte operační systém bez podpory PnP), měl by Linux přímo použít konfiguraci, kterou nastavil a do své paměti zapsal BIOS. Windows si sice údaje z této paměti přečtou taky, ale nemusí je nutně používat.

Volba přerušení pro port

Pokud používáte PnP porty, tak by je měl nastavit buď PnP BIOS nebo ovladač sériových portů a o volbu přerušení byste se neměli starat. PnP software se rozhodne, jaká přerušení jsou nejvhodnější a přiřadí je jednotlivým zařízením (volba ale nemusí být ideální). Pokud ale nastavení provádíte pomocí *isapnp* (na ISA sběrnici), nebo pokud port konfigurujete pomocí přepínačů, musíte si nějaké přerušení zvolit. Pokud už máte nějaké přerušení vybráno, můžete následující část textu přeskočit – jen byste měli vědět, že přerušení 0 má speciální význam (viz následující odstavec).

IRQ 0 není přerušení

Zatímco hardwarově je přerušení 0 přiděleno časovači, při nastavování sériových portů má speciální význam. Říká ovladači, že port nepoužívá přerušení a že jej má ovladač obsluhovat dotazovací metodou. Tato metoda má zvýšené nároky na procesor, ale je možné ji vyzkoušet, pokud dochází ke konfliktům přerušení nebo pokud přerušení nefunguje. Výhodou použití přerušení 0 je,

že nepotřebujete vědět, jaké přerušení hardware opravdu používá. Toto nastavení byste ale měli použít opravdu jen dočasně, dokud nezjistíte, jaké přerušení použít.

Sdílení přerušení, jádra 2.2+

Sdílení přerušení je situace, kdy dvě zařízení používají stejné přerušení. Obecně to nebylo možné na ISA sběrnici. PCI sběrnice umožňuje sdílet přerušení, ale nelze sdílet jedno přerušení mezi ISA a PCI sběrnicí. Většina multiportových karet sdílení přerušení podporuje. Sdílení poněkud snižuje efektivitu, protože kdykoliv vznikne přerušení, musí se nejprve zjistit, od kterého zařízení pochází. Pokud je to tedy možné, je rozumnější každému zařízení přiřadit vlastní přerušení.

Před jádry 2.2 nebylo možné sdílet přerušení mezi sériovými porty s výjimkou většiny multiportových karet. Počínaje jádrem 2.2 je sdílení přerušení mezi sériovými porty za jistých okolností možné. Aby sdílení v jádrech 2.2 fungovalo, musí být jádro přeloženo s volbou CONFIG_SERIAL_SHARE_IRQ a sdílení přerušení musí podporovat i použité karty. Protože sdílení přerušení je součástí specifikace PCI sběrnice, měly by je podporovat všechny karty.

Které přerušení zvolit?

Samotný hardware sériového portu typicky umožňuje využít jen některá přerušení. Navíc určitě nechcete vytvořit konflikty přerušení, takže většinou nemáte moc na výběr. Za normálních okolností se pro porty *ttyS0* a *ttyS2* používá přerušení 4, pro porty *ttyS1* a *ttyS3* přerušení 3. Soubor */proc/interrupts* říká, která přerušení jsou využívána právě spuštěnými programy. Asi nebudete chtít použít žádné z nich. Než se začalo přerušení 5 používat pro zvukové karty, často sloužilo pro sériové porty.

Následující příklad ukazuje, jak přerušení nastavil Greg (původní autor dokumentu *Serial-HOWTO*) v souboru */etc/rc.d/rc.serial*. Soubor *rc.serial* je skript, který se spouští při startu systému (na různých distribucích se může jmenovat různě). Počínaje *setserial*em verze 2.15 se nastavení neprovádí vždy tímto způsobem, příklad nicméně ukazuje použitá přiřazení:

```
/sbin/setserial /dev/ttyS0 irq 3        # sériová myš
/sbin/setserial /dev/ttyS1 irq 4        # terminál Wyse
/sbin/setserial /dev/ttyS2 irq 5        # modem Zoom
/sbin/setserial /dev/ttyS3 irq 9        # modem USR
```

Standardně se přerušení přiřazují takto:

IRQ 0 Časovač (může mít význam „bez přerušení“)
IRQ 1 Klávesnice
IRQ 2 Kaskáda na druhý řadič přerušení
IRQ 3 Sériový port 2
IRQ 4 Sériový port 1
IRQ 5 Paralelní port 2, zvuková karta
IRQ 6 Disketová mechanika
IRQ 7 Paralelní port 1
IRQ 8 Hodiny reálného času
IRQ 9 Přesměrováno na IRQ 2
IRQ 10 nepřiřazeno
IRQ 11 nepřiřazeno
IRQ 12 nepřiřazeno
IRQ 13 Matematický koprocesor

IRQ 14 Primární řadič pevného disku

IRQ 15 Sekundární řadič pevného disku

Při volbě přerušení neexistuje žádný doporučený postup. Zkuste najít takové, které nepoužívá ani základní deska, ani ostatní karty v systému. Možnými kandidáty jsou 2, 3, 4, 5, 7, 10, 11, 12 nebo 15. IRQ 2 je to samé jako IRQ 9. Ať už mu budete říkat 2 nebo 9, sériový ovladač vám bude rozumět. Pokud máte velmi starou sériovou kartu, nebude možná vůbec umět využít přerušení 8 a vyšší.

Rozhodně nepoužívejte přerušení 1, 6, 8, 13 nebo 14! Ty používá základní deska počítače a vůbec by se jí nelíbilo, kdybyste jí je chtěli sebrat. Jakmile si vyberete, zkontrolujte ještě jednou soubor */proc/interrupts* a ujistěte se, že vámi vybrané hodnoty nebudou s ničím kolidovat.

Volba adresy – konflikt ttyS3 a videokarty

Některé staré sériové karty vytvářejí následující problém. Adresa videokarty IBM 8514 (a některých dalších) je *0x?2e8*, kde *?* je *2*, *4*, *8* nebo *9*. Zde může vznikat konflikt s portem *ttyS3* na adrese *0x02e8*. Možná vám to přijde divné, protože adresa je jiná. Bohužel, některé staré a špatně navržené sériové karty ignorují nejvyšší část adresy a reagují na *cokoliv, co končí* adresou *2e8*.

U ISA karet byste měli použít standardní adresy uvedené níže. PCI karty používají jiné adresy, aby nedocházelo ke konfliktům s ISA kartami. Níže uvedené adresy jsou vlastně vždy první adresa z 8bajtového rozsahu. Například adresa *3f8* ve skutečnosti znamená rozsah adres *3f8* až *3ff*. Každé sériové zařízení (stejně jako jiná zařízení) potřebuje svůj vlastní jedinečný adresový rozsah. Tyto rozsahy se nesmějí překrývat. U sériových portů na ISA sběrnici se používají následující adresy:

ttyS0	0x3f8
ttyS1	0x2f8
ttyS2	0x3e8
ttyS3	0x2e8

Předpokládejme, že existuje adresní konflikt (přijdete na něj příkazem *setserial -g /dev/ttyS**) mezi skutečným portem a jiným portem, který fyzicky neexistuje (ve výpisu vidíte „UART: unknown“). Takovýto „konflikt“ by samozřejmě neměl způsobit žádný problém, u starších jader ale někdy vadí. V takovém případě buď nastavte porty tak, aby ke konfliktu nedocházelo, anebo smažte soubory */dev/ttySx* těch portů, které fyzicky neexistují.

Nastavení adresy a přerušení (hlavně pro PnP)

Jakmile nastavíte adresu a přerušení na hardwaru, nezapomeňte je také sdělit ovladači. U portů bez PnP se nastavení provádí buď pomocí přepínačů, nebo nějakým obslužným dosovým programem (takzvané „jumperless“ nastavení), tímto program také může jít vypnout PnP u PnP karet. Zbytek už se vztahuje pouze na PnP porty.

Dále uvádíme možné konfigurační postupy pro PnP karty:

- Pomocí konfigurační nabídky PnP BIOSu (obvykle je to možné pouze pro externí modemy na portech ttyS0 (COM1) a ttyS1 (COM2)).
- Ponechat konfiguraci na PnP BIOSu
- Pokud ovladač kartu najde, nenastavovat nic.
- Pomocí isapnp pro ISA karty.
- Pomocí setpci (balík pciutils nebo pcitools) pro PCI karty.

Adresy a přerušení se musí nastavit v PnP registrech karty vždy po zapnutí počítače, protože tato nastavení se po vypnutí napájení nezachovávají. Nejjednodušší metoda je nastavit BIOS tak, že nemáte PnP operační systém, a pak by měl BIOS potřebná nastavení provést automaticky sám. Spuštění Windows (které PnP podporují) s BIOSem nastaveným tak, že máte systém bez podpory PnP, ale může způsobovat problémy. Viz dokument *Plug-and-Play-HOWTO*.

Technologie Plug-and-Play (PnP) je určena k tomu, aby automatizovala nastavení adres a přerušení – ve vztahu k Linuxu však toto nastavení zpočátku dost zkomplikovala. V moderních Linuxech (jádra 2.4 a částečně i 2.2) musí ovladač každého zařízení samostatně řešit PnP nastavení svého zařízení. Neexistuje zde bohužel žádné centralizované plánování přidělení adres a přerušení, jaké funguje ve Windows. Nicméně i tak tyto karty v Linuxu obvykle fungují bez problémů.

Konfigurace pomocí PnP BIOSu

Způsob použití nástrojů jako *setpci* nebo *isapnp* je uveden v dokumentaci k těmto programům – ovšem tato samozřejmost neplatí pro BIOS. Ne všechny PnP BIOSy umožňují taková nastavení provést. Typicky umožňuje nastavení pouze prvních dvou sériových portů, a i v tomto případě může být obtížné najít, kde tato nastavení zadat. U BIOSu Award naleznete příslušné volby v části „Chipset features setup“ a obvykle není moc z čeho vybírat. Pokud není řečeno jinak, obvykle tyto dva porty používají standardní nastavení adres a přerušení.

Ať se vám to líbí nebo ne, jakmile zapnete počítač, PnP BIOS zahájí konfiguraci adres a přerušení PnP zařízení. Tuto konfiguraci může provést jen z části a zbytek ponechat na PnP operačním systému (jímž Linux částečně je), anebo může konfiguraci provést úplně, pokud mu řeknete, že nemáte PnP systém. Samozřejmě nemá jak nastavit ovladače příslušných zařízení. Úplné nastavení BIOSem je většinou nejvhodnější, ovladače ale musíte nastavit sami a ne vždy je jednoduché zjistit, jak BIOS jednotlivá zařízení nakonfiguroval.

Pokud BIOSu řeknete, že máte operační systém bez podpory PnP, měl by BIOS nastavit všechny sériové porty – ne jen první dva. Nepřímá možnost, jak ovlivnit to, co BIOS provede, je ve Windows (pokud je na stejném počítači máte) „vynutit“ požadovanou konfiguraci. Viz dokument *Plug-and-Play-HOWTO* a hledejte v něm slůvko *forced*. Jednodušší ovšem je použít konfigurační nabídku BIOSu, kde můžete přepsat to, co jste ve Windows „vynutili“. Můžete také nalézt volbu, která povoluje nebo zakazuje vynucené přepsání konfigurace.

Když přidáte nové PnP zařízení, BIOS by je měl nakonfigurovat. Může dokonce změnit adresy a přerušení stávajících zařízení v případě, že je to nutné k vyřešení konfliktů. Aby mohl konfliktům zabránit, musíte BIOSu říct, jaká ne-PnP zařízení jsou nainstalována a s jakými nastaveními. Jedna z možností, jak mu to říct, je program ICU v DOSu/Windows.

Jak ale zjistíte, co BIOS nastavil, abyste mohli tyto hodnoty sdělit ovladači zařízení? BIOS by vás o nastavených hodnotách mohl informovat – buď ve svém konfiguračním prostředí, nebo prostřednictvím výpisů při startu počítače. Další možnost zjištění představují programy *lspci* a *isapnp --dumpregs*. Jejich výstup ale začátečník nemusí hned pochopit.

Předání adresy a přerušení ovladači

Jakmile nastavíte adresu a přerušení hardwarového zařízení (nebo jste ji nechali nastavit pomocí PnP), musíte také zajistit, aby se vždy při startu systému spouštěl i program *setserial* s příslušnými hodnotami. Viz *Konfigurace při startu systému*.

Konfigurace sériového portu: vyšší úroveň

Úvod

Tuto konfiguraci normálně provádí komunikační program, například *wvdial*. Větší část konfigurace může provést aniž byste dokonce věděli, co dělá. Dříve se tato nastavení prováděla nástrojem *stty*. Pokud tímto nástrojem něco nastavíte, komunikační program si to může změnit, takže obvykle je nejrozumnější ponechat nastavení na konfiguračním programu.

Hardwarové řízení přenosu (RTS/CTS)

Pokud je to možné, měli byste vždy používat hardwarové řízení přenosu. Komunikační program by měl mít volbu, která tuto metodu řízení zapíná. Musíte ji zapnout jednak v modemu (pomocí nějakého inicializačního řetězce) a jednak v ovladači. Komunikační program by měl provést obojí nastavení (je-li nakonfigurován správně).

Pokud se vám tímto postupem nepodaří dosáhnout hardwarového řízení přenosu, musíte je nastavit ručně. U modemu zkontrolujte, že to provede inicializační řetězec, anebo že je modem takto nastaven standardně. Pokud potřebujete nastavit ovladač zařízení, nejlépe se to provede přidáním příslušného příkazu do nějakého konfiguračního skriptu, který se spouští automaticky při startu systému – viz *Konfigurace při startu systému*. Pro každý sériový port, pro nějž chcete zapnout hardwarové řízení přenosu, musíte do nějakého konfiguračního skriptu přidat následující příkaz (příklad pro port *ttyS2*):

```
stty crtscts < /dev/ttyS2
```

nebo

```
stty -F /dev/ttyS2 crtscts
```

Zajímá-li vás, jak je řízení přenosu nastaveno, spusťte *minicom* (nebo jiný komunikační program) a zadejte příkaz *AT&V* (zobrazení konfigurace modemu) a hledejte odpověď *&K3* (hardwarové řízení přenosu). Pak aniž byste komunikační program ukončili ověřte příkazem *stty -F /dev/ttyS2 -a*, zda toto nastavení používá i ovladač (hledejte text *crtscts* bez symbolu minus znamenající vypnutí).

Nastavení rychlosti

Kromě nastavení řízení přenosu je nutné nastavit i rychlost – viz *Jakou rychlost použít*. Kromě toho je ještě třeba nastavit paritu a počet bitů. Obvykle se používá nastavení 8N1 (8 datových bitů, bez parity a jeden stop bit). Používáte-li PPP, musíte toto nastavení použít.

Ignorování signálu CD

Pokud modem neposílá signál CD a je vypnut příznak *clocal* (*stty* vypisuje *-clocal*), pak se programu nemusí podařit otevřít sériový port. Pokud se port nepodaří otevřít, program se může zaseknout a marně čekat na signál CD od modemu. Není ovšem problém napsat program tak, aby se port otevřel i v případě, že CD a *-clocal* říkají, že to nejde, takže ne vždy tím musí vznikat problémy.

Jedna možnost, jak se možným potížím vyhnout, je poslat modemu příkaz *AT&C*, čímž modemu řeknete, aby nechal signál CD trvale nastaven. Trvalé nastavení signálu CD nevadí při navazování odchozích hovorů, pokud ale přijímáte příchozí hovory, používá se tento signál občas (i když málokdy) k jejich detekci.

Minicom si při svém spuštění automaticky nastavuje příznak *clocal*, takže nemá problémy. Potíže měl ale Kermit verze 6.0.192. Problémy byly i ve starších verzích *stty* – pokud bylo nastaveno *-clocal* a modem neposílal signál CD, *stty* se zasekl a zdánlivě nebyla žádná možnost, jak *clocal* nastavit (kromě použití *minicom*u, jenže ten při svém ukončení obnoví původní nastavení). Jedna možnost je pomocí *minicom*u poslat modemu příkaz *AT&C* a pak *minicom* ukončit bez resetování modemu, takže signál CD zůstane nastaven. Pak už budete moci použít *stty*.

K čemu je stty?

Program *stty* je něco podobného jako *setserial*, ale slouží k nastavení přenosové rychlosti, řízení přenosu a dalších parametrů sériového portu. Příkazem *stty -F /dev/ttyS2 -a* zjistíte, jak je nastaven port *ttyS2*. Většina nastavení se modemu vůbec netýká a vztahuje se pouze na textové terminály (některé parametry mají význam pouze pro historické terminály ze 70. let). Komunikační program by měl automaticky umět nastavit parametry potřebné pro modem, proto také většinou program *stty* nebudete potřebovat a proto o něm zde příliš nemluvíme. Někdy se ale tento program hodí při diagnostice potíží. Více se o něm dočtete v dokumentech *Serial-HOWTO* a *Text-Terminal -HOWTO*.

Konfigurace modemu

Nalezení modemu

Než začnete trávit spousty času hledáním, jak modem nakonfigurovat, zkontrolujte nejprve, že byl opravdu nalezen a že mu můžete posílat AT příkazy. Doporučuji proto nejprve modem zkusit nastavit přímo pomocí komunikačního programu a zjistit, zda vůbec funguje. Pokud ano, můžete dále konfiguraci vylepšovat. Pokud ne, viz *Modem je připojen ale nefunguje*.

AT příkazy

Port, k němuž je modem připojen, musí být nakonfigurován, a stejně tak musí být nakonfigurován i modem. Konfigurace modemu se provádí pomocí AT příkazů, které se mu posílají stejným způsobem jako data.

Většina modemů používá AT příkazy, což jsou krátké ASCII příkazové sekvence, začínající vždy znaky AT. Znaky *AT* znamenají *ATtention (pozor), následuje příkaz*. Například *ATZ&K3* je řetězec obsahující dva příkazy: *Z* a *&K3*. Příkaz *Z* je zkrácená verze příkazu *Z0*, se kterou se spokojí většina modemů. Platí to i pro všechny ostatní příkazy končící nulou. Příkaz je ukončen znakem <return>. Delší příkazové řetězce (40 a více znaků) mohou vadit starším modemům, a je nutné je rozdělit. Můžete používat velká i malá písmena.

Bohužel, AT příkazy existují v řadě různých variant, takže co funguje pro jeden modem, nemusí fungovat pro jiný. Proto taky nemůžeme zaručit, že dále uvedené AT příkazy budou fungovat i s vaším modemem. Jednotlivé příkazy posílá modemu buď automaticky komunikační program, nebo je můžete poslat vy sami ručně. Většina komunikačních programů nabízí možnost upravit a uložit inicializační řetězec, který program používá. Samotný modem má v sobě rovněž uloženu svou konfiguraci (takzvaný profil). Můžete jej podle potřeb upravit. Ve většině případů můžete mít v modemu vytvořeno více profilů a zvolit vždy ten, který právě potřebujete.

Pokud máte k modemu návod, měl by obsahovat i seznam AT příkazů. Modemy 3Com (a možná i jiné) mají vestavěnou nápovědu, kterou vyvoláte příkazem *AT$*.

Popis AT příkazů najdete také na mnoha různých stránkách na Internetu. Nejprve hledejte na stránkách výrobce vašeho modemu. Pokud to nepomůže, zkuste hledat nějaké řetězce, které se

v AT příkazech vyskytují, například *&C1*, *&D3* a podobně. Tím byste měli nalézt stránky obsahující seznamy AT příkazů a ne stránky, které o nich jen obecně hovoří. Můžete také vyzkoušet některé ze stránek uvedených mezi *Odkazy*. Nezapomínejte ale, že AT příkazy se mohou u různých modemů lišit.

Uložení a vyvolání inicializačních řetězců

Následující příklady používají množinu AT příkazů kompatibilních s modemy Hayes. Všechny příkazové řetězce musí vždy začínat znaky *AT*, například *AT&C1&D3^M* (*^M* je znak <return>). Po zapnutí se modem nastaví podle konfigurace, kterou má uloženu v paměti. Pokud vám tato konfigurace bude vyhovovat, nemusíte už dále nic měnit.

Pokud konfigurace nevyhovuje, můžete ji buď jednorázově změnit, nebo po každém zapnutí poslat modemu vlastní inicializační řetězec. To normálně dělá komunikační program. Co přesně modemu pošle, závisí na tom, jak jej nastavíte. Komunikační program by vám měl umožnit libovolné úpravy inicializačního řetězce. V některých případech se komunikační program zeptá, jaký typ modemu používáte, a pošle inicializační řetězec, který je (podle něj) pro daný modem nejvhodnější.

Konfigurace modemu po zapnutí je rovněž popsána nějakým inicializačním řetězcem, který se označuje jako profil. Když komunikační program pošle modemu další řetězec, dojde ke změně výchozí konfigurace modemu. Poslaný inicializační řetězec může obsahovat třeba jen dva příkazy a změní se tak pouze dvě nastavení výchozí konfigurace. Existují příkazy, kterými můžete v modemu vyvolat nastavení podle zvoleného uloženého profilu, a jedním příkazem tak modem úplně přenastavit.

Moderní modemy mívají uloženo více profilů, ze kterých si můžete vybrat. Můj modem obsahuje dva tovární profily (0 a 1), které nejde změnit, a dva uživatelské profily (0 a 1), které může uživatel upravit a uložit. Jiné modemy povolují více profilů. Profily zobrazíte příkazem *&V*. Při zapnutí modemu se nahraje jeden z uživatelských profilů. Pokud zadáte příkaz *&Y0* (nebo jen *Y0* na modemech 3Com), bude se při zapnutí automaticky používat profil 0.

Dalším příkazem můžete aktivovat některý z uložených profilů. Můžete jej zadat v rámci inicializačního řetězce. Samozřejmě pokud budete požadovat nahrání toho profilu, který se nahrál automaticky po zapnutí modemu, nestane se nic (pokud jste od zapnutí konfiguraci modemu nějak nezměnili). Protože se nastavení modemu může měnit, je vždy rozumné nějaký inicializační řetězec poslat – i kdyby měl pouze vyvolat jeden z uložených profilů.

Příklady nahrání profilů: *Z0* nahraje uživatelský profil 0 a provede reset modemu (zavěšení a podobně). *&F1* nahraje tovární profil 1.

Jakmile modem nastavíte tak, jak potřebujete (například nahráním továrního profilu a jeho upravením), můžete aktuální konfiguraci uložit jako uživatelský profil. Příkaz *&W0* uloží aktuální konfiguraci jako uživatelský profil 0.

Řada uživatelů se neobtěžuje s ukládáním konfigurace do modemu a namísto toho při každém použití modemu posílají delší inicializační řetězec. Další možnost je použít v inicializačním řetězce příkaz např. *&F1* (vyvolávající tovární nastavení 1) následovaný několika příkazy, který toto nastavení podle potřeb upraví. Protože tovární nastavení modemu není možné změnit, zamezí se tak tomu, aby kdokoliv změnil (a uložil) konfiguraci založenou na uživatelském profilu.

Inicializační řetězce uváděné jako příklady u některých komunikačních programů mohou obsahovat znaky, které netvoří platné příkazy pro modem. Jde o příkazy samotného komunikačního programu a nebudou se posílat modemu. Například znak ~ může znamenat krátkou pauzu.

Kde je uložen inicializační řetězec

To záleží na použitém komunikačním programu. Například:

- Gnome: spusťte *pppsetup*
- wvdial: upravte soubor */etc/wvdial.conf*
- minicom: stiskněte ^Ao (nebo jen Alt+o) a zvolte „Modem and Dialing“

Další AT příkazy

Všechny řetězce musí začínat znaky AT. Dále uvádíme několik AT příkazů, které se obvykle uvádějí v inicializačním řetězci (pokud nejsou standardně nastaveny ve výchozí konfiguraci modemu).

- E1 zapíná echo
- Q0 zapíná ohlašování kódů výsledků
- V1 zapíná podrobný výpis kódů výsledků
- S0=0 modem nebude přijímat hovory

A ještě několik dalších příkazů:

- &C1 signál CD bude aktivní jen je-li modem spojen
- &S0 signál DSR bude trvale aktivní
- X3 při vytáčení se nečeká na přítomnost vytáčecího tónu (používá se v situacích, kdy na telefonní lince není vytáčecí tón)

Jaké je aktuální nastavení modemu?

Profil modemu můžete zobrazit pomocí programu *minicom*. Je rozumné, aby v té době nepoužívaly port modemu žádné jiné procesy. Pokud už máte *minicom* pro použití s modemem nastaven, můžete na příkazovém řádku zadat *minicom -o*, čímž se program spustí bez obnovení uloženého profilu modemu. Pak příkazem *AT&V* zobrazíte profil modemu. Abyste *minicom* ukončili bez narušení aktuálního profilu, použijte příkaz *q*, který ukončí program bez resetu modemu.

Tento postup nemusí z různých důvodů fungovat. Pokud je modem nastaven tak, aby nevypisoval výsledkové kódy, nemusí vypsat ani vlastní profil. Pokud na portu modemu ve stejné době běží i jiný proces, můžete uvidět jen část profilu.

Pokud port modemu používají i další procesy a vy je zabijete, může dojít k resetu modemu a nemusíte uvidět původní profil. Stává se to, pokud zabijete proces *getty* (nebo jeho následníky *login* či *bash*) a máte nastaveno *&D3*. Zabitím procesu *getty* (nebo dalších) se shodí signál DTR a dojde k resetu modemu do iniciálního stavu. Abyste zabránili opětovnému spuštění procesu *getty*, zakomentujte jej v */etc/inittab*, spusťte *init q* a teprve pak jej zabijte.

Režimy práce modemu

Protože pro posílání AT příkazů na modem slouží stejný kanál jako pro přenos dat, je nutné umět odlišit AT příkazy od dat.

Po zapnutí se modem nachází v příkazovém režimu (označuje se jako *command mode*, *terminal mode*, *idle state* nebo *AT-command mode*). Cokoliv, co z počítače pošlete na modem, bude považováno za AT příkazy. Jakmile pošlete příkaz pro vytáčení (*ATD...*), modem vytočí číslo a spojí se s modemem na protější straně. Nyní je v on line datovém režimu a odesílá a přijímá data. Pokud se nyní modemu pokusíte poslat nějaký AT příkaz, bude jej ignorovat a namísto toho jej odešle na druhou stranu. Jedinou výjimkou je „únikový“ příkaz. Jde o sekvenci znaků +++ se správ-

nou prodlevou před a po nich. Tato prodleva slouží k tomu, aby modem mohl odlišit tento příkaz od sekvence znaků +++ v přenášených datech.

Zatím tedy máme dva režimy – příkazový a on line datový. Třetím důležitým stavem je on line příkazový režim, kdy modem udržuje spojení s protistranou, ale nepřijímá a neodesílá data a cokoliv přijatého z počítače interpretuje jako AT příkazy. Tento režim se nastavuje po přijetí únikové sekvence +++ nebo shozením signálu DTR v případě, že je modem nastaven na *&D1*. V tomto režimu je možné modemu posílat příkazy včetně příkazů, které tento režim ukončí a vrátí se do některého ze dvou předchozích.

Kromě toho existují i další stavy – vytáčení a iniciální komunikace s protějším modemem, ty však za normálních okolností vedou k přechodu do on line datového režimu. Pokud ne, modem zavěsí a vrátí se do počátečního příkazového režimu.

Sériová zařízení

Postup pro vytvoření zařízení v adresáři */dev* je popsán v dokumentu *Serial-HOWTO* v části *Creating Devices In the /dev Directory*.

Devfs (nový Souborový systém zařízení)

Jedná se o nové rozhraní k zařízením v Linuxu. Jeho použití je možné počínaje jádry 2.4. Je efektivnější než klasické rozhraní a snáze se s jeho pomocí pracuje s velkým počtem zařízení. Zároveň došlo ke změně názvů jednotlivých zařízení, je ale možné i nadále používat staré názvy. Podrobnější popis najdete na adrese http://www.atnf.csiro.au/~rgooch/linux/docs/devfs.html.

Pokud používáte nové názvy, změnily se takto: *ttyS2* na *tts/2* (sériový port), *tty3* na *vc/3* (virtuální konzola), *ptyp1* na *pty/m1* (PTY master), *ttyp2* na *pty/s2* (PTY slave). *tts* je nyní adresář, obsahující jednotlivé soubory zařízení – *0*, *1*, *2* atd. Nové názvy jsou stále umístěny v adresáři */dev*, ale je možné je přesunout i jinam.

Názvy a čísla sériových zařízení

Zařízení v Linuxu (pokud nepoužíváte nový *devfs*) mají hlavní a vedlejší číslo. Sériové porty *ttySx* mají hlavní číslo 4. Můžete to vidět (spolu s vedlejšími čísly), když v adresáři */dev* zadáte příkaz *ls -l ttyS**.

Každý sériový port měl v minulosti přidruženo také zařízení *cua*, které se chovalo poněkud jinak. Například portu *ttyS2* odpovídalo zařízení *cua2*. Hlavní číslo *cua* zařízení bylo 5, vedlejší čísla začínala od 64. V adresáři */dev* možná stále najdete zařízení *cua**, dnes už se ale nepoužívají. Jejich ovladač se chová poněkud jinak než u zařízení *ttyS**.

DOS a Windows používají názvy *COMx*, program *setserial* pak názvy *tty00*, *tty01*, atd. Nepleťte si tyto názvy s */dev/tty0*, */dev/tty1*, atd. – to jsou virtuální konzoly, nejde o sériové porty. Následující tabulka platí ve „standardním" případě, pokud máte nějakou atypickou konfiguraci, může to být jinak. Pokud používáte *devfs*, neexistuje nic takového jako hlavní a vedlejší čísla zařízení. Jsou-li sériové porty na PCI sběrnici, používají jiné vstupně-výstupní adresy.

DOS	Linux	hl. č.	vedl. č.	adresa	devfs	devfs usb	acm modem
COM1	/dev/ttyS0	4	64	3F8	/dev/tts/0	/dev/usb/tts/0	/dev/usb/acm/0
COM2	/dev/ttyS1	4	65	2F8	/dev/tts/1	/dev/usb/tts/1	/dev/usb/acm/1
COM3	/dev/ttyS2	4	66	3E8	/dev/tts/2	/dev/usb/tts/2	/dev/usb/acm/2
COM4	/dev/ttyS3	4	67	2E8	/dev/tts/3	/dev/usb/tts/3	/dev/usb/acm/3

USB porty

Sériové porty na USB sběrnici se označují */dev/ttyUSB0*, */dev/ttyUSB1* atd. Při použití *devfs* mají názvy */dev/usb/tts/0*, */dev/usb/tts/1* atd. Další informace naleznete v dokumentaci jádra v souborech *acm* a *usb-serial*.

Linkování ttySx na /dev/modem

U některých instalací se vytvářejí dvě nová zařízení – */dev/modem* pro modem a */dev/mouse* pro myš. Obě jsou symbolický odkaz na příslušné sériové zařízení, které zvolíte při instalaci. (Pokud nemáte sériovou myš, bude */dev/mouse* odkaz na jiné zařízení.)

Dříve se nedoporučovalo vytvářet */dev/modem* jako odkaz na sériové zařízení, protože systém zámkových souborů neuměl poznat, že jde o zařízení např. */dev/ttyS2* a mohlo docházet ke konfliktům při přístupu k modemu. Nový systém zámků už tímto problémem netrpí a použití takových odkazů tedy nebude způsobovat problémy.

Zajímavé programy

setserial

Následující text najdete v dokumentech *Modemy*, *Serial-HOWTO* i *Text-Terminal-HOWTO* v drobných modifikacích.

Úvod

Pokud máte notebook (PCMCIA modem), přečtěte si nejprve část *Notebooky a PCMCIA*. Program *setserial* umožňuje ovladači sériového portu říct, jaké jsou vstupně-výstupní adresy portů, jaké používají přerušení, jaký UART čip a podobně. Protože je vysoká pravděpodobnost, že sériové porty budou nalezeny a detekovány automaticky, většina lidí tento program nikdy nepoužije. Program samozřejmě nebude fungovat, pokud jádro neobsahuje podporu sériových portů, ať už přímo nebo jako modul.

Programem *setserial* lze také zjistit, jak je ovladač nastaven. Kromě toho jej lze použít k detekci hardwarových portů a zjištění UART a přerušení, tato detekce má však svá omezení (viz *Detekce*). Program neumí nastavit adresu nebo přerušení v hardwaru PnP portů (může to ale umět sériový ovladač). Neumí také přečíst data v konfiguračních registrech PnP zařízení.

Pokud máte jeden nebo dva sériové porty, obvykle budou správně nastaveny bez jakékoliv práce. Pokud ale přidáváte další porty (např. modem), budete je možná muset nastavit. Kromě manuálových stránek programu *setserial* se podívejte i na dokumentaci v adresáři */usr/doc/setserial* nebo */usr/share/doc/setserial*. Zde byste se měli dozvědět, jak *setserial* pracuje konkrétně ve vaší distribuci.

Program se často spouští automaticky při startu systému, aby mohl ovladači sdělit nastavení portů. Program bude pracovat pouze tehdy, obsahuje-li jádro podporu sériových portů, nebo je-li sé-

riový ovladač nahrán jako modul. Pokud později dojde k odstranění modulu, ztratí se nastavené údaje. Moderní distribuce někdy obsahují skripty, které umějí nastavení přečíst a obnovit. Pokud ne, musíte po novém nahrání modulu spustit *setserial* znovu. Kromě spuštění *setserial* při startu systému z nějakého skriptu se něco podobného spouští také dříve, při zavádění sériového ovladače. Sledujete-li výpis hlášení při startu systému, uvidíte, že se program *setserial* jakoby spustil dvakrát, což je také víceméně pravda.

Program nenastavuje adresy a přerušení samotného hardwaru sériového portu. To se řeší buď přepínači nebo pomocí PnP. Programu *setserial* musíte říct stejné hodnoty, jaké jsou v hardwaru nastaveny. Nepokoušejte se prostě vymyslet nějaké pěkné hodnoty, které by se vám líbily. Pokud znáte adresu portu a neznáte jeho přerušení, můžete se programem *setserial* pokusit přerušení zjistit.

Seznam možných použití získáte po zadání příkazu *setserial* bez parametrů. Nedozvíte se tak ale jednoznakové přepínače jako třeba *-v*, který se často používá při hledání problémů. Všimněte si také, že program *setserial* označuje vstupně-výstupní adresu jako „port“. Zadáte-li příkaz:

```
setserial -g /dev/ttyS*,
```

dozvíte se, jak je sériový ovladač nastaven. Pokud u některého z portů uvidíte „UART: unknown“, znamená to, že program nebyl schopen zjistit typ UART čipu (řídicí čip sériového portu) a typicky to znamená, že takový port fyzicky neexistuje a všechny ostatní údaje nemají význam. Pokud daný port opravdu existuje, *setserial* jej nepoznal a je třeba to napravit.

Pokud kromě parametru *-g* použijete ještě *-a*, dozvíte se ještě dodatečné informace, které potřebuje (a rozumí jim) jen málokdo – většinou totiž správně fungují standardní nastavení. Za normálních okolností zobrazuje *setserial* správné informace o tom, jak je hardware portu nastaven, pokud ale máte nějaké problémy, může se stát, že se program plete. Můžete jej klidně spustit a „vytvořit“ sériový port s libovolným nastavením adresy, přerušení a UART. Při příštím spuštění programu tento vymyšlený port normálně uvidíte. Je jasné, že pokud se jej pokusíte použít, ovladač nebude fungovat správně. A tak je to s jakýmkoliv nastavením programu *setserial* – tedy až na výjimky. Pokud se pokusíte přidělit jednomu portu adresu, kterou už používá jiný port (například *3e8*), nedovolí to. Pokud se ale pokusíte přiřadit adresu *3e9*, tak už to povolí. Bohužel, tato adresa je ve skutečnosti také obsazena, protože adresa *3e8* v sobě ve skutečnosti skrývá celý rozsah adres *3e8* až *3ef*. Důležité tedy je, abyste programu vždy nastavovali správné hodnoty.

I když nastavené hodnoty se při vypnutí počítače ztratí, mohou být uloženy v konfiguračním skriptu a automaticky se obnovit při novém spuštění počítače. V novějších verzích se změny provedené programem *setserial* automaticky zapisují do konfiguračního souboru. Ve starších verzích se konfigurační soubor mění pouze explicitním zásahem.

Detekce

Než začnete porty detekovat programem *setserial*, vyzkoušejte *scanport*, který by vám měl říct, kde porty hledat. Tento program se pokusí odhadnout, co je za zařízení na různých adresách, nedetekuje ale přerušení těchto zařízení. Je to ale rychlý začátek. I když by mohl způsobit zablokování celého počítače, zatím se mi to ještě nestalo. Program *scanport* je součástí distribuce Debian a v jiných se neobjevuje.

Se správnými parametry se program *setserial* pokusí detekovat sériový port na zadané adrese, ovšem tu adresu musíte uhodnout. Pokud jej necháte například detekovat port */dev/ttyS2*, bude hledat pouze na adrese, kde by tento port měl normálně být (0x2f8). Řeknete-li mu, že je na jiné adrese, bude jej hledat tam.

Cílem detekce je zjištění, zda se na dané adrese nachází UART čip, a pokud ano, jaké přerušení používá. Program *setserial* použijte až jako poslední instanci, protože existují rychlejší metody – například program *wvdialconf* zkouší nalézt modem podle hlášení vypsaných při startu systému, nebo pomocí příkazu *pnpdump --dumpregs*. Chcete-li detekovat fyzický hardware, použijte například:

```
setserial /dev/ttyS2 -v autoconfig
```

Jestliže se ve výsledném hlášení objeví UART například 16550A, je všechno v pořádku. Pokud program vypíše neznámý typ UART, pak se na dané adrese pravděpodobně sériový port nenachází. Některé levné sériové řadiče se ovšem správně neidentifikují, takže i v případě, že program nalezne „UART unknown“, existuje možnost, že port ve skutečnosti existuje.

Kromě automatické detekce UART čipu umí *setserial* detekovat i použité přerušení. Ani tato metoda však nemusí fungovat spolehlivě. Stalo se mi, že napoprvé detekoval chybné přerušení, a až při druhém spuštění zjistil přerušení správně. Od *setserial* verze 2.15 je možné výsledek poslední detekce automaticky uložit do konfiguračního souboru */etc/serial.conf*, takže se použije při dalších spuštěních systému. Kromě toho se detekce UART provádí automaticky při zavádění sériového ovladače, zde zjištěná přerušení však nemusí být správná. Druhý výpis těchto informací při startu je výsledkem inicializačního skriptu, který se typicky nepokouší nic detekovat a zobrazené informace tedy nemusí odpovídat skutečnému nastavení hardwaru. Jde pouze o konfigurační data, která někdo zapsal do inicializačního skriptu nebo souboru *serial.conf*.

Může se stát, že dva sériové porty používají stejnou vstupně-výstupní adresu. To samozřejmě není povoleno, nicméně se to občas stává. Pak detekce nalezne pouze jeden port, i když jsou tam ve skutečnosti dva. Pokud navíc porty používají různá přerušení, bude nejspíš detekovat přerušení 0.

Konfigurace při startu systému

Když jádro nahrává modul sériového ovladače (nebo je-li ovladač vestavěn přímo do jádra), provádí se automatická detekce pouze portů *ttyS0-3* a ovladač používá pouze přerušení 3 a 4 (bez ohledu na to, jaká přerušení hardwaru ve skutečnosti používá). Ve výpisech při spouštění systému se objeví stejné údaje, jako by byl spuštěn program *setserial*.

Aby se odstranily možná chybná nastavení přerušení (a případně i z jiných důvodů) může existovat soubor, který spustí program *setserial* později znovu. Pokud ovšem i v tomto souboru budou přerušení definována chybně, nic se tím nenapraví. Tento skript by měl být spuštěn hned zpočátku startu systému, ještě předtím, než se spustí jakékoliv procesy používající sériové porty. Ve většině případů jsou distribuce udělány tak, že se program *setserial* spouští hned při startu prostřednictvím nějakého inicializačního skriptu. Podrobnosti o tom, jak je spouštění řešeno v konkrétní distribuci, byste měli najít v dokumentaci v souboru *setserial...* v adresáři */usr/doc* nebo */usr/share/doc*.

Než začnete upravovat konfigurační soubor, můžete vyzkoušet „doporučené“ hodnoty manuálním spuštěním programu *setserial*. V některých případech se použité parametry automaticky uloží do souboru */etc/serial.conf*. Pokud tedy všechno bude fungovat správně, nemusíte už nic modifikovat.

Konfigurační skripty

Naším cílem je upravit (nebo vytvořit) skript v adresáři */etc*, který při startu systému spustí program *setserial*. Většina distribucí takový nějaký soubor obsahuje, i když nemusí být umístěn v adresáři */etc*. Navíc *setserial* verze 2.15 a vyšší často používá soubor */etc/serial.conf*, a není tak nutné parametry editovat přímo ve spouštěcím skriptu. Navíc přímé spuštění programu *setserial* z příkazového řádku může provedená nastavení rovnou zapsat i do konfiguračního souboru.

Před verzí 2.15 tedy musíte měnit spouštěcí skript. Od verze 2.15 máte na výběr ze tří možností: změnit skript, změnit */etc/serial.conf* anebo spustit *setserial* z příkazového řádku, což může vést k automatické změně souboru *serial.conf*. Kterou metodu použijete, závisí na konkrétní distribuci a nastaveních.

Editace skriptu (pro verzi < 2.15)

Před verzí 2.15 neexistoval soubor */etc/serial.conf* sloužící ke konfiguraci programu *setserial*. Potřebovali jste tedy najít skript, z nějž se *setserial* při startu spouštěl, a upravit jej. Pokud neexistoval, museli jste jej vytvořit. Pokud takový soubor existuje, pravděpodobně se nachází někde v adresáři */etc*. Konkrétně RedHat < 6.0 jej ale umisťoval v adresáři */usr/doc/setserial* a nejprve jste jej do adresáře */etc* museli zkopírovat. Můžete jej zkusit najít příkazem *locate*, například *locate *serial**.

Běžně se používal skript */etc/rc.d/rc.serial*. Debian používal skript */etc/rc.boot/0setserial*. Další možnost může být soubor */etc/rc.d/rc.local*, ale to není moc rozumné, protože skript nemusí být spuštěn včas. Stává se, že nějaký program zkouší otevřít sériový port ještě předtím, než se provede *rc.local* a program pak hlásí chyby komunikace. Dnes se skript obvykle nachází v */etc/init.d/*, ale tento adresář není určen k přímé editaci.

Pokud skript existuje, měl by obsahovat celou řadu zakomentovaných příkladů. Jejich odkomentováním a/nebo modifikací byste měli být schopni nastavit systém správně. Zkontrolujte, že uvádíte správnou cestu k programu *setserial* a platné názvy zařízení. Nastavení můžete otestovat tak, že skript jednoduše spustíte. Tento způsob testování je výrazně rychlejší, než opakované starty systému.

Používáte-li *setserial* verze >= 2.15 (a za předpokladu, že distributor implementoval změny, což například RedHat neudělal), může být tato metoda nastavení složitější, protože spouštěcí soubor */etc/init.d/setserial* nebo podobný není určen k přímé editaci. V tomto případě raději zvolte dále popsanou novou metodu konfigurace.

Pokud chcete, aby *setserial* automaticky zjistil UART a přerušení portu *ttyS3*, zadáte něco jako:

```
/sbin/setserial  /dev/ttyS3 auto_irq skip_test autoconfig
```

Toto nastavení proveďte pro všechny porty, které chcete nakonfigurovat. Dejte pozor, abyste uváděli jména zařízení, která v počítači opravdu existují. V některých případech (u hloupého hardwaru) nemusí automatická konfigurace fungovat správně. Pokud znáte typ UART a hodnotu přerušení, můžete je nastavit explicitně, například:

```
/sbin/setserial /dev/ttyS3 irq 5 uart 16550A  skip_test
```

Novější konfigurační metoda prostřednictvím /etc/serial.conf

Počínaje verzí 2.15 se neprovádí přímá editace spouštěcího skriptu, namísto toho se konfigurace načítá ze souboru */etc/serial.conf*. V některých případech dokonce ani nebudete muset tento soubor editovat, protože příkaz *setserial* po spuštění na příkazovém řádku může automaticky provést potřebnou editaci souboru.

Smyslem tohoto mechanismu je, abyste nemuseli editovat žádný konfigurační soubor a přitom se konfigurace systému po novém spuštění zachovávala. Toto řešení má ale jistá úskalí, protože soubor *serial.conf* není editován přímo programem *setserial*. Různé distribuce to řeší různě a odtud pramení možné problémy.

Děje se toto: když počítač vypínáte, skript, který při jeho startu spustil program *setserial*, se spouští znovu, tentokrát ale provádí operace definované v části *stop*. Konkrétně spustí *setserial*, zjistí, ja-

ká jsou aktuální nastavení portů, a tyto hodnoty zapíše do *serial.conf*. Když tedy spustíte *setserial*, změny se do konfiguračního souboru nezapíší hned, ale až při (korektním) vypnutí systému.

Jistě si umíte představit, jaké potíže mohou nastat. Řekněme, že systém nebude korektně vypnut (například někdo vypne proud) – pak se změny neuloží. Na druhé straně může nastat situace, že budete s programem *setserial* experimentovat a zapomenete nakonec obnovit původní stav, nebo jej obnovíte špatně. Pak se uloží tato „experimentální" nastavení.

Upravíte-li soubor *serial.conf* ručně, změny se po vypnutí ztratí, protože obsah souboru se přepíše podle stavu při vypínání. Existuje možnost, jak funkci automatického uložení konfigurace vypnout – stačí na prvním řádku souboru *serial.conf* odstranit text „###AUTOSAVE###". V některých instalacích se tato volba odstraní automaticky hned při prvním vypnutí počítače po instalaci. V souboru *serial.conf* by měly být uvedeny komentáře, které chování daného systému popisují.

Dnes se *setserial* při startu systému nejčastěji spouští skriptem */etc/init.d/setserial* (Debian) nebo */etc/init.d/serial* (RedHat) nebo podobným. Tento skript byste ale neměli editovat. Pro verzi *setserial* 2.15 obsahuje RedHat 6.0 soubor */usr/doc/setserial-2.15/rc.serial*, který stačí přesunout do */etc/init.d* a zajistí se tak spouštění programu při startu systému.

Chcete-li nějaký port vypnout, nastavte mu *uart none*. Formát souboru */etc/serial.conf* jsou vlastně jednotlivé parametry příkazu *setserial* tak, jak se zadávají na příkazovém řádku. Pokud nepoužíváte automatické uložení nastavení, musíte soubor */etc/serial.conf* upravit ručně.

Pokud chcete aktuální nastavení programu uložit do konfiguračního souboru, aniž byste vypínali systém, spusťte skript */etc/init.d/(set)serial stop*. Parametr *stop* říká, že se má konfigurace uložit, sériové porty budou fungovat i nadále.

V některých případech můžete narazit na směs konfigurace starou i novou metodou, při spuštění systému by se však měla provést jen jedna. Distribuce Debian označuje starší soubory jako *...pre-2.15*.

Přerušení

Standardně porty *ttyS0* a *ttyS2* sdílejí přerušení 4, porty *ttyS1* a *ttyS3* sdílejí přerušení 3. Sdílení sériových přerušení (tedy jejich současné používání více běžícími programy) ovšem není možné, pokud: 1. nemáte jádro 2.2 a vyšší, 2. pokud jste je nepřeložili s podporou přerušení a 3. pokud to nepodporuje hardware sériového portu. Viz *Sdílení přerušení a jádra 2.2+*. Pokud máte pouze dva sériové porty, *ttyS0* a *ttyS1*, všechno je v pořádku, protože u neexistujících zařízení ke konfliktům přerušení nemůže dojít.

Pokud přidáte interní modem a ponecháte si porty *ttyS0* a *ttyS1*, měli byste se pokusit najít nějaké nepoužívané přerušení, a nastavit je jak na sériovém portu (modemové kartě), tak prostřednictvím programu *setserial* v ovladači sériového portu. Nepoužíváte-li přerušení 5 pro zvukovou kartu, je to vhodný kandidát na použití modemem. Nastavení přerušení v hardwaru provedete pomocí *isapnp*, PnP BIOSu, nebo doplněním Linuxu o podporu PnP. Chcete-li zjistit, jaká přerušení mohou být volná, zkuste *man setserial* a hledejte například *IRQ 11*.

Notebooky a PCMCIA

Pokud máte notebook, přečtěte si popis konfigurace sériového portu v dokumentu *PCMCIA-HOWTO*. Je-li sériový port součástí základní desky, používá se *setserial* stejně jako u normálních počítačů. Pokud ale používáte PCMCIA karty (například PCMCIA modem), jde o jinou situaci. Konfigurace systému PCMCIA by měla automaticky spustit *setserial*, takže byste jej nemuseli spouštět ručně. Pokud jej spustíte, jeho konfigurace nemusí vyhovovat a mohou vzniknout problémy. Funkce automatického ukládání souboru *serial.conf* by neměla konfiguraci PCMCIA karty ukládat. Samozřejmě můžete příkaz *setserial* použít k tomu, abyste zjistili parametry portu.

isapnp

Program *isapnp* slouží ke konfiguraci PnP zařízení na ISA sběrnici. Distribuuje se jako součást balíku *isapnptools*, který obsahuje i další program, *pnpdump*, jenž nalezne všechna ISA PnP zařízení a zobrazí jejich konfigurační volby ve formátu, vhodném pro přidání do konfiguračního souboru */etc/isapnp.conf*. Můžete jej také spustit s parametrem *--dumpregs*, který zobrazí aktuální nastavení vstupně-výstupních adres a přerušení. Program *isapnp* můžete přidat do inicializačních souborů, takže se bude spouštět vždy při startu systému a nastaví ISA PnP zařízení. Můžete jej použít i v případě, že váš BIOS PnP zařízení nepodporuje. Další informace najdete v dokumentu *Plug-and-Play-HOWTO*.

wvdialconf

Program *wvdialconf* se pokouší zjistit, ke kterému sériovému portu (*ttyS?*) je připojen modem. Vytvoří také konfigurační soubor pro program *wvdial*, který slouží pro připojení protokolem PPP k poskytovateli internetového připojení. Abyste mohli *wvdialconf* použít, nemusíte mít PPP nainstalováno. Můžete jej pouze použít k nalezení modemu. Vytvoří také „vhodný“ inicializační řetězec, i když ne vždy musí mít pravdu. Program se spouští bez parametrů a jeho použití je tedy velmi jednoduché – musíte mu pouze zadat jméno souboru, do nějž má doporučená inicializační data uložit, například *wvdialconf konfigurační_soubor*.

Testování modemu

Jste připraveni na volání ven?

Jakmile máte modem připojen a víte, jaký používá sériový port, můžete jej vyzkoušet. Máte-li už účet u nějakého poskytovatele, můžete zkusit program jako *wvdial* a připojit se k Internetu protokolem PPP.

Jako alternativu k tomuto postupu, který přece jen představuje příliš velký krok najednou, můžete použít dvoufázový postup: Nejprve modem vyzkoušejte bez použití PPP (například programy Minicom nebo Kermit). Teprve pokud modem funguje správně, zkuste *wvdial* (nebo jiný podobný program) a připojte se k Internetu. Opačná strategie je nejprve vyzkoušet připojení k Internetu a, teprve pokud nefunguje, vrátit se zpátky k programu Minicom či Kermit a vyzkoušet, zda vůbec funguje modem. Znalost programů Minicom nebo Kermit je užitečná také pro případ, že byste se chtěli někam připojovat přímo, bez připojení k Internetu. Abyste to mohli vyzkoušet, musíte nejprve získat nějaké telefonní číslo, kam je možné se modemem připojit (bez použití PPP). Takovou možnost nabízejí někdy knihovny pro přístup k elektronickému katalogu.

Dále si zkontrolujte, zda opravdu můžete začít vytáčet. Znáte sériový port modemu? Měli byste to vědět z doby, kdy jste sériové porty konfigurovali. Víte, jakou rychlost budete na portu používat? Stručný přehled naleznete v *Tabulce rychlostí*, nebo se podívejte na část *Jakou rychlost použít*. Pokud nemáte tušení, jakou rychlost zvolit, zkuste nastavit několikanásobek avizované maximální rychlosti vašeho modemu. Pokud narazíte na nabídku, kde budete moci zapnout hardwarové řízení toku, „RTS/CTS“, nebo něco podobného, tak to udělejte. Máte modem připojen k živé telefonní lince?

Nyní musíte zvolit komunikační program, který k vytáčení použijete. Mezi jednoduché programy patří *minicom*, *seyon* (X Windows) a *kermit*. Pro přístup na Internet protokolem PPP se používají *wvdial*, *kppp* (KDE), *gnome-ppp* (GNOME). Další podrobnosti naleznete v části *Komunikační programy*. Dále si ukážeme použití tří programů – *wvdial*, *minicom* a *kermit*.

Volání s wvdial

Program *wvdial* nejen naváže spojení se vzdáleným modemem, ale spustí také protokol PPP a přihlásí vás k účtu poskytovatele Internetu. Lze jej nakonfigurovat při instalaci, nebo programem *wvdialconf*. Podívejte se na manuálové stránky programu *wvdialconf* i *wvdial*. Než ale *wvdial* použijete, musíte udělat dvě věci, které nejsou v jeho dokumentaci popsány:

- Nastavit síťové služby. Podívejte se do stařičkého dokumentu *ISP-Hookup-HOWTO*, kde naleznete užitečné informace.
- Nastavit prohlížeč.

Volání s minicomem

Minicom je součástí většiny distribucí Linuxu. Abyste jej mohli nastavit, musíte jej spustit jako *root* příkazem *minicom -s*. Tím se dostanete přímo do konfigurační nabídky. Pokud program spustíte pouze příkazem *minicom* a pak jej nakonfigurujete, budete jej muset ukončit a pak znovu spustit, aby se nastavené změny projevily. Máte-li program spuštěn, *^A* zobrazí spodní stavový řádek. Na něm se dozvíte, že *^A Z* vyvolá nápovědu. (Protože jste *^A* už stiskli, stačí nyní stisknout *Z*.)

Řadu voleb není nutné pro jednoduché vytáčení nastavovat. V rámci konfigurace musíte nastavit pouze několik základních údajů: název sériového portu, k němuž je modem připojen, a rychlost sériového portu. Ty se nastavují v nabídce *Serial port setup*. Nastavte je. Pokud je to možné, nastavte i hardwarové řízení přenosu (RTS/CTS). Pak nastavení uložte. Při zadávání rychlosti byste měli narazit i na něco jako *8N1*, to nechejte beze změny – znamená to 8 datových bitů, žádná parita a jeden stop bit. Pokud nevíte, jakou rychlost použít, na zkoušku určitě bude fungovat nějaká nízká rychlost. Stiskem klávesy Enter nastavování ukončete a uložte konfiguraci jako výchozí (*dfl*). Pokud jste *minicom* nespustili s parametrem *-s*, budete jej muset nyní ukončit a spustit znovu, aby použil zvolená nastavení a inicializoval modem.

Teď můžete začít vytáčet. Nejprve se ale v hlavní obrazovce přesvědčete, že modem reaguje. Zadejte *AT* a stiskněte *Enter*. Měl by se objevit text *OK*. Pokud se neobjeví, není něco v pořádku a nemá cenu zkoušet vytáčení.

Pokud se objeví *OK*, podívejte se znovu na nápovědu a otevřete adresář telefonních čísel. Doplňte do něj telefonní číslo a další údaje a příkazem *dial* je zkuste vytočit. Druhá možnost je vytočit číslo ručně (příkazem *manual* a zadáním čísla na klávesnici). Pokud vytáčení nefunguje, sledujte chybová hlášení a zkuste zjistit, co není v pořádku.

Volání s kermitem

Novou verzi programu *kermit* najdete na adrese http://www.columbia.edu/kermit/. Předpokládejme nyní, že máte modem připojen k portu *ttyS3* a chcete s ním komunikovat rychlostí 115 200 b/s. Postup bude následující:

```
linux# kermit
C-Kermit 6.0.192, 6 Sep 96, for Linux
 Copyright (C) 1985, 1996,
  Trustees of Columbia University in the City of New York.
Default file-transfer mode is BINARY
Type ? or HELP for help.
C-Kermit>set line /dev/ttyS3
C-Kermit>set carrier-watch off
C-Kermit>set speed 115200
/dev/ttyS3, 115200 bps
```

```
C-Kermit>c
Connecting to /dev/ttyS3, speed 115200.
The escape character is Ctrl-\ (ASCII 28, FS)
Type the escape character followed by C to get back,
or followed by ? to see other options.
ATE1Q0V1                          ; zadejte inicializační řetězec + Enter
OK                                ; toto by měl modem odpovědět
```

Pokud modem odpoví na AT příkaz, můžeme předpokládat, že funguje správně. Nyní zkuste vytočit číslo příkazem:

```
ATDT7654321,
```

kde *7654321* je volané číslo. Pokud používáte pulsní volbu, použijte namísto příkazu *ATDT* příkaz *ATDP*. Jestliže se hovor spojí, modem funguje.

Návrat do nabídky programu provedete stisknutím klávesy Ctrl, pak obrácené lomítko, stále držíte control a nakonec stisknete C:

```
C-Kermit>quit
linux#
```

Tím jsme jenom vyzkoušeli jednoduchou metodu ručního volání. Za normálních okolností necháte *kermit* vytáčet na základě jeho databáze informací o modemech a pomocí funkce automatického vytáčení. Používáte-li například modem US Robotics (USR):

```
linux# kermit
C-Kermit 6.0.192, 6 Sep 1997, for Linux
 Copyright (C) 1985, 1996,
  Trustees of Columbia University in the City of New York.
Default file-transfer mode is BINARY
Type ? or HELP for help
C-Kermit>set modem type usr       ; Vyberete typ modemu
C-Kermit>set line /dev/ttyS3      ; Vyberete port
C-Kermit>set speed 115200         ; Nastavíte rychlost
C-Kermit>dial 7654321             ; Vytáčíte
 Number: 7654321
 Device=/dev/ttyS3, modem=usr, speed=115200
 Call completed.<BEEP>
Connecting to /dev/ttyS3, speed 115200
The escape character is Ctrl-\ (ASCII 28, FS).
Type the escape character followed by C to get back,
or followed by ? to see other options.

Welcome to ...

login:
```

Jakou rychlost použít

U moderních modemů nemáte možnost volit rychlost, jakou modem komunikuje po telefonní lince, protože si automaticky zvolí nejvyšší rychlost, která je za daných podmínek použitelná. Pokud je jeden modem pomalejší a druhý rychlejší, bude rychlejší modem přenášet rychlostí pomalejšího modemu. Na nekvalitní telefonní lince modemy rychlost automaticky sníží.

Rychlost přenosu po telefonní lince volí modemy automaticky, vy však volíte rychlost přenosu mezi modemem a počítačem. Někdy se tato rychlost označuje jako DTE rychlost (termín DTE znamená Data Terminal Equipment, v tomto případě váš počítač). Tuto rychlost musíte nastavit dostatečně vysoko, aby linka mezi modemem a počítačem nepředstavovala úzké místo. Nastavená rychlost bude zároveň nejvyšší možná rychlost přenosu přes telefonní linku, ve většině případů je však telefonní linka výrazně pomalejší.

U externího modemu je DTE rychlost, kterou se budou data přenášet kabelem mezi počítačem a modemem. U interního modemu je myšlenka analogická, protože interní modem rovněž emuluje sériový port. Může to vypadat podivně, že existuje nějaký limit rychlosti pro přenos mezi počítačem a modemovou kartou na rychlé sběrnici počítače. Tak to ale funguje, protože modemová karta typicky obsahuje vlastní sériový port, který už má limitovanou rychlost.

Rychlost a komprese dat

Jakou rychlost zvolit? Pokud byste nepoužívali kompresi dat, můžete DTE rychlost nastavit stejně jako rychlost modemu. Při použití datové komprese se ale data poslaná z počítače na modem komprimují a po telefonní lince se přenáší menší objem dat. Jestliže je například rychlost přenosu mezi počítačem a modemem 20 000 B/s a kompresní poměr je 2, po telefonní lince se bude přenášet pouze 10 000 B/s. Při kompresním poměru 2:1 tedy musí být DTE rychlost dvojnásobkem maximální přenosové rychlosti modemu. Při poměru 3: 1 už musí být trojnásobkem.

Kde rychlost nastavit?

Rychlost přenosu mezi počítačem a modemem obvykle nastavujete v konfigurační nabídce komunikačního programu. Rychlost přenosu mezi modemy nastavit nemůžete, tu si modemy volí samy jako maximální vyhovující rychlost. Ve skutečnosti můžete nastavit i rychlost přenosu mezi modemy zápisem do registru *S37* modemu, ale tento postup se nedoporučuje. Pokud by takto byla na obou modemech nastavena různá rychlost, nikdy by se spolu nedomluvily.

Nelze nastavit dostatečnou rychlost

Rychlosti nad 115,2k

Až do poloviny 90. let byla nejvyšší standardní rychlostí rychlost 115,2k. V roce 2000 už ale existují sériové porty podporující rychlosti 230,4k a 460,8k. Některé dokonce podporují rychlost 921,6k. Linux takové rychlost bohužel používá jen málokdy v důsledku nedostatku ovladačů. I rychlé porty pak pracují s rychlostí 115,2k, pokud není nainstalován speciální software. Abyste mohli tyto vysoké rychlosti dosáhnout, musíte si přeložit speciálně upravené jádro.

Výrobci sériových portů se bohužel nikdy nedohodli na standardní metodě podpory vysokých rychlostí, takže sériový ovladač musí umět podporovat různé typy hardwaru. Jakmile je umožněno použití vysokých rychlostí, standardní způsob jejich volby je nastavit hodnotu *baud_base* programem *setserial* na maximum (pokud to neprovede automaticky ovladač portu). Pak software nastaví dělitele na 1 a dosáhne se tak nejvyšší možné rychlosti. Všechny tyto funkce už by měly být novými jádry podporovány.

Nestandardní způsob spočívá v nastavení hodnoty dělitele na nějakou vysokou hodnotu – v tomto případě už vůbec nejde dělit. Je to prostě speciální kód, kterým se portu říká, jakou rychlostí má pracovat. V těchto případech musíte mít speciálně upravené jádro.

Jedna z úprav jádra pro podporu této druhé metody nastavení vysokých rychlostí se jmenuje *shsmod* (Super High Speed Mode). Existuje ve verzi pro Linux i pro Windows, viz http://www.dev-drv.com/shsmod/. Existuje rovněž modul pro čip VIA VT82C686, viz http://www.kati.fi/viahss/.

Interní modemy jen zřídka podporují rychlosti vyšší než 115,2k.

Jak se rychlost nastavuje v hardwaru portu

Změna rychlosti se provádí změnou hodinové frekvence sériového portu. Samotná změna se samozřejmě neprovádí fyzicky změnou rychlosti oscilátoru, ale „vydělením" jeho rychlosti. Dělíte-li například dvěma, prostě se ignoruje každé druhé „tiknutí" hodin a rychlost se tak sníží na polovinu. Chcete-li tedy snížit rychlost, musíte pouze nastavit hodnotu dělitele. Tuto hodnotu nastavuje zápisem do příslušného registru ovladač sériového portu.

Pokud hodiny běží na nejvyšší rychlosti 115 200 b/s (což je obvyklé), tak dělitelé pro různé rychlosti vypadají takto: 1 (115,2k), 2 (57,6k), 3 (38,4k), 6 (19,2k), 12 (9,6k), 24 (4,8k), 48 (2,4k), 96 (1,2k) atd. Ovladač sériového portu nastavuje rychlost pouze tak, že samotnému portu pošle hodnotu dělitele. Touto hodnotou se dělí maximální rychlost portu a výsledkem je snížení rychlosti (dělitel 1 samozřejmě znamená nejvyšší rychlost).

Výše popsaný postup neplatí vždycky, protože u některých portů se rychlosti na 115,2k nastavují tak, že se použije hodně vysoká hodnota dělitele. Tuto výjimku mějme na paměti a můžeme pokračovat ve čtení. Za normálních okolností když nastavíte v komunikačním programu rychlost 115,2k, ovladač sériového portu nastaví na portu dělitele 1, což znamená největší rychlost.

Pomineme-li metodu, kdy se vysoké rychlosti přenosu nastavují prostřednictvím hodně vysoké hodnoty dělitele, standardně vypadá situace takto: Budete-li mít modem s maximální rychlostí řekněme 230,4k, a zvolíte-li rychlost 115,2k, komunikační program nastaví dělitele na hodnotu 1. Toto nastavení ovšem povede k reálné rychlosti 230,4k. Ať nastavíte jakoukoliv rychlost, reálná rychlost bude vždy dvojnásobná. Pokud byste měli port s maximální rychlostí 460,8k, byla by rychlost čtyřnásobná.

Nastavení dělitele

Tento problém můžete vyřešit programem *setserial*, kterým nastavíte hodnotu parametru *baud_base* na skutečnou hodnotu maximální rychlosti vašeho portu. Pak budou komunikační programy a další aplikace používat dělitele vycházející z této nastavené hodnoty, a ne z implicitního maxima 115,2k.

Některé sériové porty používají k nastavení vyšších rychlostí vysoké hodnoty dělitele. V těchto případech nemůžete *setserial* použít, protože pokud se pak pokusíte nastavit dělitele na řekněme 32 770, bude se *setserial* domnívat, že port pracuje na velmi nízké rychlosti a vypne podporu bufferů.

Tabulka rychlostí

Máte-li modem s rychlostí 56k, nejvhodnější je sériový port s čipem UART 16 650, ten ale obsahuje jen velmi málo modemů a portů. Druhým nejlepším je typ 16 550 nastavený na rychlost 230,4k. Většinou se ale používá 16 550 na rychlosti 115,2k. Ve většině případů by to nemělo ovlivnit propustnost modemu, protože kompresní poměr je typicky 2:1 (případně, při stahování komprimovaných souborů samozřejmě 1:1). Následující tabulka uvádí doporučené rychlosti sériového portu v závislosti na rychlosti modemu:

56k (V.92)	115,2k nebo (lépe) 230,4k
56k (V.90)	115,2k nebo (lépe) 230,4k

33,6k (V.34bis)	115,2k
28,8k (V.34)	115,2k
14,4k (V.32bis)	57600
9,6k (V.32)	38400
< 9600 (V.32)	použijte stejnou rychlost jako je rychlost modemu

Výše uvedené doporučené rychlosti počítají s použitím datové komprese podle standardu V.42bis a chybové korekce podle standardu V.42. Pokud nepoužíváte kompresi dat, můžete rychlost portu snížit, samozřejmě ne pod rychlost modemu.

Problémy

Modem je připojen, ale nefunguje

Objevuje se chybové hlášení typu „No modem detected“, „Modem not responding“, případně žádné chybové hlášení. Pokud máte nainstalován interní modem s vestavěným sériovým portem, nebo externí modem a nevíte, na kterém portu je připojen, pak je základem zjistit, zda vůbec funguje sériový port. *Viz Sériový port existuje, ale nelze jej nalézt.* Dále se budeme zabývat tím, jak zjistit, ke kterému portu je modem připojen.

Existuje program *wvdialconf*, který hledá modem na běžně používaných portech. Stačí zadat příkaz *wvdialconf nějaký_název_souboru*. Program vytvoří nový konfigurační soubor, který je vám ale k ničemu, pokud nehodláte používat program *wvdial*. Bohužel, pokud je modem právě v datovém režimu, *wvdialconf* jej nenajde. Viz *Modem nereaguje na AT příkazy.*

Problém může být způsoben také tím, že máte winmodem, který typicky v Linuxu nefunguje. Viz *Softwarové modemy*. Pomocí programu *setserial* můžete detekovat existující sériové porty, nelze jím ale detekovat připojené modemy. K tomu je nejvhodnější program *wvdialconf*.

Další možnost, jak zjistit, zda je na daném portu funkční modem, je spustit program *minicom* (který musíte nejprve správně nastavit). V něm zadejte text *AT* a měli byste dostat odezvu *OK*. Pokud se okamžitě neobjeví *OK*, pak:

- Neobjeví se nic. Viz Modem nereaguje na AT příkazy.
- Odezva se objeví až po dlouhé době. Viz *Hodně pomalý port.*
- Objeví se nějaké nesmyslné znaky. Modem je pravděpodobně spojen s protistranou, nereaguje na AT příkazy a namísto toho zobrazuje přijatá data.

Modem nereaguje na AT příkazy

Zadáte-li (pomocí programu *minicom* nebo podobného) modemu příkaz *AT*, měl by odpovědět textem *OK*. Pokud se tato odpověď neobjeví (a ve většině případů ani neuvidíte vámi napsané *AT*), znamená to, že modem nereaguje – velmi často proto, že to, co píšete, se k němu vůbec nedostane.

Nejběžnější příčina je ta, že na sériovém portu, na němž komunikujete, není modem připojen. V případě interního modemu nemusí dokonce správný port vůbec existovat. To může být způsobeno tím, že vestavěný sériový port PnP modemové karty buď nebyl nakonfigurován vůbec, nebo nebyl nakonfigurován správně. Viz *Sériový port existuje, ale nelze jej nalézt.*

Pokud se příkazy sice na modem dostanou, ale modem na ně nereaguje, může to být způsobeno tím, že modem je v on line datovém režimu a na AT příkazy nereaguje. Může to nastat v případě,

že jste modem používali a pak jste příslušný proces násilně ukončili. V takové situaci nedojde k přepnutí modemu zpět do příkazového režimu. *Minicom* za těchto okolností ohlásí: „You are already online. Hang up first.“ Wvdial hlásí: „modem not responding.“

Situaci můžete vyřešit restartem počítače. Můžete ovšem nejprve zkusit poslat modemu sekvenci znaků +++, která by jej měla přepnout zpět do příkazového režimu. Před i po této sekvenci musí následovat přibližně sekundová pauza. Nebude to fungovat v případě, že s modemem komunikuje nějaký jiný proces, který mezitím bude na modem posílat jiná data.

„Modem is busy“

Význam tohoto hlášení závisí na programu, který jej zobrazil. Může to znamenat, že modem je skutečně používán jiným procesem. Na distribucích SuSE se toto hlášení objevovalo v případě, že byly funkční dva sériové ovladače – jeden jako součást jádra a druhý jako nahraný modul.

V programu *kppp* se toto hlášení objeví, když se nepodaří přečíst nebo nastavit parametry portu. Může to znamenat, že ovladač nezná správnou adresu sériového portu.

Máte-li PCI modem, zkuste pomocí příkazů *lspci*, *cat /proc/pci* anebo *dmesg* zjistit správnou adresu a přerušení sériového portu. Pak zkontrolujte, zda program *setserial* zobrazí stejné hodnoty. Pokud ne, musíte pomocí programu *setserial* sdělit ovladači správné hodnoty.

„You are already online! Hang up first.“ (minicom)

Je aktivní signál CD modemu. Buď modem skutečně přijímá nosnou, nebo je nastaven tak, aby trvale signalizoval přítomnost nosné. V *minicom*u zadejte *at&v* a podívejte se, zda je nastaveno *&C* nebo *&C0*. Pokud ano, bude modem signalizovat přítomnost nosné i v případě, že není připojen, a chyba se bude objevovat trvale. V inicializačním řetězci modemu nastavte *&C1*.

Na 56k modemu nedosahuji rychlost 56k

Abyste dosáhli rychlosti kolem 56k, musí být telefonní linka velice kvalitní. Některé linky jsou v takovém stavu, že maximální dosažitelná rychlost je mnohem nižší – kolem 28,8k nebo i méně. Někdy může podobné problémy způsobovat paralelní přípojka. Pokud je to možné, zkuste z telefonní linky odpojit všechna ostatní zařízení a připojte na ni pouze modem.

Upload/download souborů nefunguje dobře

Nemusí fungovat správně řízení přenosu. Jsou-li problémy s uploadem: Máte-li nastavenu vysokou DTE rychlost (115,2k nebo tak nějak), pak bude bez problémů fungovat směr od modemu k počítači, ale v opačném směru může docházet k potížím, protože telefonní linka je obecně pomalá. Výsledkem bude spousta chyb a opakovaných odesílaní paketů. Odesílání delších souborů pak může trvat hodně dlouho, případně takové soubory nepůjde odeslat vůbec.

Jsou-li problémy s downloadem: Pokud stahujete dlouhé nekomprimované soubory, nemáte na modemu zapnutu kompresi a máte nastavenu nízkou DTE rychlost, pak může nefunkční řízení přenosu ovlivnit i download dat.

Stále se mi objevuje „Id „S3“ respawning too fast: disabled for 5 minutes“

Konkrétně *Id „S3“* je jenom příklad. Zkuste v souboru */etc/inittab* najít řádek začínající textem *S3*, který volá *getty*. Ten způsobuje problémy. Zkontrolujte, že je v pořádku syntaxe, a že zařízení (*ttyS3*) existuje. Pokud modem nenastaví signál CD a *getty* otevře port, bude se toto hlášení obje-

vovat, protože nepřítomnost signálu CD proces *getty* zabije. Ten se vzápětí obnoví a bude znovu zabit atd. K těmto chybám dochází pokud není kabel k modemu správně připojen, nebo pokud není modem správně nastaven. Zkuste AT příkazy E a Q.

Po zavěšení vzdálenou stranou se modem zasekne

Stává se to, pokud se modem neresetuje po ztrátě signálu DTR. Někdy se tato situace pozná podle nesmyslného blikání diod RD a SD. Musíte resetovat modem. U většiny modemů kompatibilních se standardem Hayes to provedete příkazem *&D3*, u modemů USR Courier, SupraFax a některých dalších příkazem *&D2*.

NO DIALTONE

Tato chyba znamená přesně to, co říká. Telefonní linku používá někdo jiný, modem není připojen na telefonní linku, anebo je na lince porucha. Zkuste namísto modemu připojit normální telefon a zkontrolujte, zda je linka živá[3].

NO CARRIER

Znamená to, že se ztratila nosná sinusová vlna od protějšího modemu. Pokud už jste měli navázáno spojení, tak se pravděpodobně rozpadlo. Na lince se mohl objevit šum, nebo jiné problémy. Protistrana mohla z nějakého důvodu zavěsit – neproběhlo správně přihlášení, nespustilo se správně PPP, vypršely časové limity.

Pokud se tato chyba objeví před navázáním spojení, znamená to, že se nedaří detekovat protější modem. Nejběžnější situace je ta, že na protější straně linky není modem, ale například záznamník, který „zvedl“ váš telefon dřív. Další příčinou může být to, že se modemy nedokázaly dohodnout na komunikačním protokolu. Může k tomu dojít v případě, že máte starší V.90 modem, který se nejprve pokouší komunikovat protokoly X2 a K56flex. Oba tyto protokoly jsou dnes již zastaralé a protistrana obvykle zavěsí, protože jim nerozumí a nečeká dostatečně dlouho na to, než váš modem zkusí komunikaci protokolem V.90.

Sériový port existuje, ale nelze jej nalézt

Pokud fyzické zařízení (například modem) vůbec nefunguje, je to obvykle proto, že je vypnuto a nemá přidělenu žádnou adresu (PnP zařízení neaktivováno), nebo je sice zapnuto, ale pracuje na jiné adrese, než si ovladač myslí, a proto s ním nekomunikuje.

Zkontrolujte nastavení BIOSu a hlášení BIOSu při startu počítače. Máte-li modem na PCI sběrnici, zkuste *lspci* nebo *scanpci*. Pokud je na ISA sběrnici, zkuste *pnpdump --dumpregs* a přečtěte si *Plug-and-Play-HOWTO*.

Program *scanport* (Debian) projde všechny porty ISA sběrnice a může najít modem na neznámém portu. Na druhé straně může také počítač zhroutit. Pokud se přes port nic nepřenáší, může být problém ve špatně nastaveném přerušení. Spusťte *setserial -g* a zkontrolujte konflikty adres a přerušení.

Pokud dva porty používají stejnou adresu, automatická detekce najde pouze jeden port. Jsou-li porty PnP, neměla by taková situace nastat, může k ní ale dojít v případě, že jeden z portů není PnP. Pokud dvě zařízení sdílejí stejné adresy, může docházet prakticky k jakýmkoliv chybám.

---

[3] Pozn. překladatele: V našich zeměpisných šířkách a délkách je obvykle problém v tom, že modem očekává na lince něco jiného, než co na ní produkuje operátor. Zkuste vypnout detekci vytáčecího tónu příkazem *ATX3*.

Hodně pomalý port

S největší pravděpodobností jde o problém s přerušením. Typické projevy jsou: Cokoliv, co napíšete, se zobrazí až o několik sekund později. Zobrazí se pouze poslední zapsaný znak, kterým může být neviditelný znak <return>, takže dojde pouze k odřádkování. Mělo by se zobrazit hodně znaků, ale zobrazí se jich pouze 16. Může být hlášena chyba „input overrun".

Podrobnější popis příznaků a řešení najdete v dokumentu *Serial-HOWTO*. Jde-li o PnP zařízení, podívejte se také na *Plug-and-Play-HOWTO*.

Jako jednoduchou kontrolu zkuste nastavit přerušení na *0*. Tím ovladači říkáte, aby nečekal na přerušení od hardwaru, ale aby port periodicky kontroloval. Pokud se problém odstraní, docházelo ke konfliktům nebo ztrátám přerušení a vy musíte vyřešit tento primární problém, protože dotazovací metoda obsluhy portu je náročná na procesor počítače.

Nalezení příčiny konfliktu nemusí být jednoduché, protože Linux se snaží takovým konfliktům zabránit, a pokud se domnívá, že k němu může dojít, zobrazí chybu */dev/ttyS?: Device or resource busy*. Ke konfliktu nicméně může dojít v případě, že má jádro nesprávné informace. Musíte zjistit, co si myslí program *setserial* (který sděluje parametry jádru) a dále musíte zjistit, jak je hardware opravdu nastaven.

Trochu pomalé – čekal bych něco víc

Klasická příčina je nízké nastavení přenosové rychlosti. Může k tomu dojít, pokud se rychlost pokusíte nastavit na vyšší hodnotu, než jakou hardware podporuje.

Dalším důvodem může být, že připojené zařízení (modem, terminál, tiskárna) prostě nefunguje tak rychle, jak byste čekali. Modem o rychlosti 56k této rychlosti dosahuje jen málokdy a pokud na protější straně není modem s digitálním připojením, pak není možná rychlost nad 33,6k.

Ještě další problém může být v tom, že máte staříčký sériový port – UART 8 250, 16 450 nebo první verze 16 550 (anebo si to myslí váš ovladač). Viz *What are UARTs* v dokumentu *Serial-HOWTO*. Zkuste *setserial -g*. Pokud zobrazí cokoliv jiného než 16 550A, může to být příčina.

Při startu počítače se zobrazí nesprávná přerušení sériových portů

Pokud port není PnP, Linux při startu neprovádí detekci přerušení. Když se nahrává ovladač sériových portů, testuje pouze jejich přítomnost. To, co říká o přerušeních, nemá žádnou váhu, protože pouze odhaduje standardní hodnoty. Detekce přerušení totiž není spolehlivá a nemusí skončit dobře. Pokud se ale později spouští *setserial*, sdělí ovladači nová přerušení a zobrazí je – tyto hodnoty by už měly být správné. Jestliže se tedy v pozdější fázi startu systému neobjeví správné hodnoty, není něco v pořádku.

Cannot open /dev/ttyS?: Permission denied

Zkontrolujte přístupová práva k portu příkazem *ls -l /dev/ttyS?*. Potřebujete mít práva čtení i zápisu. Práva můžete změnit příkazem *chmod*.

Operation not supported by device

Toto hlášení znamená, že operace požadovaná programem *setserial*, *stty* a podobně není podporována jádrem. Dříve se to stávalo v případě, že nebyl nahrán ovladač sériového portu. V době PnP to obvykle znamená, že na adrese, na které to ovladač očekává, není žádný sériový port.

Cannot create lockfile

Když nějaký program otevře port, vytvoří v adresáři */var/lock* zámek. Pokud jsou nastavena špatná přístupová práva na tento adresář, nepůjde zámek vytvořit. Ověřte nastavení práv příkazem *ls -ld /var/lock* – typicky jsou nastavena na *rwx* pro všechny. Další podrobnosti viz dokument *Serial-HOWTO*, část *What Are Lock Files*.

Device /dev/ttyS? is locked

Znamená to, že sériový port používá někdo jiný (nebo jiný proces). Existuje více způsobů, jak zjistit, který proces to je. Jedna možnost je zkontrolovat soubor se zámkem (*/var/lock/LCK...*). Měl by obsahovat ID procesu, který port používá. Je-li ID procesu například 100, zadejte příkaz *ps 100* a zjistíte, o jaký proces jde. Pokud jde o nějaký nepotřebný proces, můžete jej zabít příkazem *kill 100*. Jestliže to nezabere, zabijte jej příkazem *kill -9 100*. V tomto případě ale nedojde ke smazání souboru se zámkem a budete jej muset smazat ručně. Stejně tak pokud k tomuto souboru neexistuje žádný proces.

/dev/tty? Device or resource busy

Znamená to, že zařízení, k němuž chcete přistupovat, je (asi) právě používáno, nebo jsou (asi) používány nějaké prostředky (například přerušení), které zařízení potřebuje. Jednodušší varianta nastává v případě, že je používáno přímo požadované zařízení. Častěji jde ale o problém v konfliktu o jiné prostředky. Ještě horší je, že v některých případech není „busy" nic, co by přicházelo v úvahu.

Hlášení „resource busy" například pro port *ttyS2* totiž často říká vlastně jen „nemůžete použít *ttyS2*, protože jeho přerušení používá jiné zařízení". Ovšem toto upozornění na potenciální konflikt přerušení vychází z toho, co si o nastavení portu myslí program *setserial*. Přesnější chybové hlášení by mělo říkat „nemůžete použít *ttyS2*, protože si jádro myslí, že budete používat přerušení, které už používá někdo jiný". Pokud dvě zařízení používají stejné přerušení a pracujete pouze s jedním z nich, je všechno v pořádku, protože zatím nedochází ke konfliktu. Pokud se ale pokusíte otevřít druhé zařízení (aniž zavřete první), objeví se výše zmíněné chybové hlášení. Jádro totiž sleduje, která přerušení jsou aktuálně využívána a vychází z toho, že ke konfliktům nemůže dojít, pokud nejsou zařízení otevřena. Podobná situace vzniká v případě konfliktu vstupně-výstupních adres.

Někdy může být tato chyba dána tím, že máte aktivní dva ovladače sériových portů – jeden přímo v jádře a druhý jako modul. Oba ovladače se pokoušejí pracovat se stejnými zařízeními a jeden z nich tedy zjistí, že zařízení je používáno.

Obecně může mít tato chyba dvě příčiny:

- Skutečně může nastávat konflikt mezi prostředky.
- Setserial je špatně nastaven a pouze hlásí konflikt, který reálně neexistuje.

Musíte zjistit, co se *setserial* domnívá, že používáte za přerušení.

Zkuste problém vyřešit buď restartem počítače, nebo zabitím všech potenciálně konfliktních procesů. Pokud budete restartovat počítač, pozorně sledujte hlášení při jeho startu.

Pokud si myslíte, že správně víte, které přerušení sériový port používá, zkuste se podívat do souboru */proc/interrupts* a zjistit, kdo další dané přerušení používá.

Setserial hlásí Input/output error

Možná jste místo *ttyS* napsali *ttys*. Obecně se tato chyba objeví, pokud se *setserial* pokouší pracovat s jiným zařízením, než se sériovým portem. Dále může být příslušný sériový port zrovna používán jiným procesem, anebo na zadané vstupně-výstupní adrese nemusí žádný sériový port existovat.

Overrun error

Došlo k přeplnění hardwarového bufferu sériového portu. Viz *Higher Serial Throughput* v dokumentu *Serial-HOWTO*.

Port vynechává znaky na příjmu

Port může být používán jiným programem. Zkuste *top* nebo *ps -alxw*. Program *gpm* pro obsluhu myši často otevírá sériové porty.

Ladicí nástroje

Některé užitečné programy, které můžete použít při diagnostice problémů:

- *lsof /dev/ttyS** zobrazí otevřené sériové porty
- *setserial* zobrazuje a nastavuje nízkoúrovňovou konfiguraci portu
- *stty* zobrazuje a nastavuje konfiguraci portu na vyšší úrovni
- *modemstat* nebo *statserial* zobrazí stav různých řídicích signálů modemu (DTR, CTS a podobně)
- *irqtune* zvýší prioritu přerušení sériového portu a zlepší tak výkon
- *hdparm* může doladit parametry pevného disku
- *lspci* zobrazí přerušení, adresy a podobně zařízení na PCI sběrnici
- *pnpdump --dumpregs* zobrazí adresy, přerušení a podobně PnP zařízení na ISA sběrnici
- některé soubory v /proc (například ioports, interrupts a tty/driver/seriál)

Odkazy

Různé

- Manuálové stránky příkazů agetty(8), getty(1), gettydefs(5), init(1), isapnp(8), login(1), mgetty(8), setserial(8).
- Návod k modemu (pokud jej máte).
- Poštovní konference *serial*. Přihlásit se můžete posláním e-mailu s textem *subscribe linux-serial* na adresu *majordomo@vger.kernel.org*.

Dokumenty HOWTO

- Cable-Modem mini-howto
- SuSE ISDN Howto (*není součástí LDP*), viz http://sol.parkland.cc.il.us/sdb/en/html/-isdn.html nebo http://brenner.chemietechnik.uni-dortmund.de/doc/sdb/en/html/-isdn.html.

- Sdílení modemů (kapitola 12). *Sdílení jednoho modemu v síti.*
- PPP-HOWTO. *Nastavení PPP včetně nastavení modemu*
- Serial-HOWTO
- Text-Terminal-HOWTO. *Hovoří i o modemech*

Webové stránky

- Seznam modemů, které fungují/nefungují v Linuxu, *http://www.idir.net/~gromitkc/winmodem.html*
- Domovská stránka sériového ovladače, hovoří i o podpoře PCI modemů, *http://serial.sourceforge.net/*
- AT příkazy Hayes modemů, *http://www-dcg.fnal.gov/Net/HYSTRM20.TXT*
- *http://www.cisco.com/univercd/cc/td/doc/product/access/acs_mod/cis3600/analogfw/analogat.htm*
- *http://www.zoltrix.com/support_html/modem/USEMODEM.HTM*
- *http://modemfaq.home.att.net/*
- *http://www.net-boy.com/*
- Informace o 56k modemech, *http://808hi.com/56k/*
- Odkazy na stránky výrobců modemů, *http://www.56k.com/links/Modem_Manufacturers/*
- Další odkazy na výrobce modemů, *http://modems.rosenet.net/*

Digitální modemy: ISDN, DSL, RAS

Úvod

V tomto dokumentu jsme hovořili pouze o normálních analogových modemech používaných pro připojení k analogovým telefonním linkám. Standardní definice modemu se ale někdy rozšiřuje i o digitální „modemy“. V současné době se rozšiřují možnosti přímého digitálního přístupu k telefonním linkám. Stále je ale zapotřebí zařízení, které konvertuje signál z počítače na signál odpovídající požadavkům telefonní linky. Tato zařízení se rovněž označují jako modemy.

ISDN modemy

Tento „modem“ je ve skutečnosti tzv. Terminal Adapter (TA). Některé typy jsou podporovány jádrem 2.4, podrobnosti najdete v dokumentaci jádra v adresáři *isdn*. Ke konfiguraci můžete použít program *isdn-config*, v distribuci Debian pak *isdnutils*. Další informace obsahuje dokument *ISDN HOWTO* (nejde o součást LDP) od SuSE – http://sol.parkland.cc.il.us/sdb/en/html/isdn.html. Podporu ISDN zajišťuje balík *isdn4linux*.

Digital Subscriber Line (DSL)

DSL využívá stávajícího telefonního vedení od vás k ústředně. Pokud je přípojka kvalitní, může přenášet podstatně vyšší rychlostí, než normální modem. Na straně ústředny je umístěno speciální zařízení, které umožňuje velmi rychlý přenos dat. Na vaší straně je rovněž připojen tzv. „DSL modem“.

Digitální modemy 56k

Nejde o modemy, které byste používali doma u počítače. Tyto „modemy“ používá poskytovatel, k němuž se připojujete. Poskytovatel musí být připojen přímo k digitálnímu systému telefonního operátora, pak může být ve směru od klienta použita rychlost 56k. Poskytoval má s velkou pravděpodobností celou modemovou banku připojenou k „tlusté“ telefonní lince (T1, E1, ISDN PRI apod.), takže je schopen zpracovávat mnoho hovorů současně. Takováto zařízení se označují jako Remote Access Server (RAS).

Sdílení modemů

Originál: http://tldp.org/HOWTO/mini/Linux-Modem-Sharing/

Popisuje nastavení linuxového systému tak, aby mohl sdílet modem s dalšími systémy v síti.

Strana serveru

Předpokládám, že serverem je Linux, který má:

- buď modem připojený k zařízení */dev/ttySx*,
- nebo modem emulovaný přes *isdn4linux* na zařízení */dev/ttyIx*.

Nejjednodušší nastavení, které mě napadá, je pětiřádkový skript, který implementuje modemového démona v */usr/sbin/modemd*:

```
#!/usr/bin/perl
select((select(STDOUT), $| = 1)[$[]);
select((select(STDIN), $| = 1)[$[]);
exec 'cu -s 115200 -l /dev/ttyS1';
die '$0: Cant exec cu: $!\n';
```

Modemový démon je spouštěn procesem *inetd* jakmile se klient připojí k příslušnému portu. *modemd* jednoduše propojí handle soketu s *STDIN* a *STDOUT* příkazu *cu* a skutečnou práci s modemem nechá na programu *cu*.

Existenci modemového démona je nutné procesu *inetd* oznámit úpravou jeho konfiguračního souboru, typicky */etc/inetd.conf*:

```
#
# modem daemon
#
modem stream tcp nowait root /usr/sbin/tcpd /usr/sbin/modemd /dev/ttyS1
```

Aby to fungovalo, musíte do */etc/services* přidat následující položku:

```
modem           2006/tcp        modemd
```

Tím se konkrétnímu portu (v našem případě 2 006) přidělí symbolické jméno. Jako číslo portu můžete použít cokoliv, co není přiděleno nějaké používané službě. Po provedení potřebných změn musíte procesu *inetd* poslat signál nařizující, aby si znovu načetl konfiguraci:

```
bash# ps | grep inetd
194  ?  S     0:00 /usr/sbin/inetd
bash# kill -HUP 194
```

Nyní je server připraven na přijímání požadavků od klientů. Jeho správnou funkci můžete ověřit tímto postupem:

```
bash$ telnet localhost modem

Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

Nyní jste připojeni k modemu. Konfiguraci můžete ověřit zadáním nějakých AT příkazů:

```
atz
atz
OK

ati1
ati1
Linux ISDN
OK

^]
telnet>quit
bash$
```

Abyste jako server nemuseli používat perlový skript, existuje rovněž program *Masqdialer*, který najdete na adrese http://w3.cpwright.com/mserver/.

Pomocí tohoto programu můžete exportovat libovolný počet modemů připojených k serveru kterémukoliv systému, jenž se k němu může pomocí TCP/IP připojit na zadaném portu.

Instalace programu Masqdialer

Před překladem zkontrolujte soubor *config.h* a podle svých představ upravte:

- cestu ke konfiguračnímu souboru,
- cestu k souboru se zámkem.

Spusťte *make all*.

Zkopírujte binární soubory (*mserver* a *tcpconn*) do vhodného adresáře, např. */usr/local/sbin*. Souboru *mserver.conf* zkopírujte do adresáře, který jste v *config.h* uvedli jako konfigurační.

Program je možné spustit z inicializačních skriptů prostým příkazem */usr/local/sbin/mserver*.

Konfigurace programu Masqdialer

Řádky v souboru *mserver.com* vypadají takto:

```
5800 /dev/ttyS1      115200,8,N,1    *.foo.org,192.168.2.1
```

Tento řádek říká, že modem připojený k portu */dev/ttyS1* může být použit na portu 5 800 jakýmkoliv systémem z domény *foo.org* a z počítače 192.168.2.1. Ostatní počítače budou odmítnuty. Pozor! Pokud neuvedete seznam povolených počítačů, bude povolen přístup *VŠEM*. Nastavení sériového portu je fixní a klient je nemůže změnit. Na jednom portu můžete exportovat více modemů. Program zamyká modemy běžným mechanismem zámkových souborů pouze v době, kdy jsou skutečně používány, takže je mohou využívat i jiné programy.

Strana klienta

V současné verzi dokumentu popisujeme pouze nastavení klienta pro Windows. Na klientském počítači potřebujeme přesměrovat COM port na TCP/IP. Nejlepším programem pro tento účel je zřejmě *DialOut/IP* společnosti *Tactical Software* pro Windows 3.1 a Windows 95.

Program *DialOut/IP* zpřístupní sdílený modem na nově vytvořeném virtuálním COM portu. Ten pak můžete v programech pro Windows použít stejně jako by byl modem připojen přímo. Většina programů (včetně programu pro telefonické připojení ve Windows) toto uspořádání bez potíží akceptuje a pracuje stejně, jako by šlo o skutečný COM port a skutečný modem. Výjimkou jsou různé faxové aplikace a další programy, které potřebují přímý přístup k UART čipu. Program *DialOut/IP* umožňuje zpracovávat protokol *telnet*, ale toto nastavení je vhodné pro některé modemové banky a nehodí se pro server nastavený výše uvedeným postupem. Navzdory svému názvu lze *DialOut/IP* použít i pro aplikace, které čekají na příchozí hovory.

Na adrese http://www.tactical-sw.com/ si můžete stáhnout plně funkční verzi s životností 1–2 týdny. Instalaci a konfiguraci provádí přiložený instalační program, podrobnosti jsou popsány v souboru *README.TXT*. Při spuštění programu *DialOut/IP* pak už jen zadáváte IP adresu a port, na němž se připojujete ke sdílenému modemu.

Program *DialOut/IP* je komerční produkt licencovaný podle počtu sdílených modemů. Licence říká, že jej můžete nainstalovat na libovolný počet počítačů, které modemy sdílejí.

Bezpečnostní úvahy

Pokud v celé síti používáte pouze jediný modem, nemusíte se s bezpečností asi nijak zatěžovat. Pokud jsou ale některé počítače v síti připojeny k Internetu i jinak než prostřednictvím sdíleného modemu, musíte nějakým mechanismem zajistit, aby se kdokoliv nemohl přes Internet připojit k vašemu modemovému serveru a vytáčet mezinárodní a jiné podobně drahé hovory.

Doporučuji nainstalovat a nakonfigurovat tcp_wrapper chránící modemový server před neautorizovaným přístupem.

Příklady

Výše popsanou konfiguraci používám ke spouštění programu *Quicken* na Windows 95 a k přístupu k internetovému bankovnictví přes modem připojený k linuxovému počítači. V mém případě nesdílím „opravdový“ modem, ale emulovaný modem na kartě ISDN. *Quicken* prostě jenom vidí COM port a netuší, že k němu připojené zařízení je fyzicky někde úplně jinde a že ani nejde o standardní analogový modem, ale o ISDN zařízení, které shodou okolností rozumí AT příkazům.

KAPITOLA 13

Přechod z DOSu/Windows na Linux

Originál: http://tldp.org/HOWTO/DOS-Win-to-Linux-HOWTO.html

Úvod

Je pro vás Linux vhodný?

Začněme politicky korektně. Přestože v celém dokumentu používám termín „Linux“, mám na mysli „GNU/Linux“. Vysvětlení najdete na adrese http://www.gnu.org/gnu/linux-and-gnu.html.

Chcete opustit svět DOSu/Windows a přejít k Linuxu? Dobrý nápad – Linux je technicky dokonalejší než DOS, Windows 9x a dokonce i Windows NT. Ovšem pozor – nemusí být pro vás vhodný. Mezi DOSem/Windows a Linuxem jsou zásadní rozdíly:

- Ve Windows běží Microsoft Office a spousta her; jdou poměrně snadno nainstalovat a nastavit, jsou vyhlášené nestabilitou, mají slabý výkon a často havarují.
- V Linuxu běží StarOffice, spousta technických programů a málo her; instalace a nastavení může být obtížnější, je mimořádně stabilní, má vysoký výkon a k haváriím dochází jen zcela výjimečně.

Záleží jen na vás, abyste se rozhodli, co potřebujete. Linux vám sice přináší spoustu možností, na druhé straně bude chtít trochu času, než se s ním naučíte zacházet. Pokud tedy používáte zejména komerční programy, nebo pokud se vám nechce učit nové příkazy a poznatky, dělejte něco jiného. Spousta začátečníků se vzdala právě kvůli počátečním potížím.

Stále se pracuje na tom, aby byl Linux použitelný co nejsnáze, nicméně *nečekejte, že jej budete dobře ovládat bez pročtení spousty dokumentace a bez několika měsíců praxe*. Nečekejte okamžité výsledky. Na druhé straně, pokud jste ten správný typ uživatele, jsem přesvědčen, že Linux pro vás bude stoprocentní počítačovou nirvánou. A mimochodem – Linux a DOS/Windows mohou bez potíží fungovat na jednom počítači.

K dalšímu čtení předpokládám, že:

- znáte základy DOSu a jeho příkazy

- máte na počítači správně nainstalován Linux a případně systém X Window (zkráceně X11),
- jako příkazový interpret (ekvivalent `COMMAND.COM`u) používáte `bash`.

Není-li řečeno jinak, všechny další popisy se vztahují ke starému dobrému DOSu. Tu a tam se zmíníme o Windows, nezapomínejte ale, že Windows a Linux jsou dvě úplně jiné věci, zatímco DOS je takový hodně chudý příbuzný UNIXu.

Tento text také nepředstavuje kompletní slabikář nebo konfigurační příručku!

Dobře, co dál?

Máte na počítači nainstalován Linux a potřebné programy. Máte vytvořen uživatelský účet (pokud ne, zadejte *hned* příkaz `adduser` *`vaše_jméno`*) a máte spuštěný Linux. Právě jste zadali jméno a heslo a teď se díváte na obrazovku a myslíte si „Co teď"?

Nepropadejte panice. Prakticky hned můžete dělat to samé co v DOSu/Windows a ještě daleko více. Když jste provozovali DOS/Windows, určitě jste dělali věci jako:

- spouštění programů, vytváření, kopírování, prohlížení, mazání, tisk a přejmenování souborů;
- vytváření, procházení a rušení adresářů;
- formátování disket a kopírování dat na diskety;
- úpravy systému;
- práce s Internetem;
- vytváření souborů BAT a programování v oblíbeném programovacím jazyce;
- a zbylé 1 % činností.

Jistě rádi uslyšíte, že všechny tyto věci jde v Linuxu dělat velmi podobně jako v DOSu. Běžný uživatel používá v DOSu jen velmi malou část z více než stovky příkazů – a to samé platí i v Linuxu.

Základní poznatky

Nejlepší způsob, jak se něco naučit, je si to vyzkoušet. Vřele vám doporučujeme si s Linuxem hrát a experimentovat – pokud nejste přihlášeni jako „root", nemůžete tím systému ublížit. Několik základních informací:

- Nejprve jak Linux korektně ukončit. Pokud pracujete v textovém režimu, stiskněte Ctrl+Alt+Del, počkejte na restart počítače a vypněte jej. Pokud pracujete v systému X Window, stiskněte nejprve Ctr+Alt+Backspace a teprve pak Ctrl+Alt+Del. *Nikdy* počítač přímo nevypínejte, mohli byste tím poškodit souborový systém.
- Na rozdíl od DOSu a Windows obsahuje Linux vestavěné bezpečnostní mechanismy. Soubory a adresáře mají svá přístupová práva, výsledkem pak je, že běžný uživatel nemůže k některým souborům přistupovat (viz část Práva a vlastníci). V DOSu a Windows naopak můžete kdykoliv smazat celý obsah disku.
- V Linuxu existuje speciální uživatel nazvaný „root" (superuživatel), který funguje jako správce systému a má plnou moc nad počítačem. Pokud pracujete na svém vlastním počítači, určitě budete pracovat i jako superuživatel. Práce jako superuživatel je *nebezpečná*, každá chyba může vést k vážnému poškození nebo zničení systému stejně jako v DOSu/Windows. Pokud to není absolutně nutné, tak jako superuživatel nepracujte.

- Složitost Linuxu je dána hlavně jeho vysokou konfigurovatelností – prakticky každou funkci a každou aplikaci je možné nastavit pomocí jednoho nebo více konfiguračních souborů. Složitost je cena, která se platí za tuto pružnost.
- Možnost přesměrování existuje okrajově i v DOSu, v Linuxu jde o velmi důležitou a poměrně mocnou funkci. Je možné za sebou řetězit jednoduché příkazy a provádět tak komplikované operace. Vřele vám doporučujeme naučit se, jak je používat.

Kde získat další informace

Možností, kde získat nápovědu, je při práci s Linuxem celá řada. Nejdůležitější jsou:

- *dokumentace* – a to myslíme vážně. Přestože vám tento text může posloužit jako úvod do práce s Linuxem, existuje několik knih, které byste si měli přečíst. Minimálně to jsou „Linux Installation and Getting Started" Matta Welshe (http://www.linuxdoc.org/LDP/gs/gs.html) a Linux FAQ (http://www.linuxdoc.org/FAQ/Linux-FAQ/index.html). Dokud si některou z nich nepřečtete, berte to jako velké minus!
- Dokumentaci k balíkům, které máte na počítači nainstalovány, najdete obvykle v adresáři */usr/doc*.
- Nápovědu k „interním příkazům" příkazového interpretu získáte pomocí příkazů `help`, nebo lépe `man bash` nebo `info bash`.
- Nápovědu ke konkrétnímu příkazu získáte příkazem `man` *příkaz* (čímž vyvoláte manuálovou stránku daného příkazu). Případně můžete vyzkoušet `info` *příkaz* – *info* je hypertextový dokumentační systém, možná pro začátek trochu málo intuitivní. Konečně můžete vyzkoušet whatis *příkaz*. Ve všech případech opustíte nápovědu klávesou „q".
- Dalším místem je samozřejmě Internet, například usenetová adresa news:comp.os.linux.-setup. Rozhodně nežádejte pomoc od autora tohoto dokumentu.

Konvence

V dalším textu používáme v příkladech obvykle následující notaci: `<...>` znamená povinný parametr, zatímco `[...]` je nepovinný parametr. Například:

```
$ tar -tf <soubor.tar> [> vystupni_soubor]
```

`soubor.tar` musí být uveden, přesměrování do `vystupni_soubor` je nepovinné.

„RMP" znamená „please Read the Man Pages for further information" – podrobnosti naleznete v manuálu. Čtení dokumentace je zkrátka nesmírně důležité.

Pokud je příkaz uveden výzvou #, znamená to, že jej může provést pouze superuživatel.

Pro netrpělivé

Chcete už začít? Tak se podívejte na následující tabulku:

DOS	Linux	Poznámka
`ATTRIB (+-) attr soubor`	`chmod <mód> soubor`	**úplně jiné**
`BACKUP`	`tar -Mcvf zařízení /adresář`	**dtto**
`CD adresář\`	`cd adresář/`	**téměř stejná syntaxe**

DOS	Linux	Poznámka
`COPY soubor1 soubor2`	`cp soubor1 soubor2`	dtto
`DEL soubor`	`rm soubor`	pozor – neexistuje „undelete“
`DELTREE adresář`	`rm -R adresář/`	dtto
`DIR`	`ls`	poněkud jiná syntaxe
`DIR soubor /S`	`find . -name soubor`	něco úplně jiného
`EDIT soubor`	`vi soubor jstar soubor`	to se vám asi líbit nebude skoro jako dosový edit
`EDLIN soubor`	`ed soubor`	zapomeňte na to
`FORMAT`	`fdformat, mount, umount`	něco docela jiného
`HELP příkaz`	`man příkaz info příkaz`	stejná filozofie
`MD adresář`	`mkdir adresář/`	téměř stejná syntaxe
`MORE < soubor`	`less soubor`	mnohem lepší
`MOVE soubor1 soubor2`	`mv soubor1 soubor2`	dtto
`NULL`	`/dev/null`	dtto
`PRINT soubor`	`lpr soubor`	dtto
`PRN`	`/dev/lp0 /dev/lp1`	dtto
`RD adresář`	`rmdir adresář/`	téměř stejná syntaxe
`REN soubor1 soubor2`	`mv soubor1 soubor2`	nefunguje pro více souborů
`RESTORE`	`tar -Mxpvf zařízení`	jiná syntaxe
`TYPE soubor`	`less soubor`	mnohem lepší
`WIN`	`startx`	nebe a dudy!

Pokud potřebujete více než jen tabulku příkazů, čtěte dál.

Poznáváme bash

Dobrá zpráva: v Linuxu píšete na příkazové řádce mnohem méně, protože shell `bash` umí psát za vás a nabízí šikovné editační funkce. Začněme třeba tím, že stiskem šipky nahoru vyvoláte poslední zadaný příkaz – to ale není zdaleka všechno. Klávesou Tab dokončujete názvy souborů a adresářů, takže když zadáte:

```
$ ls /uTABloTABbTAB
```

je to jako byste zadali

```
$ ls /usr/local/bin
```

V případě nejednoznačnosti, například

```
$ ls /uTABloTABiTAB
```

`se bash` zastaví, protože neví, zda myslíte */usr/local/info* nebo */usr/local/include*. Přidejte ještě nějaké znaky a stiskněte Tab znovu.

Často používané klávesové kombinace jsou Esc+Backspace, které vymaže slovo nalevo; Esc+D vymaže slovo napravo; Esc+F posune kurzor o jedno slovo doprava; Esc+B o jedno doleva; Ctrl+A přejde na začátek řádku, Ctrl+E na konec. Klávesa Alt funguje stejně jako Esc.

To prozatím stačí. Jakmile si na tyto zkratky zvyknete, zjistíte, že prompt DOSu je hrozně nešikovný...

Soubory a programy

Soubory – úvodní poznámky

Linux používá strukturu adresářů a souborů, která je velmi podobná DOSu/Windows. Soubory mají názvy, které se řídí nějakými pravidly, ukládají se do adresářů, některé z nich jde spustit. Většina spustitelných umožňuje zadat nějaké parametry. Můžete používat i zástupné znaky, přesměrování a rouru. Snad jen některé drobné rozdíly:

- v DOSu používají názvy souborů notaci 8.3, například DOSTMALO.TXT. V Linuxu je to lepší. Pokud použijete souborový systém ext2 nebo umsdos, můžete používat delší názvy souborů (až do 255 znaků) a dokonce s více tečkami, například malá písmenka – ono totiž...
- Linux rozlišuje velká a malá písmenka v názvech souborů a příkazů. Takže SOUBOR.tar.gz a soubor.tar.gz jsou dvě různé věci. ls je příkaz, LS je chyba.
- Uživatele Windows upozorňujeme na používání dlouhých názvů souborů v Linuxu. Pokud název obsahuje mezeru (což se sice nedoporučuje, ale je to možné[1]), je nutné při každé manipulaci s takovýmto souborem uzavřít jeho název do uvozovek. Například:

```
$ # nasledujici prikaz vytvori adresar "Zaloha starych souboru"
$ mkdir "Zaloha starych souboru"
$ ls
Zaloha starych souboru    bin     tmp
```

  Kromě toho by se neměly používat některé znaky, například `!*$&#`.

- Neexistují žádné povinné přípony jako .COM a .EXE pro programy nebo .BAT pro dávkové soubory. Spuštěné soubory poznáte podle toho, že při výpisu příkazem ls -F mají na konci názvu uvedenu hvězdičku. Například:

```
$ ls -F
I_am_a_dir/   cindy.jpg    cjpg*   letter_to_Joe    my_1st_script*  old~
```

  Soubory `cjpg` a `my_1st_script` jsou spustitelné, tedy „programy“. V DOSu se zálohy souborů označují příponou .BAK, v Linuxu končí tildou „~“. Pokud název souboru začíná tečkou, je soubor považován za skrytý. Například soubor *.Skryty.soubor* se neobjeví ve výpisu příkazem `ls .`

- Přepínače dosových programů se zadávají jako /přepínač, Linux používá -přepínač nebo --přepínač[2]. Například dir /S je ekvivalentní příkazu ls -R. Můžete si všimnout, že tuto notaci používá i řada dosových programů, například PKZIP nebo ARJ.

Teď můžete přejít k části *Překlad dosových příkazů do Linuxu*, ale na vašem místě bych pokračoval ve čtení.

1 Pozn. překladatele: Rovněž tak je možné, ale nedoporučuje se, používat v názvech souborů české znaky. Vede to k řadě problémů při přenosu souborů na jiné operační systémy. K tomuto tématu se ještě vrátíme v části věnované práci se zařízeními.

2 Pozn. překladatele: Obvykle platí, že jednou pomlčkou se uvozují „krátké“ přepínače, dvěma „dlouhé“ přepínače. Například -h a --help bude mít obvykle stejný efekt.

Symbolické odkazy

UNIX používá jeden typ souborů, které v DOSu neexistují – symbolické odkazy. Představují vlastně ukazatel na soubor nebo adresář a dá se s nimi pracovat místo souboru nebo adresáře, na nějž ukazují; je to podobné zástupcům ve Windows. Příkladem symbolického odkazu jsou `/usr/X11`, ukazující na `/usr/X11R6`; nebo `/dev/modem`, ukazující na `/dev/ttyS0` nebo `/dev/ttyS1`.

Symbolický odkaz vytvoříte příkazem

```
$ ln -s <soubor_nebo_adresář> <název_odkazu>
```

Například:

```
$ ln -s /usr/doc/g77/DOC g77manual.txt
```

Teď můžete namísto s `/usr/doc/g77/DOC` pracovat s `g77manual.txt`. Odkazy se ve výpisu souborů ukazují takto:

```
$ ls -F
g77manual.txt@
$ ls -l
(něco něco...)           g77manual.txt -> /usr/doc/g77/DOC
```

Práva a vlastníci

Soubory a adresáře v DOSu mají následující příznaky: A (archivní), H (skrytý), R (jen ke čtení), a S (systémový). V Linuxu mají význam pouze příznaky H a R, skryté soubory začínají tečkou, o souborech jen ke čtení se dozvíme dále.

V Linuxu má každý soubor „práva“ a vlastníka, který je dále členem nějaké „skupiny“. Podívejme se na tento příklad:

```
$ ls -l /bin/ls
-rwxr-xr-x  1  root  bin  27281 Aug 15 1995 /bin/ls*
```

První položka udává práva souboru `/bin/ls`, který patří uživateli root, skupině bin. Přeskočíme ostatní údaje a zapamatujeme si, že `-rwxr-xr-x` zleva doprava znamená:

`-` je typ souboru (`-` = normální soubor, `d` = adresář, `l` = odkaz atd.); `rwx` jsou práva vlastníka souboru (čtení, zápis, spuštění[3]); `r-x` jsou práva skupiny (čtení a spuštění). (O skupinách nebudeme hovořit, jako začínající uživatel se bez nich obejdete.) Konečně `r-x` jsou práva všech ostatních uživatelů (čtení a spuštění).

I adresář `/bin` má práva, více se o nich dozvíte v části *Práva adresářů*. Práva jsou důvod, proč nemůžete jako běžný uživatel soubor /bin/ls smazat – nemáte k tomu práva. Práva souborů můžete měnit takto:

```
$ chmod <kdoXpráva> <soubor>
```

kde *kdo* je buď u (uživatel, tedy vlastník), g (skupina) nebo o (ostatní). X je + nebo - a práva jsou r, w nebo x. Typické příklady použití příkazu chmod vypadají takto:

```
$ chmod +x soubor
```

nastaví soubor jako spustitelný.

```
$ chmod go-rw soubor
```

odstraní práva čtení a zápisu pro všechny kromě vlastníka.

3 Pozn. překladatele: Písmenka rwx samozřejmě pocházejí z anglických slov read, write a eXecute.

```
$ chmod ugo+rwx soubor
```

všichni budou mít práva čtení, spuštění i zápisu.

```
# chmod +s soubor
```

vytvoří soubor s příznakem „setuid“ nebo „suid“ – tedy soubor, který může kdokoliv spustit s právy jeho vlastníka. Typicky potkáte soubory „suid root“, to je většina důležitých systémových souborů, například X server.

Rychlejší způsob práce s právy je jejich číselné vyjádření: `rwxr-xr-x` lze zapsat jako 755 (každé písmeno odpovídá jednomu bitu, `---` je 0, `--x` je 1, `-w-` je 2, `-wx` jsou 3...). Vypadá to složitě, ale s trochou praxe to snadno pochopíte. Superuživatel může změnit práva jakéhokoliv souboru. RMP.

Soubory – překlad příkazů

Vlevo je uveden příkaz DOSu, vpravo jeho linuxový ekvivalent.

ATTRIB	chmod
COPY	cp
DEL	rm
MOVE	mv
REN	mv
TYPE	more, less, cat

Operátory přesměrování a zřetězení: `< > >> |`

Zástupné znaky: `* ?`

```
nul: /dev/null
prn, lpt1: /dev/lp0 nebo /dev/lp1; lpr
```

Příklady

DOS	Linux
`C:\GUIDO>ATTRIB +R FILE.TXT`	`$ chmod 400 file.txt`
`C:\GUIDO>COPY JOE.TXT JOE.DOC`	`$ cp joe.txt joe.doc`
`C:\GUIDO>COPY *.* TOTAL`	`$ cat * > total`
`C:\GUIDO>COPY FRACTALS.DOC PRN`	`$ lpr fractals.doc`
`C:\GUIDO>DEL TEMP`	`$ rm temp`
`C:\GUIDO>DEL *.BAK`	`$ rm *~`
`C:\GUIDO>MOVE PAPER.TXT TMP\`	`$ mv paper.txt tmp/`
`C:\GUIDO>REN PAPER.TXT PAPER.ASC`	`$ mv paper.txt paper.asc`
`C:\GUIDO>PRINT LETTER.TXT`	`$ lpr letter.txt`
`C:\GUIDO>TYPE LETTER.TXT`	`$ more letter.txt`
`C:\GUIDO>TYPE LETTER.TXT`	`$ less letter.txt`
`C:\GUIDO>TYPE LETTER.TXT > NULL`	`$ cat letter.txt > /dev/null`
není	`$ more *.txt *.asc`
není	`$ cat section*.txt | less`

Poznámky:

- * je v Linuxu chytřejší, reprezentuje všechny soubory kromě skrytých. Skryté soubory se vyberou jako .* (čímž se ale vybere také aktuální adresář „." a rodičovský adresář „.."!). Výraz *.* označuje pouze soubory, jejichž název obsahuje tečku, nebo které hvězdičkou končí; `p*r` znamená „peter" i „piper"; `*c*` znamená „picked" i „peck".
- Při použití příkazu more se mezerou posouváte v souboru, q ukončí prohlížení. Příkaz less je pohodlnější a umožňuje používat kurzorové šipky.
- V Linuxu neexistuje nic jako UNDELETE, takže před mazáním se vždy dvakrát zamyslete!
- Kromě operátorů < > a >> v DOSu používá Linux ještě 2> k přesměrování chybových hlášení (standardního chybového výstupu, stderr). Navíc 2>&1 přesměruje stderr na stdout (chybový výstup na standardní výstup), 1>&2 přesměruje stdout na stderr.
- Linux používá ještě další zástupné znaky: []. Použití: [abc]* označuje všechny soubory začínající a, b nebo c; *[I-N1-3] označuje soubory končící na I, J, K, L, M, N, 1, 2 nebo 3.
- Příkaz Ipr <soubor> vytiskne soubor na pozadí. Stav tiskové fronty zjistíte příkazem lpq, úlohu z fronty odstraníte příkazem lprm.
- Přejmenování souborů se v Linuxu chová jinak než v DOSu. Příkaz mv *.xxx *.yyy nebude fungovat. Příkaz s podobným chování jako REN najdete v ftp://metalab.unc.edu/pub/Linux/utils/file.
- Pokud chcete být upozorněni při přepisování souborů, použijte příkazy cp -i a mv -i.

Spouštění programů: multitasking a uživatelské relace

Chcete-li spustit nějaký program, stejně jako v DOSu zadejte jeho název. Pokud je adresář (viz *Práce s adresáři*), v němž se program nachází, uveden v proměnné PATH (viz *Inicializační soubory systému*), program se spustí. Výjimka: Na rozdíl od DOSu se v Linuxu nespustí program umístěný přímo v aktuálním adresáři, pokud aktuální adresář není v cestě explicitně uveden. Řešení: chcete-li spustit program prog v aktuálním adresáři, zadejte ./prog.

Typický příkazový řádek vypadá takto:

```
$ command [-s1 [-s2] ... [-sn]] [par1 [par2] ... [parn]] [< input] [> output]
```

kde `-s1`, ..., `-sn` jsou přepínače programu, `par1`, ..., `parn` jsou parametry programu. Více příkazů na jednom řádku můžete spustit takto:

```
$ command1 ; command2 ; ... ; commandn
```

A to je všechno o spouštění programů – nicméně můžeme zajít dál. Jedním z hlavních důvodů pro použití Linuxu je to, že jde o víceúlohový operační systém – v jednom okamžiku může být spuštěno více programů (dále budeme říkat *procesů*). Můžete spustit nějaký proces na pozadí a pokračovat dál v práci. Navíc Linux umožňuje mít současně spuštěno více uživatelských relací – je to, jako kdybyste měli současně k dispozici více počítačů!

- Pro přepínání mezi virtuálními konzolami použijte klávesy Alt+F1 až Alt+F6.[4]
- Pokud chcete na stávající konzole spustit novou uživatelskou relaci, aniž byste ukončili právě běžící, zadejte su – <uživatelské_jméno>. Příklad: su – root. Je to užitečné, pokud chcete udělat něco, co může provést jenom superuživatel.

4 Pozn. překladatele: Pokud standardně pracujete v grafickém prostředí, přepnete se na virtuální textové konzoly klávesami Ctrl+Alt+F1 až F6. V rámci textových konzol pak funguje přepínání pomocí Alt+F1 až Alt+6. Stiskem Alt+F7 se vrátíte zpátky na grafickou konzolu.

- Relaci ukončíte příkazem exit. Pokud existují nějaké zastavené úlohy (viz dále), objeví se upozornění.
- Chcete-li spustit proces na pozadí, doplňte na konec příkazového řádků ampersand „&“:

```
$ progname [-přepínače] [parametry] [< input] [> output] &
[1] 123
```

  shell identifikuje spuštěnou úlohu jejím číslem (v tomto případě [1]) a identifikačním číslem procesu (PID, v tomto případě 123).
- Chcete-li vidět, které procesy běží, zadejte ps ax. Zobrazí se všechny běžící procesy.
- Zabít (ukončit) běžící proces lze příkazem kill <PID>. Zabít proces můžete potřebovat v situaci, kdy jej nejste schopni ukončit korektně. Pokud nejste superuživatel, nemůžete zabít procesy jiných uživatelů. Občas se stane, že proces jde zabít pouze příkazem kill -SIGKILL <PID>. Kromě toho máte možnost proces zastavit nebo dočasně pozastavit, poslat proces na pozadí a přenést proces z pozadí na popředí. V tomto kontextu se procesy označují jako úlohy (jobs).
- Počet běžících úloh zjistíte příkazem jobs. Úlohy jsou identifikovány svými čísly, nikoliv svými PID.
- Proces běžící na popředí můžete zastavit stiskem Ctrl+C (nefunguje to vždy).
- Proces běžící na popředí můžete pozastavit stiskem Ctrl+Z (opět nefunguje vždy).
- Pozastavený proces odešlete na pozadí příkazem bg <%job> (z procesu se stane úloha).
- Přenést úlohu na popředí můžete příkazem fg <%job>. Pokud chcete přenést na popředí úlohu, kterou jste poslali na pozadí jako poslední, stačí zadat pouze příkaz fg.
- Úlohu můžete zabít příkazem kill <%job>, kde <job> je 1, 2, 3...

Tyto příkazy jsou užitečné při formátování disků, vytváření velkých archivů, rozbalování archivů a podobně – to všechno současně a stále můžete normálně pracovat. Zkuste si to ve Windows, čistě abyste viděli rozdíl ve výkonu (samozřejmě pokud Windows raději nespadnou).

Spouštění programů na vzdáleném počítači

Chcete-li spustit nějaký program na vzdáleném počítači, pojmenovaném `remote.machine.edu`, použijte příkaz.

```
$ telnet remote.machine.edu
```

Po přihlášení můžete spouštět libovolné programy. Není snad nutné zdůrazňovat, že k tomu potřebujete mít na vzdáleném počítači uživatelský účet.

Pokud používáte systém X Window, můžete na vzdáleném počítači spouštět i grafické aplikace, přičemž jejich výstup vidíte na své obrazovce. Předpokládejme, že `remote.machine.edu` je vzdálený počítač se systémem X Window, a `local.linux.box` je váš místní počítač. Chcete-li z místního počítače spustit grafickou aplikaci na vzdáleném počítači, postupujte takto:

- spusťte systém X window a spusťte `xterm` nebo nějaký jiný emulátor terminálu, a zadejte:

```
$ xhost +remote.machine.edu
$ telnet remote.machine.edu
```

- po přihlášení zadejte:

```
remote:$ DISPLAY=local.linux.box:0.0
remote:$ progname &
```

(Namísto `DISPLAY...`, můžete zadat `setenv DISPLAY local.linux.box:0.0`. Závisí to na nastavení prostředí vzdáleného počítače.)

A to je celé! Právě jste na vzdáleném počítači spustili program `progname` a jeho výstup vidíte na svém počítači. Nezkoušejte to ale přes modemovou linku, odezva by byla příliš pomalá. Navíc jde o poměrně nepěkný a málo bezpečný způsob. Bezpečnější metodu popisuje dokument „Remote X Apps mini-HOWTO“ na adrese http://www.linuxdoc.org/HOWTO/mini/Remote-X-Apps.html.

Práce s adresáři

Úvodní poznámky

Už jsme si popsali rozdíly mezi soubory v DOSu/Windows a v Linuxu. Co se týče adresářů, v DOSu/Windows se kořenový adresář označuje jako \, v Linuxu jako /. Vnořené adresáře se analogicky oddělují v DOSu/ Windows znakem \, v Linuxu znakem /. Příklady souborových cest:

```
DOS:    C:\PAPERS\GEOLOGY\MID_EOC.TEX
Linux:  /home/guido/papers/geology/middle_eocene.tex
```

Známým způsobem se rodičovský adresář označuje znaky.., aktuální adresář znakem .. Nezapomínejte, že systém vám nedovolí provádět příkazy cd, rd nebo md kde se vám zachce. Každý uživatel má pro svá data vyhrazen svůj domovský adresář, například na mém počítači mám domovský adresář `/home/guido`.

Práva adresářů

Adresáře, stejně jako soubory, mají přístupová práva. To,co jsme popisovali v části *Práva a vlastníci* (vlastník, skupina, ostatní a podobně), platí i pro adresáře. Práva rx pro adresář znamenají, že do něj můžete vstoupit, právo w znamená, že v něm můžete mazat soubory (pokud máte příslušná práva pro daný soubor), případně můžete smazat celý adresář.

Pokud budu například chtít, aby ostatní uživatelé nemohli prohlížet adresář `/home/guido/text`, zadám:

```
$ chmod o-rwx /home/guido/text
```

Adresáře – překlad příkazů

DOS	Linux
DIR	ls, find, du
CD	cd, pwd
MD	mkdir
RD	rmdir
DELTREE	rm –rf
MOVE	mv

Příklady:

DOS	Linux
C:\GUIDO>DIR	$ ls
C:\GUIDO>DIR FILE.TXT	$ ls file.txt
C:\GUIDO>DIR *.H *.C	$ ls *.h *.c
C:\GUIDO>DIR/P	$ ls | more
C:\GUIDO>DIR/A	$ ls -l
C:\GUIDO>DIR *.TMP /S	$ find / -name "*.tmp"
C:\GUIDO>CD není k dispozici (viz poznámky) dtto dtto	$ pwd $ cd $ cd ~ $ cd ~/temp
C:\GUIDO>CD \OTHER	$ cd /other
C:\GUIDO>CD ..\TEMP\TRASH	$ cd ../temp/trash
C:\GUIDO>MD NEWPROGS	$ mkdir newprogs
C:\GUIDO>MOVE PROG ..	$ mv prog ..
C:\GUIDO>MD \PROGS\TURBO	$ mkdir /progs/turbo
C:\GUIDO>DELTREE TEMP\TRASH	$ rm -rf temp/trash
C:\GUIDO>RD NEWPROGS	$ rmdir newprogs
C:\GUIDO>RD \PROGS\TURBO	$ rmdir /progs/turbo

Poznámky:

- Chcete-li použít příkaz `rmdir`, adresář musí být prázdný. Pokud chcete smazat adresář i s jeho obsahem, použijte (na vlastní nebezpečí) příkaz rm -rf.
- Znak „~“ je zkratka vašeho domovského adresáře. Příkazem cd nebo cd ~ se dostanete odkudkoliv přímo do svého domovského adresáře, příkaz cd ~/tmp vás přesune přímo do adresáře /home/váš_adresář/tmp.
- Příkaz cd – provede „undo“ poslední změny adresáře.

Diskety, pevné disky a podobně

V Linuxu jsou dvě možnosti jak pracovat se zařízeními – dosový způsob a linuxový způsob. Vyberte si sami.

Dosový způsob

Většina distribucí Linuxu obsahuje balík Mtools, což je skupina příkazů, které přesně odpovídají svým dosovým jmenovcům, pouze jejich názvy začínají písmenem „m“, například mformat, mdir, mdel, mmd a podobně. Umějí dokonce zachovávat dlouhé názvy souborů, ne už však jejich práva. Pokud si upravíte konfigurační soubor /etc/mtools.conf (jeho příklad je součástí distribuce), můžete pracovat i s dosovými oblastmi, s CD-ROM mechanikami a Zip mechanikami. Nicméně naformátování čisté diskety příkazem mformat se vám nepovede, k tomu musíte jako superuživatel zadat příkaz fdformat /dev/fd0H1440.

K souborům na disketě nemůžete přistupovat běžnými příkazy, například less a:soubor.txt. To je nevýhoda dosového postupu.

Linuxový způsob

UNIX používá jinou filozofii práce se zařízeními. Nemá žádné samostatné disky jako A: nebo C:, jakýkoliv disk, ať už pevný nebo disketa, se prostřednictvím operace nazvané *připojení (monnt)* stává součástí lokálního souborového systému. Když přestanete disk používat a chcete jej vyjmout, musíte jej *odpojit*.

Fyzické naformátování disku je jedna operace, vytvoření souborového systému je jiná věc. Příkaz FORMAT A: v DOSu provede oba úkony, v Linuxu jsou pro ně samostatné příkazy. Příkaz pro naformátování diskety byl uveden výše, souborový systém se vytváří příkazem:

```
# mkfs -t ext2 -c /dev/fd0H1440
```

Namísto souborového systému ext2 můžete použít i jiné typy souborových systémů – například dos nebo vfat (ten je nejvhodnější). Jakmile je disk připraven, připojíte jej příkazem:

```
# mount -t ext2 /dev/fd0 /mnt,
```

přičemž pokud nepoužíváte souborový systém ext2, musíte uvést typ vámi používaného souborového systému. Od této chvíle se soubory na disketě nacházejí v adresáři /mnt. Například:

DOS	**Linux**
`C:\GUIDO>DIR  A:`	`$ ls /mnt`
`C:\GUIDO>COPY A:*.*`	`$ cp /mnt/* .`
`C:\GUIDO>COPY *.ZIP A:`	`$ cp *.zip /mnt`
`C:\GUIDO>EDIT A:FILE.TXT`	`$ jstar /mnt/file.txt`
`C:\GUIDO>A:`	`$ cd /mnt`
`A:> _`	`/mnt/$ _`

Když skončíte práci, *musíte* disketu před vyjmutím z mechaniky odpojit:

```
# umount /mnt
```

Příkazy fdformat a mkfs se samozřejmě používají pouze na nenaformátované diskety. Pokud chcete pracovat s mechanikou B:, použijte v předchozích příkazech zařízení fd1H1440 a fd1 místo fd0H1440 a fd0.

Je zbytečné říkat, že to, co platí pro diskety, platí i pro jiná zařízení, například pokud budete chtít připojit jiný pevný disk nebo CD-ROM. CD disk připojíte následujícím příkazem:

```
# mount -t iso9660 /dev/cdrom /mnt
```

Takto vypadají „oficiální“ způsoby připojení disků, nicméně máme v zásobě i jeden trik. Vzhledem k tomu, že nutnost práv superuživatele k připojení diskety nebo CD disku je poměrně nepříjemná, existuje způsob, jak tuto operaci umožnit i normálním uživatelům.

- jako uživatel proveďte

```
# mkdir /mnt/floppy ; mkdir /mnt/cdrom
# chmod 777 /mnt/floppy /mnt/cd*
# # zkontrolujte, zda máte správně definovánu CD mechaniku[5]
# chmod 666 /dev/hdb ; chmod 666 /dev/fd*
```

5 Pozn. překladatele: Záleží na tom, jak máte CD mechaniku fyzicky připojenu (master/slave na primárním/sekundárním IDE) – pak může jít o zařízení /dev/hda až /dev/hdd.

- do souboru /etc/fstab přidejte

```
/dev/cdrom      /mnt/cdrom  iso9660 ro,user,noauto          0       0
/dev/fd0        /mnt/floppy vfat    user,noauto             0       0
```

K připojení diskety nebo CD mechaniky nyní můžete použít příkazy:

```
$ mount /mnt/floppy
$ mount /mnt/cdrom
```

Do adresářů /mnt/floppy a /mnt/cdrom nyní může přistupovat každý uživatel. Tento typ přístupu nicméně představuje bezpečnostní díru, pokud vás to tedy trápí...

Dva užitečné příkazy jsou df, který nabídne informace o připojených souborových systémech, a du <adresář>, který zjistí, jaká je velikost daného adresáře.

Zálohování

Pro potřeby zálohování existuje několik specializovaných programových balíků, přinejmenším však (jako superuživatel) můžete zálohu na více disket vytvořit takto:

```
# tar -M -cvf /dev/fd0H1440 adresář_k_zálohování/
```

Mějte v mechanice připravenou prázdnou naformátovanou disketu a příslušný počet dalších v zásobě. Obnovení zálohy provedete následujícím příkazem (v mechanice musí být první disketa archivu):

```
# tar -M -xpvf /dev/fd0H1440
```

A co Windows?

„Ekvivalentem“ Windows je grafický systém X Window System. Na rozdíl od Windows nebo MacOS nebyl systém X11 navržen tak, aby byl jednoduchý nebo aby vypadal pěkně, jeho úlohou je čistě řešit grafické rozhraní unixových pracovních stanic. Uveďme si hlavní rozdíly:

- Zatímco Windows vypadají na celém světě stejně, pro X11 to neplatí – jsou mnohem více konfigurovatelné. Vnější vzhled systému X11 je určen klíčovou komponentou zvanou „správce oken“. Zde máte hodně na výběr: `fvwm`, jednoduchý, ale pěkný a nenáročný na paměť, `fvwm2-95`, `Afterstep`, `WindowMaker`, `Enlightenment` a řada dalších. Správce oken se obvykle spouští ze souboru .xinitrc.
- Správce oken lze nastavit tak, aby se okna chovala stejně jako ve Windows: po klepnutí myší na okno toto přejde do popředí. Další možnost je, že přejde do popředí bezprostředně poté, co na okno najedete myší. Lze také nastavit, zda se mají okna na obrazovce umisťovat automaticky nebo manuálně. Pokud se místo okna s programem objeví pouze rámeček, klepněte myší tam, kde jej chcete mít.
- Většinu funkcí lze nastavit úpravou nějakého konfiguračního souboru. Přečtěte si dokumentaci ke správci oken – konfigurační soubor se může jmenovat *.fvwmrc*, *.fvwm2rc95*, *.steprc* a tak dále. Vzorový konfigurační soubor se typicky nachází v adresáři */etc/X11/název_správce_oken/system.název_správce_oken*.
- Grafické aplikace se vytvářejí pomocí speciálních knihoven (tzv. „widgetů“). Existuje jich několik a proto mohou aplikace vypadat různě. Nejjednodušší aplikace používají widgety Athena (*xdvi*, *xman*, *xcalc*), další Motif (*netscape*), jiné Tcl/Tk, Qt, Gtk, XForms a jiné. Prakticky všechny tyto knihovny poskytují přibližně stejný „windowsoidní“ vzhled aplikací.

- Chování aplikací se může občas lišit. Pokud například myší označíte kus textu a stisknete <Backspace>, očekáváte, že se text smaže. Nebude to ovšem fungovat v aplikacích založených na widgetech Athena, v jiných to fungovat bude.
- Chování posuvných lišt a změny velikosti oken závisí na správci oken a použité sadě widgetu. Tip: pokud zjistíte, že posuvné lišty se nechovají podle očekávání, zkuste místo levého tlačítka použít prostřední tlačítko (nebo levé a pravé současně).
- Aplikace standardně nemají ikonu, lze jim však libovolnou nastavit. Většina správců oken umožňuje klepnutím pravého tlačítka na ploše (kořenovém okně) vyvolat nabídku s různými volbami. Vzhled kořenového okna můžete změnit příkazy *xsetroot* nebo *xloadimage*.
- Schránka může obsahovat pouze text a chová se poněkud zvláště. Jakmile nějaký text označíte, kopíruje se do schránky okamžitě – přesuňte se myší kamkoliv a vložte text stisknutím prostředního tlačítka. Aplikace *xclipboard* umožňuje vytvářet více schránek.
- Funkce „táhni a pusť" je k dispozici pouze v aplikacích a ve správcích oken, které ji podporují.

A teď dobré zprávy. Existují projekty, které se snaží o to, aby se X11 aplikace chovaly stejně jako ve Windows. Jsou to například Gnome (http://www.gnome.org) nebo KDE (http://www.kde.org). S největší pravděpodobností jsou oba součástí vaší distribuce. Plocha Windows vám tak už nikdy nebude chybět.

Úpravy systému

Inicializační soubory systému

V DOSu jsou dva důležité soubory – `AUTOEXEC.BAT` a `CONFIG.SYS`, které slouží k inicializaci systému při jeho spouštění, k nastavení některých proměnných prostředí (jako jsou PATH nebo FILES) a k případnému spouštění dalších programů. Windows navíc mají nechvalně známý registr – jeden z nejhorších nápadů v celé historii výpočetní techniky.

V Linuxu existuje řada inicializačních souborů, některé z nich je přitom velmi rozumné neměnit, pokud přesně nevíte, co děláte. Najdete je v adresáři */etc*. Všechny konfigurační soubory mají formát prostého textu. Pokud chcete pouze nastavovat proměnnou PATH a jiné proměnné prostředí, nebo chcete změnit přihlašovací výzvu nebo automaticky po přihlášení spustit nějaký program, podívejte se na následující soubory:

Soubor	Poznámka
/etc/issue	Výzva před přihlášením
/etc/motd	Výzva po přihlášení
/etc/profile	Nastavuje proměnnou PATH a další
/etc/bashrc	Nastavuje aliasy, funkce a další
/home/*váš_domovský_adresář*/.bashrc	Nastavuje vaše aliasy a funkce
/home/*váš_domovský_adresář*/.bash_profile nebo /home/*váš_domovský_adresář*/.profile	Nastavuje vaše prostředí a spouští programy

Pokud existuje poslední uvedený soubor (všimněte si, že jde o skrytý soubor), bude po přihlášení přečten a vykonají se v něm uvedené příkazy.

Podívejme se na příklad souboru *.bash_profile*:

```
# Toto je komentář
echo Prostředí:
printenv | less   # ekvivalent příkazu SET v DOSu
alias d='ls -l'   # co je alias je snad jasné...
alias up='cd ..'
echo "Spouštěcí cesta je "$PATH
echo "Dneska je `date`"  # vytiskne výstup příkazu 'date'
echo "Přeji pěkný den, "$LOGNAME
# Toto je "funkce příkazového interpretu":
ctgz() # Vypíše obsah archivu .tar.gz
{
  for file in $*
  do
    gzip -dc ${file} | tar tf -
  done
}
# konec souboru .profile
```

`$PATH` a `$LOGNAME`, jak jste jistě uhodli, jsou proměnné prostředí. Těchto proměnných existuje celá řada. Přečtěte si manuálové stránky aplikací jako *less* nebo *bash*.

Následující řádek v */etc/profile* zajistí přibližně podobné chování jako DOSový příkaz `PROMPT $P$G`:

```
export PS1="\w\\$ "
```

Inicializační soubory programů

V Linuxu lze prakticky všechno upravit podle vašich potřeb. Většina programů má jeden nebo více konfiguračních souborů, se kterými si můžete hrát. Typicky jde o soubor *.název_programu* v domovském adresáři. První kandidáti na úpravy jsou:

- `.inputrc`: používá jej `bash` k nastavení chování kláves;
- `.xinitrc`: používá jej `startx` k inicializaci systému X Window;
- `.fvwmrc`: používá jej správce oken `fvwm`.
- `.joerc, .jstarrc`: používá jej textový editor `joe`;
- `.jedrc`: používá jej textový editor `jed`;
- `.pinerc`: používá jej poštovní klient `pine`;
- `.Xdefaults`: používá jej řada grafických programů.

U všech těchto souborů, i u dalších, na které dříve či později narazíte, doporučujeme přečíst si manuál. Kromě toho vám můžeme doporučit praktický návod „Konfigurace systému (kapitola 1)“, http://www.linuxdoc.org/HOWTO/Config-HOWTO.html?

Sítě – základy

I v Linuxu existuje „Telefonické připojení“, navíc je rychlejší a stabilnější. Tato hračka se jmenuje „PPP“, což je protokol používaný k propojení počítačů modemem. Jediné co potřebujete je nástroj pro vytočení čísla a navázání spojení.

Ke čtení pošty ze serveru vašeho poskytovatele potřebujete nástroj nazvaný „email fetcher“, který používá protokol POP. Ten automaticky stahuje vaši poštu a ta se tváří, jako by dorazila přímo na váš počítač. Pak použijete MUA (Mail User Agent – tedy poštovní klient) jako *pine*, *mutt* nebo *elm* k práci s poštou.

Zatímco ve Windows se telefonické připojení aktivuje automaticky po spuštění nějaké internetové aplikace, v Linuxu to funguje opačně – nejprve navážete připojení a pak spustíte aplikaci. Existuje program *diald*, který zajišťuje podobné chování jako ve Windows. Instalace a konfigurace telefonického připojení k síti patřila k jedněm z nejsložitějších úkonů – to už ale dávno není pravda. Přečtěte si praktický návod „Konfigurace systému (kapitola 1)“.

A konečně něco o „Okolních počítačích“ – můžete dosáhnout toho, že linuxový počítač se bude v lokální síti s počítači Windows objevovat úplně stejně jako jiné stanice. Ta kouzelná věc se jmenuje Samba – a nejde o známý latinskoamerický tanec, ale o implementaci protokolu SMB v Linuxu. Viz `htttp://www.samba.org/`

Něco o programování

Skripty příkazového interpretu – nadupané soubory .BAT

Pokud jste soubory .BAT používali pouze jako zkratku za dlouhé příkazy, můžete stejného efektu dosáhnout vytvořením příslušných aliasů (viz příklad výše) v souborech *.profile* nebo *.bash_profile*. Pokud jste ale používali složitější operace, pak se vám určitě hodně zalíbí skriptovací jazyk obsažený v příkazovém interpretu – je stejně mocný jako starý dobrý Basic, ne-li ještě lepší. Má proměnné, struktury jako *while*, *for*, *case*, *if...then...else* a spoustu dalších věcí – lze jej použít jako slušnou alternativu „opravdových“ programovacích jazyků.

K vytvoření skriptu (tedy ekvivalentu souboru .BAT v DOSu) stačí vytvořit normální textový soubor s příkazy, uložit jej a pak jej nastavit jako spustitelný příkazem *chmod +x <název_skriptu>*. Spustíte jej zadáním jeho názvu.

Malé upozornění. Velmi rozšířený textový editor se jmenuje *vi*, a podle mých zkušeností jej noví uživatelé považují za velmi komplikovaný. Nebudu vysvětlovat, jak jej použít; doporučuji knihu Matta Welshe nebo nějaký jiný návod na Internetu. Pro tuto chvíli nám stačí vědět, že:

- Při vkládání textu zadejte `i` a pak text.
- K mazání znaků stiskněte <ESC> a pak `x`.
- K ukončení editace bez uložení stiskněte <ESC> a pak `:q!`
- K ukončení editace s uložením stiskněte <ESC> a pak `:wq`.

Dobrým editorem pro začátečníky je *joe*. Spustíte-li jej příkazem *jstar*, bude se chovat jako běžný editor v DOSu/Windows. Ještě lepší je *jed* v režimu WordStar nebo IDE. Kde tyto editory sehnat se dočtete v části *Kde hledat aplikace*.

Vytváření skriptů pro shell je natolik rozsáhlé téma, že vydá na samostatnou knihu, a my se této problematice nebudeme podrobněji věnovat. Uvádíme pouze příklad skriptu, z nějž můžete pochopit některá základní pravidla.

```
#!/bin/sh
# sample.sh
# Toto je komentář
# Neměňte první řádek, je nutný!!!
echo "Pracujete v systému: `uname -a`" # použije výstup příkazu
echo "Můj název je $0" # vestavěná proměnná
echo "Zadali jste mi $# následujících parametrů: "$*
echo "První parametr je: "$1
echo -n "Jak se jmenuješ? " ; read your_name
echo Všimněte si rozdílu: "hi $your_name" # uvození znakem "
echo Všimněte si rozdílu: 'hi $your_name' # uvození znakem '
DIRS=0 ; FILES=0
for file in `ls .` ; do
  if [ -d ${file} ] ; then # pokud je to adresář
    DIRS=`expr $DIRS + 1`  # DIRS = DIRS + 1
  elif [ -f ${file} ] ; then
    FILES=`expr $FILES + 1`
  fi
  case ${file} in
    *.gif|*jpg) echo "${file}: grafický soubor" ;;
    *.txt|*.tex) echo "${file}: textový soubor" ;;
    *.c|*.f|*.for) echo "${file}: zdrojový soubor" ;;
    *) echo "${file}: obecný soubor" ;;
  esac
done
echo "Našel jsem ${DIRS} adresářů a ${FILES} souborů"
ls | grep "ZxY--%%WKW"
if [ $? != 0 ] ; then # návratový kód posledního příkazu
  echo "ZxY--%%WKW nenalezen"
fi
echo "A to je vše... Chceš-li více, zadej 'man bash'"
```

Jazyk C

V Unixu je hlavním systémovým jazykem C, ať už se vám to líbí nebo ne. Existují samozřejmě i implementace řady dalších jazyků (Java, FORTRAN, Pascal, Lisp, Basic, Perl, awk...).

Vycházejme z toho, že jazyk C znáte – uvedeme si základní návod pro ty, kteří jsou zkažení nástroji jako Turbo C++ a podobnými dosovými věcmi. Překladač jazyka C v Linuxu se jmenuje *gcc* a neobsahuje žádné z parádiček, které obvykle najdete v dosových nástrojích: nemá grafické prostředí, nápovědu, vestavěný debugger, ani nic podobného. Je to prostý řádkový překladač, nesmírně mocný a efektivní. K překladu standardního souboru *hello.c* zadáte:

```
$ gcc hello.c
```

Tím vznikne spustitelný soubor pojmenovaný *a.out*. Má-li se jmenovat jinak, zadejte:

```
$ gcc -o hola hello.c
```

Ke slinkování programu s nějakou knihovnou použijte přepínač *-l<název_knihovny>*. K přilinkování matematické knihovny tedy zadáte:

```
$ gcc -o mathprog mathprog.c -lm
```

(Přepínač *-l<název_knihovny>* způsobí přilinkování knihovny */usr/lib/lib<název_knihovny>.so*, tedy *-lm* přilinkuje knihovnu */usr/lib/libm.so*.)

Až sem je to jednoduché. Pokud se ovšem program skládá z více zdrojových souborů, budete muset použít nástroj *make*. Řekněme, že jste napsali program pro zpracování výrazů. Zdrojový soubor se jmenuje *parser.c* a #includuje dva soubory *parser.h* a *xy.h*. Funkce programu *parser.c* pak chcete použít v programu, řekněme *calc.c*, který #includuje *parser.h*. Správný chaos, že? Jak teď přeložíme program *calc.c*?

Budete muset vytvořit soubor *Makefile*, který překladači vysvětluje závislosti mezi zdrojovými a objektovými soubory. V našem příkladu:

```
# Makefile, pro překlad calc.c
# Kde je uvedeno, použijte klávesu <TAB>!

calc: calc.o parser.o
<TAB>gcc -o calc calc.o parser.o -lm
# calc závisí na dvou objektových souborech: calc.o a parser.o

calc.o: calc.c parser.h
<TAB>gcc -c calc.c
# calc.o závisí na dvou zdrojových souborech

parser.o:  parser.c parser.h xy.h
<TAB>gcc -c parser.c
# parser.o závisí na třech zdrojových souborech

# konec souboru Makefile.
```

Uložte tento soubor jako *Makefile* a příkazem *make* přeložte program. Nebo jej také můžete uložit jako *calc.mak* a zadat *make -f calc.mak*. A samozřejmě si přečtete manuál. V manuálu můžete získat nápovědu i k různým funkcích jazyka C, jsou popsány ve třetí části manuálu. Například:

```
$ man 3 printf
```

K ladění programů slouží *gdb*. Jak s ním zacházet se dozvíte příkazem *info gdb*.

Existuje celá řada knihoven – mezi prvními, které budete možná používat, budou *ncurses* (grafické efekty v textovém režimu) a *svgalib* (konzolová grafika). Řada editorů umí fungovat v režimu vývojového prostředí – například *emacs* i *jed* podporují barevné zvýraznění syntaxe, automatické odsazování a podobně. Můžete si také pořídit program *rhide* (ftp://metalab.unc.edu:/pub/Linux/devel/debuggers/) – je to klon vývojového prostředí Borland a možná se vám bude líbit.

Programování pro X11

Pokud se cítíte na programování grafických aplikací (není to tak složité), existuje několik knihoven, které to výrazně usnadňují. Důležitá místa k návštěvě jsou stránky http://www.gtk.org projektu GTK+ a http://www.troll.no projektu Qt. GTK+ je skupina widgetů původně sloužících při tvorbě grafického editoru The GIMP (http://www.gimp.org) a používá je prostředí Gnome. V prostředí KDE se používá objektová knihovna Qt. Jednu z nich budete pravděpodobně používat.

Mezi nejlepší nástroje pro vizuální programování patří Kdevelop pro Qt, http://www.kdevelop.org, a Glade pro GTK+, http://glade.pn.org. Další informace najdete na adrese – http://www.free-soft.org/guitool/.

Multiplatformní programování

Nebylo by pěkné napsat zdrojový kód, který se přeloží pod Linuxem i pod Windows? V době vzniku tohoto textu existuje několik widgetů, které umožňují více či méně stabilní multiplatformní programování. Co se týče stability a úplnosti, doporučuji víceméně pouze jediný nástroj – FLTK, Fast Light Tool Kit http://www.fltk.org. Je malý, rychlý a stabilní. Obsahuje také semi-vizuální nástroj pojmenovaný Fluid.

Zbývající jedno procento

Tedy popravdě řečeno, víc než jedno procento...

Spouštění DOS/Win aplikací

Ano, s jistými omezeními lze v Linuxu spouštět aplikace určené pro DOS a Widows. Existují dva poměrně dobré emulátory: Dosemu (http://www.dosemu.org) a Wine (http://www.winehq.com). Wine je čím dále lepší a lepší a rozrůstá se seznam použitelných aplikací. Dokáže dokonce spustit i Word a Excel!

Programy tar a gzip

V Linuxu existuje několik rozšířených aplikací pro archivaci a komprimaci souborů. K vytváření archivů slouží program *tar* – je to něco jako PKZIP nebo Winzip, neprovádí však komprimaci, pouze archivuje. Nový archiv vytvoříte příkazem:

```
$ tar cvf <jméno_archivu.tar> <soubor> [soubor...]
```

K vybalení souborů z archivu slouží příkaz:

```
$ tar xvf <jméno_archivu.tar> [soubor...]
```

Obsah archivu vypíšete příkazem:

```
$ tar tf <jméno_archivu.tar> | less
```

Soubory můžete komprimovat příkazem *compress*, který je zastaralý a jeho používání se nedoporučuje, nebo *gzip*:

```
$ compress <soubor>
$ gzip <soubor>
```

Vzniknou komprimované soubory s příponou *.Z* (*compress*) nebo *.gz* (*gzip*). Tyto programy umějí komprimovat vždy jen jeden soubor. Dekomprimaci provedete příkazem:

```
$ compress -d <file.Z>
$ gzip -d <file.gz>
```

Viz manuál.

Existují také programy *unarj*, *zip* a *unzip* (kompatibilní s PK ZIP). Stejně běžné jako soubory .ZIP v DOSu jsou v unixovém světě soubory s příponou *.tar.gz* nebo *.tgz* (archivované *tar* a komprimované *gzip*). Obsah archivu *.tar.gz* vypíšete příkazem:

```
$ tar ztf <soubor.tar.gz> | less
```

Instalace aplikací

První věc: instalace aplikací je práce superuživatele. Většina linuxových aplikací se distribuuje jako archivy *.tar.gz*, které typicky obsahují adresář s potřebnými soubory. Rozumné je tyto balíky instalovat do */usr/local* příkazem:

```
# tar zxf <archív.tar.gz>
```

a pak si přečíst soubor README nebo INSTALL. Ve většině případů se aplikace distribuují ve zdrojové formě a budete je muset přeložit. Typicky stačí spustit příkazy *make* a pak *make install*. Pokud archiv obsahuje skript *configure*, spouští se jako první. Samozřejmě budete potřebovat překladač *gcc* nebo *g++*.

Jiné archivy je nutné rozbalovat v kořenovém adresáři, typicky je to příklad archivů *.tgz* distribuce Slackware. V některých případech archiv obsahuje soubory, ale ne adresář – takže pozor, ať na disku nevyrobíte zmatek. Před rozbalením archivu se vždycky podívejte, co obsahuje.

Debian a RedHat (SuSE, Mandrake a další) používají vlastní formáty archivů – *.deb* a *.rpm*. Poslední zmíněný formát používají i jiné distribuce, k instalaci *rpm* balíku použijte příkaz:

```
# rpm -i package.rpm
```

Užitečné tipy

Skrolování obrazovky: <Shift><PageUp> umožňuje návrat na několik předchozích stránek na konzole.

Reset konzoly: Pokud se vám podaří příkazem *more* nebo *cat* vypsat binární soubor, může obrazovka vypadat tragicky. Napravíte to tak, že naslepo zadáte příkaz *reset* nebo *echo* <Ctrl>V <Esc> c <Return>.

Vkládání textu: Na konzole, viz níže, v systému X Window označte text myší a vložte jej stisknutím prostředního tlačítka (nebo levého a pravého současně, máte-li dvoutlačítkovou myš). Existuje rovněž aplikace *xclipboard*, nenechte se zaskočit její pomalou odezvou.

Použití myši: Pokud máte instalován program *gpm*, ovladač myší pro konzolu, můžete text rovněž označit myší a vložit stisknutím prostředního tlačítka. Tato funkce funguje i mezi virtuálními konzolami.

Zprávy systému: Jako superuživatel se podívejte na soubory */var/adm/messages* nebo */var/log/messages*, kde se dozvíte, co vám chce říct jádro systému. Užitečný je také příkaz *dmesg*.

Kde hledat aplikace

Pokud sháníte náhradu svých oblíbených DOS/Win programů jejich linuxovými protějšky, doporučujeme hlavní archiv linuxových programů na ftp://metalab.unc.edu/pub/Linux. Další dobré místo je „Linux Applications and Utilities Page" http://www.xnet.com/~blatura/linapps.shtml, „oficiální" linuxová stránka http://www.linux.org, a archivy http://freshmeat.net a http://www.sourceforge.net.

Věci, které v DOSu nejde udělat

Linux dokáže spoustu věcí, které by se v DOS/Windows dělaly jen velmi obtížně, komplikovaně nebo vůbec. Krátký seznam toho, co můžete jednoduše udělat:

- Program *at* umožňuje spuštění programu v nastavenou dobu.

- *awk* je jednoduchý a přitom mocný jazyk pro práci s (nejen) datovými soubory. Řekněme, že *data.dat* je soubor s daty, kdy každý řádek obsahuje několik údajů. Příkazem

  ```
  $ awk '$2 ~ "abc" {print $1, "\t", $4}' data.dat
  ```

  vytisknete 1. a 4. údaj na každém řádku, jehož 2. údaj obsahuje text „abc".

- Program *cron* umožňuje opakovaně spouštět úlohy v nastavenou dobu. Viz *man 5 crontab*.
- Příkaz *file <soubor>* vám řekne, co je soubor zač (text, program, archiv a podobně).
- Příkaz *find* (viz také sekce *Adresáře – překlad příkazů*) je nesmírně mocný a užitečný příkaz. Umožňuje hledat soubory, které splňují nějaké podmínky a provádět s nimi operace. Obecné použití tohoto příkazu je:

  ```
  $ find <adresář> <výraz>,
  ```

  kde *<výraz>* obsahuje vyhledávací kritéria a prováděné akce. Například:

  ```
  $ find . -type l -exec ls -l {} \;
  ```

  nalezne všechny symbolické odkazy v aktuálním adresáři a ukáže, kam směřují.

  ```
  $ find / -name "*.old" -ok rm {} \;
  ```

  nalezne všechny soubory odpovídající zadanému požadavku a vymaže je, přičemž se ptá na potvrzení.

  ```
  $ find . -perm +111
  ```

  nalezne všechny soubory s právy 111 (spustitelné).

  ```
  $ find . -user root
  ```

  nalezne všechny soubory patřící uživateli *root*. Alternativ je celá řada – viz manuál.

- Příkaz *grep* hledá v souborech text. Například:

  ```
  $ grep -l "geology" *.tex
  ```

  vypíše soubory *.tex, které obsahují slovo „geology". Mutace *zgrep* hledá v komprimovaných archivech. Viz manuál.

- Regulární výrazy představují složitý ale nesmírně mocný způsob provádění operací nad textem. Například `^a[^a-m]X{4,}txt$` vybírá řádky začínající znakem „a" následovaným libovolným znakem kromě znaků „a" až „m", za nímž následuje 4 a více „X" a končí „txt". Regulární výrazy se používají v textových editorech, s příkazem less a u celé řady dalších programů. Viz *man grep*.
- Příkaz *script <soubor>* vypisuje obsah obrazovky do souboru, dokud nezadáte příkaz *exit*. Užitečné pro ladění.
- Příkaz *sudo* umožňuje běžným uživatelům provádět některé úkony, které typicky může provést pouze superuživatel (například připojování disků, formátování a podobně). Viz manuál.
- Příkaz *uname* zobrazí informace o systému.
- Příkazy *zcat* a *zless* jsou užitečné k prohlížení a zpracovávání archivů bez nutnosti jejich rozbalení. Například:

```
$ zless textfile.gz
$ zcat textfile.gz | lpr
```

- Další užitečné příkazy jsou *bc*, *cal*, *chsh*, *cmp*, *cut*, *fmt*, *head*, *hexdump*, *nl*, *passwd*, *printf*, *sort*, *split*, *strings*, *tac*, *tail*, *tee*, *touch*, *uniq*, *w*, *wall*, *wc*, *whereis*, *write*, *xargs*, *znew*. Viz manuál.

Zkuste si Unix pod DOSem

Věřte tomu nebo ne, existují nástroje, které pod DOS/Windows nabízejí podobné prostředí jako v Unixu. Jedním z nich je balík Djgpp (http://www.delorie.com/djgpp/) pro DOS, další je Cygwin (http://www.cygnus.com/cygwin) pro Windows. Oba obsahují stejné nástroje jako Linux, nezískáte však samozřejmě stabilitu a výkon Linuxu.

Pokud si chcete chování Linuxu vyzkoušet, použijte Djgpp. Stáhněte si a nainstalujte soubory (aktuální verze v době vzniku tohoto dokumentu byla 2.02): `djdev202.zip`, `bnu281b.zip`, `bsh1147b.zip`, `fil316b.zip`, `find41b.zip`, `grep22b.zip`, `gwk303b.zip`, `lss332b.zip`, `shl112b.zip`.. Instalační instrukce jsou přiloženy, další informace můžete najít na news:comp.os.msdos.djgpp.

Použití *bash*e v DOSu/Windows působí jako závan čerstvého vzduchu. Ke správnému nastavení upravte soubor BOOT.BAT podle vašeho systému a pak nahrajte do domovského adresáře (Windows) příslušné soubory:

```
# _bashrc

LS_OPTIONS="-F -s --color=yes"
alias cp='cp -i'
alias d='ls -l'
alias l=less
alias ls="ls $LS_OPTIONS"
alias mv='mv -i'
alias rm='rm -i'
alias u='cd ..'
# this is _bprof
if [ -f ~/_bashrc ]; then
  . ~/_bashrc
fi
PS1='\w\$ '
PS2='> '
CDPATH="$CDPATH:~"
# pro less(1)
LESS="-M-Q"                     # tichá dlouhá výzva
LESSEDIT="%E ?lt+%lt. %f"       # editace horního řádku
VISUAL="jed"                    # editor
LESSCHARSET=latin1              # zobrazit znaky s diakritikou
export PS1 PS2 CDPATH LS_OPTIONS LESS LESSEDIT LESSOPEN VISUAL LESSCHARSET
```

Běžné přípony a odpovídající programy

Různých přípon existuje strašně moc. Základní přehled bez různých výjimek (fonty a podobně) vypadá takto:

- `1 ... 8`: manuálové stránky. Čtěte příkazem `groff -Tascii -man <soubor.1>`.
- `arj`: archivy programu `arj`.
- `dvi`: výstup progamu TeX (viz níže). Zobrazte programem `xdvi` , konverzi na PostScript provedete příkazem `dvips`.
- `gz`: archivy programu `gzip`.
- `info`: info soubor (Něco jako manuál). Příkaz `info`.
- `lsm`: Linux Software Map. Textový soubor obsahující popis balíku.
- `ps`: PostScript. Zobrazte nebo vytiskněte programem `gs`, popř. `ghostview` nebo `gv`.
- `rpm`: balíček RPM. Instaluje se programem *rpm*.
- `taz, tar.Z`: archivy programu *tar* komprimované programem *compress*.
- `tgz, tar.gz`: archivy programu tar komprimované programem *gzip*.
- `tex`: textový soubor pro TeX, výkonný sázecí systém. Program *tex* existuje pro většinu distribucí.
- `texi`: soubor texinfo, umí vytvořit jako soubory TeX, tak info. Příkaz `texinfo`.
- `xbm, xpm, xwd`: grafické soubory.
- `Z`: archivy programu *compress*.

Konverze souborů

Pokud potřebujete přenášet textové soubory mezi DOSem/Windows a Linuxem, narazíte na problém konce řádku. V DOSu končí každý řádek textu znaky CR/LF (tedy ASCII 13 a ASCII 10), zatímco v Linuxu pouze znakem LF. Pokud budete DOSový soubor editovat v Linuxu, bude asi každý řádek končit divným znakem „M“, linuxový soubor v DOSu bude vypadat jako jeden jediný dlouhatánský řádek. Ke konverzi slouží dvojice nástrojů *dos2unix* a *unix2dos*.

Pokud používáte diakritiku, používejte kódování Windows (například program Notepad) a ne kódování DOSu. V opačném případě vznikne zmatek.

Kancelářské balíky

Ano, pro Linux existují kancelářské balíky zadarmo!

Asi nejlepší je dnes OpenOffice.org (http://www.openoffice.org), který nabízí podobné funkce jako Microsoft Office. Umí dokonce číst a ukládat soubory ve formátu Word a Excel, i když převod vždy nedopadne perfektně. Existuje i v české verzi (http://oo-cs.sf.net).

Dalšími alternativami jsou Koffice (http://www.koffice.org) a Corel WordPerfect, http://www.corel.com.

KAPITOLA 14

Linux a Windows NT

Originál: http://tldp.org/HOWTO/mini/Linux+WinNT.html

Tento dokument popisuje několik způsobů, jak nainstalovat Linux a Windows NT na stejném počítači a jak tyto systémy zavádět prostřednictvím programu LILO. Existuje rovněž dokument „Linux+NT-Loader“, který popisuje, jak tyto systémy zavádět pomocí programu NT Loader. Protože Windows 2000 považuji za NT 5.0, popisujeme také upgrade NT 4.0 na 2000.

Úvod

Ve verzi 1.1 dokumentu Linux+WindowsNT mini-HOWTO píše Bill Wohler:

Díky hardwarovým nebo softwarovým problémům, nebo díky neschopnosti užívatele jsem strávil několik těžkých dní pokusy o instalaci NT a Linuxu na nový počítač HP Vectra.

Sám za sebe mohu říci, že jsem se potkal se stejným problémem, nicméně někdy prostě **opravdu** potřebujete na jednom počítači provozovat Windows NT i Linux a přepínat se mezi těmito systémy.

Za žádných okolností neformátujte oddíly programem Disk Administrator z NT 3.51. Ptá se, zda může na disk zapsat signaturu, která „nezpůsobí žádné potíže“. Jakmile to provedl, poškodil tabulku tak, že ji bylo možné opravit až níže popsanými kroky 3 a 7. Díky tomu jsem byl omezen na jedinou z diskových oblastí NT FAT. Kromě toho, i kdyby se vám podařilo Disk Administrator přimět k rozumné činnosti, tak pokud nepoužíváte Linux s podporou NTFS, budete potřebovat ještě jednu samostatnou FAT oblast pro přenos souborů mezi Linuxem a Windows NT.

Disk Administrator z Windows NT je vhodný nástroj k tomu, abyste zjistili, jak vaše disky vypadají před a poté, co použijete nástroj Partition Magic společnosti Power Quest. Budete jej možná potřebovat ke zmenšení velikosti NT oblasti (ať už NTFS nebo FAT) kvůli vytvoření volného místa pro linuxovou oblast. (Později jsem zjistil, že zmenšení používaných oblastí nemusí být nutné. Pokud začínáte instalaci od počátku, bude asi nejrozumnější od počátku rozdělit celý disk programem FDISK. Budete potřebovat spouštěcí disketu DOSu s programy FDISK a FORMAT. Podrobnosti později.)

Nejprve jsem instaloval Linux a pak NT, nicméně teď už bych zvládl i nejprve instalovat NT a pak Linux.

Samozřejmě, že zvládneme nejprve instalovat NT a pak Linux. Ukážeme si, jak to udělat a jak pomocí programu LILO (LInux LOader) volit spouštěný operační systém. Než k tomu dojdeme, začneme nejprve postupem, který použil Bill Wohler, předchozí správce tohoto dokumentu.

Jak instalovat: Nejprve Linux, pak Windows NT

1. Nainstalujte nutné minimum Linuxu (nemá smysl instalovat zbytek, dokud nevyřešíte problémy s Linuxem a NT). Veškeré vytváření oddílů proveďte v Linuxu včetně vytvoření oddílu pro NT (vytvořte jej jako FAT). Nepodařilo se mi vytvořit více než jeden oddíl pro NT. Sám jsem tento oddíl vytvořil jako první, nevím ale, jestli to je či není nutné.
2. Upravte */etc/lilo.conf* a nastavte *boot=/dev/sda*. (Nepodařilo se mi nainstalovat LILO na linuxovou oblast, v mém případě */dev/sda3*.) Spusťte *lilo*.
3. Uložte MBR záznam příkazem *dd if=/dev/sda of=/dev/fd0 bs=512 count=1*. Opravdu jej uložte na disketu. Věřte mi. Udělejte to pokaždé, když změníte tabulku tabulku diskových oblastí.
4. Proveďte první část instalace NT. Když se budou v průběhu instalace rebootovat, spusťte Linux.
5. Do */etc/lilo.conf* doplňte záznam pro Windows NT:

   ```
   other=/dev/sda1
   label=NT
   table=/dev/sda
   ```

   a spusťte *lilo*. Pokud si bude lilo stěžovat (zapomněl jsem přesné znění oné hlášky), doplňte do */etc/lilo.conf* vedle klíčového slova „compact“ ještě údaj „linear“. Jestliže NT poškodily tabulku diskových oblastí, budete muset buď použít volbu „ignore-table“, nebo použijte postup podle bodu 7. Viz také volba „fix-table“. Vaším nejlepším přítelem je dokument LILO HOWTO.
6. Restartujte počítač, nabootujte NT a dokončete instalaci!
7. Zpátky v Linuxu spusťte *fdisk* a ověřte, zda na linuxových oddílech neuvidíte hlášení „partition doesn't end on cylinder boundary“. Na oddílu NT se toto hlášení možná objeví, ale tam to zjevně nevadí.

   ```
   /dev/sda1            1        1      322   329301    6  DOS 16-bit =32M
   Partition 1 does not end on cylinder boundary:
   phys=(321, 39, 9) should be (321, 63, 32)
   ```

   cfdisk hlásí divné údaje, ale vše vypadá v pořádku:

   ```
                                      Unusable                    0.04*

   /dev/sda1              Primary     DOS 16-bit =32Mb          321.59*

                                      Unusable                    0.39*
   ```

   Pokud se objeví hlášení o hranicích cylindrů u oddílů s Linuxem, stačí použít *cfdisk* a provést něco nevinného, například změnit boot sektor.

   Pokud ovšem NT poškodily tabulku diskových oblastí a *cfdisk* ani nejde spustit, protože hlásí, že nemůže otevřít */dev/sda*, musíte přistoupit k radikálnějšímu kroku. Budete potřebovat dříve uložený MBR. Přepište MBR a obnovte jej z diskety (ovšem bez signatury):

   ```
   dd if=/dev/zero of=/dev/sda bs=512 count=1
   dd if=/dev/fd0 of=/dev/sda bs=510 count=1
   ```
8. Nainstalujte zbytek Linuxu. Jednoduché, že?

Pokud chcete, aby do MBR zapisovaly NT a ne LILO, budete muset MBR nejprve přepsat následujícím postupem:

- *dd if=/dev/zero of=/dev/sda bs=446 count=1* (v Linuxu), nebo pomocí nástrojů SCSI disku proveďte nízkoúrovňový formát disku. (Nízkoúrovňový formát IDE disků je prý může poškodit, tak jej nepoužívejte.)
- Spusťte *fdisk /mbr* (máte přece vytvořenu bootovací disketu DOSu s programem FDISK)
- Vymažte NT oblast a znovu ji vytvořte při instalaci NT.
- Dokončete instalaci NT.

Epilog: Po prvním zveřejnění tohoto textu jsem od různých lidí zaslechl, že neměli s Disk Administratorem z NT žádné problémy, a že se jim podařilo bez komplikací jako první nainstalovat ať Linux nebo NT. S trochou štěstí rozšíříte jejich řady, pokud ne, snad vám bude předchozí text užitečný.

Jedná se o první verzi tohoto dokumentu, existují však i další dokumenty na toto téma. Přečtěte si je také. Pokud se vaše zkušenosti liší od zde popsaných, buď napište vlastní mini-HOWTO, anebo pošlete doplněk k tomuto dokumentu jeho správci. Možná časem někdo zkombinuje všechny podobné dokumenty do jednoho konzistentního.

Konečně musím dodat, že výše uvedené je vše, co o daném problému vím. Další informace získáte nejlépe prostřednictvím vhodné diskusní skupiny. Sám používám NT zhruba jeden den v roce, a to ještě pod nátlakem.

Jak instalovat: Nejprve Windows NT, pak Linux

Pokud máte *pouze jeden* IDE disk

- Nejprve vám doporučuji nainstalovat nově Windows NT 4.0. Předpokládám, že důležitá data už zálohovaná máte, takže s instalací by neměl být problém. Při instalaci NT se vás instalační program nebude ptát, kam má umístit zavaděč NT a umístí jej do MBR (Master Boot Record, hlavní bootovací záznam) vašeho disku. Existuje možnost, že v MBR zůstanou nějaké údaje ze starších instalací (zejména pokud bylo instalováno i LILO), takže doporučuji *před* instalací Windows NT nabootovat systém z dosové diskety a spustit *fdisk /mbr*. Pak počítač restartujte (bez diskety) a instalujte.
- Po úspěšné instalaci NT zjistíte, že k instalaci byl použit celý disk nebo konkrétní oddíl (podle toho, co jste při instalaci zvolili). Budete tedy možná nuceni zmenšit oddíl s NT tak, abyste získali nějaké volné místo na disku. Do tohoto volného místa později budeme instalovat Linux. Po nastavení a spuštění NT nabootujte počítač z diskety s programem Partition Magic společnosti Power Quest. Je to grafický nástroj, který zobrazí všechny oddíly na všech discích. Nejlepší na něm je to, že můžete změnit vlastnosti oddílů *aniž* byste přišli o data. Jedna z možností je právě zmenšení oddílu tak, abyste na disku získali volné místo pro nějaké jiné účely. I když se doporučuje před změnou velikosti oblsti zálohovat veškerá data, raději tuto operaci provádím předtím, než v NT instaluji cokoliv dalšího (takže případná nová instalace NT není tak tragická). Nuže programem Partition Magic (nebo jiným nástrojem, který k tomuto účelu používáte), zmenšíte velikost NT oddílu (ať už NTFS nebo FAT) a umístíte jej buď na začátek nebo na konec původně použitého prostoru. (Obvykle nechávám NT na začátku disku, takže zbytek disku je „volné místo“.) Po zmenšení oblasti zkuste NT spustit a zkontrolujte, zda je všechno v pořádku.

- Jak už bylo řečeno v úvodu, *nemusí* být nutné používat nástroje jako je Partition Magic. Tento nástroj je vynikající v případě, že máte NT nainstalovány už delší dobu a nechcete je instalovat znovu, tedy v případě, že vám stávající instalace NT a dalších aplikací plně vyhovují. Nechcete je tedy celé zrušit, nicméně máte na příslušném diskovém oddílu volné místo. V takovém případě je Partition Magic nástroj přesně pro vás. Pokud ale začínáte s čistým počítačem, nebo pokud vám nevadí, že disk přeformátujete, je jednodušší začít s čistou disketou, kterou naformátujete jako bootovací a nakopírujete na ni dva dosové nástroje: FDISK a FORMAT. Pak počítač z této diskety nabootujete a zadáte příkaz *fdisk*. Objeví se různé volby, které vám umožní změnit rozdělení disku. Teď můžete na části disku vytvořit oddíl se souborovým systémem FAT (kde později nainstalujete své oblíbené NT). Zbytek disku nechte raději volný (nepokoušejte se na něm pomocí dosového FDISKu vytvořit oddíl pro Linux). Pokud byste trvali na tom, že chcete vytvořit linuxový oddíl právě teď, budete muset použít linuxovou verzi programu FDISK.
- Až sem je to jednoduché. Dalším krokem je instalace Linuxu. Pokud používáte distribuci RedHat, stačí vložit do mechaniky instalační CD a restartovat počítač (předpokládám, že u ostatních distribucí bude situace stejná nebo podobná). Dále musíte zvolit, jaký typ instalace provedete (stanice s Gnome nebo KDE, vlastní instalace a podobně). Zvolte co chcete, doporučuji ale instalovat pracovní stanici. Je to výhodné, protože instalační proces automaticky najde volné místo na (prvním) pevném disku, vytvoří všechny oddíly potřebné pro Linux, správně je naformátuje a standardně nastaví většinu věcí tak, že se při instalaci nebudete příliš trápit. (Později můžete doinstalovat chybějící komponenty, nebo provést novou vlastní instalaci přes existující linuxové oddíly.) LILO je nutné zavést do MBR.
- **Nezapomeňte vytvořit spouštěcí disketu. Nikdy nevíte, kdy ji budete potřebovat. Pokud by došlo k nějakému poškození MBR a vy nebudete mít spouštěcí disketu, linuxové oddíly budou nedostupné a budete nuceni systém nově nainstalovat**.
- Po skončení instalace Linuxu restartujte počítač. Zjistíte, že LILO nabízí pouze jedinou volbu – spuštění Linuxu (nebo možná více voleb pro spuštění Linuxu například v případě, že máte víceprocesorový počítač, nebo něco podobného). Nelekejte se! Windows NT jsou stále v počítači tam, kde jste je instalovali před Linuxem. Teď se musíte s Linuxem co nejrychleji spřátelit, abyste byli schopni najít a upravit soubor */etc/lilo.conf*. Když jej otevřete poprvé, zjistíte, že je v něm jediný (nebo více) záznam pro spuštění Linuxu. Měli byste znát přesné místo (tedy oddíl), kde jsou nainstalovány Windows NT, takže byste měli být schopni do souboru */etc/lilo.conf* dopsat příslušný záznam. Po provedení změn spusťte příkazem */sbin/lilo* LILO a po restartu počítače byste měli v nabídce vidět možnost nabootovat jak Linux, tak NT (nebo DOS nebo něco).
- Já jsem kvůli instalaci NT přidával tyto údaje:

```
other=/dev/hda1
label=nt
```

Celý soubor */etc/lilo.conf* tedy vypadá takto:

```
boot=/dev/hda
timeout=50
prompt
    default=linux
    vga=normal
    read-only
image=/boot/vmlinuz-2.2.12-20
```

```
    label=linux
    root=/dev/hda3
other=/dev/hda1
    label=nt
```

- Ještě trochu vysvětlení k obsahu mého souboru */etc/lilo.conf*. Po instalaci Windows NT je jejich oddílu přiřazeno označení C:. Kromě toho jsem si vytvořil ještě jednu NTFS oblast pro případ, abych mohl uložit a zálohovat důležité soubory, kdyby bylo nutné NT reinstalovat. Tato partice se hlásí jako disk D: Každá má velikost přibližně 3 GB a instalátor Linuxu je našel jako zařízení */dev/hda1* a */dev/hda2*. Zbylý volný prostor měl velikost asi 2 GB, a na něm jsem vytvořil oddíl */root* o velikosti asi 1,9 GB a */swap* o velikosti přibližně 100 MB (zařízení */dev/hda3* a */dev/hda4*). LILO je umístěno v MBR a všechno fungovalo bez problémů.

Podotýkám, že jsem tehdy linuxové oddíly takto chtěl vytvořit sám. Později jsem zjistil, že to vůbec nebylo nutné a dneska už nechávám Linux, ať si na volném diskovém prostoru vytvoří oddíly sám jak chce. Věřím mu :-)

Pokud máte více než jeden (SCSI) disk

Poznámka: Budu popisovat situaci s jedním počítačem, který obsahuje několik SCSI disků. Proto jsem v závorce uvedl „SCSI“. V tom počítači je několik řadičů SCSI, SCSI CD-ROM a SCSI pásková jednotka. Neznamená to ale, že *vás* se budou věci kolem SCSI týkat. Pravděpodobnější bude situace, kdy v jednom počítači máte více IDE disků, nicméně instalační postup se tím prakticky nezmění.

- Nejprve vám doporučuji nainstalovat nově Windows NT 4.0 na **první** pevný disk. Předpokládám, že důležitá data už zálohovaná máte, takže s instalací by neměl být problém. Klidně můžete data zálohovat na druhý disk, nebo něco podobného. Při instalaci NT se vás instalační program nebude ptát, kam má umístit zavaděč NT a umístí jej do MBR (Master Boot Record, hlavní bootovací záznam) **prvního** disku. Existuje možnost, že v MBR zůstanou nějaké údaje ze starších instalací (zejména pokud bylo instalováno i LILO), takže doporučuji *před* instalací Windows NT nabootovat systém z dosové diskety a spustit *fdisk /mbr*. Pak počítač restartujte (bez diskety) a instalujte. Pokud si chcete být jisti, že počítač je „čistý“, můžete také v průběhu instalace smazat oddíly na ostatních discích – ovšem pouze v případě, že jste je **nepoužili na zálohování dat!**
- Po úspěšné instalaci NT zjistíte, že k instalaci byl použit celý disk nebo konkrétní oddíl (podle toho, co jste při instalaci zvolili). Budete tedy možná nuceni zmenšit oddíl s NT tak, abyste získali nějaké volné místo na disku. Do tohoto volného místa později budeme instalovat Linux. Můžete také zvolit variantu (tak jako já), že Linux nainstalujete na další disk(y). V takovém případě při instalaci Linuxu použijete k instalaci disk */dev/sdb* (nebo *sdc* nebo *sdd*, pokud máte SCSI systém, nebo */dev/hdb* (nebo *hdc* nebo *hdd*, pokud máte IDE systém). I když při instalaci bylo možné zvolit cokoliv, co jsem chtěl, ukázalo se, že po dokončení nebylo LILO schopno nabootovat vůbec nic. Vždy se zastavilo po vypsání „LI“ a dál nešlo nic dělat. Po nějaké době experimentování jsem se nakonec rozhodl nainstalovat všechno na **první** disk. Po nastavení a spuštění NT jsem zavedl počítač z diskety s programem Partition Magic společnosti Power Quest. Je to grafický nástroj, který zobrazí všechny oddíly na všech discích. Nejlepší na něm je to, že můžete změnit vlastnosti oddílů, *aniž* byste přišli o data. Jedna z možností je právě zmenšení oddílu tak, abyste na disku získali volné místo pro nějaké jiné účely. I když se doporučuje před změnou velikosti oblastd zálohovat veškerá data, raději tuto operaci provádím předtím, než v NT

instaluji cokoliv dalšího (takže případná nová instalace NT není tak tragická). Nuže programem Partition Magic (nebo jiným nástrojem, který k tomuto účelu používáte), zmenšíte velikost NT oddílu (ať už NTFS nebo FAT) a umístíte jej buď na začátek nebo na konec původně použitého prostoru. (Obvykle nechávám NT na začátku disku, takže zbytek disku je „volné místo". Zjistil jsem, že pokud volný necháte začátek disku, může to vést k problémům. Později se k tomuto tématu vrátíme.) Po zmenšení oblasti zkuste NT spustit a zkontrolujte, zda je všechno v pořádku.

- Jak už bylo řečeno, *nemusí* být nutné používat nástroje jako je Partition Magic. Tento nástroj je vynikající v případě, že máte NT nainstalovány už delší dobu a nechcete je instalovat znovu, tedy v případě, že vám stávající instalace NT a dalších aplikací plně vyhovují. Nechcete je tedy celé zrušit, nicméně máte na příslušném diskovém oddílu volné místo. V takovém případě je Partition Magic nástroj přesně pro vás. Pokud ale začínáte s čistým počítačem, nebo pokud vám nevadí, že disk přeformátujete, je jednodušší začít s čistou disketou, kterou naformátujete jako bootovací a nakopírujete na ni dva dosové nástroje: FDISK a FORMAT. Pak počítač z této diskety nabootujete a zadáte příkaz *fdisk*. Objeví se různé volby, které vám umožní změnit rozdělení disku. Teď můžete na části disku vytvořit oddíl se souborovým systémem FAT (kde později nainstalujete své oblíbené NT). Zbytek disku nechte raději volný (nepokoušejte se na něm pomocí dosového FDISKu vytvořit oddíl pro Linux). Pokud byste trvali na tom, že chcete vytvořit linuxový oddíl právě teď, budete muset použít linuxovou verzi programu FDISK.
- Až sem je to jednoduché. Dalším krokem je instalace Linuxu. Pokud používáte distribuci RedHat, stačí vložit do mechaniky instalační CD a restartovat počítač (předpokládám, že u ostatních distribucí bude situace stejná nebo podobná). Dále musíte zvolit, jaký typ instalace provedete (stanice s Gnome nebo KDE, vlastní instalace a podobně). Zvolte co chcete, doporučuji ale instalovat pracovní stanici. Je to výhodné, protože instalační proces automaticky najde volné místo na (prvním) pevném disku, vytvoří všechny oddíly potřebné pro Linux, správně je naformátuje a standardně nastaví většinu věcí tak, že se při instalaci nebudete příliš trápit. (Později můžete doinstalovat chybějící komponenty, nebo provést novou vlastní instalaci přes existující linuxové oddíly.) LILO je nutné zavést do MBR **prvního** disku.
- Po skončení instalace Linuxu restartujte počítač. Zjistíte, že LILO nabízí pouze jedinou volbu – spuštění Linuxu (nebo možná více voleb pro spuštění Linuxu například v případě, že máte víceprocesorový počítač, nebo něco podobného). Nelekejte se! Windows NT jsou stále v počítači tam, kde jste je instalovali před Linuxem. Teď se musíte s Linuxem co nejrychleji spřátelit, abyste byli schopni najít a upravit soubor */etc/lilo.conf*. Když jej otevřete poprvé, zjistíte, že je v něm jediný (nebo více) záznam pro spuštění Linuxu. Měli byste znát přesné místo (tedy oddíl), kde jsou nainstalovány Windows NT, takže byste měli být schopni do souboru */etc/lilo.conf* dopsat příslušný záznam. Po provedení změn spusťte příkazem */sbin/lilo* LILO a po restartu počítače byste měli v nabídce vidět možnost nabootovat jak Linux, tak NT (nebo DOS nebo něco).
- Já jsem kvůli instalaci NT přidával tyto údaje:

```
other=/dev/sda1
      label=nt
```

Celý soubor */etc/lilo.conf* tedy vypadá takto:

```
      boot=/dev/sda
      map=/boot/map
```

```
install=/boot/boot.b
prompt
timeout=50
default=linux
image=/boot/vmlinuz-2.2.12-20smp
    label=linux-mp
    initrd=/boot/initrd-2.2.12-20smp.img
    read-only
    root=/dev/sda6
image=/boot/vmlinuz-2.2.12-20
    label=linux-up
    initrd=/boot/initrd-2.2.12-20.img
    read-only
    root=/dev/sda6
other=/dev/hda1
    label=nt
```

- Ještě trochu vysvětlení k obsahu mého souboru */etc/lilo.conf*. Po instalaci Windows NT na **první** disk jsem jejich oddílu přiřadil označení C:. Po vytvoření dostatečného volného místa *za* NTFS oddílem jsem nechal instalátor Linuxu, aby si volné místo na disku rozdělil sám. I když by Linux měl podporovat až čtyři primární oblasti na jednom disku, ukázalo se, že pokud instalátor na disku nalezne existující primární oblast, vytvoří raději rozšířenou oblast. Předpokládáme-li, že primární oblasti (včetně rozšířených) jsou číslovány */dev/sda1* až */dev/sda4*, pak by měla být rozšířená oblast označena jako */dev/sda4*. Ve světle této logiky pak bude první logický disk na rozšířené oblasti označen jako */dev/sda5* (v mém případě to byl oddíl */boot*, umístěný fyzicky na úplném začátku prvního pevného disku). Oddíly */root* a */swap* pak byly */dev/sda6* a */dev/sda7*. Můžete to poznat podle údaje „root=-/dev/sda6“. LILO bylo opět uloženo do MBR a vše fungovalo bez potíží.
- Můžete se nyní zeptat: Co dělat v případě, že mám v počítači více disků? Dobrá otázka. Já mám v počítači *čtyři* SCSI disky a samozřejmě jsem zkoušel Linux instalovat na druhý, třetí a nakonec i na čtvrtý disk. Ať už jsem udělal cokoliv (instalátoru to nikdy nevadilo), po závěrečném restartu se LILO vždycky zastavilo po vypsání „LI“ a to bylo všechno. Nakonec jsem došel k závěru, že problém se všemi těmi disky je v tom, že jsou příliš „daleko“ od MBR, který je na začátku **prvního** disku. Proto jsem doporučoval, ať všechno (NT i Linux) nainstalujete na **první** disk. Jakmile vám oba systémy poběží, není problém v nich zpřístupnit i ostatní disky. Předpokládám, že se vám všechno na první disk vejde. (NT ke své instalaci potřebují alespoň 150 MB, u Linuxu záleží, co instalujete: Pracovní stanice s Gnome nebo KDE potřebují asi 580 MB, úplná instalace kolem 1,4 GB. Disk o velikosti 2,4 GB by tedy měl bohatě stačit.)
- Dále se můžete zeptat: Co kdybych zmenšil NT oblast tak, aby volné místo zůstalo na začátku disku? Zkoušel jsem to. Nejprve jsem nainstaloval pracovní stanici s Gnome (395 balíčků, 570 MB). Při instalaci nejprve vše vypadalo v pořádku, ale nepodařilo se ani nainstalovat LILO, ani vytvořit spouštěcí disketu. Jak asi tušíte, Linux nefungoval. Pak jsem zkoušel instalovat pracovní stanici s KDE (377 balíčků, 582 MB). I teď všechno vypadalo v pořádku, ale opět se nepodařilo nainstalovat LILO ani vytvořit disketu. Pak jsem zjišťoval, jak vlastně vypadá struktura oblastí na disku. Překvapilo mě, že nově vytvořené logické oblasti (na nově vytvořené rozšířené oblasti) byly číslovány tak, jako by byly fyzicky *za* oblastí s NT. Jinak řečeno, dostal jsem zajímavé pořadí oblastí: */dev/sda5*, */dev/sda6*, */dev/sda7* a nakonec */dev/sda1*. To zřejmě systém zmátlo. Proto jsem doporučoval, ať volné místo vytvoříte *za* oblastí s NT.

- Všimněte si dvou spouštěcích voleb Linuxu (liší se v „smp“). Jedná se o server s podporou více procesorů. Instalační proces tento hardware správně detekoval a nainstaloval systém pro jeden procesor i pro více procesorů. Dodnes je tam ovšem fyzicky pouze jeden procesor.

Přechod z Windows NT na Windows 2000

Abych řekl pravdu, nejde o opravdový „přechod“ stávající instalace NT na 2000, ale o novou instalaci Windows 2000 Professional. Předpokládám, že tento postup bude fungovat i pro jiné produkty Windows 2000. Nezkoušel jsem to s instalací serveru, ale pokud si vzpomínám, nikdy jsem s NT neměl žádné problémy, ať už šlo o server nebo pracovní stanici.

- Jako první je třeba provést zálohu všech důležitých dat! Zálohoval jsem je na samostatný diskový oddíl, kam jsem prostě všechny potřebné dokumenty zkopíroval. S tímto oddílem jsem pak při instalaci nemanipuloval. Po pořízení záloh můžeme začít s instalací.
- Vždycky při (re)instalaci Windows je nejraději instaluji na čistý diskový oddíl. Proto jsem odstranil všechny stávající oddíly systému NT a na vzniklém volném místě jsem vytvořil nový oddíl typu NTFS. Při instalaci jsem pak zvolil umístění na tento nový oddíl.
- V průběhu instalace dojde k odstranění LILO z MBR, takže při několika instalačních restartech se neobjeví volby LILO.
- Po dokončení instalace systému můžete nainstalovat oblíbené aplikace a obnovit zálohovaná data. Zkontrolujte, zda nová instalace Windows funguje tak, jak má.
- Až sem to bylo jednoduché. Teď musíte najít spouštěcí disketu Linuxu. Pokud ji nenajdete, je to smutné. Nejsem si vůbec jistý, zda je možné Linux bez této diskety oživit. Někteří odborníci tvrdí, že Linux můžete zavést z instalačního CD, ovšem nevím, co dělat, pokud váš počítač neumožňuje bootování z CD-ROM mechaniky... Nějaké nápady?
- Po úspěšném spuštění Linuxu budeme opět upravovat soubor */etc/lilo.conf*. Měly by v něm být minimálně dvě volby: Linux a NT. Pokud tam volba pro spuštění vašich oblíbených Windows chybí, přidejte ji. Postup byl popsán v předchozích částech. Pak musíte spustit */sbin/lilo*, čímž se zavaděč LILO zapíše do MBR.
- A to je všechno. Při dalším startu počítače se objeví nabídka, který systém chcete spustit.
- Z vlastní zkušenosti – všechno funguje skvěle. I když jsem musel odstranit předchozí instalaci WinNT kvůli instalaci Win2000 a (dočasně) jsem přišel o LILO, na konci bylo všechno tak, jak je potřeba – Windows 2000 a Linux spolupracující stejně jako předtím Windows NT a Linux!

Jak instalovat Windows 2000 vedle Linuxu a Windows 98

Poznámka: Když říkám *vedle Linuxu a Windows 98*, myslím tím, že Linux a Windows 98 **už jsou** na počítači nainstalovány a spolupracují. Nainstalovali jsme je dříve, ještě před rozhodnutím instalovat i Windows 2000. Na jednom z počítačů v práci máme Linux i Win98, oba systémy se spouštějí přes LILO.

V zásadě není problém nainstalovat Windows 2000 na počítač, kde už jsou nějaké Windows nainstalovány. V mém případě šlo o Windows 98, takže stačilo pouze vložit do mechaniky CD s Windows 2000. Instalátor nalezl „starší" verzi Windows a nabídl mi buď aktualizaci této verze, nebo novou čistou instalaci. Nejprve jsem vyzkoušel aktualizovat Windows 98 na W2000, protože jsem chtěl dostat podobnou konfiguraci, jakou mám doma (a kterou jsem popisoval výše). Instalace však detekovala problémy s kompatibilitou hardwaru a softwaru a některé komponenty by po aktualizaci nemusely pracovat správně. Proto jsem zvolil samostatnou instalaci vedle Windows 98 a Linuxu. Naštěstí jsem krátce předtím do počítače instaloval nový pevný disk, takže jsem se nemusel trápit s prvním diskem, na kterém byly nainstalovány Windows 98 a Linux. Skutečným důvodem pro pořízení nového disku bylo hlavně to, že první už byl docela plný. Pokud bych tedy na něj chtěl instalovat třetí operační systém, musel bych jej hodně pročistit. Takže když se spustila instalace Windows 2000, nechal jsem je nainstalovat na druhý disk, který si instalátor sám rozdělil a vytvořil na něm systém NTFS. (Jen podotýkám, že Windows 98 používaly systém FAT32 a Linux ext2.)

Asi po hodině instalace Windows 2000 skončila. V průběhu instalace došlo k několika restartům. Když mělo dojít k prvnímu, byl jsem docela zvědavý, zda instalace přepsala MBR, kde bylo LILO (v předchozí kapitole jsem říkal, že Windows 2000 s největší pravděpodobností MBR přepíší). Zajímavé je, že tentokrát k tomu nedošlo a na obrazovce se objevila výzva „LILO boot:". Objevila se ale *nová* věc – LILO začalo nabízet navíc zavaděč Windows 2000, který pak následně nabídl možnost nabootovat jak Windows 2000, tak i „Windows" – tedy staré Windows 98.

Podle toho, který systém chci spustit, pak musím volit různé postupy, ale je to pořád velmi jednoduché. Například:

- Chci-li spustit Linux, použiji k tomu LILO, kde zvolím volbu „linux" (a mám ji nastavenu v */etc/lilo.conf* jako standardní).
- Chci-li spustit Windows, vyberu v LILO druhou volbu, která spustí zavaděč Windows 2000. Pak mám možnost se rozhodnout pro W2000 nebo Windows 98. Oba systémy naběhnou bez problémů.

Jak instalovat Windows NT/2000 a Linux na laptopu

Pokud chcete mít současně NT a 2000 a Linux

Před nějakým časem jsme v práci pořídili nové notebooky HP Omnibook 6000, tak jsem se hned vrhl na nové hračky. Šlo o modely s Pentiem 3 na 1 GHz a 128 MB RAM. Kromě toho obsahovaly 30GB disk, výměnné DVD a FDD mechaniky, sekundární baterii do šachty pro mechaniky a kombinovanou síťovou a modemovou kartu.

- Notebooky jsme koupili s předinstalovaným systémem Windows 2000 Professional a těšil jsem se, jak disk přeformátuji a začnu od začátku. Měl jsem v plánu opět použít Partition Magic společnosti Power Quest. Nakonec jsem všechny stávající oblasti smazal, protože jsem si všiml jedné malé oblasti hned zkraje disku, která mi byla podezřelá (později jsem zjistil, že sloužila k hibernaci a diagnostickým účelům).
- Následně jsem zkusil zopakovat proceduru, na kterou jsem byl vždycky velice pyšný: vytvořit dvojici FAT oblastí (zhruba 2 037 MB), kam přijdou nainstalovat NT a 2000. Neptejte se mě, proč používám starý formát FAT – je to čistě kvůli tomu, aby tyto oblasti byly přístupné všem systémům Windows a abych mohl v případě potřeby sdílet soubory s jakýmikoliv jinými operačními systémy. Kromě těchto oblastí (které samozřejmě začínají na začátku disku), jsem vytvořil ještě další FAT oblasti, tentokrát ale na *konci* disku. Partition Magic to dovoluje (teď by mi měli hoši z Power Questu poslat pár peněz za reklamu na jejich produkt).

  Možná vás zajímá, proč si někdo nechává volný prostor uprostřed disku. Odpověď je, aby mohla být */boot* oblast Linuxu umístěna pod známým limitem 1 024 cylindrů. Pokud její velikost tento limit překročí, tak to už nevadí, důležitý je začátek. Tím pádem můžete ve vzniklé volné oblasti vytvořit linuxovou oblast */boot* (bez ohledu na její velikost), a zbytek můžete použít pro sdílenou (FAT) oblast mezi Windows a Linuxem.

  Zatím je všechno v pořádku. Teď vložíme instalační disk Windows NT. (Použijte „opravdový" instalační disk, *ne* ten, který byl dodán s notebookem, protože jeho účelem je „obnovit" instalaci včetně hibernační oblasti. Proč nám tato oblast vadí, to uvidíme později.)
- Po skončení instalace NT je čas vložit instalační CD Windows 2000 (opět použijte „opravdový" instalační disk, ne to, co jste dostali k počítači). Ze spuštěných NT jsem tedy provedl instalaci 2000 na druhou oblast (samozřejmě, že v prostředí NT se spustil pouze začátek instalačního procesu, ale to je v pořádku). Po skončení instalace máme na počítači dvoje Windows – každé na jedné oblasti.
- Všechno vypadá být v pořádku, kromě diagnostiky. Měla by se aktivovat stiskem F10 při startu počítače, **pokud** bychom použili originální dodané CD. Mimochodem, „obnovená" instalace skončí s ještě větší oblastí (asi 500 MB). A co víc, tato oblast bude *primární*. Při instalaci více operačních systémů by vám takto rychle došly primární oblasti. Já jsem se nicméně rozhodl, že diagnostické nástroje nepotřebuji a tak stisk F10 při startu počítače už nic nedělá :-).
- Abych nezapomněl, zavaděč Windows 2000 by měl při dodržení výše popsaného postupu nabízet možnost spuštění jak NT, tak 2000. (Pokud nechcete mít na počítači i NT i 2000, můžete klidně přejít k další části.)
- V této chvíli je pravý čas poohlédnout se po instalačním CD Linuxu. Vložíme je do DVD (nebo CD) mechaniky a restartujeme systém. Několik sekund po zapnutí systému bude třeba jednou nebo dvakrát zmáčknout Escape a změnit pořadí bootovacích zařízení (potřebujeme samozřejmě, aby systém bootoval z CD/DVD).
- Zbytek instalace Linuxu proběhne jako obvykle. Nezapomeňte vytvořit spouštěcí disketu, nikdy nevíte, kdy může být užitečná.
- A to by mělo být všechno. Při dalším startu systému by se měl spustit Linux. Nenechte se překvapit, že se neobjeví zavaděč Windows. Pomůže vám další skvělý dokument, LILO mini-HOWTO. Tam se dozvíte, jak funguje zavaděč Linuxu LILO (Linux Loader).

- Pokud vám to přijde jednodušší, podívejte se o pár kapitol dopředu na *Přechod z Windows NT na Windows 2000*, kde se dozvíte, jak zavádění systému nastavit.

Pokud chcete mít pouze Windows 2000 a Linux

Výše popsaný postup vedl k vytvoření počítače s více (konkrétně se třemi) operačními systémy. Pokud chci spustit Linux, vyberu jej v nabídce LILO. Chci-li spustit některé Windows, vyberu v nabídce LILO volbu Windows, čímž se spustí zavaděč Windows 2000. V něm si pak zvolím, zda chci spustit NT nebo 2000.

Kvůli nějakým jiným věcem jsem potřeboval do obou Windows nainstalovat Norton System Works. Dopadlo to dobře, ale později jsem zjevně zadal nějaký příkaz, který vedl k závažné změně v konfiguraci. Výsledkem bylo, že ani jedny Windows nebylo možné spustit. Což byla správná chvíle na reinstalaci :-)

- Tentokrát jsem se rozhodl použít „záchranné“ CD, dodané s notebookem. Musím říct, že se mi strašně líbilo modré logo „HP Invent“ na pozadí, stejně jako text „Manufactured by Hewlett Packard“ ve Vlastnostech Mého počítače (teď čekám pár peněz za reklamu od HP).
- Použitím „záchranného“ CD si ušetříte čas při instalaci NT/2000, protože dojde k replikaci – nebo, chcete-li „naklonování“ – celého obrazu pevného disku a proces je velmi rychlý. Navíc dojde k vytvoření hibernační/diagnostické oblasti. A konečně, pouze z tohoto CD získáte některé nástroje od HP.
- Po pěti až šesti minutách jsou NT (nebo 2000) připraveny k práci. Podobně jako při „opravdové“ instalaci (popsané v předchozí části) je nyní čas vložit instalační CD Linuxu.
- Zbytek instalace Linuxu proběhne jako obvykle. Nezapomeňte vytvořit spouštěcí disketu, nikdy nevíte, kdy může být užitečná.
- A to by mělo být všechno. Při dalším startu systému by se měl spustit Linux. Nenechte se překvapit, že se neobjeví zavaděč Windows. Pomůže vám další skvělý dokument, LILO mini-HOWTO. Tam se dozvíte, jak funguje zavaděč Linuxu LILO (Linux Loader).
- Pokud vám to přijde jednodušší, podívejte se o pár kapitol dopředu na *Přechod z Windows NT na Windows 2000*, kde se dozvíte, jak zaváděné systémy nastavit.
- Určitě se mnou budete souhlasit, že život by byl daleko jednodušší, kdyby neexistovaly různé potíže, jako je třeba „tolerance“ mezi světem Linuxu a Windows. Možná jsem při instalaci někde opět udělal nějakou chybu, nicméně záhy po skončení instalace Linuxu – konkrétně RedHat 7.1 (máme 4. května 2002 a pořád nemám žádnou lepší distribuci... Nějaký dárek od laskavých čtenářů?) – jsem zjistil, že stisk F10 při startu počítače už nespustí diagnostické nástroje HP. Bez ohledu na stisk F10 se objevuje rovnou nabídka LILO. *Před* instalací Linuxu fungovala diagnostika bez potíží.
- Při příštím spuštění Windows 2000 ukázal jejich Správce disku následující rozdělení:

	Velikost	Formát	Název	Typ
1	15 MB	FAT	-	Primary
2	7.30 GB	FAT32	HPNOTEBOOK C:	Primary
3	52 MB	-	-	Primary
4	18.37 GB	-	-	Logical
5	258 MB	-	-	Logical
6	1.96 GB	volné místo	-	-

- Partition Magic to ale viděl takto:

	Velikost v MB	Formát	Název	Typ
1	14,7	FAT	save to disk	Primary
2	7471,4	FAT32	HPNOTEBOOK	Primary
3	51,7	ext2	/boot	Primary
4	21077,9	extended	-	Primary
5	18811,4	ext2	/	Logical
6	258,4	swap	-	Logical
7	2008,1	volné místo	-	Logical

- Vysvětlení k těmto tabulkám: Diagnostická oblast, vytvořená „záchranným“ diskem Windows 2000, je poměrně malá – cca 15 MB (v porovnání s touž oblastí vytvořenou při obnově Windows NT, kdy má velikost asi 500 MB). Bez ohledu na velikost je to *primární* oblast. Je tedy třeba dávat pozor, kolik je vlastně v systému primárních oblastí. Nejsem si sice jistý, ale právě to může být důvod, proč po skončení instalace všech operačních systémů nejsou diagnostické nástroje dostupné.

  Dále můžete vidět, že velikost FAT32 oblast s Windows 2000 je zmenšena na cca 7,5 GB, abychom získali místo pro instalaci Linuxu. Doporučuji, abyste využití vzniklého volného místa ponechali na instalaci Linuxu, není nutné vytvářet oblasti ručně. Jak vidíte v tabulce, zhruba 50 MB má zaváděcí disková oblast, asi dvojnásobek fyzické paměti (2 x 128 = 256 MB) je swapovací oblast a zbytek jsou ostatní části Linuxu. Zbylé volné místo jsou zřejmě nevyužitelné pozůstatky po konverzi různých souborových systémů.

- Současný stav tedy je: Linux funguje, Windows fungují. Diagnostika nefunguje. Hibernaci (ve Windows) jsem zatím nezkoušel. Kombinovaná 3Com síťová/modemová karta funguje v Linuxu jen částečně – síťová karta funguje, ale modem se tváří jako „winmodem“. Nikdo neví, zda s tím HP hodlá něco dělat. Jediné, co můžu říct, je, že pokud modem potřebujete, tak se notebooku HP Omnibook 6000 vyhněte. Anebo můžete samozřejmě utratit nějaké peníze navíc a koupit si podporovanou modemovou kartu. 15“ obrazovka nabízí skvělý obraz v rozlišení až 1 400 x 1 050 a 16bitové barevné hloubce. Zajímavé je, že toto rozlišení se ve Windows nastaví *standardně*, v Linuxu jej lze bez potíží dosáhnout také. Touchpad je poněkud přecitlivělý, a zbytečně často detekuje „klepnutí“. Občas mi to vadí, ale vy můžete mít jinou zkušenost.

Další plán je provést konverzi FAT32 oblasti na NTFS kvůli zvýšení spolehlivosti Windows. Dále bude potřeba zmenšit velikost linuxové oblasti a získat tak něco volného místa, které bude sloužit ke sdílení souborů mezi oběma systémy (bude tam pravděpodobně systém FAT).

Soužití Linuxu a Win9x+ pomocí zavaděče Grub

Originál: http://tldp.org/HOWTO/Linux+Win9x+Grub-HOWTO/

Tento dokument popisuje, jak pomocí zavaděče GRUB nastavit počítač tak, aby umožnil spouštění Windows i Linuxu bez poškození existující instalace Linuxu.

Úvod

Proč tento způsob použít

Jsem velký fanoušek věcí, které nepocházejí od Microsoftu, takže když jsem si pořizoval nový počítač, nechal jsem si na něj nainstalovat Linux. Je to skvělý počítač, jenže mi na něm chyběly mé oblíbené hry z Windows, a taky GIMP není úplně to, co by mi stačilo. Takže jsem se nakonec rozhodl, že budu mít na počítači oba systémy. Když jsem se snažil zjistit, jak to udělat, našel jsem pouze návod, jak nainstalovat Linux na počítači, na němž už jsou nainstalovány Windows, a jak je spouštět programem *loadlin*. Bohužel se ale *loadlin* spouští z DOSu, takže celá tato metoda vyžaduje, aby byly Windows nainstalovány na první oblasti prvního pevného disku, poněvadž Windows jsou přesvědčeny, že jsou středem vesmíru a všechny ostatní operační systémy musí obíhat kolem. To je ovšem problém, pokud už máte na této oblasti nainstalován Linux a nechcete kvůli instalaci Windows celý systém rušit.

A to je chvíle, kdy vstupuje do hry GNU GRUB – GRand Unified Bootloader. Od zavaděčů jako je LILO se liší v tom, že umí Windows lhát a přesvědčí je, že jsou nainstalovány na první oblasti i v případě, kdy to není pravda. Takže si můžete Linux nechat nainstalovaný tam kde je, a Windows doinstalovat někam na kraj.

Požadavky

Co budete potřebovat

- Počítač s funkční instalací Linuxu

- Dostatek místa na disku pro oblast Windows
- Editor diskových oblastí, jako je například GNU Parted, http://www.gnu.org/software/parted
- RPM balíček nebo zdrojové soubory GRUB
- Spouštěcí disketu DOSu/Windows s podporou CD-ROM a instalační CD Windows 95, nebo spouštěcí instalační CD Windows 9x.
- (*doporučeno*) Spouštěcí disk Linuxu.

Postup

Vytvoření oblasti pro Windows

Pokud máte na disku volný oddíl pro instalaci Windows, můžete tuto část přeskočit. V opačném případě budete muset spustit editor diskových oblastí, defragmentovat disk a vytvořit nový oddíl pro Windows. Já jsem k instalaci použil GNU Parted, stejně dobře by posloužil sfdisk. *Nepoužívejte* fdisk, přišli byste o data. Upozorňujeme, že úspěch následujícího postupu není zaručen, takže si pro jistotu zálohujte všechna důležitá data.

Takto vypadala tabulka diskových oblastí mého disku před zahájením instalace:

```
Device      Start       End         System
/dev/hda1    1          3           Linux
/dev/hda2    4          1222        Extended
/dev/hda5    4          36          Linux swap
/dev/hda6    37         1222        Linux
```

/dev/hda1 je bootovací oddíl Linuxu (v Linuxu můžete mít samostatný bootovací oddíl), /dev/hda2 je rozšířený oddíl, který obsahuje dvě logické jednotky, */dev/hda5 a* /dev/hda6, které obsahují swapovací oddíl a instalační oddíl.

Jako superuživatel spusťte editor diskových oblastí. Zmenšete velikost linuxového oddílu podle vlastní úvahy (nemůžete jej zmenšit pod velikost dat, která již na disku jsou). Program bude nějaký čas defragmentovat disk a pak změní tabulku diskových oblastí. Dále odpovídajícím způsobem zmenšete velikost rozšířeného oddílu, na němž máte umístěn linuxový oddíl. Teď byste měli mít dostatek místa, abyste na něm mohli vytvořit nový oddíl pro instalaci Windows. Vytvořte tento oddíl se souborovým systémem Win95 FAT32 (LBA). Teď by měla tabulka diskových oblastí vypadat nějak takto:

```
Device       Start          End      System
/dev/hda1     1             3        Linux
/dev/hda2     4             905      Extended
/dev/hda3     906           1222      Win95 FAT32 (LBA)
/dev/hda5     4             36       Linux swap
/dev/hda6     37            905      Linux
```

Vřele vám doporučuji v tomto okamžiku spustit fdisk a zapsat si na papír všechny informace o rozdělení disku. Pokud uděláte při další instalaci chybu, Windows nesmírně ochotně přepíší tabulku diskových oblastí tak, jak má podle nich vypadat, a v takovém případě se vám bude hodit údaj o tom, jak disk opravdu vypadal.

Instalace GRUB

Instalace

Měli byste mít na disku buď RPM balíček nebo zdrojový balík programu GRUB. Nainstalujte jej způsobem, jenž odpovídá vašemu systému. V tomto okamžiku budete mít GRUB nainstalován, nicméně nebude aktivní. Jako superuživatel spusťte:

```
# /sbin/grub-install /dev/hda
```

Tím se GRUB nahraje do hlavního bootovacího záznamu vašeho disku. Dále musíte upravit soubor */boot/grub/grub.conf* a nastavit bootovací nabídku programu GRUB.

Editace souboru grub.connf

```
default=0
timeout=10 splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.7-10)
        root (hd0,0)
        kernel /vmlinuz-2.4.7-10 ro root=/dev/hda6
        initrd /initrd-2.4.7-10.img

title Windows 98
        map (hd0,0) (hd0,2)
        map (hd0,2) (hd0,0)
        rootnoverify (hd0,2)
        chainloader +1

title DOS Boot Disk
        map (hd0,0) (hd0,2)
        map (hd0,2) (hd0,0)
        chainloader (fd0)+1
```

Podívejme se, co jednotlivé údaje znamenají.

„default=0“ a „timeout=10“ znamenají, že pokud na nic nebudete sahat, po deseti sekundách se automaticky spustí Linux. Údaj „splashimage“ mluví sám za sebe a není příliš důležitý.

Údaj „title“ uvozuje každou bootovací volbu a definuje text, který se objeví v nabídce při spouštění systému. Údaj „root“ říká, na kterém oddílu je uložen obraz jádra Linuxu (může a nemusí to být kořenový oddíl). Údaj „root (hd0,0)“ tedy programu GRUB říká, že jádro je na prvním oddílu disku */dev/hda*. Můžete si všimnout, že GRUB používá zábavné číslování – 0 – 3 jsou primární oddíly, 4 a výše logické oddíly. Další řádek říká, kde najít jádro a kde je umístěn kořenový oddíl. Údaj „initrd“ říká, kde se nachází obraz inicializačního RAM disku. Tyto údaje budou poněkud odlišné, pokud nepoužíváte samostatný bootovací oddíl. Další informace zjistíte na manuálových stránkách programu GRUB.

Položky „map“ v části Windows 98 jsou rozhodující pro správnou funkci celé instalace. Jde o magické údaje, které donutí Windows věřit, že se nacházejí na první oblasti prvního disku. Pokud neprovedete namapování oddílu s Windows na (hd0,0), Windows vám zničí tabulku diskových oblastí a nepodaří se vám nabootovat nic.

Údaj „rootnoverify“ říká programu, že má bootovat z oddílu s Windows, ale nemá se jej pokoušet namountovat, a údaj „chainloader +1“ říká, že zavádění systému má pokračovat zavaděčem Windows.

Část „DOS boot disk“ je nezbytná, pokud budete chtít při instalaci Windows bootovat z diskety, nicméně může se hodit i kdykoliv jindy po instalaci, když budete chtít z jakéhokoliv důvodu nabootovat z diskety. Pokud budete instalovat z bootovacího CD, musíte si vytvořit položku, v níž předáte zavádění systému na zařízení odpovídající vaší CD mechanice.

Uložte soubor *grub.conf* a ukončete editor. Na rozdíl od programu LILO nevyžaduje GRUB po úpravě konfigurace spuštění žádného programu. Připravte si k ruce záchrannou disketu Linuxu a zkuste restartovat počítač. Vzápětí po startu byste měli vidět nabídku programu GRUB se třemi možnostmi. Zvolte Linux. Při troše štěstí se Linux obvyklým způsobem spustí, což znamená, že jste GRUB nainstalovali správně. Pokud se to nepovede, tak přijde ke slovu zmíněná záchranná disketa, z níž nabootujete Linux a opravíte soubor *grub.conf*. *Až se vám podaří nakonfigurovat GRUB tak, aby Linux správně bootoval, můžete přejít k instalaci Windows.*

Instalace Windows

Vyjměte z mechanik všechny diskety a CD a restartujte počítač. *Nebootujte počítač přímo z instalačního média!* BIOS by totiž spustil instalaci Windows přímo z tohoto média a GRUB by neměl šanci uplatnit potřebné přemapování. Až se objeví nabídka GRUBu, vložte bootovací disketu nebo CD a zvolte odpovídající položku z nabídky. Instalátor by měl oblast, kterou jste pro něj vytvořili, vidět jako disk C:. Vesele pokračujte a nainstalujte Windows.

A je to. Váš počítač nyní umožňuje práci s oběma operačními systémy.

Tisk na Windows

Originál: http://tldp.org/HOWTO/mini/Print2Win/

Tento dokument uvádí obecné informace jak z Linuxu tisknout na sdílené tiskárně Windows.

Úvod

Protože chceme nastavit tiskovou službu tak, aby fungovala, budeme celé nastavování provádět postupně krok za krokem, abychom se vyvarovali chyb nebo špatných nastavení.

Nejprve nastavíme server, a potom budeme nastavovat klienty.

Server (Windows)

Pro konfiguraci serveru není nutné uvádět žádná zvláštní pravidla. Nainstalujte ve Windows ovladač tiskárny, otestujte, zda tiskárna správně funguje, a pak povolte sdílení tiskárny.

Klient (Linux)

Konfigurace klienta v Linuxu je podobná jako při tisku z Linuxu na Linux.

Standardní lpr

Nejjednodušší metoda je přidat záznam do souboru */etc/printcap*. Jako jednoduchý příklad pro obecnou jehličkovou tiskárnu uvádíme:

```
# EPSON LX300
epson:\
:sd=/var/spool/lpd/epson:\
:mx#0:\
:sh:\
:if=/var/spool/lpd/epson/filter:\
:lp=/dev/null:
```

Zkontrolujte, že používáte správný filtr, nebo použijte univerzální filtr.

Vytvořte adresář */var/spool/lpd/epson* a uživatelům, kteří mají na tiskárně tisknout, nastavte práva pro přístup do tohoto adresáře.

Předpokládejme, že server Windows se jmenuje *meriadoc*, má IP adresu 192.168.1.49, a že tiskárna je sdílena pod názvem *epsonLX*.

Pak musíte soubor */var/spool/lpd/epson/.config* nastavit takto:

```
share='\\meriadoc\epsonLX'
hostip=192.168.1.49
```

Kde:

- share = „\\ *název_serveru_windows*\ *název_sdílené_tiskárny*“
- hostip = *ip_adresa_serveru_windows*

Jakmile máte soubor */etc/printcap* správně nastaven, zbývá povolit sdílení tiskárny:

```
[localhost]$ lpc up epson
[localhost]$ lpc enable epson
```

Pokud je všechno v pořádku, můžete tisknout:

```
[localhost]$ lpr -Pepson soubor
```

Tipy

Ve výše uvedeném příkladu souboru */etc/printcap* jsem použil následující záznam, který ale obecně není rozumný:

```
...
:lp=/dev/null:\
```

Tiskový démon *lpr* totiž soubor specifikovaný parametrem *lp*= otevírá „exkluzivně“. Dělá to proto, aby se zabránilo více procesům současně tisknout na stejné tiskárně.

Vedlejší efekt použitého nastavení tedy bude, že žádné z tiskáren v systému nebudou moci tisknout současně (to většinou nevadí, protože tiskárny tisknou rychle a pomocí front, takže si toho nejspíš ani nevšimnete), ale navíc *jakýkoliv* proces pokoušející se o zápis do */dev/null* skončí chybou!

Na jednouživatelském systému to moc nevadí. Pokud ale máte systém se spoustou uživatelů a tiskáren, pak by to problém byl.

Řešením je vytvoření falešných souborů pro každou zvlášť, např. *touch /dev/eng*:

```
[localhost]$ touch /dev/eng
[localhost]$ touch /dev/colour
```

Winmodemy a Linux

Originál: http://tldp.org/HOWTO/Linmodem-HOWTO.html

Tento dokument by měl pomoci majitelům winmodemů zprovoznit je pod Linuxem.

Co jsou Winmodemy?

Jsou to modemy...

Winmodem se, stejně jako ostatní modemy, používá pro telefonický přístup ke službám jako jsou BBS, Internet, VoicePhone, Fax a podobně. Je připojen k telefonní lince a jeho důležitou charakteristikou je jeho rychlost. Pokud se chcete dozvědět o modemech více, doporučuji praktický návod „Modemy“ (kapitola 11).s

...ale ne jako opravdové!

Jsou to ale WINmodemy. Znamená to, že ke své práci potřebují Windows. Proč? Jednoduše proto, poněvadž jsou hloupé. Potřebují speciální software, ovladač, aby byly schopné plnit svou funkci. Když mluvíme o softwaru, musíme zmínit i operační systém – a ovladače dodávané s těmito modemy jsou z 99 % určeny pouze pro Windows. Někteří výrobci, například LT nebo Motorola, se rozhodli vytvořit i linuxové verze ovladačů. Nepochopili však filozofii Linuxu. Jimi dodávané ovladače samozřejmě fungují, jsou ale „Closed-Source“. Jsou zdarma, ale nejsou licencovány podle GPL. Jinak řečeno, nemáte k dispozici jejich zdrojové kódy.

Občas se někdo rozhodne vytvořit Open-Source ovladač, problém je ale s informacemi o modemech, protože výrobci neradi zveřejňují jejich specifikaci. Proto jsou Open-Source ovladače většinou ve verzích alfa nebo beta.

Jak poznám, že mám Winmodem?

Zkuste zjistit název sériového portu, k němuž je modem připojen (ve Windows a v DOSu COM1, COM2...)

V Linuxu se porty označují jako */dev/ttySx*, kde *x* je číslo sériového portu v DOSu minus 1. Tedy COM1 v DOSu je *ttyS0* v Linuxu, COM3 v DOSu je *ttyS2* v Linuxu.

Následujícími příkazy vytvořte odkaz z příslušného */dev/ttySx* na */dev/modem*:

```
rm -f /dev/modem
ln -s /dev/ttySx /dev/modem
```

Stáhněte si a nainstalujte balíček *minicom*. Spusťte jej příkazem *minicom -s*.

Zvolte „Serial Port Setup“, zadejte „A“ pro „Serial Device“, smažte celý řádek a napište */dev/modem*. Potvrďte klávesou *Enter*. Stiskněte *Esc*, zvolte „Save setup as dfl“ a pak „Exit“.

Chvíli počkejte a napište „AT“ – pokud modem odpoví „OK“, pak *nemáte* Winmodem, ale standardní modem. Pokud inicializace trvá příliš dlouho, máte Winmodem. Následující text vám možná pomůže jej zprovoznit.

Stiskem *Ctrl+A* a *X* ukončíte minicom.

ISA nebo PCI?

PCI?

Takže máme Winmodem… Jenže jsou dva typy: ISA nebo PCI. Aby to bylo složitější, tato dvě rozhraní jsou úplně odlišná. Zkusíme tedy zjistit, zda máme PCI modem nebo ISA modem. Nejprve musíme nakonfigurovat jádro. V průběhu *make *config* musíte odpovědět *yes* na následující otázky:

- V části Loadable module support: „Enable loadable module support (CONFIG_MODULES)“ a „Set version information on all modules for symbols (CONFIG_MODVERSIONS)“ (musíte mít nainstalován balík *modutils*, více viz *Kernel-HOWTO*)
- V části General setup:

  „PCI support“ (CONFIG_PCI)

  V „PCI Access Mode“ (CONFIG_PCI_GOBIOS) zvolte „Any“.

  „PCI quirks“ (CONFIG_PCI_QUIRKS)

  „Backward compatible /proc/pci“ (CONFIG_PCI_OLD_PROC)
- Pokud můžete nastavit PCI Device Name Database (v jádrech 2.4+), získáte lépe čitelný výstup v souboru */proc/pci* (CONFIG_PCI_NAMES).
- V části Filesystems
- „/proc filesystem support“ (CONFIG_PROC_FS)

Pokud jste jádro nikdy nepřekládali, můžete spoléhat na to, že novější distribuce potřebná nastavení obsahují. Chcete-li si jádro přeložit a nevíte jak, přečtěte si Kernel-HOWTO.

Dále budeme předpokládat, že máte jádro správně nastaveno.

Spusťte *cat /proc/pci*. Pokud ve výstupu najdete název svého modemu, pak máte PCI modem. Přejděte na část *Instalace ovladače*.

nebo ISA?

Pokud jste modem nenašli, možná bude ISA. Jak to zjistit?

Podpora ISA nebyla součástí starších jader, v jádrech 2.4 už jsou podporována. Pokud používáte takové jádro, přeskočte rovnou na následující část. Existuje pomocný software, *isapnp*, který s těmito zařízeními pracuje. Nejprve jej potřebujete nainstalovat. Pokud jej nemáte, stáhněte si balík *isapnptools*.

Po nainstalování balíku spusťte program *pnpdump*, který bude hledat ISA zařízení. Pak se pokusí zjistit prostředky, které tato zařízení používají, a vypíše je do souboru */etc/isapnp.conf*. Tento soubor můžete následně upravit a spustit další program, *isapnp*, který si přečte údaje v *isapnp.conf*

a provede automatickou konfiguraci připojených zařízení. Zní to složitě? Proveďte následující kroky:

1. Spusťte *pnpdump > /etc/isapnp.conf*.
2. Otevřete jej ve svém oblíbeném editoru.
3. Najděte část, kde se hovoří o vašem modemu.
4. Odkomentujte některé řádky (odstraněním znaku # na začátku řádku). Budete potřebovat:

   1 řádek *(IO 0…)*

   1 řádek *(INT 0…)*

   1 řádek *(DMA 0…)*

   1 řádek *(DMA 1…)*

   1 řádek *(IO 1…)*
5. Na všech odkomentovaných řádcích odstraňte *(CHECK)*.
6. Spusťte *isapnp /etc/isapnp.conf*. Pokud se objeví nějaké chyby, zkuste soubor *isapnp.conf* upravit a změnit nějaké parametry, orientujte se podle textu chybových hlášení. Pokud se žádné chyby neobjeví, upravte soubor tak, že v něm v části týkající se modemu odstraníte řádek *(ACT Y)*. Znovu spusťte *isapnp /etc/isapnp.conf*. Měl by vypsat *[Název_modemu] Enabled OK*.
7. Přidejte *isapnp /etc/isapnp.conf* do souboru */etc/rc.d/rc.local* příkazem *echo „isapnp /etc/isapnp.conf“ >> /etc/rc.d/rc.local*. Tím se zajistí automatická konfigurace zařízení při startu systému.

Pokud vám to pomůže, uvádím příklad svého souboru */etc/isapnp.conf*. (Soubor je okomentovaný, původně zakomentované řádky jsou uvedeny znaky ##.)

```
# PłeskoŁeno...
## (DEBUG)
(READPORT 0x0203)
(ISOLATE PRESERVE)
(IDENTIFY *)
(VERBOSITY 2)
(CONFLICT (IO FATAL)(IRQ FATAL)(DMA FATAL)(MEM FATAL)) # or WARNING

# Identifikace modemové karty

## Card 1: (serial identifier e2 00 00 01 00 05 50 c3 1e)
## Vendor Id GVC5005, Serial Number 256, checksum 0xE2.
## Version 1.0, Vendor version 0.1
## ANSI string -->LT Win Modem<--
##
## Logical device id HSM0140
##     Device support I/O range check register
#

# Chceme nakonfigurovat kartu GVC5005/256

(CONFIGURE GVC5005/256 (LD 0

# I/O adresa zaŁ□nÆ na 0x03f8, rozsah je 8
(IO 0 (SIZE 8) (BASE 0x03f8) )
# IRQ 4
```

```
(INT 0 (IRQ 4 (MODE +E)))
# DMA 5
(DMA 0 (CHANNEL 5))
# DMA 7
(DMA 1 (CHANNEL 7))
# DruhÆ I/O adresa zaŁ□nÆ na 0x0100, rozsah je 8
(IO 1 (SIZE 8) (BASE 0x0100) )
# Název karty
 (NAME "GVC5005/256[0]{LT Win Modem        }")
# Aktivujeme ji
(ACT Y)
# Konec konfigurace
))

######## Konfigurace zbývajících ISA karet ###############

## VrÆt□ v□echny karty do stavu □ŁekÆn□ na odezvu□
(WAITFORKEY)
```

ISA a jádra 2.4

Pokud používáte jádro 2.4 (což můžete zjistit příkazem *uname -r*), nemusíte se zaobírat programem *isapnp* a jeho konfiguračním souborem, protože v jádrech 2.4 je pro konfiguraci ISA karet implementován podobný mechanismus jako pro karty PCI. Abyste tuto funkci zapnuli, musíte při konfiguraci jádra odpovědět *Y* nebo *M* na otázku „Plug-and-Play support (CONFIG_PNP) a ISA Plug-and-Play suppor“ (CONFIG_ISAPNP) (tyto volby jsou v části „Plug-and-Play configuration“). Jádro přeložte a nainstalujte. Pokud jste u ISA Plug-and-Play support zvolili *M*, bude dobré zadat *modprobe isapnp*. Pro použití s touto novou metodou budete potřebovat ovladač, který ji podporuje (například *ltmodem 5.78*, viz dále).

Instalace ovladače modemu

Tuto kapitolu nemusíte číst celou, stačí si vybrat část odpovídající vašemu modemu. V tomto okamžiku zde popisujeme pouze instalaci modemu LT Modem pomocí ovladačů Lucent a Open Source ovladače. Pokud se vám podařilo zprovoznit jiný modem s jiným ovladačem, napište mi (alexandre12@mageos.com) a já tento popis doplním.

ltmodem 5.78

URL: http://www.tux.org/pub/dclug/marvin/ltmodem-5.78e.tar.gz

Autor: Lucent

Licence: Ne-GPL

Podporuje: ISA/PCI modemy s chipsetem Lucent (rodina Mars)

Funkce: modul jádra, simuluje sériový port. Podporuje PPP, fax a voice

Aktuální verze: 5.78e

Dokumentace: README-1ST

Vyžaduje: 2.2.x nebo 2.4.x; podporu ISA PNP; podporu modulů; překladač C

Jak na něj: Přečtěte si README-1ST, je to jednoduché.

Problémy:

- *Unresolved symbols: xxx_isapnp_xxxx*: spusťte *modprobe isapnp*.
- *Device or ressource busy*: modem není ovladačem podporován.

Komentář: Lucent se tentokrát vyhnul problémům s verzemi jádra a distribuce tím, že část ovladače je dodávána ve zdrojové podobě.

LT WinModem (od Lucentu) 5.68 (ZASTARALÉ)

URL: http://www.linmodems.org/linux568.zip

Autor: Lucent Technologies

Licence: Ne-GPL

Podporuje: ISA/PCI modemy s chipsetem Lucent (rodina Mars)

Funkce: modul jádra, simuluje sériový port; Podporuje PPP, fax a voice

Aktuální verze: 5.68

Dokumentace: README

Vyžaduje: 2.2.12 nebo vyšší jádro RedHat; podporu modulů

Jak na něj: Rozbalte distribuci (*unzip linux568.zip*) a spusťte *./ltinst*. To je celé!

Problémy:

- *„insmod: ltmodem: Unresolved symbol(s) *****"*: Jádro není kompatibilní s ovladačem. Nainstalujte si jádro 2.2.12 nebo vyšší.
- *„Warning: kernel version mismatch..."*: Nepoužíváte jádro 2.2.12-20. Jde pouze o varování, nebude to vadit ve funkci.
- *„ltmodem: init_module: device or resource busy"*:
  - Nemáte nainstalován modem *lt* ani kompatibilní.
  - Pokud máte ISA modem, nemáte nakonfigurováno *isapnp*.
  - Pokud máte PCI modem, nemáte nakonfigurovánu podporu PCI.

Program LTMODEM (OpenSource ovladač)

URL: http://www.close.u-net.com

Autoři: Richard Close a Pavel Machek

Licence: GPL

Podporuje: ISA/PCI modemy s chipsetem Lucent (rodina Mars)

Funkce: Ovladač v uživatelském prostoru, podporuje voice, nepodporuje PPP (V90)

Aktuální verze: 0.99

Dokumentace: README

Vyžaduje: překladač GNU C, automake

Jak na něj:

- PCI: *make* ; *make install* – Pak použijte program *ltmodem*.
- ISA: (přečtěte si README.ISA) *mv Makefile Makefile.PCI* ; *mv Makefile.ISA Makefile* ; editujte *config.h* a nastavte správné hodnoty pro váš modem (I/O, DMA, IRQ); *make* ; *make install* – Pak použijte program *ltmodem*.

Problémy:

- *„Sorry, I can't found any modem…"*:
  - Nemáte modem LT.
  - Máte ISA modem nenakonfigurovaný přes *isapnp*.
  - Máte nakonfigurovaný ISA modem, ale *ltmodem* není přeložen s podporou ISA.
  - Máte PCI modem a nemáte zapnutu podporu PCI v jádře.
  - Máte PCI modem, správné jádro, ale *ltmodem* není přeložen s podporou PCI.

Informace

- Domovská stránka projektu Linmodems: http://www.linmodems.org
- Domovská stránka projektu LTModem: http://www.close.u-net.com
- Domovská stránka společnosti Lucent Technologies: http://www.lucent.com
- Linuxové jádro: http://www.kernel.org
- Domovská stránka balíku *isapnptools*: http://www.roestock.demon.co.uk/isapnptools
- Dokumentační projekt: http://www.tldp.org

Zvuk na Linuxu

Originál: http://tldp.org/HOWTO/Sound-HOWTO

Tento dokument popisuje podporu zvuku v Linuxu. Uvádí seznam podporovaného zvukového hardwaru, popisuje jak nakonfigurovat ovladače jádra a odpovídá na často kladené otázky. Jeho záměrem je pomoci začínajícím uživatelům a omezit provoz v diskusních skupinách.

Úvod

Tento dokument je zamýšlen jako rychlá referenční příručka pokrývající všechno, co potřebujete vědět k instalaci a konfiguraci podpory zvuku v Linuxu. Odpovídá se v něm na často kladené otázky a uvádějí se odkazy na další informační prameny týkající se různých témat vztahujících se k počítačem generovanému zvuku a hudbě.

Záběr dokumentu se omezuje pouze na ty vlastnosti zvukových karet, které mají vztah k Linuxu. Obecnější informace o zvukových kartách a počítačově generovaném zvuku a hudbě naleznete mezi odkazy.

6850 UART MIDI Interface	AD1816/AD1816A based cards	AD1816/AD1816A sound chip	AD1848 sound chip
ADSP-2115	ALS-007 based cards (Avance Logic)	ALS-1x0 sound chip	ATARI onboard sound
ATI Stereo F/X	Acer FX-3D	AdLib	Amiga onboard sound
Audio Excel DSP 16	AudioDrive	Aztech Sound Galaxy Washington 16	Aztech Sound Galaxy WaveRider 3D
Aztech Sound Galaxy WaveRider Pro32	Beethoven ADSP-16	CMI8330 sound chip	CMI8338/8378 sound chip
Cardinal DSP16	Compaq Deskpro XL onboard sound	Corel Netwinder WaveArtist	Crystal CS423x
Crystal CS4280	Crystal CS46xx	ES1370 sound chip	ES1371 sound chip
ESC614 sound chip	ESS Maestro 1/2/2E sound chip	ESS Solo1 sound chip	ESS1688 sound chip
ESS1788 sound chip	ESS1868 sound chip	ESS1869 sound chip	ESS1887 sound chip
ESS1888 sound chip	ESS688 sound chip	Ensoniq AudioPCI (ES1370)	Ensoniq AudioPCI 97 (ES1371)

Ensoniq/Reveal/Spea SoundScape	Gallant SC-6000	Gallant SC-6600	Gravis Ultrasound
Gravis Ultrasound ACE	Gravis Ultrasound Max	Gravis Ultrasound with 16 bit option	HP Kayak
Highscreen Sound-Booster32 Wave3D	IBM MWAVE	Jazz 16	Logitech Sound Man 16
Logitech SoundMan Games	Logitech SoundMan Wave	MAD16 Pro (OpTi 82C9xx chipsets)	Media Vision Jazz16
MediaTriX AudioTriX Pro	Microsoft Windows Sound System	MiroSOUND PCM12	Mozart (OAK OTI-601)
NeoMagic 256AV/256ZX	OpTi 82C931	Orchid SW32	Personal Sound System (PSS)
Pinnacle MultiSound	Power Mac onboard sound	Pro Audio Spectrum 16	Pro Audio Studio 16
Pro Sonic 16	Q40 onboard sound	Roland MPU-401 MIDI interface	S3 SonicVibes
SGI Visual Workstation	SM Games	SY-1816	SoundBlaster 1.0
SoundBlaster 16	SoundBlaster 16ASP	SoundBlaster 2.0	SoundBlaster 32
SoundBlaster 64	SoundBlaster AWE32	SoundBlaster AWE64	SoundBlaster Live!
SoundBlaster PCI 128	SoundBlaster PCI 512	SoundBlaster Pro	SoundBlaster Vibra16
SoundBlaster Vibra16X	TI TM4000M notebook	Terratec Base 1	Terratec Base 64
ThunderBoard	Trident 4DWave DX/NX	Trident Ali 5451	Trident SiS 7018
Turtle Beach Maui	Turtle Beach MultiSound Classic	Turtle Beach MultiSound Fiji	Turtle Beach MultiSound Hurricane
Turtle Beach MultiSound Monterey	Turtle Beach MultiSound Pinnacle	Turtle Beach MultiSound Tahiti	Turtle Beach WaveFront Maui
Turtle Beach WaveFront Tropez	Turtle Beach WaveFront Tropez+	VIA 82Cxxx chip set	VIDC 16-bit sound
Yamaha OPL2 sound chip	Yamaha OPL3 sound chip	Yamaha OPL3-SA1 sound chip	Yamaha OPL3-SA2 sound chip
Yamaha OPL3-SA3 sound chip	Yamaha OPL3-SAx sound chip	Yamaha OPL4 sound chip	Yamaha YM3812 sound chip

Technologie zvukových karet

Tato kapitola představuje *velmi* stručný přehled počítačových zvukových technologií, jejím smyslem je usnadnit pochopení některých věcí, uváděných dále v textu. Podrobnější informace můžete najít v nějaké knize věnované digitálnímu zvuku nebo digitálnímu zpracování signálu.

Zvuk je *analogová* veličina, může nabývat libovolné hodnoty z určitého rozsahu. Počítače jsou *digitální*, pracují pouze s konkrétními (diskrétními) hodnotami. Zvukové karty používají zařízení, které se označuje jako *analogově-digitální převodník* (A/D nebo ADC), a jež převádí hodnoty napětí odpovídající analogovým zvukovým vlnám na digitální nebo číselné hodnoty, které je možné uložit v paměti. Podobně pak *digitálně-analogový převodník* (D/A, DAC) převádí číselné hodnoty zpátky na analogové hodnoty napětí, kterými se pak dá budit reproduktor, generující zvuk.

Proces konverze analogové veličiny na digitální, takzvané vzorkování (*sampling*), vnáší do záznamu chyby. Existují dvě hodnoty, které jsou klíčové pro stanovení, nakolik věrně bude navzorkovaný signál odpovídat originálu. *Vzorkovací kmitočet* udává počet vzorků „odebraných" za jednotku času (typicky se uvádí ve vzorcích za sekundu nebo v hertzech). Nízká vzorkovací frekvence přináší méně přesnou reprezentaci analogového signálu. *Velikost vzorku* je rozsah hodnot použitý k reprezentaci každého vzorku, typicky se uvádí v bitech. Čím větší velikost vzorku, tím přesnější bude digitalizovaný signál. Zvukové karty typicky pracují s 8 nebo 16bitovými vzorky a se vzorkovacími kmitočty od 4 000 do 44 000 vzorků za sekundu. Digitalizovaný signál navíc může obsahovat jeden nebo dva kanály (mono nebo stereo).

FM syntéza je starší technika generování zvuku. Je založena na kombinování různých vln (sinusových, trojúhelníkových, obdélníkových). Snáze se hardwarově implementuje než D/A převod, hůře se ale programuje a její možnosti jsou omezené. Řada zvukových karet nabízí FM syntézu kvůli zpětné kompatibilitě se staršími kartami a programy. Obvykle karta nabízí několik nezávislých zvukových generátorů, *kanálů*.

Tabulková syntéza kombinuje flexibilitu D/A konverze s vícekanálovými možnostmi FM syntézy. Při tomto řešení je možné nahrát digitalizované vzorky do vyhrazené oblasti paměti, a pak je přehrávat, kombinovat a upravovat s malými nároky na procesor. Všechny standardní zvukové karty dnes tabulkovou syntézu podporují.

Většina zvukových karet umožňuje *mixování*, tedy skládání signálů z více vstupních zdrojů a řízení jejich hlasitosti.

MIDI je zkratka „Musical Instrument Digital Interface", což je standardizovaný hardwarový a softwarový protokol, který umožňuje hudebním nástrojům komunikovat mezi sebou. Události posílané po MIDI sběrnici je možné ukládat jako MIDI soubory a později je použít k úpravám a přehrávání hudby. Řada zvukových zařízení obsahuje MIDI rozhraní. Ty, které je neobsahují, mohou alespoň přehrávat MIDI soubory prostřednictvím generátorů na kartě.

Soubory MOD představují běžný formát počítačově generované hudby. Kromě informací o přehrávaných tónech obsahují také digitalizované vzorky jednotlivých nástrojů (kanálů). Soubory MOD pocházejí z počítačů Amiga, s příslušným softwarem je však lze přehrávat i na jiných systémech včetně Linuxu.

Soubory MP3 představují oblíbený formát pro počítačovou distribuci hudby a zvuku. MP3 používá komplikované kompresní schéma (MPEG layer 3), kterým se dosahuje zhruba desetinásobná komprese s nepatrnou ztrátou kvality, porovnáváno se záznamem na zvukovém CD.

Podporovaný hardware

V této kapitole uvádíme seznam zvukových karet a rozhraní, které Linux aktuálně podporuje. Informace se vztahují k nejnovějšímu jádru, což bylo v době vzniku textu jádro 2.4.4. Uvádíme pouze ovladače, které jsou součástí standardní zdrojové distribuce jádra. Kromě toho jsou k dispozici i další ovladače (viz část *Alternativní ovladače*). Aktuální informace o podporovaných zvukových kartách a jejich funkcích naleznete jako součást zdrojové distribuce jádra, typicky nainstalované v adresáři */usr/src/linux/Documentation/sound*.

Platformy

Informace v tomto dokumentu se vztahují k Linuxu na platformě Intel x86.

Zvukové ovladače by měly pracovat také s většinou zvukových karet na platformě Alpha. U některých karet však může na platformě Alpha docházet ke konfliktům vstupně-výstupních portů,

přestože na platformě x86 pracují bezchybně. Není tedy možné obecně říct, zda daná karta pracovat bude či nebude, dokud to někdo nevyzkouší...

Podle informací od uživatelů nefungují ovladače zvuku na platformě PowerPC, ale její podpora by měla být v budoucnu vylepšena. Na platformě MIPS je možné podporu zvuku nastavit, a některé počítače MIPS jsou vybaveny EISA sloty, a/nebo vestavěným zvukovým hardwarem.

Jádro Linuxu dále obsahuje samostatné ovladače pro Amiga a Atari verze Linuxu, které implementují kompatibilní podmnožiny funkcí zvukového ovladače platformy Intel přímo na vestavěném zvukovém hardwaru těchto platforem.

Verze Linuxu pro SPARC obsahuje podporu zvuku pouze pro některé modely pracovních stanic Sun. Pokud vím, funguje vestavěná zvuková karta, avšak není podporováno externí DSP zařízení, protože Sun neuvolnil jeho specifikaci.

Typy zvukových karet

Existuje celá řada zvukových karet pro různé dostupné architektury sběrnic. Dále uvádíme stručný přehled běžnějších typů a jejich funkcí.

Karty *ISA* představují nejstarší typ zvukových karet, určených pro původní ISA sběrnici (bez podpory Plug-and-Play). Typicky se nastavují pomocí přepínačů, kterými se volí vstupně-výstupní adresy, přerušení a DMA kanál. Dnes už je pravděpodobně nikdo nevyrábí.

Karty *ISA Plug-and-Play* používají rozšířenou verzi ISA sběrnice s podporou softwarové identifikace a konfigurace nastavení karet. Některé se možná ještě vyrábějí.

Karty *PCI* používají rychlejší PCI sběrnici, která podporuje softwarovou identifikaci a konfiguraci karet. Většina dnes vyráběných karet používá právě PCI sběrnici. Většina základních desek s integrovanými zvukovými kartami připojuje tyto karty rovněž na PCI sběrnici.

USB je nová architektura sběrnice pro připojování externích zařízení za provozu. Teoreticky mohou existovat i USB zvukové karty, momentálně však, pokud vím, existují pouze na USB sběrnici připojované reproduktory.

Zvukové karty

Následující zvukové karty jsou podporovány jádrem Linuxu. Některé z položek seznamu představují zvukové čipy a ne přímo modely zvukových karet. Seznam je samozřejmě neúplný, protože existuje celá řada karet, které jsou s níže uvedenými kompatibilní. Aby to bylo ještě zamotanější, někteří výrobci pravidelně mění design svých karet, čímž vznikají různé nekompatibility, a tyto karty pak stále prodávají jako stejný model.

Poznámka ke kompatibilitě: Většina zvukových karet je označována jako *SoundBlaster compatible*, avšak jen velmi málo z nich je opravdu kompatibilních natolik, aby fungovaly s linuxovým ovladačem karet SoundBlaster. Tyto karty obvykle fungují lépe s ovladači MSS/WSS nebo MAD16. S ovladačem SoundBlaster fungují pouze opravdové SoundBlaster karty vyrobené v Creative Labs, které obsahují zvukový čip Creative (tedy např. SoundBlaster16 Vibra), MV Jazz16 nebo ESS688/1688. Pokus o provozování „SoundBlaster Pro Compatible“ karet s ovladačem SoundBlaster je typicky pouze ztráta času.

Jádro Linuxu podporuje SCSI port na některých zvukových kartách (např. na ProAudioSpectrum 16) a proprietární rozhraní některých CD-ROM mechanik (například SoundBlaster Pro). Další informace viz dokumenty SCSI HOWTO a CDROM HOWTO.

Ovladač podporující port joysticku, který je součástí některých zvukových karet, je součástí jádra 2.2 a novějších. Ovladače SCSI, CD-ROM, joysticku a zvukové ovladače jsou jeden na druhém úplně nezávislé.

Alternativní zvukové ovladače

OSS/4Front

Zvukovou podporu v linuxovém jádře původně vytvořil Hannu Savolainen. Poté začal vyvíjet Open Sound System, komerční zvukové ovladače společnosti 4Front Technologies, podporující celou řadu unixových systémů. Vývoj modulárních zvukových ovladačů převzal Alan Cox a celá řada dalších lidí se podílela na různých opravách a vývoji ovladačů pro různé nové zvukové karty. Tyto ovladače byly součástí distribuce RedHat 5.0 až 5.2. Od verze jádra 2.0 jsou standardní součástí jádra. Alan Cox má podporu zvuku stále na starosti, a Hannu pravidelně přispívá kódem z komerčních ovladačů.

Komerční ovladač Open Sound System společnosti 4Front Technologies je snáze konfigurovatelný a podporuje více zvukových karet, zejména novějších modelů. Je rovněž kompatibilní s aplikacemi určenými pro standardní zvukové ovladače jádra. Nevýhodou je, že za tyto ovladače je nutné platit a nezískáte jejich zdrojový kód. Verzi k odzkoušení si můžete stáhnout zdarma. Další informace naleznete na stránce 4Front Technologies na adrese http://www.opensound.com.

ALSA

Jaroslav Kysela a další začali vyvíjet alternativní zvukový ovladač pro kartu Gravis UltraSound. Projekt byl později přejmenován na *Advanced Linux Sound Architecture* (ALSA) a autoři věří, že nyní jde o obecné zvukové ovladače, které lze použít namísto standardních jaderných ovladačů. Ovladače ALSA podporují řadu běžných zvukových karet, jsou plně duplexní, plně modulární a kompatibilní se zvukovou architekturou jádra. Stránky projektu najdete na adrese http://www.alsa-project.org. Překladu a instalaci těchto ovladačů se věnuje samostatný dokument „Alsa-sound -mini-HOWTO“. Ovladače ALSA se staly standardní součástí jádra od vývojové verze 2.5.

Turtle Beach

Markus Mummert (mum@mmk.e-technik.tu-muenchen.de) napsal ovladače pro zvukové karty Turtle Beach MultiSound, Tahiti a Monterey. Dokumentace říká:

Jsou určeny k vysoce kvalitnímu záznamu a přehrávání na disk bez ztráty synchronizace i na zatížených systémech. Další funkce, jako vlnová syntéza, MIDI a DSP nejsou funkční. Není možné současné přehrávání i záznam. Ovladače slouží jako náhrada VoxWare a byly testovány na různých jádrech od 1.0.9 po 1.2.1. Lze je také nainstalovat na systémech SysV386R3.2.

Najdete je na adrese http://www.cs.colorado.edu/~mccreary/tbeach.

Roland MPU-401

Kim Burgaard (burgaard@daimi.aau.dk) napsal ovladač a nástroje pro MIDI rozhraní Roland MPU-401. Z popisu uvádíme:

Ovladač pro MIDI rozhraní kompatibilní s Roland MPU-401 (včetně Roland SCC-1 a RAP-10/ATW-10). Obsahuje užitečnou kolekci nástrojů pro přehrávání a záznam souborů MIDI.

Od verze 0.11a došlo k řadě vylepšení. Mimo jiné nyní ovladač podporuje sdílení přerušení a je kompatibilní s modulární architekturou jádra. Nabízí funkce metronomu, možnost synchronizace například s grafickými rozhraními bez ztráty přesnosti, rozhraní pro přehrávání a nahrávání, a další a další.

Naleznete jej na adrese ftp://www.ibiblio.org/pub/Linux/kernel/sound/mpu401-0.2.tar.gz.

SoundBlaster Live!

Creative Labs nabízejí linuxové ovladače pro různé karty včetně SoundBlaster Live! na adrese http://opensource.creative.com.

Packet Radio

Relativně nová možnost použití zvukových karet je jako modem pro amatérské rádio. Jádra 2.1 a novější obsahují ovladač, který pracuje s kartami SoundBlaster a kartami kompatibilními s Windows Sound System, a umožňuje je použít s protokoly AFSK 1200 bps a FSK 9600 bps. Podrobnosti viz Linux AX25 HOWTO. (Mimochodem, volačka autora dokumentu je VE3ICH.)

Interní reproduktor

Existuje alternativní zvukový ovladač, který nevyžaduje použití žádného zvukového hardwaru, využívá vestavěný reproduktor. Je softwarově kompatibilní s ovladači „opravdových" zvukových karet, jak se ale dá očekávat, výstupní kvalita je podstatně nižší a ovladač má vysoké nároky na procesor. Výsledky jsou velmi proměnlivé, závisí hodně na vlastnostech konkrétního reproduktoru. Podrobnější informace naleznete v dokumentaci, která je součástí programu.

Nejnovější verzi tohoto ovladače najdete na adrese ftp://ftp.infradead.org/pub/pcsp/.

Paralelní port

Další možnost je postavit digitálně-analogový převodník pomocí paralelního portu a několika součástek. Dosáhne se tak vyšší kvality než s použitím vestavěného reproduktoru, nároky na procesor jsou však stále vysoké. Software popsaný v předchozí části podporuje i toto uspořádání a v dokumentaci popisuje i návod ke konstrukci potřebného hardwaru.

Instalace

Konfigurace podpory zvuku v Linuxu obnáší následující kroky:

1. Instalace zvukové karty.
2. Konfigurace Plug-and-Play (pokud je to nutné).
3. Konfigurace zvukové podpory v jádře a překlad jádra.
4. Vytvoření souborů zařízení.
5. Spuštění nového jádra a otestování instalace.

Některé distribuce Linuxu obsahují nástroj pro konfiguraci zvukového ovladače, který provede detekci zvukové karty a nastaví všechny potřebné konfigurační soubory tak, aby se nahrály ovladače potřebné pro danou kartu. Například distribuce RedHat obsahuje nástroj *sndconfig*. Pokud ve vaší distribuci nějaký takový nástroj je, doporučuji jej použít. Pokud bude fungovat, můžete přeskočit zbytek pokynů v této kapitole.

Pokud fungovat nebude nebo pokud dáte přednost ručnímu nastavení, abyste přesně rozuměli tomu, co a proč se nastavuje, následující kapitoly pokrývají celý postup krok za krokem.

Instalace zvukové karty

Držte se pokynů výrobce karty, nebo si nechte kartu nainstalovat dodavatelem počítače.

Starší karty používají různé přepínače pro nastavení přerušení, DMA kanálů a podobně. Poznamenejte si nastavené hodnoty. Pokud si nejste jisti, použijte hodnoty nastavené výrobcem. Při nastavování se pokud možno vyhněte konfliktům s jinými zařízeními (síťové karty, adaptéry SCSI, sériové a paralelní porty).

Obvykle byste měli být schopni použít stejné hodnoty vstupně-výstupních portů, přerušení a DMA, které fungují i pod DOSem. V některých případech (zejména při použití PnP karet) budete muset použít jiné hodnoty. Neobejdete se možná bez nějakého experimentování.

Konfigurace ISA Plug-and-Play

Některé karty používají k nastavení vstupně-výstupních adres, přerušení a kanálu DMA protokol ISA Plug-and-Play. Pokud máte zvukovou kartu PCI, nebo hodně starou ISA kartu, kde se nastavení provádí pomocí přepínačů, můžete tuto část přeskočit.

Doporučená metoda konfigurace Plug-and-Play karet je použití nástroje *isapnp*, který je součástí většiny distribucí Linuxu (případně si jej můžete stáhnout ze serveru RedHat na adrese http://www.redhat.com).

Nejprve se podívejte do dokumentace k vaší distribuci. Možná už obsahuje podporu Plug-and-Play, anebo ji používá poněkud jinak, než dále popisujeme. Pokud budete celou podporu nastavovat sami, naleznete potřebné informace na manuálových stránkách nástroje *isapnp*. Použitý postup vypadá zhruba takto:

- Pomocí nástroje *pnpdump* si zjistěte možná nastavení všech vašich Plug-and-Play zařízení, výsledek uložte do souboru */etc/isapnp.conf*.
- Zvolte taková nastavení zvukové karty, která nevedou ke konfliktu s jinými zařízeními v systému, a odkomentujte příslušné řádky v souboru */etc/isapnp.conf*. Nezapomeňte odkomentovat řádek (ACT Y) u konce souboru.
- Zkontrolujte, zda se při startu systému některým z inicializačních skriptů spouští program *isapnp*. Restartujte systém, nebo spusťte *isapnp* ručně.

Pokud z nějakých důvodů nechcete nebo nemůžete použít nástroje *isapnp*, máte několik dalších možností. Používáte-li kartu pod Windows 95 nebo 98, můžete kartu nastavit pomocí správce zařízení a pak provést teplý restart Linuxu programem LOADLIN. Windows i Linux musí používat stejné nastavení karty.

Pokud kartu používáte pod DOSem, můžete ji nastavit pomocí nástroje *icu*, dodávaného s kartami SoundBlaster16 PnP a pak provést teplý restart Linuxu programem LOADLIN. Opět se musí v DOSu a Linuxu používat stejná nastavení karty.

V jádrech 2.4 a vyšších je implementována podpora ISA PnP. Ovladače některých karet podporují automatickou detekci a konfiguraci karty bez použití *isapnp*. Podrobnosti byste měli nalézt v dokumentaci k ovladači karty.

Konfigurace jádra

Potřebujete, aby jádro obsahovalo ovladače vaší zvukové karty. Je možné, že jádro, které používáte, tyto ovladače již obsahuje. Ve většině případů jsou ovladače vytvořeny jako samostatné moduly. Které moduly jsou k dispozici můžete zjistit v adresáři */lib/modules*. U jádra 2.4.4 najdete ovladače zvukových karet v adresáři */lib/modules/2.4.4/kernel/drivers/sound*. Pokud zde naleznete modul ovladače vaší karty, můžete jej použít a vyhnete se překladu celého jádra.

Pokud ovladače jádra k dispozici nejsou, budete muset nakonfigurovat a přeložit nové jádro. Ovladače můžete přeložit přímo jako součást jádra, nebo jako samostatné moduly. Ve většině případů se doporučuje jejich překlad jako samostatných modulů, protože tak můžete snáze experimentovat s různými ovladači v případě, že si nejste jisti tím správným. Navíc můžete ovladač odstranit, když jej nepotřebujete, a ušetřit tak paměť. Přeložení ovladačů přímo jako součást jádra je vhodné tehdy, pokud problematice modulů nerozumíte a hledáte jednodušší řešení.

Podrobnosti o překladu jádra naleznete v dokumentu Linux Kernel HOWTO. Dále uvádíme pouze některé postřehy týkající se specificky zvukových karet.

Pokud jste podporu zvuku v jádře nikdy dříve nenastavovali, je rozumné přečíst si dokumentaci dodávanou se zvukovými ovladači, zejména dokumentaci vztahující se k vaší zvukové kartě. Příslušné soubory naleznete v dokumentaci jádra, která se typicky instaluje do adresáře */usr/src/linux/Documentation/sound*. Pokud tento adresář nenaleznete, používáte buď velmi starou verzi jádra, anebo nemáte zdrojové kódy jádra nainstalovány.

Jádro sestavte běžným způsobem. Momentálně jsou k dispozici tři rozhraní pro konfiguraci jádra. Pokud používáte prostředí systému X Window, můžete použít příkaz *make xconfig*. V textovém režimu můžete použít *make menuconfig*. Konečně klasická řádková metoda se spouští příkazem *make config*.

Při konfiguraci narazíte na celou řadu voleb zvukové karty a použitých ovladačů. Co která možnost znamená byste měli zjistit z nápovědy konfiguračního systému. Zvolte nejvhodnější volby podle svého vědomí a svědomí.

Po nastavení příslušných voleb jádro přeložte a nainstalujte postupem podle dokumentu Linux Kernel HOWTO.

Vytvoření souborů zařízení

Ke správné funkci musí být vytvořeny soubory reprezentující zvuková zařízení. Obvykle se vytvářejí při instalaci Linuxu. Rychlou kontrolu můžete provést níže uvedenými příkazy. Pokud výstup vypadá stejně jako v našem příkladu (datové údaje se mohou lišit), je s největší pravděpodobností vše v pořádku.

```
% ls -l /dev/dsp
crw-rw-rw-   1 root     root      14,   3 Apr 25  1995 /dev/dsp
```

Samotná existence souborů zařízení nic nezaručuje. Aby zařízení pracovalo, musí být přeloženy nebo nahrány i příslušné ovladače jádra (viz dále).

V řídkých případech, nebo pokud se domníváte, že jsou příslušné soubory poškozeny, je můžete vytvořit znovu. Většina distribucí Linuxu obsahuje skript */dev/MAKEDEV*, který slouží k tomuto účelu.

Pokud používáte souborový systém *devfs* podporovaný jádry 2.4, soubory zvukových zařízení se nacházejí v */dev/sound*, existují však odkazy na starší zařízení, jako je */dev/dsp*.

Spuštění Linuxu a testování instalace

Nyní byste měli být schopni nabootovat nové jádro a otestovat ovladače zařízení. Při instalaci a spuštění nového jádra použijte běžný postup (nezapomeňte si samozřejmě zálohovat původní jádro).

Pokud používáte modulární ovladače, budete je muset nahrát příkazem *modprobe*, například *modprobe sb* pro karty SoundBlaster.

Po nabootování jádra nebo nahrání modulů hledejte pomocí příkazu *dmesg* hlášení jako je:

```
Soundblaster audio driver Copyright (C) by Hannu Savolainen 1993-1996
sb: Creative SB AWE64  PnP detected
sb: ISAPnP reports 'Creative SB AWE64  PnP' at i/o 0x220, irq 5, dma 1, 5
SB 4.16 detected OK (220)
sb: 1 Soundblaster PnP card(s) found.

Crystal 4280/46xx + AC97 Audio, version 1.22.32, 10:28:40 Apr 28 2001
cs46xx: Card found at 0xf4100000 and 0xf4000000, IRQ 11
cs46xx: Thinkpad 600X/A20/T20 (1014:0153) at 0xf4100000/0xf4000000, IRQ 11
ac97_codec: AC97 Audio codec, id: 0x4352:0x5914 (Cirrus Logic CS4297A rev B)
```

Z hlášení by mělo vyplývat, že byla nalezena zvuková karta a že její parametry odpovídají jejímu typu a případnému nastavení. Můžete případně nalézt chybová hlášení nebo upozornění v případě, že ovladač není správně nakonfigurován nebo zvolen.

Ve starších verzích tohoto dokumentu se doporučovala kontrola obsahu souboru */dev/sndstat*. Ten už není v jádrech 2.4 a vyšších podporován.

Měli byste být schopni přehrát zvukový soubor. Sežeňte si nějaký zvukový soubor a proveďte základní kontrolu zvukového zařízení:

```
% cat endoftheworld >/dev/dsp
% cat crash.au >/dev/audio
```

Použití příkazu *cat* není nejvhodější metoda přehrávání zvukových souborů, jde pouze o jednoduchou kontrolu. Určitě si později opatříte nějaký přehrávač, který poslouží lépe.

Pokud výše uvedené příkazy vypíší „I/O error“, podívejte se na konec výpisu hlášení jádra pomocí příkazu *dmesg*. Měli byste tam nalézt nějaké chybové hlášení. Velmi často to bude „Sound: DMS (output) timed out – IRQ/DRQ config error?“ Znamená to, že ovladač nepřijal od karty očekávané přerušení. Ve většině případů to znamená, že nefunguje přerušení nebo DMA nastavené v ovladači. Nejjednodušší metoda je vyzkoušet všechna přerušení a DMA podporovaná kartou.

Další možná příčina je ta, že zařízení není kompatibilní s použitým ovladačem. Prakticky s jistotou to nastane, pokud použijete kartu „SoundBlaster (Pro/16) Compatible“ a ovladač SoundBlaster. V takovém případě potřebujete zjistit, s čím je vaše zvuková karta *opravdu* kompatibilní. (Například dotazem na *comp.os.linux.hardware*.)

Příklady zvukových souborů můžete získat na adrese ftp://tsx-11.mit.edu/pub/linux/packages/sound/snd-data-0.1.tar.Z.

Dále můžete ověřit nahrávání. Připojte signál do vstupu karty a následujícími příkazy proveďte jednoduchý test:

```
# 4 sekundy záznamu z mikrofonu
% dd bs=8k count=4 </dev/audio >sample.au
4+0 records in
4+0 records out
```

```
# płehrÆn zÆznamu
% cat sample.au >/dev/audio
```

Samozřejmě nesmíte zapomenout připojit mikrofon a mluvit do něj. Kromě toho možná budete potřebovat nějaký mixážní program, abyste mikrofon nastavili jako vstupní zařízení a nastavili správně citlivost vstupu.

Pokud testy dopadnou úspěšně, lze předpokládat, že zvukový D/A a A/D hardware i software funguje. Pokud narazíte na nějaké problémy, čtěte dále.

Problémy

Pokud zvuková karta stále nefunguje správně, uvádíme dále, co byste měli zkontrolovat. Jednotlivé rady jsou seřazeny od nejjednodušších po nejsložitější. Narazíte-li na nějaký problém, nejprve jej vyřešte a pak pokračujte dále.

Zkontrolujte, zda používáte nově přeložené jádro

Zkontrolujte datum vytvoření jádra, abyste si ověřili, že opravdu používáte jádro, které jste přeložili s podporou zvuku. Můžete to provést příkazem *uname*:

```
% uname -a
Linux fizzbin 2.2.4 #1 Tue Mar 23 11:23:21 EST 1999 i586 unknown
```

Anebo vypsáním souboru */proc/version*:

```
% cat /proc/version
Linux version 2.2.4 (root@fizzbin) (gcc version 2.7.2.3) #1 Tue Mar 23 11:23:21
EST 1999
```

Pokud datum neodpovídá tomu, kdy jste jádro přeložili, pak používáte staré jádro. Opravdu jste restartovali počítač? Pokud používáte LILO, reinstalovali jste zavaděč (typicky příkazem *lilo*)? Pokud bootujete z diskety, vytvořili jste novou disketu?

Pokud používáte modulární ovladač, příkazem *lsmod* ověřte, zda jsou moduly zavedeny:

```
% /sbin/lsmod
Module                  Size  Used by
sb                      6320   0 (unused)
sb_lib                 35040   0 [sb]
uart401                 6544   0 [sb_lib]
sound                  59888   0 [sb_lib uart401]
soundcore               4144   5 [sb_lib sound]
isa-pnp                28304   0 [sb]
...
```

Detekovalo jádro při startu zvukovou kartu?

Zkontrolujte, zda jádro při startu detekovalo zvukovou kartu. Měli byste vidět příslušná hlášení při startu systému. Pokud jste je nestihli přečíst, můžete je znovu vypsat příkazem *dmesg*:

```
% dmesg
```

nebo

```
% tail /var/log/messages
```

Pokud karta nebyla nalezena, něco není v pořádku. Zkontrolujte, zda je skutečně správně nainstalována. Pokud funguje pod DOSem, můžete si být přiměřeně jisti, že hardware funguje a problém

je v konfiguraci jádra. Buď jste nakonfigurovali špatný typ karty nebo špatné parametry, případně karta není kompatibilní s linuxovými ovladači.

Jedna možná příčina je ta, že používáte některý z *kompatibilních* typů karet, které vyžadují inicializaci dosovým ovladačem. Zkuste nabootovat DOS a nahrát ovladač karty poskytnutý výrobcem. Pak nabootujte Linux stiskem Ctrl+Alt+Del. Zkontrolujte, že nastavení vstupně-výstupní adresy, DMA a přerušení jsou v Linuxu stejná jako v DOSu. Přečtěte si soubor Readme.cards v distribuci ovladače, který obsahuje doporučení pro nastavení různých typů karet.

Pokud karta není v souboru uvedena, je možné, že ji linuxové ovladače nepodporují. Zkuste zjistit další informace v kapitole odkazů na konci tohoto dokumentu.

Lze číst ze zařízení dsp?

Zkuste číst ze zařízení */dev/audio* příkazem *dd* uvedeným výše. Příkaz by měl proběhnout bez chyb. Pokud nefunguje, je možný konflikt v nastaveních IRQ nebo DMA nebo je problém s kompatibilitou hardwaru (zařízení není v Linuxu podporováno, anebo je ovladač nakonfigurován pro jiné zařízení).

Méně pravděpodobná možnost je hardwarový problém. Pokud to jde, otestujte kartu pod DOSem, abyste tento problém vyloučili.

Když všechno selže

Pokud karta stále nefunguje, vyzkoušejte tato závěrečná doporučení:

- znovu si přečtěte tento dokument,
- přečtěte si informace, na něž se odkazujeme na konci dokumentu a relevantní dokumentaci ke zdrojovým kódům jádra,
- zeptejte se v diskusních skupinách *comp.os.linux* nebo podobně,
- použijte nějaký vyhledávací nástroj, například http://www.google.com, a inteligentně zvolte vyhledávací kritéria,
- napište autorovi zvukového ovladače,
- napište autorovi tohoto dokumentu.

Aplikace pro práci se zvukem

Dále uvádíme přehled některých aplikací, které asi budete chtít použít se zvukovou kartou. Aktuálnější informace můžete získat na Linux Software Map, v internetových archivem a přímo v dokumentaci k vaší distribuci.

Pokud používáte grafické prostředí jako KDE nebo GNOME, měli byste mít k dispozici přímo řadu multimediálních aplikací, které jsou součástí těchto prostředí.

Minimálně budete zřejmě potřebovat následující aplikace:

- program pro konverzi zvukových formátů (např. *sox*)
- mixér (například *aumix* nebo *xmix*)
- přehrávač digitalizovaných záznamů (například *play* nebo *wavplay*)
- přehrávač souborů MOD (například *tracker*)
- přehrávač souborů MIDI (například *playmidi*)

Většina těchto nástrojů existuje v textové i v grafické verzi. Existují samozřejmě i další aplikace, například pro syntézu a rozpoznávání řeči, které si můžete vyzkoušet.

Odpovědi na časté otázky

Tato kapitola odpovídá na některé otázky, které se často vyskytují v diskusních skupinách a poštovních konferencích. Odpovědi na další otázky můžete najít také na stránkách ovladače OSS.

Co jsou jednotlivé soubory zvukových zařízení?

Uvádíme seznam obvyklých souborů zvukových zařízení, některé distribuce Linuxu však mohou používat trochu jiné značení.

`/dev/audio`	Standardně odkaz na */dev/audio0*.
`/dev/audio0`	Zvukové zařízení kompatibilní se Sun Workstation (jde o částečnou implementaci bez podpory *ioctl* rozhraní společnosti Sun, podporováno je pouze u-law kódování).
`/dev/audio1`	Druhé zvukové zařízení (pokud je podporováno zvukovou kartou, nebo jsou-li instalovány dvě zvukové karty).
`/dev/dsp`	Standardně odkaz na */dev/dsp0*.
`/dev/dsp0`	První digitální vzorkovací zařízení.
`/dev/dsp1`	Druhé digitální vzorkovací zařízení.
`/dev/mixer`	Standardně odkaz na */dev/mixer0*.
`/dev/mixer0`	První zvukový mixér.
`/dev/mixer1`	Druhý zvukový mixér.
`/dev/music`	Vysokoúrovňové rozhraní sekvencéru.
`/dev/sequencer`	Nízkoúrovňový přístup k MIDI, FM a GUS.
`/dev/sequencer2`	Standardně odkaz na */dev/music*.
`/dev/midi00`	První přímý port MIDI.
`/dev/midi01`	Druhý přímý port MIDI.
`/dev/midi02`	Třetí přímý port MIDI.
`/dev/midi03`	Čtvrtý přímý port MIDI.
`/dev/sndstat`	Zobrazuje status zvukového zařízení (dostupné též jako */proc/sound*, v jádrech 2.4 odstraněno).

Ovladač interního reproduktoru nabízí následující zařízení:

`/dev/pcaudio`	Ekvivalentní k */dev/audio*.
`/dev/pcsp`	Ekvivalentní k */dev/dsp*.
`/dev/pcmixer`	Ekvivalentní k */dev/mixer*.

Jak můžu přehrát vzorek zvuku?

Zvukové soubory ve formátu Sun workstation (*.au*) je možné přehrát odesláním na */dev/audio*. Čisté vzorky je možné odeslat na */dev/dsp*. Obecně výsledek nebude valný a doporučuje se použít programy jako *play*, které rozeznají použitý formát souboru a správně nastaví vzorkovací kmitočet zvukové karty a podobně.

Pokud používáte grafické prostředí jako KDE nebo GNOME, měli byste mít k dispozici grafické programy pro přehrávání zvuku. Nejlepší výsledky při přehrávání souborů WAV zajišťují programy *wavplay* nebo *vplay* (v balíčku snd-util). Neumějí však přehrát WAV soubory komprimované

kompresí Microsoft ADPCM. Starší verze programu *play* (z balíčku Lsok) nepřehrávají správně 16bitové soubory WAV.

K přehrání většiny zvukových souborů lze použít program *splay* z balíčku snd-util, pokud mu předáte správné parametry a soubory.

Jak můžu zaznamenávat zvuk?

Čtením zařízení */dev/audio* nebo */dev/dsp* získáte navzorkovaná data, která můžete přesměrovat do souboru. Jednodušší řízení vzorkovací frekvence a dalších parametrů nabízí programy jako *vrec*. K nastavení správného vstupního zařízení budete možná potřebovat nějaký mixážní program.

Můžu nainstalovat více zvukových karet?

Se stávajícími ovladači je možné mít současně nainstalováno více karet SoundBlaster, SoundBlaster/Pro, SoundBlaster16, MPU-401 nebo MSS. Instalace dvou karet SoundBlaster vyžaduje ruční úpravu souboru *local.h* a nadefinování maker SB2_BASE, SB2_IRQ, SB2_DMA a případně SB2_DMA2. Je také možné současně používat SoundBlaster a PAS16.

U jader 2.0 a novějších, v nichž se zvuk nastavuje příkazem *make config*, se namísto souboru *local.h* edituje soubor */usr/include/linux/autoconf.h*. Za částí, která obsahuje následující řádky:

```
#define SBC_BASE 0x220
#define SBC_IRQ (5)
#define SBC_DMA (1)
#define SB_DMA2 (5)
#define SB_MPU_BASE 0x0
#define SB_MPU_IRQ (-1)
```

doplňte následující (s hodnotami, které odpovídají vašemu systému):

```
#define SB2_BASE 0x330
#define SB2_IRQ (7)
#define SB2_DMA (2)
#define SB2_DMA2 (2)
```

Následující ovladače neumožňují použití více karet:

- GUS (omezení ovladače)
- MAD16 (hardwarové omezení)
- AudioTrix Pro (hardwarové omezení)
- CS4332 (hardwarové omezení)

Chyba „No such file or directory" pro zvuková zařízení

Musíte mít vytvořeny soubory zvukových zařízení. Viz část věnovaná jejich vytvoření. Pokud soubory existují, ověřte, zda mají správně nastaveno hlavní a vedlejší číslo zařízení (starší distribuce Linuxu mohly vytvořit tyto soubory se špatnými čísly).

Chyba „No such device" pro zvuková zařízení

Jádro neobsahuje podporu zvukového zařízení nebo nastavená vstupně-výstupní adresa neodpovídá nastavení karty. Zkontrolujte, zda používáte správné jádro a zda parametry ovladače odpovídají nastavení karty.

Chyba „No space left" pro zvuková zařízení

K této chybě může dojít při pokusu o přehrávání dat přes zařízení */dev/audio* nebo */dev/dsp,* pokud příslušné soubory zařízení neexistují. Zvukové zařízení se pak chová jako normální soubor a zápis vede k zaplnění disku. Musíte spustit skript uvedený v části *Vytvoření souborů zařízení*.

K této chybě může dojít také s jádrem 2.0 a vyšším, pokud není dostatek volné paměti RAM. Ovladač zařízení potřebuje minimálně dvě stránky (8K) souvislého fyzického paměťového prostoru pro každý DMA kanál. Stává se to občas u počítačů s méně než 16 MB RAM nebo u počítačů, které hodně dlouho běží. Buffery pro DMA přenos můžete alokovat při nahrávání ovladače parametrem jádra *dma_buf=1*.

Chyba „Device busy" pro zvuková zařízení

V jednom okamžiku může mít zvukové zařízení otevřen jen jeden proces. S největší pravděpodobností má požadované zařízení otevřen jiný proces. Lze to zjistit například příkazem *fuser*:

```
% fuser -v /dev/dsp
/dev/dsp:             USER       PID ACCESS COMMAND
                      tranter    265 f....  tracker
```

Výše uvedený příklad ukazuje, že zařízení *dsp* má otevřen proces 265. Můžete počkat, až proces skončí, nebo jej můžete zabít. Abyste viděli zařízení použitá jinými uživateli, než jste vy sami, musíte příkaz *fuser* spustit jako *root*.

V prostředí KDE řídí obvykle přístup ke zvukovému zařízení zvukový server *artsd*. Aplikace musejí o přehrávání zvuku žádat zvukový server, anebo musíte zvukový server zastavit. Obdobná situace nastává v prostředí GNOME se zvukovým serverem *esd*.

Stále dostávám chybu Device busy!

Podle Briana Gougha může potenciálně docházet ke konfliktu karet SoundBlaster s nastaveným kanálem DMA 1 a páskovou jednotkou QIC-02, která rovněž používá DMA kanál 1. Pokud používáte FTAPE, možná máte tento ovladač zapnutý. Podle dokumentu FTAPE-HOWTO není ovladač QIC-02 nezbytný, stačí ovladač QIC-117. Nastavte jádro tak, aby používalo ovladač QIC-117 a ne QIC-02 – pak by mělo být možné současně použít FTAPE i zvukové zařízení.

Částečné přehrávání digitalizovaného zvuku

Chyba se projevuje typicky tak, že soubor se přehrává asi sekundu a pak přehrávání úplně skončí, nebo vyhlásí chybu „missing IRQ" nebo „DMA timeout". S největší pravděpodobností máte chybně nastaveno přerušení nebo DMA kanál. Ověřte, že parametry ovladače odpovídají nastavení karty a že nedochází ke konfliktu s jinými zařízeními.

Podobnou chybou je, že při přehrávání dochází k zacyklení. S největší pravděpodobností jde o konflikt přerušení.

Při přehrávání souborů MOD dochází k zasekávání

Přehrávání souborů MOD je náročné na procesor. Buď máte spuštěno příliš mnoho procesů, anebo je váš počítač příliš pomalý, než aby zvládl přehrávání těchto souborů. Můžete vyzkoušet:

- zkusit přehrávání s nižší vzorkovací frekvencí nebo v režimu mono
- vypnutí jiných procesů

- koupě nového počítače
- koupě výkonnější zvukové karty (například Gravis UltraSound)

Pokud používáte kartu Gravis UltraSound, měli byste použít přehrávač souborů MOD určený speciálně pro GUS (například *gmod*).

Chyby při překladu zvukových aplikací

Verze 1.0c a starší zvukového ovladače používaly odlišné a nekompatibilní schéma *ioctl* volání. Sežeňte si novější verzi aplikace, nebo upravte zdrojové kódy tak, aby používaly nová volání. Podrobnosti viz soubor Readme zvukového ovladače.

Zkontrolujte také, zda používáte poslední verze souborů *soundcard.h* a *ultrasound.h*. Viz návod k instalaci na začátku tohoto dokumentu.

Chyba SEGV při spuštění zvukových programů, které dříve fungovaly

Pravděpodobně stejná příčina jako u předchozí otázky.

Jaké jsou známé chyby nebo omezení zvukového ovladače?

Viz soubory, které jsou součástí zdrojového kódu ovladače.

Kde jsou dokumentovány ioctl volání ovladače a podobně?

V současnosti nejlepší dokumentace, kromě samotných zdrojových kódů, je k dispozici na stránkách společnosti 4Front Technologies na adrese http://www.opensound.com. Dalším dobrým informačním pramenem je *Linux Multimedia Guide*, viz odkazy na konci tohoto dokumentu.

Jaký je potřeba procesor k plynulému přehrávání/záznamu?

Na tuto otázku neexistuje jednoduchá odpověď, protože to závisí na řadě faktorů:

- zda se používá PCM modulace nebo FM syntéza,
- vzorkovací kmitočet a velikost vzorku,
- aplikace použitá k nahrávání/přehrávání,
- použitá zvuková karta,
- propustnost disku, sběrnice, velikost vyrovnávací paměti, atd...

Obecně by jakákoliv 386 a lepší měla zvládnout přehrávání vzorkovaného zvuku i FM syntézu na 8bitové zvukové kartě.

Přehrávání souborů MOD má vyšší nároky na procesor. Experimentální měření ukazují, že přehrávání se vzorkovací frekvencí 44 kHz vyžaduje více než 40 % procesoru na 486/50, a že 386/25 nezvládne více než 22 kHz (platí pro 8bitové karty). Karty jako Gravis UltraSound přenášejí část funkcí na hardware karty a mají menší nároky na samotný procesor. Výše uvedené platí v případě, že počítač není zatížen jinými operacemi.

Konverze zvukových formátů a vytváření efektů pomocí programů jako je *sox* je výrazně rychlejší, pokud máte matematický koprocesor. Samotný ovladač koprocesor nevyužívá.

Problémy s kartou PAS16 a SCSI adaptérem Adaptec 1542

Linux rozeznává ovladač 1542 pouze na adresách 330 (standardně) nebo 334, PAS umožňuje emulaci MPU-401 pouze na adrese 330. Dokonce i pokud vypnete softwarovou podporu MPU-401, stále dochází ke konfliktu s adaptérem 1542. Stačí přepnout adaptér 1542 na adresu 334.

Navíc 1542 i u PAS-16 používají 16bitové DMA, takže pokud pracujete s 16bitovými stereo vzorky s frekvencí 44 kHz a zároveň zapisujete na SCSI disk na řadiči 1542, vznikají problémy. DMA se překrývají a není dostatek času na obnovování paměti, takže dochází k chybám „PARITY ERROR – SYSTEM HALTED" bez zjevné příčiny. Navíc někteří výrobci, například u páskové jednotky QIC-117, doporučují nastavení časů připojení/odpojení sběrnice tak, že je řadič 1542 aktivní déle než obvykle. Použijte program SCSISEL.EXE od Adaptecu a snižte čas BUS ON nebo zvyšte čas BUS OFF dokud problémy nezmizí. Pak pro jistotu posuňte nastavení ještě o hodnotu výš. Program SCSISEL mění nastavení přímo v paměti EEPROM, takže je trvalé na rozdíl od nastavení v souboru CONFIG.SYS a bude tedy funkční i v Linuxu.

Poslední problém – starší chipsety Symphony výrazně omezovaly časování vstupně-výstupních cyklů, aby se zvýšila rychlost sběrnice. U všech takových desek, které jsem potkal, to ničemu nevadilo, kromě karty PAS-16. Zkuste sehnat program SYMPFIX.EXE, který může tento problém (bez záruky) vyřešit nastavením diagnostického příznaku na řadiči sběrnice. Další možnosti jsou:

- požádat výrobce desky o výměnu čipu řídicího sběrnici,
- vyměnit základní desku,
- vyměnit zvukovou kartu.

Lze současně přehrávat i zaznamenávat zvuk?

Ovladače některých zvukových karet podporují plně duplexní režim. Viz dokumentace na stránkách společnosti 4Front Technologies.

Používám kartu SoundBlaster s IRQ 2, jádro však nelze na tuto hodnotu nastavit

Na počítačích 286 a vyšších je IRQ 2 použito jako kaskádní připojení druhého řadiče přerušení. Je ekvivalentní s IRQ 9.

Pokud používám Linux a pak spustím DOS, objevují se chybová hlášení a/nebo zvukové aplikace nefungují správně

Dochází k tomu při měkkém restartu do DOSu. Někdy chybová hlášení nesprávně poukazují na chyby v souboru CONFIG.SYS.

Většina současných zvukových karet umožňuje softwarové nastavení hodnot přerušení a DMA. Pokud v Linuxu a v DOSu používáte jiná nastavení, může docházet k chybám. Některé karty se nedají znovu nastavit bez úplného restartu.

Jednoduché řešení je restartovat vypnutím a zapnutím počítače nebo tlačítkem reset, nikoliv stiskem Ctrl+Alt+Del.

Systémové řešení je používat v Linuxu i v DOSu stejné přerušení a DMA, anebo nepoužívat DOS :-)

Problémy s DOOMem v Linuxu

Tato poznámka může být zajímavá pro ty, kteří hrají linuxový port DOOM.

Aby zvukový výstup fungoval správně, potřebujete zvukový ovladač verze 2.90 a vyšší, které podporují režim reálného času, využívaný v DOOMu. Zvuky používají 16bitové vzorkování. Pokud máte 8bitovou zvukovou kartu, můžete zvuk provozovat s pomocí některého z programů, které jsou k dispozici na ftp://www.ibiblio.org/pub/Linux/games/doom.

Je-li výkon hry slabý, může pomoci vypnutí zvuku (přejmenováním souboru *sndserver*).

Standardně má DOOM (stejně jako v DOSu) podporu zvuku vypnutou. Programem *musserver* je možné zvuk v Linuxu zapnout. Program můžete získat na adrese ftp://pandora.st.hmc.edu/pub/linux/musserver.tgz.

Jak lze omezit šum?

Používejte kvalitní stíněné kabely, popřípadě vyzkoušejte zvukovou kartu v jiných slotech. Pokud karta umožňuje nastavení hlasitosti, vyzkoušejte různé hodnoty (nejlepší bude zřejmě maximum). Pomocí mixéru zajistěte, že jsou vypnuté všechny nepoužívané vstupy (například mikrofon).

Philipp Braunbeck uvádí, že na zvukové kartě ESS-1868 je přepínač, kterým je možné zapnout vestavěný zesilovač a omezit tak šum.

Na některých systémech pomůže snížení šumu parametr *no-hlt* jádra. Tím se jádru říká, že v nečinných cyklech nemá používat instrukci HALT. Parametr můžete zadat ručně při spouštění systému, nebo můžete uvést parametr *append="no-hlt"* v konfiguračním souboru zavaděče.

Některé zvukové karty jsou prostě špatně stíněné a zeměné, a tedy náchylné k šumu.

Můžu zvuk přehrávat, ale ne zaznamenávat

Pokud přehrávání funguje, a záznam ne, zkuste tyto kroky:

- pomocí mixéru zvolte správné vstupní zařízení,
- pomocí mixéru nastavte citlivost vstupního zařízení,
- je-li to možné, ověřte funkci karty v DOSu, abyste vyloučili hardwarový problém.

Někdy se pro záznam zvuku používá jiný DMA kanál než pro přehrávání zvuku. V takovém případě může být příčina ve špatném nastavení DMA.

Moje „kompatibilní" karta funguje jen s předchozí inicializací v DOSu

Ve většině případů fungují karty „SoundBlaster Compatible" v Linuxu správně s jiným ovladačem, než s ovladačem SoundBlaster. Většina zvukových karet se vydává za „16 bit SB compatible", „SB compatible" a podobně – ve skutečnosti se však jedná pouze o drobné úpravy kvůli kompatibilitě s dosovými hrami. Tyto karty obvykle pracují v nějakém nativním 16bitovém režimu, který je v Linuxu s velkou pravděpodobností podporován.

Pouze u některých (velmi starých) karet je nutné je provozovat v režimu SoundBlaster. Jediné novější karty, které jsou výjimkou z tohoto pravidla, jsou karty založené na čipu Mwave.

Moje „SB 16bit compatible“ karta funguje v Linuxu pouze jako 8bitová

16bitové karty označované jako SoundBlaster kompatibilní jsou opravdu kompatibilní pouze s 8bitovým SoundBlaster Pro. Typicky mají 16bitový režim, který není kompatibilní s režimem SoundBlaster 16 a nepracuje tedy s linuxovým ovladačem.

Lepších výsledků můžete dosáhnout s ovladači MAD16 nebo MSS/WSS.

Kde najdu zvukové aplikace pro Linux?

Následující seznam uvádí několik dobrých archivů linuxových programů:

- ftp://www.ibiblio.org/pub/Linux/kernel/sound/
- ftp://www.ibiblio.org/pub/Linux/apps/sound/
- ftp://tsx-11.mit.edu/pub/linux/packages/sound/
- ftp://nic.funet.fi/pub/Linux/util/sound/
- ftp://nic.funet.fi/pub/Linux/xtra/snd-kit/
- ftp://nic.funet.fi/pub/Linux/ALPHA/sound/

Viz také odkazy na konci dokumentu.

Lze zvukové ovladače přeložit jako moduly jádra?

V současných jádrech jsou zvukové ovladače podporovány jako samostatné moduly.

Podrobnosti viz */usr/src/linux/Documentation/sound*, zejména soubory *Introduction* a *README.modules*.

Můžu zvukovou kartu použít pro výstup pípání konzoly?

Zkuste program *oplbeep* na adrese ftp://www.ibiblio.org/pub/Linux/apps/sound/oplbeep-2.3.tar.gz

Další možnost je program *beep* na adrese ftp://www.ibiblio.org/pub/Linux/kernel/patches/misc/modreq_beep.tgz

Balíček *modutils* obsahuje příklad programu a úpravy jádra, která umožňuje spuštění jakéhokoliv externího programu když jádro požaduje generování zvukového signálu.

Verze 2.0 a vyšší prostředí KDE umožňují přehrávání zvukového souboru místo pípnutí v aplikacích KDE.

Konečně některé zvukové karty umožňují připojit výstup reproduktoru ke kartě, takže veškeré zvuky jdou přes reproduktory zvukové karty.

Co je to VoxWare?

Komerční verze zvukových ovladačů, prodávaných společností 4Front Technologies, byla dříve označována jako *VoxWare*, *USS* (Unix Sound System) nebo *TASD* (Temporary Anonymous Sound Driver). Nyní se označuje jako *OSS* (Open Sound System). Verze dodávaná s Linuxem se někdy označuje jako *OSS/Free*.

Další informace najdete na stránkách společnosti 4Front Technologies na adrese http://www.-opensound.com/.

Sox/play/vplay hlásí „invalid block size 1024“

Změna v ovladači verze 1.3.67 způsobila chybu některých programů, které (chybně) zjišťovaly, zda je výstup volání SNDCTL_DSP_GETBLKSIZE větší než 4 096. Novější verze ovladače byly opraveny a nedovolují alokaci fragmentů kratších než 4 096 bajtů, což řeší problémy starších programů.

Při nahrání modulu ovladače se ztratí nastavení mixéru

Ovladač můžete přeložit jako modul a použít *kerneld* k jeho automatickému zavádění a odstraňování. Tím může vzniknout jeden problém – kdykoliv dojde k nahrání modulu, nastaví se mixér zpět na výchozí hodnoty. U některých karet to může být příliš nahlas (např. SoundBlaster16), u jiných příliš potichu. Markus Gutschke (gutschk@uni-muenster.de) našel následující řešení. V souboru */etc/modules.conf* uveďte následující:

```
options sound dma_buffsize=65536
post-install sound /usr/bin/setmixer igain 0 ogain 0 vol 75
```

Tím způsobíme, že se mixér (v našem případě program *setmixer*) spustí ihned po nahrání ovladače. Parametr *dma_buffersize* je smyšlená hodnota potřebná proto, že příkaz *option* vyžaduje zadání řádkového parametru. Změňte hodnoty podle toho, jaký používáte mixér a jaké úrovně hlasitosti potřebujete nastavit.

Pokud jste ovladač přeložili jako součást jádra a chcete mixér spouštět hned při startu systému, můžete příkaz pro jeho spuštění umístit do spouštěcího skriptu, například */etc/rc.d/rc.local*.

Zvuk může zaznamenávat pouze root

Standardně se soubory zvukových zařízení vytvářejí tak, aby je mohl číst pouze superuživatel. Jedná se o bezpečnostní opatření. V síťovém prostředí se může uživatel vzdáleně přihlásit k počítači se zvukovou kartou a mikrofonem, a odposlouchávat. Pokud vám to nevadí, můžete práva souborů změnit.

Standardní práva umožňují všem uživatelům zvukové soubory přehrávat. Nejedná se o bezpečnostní riziko, může to ale být nepříjemné.

Je podporován zvukový hardware IBM ThinkPad?

Informace o práci se zvukovými kartami MWAVE v laptopech IBM ThinkPad naleznete v souboru */usr/src/linux/Documentation/sound/mwave*, který je součástí zdrojové distribuce jádra. Nezapomeňte, že ne všechny laptopy IBM ThinkPad musí používat čip MWAVE.

Aplikace končí chybou, protože zvuková karta nemá mixér

Některé staré 8bitové karty SoundBlaster nejsou vybaveny mixérem. Některé aplikace potřebují ke své činnosti otevřít mixér a s těmito kartami nefungují. Jens Werner (werner@bert.emv.ing.tu-bs.de) navrhuje následující řešení – vytvořit */dev/mixer* jako odkaz na */dev/null*, pak by mělo všechno fungovat.

Problémy s kartou SB16 CT4170

Jedná se o novější typ karet SoundBlaster, který je vybaven pouze jedním DMA kanálem. Jádro má tedy problémy při použití 16bitového DMA. Řešením je nastavit druhý DMA kanál na 1, pak bude karta fungovat správně.

Jak připojit MIDI klávesy ke zvukové kartě?

Klávesy MIDI-master jsou klávesy bez syntezátoru, jsou vybaveny pouze výstupním MIDI konektorem. Ten lze pomocí vhodného kabelu propojit s 15pinovým MIDI konektorem na většině zvukových karet. Klávesy pak lze použít k ovládání MIDI syntezátorů přímo na zvukové kartě. Přeložte a spusťte (například příkazem *gcc -o prog prog.c*) následující program:

```
#include <fcntl.h>

main()
{
  int fil, a;
  char b[256];

  fil=open("/dev/midi", O_RDWR);
  for(;;)
    {
      a=read(fil, b, 256);
      write(fil, b, a);
    }
}
```

Problémy s IRQ 15 a Ensoniq PCI 128

Problém se projevuje tak, že karta se standardně snaží použít přerušení 15. Toto přerušení je určeno pro sekundární řadič IDE a nelze je sdílet s jinými zařízeními. Potřebujeme nějakým způsobem donutit kartu, aby používala jiné přerušení (například přerušení 11, tak jak to dělá ve Windows).

Funguje následující postup:

a) v BIOSu nastavíme, že počítač nemá PnP operační systém,

b) v BIOSu nastavíme přerušení 15 jako rezervované pro ISA zařízení.

Po spuštění Linuxu by karta měla fungovat. Co se týče Windows, zvuková karta v nich i nadále fungovala bez potíží.

Kde získat MIDI patche pro SoftOSS?

SoftOSS je softwarový tabulkový syntezátor, který je součástí zvukového ovladače karet kompatibilních s Gravis Ultrasound. Ovladače ke své činnosti potřebuje patche pro MIDI. Dokumentace říká, že „zdarma dostupné patche MIDIA jsou k dispozici na různých FTP serverech". SoftOSS není součástí jader 2.4.

Jak říká dokumentace na stránkách společnosti 4Front Technologies, http://www.opensound.com/softoss.html, lze je získat na adrese ftp://archive.cs.umbc.edu/pub/midia/instruments.tar.gz.

Odkazy

Zde uvádíme odkazy na různé další užitečné zdroje informací.

Pokud máte zvukovou kartu, která zároveň podporuje CD-ROM nebo SCSI, mohou vám pomoci dokumenty *SCSI HOWTO* (http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-for-

mats/html_single/SCSI-Generic-HOWTO.html) a *CD-ROM HOWTO* (http://www.ibiblio.org/pub/-Linux/docs/HOWTO/other-formats/html_single/CDROM-HOWTO.html).

Dokument *Sound Playing HOWTO* (http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/Sound-Playing-HOWTO.html) popisuje, jak přehrávat různé typy souborů

Dokument *SoundBlaster AWE mini-HOWTO* (http://www.ibiblio.org/pub/Linux/docs/HOWTO/mini/other-formats/html_single/Soundblaster-AWE.html) popisuje, jak v Linuxu zprovoznit karty SoundBlaster 32 a 64.

Programátorské informace najdete na stránkách společnosti 4Front Technologies, http://www.opensound.com/pguide.

Následující seznamy otázek jsou pravidelně publikovány na Usenetu ve skupině news:news.announce, a archivují se na ftp://rtfm.mit.edu/pub/usenet/news.answers:

- PCsoundcards/generic-faq (obecné otázky ohledně zvukových karet)
- PCsoundcards/soundcard-faq (otázky z konference comp.sys.ibm.pc.soundcard)
- PCsoundcards/gravis-ultrasound/faq (otázky ke kartám Gravis UltraSound)
- audio-fmts/part1 (popis formátu zvukových souborů)
- audio-fmts/part2 (popis formátu zvukových souborů)

V těchto otázkách a odpovědích naleznete také odkazy na konference ke konkrétním produktům a odkazy na různé archivy. Problematice zvuku a/nebo hudby se věnují následující skupiny na usenetu:

- alt.binaries.sounds.* (různé skupiny pro zveřejňování zvukových souborů)
- alt.binaries.multimedia (pro zveřejňování multimediálních souborů)
- alt.sb.programmer (programování karet soundblaster)
- comp.multimedia (problematika multimédií)
- comp.music (teorie počítačové hudby)
- comp.sys.ibm.pc.soundcard.* (různé skupiny o zvukových kartách pro PC)

Webovou stránku věnovanou multimédiím naleznete na adrese http://www.scala.com/multimedia/. Další dobré stránky ohledně zvukových aplikací a MIDI pod Linuxem jsou na adrese http://sound.condorow.net/. Creative Labs mají stránku http://www.creaf.com/.

Poštovní konference Linuxu obsahují řadu „kanálů“ věnovaných různým tématům. Jak se přihlásit zjistíte po zaslání e-mailu s textem „help“ v těle na adresu mailto:majordomo@vger.kernel.org.

Jak už bylo několikrát řečeno, zvukový ovladač v jádře obsahuje řadu souborů Readme, v nichž naleznete spoustu užitečných informací. Typicky je najdete v adresáři */usr/src/linux/drivers/sound*.

Informace o OSS, komerčním zvukovém ovladači pro Linux a další Unixové systémy najdete na webových stránkách společnosti 4Front Technologies na adrese http://www.opensound.com/.

Vynikající místo pro hledání programů pro Linux je *Linux Software Map* na adrese http://www.execpc.com/lsm/. On line verzi najdete na adrese http://www.boutell.com/lsm/.

Další vynikající místo s programy pro Linux je http://www.freshmeat.net/.

Řadu příruček věnovaných Linuxu najdete na stránkách *Linux Documentation Projectu*, http://www.tldp.org. Můžete si je stáhnout, anebo si objednat jejich tištěnou podobu.

A na závěr malá reklama: Pokud se o multimédiích v Linuxu chcete dozvědět mnohem více (zejména na téma aplikací a programování CD-ROM a zvukových karet), doporučuji svou knihu *Linux Multimedia Guide*, vydanou nakladatelstvím O'Reilly and Associates, ISBN 1-56592-219-0.

MP3 v Linuxu

Originál: http://tldp.org/HOWTO/MP3-HOWTO.html

Tento dokument popisuje hardware, software a postupy potřebné ke kódování, přehrávání, mixování a streamování zvukových MP3 souborů v Linuxu.

Úvod

Tento dokument popisuje hardware, software a postupy potřebné ke kódování, přehrávání, mixování a streamování zvukových MP3 souborů v Linuxu. Pokrývá následující témata:

- Vytváření MP3 souborů z živých nebo externích zdrojů
- Vytváření MP3 souborů ze zvukových CD
- Streamování MP3 souborů po síti
- Přehrávání MP3 souborů
- Vytváření MP3 streamů
- Mixování MP3 souborů
- Editaci ID3 záznamů

Hardwarové požadavky a problémy s výkonem

Digitální zpracování zvuku je náročný úkol, který silně závisí na výpočetních a vstupně-výstupních možnostech systému. Jako naprosté minimum doporučuji počítač třídy Pentium.

Pokud budete data kódovat z analogového zdroje přes mikrofonní nebo linkový vstup, vhodná je PCI zvuková karta. Rozdíl ve vstupně-výstupních možnostech PSI a ISA karet je značný, u PCI karet dělá až 132 MB/s (údaj podle *PCI-HOWTO*). Je jasné, že čím kvalitnější zvuková karta (ve smyslu odstupu signál-šum), tím kvalitnější jsou výsledné MP3 soubory. Používal jsem Soundblaster PCI128 a nedávno přešel na Soundblaster Live Value; obě karty byly po výkonnostní stránce dostačující, nicméně série Live má lepší odstup signál-šum, který vyhovuje i pro poloprofesionální práci. Nezapomínejte na zlaté pravidlo jakéhokoliv datového zpracování: garbage in – garbage out!

Společnost Creative Labs poskytuje pro karty Soundblaster Live! linuxový ovladač na adrese http://developer.soundblaster.com/linux/.

Při záznamu analogových zvukových dat na disk, takzvaný *direct to disk* nebo *d2d* záznam, je kritický výkon disku a jeho rozhraní. Pokud používáte IDE systém, doporučujeme režim 4 nebo UDMA, jejichž přenosová rychlost je dostatečná pro bezproblémový a spolehlivý záznam dat.

Ideálním řešením je SCSI systém, jehož disky i rozhraní mají výrazně vyšší propustnost, od 5 MB/s u SCSI 1 po 40 – 80 MB/s u ultra-ultra2/wide SCSI. Špičková rychlost u IDE zařízení může být kdekoliv od 8,3 MB/s po 66 MB/s u Ultra DMA mode 4, jenže tyto rychlosti jsou špičkové a průměrná propustnost je nižší. Pokud si můžete dovolit AV SCSI disk, neváhejte. AV disky mají systém zápisových hlaviček optimalizovaný pro kontinuální přenos; běžné IDE a SCSI disky nejsou obvykle schopny zajistit dostatečný výkon, protože zápisové hlavičky se postupně zahřívají.

Je jasné, že disky s vyrovnávací pamětí poskytují vyrovnanější výkon, protože vyrovnávací paměť pomůže překlenout okamžiky, kdy dochází ke kalibraci hlaviček nebo k přenosovým špičkám.

Pokud parametry disku nevyhovují, bude při záznamu docházet k výpadkům a poruchám, když disk nebude stíhat zápis. Pokud zaznamenáváte jednorázové děje, investujte do slušného SCSI systému.

Další příčinou výpadků při *d2d* záznamech je hodně zatížený systém. Úlohy běžící na pozadí mohou vést ke krátkodobému zablokování systému. Proto je vhodné omezit počet procesů na pozadí, zejména síťových služeb. Podrobnosti o nastavení síťových služeb najdete v příručkách správce systému a správce sítě.

Výpadky mohou vznikat i v důsledku stránkování virtuální paměti, proto používejte co nejvíce paměti RAM. Doporučuji minimálně 32 MB, klidně však můžete použít i mnohem více.

Chcete-li ze svého systému vyždímat maximum, pravděpodobně ničemu neublíží optimalizace jádra.

Při streamování MP3 souborů má vliv na propustnost i síťové rozhraní – jasné je, že 100Mb karta povede k lepším výsledkům než 10Mb karta.

Výše uvedené požadavky definují systém dostatečně vhodný pro kódování zvukových dat. Nebojte se ale použít i starší a slabší počítač, pokud nemáte nic lepšího k dispozici.

Vhodným oříškem pro správce je vyladit slabší systém tak, aby poskytoval dostatečně dobré výsledky. Celkové chování počítače se tím určitě zlepší :-)

Dalším důležitým bodem je kabeláž. Levné nekvalitní kabely a konektory budou mít za následek nízkou kvalitu záznamu. Pokud vaše zvuková karta umožňuje použít gramofonové, takzvané RCA konektory, použijte je. Pomůže i použití pozlacených konektorů a dostatečný odstup mezi zvukovými a datovými kabely, aby nedocházelo ke vzájemnému rušení.

Samozřejmě nemá smysl utrácet hromady peněz za kabely, pokud zbytek systému nestojí za nic.

Při čtení souborů z CD-ROM mechaniky bude mít rychlost a typ mechaniky vliv na to, jak dlouho bude načtení dat trvat. Stařičká jednorychlostní mechanika pravděpodobně uspokojí jen ty nejtrpělivější.

Pokud chcete poslouchat, co vlastně zaznamenáváte, musíte CD-ROM mechaniku propojit se zvukovou kartou – ať už interním kabelem, nebo přes sluchátkový výstup mechaniky. V takovém případě ale nebudete moci poslouchat přehrávané MP3 soubory.

Podrobné informace o nastavení různých zvukových karet naleznete v praktickém návodu „Zvuk na Linuxu“ (kapitola 18).

Softwarové požadavky

Konverze zvuku na MP3 je normálně dvoufázový proces, kdy se zvuk nejprve zaznamená ve formátu WAV a pak se provede jeho konverze do MP3. Některé nástroje umožňují tento proces provést najednou.

Podle toho, z jakého zdroje chcete zvuk kódovat, musíte zvolit vhodný program pro záznam WAV souboru. Pokud chcete například zaznamenávat zvuk z analogového vstupu zvukové karty, potřebujete program, který tento vstup čte a zapisuje do WAV souboru. Následující přehled uvádí některé užitečné nástroje (popisy jsou vesměs přebrány z webových stránek příslušných programů).

Záznam zvuku

Záznam zvuku z analogového vstupu

Wavrec

Program je distribuován jako součást aplikace *wavplay*, kterou můžete získat na adrese ftp://metalab.unc.edu/pub/Linux/apps/sound/players/.

Konverze dat ze zvukového CD do WAV (takzvaný „CD ripping")

CDDA2WAV

http://metalab.unc.edu/pub/Linux/apps/sound/cdrom/

Cdparanoia

Cdparanoia je „Compact Disc Digital Audio (CDDA) extraction tool", běžně označovaný jako „ripper". Program využívá služeb knihovny Paranoia, která provádí celou práci (zdrojové kódy této knihovny jsou součástí distribuce programu *cdparanoia*). Stejně jako původní program *cdda2wav* čte program *cdparanoia* zvukový záznam z CD-ROM přímo jako data, bez konverze na analogový signál, a zaznamenává je jako WAV, AIFC nebo 16bitový PCM signál. V porovnání s *cdda2wav* je *cdparanoia* výrazně pomalejší, poskytuje však nejlepší možné výsledky při čtení z disků, které jsou poškrábané nebo z jiných důvodů obtížně čitelné.

http://www.xiph.org/paranoia/index.html

RipEnc

RipEnc je skriptové rozhraní k programům *cdparanoia*, *cdda2wav*, *tosha* a *Bladeenc*, *8hz-mp3*, *l3enc*. Provádí vyhledávání v CDDB databázi a automatizuje tak pojmenovávání ukládaných skladeb. Nabízí také možnost jejich ručního pojmenování. Můžete číst celé CD, nebo jen vybrané skladby. Podporuje také práci s ID3 údaji.

http://www.asde.com/~mjparme/index.htm

Cd2mp3

cd2mp3 je program, který slouží k obsluze dvou dalších programů – *cdda2wav* (pro záznam stop z CD) a *lame* (pro kódování do MP3).

http://sertaozinho.org/cd2mp3/index.html

RipperX

RipperX je grafický program pro záznam CD a kódování do MP3. Obsahuje rozhraní pro práci s programy *cdparanoia*, *BladeEnc*, *Lame*, *XingMp3enc*, *8hz-mp3* a *ISO v2 encoder*. Podporuje rovněž CDDB a ID3.

http://www.digitallabyrinth.com/linux/ripperX/

Grip

Grip je grafický přehrávač CD s možností záznamu a kódování do MP3. Pro záznam obsahuje vestavěný program *cdparanoia*, umožňuje však použít i externí programy pro záznam (například *cdda2wav*). Obsahuje také rozhraní ke kódovacím programům, takže usnadňuje konverzi CD do MP3. K získávání informací o skladbách používá CDDB. Grip používá DigitalDJ a vytváří tak jednotné „počítačové" rozhraní k hudebním kolekcím.

http://www.nostatic.org/grip/

Kódovací programy

Tyto programy potřebujete ke konverzi formátu WAV do MP3.

Blade's MP3 Encoder

BladeEnc je freewarový kódovací program. Používá stejné kompresní rutiny jako *mpegEnc*, takže můžete čekat přibližně stejnou, případně lepší kvalitu. Hlavní rozdíly jsou ve vzhledu a rychlosti. BladeEnc na rozdíl od mpegEnc neobsahuje pěkné uživatelsky přítulné rozhraní, je však více než třikrát rychlejší a podporuje práci s několika oblíbenými uživatelskými rozhraními.

http://bladeenc.cjb.net

Lame

V úžasné historii volby jmen pro GNU projekty zaujímá LAME čestné místo – LAME znamená „LAME Ain't an Mp3 Encoder", LAME není mp3 kodér. Jedná se o modifikaci zdrojových kódů dist10 ISO, uvolněnou pod licencí GPL. LAME není schopen výroby MP3 streamu, není možné jej ani samostatně přeložit. K jeho činnosti potřebujete i zdrojové kódy ISO. Tyto zdrojové kódy jsou rovněž k dispozici zdarma, k jejich komerční distribuci (včetně distribuce zdarma šířených kódovacích produktů) však může být nutný souhlas FhG (Fraunhofer Gesellschaft, Germany).

http://www.sulaco.org/mp3/

Gogo

Jedná se o velmi rychlý kódovací program pro platformu *x86*, založený na LAME verze 3.29beta a optimalizovaný PEN@MarineCat, Keiichi SAKAI, URURI, kei a shigeo. Pro překlad budete potřebovat také zdrojové kódy NASM, které najdete na adrese http://www.web-sites.co.uk/nasm/.

http://homepage1.nifty.com/herumi/gogo_e.html

Přehrávače

K přehrávání MP3 souborů přirozeně potřebujete přehrávač.

Xmms (dříve označovaný jako X11Amp)

Tento přehrávač obsahuje většinu funkcí známých z přehrávače Winamp pro Windows 95/98/NT, obsahuje však také některé specifické funkce dostupné pouze v Linuxu.

http://www.xmms.org

Xaudio

Xaudio je velice rychlý a robustní multiplatformní přehrávač společnosti Digital Audio, zaměřený zejména na přehrávání MPEG (MP1, MP2 a MP3) zvukových souborů.

http://www.xaudio.com

AlsaPlayer

AlsaPlayer je nový typ PCM přehrávače. Je silně vícevláknový a intenzivně využívá knihovny a ovladače ALSA. Obsahuje některé velice zajímavé funkce jedinečné pro přehrávače na platformě Linux/Unix. Cílem je vytvořit modulární přehrávač všech typů médií se zaměřením na PCM zvuková data. Jako první a jediný přehrávač umožňuje plynulé řízení rychlosti přehrávání – a to jak zrychlování, tak i zpomalování.

http://www.alsa-project.org/~andy/

mpg123

Co je mpg123? Je to rychlý, zdarma dostupný a přenositelný přehrávač MPEG audia pro Unix. Podporuje kódování MPEG 1.0/2.0 typu 1, 2 a 3 (tedy populární MP3 soubory) a byl otestován na celé řadě platforem včetně Linuxu, FreeBSD, NetBSD, SunOS, Solaris, IRIX, HP-UX a dalších. K přehrávání v CD kvalitě (44 kHz, 16 bitů, stereo) je nutný procesor Pentium (nebo rychlý 486), SPARCstation10, DEC Alpha nebo podobný. Přehrávání v monofonním režimu nebo se sníženou kvalitou (22 kHz nebo 11 kHz) je možné i na pomalejších 486.

http://www.mpg123.org

Freeamp

FreeAmp je rozšiřitelný víceplatformní přehrávač. Obsahuje optimalizovanou verzi MPEG dekodéru Xing, což z něj dělá jeden z nejrychlejších a nejkvalitnějších přehrávačů. Obsahuje řadu běžných funkcí, dostupných z jednoduchého a snadno použitelného uživatelského rozhraní.

http://www.freeamp.org/

Streamovací servery

Streamovací servery umožňují „vysílat" MP3 soubory po síti, ať už jde o intranet, nebo přímo Internet.

Icecast

Icecast je systém pro vysílání zvuku v kódování Mpeg Layer III od týmu linuxpower.org. Icecast obsahuje iceplay a icedir. Iceplay je streamer playlistu, který umožňuje předávat předkódovaná data serveru Icecast.

http://www.icecast.org/

Fluid

Fluid Streaming Server je program pro streamování médií po síti ve formátu mp3.

http://www.subside.com/fluid/ (stará adresa)

http://fluid.sourceforge.net/ (nová adresa)

Litestream

Litestream je Open-Source, široce rozšiřitelný vysokokapacitní streamovací systém MP3 pro Unix.

http://www.litestream.net/

Apache::MP3

Modul pro streamování MP3 prostřednictvím webového serveru Apache.

Tento modul převezme adresářovou strukturu obsahující MP3 soubory a nabízí je jako prohlížitelnou knihovnu ke streamování přes web.

[Dostupné na stránkách CPAN]

Mixování

LiveIce

LiveIce slouží jako datový zdroj pro Icecast, v reálném čase provádí kódování MPEG streamů. Na rozdíl od klientů jako Shout nebo IceDJ umožňuje vysílání živého zvuku, ne jen uložených mp3 dat.

LiveIce je součástí balíku Icecast, nové verze a dokumentaci naleznete na adrese:

http://star.arm.ac.uk/~spm/software/liveice.html

eMixer

eMixer je snadno použitelné rozhraní programu mpg123, které umožňuje přehrávat a mixovat dva mp3 streamy. Možnost mixovat zvuk ze dvou zdrojů umožňuje plynulé přechody mezi skladbami a nabízí tak „DJ" funkce přímo z konzoly. eMixer se také hodí pro „real time" party. eMixer je založen na původním mixážním kódu mp3, na němž je vystavěna i mixážní komponenta v programu LiveIce.

http://emixer.linuxave.net/

GDAM

GDAM je balík pro mixování v reálném čase. Umožňuje současné přehrávání a mixování libovolného počtu mp3 souborů. Umožňuje dynamické vkládání a úpravy efektů. GDAM je založen na architektuře klient-server. Veškerý zvuk je vytvářen serverem, který přijímá pokyny od libovolného množství klientů. Mezi další funkce patří doplňky pro vytváření efektů, ukládání a smyčkování, sekvencér, prohlížeč zvuku a kalkulátor rytmu, plynulé skládání (bez mezer mezi skladbami v seznamu), online nápověda, klon programu mpg123 běžící na serveru, pružné řádkové rozhraní pro přímé ovládání serveru, záznam mixovaného výsledku nebo jeho části, podpora více zvukových zařízení a hardwarové řízení MIDI.

http://gdam.org/ nebo http://gdam.sourceforge.net

Editory ID3

id3ed

id3ed je editor ID3 záznamů v mp3 souborech. Jednotlivé texty je možné měnit interaktivně nebo z příkazového řádku, případně kombinovaně. Žánr je možné definovat slovně nebo číselně. Údaje je možné také prohlížet a mazat.

http://www.azstarnet.com/~donut/programs/id3ed.html

mp3info

MP3info je jednoduchý nástroj pro čtení a záznam ID3 údajů v MP3 souborech. Obsahuje textové i grafické rozhraní.

http://metalab.unc.edu/mp3info/

Záznam streamů

Streamripper

Streamripper zaznamenává vysílané streamy. Pokud stream obsahuje údaje o skladbách (metadata), vytváří streamripper pro každou skladbu samostatné soubory.

http://streamripper.sourceforge.net/

Wget

GNU Wget je nástroj pro stahování souborů protokoly HTTP a FTP. Nepracuje interaktivně a umožňuje tak spuštění na pozadí, bez nutnosti přihlášení k počítači. *Wget umožňuje daleko více než jen záznam streamů!*

ftp://ftp.gnu.org/gnu/wget/

Různé

Normalizace hlasitosti

Wavnorm

Pokud pracujete s digitálně zaznamenaným zvukem nebo záznamem ze starších CD, pravděpodobně narazíte na rozdíly v hlasitosti skladeb. Změnu hlasitosti záznamů kódovaných v MP3 umožňuje program *wavnorm*.

http://www.zog.net.au/computers/wavnorm/

SOX

Sox je velice užitečný konverzní nástroj, který by vám rozhodně neměl chybět. Budete jej potřebovat, chcete-li použít program *wavnorm*. ftp://metalab.unc.edu/pub/Linux/apps/sound/convert/.

Možná budete potřebovat i nějaký mixážní program; vhodný je Xmixer, který je součástí většiny distribucí.

Nastavení systému

V této části popisujeme základní informace o nastavení linuxového systému pro záznam zvuku z analogových zdrojů nebo z CD-ROM. Text je založen na mém systému s procesorem Intel a distribucí RedHat, měl by však platit víceméně obecně. Pokud máte nějaké zkušenosti s jinými platformami, kontaktujte autora dokumentu.

Přirozeně hlavním požadavkem je funkční zvuková karta. V této chvíli vás odkážu na vynikající praktický návod „Zvuk na Linuxu“ (kapitola 18) Jeffa Trantera. Dobré čtení je také *Linux Sound Playing HOWTO* od Yoo C. Chunga. Oba dokumenty popisují zprovoznění zvukového systému podstatně podrobněji, než to dokážu já.

Nastavení pro záznam analogového zvuku

Nejprve je nutné připojit zvukový zdroj. Existuje řada způsobů, jak do linuxového systému přivést zvuk, nejběžnější jsou:

Výstup „Line Out“ do konektoru „Line In“ zvukové karty. Většina audiozařízení obsahuje konektor „Line Out“. Napěťová úroveň na tomto konektoru je definována standardem, a pokud si dobře pamatuji, je to 500 mV u domácích a poloprofesionálních zařízení, a 750 mV u profesionálních zařízení. Troufám si hádat, že většina zvukových karet předpokládá napěťovou úroveň 500 mV, některé nové „Pro“ modely pak možná úroveň 750 mV. Nemělo by to mít velký vliv, pokud nezaznamenáváte hodně silný signál.

Výstup „Line Out“ typicky slouží k připojení Hi-Fi zařízení k zesilovači, takže tímto způsobem by měly jít bez problémů připojit magnetofony, tunery, CD přehrávače, přehrávače DAT a minidisků. Složitější je to s gramofony, jak uvidíme dále.

Zvuk můžete přijímat také z videopřehrávače. Většina videí má buď přímo konektor „Line Out“, nebo můžete zvuk získat z konektoru SCART.

Další propojení může být „Tape Out“ zesilovače do „Line In“ zvukové karty a „Line Out“ zvukové karty do „Tape In“ zesilovače. V této konfiguraci vlastně nahradíte normální magnetofon počítačem. Propojení výstupu zvukové karty na vstup zesilovače vám umožní sledovat hlasitost výstupního signálu.

Mikrofon do vstupu „Mic In“ zvukové karty. Napětí generované mikrofonem je výrazně nižší než hodnoty používané na linkových vstupech a výstupech. Pokud byste mikrofon připojili do linkového vstupu zvukové karty, pravděpodobně nezaznamenáte vůbec nic.

> **Upozornění!** Při opačném zapojení (linkový výstup na mikrofonní vstup) může dojít k poškození zvukové karty!
>
> Gramofon do mikrofonního vstupu. Upozorňuji na následující poznámku Marka Tranchanta:
>
> *Přímý výstup z gramofonu má velmi malou napěťovou úroveň. Nelze jej však připojit přímo do mikrofonního vstupu a čekat dobrý výsledek. Výstup gramofonu je nutné zpracovat ekvalizérem, protože nahrávky jsou zaznamenány s potlačením hloubek a zesílením výšek, aby se optimalizoval pohyb jehly gramofonu. Tyto posuny jsou přesně definovány a označují se jako RIAA ekvalizace. Proto je nutné výstup přivést do předzesilovače a pak na linkový vstup.*

Elektronické klávesy a syntezátory je možné připojit přímo do vstupu „Line In“ zvukové karty, kytary pak přes DI (Direct Injection) zařízení, které provede zesílení na potřebnou úroveň.

Než do zvukové karty cokoliv připojíte, zkontrolujte, že hlasitost je nastavena na minimum. Při připojování mikrofonu pak zkontrolujte, zda je vypnutý a není v blízkosti reproduktorů.

Nastavení pro záznam zvuku z CD-ROM

Záznam zvuku z CD-ROM je poměrně prostý. Pokud vám funguje přehrávání zvukových CD v mechanice, lze předpokládat, že budete schopni provádět i jejich záznam.

Další nastavení

Přihlašte se do systému jako normálně a pak mixážním programem nastavte záznamové úrovně tak, aby byly dostatečně hlasité, ale aby nedocházelo ke zkreslení. Běžně provádím toto nastavení „od ucha“, s trochou praxe zjistíte, jaké hodnoty jsou pro vaše zařízení vhodné.

Dále doporučuji buď vypnout všechny nepotřebné služby, nebo přejít do jednouživatelského režimu, zejména při záznamu z externích zdrojů. Tím se zajistí, že systém bude provádět minimum jiné činnosti a vyloučí se tak vznik výpadků při záznamu. Pro záznam zvuku používám vyhrazený SCSI disk, který budu označovat jako */mp3*. Je to dáno zejména výkonem SCSI zařízení. Záznam na vyhrazený disk má výhodu také v tom, že si můžete být téměř jisti, že při záznamu zvuku nepřeskočí hlavičky náhle někam jinam.

Podrobnosti o použití více disků v linuxovém systému uvádí dokument *Multi-Disk-HOWTO* Steina Gjena.

Záznam z externího zdroje

Nejprve zkontrolujte, zda máte na disku dostatek místa. Při záznamu v CD kvalitě (16bitové stereo, 44,1 kHz) potřebujete na jednu minutu necelých 10 MB (5 MB na kanál).

Obvykle používám DAT kvalitu, tedy 16bitové stereo, 48 kHz.

Při použití programu *wavrec* zadáte následující příkaz:

```
/usr/local/bin/wavrec -t 60 -s 48000 -S /mp3/temp.wav
```

První část příkazu je explicitně definovaná cesta k programu *wavrec*. Parametr „-t 60“ definuje délku záznamu v sekundách. Třetí část, „-s 48000“, definuje vzorkovací frekvenci ve vzorcích za sekundu. (48000 je frekvence DAT záznamu, 44100 frekvence CD záznamu). Poslední část představuje cestu k výstupnímu souboru.

Úplný seznam parametrů získáte příkazem *wavrec -help*, nebo na manuálové stránce.

Tímto postupem získáme WAV soubor, který následně konvertujeme na MP3.

Při konverzi programem *bladeenc* použijeme příkaz:

```
/usr/local/bin/bladeenc zdrojový_soubor cílový_soubor -br 256000
```

Parametr -br nastavuje hodnotu „bit rate“, používám 256 kb/s. Nezapomeňte, že ve vašem systému se může cesta k programu *bladeenc* lišit. Úplný seznam parametrů získáte příkazem *bladeenc -help*, což je sice neplatný parametr, ale vypíše seznam platných.

Pokud chcete kódování provést programem *lame* (nebo *gogo*, který je na Lame založen), zadejte:

```
/usr/local/bin/lame zdrojový_soubor cílový_soubor -b 256
```

Záznam z CD-ROM

Podobně jako při záznamu externího zvuku je i záznam z CD-ROM dvoufázový proces. Nejprve dojde ke čtení dat z CD a k jejich záznamu do WAV souboru. Pak se provede konverze WAV souboru do MP3.

Typicky existují dva typy kódovacích programů – textové a grafické. Oba dělají to samé, grafické se ale snáze používají a jsou hezčí.

Řádkové kódování

Napsal jsem jednoduchý skript v Perlu, který přečte a zakóduje skladby z CD.

```
#!/usr/bin/perl

if ($ARGV[0] ne "") {

$count = 1;

do {

$cdcap = system("cdparanoia", $count, "/mp3/cdda.wav");
$track = "$ARGV[1]/track".$count.".mp3";
$enc = system("bladeenc  /mp3/cdda.wav $track -br 256000");
$count++;

}
until $count  $ARGV[0];
exit;
}

else {
print "Usage cdriper [no of tracks] [destination directory]\n\n";
}
```

Poznámka: Skript je velice jednoduchý a neobsahuje žádné vychytávky jako je kontrola chyb nebo práce s CDDB. Upravte si jej dle svého :-)

Nejdůležitější části jsou:

```
$cdcap = system("cdparanoia", $count, "/mp3/cdda.wav");
```

Tento řádek volá program pro čtení CD, *cdparanoia*. Tento program čte data z CD a zapisuje je do formátu WAV. Používám program *cdparanoia*, pokud byste ale chtěli použít *cdda2wav*, upravte řádek takto:

```
$cdcap = system("cdda2wav", $count, "/mp3/cdda.wav");
```

Parametry jsou *$count*, udávající počet zaznamenávaných stop, a dále cesta k výstupnímu souboru. V mém případě používám záznam do dočasného adresáře na SCSI disku, rezervovaném pro MP3.

Konverzi WAV souboru do MP3 provedeme programem Bladeenc.

Tento skript jsem vytvořil kvůli záznamu CD bez nutnosti samostatně zapisovat a kódovat každou stopu a bez nutnosti použít dávkový režim programu *cdparanoia*. Tím se redukují požadavky na volný diskový prostor, protože *cdparanoia* načte celý disk a spotřebuje tak až 600 MB místa.

Pokud budete chtít použít Lame nebo Gogo, bude příkaz vypadat takto:

```
$enc = system("lame  /mp3/cdda.wav $track -b 256");
```

nebo

```
$enc = system("gogo  /mp3/cdda.wav $track -b 256");
```

Dále uvádíme seznam parametrů jednotlivých kódovacích programů.

Bladeenc

Použití: *bladeenc [globální_přepínače] vstup1 [výstup1 [parametry1]] vstup2 ...*

Obecné přepínače

-[kbit], -br [kbit]	Nastavení bitrate. Implicitně 128 kb/s (64 u mono záznamu).
-crc	Vytvoří MP3 s kontrolním součtem.
-delete, -del	Po zakódování smaže vstupní data.
-private, -p	U výstupního souboru nastaví privátní příznak.
-copyright, -c	U výstupního souboru nastaví příznak copyrightu.
-mono, -dm	Vytvoří monofonní záznam kombinací obou kanálů.
-leftmono, -lm	Vytvoří monofonní záznam z levého kanálu.
-rightmono, -rm	Vytvoří monofonní záznam z pravého kanálu.
-swap	Prohodí levý a pravý kanál.
-rawfreq=[freq]	Definuje vzorkovací frekvenci u čistého vzorku. Implicitní je 44 100.
-rawbits=[bits]	Definuje počet bitů na kanál u čistého vzorku. Implicitní je 16.
-rawmono	Udává, že vzorek je monofonní, ne stereofonní.
-rawstereo	Udává, že vzorek je stereofonní (implicitní chování).
-rawsigned	Udává, že vzorek je podepsaný (implicitní chování).
-rawunsigned	Udává, že vzorek je nepodepsaný.

-rawchannels=[1|2] Definuje počet kanálů ve vzorku, ekvivalentní s parametry *rawmono*, respektive *rawstereo*.

Globální přepínače

-quit, -q Po dokončení ukončí program bez čekání na stisk klávesy.

-outdir=[dir] Ukládá výsledné soubory do zadaného adresáře.

-quiet Nevypisuje nic na obrazovku.

-nocfg Nepoužívá nastavení z konfiguračního souboru.

-prio=[prio] Nastavuje prioritu procesu, platné hodnoty jsou HIGHEST, HIGHER, NORMAL, LOWER, LOWEST (implicitně) a IDLE.

-refresh=[rate] Frekvence obnovování indikátoru postupu.

-progress=[0-8] Typ použitého indikátoru. 0=žádný, 1=standardní.

Vstupní a výstupní soubory je možné nahradit STDIN a STDOUT.

Lame

Použití: *lame [parametry] vstupní_soubor [výstupní_soubor]*

Místo vstupního a výstupního souboru je možné zadat „-“, což znamená stdin/stdout.

Parametry

`-m mode` (S)tereo, (j)oint, (f)orce nebo (m)ono. Implicitní je *j. force* znamená vynucení stereo režimu; rychlejší a využívá speciální režim maskování.

`-b bitrate` Nastavuje bitrate, implicitní je 128 kb/s (u VBR nastavuje povolenou minimální hodnotu).

`-s freq` Vzorkovací frekvence vstupního souboru (v kHz) – implicitní hodnota je 44,1.

`--resample freq` Vzorkovací frekvence výstupního souboru (v kHz) – implicitní hodnota je stejná jako u vstupního souboru.

`--mp3input` Vstupní soubor je MP3.

`--voice` Experimentální režim mluveného slova.

`-v` Použít proměnný bitrate (VBR).

`-V n` Nastavení kvality pro VBR, implicitní hodnota *n*=4. 0=vysoká kvalita, velké soubory, 9=nízká kvalita, malé soubory.

`-t` Vypíná informativní příznak VBR.

`--nohist` Vypíná zobrazení VBR histogramu.

`-h` Používá (možná) vylepšení kvality.

`-f` Rychlý režim (nízká kvalita).

`-k` Vypíná ořezání při sfb=21.

`-d` Povoluje různé typy bloků u jednotlivých kanálů.

`--athonly` K maskování používá pouze ATH.

`-r` Vstup je čistá PCM.

`-x` Provede přehození bajtů vstupních dat.

`-a` Převádí stereozáznam na monozáznam.

`-e emp`	Ruší zvýraznění n/5/c (zastaralá volba).
`-p`	Chybová ochrana. Ke každému rámci přidává 16bitový kontrolní součet.
`-c`	Označí data příznakem copyrightu.
`-o`	Označí data jako nepůvodní.
`-S`	Potlačí zobrazení indikátoru průběhu.

Následující parametry doplňují ID3 data:

`--tt název`	Název skladby (max. 30 znaků).
`--ta interpret`	Interpret skladby (max. 30 znaků).
`--tl album`	Album z nějž skladba pochází (max. 30 znaků).
`--ty rok`	Rok vzniku skladby (max. 4 znaky).
`--tc komentář`	Další údaje (max. 30 znaků).

Vzorkovací frekvence pro MPEG1 (kHz): 32 44,1 48

bitrate (kb/s): 32 48 56 64 80 96 112 128 160 192 224 256 320

Vzorkovací frekvence pro MPEG2 (kHz): 16 22,05 24

bitrate (kb/s): 8 16 24 32 40 48 56 64 80 96 112 128 144 160

Gogo

Použití: *gogo vstupní_soubor [výstupní_soubor] [parametry]*

Parametry:

`-b kbps`	bitrate [kbps]
`-br bps`	bitrate [bps]
`-silent`	nevypisuje indikátor průběhu
`-off`	{3dn,mmx,kni(sse),e3dn}
`-v {0..9}`	VBR, 0=nejvyšší kvalita, 9=nejvyšší komprese. Doporučuje se kombinovat s parametrem *-b* v případě čistého PCM vstupu
`--offset bytes`	přeskočit hlavičku souboru
`-8bit`	8bitová PCM (implicitně 16bitová)
`-mono`	mono PCM (implicitně stereo)
`-bswap`	záměna vyššího a nižšího bajtu u 16bitové PCM
`-s kHz`	frekvence PCM (implicitně 44,1 kHz)
`-nopsy`	vypnutí psychoakustiky
`-m {s,m,j}`	výstupní formát: s=stereo, m=mono, j=spojené stereo
`-d kHz`	změna vzorkovací frekvence výstupního MP3
`-emh {n,c,5}`	potlačení zvýraznění
`-lpf {on,off}`	16kHz filtr (implicitně se používá, je-li vstupní frekvence <= 128 kbps)
`-test`	testovací režim
`-delete`	smaže vstupní soubor po skončení kódování

RipEnc

Program RipEnc dělá to samé, jako výše uvedený kód, je ale napsán v shellu a snáze se používá. Ukážeme si, jak vypadá:

```
RipEnc version 0.7, Copyright (C) 1999  Michael J. Parmeley
<mjparme@asde.com, RipEnc comes with ABSOLUTELY NO WARRANTY

There is currently NO encoding process running in the background
Your encode.log file is 982607 bytes long.

<Enter 'd' for details, 'v' to view the encode log, or 'del' to delete the
encode log

1) Change working directory.....................[/megajukebox/tmp]
2) Choose encoder...............................[lame]
3) Choose ripper................................[cdparanoia]
4) Choose id3 tool..............................[none]
5) Toggle between Manual and CDDB naming.......[manual]
6) Setup XMCD_LIBDIR variable for CDA..........[/var/X11R6/lib/xmcd]
7) Set preferred naming convention..............[artist-name_of_song.mp3]
8) Rip whole CD?................................[no]
9) Set small hard drive option?.................[no]
10) Please select your Cd-Rom device............[/dev/cdrom]
11) Set the Bitrate for the encoded MP3's......[256]
12) List the files in your working directory
13) Start
14) About
15) Exit
?
```

CD2MP3

Cd2mp3 je jednoprůchodový program pro kódování CD do formátu MP3. Vytváří MP3 soubory jednotlivých stop bez dočasných WAV souborů. Takto vypadá v činnosti:

```
$ cd2mp3 options ALL
Using language: 1 - English.
Translator: Gustavo Sverzut Barbieri(k-s) <gsbarbieri@hotmail.com

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Cd2Mp3 1.0 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Author: Gustavo Sverzut Barbieri (k-s) <gsbarbieri@hotmail.com.br
*** device: -D/dev/cdrom        type: -Icooked_ioctl
*** audio device: /dev/dsp
*** preset: tape
*** copyright: Yes
*** author:
*** album:

recording: '1' as 'track-1.mp3' (wait)
        Ok! (recorded)
recording: '2' as 'track-2.mp3' (wait)
```

Samozřejmě můžete použít řádkové parametry. Takto vypadá jejich seznam:
Použití:

cd2mp3 parametry číslo_stopy="název_stopy" ... číslo_stopy=„název_stropy"

nebo

cd2mp3 parametry ALL (k záznamu všech stop)

nebo

cd2mp3 parametry LIST=seznam (čte dvojice stopa/název ze souboru)

nebo

cd2mp3 parametry PLAY=číslo_stopy VOL=[0..100] (pouze přehrává)

Parametry:

DEV=zařízení_cdrom	implicitně /dev/cdrom
DEV_TYPE=[generic_scsi | cooked_ioctl]	implicitně cooked_ioctl
AUDIO_DEV=audio_zařízení	implicitně /dev/dsp
LANGUAGE=číslo_jazyka	LANGUAGE=help zobrazí podporované jazyky
COPYRIGHT=[YES | NO]	
PRESET=[phone | voice | fm | tape | hifi | cd | studio]	PRESET=help zobrazí nápovědu
ALBUM=„název_alba"	AUTHOR=„jméno_interpreta"

Grafické kódovací programy

Grafické kódovací programy nabízejí stejné funkce jako řádkové, obalují je však do pěkného a snadno použitelné rozhraní. Příkladem mohou být Grip nebo RipperX, které se chovají velmi podobně. Oba nabízejí možnost zaznamenat jednu stopu, více stop, nebo celé album. Podporují také práci s CDDB, což umožňuje načíst informace o albu a jednotlivých stopách ze serveru a není je tak nutné zadávat ručně.

Výkon kódovacích programů

V části popisující kódovací programy jsem mluvil o třech – *bladeenc*, *lame* a *gogo*. Hlavní rozdíl mezi nimi spočívá v rozdílném výkonu při kódování (i když jsou mezi nimi i drobné rozdíly v parametrech, jimiž se řídí). Malý příklad. Načetl jsem jednu stopu z CD a nechal ji zakódovat všemi třemi programy. Podmínky byly ve všech případech stejné, výstupem byl stereofonní mp3.

```
[dj@megajukebox]$ ls -l cdda.wav
-rw-rw-r--   1 dj       dj       59823164 Feb 10 00:56 cdda.wav

[dj@megajukebox]$ bladeenc cdda.wav -br 256

BladeEnc 0.91    (c) Tord Jansson          Homepage: http://bladeenc.mp3.no
================================================================================
BladeEnc is free software, distributed under the Lesser General Public License.
See the file COPYING, BladeEnc's homepage or www.fsf.org for more details.

Files to encode: 1

Encoding:  ../test.wav
Input:     44.1 kHz, 16 bit, stereo.
```

```
Output:    128 kBit, stereo.

Completed. Encoding time: 00:05:58 (0.78X)

All operations completed. Total encoding time: 00:05:58

-------------------------------------------------------------------------------

[dj@megajukebox]$ lame cdda.wav -b 256
LAME version 3.50 (www.sulaco.org/mp3)
GPSYCHO: GPL psycho-acoustic model version 0.74.
Encoding ../test.wav to ../test.wav.mp3
Encoding as 44.1 kHz 128 kbps j-stereo MPEG1 LayerIII file
    Frame          |  CPU/estimated  |  time/estimated | play/CPU |   ETA
 10756/ 10756(100%)| 0:02:28/ 0:02:28| 0:02:29/ 0:02:29|    1.9074| 0:00:00

-------------------------------------------------------------------------------

[dj@megajukebox]$ gogo cdda.wav -m s -b 256
GOGO-no-coda ver. 2.24 (Feb 12 2000)
Copyright (C) 1999 PEN@MarineCat and shigeo
          Special thanks to Keiichi SAKAI, URURI, Noisyu and Kei
MPEG 1, layer 3 stereo
inp sampling-freq=44.1kHz out sampling-freq=44.1kHz bitrate=256kbps
inp sampling-freq=44.1kHz out sampling-freq=44.1kHz bitrate=128kbps
input  file `../test.wav'
output file `../test.mp3'
{  10751/  10755} 100.0% (  2.94x)  re:[00:00:00.03] to:[00:01:35.42]
End of encoding
time=  95.430sec
```

Ukazuje se, že Gogo používá podstatně lépe optimalizované kompresní algoritmy než Bladeenc a Lame.

Streamování MP3

Streamovací server umožňuje vysílat MP3 soubory po TCP síti. Sítí může být přímo Internet, nebo nějaká lokální síť či intranet.

Principy činnosti jsou podobné jako u webových serverů, soubory se přenášejí, jakmile se nějaký klient (přehrávač MP3) připojí k serveru.

Icecast

Po stažení a rozbalení toho programu je rozumné podívat se do adresáře *doc*, kde se nachází užitečný a vyčerpávající manuál ve formátu HTML.

Pokud jste si stáhli zdrojový kód, držte se při překladu instrukcí platných pro váš systém. Icecast nebude správně pracovat, pokud v konfiguračním souboru */etc/icecast.conf* nebude nastaveno jméno serveru. To musí odpovídat jménu, zjištěnému zpětným překladem adresy IP.

Pokud při startu uvidíte následující hlášení, něco není v pořádku:

```
- [05/Jan/2000:17:21:04] WARNING: Resolving the server name [your.server.name]
  does not work!
```

Upravte soubor */etc/icecast.conf*, ve kterém najdete řádek „server_name“. Zde zadejte název vašeho systému. Pokud jej nevíte, můžete použít příkaz *hostname*, nebo se podívat do souboru */etc/hosts*.

Po provedení příslušných změn musíte konfigurační soubor buď zkopírovat do adresáře *bin*, anebo můžete *icecast* spustit s parametrem *-c*, který udává umístění konfiguračního souboru:

```
[dj@megajukebox bin]$ ./icecast -c ../etc/icecast.conf
```

Pokud je vše nastaveno správně, mělo by se objevit něco takového:

```
[dj@megajukebox bin]$ ./icecast -c ../etc/icecast.conf -d /home/dj/mp3/icecast/
Icecast Version 1.3.0 Starting...
Icecast comes with NO WARRANTY, to the extent permitted by law.
You may redistribute copies of Icecast under the terms of the
GNU General Public License.
For more information about these matters, see the file named COPYING.

[05/Jan/2000:18:36:30] Icecast Version 1.3.0 Starting..
[05/Jan/2000:18:36:30] Using stdin as icecast operator console
[05/Jan/2000:18:36:30] Tailing file to icecast operator console
[05/Jan/2000:18:36:30] Server started...
[05/Jan/2000:18:36:30] Listening on port 8000...
[05/Jan/2000:18:36:30] Using [megajukebox] as servername...
[05/Jan/2000:18:36:30] Max values: 1000 clients, 1000 clients per source, 10
sources, 5 admins
[05/Jan/2000:18:36:30] [Bandwidth: 0.000000MB/s] [Sources: 0] [Clients: 0]
[Admins: 1] [Uptime: 0 seconds]
```

Parametrem *-d* se definuje adresář s logovacími soubory a šablonami. Dále uvádíme seznam všech řádkových parametrů.

-c soubor	Definuje umístění konfiguračního souboru. Jakékoliv řádkové parametry uvedené po tomto parametru přepíší nastavení konfiguračního souboru. Kromě toho se vždy zpracovává soubor *icecast.conf* v aktuálním adresáři, takže budou použity všechny v něm uvedené volby, které nejsou přepsány nastaveními v novém souboru.
-P port	Port používaný všemi klienty, zdroji, a správcem. Implicitně je to 8 000.
-m max_klientů	Maximální počet klientů, které je možné obsluhovat. Při dosažení tohoto limitu budou další klienti odmítáni s hlášením „HTTP/1.0 504 Server Full“.
-p heslo_kodéru	Nastavuje heslo, které musí kodér uvést, any mohl dodávat data serveru. Pokud jste server přeložili s podporou kryptování, musí být parametrem zašifrovaný řetězec.
-b	Spustí server na pozadí, tedy jako démona. Administraci serveru je pak možné provádět připojením se na server pomocí nějakého *telnet* klienta.
-d adresář	Všechny ukládané logy a prohledávané šablony se ukládají relativně k tomuto adresáři.

Server je nyní spuštěn, teď k němu musíme připojit zdroj MP3 dat. Data můžete serveru doručovat dvěma programy – Shout nebo LiveIce.

Shout

Program Shout poskytuje serveru statický seznam MP3 souborů. Je součástí distribuce Icecast.

Seznam MP3 souborů, které má server nabízet, vytvoříte následujícím způsobem:

```
find [adresář s MP3] -name '*.mp3' -print>playlist
```

Nejjednodušší způsob spuštění služby *shout* je následující:

```
[dj@megajukebox bin]# ./shout megajukebox -P hackme -p playlist
```

Parametr -P udává heslo potřebné pro připojení k serveru Icecast, v našem případě je nastaveno na *hackme*... Rozhodně doporučuji použít něco jiného :-) Parametr -p definuje umístění seznamu skladeb.

Dále uvádíme seznam řádkových parametrů:

Použití: *shout název_serveru [parametry] [[-b bitrate] soubor.mp3] ...*

-B adresář	Adresář se všemi soubory programu shout
-C soubor	Konfigurační soubor
-D dj_soubor	Spustí se před každou skladbou
-P heslo	Heslo pro připojení k serveru
-S	Vypíše všechna nastavení a skončí
-V	Použije „ukecaný“ výstup
-X desc	Použije zadaný deskriptor
-a	Použije automatickou korekci přenosové rychlosti
-b bitrate	Použije nastavenou přenosovou rychlost
-d	Aktivuje funkci „dj“
-e port	Port serveru, na nějž se připojuje
-f	Přeskakuje soubory, které mají jiný bitrate
-g žánr	Použije zadaný žánr
-h	Vypíše nápovědu
-i	Použije starý formát hlaviček
-k	Nezkracuje interní seznam skladeb
-l	Nekonečný běh (ve smyčce)
-m mount	Použije zadaný připojovací bod
-n name	Použije zadaný název
-o	Vypíná automatickou detekci přenosové rychlosti
-p playlist	Načte seznam skladeb ze souboru
-r	Náhodné pořadí přehrávání skladeb
-s	Neposílá adresářovému serveru metadata
-u url	Použije zadané URL
-v	Zobrazí číslo verze
-x	Neaktualizuje soubor stop (šetří procesor)
-z	Přepne se na pozadí
-t	Zapíná vysílání názvů skladeb

LiveIce

LiveIce může pracovat ve dvou režimech – buď serveru IceCast předává statické skladby, nebo může předávat živý zvuk z externího zdroje.

Po rozbalení programu a přečtení souboru README ověřte, zda máte nainstalován program mpg123, protože LiveIce jej potřebuje ke své činnosti.

Existují dva způsoby, jak program LiveIce nakonfigurovat. Buď ruční editací konfiguračního souboru, nebo pomocí grafického konfiguračního nástroje.

Nejpodrobnější informace o parametrech v souboru *liveice.cfg* naleznete přímo na webových stránkách programu.

Dále uvádím kopii svého nastavení programu LiveIce, kdy funguje v mixážním režimu (přenáší data ze statického seznamu MP3 souborů).

```
# liveice configuration file
# Automatically generated

SERVER megajukebox              # Název serveru, MUSÍ SOUHLASIT S REVERZNÍM
ZÁZNAMEM DNS
PORT 8000                       # Port na němž IceCast běží

NAME Megajukebox                # Název serveru, který se předává přehrávačům
                                  MP3 a
                                # adresářovým serverům. Např. "Sarah FM" nebo
                                # "ThisTown: Loud and Heavy Jazz - Internet Ra-
                                  dio 24/7"

GENRE Live                      # Informace o žánru, např. 'Talk' nebo 'Dance'

DESCRIPTION                     # Informace o stanici, např. 'The best for reg-
                                  gae in the North'

URL http://megajukebox:8000     # URL a port serveru

PUBLIC 0                        # Nastavte na 1, pokud má Icecast oznamovat vaši
                                  stanici adresářovým serverům

XAUDIOCAST_LOGIN                # buď ICY_LOGIN nebo X_AUDIOCAST_LOGIN. Lepší je
                                  X_AUDIOCAST

MOUNTPOINT /techno              # Připojovací bod streamu. Používá se pouze je
                                  -li zapnuto
                                # X_AUDIOCAST, jinak je implicitně icy_0

PASSWORD hackme                 # Heslo správce

SAMPLE_RATE 44100               # Vzorkovací frekvence streamu
STEREO                          # MONO nebo STEREO

NO_SOUNDCARD                    # viz níže

HALF_DUPLEX                     # Duplexní režim zvukové karty, buď HALF_DUPLEX
                                  nebo FULL_DUPLEX
```

```
USE_GOGO                        # Volba použitého kodéru, více v README souboru
BITRATE 128000                  # Nastavuje bitrate streamu, viz níže
VBR_QUALITY 1                   # Nastavuje proměnný bitrate

MIXER                           # viz níže

PLAYLIST /megajukebox/playlist # Umístění seznamu souborů

TRACK_LOGFILE track.log         # Umístění logu vysílaných souborů
```

Po nastavení konfiguračního souboru můžete spustit LiveIce takto:

```
[dj@megajukebox liveice]$ ./liveice
/megajukebox/playlist
1
opening connection to megajukebox 8000
Attempting to Contact Server
connection successful: forking process
opening pipe!...
writing password
Setting up Interface
Soundcard Reopened For Encoding
Input Format: 16Bit 44100Hz Stereo
Output Format: 256000 Bps Mpeg Audio
IceCast Server: megajukebox:8000
Mountpoint: /techno
Name: megajukebox - this and that radio - broadcasting 24/7
Genre: Techno
Url: http://megajukebox.com
Description: a load of digital noise - but i know you like it :)

 Press '+' to Finish
adding /megajukebox/demotunes/track_1.mp3
adding /megajukebox/demotunes/track_2.mp3
adding /megajukebox/demotunes/track_3.mp3
adding /megajukebox/demotunes/track_4.mp3
/megajukebox/demotunes/track_4.mp3
Adding New Channel 1
Adding New Channel 2
Channel 1 selecting
 /megajukebox/demotunes/track_1.mp3
Channel 2 selecting
 /megajukebox/demotunes/track_1.mp3
Playing track_1.mp3
searching for Id3v2
searching for Id3v1
copying the data
fixing the nulls
adding the url
closing input file
Using log track.log
```

Poslední řádek je indikátor zatížení. Spuštěný server v režimu mixéru je možné ovládat klávesami:

Akce	pro kanál 1	pro kanál 2
přechod na další stopu	1	a
přechod na předchozí stopu	q	z
zapne/vypne kanál	2	s
reset kanálu	w	x
zvýšení hlasitosti kanálu	3	d
snížení hlasitosti kanálu	e	c
zvýšení rychlosti kanálu	4	f
snížení rychlosti kanálu	r	v
sticky režim zap/náhodný/vyp	5	g
náhled kanálu	t	b
náhodná stopa	u	m

Výše uvedený příklad konfigurace spouští LiveIce v režimu přehrávání statických souborů. Pokud chcete vysílat živý zvuk, změňte hodnotu MIXER na NOMIXER a hodnotu NO_SOUNDCARD na SOUNDCARD.

Nevhodná nastavení mohou vést k zajímavým chybovým hlášením :-)

```
946:Error: Line In mode *and* no soundcard??????? Eeejit!
```

Po připojení vnějšího zdroje a správném nastavení všeho možného by měl server fungovat :-)

```
[dj@megajukebox liveice]$ ./liveice
/megajukebox/playlist
0
Initialising Soundcard
16Bit 22050Hz Stereo Full Duplex
opening connection to megajukebox 8000
Attempting to Contact Server
connection successful: forking process
opening pipe!...
writing password
Setting up Interface
Soundcard Reopened For Encoding
Input Format: 16Bit 22050Hz Stereo
Output Format: 32000 Bps Mpeg Audio
IceCast Server: megajukebox:8000
Mountpoint: /daves_band_live_at_the_club
Name: megajukebox - Dave and the Dynamite - Live at the Roxy
Genre: Live/Rock
Url: http://megajukebox
Description: megajukebox::Louder than a frog in a trashcan..... and almost as
musical

 Press '+' to Finish
Lvl: L:   8704 R:  11776
```

Poslední řádek ukazuje sílu signálu. Pokud je vstupní signál příliš silný, objeví se varování. Můžete-li, snižte v takovém případě hlasitost externího zdroje.

Pokud jste se dívali pozorně, mohli jste si na začátku konfiguračního souboru všimnout hlášení, že byl vygenerován automaticky. Použijete-li konfigurační nástroj *liveiceconfigure.tk*, přijdete o změny, které jste v souboru provedli ručně.

Fluid

Po rozbalení zdrojového archivu si přečtěte soubor README :-)

Program Fluid nabízí tři režimy činnosti – transmit, relay a forward. My se budeme zabývat pouze režimem transmit.

Konfigurační soubory týkající se vysílání jsou uloženy v *config/MP3TX.cfg*. K otestování můžete server spustit následujícím příkazem, v tomto okamžiku bude vyhovovat implicitní konfigurace:

```
java Fluid TX
```

Je jasné, že musíte mít nainstalovánu nějakou Javu. Můžete použít buď portaci JDK od Blackdownu, http://www.blackdown.org, nebo, pokud používáte RedHat, pak Kaffe.

Fluid se dodává s několika ukázkami MP3 souborů, takže pokud je všechno v pořádku, měli byste vidět něco takového (spustil jsem server pomocí programu *kaffe*, vy můžete použít program *java*):

```
[dj@megajukebox Fluid-Beta2J]$ kaffe Fluid tx
------- Fluid Streaming Server Beta 2 -------
This program is ShareWare(tm) and it will not
be crippled in any way because of it. However
if you do like the program and will use it
commercial purposes, we ask of you to contact
us at the address below for pricing info:

 Eldean AB                  E-mail:
 Sjoangsvagen 7             fluid@subside.com
 S-192 72 Sollentuna
 SWEDEN

   Fluid is Copyright Subside (C) 1998
       written by Lars Samuelsson
         http://www.subside.com
-----------------------------------------------

* Transmission mode *
Reading config from: config/MP3TX.cfg
Reading playlist: playlist.m3u
Server started on port: 2711
Accepting administrator login on port: 2710
P| Dr. Nick - Hello Everybody
```

Pokud jste se dostali až sem, jistě budete chtít vysílat i více než jen ukázkové soubory! Budete muset vytvořit seznam MP3 souborů, které chcete přehrávat. Půjde o statický seznam, uživatelé jej nebudou moci měnit ani nebudou moci požadovat jednotlivé soubory. Seznam se jmenuje *playlist.m3u* a standardně je uložen v kořenovém adresáři. Chcete-li sestavit seznam všech MP3 souborů v daném adresáři nebo disku, použijte následující příkaz:

```
find [MP3 directory] -name "*.mp3" -print>playlist.m3u
```

Server standardně používá port 2 711, pokud jej chcete změnit, upravte konfigurační soubor. Server je možné spravovat vzdáleně, implicitně na portu 2 710:

```
[dj@megajukebox Fluid-Beta2J]$ telnet localhost 2710
Trying 127.0.0.1..megajukebox
Connected to localhost.localdomain.
Escape character is '^]'.
jaguar
You are connected to the -Fluid- Streaming Server
Type "help" for a command reference
help
The following commands are available:
 help conn curr exit
curr
Information about the currently broadcasted song:
Title:   Beer Talk
Artist:  Homer Simpson
Album:   The Simpsons
Year:    1996
Comment: Borrowed this as an example
Genre:   Comedy
```

Slovíčko *jaguar* je implicitní heslo správce. Neobjevuje se žádná výzva k zadání hesla, takže na ni zbytečně nečekejte. Doporučuji heslo změnit na nějakou jinou hodnotu, jinak se vám do serveru rychle někdo naboural. Změnu provedete v konfiguračním souboru, který vypadá takto:

```
[dj@megajukebox config]$ cat MP3TX.cfg
2711
2710
5
4096
32
1000
jaguar
playlist.m3u
current.txt

# JednotlivØ ŁÆdky znamenaj□
# 1. ¨□slo portu serveru
# 2. ¨□slo portu pro vzdÆlenou sprÆvu
# 3. MaximÆln□ poŁet klientø
# 4  Velikost paketů při čtení/posílání
# 5. Přenosová rychlost v kb/s (všechny mp3 musí mít stejnou)
# 6. Pauza mezi skladbami (v ms)
# 7. Heslo pro vzdálenou správu
# 8. Seznam skladeb (ve formátu m3u)
# 9. Soubor, do n□j□ se zapisuj□ informace o skladbÆch
```

Seznam skladeb ve formátu m3u můžete vytvořit výše uvedeným příkazem *find*.

Aktualizace ve verzi RC1

Nová verze programu *fluid* byla uvolněna pod licencí GPL. Hlavní změny jsou:

- Nový formát souboru *fluid.conf*.
- Nový implicitní port 4 711.
- Nový způsob spuštění a zastavení. Nyní se používají skripty *fluid.start* a *fluid.stop*.
- Součástí distribuce je zdrojový kód.

Litestream

Program Litestream funguje podobně jako Icecast. Systém se skládá ze streameru a ze zdroje. Streamer se spouští následujícím příkazem:

```
litestream port_zdroje stream_server stream_port max_klientů log_id [server_yp
port_yp]
```

Jednotlivé parametry mají následující význam:

port_zdroje	Port, kterým se připojujete ke zdroji zvuku.
stream_server	IP adresa nebo FQDN vašeho serveru (název zjistíte příkazem *hostname*, IP adresu v souboru */etc/hosts*).
stream_port	Port, k němuž se mají připojovat klienti.
max_klientů	Maximální počet klientů, kteří se mohou k serveru připojit.
log_id	Hlášení serveru se zapisují do souboru */var/log/messages*. Tímto parametrem definujete řetězec, kterým jsou záznamy identifikovány. Pokud použijete řetězec „Litestream“, budou záznamy v souboru */var/log/messages* vypadat takto: `Sep 18 19:32:20 linux Litestream[1901]: stream.c:555: main: 'server started'.`
server_yp port_yp	Adresa a port yp serveru, který se použije k oznámení existence vašeho serveru adresářovým serverům mp3.

Takto vypadá příklad spuštění serveru:

```
[dj@megajukebox]$ litestream 5000 megajukebox 5555 1000 LitestreamServer
```

Tím je spuštěn server. Ten už teď jenom sedí a čeká na připojení zvukového zdroje, což se provede příkazem *source*. Formát příkazu vypadá takto:

```
source ip_nebo_nÆzev port nÆzev □Ænr url irc icq aim veřejnØ?(0,1)
bitrate(16,18,56,128,...) playlist.txt log_id
```

Jednotlivé parametry mají následující význam:

ip_nebo_název	Počítač, na němž běží Litestream server.
port	Port, na němž Litestream server očekává zvukový zdroj.
název	Název streamu.
žánr	Žánr streamu.
url	URL webových stránek streamu.
irc	IRC kanál streamu.
icq	Váš ICQ identifikátor.
aim	Váš AIM identifikátor
veřejné	0 nebo 1, 0=privátní, 1=veřejné

bitrate	Přenosová rychlost, která se bude avizovat yp serveru. Nemusí nutně odpovídat skutečně použité rychlosti.
playlist.txt	Soubor se seznamem skladeb.
log_id	Identifikátor záznamů v logu.

Takto vypadá příklad spuštění zvukového zdroje:

```
[dj@megajukebox]$ source megakukebox 5000 'Megajukebox  The best jukebox ever'
Various  http://www.megajukebox.com \#megajukebox 0 N/A 1 128 /jukebox/playlist
LitestreamSource
```

Modul Apache::MP3

Module Apache::MP3 umožňuje webovému serveru Apache streamovat MP3 soubory. Nenabízí stejné množství funkcí jako programy Fluid a Litestream & Icecast.

Modul ke své činnosti vyžaduje moduly MP3::Info a mod_perl, všechny jsou dostupné na stránkách CPAN.

Po stažení příslušných souborů doporučujeme přečíst soubor README!

Nejprve sestavte modul a pak proveďte potřebné změny v konfiguračních souborech serveru Apache (budete zřejmě modifikovat buď *httpd.conf* nebo *srm.conf*).

Kromě toho budete muset vytvořit adresáře pro ikony a adresářovou strukturu pro MP3 soubory.

Šestá část souboru README popisuje podrobnosti o úpravách (nebo vytvoření) konfiguračního souboru Perlu.

Dále je nutné nahrát modul MP3::Info, bez kterého by mohlo dojít k havárii serveru.

V souboru *httpd.conf* doplňte položku typu

```
 Perlrequire /etc/httpd/conf/startup.perl
```

Upravte odkazy v souborech v */etc/httpd/conf* tak, aby ukazovaly na umístění konfiguračních souborů ve vašem systému. Pak na příslušném místě vytvořte soubor *startup.perl*. Minimálně by měl obsahovat následující (případně upravte cesty tak, aby odpovídaly vašemu systému):

```
[dj@megajukebox conf]$ cat /etc/httpd/conf/startup.perl

#!/usr/bin/perl

use MP3::Info();
```

Pak restartuje Apache a v prohlížeči otevřete některý z adresářů s MP3 soubory.

K modulu Apache::MP3 existuje další dokumentace, spusťte příkaz:

```
[dj@megajukebox conf]$ perldoc Apache::MP3
```

Překonání firewallu

Řada společností používá firewally, které blokují možnost připojit se na určité porty vzdálených systémů.

Jedna možnost, jak to obejít, je provozovat MP3 server na portu 80. Na stejné adrese však nebudete moci provozovat webový server.

Pokud chcete Icecast spustit tímto způsobem, nezapomeňte příslušným parametrem nastavit port na hodnotu 80:

```
[dj@megajukebox bin]$ ./shout megajukebox -e 80 -P hackme -p ../playlist
```

Nároky na přenosovou kapacitu

Streamování zvuku s nastavenou vyšší přenosovou rychlostí může mít velké nároky na přenosovou kapacitu sítě. Představme si následující situaci: Linka typu T1 má kapacitu přibližně 1,55 Mb/s. Pokud vysíláte MP3 soubory v kvalitě 128 kb/s stereofonně, každý připojený klient spotřebuje 256 kb/s kapacity, takže se jich k vám může připojit maximálně 6. Navíc se k vám těžko připojí uživatelé modemů.

Musíte tedy kvalitu volit nejen podle kapacity vaší internetové linky, ale také podle toho, jak se k vám budou připojovat uživatelé. Rychlost 24 kb/s stereo představuje rozumně kvalitní přenos, který mohou využít i uživatelé modemů s rychlostí 56 kb/s. Stejná T1 linka vám pak umožní současně až 32 připojení.

I pokud vysíláte pouze pro interní síť, stále musíte mít přenosové nároky na paměti, zejména pokud máte síť s rychlostí 10 Mb/s.

Problematika autorských práv

Myslím, že je rozumné předpokládat, že hudební vydavatelství nebudou šťastná, pokud budete jejich skladby šířit bez povolení a bez příslušných poplatků. Co tedy vysílat můžete?

Měli byste si být vědomi právních úprav, protože zodpovědnost za vysílání bude ležet na vás. Níže uvádíme dva odkazy – jeden na Electronic Frontier Foundation, která usiluje o zmírnění omezení vztahujících se k podobným technologiím. Druhý je na Recording Industry Association of America, která se snaží chránit práva autorů před pirátstvím.

Vřele vám doporučuji obě adresy navštívit, včetně stránek institucí, které se vztahují na místa, kde fyzicky působíte.

http://www.eff.org/cafe/

http://www.riaa.com/weblic/weblic.htm

Přehrávání MP3

Předpokládáme, že nějaké MP3 soubory vlastníte a máte tedy možnost je poslouchat přímo, nebo prostřednictvím streamu.

Přehrávání ze souboru

Přehrávání ze souborů je u všech přehrávačů poměrně přímočaré. Jediným rozdílem je to, že některé přehrávače jsou řádkově ovládané, jiné jsou grafické.

Chcete-li přehrát MP3 soubor, musíte jeho název předat jako parametr, například:

```
[dj@megajukebox]$ mpg123 /mp3_files/SampleFile.mp3
```

nebo

```
[dj@megajukebox]$ xaudio /mp3_files/SampleFile.mp3
```

Chcete-li přehrávat více souborů, uveďte je všechny:

```
[dj@megajukebox]$ alsaplayer /mp3_files/SampleFile1.mp3
/mp3_files/SampleFile2.mp3
```

Pokud chcete přehrát všechny soubory v určitém adresáři, můžete je specifikovat takto:

```
[dj@megajukebox]$ xmms /mp3_files/*.mp3
```

Přehrávání MP3 streamů

Přehrávání MP3 ze streamovacích serverů je velmi jednoduché, prostě nahraďte název souborů příslušnou adresou a číslem portu:

```
[dj@megajukebox]$ mpg123 http://localhost:8000
```

nebo

```
[dj@megajukebox]$ freeamp http://megajukebox:2711
```

Záznam MP3 streamů

Streamy z MP3 serverů je možné zaznamenat a lokálně uložit následujícími způsoby.

Streamripper

Záznam MP3 streamů programem *streamripper* můžete provést takto:

```
[dj@megajukebox]$ streamripper -h megajukebox -p 8000
Press CTRL-C to stop
name: Megajukebox
reponse: 200
genre: Megajukebox
url: http://www.megajukebox.com/
bitrate: 128
This stream contains no meta data, ripping as one large ass track
1st track ripped
socket error: : Success
```

Dále uvádíme seznam parametrů programu.

Použití: *streamripper -h hostitel -p port [-d adresář]*

-h hostitel	Název serveru. Povinný parametr.
-p port	Port na serveru. Povinný parametr.
-d adresář	Adresář, do nějž se mají soubory ukládat.
-l logovací_soubor	Zaznamenávat provedené aktivity do souboru.
-v	Podrobnější hlášení programu.
-c	Nezobrazovat stavové počitadlo.
-q	Tichý režim, nevytváří mp3 soubory.
-s	Předává stream na stdout.
-n	Před zahájením zápisu počká na začátek stopy.
-o	Vytvoří adresář s názvem zaznamenávaného streamu.

wget

Pomocí programu *wget* můžete stream uložit takto:

```
[dj@megajukebox]$ wget http://megajukebox:8000 -O download01.mp3
--13:41:41--  http://megajukebox:8000/
          = `download01.mp3'
Connecting to megajukebox:8000... connected!
HTTP request sent, awaiting response... 200 OK
Length: unspecified [audio/mpeg]

    0K - ......... ......... ......... ......... .........
   50K - ......... ......... ......... ......... .........
  100K - ......... ......... ......... ......... .........
  150K - ......... ......... ......... ......... .........
  200K - ......... ......... ......... ......... .........
  250K - ......... ......... ......... ......... .........
  300K - ......... ......... ......... ......... .........
  350K - ......... ......... ......... ......... .........
  400K - ......... ......... ......... ......... .........
  450K - ......... ......... ......... ......... .........
  500K - ......... ......... ......... ......... .........
```

Mixování

eMixer

Program *eMixer* umožňuje mixovat MP3 podobně jako na mixážním pultu.

Novější verze podporuje dvě zvukové karty, takže výstup můžete posílat jednou zvukovou kartou a přes druhou můžete poslouchat další stopu.

Po rozbalení archivu si opět přečtěte soubor s instrukcemi, jak program přeložit.

Dále budete muset vytvořit seznam MP3 souborů, příslušný příkaz jsme uváděli v části věnované streamování.

Před spuštěním programu *eMixer* budete muset mít nainstalován program *mpg123*.

Dále uvádíme ovládací klávesy programu:

nahoru, dolů Procházení seznamem skladeb

Page Up, Page Down Procházení seznamem skladeb

0.Blues	1.Classic Rock	2.Country	3.Dance
4.Disco	5.Funk	6.Grunge	7.Hip-Hop
8.Jazz	9.Metal	10.New Age	11.Oldies
12.Other	13.Pop	14.R&B	15.Rap
16.Reggae	17.Rock	18.Techno	19.Industrial
20.Alternative	21.Ska	22.Death Metal	23.Pranks
24.Soundtrack	25.Euro-Techno	26.Ambient	27.Trip-Hop
28.Vocal	29.Jazz+Funk	30.Fusion	31.Trance
32.Classical	33.Instrumental	34.Acid	35.House
36.Game	37.Sound Clip	38.Gospel	39.Noise
40.AlternRock	41.Bass	42.Soul	43.Punk
44.Space	45.Meditative	46.Instrumental Pop	47.Instrumental Rock
48.Ethnic	49.Gothic	50.Darkwave	51.Techno-Industrial
52.Electronic	53.Pop-Folk	54.Eurodance	55.Dream
56.Southern Rock	57.Comedy	58.Cult	59.Gangsta
60.Top 40	61.Christian Rap	62.Pop/Funk	63.Jungle
64.Native American	65.Cabaret	66.New Wave	67.Psychadelic
68.Rave	69.Showtunes	70.Trailer	71.Lo-Fi
72.Tribal	73.Acid Punk	74.Acid Jazz	75.Polka
76.Retro	77.Musical	78.Rock & Roll	79.Hard Rock

Enter	Spuštění/zastavení stopy

80.Folk	81.Folk-Rock	82.National Folk	83.Swing
84.Fast Fusion	85.Bebob	86.Latin	87.Revival
88.Celtic	89.Bluegrass	90.Avantgarde	91.Gothic Rock
92.Progressive Rock	93.Psychedelic Rock	94.Symphonic Rock	95.Slow Rock
96.Big Band	97.Chorus	98.Easy Listening	99.Acoustic
100.Humour	101.Speech	102.Chanson	103.Opera
104.Chamber Music	105.Sonata	106.Symphony	107.Booty Bass
108.Primus	109.Porn Groove	110.Satire	111.Slow Jam
112.Club	113.Tango	114.Samba	115.Folklore
116.Ballad	117.Power Ballad	118.Rhythmic Soul	119.Freestyle
120.Duet	121.Punk Rock	122.Drum Solo	123.A capella
124.Euro-House	125.Dance Hall		

Tab	Změna kanálu
}]	Přepíná mezi ovládáním hlasitosti a rychlosti
mezerník	Restartuje aktivní stopu
vlevo, vpravo	Ovládání faderu
Insert	Snižuje hlasitost/rychlost kanálu 1
Home	Zvyšuje hlasitost/rychlost kanálu 1
Delete	Snižuje hlasitost/rychlost kanálu 2
End	Zvyšuje hlasitost/rychlost kanálu 2
< /	Polohování faderu
+ =	Přepínání mezi fadery
q	Start/stop kanálu 1
w	Start/stop kanálu 2
p	Přepíná režimy přehrávání (single, loop, continuous, random)
a	Zastaví všechny kanály
f	Souborová nabídka
u	Nabídka nástrojů
h	Nabídka nápovědy
~ `	Ruší nabídku
s	Zapne simultánní přehrávání obou kanálů

GDAM

GDAM je grafický MP3 mixér s řadou vestavěných efektů.

Editace ID3 záznamů

ID3 záznamy jsou informační údaje uložené v MP3 souborech, které obsahují data jako interpret skladby, album a podobně.

Pomocí příslušných editorů můžete tyto údaje měnit.

id3ed

Použití:

```
id3ed [-s skladba] [-n interpret] [-a album] [-y rok] [-c komentář]
      [-k číslo_stopy] [-g žánr] [-q] [-SNAYCKG] [-l/-L] [-r]
      [-i] mp3_soubory [-v]
```

-q	Bez řádkového rozhraní, nastaví pouze hodnoty předané pomocí parametrů. Dvojí zadání parametru potlačí jakýkoliv výstup programu.
-SNAYCKG	Vyzve k zadání pouze vybraných údajů.
-l/-L	Vypíše seznam žánrů.
-r	Odstraní stávající informace.
-i	Vypíše stávající informace bez jejich úpravy.
-v	Vypíše verzi programu.

```
[dj@megajukebox MyBand]$ id3ed track01.mp3

File track01.mp3: (tag v1.1)
songname[max:30]: Our Kick Ass Demo
artist[max:30]: Us
album[max:30]: White Album
year[max:4]: 1999
comment[max:28]: Will be a classic some day!
tracknum[max:3]: 1
genre[0-255/name]: 5
```

Chceme-li informace zobrazit, použijeme parametr *-i*:

```
[dj@megajukebox MyBand]$ id3ed -i track01.mp3
track01.mp3: (tag v1.1)
songname: Our Kick Ass Demo
artist: Us
album: White Album
year: 1999
comment: Will be a classic some day!
tracknum: 1
genre: Funk(5)
```

mp3info

Program MP3info funguje podobně jako *id3ed*, dále uvádíme část z jeho manuálové stránky:

Použití:

```
mp3info [-parametr hodnota] soubor(y)
```

-T	Nečíst informace z CD. Vhodné pro pomalá média, protože ID3 informace jsou uloženy na konci souboru.
-s 0/1	Vypsat informace. Zastaralá volba, měla by být vždy 1.
-f *format*	Formát vypisovaného řetězce. Podrobnosti viz README.
-F *čísloFormátu*	Použít předdefinovaný formát. Vyzkoušejte.
-w	Zapíše informace na konec souboru.

[1] Pozn. překladatele: Ano, takové situace jsou :-) Pokud vlastníte jako já fotoaparát HP PhotoSmart 715, nezbude vám nic jiného, než aplikovat patch např. podle , a pak si přeložit jádro znovu. Mnoho štěstí!

`-W`	Vymaže informace ze souboru. Přepíše všechna nastavení datových příznaků, nastavuje *–s off*. Chcete-li informace současně smazat a zobrazit, zadejte *–W –s 1*.
`-n` *název*	Nastavuje název skladby. Předpokládá volbu *–w*. Zastaralý parametr, používejte *–t*.
`-t` *název*	Nastavuje název skladby. Předpokládá volbu *–w*.
`-a` *interpret*	Nastavuje jméno interpreta. Předpokládá volbu *–w*.
`-l` *album*	Nastavuje název alba. Předpokládá volbu *–w*.
`-y` *rok*	Nastavuje rok vydání. Předpokládá volbu *–w*.
`-c` *komentář*	Nastavuje komentář. Předpokládá volbu *–w*.
`-g` *žánr*	Nastavuje číslo žánru. Předpokládá volbu *–w*.
`-G` *žánr*	Nastavuje název žánru. Předpokládá volbu *–w*.

Identifikátory žánru

Tyto identifikátory slouží k nastavení žánru skladby, jsou převzaty z http://www.id3.org/id3v2-00.-txt.

Jsou definovány následující žánry: in ID3v1

Následující žánry jsou rozšíření Winampu:

MP3 na Minidisc

Záznam MP3 na minidisc je možný dvěma metodami: analogově a digitálně. Každá z metod závisí na typu zvukové karty, kterou máte nainstalovánu. Hlavní rozdíl mezi analogovým a digitálním záznamem je v kvalitě pořízeného záznamu. U analogového záznamu může vznikat šum a jiné rušení, i když se je některé zvukové karty snaží minimalizovat. Pokud máte pouze analogovou kartu, neznamená to, že byste nemohli dosáhnout kvalitní nahrávky. Pouze si budete muset pohrát s nastaveními mixéru tak, abyste šum minimalizovali.

Nepotřebujete žádný speciální program, pouze normální přehrávač MP3 souborů. Samozřejmě obě metody vyžadují funkční propojení výstupu zvukové karty a vstupu minidiscu.

Analogový záznam

Pokud vaše zvuková karta nemá digitální výstup, můžete zvolit nahrávání analogovou linkou.

Propojte výstup „Line Out“ zvukové karty se vstupem „Line In“ Minidisku vhodným analogovým kabelem. Obvyklým způsobem aktivujte na Minidisku režim záznamu a spusťte přehrávání MP3. Minidisk by měl zobrazovat záznamovou úroveň. Při přehrávání se možná objeví větší či menší šum, způsobený zvukovou kartou.

Abyste šum při záznamu minimalizovali, nastavte záznamovou úroveň Minidisku v rozmezí –3 dB až 0 dB. Další možnost je výše popsaným způsobem Minidisk připojit, zapnout nahrávání, ale nespustit přehrávání MP3. Takto můžete nastavováním mixéru šum co nejvíce snížit.

Digitální záznam

Při použití digitálního záznamu dosáhnete lepší kvality. Připojte optický kabel do výstupu „Digital Out“ zvukové karty. (Pokud budete kabel kupovat, zvolte správnou velikost podle portů na zvukové kartě a Minidisku. Obvyklá velikost je 3,5 mm, ale podívejte se raději do návodu.)

Volný konec kabelu by měl svítit. Nyní jej připojte do digitálního vstupu Minidisku. Zapněte nahrávací režim a začněte přehrávat MP3.

Seznam skladeb

Poslední problém se týká přehrávání série MP3 souborů: jak má Minidisk poznat, kdy dochází k přechodu z jedné skladby na druhou a jak má tedy automaticky číslovat stopy? Řešení přináší MP3 soubor, který si můžete stáhnout z následující adresy: http://www.prongs.org/minidisc/2sec.zip.

Jedná se o MP3 soubor obsahující dvě sekundy ticha, který můžete vložit mezi jednotlivé skladby. Podle ticha pak Minidisk pozná, že dochází ke změně skladby.

USB Digitální fotoaparáty

Originál: http://tldp.org/HOWTO/USB-Digital-Camera-HOWTO/

Úvod

Záměr

Tento dokument je určen uživatelům Linuxu, kteří mají digitální fotoaparát s rozhraním USB a implementací USB Mass Storage, nebo kteří si chtějí takový aparát koupit. V mém případě jsem jej dostal k narozeninám a nešlo jej vrátit, abych neurazil něčí city. Bylo tedy nutné vzít dokumentaci a zprovoznit jej. Není nic, co bychom nemohli dokázat, nejsou hloupé otázky, jsou pouze informace, které zatím nemáme.

Pokrývaná zařízení

Popsaný postup funguje s jádrem 2.4.8 a testoval jsem jej s aparátem Sony P-50 Cybershot s 4 MB a 64 MB paměti, a se čtečkou karet fotoaparátu Olympus. Podle dostupných informací by měl postup fungovat zpět až k jádrům 2.2.19, ale nemám to vyzkoušeno. Pozitivně vím, že postup nefunguje s jádrem 2.2.15.

Dále uvedené informace pomohly vyřešit mé problémy. Existují i jiné postupy, ale ty mohou vyžadovat překlad nového jádra, čemuž jsem se chtěl vyhnout[1]. Podle informací v tomto dokumentu byste měli být schopni zprovoznit USB Mass Storage při startu systému. Linux se neustále vyvíjí a i když zde nenajdete řešení svého problému, měli byste minimálně získat základ pro další experimenty. Následující citace z dokumentu „gphoto2 README“ naznačuje, pro které další fotoaparáty může uvedený postup fungovat. Aktuální verzi najdete na adrese http://www.gphoto.org.

> Existují fotoaparáty podporující takzvaný USB Mass Storage Protocol. Tento protokol je veřejně publikován a umožňuje přístup k různým typům datových zařízení, ať už je to fotoaparát nebo disk, připojeným přes USB. Pro tento protokol existují samostatné ovladače, nepotřebujete speciální programy jako je *gphoto2*.
>
> V současné době je protokol podporován následujícími zařízeními:

- Casio QV [2x00,3x00,8000]
- Fuji FinePix S1 Pro, [1400,2400,4700]Zoom, 1300, 4500

- HP PhotoSmart 315, 618, 912
- Leica Digilux 4.3
- Konica KD300Z
- Kyocera Finecam s3
- Minolta Dimage 7
- Nikon Coolpix 995
- Olympus C-100, C-200Z, C-700, C-860L, C-2040, C-3020Z, C-3040Z, C-4040Zoom, D-510, E-10
- Pentax Optio 330
- Sony DSC-F505(V), DSC P5, DSC-F707

Programem *gphoto2* nelze s těmito fotoaparáty pracovat.

Jiné aparáty podporují protokol PTP nebo USB Imaging Devices, což je protokol vyvinutý společností Kodak a dalšími. Program *gphoto* jej nepodporuje, je však podporován programem *jPhoto* (http://jphoto.sourceforge.net). Následující seznam uvádí aparáty podporující tento protokol:

- Kodak DC-4800, DX-3215, DX-3500, DX-3600, DX-3700, DX-3900, MC3 a všechny aparáty systému Kodak Easy Share™.
- Sony DSC-P5, DSC-F707 (oba vyžadují uživatelské nastavení aparátu).

Tyto aparáty nebudou podporovány, dokud nebude *gphoto2* implementovat protokol PTP.

Doporučené materiály

Doporučujeme vám přečíst následující dokumentaci. Obsahuje užitečné informace.

Manuálové stránky

Manuálové stránky programů *lsmod*, *modprobe*, *mount*, *mv*, *su*, *fstab*, *mtab*, *dir*, *install*, *mknod* a *chown*.

Dokumenty HOWTO

Module-HOWTO, *Bash-Prog-Intro-HOWTO*, *SCSI-2.4-HOWTO* a *Hardware-HOWTO*.

Webové stránky

http://www2.one-eyed-alien.net/~mdharm/Linux-usb/ a http://www.Linux-usb.org/USB-guide-/book1.html.

Předpoklady

Můj počítač je Athlon 900 s 40GB diskem. Nepoužívám žádná SCSI a USB zařízení. Jádro bylo přeloženo bez ovladačů SCSI a USB, ty jsou přeloženy jako moduly. Používám jádro 2.4.8_26mdk z distribuce Mandrake 8.1. Vzhledem k tomu, že budeme pracovat na úrovni jádra, měly by popsané postupy fungovat pro libovolnou distribuci.

Dále předpokládám:

- Nemáte nahrány moduly SCSI a USB a nemáte připojena žádná taková zařízení.

- Znáte heslo superuživatele.
- Slovem „fotoaparát“ budeme označovat zařízení, která mohou být fotoaparát, čtečka karet nebo jakékoliv zařízení USB Mass Storage.
- Mountovací bod „camera“, */mnt/camera* nedefinuje zařízení ve výše uvedeném významu. V adresáři */mnt* najdete pravděpodobně adresáře *cdrom*, *disk*, *floppy* a případně další. Já zde mám navíc i adresář *camera*. Jedná se o adresáře používané jako připojovací body.
- Výzva „[bash]$“ je výzvou shellu. Při zadávání příkazů ji neopisujte.

Předběžná nastavení

V tomto okamžiku je nutné provést některá rozhodnutí. Je nutné vytvořit připojovací adresář a pojmenovat jej. Já používám adresář *camera* v */mnt*. Název adresáře může být libovolný. Adresář vytvoříte následujícím příkazem:

```
[bash]$ mkdir -m 777 /mnt/camera
```

Všechny fotky ukládám do jednoho adresáře s podadresáři, které slouží k tematickému členění. Proto jsem vytvořil adresář *picture* v domovském adresáři. Název adresáře může být libovolný. Adresář vytvoříte následujícím příkazem:

```
[bash]$ mkdir -m 777 ~/picture
```

Viz část *A.1* na konci tohoto dokumentu.

Důležité rozhodnutí! Budete pracovat jako normální uživatel nebo jako superuživatel?

Skripty

Následující skripty vznikly na základě čtení řady diskusních skupin, praktických návodů a manuálů. Netvrdím, že jsou originální, jde o kompilaci podle řady různých rad a doporučení. Chtěl bych poděkovat všem, kteří mi s jejich vznikem pomohli.

Spusťte svůj oblíbený textový editor, zvolte název souboru a vytvořte skripty pro normálního uživatele nebo superuživatele.

Normální uživatel

```
echo "Zadejte nazev adresare pro ulozeni fotek."
read DIRPATH
mkdir ~/picture/$DIRPATH
su -c "/sbin/modprobe usb-storage; mount -t vfat /dev/sda1 /mnt/camera;
/etc/rc.d/init.d/usb start;
mv /mnt/camera/dcim/100msdcf/*.jpg ~/picture/$DIRPATH;
umount /mnt/camera;
chown -R va□e_płihla□ovac□ jmØno ~/picture/$DIRPATH"
```

Superuživatel

```
echo "Zadejte nazev adresare pro ulozeni fotek."
read DIRPATH
mkdir picture/$DIRPATH
/sbin/modprobe usb-storage
mount -t vfat /dev/sda1 /mnt/camera
```

```
/etc/rc.d/init.d/usb start
mv /mnt/camera/dcim/100msdcf/*.jpg picture/$DIRPATH;
umount /mnt/camera
chown -R vaše_přihlašovací_jméno picture/$DIRPATH
```

Nastavte skripty jako spustitelné

Nyní je nutné skripty nastavit jako spustitelné. Provedete to příkazem:

Jako uživatel:

```
[bash]$ su -c "chmod a=r+w+x název_skriptu"
```

Jako superuživatel:

```
[bash#] chmod a=r+w+x název_skriptu
```

Co se děje, když skript spustíte

Po spuštění skript vytvoří adresář pro uložení fotek. Název adresáře zadáte po výzvě. Jmenuje-li se skript *getcamJ* (*J* protože stahuje fotky ve formátu *jpg*), bude jeho běh vypadat takto:

```
[bash]$ getcamJ
Zadejte nazev adresare pro ulozeni fotek.
nejaky_adresar
Password:
heslo_superuživatele
```

Pokud skript spouštíte jako superuživatel, zbytek odstavce pro vás neplatí. K prováděným operacím potřebujeme práva superuživatele. Proto používáme příkaz *su*. Parametr *-c* slouží k provedení jednoho příkazu. Uvozovky nám umožňují zadat příkaz obsahující mezery, středník pak odděluje příkazy, které se provedou jeden po druhém

/sbin/modprobe usb-storage : nainstaluje modul USB Mass Storage společně s dalšími potřebnými moduly a ovladači, zejména půjde o ovladač SCSI. Zkontrolujte, že máte v adresáři */dev* vytvořeny položky *sda0*, *sda1*, *sda2*, *sda3*, *sda4*, *sdb0*, *sdb1*, *sdb2*, *sdb3*, *sdb4*. Pokud používáte i jiná SCSI zařízení, opravte *sda1* na jiné, pravděpodobně *sdb1*.

mount -t vfat /dev/sda1 /mnt/camera : Připojí SCSI ovladač.

/etc/rc.d/init.d/usb start : Spustí služby USB.

mv /mnt/camera/dcim/100msdcf/.jpg picture/$DIRPATH;* : Přesune fotky z fotoaparátu na disk, zároveň je odstraní z aparátu.

umount /mnt/camera : Odpojí SCSI ovladač.

chown -R your_login_name picture/$DIRPATH : Nakopírované soubory vlastní uživatel *root* a normální uživatel by k nim neměl přístup. Tímto příkazem se změní vlastník souborů, podrobnosti viz manuál.

Používám systém, v němž ovladače USB a SCSI nejsou součástí jádra, zavádějí se jako moduly. Skript předpokládá, že váš systém je nastaven stejně. Pokud ne, budete jej muset opravit. Přečtěte si relevantní manuálové stránky a praktické návody (dokumenty HOWTO), případně se někoho zeptejte.

Doladění

Všechno by mělo být připraveno k finálním úpravám. Nafoťte nějaké zkušební snímky ve všech možných formátech. Můj aparát podporuje formáty TIFF, GIF, JPEG a MPEG a rovněž nabízí k jednotlivým fotkám náhledy. Fotky jsou uloženy v adresářích *100msdcf*, *imcif100*, *thm* a *moml0001*, které se nacházejí v adresářích *dcim* a *mssony*. Musíte si zjistit, jak adresáře pojmenovává váš aparát. Můžete to provést následujícím postupem:

- Zkopírujte výše uvedený skript do pomocného souboru
- Pomocí textového editoru změňte řádek mv /mnt/camera/dcim/100msdcf/*.jpg picture/$DIRPATH na mv /mnt/camera/* picture/$DIRPATH
- Spusťte skript takto:

  ```
  [bash]$ ./název_skriptu
  ```

  (Všimněte si tečky a lomítka.)

Nepropadejte panice

Upravený skript může hlásit různé chyby. Pro tuto chvíli je ignorujte. Než začnete tvrdit, že nic nefunguje, zkuste se podívat, zda jste náhodou nezískali nějaké fotky. Pokud ano, zapište si názvy adresářů. Pak znovu spusťte textový editor a opravte cestu *dcim/100msdcf/*.jpg* tak, aby odpovídala vašim adresářům. Můžete si vytvořit několik skriptů pro manipulaci s různými typy souborů.

V tomto okamžiku by měl adresář s fotkami na disku vypadat tak nějak, jak uvádíme v příloze *A.2*. Ověřte si to následujícím příkazem:

```
[bash]$ dir -R adresář_s_fotkami
```

Podle informací v tomto dokumentu, manuálu a dalších HOWTO by všechno mělo fungovat.

Hodně štěstí.

Problémy

Pokud něco nefunguje správně, je čas na trochu diagnostiky. Podívejte se na foťák, zda v něm snímky stále jsou. Pokud ano, zbytek tohoto odstavce přeskočte. Pokud fotky nejsou ve foťáku, jsou někde na disku. Hledejte. Pokud je nenajdete, nafoťte nějaké další, vypněte foťák, připojte jej k počítači a zapněte.

Zkontrolujte, zda existuje připojovací bod */mnt/camera*. Pokud ne, vytvořte jej. Občas se stane, že zmizí. Stalo se mi také, že zmizelo zařízení */dev/sda1*. Pokud je to váš případ, znovu jej vytvořte.

Smažte všechny adresáře, které vytvořil modifikovaný skript, a spusťte jej se správně nastavenými cestami. Smazání všech adresářů a souborů snáze provedete jako superuživatel – nezapomeňte se pak vrátit do normálního uživatelského režimu. Viz příloha C.

Zadejte příkaz

```
[bash]$ dmesg
```

Mělo by se objevit něco jako:

```
hub.c: USB new device connect on bus1/1, assigned device number 2
usb.c: USB device 2 (vend/prod 0x54c/0x10) is not claimed by any active driver.
```

(0x54c/0x10 se bude lišit podle typu foťáku.)

Pokud se něco podobného objevilo, bylo zařízení USB Mass Storage rozeznáno.

Nyní zapněte foťák a spusťte upravený skript. Po novém zadání příkazu *dmesg* by se mělo objevit něco jako:

```
[bash]$ dmesg

SCSI subsystem driver Revision: 1.00
Initializing USB Mass Storage driver...
usb.c: registered new driver usb-storage
scsi0 : SCSI emulation for USB Mass Storage devices
Vendor: Sony Model: Sony DSC Rev: 3.22
Type: Direct-Access ANSI SCSI revision: 02
WARNING: USB Mass Storage data integrity not assured
USB Mass Storage device found at 2
USB Mass Storage support registered.
Attached scsi removable disk sda at scsi0, channel 0, id 0, lun 0
SCSI device sda: 126848 512-byte hdwr sectors (65 MB)
sda: Write Protect is off
/dev/scsi/host0/bus0/target0/lun0: p1
usb-uhci.c: interrupt, status 3, frame# 1628
```

Nyní spusťte následující příkaz a podívejte se do přílohy B.

```
[bash]$ lsmod
```

Pokud informace vypsané příkazem *lsmod* odpovídají příloze B, a pokud výpis příkazu *dmesg* ukazuje to, co jsme viděli výše, pak netuším, v čem by mohl být problém. Jediné, co můžu poradit, je zkusit celý postup znovu. Tentokrát však přesměrujte výstupy do souboru a výsledky pošlete s popisem toho, co jste udělali, do vaší oblíbené linuxové konference.

A nakonec

Pokud všechno funguje správně a tak jak chcete, můžete ještě udělat poslední věc. Není ovšem nutná. Skript pro stažení fotek můžete spouštět z adresáře, kde jej máte uložen pomocí „./“ – anebo jej můžete umístit do některého z adresářů, kde jsou uloženy i jiné spustitelné soubory. Doporučuji adresář */usr/sbin*. Provedete to příkazem:

```
[bash]$ install název_skriptu /usr/sbin
```

Příloha A

Část 1

Takto mám organizovány snímky – primární adresář *picture* a v něm podadresáře pro různé kategorie:

```
picture/n□co:
dsc00117.jpg dsc00120.jpg dsc00123.jpg dsc00126.jpg dsc00129.jpg
dsc00118.jpg dsc00121.jpg dsc00124.jpg dsc00127.jpg dsc00130.jpg
dsc00119.jpg dsc00122.jpg dsc00125.jpg dsc00128.jpg dsc00131.jpg
```

Část 2

Po experimentu s upraveným skriptem budou snímky rozmístěny v různých adresářích. Pro tuto chvíli je to to, co chceme:

```
picture/trash:
camera

picture/trash/camera:
dcim mssony

picture/trash/camera/dcim:
100msdcf

picture/trash/camera/dcim/100msdcf:
dsc00357.jpg dsc00360.jpg dsc00363.jpg txt00365.gif
dsc00358.jpg dsc00361.jpg dsc00364.jpg txt00365.thm
dsc00359.jpg dsc00362.jpg dsc00366.jpg

picture/trash/camera/mssony:
imcif100

picture/trash/camera/mssony/imcif100:
dsc00364.jpg dsc00366.tif
```

Příloha B

Co potřebujeme najít je *usb-storage* ve sloupci *Used by*:

```
Module       Size      Used by
nls_iso8859-12880  0   (autoclean)
nls_cp437 4400  0   (autoclean)
sd_mod11792  0   (autoclean)
vfat 9968  0   (autoclean)
fat 32192  0  (autoclean) [vfat]
usb-storage 52528 0
scsi_mod 91072 2 [sd_mod usb-storage]
ppp_deflate 42208 0  (autoclean)
bsd_comp  4576  0  (autoclean)
ppp_async 6672  0  (autoclean)
ppp_generic 19616  0  (autoclean) [ppp_deflate bsd_comp ppp_async]
slhc 5136  0  (autoclean) [ppp_generic]
parport_pc  20240 1  (autoclean)
lp 5808  0  (autoclean)
parport  24768  1 (autoclean) [parport_pc lp]
es1371 26768  1
soundcore 4208  4 [es1371]
ac97_codec  9312  0  [es1371]
gameport  1856  0  [es1371]
af_packet  12560  0 (autoclean)
ip_vs 62000  0  (autoclean)
usb-uhci  21232  0  (unused)
usbcore  50752  1 [usb-storage usb-uhci]
rtc 5600  0  (autoclean)
```

Příloha C

Budete-li chtít smazat všechny testovací adresáře, udělejte to následujícím příkazem – *ale opatrně*!

```
[bash]$ rm -Rf picture/testovac□_adresÆł
```

Snadno můžete smazat i něco, co nechcete!

Vypalování CD z MP3

Originál: http://tldp.org/HOWTO/CD-Writing-HOWTO.html

Tento dokument vysvětluje, jak zapisovat na CD-ROM v systému Linux.

Úvod

Mnoho lidí používá Linux pro vypalování CD-ROMů, protože je to spolehlivé a jednoduché. Žádné modré obrazovky při vypalování a žádná starost ohledně správné kombinace hardware a software. Jakmile je to správně nastaveno, prostě to funguje. CD-Writing HOWTO je obsažen rovněž v této knize jako praktický návod „Vypalujeme CD s MP3“ (kapitola 22) popisuje nastavování, umisťování dat na média a zmiňuje zajímavé aplikace zaslané laskavými čtenáři.

Dostupnost

Jako vydavatel tohoto dokumentu většinou shrnuji to, co mi napsali jiní lidé. Nejsem vývojář softwaru a ani expert na hardware, takže se na specifické problémy s hard- nebo softwarem ptejte někoho jiného. Co má vždy smysl, je oznámení řešení problémů, které ještě nejsou popsány v tomto praktickém návodu, mně.

Ročně dostávám několik set e-mailů ohledně tohoto praktického návodu. Takže prosím buďte trpěliví, jelikož nemohu vždy odpovědět do hodiny. Samozřejmě, čtu vše hned a dávám si to do mé CDR-fronty. Než se na něco zeptáte, ujistěte se prosím, že máte nejnovější verzi tohoto dokumentu; je vždy k dispozici na adrese http://www.guug.de/~winni/linux/.

Doporučená četba

Možná budete potřebovat příručku k vaší distribuci Linuxu, abyste se dozvěděli něco o instalaci nového jádra. Tuto problematiku neovládám pro jiné distribuce Linuxu, než používám já.

CD-R FAQ je obecný FAQ o zapisovatelných kompaktních discích (CD-R), zapisovacích mechanikách a požadovaném softwaru. Většina zapisovacích mechanik může být použita také pro čtení CD-ROMů, možná byste si mohli přečíst *Linux CD-ROM HOWTO*, *Linux SCSI HOWTO* a *Linux Kernel HOWTO*.

Terminologie ... lasery na maximum ... pal!

CD-ROM znamená *Compact Disc Read Only Memory*, médium úložiště využívající optický laser pro čtení mikroskopických dírek na barevném lesknoucím se disku. Díry reprezentují bity informace a jsou tak maličké, že se jich na disk vejde několik miliard. Proto je CD vysokokapacitní médium.

Termín *CD-R* je zkrácená podoba *CD-ROM recordable* a značí, že jde o CD, které na svém povrchu nemá tyto mikroskopické díry. Takže je prázdné. CD-R má uvnitř speciální chemický film, do kterého se dírky vypalují. To se dělá přidáním energie laseru, který normálně pouze snímá dírky, který pak tyto dírky vytváří. Tato akce může být na CD-R provedena **pouze** jednou. Některé oblasti můžete ponechat pro pozdější zápis, čímž se vytváří takzvané *multi-session CD*.

CD-ROM rewritable (krátce: *CD-RW*) bylo vyvinuto odstraněním omezení média CD-R. S vypalovačkou CD-RW může laser dělat obojí – vytvářet dírky do média a také vracet médium do jeho původního stavu. To je možné, protože laser ve skutečnosti nedělá dírky do média, které by se ztrácely v oblacích kouře. Dobrou analogií pro techniku je hra ledního hokeje: jízdou po ledu na něm hráči (laser) vytváří škrábance. Vzorek na ledu (médiu) je zápisem toho, co se stalo na ledu v průběhu jedné třetiny. Mezi třetinami čistící vůz Zamboni jezdí po ledu a vyplňuje škrábance rozpuštěním horní vrstvičky ledu. (Zamboni je název druhu čistícího vozu na stadionech ledního hokeje). Tímto způsobem je vzorek na ledu odstraněn a může začít nová třetina. Vědecký termín pro odpaření, kondenzaci, rozpuštění a zmrazení je „změna fáze“, což dává vypalovačkám CD-RW název „zařízení pro změnu fáze „.

Tento dokument se zabývá otázkou zápisu na média CD-R a CD-RW. Vítejte na palubě, kapitáne.

Adaptor vs. Adapter

Nejčastěji používaným ve zdrojích jádra je adapter (adapter: 4283, adaptor: 154). Ještě důležitější jsou parametry možností modulů a aliasy, jako u „scsi_hostadapter“. Takže z důvodu konzistentnosti terminologie v příkladech konfigurace a textu dokumentu používám tuto konvenci bez ohledu na správnost terminologie.

Podporované zapisovací jednotky CD

Zapisovací jednotky USB aktuálně nejsou podporovány. Až na toto můžete bezpečně předpokládat, že nejnovější zapisovací jednotky IDE/ATAPI a SCSI v systému Linux pracují. Novější jednotky jsou většinou kompatibilní s MMC a jsou proto podporovány. Pokud verze SCSI určité zapisovací jednotky pracuje, verze IDE/ATAPI bude pravděpodobně pracovat také a naopak. Samozřejmě, někteří lidé chtějí mít příjemný pocit po přečtení konkrétního modelu jejich zapisovací jednotky v nějakém seznamu kompatibilních jednotek. To je důvod, proč jsem nemohl vynechat následující seznam z HOWTO. Zde je kompletní shrnutí podporovaných jednotek:

```
Acer:           CDRW 4432A, CDRW 6206A, CD-R/RW 6X4X32, 8432A
BTC:            BCE 621E (IDE)
Compro:         CW-7502, CW-7502B
Creative:       MK 4211, RW 4224E,
Delta:          OME-W 141
Dysan:          CRW-1622
Elite:          Elite b444.41
Goldstar:       CED-8041B
Grundig:        CDR 100 IPW
Guillemot:      Maxi CD-R 4X/8X
HP:             SureStore 4020i, SureStore 6020i,
                C4324, C4325
                CD-writer+ 7100, 7200i, 7500e, 8100i, 8110i, 8200i Plus,
                           8250i, 9100i, 9110i, 9200e, 9210, 9300i, 9310i
Hi-Val:         CDD 2242, CDD-3610,
Iomega:         ZIPCD 4x650
JVC:            XR-W 2001, XR-W 2010, XR-W 2040, XR-W 2042, XR-RW 2224,
                YR 2626
Kiss:           CDRW (model nezveden)
Kodak:          PCD 200, PCD 225, PCD 260, PCD 600
```

Matsushita:	matsushita je dostupná pro panasonic, podívejte se tam
Memorex:	CRW-620, CDR-622, CRW-1622, CRW-2224, CDRW-4420
Microboards:	PlayWrite 2000, PlayWrite 4000 RW, PlayWrite 4001 RW
MicroNet:	MasterCD Plus 4x4, MasterCD Plus 4x6
Mitsubishi:	CDRW-226
Mitsumi:	CR-2401-TS, CR-2600 TE, CR-2801 TE, CR-4801 TE, CR-4802 TE, CR-4804 TE
Nomai:	680.RW
Olympus:	CDS 615E, CDS 620E
Optima:	DisKovery 650 CD-R
OTI:	CDRW 965, CDRW 975 (Socrates 1.0)
Panasonic:	CW-7285, CW-7502, CW-7503, CW-7582
Philips:	CDD-521/10, CDD-522, CDD-2000, CDD-2600, CDD-3600, CDD-3610, CDD 4201 PCA 267cr, PCA 460 RW, PCRW 404, Omniwriter 26, Omniwriter 26A, CDRW800
Pinnacle:	RCD-100, RCD-1000, RCD-5020, RCD-5040
Pioneer:	DW-S114X
Plasmon:	CDR 480, CDR 4220, RF-4100, RF-4102, CDR 4400
Plextor:	CDR PX-24 CS, PX-412 C, PX-R412 C PX-R 810Ti, PX-R 820T, PX-W 4220Ti, PX-W 8220T, PX-W 8432T Plexwriter RW 4/2/20
Procom:	PCDR 4
REC:	820s
Ricoh:	RO-1420C+, MP 1420C, MP 6200S, MP 6201S, MP 7040A, MP-7060A
Samsung:	SW-204
Sanyo:	CRD-R24S
Smart and Friendly:	CD-RW 226, CD-R 1002, CD-R 1002/PRO, CD-R 1004, CD-R 2004, CD-R 2006 PLUS, CD-R 2006 PRO, CD-RW 2224, CD-R 4000, CD-R 4006, CD-R 4012, CD-RW 4424A CD-R 8020, CD-R 8220
Sony:	CDRX 100E, CDRX 120E, CDRX 140S-RP, CDU 920S, CDU 924, CDU 926S, CDU 928E, CDU 948S
Taiyo Yuden:	EW-50
TEAC:	CD-R50S, CD-R55S, CDR-55S, CDR-55K, CDR-56S-400, CD-R56S-600, R56S-614
Traxdata:	CRW 2260, CDR 4120, CDR 4120 Pro, CDRW 4260, CDRW 4424, CDR 4800
Turtle Beach:	2040R
Waitec:	wt 2036, wt 2444ei
WPI (Wearnes):	CDRW-622, CDR-632P
Yamaha:	CDR-100, CDR 102, CDR-200, CDR-200t, CDR-200tx CDR-400, CDR-400c, CDR-400t, CDR-400tx, CDR-400Atx CDW-2216E, CRW-2260, CRW-2260t, CRW-4250tx, CRW-4260t, CRW-4260tx, CRW-4261, CRW-4416S, CRW-6416S, CRW-8424E

Tabulka 1: Zapisovací CD mechaniky podporované systémem Linux

Detailní seznam modelů, které byly zjištěny jako funkční nebo nefunkční v různých unixových systémech je k dispozici online na adrese http://www.guug.de:8080/cgi-bin/winni/lsc-orig.pl.

Pokud váš hardware není podporován, můžete stále vytvářet obraz CD v systému Linux. Možná to tak budete chtít udělat, protože většina vypalovacího software pro DOS nepracuje s rozšířeními RockRidge (unixové souborové systémy na CD-ROM). V dalším kroku můžete vypálit obraz na CD-R v softwaru pro DOS nebo Macintosh.

Podporované funkce

Existují dvě třídy nástrojů: ovladače hardwaru a software pro formátování dat. Ovladače hardwaru podporují následující funkce:

Podporovaná funkce	cdwrite-2.1	cdrecord-1.6	cdrdao
IDE/ATAPI	ano	ano	ano
Paralelní Port	ne	ano	ano
CD-RW	ne	ano	ano
Audio CD	ano	ano	ano
Datový CD-ROM	ano	ano	částečně
Multisession	částečně	ano	ne
TAO (track-at-once)	ano	ano	ano
DAO (disk-at-once)	ne	částečně	ano
Packetový zápis	ne	ne	ne

`cdwrite` je neudržovaný software uvedený pouze pro úplnost. Používejte místo něj prosím `cdrecord`, jelikož podporuje širší paletu hardwaru a má výrazně více funkcí. Hlavní výhodou cdrdao je schopnost vytváření hudebních CD bez dvou sekund ticha mezi stopami (zápisem v režimu disk-at-once; DAO.

Nástroje, klasifikované jako formátovače dat (data-formatters) organizují data na médiu (vkládají na něj systém souborů).

Funkce	mkisofs	mkhybrid	mkvcdfs
ISO 9660	ano	ano	ne
RockRidge	ano	ano	ne
El Torito	ano	ano	ne
HFS	ne	ano	ne
Joliet	ano	ano	ne
Multisession	ano	ano	ne
CD-Extra	ano	ano	ne
Video-CD	ne	ne	ano

Největším rozdílem mezi souborovými systémy ISO 9660 a ReiserFS nebo Ext -2: je, že jakmile jsou zapsány, nemůžete upravovat soubory. Další omezení souborového systému ISO-9660 zahrnují:

- pouze 8 povolených úrovní podadresářů (počítáno od adresáře nejvyšší úrovně CD)
- maximální délka názvů souborů: 32 znaků
- kapacita 650 MB

RockRidge je rozšíření, které umožňuje delší názvy souborů a hlubší hierarchii adresářů než souborový systém ISO-9660. Při čtení CD-ROM s rozšířením RockRidge pod systémem Linux se objevují všechny známé vlastnosti souborů, jako jsou vlastník, skupina, oprávnění a symbolické odkazy (jako unixový souborový systém). Tato rozšíření nejsou dostupná, když čtete CD-ROM pod DOSem nebo heterogenní skupinou operačních systému rodiny Windows.

El Torito lze použít pro vytváření zaveditelných CD-ROM. Aby tato funkce pracovala, musí ji podporovat BIOS vašeho PC. Nepřesně řečeno, prvních 1.44 (nebo 2.88, pokud je podporováno) Mbajtů CD-ROM obsahuje obraz vámi zadané diskety. Tento obraz je BIOSem zpracováván jako disketa a je z něj spuštěn systém (v důsledku toho, když spouštíte z této virtuální diskety, nemusí být vaše původní jednotka A: (/dev/fd0) přístupná).

HFS umožňuje počítačům Macintosh načítat CD-ROM jako by to byl svazek HFS (nativní souborový systém MacOS).

Joliet přináší (mezi jinými věcmi) dlouhé názvy souborů pro novější varianty Windows (95, 98, NT). Autor samozřejmě nezná žádný nástroj, který by umožňoval dlouhé názvy souborů v samotném DOSu nebo Windows 3.11.

Video-CD mohou být přímo přehrávána na jednotkách DVD.

V části 2.8 je uveden seznam dostupnosti zmíněného softwaru.

E-mailové diskusní skupiny

Pokud se chcete připojit k vývojovému týmu (se záměrem jim aktivně *pomáhat*), odešlete e-mail na cdwrite-request@other.debian.org a napište do těla zprávy slovo `subscribe`.

Nastavení systému Linux pro vytváření CD-ROMů

Tato část se týká následujících druhů zapisovacích mechanik: SCSI, IDE/ATAPI a zařízení pro paralelní port. Zapisovací mechaniky USB do května 2000 nebyly podporovány. Zapisovací mechaniky jiné než SCSI vyžadují ovladače pro kompatibilitu, což umožní, aby se objevovaly jako skutečná zařízení SCSI. Na jednu stranu je taková unifikující strategie jednoduchá („všechno je SCSI"), protože na úrovni aplikace můžete sdílet své znalosti s ostatními uživateli bez ohledu na druh jejich zapisovací mechaniky. Na druhou stranu, musíte překonfigurovat aplikace jako přehrávače hudebních CD nebo nástroj pro připojování, aby odpovídaly změně názvu ovladače. Například, pokud jste dříve prováděli přístup k vaší zapisovací jednotce ATAPI skrze soubor zařízení /dev/hdc, po aktivování ovladačů kompatibility s SCSI k ní budete muset přistupovat skrze /dev/scd0.

Jakmile jste úspěšně nastavili váš hardware a zbytek vašeho systému Linux, můžete pomocí příkazu `cdrecord -scanbus` zobrazit seznam zařízení na vašich sběrnicích SCSI. Cílem této části je provést vás nastavením vašeho systému Linux, abyste nakonec viděli něco takového:

```
shell> cdrecord -scanbus
Cdrecord release 1.7a1 Copyright (C) 1995-1998 Jörg Schilling
scsibus0:
      0,0,0) 'Quantum ' 'XP34300         ' 'F76D' Disk
      0,1,0) 'SEAGATE ' 'ST11200N        ' '8334' Disk
      0,2,0) *
      0,3,0) 'TOSHIBA ' 'MK537FB/        ' '6258' Disk
      0,4,0) 'WANGTEK ' '5150ES SCSI 36  ' 'ESB6' Removable Tape
      0,5,0) 'EXABYTE ' 'EXB-8500-85QUE  ' '0428' Removable Tape
      0,6,0) 'TOSHIBA ' 'XM-3401TASUNSLCD' '3593' Removable CD-ROM
      0,7,0) *
scsibus1:
      1,0,0) 'Quantum ' 'XP31070W        ' 'L912' Disk
      1,1,0) *
      1,2,0) *
      1,3,0) 'TEAC    ' 'CD-R55S         ' '1.0H' Removable CD-ROM
      1,4,0) 'MATSHITA' 'CD-R   CW-7502  ' '4.02' Removable CD-ROM
      1,5,0) *
```

```
      1,6,0) 'YAMAHA  ' 'CDR400t          ' '1.0d' Removable CD-ROM
      1,7,0) *
```

Výpis 1: Detekování zařízení na vaší sběrnici SCSI

Příklad poskytl Jörg Schilling a zobrazuje celkem čtyři zapisovací mechaniky. Všimněte si, že -scanbus také vypisuje ostatní zařízení, např. normální jednotky CD-ROM a jednotky pevných disků. Poslední sloupec uvádí popis zařízení SCSI, podle kterého nemůžete zřetelně odlišit obyčejné jednotky CD-ROM od těch zapisovacích. Ale identifikace produktu (prostřední sloupec) často obsahuje zmínky o funkci ve tvaru R, -R nebo -RW.

Rychlý start

Tato část je pokusem o rychlý a jednoduchý popis konfigurace. Ne všechna možná nastavení jsou popsána, ale jděte na to a v každém případě to vyzkoušejte. Nejdříve ze všeho se podívejte na verzi jádra operačního systému Linux vypsanou příkazem „uname -r“. Může to být něco jako 2.0.X nebo 2.2.Y, kde X je vyšší než 36 a Y je vyšší než 11. Pokud máte starší verze nebo vývojové jádro, je to na vás. Instalace nového jádra je asi tolik práce jako oprava toho starého, takže jsem odstranil všechny pokyny, které jsou zapotřebí pro špatná jádra.

Níže uvedený seznam obsahuje sadu příkazů, kterými byste měli začít. Příkazy vytváří soubory zařízení v /dev, pokud ještě neexistují.

```
test 'whoami' = 'root' || echo "Pro spuštění těchto příkazů musíte být root."
cd /dev/
umask -S u=rwx,g=rwx,o-rwx
[ -f loop0 ] \
    || ./MAKEDEV loop \
    || for i in 0 1 2 3 4 5 6 7; do mknod loop$i b 7 $i; done
[ -f sg0 -o -f sga ] \
    || ./MAKEDEV sg \
    || for i in 0 1 2 3 4 5 6 7; do mknod sg$i  c 21 $i; done
```

Výpis 2: Vytváření souborů zařízení

Přístup k hardwaru je v systému Linux obvykle implementován pomocí souborů zařízení. Takže než budete dělat něco dalšího, ujistěte se, že tyto soubory v adresáři /dev existují. Pořád mi nikdo neuměl sdělit důvod, proč to nebylo zautomatizováno pomocí technik jako jsou souborové systémy zařízení (device filesystem; devfs). Devfs je k dispozici roky, přináší bezpečnější (!) a mnohem čistší pojmenování zařízení a vytváří položky v /dev automaticky. Někteří prominenti argumentují tím, že devfs není perfektní řešení, ale nepřišli s ničím lepším, ani ničím srovnatelným a v neposlední řadě není nyní k dispozici nic a nic se netestuje. Začněme pomocí devfs, takže mohu odstranit výše uvedené příkazy z tohoto dokumentu (http://www.atnf.CSIRO.AU/~rgooch/linux/kernel-patches.html).

Další věcí, kterou je potřeba zajistit, je to, aby bylo jádro systému Linux vybaveno nezbytnými ovladači. Následující příkazy kontrolují přítomnost různých souborů ovladačů ve spuštěném jádře systému Linux. Obvykle by měl příkaz „cdrecord -scanbus“ zapnout automatické načítání všech ovladačů. V případě, že ovladač pak v jádře není, je to oznámeno a je ručně načten modularizovaný ovladač (modul) skrze insmod.

```
test 'whoami' = 'root' || echo "Pro provedení těchto příkazů musíte být root."
cdrecord -scanbus > /dev/null
if ! (pidof kerneld || test -f "/proc/sys/kernel/modprobe"); then
    echo "Ani kerneld, ani kmod neběží, takže moduly nelze načíst automaticky
".
fi
report_no_autoload() {
    echo "Příště si ověřte, že je načten modul $1."
}
if test ! -f "/proc/scsi/scsi"; then
    report_no_autoload scsi_mod  &&  insmod scsi_mod
fi
if ! grep "^........ sg_" /proc/ksyms > /dev/null; then
    report_no_autoload sg  &&  insmod sg
fi
if ! grep "^........ sr_" /proc/ksyms > /dev/null; then
    report_no_autoload sr_mod  &&  insmod sr_mod
fi
if ! grep "^........ loop_" /proc/ksyms > /dev/null; then
    report_no_autoload loop  &&  insmod loop
fi
if ! grep iso9660 /proc/filesystems > /dev/null; then
    report_no_autoload iso9660  &&  insmod iso9660
fi
echo "Následující část je potřeba pouze pro zapisovací mechaniky IDE/ATAPI."
if ! grep ide-scsi /proc/ide/drivers > /dev/null; then
    report_no_autoload ide-scsi  &&  insmod ide-scsi
fi
cdrecord -scanbus
```

Výpis 3: Testování ovladačů

Pokud insmod nadává na chybějící soubory modulu, přečtěte si prosím následující kapitolu. Pokud jste v textovém režimu (konsole), může načítání modulů vyvolat zobrazení některých zpráv na obrazovce. Pokud jste v grafickém režimu (X11, KDE, Gnome), můžete se k těmto zprávám vrátit příkazem dmesg.

Existuje několik způsobů načtení modulů při příštím spuštění systému Linux:

```
(1) Vložení relevantního příkazu insmod do spouštěcí sekvence
    (skript příkazového interpretu s názvem rc.local nebo jeho ekvivalent).
(2a) Spuštění kerneld nebo kmod a
(2b) jejich konfigurace v /etc/modules.conf (abych byl přesnější,
     konfigurujete nástroj modprobe, který je volán démony)
```

Lidé se zapisovací mechanikou SCSI mohou přeskočit zbytek této části, protože cdrecord bude pravděpodobně stejně jejich hardware detekovat. Pokud ne, pak mi prosím pošlete e-mail s nějakými informacemi o vašem nastavení, abych mohl vylepšit část o zapisovacích jednotkách SCSI.

Nyní k lidem se zapisovacími mechanikami pro IDE/ATAPI. Jak bylo napsáno v předchozí kapitole, musíte načíst ovladač kompatibility ide-scsi. Ale tento ovladač má přístup k vaší zapisovací mechanice pouze pokud zde není jiný ovladač, který to již dělá. Jinými slovy, musíte říci vašemu ovladači IDE, aby ponechal vaši zapisovací mechaniku nerozpoznanou, aby ji mohl převzít ovladač ide-scsi.

```
hda = sběrnice IDE/konektor 0 zařízení master
hdb = sběrnice IDE/konektor 0 zařízení slave
hdc = sběrnice IDE/konektor 1 zařízení master
hdd = sběrnice IDE/konektor 1 zařízení slave
```

Výše uvedená tabulka zobrazuje vztah názvů souborů zařízení a umístění zařízení na sběrnicích IDE. Název souboru zařízení reprezentující vaši zapisovací mechaniku musí být předán ovladači v jádře systému Linux. Příklad: hdb=ide-scsi. Takové nastavení by mělo být přidáno do lilo.conf nebo chos.conf pokud je ovladač staticky zakompilován do vašeho jádra, což se zdá být nejobvyklejším nastavením. Pokud potřebujete předat jádru více než jeden parametr, pak je oddělte mezerami (jak bylo zobrazeno v příkladu chos). Následující dva výpisy zobrazují příklady konfigurací obsahujících více řádku než relevantní řádek append. Všimněte si prosím, že položky append- a cmdline- jsou určeny pro jednotlivé druhy jádra (tj. nepřidávejte je přímo na začátek).

```
image=/boot/zImage-2.2.14
  label=Linux
  read-only
  append="hdb=ide-scsi"
```

Výpis 4: Příklad konfigurace pro lilo (/etc/lilo.conf)

```
linux "Linux 2.1.14" {
        image=/boot/zImage-2.0.37
        cmdline= root=/dev/hda5 readonly hdb=ide-scsi
}
```

Výpis 5: Příklad konfigurace pro chos (/etc/chos.conf)

Pokud je ovladač pro IDE/ATAPI CD-ROM načten jako modul, pak se výše uvedené pro vás nebude lišit, ale ujistěte se, že jste vložili řádek options z následujícího výpisu. Poslední tři řádky výpisu jsou obecně doporučovány pro lepší automatizaci načítání požadovaných modulů.

```
options ide-cd ignore=hdb            # říká modulu ide-cd, aby ignoroval hdb
alias scd0 sr_mod                    # načtení sr_mod při přístupu k scd0
#pre-install ide-scsi modprobe imm    # odkomentujte pouze pro některé jednotky
ZIP
pre-install sg     modprobe ide-scsi # načíst ide-scsi před sg
pre-install sr_mod modprobe ide-scsi # načíst ide-scsi před sr_mod
pre-install ide-scsi modprobe ide-cd # načíst ide-cd   před ide-scsi
```

Výpis 6: Příklad konfigurace pro /etc/modules.conf

Pokud je vaše zapisovací mechanika jedinou mechanikou CD-ROM připojenou k vašemu počítači, pak si pamatujte, že máte přístup k jednotce CD-ROM v zapisovači pomocí souboru zařízení /dev/scd kde =0,..,8. Možná budete chtít změnit symbolický název zařízení cdrom, aby ukazoval na nový soubor zařízení. Níže uvedený výpis ukazuje příkaz, kterým toho dosáhnete pro scd0.

```
cd /dev && rm cdrom && ln -s scd0 cdrom
```

Výpis 7: Nastavení cdrom jako symbolického názvu pro scd0

Pokud jsou vaše zapisovací mechanika a jednotka CD-ROM dvě různá zařízení, pak symbolický odkaz cdrom neměňte.

Speciální poznámky k zapisovacím mechanikám SCSI

Ujistěte se prosím, že je vaše zapisovací jednotka rozpoznána BIOSem vaší karty hostitelského adaptéru SCSI. Každý hostitelský adaptér SCSI zkoumá po zapnutí sběrnici SCSI a hlásí všechna nalezená zařízení připojená ke sběrnici. Zpráva zahrnuje SCSI ID zařízení a označení produktu. Nemá smysl nic provádět, dokud není vaše zapisovací jednotka uvedena v této zprávě.

Pokud chcete připojit vaše zařízení SCSI přes paralelní port (nepleťte si to s jednotkami IDE pro paralelní port), potřebujete speciální aktivní kabel a speciální ovladač jádra. O této možnosti se dozvíte více na adrese http://www.torque.net/parport/parscsi.html.

Speciální poznámky k zapisovacím mechanikám pro paralelní port

Promiňte, ale nic o tom nevím. Přečtěte si prosím http://www.torque.net/parport/paride.html nebo váš lokální soubor /usr/src/linux/Documentation/paride.txt.

Kompilace chybějících modulů jádra (volitelně)

Tuto část nemusíte číst, pokud je váš hardware úspěšně rozpoznán a nastaven výše popsanými kroky konfigurace.

Jádro systému Linux může být vybaveno ovladači pro různé funkce. Ovladače do jádra můžete zakompilovat staticky nebo je můžete zkompilovat jako modul pro načítání na vyžádání. Druhá metoda je preferována pro ovladače, které nejsou absolutně nutné pro uvedení vašeho systému Linux do života, protože pak bude vaše jádro menší a rychlejší. Samozřejmě, některé ovladače jsou nezbytné pro spuštění systému a neměli byste je kompilovat jako modul. Příklad: pokud je váš systém uložen na pevném disku IDE, musíte mít ovladač pro pevné disky IDE v jádře – nikoliv jako modul.

Existují tři různé druhy zapisovacích mechanik: SCSI, IDE/ATAPI a externí zapisovací mechaniky, které pracují přes paralelní port. Tabulka zobrazuje jak nastavit jádro systému Linux pro tyto druhy hardware. První sloupec tabulky je část konfigurační nabídky jádra, kde můžete nalézt nastavení. Druhý sloupec je popis funkce (převzatý také z konfigurační nabídky jádra). Třetí sloupec udává název vyplývajícího modulu. Sloupce s názvy SCSI, IDE a PP obsahují nezbytné volby pro asociovaný hardware (PP = paralelní port).

```
Odd.   Popis                                    Modul    SCSI  IDE   PP
-----------------------------------------------------------------
BLOCK  Enhanced IDE/MFM/RLL...                                 Y
BLOCK  IDE/ATAPI CDROM                          ide-cd         M
BLOCK  SCSI emulation support                   ide-scsi       M
BLOCK  Loopback device                          loop       M   M     M

PARIDE Parallel port IDE device                 paride               Y/M
PARIDE Parallel port ATAPI CD-ROMs                                    M
PARIDE Parallel port generic ATAPI                                    M
PARIDE (vyberet vhodný nízkoúrovňový ovladač)                         Y

SCSI   SCSI support                             scsi_mod   Y/M  Y/M
SCSI   SCSI CD-ROM support                      sr_mod     Y/M  Y/M
SCSI     Enable vendor-specific                            Y    Y
SCSI   SCSI generic support                     sg         Y/M  Y/M
SCSI   (vyberet vhodný nízkoúrovňový ovladač)              Y

FS     ISO 9660 CDROM filesystem   iso9660                 Y/M  Y/M   Y/M
FS     Microsoft Joliet cdrom...   joliet                   Y    Y     Y
```

Tabulka 2: Výběr ovladače pro různé druhy zapisovacích mechanik

Y znamená ano a znamená to, že byste měli vložit tohoto démona do jádra. M znamená modul a to znamená, že byste měli nebo budete muset zkompilovat tuto funkci jako modul. Y/M vám dává na výběr (pořadí indikuje volby s menším počtem potenciálních problémů). Prázdná nastavení není potřeba měnit a jejich ponechání zvyšuje šanci, že výsledné jádro bude pracovat (pokud předtím pracovalo...). Zejména v prostředích, kde jsou zařízení SCSI i ATAPI, je lepší sestavovat většinu věcí jako moduly.

Kompilace zařízení loopback je nepovinná. Umožňuje vám otestovat obraz před jeho zápisem na médium. Pokud chcete číst CD-ROMy, potřebujete podporu pro souborový systém ISO 9660. Tento ovladač automaticky zahrnuje rozšíření RockRidge. Rozšíření Microsoft Joliet CD-ROM musí být přidáno do souborového systému ISO 9660 explicitně. V některém případě budete potřebovat pro váš hardware nízkoúrovňový ovladač. Nízkoúrovňový ovladače je ten, který spolupracuje přímo s hardwarem. Pro SCSI a paralelní port existuje mnoho nízkoúrovňových ovladačů.

Instalace výsledného jádra systému Linux přesahuje rámec tohoto dokumentu. Podívejte se, prosím, do dokumentace vaší distribuce systému Linux.

Uživatelé systému RedHat Linux, vězte, že musíte zakompilovat funkce „Ramdisk support“ a „Initial ramdisk“. Kromě toho musíte vygenerovat nový ramdisk s novými moduly spuštěním příkazu jako „mkinitrd – preload ide-cd initrd-2.2.14.img 2.2.14“.

Získání uživatelského softwaru pro vypalování CD-R

Detailnější seznam nástrojů týkajících se produkování CD-ROMů je k dispozici na adrese http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdb.html.

Nástroje příkazového řádku

Jeden z následujících balíčků je potřeba pro generování obrazů CD-R (vyžadováno pouze pro datové CD-ROMy):

ftp://tsx-11.mit.edu/pub/linux/packages/mkisofs/ (mkisofs)

ftp://ftp.ge.ucl.ac.uk/pub/mkhfs (mkhybrid)

Pro zápis obrazů na CD-R potřebujete jeden z následujících softwarových balíčků:

ftp://ftp.fokus.gmd.de/pub/unix/cdrecord/ (cdrecord)

http://www.ping.de/sites/daneb/cdrdao.html (cdrdao)

http://www.munich-vision.de/vcd/ (mkvcdfs)

Nevěřte manuálové stránce starých verzí `mkisofs`, která tvrdí, že potřebujete verzi 1.5 programu `cdwrite`. Použijte pouze cdrecord a budete v pohodě. Pamatujte si prosím, že novější verze cdrecord jsou dodávány s rozšířenou verzí mkisofs a některé další nástroje jsou v podadresáři misc/ (readcd, isosize) a nikde jinde.

Grafická uživatelská rozhraní (volitelně)

Rozhraní (Front-end) jsou v Linuxu opravdovým rozhraním. To znamená, že pořád musíte nainstalovat nástroje příkazového řádku, ale budete k nim mít přístup lépe vypadajícím způsobem.

X-CD-Roast je balíček programů určený pro jednoduché vytváření CD v Linuxu. Kombinuje nástroje příkazového řádku jako `cdrecord` a `mkisofs` s hezkým grafickým uživatelským rozhraním.

http://www.fh-muenchen.de/home/ze/rz/services/projects/xcdroast/e_overview.html

BurnIT je rozhraní napsané v jazyce JAVA pro cdrecord, mkisofs a cdda2wav-0.95, čímž tvoří kompletní balíček pro vypalování CD na unixové platformě. Je k dispozici na adrese

http://sunsite.auc.dk/BurnIT/

CD-Tux je textově orientované rozhraní pro programy mkisofs a cdrecord. „Vytváří jednoduše použitelné prostředí pro skoro všechno, co byste mohli dělat s CD v plných barvách pomocí (i nechvalně) známé knihovny NCURSES. A všechno tohle dělá spustitelný soubor menší než 75k.

http://www.datadictator.co.za/cdtux/

Vypalování CD-R

Vytváření CD-ROMů se v systému Linux skládá ze dvou kroků:

- zabalení požadovaných dat (souborů, hudby nebo obojího) do souborů se speciálními formáty
- zápis dat ze souborů na CD-R pomocí utility `cdrecord`

Tato kapitola detailně popisuje kroky pro datové a hudební CD.

Vytváření CD-ROMů (jen data)

Pamatujte si, že shromažďování dat, která chcete umístit na CD, obvykle trvá déle než by jeden čekal. Uvědomte si, že chybějící soubory nelze přidat na CD, pokud je zapsáno a ukončeno. To platí i ohledně CD-RW, které může být přepsáno pouze celé. Použití funkce multi-session není volbou pro jednotlivé soubory, protože zabírá moc místa pro nový kompletní obsah (table of contents; TOC). UDF na Linuxu ještě není připraveno.

Také pamatujte na to, že je určité množství volného místa na CD použito pro uložení informací o souborovém systému ISO-9660 (obvykle několik MB). 620 MB dat se vždy na 650MB CD-R vejde.

Vytváření obrazu pozdějšího CD-ROM

Než může být nějaké médium úložiště (disketa, pevný disk nebo CD) použito, musí na něm být systém souborů (DOSovsky řečeno: musí být zformátováno). Systém souborů odpovídá za organizaci a začleňování souborů, které by měly být uloženy na médiu.

Obvyklé utility pro vytváření souborových systémů na oddílech pevného disku na ně zapisují prázdný systém souborů, který je pak připojen a podle potřeby zaplněn soubory uživatelů. Na CD se dá zapsat jen jednou, takže když jsme psali o prázdném systému souborů, bylo by zformátováno a zůstalo úplně prázdné navždy. To platí i pro přepisovatelná média, jelikož nemůžete libovolně měnit sektory; musíte vymazat celý jejich obsah.

Takže to, co potřebujeme je nástroj, který vytváří systém souborů při kopírování souborů na CD. Tento nástroj se nazývá `mkisofs`. Příklad využití:

```
mkisofs  -r   -o cd_image   private_collection/
              '----------'  '------------------'
                   |              |
            zapsat výstup do  vzít adresář jako vstup
```

Volba '-r' nastavuje oprávnění pro všechny soubory, aby byly veřejně dostupné na CD a zapíná rozšíření RockRidge. Pravděpodobně nechcete tuto volbu použít dokud nevíte, co opravdu dělá (pozn.: bez '-r' připojovací body převezmou oprávnění z `private_collection`!).

`mkisofs` se pokusí mapovat všechny názvy souborů na formát 8.3 používaný systémem DOS pro zajištění nejvyšší možné kompatibility. V případě konfliktu názvů (různé soubory mají stejný název 8.3) se použijí v názvech souborů čísla a informace o zvoleném názvu je vypsána na STDERR (obvykle na obrazovku). Nepanikařte: V Linuxu nikdy neuvidíte tyto divné názvy souborů 8.3, protože Linux používá rozšíření Rock Ridge, která obsahují informace o původním souboru (oprávnění, název souboru, atd.).

Nezapomeňte použít volbu -J (rozšíření MS Joliet) nebo použít mkhybrid pokud chcete generovat CD-ROM přátelštější pro Windows. Pro CD-ROMy HFS používané na počítačích Macintosh, byste si měli spíše přečíst manuálovou stránku větší sestřičky mkisofs „mkhybrid", kde najdete detaily k různým volbám.

Nyní byste se mohli divit, proč není výstup `mkisofs` přímo odeslán na zapisovací zařízení. Existují tři důvody:

- `mkisofs` neví nic o ovládání zapisovacích jednotek CD.
- Mohli byste chtít obraz před jeho vypálením otestovat.
- Na pomalých počítačích by to nebylo spolehlivé.

Existuje metoda zápisu na CD-R najednou, která bude popsána níže.

Někdo by také mohl přemýšlet o vytváření dalšího oddílu a zápisu obrazu do tohoto oddílu místo do souboru. Jsem proti této strategii, protože pokud zapíšete kvůli překlepu na špatný oddíl, můžete přijít o celý systém Linux. Čti: mně se to stalo... Kromě toho je to mrhání místem na disku, protože obraz CD jsou dočasná data, která mohou být po vytvoření CD odstraněna. Samozřejmě, použití základních oddílů šetří čas pro odstraňování souborů o velikosti 650 MB.

Testování obrazu CD

Linux umí připojovat soubory jako by to byly oddíly disku. Tato funkce je užitečná pro kontrolu, že rozvržení adresáře a oprávnění k přístupu k souborům obrazu CD odpovídá vašim požadav-

kům. Ačkoliv je dnes médium velmi levné, proces zápisu pořád nějakou dobu trvá a mohli byste chtít třeba alespoň ušetřit čas pomocí rychlého vyzkoušení.

Pro připojení souboru `cd_image` vytvořeného výše na adresář `/cdrom` zadejte příkaz

```
mount -t iso9660 -o ro,loop=/dev/loop0 cd_image /cdrom
```

Nyní se můžete podívat na soubory v adresáři `/cdrom` – objevují se přesně jako by byly na skutečném CD. Pro odpojení obrazu CD jednoduše zadejte `umount /cdrom` (Varování: na jádrech systému Linux před verzí 2.0.31 nemusí být poslední soubor v `/cdrom` plně čitelný. Použijte prosím novější jádro jako např. 2.0.36. Volba -pad pro cdrecord se týká pouze hudebních CD a volba -pad pro mkisofs vyžaduje cestu, kterou je těžší zadávat než inovovat na bezchybné jádro).

> **POZNÁMKA:** Některé archaické verze `mount` neumí pracovat se zařízeními loopback. Pokud máte takovou starou verzi `mount`, pak inovujte svůj systém Linux. Někteří lidé také doporučovali doplnění informací o tom, jak získat nejnovější nástroje pro připojování, do tohoto dokumentu. Vždycky to odmítám. Pokud distribuce vašeho systému Linux obsahuje zastaralý `mount`, nahlašte to jako chybu. Pokud není distribuce Linuxu snadno inovovatelná, nahlašte to jako chybu.

Pokud bych přidal všechny informace, které jsou potřeba pro odstranění chyb ve špatně navržených distribucích systému Linux, byl by tento praktický návod o hodně delší a bylo by těžší jej číst.

Zápis obrazu CD na CD

Tato část zahrnuje pouze zápis dat na CD v režimu TAO, protože je to nejpoužívanější režim pro data. Více informací o rozdílech režimů TAO a DAO najdete v kapitole o hudebních CD-R. Pokud používáte režim DAO s nástrojem cdrdao, pak nezapomeňte přidat fiktivní hudební stopu na konec souboru TOC (viz README).

Už toho moc nezbylo. Pokud jste to ještě nezkoušeli, tak teď je ten správný čas pro příkaz

```
cdrecord -scanbus
```

To vám řekne, ke kterému zařízení SCSI patří vaše zapisovací jednotka CD. Všechny ostatní metody odhadování informací, které dobře popisuje cdrecord, byly z tohoto HOWTO odstraněny.

Před předvedením posledního příkazu mi dovolte, abych vás varoval, že zapisovací jednotky CD chtějí být krmeny konstantním proudem dat. Takže proces zápisu obrazu CD na CD nesmí být přerušen, jinak vytvoříte poškozené CD. Datový proud je jednoduché přerušit odstraněním velmi velkého souboru. Příklad: pokud odstraníte starý obraz CD o velikosti 650 Mbajtů, musí jádro inovovat informace o 650,000 bloků na pevném disku (za předpokladu, že váš systém souborů má velikost bloku 1 Kbajt). To zabere nějaký čas a velmi pravděpodobně to zpomalí činnost disku na moc dlouho a vznikne tak několikasekundová časová mezera v proudu dat. Čtení pošty, procházení na Internetu nebo dokonce kompilace jádra samozřejmě na moderních počítačích obecně neovlivní proces zápisu.

Pamatujte si, prosím, že žádná zapisovací jednotky nemůže vrátit svůj laser a pokračovat od původního bodu na CD, ve kterém byla přerušena. Proto silné vibrace nebo jiné mechanické rázy pravděpodobně zničí CD, které vytváříte.

Když už jste duševně připraveni, oblečte se do černé róby, vynásobte identifikátor SCSI zapisovací jednotky CD jeho verzí SCSI a zapalte tolik svíček, vyslovte dvě verze ASR-FAQ (diskusní skupina alt.sysadmin.recovery) a nakonec napište:

```
shell> SCSI_BUS=0   # převzato z výpisu 1 "scsibus0:"
shell> SCSI_ID=6    # převzato z výpisu 1 "TOSHIBA XM-3401"
shell> SCSI_LUN=0
shell> cdrecord -v speed=2 dev=$SCSI_BUS,$SCSI_ID,$SCSI_LUN \
                  -data  cd_image

# stejné jako výše, ale kratší:
shell> cdrecord -v speed=2 dev=0,6,0  -data  cd_image
```

Pro lepší čitelnost jsou parametry zapisovací jednotky uloženy do tří proměnných prostředí s názvy: SCSI_BUS, SCSI_ID, SCSI_LUN.

Pokud používáte cdrecord pro přepis CD-RW, musíte přidat volbu „blank=...“ pro odstranění starého obsahu. Přečtěte si prosím manuálovou stránku, kde se dozvíte více o různých metodách vymazání CD-RW.

V době, kdy všichni kromě mě měli 400MHz počítač, lidé zadávali výstup z mkisofs přímo do cdrecord:

```
shell> IMG_SIZE='mkisofs -R -q -print-size private_collection/  2>&1 \
| sed -e "s/.* = //"'
shell> echo $IMG_SIZE
shell> [ "0$IMG_SIZE" -ne 0 ] && mkisofs -r  private_collection/  \
|cdrecord  speed=2  dev=0,6,0
            tsize=${IMG_SIZE}s  -data  -
#       nezapomeňte na --^         ^-- čti data ze STDIN
```

První příkaz je prázdné spuštění pro zjištění velikosti obrazu (aby to fungovalo, potřebujete mkisofs z distribuce cdrecord). Je potřeba zadat všechny parametry, které použijete při konečném spuštění (např. -J nebo -hfs). Vaše zapisovací jednotka možná nepotřebuje znát velikost obrazu, který má být zapsán, takže můžete toto spuštění vypustit. Vypsaná velikost musí být předána jako parametr tsize do cdrecord (je uložena v proměnné prostředí IMG_SIZE). Druhý příkaz je sekvence mkisofs a cdrecord, spojená rourou.

Vytváření hudebních CD

Vytváření hudebních CD je velmi podobné krokům popsaným výše pro datová CD. Můžete si vybrat mezi dvěma technikami: DAO nebo TAO. TAO (stopa najednou – track at once) se méně hodí pro hudbu, protože uslyšíte lupnutí mezi jednotlivými stopami. TAO byla popsána jako první, protože je o něco jednodušší a DAO není ještě k dispozici pro všechny jednotky.

Hlavní rozdíl oproti zapisování datových CD-R je formát obrazů. ISO-9660 (nebo jakýkoliv systém souborů, kterému dáváte přednost) nelze použít, protože žádný přehrávač hudebních CD neumí pracovat se systémy souborů. Namísto toho musí být hudební data zapsána jako „16bitové stereo vzorky v kódování PCM při 44100 vzorcích/sekundu (44,1 kHz)“.

Jedním nástrojem pro konverzi vašich zvukových souborů do požadovaného formátu je sox. Jeho používání je jednoduché:

```
shell> sox  killing-my-software.wav  killing-my-software.cdr
```

Tento příkaz by zkonvertoval píseň killing-my-software z formátu WAV do CDR s hudebním formátem. Podívejte se na manuálovou stránku sox, kde najdete detailní informace o formátech a rozšířeních systému souborů, které sox zná. Protože výstup ruční konverze zabírá moc místa na disku, byla vytvořena funkce vestavěná do cdrecord pro hudební formáty WAV a AU. Takže pokud

vaše hudební soubory mají přípony .wav nebo .au (a vzorkování „stereo, 16 bitů, 44,1 kHz"), můžete je použít jako hudební stopy bez ruční konverze do formátu CDR. cdrecord samozřejmě vyžaduje, aby velikost hudebních dat byla celé číslo násobek 2352 a aby byla větší než 705 600 bajtů, což není u některých souborů WAV splněno. Pro použití takových souborů v sox je potřeba doplnit hudebními daty na 2352 bajtů.

Zapisování hudebních CD (TAO)

Hudební CD se skládá z hudebních stop, které jsou v režimu TAO uspořádány jako samostatné obrazy. Takže pokud chcete mít na vašem CD deset stop, musíte vytvořit deset obrazů.

Cdrecord zapisuje obrazy CD jako hudební stopy pokud je uvedena volba -audio. Další volby jsou shodné s těmi, které se používají při vytváření datových CD (pokud nemáte velmi speciální požadavky). Tyto tři příklady dělají všechny to samé, ale čtou stopy z různých formátů zvukových souborů:

```
shell> cdrecord -v speed=2 dev=0,6,0  -audio  track1.cdr track2.cdr...
shell> cdrecord -v speed=2 dev=0,6,0  -audio  track1.wav track2.wav...
shell> cdrecord -v speed=2 dev=0,6,0  -audio  track1.au  track2.au...
```

Tímto vytvoříte hudební CD, které bude mít 2sekundové pauzy mezi hudebními stopami. Jedním zvláštním formátem, který není přímo čitelný v cdrecord, je MPEG Layer 3. Pro konverzi souborů v tomto formátu na formát CDR můžete použít příkaz „mpg123 –cdr - track1.mp3 > track1.cdr". Volba --cdr zajišťuje, aby byla stopa kódována v požadovaném formátu (viz výše). Starší verze mpg123 vyžadují -s namísto jednoduchého – pro zápis na standardní výstup. Jiný směr konverze z WAV do MPEG může být proveden pomocí LAME pro soubory WAV (extrahujte stopu z hudebního CD pomocí cdda2wav a zakódujte ji pomocí LAME do MP3).

Pro vytvoření CD-R z více souborů MP3 můžete použít následující sekvenci příkazů:

```
for I in *.mp3
do
    mpg123 --cdr - "$I" | cdrecord -audio -pad -nofix -
done
cdrecord -fix
```

V závislosti na rychlosti vašeho počítače možná budete chtít zpomalit zápis na „speed=1" (volba cdrecord). Pokud používáte „speed=4", musí být váš počítač schopen přehrávat soubor MP3 čtyřnásobnou rychlostí. mpg123 spotřebovává mnoho času procesoru! Pokud si nejste jisti, zkuste příkaz spustit naprázdno s -dummy (ponechá laser vypnutý).

DAO

Pokud se chcete zbavit mezer mezi hudebními stopami, musíte místo výše popsaného režimu TAO použít zápis DAO (celý disk najednou – disk-at-once). Podpora DAO je aktuálně lepší v cdrdao. Podívejte se prosím na detailní informace na jeho domovské stránce.

Pokud připravujete CD v režimu DAO, pak použijete jednolitý obraz (zvukový soubor) a budete řídit informace o stopách pomocí konfiguračního souboru.

```
CD_DA
TRACK AUDIO
FILE "live.wav" 0 5:0:0
INDEX 3:0:0
TRACK AUDIO
```

```
FILE "live.wav" 5:0:0 5:0:0
TRACK AUDIO
FILE "live.wav" 10:0:0 5:0:0
INDEX 2:0:0
```

Smíšené CD-ROMy

O tomto tématu se toho nedá říct mnoho. Prostě indikujte druh (následujících) obrazů volbami -data a -audio. Příklad:

```
cdrecord  -v dev=0,6,0  -data  cd_image  -audio  track*.cdr
```

Drahý Winfriede,...

Toto je část, která se obvykle nazývá „často kladené dotazy s odpověďmi“. Pokud máte problém s vaším partnerem, dětmi nebo psem, pošlete je, pokud se vztahují k vytváření CD-R nebo jsou jinak zábavné.

Jak citlivý je proces vypalování?

Zkuste to. Použijte volbu -dummy pro spuštění cdrecord naprázdno. Dělejte všechno, co byste jinak dělali, a sledujte, jestli to proces vypalování přežije.

Pokud jste cdrecord nakrmili přímo z mkisofs, pak náročné diskové procesy jako např. aktualizace databáze *locate*, snižují maximální rychlost toku dat a mohou poškodit CD. Je lepší zkontrolovat, že takové procesy nejsou spuštěny přes `cron`, `at` nebo `anacron`, pokud vypalujete CD-R na starších počítačích.

Má fragmentace souboru špatný vliv na propustnost?

Fragmentace souborů je obvykle tak malá, že její vliv není pozorovatelný. Samozřejmě můžete jednoduše vytvořit patologické případy fragmentace, které sníží propustnost vašich pevných disků pod 100 kbajtů/sekundu. Ale nedělejte to. :-) Ano, soubory na pevném disku se během let stanou fragmentovanými. Čím prázdnější disk, tím rychlejší systém souborů je. Vždy ponechte 10 % nebo 20 % volného místa a mělo by vše běžet s ohledem na vytváření CD dobře.

Pokud si nejste jisti, podívejte se na zprávy vypisované při spouštění. Procento fragmentace je hlášeno při kontrole systémů souborů. Tuto hodnotu můžete zkontrolovat pomocí velmi nebezpečného příkazu

```
shell> e2fsck -n  /dev/sda5        # '-n' je důležité!
[další řádky vynechány -- chyby ignorujte]
/dev/sda5: 73/12288 files (12.3% non-contiguous)
```

V tomto případě fragmentace vypadá velmi vysoká – ale v systému souborů je pouze 73 velmi malých souborů. Takže tato hodnota *není* alarmující.

Existuje experimentální nástroj, který se jmenuje e2defrag a slouží pro defragmentaci systémů souborů extended-2. Aktuální verze nepracuje dostatečně spolehlivě, aby se dala použít i jen pro privátní prostředí. Pokud chcete opravdu defragmentovat váš systém souborů, vytvořte záložní kopii (lepší jsou dvě kopie), procvičte si obnovování dat, pak vytvořte nový systém souborů (který zničí ten starý) a obnovte data. Toto je aktuálně nejbezpečnější technika.

Je možné ukládat obraz CD na systém souborů UMSDOS?

Ano. Jediný systém souborů, který není dostatečně rychlý a spolehlivý pro vytváření CD-ROMů, je *network filesystem* (*NFS*). Já sám jsem používal UMSDOS pro sdílení diskového prostoru mezi Linuxem a DOS/Win na PC (486/66) určeném pro vytváření CD-ROMů.

Neexistuje nějaký způsob pro obejití omezení ISO-9660?

Ano. Na CD můžete umístit systém souborů, který budete chtít. Ale jiné operační systémy než Linux nebudou umět s tímto CD pracovat. Zde je návod:

- Vytvořte prázdný soubor o velikosti 650 MB.

```
dd if=/dev/zero of="empty_file" bs=1024k count=650
```

- Vytvořte v tomto souboru systém souborů ext2

```
shell> /sbin/mke2fs  -b 2048  empty_file
empty_file is not a block special device.
Proceed anyway? (y,n) y
```

- Připojte tento prázdný soubor pomocí zařízení loopback (potřebujete jej připojit; viz výše).

```
mount -t ext2 -o loop=/dev/loop1 empty_file /mnt
```

- Zkoírujte soubory do /mnt a pak soubor odpojte.
- Použijte `cdrecord` na empty_file (který již není prázdný) jako by to byl obraz ISO-9660.

Pokud chcete vytvořit položku v `/etc/fstab` pro takové CD, pak vypněte kontrolu souboru zařízení při spouštění systému. Např.:

```
/dev/cdrom  /cdrom  ext2  defaults,ro  0 0
```

První 0 znamená „zazálohovat pomocí dump“ (záloha), druhá (=důležitá) znamená „při startu nekontrolovat chyby „ (fsck by selhal při kontrole chyb na CD).

Jak přečíst stopy z hudebních CD?

Existuje více softwarových balíčků. Nejnovějším je „cdparanoia“ a může být stažen z

http://www.xiph.org/paranoia/

Nebo můžete vyzkoušet kombinaci „cdda2wav“ a „sox.

`cdda2wav` vám umožňuje dosáhnout specifického intervalu (nebo celé stopy) z vašeho hudebního CD a převést jej do souboru .wav. `sox` konvertuje soubory WAV zpátky do (hudební CD) formátu cdda, aby je bylo možno zapsat na CD-R pomocí `cdrecord`. `sox` nepotřebujete nutně, pokud používáte nejnovější verzi `cdrecord`, protože má vestavěnou podporu souborů .au a .wav.

Jak zkontrolovat zařízení SCSI po spuštění?

Soubor `drivers/scsi/scsi.c` obsahuje informace

```
/*
 * Usage: echo "scsi add-single-device 0 1 2 3" >/proc/scsi/scsi
 * with  "0 1 2 3" replaced by your "Host Channel Id Lun".
 * Consider this feature BETA.
 *     CAUTION: This is not for hot plugging your peripherals. As
```

```
 *     SCSI was not designed for this you could damage your
 *     hardware !
 * However perhaps it is legal to switch on an
 * already connected device. It is perhaps not
 * guaranteed this device doesn't corrupt an ongoing data transfer.
 */
```

Pamatujte si, že by toto mělo být použito pouze pokud přidáváte zařízení SCSI na konec řetězce. Vložení nových zařízení SCSI do stávajícího řetězce ruší přidělování názvů zařízení (adresář /dev) a může zničit kompletní obsah vašeho pevného disku.

Některé verze jádra vůbec nemají rády znovuprohledání sběrnice SCSI a váš systém může solidně zamrznout, když to zkoušíte. Byli jste varováni.

Je možné vytvořit kopii datového CD 1:1?

Ano. Ale měli byste vědět, že když se vyskytnou chyby při čtení originálu (z důvodu prachu nebo škrábanců), projeví se to špatnou kopií. Pamatujte si, že obě metody selžou na hudebních CD! Na hudebních CD musíte použít cdrdao nebo cdda2wav.

První případ: máte zapisovací jednotku CD a oddělenou jednotku CD-ROM. Provedením příkazu

```
cdrecord -v dev=0,6,0 speed=2 -isosize /dev/scd0
```

načtete datový proud z jednotky CD-ROM připojené jako `/dev/scd0` a zapíšete jej přímo na zapisovací jednotku.

Druhý případ: nemáte oddělenou jednotku CD-ROM. V tomto případě musíte použít zapisovací jednotku nejprve pro načtení CD-ROMu:

```
dd if=/dev/scd0 of=cdimage
```

Tento příkaz přečte obsah CD-ROMu v zařízení /dev/scd0 a zapíše jej do souboru „cdimage“. Obsah tohoto souboru je ekvivalentem toho, co produkuje `mkisofs`, takže můžete postupovat tak, jak bylo popsáno výše v tomto dokumentu (což znamená předat soubor cdimage jako vstup cdrecord). Pokud chcete vidět ukazatel průběhu a další umělecké věci, také můžete použít `sdd` od Jörga Schillingse.

V případě, že nastávají chyby, nainstalujte nejnovější verzi cdrecord, která obsahuje nástroj nazvaný „readcd“ (je pod misc/). Umožňuje stejný výsledek jako sdd, ale čte sektory na CD-ROM v případě chyb několikrát.

Může Linux načítat CD-ROMy Joliet? (zastaralá odpověď)

Ano. Novější jádra (2.0.36 a nadcházející 2.2) mají vestavěnou podporu pro formát Joliet. Pamatujte si, že musíte použít ve vašem /etc/fstab obě volby: klíčová slova iso9660 a joliet (druhý je opravdu rozšířením). Více informací najdete na adrese http://www-plateau.cs.berkeley.edu/people/chaffee/joliet.html.

Jak mám načítat/připojovat CD-ROMy pomocí zapisovací jednotky?

Stejně jako to děláte u běžných jednotek CD-ROM. Není v tom žádný trik. Pamatujte si, že musíte použít zařízení scd (SCSI CD-ROM) pro připojení CD-ROMů pro čtení, i když máte CD-ROM

ATAPI (vzpomeňte si, že jste konfigurovali vaše zařízení ATAPI tak, aby se chovalo jako SCSI). Příklad /etc/fstab:

```
/dev/scd0  /cdrom  iso9660  ro,user,noauto  0  0
```

Jak uložit více dat na CD-R?

Použijte `bzip2` místo jiných komprimačních programů jako `gzip` nebo `pkzip`. Ušetří vám to až 30 % místa na disku pro větší soubory (>100kb). Můžete jej stáhnout na adrese

http://www.muraroa.demon.co.uk/

Místo vytvoření skutečného hudebního CD můžete volitelně převést vaše hudební soubory WAV na hudební soubory MP3 a uložit je na souborový systém ISO-9660 jako běžné soubory. Obvykle MPEG III poskytuje kompresi 1:10. Samozřejmě, většina přehrávačů CD neumí načítat soubory... to je nevýhoda. Na druhou stranu, proč nespouštět hudbu na vaší příští párty z pevného disku? 18 Gbajtů stačí na 3000-4000 titulů. :-)

Software pro komprimaci do MPEG III je k dispozici na adrese

http://www.sulaco.org/mp3/

Přehrávač MPEG III je k dispozici na adrese

http://www.mpg123.org/

Pro záznam řeči byste mohli chtít vyzkoušet snížit jeho velikost pomocí `shorten` nebo „GSM lossy speech compression“:

ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/

http://kbs.cs.tu-berlin.de/~jutta/toast.html

Jak vytvářet spustitelné CD-ROMy?

Musíte mít spustitelnou disketu 1,44 MB. Vytvořte přesný obraz této diskety spuštěním příkazu

```
dd if=/dev/fd0 of=boot.img bs=18k
```

Umístěte obraz této diskety do adresáře, který obsahuje sadu vašich souborů (nebo do jeho podadresáře, jak budete chtít). Řekněte mkisofs o tomto souboru volbou ‘-b’ a také použijte ‘-c’. Detailnější informace najdete v souboru README.eltorito v distribuci mkisofs.

Zajímavá aplikace pro vlastní spustitelná CD je nezavirovaný systém DOS nebo Windows. Šetří to peníze za pevné disky (pokud máte síť a používáte sambu pro umísťování uživatelských dat na souborový server). V německém počítačovém časopisu c't o tom byl článek v čísle 11/99, strana 206 (http://www.heise.de/).

Detaily o spustitelném CD-ROMu RedHat jsou k dispozici na adrese http://members.bellatlantic.net/~smithrod/rhjol-technical.html.

Jak lze zapisovat na CD-ROMy jako na pevný disk?

Existuje souborový systém *overlay* dostupný pro Linux, který je připojen přes CD-ROM a brání všem operacím zápisu. Nové a upravené soubory jsou uloženy jinde, ale pro uživatele to vypadá, jako by se obsah CD-ROM měnil. Více informací najdete na adrese http://home.att.net/~artnaseef/ovlfs/ovlfs.html.

Pokud to vašim potřebám nevyhovuje: počkejte si, až bude systém souborů UDF podporován Linuxem nebo pomozte při jeho vývoji (viz. http://trylinux.com/projects/udf/. V tomto okamžiku je podporováno z důvodu omezení v ovladačích CD-ROM v jádře Linuxu pouze čtení médií CD.

Je možné použít více zapisovacích jednotek najednou?

Ano. Je hlášeno, že pracují alespoň 3 zapisovací jednotky na plné rychlosti (6x) na PC s 233 MHz, jedinou sběrnicí SCSI a jádrem 2.2.12. Potřebujete novou verzi jádra Linuxu (2.2.12 nebo vyšší).

Co Solaris, *BSD, AIX, HP-UX, atd.? Je má varianta Unixu podporována?

Pouze kapitola 2 je určena pouze pro Linux. Kapitoly 3 a 4 můžete použít i když máte operační systém jiné rodiny než Linux. Podívejte se prosím do souborů README.NetBSD, README.aix, README.hpux, README.next, README.solaris, README.sunos, README.vms nebo README.xxxBSD v distribuci cdrecord.

Pravděpodobně ano. Zkompilujte cdrecord pro vaši platformu a spusťte příkaz „cdrecord -scanbus“. Přečtěte si soubor README.* pro váš Unix distribuovaný se zdroji cdrecord. Samozřejmě, ne všechny varianty Unixu umí číst rozšíření RockRidge, Joliet nebo HFS na vašem nově vytvořeném CD-R.

Kam trvale uložit místní konfiguraci?

Máte dvě možnosti. Buď použijete vestavěný konfigurační soubor pro cdrecord, nebo použijete příkazový interpret jako v ukázce dole. Tento skript příkazového interpretu načte konfigurační soubor, který vypíše možnosti a parametry pro cdrecord řádek po řádku. Názvy jsou úplně stejné jako na příkazové řádce, ale bez počáteční čárky. Komentáře jsou povoleny. Příklad:

```
# buďte upovídaní
v
# nastavte rychlost zapisovací jednotky
speed=2
# parametry zařízení ve tvaru BUS,ID,LUN
dev=0,6,0
```

Konfigurační soubory pro obal patří do /etc/cdrecord/ a musí být uvedeny na příkazovém řádku. Příklad: pokud se chcete odkazovat na konfiguraci /etc/cdrecord/mywriter.cfg, pak můžete spustit příkaz „cdrecord.sh mywriter.cfg -audio track1...“. Všechno za mywrite.cfg je předáno cdrecord.

```
#! /bin/bash

CFGDIR="/etc/cdrecord"

CFG="$1"
shift
ARGS_LEFT="$@"

if [ ! -f "$CFGDIR/$CFG" ]
then
    echo "Konfigurační soubor $CFGDIR/$CFG nebyl nalezen. Konec."
    exit 1
fi
```

```
while read LINE
do
    case $LINE in
        \#*|"") continue;;
    esac
    old_IFS="$IFS"
    IFS="$IFS="
    set -- $LINE
    IFS="$old_IFS"
    O_NAME="$1"
    O_VALUE=""
    while shift
    do
        case $1 in
            "") continue;;
        esac
        O_VALUE="$1"
    done

    if [ -z "$O_VALUE" ]
    then
        O_CDRECORD="$O_CDRECORD -$O_NAME "
        continue
    fi
    O_CDRECORD="$O_CDRECORD $O_NAME=$O_VALUE "

done < "$CFGDIR/$CFG"

set -x    #DEBUG
exec cdrecord $O_CDRECORD $ARGS_LEFT
echo "Spuštění programu selhalo."
```

Jak získat informace o CD?

Někde před prvními 32 k CD je umístěn blok informací o CD. Tyto informace můžete načíst pomocí následujícího skriptu shellu:

```
#! /bin/bash

RD=/dev/cdrom
for i in 32768,7   32776,32 32808,32 32958,128 33086,128 33214,128 \
         33342,128 33470,32 33581,16 33598,16  33615,16  33632,16
do
    old_IFS="$IFS"
    IFS=","
    set -- $i
    IFS="$old_IFS"
    OFFSET=$1
    LENGTH=$2
    echo "*'dd if=$RD bs=1 skip=$OFFSET count=$LENGTH   2> /dev/null'#"
done
```

Co přepisování?

Když přepisujete média CD-RW, uveďte parametr blank=fast u cdrecord. To je vše. Detailní informace o tomto parametru najdete v manuálových stránkách pro cdrecord.

Jak vytvářet multi-session CD?

Nejprve ze všeho musí být obraz pro multi-session CD formátován pomocí souborového systému ISO-9660 s rozšířením RockRidge. A musíte použít volbu -multi u cdrecord pokud chcete přidat další data. Takže alespoň u prvního zápisu musíte zadat volbu -multi.

Některé zapisovací jednotky nemají podporu pro CD-ROM XA mode 2 nebo session-at-once (SAO), takže je potřeba zadat přepínač -data u cdrecord na příkazovém řádku.

Generování obrazů pro druhý a následující záznamy je trošku komplikovanější. Mkisofs musí vědět, kde začíná volné místo na CD-R. Tuto informaci je možno získat pomocí volby -msinfo u cdrecord (viz. níže uvedený příklad).

```
shell> NEXT_TRACK='cdrecord -msinfo dev=0,6,0'
shell> echo $NEXT_TRACK
shell> mkisofs -R -o cd_image2 -C $NEXT_TRACK -M /dev/scd5
         private_collection/
```

Více informací najdete v souboru README.multi, který je distribuován s cdrecord.

Mám použít adaptér SCSI dodaný se zapisovací jednotkou?

Hlášeno e-mailem: Většina dokumentů k zapisovacím jednotkám doporučuje použít oddělenou sběrnici SCSI, pokud zapisujete z jednotky CD-ROM na zapisovací jednotku a já sám jsem to viděl v následující variantě:

Karta Adaptec 2940UW SCSI, 24x SCSI CD-ROM a zapisovací jednotka 4x4 SCSI. Když jsem dostal zapisovací jednotku, byla s ní dodána karta ISA SCSI, která měla pracovat pouze s jedním zařízením. Myslel jsem, že ji zahodím a použiji mou lepší kartu Adaptec pro všechna zařízení. Všiml jsem si, že to bylo náchylnější na podtečení vyrovnávací paměti při zápisu rychlostí 4x, ale jakmile jsem zkusil zapojit kartu ISA SCSI, neměl jsem žádné problémy. Znám další dva lidi (oba používají karty adaptec 2940), kteří se setkali s přesně stejnými symptomy, obvykle při zápisu z jednotky CD-ROM na zapisovací jednotku. I když jsem se nikdy nesetkal s problémem při vypalování z pevného disku na zapisovací jednotku na stejné sběrnici.

Jak pálit přes síť?

Obvykle je přenos souborů přes FTP dostatečně rychlý, i když jen přes 10Mb ethernet, pro krmení zapisovací jednotky se čtyřnásobnou (4x) rychlostí. Klienta FTP a cdrecord můžete propojit skrze fifo. Nejprve vytvořte fifo s názvem cdimage:

```
mkfifo cdimage
ftp other.host.org
get cdimg cdimage
```

Pak zpracujte cdimage jako obyčejný soubor, tj. spusťte následující příkaz:

```
cdrecord dev=0,1,0 speed=2 cdimage
```

Váš klient FTP si všimne, že cdrecord chce číst ze souboru a začne přenášet data z hostitele FTP.

Slyším lupnutí nebo cvaknutí na konci každé stopy.

Abyste se lupnutí zbavili, musíte použít režim DAO (disk-at-once).

Jak se dá nastavit, aby mohl uživatel vypalovat CD bez toho, aby musel být root?

Můžete přidat bit setuid u spustitelného souboru cdrecord. Samozřejmě zde může být bezpečnostní riziko. Jednoduché nastavení oprávnění pro soubory zařízení nepomůže, protože cdrecord spouští privilegované příkazy skrze obecná rozhraní SCSI.

```
which cdrecord
chown root.root /usr/bin/cdrecord
chmod 4111 /usr/bin/cdrecord
```

Kde získám standardy „Yellow Book“ a „Orange Book“?

Tištěné specifikace můžete získat od společnosti Philips a jsou drahé.

Hledal jsem informace o vypalování Video-CD pod Linuxem.

Zde můžete nalézt nástroje pro vytváření videí MPEG a Video-CD: http://www.mainconcept.de/

http://www.johanni.de/munich-vision/vcd/

Korektní lidé se zmiňují o nástrojích Berkeley a jiných strategiích YUV. Jejich použití je komplikované, zabírá mnoho času a místa na pevném disku a neumožňují hudební stopy. Doporučuji použít aplikace koncepčně vyšší úrovně než výše uvedené.

Co je jednodušší nastavit, IDE nebo SCSI?

Zapisovací jednotky SCSI se trochu jednodušeji nastavují pro zápis na CD pod Linuxem. A je hlášeno, že mají lepší opravy chyb. Jestli to vyvažuje vyšší cenu, nelze obecně říci.

Jak mohu přepálit CD pomocí {cdrecord,cdrdao}?

Přepalování CD-R není nic speciálního. Je na vaše vlastní riziko, že budou data umístěna na CD-ROM, ale to je vše. V softwaru pro Linux neexistují žádná omezení 650 Mbajtů.

Co cdrecord udělá, když se zastaví vstup z roury?

Dokončí zápis. Takže můžete prostě spojit váš oblíbený nástroj pro zálohování s cdrecord pomocí roury, jako v „bru -size=640m -f - | cdrecord dev=0,1,0 speed=2 -“. Musíte věnovat speciální péči zálohovacímu nástroji, pokud záloha zabírá více CD-R.

Existuje ekvivalent ignore=hdX pro emulaci ide-scsi?

Neznám žádný způsob, ale někdo by mohl tuto funkci přidat do zdrojů jádra Linuxu.

Kolikrát mohu znovu použít CD-RW než se poškodí?

Dobrá otázka.

Který formát bych měl zvolit pro na platformě nezávislý CD-ROM?

CD-ROM by měl být čitelný pro všechny systémy pouze při použití holého formátu ISO 9660. To znamená stupidní názvy souborů 8+3 ze starého MS-DOSu a bez rozšíření HFS (Macintosh), Joliet (Microsoft) nebo RockRidge (novější Unixy). Neexistuje rozšíření pro delší názvy souborů, které by byly čitelné všemi operačními systémy.

Je možné multi-session pro hudební stopy?

Přehrávače hudebních CD umí pracovat pouze s hudebními stopami uloženými v první session. Jinými slovy, nemůžete přidávat hudební stopy pomocí dalších sessions. Samozřejmě, zápis datových stop do druhé session je efektivně skryje před přehrávači hudebních CD. Tímto způsobem zabráníte tomu, abyste měli tichou stopu na vašem CD ve smíšeném režimu (smíšení hudby a dat).

Jaké hardwarové prostředky potřebuji? Stačí staré Pentium?

Otázka závisí na vašich požadavcích. Pokud potřebujete důvod pro zakoupení nového počítače, je zde odpověď od mezinárodní asociace výrobců počítačů: Pro cokoliv, co chcete dělat, potřebujete procesor s 800 MHz. Protože nepůjde do stávající základní desky, potřebujete také novou základní desku. Nejjednodušším řešením je zakoupení typické kompletní nabídky, kterou vidíte v reklamách v televizi. Ignorujte prosím zbytek této části.

Nyní případ, kdy chcete rozumnou odpověď: Vytvořil jsem úspěšně více CD-ROMů na „486“ s 66 Mhz. Ačkoliv to MS už nepovažuje za PC a doporučuje na něm spouštět jejich verzi CE (viděno na CeBITu), Linux běží stejně dobře na předchůdcích procesoru Pentium a dokonce na nich umí vytvářet CD. Jednoduše můžete zjistit zda výkon vašeho hardwaru stačí pro vytváření CD-ROMů tím, že to vyzkoušíte. Prostě přidejte na příkazovém řádku přepínač -dummy pro spouštění cdrecord a laser zůstane vypnutý. Sledujte proces vypalování.

Odstraňování problémů

Pamatujte, že vadná CD můžete stále používat jako podložky pod nápoje :-)

Nepracuje pod Linuxem

Nejprve prosím zkontrolujte, že zapisovací jednotka pracuje v softwaru, se kterým byla dodána (=pod jiným operačním systémem). Konkrétně:

- Najde řadič zapisovací jednotku jako zařízení SCSI?
- Najde software ovladače zapisovací jednotku?
- Je možné vytvořit CD pomocí dodaného softwaru?

Pokud nepracuje ani s dodaným softwarem, je zde konflikt hardwaru nebo vadný hardware. Pokud pracuje a používáte loadlin pro spouštění systému Linux, pak je problém v loadlin. Loadlin dělá „teplý start“ s většinou hardwaru již inicializovanou a to může zmást jádro Linuxu.

Chybová zpráva: No read access for ‘dev=0,6,0’

Pod Linuxem jsou některé verze knihovny C nekompatibilní (chybné), takže aplikace sestavená pro jednu verzi nebude pracovat s jinou. Zde je příklad pro chybu vyvolanou předkompilovanými binárními soubory:

```
[root@Blue /dev]# cdrecord -eject dev=0,6,0
cdrecord: No such file or directory. No read access for 'dev=0,6,0'.
```

Řešením je instalace novější knihovny C.

Nepracuje pod DOSem a spol.

Zkuste použít Linux. Instalace a konfigurace ovladačů SCSI pro DOS je peklo. Linux že je příliš komplikovaný? Ha!

Chyby SCSI v průběhu vypalování

Tyto chyby jsou nejpravděpodobněji způsobeny

- chybějící funkcí odpojování/znovupřipojování na sběrnici SCSI
- nedostatečně chlazeným hardwarem
- vadným hardwarem (mělo by být detekováno v 5.1.)

Za různých okolností se zařízení SCSI odpojují a znovu připojují (elektronicky) ze sběrnice SCSI. Pokud tato funkce není k dispozici (zkontrolujte řadič a parametry jádra), mohou mít některé zapisovací jednotky problémy v průběhu vypalování nebo zakončování CD-R.

Zejména ovladač SCSI NCR 53c7,8xx má tuto funkci implicitně vypnutou, takže byste ji měli nejdříve zkontrolovat:

```
NCR53c7,8xx SCSI support                  [N/y/m/?] y
   always negotiate synchronous transfers [N/y/?] (NEW) n
   allow FAST-SCSI [10MHz]                [N/y/?] (NEW) y
   allow DISCONNECT                       [N/y/?] (NEW) y
```

Chyby médií

Pokud cdrecord hlásí chyby médií ve tvaru „Sense Key: ... Medium Error, Segment ...“, pak médium není prázdné. Pokud používáte CD-RW, zkuste přepnout blank=fast na spolehlivější blank=all. Pokud používáte CD-R, pak se ujistěte, že CD-R nikdy dříve nevidělo zapisovací jednotku, nebo vyzkoušejte CD od jiného výrobce.

Nově vytvořená CD nejsou v některých přehrávačích čitelná

Někteří lidé hlásili problémy s přehráváním jimi vytvořených CD. Velmi staré audio přehrávače nebo autopřehrávače mohou mít problémy s CD-R, ačkoliv je to extrémně vzácné. Celkem časté jsou problémy s CD-RW, protože neodráží laserový paprsek tak dobře jako CD-R a lisované „stříbrné“ disky.

Můj skener přestal pracovat poté, co jsem zavedl modul ide-scsi

Vložením emulace hostitelského adaptéru SCSI se mění názvy zařízení SCSI. Pokud byl váš skener předtím /dev/sg0, mohl by být nyní /dev/sg1 nebo /dev/sg2. Výborní vývojáři jádra v minulosti nemysleli, že je to úplně na mrtvici a zakázali řešení jako devfs. Ale to je jiný příběh. Nejprve byste měli vyzkoušet nastavení odkazu /dev/scanner tak, aby ukazoval na platné generické zařízení SCSI. Příklady:

```
cd /dev
ls -l scanner     # zobrazuje aktuální nastavení
ln -sf sg2 scanner
# otestuj scanner
ln -sf sg1 scanner
# otestuj the scanner
# atd.
```

Vývojáři aplikací by měli hodně myslet na podporu tohoto nebezpečného a k chybám náchylného schématu přidělování názvů. Zvažte prosím alespoň použití přechodných řešení jako jsou parametry SCSI používané nástrojem cdrecord.

Zásluhy

Mnoho díků patří čtenářům tohoto dokumentu, kteří přispívali aktivně do jeho obsahu. Jelikož nemám přístup k zapisovací jednotce mnoho let, byly pro mne zprávy o nastaveních ze skutečné praxe a zkušenosti opravdu cenné.

Doug Alcorn <doug@lathi.net>

pomáhal vylepšit manipulaci s novějšími jádry

Kalle Andersson <kalle@sslug.dk>

Jak zapisovat hudební CD přímo z MP3.

Alan Brown <alan@manawatu.net.nz>

Rick Cochran <rick@msc.cornell.edu>

podněty k odpojení/znovupřipojení v ovladači ncr implicitně vypnutém

Robert Doolittle <bob.doolittle@sun.com>

dobré argumenty pro vypuštění cdwrite z tohoto dokumentu

Markus Dickebohm <m.dickebohm@uni-koeln.de>

Thomas Duffy <tduffy@sgi.com>

Hlavní vyčištění syntaxe a pravopisu

Dave Forrest <dforrest@virginia.edu>

Oprava problémů s pojmy adaptéru

Jos van Geffen <jos@tnj.phys.tue.nl>

oznámil problém v kapitole 4.

Bernhard Gubanka <beg@ipp-garching.mpg.de>

Upozornil na potřebu nové verze mount pro práci se zařízením loopback

Stephen Harris <sweh@mpn.com>

Přispěl poznámkou k vytváření hudebních CD

Janne Himanka <shem@oyt.oulu.fi>

Odkaz na záplatu jádra pro čtení CD-ROMů Joliet

Stephan Noy <stnoy@mi.uni-koeln.de>

Informace a zkušenosti s vytvářením hudebních CD

Don H. Olive <don@andromeda.campbellsvil.edu>

URL nástroje mkhybrid

Jesper Pedersen <jews@imada.ou.dk>

Pierre Pfister <pp@uplift.fr>

Pomáhali při vývoji návodu na kopie 1:1.

Daniel A. Quist <dquist@cs.nmt.edu>

Informace o IDE CD-R a novějších verzích jádra

Martti.Rahkila@hut.fi

Oznámil problém s přednastavenými zapisovacími jednotkami při spouštění pomocí loadlin.

Dale Scheetz <dwarf@polaris.net>

Joerg Schilling <schilling@fokus.gmd.de>

Mnoho informací o cdrecord

Martin Schulze <joey@Infodrom.North.DE>

Podal informace o diskusní skupině cdwrite

Gerald C Snyder <gcsnyd@loop.com>

Testoval zápis CD se systémem souborů ext2 (viz. 4.4)

Art Stone <stone@math.ubc.ca>

Měl nápad k umisťování systémů souborů jiných než ISO-9660 na CD

The Sheepy One <kero@escape.com>

Doporučil používat vadné CD-ROMy jako tácky pod nápoje

Erwin Zoer <ezoer@wxs.nl>

Kromě toho bych rád poděkoval následujícím lidem za nahlášení chyb v pravopisu: Bartosz Maruszewski <B.Maruszewski@zsmeie.torun.pl>, Alessandro Rubini <rubini@prosa.it>, Ian Stirling <ian@opus131.com>, Brian H. Toby.

Konec Linux CD-Writing HOWTO. (Už můžete přestat číst)

Vypalování CD s MP3

Originál: http://tldp.org/HOWTO/mini/MP3-CD-Burning/

Úplný návod na vytváření zvukových a datových CD z MP3 souborů.

Úvod

Tento dokument vznikl v důsledku mých zkušeností s vypalováním zvukových CD a kvůli nedostatku informací o normalizaci zvuku na Internetu. Typicky vypaluji zvuková CD jako mixy – různé skladby z různých zdrojů. Velmi často se hlasitost nahrávek výrazně liší. To je první problém. Další problém je v tom, že řada zvukových souborů na Internetu není kompatibilní s CD formátem (16bitové stereo, 44,1 kHz) a je nutné je konvertovat. Existuje řada programů na vypalování zvukových CD z MP3 souborů a spousta z nich provádí tyto konverze automaticky. Nenarazil jsem však na jediný nástroj, který by normoval hlasitost nahrávek, a proto nabízím následující přepis.

Tento dokument se týká jedné jediné věci – vypalování MP3 souborů na zvuková CD, aby je bylo možné poslouchat. Podrobnější informace o MP3 souborech naleznete v dokumentu MP3 HOWTO, obecné informace o vypalování CD pak v dokumentu CD-Writing HOWTO.

Předpokládám, že si chcete vypálit CD se skladbami z různých zdrojů, které se liší kvalitou, a přitom požadujete co nejlepší výsledek. Tento dokument vám může pomoci.

Zvuková CD

Příprava stop

Poznámka: U všech příkazů předpokládáme příkazový interpret bash.

1. Shromážděte všechny MP3 soubory v jednom adresáři.
2. Pokud některé ze souborů pocházejí ze systémů DOS/Windows, mohou být v jejich názvech použita velká písmena. Můžete provést konverzi na malá písmena buď u celých názvů, nebo jenom u přípon. Konverzi celých názvů provedete příkazem:

   ```
   for i in *.[Mm][Pp]3; do mv "$i" `echo $i | tr '[A-Z]' '[a-z]'`; done
   ```

   konverzi pouze přípon provedete příkazem:

```
for i in *.MP3; do mv "$i" "`basename "$i" .MP3`.mp3; done
```

3. Pokud názvy některých souborů obsahují mezery, převeďte je na podtržítka:

```
for i in *.mp3; do mv "$i" `echo $i | tr ' ' '_'`; done
```

4. Následujícím příkazem proveďte konverzi na formát WAV:

```
for i in *.mp3; do mpg123 -w `basename $i .mp3`.wav $i; done
```

Pokud dekódujete 22kHz MP3 soubory, může být výsledek zkreslený. Napravíte to příkazem:

```
for i in *.mp3; do mpg123 --rate 44100 --stereo --buffer 3072 --resync -
w `basename $i .mp3`.wav $i; done
```

Program *mpg123* by měl být součástí každé linuxové distribuce, pokud jej ale náhodou nemáte, můžete jej získat na adrese http://www.mpg123.de/.

Poznámka: U některých MP3 souborů jsem zjistil, že výstup programu *mpg123* je zkreslený. Nejprve jsem si myslel, že jde o chybu v MP3 souboru, jenže jiné přehrávače jej interpretovaly správně. Proto jsem hledal přehrávač MP3 souborů, který umí vytvářet formát WAV.

Našel jsme program MAD na adrese http://www.mars.org/home/rob/proj/mpeg/. S tímto programem vypadá příkaz pro konverzi takto:

```
for i in *.mp3; do madplay -o `basename $i .mp3`.wav $i; done
```

Konverzi je možné provést ještě jinak. Některé MP3 soubory při dekódování zjevně zmatou jak *mpg123*, tak i *madplay*. Tyto obtížné případy dobře zvládá program *lame*, který najdete na adrese http://www.mp3dev.org/mp3/:

```
for i in *.mp3; do lame --decode $i `basename $i .mp3`.wav; done
```

Poznámka: Sekvence *`basename $i .mp3`.wav* nahrazuje příponu *mp3* příponou *wav*. Existuje 101 různých způsobů, jak to udělat, například také *`echo „$i“ | sed 's/\.mp3$/.wav/'`*

1. Spusťte příkaz *file *.wav* a zkontrolujte, zda nějaký soubor není jiného typu, než 16bitový, stereo, 44 100 Hz.
2. Pokud by se takový soubor objevil, proveďte jeho konverzi na požadovaný formát. Například konverzi souboru *track01.wav* na vzorkovací frekvenci 44,1 kHz provedete příkazem:

```
sox track01.wav -r 44100 track01-new.wav resample
```

Pokud by tím při konverzi mono souborů vznikalo praskání, použijte příkaz:

```
sox track01.wav -r 44100 -c 2 track01-new.wav
```

sox je velmi populární program a pravděpodobně bude součástí vaší distribuce Linuxu. Pokud ne, najdete jej na adrese http://www.spies.com/Sox. Jeho řádkové parametry jsou ale pro příležitostného uživatele (jako já) dost záhadné, proto doporučuji http://www.spies.com/Sox/sox.tips.html, kde najdete příklady použití.

3. Dalším krokem je normalizace *wav* souborů, abychom odstranili výrazné rozdíly v hlasitosti. Používám program *normalize* Chrise Vailla (cvaill@cs.columbia.edu), který najdete na

adrese http://www.cs.columbia.edu/~cvaill/normalize. Používám následující syntaxi (*-m* zapíná mixážní režim, kdy budou všechny soubory tak nejtišší, jak je to možné):

```
normalize -m *.wav
```

Vypalování CD

Existuje řada programů, které z *wav* souborů vytvoří zvuková CD. K vypalování v řádkovém režimu používám program *cdrecord*, pro grafické rozhraní program *xcdroast*. Při použití programu *cdrecord* potřebujete vědět, které SCSI zařízení je vaše vypalovačka. Pokud používáte ATAPI vypalovačku, musíte použít emulaci SCSI (modul jádra *ide-scsi*). Předpokládejme, že máte ATAPI vypalovačku jako master na sekundárním IDE rozhraní. Bude jí tedy odpovídat soubor */dev/hdc*. Abyste jádru řekli, že ji má považovat za SCSI zařízení, musíte do souboru */etc/lilo.conf* doplnit řádek

```
append=" hdc=ide-scsi"
```

Pokud jádro nenahraje modul *ide-scsi* automaticky, musíte do souboru *rc.local* (nebo odpovídajícího) doplnit příkaz *insmod ide-scsi*. Jakmile se vypalovačka hlásí jako SCSI zařízení, použijte příkaz *cdrecord --scanbus* ke zjištění, o jaké zařízení jde. Na mém systému vypadá výstup takto:

```
scsibus1:
1,0,0 100) 'IOMEGA ' 'ZIP 250 ' '51.G' Removable Disk
1,1,0 101) 'HP ' 'CD-Writer+ 7100 ' '3.01' Removable CD-ROM
```

Při spouštění programu *cdrecord* budete tedy vypalovačku definovat parametrem *dev=1,1,0*. Celý příkaz pro vypálení disku pak vypadá takto:

```
cdrecord dev=1,1,0 -eject speed=2 -pad -audio *.wav
```

Poznámka: Parametr *-pad* je nutný, protože zvukové stopy na CD musí být zarovnány na určitou délku, což u MP3 souborů nemusí být splněno.

Vypalování DAO CD

DAO, Disc-At-Once, je nová metoda vypalování CD bez dvousekundové pauzy mezi stopami. Je vhodná k vypalování různých party-mixů :-). Tento režim vypalování umožňuje program *cdrdao*, který můžete najít na SourceForge, http://sourceforge.net/projects/cdrdao/.

Program *cdrdao* používá popisný soubor nazvaný *TOC* (samozřejmě Table Of Contents, tedy *obsah*). Tento soubor lze vytvořit dvěma způsoby. Jedna možnost je použít skript dodávaný se zdrojovou verzí programu *cdrdao* (nachází se v adresáři *contrib* a jmenuje se *generate_toc.sh*). Jako parametr mu předáte seznam *wav* souborů, výstupem je soubor *cd.toc*. Druhá možnost je vytvořit si tento soubor ručně v oblíbeném textovém editoru. Takto vypadá příklad:

```
CD_DA

TRACK AUDIO
AUDIOFILE "mix-01.wav" 0

TRACK AUDIO
AUDIOFILE "mix-02.wav" 0

TRACK AUDIO
AUDIOFILE "mix-03.wav" 0
```

```
TRACK AUDIO
AUDIOFILE "mix-04.wav" 0

TRACK AUDIO
AUDIOFILE "mix-05.wav" 0
```

Hodnota *0* (nula) za názvem souboru znamená, že se má začít od začátku souboru. Kromě toho je možné uvést i další hodnotu, udávající, jaká délka souboru má být použita. Program *xcdroast* vytváří podobné *TOC* soubory, příklad najdete v adresáři *testtocs* zdrojové distribuce *cdrdao*.

Program *cdrdao* standardně používá k vypalování zařízení */dev/cdrecorder*, což musí být odkaz na vaši vypalovačku. Předpokládejme, že vypalovačka se hlásí jako */dev/scd0*, pak příslušný odkaz vytvoříte (jako superuživatel) příkazem:

```
ln -s /dev/scd0 /dev/cdrecorder
```

Pokud je soubor s obsahem disku pojmenován *cd.toc*, samotný příkaz pro vypálení je velmi jednoduchý:

```
cdrdao wrote cd.toc
```

Software

Existují programy, které automatizují proces vytváření CD z MP3 souborů. Krátký výběr vypadá takto:

- *burnmp3* – program automatizující vypalování metodou DAO. http://richardsnow.bizland.com/burnmp3/.
- *mp32dao* – skript z distribuce *cdrdao*, v adresáři *contrib*. http://cdrdao.sourceforge.net/.

Datová CD

Poznámka: Na této části se pracuje, díváte se na první, velmi stručnou verzi.

S rostoucí oblibou přehrávačů CD/MP3 roste význam vypalování datových CD k poslechu hudby. Hlavní výhodou je bez debaty možnost vměstnat na jeden disk až desetinásobné množství hudby (jde o velmi přibližný odhad). Co se týče datových CD s MP3 soubory, jde o naprosto běžná datová CD ve formátu ISO9660, jednotlivé MP3 jsou nahrány jako obyčejné soubory. Všechny mi známé přehrávače CD/MP3 zvládnou zpracovat CD s adresáři. Při vypalování CD používám rozšíření Joliet. Při vypalování takovéhoto CD v Linuxu musíte nejprve vytvořit ISO obraz disku a pak je vypálit tak, jak ukazuje následující příklad:

```
mkisofs -J -o /tmp/mymp3s.iso /home/greg/mp3s/
cdrecord dev=1,0,0 speed=16 /tmp/mymp3s.iso
```

A to je všechno.

Ještě musím vyzkoumat možnost přímé normalizace hlasitosti MP3 souborů – obávám se ale, že to vždy vyžaduje dekompresi, normalizaci a novou kompresi, což vede ke ztrátě kvality... Zůstaňte s námi!

Linux a DVD

Originál: http://tldp.org/HOWTO/DVD-HOWTO.html

(Doufejme) snadno pochopitelné vysvětlení, jak pod Linuxem přehrávat DVD.

Úvod

Účelem tohoto textu je poskytnout jednoduchý návod, jak pod Linuxem přehrávat DVD filmy. Snažil jsem se vyjadřovat co nejjasněji, pokud byste ale něčemu nerozuměli nebo jste objevili chybu, informujte mě a já to napravím. Pokud máte nějaké dotazy, můžete se přihlásit do konference zasláním prázdného e-mailu na adresu livid-user-subscribe@linuxvideo.org. Další informace o problematice můžete najít také na adrese http://www.linuxvideo.org.

Požadavky

V dokumentu předpokládáme, že máte:

- Jádro 2.2.x nebo vyšší. Doporučujeme jádro s podporou *ioctl* volání DVD služeb.
- XFree86 3.3.x nebo 4.x.
- DVD-ROM/RAM/RW mechaniku podporovanou Linuxem (což je většina).
- Základní znalosti GLOBAL: shell – příkazový interpret a angličtiny.

Soubory

Abychom začali od začátku, potřebujete jádro s MTRR (viz dále) a s podporou DVD volání. Zatímco MTRR je v jádře od verze 2.2.11 (opravte mne, pokud se mýlím), kvůli podpoře DVD volání budete možná muset aktualizovat jádro. Příslušné opravy najdete na http://www.kernel.dk, samotné jádro pak najdete na svém oblíbeném zrcadle serveru http://www.kernel.org.

Jako další věc budete potřebovat nástroje LiViD. Můžete je získat dvěma způsoby:

- Nahrát si je ze serveru CVS, takže získáte aktuální verzi. Tento postup doporučujeme.

Následující příkazy nahrají aktuální verzi LiViD. (Pokud nechcete nastavovat proměnnou CVSROOT, můžete použít parametr -d.)

```
# mkdir ~/livid
# cd ~/livid
# export CVSROOT=:pserver:anonymous@cvs.linuxvideo.org:/cvs/livid
# cvs login
(Logging in to anonymous@cvs.linuxvideo.org)
CVS password:
```

```
Anonymn□ pł□stup je bez hesla, pouze stiskn□te Enter.

# cvs -z3 co -P ac3dec mpeg2dec oms

Pł□slu□nØ soubory se nahraj□ do adresÆłø, kam patł□.
```

- Pokud CVS nechcete použít, nebo nemůžete kvůli omezením firewallem, můžete si aktuální verzi nahrát na adrese http://www.linuxvideo.org/developer/dl.phtml.

Instalace

Jádro

Jádra od verze 2.2.16 a všechna jádra 2.4.x obsahují podporu volání DVD služeb, takže potřebujete pouze zjistit, zda máte zapnutu podporu MTRR, a jádro běžným způsobem přeložit a nainstalovat.

Pokud budete chtít upravovat starší jádro, začněte na adrese, kde si nahrajete příslušné opravy.

Nevíte-li, jak jádro upravit a přeložit, přečtěte si dokument Kernel HOWTO na adrese http://howto.tucows.com/LDP/HOWTO/Kernel-HOWTO.html.

Nástroje LiViD

Při instalaci nástrojů LiViD zkontrolujte, že máte v souboru */etc/ld.so.conf* uveden adresář */usr/local/lib*.

K přehrávání DVD potřebujete sestavit pouze modul OMS, který si automaticky sestaví kodeky ac3 a mpeg2. Pokud ale chcete, můžete experimentovat i se samostatnými kodeky. Následujícími příkazy sestavíte a nainstalujete nástroje LiViD. Pokud by došlo k nějakým potížím, podívejte se do části *Problémy*.

```
ac3dec: (nepovinné)

  # cd ~/livid/ac3dec
  # ./autogen.sh
  # make
  # make install

mpeg2dec: (nepovinné)

  # cd ~/livid/mpeg2dec
  # ./autogen.sh
  # make
  # make install

oms:

  # cd ~/livid/oms
  # ./autogen.sh
  # ./configure
  # make
  # make install
```

Můžete použít i různé další volby, které se předávají jako parametry skriptu *./configure*. OMS používá parametr *--enable-devel*, kterým se zapínají některé vývojové a experimentální funkce kódu. Kromě toho je možné použít standardní parametry automatické konfigurace, například *--prefix*, pokud chcete instalovat jinam, než do adresáře */usr/local*. K instalaci do podadresáře zdrojového adresáře můžete například zadat příkaz:

```
# ./configure --prefix=`pwd`/inst
```

Úplný seznam parametrů získáte příkazem *./configure --help*. Pokud máte systém X Window nainstalován na nějakém neobvyklém místě (třeba v případě, že máte současně nainstalovány XFree-86 3.3.x i 4.x), budete možná potřebovat parametry *--x-includes* a *--x-libraries*.

Tímto by měly být nainstalovány nezbytné součásti LiViD. Následující postup není nutný, pokud ale chcete použít „rourové“ pluginy, budete muset vytvořit některé speciální FIFO roury, aby měla data z DVD kudy cestovat. Můžete je vytvořit následujícími příkazy (pokud už tedy neexistují):

```
# mkfifo /tmp/video
# mkfifo /tmp/audio
```

Další možnost řízení datového toku, která je dostupná ve verzích oms_devel, jsou přímé vstupně -výstupní operace, podporované nejnovějšími linuxovými jádry. Pokud takové jádro nemáte, budou se data číst ze standardních zařízení. Použití přímých vstupně-výstupních operací je sice výhodné, není to ale nutné. Pokud neexistují, vytvořte následující dvě zařízení:

```
# mknod /dev/rawctl c 162 0
# mknod /dev/raw1 c 162 1
```

A konečně poslední věc. OMS potřebuje, aby */dev/dvd* byl symbolický odkaz na vaši DVD mechaniku, tedy na */dev/hdb1*, */dev/scd0* a podobně. Pokud máte DVD mechaniku přístupnou jako */dev/cdrom*, vytvoříte odkaz příkazem:

```
# ln -s /dev/cdrom /dev/dvd
```

Obecně není příliš rozumné vytváření odkazů na odkazy, protože se tím zvyšuje počet potřebných vstupně-výstupních operací. Namísto */dev/cdrom* tedy zadejte skutečný název vašeho DVD zařízení.

Přehrávání

Nejprve budete potřebovat konfigurační soubor. Zkopírujte soubor *doc/config.sample* ze zdrojového adresáře OMS do *.oms/config* ve svém domovském adresáři. Upravte hodnoty tak, aby odpovídaly vašim ovladačům, nastavením, zařízením a podobně.

Nenechte se překvapit spoustou textových hlášení. Jedná se stále o vývojovou verzi. Pokud uvidíte spousty textu, ale neobjeví se okno s videem, asi budete mít chybu v konfiguračním souboru. Zkontrolujte, zda používáte správně doplňky a zařízení.

Abyste mohli přehrávat DVD, musíte mít systém X Window spuštěn v 16bitovém barevném rozlišení. Vložte DVD disk do mechaniky a zadejte:

```
# oms
```

Starší verze OMS používají roury, které jsme si vytvořili výše. Většina uživatelů může tuto metodu ignorovat. Potřebujete-li použít roury, zadejte:

```
# ac3dec /tmp/audio& mpeg2dec /tmp/video& oms
```

V tomto případě bude možná nutné procesy *ac3dec* a *mpeg2dec* zabít ručně.

Pokud šlo všechno dobře, měl by se objevit nějaký text a následně dotaz, zda chcete název disku vyhledat v databázi DVDDB. Klidně můžete říct, že ano. Pak by na vás měl vyskočit ovládací panel OMS. Klepněte na tlačítko *playlist*, pak na *scan dvd* a nakonec na *play*.

Další záležitosti

Nastavení MTRR

Poznámka pro majitele videokaret Matrox:

Pokud máte v jádře přeloženu podporu matroxfb, je dobrá šance, že MTRR už je také nastaveno. V tom případě můžete následující část přeskočit.

Nastavením MTRR můžete v některých případech o něco zvýšit výkon přehrávání videa, takže se každopádně jedná o rozumnou volbu. Nejprve musíte zjistit, zda máte podporu MTRR zapnutu v jádře:

```
# ls /proc/mtrr
```

Pokud se dozvíte něco jako že */proc/mtrr* neexistuje, musíte si přeložit nové jádro s podporou MTRR (v konfiguraci jádra se tato volba nachází pod „Processor type and features“).

Jakmile podpora MTRR funguje, potřebujete vědět bázovou adresu vaší videokarty v paměti, a velikost paměti RAM na videokartě. Nejjednodušší způsob, jak to zjistit, je sledovat hlášení systému X Window při jeho spouštění. Protože tato hlášení obvykle velmi rychle odběhnou mimo obrazovku a ztratí se při přepnutí na jiný terminál, musíte si výstup systému X Window přesměrovat do souboru (řekněme *xoutput*), a posléze si jej v klidu prohlédnout. Můžete to provést příkazem:

```
# startx 2>xoutput
```

Řádek s potřebnými informacemi bude typicky někde uprostřed výstupu a měl by vypadat nějak takto:

```
0xe2000000 (--) SVGA: PCI: NVidia Riva TNT2 rev 17, Memory @ 0xee000000,
```

Když tento údaj najdete, opište si poslední uvedenou adresu, v tomto případě 0xee2000000. V závislosti na vaší videokartě můžete a nemusíte vidět více adres, takže se nelekněte, pokud výpis vypadá trochu jinak. Po zapsání tohoto údaje můžete soubor *xoutput* smazat.

Dále musíte vytvořit nový MTRR. Abyste to mohli udělat, musíte znát velikost videopaměti na vaší kartě, a to v šestnáctkových hodnotách. Typické hodnoty jsou:

```
4MB  -- 0x400000
8MB  -- 0x800000
16MB -- 0x1000000
32MB -- 0x2000000
64MB -- 0x4000000
```

MTRR vytvoříte příkazem

```
# echo "base=0xe2000000 size=0x2000000 type=write-combining" > /proc/mtrr
```

kde „0xe2000000“ a „0x2000000“ nahradíte bázovou adresou a velikostí paměti vašeho systému.

Teď by mělo MTRR fungovat, což ověříte příkazem:

```
# cat /proc/mtrr
```

Měli byste dostat výstup přibližně jako:

```
reg00: base=0x00000000 (   0MB), size= 128MB: write-back, count=1
reg01: base=0xe2000000 (3616MB), size=  32MB: write-combining,
count=1
```

Počet údajů bude pravděpodobně odlišný, tak se nenechte zmást.

Problémy

Chyby za běhu

Uvádíme krátký seznam známých chyb za běhu.

Illegal Instruction Error

Pokud používáte jiný chipset než Intel (typicky K6) a dochází k chybám při spuštění mpeg2video, zkuste upravit soubor *nist/configure.in* na řádcích 129 a 130:

```
CFLAGS="$CFLAGS -DHAVE_MMX -DLINUX -march=i686 -fschedule-insns2 -malign-doub
CXXFLAGS="$CXXFLAGS -DHAVE_MMX -DLINUX -march=i686 -fschedule-insns2 -malign-
```

nahraďte „-march=i686“ na obou řádcích textem „-march=i586“, a znovu přeložte a nainstalujte.

Chyby překladu

Následuje seznam typických chyb při překladu, u nichž je známa náprava.

‘dvd_struct’ undeclared...

Nejčastější problémy při překladu vznikají tím, že OMS hledá hlavičky jádra na špatném místě. Standardně je hledá v adresáři */usr/include/[linux|asm]*, nicméně to jsou hlavičkové soubory stabilního jádra, takže při překladu normálních programů se používají starší hlavičky. OMS potřebuje hlavičkové soubory s podporou *ioctl* volání pro DVD. Pokud je někde něco špatně nastaveno, není deklarována spousta věcí a překlad skončí chybou. Nejjednodušší metoda je použít konfigurační volbu *--with-kernel-headers*=(cesta k hlavičkovým souborům). Další možnost je upravit hlavičky na standardním umístění:

```
# mkdir /usr/include/old
# mv /usr/include/linux /usr/include/old/linux
# mv /usr/include/asm /usr/include/old/asm
# mv /usr/include/scsi /usr/include/old/scsi
# ln -s /usr/src/linux/include/linux /usr/include/linux
# ln -s /usr/src/linux/include/scsi /usr/include/scsi
# ln -s /usr/src/linux/include/asm /usr/include/asm
```

Can’t determine absolute dir of ‘../../../../src/plugin/codec/mpeg2dec/.libs’

Další běžné chyby vznikají s adresářem *oms/src/plugin/codec/mpeg2dec/.libs*. Z nějakého důvodu tento adresář neexistuje, k překladu je ale zapotřebí. Stačí jej vytvořit:

```
# mkdir src/plugin/codec/mpeg2dec/.libs
```

Can't find libXv.so or libXxf86dga.so

Xfree86 4.x nemá sestaveny sdílené knihovny pro Xv a Xxf86dga. K překladu OMS jsou ale tyto knihovny zapotřebí. V adresáři s knihovnami je sestavte příkazem:

```
# ld --whole-archive -shared -o libXv.so libXv.a
# ld --whole-archive -shared -o libXxf86dga.so libXxf86dga.a
```

Chyby segmentace bez zřejmého důvodu

Někdy se stane, že se omylem použijí starší knihovny zapomenuté v systému. To vede k nechtěným chybám a haváriím. Smažte z */usr/local/lib* všechny staré knihovny a mělo by to pomoci.

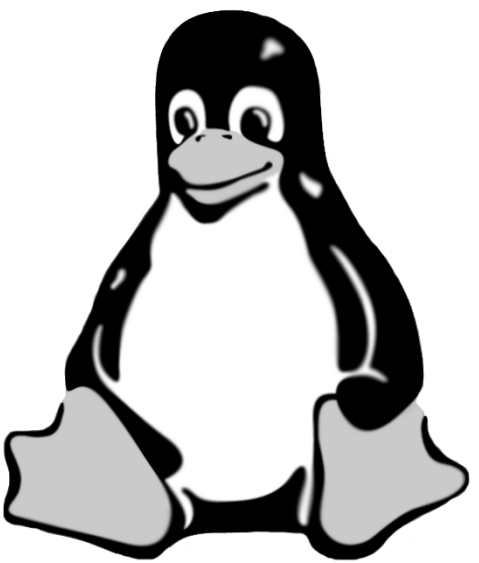
Nefungující funkce

Některé funkce nemusejí ve stávajících verzích fungovat správně:

- Nemusí fungovat procházení kapitolami.
- Nemusí fungovat tlačítka přehrávání/zastavení/pozastavení.
- Zvuk může přeskakovat nebo znít divně. Je to dáno nedostatečnou implementací vhodné synchronizace a zahazování rámců.

Další chyby

Pokud máte problém, který zde není popsán, může aktuální verze CVS kódu obsahovat chybu nebo v něm chybí nějaká funkce. Buď můžete doufat, že brzy dojde k nápravě, nebo se můžete přihlásit do konference a ohlásit chybu nebo požádat o implementaci funkce. Přihlásíte se zasláním prázdného mailu na adresu livid-user-subscribe@linuxvideo.org.



ČÁST V

Všeobecná veřejná licence: GNU

Preambule

Licence pro většinu programového vybavení jsou navrženy tak, že vám odebírají právo jeho volného sdílení a úprav. Smyslem Obecné veřejné licence GNU je naproti tomu zaručit volnost sdílení a úpravy volného programového vybavení – pro zajištění volného přístupu k tomuto programovému vybavení pro všechny jeho uživatele. Tato Obecná veřejná licence GNU se vztahuje na většinu programového vybavení nadace Free Software Foundation a na jakýkoli jiný program, jehož autor se přikloní k jejímu používání. (Některé další programové vybavení od Free Software Foundation je namísto toho pokryto Obecnou knihovní veřejnou licencí GNU.) Můžete ji rovněž použít pro své programy.

Pokud mluvíme o volném programovém vybavení, máme na mysli volnost, nikoliv cenu. Naše Obecná veřejná licence je navržena pro zajištění toho, že můžete volně šířit kopie volného programového vybavení (a účtovat si za tuto službu, pokud chcete), že obdržíte zdrojový kód anebo jej můžete získat, pokud chcete, že můžete tento software měnit nebo jeho části použít v nových programech; a že víte, že tyto věci smíte dělat.

Abychom mohli vaše práva chránit, musíme vytvořit omezení, která zakáží komukoli vám tato práva odepírat nebo vás žádat, abyste se těchto práv vzdal. Tato omezení se promítají do jistých povinností, kterým musíte dostát, pokud šíříte kopie dotyčného programového vybavení anebo ho modifikujete.

Například, šíříte-li kopie takového programu, ať již zdarma nebo za poplatek, musíte poskytnout příjemcům všechna práva, která máte sám. Musíte zaručit, že příjemci rovněž dostanou anebo mohou získat zdrojový kód. A musíte jim ukázat tyto podmínky, aby znali svá práva.

Vaše práva chráníme ve dvou krocích: (1) autorizací programového vybavení, a (2) nabídkou této licence, která vám dává právoplatné svolení ke kopírování, šíření a modifikaci programového vybavení.

Kvůli ochraně každého autora i nás samotných chceme zaručit, aby každý chápal skutečnost, že pro volné programové vybavení nejsou žádné záruky. Je-li programové vybavení někým jiným modifikováno a posláno dále, chceme, aby příjemci věděli, že to, co mají, není originál, takže jakékoliv problémy vnesené jinými se neodrazí na reputaci původních autorů.

Konečně, každý volný program je neustále ohrožen softwareovými patenty. Přejeme si zamezit nebezpečí, že redistributoři volného programu obdrží samostatně patentová osvědčení a tím učiní program vázaným. Abychom tomu zamezili, deklarovali jsme, že každý patent musí být buď vydán s tím, že umožňuje volné užití, anebo nesmí být vydán vůbec.

Přesná ustanovení a podmínky pro kopírování, šíření a modifikaci jsou uvedeny dále.

Ustanovení a podmínky pro kopírování, distribuci a modifikaci

Tato licence se vztahuje na kterýkoliv program či jiné dílo, které obsahuje zmínku, umístěnou v něm držitelem autorských práv, o tom, že dílo může být šířeno podle ustanovení Obecné veřejné licence GNU. V dalším textu znamená „Program“ každý takový program nebo dílo a „dílo založené na Programu“ znamená buď Program samotný anebo každé jiné dílo z něj odvozené, které podléhá autorskému zákonu: tím se míní dílo obsahující Program nebo jeho část, buď doslovně anebo s modifikacemi, popřípadě v překladu do jiného jazyka. (Nadále je překlad zahrnován bez omezení pod pojmem „modifikace“ .) Každý uživatel licence je označován jako „vy“ .

Jiné činnosti než kopírování, šíření a modifikace nejsou pokryty touto licencí; sahají mimo její rámec. Akt spuštění programu není omezen a výstup z Programu je pokryt pouze tehdy, jestliže obsah výstupu tvoří dílo založené na Programu (nezávisle na tom, zda bylo vytvořeno činností Programu). Posouzení platnosti předchozí věty závisí na tom, co Program dělá.

1. Smíte kopírovat a šířit doslovné kopie zdrojového kódu Programu tak, jak jste jej obdržel, a na libovolém médiu, za předpokladu, že na každé kopii viditelně a náležitě zveřejníte zmínku o autorských právech a absenci záruky; ponecháte nedotčené všechny zmínky vztahující se k této licenci a k absenci záruky; a dáte každému příjemci spolu s Programem kopii této licence.

   Za fyzický akt přenesení kopie můžete žádat poplatek a podle vlastního uvážení můžete nabídnout za poplatek záruční ochranu.

2. Můžete modifikovat vaši kopii či kopie Programu anebo kterékoliv jeho části, a tak vytvořit dílo založené na Programu, a kopírovat a rozšiřovat takové modifikace či dílo podle platné podmínky sekce 1, za předpokladu, že splníte všechny tyto podmínky:

   a) Modifikované soubory musíte opatřit zřetelnou zmínkou uvádějící, že jste soubory změnil a datum každé změny.

   b) Musíte umožnit, aby jakékoliv vámi publikované dílo či rozšiřované dílo, které obsahuje zcela nebo jen zčásti Program nebo jakoukoli jeho část, popřípadě je z Programu nebo jeho části odvozeno, mohlo být jako celek bezplatně poskytnuto každé třetí osobě v souladu s ustanoveními této licence.

   c) Pokud modifikovaný program pracuje tak, že čte interaktivně povely, musíte zjistit, že při nejběžnějším způsobu jeho spuštění vytiskne nebo zobrazí hlášení zahrnující příslušnou zmínku o autorském právu a uvede, že neexistuje žádná záruka (nebo popřípadě, že záruku poskytujete vy), a že uživatelé mohou za těchto podmínek Program redistribuovat, a musí uživateli sdělit, jakým způsobem může nahlédnout do kopie této licence. (Výjimka: v případě, že sám program je interaktivní, avšak žádné takové hlášení nevypisuje, nepožaduje se, aby vaše dílo založené na Programu takové hlášení vypisovalo.)

   Tyto požadavky se vztahují k modifikovanému dílu jako celku. Pokud lze identifikovat části takového díla, které zřejmě nejsou odvozeny z Programu a mohou být samy o sobě rozumně považovány za nezávislá a samostatná díla, pak se tato licence a její ustanovení nevztahují na tyto části, jsou-li šířeny jako nezávislá díla. Avšak jakmile tyto části rozšiřujete jako části celku, jímž je dílo založené na Programu, musí být rozšiřování tohoto celku podřízeno ustanovení této licence tak, že povolení poskytnutá dalším uživatelům se rozšíří na celé dílo, tedy na všechny jeho části bez ohledu na to, kdo kterou část napsal.

   Smyslem tohoto paragrafu tedy není získání práv na dílo zcela napsané vámi ani popírání vašich práv vůči němu; skutečným smyslem je výkon práva na řízení distribuce odvozených nebo kolektivních děl založených na Programu.

   Pouhé spojení jiného díla, jež není na Programu založeno, s Programem (anebo dílem založeným na Programu) na paměťovém nebo distribučním médiu neuvazuje toto jiné dílo do působnosti této licence.

3. Můžete kopírovat a rozšiřovat Program (nebo dílo na něm založené, viz sekce 2 v objektové anebo spustitelné podobě podle ustanovení sekcí 1 a 2 výše, pokud splníte některou z následujících náležitostí:

a) Doprovodíte jej zdrojovým kódem ve strojově čitelné formě. Zdrojový kód musí být rozšiřován podle ustanovení sekcí 1 a 2, a to na médiu běžně používaném pro výměnu programového vybavení; nebo

b) Doprovodíte je písemnou nabídkou nejméně na tři roky, podle níž poskytnete jakékoli třetí straně, za poplatek nepřevyšující vaše výdaje vynaložené na fyzickou výrobu zdrojové distribuce, kompletní strojově čitelnou kopii odpovídající zdrojovému kódu, jenž musí být šířen podle ustanovení sekcí 1 a 2 na médiu běžně používaném pro výměnu programového vybavení; nebo

c) Doprovodíte jej informacemi, které jste dostal ohledně nabídky na poskytnutí zdrojového kódu. (Tato alternativa je povolena jen pro nekomerční šíření a jenom tehdy, pokud jste obdržel program v objektovém nebo spustitelném tvaru spolu s takovou nabídkou, v souladu s položkou b výše.)

Zdrojový kód k dílu je nejvhodnější formou díla z hlediska jeho případných modifikací. Pro dílo ve spustitelném tvaru znamená úplný zdrojový kód veškerý zdrojový kód pro všechny moduly, které obsahuje, plus jakékoli další soubory pro definici rozhraní, plus dávkové soubory potřebné pro kompilaci a instalaci spustitelného programu. Zvláštní výjimkou jsou však ty programové komponenty, které jsou normálně šířeny (buď ve zdrojové nebo binární formě) s hlavními součástmi operačního systému, na němž spustitelný program běží (tj. s překladačem, jádrem apod.). Tyto komponenty nemusí být šířeny se zdrojovým kódem, pokud ovšem komponenta sama nedoprovází spustitelnou podobu díla.

Je-li šíření objektového nebo spustitelného kódu činěno nabídkou přístupu ke kopírování z určitého místa, potom se za distribuci zdrojového kódu počítá i nabídnutí ekvivalentního přístupu ke kopírování zdrojového kódu ze stejného místa, byť přitom nejsou třetí strany nuceny ke zkopírování zdrojového kódu spolu s objektovým.

4. Nesmíte kopírovat, modifikovat, poskytovat sublicence anebo šířit Program jiným způsobem než výslovně uvedeným v této licenci. Jakýkoli jiný pokus o kopírování, modifikování, poskytnutí sublicence anebo šíření Programu je neplatný a automaticky ukončí vaše práva daná touto licencí. Strany, které od vás obdržely kopie anebo práva v souladu s touto licencí, však nemají své licence ukončeny, dokud se jim plně podřizují.

5. Není vaší povinností tuto licenci přijmout, protože jste ji nepodepsal. Nic jiného vám však nedává možnost kopírovat nebo šířit Program nebo odvozená díla. V případě, že tuto licenci nepřijmete, jsou tyto činnosti zákonem zakázány. Tím pádem modifikací anebo šířením Programu (anebo každého díla založeného na Programu) vyjadřujete své podřízení se licenci a všem jejím ustanovením a podmínkám pro kopírování, modifikování a šíření Programu a děl na něm založených.

6. Pokaždé, když redistribuujete Program (nebo dílo založené na Programu), získává příjemce od původního držitele licence právo kopírovat, modifikovat a šířit Program v souladu s těmito ustanoveními a podmínkami. Nesmíte klást žádné překážky výkonu zde zaručených příjemcových práv. Nejste odpovědný za vymáhání dodržování této licence třetími stranami.

7. Jsou-li vám z rozhodnutí soudu, obviněním z porušení patentu nebo z jakéhokoli jiného důvodu (nejen v souvislosti s patenty) uloženy takové podmínky (ať již příkazem soudu, smlouvou nebo jinak), které se vylučují s podmínkami této licence, nejste tím osvobozen od podmínek této licence. Pokud nemůžete šířit Program tak, abyste vyhověl zároveň svým závazkům vyplývajícím z této licence a jiným platným zůvazkům, nesmíte jej v důsledku toho šířit vůbec. Pokud by například patentové osvědčení nepovolovalo bezplatnou redis-

tribuci Programu všemi, kdo vaším přičiněním získají přímo nebo nepřímo jeho kopie, pak by jediný možný způsob jak vyhovět zároveň patentovému osvědčení i této licenci spočíval v ukončení distribuce Programu.

Pokud by se za nějakých specifických okolností jevila některá část tohoto paragrafu jako neplatná nebo nevynutitelná, považuje se za směrodatnou rovnováha vyjádřená tímto paragrafem a paragraf jako celek se považuje za směrodatný za jiných okolností.

Smyslem tohoto paragrafu není navádět vás k porušování patentů či jiných ustanovení autorského práva, anebo tato ustanovení zpochybňovat; jediným jeho smyslem je ochrana integrity systému šíření volného programového vybavení, které je podloženo veřejnými licenčními předpisy. Mnozí lidé poskytli své příspěvky do širokého okruhu programového vybavení šířeného tímto systémem, spolehnuvše na jeho důsledné uplatňování; záleží na autorovi/dárci, aby rozhodl, zda si přeje šířit programové vybavení pomocí nějakého jiného systému a žádný uživatel licence nemůže takové rozhodnutí zpochybňovat.

Smyslem tohoto paragrafu je zevrubně osvětlit to, co je považováno za důsledek plynoucí ze zbytku této licence.

8. Pokud je šíření či použití Programu v některých zemích omezeno buď patenty anebo autorsky chráněnými rozhraními, může držitel původních autorských práv, který svěřuje Program do působnosti této licence, přidat výslovně omezení pro geografické šíření, vylučující takové země, takže šíření je povoleno jen v těch zemích nebo mezi těmi zeměmi, které nejsou tímto způsobem vyloučeny. Tato licence zahrnuje v tomto případě takové omezení přesně tak, jako bylo zapsáno v textu této licence.
9. Free Software Foundation může čas od času vydávat upravené nebo nové verze Obecné veřejné licence. Takové nové verze se budou svým duchem podobat současné verzi, v konkrétních věcech se však mohou lišit s ohledem na nové problémy či zájmy.

   Každé nové verzi je přiděleno rozlišující číslo verze. Pokud Program specifikuje číslo verze, která se na něj vztahuje, a „všechny následující verze“ , můžete se podle uvážení řídit ustanoveními a podmínkami buďto oné konkrétní verze anebo kterékoliv následující verze, kterou vydala Free Software Foundation. Jestliže Program nespecifikuje číslo verze této licence, můžete si vybrat libovolnou verzi, kterou kdy Free Software Foundation vydala.
10. Pokud si přejete zahrnout části Programu do jiných volných programů, jejichž distribuční podmínky jsou odlišné, zašlete autorovi žádost o povolení. V případě programového vybavení, k němuž vlastní autorská práva Free Software Foundation, napište Free Software Foundation; někdy činíme výjimky ze zde uvedených ustanovení. Naše rozhodnutí bude vedeno dvěma cíli; zachováním volné povahy všech odvozenin našeho volného programového vybavení a podporou sdílení a opětovného využití programového vybavení obecně.

ZÁRUKA SE NEPOSKYTUJE

11. VZHLEDEM K BEZPLATNÉMU POSKYTNUTÍ LICENCE K PROGRAMU SE NA PROGRAM NEVZTAHUJE ŽÁDNÁ ZÁRUKA, A TO V MÍŘE POVOLENÉ ZÁKONEM. POKUD NENÍ PÍSEMNĚ STANOVENO JINAK, POSKYTUJÍ DRŽITELÉ AUTORSKÝCH PRÁV POPŘÍPADĚ JINÉ STRANY PROGRAM „TAK, JAK JE“, BEZ ZÁRUKY JAKÉHOKOLIV DRUHU, AŤ VÝSLOVNÉ NEBO VYPLÝVAJÍCÍ, VČETNĚ, ALE NIKOLI JEN, ZÁRUK PRODEJNOSTI A VHODNOSTI PRO URČITÝ ÚČEL. POKUD JDE O KVALITU A VÝKONNOST PROGRAMU, LEŽÍ VEŠKERÉ RIZIKO NA VÁS. POKUD BY SE U PROGRAMU PROJEVILY ZÁVADY, PADAJÍ NÁKLADY ZA VŠECHNU POTŘEBNOU ÚDRŽBU, OPRAVY ČI NÁPRAVU NA VÁŠ VRUB.

12. V ŽÁDNÉM PŘÍPADĚ, S VÝJIMKOU TOHO, KDYŽ TO VYŽADUJE PLATNÝ ZÁKON, ANEBO KDYŽ TO BYLO PÍSEMNĚ ODSOUHLASENO, VÁM NEBUDE ŽÁDNÝ Z DRŽITELŮ AUTORSKÝCH PRÁV ANI ŽÁDNÁ JINÁ STRANA, KTERÁ SMÍ MODIFIKOVAT ČI ŠÍŘIT PROGRAM V SOULADU S PŘEDCHOZÍMI USTANOVENÍMI, ODPOVĚDNÝ ZA ŠKODY, VČETNĚ VŠECH OBECNÝCH, SPECIÁLNÍCH, NAHODILÝCH NEBO NÁSLEDNÝCH ŠKOD VYPLÝVAJÍCÍCH Z UŽÍVÁNÍ ANEBO NESCHOPNOSTI UŽÍVAT PROGRAMU (VČETNĚ ALE NIKOLI JEN, ZTRÁTY NEBO ZKRESLENÍ DAT, NEBO TRVALÝCH ŠKOD ZPŮSOBENÝCH VÁM NEBO TŘETÍM STRANÁM, NEBO SELHÁNÍ FUNKCE PROGRAMU V SOUČINNOSTI S JINÝMI PROGRAMY), A TO I V PŘÍPADĚ, ŽE TAKOVÝ DRŽITEL AUTORSKÝCH PRÁV NEBO JINÁ STRANA BYLI UPOZORNĚNI NA MOŽNOST TAKOVÝCH ŠKOD.

Jak uplatnit tato ustanovení na vaše nové programy

Pokud vyvinete nový program a chcete, aby byl veřejnosti co nejvíce k užitku, můžete toho nejlépe dosáhnout tím, že jej prohlásíte za volné programové vybavení, které může kdokoliv redistribuovat a měnit za zde uvedených podmínek.

K tomu stačí připojit k programu následující údaje. Nejbezpečnější cestou je jejich připojení na začátek každého zdrojového souboru, čímž se nejúčinněji sdělí vyloučení záruky; a v každém souboru by pak měla být přinejmenším řádka s „copyrightem" a odkaz na místo, kde lze nalézt úplné údaje.

```
řádka se jménem programu a nástinem toho, co dělá
Copyright (C) 19yy  jméno autora

Tento program je volné programové vybavení; můžete jej šířit a
modifikovat podle ustanovení Obecné veřejné licence GNU, vydávané Free
Software Foundation; a to buď verze 2 této licence anebo (podle vašeho
uvážení) kterékoli pozdější verze.

Tento program je rozšiřován v naději, že bude užitečný, avšak BEZ
JAKÉKOLI ZÁRUKY; neposkytují se ani odvozené záruky PRODEJNOSTI anebo
VHODNOSTI PRO URČITÝ ÚČEL.
Další podrobnosti hledejte v Obecné veřejné licenci GNU.

Kopii Obecné veřejné licence GNU jste měl obdržet spolu s tímto
programem; pokud se tak nestalo, napište o ni Free Software Foundation,
Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Připojte rovněž informaci o tom, jak je možné se s vámi spojit elektronickou a papírovou poštou.

Pokud je program interaktivní, zařiďte, aby se při startu v interaktivním módu vypsalo hlášení podobné tomutu:

```
Gnomovision verze 69, Copyright (C) 19yy jméno autora
Gnomovision je ABSOLUTNĚ BEZ ZÁRUKY; podrobnosti se dozvíte zadáním
show w. Jde o volné programové vybavení a jeho šíření za jistých podmínek
je vítáno; podrobnosti získáte zadáním show c.
```

Hypotetické povely `show w` a `show c` by měly zobrazit příslušné pasáže Obecné veřejné licence. Odpovídající povely ovšem nemusí být právě `show w` a `show c`; mohou to být třeba stisky tlačítka na myši nebo položky v menu - cokoliv, co se do vašeho programu hodí.

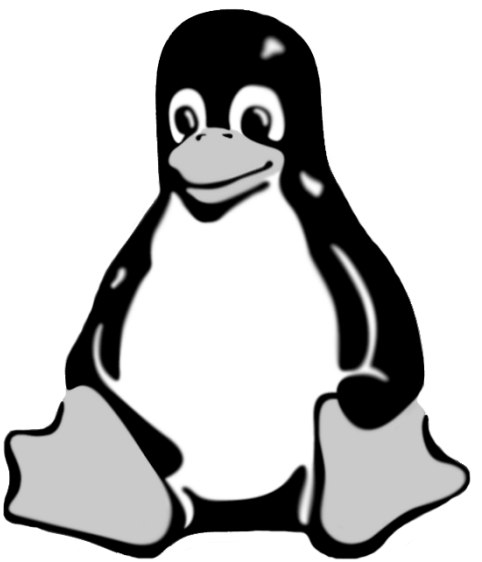
Pokud je to nutné, měl byste také přimět svého zaměstnavatele (jestliže pracujete jako programátor) nebo představitele vaší školy, je-li někdo takový, k tomu, aby podepsal „zřeknutí se autorských práv“ . Zde je vzor; jména pozměňte:

```
Yoyodyne, a.s., se tímto zříká veškerého zájmu o autorská práva k programu
`Gnomovision' (překladač s nakladačem) napsaného Jamesem Hackerem.

Tomáš Složitý --- podpis}, 1. dubna 1989
Tomáš Složitý, více než prezident
```

Tato Obecná veřejná licence neumožňuje zahrnutí vašeho programu do jiných než volných programů. Je-li váš program knihovnou podprogramů, můžete zvážit, zda je užitečné umožnit sestavování i vázaných aplikačních programů s vaší knihovnou. V takovém případě použijte Obecnou knihovní licenci GNU namísto této licence.

Dotazy a připomínky ohledně stránky posílejte na hubicka@paru.cas.cz.



Rejstřík

C

Č

D

E

F

G

H

Ch

I

J

K

L

M

N

O

P

Q

R

Ř

S

Š

T

U

V

W

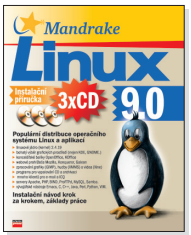
X

Y

Z

Ž

Další knihy z nabídky nakladatelství Computer Press



K0822 56 stran + 3xCD-ROM

Vlastimil Pošmura, Ivan Bíbr:

Linux Mandrake 9.0
3xCD a Instalační příručka

295 Kč

Mandrake je dnes zřejmě nejoblíbenější linuxovou distribucí, snadnou pro obyčejné uživatele a spolehlivou pro serverové řešení. Obsahuje mix různých grafických prostředí (KDE, GNOME, IceWM, WindowMaker aj.), kancelářský balík OpenOffice, nejrůznější serverové produkty, vývojářské nástroje i multimediální aplikace. Doprovodná příručka popisuje krok po kroku instalaci, konfiguraci i základy práce.

K0866 280 stran

Vilém Vychodil:

Linux
Příručka českého uživatele systému

269 Kč

První tištěné vydání publikace, jež získala početnou čtenářskou obec již ve své elektronické podobě, nyní aktualizovaná a doplněná. Obsahuje vysvětlení základních principů i detaily uživatelské práce, souborového systému, systému procesů, příkazového interpretu, zpracování textu, síťového prostředí, grafického síťového rozhraní i specifika a možnosti českého prostředí.

K0815 642 stran + CD-ROM

Michael Lucas:

FreeBSD
Podrobný průvodce síťovým operačním systémem

490 Kč

BSD je volně šiřitelný operační systém se čtvrtstoletou historií, který byl v posledních letech dotažen do podoby, v níž předčí řadu jiných systémů unixového typu – například v kompatibilitě a bezpečnosti. Proto se čím dál více nasazuje především jako internetový, aplikační nebo databázový server, ale stále populárnější je i mezi samostatnými uživateli. Kniha důkladně provádí začínajícího uživatele instalací a všemi oblastmi provozu a správy FreeBSD, včetně nastavení českého prostředí, které FreeBSD výborně podporuje. Díky hloubce i šíři záběru přinese mnoho informací i zkušeným administrátorům. Použitelná i pro varianty OpenBSD a NetBSD. Obsahuje CD s kompletní instalací FreeBSD 4.7.

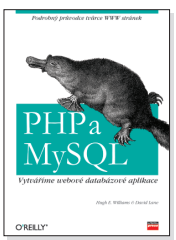
*K0844 192 stran + CD-ROM **

Martin Kysela:

Přecházíme na Linux

189 Kč *

Publikace určená pro uživatele, kteří umějí samostatně pracovat s počítačem, avšak jsou v operačním systému Linux úplnými nováčky. Seznámíte se s výhodami a nevýhodami Linuxu, rozdíly proti jiným systémům, prací v textovém i grafickém režimu, základními příkazy atd. Na CD množství linuxového softwaru.



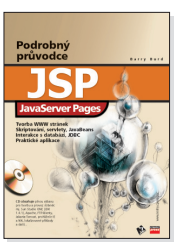
K0812 552 stran

Hugh E. Williams, David Lane:

PHP a MySQL
Vytváříme webové databázové aplikace

490 Kč

Skriptovací jazyk PHP nachází plné uplatnění až v kombinaci s databázovým systémem MySQL. Kniha prestižního nakladatelství O'Reilly seznamuje začínající a částečně znalé tvůrce webů se základy obou technologií a s principy fungování dynamické internetové aplikace. Postupy procvičíte na ukázkové aplikaci, dostatek informací zúročíte nebo kdykoli dohledáte při tvorbě, zabezpečení a údržbě vašich aplikací.



K0821 416 stran + CD-ROM

Barry Burd:

JSP: JavaServer Pages
Podrobný průvodce

369 Kč

Velmi srozumitelný a prakticky koncipovaný průvodce vás krok po kroku seznámí s tvorbou dokumentů a aplikací na platformě Javy. Zvládnete základy technologie JSP, tvorbu skriptů, servletů, apletů i plných aplikací fungujících na Windows, Linuxu i Unixu. Konkrétně pak integraci stránek s databázemi, efektivní techniky programování i testování stránek, využití JavaBeans či užitečné podrobnosti protokolu HTTP. Příklady zahrnují vytvoření internetového nákupního košíku, vlastního webového portálu, diskusního serveru či interaktivní hry. Na CD najdete vše, co je potřeba k tvorbě i provozu stránek v JSP, včetně Java Sun ONE Studia, Apache, JDataConnect, prohlížečů IE6 a NN7, IW FTPort Clientu, příkladů z knihy a dalšího užitečného softwaru.

* Předběžné údaje.

Všechny tyto a mnohé další zajímavé publikace si můžete objednat u zásilkové služby našeho vydavatelství, nebo je žádejte u vašeho knihkupce.

Zásilková služba pro ČR: Computer Press, a.s., Nám. 28. dubna 48, 635 00 Brno
Ojednávejte na: www.knihy.cpress.cz, distribuce@cpress.cz, bezplatná telefonní linka: 800 555 513

Zásilková služba pro SR: Computer Press Bratislava, Hattalova 12, 831 03 Bratislava, Slovenská republika,
tel. +421 (2) 4445 2048, e-mail:distribucia@cpress.sk