LINUX

Příručka správce operačního systému Linux

Kapitola

1

Úvod

Na počátku byl soubor nesličný a pustý, a prázdno se vznášelo nad povrchem bitů. A prsty Autorovy dosedly na povrch klávesnice i řekl Autor: Buďte slova! A stalo se tak.

Manuál „Průvodce správce operačního systému Linux“ popisuje ty aspekty používání operačního systému, jež se vztahují k jeho správě neboli administraci. Je určený 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“. Bližší informace o sadě manuálů 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 (v případě potřeby jejich obnovování), instalaci nových programů, vytváření uživatelských účtů (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. Místo prostého pojmu „údržba systému“ se používá termín „administrace“, aby tato oblast nevypadala na první pohled zas až tak jednoduše.[1]

Příručka je strukturovaná tak, že většinu kapitol můžete číst nezávisle na sobě. Když například hledáte nějaké informace o zálohování, stačí, když si přečtete příslušnou kapitolu.[2] Doufáme, že díky tomu bude možné používat tuto knihu i jako referenční příručku a v případě

1 Přesto se najdou lidé, kteří toto označení *používají*, ale jenom proto, že ještě nikdy nečetli tento manuál. Nešťastníci...

2 Tedy, jestli se k vám – zcela náhodou, dostane verze, která kapitolu o zálohování obsahuje.

potřeby místo „louskání“ celého textu číst pouze jeho menší části. Přesto manuál zůstává především a převážně učebnicí a jakýmsi průvodcem. To, že poslouží i jako příručka referencí je pouze šťastným řízením osudu.

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 i manuálové stránky, po kterých by správce měl sáhnout pokaždé, když si není funkcí některého příkazu zcela jist.

Manuál je zaměřen především na operační systém Linux, ale v obecných principech může být užitečný i pro správce jiných unixových systémů. Bohužel je mezi různými verzemi Unixu tolik rozdílů (o správě systému to platí dvojnásob), že není prakticky možné postihnout všechny známé variety. Je totiž obtížné – vezmeme-li v potaz způsob, jakým se Linux vyvíjí – pokrýt i všechny možnosti tohoto operačního systému samotného.

Kromě toho 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 (i když autor osobně dává téměř výlučně přednost systému Debian GNU/Linux). V rámci možností se v příručce snažíme upozornit na takovéto odlišnosti a objasnit i jiné možné alternativy.

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 manuálu proto najdete hodně informací, které nejsou nezbytné pro každého správce. 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, no a pak by pro vás mohly být o něco příjemnější i jeho používání a jeho správa.

Tak jako vše ostatní spojené s vývojem Linuxu, byla i tato práce založená 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, takže to berte jako varování pro každý případ.

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 je-

nom 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.

Pokoušeli jsme se napsat tuto příručku co nejlépe, ale budeme vděčni za všechny vaše nápady jak ji vylepšit. Gramatické a věcné chyby, nápady týkajíce 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/liw/mail-to-lasu.html.* Na této stránce najdete i pokyny potřebné pro doručení elektronické pošty přes filtry nevyžádaných zpráv.

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, Theodore Ts'o a Stephen Tweedie odvedli kus práce, kterou jsme si formou odkazů a referencí „zapůjčili" (tím pádem je naše kniha na pohled tenčí a o to víc působivá). 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 Komarinskemu 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.

Linux – dokumentační projekt

Projekt tvorby dokumentace pro operační systém Linux (angl. The Linux Documentation Project, zkráceně LDP) je volné sdružení autorů, korektorů a editorů, kteří spolupracují na úplné dokumentaci systému. Hlavním koordinátorem projektu je Greg Hankins.

Tento manuál je jedním ze sady dokumentů, které jsou v rámci LDP šířeny. Tato sada obsahuje Průvodce uživatele systému, Průvodce správce systému, Průvodce správce sítě, Průvodce jádrem systému. Všechny příručky jsou k dispozici ve formátu zdrojového kódu pro sázecí systém LaTex, ve formátu .dvi a v jazyce Postscript, a to na anonymním serveru FTP s adresou `ftp://sunsite.unc.edu`, v adresáři `/pub/Linux/docs/LDP`.

Rádi bychom v závěru dodali odvahy všem čtenářům se sklonem k tvůrčímu psaní či redigování, aby se připojili k naší snaze o zdokonalení dokumentace operačního systému Linux. Máte-li zájem a přístup k účtu pro elektronickou poštu, kontaktujte prostřednictvím Internetu Grega Hankinse na adrese `gregh@sunsite.unc.edu`.

Óda na LDP[3]

Jak úžasná věc
a krásná
napsat knihu.

Zpíval bych rád
i o potu, krvi a slzách.
i z těch kniha vzniká.

Začalo to kdysi
v dvaadevadesátém,
uživatelé kňučeli
„Nemůžem nic dělat!“

Chtěli jenom vědět,
v čem je jejich problém
a jak jej vyřešit
(nejlépe do zítřka).

Dali jsme jim odpovědi
v Č-K-D[4] pro Linux,
doufajíce, že je amen –
hlavně žádné další psaní.

„Příliš dlouhé,
nepřehledné,
k nepřečtení –
ať děláme, co děláme!“
Pak několik z nás
spojilo síly
(samozřejmě virtuálně)
a odstartoval LDP.

Začali jsme psát,
plánovat,
přinejmenším několik knih
pro každou z oblastí.

Začátek byl zábavný,
dlouhé hovory,
hrubý obrys
a pak zával.

Padlo ticho,
začala práce,
někdo psal míň,
někdo víc.

Prázdná obrazovka,
ach, ta hrůza,
sedí a směje se
vám do očí.

Stále čekáme
na poslední den,
kdy řekneme
hotovo.

Než přijde,
je vše, co máme,
pouhým návrhem,
čekajícím na vaše
připomínky.

3 Autor si přeje zůstat v anonymitě (dílko zaslal do diskusní skupiny LDP Matt Welsh).

4 Často kladené dotazy, angl. Frequently-Asked Question, zkráceně F-A-Q (poznámka překladatele).

Kapitola

2

Operační systém Linux – přehled

A viděl Bůh, že vše, což učinil, bylo velmi dobré.
Genesis 1:31

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 probraná později, na jiném místě knihy.

2.1 Různé části operačního systému

Operační systém typu Unix se skládá z **jádra systému** a **systémových programů**. Uživatel systému pracuje s **aplikačními programy**. Jádro je srdcem operačního systému[1]. Udržuje záznamy souborů 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 zařízením přímo. Když chtějí uživatelé a procesy používat technické prostředky, musí používat nástroje, které nabízí jádro systému. Tímto způsobem je zabezpečená i vzájemná ochrana uživatelů. Nástroje jádra systému, o nichž byla řeč, lze využívat prostřednictvím **volání systému** (angl. **system calls**). Podrobnější informace o systémových voláních uvádí sekce 2 manuálových stránek.

---

1 Skutečně často se jádro systému chybně ztotožňuje se samotným operačním systémem. Ale operační systém poskytuje ve srovnání s prostým jádrem o hodně více služeb.

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** (angl. **user mode**). 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, `telnet` je systémový program. Hranice mezi aplikačními a systémovými programy je často dost neostrá. Lze prohlásit, že takovéto rozdělení je samoúčelné – důležité čistě pro vymezení samotných klasifikačních kritérií.

Součástí operačního systému mohou být i překladače programovacích jazyků a jejich knihovny (v případě Linuxu knihovny překladačů GCC a C). Součástí operačního systému ale nejsou všechny programovací jazyky. Naopak, za jeho součást se často považuje 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.

2.2 Důležité části jádra systému

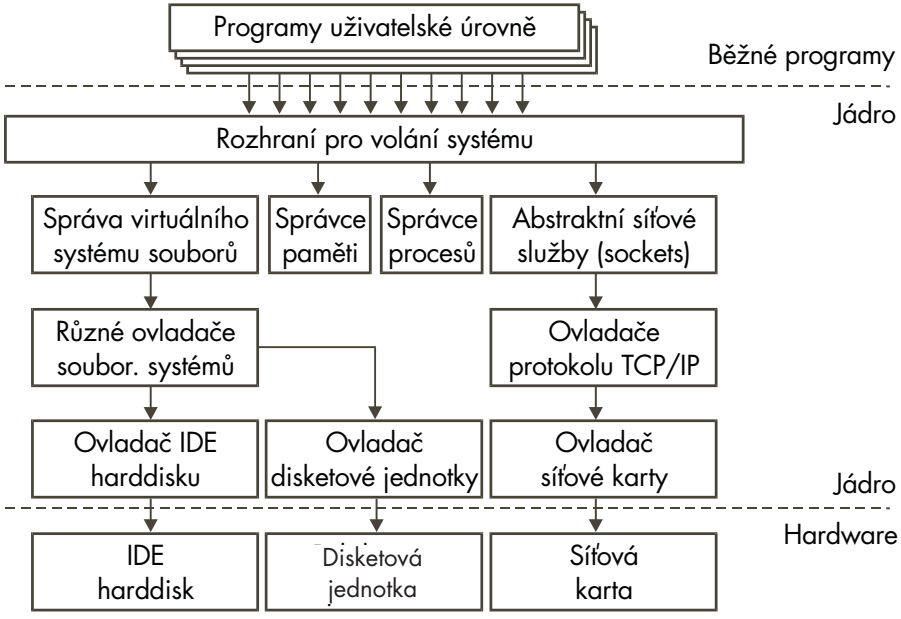
Jádro Linuxu sestává z několika důležitých subsystémů. Jsou to části řízení procesů, správy paměti, ovladačů technických prostředků, ovladačů souborových systémů, správy sítě a různé další kusy a kousky. Některé z nich jsou zobrazeny na obrázku 2.1.

Snad nejdůležitějšími subsystémy (bez nichž nic jiného nefunguje) jsou správa paměti a správa procesů. Subsystém správy paměti zajišťuje přidělování paměťových oblastí a odkládacího prostoru (angl. swap space) jednotlivým procesům, částem jádra a vyrovnávací paměti (angl. buffer cache). 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, které 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. Ku příkladu různé síťové protokoly byly vyčleněné do jednoho programového rozhraní – knihovny „BSD socket library“. Dalším příkladem je

vrstva **virtuálního souborového systému** (angl. **virtual filesystem**, zkráceně 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 směřuje k požadovanému ovladači konkrétního systému souborů.



Obrázek 2.1
Některé z důležitějších částí jádra systému Linux

2.3 Nejdůležitější služby v unixovém systému

Tato podkapitola popisuje některé významnější služby systému Unix, avšak opět bez větších podrobností. Všechny služby budou později podrobně vysvětlené v dalších kapitolách.

2.3.1 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 atd.).

Přesný seznam úloh, které `init` při zavádění dělá, závisí na verzi tohoto programu i operačního systému. Je na výběr několik možností. Proces `init` často obstarává tzv. **jednouživatelský režim** (angl. **single user mode**). V jednouživatelském režimu se do systému nemůže nikdo přihlásit. Příkazový interpret může z konzoly používat pouze superuživatel (správce). Běžným režimem práce je **víceuživatelský režim** (angl. **multiuser mode**). Tyto režimy práce některé systémy Unix zobecňují do tzv. **úrovní běhu systému** (angl. **run levels**). 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.

V běžné situaci program `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 „umřeli“. 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.

2.3.2 Přihlášení z terminálů

Přihlášení uživatelů prostřednictvím terminálů (pomocí sériových linek) a konzoly (v případě, že neběží X Window) do systému 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 další 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 pojem o přihlašování uživatelů do systému. Všechno kolem toho obstarávají systémové programy.

2.3.3 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 starost, 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 směřována 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.

2.3.4 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 tabulku `crontab`, jež obsahuje seznam příkazů, které chce vlastník spustit, a časy, kdy se mají tyto příkazy provést. Démon `cron` má na starost 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.

2.3.5 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 neimplementují uživatelské rozhraní. X Window pouze zavádí systém oken, tedy sadu nástrojů, pomocí kterých může být grafické uživatelské rozhraní implementované.

Tři z nejpopulárnějších stylů uživatelských rozhraní postavených na X jsou Athena, Motif a Open Look.*

* Poznámka korektora: V současné době patří mezi nejpopulárnější uživatelská rozhraní KDE (http://www.kde.org) nebo GNOME (http://www.gnome.org).

2.3.6 Komunikace prostřednictvím počítačové sítě

Komunikace pomocí počítačové sítě neboli síťování (angl. networking) je propojení dvou nebo více počítačů tak, že mohou komunikovat navzájem každý s každým. V současnosti používané metody propojování a komunikace jsou o něco komplikovanější, ale výsledný efekt stojí za to.

Operační systémy Unix mají řadu síťových funkcí. Většinu základních služeb – služby souborových systémů, tisky, zálohování atd., 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 „Průvodce správce sítě".

2.3.7 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. Tímto spojením se může realizovat libovolný počet běžných přihlášení[2]. 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`.

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`.*

2.3.8 Síťové souborové systémy

Jednou z nejužitečnějších věcí, kterou lze využít díky síťovým službám, je sdílení souborů pomocí **síťového souborového systému** (angl. **network file system**). Jeden z nejběžněji používaných typů se nazývá Network File System, zkráceně NFS, a byl vyvinut společností Sun.

---

2 No, může jich být přinejmenším mnoho. Ještě stále je totiž šířka přenosového pásma sítí problémem, takže existuje jakási praktická horní hranice počtu současných přihlášení do systému prostřednictvím jediného síťového spojení.

* Poznámka korektora: Od služeb `telnet` a `rlogin` se upouští a nahrazuje je program `ssh`.

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.

2.3.9 Pošta

Elektronická pošta je obvykle tím nejdůležitějším způsobem počítačové komunikace. Elektronický dopis je uložený v souboru se zvláštním formátem. K jeho odeslání nebo přečtení se používají speciální programy.

Každý uživatel systému má vlastní **schránku na příchozí poštu** (angl. **incoming mailbox**). Je to soubor určitého formátu, ve kterém jsou uložené 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 starost první z nich - **agent pro přenos pošty** (angl. **Mail Transfer Agent** - zkráceně **MTA**), např. `sendmail` nebo `smail`. Uživatelé používají ke čtení pošty množství různých programů, tzv. **uživatelských poštovních agentů** (angl. **Mail User Agent** - zkráceně **MUA**), např. `pine` nebo `elm`. Poštovní schránky uživatelů jsou obvykle uloženy v adresáři `/var/spool/mail`.

2.3.10 Tisk

Tiskárnu může současně využívat pouze jeden uživatel. Nesdílet tiskárny mezi uživateli je ale dost neekonomické. Tiskárnu proto řídí program, jenž realizuje tzv. **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 nutnosti soupeřit a handrkovat se o přístup k tiskárně.[3]

Program pro obsluhu tiskové fronty navíc **ukládá metodou „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 tis-

[3] Nikdo se od programu, jenž obsluhuje tiskovou frontu, přesně nedoví, kdy bude jeho tiskový výstup skutečně ukončený. Uživatelé tedy místo toho - čekajíce na své tiskové výstupy - vytvoří novou frontu *u* tiskárny. To je ohromný příspěvek v oblasti podpory sociálních vztahů na pracovišti.

kovou 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.

2.4 Základní rysy systému souborů

Souborový systém je rozdělený na několik částí. Obvykle jsou hierarchicky uspořádané a nejvýše stojí kořenový souborový systém „root“ (angl. root filesystem). Říká se mu rovněž kořenový svazek, označuje se „/“. Souborový systém „root“ obsahuje adresáře `/bin`, `/lib`, `/etc`, `/dev` a několik dalších. Dalším je systém souborů `/usr`. Obsahuje programy a data, která se nemění. Následuje souborový systém `/var`. Ukládají se v něm data, jež se naopak často mění (například tzv. log-soubory). Posledním je souborový systém `/home`.

Ukládají si v něm svá data a soubory 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 uložené i v jediném systému souborů.

Některé další detaily týkající se uspořádání systému souborů popisuje kapitola 3. Ještě více podrobností o tomto tématu přináší „Standard systému souborů operačního systému Linux“*.

* Poznámka korektora: Dokument „Standard systému souborů operačního systému Linux“ se nyní jmenuje FHS (Filesystem Hierarchy Standard).

Kapitola

3

Přehled struktury adresářů

Dva dny nato seděl Pooh 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 systému souborů standardu FSSTND. Načrtneme v ní také běžný způsob rozdělení struktury adresářů do samostatných souborových systémů (svazků) s odlišnými účely a tento způsob rozdělení zdůvodníme. Naznačíme i některé alternativní způsoby rozdělení.

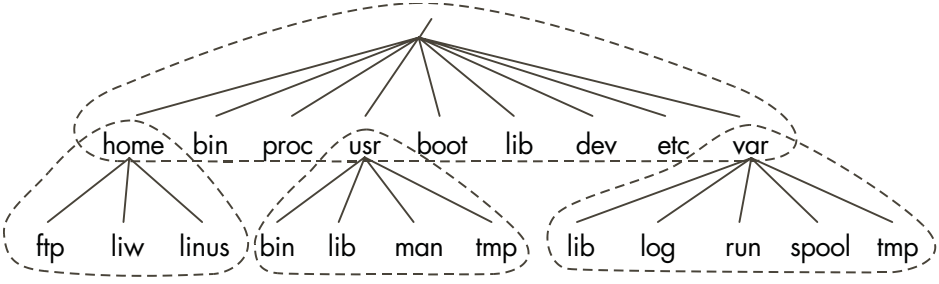
3.1 Základy

Tato kapitola volně vychází z normy „Standard systému souborů operačního systému Linux FSSTND, verze 1.2“ (viz bibliografie [Qui95]), který je pokusem zavést jisté konvence do organizace adresářového stromu operačního systému Linux. 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 FSSTND, nejsou-li pro to velmi závažné důvody. Norma FSSTND 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 FSSTND. Správce systému – chce-li plně proniknout do problematiky systémů souborů – by si měl přečíst i normu FSSTND.

Kapitola rovněž nepopisuje podrobnosti týkající se všech typů souborových systémů. Nebylo také cílem popsat všechny adresáře a konfigurační 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, tzv. svazky. Každá z těchto částí může být umístněna na vlastním disku nebo samostatné diskové oblasti – logické sekci. Tak se lze jednoduše přizpůsobit omezením velikosti disků a zároveň usnadnit zálohování i ostatní úkoly spojené se správou systému. Nejdůležitější z těchto částí jsou souborový systém „root“ (kořenový svazek), dále souborové systémy `/usr`, `/var` a `/home` (viz obrázek 3.1). Každý systém souborů má jiné určení. Adresářová stromová struktura byla navržená 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 (např. CD-ROM), nebo pomocí sítě se systémem NFS.



Obrázek 3.1
Části adresářové struktury Unixu. Přerušované čáry označují hranice diskových oblastí.

Určení takto vymezených částí adresářové struktury je popsané v dalším textu.

- Souborový systém „root“ (kořenový svazek) je specifický pro každý počítač. Obecně je uložen na lokálním disku (avšak může to být i virtuální disk v paměti RAM – tzv. „ramdisk“, nebo síťová disková jednotka). Kořenový svazek obsahuje soubory nutné pro zavedení systému a jeho uvedení do stavu, ve kterém mohou být připojené 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 svazku `/usr` by neměl být specifický pro některý 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 výhod-

né 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 (např. 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, tzv. log-soubory, formátované manuálové stránky a dočasné soubory. Historicky bývaly všechny soubory, které jsou nyní uloženy v systému /var, v souborovém systému /usr. To znemožňovalo připojit svazek /usr pouze pro čtení.

- Souborový systém /home obsahuje domovské adresáře uživatelů. Vyčlenění uživatelských domovských adresářů do vlastní adresářové stromové struktury 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, např. /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 vyhrazený pro různé účely. Jedinou věcí, na kterou je potřeba dbát, jsou standardní unixová *jména*. Ty 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/adm/messages. Totéž platí i v případě, že se například přesune adresář /var do adresáře /usr/var a na původním místě přesunutého adresáře bude symbolický link z adresáře /usr/var.

Struktura souborového systému Unixu sdružuje soubory podle jejich účelu. Jenom tak lze zaručit, že budou například 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 slo-

žité 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.

3.2 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 na tomto svazku nebude možné zavést. Tento problém lze řešit pouze pomocí speciálních opatření (např. zavedením systému z diskety), a to by chtěl riskovat asi málokterý správce. Obecně by kořenový adresář systémového svazku 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 pro zavedení systému a pro práci běžných uživatelů po jeho zavedení.
`/sbin`	Stejné jako u adresáře `/bin`. Příkazy v tomto podadresáři ale nejsou určeny běžným uživatelům, i když je též mohou použít (je-li to nutné nebo možné).
`/etc`	Konfigurační soubory specifické pro daný počítač.
`/root`	Domovský adresář superuživatele.
`/lib`	Sdílené knihovny pro programy v kořenovém souborovém systému.
`/lib/modules`	Zaváděcí moduly jádra systému – zvláště ty, které jsou potřeba pro zavedení systému při zotavení po neočekávaných událostech (např. síťové ovladače a ovladače pro souborový systém).
`/dev`	Speciální soubory.
`/tmp`	Dočasné soubory. Programy, které se spouští až po zavedení systému, by správně měly používat místo adresáře `/tmp` adresář `/var/tmp`, protože je velmi pravděpodobné, že leží na větším disku.
`/boot`	Soubory, jež používá zavaděč operačního systému (angl. bootstrap loader), např. LILO. Je dobré mít v tomto podadresáři uložené obrazy jádra (místo toho, aby se ukládaly přímo v kořenovém adresáři). V přípa-

dě, že jich máte víc, může obsah adresáře `/boot` značně narůst. V tom případě bude lepší mít jej v samostatném souborovém systému. Tím se také zajistí, že obrazy jádra budou uloženy na prvních 1024 cylindrech disku IDE.

`/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 (např. `/mnt/dosa` pro disketovou mechaniku používanou v souborovém systému MS-DOS, `/mnt/exta` pro tutéž mechaniku využívanou v souborovém systému ext2 a podobně).

`/proc`, `/usr`, `/var`, `/home` Přípojná místa pro další souborové systémy.

3.2.1 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 popsané v „Průvodci správce sítě operačního systému Linux“.

`/etc/rc`, `/etc/rc.d` a `/etc/rc?.d`
Skripty a adresáře skriptů, které se spouští při startu, nebo v případě, že se mění úroveň běhu systému. Podrobnější informace najdete v kapitole o procesu `init`.

`/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 pro program *passwd*.

`/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` (skriptem `/etc/rc`, nebo odpovídajícím souborem, jenž se spouští při startu systému). V systému Linux obsahuje rovněž informace o odkládacích oblastech „swap“, které automaticky používá příkaz `swapon -a`. Podrobnější informace viz podkapitola 4.8.5 a manuálové stránky k příkazu *mount*.

`/etc/group` Soubor podobný souboru `/etc/passwd`, ale místo uživatelů popisuje pracovní skupiny. Podrobnější informace viz manuálová stránka pro soubor `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 hlášku. 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` Tzv. „**zpráva pro tento den“** (angl. **message of the day**) – automatický výstup na terminál uživatele po úspěšném přihlášení do systému. Obsah volí správce systému. Často se využívá pro předávání informací (např. 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, např. při zadání příkazu `df`.

`/etc/shadow` Soubor tzv. „stínových“ přístupových hesel (shadow password) uživatelů v systémech, které mají nainstalovanou podporu systému stínových hesel. Kódovaná stínová hesla jsou z bezpečnostních důvodů přeneseny ze souboru `/etc/passwd` do souboru `/etc/shadow`, který může číst pouze superuživatel. Snižuje se tak pravděpodobnost odhalení některého z přístupových hesel při průniku do systému.

`/etc/login.defs`
Konfigurační soubor příkazu `login`.

`/etc/printcap` Podobně jako u souboru `/etc/termcap`, až na to, že soubor je určený pro tiskárny. Odlišná je i jeho syntaxe.

`/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 stejné pro všechny uživatele. Viz manuálové stránky k příslušným interpretům příkazů.

`/etc/securetty` Soubor identifikuje zabezpečené terminály, tedy terminály, ze kterých se může přihlašovat superuživatel. Typicky je v seznamu uvedena pouze virtuální konzola, takže je nemožné (nebo přinejmenším těžší) získat oprávnění superuživatele přihlášením se po modemu nebo ze sítě.

`/etc/shells` Soubor, jenž uvádí seznam důvěryhodných interpretů příkazů. Příkaz `chsh` umožňuje uživatelům změnit shell 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 shell 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`.

3.2.2 Adresář /dev

Adresář `/dev` obsahuje speciální soubory pro všechna zařízení. Speciální soubory jsou pojmenované podle určitých konvencí. Ty jsou podrobně popsán v „Seznamu zařízení operačního systému Linux“ (viz [Anv]). 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 linky. Lokální speciální soubory jsou ty, které nejsou vytvořené standardním postupem pomocí skriptu `MAKEDEV`, typicky například speciální soubory pro některé nestandardní ovladače zařízení.

3.3 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 popsané níže, ty méně významné neuvádíme. Více informací přináší popis standardu FSSTND.

/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 vytvořený pod adresářem /usr samotným.

/usr/X386 — Podobně jako u předchozího adresáře /usr/X11R6, ale pro systém X11 Release 5.

/usr/bin — Zde se nachází téměř všechny uživatelské příkazy. Některé další příkazy jsou uložené v adresáři /bin nebo /usr/local/bin.

/usr/sbin — Obsahuje ty příkazy pro správu systému, které nejsou potřeba přímo v souborovém systému „root“ (zde je například uložená převážná většina serverových programů).

/usr/man, /usr/info, /usr/doc — Manuálové stránky, informační dokumenty o projektu 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 umístěné 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.

3.4 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/catman — Vyrovnávací paměť pro manuálové stránky, které jsou formátované na požádání. Zdroje pro tyto manuálové stránky jsou obvykle uložené v adresáři /usr/man/man*. Manuálové stránky v předem formátované verzi jsou uložené v adresáři /usr/man/cat*. Manuálové stránky je běžně třeba při prvním prohlížení formátovat. Formátované verze jsou pak uložené právě v adresáři /var/catman. Další uživatel, který si chce stejné stránky prohlížet, tak nemusí čekat na jejich opakované formátování. (Soubory v uvedeném adresáři /var/catman je obvykle potřeba mazat, stejně jako dočasné soubory v adresářích /tmp a /var/tmp.)

/var/lib — Soubory, které se při normálním provozu systému mění.

/var/local — Měnící se data pro programy instalované v adresáři /usr/local (tj. programy instalované správcem systému). Upozorňujeme, že lokálně instalované programy by měly používat i ostatní podadresáře nadřazeného adresáře /var, např. /var/lock.

/var/lock — Soubory tzv. 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 tzv. log-soubory různých programů. Například program login zaznamenává (do souboru /var/log/wtmp) všechna přihlášení a odhlášení uživatelů systému, program syslog ukládá (do souboru /var/log/messages) všechny hlášky jádra systému a systémových programů. Velikost souborů v adresáři /var/log dost často nekontrolovaně roste. Proto se musí v pravidelných intervalech mazat.

/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 současně přihlášených uživatelích systému.

`/var/spool` Adresáře pro elektronickou poštu, systém „news“, tiskové fronty a další subsystémy, které využívají metodu „spool“ a princip řazení úloh do fronty. Každý z těchto subsystémů má v tomto adresáři svůj vlastní podadresář, např. poštovní schránky uživatelů jsou uložené v podadresáři `/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.)

3.5 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 informací o procesu číslo 1. Každý z procesů má v adresáři `/proc` vlastní podadresář, kterého jméno je stejné, jako identifikační číslo procesu.

`/proc/cpuinfo` Různé informace o procesoru. Například typ, výrobce model, atd.

`/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, i o historii žádostí o využití každého z nich.

`/proc/ioports` Informuje o tom, který z vstupně-výstupních portů se momentálně využívá.

`/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 využívá 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 (swap).
`/proc/modules`	Informuje o tom, které moduly jádra jsou právě zavedeny v paměti.
`/proc/net`	Informace o stavu síťových protokolů.
`/proc/self`	Symbolický link 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 linky. Tímto způsobem se mohou programy pohodlně a jednoduše dostat k vlastnímu adresáři.
`/proc/stat`	Různé statistiky týkající se systému. Např. počet chyb stránkování během zavádění systému a podobné.
`/proc/uptime`	Informuje o tom, jak dlouho systém běží.
`/proc/version`	Verze jádra systému.

Většina výše popsaných souborů má textovou formu. Jsou tedy celkem dobře čitelné pomocí standardních nástrojů. Avšak velmi často jsou formátované takovým způsobem, že informace v nich obsažené jsou pro běžného uživatele na pohled dost těžce „stravitelné“. Proto existuje mnoho příkazů, které – kromě toho, že čtou informace obsažené v souborech systému `/proc`, upravují jejich formát do srozumitelnější podoby. Tak například program `free` čte data ze souboru `/proc/meminfo`, převádí velikost v bajtech na kilobajty a přidá něco málo dalších informací o využití paměti.

Kapitola

4

Používání disků a jiných záznamových médií

Na čistém disku lze hledat navěky.

Z pohledu diskového subsystému vás jako správce čeká skutečně nejvíc práce při instalaci operačního systému Linux, případně instalace jeho vyšší verze. 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 disků formátovaná výrobcem. Formátování po připojení do systému tedy 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, segmenty (angl. partitions). 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 lze vytvořit samostatný souborový systém. Z pohledu operačního systému nestačí disk pouze zapojit, případně rozdělit na samostatné diskové oblasti. Linux může disk používat až poté, co na něm vytvoříte nějaký souborový systém. Pak lze na disk ukládat soubory a přistupovat k nim.

- Vytvoření jednoduché 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 5 obsahuje i informace o virtuální paměti a diskové vyrovnávací paměti. Pracujete-li s disky, měli byste být s jejich principy obeznámeni.

V této kapitole najdete vše, co jako správce systému potřebujete vědět o pevných discích, disketách, jednotkách CD-ROM a páskových jednotkách.

4.1 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 (např. pásky a sériové linky). Znaková zařízení mohou být buď sériová, nebo s náhodným přístupem. Každému z podporovaných zařízení odpovídá v systému souborů jeden nebo několik speciálních souborů. Když se čtou či zapisují data z anebo do speciálního souboru, 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 např. obsluha přerušení nebo nastavování parametrů sériového portu). Když například chcete vytisknout nějaký soubor na tiskárně, stačí zadat

```
$ cat jméno_souboru > /dev/lp1
$
```

Obsah tohoto 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, se pro tiskové úlohy běžně používá zvláštní program (nejčastěji `lpr`). Tento program zajistí, že se v určitém okamžiku bude tisknout pouze jeden soubor. Ihned po ukončení tiskové úlohy automaticky pošle na tiskárnu další ze souborů. Podobný mechanismus přístupu vyžaduje většina zařízení. Takže ve skutečnosti se běžný uživatel o speciální soubory skoro vůbec nemusí zajímat.

Protože se zařízení chovají jako soubory souborového systému (uložené v adresáři `/dev`), lze velmi lehce zjistit, které speciální soubory existují. Lze použít například příkaz `ls` nebo jiný vhodný program. v prvním sloupci výstupu příkazu `ls -l` je uvedený typ takového souboru a jeho přístupová práva. Chcete-li si prohlédnout sériová zařízení systému, zadáte

```
$ ls –l /dev/cua0
crw-rw-rw- 1 root uucp 5, 64 Nov 30 1993 /dev/cua0
$
```

Podle prvního písmene prvního sloupce, tedy písmene „`c`“ v řetězci „`crw-rw-rw-`“, 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).

4.2 Pevné disky

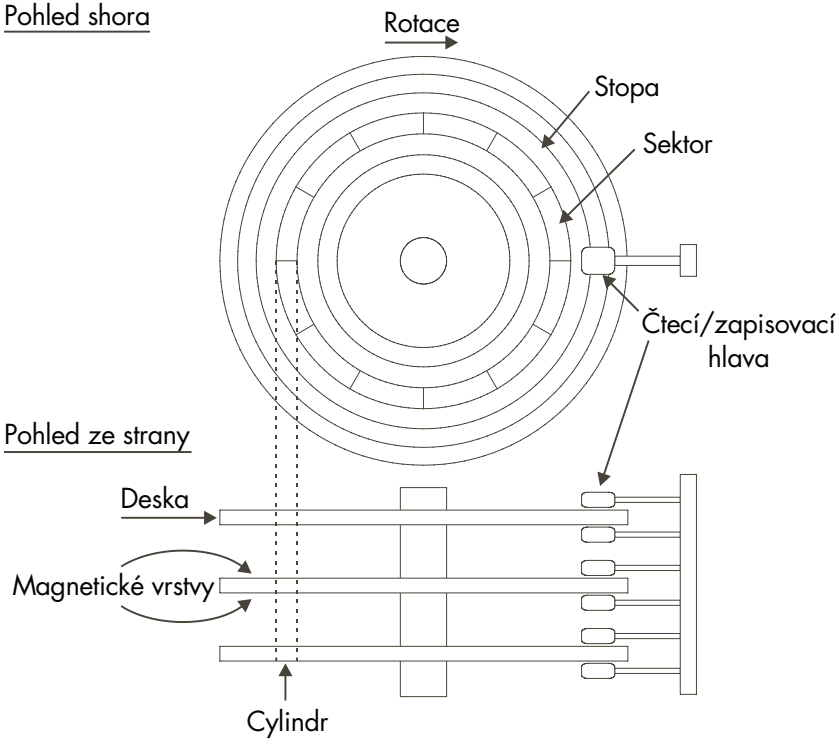
Tento odstavec zavádí terminologii, která se v oblasti pevných disků používá. Znáte-li tyto pojmy a principy, můžete tuto část přeskočit. Na obrázku 4.1 jsou schematicky znázorněny důležité části pevného disku, který se skládá z jedné nebo více kruhových **desek**.[1] Povrch, případně obě **strany** těchto desek, jsou pokryty magnetickou vrstvou, na kterou se zaznamenávají data. Každé takovéto vrstvě odpovídá jedna **čtecí a zapisovací hlava**, která čte nebo zaznamenává údaje. Disky rotují kolem běžné hřídele. Typická rychlost otáčení je 3600 otáček za minutu. Pevné disky s vysokým výkonem používají i vyšší rychlosti. Hlavy se pohybují po obvodu disků. Díky tomuto pohybu spojenému s rotací kruhových desek může hlava přistupovat ke všem částem magnetických povrchů pevných desek disku.

Procesor (CPU) a vybraný 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 mají 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

[1] Tyto desky jsou vyrobeny z pevného materiálu, např. hliníku. Proto se pevným diskům říká „pevné“.

a dělá další nepříjemností, které je 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 dělá některé další operace, obsluhuje například vyrovnávací paměť, automaticky nahrazuje vadné sektory atd.

Výše uvedené obvykle každému postačí k pochopení toho, jak hardware pevného disku funguje. Existuje pochopitelně ještě řada dalších detailů, např. řízení motoru, který otáčí disky a přemísťuje hlavy, elektronika, jež řídí operace dalších mechanických částí atd., které ale většinou nejsou pro pochopení principu práce pevného disku tak důležité.



Obrázek 4.1
Schematické zobrazení pevného disku

Magnetické vrstvy disku jsou obvykle rozděleny do soustředných kružnic, kterým se říká **stopy** (angl. **tracks**). Stopy jsou po obvodu rozděleny 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 sek-

torů (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.

Každá vrstva je rozdělena na stopy a sektory stejným způsobem. Takže když je hlava jedné vrstvy 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 uložené na několika místech, se říká **fragmentované**.

Počet vrstev (resp. 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**. Z paměti CMOS RAM si geometrii disku operační systém načítá vždy během zavádění nebo inicializace ovladače disku.

Na neštěstí má BIOS[2] omezení, které neumožňuje adresovat v CMOS RAM počet stop větší, než 1024. To je pro pevné disky velké kapacity příliš málo. Uvedené omezení lze obejít tak, že řadič pevného disku nebude říkat ostatním prvkům výpočetního systému pravdu o jeho skutečné geometrii a bude **překládat adresy**, které zbytek systému požaduje, do něčeho, co odpovídá realitě. Mějme například pevný disk, jenž má 8 hlav, 2048 stop a 35 sektorů na stopu[3]. Řadič tohoto disku bude zbytku systému „lhát“ a tvrdit, že má 16 hlav, 1024 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.

2 Systém BIOS (Basic Input Output System) je software uložený v paměti ROM. Kromě jiného má na starost úvodní procedury zavádění systému.

3 Tato čísla jsou zcela smyšlená.

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 uspořádanou 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 atd.

Každý pevný disk je v systému reprezentován samostatným speciálním souborem. Nejčastěji budou v systému buď dva, nebo čtyři pevné disky IDE. Zastupují je pak speciální soubory `/dev/hda`, `/dev/hdb`, `/dev/hdc`, případně `/dev/hdd`. Pevné disky SCSI reprezentují speciální soubory `/dev/sda`, `/dev/sdb` atd. Podobné konvence týkající se názvů speciálních souborů platí i pro pevné disky jiných typů. Podrobnější informace uvádí [Anv]. Pamatujte na to, že speciální soubory zastupující pevné disky umožňují přístup k disku jako celku, bez ohledu na jeho diskové oblasti (o těch bude řeč později). 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 zaváděcímu sektoru disku (angl. master boot record, zkráceně MBR), o kterých se také dozvíte víc v dalších částech této kapitoly.

4.3 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 1,44MB disketami. 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` ve jménu speciálního souboru). To jsou běžné 3,5palcové diskety HD (High Density).

Podrobnější informace o konvencích pro pojmenovávání speciálních souborů reprezentujících disketové mechaniky uvádí [Anv].

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 nejdřív naformátovaná. Automatická zařízení zastupují speciální soubory `/dev/fd0`, `/dev/fd1` atd.

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 disketami, jejich příčinou mohla být právě nefunkční indikace výměny média. Jediným způsobem, jak lze tyto problémy odstranit, je nechat disketovou mechaniku opravit.

4.4 Jednotky CD-ROM

Jednotky CD-ROM čtou opticky data z plastických disků. Informace jsou zaznamenány na povrchu těchto disků[4] 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

4 Přesněji, na povrchu *uvnitř* disků – tedy na tenkém kovovém disku uvnitř plastového pláště.

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 kilobajt 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 tzv. „live“ souborové systémy na CD-ROM. Ty fungují tak, že část souborů se při instalaci nekopíruje 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 s ním měly – jako se svým původním – umět bez problémů pracovat 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 extension“. Rock Ridge například umožňuje používat dlouhá jména souborů, symbolické linky 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 extension. 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 programů není určena pro Linux (možná s výjimkou některých programů, jež běží pod `dosemu`, linuxovým emulátorem systému MS-DOS).

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 jed

notlivých způsobů připojení jednotky CD-ROM jdou nad rámec této knihy. Pro vás je podstatné, že podobně jako u jednotek pružných disků určuje způsob připojení vždy jiný speciální soubor. Další podrobnosti, které by mohly více objasnit tuto problematiku, uvádí [Anv].

4.5 Pásky

Magnetopáskové jednotky používají pásky podobné[5] 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 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.

4.6 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 systému MS-DOS se pod pojmem „formátování" rozumí i proces vytváření 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** (angl. **low-level formatting**) a pro vytvoření souborového systému označení **vysokoúrovňové formátování** (angl. **high-level formatting**). Terminologie používaná v unixovém světě označuje obě tyto činnosti tradičními pojmy „formátování" a „vytvoření souborového systému". Nejinak tomu bude i v této knize.

5 Ale jinak, pochopitelně, úplně jiné.

Disky IDE a některé disky SCSI jsou formátovány ve výrobě a není třeba je po připojení formátovat opakovaně. Možná proto se o formátování disků málokdo zajímá. Ale právě nesprávné formátování pevného disku může být skutečnou příčinou toho, že disk nepracuje správně (některé disky vyžadují zvláštní způsob formátování, který pak umožňuje například automatické přemísťování vadných sektorů).

Navíc disky, které je potřeba (a které lze) formátovat často, vyžadují 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.

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 popsaný v dalších částech této kapitoly. Jinou alternativou je vytvoření malé diskové oblasti (partition), 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 (např. `/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 logická chyba (špatná úroveň signálu po zápisu znečištěnou zápisovou hlavou), program `fdformat` si nebude „stěžovat“. Naopak skutečná chyba – fyzicky poškozený sektor, přeruší proces kontroly povrchu diskety a jejího formátování. Jádro systému zapíše hlášení do log-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`. Ten ale neumí disk zformátovat, takže jej lze použít jenom ke kontrole existujících souborových systémů. 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`, jimž 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. Takováto vlastnost diskové jednotky by měla být dokumentovaná v jejím manuálu. V něm byste měli najít podrobnější informace o způsobu jak zjistit, jestli se na disku, který používáte, vyskytují vadné bloky, které řadič disku popsaným způsobem nahradil správnými sektory. Avšak i disky tohoto typu by teoreticky mohli selhat, kdyby počet vadných bloků výrazně vzrostl. Nicméně je pravděpodobnější, že než by se tak stalo, byl by disk již tak opotřebovaný, že by byl prakticky nepoužitelný.

4.7 Diskové oblasti

Pevný disk může být rozdělen na několik **diskových oblastí,** neboli segmentů (angl. **partitions**). Každá oblast se chová tak, jako by byla samostatným diskem. Využití diskových oblastí má význam v případě, že máte jeden pevný disk a chcete na něm používat například dva operační systémy. Disk můžete rozdělit na dva segmenty. Každý operační systém pak 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 segmenty nedělí. 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.

4.7.1 Zaváděcí sektor disku, zaváděcí sektory operačních systémů a tabulka 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). První sektor je tzv. **zaváděcí sektor disku** (angl. **master boot record**, zkráceně 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 rozdělení disku na oblasti (angl. partition table) a zjistí, která oblast disku je aktivní (tedy označená jako zaváděcí). Pak přečte první sektor této oblasti, neboli **zaváděcí sektor** (angl. **boot sector**) této oblasti. (MBR je také zaváděcí sektor, ale má zvláštní postavení, a proto i zvláštní označení.) Zaváděcí sektor diskového segmentu obsahuje další krátký program, který načítá první část operačního systému, jenž je na této diskové oblasti uložený (samozřejmě za předpokladu, že je daná oblast označena jako aktivní – zaváděcí) a spouští jej.

Metoda dělení disku na diskové oblasti není hardwarově implementovaná a není ani součástí systému BIOS. Jde čistě o nepsaná pravidla podporovaná i jinými operačními systémy. Ale ne všechny operační systémy se chovají podle těchto konvencí, jsou i výjimky. Některé operační systémy sice podporují dělení disku na diskové oblasti, ale obsadí jeden diskový segment a v rámci této oblasti používají svou vlastní interní metodu dělení. Avšak i tyto typy operačních systémů mohou bez jakýchkoliv zvláštních prostředků spolupracovat s jinými systémy (včetně operačního systému Linux). Naopak, takový operační systém, jenž rozdělení disku na diskové oblasti nepodporuje, nemůže koexistovat na stejném pevném disku s jiným operačním systémem, který tyto konvence podporuje.

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`.) Některé 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
$
```

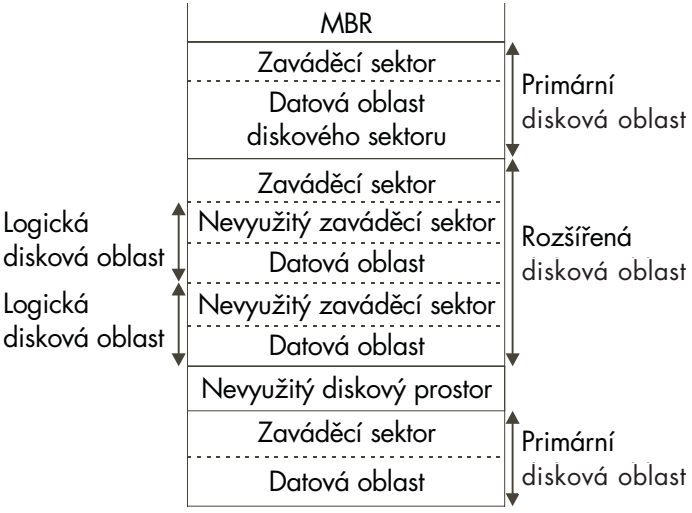
4.7.2 Rozšířené a logické diskové oblasti

Původní schéma dělení disků počítačů PC na diskové oblasti umožňuje vytvořit pouze čtyři diskové segmenty. 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 prostor pro „swap“ na samostatném diskovém segmentu (podrobnosti uvádíme v dalších částech manuálu).

Aby bylo možné omezení počtu diskových oblastí obejít, byly zavedeny tzv. **rozšířené diskové oblasti** (angl. **extended partitions**). Tento trik umožňuje rozdělit **primární diskové oblasti** (angl. **primary partitions**) na podoblasti. Takto rozdělená primární oblast je onou rozšířenou oblastí a její části (podoblasti) jsou tzv. **logické oblasti** (angl. **logical partitions**).

Chovají se jako primární[6], ale jsou vytvořené jiným způsobem. Není mezi nimi ale žádný rozdíl v rychlosti.

Struktura oblastí pevného disku by mohla vypadat například tak, jak je uvedeno na obrázku 4.2. Disk je rozdělen na tři primární diskové oblasti. Druhá z nich je rozdělena na dvě logické diskové oblasti. Část disku nepatří vůbec žádné oblasti. Disk jako celek a každá primární oblast má svůj zaváděcí sektor.



Obrázek 4.2
Příklad rozdělení pevného disku na diskové oblasti

4.7.3 Typy diskových oblastí

Tabulky oblastí disku (jedna v MBR a další na rozšířených diskových oblastech) mají vyhrazený jeden bajt pro každou oblast, jež identifikuje její typ. Typ diskové oblasti určuje operační systém, který daný segment využívá, případně jiný účel, pro který tuto oblast operační systém používá (např. odkládací prostor „swap“). Informační bajt by měl zamezit tomu, aby mohly být v počítačovém systému nainstalovány dva operační systémy, které by náhodně využí-

6 Nelogické?

valy stejnou oblast disku. 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 významnější část tohoto bajtu, kdežto zbylé ne.

Žádný z úřadů pro normalizaci nespecifikoval, co která hodnota bajtu znamená. Některé obecně přijaté hodnoty a odpovídající typy uvádí tabulka 4.1. Stejný seznam vypíše linuxový program `fdisk`.

4.7.4 Dělení pevného disku na diskové oblasti

Je mnoho programů, které umí vytvářet a mazat diskové oblasti. Součástí většiny operačních systémů je nějaký nástroj pro práci s diskovými segmenty. Je vždy lepší používat program, jenž je součástí operačního systému, v prostředí kterého se segmenty pracujete. Pouze v případě, že by se tento program choval nezvykle, se doporučuje použít jiný. 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.

0	Nevyužitá oblast	40	Venix 80286	94	Amoeba BBT
1	DOS (12bit. FAT)	51	Novell?	a5	BSD/386
2	XENIX (oblast „root“)	52	Microport	b7	BSDI fs
3	XENIX (oblast „usr“)	63	GNU HURD	b8	BSDI swap
4	DOS (16bit) < 32M	64	Novell	c7	Syrinx
5	Rozšířená oblast	75	PC/IX	db	CP/M
6	DOS (16bit) > 32M	80	Starší MINIX	e1	DOS (přístupná)
7	OS/2 HPFS	81	Linux/MINIX	e3	DOS (pouze pro čtení)
8	AIX	82	Linux swap	f2	DOS (sekundární)
9	AIX zaváděcí	83	Linux nativní	ff	BBT
a	OS/2 Boot Manager	93	Amoeba		

Tabulka 4.1
Typy diskových oblastí (výstup programu `fdisk` pro Linux)

V případě, že používáte disky IDE, musí být celá zaváděcí disková oblast (tedy segment, na kterém jsou uloženy soubory obrazů jádra) na prvních 1024 cylindrech. To proto, že při zavádění operačního systému (předtím, než systém přechází do chráněného režimu) používá disk systém BIOS a ten neumí pracovat s více než 1024 cylindry. V některých případech je

možné používat zaváděcí diskovou oblast, která leží na prvních 1024 cylindrech, jenom zčásti. To je možné jenom když budou uloženy na prvních 1024 cylindrech všechny soubory, které při inicializaci čte systém BIOS. Vzhledem k tomu, že takovéhoto uspořádání je velmi obtížné dosáhnout, *je lepší jej 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 oblast vašeho systému celá na prvních 1024 cylindrech. Některé nové verze systémů BIOS a disků IDE již umí pracovat i s disky, které mají více než 1024 cylindrů. Máte-li takovýto systém, můžete na tento problém zapomenout. Jestli si v tom nejste zcela jisti, umístněte raději podle doporučení zaváděcí diskovou oblast na prvních 1024 cylindrů.

Každá disková oblast by měla mít sudý počet sektorů, protože souborové systémy Linuxu používají bloky 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 z tohoto segmentu disku uchovat (pro každý případ raději celý disk nebo oblast), pak tento segment smazat, vytvořit nový a obnovit na nové diskové oblasti soubory ze zálohy. Chcete-li zvětšit velikost 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 segmentů. Pro jiné souborové systémy je to zatím nadále nevyhnutné.

4.7.5 Speciální soubory a diskové oblasti

Každá disková oblast a rozšířená disková oblast má svůj vlastní speciální soubor. Podle konvence pro konstrukci názvů speciálních souborů se číslo diskové oblasti připojí za jméno celého disku s tím, že 1–4 budou primární diskové oblasti (podle toho kolik primárních oblastí bylo vytvořeno) a 5–8 jsou logické diskové oblasti (podle toho, na které primární oblasti jsou

vytvořené). 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. Více informací najdete v seznamu zařízení, jenž je uveden v [Anv].

4.8 Souborové systémy

4.8.1 Co jsou souborové systémy?

Souborový systém tvoří metody a struktury dat, pomocí kterých operační systém udržuje záznamy souborů na disku nebo diskové oblasti. Jde tedy o způsob, jakým jsou soubory na disku organizované. Tento termín se ale používá i ve vztahu k diskové oblasti nebo disku, na kterém se ukládají soubory, nebo ve významu typu souborového systém. Takže když někdo řekne „mám dva souborové systémy“, může mít namysli to, že má disk rozdělen na dvě diskové oblasti, nebo to může znamenat, že používá „rozšířený souborový systém“, tedy určitý typ souborového systému.

Rozdíl mezi diskem či diskovou oblastí a souborovým systémem, který je na nich vytvořený, je dost podstatný. Některé programy (mezi nimi – zcela logicky – programy, pomocí kterých se souborové systémy vytváří) pracují přímo se sektory disku nebo diskového segmentu. Jestli na nich byl předtím vytvořený systém souborů, bude po spuštění takovýchto programů zničený nebo vážně poškozený. Většina aplikací ale pracuje se souborovým systémem. Proto je nelze použít na diskové oblasti, na které není vytvořený žádný systém souborů (nebo na oblasti, na které je vytvořený souborový systém nesprávného typu).

Při tím, než bude možné diskovou oblast nebo disk použít, je potřeba je inicializovat – musí se na ně zapsat určité účetní 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**, **i-uzel**, **datový blok**, **adresářový blok** a **nepřímý blok**. Superblok obsahuje informace o souborovém systému jako celku, například jeho velikost (zrovna u této položky závisí přesná hodnota na konkrétním souborovém systému). I-uzel obsahuje všechny informace o souboru, kromě jeho jména. Jméno souboru je uloženo v adresáři, společně s odpovídajícím číslem i-uzlu. Adresářová položka obsahuje jména souborů a čísla i-uzlů, které tyto soubory reprezentují. I-uzel dále obsahuje čísla datových bloků, v nichž jsou uložená data souboru, který daný i-uzel zastupuje. V i-uzlu 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 tzv. nepřímé bloky. 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**“ (angl. **hole**), a to pomocí programu `lseek` (podrobnosti uvádí příslušná manuálová stránka). Vypadá to tak, že souborový systém pouze „předstírá“, že je na konkrétním místě souboru uloženo nula bajtů dat, takže pro toto místo nejsou vyhrazeny žádné sektory 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. (Tato „prázdná místa“ jsou implementována tak, že se jako adresa datového bloku v nepřímém bloku nebo i-uzlu uloží zvláštní hodnota. Tato zvláštní adresa znamená, že pro určitou část souboru není alokován žádný datový blok, tedy že je v tomto souboru „díra“.)

Prázdná místa v souborech lze využít. Na autorově systému bylo jednoduchými prostředky dokázáno, že tímto způsobem lze potencionálně ušetřit asi 4 MB z celkových 200 MB diskového prostoru. A to je na tomto systému poměrně málo programů a vůbec žádné databázové soubory. Uvedené diagnostické nástroje jsou popsané v příloze A.

4.8.2 Široká paleta souborových systémů

Linux podporuje několik typů souborových systémů. Mezi nejdůležitější patří:

minix — Je nejstarší a je považovaný za nejspolehlivější. Má ale několik omezení – chybí časová razítka (angl. time stamps), 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 velmi 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, linky a speciální soubory. To umožňuje používat běžný souborový systém `msdos` jako by byl původním 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 extension" 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.

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. Jestli 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 žádné místo na disku.

4.8.3 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 k 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.

4.8.4 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 tzv. „front end“, tedy 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 uvedené níže. Více informací najdete v manuálových stránkách.

`-t` *typ_fs* Výběr typu souborového systému.

`-c` Hledání vadných bloků a aktualizace jejich seznamu.

`-l` *jméno_souboru* Načtení seznamu vadných bloků ze souboru *jméno_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 argument `-c` programu `mkfs` a název souboru 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
```

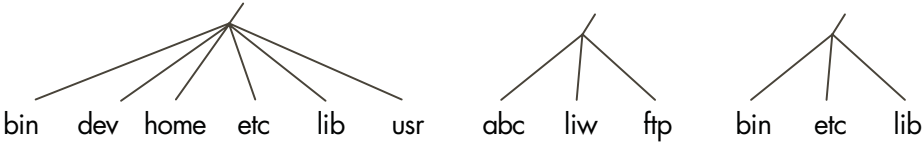
```
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

Volba `-c` programu `mkfs` je sice vhodnější, než použití programu `badblocks`, ale příkaz `badblocks` je vždy nutné použít pro kontrolu vadných bloků 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, až na to, že není nutné je formátovat.

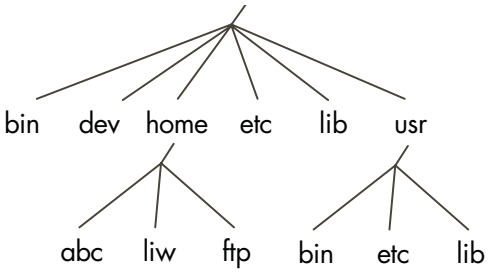
4.8.5 Připojení a odpojení

Souborový systém se musí před použitím **připojit**, namontovat. 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 4.3 jsou například zobrazeny tři samostatné souborové systémy, každý se svým vlastním kořenovým adresářem. Když budou poslední dva souborové systémy připojeny pod adresář `/home` a `/usr` prvního systému souborů, dostaneme jeden adresářový strom, který je vyobrazen na obrázku 4.4.



Obrázek 4.3
Tři samostatné souborové systémy



Obrázek 4.4
Připojené souborové systémy `/home` a `/usr`

Souborový systém lze připojit například zadáním následujících příkazů:

```
$ mount /dev/hda2 /home
$ mount /dev/hda3 /usr
$
```

Příkaz `mount` má dva argumenty. Prvním je speciální soubor 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`. Říká se, že „`/dev/hda2` je připojený na adresář `/home`“ a to samé platí i o adresáři `/usr`. Když si pak chcete prohlédnout některý ze souborových systémů, procházíte strukturou adresářů, ke kterým jsou připojené, jako by to byly jakékoliv jiné adresáře. Je důležité si uvědomit rozdíl mezi speciálním souborem `/dev/hda2` a adresářem `/home`, ke kterému je systém souborů připojen. Speciální soubor umožňuje přístup k „syrovému“ obsahu pevného disku, kdežto prostřednictvím adresáře `/home` se přistupuje k souborům, které jsou na tomto disku uložené. Adresář, ke kterému je souborový systém připojený, se nazývá **přípojný bod** (angl. **mount point**).

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. Soubory, jež byly původně uloženy v adresáři, ke kterému je připojený nový souborový systém, budou nepřístupné, a to až do jeho odpojení. (Soubory, které byly v době připojení otevřené, budou nadále přístupné. Soubory, které jsou nalinkované z jiných adresářů, budou přístupné prostřednictvím těchto původních souborů.) Nehrozí přitom 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 link adresáře `/tmp` do adresáře `/var/tmp`. Předtím, než se při zavádění systému připojí souborový systém `/usr`, je adresář `/var/tmp` v kořenovém souborovém systému. Poté, co se připojí systém souborů `/usr`, bude adresář `/var/tmp` v kořenovém souborovém systému nepřístupný. V případě, že by adresář `/var/tmp` v souborovém systému „root“ 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. Rovněž jádro samotné má zakázáno aktualizovat položku času posledního přístupu v i-uzlech souborů. Připojení systému souborů pouze pro čtení je podmínkou u médií, na která nelze zapisovat, např. u disků CD-ROM.

Pozorný čtenář si již uvědomil drobný logistický problém. Jakým způsobem se připojuje první souborový systém, tedy kořenový svazek (angl. **root filesystem**), někdy označovaný jako **kořenový souborový systém** (obsahuje kořenový adresář), když zjevně nemůže být připojený na jiný souborový systém? Odpověď je jednoduchá – je to „kouzlo“.[7] 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 magicky připojuje jako kořenový, je buď zkompilované v jádře systému, nebo se nastavuje zavaděčem LILO, případně programem `rdev`. Kořenový svazek se pokaždé 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 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* 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 „spláchne“ všechna metadata, která má souborový systém uložená v paměti.

[7] Víc informací najdete ve zdrojovém kódu jádra systému nebo v Průvodci jádrem systému.

V některých systémech se používají i jiné souborové systémy, které lze automaticky připojit při zavádění systému. 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. K dalšímu rozšíření znalostí o připojování souborových systémů při zavádění systému lze doporučit závěr kapitoly, jež pojednává o zavádění operačního systému.

Když už není třeba mít souborový systém připojený, je možno jej odpojit příkazem `umount`[8]. Program `umount` má jediný argument, buďto speciální soubor, nebo bod přístupu. Například odpojení adresářů připojených v předchozím příkladu lze provést příkazy

```
$ umount /dev/hda2
$ umount /usr
$
```

Podrobnější instrukce jak používat příkaz `umount` najdete v manuálové stránce k tomuto programu. Důležitý imperativ, na který je potřeba upozornit – připojenou disketovou jednotku je vždy potřeba odpojit. *Nestačí jenom vytáhnout disketu z mechaniky!* Když systém používá vyrovnávací paměť, v okamžiku odpojení jednotky nemusí být data ze zásobníku paměti „cache“ zapsaná na disketu! Předčasné vytažení diskety z mechaniky by mohlo způsobit poškození jejího obsahu. Když z diskety pouze čtete, není její poškození moc pravděpodobné, ale když zapisujete – navíc náhodně – 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 trojského 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:

- Sdělit uživatelům heslo superuživatele. To je z hlediska bezpečnosti zjevně špatné, avšak to nejjednodušší řešení. Funguje dobře v případě, že se vůbec není potřeba zabývat zabezpečením systému. To se týká řady osobních systémů – tedy systémů, které nejsou připojené do počítačové sítě.

8 Logicky správný tvar je pochopitelně `unmount`, ale písmenko „n“ v sedmdesátých letech záhadně zmizelo a od té doby jej již nikdo víckrát nespatřil. Jestli jej náhodou najdete, vraťte jej laskavě do Bell Labs, NJ.

- 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.[9]
- 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 nevhodné.
- 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): speciální soubor, který se má připojit; adresář, na který se má toto zařízení namontovat (přípojný bod); typ souborového systému; některé další volby; 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 (tj. zakáže připojení příkazem `mount -a`). Argument `user` umožní kterémukoliv uživateli připojit si souborový systém. Z důvodů bezpečnosti zamezí možnosti spouštět programy (jak běžné, tak programy s příznakem setuid) a interpretaci speciálních souborů z připojeného souborového systému. Pak si každý uživatel může namontovat disketovou jednotku se souborovým systémem `msdos` tímto příkazem:

```
$ mount /floppy
$
```

Disketu můžete (samozřejmě pokaždé musíte) odpojit 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`:

9 Uživatelé by nad tím pravděpodobně museli několik sekund přemýšlet.

```
/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 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.

4.8.6 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 automaticky opravoval méně závažné chyby a varoval uživatele na výskyt chyb, které nelze odstranit. Naštěstí je kód, kterým se implementují souborové systémy, poměrně efektivně odladěný, takže problémy vznikají velmi 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.

Jenomže kontrola velkých souborových systémů programu `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 vy-hnout. První z nich: 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 odpojený 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 validace 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 tuto podmínku 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, např. 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 týkající 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.

4.8.7 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.

4.8.8 Boj s fragmentací

Když se soubor ukládá na disk, nemůže být vždy zapsaný do po sobě jdoucích bloků. Soubor, jenž není uložen do sekvenční řady za sebou jdoucích bloků je **fragmentovaný**. Trvá pak déle takovýto fragmentovaný soubor načíst, protože čtecí hlava disku se musí při čtení víc pohybovat. Proto je nežádoucí připustit fragmentaci souborů, i když ta není až tak velkým problémem pro systémy s velkou vyrovnávací pamětí. Pomocí paměti „cache“ lze totiž načítat data napřed.

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, je ukládá tak, aby byly co nejvíce pohromadě. Systém souborů ext2 navíc vždy efektivně alokuje volné sektory, které jsou nejblíže k zbylým blokům ukládaného souboru. Používáte-li proto systémy souborů ext2, nemusíte se fragmentace obávat. Přesto existuje program, jenž umí defragmentovat tento souborový systém – viz v bibliografii uvedený odkaz [TV].

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.

4.8.9 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 podkapitolu 5.6) na disk. Jen zřídkakdy je potřeba zadávat jej ručně, automaticky to totiž dělá démon `update`. Má to význam především v případě nečekaných událostí, například když jsou procesy `update` nebo jejich pomocný proces `bdflush` nečekaně ukončeny, nebo v případě, že musíte ihned vypnout napájení a nemůžete čekat, než se opět spustí program `update`.

4.8.10 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 program „front end“ nezávislý na typu souborového systému, zná souborový systém ext2 některé další nástroje, jež mohou být užitečné při správě systému.

Program `tune2fs` umí upravit parametry souborového systému. Uvedeme některé z těch významnějších:

- Maximální počet připojení programem `mount`. Příkaz `e2fsck` kontroluje, zda nebyl souborový systém připojený vícekrát, než je povoleno, a to i v případě, že je nastavený příznak „clean“. 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, opět i v případě, že je nastavený příznak „clean“ a souborový systém nebyl připojen vícekrát, než je povoleno. Opakované kontroly lze zakázat.

- Počet bloků vyhrazených pro kořenový souborový systém. Souborový systém ext2 rezervuje některé bloky pro kořenový svazek. Když se pak souborový systém jako celek zaplní, není potřeba nic mazat a systém lze v omezené míře spravovat. Rezervovaný počet bloků pro souborový systém „root“ je primárně nastaven na 5 %, což u většiny disků stačí k tomu, aby se zamezilo jejich přeplnění. Nemá smysl rezervovat bloky pro kořenový systém souborů na disketách.

```
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 4.5
Příklad výstupu programu `dumpe2fs`

Více informací najdete v manuálové stránce programu *tune2fs*.

Program `dumpe2fs` vypisuje informace o daném souborovém systému typu ext2, většinou z jeho superbloku. Na obrázku 4.5 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 většinu správců – laiků.

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 defektů na disku, které nelze opravit automaticky programem `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 10.

4.9 Disky bez souborových systémů

Ne na všech discích nebo diskových oblastech se vytváří souborové systémy. Například disková odkládací oblast (swap) 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 „syrové“ 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 nějaké účetní režijní náklady. 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 samozřejmě byly tyto příkazy k ničemu.)

4.10 Přidělování diskového prostoru

4.10.1 Způsoby rozdělování disku na diskové oblasti

Rozdělit disk na 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 souborový systém „root“, jenž obsahuje adresáře `/bin`, `/etc`, `/dev`, `/lib`, `/tmp`, 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 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 (swap). Tím, že se vyčlení vlastní disková oblast (disk) domovským adresářům, v nichž jsou uloženy soubory jednotlivých uživatelů, se zjednoduší zálohování, protože programy, které jsou uložené v adresáři `/usr`, obvykle není potřeba zálohovat. V síťovém prostředí je navíc možné adresář `/usr` sdílet mezi několika počítači (např. 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 uložené 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 umístnit 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. Nicméně je to jediný způsob, jak si ověřit správnost rozdělení disku.

4.10.2 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.[10]

O přidělování odkládacího prostoru pojednává odstavec 5.5.

4.10.3 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 disková 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 klaustrofo-

[10] Sic transit discus mundi.

bie. Odkládacímu prostoru (swap) 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 souborové systémy „root“, `/usr` a domovské adresáře `/home` nemá praktický význam.

Když pak nebude třeba systém MS-DOS, lze změnit rozdělení disku tak, že se vyhradí 12 MB pro odkládací prostor Linuxu a zbytek bude opět jako jeden souborový systém.

Disk o velikosti 330 MB lze rozdělit na několik diskových oblastí následujícím způsobem:

5 MB	kořenový souborový systém
10 MB	oblast pro odkládací prostor (swap)
180 MB	souborový systém `/usr`
120 MB	souborový systém `/home`
15 MB	neobsazená disková oblast

Neobsazenou diskovou oblast lze využívat na různé „experimenty“, při nichž je potřeba mít samostatný diskový segment, např. při testování různých distribucí Linuxu, nebo srovnávání rychlosti souborových systémů a podobně. Jinak lze neobsazenou diskovou oblast rovněž využít jako odkládací prostor (hlavně v případě, že máte rádi hodně otevřených oken).

4.10.4 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.

4.10.5 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 verzi GNU 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` spakují (a rozbalí) jednotlivé soubory, případně celé skupiny souborů. Systém `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. (Systém `DouBle` pracuje na stejném principu jako program Stacker pro operační systém MS-DOS, který možná znáte.)*

* Poznámka korektora: Dalším používaným programem pro komprimaci je `bzip2`, pomocí něhož jsou komprimovány i zdrojové texty jádra operačního systému Linux.

Kapitola

5

Správa paměti

Minnet, jag har tappat mitt minne,
är jag svensk eller finne
kommer inte ihĺg. . .
(Bosse Österberg)

V této kapitole jsou popsány hlavní rysy systému správy paměti operačního systému Linux, tedy subsystémy virtuální paměti a diskové vyrovnávací paměti (angl. disk buffer cache). Kapitola dále popisuje jejich účel, funkce a další podrobnosti, které správce systému musí vzít v úvahu.

5.1 Co je virtuální paměť?

Operační systém Linux podporuje systém **virtuální paměti**, 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 RAM. To vše probíhá z pohledu uživatele zcela transparentně. Programy běžící pod Linuxem si zjišťují pouze velikost dostupné paměti RAM 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** (angl. **swap space**).

Linux může použít pro „swap“ jak normální soubor uložený v souborovém systému, tak zvláštní diskovou oblast. Předností samostatného diskového segmentu je rychlost, výhodou odkládacího souboru je jednodušší možnost změny celkové velikosti odkládacího prostoru. Není přitom totiž potřeba měnit rozdělení celého pevného disku na oblasti, kdy navíc hrozí nutnost kompletní reinstalace systému (při případném nezdaru této operace). Víte-li, jak velký odkládací prostor budete potřebovat, zvolte odkládací prostor na zvláštní oblasti disku. Pokud si nároky na „swap“ 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 pro „swap“ navíc.

Poznámka k terminologii používané v oblasti operačních systémů: v odborných kruzích se obvykle rozlišuje mezi odkládáním (angl. swapping), tedy zapsáním celého procesu do odkládacího prostoru na disku, a stránkováním (angl. 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).[1]

5.2 Vytvoření odkládacího prostoru na disku

Odkládací soubor (swap) je běžný soubor a není pro jádro systému ničím zvláštní. Jediná vlastnost, která má pro jádro význam, je, že odkládací soubor nemá tzv. „prázdná místa“ (angl. holes) a že je připraven pro použití programem `mkswap`. Musí být navíc (z důvodů implementace) uložený na lokálním disku, takže nemůže být uložen v souborovém systému, který je připojen pomocí NFS.

Zmínka o „dírách“ je důležitá. Odkládací soubor si 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 pro odkládací soubor vyhrazeny. Protože „prázdné místo“ v souboru znamená, že tomuto místu souboru nejsou přiděleny žádné diskové sektory, nebylo by pro jádro systému dobré je využívat.

[1] Což pokaždé zcela zbytečně hrozným způsobem rozladí množství počítačových odborníků.

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 uvedená 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 poslední pár kilobajtů nevyužitý.

Disková oblast pro „swap“ rovněž není ničím neobvyklým. Vytvoří se stejně, jako každá jiná disková oblast. Jediným rozdílem je, že se používá jako „syrové“ 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 byli vytvořeny diskový segment pro odkládací prostor nebo odkládací soubor, je potřeba zapsat na jejich začátek odpovídající označení, které obsahuje některé informace významné z hlediska správy systému a informace, jež využívá jádro systému. Provede se to příkazem `mkswap` tímto způsobem:

```
$ mkswap /extra-swap 1024
Setting up swapspace, size = 1044480 bytes
$
```

Je důležité si uvědomit, že odkládací prostor systém zatím nevyužívá. 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 segmenty!* Naštěstí budete tento příkaz potřebovat pouze když instalujete operační systém.

Linuxový manažer paměti omezuje velikost každého z odkládacích prostorů na asi 127 MB (z různých technických důvodů je současný limit (4096-10) x 8 x 4 096=133 890 048 bajtů neboli 127,6875 MB).* Avšak současně můžete využívat až 16 odkládacích prostorů, tedy celkem téměř 2 GB.[2]

* Poznámka korektora: Dnes už existují i záplaty (ale nikoli chybové) do jádra, které umožňují využívat odkládací soubory o velikosti až do 2GB.

2 Gigabajt sem, gigabajt tam, o reálné paměti jsme začali mluvit dost brzo.

5.3 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 argument 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 často 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 (položka `Mem:`) se zobrazuje velikost fyzické paměti. Sloupec `total` neukazuje skutečnou velikost fyzické paměti, kterou využívá jádro systému, ta má obvykle asi jeden megabajt. v sloupci `used` je zobrazeno množství využívané paměti (v druhém řádku chybí údaj o zásobnících vyrovnávací paměti). Sloupec `free` udává celkové množství nevyužité paměti. Sloupec `shared` ukazuje paměť sdílenou několika procesy – platí čím více, tím lépe. v sloupci `buffers` je zobrazena aktuální velikost vyrovnávací diskové paměti.

v posledním řádku (`Swap:`) jsou analogické informace, jež se týkají odkládacích prostorů. 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, něž vyřadíte některý odkládací prostor z činnosti, zkontrolovat (např. příkazem `free`), jestli máte dostatek volné paměti.

Všechny odkládací prostory, které se používají automaticky po zadání příkazu `swapon -a`, lze vyřadit z činnosti příkazem `swapoff -a`. Příkaz vyřadí z činnosti pouze 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.

5.4 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 se tomu trochu více věnovat. V publikaci „Tips – HOWTO“ je uvedeno několik praktických rad, jak systém sdílení odkládacího prostoru realizovat.

5.5 Přidělování odkládacího prostoru

Možná někdy narazíte na radu, že máte přidělovat dvakrát tolik odkládacího prostoru, než máte fyzické paměti. To je pouhá pověra. Uvádíme proto správný postup:

1. Zkuste odhadnout, jaké jsou vaše celkové potřeby paměti, tedy největší množství paměti, které budete pravděpodobně v jednom okamžiku potřebovat. To je dané součtem paměťových nároků všech programů, které budou (pravděpodobně vždy) běžet současně.

   Když například chcete, aby běželo grafické uživatelské rozhraní X Window, měli byste mu přidělit asi 8 MB paměti RAM. 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`.

2. K odhadu podle kroku 1 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.

3. 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 paměť a není třeba odečítat velikost fyzické paměti od celkových nároků na paměťový prostor.)

4. 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 případné odkládání – data lze takto odložit dřív, 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.

5.6 Disková vyrovnávací paměť

Čtení z disku[3] je ve srovnání s přístupem k (reálné) 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. Zkuste si 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, než je nebude potřeba, lze zrychlit především operace čtení z disku. Paměť vyhrazená pro tyto účely, tedy pro ukládání z disku načítaných a na disk zapisovaných dat (angl. **disk buffering**), se nazývá **disková vyrovnávací paměť** (angl. **buffer cache**).

Protože paměť je naneštěstí omezený a navíc „vzácný“ systémový zdroj, nemůže být vyrovnávací paměť obvykle příliš velká (nemohou v ní být totiž uložena úplně všechna data, která by chtěl někdo používat). Když se vyrovnávací paměť zaplní, data, jež se nepoužívala nejdéle, se odloží na disk a takto uvolněná vyrovnávací paměť se využije na ukládání dalších dat.

Vyrovnávací paměť funguje při zápisu na disk. Jednak proto, že data, která se zapisují na disk, se velmi často brzo opakovaně načítají (např. zdrojový kód je ukládán do souboru a pak jej opět načítá kompilátor), takže je výhodné uložit data zapisovaná na disk i do vyrovnávací pa-

3 Pochopitelně s výjimkou disku RAM.

měti. Jednak již pouhým uložením dat do paměti „cache“ a tím, že se nezapisují na disk ihned, 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. Část z nich se označuje jako **vyrovnávací paměti s přímým zápisem** (angl. **write-through**). Zapisují data na disk ihned (data přirozeně zůstanou rovněž uložena ve vyrovnávací paměti). Je-li zápis odložen na pozdější dobu, označují se tyto vyrovnávací paměti jako **paměti s odloženým zápisem** (angl. **write-back cache**). Posledně uvedený systém je zřejmě efektivnější, než prvně jmenovaný, 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 6). 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 kontrolní dioda LED mechaniky pružného disku. Příkaz `sync` **vyprázdní** vyrovnávací paměť, tedy uloží všechna nezapsaná data na disk. Lze jej použít vždy, když si nejste zcela jisti, jestli se obsah vyrovnávací paměti bezpečně uložil na disk. V tradičních systémech Unix existuje program `update`, který běží na pozadí a spouští příkaz `sync` každých 30 sekund. V těchto systémech obvykle není 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 toho obávat. Proces `bdflush` lze spustit ručně (příkazem `/sbin/update`).

Ve skutečnosti se obvykle do vyrovnávací paměti neukládají 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 nepoužitelná. Bude v ní uloženo tak málo dat, že se vyrovnávací paměť vždy vyprázdní předtím, než mohou být data použita. Kritická velikost záleží na tom, jaké množství dat se z disků čte a na disky zapisuje a jak často se k těmto údajům 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 paměť a zapříčinit časté odkládání (jež je navíc velmi 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.

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.

Kapitola

6

Zavádění systému a ukončení jeho běhu

Spusť mě
Ó... nesmíš... nesmíš
Nikdy, nikdy, nikdy nesmíš zastavit
Rozjeď to
Ó... rozjeď to, nikdy, nikdy, nikdy
Nezklameš,
nezklameš
(Rolling Stones)

Tato kapitola popisuje, co se děje po tom, co je systém Linux spuštěn, a poté, co je zastaven. Dále uvádí správný postup procedur zavedení a zastavení systému. Když se tyto postupy nedodrží, mohou se některé soubory poškodit anebo ztratit.

6.1 Zavádění a ukončení práce systému - přehled

Zapnutí počítače, po němž se zavede operační systém[1], se říká **zavedení systému** (angl. **booting**). Tento původní termín 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.

[1] První počítače nestačilo pouze zapnout. Bylo ještě potřeba manuálně zavést operační systém. To až tyhle nově opentlovaná „udělátka“ zařídí všechno sami.

V průběhu zaváděcích sekvencí počítač nejdříve zavede krátký kód, kterému se říká **zavaděč** (angl. **bootstrap loader**), 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 část kódu, kterou počítač zavádí jako první, je velmi krátká (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** (angl. **boot sector**). 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 výpočetního systému 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 níže.

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.

6.2 Proces zavádění systému při pohledu z blízka

Systém Linux lze zavést buď z diskety, z pevného disku nebo i z CD. Příslušná kapitola příručky „Průvodce instalací a začátky“ ([Wel]), jež podrobně pojednává o instalaci operačního systému Linux, uvádí návod jak systém instalovat oběma z výše uvedených způsobů.

Když počítač PC startuje, systém BIOS provádí různé testy a kontroluje, zda je vše v pořádku[2]. Až pak zahájí skutečné zavádění operačního systému. Vybere zaváděcí diskovou jednotku. Typicky první disketovou mechaniku – je-li v mechanice zasunutá disketa, jinak první pevný disk – nejde-li o bezdiskový počítač. Pořadí prohledávání lze konfigurovat. Poté se na-

2 Říká se tomu **test systému při zapnutí** (angl. **power on self test**, zkráceně **POST**).

čte první sektor tohoto disku, kterému se říká **zaváděcí sektor** (angl. **boot sector**). U pevných disků se označuje jako **zaváděcí sektor disku** (angl. **master boot record**), 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), úkolem kterého je načíst z disku operační systém a spustit jej. Pokud jde o zavádění systému z diskety – zaváděcí sektor pružného disku obsahuje 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 obvykle 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 (angl. LInux LOader, zkráceně LILO).

Když se systém zavádí z pevného disku, kód obsažený v zaváděcím sektoru disku si prohlédne tabulku rozdělení disku, která je v tomto sektoru také uložená. Pak zaváděcí kód identifikuje aktivní diskovou oblast (oblast, která je označená jako zaváděcí), načte zaváděcí sektor aktivní oblasti a spustí kód uložený v tomto zaváděcím sektoru. Kód v zaváděcím sektoru diskové oblasti dělá v podstatě to samé, co kód obsažený v zaváděcím sektoru diskety. Načte jádro systému z aktivní 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 diskové oblasti nemůže jednoduše sekvenčně číst data z disku, jako při zavádění systému z diskety. Musí hledat příslušné sektory všude, kam je souborový systém uložil. 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 dál v zaváděcí sekvenci, takže zavaděč LILO načte a zavede implicitně vybrané jádro Linuxu. Je ale možné nakonfigurovat jej tak, aby zavedl některý z více obrazů jádra 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 Ctrl v okamžiku zavádění systému (tj. 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 argumenty příkazové řádky, které se zadávají za jménem jádra nebo operačního systému, který se zavádí.

META: Existují i jiné zavaděče, než LILO. Informace o nich budou přidány v některé z dalších verzí manuálu. `loadlin`.

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 je výrazně pracnější konfigurovat operační systém tak, aby se zaváděl z disku. Proto mnoho uživatelů dává v první fázi přednost zavádění systému z diskety a až pak, když bude nainstalovaný systém fungovat, doinstalují zavaděč LILO a budou Linux zavádět z pevného disku.

Až poté, co se jádro Linuxu některým z výše uvedených způsobů načte do paměti, je skutečně spuštěno. Probíhají zhruba tyto věci:

- Jádro Linuxu se instaluje v komprimovaném tvaru, takže se před samotným zavedením 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 video-mód implicitně nastavit – pak se systém při startu na video-režim 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 atd.) 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, jenž používá autor, se vypisují tyto hlášky:

```
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 data)
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 program `init`, jenž se nachází v adresáři `/sbin`. Program `init` bude vždy procesem číslo 1 a jeho úkolem jsou různé „úklidové práce“ spojené se startem systému. Přesný postup toho, co jádro systému v tomto kroku dělá, závisí na tom, jak je konfigurováno. Více informací uvádí kapitola 7. Jádro systému v této fázi přinejmenším spustí na pozadí některé nezbytné démony.

- Proces `init` pak přejde do víceuživatelského režimu, spustí procesy `getty` pro virtuální konzolu a sériové linky. Proces `getty` je program, který umožňuje uživatelům přihlásit se prostřednictvím virtuální konzoly a 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.

6.3 Podrobněji o zastavení systému

I při zastavování operačního systému Linux je důležité dodržovat správný postup. Jestli se správná procedura zastavení systému nedodrží, budou souborové systémy pravděpodobně „na vyhození“. Obsah jednotlivých souborů se totiž může náhodně promíchat. To proto, že operační systém využívá diskovou vyrovnávací paměť, jež nezapisuje některé změny na disk ihned, ale v určitých časových intervalech. To sice významně zvyšuje výkon systému, ale následkem toho, že se z ničeho nic vypne napájení ve chvíli, kdy paměť „cache“ 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 pouhému „cvrknutí“ do síťového vypínače je to, že v systému se souběžným zpracováním úloh (multitaskingem) 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 poslední z nich jako superuživatel. Jestli jste přihlášeni jako uživatel `root`, je potřeba se vrátit do kořenového adresáře. Vyhnete se tak problémům při odpojení souborových systémů. Pak můžete zadat příkaz `shutdown -h now`. Mezi zadáním příkazu `shutdown` a zastavením systému bude určité časové prodlení, když nahradíte argument `now` znaménkem plus a počtem minut (přece jenom – běžně nejste jediným uživatelem 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 systémového svazku), uživatelské procesy (je-li někdo stále přihlášen) se ukončí, běžící démoni se zastaví, všechny připojené svazky se odpojí, obrazně řečeno – všechno se utiší. Poté proces `init` vypíše zprávu, že lze počítač vypnout. Až pak *a jenom pak* lze šáhnout na síťový vypínač.

v některých případech (zřídkakdy u správně nakonfigurovaného systému) není možné zastavit systém správně. Například když jádro systému zpanikaří, havaruje, hoří a vůbec jinak zlobí, může být zcela nemožné zadat jakýkoliv další příkaz. Pochopitelně, v takovéto situaci je i 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 jenon 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 zásobník vyrovnávací paměti) a potom jednoduše vypnout proud.

Někteří uživatelé rádi používají při zastavení systému příkaz `sync`[3] 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 „clean filesystem“ u souborových systémů ext2fs. Proto se metoda trojího zadání příkazu `sync` *nedoporučuje*.

(Možná vás nenapadá proč zrovna *trojí* zadání příkazu `sync` – v ranných dobách Unixu se všechny příkazy musely „naťukat“ zvlášť, takže bylo obvykle dost času na to, aby se ukončila většina diskových vstupně-výstupních operací.)

6.4 Znovuzavedení systému

Pod znovuzavedením operačního systému (angl. rebooting) se rozumí jeho opakované zavedení, tj. 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. Ale to, co se stane po zmáčknutí kláves Ctrl+Alt+Del lze v systému Linux nastavit při jeho konfiguraci. Na víceuživatelském počítači by například bylo lepší před znovuzavedením systému povolit určité časové prodlení. Naopak systémy, které jsou fyzicky přístupné komukoliv, by bylo lepší nastavit tak, aby se při zmáčknutí kombinace kláves Ctrl+Alt+Del nedělo vůbec nic.

[3] Příkaz `sync` vyprázdní zásobník diskové vyrovnávací paměti.

6.5 Jednouživatelský režim

Příkaz `shutdown` lze použít i při přechodu 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, systém běží v normálním (víceuživatelském) režimu.

6.6 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 navíc fungovat vždy (když samozřejmě funguje hardware). Pro počítače PC je takovou alternativou zavádění systému z diskety.

Většina distribucí operačního systému Linux umožňuje vytvořit tzv. **záchrannou zaváděcí disketu** již při instalaci systému. Doporučujeme to udělat. Avšak některé takovéto záchranné diskety obsahují pouze jádro systému. 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 Linux.

Proto by si správce měl vytvořit vlastní zaváděcí diskety, přizpůsobené konkrétním potřebám. Pokyny jak na to jsou obsaženy v příručce „HOWTO – Bootdisk“ od Grahama Chapmana ([Cha]). Musíte mít přirozeně neustále na paměti, abyste měli záchranné zaváděcí a „rootovské“ diskety stále aktuální.

Disketovou mechaniku, která se využívá pro připojení superuživatelské zaváděcí diskety, nebudete moci použít k čemukoliv 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.

Kapitola

7

Proces init

Uuno on numero yksi

Tato kapitola popisuje proces `init`, jenž je vždy prvním procesem uživatelské úrovně, 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 atd. V této kapitole bude vysvětleno, jak se proces `init` konfiguruje a jak se zavádí různé úrovně běhu systému.

7.1 Proces init přichází první

Proces `init` je jedním z programů, jenž 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` většinou „zavadíte“ jenom pokud připojujete nové sériové terminály, modemy pro příchozí volání, tzv. „dial-in“ (nikoliv tedy modemy, pomocí kterých se budete připojovat do jiných systému, tzv. „dial-out“) a když potřebujete změnit implicitní úroveň běhu systému.

Když se zavede jádro systému (načte se do paměti, spustí se, inicializuje ovladače zařízení, datové struktury atd.), ukončí svou roli v proceduře zavádění operačního systému tím, že spustí první program uživatelské úrovně – proces `init`. Takže program `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 umístnění v systému Linux je `/sbin/init`. Nenajde-li jádro program `init`, pokusí se spustit program `/bin/sh` a pokud neuspěje, skončí neúspěšně i celá procedura startu systému.

Když proces `init` startuje, ukončí se proces zavedení systému tím, že se provede několik administrativních úkolů, například kontrola souborových systémů, „úklid“ v adresáři `/tmp`, start obsluhy různých služeb a spuštění procesu `getty` pro každý terminál nebo virtuální konzolu, prostřednictvím nichž se uživatelé mohou přihlašovat do systému atd. (viz kapitola 8).

Po správném zavedení systému proces `init` po každém odhlášení uživatele restartuje procesy `getty` pro příslušný terminál. Umožní tím případné další přihlášení jiných uživatelů. Program `init` si také osvojuje všechny „osiřelé“ procesy. Když některý z procesů spustí další proces (svého potomka) a později ukončí svou činnost dřív, než některý z potomků, sirotci se okamžitě stanou potomky procesu `init`. Adopce sirotků má význam především z různých technických důvodů, je ale dobré o tom vědět, protože je pak snadnější pochopit význam položek seznamu procesů a grafů hierarchického stromu běžících procesů.[1]

Správce systému má na výběr několik verzí programu `init`. Většina distribucí operačního systému Linux používá program `sysvinit` (autorem programu je Miquel van Smoorenburg), jehož předlohou je program `init` pro Unix System V. Unix verze BSD používá odlišný program `init`. Primárním rozdílem mezi uvedenými verzemi programů jsou úrovně běhu systému. Unix System V je implementuje, kdežto verze BSD nikoliv (alespoň pokud jde o jejich tradiční verze). Tento rozdíl není podstatný, a tak se podíváme pouze na program `sysvinit`.

7.2 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[2]. Tato vlastnost umožňuje měnit konfiguraci programu `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 řádku programu `getty`. Jednotlivé řádky souboru `/etc/inittab` sestávají ze čtyř polí oddělených dvojtečkou:

1 Proces `init` samotný nelze ukončit. Nelze jej „zabít“ ani odesláním signálu SIGKILL.

2 Například zadáním příkazu `kill -HUP 1` jako uživatel „`root`“.

```
id:úrovně_běhu:akce:proces
```

Uvedené položky budou popsán níže. Soubor `/etc/inittab` může kromě řádkových záznamů obsahovat i prázdné řádky a ty, jež začínají znakem „#“. Ty program `init` ignoruje.

id — První položka identifikuje každý z řádků konfiguračního souboru. Řádky procesů `getty` blíže specifikují terminál, který daný proces obsluhuje (rozlišuje se písmeny následujícími příslušný název 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, např. `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 vykonat.

Chcete-li spustit programu `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.[3]

Když budete potřebovat přidat do systému další terminály nebo modemové linky pro příchozí volání (dial-in), měli byste rozšířit soubor `/etc/inittab` o další řádky, vždy jeden pro každý terminál, resp. modemovou linku. Více informací najdete v manuálových stránkách *init*(8), *inittab*(5) a *getty*(8).

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 tento proces spustí, ten se neprovede úspěšně a ukončí se,

[3] Různé verze programu `getty` se chovají různě. Ověřte si vlastnosti programu `getty` v jeho manuálové stránce, ale nezapomeňte se také ujistit, že si čtete v správné manuálové stránce odpovídající verzi programu.

`init` jej opakovaně spustí, proces se ukončí, `init` jej spustí, proces se ukončí a tak dál, až do nekonečna. 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, `init` před dalším pokusem o provedení příkazu vyčká pět minut.

7.3 Úrovně běhu systému

Úrovní běhu systému se rozumí určitý stav procesu `init` i celého systému. Tento stav určuje, které ze služeb se nabízí. Jednotlivé úrovně se rozlišují čísly, uvedenými v tabulce 7.1. 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 atd. 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 zvláštní úkoly, spojené s administrací systému).
2–5	Běžný provoz (uživatelsky definovaný).
6	Znovuzavedení systému.

Tabulka 7.1
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 návěstí, druhé pole znamená, že se tento záznam (řádek) uplatní při úrovni 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` 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 sadu služeb, které zatím neběží, a pozastaví obsluhu těch, 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, obecnou 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 argument příkazové řádky `single` nebo `emergency`.[4] Tím si v podstatě vyberete jednouživatelský režim (1. úroveň běhu systému), jenž je popsán v odstavci 7.5.

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.

7.4 Zvláštní konfigurace v souboru /etc/inittab

Soubor `/etc/inittab` má některé zvláštní rysy, jež umožňují procesu `init` reagovat i na některé zvláštní situace. Tyto speciální vlastnosti procesu `init` určují 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` — Umožňuje procesu `init` znovu zavést systém, když uživatel současně zmáčkne klávesy Ctrl+Alt+Del. 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.[5]

`sysinit` — Příkaz, jenž se provede při zavádění systému. Tímto způsobem se například obvykle mažou soubory v adresáři `/tmp`.

4 Parametry příkazové řádky jádra systému lze zadat například pomocí zavaděče LILO. Viz odstavec 7.5.

5 Nebo se spustí program `nethack`.

Výše uvedený seznam klíčových slov není úplný. Další možnosti i podrobnosti týkající se těch, o kterých se zmiňujeme, uvádí manuálová stránka souboru *inittab*(5).

7.5 Zavádění systému v jednouživatelském režimu

Důležitou úrovní běhu systému je tzv. 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.[6] 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`. Program `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 nejdřív.

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 správce systému připraven i na tuto situaci.

Správně nakonfigurovaný systém se před spuštěním shellu v jednouživatelském režimu zeptá na přístupové heslo superuživatele. Je to důležité bezpečnostní opatření, protože jinak by bylo možné jednoduše zadat vhodný parametr příkazové řádky zaváděči systému LILO a dostat se tak k účtu a 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.)

6 Pravděpodobně jej nebude možno použít k hraní hry `nethack`.

8

Přihlašování do systému a ukončování sezení

Tato kapitola by potřebovala citát. Má někdo nějaký návrh?

Tato část knihy popisuje, co se v systému děje poté, co se do něj uživatel přihlásí a zahájí sezení a následně se ze systému odhlásí. Dále budou detailněji popsán různé interakce některých procesů běžících na pozadí, při zahajování a ukončování sezení používané „log"-soubory, konfigurační soubory a další.

8.1 Přihlašování přes terminály

Na obrázku 8.1 je graficky zobrazený 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`.

Je důležité si uvědomit, že jediným novým procesem je ten, jenž vytvoří program `init` (použitím systémového volání `fork`). Procesy `getty` a `login` nahrazují právě tento nový proces (použitím volání systému `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 nečinnosti opět aktivní. Program `getty` rovněž přizpůsobuje přenosové rychlosti a další nastavení pro konkrétní spojení. Takovéto změny parametrů připojení jsou obzvlášť důležité v případě, že systém odpovídá na příchozí modemové žádosti o připojení. V tomto případě se totiž přenosové parametry běžně mění případ od případu.

V současnosti se používá několik různých verzí programů `getty` a `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 `init` pro vás budou i nadále důležité.

8.2 Přihlášení prostřednictvím sítě

Dva počítače, které jsou zapojené 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“. Proto, že 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 sdílelno 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 „Průvodce správce sítě systému Linux“.

8.3 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 „log“-souboru (pomocí programu `syslog`). Rovněž zaznamenává úspěšné i neúspěšné pokusy o přihlášení superuživatele. Oba druhy záznamů jsou užitečné při pátrání po případných „vetřelcích“.

Současně 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), uvedené 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`.[1] 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.

8.4 X a xdm

META: X implementují přihlášení prostřednictvím procesu `xdm` a rovněž `xterm -ls`.

8.5 Řízení přístupu

Databáze uživatelů je tradičně uložena v souboru `/etc/passwd`. Některé systémy používají tzv. **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, jenž se implicitně spouští po přihlášení atd. 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 hesla rozšifrovat 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 souboru, 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.

[1] Dobré distribuce operačního systému Linux to zajistí automaticky.

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.

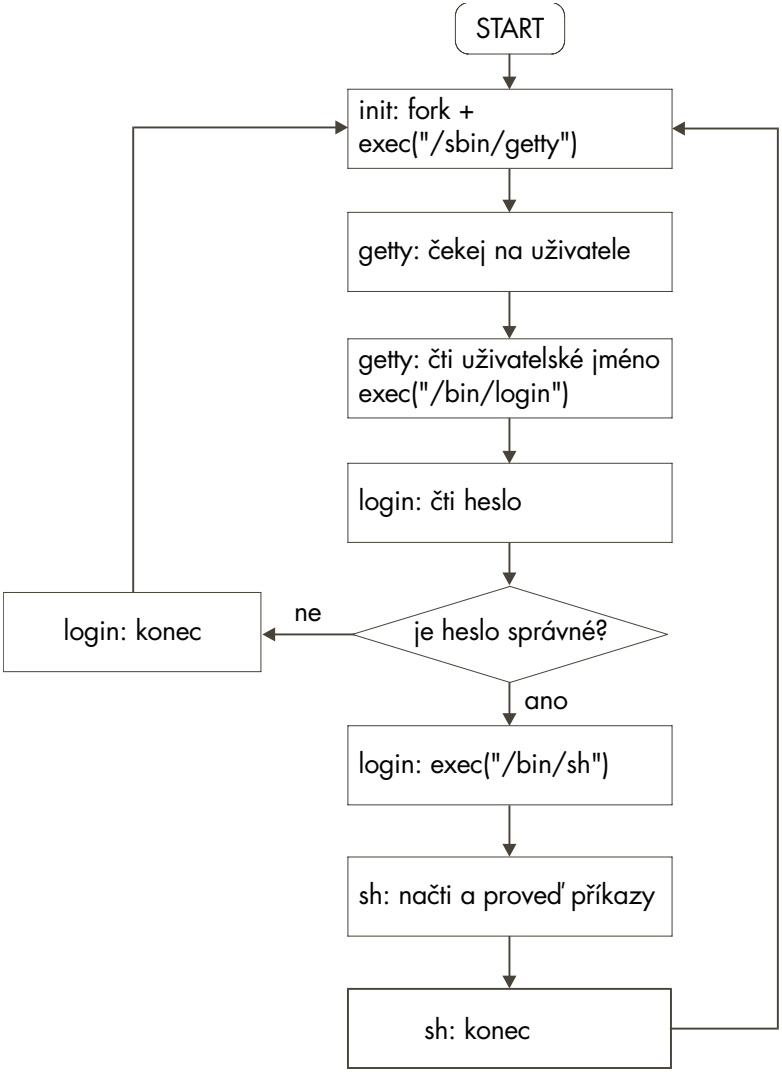
Databáze skupin uživatelů je uložena v souboru `/etc/group`, u systémů se stínovými hesly v souboru `/etc/shadow.group`.

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`.

8.6 Spouštění interpretu příkazů

Při startu každý interaktivní příkazový interpret (shell) automaticky spouští jeden či více předem určených souborů. Různé interprety spouští různé konfigurační soubory. Podrobnější informace najdete v dokumentaci k jednotlivým typům interpretů.

Většina shellů 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 upravil si předem nastavené běžné prostředí podle svého vlastního vkusu.



Obrázek 8.1
Připojení přes terminály: vztahy mezi procesy `init`, `getty`, `login` a interpretem příkazů

Kapitola

9

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.

9.1 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 pouze jedna osoba, což se týká převážné většiny mikropočítačů.[1] 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 výpočetním 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.

[1] Dost by mi například vadilo, kdyby si má sestra četla v mé milostné korespondenci.

9.2 Vytváření uživatelských účtů

Samotné jádro systému Linux považuje uživatele systému za pouhé číslo. Každého uživatele lze totiž identifikovat podle jednoznačného celého čísla, tzv. **identifikačního čísla uživatele** (angl. **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 a přidělená **uživatelská jména** se udržují ve zvláštní databázi mimo vlastní jádro. Tato databáze obsahuje též další informace o uživatelích systému.

Když potřebujete vytvořit nový uživatelský účet, musíte přidat informace o novém uživateli do 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 utility 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 9.2.4.

9.2.1 Soubor /etc/passwd a další informační soubory

Základní databází uživatelů v systému Unix je textový soubor `/etc/passwd` (angl. **password file**), v němž jsou uvedeny 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:

1. Uživatelské jméno.
2. Heslo v zakódované podobě.
3. Identifikační číslo uživatele.
4. Identifikační číslo pracovní skupiny (angl. group ID, zkráceně GID).
5. Skutečné jméno uživatele, případně popis účtu.
6. Domovský adresář.
7. Příkazový interpret (nebo program), který se spustí po přihlášení.

Formát jednotlivých políček je podrobněji popsán v manuálové stránce *passwd*(5).

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 (např. krátké slovo, jenž lze najít v nějakém slovníku, jméno nebo příjmení uživatele atd.). Proto z hlediska bezpečnosti není dobré mít hesla uložená přímo v souboru `/etc/passwd`.

Řada systémů Linux používá systém tzv. **stínových hesel** (angl. **shadow passwords**). Jde o alternativní způsob uložení uživatelských přístupových hesel, jež se zašifrovaná ukládají do jiného souboru (`/etc/shadow`), jenž může číst pouze superuživatel. Soubor `/etc/passwd` pak obsahuje v druhém poli pouze speciální znak. Program, který si potřebuje ověřit totožnost uživatele a má propůjčená přístupová práva vlastníka souboru (pomocí volání jádra setuid), může soubor `/etc/shadow` číst. Běžné programy, které používají pouze některé z dalších položek souboru `/etc/passwd`, přístup k heslům uživatelů systému nemají.[2]

9.2.2 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 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 (viz [Kir]).

META: možná na nesprávném místě? Měli byste se také vyvarovat opakovanému přidělování stejných uživatelských čísel (i textových uživatelských jmen), protože nový vlastník UID (případně uživatelského jména) by tak měl přístup k souborům bývalého vlastníka, k jeho elektronické poště a dalším informacím.

[2] Ano, to znamená, že soubor s uživatelskými hesly `/etc/passwd` obsahuje všechny informace o uživatelích, kromě jejich hesel. Překvapení, která přináší vývoj.

9.2.3 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 si 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.

9.2.4 Manuální vytváření uživatelských účtů

Nový uživatelský účet lze vytvořit ručně tímto postupem:

1. Upraví se soubor `/etc/passwd`, například příkazem `vipw`(8), tak, že se do souboru hesel přidá další řádek nového uživatelského účtu. Je nutné dodržovat správnou syntaxi. *Není vhodné upravovat tento soubor přímo běžným editorem!* Program `vipw` soubor `/etc/passwd` uzamkne, takže se ostatní programy nebudou pokoušet jej změnit. Do pole pro heslo se vloží znak „*“, takže zatím nebude možné se prostřednictvím tohoto účtu do systému přihlásit.

2. Jestli je potřeba vytvořit i novou pracovní skupinu, upraví se podobným způsobem i soubor `/etc/group`, a sice programem `vigr`.

3. Příkazem `mkdir` se pro nového uživatele vytvoří domovský adresář.

4. Do nově vytvořeného domovského adresáře se nakopírují konfigurační soubory z adresáře `/etc/skel`.

5. Příkazy `chown` a `chmod` se upraví jejich vlastnická 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= .
```

6. Zadejte heslo programem `passwd`(1).

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[3]. 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 vskutku lepší zřizovat takovéto anonymní účty bez hesla. Jinak by k nim měl přístup pouze uživatel, který by předtím musel získal oprávnění superuživatele, protože pouze uživatel `root` má přístup k ostatním uživatelským účtům.

9.3 Změny vlastností uživatelských účtů

Je několik příkazů, kterými lze měnit různé vlastnosti uživatelských účtu (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.

3 Surreální uživatelé?

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` (opakovaně 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 „přezdívku“, tedy alias.[4]

9.4 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
```

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 (viz odstavec 2.3.8), 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.

[4] Uživatelské jméno se může změnit například z důvodů sňatku. Uživatelka by mohla chtít používat uživatelské jméno, které odpovídá jejímu novému příjmení.

9.5 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 neznámy „hacker“ prolomil přístupové heslo uživatele některého účtu.

Nejvhodnějším způsobem jak zamezit přístup k podezřelému účtu, je zaměnit nastavený příkazový interpret zvláštním programem, který na terminál vypíše určitou hlášku. 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.[5]

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 - tester
Tento účet byl z důvodů porušení bezpečnostních opatření zablokován.
Volejte prosím číslo 555-1234 a vyčkejte příjezdu mužů v černém.
#
```

5 Máte-li pověst zlomyslného BOFH, tak vám možná budou připadat celkem legrační.*

* Poznámka překladatele: Za zkratkou BOFH (The Bastard Operator From Hell) se skrývá imaginární postava zlotřilého, vychytralého a zlomyslného správce systému (pohybujícího se obvykle v prostředí univerzitních počítačových učeben plných usilovně pracujících studentů – jinak počítačově nevelmi zdatných, avšak o to aktivnějších laiků), který se náramně baví tím, že si ze svých všetečných uživatelů, kteří jej neustále (a velmi často bezdůvodně) otravují se svými problémy nebo pseudoproblémy, na oplátku (ale velmi často i bezdůvodně) dělá legraci, tahá je za nos a „školí“ je kanadskými žertíky nejhrubšího zrna. Sarkastické příběhy si pochopitelně vymýšlí a v elektronických magazínech na Internetu publikují sami operátoři (správci systémů).

Pomocí příkazu `su` ve výše uvedeném příkladu se pochopitelně pouze testuje, zda je změna původně nastaveného interpretu funkční.

Takovéto skripty pro `tail` by měly být uloženy ve zvláštním adresáři, aby jejich jména nekolidovala s příkazy jednotlivých uživatelů.

Kapitola

10

Zálohování

Hardware je indeterministicky spolehlivý.
Software je deterministicky nespolehlivý.
Lidé jsou indeterministicky nespolehliví.
Příroda je deterministicky spolehlivá.

Tato kapitola vysvětluje proč, jak a kdy zálohovat a jak ze záloh obnovit data.

10.1 O důležitosti zálohování

Vaše data mají určitou cenu. Jejich cena je daná hodnotou vašeho času a cenou úsilí data v případě ztráty znovu vytvořit. To vše lze vyjádřit v penězích, nebo přinejmenším vyvážit zármutkem a slzami. Někdy již totiž ztracené informace nelze obnovit, například když je jejich ztráta důsledkem různých 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.[1]

1 Pátým důvodem je „něco dalšího“.

I když je v současnosti hardware relativně spolehlivý, ještě stále se může zdánlivě spontánně 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.

Vývoj u programů už vůbec nesměřuje ke spolehlivosti. 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 může selhat i zálohování. Součástí správného zálohování je i to, že se kontroluje, jestli vůbec funguje. Jistě byste si nechtěli jednoho dne „všimnout“, že se zálohy nedělaly správně.[2]

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.[3] Může se také stát, že si v momentě, kdy se pokoušíte obnovit data ze záloh, uvědomíte, že jste zapomněli zálohovat něco důležitého, například databázi uživatelů systému, která může mít třeba 15 000 položek. To „nejlepší“ nakonec – všechny zálohovací dávky mohou fungovat bezchybně, ale v poslední známé páskové jednotce, jež ještě umí přečíst typ pásky, který používáte, je vědro vody. Když totiž dojde až na zálohy, je paranoia v popisu práce.

10.2 Výběr média pro zálohování

Co se týče zálohování, je nejdůležitějším rozhodnutím výběr médií. Musíte zvážit náklady, rychlost, dostupnost a použitelnost.

2 Nesmějte se. Několika lidem se to stalo.

3 Radši u toho nebýt...

Cena je poměrně důležitá, protože byste měli mít raději několikrát větší kapacitu zálohovacích médií, než ve skutečnosti potřebujete. 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 z hlediska zálohování. Například pevný disk je typicky velmi spolehlivé médium. Nelze ale prohlásit totéž, když hodnotíme spolehlivost z hlediska zálohování a když je tento disk v tom samém počítači jako disk, který zálohujeme.

Jestli lze data zálohovat tak, že nedochází ke kolizím s jinými programy, není rychlost většinou příliš důležitá. 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 velmi rychlé, velmi dostupné, ale ne příliš použitelné v případě velkého množství dat. Magnetické pásky jsou levné i dražší, 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).

10.3 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 povětšině stejné. Oba programy umí ukládat soubory na pásky a obnovovat je, oba jsou schopné využívat téměř všechna média, protože díky ovladačům zařízení jádra systému, které mají na starost obsluhu nízkoúrovňových zařízení, se takováto zařízení chovají vůči programům uživatelské úrovně stejně. Některé unixové verze programů `tar` a `cpio` mohou mít problémy s neobvyklými soubory (symbolickými linky, 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 jeho prostřednictvím.

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 razítko (angl. time stamp). 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 čtecí hlavy disku. 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 tzv. úrovní zálohování (angl. backup levels), 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.

10.4 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** (angl. **full backup**) a v dalších krocích se tvoří tzv. **inkrementální,** neboli **přírůstkové zálohy** (angl. **incremental backups**). 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 přírůstkových záloh bývá pochopitelně mnohokrát pracnější, než obnovování z úplných záloh. Nicméně, ob-

novování dat z přírůstkových záloh lze optimalizovat tak, že vždy zálohujete všechny změny od poslední úplné zálohy. Takto je sice o něco pracnější proces zálohování, ale pak by nemělo být nikdy potřeba obnovovat víc, než některou úplnou a 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 další přírůstkové 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, např. 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.

10.4.1 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 jeho 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, byste měli 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/
....
#
```

Je-li kontrola záloh neúspěšná, nebudete moci v případě potřeby původní data z takovýchto záloh obnovit.

Přírůstkové zálohování dělá programem `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 i-uzlu souboru, například změny bitu vyhrazeného pro přístupová práva nebo změny jména souboru. Ke změnám v i-uzlech se lze dopracovat pomocí programu `find` a pak srovnávat stav dat uložených v souborovém systému se seznamem souborů, které byly posledně zálohovány. Skripty a programy, které implementují tento způsob zálohování, najdete na linuxových uzlech FTP.

10.4.2 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 databázové techniky, jež využívají náhodný přístup.

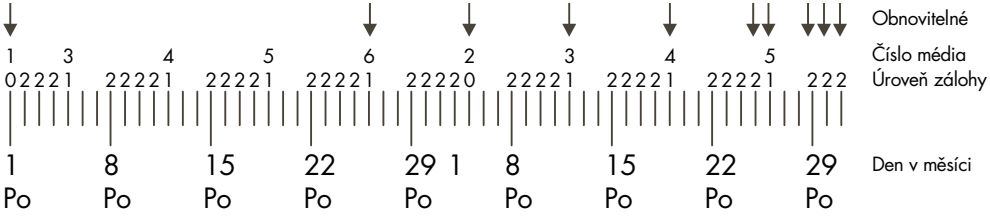
Program `tar` neumí správně zacházet se smazanými soubory. Potřebujete-li obnovit souborový systém z úplné a přírůstkové zálohy a mezi těmito zálohami jste některý ze souborů smazali, po obnovení dat ze záloh bude smazaný soubor znovu existovat. To může být dost závažný problém například v případě, že soubor obsahuje citlivá data, která by již neměla být k dispozici.

10.5 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 přírůstkové zálohování. Ty lze zobecnit do libovolného počtu dalších úrovní. Úplná záloha by mohla být úrovní 0 a další různé přírůstkové zálohy úrovněmi 1, 2, 3 atd. Na každé úrovni přírůstkového 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.



Obrázek 10.1
Příklad časového rozvrhu víceúrovňového zálohování

Víceúrovňovým zálohováním lze historii vytváření archivů prodloužit levněji. Když si například koupíte deset kazet, můžete 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 jste 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 nemůžete obnovit všechny verze zálohovaných souborů, avšak to, co obnovit lze, obvykle postačí.

Z obrázku 10.1 je patrné, která úroveň zálohování se používá v ten který den a z kterých záloh je možno na konci měsíce data obnovit.

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 přírůstkové zálohy úrovně menšího rozsahu. Ale i tak se čas zálohování zvýší, 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 10.2. Zkuste použít posloupnost úrovní zálohování: 3, 2, 5, 4, 7, 6, 9, 8, 9 atd. Snížíte tím čas zálohování i obnovování dat. Zálohovat byste měli vždy maximálně 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 (dny)	Záloha (pásky)	Obnovování
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, 7
7	6	2	1, 2, 5, 7, 8
8	9	1	1, 2, 5, 7, 8
9	8	2	1, 2, 5, 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, ...

Obrázek 10.2
Efektivnější postup zálohování s větším počtem úrovní

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.

Program `dump` má zabudovanou podporu víceúrovňového zálohování. U příkazů `tar` a `cpio` je potřeba víceúrovňové zálohování implementovat pomocí skriptů.

10.6 Co zálohovat

Je pochopitelné, že budete chtít zálohovat vše, co se vám na zálohovací média vejde. Významnou výjimkou je software, jenž lze bezbolestně obnovit z instalačních médií.[4] 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 soubor `/proc/kcore`, protože je to pouze aktuální obraz fyzické paměti, takže je dost velký.

Jakousi „šedou zónou“ jsou zprávy „news“, poštovní schránky, log-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 tabulky, které jsou často rozeseté po celém souborovém systému).

10.7 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

[4] Musíte se rozhodnout, co je jednodušší. Jsou lidé, kteří tvrdí, že je jednodušší instalovat programy z tuctů instalačních disket.

komprimovaná 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 zálohou. 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 např. program `afio` (varianta programu `cpio`).

Komprese o něco prodlouží proces zálohování. Může to způsobit, že zálohovací program nebude schopen zapisovat na pásku dostatečně rychle.[5] 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 mohli setkat u velmi pomalých počítačů.

5 Nepřenáší-li se na páskovou mechaniku data dostatečně rychle, musí se zastavit. To zálohování ještě víc zpomaluje a není to dobré ani pro pásku a ani pro páskovou mechaniku.

Kapitola

11

Udržování správného času

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 porozumět jeho principům.

11.1 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 se Země otáčí,[1] je poledne na různých místech zeměkoule v jinou dobu. Z toho vychází představa **lokálního**, **místního času**. Lidstvo měří čas v různých jednotkách, jejichž většina je rovněž svázána s přírodními jevy, jako je ona kulminace Slunce v poledne. Pokud se nacházíte 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** (angl. **universal time**, zkráceně UT nebo

[1] Dle výsledků posledních výzkumů.

UTC), dříve označovaný jako greenwichský čas (angl. Greenwich Mean Time nebo GMT), protože je místním časem ve městě Greenwich 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** (angl. **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 posunutý a mění pravidla z roka na rok. To je důvod proč nemohou být posuny časových pásem triviální.

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 východoevropském 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 upgradovat jenom tomu odpovídající část subsystému pro určování času. Stačí pak nastavit časové pásmo systému a upravit datové soubory zvoleného pásma, není potřeba se trápit s nastavováním systémového času.

11.2 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 seřizovat jako jednu z položek na obrazovce pro nastavení systému BIOS, nebo z běžícího 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 nastavený univerzální čas, protože pak není potřeba měnit nastavení hardwarových systémových hodin na začátku a konci období letního času (UTC nemá DST). 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 jeho nastavení při přechodu z a na letní čas (jinak by softwarové hodiny neukazovaly místní čas).

11.3. Zobrazení a nastavování času

v systému Debian je systémové časové pásmo určeno symbolickým linkem `/etc/localtime`. Tento link odkazuje na datový soubor pro dané časové pásmo, jež 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*(3).

Příkaz `date` ukáže aktuální datum a čas.[2] Například:

```
$ date
Sun Jul 14 21:53:41 EET DST 1996
$
```

[2] Pozor na příkaz `time`, jenž neukazuje aktuální čas.

To znamená, 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` můžeme rovněž zjistit univerzální čas:

```
$ 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“ dopředu nebo posuny vzad jsou nebezpečnější, než menší změny a posuny dopředu.

11.4 Co když jdou hodiny špatně

Softwarové hodiny systému Linux nejdou vždy přesně. V chodu je udržují pravidelná **přerušení časovače** generované 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. Jestli 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 obrazovku pro nastavení systému BIOS 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 nastavené 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.

A

Zjišťování „prázdných míst“ v souborech

v příloze A je uvedena zajímavá část programu, jenž se používá k zjišťování potenciálu „děr“ v souborovém systému. Originální distribuce této knihy obsahuje celý zdrojový kód (`sag/measure-holes/measure-holes.c`).

```
int process(FILE *f, char *filename) {
static char *buf = NULL;
static long prev_block_size = -1;
long zeroes;
char *p;

if (buf == NULL || prev_block_size != block_size) {
free(buf);
buf = xmalloc(block_size + 1);
buf[block_size] = 1;
prev_block_size = block_size;
}
zeroes = 0;
while (fread(buf, block_size, 1, f) == 1) {
for (p = buf; *p == '\0'; )
++p;
if (p == buf+block_size)
zeroes += block_size;
}
if (zeroes > 0)
```

```
printf("%ld %s\n", zeroes, filename);
if (ferror(f)) {
errormsg(0, -1, "read failed for '%s'", filename);
return -1;
}
return 0;
}
```

B

Slovníček

Knihovník Unseenské univerzity se rozhodl sám napomoci obecnému porozumění tak, že napíše orangutansko-lidský slovník.
Pracoval na něm celé tři měsíce.
Nebylo to lehké. Musel se dostat až k slůvku „oook“.
(Terry Pratchett, „Muži ve zbrani“)

Zde je stručný seznam slovních definicí pojmů, spojovaných s operačním systémem Linux, zvláště pak se správou systému.

ambice Sepsání několika zábavných vět v naději, že se dostanou mezi linuxové soubory „cookie“.

aplikační program Software, který dělá něco užitečného. Výsledek používání aplikačního programu je v podstatě důvodem zakoupení počítače. Viz také systémový program, operační systém.

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.

jádro systému Část operačního systému, která implementuje interakce s technickými prostředky výpočetního systému a sdílení zdrojů. Viz také systémový program.

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.

slovníček Seznam slov a vysvětlení jejich významu. Nezaměňovat se slovníkem, jenž je rovněž seznamem slov a jejich vysvětlení.

souborový systém Metoda a datové struktury, které používá operační systém pro ukládání záznamů souborů na disk či diskovou oblast; způsob, kterým jsou soubory na disku organizován. 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 (např. doručování elektronické pošty), ale velmi často jsou považovány za běžnou součást systému (např. kompilátor). Viz také aplikační program, jádro systému, operační systém.

volání systému Služby, které poskytuje aplikačním programům jádro systému, a způsob, jímž jsou volány. Viz sekci 2 manuálových stránek.