Průvodce Linuxem

Michal Dočekal

31. října 2007

Obsah

Seznam tabulek

Předmluva

O knize

Linux je fenoménem současnosti. Avšak zatímco v oblasti serverů je dobře znám, v oblasti desktopů je znám méně. Situace se však časem lepší. Začíná se o něm stále více hovořit a jeho uživatelská základna roste. Pokud se chcete dozvědět, co je to Linux, co vám přinese, jak jej začít používat a jak jej optimálně využít, jste na správném místě.

Tuto dokumentaci jsem zaměřil na člověka se zájmem poznat Linux poněkud více do hloubky, aby z něj mohl vytěžit maximum. Není to dokumentace určená pro naprostého začátečníka, který nemá ani ty nejzákladnější znalosti o fungování počítačů.

Dokumentaci jsem se nesnažil udělat zcela podrobně, důraz kladu především na základy a dávám k dispozici odkazy na další materiály, kde jsou příslušné záležitosti rozebrány do podrobna. Snažil jsem se kromě teoretického základu poskytnout rovněž praktické rady, které by vám měly pomoci s vytvořením těch správných návyků, abyste byli schopni s GNU/Linuxem pracovat efektivně.

Tato dokumentace je k dispozici v aktuální verzi k on-line prohlížení či stažení na webu:

http://www.poznejlinux.cz

Jakékoliv chyby nebo nejasnosti mi, prosím, nahlašte na e-mail info@poznejlinux.cz.

Licence a copyright

Kapitola 1

Úvod do GNU/Linuxu

1.1 Co je GNU/Linux?

GNU/Linux je operační systém[1] (dále OS), na jehož vývoji se podílí ohromné množství doborovolníků z celého světa, stejně jako řada komerčních firem. Primární důvod jeho existence je nabídnout uživatelům svobodný OS, tedy OS bez restrikcí a omezení. GNU/Linux je založený na Unixu, víceuživatelském a víceúlohovém OS s dlouhou tradicí a brilantně navrženou architekturou.

Jedná se o velice flexibilní systém, který je schopen zastat mnoho rolí. Naleznete jej nejenom na desktopech, serverech, routerech, clusterech, superpočítačích, ale i v embedded zařízeních jako jsou PDA, mobilní telefony nebo třeba automaty na zmrzlinu či vojenské transportéry. Vás ale bude asi nejvíce zajímat oblast desktopu, popřípadě serveru, a na tyto dvě oblasti se také zaměřím.

Tou nejdůležitější informací je však ta, že GNU/Linux není MS Windows. Tyto systémy jsou odlišné, postavené na jiných filozofiích, s jinými cíli a prioritami. To není chyba. Kdyby byly stejné, neměli byste důvod zajímat se o GNU/Linux. Na druhou stranu, k tomu, abyste se s ním naučili efektivně pracovat, je nutné, abyste jej nebrali jako kopii MS Windows, ale poněkud více otevřeli svou mysl novým poznatkům.

1.1.1 GNU a Linux

Linux je pouze jádro OS (tzv. kernel). Teprve spojením s projektem GNU vzniká plnohodnotný operační systém. To je důvod, proč nehovořím o Linuxu, ale o GNU/Linuxu.

1.1.2 Distribuce

GNU/Linux není jeden konkrétní operační systém. Ve skutečnosti tento termín zastřešuje ohromné množství relativně nezávislých projektů, jejichž integrací vzniká konkrétní operační systém. GNU/Linux je tedy možné si z oněch komponent vlastnoručně

[1] http://en.wikipedia.org/wiki/Operating_system

"postavit", ale to bývá značně složité a časově náročné. Proto existují linuxové distribuce, tedy již sestavené, konkrétní operační systémy založené na GNU/Linuxu. Výběru distribuce se budeme věnovat dále.

1.2 Proč Linux?

Pokud si kladete otázku, zda-li vám stojí GNU/Linux za vyzkoušení, možná byste rádi věděli, jaké jsou jeho stěžejní výhody. Dovolte mi však, spíše než o výhodách, pohovořit o vlastnostech GNU/Linuxu. Ostatně, výhody i nevýhody jsou vždy závislé na tom, jak se na danou vlastnost člověk dívá.

1.2.1 Svoboda softwaru

Účelem GNU/Linuxu je nabídnout uživateli svobodný operační systém. Tyto svobody jsou čtyři, a sice svoboda používat program, jak si uživatel přeje, studovat, jak funguje, upravovat jej a šířit jak původní, tak upravenou verzi.

Uživatelé se často ptají, k čemu jim tyto svobody jsou, když třeba nejsou programátoři a nechtějí si systém upravovat. Neuvědomují si přitom, že co nemohou nebo nechtějí udělat oni, mohou udělat jiní, a prospěch z toho mohou mít zpětně i ti, kteří se na změnách nepodíleli. Mezi tyto zprostředkované výhody patří mimo jiné:

- absence restrikcí (žádné ochrany proti kopírování, DRM, aktivace, atd.)
- žádné šmírování uživatele (alias spyware)
- rychlý vývoj
- brzká dostupnost aktualizací
- orientace na uživatele (programátor je sám uživatelem)
- možnost přímého kontaktu s vývojáři (tedy i ovlivnit vývoj programu)
- záruka kontinuity (odejde-li vývojář, nahradí ho jiný)
- nezávislost na jedné konkrétní firmě
- atd.

1.2.2 Stabilita, spolehlivost

GNU/Linux je stabilní a spolehlivý systém. Není náhodou, že je velmi rozšířený na serverech, kde je stabilita a spolehlivost jedním z rozhodujících kritérií. Je jasné, že stabilita je omezena použitým hardwarem, na poruchovém HW stabilitu očekávat nelze.

1.2.3 Výbava

Linuxové distribuce nejsou holé OS bez aplikací, jsou plně vybavené řadou aplikací pro běžné i speciální použití. Díky tomu můžete se systémem ihned začít pracovat, a nikoliv nejprve nahrávat jeden program za druhým.

1.2.4 Svoboda výběru

S GNU/Linuxem nejste omezeni na jediný prohlížeč, jediný přehrávač nebo jediné grafické prostředí. Máte možnost si vybrat, prakticky na každé úrovni. Jakou si zvolit distribuci, kterou variantu, jaké prostředí, jaké aplikace, atd. - záleží jen na vás. Nikdo vás nenutí jít jednou cestou.

1.2.5 Otevřenost a kontrola

GNU/Linux před vámi nic netají, v každém okamžiku můžete zjistit, co se děje i co se dělo. Stejně tak máte možnost ve svém systému změnit či upravit naprosto cokoliv. Jste to vy, kdo určuje pravidla.

1.2.6 Bezpečnost

Nebudu zastírat, že každý OS včetně GNU/Linuxu má jistá bezpečnostní rizika. Koneckonců, chyby dělá každý, včetně vývojářů GNU/Linuxu. Ale faktem je, že na tento operační systém prakticky neexistují viry ani spyware, Linux navíc sám obsahuje špičkový firewall a dobře vyladěný systém oprávnění. Aktualizace jsou dostupné velmi rychle, software se instaluje z repositářů s digitálně podepsanými balíčky. Tato opatření pomáhají efektivně chránit uživatele před bezpečnostními riziky.

1.2.7 Důvěryhodnost

GNU/Linux prochází, díky otevřenosti jeho zdrojových kódů, neustálým nezávislým bezpečnostním auditem veřejnosti. Kdokoliv se může podívat, jestli daný program nedělá něco, co nemá. U GNU/Linuxu je tedy důvodů pro pochybnosti velmi málo. Samozřejmě je ale vždy na uživateli, aby si sám určil, čemu důvěřuje. Ostatně, pokud má zájem, může projít zdrojový kód ke každému programu sám, a přesvědčit se na vlastní oči.

1.2.8 Flexibilita

Důsledkem otevřenosti a unixového charakteru GNU/Linuxu je ohromná flexibilita. GNU/Linux lze přizpůsobit prakticky jakémukoliv účelu, a to je důvodem, proč ho nalezneme téměř všude. Na osobních počítačích, serverech, superpočítačích, mobilech i PDA. Existuje také řada speciálně upravených linuxových distribucí pro řadu účelů.

1.3 Stinné stránky GNU/Linuxu

1.3.1 David a Goliáš

Nebudeme si nic nalhávat, alespoň na domácích a kancelářských počítačích má GNU-/Linux výrazně menší podíl než současný majoritní systém. Důsledkem toho je, že se řada společností vyvíjející software nebo hardware orientuje výhradně na onoho "Goliáše" a na jiné operační systémy příliš nemyslí.

To znamená, že některý hardware prostě pod GNU/Linuxem fungovat nebude, nebo bude postup jeho zprovoznění příliš složitý. Většina hardwaru však podporována je, takže není důvod panikařit. Se softwarem je to podobné. Je-li software k dispozici pouze ve verzi pro majoritní systém, pak je jasné, že v GNU/Linuxu jen tak spustit nepůjde.

Naštěstí pro tento problém existuje několik řešení. Prvním je projekt Wine a jeho komerční varianty (Cedega, Crossover). Lze si ho představit jako tlumočníka mezi Linuxem a programem pro majoritní systém. Tyto projekty samozřejmě nemají ani zdaleka 100% úspěšnost, ale žít se s nimi dá. Druhým řešením jsou ryzí emulátory (VMWare, Virtualbox, qemu, apod.), do kterých se nainstaluje majoritní systém a požadovaný software, a to celé funguje pod GNU/Linuxem.

Posledním řešením je současný provoz GNU/Linuxu a MS Windows. Oba systémy mohou koexistovat na jednom počítači a uživatel se může sám rozhodnout, který operační systém v danou chvíli spustí.

1.3.2 Patentově chráněné technologie

Některé známé a používané patentově chráněné technologie (zejména audio a video kodeky), nemohou být součástí některých distribucí (v závislosti na tom, kde se vyrábí, nebo pro koho jsou určeny), a proto bývá nutné tyto komponenty do systému doplnit. U některých distribucí je to velmi jednoduché, jinde to bývá složitější. Tyto záležitosti však bývají podrobně popsány v dokumentaci nebo na webových stránkách distribuce.

1.3.3 Čas jsou peníze

GNU/Linux není MS Windows. I když jsou po uživatelské stránce podobné, jejich správa i funkčnost se liší. Tudíž je třeba, abyste se naučili něco nového, možná přečetli nějakou dokumentaci a GNU/Linux si náležitě osahali a seznámili se s ním. To samozřejmě vyžaduje čas, který můžete a nemusíte mít. Výhodou je, že si můžete sami určit, kam až chcete v seznamování se s GNU/Linuxem zajít, co se naučit a co ne, a libovolně si rozvrhnout "učební plán" podle toho.

1.4 Exkurze do historie

1.4.1 Unix

Dva lidé, Ken Thompson a Dennis Ritchie, stáli roku 1969 v rámci Bell Laboratories u zrodu operačního systému Unix. Tento operační systém byl vytvořen od počátku tak, aby s ním mohlo současně pracovat více uživatelů (byl tedy od začátku víceúlohový a víceuživatelský). Jeho budoucnost byla zajištěna přenositelností. Za tímto účelem byl vytvořen programovací jazyk C, do kterého byl posléze Unix přepsán a přenesen na jiné platformy. Unix získal svou slávu především díky své architektuře, která má ryze geniální myšlenku:

- existuje velké množství malých prográmků

- každý z nich dělá jenom jednu věc, ale dělá ji dobře
- prográmky spolu mohou komunikovat
- lze je a jejich spolupráci řídit pomocí příkazové řádky

Kombinací těchto prográmků je možné realizovat přesně to, co je třeba. Této architektuře se říká modulární, systém je komplex relativně samostatných komponenet, které spolupracují, a řízením jejich spolupráce je možné dosáhnout přesně toho, co člověk potřebuje.

V Unixu platí ještě jedno pravidlo - "všechno je soubor". To se týká mj. i hardwaru, takže v Unixu je vaše tiskárna, pevný disk či síťová karta obyčejný soubor. A není náhodou, že ony výše zmíněné prográmky umí mj. velmi dobře zacházet se soubory.

Z těchto důvodů byl Unix velice oblíbený a netrvalo dlouho a začaly vznikat jeho další implementace.

1.4.2 GNU

Posuneme se do roku 1983, kdy Richard M. Stallman založil projekt GNU. Stallmanovi vadila tzv. proprietární (uzavřená) podstata dosavadního softwaru, tedy ta, která existuje dodnes. Spočívá v tom, že jeden subjekt daný software "uzavře" pomocí licence a autorského zákona. Uživatelům pak brání v tom, aby jej jakkoliv měnili, upravovali, popřípadě i šířili dál. Uzavřenost softwaru také způsobuje závislost uživatele na výrobci, který rozhoduje, kdy a jestli program opraví, upraví, či vylepší. Uživatel pak de facto ani nemá šanci si ověřit, že software dělá skutečně to, co dělat má, a ne nic nepříjemného navíc, což je v dnešní době velice aktuální téma (viz spyware, adware, rootkity, apod.).

Richardovi se tato situace nelíbila, byl toho názoru, že uživatelé by měli mít příslušná práva nskládání se softwarem, který používají, tedy že software by měl být svobodný. Základem pro používání jakéhokoliv softwaru je operační systém, a tak se Richard rozhodl vytvořit svobodný operační systém a založil projekt GNU, který měl vytvořit svobodnou implementaci operačního systému Unix. Vývoj zastřešil v rámci Free software foundation (Nadace svobodného softwaru).

Svobodný software je ale pojem, který v našem právním řádu neexistuje. Všechen software se musí licencovat. Pokud je software bez licence, spadá do kategorie public domain. Ale public domain software může kdokoliv uzavřít, vytvořit z něj proprietární software a svobodu jeho uživatelům odebrat. Z toho důvodu vytvořil Richard licenci GNU/GPL, která je považována za copyleft (copyright naruby). Zatímco copyright dnes slouží k tomu, aby moc nad softwarem soustředil v rukou jednoho subjektu a ostatní omezil, copyleftová licence GNU/GPL přesně tohle znemožňuje tím, že veškerá odvozená díla od GNU/GPL softwaru musí být vydána pod stejnou licencí, která svobody zajišťuje.

Licence GNU/GPL má copyleftový charakter, což znamená, že software smí být šířen pouze pod stejnou licencí. Není tedy možné vzít GNU/GPL software a udělat z něj proprietární software. Více informací o projektu GNU a myšlenkách Richarda Stallmana naleznete na stránkách projektu GNU.

1.4.3 Linux a GNU/Linux

Nyní se přesuneme do roku 1991. Projektu GNU chyběla k dokonalosti jediná věc, funkční jádro operačního systému (kernel). Ten sice projekt GNU zahrnoval (GNU-/Hurd), ale byl příliš složitý, než aby byl zralý produkčního nasazení (dodnes není).

V roce 1991 finský student, Linus Torvalds, zveřejnil svůj úmysl sestavit unixové jádro, které by, dle jeho slov, nemělo být tak profesionální jako projekt GNU. Jenomže do práce se zapojilo tak obrovské množství nadšenců, že se velmi brzy situace změnila a z Linuxu se stal profesionální unixový kernel. Sloučením softwaru z projektu GNU a linuxového jádra vznikl GNU/Linux. V roce 1994 byl vydán Linux 1.0 a o rok dříve již vznikly dvě linuxové distribuce, Slackware a Debian.

GNU/Linux začal velmi rychle nabývat na síle. Vzniklo ohromné množství dalších projektů, přinášejících další a další funkcionalitu uživatelům tohoto svobodného unixového operačního systému. GNU/Linux začaly podporovat i velké korporace jako IBM, Hewlett-Packard, Novell či Oracle.

Roku 1998 nastal linuxový boom, kdy došlo k jeho růstu na poli serverů o více než 200%, a byl tak spolehlivě operační systém s nejrychleji rostoucím podílem na trhu.

GNU/Linux ale nezískával uživatele pouze na poli serverů, ale i na poli desktopů. GNU/Linux byl, stejně jako Unix, expert friendly, tedy přívětivý pro experty. Branou na trh desktopových operačních systémů je ale především uživatelská přívětivost. A právě tady pomohla celá řada projektů, ať už to byly projekty jako KDE (ohlášen 1996, první verze v roce 1998) či GNOME (ohlášen 1997, první verze v roce 1999) přinášející komfortní grafický desktop, správci distribucí jako Mandrake (první verze 1998) a SUSE (první "SUSE" verze 1996, dříve založená na Slackware) nabízející snadnou a uživatelsky přívětivou konfiguraci systému, stejně jako celá řada vzniknuvších aplikací, které byly schopné realizovat běžné činnosti na desktopu (kancelářské balíky, multimediální přehrávače, hry, apod.).

V roce 2002 používalo dle průzkumu provedeného společností IDC tento operační systém 2.8% uživatelů desktopů. Na serverech měl v tomto roce už 25% podíl.

Novější data bohužel nemáme a webové statistiky jsou neprůkazné, nicméně lze předpokládat, že podíl desktopových uživatelů GNU/Linuxu je mezi 4-5% a na serverech má GNU/Linux více než 30% podíl a neustále roste. Na některých trzích má GNU/Linux dokonce prvenství (superpočítače) a stále více firem a institucí jej implementuje do svých IT infrastruktur.

1.5 Zdroje a odkazy

- Linux Express, Co je to Linux?
- ABC Linuxu, Leoš Literák, Co je to Linux?
- Wikipedia, Linux (česky)
- Wikipedia, Linux (anglicky)
- Web proc.linux.cz, Adam Přibyl, Proč Linux?

- Audiovizuální centrum Silicon hill, Petr Koloros, Linux? Na co? (video)

Kapitola 2

Výběr distribuce

Výběr distribuce je pro začátečníka kritický. Špatná volba ho může od GNU/Linuxu odradit, a proto je třeba vědět, jaké distribuce existují a jaké mají vlastnosti, aby si mohl vybrat tu správnou.

Proč je to tak důležité, respektive, v čem se distribuce liší? Všechno je samozřejmě GNU/Linux, nicméně odlišnosti jsou v zaměření distribuce (desktop, server, ap.), vybavení, aktuálnosti balíčků, ale tím nejpodstatnějším pro začátečníka jsou nástroje pro snadnou a pohodlnou správu systému.

Linuxových distribucí existují stovky (přibližně 350), ale není důvod k obavám, v popředí je jen několik málo kousků, které si v této knize představíme. Pokud chcete, můžete si ale samozřejmě vybrat z mnohem širšího seznamu distribucí na webu http://www.distrowatch.com.

2.1 Ubuntu

Ubuntu (http://www.ubuntu.cz/) je distribuce primárně zaměřená na začátečníky a běžné uživatele s filozofií založenou na humánních principech. Je a vždy bude ryze nekomerční, vychází zhruba každého půl roku a je možné si ji nechat zaslat zdarma poštou[1]. Nemá ještě tolik grafických konfiguračních nástrojů jako jiné distribuce zaměřené na pohodlí uživatelů, ale vyvažuje to ochotná komunita[2] i existence řady podrobných návodů[3], jak co udělat, z nichž ty nejdůležitější má distribuce přibalené a lokalizované do češtiny. Jelikož je založena na distribuci Debian (viz níže), má k dispozici ohromné množství softwaru.

2.2 Mandriva

Mandriva (http://www.mandrivalinux.cz/) je další s distribucí zaměřených na začínající a běžné uživatele. Nabízí sadu grafických konfiguračních nástrojů, snadnost použití a

[1] http://www.ubuntu.cz/ziskejte/poslat
[2] http://forum.ubuntu.cz/
[3] http://wiki.ubuntu.cz/

přívětivost. Má u nás velmi dobré zastoupení, vychází jak v placené, krabicové verzi, tak ve volně stažitelné. Ke krabicové verzi (mají ji i některá knihkupectví) dostává uživatel velmi dobře sestavenou českou příručku[4], která je k dispozici i v bezplatné elektronické podobě.

2.3 OpenSUSE

Vznikající pod taktovkou společnosti Novell, OpenSUSE (http://cs.opensuse.org/ je opět distribuce zaměřená na desktop a uživatelskou přívětivost. Obsahuje komplexní ovládací centrum Yast s řadou grafických konfiguračních nástrojů. Čeští uživatelé mají k dispozici komunitní portál[5], kde naleznou diskusní fóra, články a řadu návodů.

2.4 Fedora Cora

Fedora (http://fedoraproject.org/ se stala odnoží komerčního Red Hatu, který nadále vyvíjí podniková řešení pro servery (Red Hat Enterprise Linux, který z Fedory vzniká). Distribuce je nekomerční, obsahuje některé grafické konfigurační nástroje, takže i začátečník bude mít možnost distribuci pohodlněji používat, i když příkazové řádce se patrně nevyhne. Čeští uživatelé by měli obrátit svou pozornost k místnímu komunitnímu portálu[6], kde naleznou českou dokumentaci a diskusní fórum.

2.5 Debian GNU/Linux

Debian (http://www.debian.org) je ryze nekomerční distribuce, de facto jedna z největších a nejznámějších. Je ovšem určena spíše pokročilejším uživatelům, neboť se v ní mnohé konfiguruje ručně. Debian přibližuje začátečníkům a běžným uživatelům distribuce Ubuntu (viz výše), která z něj vychází.

Distribuce se člení se do třech větví. Stabilní, která je méně aktuální, ale, jak její název napovídá, je stabilní a spolehlivá. Jejím opakem je nestabilní větev, která má sice ten nejaktuálnější software, ale ten není důkladně otestován. Jakmile se balíček přidaný do nestabilní větve na svém místě nějakou dobu zahřeje a nikdo nehlásí žádné problémy, putuje do testovací větve. Ta se jednou za čas zmrazí a vyloupne se z ní stabilní verze.

Čeští uživatelé mohou navštívit český portál[7], kde naleznou některé návody, ale především českou e-mailovou konferenci.

[4] http://www.mandrivalinux.cz/dokumentace
[5] http://portal.suse.cz/
[6] http://www.fedora.cz/
[7] http://debian.cz/

2.6 Slackware

Slackware (http://www.slackware.com) je distribuce s velmi dlouhou tradicí, dokonce jedna z nejstarších přežívajících distribucí vůbec. Její tvůrce tvůrce je jediný člověk, Patrick Volkerding. Distribuce vyniká "čistotou" konfiguračních souborů a jednoduchostí provedení (ve smyslu minimálních úprav jednotlivých komponent, ze kterých se systém skládá). Není zaměřena na začátečníky, ruční konfigurace je nutností. Čeští uživatelé mohou navštívit komunitní portál[8].

2.7 Gentoo

Gentoo (http://www.gentoo.org) je zástupce distribucí "from scratch", tedy "od začátku" (volně přeloženo). V Gentoo si člověk vše realizuje a volí sám, software si kompiluje a optimalizuje na daný hardware. Distribuce pak vhodnými prostředky přispívá a pomáhá uživateli snadno dosáhnout kýženého stavu, ať už excelentní dokumentací, díky které se člověk dokonale naučí, jak vše v systému probíhá, nebo pomocí vhodných technických prostředků, které dané činnosti usnadní a automatizují. Pokročilým uživatelům tak dává do ruky velmi mocný nástroj, na kterém by si ale začátečník bez tun trpělivosti a volného času vylámal zuby. Čeští uživatelé mohou navštívit české fórum[9] nebo českou wiki[10].

2.8 Kterou distribuci vybrat?

To záleží na vašich požadavcích. Chcete do GNU/Linuxu rychle proniknout, máte dost času, trpělivosti a trochu nadšenecké a kutilské povahy? Zvolte nějakou distribuci pro pokročilejší uživatele. Chcete raději přítulné prostředí, které se snadno spravuje? Nechcete do GNU/Linux proniknout nebo chcete pronikat postupně? Zvolte distribuci vhodnou pro začátečníky. Pokud se nemůžete rozhodnout, klidně jich vyzkoušejte více.

2.9 Pár praktických rad

Osobně doporučuji dobře zvážit eventuelní změnu distribuce v začátcích (po aklimatizaci to už problém nebývá). Někteří lidé mívají tendenci prosazovat svoji oblíbenou distribuci a radit začátečníkovi, když žádá o pomoc, aby přešel právě na tu jejich. Není dobré takové rady následovat, protože přechod na jinou distribuci může být v začátcích lehce matoucí. Naopak, pokud zjistíte, že vám distribuce opravdu nevyhovuje, a rádi byste zkusili jinou, učiňte tak. Je to rozhodně lepší než opustit GNU/Linux kvůli tomu, že se vám nezamlouvá jedna distribuce.

Mojí druhou, možná ještě podstatnější radou, je dát si pozor na dobu vydání dané verze distribuce. Může se stát, že natrefíte na starou verzi distribuce, ať už v knihkupectvích, kde mohou některé distribuce přečkat řadu let, ačkoliv nové verze vychází

[8] http://www.czslug.cz/
[9] http://forums.gentoo.cz/
[10] http://cs.gentoo-wiki.com/

třeba každých šest měsíců, nebo na určitých serverech, kde z nějakého důvodu nebyla provedena aktualizace. Starým verzím doporučuji se obloukem vyhnout (na novějších počítačích nemusí fungovat, jejich podpora hardwaru bude omezená, nemusí k nim už být k dispozici aktualizace a přicházeli byste o všechny nové vlastnosti, které jsou v novějších verzích).

Stejně tak však varuji před horkými brambory, tedy distribucemi vydanými v průběhu posledních několika týdnů. Nestává se to zase až tak často, ale někdy se nepodaří nalézt všechny chyby a některé vážnější v distribuci zůstanou. Já doporučuji pro jistotu tak zhruba měsíc po vydání nové verze počkat, a pak prověřit poznámky k vydání (relase notes, errata, apod.), kde naleznete informace o známých chybách.

Jak zjistíte, která verze je nejnovější, popřípadě kdy byla vydána? K tomu můžete využít buď oficiální webové stránky distribuce, nebo server Distrowatch[11], který udržuje komplexní přehled o světu linuxových distribucí.

Každopádně vám přeji šťastnou ruku a pevně doufám, že vám tato dokumentace pomůže při objevování světa GNU/Linuxu prostřednictvím vámi zvolené distribuce.

2.10 Zdroje a odkazy

- Web proc.linux.cz, Adam Přibyl, Chci si Linux vyzkoušet na svém počítači
- Web proc.linux.cz, Adam Přibyl, Kde získat Linux a jaký si vybrat
- PC Svět, Jiří Formánek, Systém do kapsy – Live distribuce Linuxu
- Linux Express, Jak Linux získat?
- ABC Linuxu, Leoš Literák, Linux
- Wikipedia, Linuxová distribuce (česky)
- Wikipedia, Linux distribution (anglicky)
- Web Distrowatch, http://distrowatch.com

[11] http://distrowatch.com

Kapitola 3

Instalace distribuce

3.1 Jak si distribuci opatřit

Ať už chceme GNU/Linux vyzkoušet prostřednictvím live distribuce, nebo jsme si už vybrali klasickou, kterou zamýšlíme instalovat na pevný disk, potřebujeme si ji nejprve opatřit. V zásadě máme několik možností:

- stáhnout ISO obrazy médií a vypálit je
- nechat si je poslat za určitou částku
- nechat si je na počkání vypálit za určitou částku
- koupit ji v obchodě nebo v knihkupectví
- koupit si ji jako přílohu v časopise
- zkopírovat si ji od kamaráda
- nechat si ji bezplatně zaslat

3.1.1 Linuxové obchůdky

Nemáte-li možnost distribuci stáhnout a vypálit, můžete si ji objednat v některém linuxovém obchůdku, někde i nechat si ji na počkání vypálit. Některé z nich uvádím v následujícím seznamu, další naleznete při troše hledání v katalozích či na Googlu:

- http://linuxsoft.cz/
- http://linuxos.sk/
- http://shop.qcm.cz/
- http://dvdlinux.cz/
- http://linux-cd.cz/

3.1.2 Obchody a knihkupectví

V obchodech zabývajících se výpočetní technikou příliš zástupců linuxových distribucí, paradoxně, nenajdeme. Někde nám alespoň prodávají distribuce SUSE či Mandrake, ale žádná sláva to není. I mnohá knihkupectví jsou na tom lépe.

Pokud už sháníme nějakou distribuci v obchodech nebo knihkupectvích, měli bychom si předem ověřit, že daná verze je aktuální. Občas je možné sehnat velmi staré verze za nehorázné sumy, jelikož se prodávají jako knihy s CD/DVD přilohou. Starším distribucím bych se raději vyhnul.

3.1.3 Časopisy

Na našem trhu je to především Linux+, který je běžně k dispozici na stáncích. Má 2xDVD přílohu, jednu live distribuci s recenzovanými programy a minimálně jednu další klasickou linuxovou distribuci. Linuxové distribuce však bývají rovněž součástí řady vydání jiných časopisů (např. CHIP, apod.)

3.1.4 Kamarád linuxák

Pokud máte známého, který je vám ochotný zvolenou distribuci stáhnout, eventuelně vypálit, vypůjčit či překopírovat, tím lépe, je to legální. Jediný problém může nastat u komerčních (placených) verzí distribucí, které nemusí mít jako celek tak benevolentní licenci. Je to dáno tím, že se v takových distribucích nacházejí komponenty s jinými licencemi, které je třeba respektovat. V takovém případě doporučuji pročíst licenci k dané distribuci.

3.1.5 Zaslání zdarma poštou

Máte-li zájem o distribuci Ubuntu, můžete si její live a instalační CD nechat poslat poštou, z vaší strany zcela bezplatně. Více viz http://ubuntu.cz/. Jiné distribuce nic takového nenabízejí, alespoň pokud vím.

3.2 Instalujeme distribuci

V zásadě existují dva scénáře instalace GNU/Linuxu na váš počítač, instalace na čistý počítač (bez OS, bez dat) a instalace na užívaný počítač (s OS i s daty).

V případě, že máte na počítači nějaká existující nezazálohovaná data, tak je ještě před instalací zálohujte. Může se přihodit cokoliv, počínaje překliknutím, přes výpadek elektřiny, po chybu softwaru. Není nic horšího, než když je první zkušenost s jiným operačním systémem korunována ztrátou dat.

Už jsme si o tom říkali, ale já to raději řeknu ještě jednou. Pokud možno, neinstalujte ryzí live distribuce, i když instalaci třeba umožňují. Tato funkce bývá neoficiální a neodladěná, a i když při instalaci nepřijdete o všechna svá data, nemusíte být spokojeni s distribucí samotnou, protože jejím primárním účelem je běh z média, a tomu je přizpůsobena. Pochopitelně není problém provést adekvátní úpravy ručně, ale to je

práce pro zdatného a pokročilého linuxového uživatele. Vyberte si raději klasickou distribuci, která je pro instalaci na pevný disk navržena. Tato poznámka se samozřejmě netýká distribucí, které jsou primárně navrženy pro instalaci na pevný disk, ale nabízejí instalační CD s live systémem, jako třeba Ubuntu.

Instalujete-li GNU/Linux do VMWare nebo jiného virtuálního stroje, buďte připraveni na eventuální problémy, linuxové distribuce bývají sestavené pro provoz na skutečných, ne virtuálních počítačích.

Dále berte na vědomí, že tento průvodce je relativně stručný a velmi obecný, navíc různé distribuce mívají instalaci řešenou jinak, takže se klidně můžete setkat s distribucí, jejíž postup instalace bude jiný, některé prvky mohou být vynechány, pořadí může být změněno, ap.

Měli byste si v každém případě opatřit instalační příručku ke své distribuci a tu si důkladně pročíst. Pokud nemáte tištěnou verzi, měli byste si ji před instalací vytisknout.

Dlužno dodat, že většinu rozhodnutí za vás, pokud si zvolíte distribuci zaměřenou na běžného domácího uživatele, učiní samotný instalátor, a vy budete celou dobu spíše jen upravovat jeho nastavení.

3.3 Průběh instalace

Instalace GNU/Linuxu se běžně sestává s těchto fází:

- nabootování z instalačního CD
- rozdělení pevného disku
- výběr softwaru k instalaci
- instalace
- konfigurace zavaděče
- základní konfigurace systému (zadání hesla superuživatele, vytvoření uživatelského účtu, nastavení sítě, firewallu, apod.)

3.3.1 Nabootování z instalačního CD

Předpokládám, že to s přehledem zvládnete. Vložíte první instalační CD/DVD do mechaniky a restartujete počítač. Pokud se nezavede instalátor, v BIOSu nastavíte bootování z CD mechaniky. Pokud to nepomůže, vytvoříte bootovací disketu či konsultujete dostupnou dokumentaci.

Na (těžce) nekompatibilním hardwaru může ve fázi bootování GNU/Linuxu nastat problém. Je dobré v této situaci konzultovat dokumentaci a vyzkoušet některé parametry, které můžete předat jádru Linuxu, a tak ho donutit nepoužívat problémový hardware.

Parametr jádru zadáte tak, že na úvodní obrazovce s příkazovou řádkou, která se zpravidla objeví ještě před vlastním nabootováním GNU/Linuxu, zapíšete název jádra

(zpravidla Linux, ale mrkněte do dokumentace) a jednotlivé parametry oddělené mezerou podle toho, co chcete vypnout (poslední tři doporučuji použít až teprve v situaci, kdy první dva nezaberou):

- **noapic**
- **noacpi**
- **noapm**
- **nodma**
- **nosmp**

3.3.2 Rozdělení pevného disku

Rozdělení pevného disku je záležitost, kterou můžete provést prostřednictvím instalačního programu či jiného softwaru. Pokud máte k dispozici čistý počítač, je situace jednoduchá, prostě smažete celý disk a necháte instalátor, aby rozvrhl oddíly sám. Pozor dejte v případě, že máte na disku data, která chcete zachovat.

Máte-li na pevném disku data (či celý operační systém), která chcete zachovat, mohou nastat dvě situace. Buď máte disk už rozdělený, v tom případě stačí vybrat jeden oddíl, evakuovat jej, v instalátoru jej vymazat a vytvořit místo volného místa patřičné oddíly. Poslední varianta je situace, kdy pevný disk rozdělený nemáte, v takovém případě je nutné vzít existující oddíl a zmenšit jej. To je ze všech variant nejvíce *"nebezpečná"* varianta, takže pozor na data!

Zmenšit oddíl s Windows může udělat linuxový instalátor, dokonce i bez zničení dat na něm, ale nedokáží to všechny distribuce a u jistých proprietárních souborových systémů (jako NTFS), jejichž specifikace jsou tajeny, je tato činnost velmi riziková. V každém případě by bylo velmi vhodné konzultovat instalační příručku zvolené distribuce, eventuelně pak dát přednost jinému softwaru pro rozdělování pevných disků. Tam ale doporučuji vyhnout se vytváření oddílu s linuxovými souborovými systémy, to nechte na instalátoru GNU/Linuxu.

3.3.2.1 Pevné disky v GNU/Linuxu

Pro naše účely postačí říci, že IDE zařízení (pevné disky a CD mechaniky na většině domácích počítačů) jsou označovány jako *hda*, *hdb*, atd., oddíly na pevných discích se značí číslem (např. *hda1* nebo *hdb4*). Zařízení SCSI a SATA disky GNU/Linux rozlišuje buď jako *sda*, *sdb*,..., CD mechaniky jsou pak *scd0*, *scd1*,... a opět, u pevných disků se oddíly označují číslem (*sda1*, *sda2*, atd.). Povětšinou ale stačí sledovat kapacity disků/oddílů a symbolika nesymbolika, co je to za disk/oddíl, vám dojde.

3.3.2.2 Rozvržení linuxových oddílů

Pokud máte možnost nastavit v instalátoru automatické rozvržení oddílů, učiňte tak (ovšem překontrolujte si, zda tak náhodou nepřijdete o data na některém z existujících oddílů). V nejzazší nouzi je možné nechat instalátor vytvořit pouze jediný oddíl, ale

tento postup nedoporučuji. Určitě se hodí minimálně dva, první na OS a data, druhý na swap (odkládací oddíl, něco jako `pagefile.sys` ve Windows, doporučuje se 2x kapacita RAM, rozumné maximum se pohybuje kolem 2GB). Velmi často se používá rozdělení na tři oddíly, první na kořenový souborový systém, "`/`", kde bude veškerý software a systémová konfigurace, druhý na swap a třetí na domovské adresáře uživatelů, "`/home`", kde budou veškerá uživatelská data.

3.3.2.3 Linuxové souborové systémy

V GNU/Linuxu máte široký výběr souborových systémů. Mají svá různá specifika a užití, o jejich vlastnostech se lze dočíst více na odborněji zaměřených webech, pro domácí použití postačí *ext3* nebo *reiserfs*.

3.3.3 Výběr softwaru k instalaci

Ten má většinou podobu volení kategorií, ze kterých se zpravidla nainstaluje nějaký předem daný výběr, který by měl většině uživatelů vyhovovat. Doporučuji spíše volit méně balíčků a doinstalovávat později než naopak, abyste se v té záplavě softwaru neztratili.

3.3.4 Instalace

Pohodlně se usaďte a pozorujte progressbar nebo si jděte uvařit čaj, bude to chvilku trvat.

3.3.5 Konfigurace zavaděče

Poslední krok týkající se celého počítače a nikoliv pouze GNU/Linuxu je konfigurace zavaděče. Máte celou řadu možností, jak v tomto případě postupovat, většinou postačí zvolit instalaci do MBR (hlavního zaváděcího záznamu), linuxové zavaděče jsou schopné bez problémů zavést Windows (naopak to jde také, ale chce to trošku hackingu). GNU/Linux má k dispozici dva zavaděče, LILO a GRUB. Co zvolíte záleží na vás. Ve zkratce, Lilo je minimalistický zavaděč, Grub je poměrně komplexní a nabízí více možností, ale je o to větší.

Zaváděcí disketu je určitě dobré vytvořit, protože MS Windows mají ten neslušný zvyk při své instalaci přepsat bez ptaní MBR svým zavaděčem, a kompletně tak v tomto případě zbořit "most" ke GNU/Linuxu (nebo jiným OS). To se dá samozřejmě ex-post napravit, ale i tak je dobré mít zálohu, pokud nebudete mít nějaké záchranné CD zrovna po ruce.

3.3.6 Zadání hesla superuživatele

Superuživatel (uživatel root) v GNU/Linuxu je Bůh, může cokoliv, neplatí pro něj žádná omezení. Může systém libovolně upravit, ale může ho i zničit. Je dobré zvolit nějaké silnější heslo a dobře si jej zapamatovat. V některých distribucích (např. Ubuntu)

se heslo pro uživatele root nezadává, administrátorské činnosti se realizují přes *sudo*, kde postačí zadat heslo k uživatelskému účtu (viz níže).

3.3.7 Vytvoření uživatelského účtu

Velmi silně se nedoporučuje pracovat v GNU/Linuxu pod účtem superuživatele. Systém oprávnění je v unixových systémech vyladěn tak, aby uživateli nic nebránilo realizovat jeho práci, bude-li přihlášen jako obyčejný uživatel, a přitom byl systém samotný dobře chráněn. Pokud bude uživatel potřebovat patřičná privilegia, může si je pro daný úkol jednoduše opatřit, i když bude přihlášen jako obyčejný uživatel.

3.3.8 Nastavení firewallu

Linuxové systémy obsahují stavový firewall fungující na úrovni jádra OS. Viry a spyware v GNU/Linuxu problém nepředstavují, a proto je nastavení firewallu to jediné (kromě přirozené ostražitosti), co budete pro bezpečnost svého systému dělat. Distribuce zaměřené na běžné uživatele obsahují většinou velmi jednoduchý klikací nástroj k jeho nastavení. Pro potřeby domácího uživatele doporučuji:

- ujistit se, že je firewall zapnutý
- nepovolit přístup zvenčí k žádné službě (pokud to vyloženě nepotřebujete)

Některé distribuce (např. Ubuntu) firewall ve výchozí instalaci nenastavují, protože žádná síťová služba ve výchozím nastavení neposlouchá na síti. V takovém případě firewall opravdu není třeba.

3.4 Zdroje a odkazy

- ABC Linuxu, Rastislav Stanik, Na co se často ptáme: Organizácia disku
- ABC Linuxu, Vlastimil Ott, Na co se často ptáme 1: LILO
- Audiovizuální centrum Silicon hill, InstallFest (videa)

Kapitola 4

Práce s GNU/Linuxem

První, co nás čeká, je přihlášení do systému. Zadáme své uživatelské jméno a heslo a přihlásíme se. Nebudou-li se zobrazovat hvězdičky u hesla, nenecháme se tím vyvést z míry, je to bezpečnostní opatření zabraňující potenciálnímu útočníkovi zjistit, kolik znaků má naše heslo.

Přihlašujeme-li se do příkazového řádku, na výzvu login: zadáme uživatelské jméno a na výzvu password: heslo. Pokud jsme nainstalovali grafické rozhraní, měli bychom být schopni ho nastartovat pomocí příkazu: **startx**

Za několik sekund se dostaneme do grafického rozhraní systému GNU/Linux. Nyní záleží na tom, jaké grafické prostředí jsme si vybrali nebo nám bylo přiděleno. V úvahu pro většinu distribucí přichází dvě. Gnome a KDE.

4.1 Společné prvky Gnome a KDE

Ovládání Gnome, KDE a dalších prostředí v GNU/Linuxu má několik společných prvků. Tím prvním a nejdůležitějším je funkce levého a pravého tlačítka myši. Levé tlačítko provádí volbu, pravé tlačítko vyvolává vlastnosti. Dalším prvkem je koncept virtuálních ploch. Na každé "ploše" můžete mít spuštěné různé aplikace a mezi plochami se můžete snadno přepínat.

Plocha jako taková, tj. ikony na pozadí, je ovládacím prvkem přítomným v Gnome a KDE. V jiných, odlehčených prostředích, být nemusí (lze však doplnit jako samostatný program). Obsah plochy, pokud je plocha k dispozici, je uložen v adresáři `/home/uzivatel/Desktop`, kde *uzivatel* je vaše uživatelské jméno. Tolik ke společným prvkům. Podívejme se teď na jednotlivá prostředí.

Schránka funguje klasickým způsobem přes menu nebo klávesové zkratky (Ctrl-C nebo Ctrl-Insert kopíruje, Ctrl-V nebo Shift-Insert vkládá, Ctrl-X vyřízne). Funguje ale ještě pohodlněji. Pokud myší označíte nějaký text, automaticky se zkopíruje do schránky. Pak se stačí přesunout jinam a použít prostřední tlačítko myši, které zkopírovaný text vloží.

4.2 Prostředí KDE

Prostředí KDE má plochu s ikonami a spodní lištu. Na spodní liště úplně vlevo se nachází tlačítko K (ať už tak vypadá či ne), které je branou k aplikacím a hlavním funkcím (tj. odhlášení, nastavení prostředí, spuštění programů, vyhledávání). Po pravici tohoto tlačítka jsou tlačítka další, vedoucí k těm nejdůležitějším aplikacím. Následuje přepínač virtuálních ploch, pruh úloh (zde se zobrazují spuštěné programy), systémová lišta a hodiny.

Spodní lišta je vysoce konfigurovatelná (pravé tlačítko), lze do ní přidávat i tzv. applety (běžící prográmky, např. předpověď počasí) a různě měnit její velikost, umístění a prvky. Tlačítko K i vše ostatní lze libovolně přesouvat. Plocha je rovněž konfigurovatelná (pravé tlačítko).

Nastavení prostředí se nachází v menu K (pod položkou Nastavení systému nebo System settings). Tato nastavení se převážně týkají prostředí, nikoliv systému jako takového. Lze tu snadno nastavit rozložení klávesnice, což je dobrý první krok. Je to položka rozvržení klávesnice v místních zvyklostech a zpřístupnění. Implicitní klávesová zkratka pro přepnutí klávesnice je Ctrl-Alt-K.

Operace "táhni a pusť" se v KDE provádí pouze levým tlačítkem myši (pravým provést nejde), s tím, že po přesunutí položky se prostředí samo zeptá, co má udělat (kopírovat, přesunout, vytvořit odkaz, ap.). Podržíte-li **Shift** při pouštění položky, položka se do nového umístění přesune. Pokud podržíte **Ctrl**, položka se zkopíruje. Pokud podržíte obojí, vytvoří se odkaz.

4.3 Prostředí GNOME

Prostředí Gnome má dvě lišty a desktop. Spodní lišta obsahuje pruh úloh a přepínač virtuálních ploch. Horní lišta obsahuje tři menu, aplikace, místa a systém. Názvy jsou samovysvětlující. Menu aplikace zobrazí strukturovanou nabídku aplikací, menu místa zobrazí všechny připojené datové jednotky, včetně sítě. Nabízí i vyhledávání souborů. Systém pak nabízí systémové funkce - konfiguraci, správu prostředí i systému a odhlášení uživatele. Po pravé straně od menu se na liště nachází tlačítka pro rychlé spouštění vybraných aplikací. Následuje systémová lišta a hodiny. Jednotlivé funkční prvky se dají nastavovat po kliknutí pravým tlačítkem myši.

Do horní lišty Rozložení klávesnice lze v Gnome nastavit pomocí menu Systém, v němž stačí zvolit Nastavení-Klávesnice. Implicitní klávesová zkratka pro přenutí klávesnice je současné stlačení obou Shiftů, eventuelně Altů.

Operace "táhni a pusť" se provádí pouze levým tlačítkem myši (pravým provést nejde). Na rozdíl od KDE se po přesunutí položky provede to, co si prostředí myslí, že chcete provést. Pokud přesouváte položky ve stejném kontextu (na jedné datové jednotce), zvolí se operace "přesunout". V jiném kontextu (na různých datových jednotkách) se zvolí operace "kopírovat". Podržíte-li **Shift** při pouštění položky, položka se do nového umístění přesune. Pokud podržíte **Ctrl**, položka se zkopíruje. Pokud podržíte obojí, vytvoří se symbolický odkaz.

4.4 Tipy a triky

Potřebujete-li zadat znak z anglické klávesnice (typicky zavináč, křížek, apod.), ale zvolenou budete mít českou, nemusíte se přepínat zpět na anglickou. Podržíte pravý **Alt** a můžete vložit znaky z anglické klávesnice. Můžete při tom klidně používat **Shift**, a dostat se tak k dalším speciálním znakům.

Jednou z možností, jak rychle spustit aplikaci, je použít klávesovou zkratku Alt-F2. Zobrazí se dialog s editačním políčkem, kam stačí zapsat název aplikace, např. konsole, a "odentrovat". Pro někoho, kdo umí rychle psát a má dobrou paměť, to může být velmi užitečné.

Pokud používáte prostředí KDE a z nějakého jiného adresáře se pokusíte operací "táhni a pusť" přenést na plochu obrázek, jedna možnost v nabídce bude "Nastavit jako tapetu". Je to velmi jednoduchý způsob, jak změnit obrázek na ploše.

4.5 Příkazová řádka

K příkazové řádce se lze dostat přímo v grafickém režimu pomocí terminálového programu (konsole, gnome-terminal, xterm, eterm, ap.), který určitě najdete pod *Systémovými nástroji* (nebo něčím podobným). Grafické prostředí ale také můžete opustit, a dostat se tak do textového režimu. To provedete klávesovou zkratkou Ctrl-Alt-Fx, kde *x* je číslo od 1 zpravidla do 7, podle toho, na jakou virtuální konzoli se chcete přepnout. Ano, i textový režim má něco jako virtuální plochy. Na první virtuální konzoli se přepnete klávesovou zkratkou Ctrl-Alt-F1. Mezi jednotlivými virtuálními konzolemi se už v textovém režimu přepínáte pomocí Altu (tj. Alt-F1, Alt-F2, apod.). Na jedné z virtuálních konzolí sídlí samotné grafické prostředí, zpravidla na sedmé. Takže zpátky se dostanete pomocí Alt-F7.

4.6 Aplikace

Jaké aplikace použít, to nechám samozřejmě na vás. Jsou seřazené podle kategorií, takže by neměl být problém najít takovou aplikaci, kterou potřebujete. Doporučuji vám se podívat, jaké aplikace máte k dispozici a provést takový rychlý osobní průzkum. Když nebudete vědět, co zkusit, pak tu mám pár tipů. Firefox, Konqueror jsou webové prohlížeče (Konqueror je navíc i správce souborů). Jako kancelářský balík určitě zkuste OpenOffice. Jako jednoduchý textový editor postačí Kwrite či Gedit. Co se týče her, určitě vyzkoušejte Frozen Bubble (pozor, návykové!). K přehrávání hudby doporučuji Amarok, Rhythmbox či Xmms. K přehrávání videa můžete použít Totem nebo Mplayer.

Chcete najít linuxový ekvivalent určité aplikace? Pak se vám možná hodí tabulka ekvivalentů[1]. A nebo můžete použít databázi linuxového softwaru na adrese www.linuxsoft.cz.

[1] http://proc.linux.cz/ekvivalenty.html

4.7 Správa souborů

Jak pouštět aplikace, to už víme. Druhá nezbytná věc, kterou uživatel ke své činnosti potřebuje, je pracovat se soubory, které vytvořil, přesouvat je, kopírovat, mazat. K tomu je třeba nějakého správce souborů. Tím nejzákladnějším způsobem, jak toho lze dosáhnout, je v KDE použít ikonku "Můj adresář", v Gnome "Místa" - "Domovský adresář", čímž dosáhneme spuštění správce souborů. Mezi více okny správce souborů stačí použít operaci "táhni a pusť".

Správců souborů existuje více, já doporučím ještě dva. Prvním je vynikající správce souborů Midnight Commander pro příkazovou řádku (příkaz **mc**), druhým je Krusader, správce pro grafické prostředí. Velkou část operací lze také realizovat pomocí pravého tlačítka myši (např. zabalit, zašifrovat, apod.).

4.8 Adresářová struktura

V GNU/Linuxu jsou datové jednotky připojené do jedné adresářové struktury. Disková jednotka se pro uživatele jeví jako obyčejný adresář (tomuto adresáři se říká přípojný bod alias "mount point"). Adresářová struktura začíná kořenovým adresářem, který se označuje znakem `/`. Má svá přísná pravidla, ale o těch až jindy. Prozatím vám postačí vědět, že váš domovský adresář, kde můžete realizovat cokoliv, je `/home/uzivatel`, kde `uzivatel` je vaše uživatelské jméno. Dodatečné datové jednotky se připojují do adresáře `/mnt` nebo `/media`, zpravidla dle názvu (tj. `/media/cdrom` je přípojný bod vaší CD/DVD-ROM/RW mechaniky). V těchto adresářích můžete také najít identifikované oddíly s daty patřící jiným operačním systémům.

4.9 Velká a malá písmena

Unixové systémy rozlišují v názvech souborů a adresářů velká a malá písmena. Je tedy rozdíl mezi souborem s názvem "`ahoj`" a souborem s názvem "`Ahoj`".

4.10 Práce s médii

Přívětivé distribuce mívají vždy svůj vlastní recept, jak zacházet s médii, a proto nemohu než vám spíše ozřejmit principy této práce, s tím, že některé kroky, které nastíním, ve vámi zvolené distribuci možná nebudete muset realizovat.

Prvním krokem je připojení média k přípojnému bodu (tj. adresáři, kde budou jeho data přístupná). Detekované médium se vám velmi pravděpodobně objeví přímo na ploše.

Jakmile je médium detekováno, je třeba jej připojit. Pokud ikonu otevřeme (kliknutím nebo dvojklikem, podle nastavení), dojde k automatickému připojení. Nebo můžeme použít pravé tlačítko a zvolit "Připojit". Když budeme chtít médium odpojit, zvolíme "Odpojit" nebo "Odstranit" v případě flash disků a paměťových karet. Médium

můžeme nechat vysnout zvolením "Vysunout". Vše samozřejmě z menu pravého tlačítka.

Stane se, že nám systém oznámí, že zařízení není schopen odpojit. Je to způsobeno tím, že máte otevřený program, který k datům z média přistupuje. V takovém případě se médium odpojit nedá, dokud neukončíme práci s takovým programem. Toto chování zajišťuje konzistenci dat.

4.11 Systém oprávnění

Zkusím upustit od teorie a říci jen pár praktických poznámek. Systém oprávnění v GNU/Linuxu zajišťuje, aby uživatel nezlikvidoval omylem nebo úmyslně celý systém. Pracuje-li se pod běžným účtem, vše podléhá systému oprávnění. Uživatel je králem jen ve svém domovském adresáři, ale jinde ne. Z tohoto důvodu, budete-li pracovat jako běžní uživatelé, nemusíte se obávat, že něco poškodíte.

Při instalaci jste pravděpodobně zadávali heslo uživatele "root" (superuživatele). Tento uživatel je všemocný, tj. pro něj oprávnění neplatí, může všechno. Jako normální uživatel si můžete vypůjčit jeho oprávnění při spuštění aplikace provádějící správu systému - v grafickém prostředí budete požádáni o zadání patřičného hesla. Daný program pak bude pracovat s oprávněním uživatele root. Pouze v této době a pouze při práci s tímto programem můžete systém reálně ohrozit. Nebo pokud se jako root přímo přihlásíte, a to silně doporučuji nedělat. Je to nebezpečné a není to nutné.

4.12 Nouzové procedury

Pokud něco přestalo fungovat, můžete využít nouzovou proceduru, kterou jsem sestavil pro běžného uživatele. Následujte zvolené kroky podle situace, pokud možno v pořadí. Pomocí Alt-Tab se můžeme přepnout do jiného programu na stejné ploše. Můžeme se přepnout na jinou virtuální plochu, v KDE je to Ctrl-Fx (kde *x* je číslo od 1 do počtu virtuálních ploch).

Program můžeme ukončit pomocí Alt-F4. Vzpurný program můžeme ukončit pomocí programu xkill, který má v KDE běžně namapovanou zkratku Ctrl-Alt-Esc. Hnátu s lebkami namíříme na program k sestřelení a stiskneme levé tlačítko myši. Většinou se můžeme nouzově odhlásit pomocí Ctrl-Alt-Del. Grafické rozhraní je možné celé sestřelit pomocí Ctrl-Alt-Backspace. Můžeme se přepnout do textového režimu pomocí Ctrl-Alt-F1. V této situaci pomůže klávesová zkratka Ctrl-Alt-Del k vyvolání nouzového restartu.

Pokud jste došli až sem, znamená to, že došlo k vážnější poruše. Následující postup se možná bude zdát složitější, ale není na něm co zkazit. Respektive, zkazí se méně při jeho provedení než při studeném restartu rovnou. Mezi každým krokem je dobré pár sekund počkat. Dlužno dodat, že na některých distribucích je tato funkce vypnutá přímo v jádře (bezpečnostní opatření). Tak jako tak, lze ji alespoň vyzkoušet. Nuže:

- Alt-PrintScreen-R (uvolní klávesnici)
- Alt-PrintScreen-S (vyprázdní vyrovnávací paměti)

- Alt-PrintScreen-E (požádá programy o ukončení)
- Alt-PrintScreen-I (sestřelí všechny programy)
- Alt-PrintScreen-U (odpojí souborové systémy)
- Alt-PrintScreen-B (restartuje počítač)

Pořád nic? Pak je možné, že v jádře je tato funkce vypnutá, nebo jádro přestalo fungovat úplně. Teď už se nedá dělat nic jiného než restartovat stroj násilím (tlačítko RESET) nebo mašinu vypnout.

4.13 Rady do začátku

Pokud se rozhodnete nainstalovat a používat GNU/Linux, pak se vám do začátku určitě hodí několik postřehů, které jsem za dobu svého průzkumu problémů linuxových začátečníků učinil.

4.13.1 GNU/Linux není Windows

Jakkoliv se tato věta může zdát banální, vědomí o tomto faktu bývá často klíčem k zachování uživatelské přízně ke GNU/Linuxu. Pokud si člověk neuvědomí, že GNU/Linux nevychází z Windows, ale z Unixu, a spravuje jej stejnými postupy jako Windows, může narazit na problémy způsobené odlišností techto systémů a třeba jej zavrhnout, byť by bývalo stačilo použít linuxový postup a celou záležitost by realizoval v naprosté pohodě. Velmi častým příkladem je nepoužívání správce balíčků (viz dále). Osobně doporučuji aplikovat přístup "Jak na to v GNU/Linuxu?" místo častého a nešťastného "Proč to nejde jako ve Windows?".

4.13.2 Výběr správné distribuce

Bez správné distribuce to nejde. Každý uživatel má jiné preference a každý uživatel si od GNU/Linuxu bude slibovat něco jiného. Stejně tak bude chtít každý uživatel zajít v seznámení s GNU/Linuxem do různé hloubky. Někdo bude chtít jít “až na kost”, jinému postačí pohyb v grafickém rozhraní a spouštění aplikací. U Windows je člověk zvyklý na možnost vybírat maximálně několik prakticky shodných verzí lišících se v maličkostech, zatímco linuxových distribucí je hodně a bývají dost odlišné.

Obecně, existují distribuce vhodné pro začátečníky (user friendly) a distribuce vhodné pro pokročilé uživatele GNU/Linuxu (expert friendly). Pokud začínáme s GNU/Linuxem a nechceme jeho poznávání věnovat příliš času, bylo by vhodné zvolit uživatelsky přívětivou distribuci. Pouze pokud máme spoustu času a zájem poznat GNU/Linux do hloubky (a následně náležitě využít nabyté znalosti), použijeme některou z distribucí určených pro pokročilejší uživatele.

Dodávám, že výběr špatné distribuce bývá velmi častým, byť zbytečným, důvodem zavržení GNU/Linuxu. Proto se pokusme uvážit, jaké máme znalosti a předpoklady, časové zázemí a vůli a chuť se učit, a vyberme distribuci podle toho.

4.13.3 Číst dokumentaci

Bez dokumentace to opravdu nejde. Alespoň dokumentaci k vámi zvolené distribuci byste si měli projít. Samozřejmě čím více toho o GNU/Linuxu přečtete, tím lépe, ale když nic, tak alespoň distribuční dokumentaci. Tam najdete informace o specifikách distribuce, ale i různé tipy a triky, jak některé operace provádět rychleji a efektivněji. Zejména uživatelé přecházející z neunixového operačního systému potřebují vstřebat rozdíly mezi těmito systémy, a k tomu je dokumentace velmi důležitá.

4.13.4 Stávající operační systém ponechat

Opravdu není dobrý nápad (alespoň v začátcích) kompletně zrušit stávající operační systém, byť třeba zjistíme, že nám již nevyhovuje. Umíme se v něm totiž stále velmi snadno zorientovat, což nám pomůže v situaci, kdy honem nebudeme vědět, jak něco realizovat v GNU/Linuxu. Teprve až zjistíme, že nám původní operační systém už půl roku leží jen tak ladem a prostor, který zabírá, nám citelně chybí, začneme uvažovat o jeho kompletním zrušení.

4.13.5 Nepracovat pod rootem

Pokud jste z jiného systému zvyklí provádět běžnou práci pod administrátorským účtem (tj. pod rootem), v GNU/Linuxu to nedělejte. Opravdu to není dobrý nápad. V GNU-/Linuxu není práce pod uživatelským účtem nijak handicapovaná, protože je možné velice snadno získat rootovská oprávnění pro specifický úkon. Je dokonce běžné, že je uživatel v případě, že si z menu vybere nějaký konfigurační nástroj vyžadující práva uživatele root, automaticky požádán o rootovské heslo, a může tedy danou činnost bez problémů realizovat.

4.13.6 Neměnit distribuci v začátcích

Není to pouze specifikem českých diskusních fór (a poraden), ale velmi často se můžete setkat s tím, že vám někdo místo dobře mířené a přesné odpovědi na váš problém nabídne přechod na jinou distribuci (nezřídka stylem opravdu neslušným). Za takové příspěvky se vám předem jménem všech rozumných linuxových uživatelů omlouvám (to víte, i my máme mezi sebou tupce). Osobně doporučuji takové příspěvky ignorovat a až na výjimky neměnit v začátcích distribuci, kterou si vyberete.

Pravdou je, že žádná distribuce není zcela bezproblémová a každá má svá specifika, která je potřeba znát. Člověk, který vám bude chtít pomoci a ne si jen masírovat svoje ego, bude vaši svobodu volby zcela respektovat a pokusí se vám poradit, jak daný problém vyřešit v distribuci, kterou používáte. Bude si totiž velmi dobře uvědomovat, že jakmile byste přešli na jinou distribuci, setkali byste se s jinými problémy a nejpozději druhý den by od vás našel ve fóru či konferenci další dotaz.

Dobrou strategií do začátku je zvolit si distribuci a vydržet s ní nějakou dobu. Pokud narazíte na problémy, snažte se je vyřešit místo přechodu na jinou distribuci. Ten si schovejte na později, až budete Linux ovládat na dostatečné úrovni. Do té doby bývá

přechod na jinou distribuci kontraproduktivní, protože vás ještě více zmate a může s sebou přinést další problémy, které bude třeba řešit.

Na druhou stranu, v určitých situacích se tomu nevyhnete, například v situaci, kdy si vyberete distribuci, která se pro vás nehodí (příliš složitá nebo příliš klikací), popřípadě když budete vědět, že ta vaše má nějaké problémy, které jsou pro vás fatální, a ona druhá je nemá. To ale jistě poznáte sami.

4.13.7 Netřeba se všechno učit hned

GNU/Linux je poměrně komplexní operační systém, navíc zcela otevřený a dobře zdokumentovaný. V uživateli leckdy množství dokumentace a okolnosti vyvolávají pocit, že to všechno bude muset projít, aby systém mohl používat, ale tak tomu není. Jako uživatelé si můžete vybrat, kam až v seznamování s GNU/Linuxem budete chtít zajít. Můžete zůstat na úrovni klikání v grafickém rozhraní nebo pokročit až na úroveň zdrojového kódu, volba je čistě a pouze na vás.

4.13.8 Preferovat správce balíčků

V GNU/Linuxu je správa softwaru řešena centrálně, prostřednictvím správce balíčků. A to je přesně způsob, který by měl začátečník preferovat a, je-li možno, neinstalovat software jinak, je to složitější a má to svá úskalí. Většinou to ale není vůbec nutné, protože prakticky většina softwaru, který kdy budete chtít použít, se nainstaluje už během instalace, nebo je k dispozici prostřednictvím správce balíčků. Práce s ním je podrobně popsána v kapitole *Správa softwaru*.

4.13.9 Upřednostňovat nativní aplikace

Někdy se dostanete do situace, kdy budete mít na jedné straně možnost používat aplikaci určenou pro jiný operační systém v emulátoru a na straně druhé budete mít možnost používat nativní aplikaci pro GNU/Linux. Z hlediska praxe je vhodnější preferovat v tomto případě ekvivalentní nativní aplikaci pro GNU/Linux.

4.13.10 Nevzdávat se předčasně

Nebudu se pokoušet vám tvrdit, že GNU/Linux je zcela bezproblémový operační systém. Na nějaký problém nepochybně dříve či později narazíte. Skeptik by v této souvislosti mohl rozdíl mezi GNU/Linuxem a jiným operačním systémem vidět v různé škále problémů, které budete řešit. V této souvislosti nemohu než doufat, že se nevzdáte předčasně, pokud na nějaký problém narazíte. Přeci jen, když už ujdete tak dlouhou cestu a chcete vyzkoušet jiný operační systém, měli byste mu dát férovou šanci.

4.14 Co dál?

Pokud vás zajímá, jak GNU/Linux funguje, můžete si pročíst kapitoly zaměřené na architekturu a správu GNU/Linuxu. Určitě si pročtěte kapitolu označenou jako *Správa*

softwaru, určitě se vám hodí, až budete chtít nainstalovat nějaký program nebo aktualizovat distribuci.

4.14.1 Multimédia

Pokud zjistíte, že vaše distribuce nepodporuje přehrávání určitých multimediálních souborů, je to otázka doinstalování příslušných kodeků. Za tímto účelem nahlédněte do distribuční dokumentace nebo se podívejte do diskusních fór. Nepochybně naleznete podrobný návod, jak podporu proprietárních formátů doplnit.

4.14.2 Doplňující dokumentace

Projděte si zdroje a odkazy zejména pod touto kapitolou, kde naleznete mj. odkazy na dokumentace podobné této.

4.14.3 Odkazy

Poslední kapitolu tvoří rozsáhlá databáze odkazů na weby o GNU/Linuxu, ať už portály, dokumentační weby, distribuční weby, vyhledávací služby, apod.

4.15 Zdroje a odkazy

- ABC Linuxu, čtenáři, Učebnice GNU/Linuxu
- Petr Mach, Úvod do systému Linux
- Covex, Otázky uživatelů, kteří se začínají zajímat o Linux
- Dokumentační projekt http://www.tldp.org/
- Dokumentační archív http://docs.linux.cz/
- Tutoriály na Rootu, http://tutorialy.root.cz/
- Really Linux (anglicky), http://www.reallylinux.com/

Kapitola 5

Architektura GNU/Linuxu

Pokud chceme efektivně spravovat GNU/Linux, musíme porozumět tomu, jak vlastně funguje. Musíme se podívat "pod kapotu" a pochopit, jaké principy leží pod slupkou grafických rozhraní a pomocných grafických nástrojů pro konfiguraci. Vzhledem k tomu, že GNU/Linux je unixový systém, očekávejte v dokumentaci velmi časté zmiňování o "unixových systémech", jelikož se převážně jedná o obecně platné principy.

Základní charakteristika GNU/Linuxu je modulárnost. To znamená, že se sestává z mnoha relativně nezávislých částí. Pro většinu z nich navíc existují alternativy, a proto vždy záleží na správci distribuce, jak systém sestaví. Modulárnost také vysvětluje rozdíly mezi jednotlivými distribucemi. I když je vše GNU/Linux, odlišnosti bývají natolik markantní, že při přechodu na jinou distribuci může být uživatel minimálně lehce zmaten. Já vám budu sdělovat informace nezávislé na zvolené linuxové distribuci.

Modularita vychází z unixové filozofie. Ta je založena na existenci mnoha jednoduchých programů, které provádí jednu věc, ale provádí ji velice dobře. Jejich kombinací prostřednictvím příkazové řádky lze dosáhnout přesně toho, co člověk potřebuje. Tyto prográmky tvoří základ každého unixového systému a i když je člověk nechce využít, je dobré alespoň vědět, že tam jsou.

Je jasné, že GNU/Linux už dávno není pouze o příkazové řádce. Je to komplexní systém nabízející přívětivá grafická rozhraní a klasická integrovaná řešení (jeden program realizující mnohé). Podstatné je, že to vše stojí na pevném unixovém základě a je jenom na uživateli, jestli bude pracovat v pohodlí grafického rozhraní, jestli se posune níže nebo jestli obojí zkombinuje. Volba je na vás.

Většina záležitostí se v unixovém systému realizuje prostřednictvím souborového systému. Narozdíl od jiných systémů, v unixových systémech existuje pouze jediná adresářová struktura, do které se připojují všechy další souborové systémy (tj. média, síťové disky, flash disky, apod.). Práce se systémem i práce s hardwarem je de facto prací se soubory, neboť v unixových systémech platí heslo "vše je soubor", tudíž v těchto systémech je i vaše tiskárna či pevný disk soubor, se kterým lze zacházet prostřednictvím obslužných programů, nebo přímo pomocí unixových nástrojů.

Unixová architektura je jednoduchá, vnitřně konzistentní, a umí velmi dobře využívat svých výhod. Kupříkladu, pravidlo "vše je soubor" koresponduje s výše zmíněným pravidlem o mocných nástrojích, které mj. velmi dobře pracují se soubory. To je důvo-

dem, proč jsou tyto systémy stále tolik populární.

Navíc už od počátku byly tyto systémy navržené jako víceúlohové a víceuživatelské, a proto s nimi může bez problémů a bezpečně pracovat více uživatelů, třeba i najednou. Uživatele jednouživatelského počítače bude z těchto vlastností zajímat ona zmiňovaná bezpečnost. Tu zajišťuje systém oprávnění, který dává uživatelům pocit jistoty, že dokud nezískají oprávnění uživatele root (superuživatele), nemohou systém poškodit.

5.1 Od hardwaru až k aplikaci

Hierarchie komponent GNU/Linuxu začíná jádrem systému, tedy kernelem. Ten zodpovídá za ty nejzákladnější funkce, inicializace hardware a jeho prostředky poskytuje programům podle specifických pravidel. Zároveň hlídá oprávněnost přístupu k jednotlivým prostředkům, a zajišťuje tak bezpečí samotného systému. Programům poskytuje svoje programátorské komunikační rozhraní zvané API.

Nad jádrem stojí knihovny v čele s libc (knihovna poskytující ty nejzásadnější funkce, součást projektu GNU). Knihovna je soubor funkcí, které mohou programy volat. Díky tomu si s sebou prohlížeče obrázků nemusí vláčet schopnost zpracovat obrázek PNG. Prostě zavolá patřičné funkce knihovny libpng, která vše zařídí. Toto oddělení funkcionality má nesporné výhody. Pokud někdo nalezne způsob, jak zpracovávat tyto obrázky rychleji, stačí mu pozměnit knihovnu a všechny programy, které pracují s těmito obrázky, tuto vlastnost okamžitě přebírají.

Má to ovšem i stinné stránky. Knihovny se vyvíjejí a svoje komunikační rozhraní mění. Může se stát, že starší program bude vyžadovat starší verzi knihovny. V GNU/Linuxu s tím není problém, protože číslo její verze je součástí názvu patřičného souboru, takže můžete mít v systému současně nainstalováno více verzí jedné knihovny. Dále je jasné, že výše zmíněný prohlížeč obrázků bez správné knihovny odmítne fungovat. Proto je nutné zajistit, aby měl program k dispozici všechny potřebné knihovny.

Knihovny tedy stojí mezi jádrem a programy, ale samotné programy mohou využívat rozhraní jádra taktéž. Je jasné, že některé programy mohou pro svou funkci potřebovat další programy. Kupříkladu, známý vypalovací program K3B pro svou funkci vyžaduje řádkové nástroje pro vypalování. Vyžaduje ale i grafické prostředí a knihovnu qt. Program, který stojí přímo nad jádrem a knihovnami, a který nepotřebuje jiné programy, se nazývá nízkoúrovňový. Naopak program, který vyžaduje jiné programy je vysokoúrovňový.

Nezbytným vazbám mezi konkrétními komponenty se říká říká závislosti. Ty zpravidla řeší místo uživatele správce balíčků.

5.2 Jádro, kernel, Linux

Primární komponentou systému GNU/Linux je kernel, jádro operačního systému, Linux. Zajišťuje správu procesů (alias běžících programů), správu paměti, multitasking (práce více úloh současně) a TCP/IP síťování.

Linux je monolitické jádro, takže veškerá jeho funkcionalita je uložena v jednom celku, ovšem s rozšířením o moduly (ovladače hardwaru a další funkcionalita), které lze do jádra vložit a opět z něho vyjmout, a to přímo za běhu. Jádro běží v plně privilegovaném režimu procesoru (tj. s plným přístupem k hardwaru). Naopak všechno ostatní běží v plně neprivilegovaném režimu a musí o přístup k danému prostředku požádat jádro, které pečlivě kontroluje, má-li daný program příslušné oprávnění.

5.2.1 Kernel panic a oops

Při běhu jádra může dojít k chybovému stavu. Ty jsou podle fatálnosti dva, kernel oops a kernel panic. Kernel oops je nefatální chyba, ze které se kernel zotaví, i když při tom dojde k určitým škodám na běžícím systému, některé prostředky již nemusí být k dispozici. Informace o chybovém stavu bývají dohledatelné v systémovém logu (ve `/var/log`). Kernel panic oznamuje fatální chybu, ze které se jádro nemá šanci zotavit.

Tyto chyby mohou být způsobeny buď skutečnou programátorskou chybou v jádře, ale mohou být také projevem špatně fungujícího hardwaru nebo ovladače. Typickým problémem je chyba v paměti RAM.

5.2.2 Číslování verzí jádra

Verze jádra má tvar *A.B.C.D*, kde *A* je verze jádra, *B* je hlavní revize jádra, přičemž liché číslo značí vývojovou a sudé stabilní revizi, *C* je vedlejší revize, která se mění s provedením nějaké podstatnější změny (přidání ovladače, apod.), a konečně *D* je číslo, jehož povýšení naznačuje drobnou změnu (opravu chyby, apod.). Momentálně existují dvě řady jádra, starší *2.4* a novější *2.6*. Aktuální verzi jádra zjistíte třeba z webu kernel.org, odkud si můžete samotné jádro stáhnout.

5.2.3 Vanilla a distribuční jádra

Jádro lze různými způsoby upravovat, lze na něj aplikovat patche, které do něj přidávají další (neoficiální) funkcionalitu. Čistému jádru bez patchů se říká vanilla jádro a lze jej získat přímo ze stránek kernel.org. Jádra, která se nachází v distribucích, bývají patchovaná a upravená třeba tak, aby distribuce fungovala na téměř jakémkoliv hardwaru.

5.2.4 Kompilace jádra

Jádro si můžete sestavit sami, odstranit z něj podporu zařízení, která nepoužíváte, a přizpůsobit jej přesně pro váš počítač. Můžete sami rozhodnout, kterou funkcionalitu zkompilujete jako modul a kterou pevně zakotvíte v jádře. Můžete sami aplikovat patřičné patche, které uznáte za vhodné. A samozřejmě, můžete kód jádra upravit sami. Kompilaci jádra bych doporučil pouze v případě, kdy k tomu máte dobrý důvod.

5.2.5 Iniciální ramdisk (initrd)

Už jsme si řekli, že některá funkcionalita a některé ovladače mohou být v jádře přístupné jako moduly. Moduly se samozřejmě nacházejí mimo jádro, jsou to obyčejné soubory. Je jasné, že k tomu, aby se jádro mohlo dostat k modulu, potřebuje připojit příslušný oddíl, na kterém jsou moduly uloženy. Má-li však třeba ovladač řadiče nebo souborového systému zakompilovaný jako modul, nemá šanci se k němu dostat. V takovém případě ohlásí kernel panic a odmítne pokračovat.

Tuto situaci řeší malý komprimovaný souborový systém obsahující mj. příslušné moduly, který se zkopíruje do paměti za jádro, jádro jej rozbalí a připojí na kořenový adresář. Z něj pak natáhne příslušné moduly a může pokračovat dále.

5.2.6 Předávání parametrů jádru

Parametry se jádru předávají pomocí zavaděče (LILO nebo GRUB). Ty mu třeba řeknou, který oddíl má připojit na kořenový adresář, které činnosti má a nemá provádět, jak velikou má očekávat paměť, apod. Parametry jádra tedy umožňují upravovat jeho funkci a vyřešit některé problémy se špatně fungujícím hardwarem (tak, že danou funkcionalitu vypnou).

5.3 Soubory a adresáře

Soubory a adresáře se nachází v rámci souborových systémů, které jsou vytvořeny v oddílech pevných disků nebo výměnných zařízeních. V unixech jsou jednotlivé souborové systémy připojovány na adresáře, takže vše se jeví jako jedna veliká adresářová struktura. Nejenom z toho důvodu má adresářová struktura v unixových systémech specifická pravidla, kde se co nachází. Ale začněme od začátku.

5.3.1 Souborové systémy, diskové oddíly a pevné disky

Souborový systém je sada pravidel pro ukládání souborů a adresářů na pevný disk tak, aby bylo možné je opět přečíst. Bývá v ní vymezeno, kde na pevném disku se daný soubor nachází, jak se jmenuje, v jakém je adresáři a jaká má přístupová práva. Souborové systémy existují na pevných discích v oblasti definované jako tzv. diskový oddíl. Těch může mít pevný disk více. Výměnná zařízení obsahují zpravidla pouze jediný souborový systém, nejsou členěny na oddíly.

V unixových systémech jsou souborové systémy na oddílech pevných disků i na jiných zařízeních připojovány na adresáře. Základ tvoří kořenový souborový systém, který se připojí na kořenový adresář, a do jeho struktury se pak zapouštějí (připojují) další souborové systémy podle požadavků uživatele. Připojování souborových systémů může probíhat automaticky (např. po vložení média), nebo může být vyžadováno provést ruční připojení.

5.3.2 Soubory

V unixových systémech platí, že vše je soubor. Na rozdíl od jiných systémů nerozděluje GNU/Linux název souboru na jména a příponu. Přípona, pokud je, je prostě součástí jména souboru. Unixové systémy také rozlišují velká a malá písmena v názvech souborů, takže, kupříkladu, soubor se jménem "novak" je něco jiného než soubor "Novak". Soubor (adresář) se jménem začínajícím tečkou je považován za skrytý soubour, na což reagují příslušné programy tak, že jej nezobrazí. K zobrazení takových souborů je zpravidla nutné použít adekvátní volbu v nastavení příslušného programu. Rozlišujeme následující typy souborů:

- obyčejný soubor
- adresář (v terminologii MS Windows "složka")
- symbolický odkaz
- pevný odkaz
- znakové zařízení
- blokové zařízení
- pojmenovaná roura
- socket

5.3.2.1 Adresář (directory)

Adresář je vlastně určitý typ souboru, který obsahuje odkazy na jiné soubory. Tyto odkazy musí být alespoň dva, odkaz na sama sebe (reprezentovaný tečkou) a odkaz na předchozí adresář (reprezentovaný dvěma tečkami).

5.3.2.2 Symbolický odkaz (symbolic link)

Symbolický odkaz je de facto soubor s odkazem na jiný soubor. Vztah mezi odkazem a původním souborem řídí jádro, takže všechny programy se k symbolickým odkazům chovají stejně. Respektive, pokusíte-li se otevřít odkaz příslušným programem (třeba textovým editorem, nebo prohlížečem či jiným programem), jádro zachytí patřičný požadavek, a jelikož ví, že se jedná o odkaz, provede přesměrování na původní soubor. Takže vám se zobrazí obsah souboru, na který odkaz odkazuje. Při kterékoliv jiné operaci (kopírování, přesouvání, mazání, apod.) se změna dotkne pouze odkazu samotného, a nikoliv původního souboru. Symbolický odkaz může odkazovat i na neexistující (nebo dočasně neexistující) soubor.

5.3.2.3 Pevný odkaz (hard link)

Pevný odkaz je vlastně jiný název pro stejný soubor. Stejná data jsou přístupná pod více soubory. Na rozdíl od symbolického odkazu je tedy omezen na stejný souborový systém. Při vytvoření pevného odkazu se zvýší patřičný počet odkazů v záznamech souborového systému. Rozlišení na původní soubor a odkaz pozbývá smyslu, protože oba soubory vedou ke stejným datům. Data se však nekopírují, prostě oba soubory odkazují na stejné místo na pevném disku. Pokud smažete pevný odkaz, klesne počet odkazů na příslušná data. Klesne-li na nulu (v případě, že smažete poslední odkaz), dojde k uvolnění souboru ze souborového systému (vymazání).

5.3.2.4 Znaková a bloková zařízení

Zařízení jsou speciální soubory, kde práce s nimi je jádrem zachycena a provedena na zařízení, které reprezentují. Dělí se na znaková a bloková, přičemž bloková jsou schopna náhodného přístupu (třeba pevný disk - přečti sektor 24), zatímco znaková zařízení pracují s proudy dat (tiskárna, scanner, apod.). Tyto speciální soubory jsou umístěny v adresáři `/dev` a z jejich symboliky bych rád probral alespoň symboliku médií (hvězdička supluje libovolnou chybějící část):

- IDE zařízení - `hd*` (pevné disky i CD mechaniky)
  - primární zařízení na prvním řadiči - `hda`
  - sekundární zařízení na prvním řadiči - `hdb`
  - primární zařízení na druhém řadiči - `hdc`
  - sekundární zařízení na druhém řadiči - `hdd`
  - oddíly: první oddíl `hda1`, druhý oddíl `hda2`, třetí oddíl `hda3`, atd.
- SCSI/SATA zařízení - pevné disky `sd*`, CD mechaniky `scd*`
  - primární zařízení na prvním řadiči - `sda`
  - sekundární zařízení na prvním řadiči - `sdb`
  - primární zařízení na druhém řadiči - `sdc`
  - sekundární zařízení na druhém řadiči - `sdd`
  - oddíly: první oddíl `sda1`, druhý oddíl `sda2`, třetí oddíl `sda3`, atd.
  - rozlišení jednotlivých CD mechanik: první mechanika `scd0`, druhá `scd1`, atd.
- disketová mechanika - `fd*`
  - rozlišení jednotlivých mechanik: první mechanika `fd0`, druhá mechanika `fd1`, atd.

Mezi další zajímavá zařízení patří:

- `/dev/random` - generátor náhodných čísel

- `/dev/urandom` - generátor pseudo-náhodných čísel (je rychlejší, ale "méně náhodný")
- `/dev/null` - černá díra, cokoliv sem přesunete, to zmizí
- `/dev/zero` - generátor nul

Zařízení jsou rozpoznávána pomocí dvou čísel identifikujících příslušný hardware, tzv. major a minor. Názvy souborů zařízení jádro v úvahu nebere.

5.3.2.5 Pojmenovaná roura (named pipe)

Pojmenovaná roura je způsob, jakým si vyměňují informace procesy (běžící programy). Jejich použití je automatické, pro uživatele nemají význam. Velký význam mají naopak pro programátory. Tyto pojmenované roury se často vyskytují v adresáři `/tmp`, a proto je krajně nevhodné jej za běhu systému mazat, protože tak běžícím programům utnete jejich vzájemné komunikační kanály.

5.3.2.6 Socket

Sockety jsou záležitostí programátorů, ve zkratce, jsou to soubory, které reprezentují síťové spojení.

5.3.3 Adresářová struktura

Adresářová struktura začíná kořenovým adresářem, který se označuje normálním lomítkem. Následují jednotlivé adresáře a jejich popis:

5.3.3.1 `/bin`

Nepostradatelné spustitelné soubory využívané všemi uživateli. Nalezneme tu mj. interprety příkazové řádky (shelly) a základní řádkové nástroje pro práci se systémem.

5.3.3.2 `/boot`

Obrazy jader (zpravidla soubory začínající na vmlinuz) a iniciální ramdisky.

5.3.3.3 `/dev`

Soubory zařízení, tedy speciální soubory, které reprezentují jednotlivá zařízení.

5.3.3.4 `/etc`

Konfigurační soubory, přesněji struktura konfiguračních souborů. V podadresáři `init.d` (v některých distribucích je to `rc.d`) se ukrývají skripty potřebné pro spouštění a vypínání systémových služeb (v unixových systémech se jim říká démoni).

5.3.3.5 `/home`

Domovské adresáře uživatelů. Podadresář s vaším uživatelským jménem je vaše území, kde můžete provádět skoro cokoliv.

5.3.3.6 `/lib`

Ty nejdůležitější knihovny potřebné pro běh systému. V podadresáři modules se nachází moduly jádra.

5.3.3.7 `/media`, `/mnt`

Adresáře, kde bývají připojena média (CD, DVD, flash disky, diskety) a další diskové oddíly.

5.3.3.8 `/proc`

Speciální soubory, které tvoří komunikační rozhraní s jádrem. Zde se můžete dozvědět mnohé o činnosti jádra a vhodnými změnami příslušných souborů můžete upravovat jeho funkci.

5.3.3.9 `/opt`

Adresář vhodný pro instalaci softwaru, který není původně vytvořen pro unixové systémy a neumí využívat jeho adresářovou strukturu.

5.3.3.10 `/root`

Domovský adresář uživatele root.

5.3.3.11 `/sbin`

Nepostradatelné spustitelné soubory určené výhradně pro uživatele root.

5.3.3.12 `/usr`

Tzv. sekundární hierarchie, obsahuje mj. podadresáře `bin`, `sbin`, `lib` a další, které se nachází v kořenovém adresáři. Tyto ale nejsou nutné k běhu systému, hnízdí v nich uživatelské programy. Podadresář `local` slouží jako prostor k instalaci softwaru mimo balíčkovací systém. Podadresář `share` obsahuje zpravidla datové soubory aplikací. Podadresář `share/doc` nebo `doc` obsahuje dodatečnou dokumentaci k jednotlivým programům.

5.3.3.13 `/var`

Proměnlivá data, soubory, u kterých se očekávají změny. Zde je nutné zajistit, aby vždy bylo k dispozici nějaké volné místo. Bez toho nebude systém fungovat správně. Podadresář `log` obsahuje systémové protokoly, soubory s informacemi o tom, co se v systému událo během poslední doby. Velmi cenný informační zdroj.

5.3.3.14 `/tmp`

Dočasné soubory. Stejně jako v případě adresáře `/var` je nutné zajistit dostatek volného prostoru. Pokud místo dojde, systém přestane fungovat správně. Soubory zde uložené rozhodně nemažte při běhu systému! Byť jsou dočasné, mohou být pro právě spuštěné programy klíčový význam (viz "pojmenovaná roura" výše).

5.3.4 Adresářová struktura a software

Jak je z této struktury patrné, nejsou programy umístěny vždy v jednom adresáři, kde mají svoje spouštěcí i datové soubory, ale jejich spouštěcí soubory se nachází v `/bin` či `/usr/bin`, sdílené knihovny mají k dispozici v `/lib` či `/usr/lib`, konfigurační soubory mají zpravidla v `/etc` a jejich datové soubory se nachází v `/usr/share`. Pokud máte software, který se s takovou hierarchií nevypořádá, pak můžete použít adresář `/opt`, který je tomuto účelu vyhrazen. Teď vás možná napdá, jak odlišit, co patří jednotlivým programům. To má na starosti balíčkovací systém, který ví, co patří ke kterému programu a je schopný selektivně odstranit daný program tak, že po něm nezbyde vůbec nic, přičemž je schopen se vypořádat se závislostmi.

5.4 Uživatelé a skupiny

Každý uživatel je v systému veden pod určitým číslem, kterému se příká UID. Může být zařazen do jedné nebo více skupin, které jsou v systému vedené také pod určitým číslem, kterému se říká GID. Skupiny slouží v rámci systému přístupových práv, aby uživatelé mohli pracovat na společných projektech bez toho, aby dávali práva ke svým datům všem uživatelům.

Domovské adresáře uživatelů nalezneme v adresáři `/home`. Kromě domovského adresáře má uživatel jen velmi omezená práva měnit cokoliv v systému, tudíž i zapisovat do mnoha adresářů. Jeden z adresářů, do kterých kterýkoliv uživatel zapisovat smí, je `/tmp`. Speciálním případem je uživatel root, přezdívaný superuživatel. Pro něj žádná omezení neplatí, on může vše.

Přidání nového uživatele a odebrání stávajícího lze realizovat v pohodlí grafického rozhraní snad ve všech distribucích. Informace o uživatelích a skupinách naleznete v následujících konfiguračních souborech:

- `/etc/passwd` - základní informace o uživatelích: login, heslo (bývalo dříve, nyní je v `/etc/shadow`), uid, gid, jméno uživatele, adresář, implicitní shell
- `/etc/shadow` - obsahuje mj. md5 hashe hesel, takže, na rozdíl od `/etc/passwd`, není čitelný obyčejnými uživateli

- `/etc/group` - obsahuje informace o skupinách a kdo do nich patří

Při založení nového uživatele se zpravidla z adresáře `/etc/skel` zkopíruje vše do uživatelova domovského adresáře. Po přihlášení uživatele do příkazové řádky se mu zobrazí obsah souboru `/etc/motd`.

5.5 Přístupová práva

Model přístupových práv v unixových systémech je velmi jednoduchý. Každý soubor má svého vlastníka a skupinu, do které patří. Oprávnění mohou být různá pro vlastníka, skupinu a ostatní uživatele. Přístupová práva jsou tři, právo číst (`r`, jako read), právo zapisovat (`w`, jako write) a právo vykonat kód, tedy spouštět (`x`, jako eXecute).

Mám-li soubor s vlastníkem "michal", skupinou "projekt" a oprávněními `rwxr--xr--`, znamená to, že vlastník má právo soubor číst, zapisovat do něj a spustit jej. To nám říkají první tři znaky tohoto symbolického zápisu. Další tři znaky nám říkají, co mohou se souborem provádět uživatelé patřící do skupiny "projekt". Ti mohou soubor číst a spouštět jej, nemají však právo do něj zapisovat. Jak si můžete všimnout, místo práva `w` je v symbolickém zápisu pomlčka, která nám říká, že dané právo přiděleno nebylo. Ostatní uživatelé, soudě dle posledních třech znaků symbolického zápisu, mají pouze oprávnění soubor číst. K zápisu a ke spuštění oprávnění nemají.

5.5.1 Práva adresářů

Jak jsme si řekli, adresář je speciální typ souboru, který obsahuje odkazy na jiné soubory. Proto na něj mají práva poněkud jiný účinek. Právo čtení nám umožní přečíst si jeho obsah, tj. jaké soubory se v něm nachází. Právo zápisu nám umožní vytvářet či mazat soubory, které se v něm nachází. Ano, když soubor ochráníme proti zápisu i čtení, ale je v adresáři, do kterého má jiný uživatel povolený zápis, může soubor vymazat. Výmaz souboru je v podstatě zápis do příslušného adresáře, kde se odstraní položka ukazující na daný soubor. Existuje však možnost, jak i to ošetřit, viz "Sticky bit" níže. Právo ke spuštění nám umožní adresář otevřít.

5.5.2 Speciální práva

Kromě výše uvedených práv existují ještě tři takové malé "hacky", sticky bit, setuid bit a setgid bit.

5.5.3 Sticky bit

Jak už jsem podotkl, na výmaz souboru stačí mít oprávnění k zápisu do adresáře, kde se nachází. Pak i v případě, že bude daný soubor proti nám chráněn sebelépe, stejně jej budeme moci vymazat. Tato situace určitě není přijatelná, a proto existuje způsob, jak tomu zabránit. Tím způsobem je nastavit danému adresáři sticky bit. V takto ošetřeném adresáři bude moci soubory mazat jenom jejich vlastník.

5.5.4 SetUID a SetGID

Pro některé operace může být z nějakého důvodu třeba vyšších práv, než má uživatel k dispozici. V případě vypalování je tím důvodem přímý přístup k mechanikám. Pokud uživatel tato oprávnění nemá, ale přesto chce danou činnost provádět, může být řešením právě setuid nebo setgid oprávnění, které způsobí, že se při spuštění daného programu nastaví jeho "výkonný" vlastník (setuid) nebo skupina (setgid) na vlastníka souboru. Pokud je jím root, získá program jeho oprávnění a má mj. plný přístup k hardwaru. Ale tato oprávnění získá pouze daný program při svém provádění, uživatel, který jej spustil, je nezíská. Pomocí těchto oprávnění je tedy možné, aby uživatel pracoval bez oprávnění uživatele root, a přesto prováděl operace, které vyžadují vyšší oprávnění, než má k dispozici.

5.6 Procesy

Běžícím programům se říká procesy. GNU/Linux je (stejně jako ostatní unixové systémy) víceúlohový systém, což znamená, že v něm může najednou běžet více procesů (na jednom či více procesorech). Procesy jsou v systému identifikovány číslem zvaným PID (process ID). Rodičem všech procesů je init (`/sbin/init`, PID=1), který je zodpovědný za nastartování celého systému.

5.6.1 Řízení procesů

Procesy řídí jádro, které jim přiděluje systémové prostředky. Čas procesoru je procesům přidělován podle jejich "niceness" (slušnost). Ta může nabývat hodnot od `-20` do `19`, přičemž čím je číslo nižší, tím je na běh programu kladen větší důraz a je mu přidělováno více času procesoru. Pouze uživatel root je oprávněn nastavit procesům slušnost menší než nula.

I když je systém plně zažížen, jádro vždy uspokojuje všechny procesy, takže i když vám zrovna běží nějaký žrout obecný se slušností `-20`, dostane se časem ke slovu i slušňák devatenáctého stupně.

5.6.2 Signály

Procesům se dají posílat signály, které mohou program donutit chovat se určitým způsobem či provést určitou operaci. Z hlediska uživatele jsou nejpodstatnější dva signály, signál TERM a signál KILL. První požádá program o ukončení, druhý ho bez milosti sestřelí. Pro uživatele není až tak podstatné se učit význam signálů, ale programátor by je znát měl. V souvisejících odkazech naleznete na toto téma velmi dobrý článek.

5.6.3 Démoni

Démoni jsou procesy běžící na pozadí, ty, které uživatel nemá přímo pod svou kontrolou, neovládá je interaktivně, ale které zajišťují určitou činnost. Jejich ovládací skripty (umožňující je spustit, zastavit či restartovat) se nachází zpravidla v `/etc/init.d`.

5.7 Uživatelská rozhraní

Dnes existují všeho všudy dvě uživatelská rozhraní, textové (příkazová řádka) a grafické. GNU/Linux nabízí obojí a u každého typu ještě řadu možností k výběru.

5.7.1 Textový režim, shell

Základním rozhraním GNU/Linuxu (tedy tím, které by mělo fungovat vždy a všude), je příkazová řádka. Shell je interpret příkazové řádky, program, který zpracovává příkazy, které píšete. Těch má GNU/Linux celou řadu, ale tím nejpoužívanějším je Bash, inovovaná verze unixového Bourne Shellu. Existují samozřejmě i jiné shelly.

V příkazové řádce je možné realizovat mnohé, a to rychle a efektivně. Bohužel, tyto možnosti jsou vyváženy nutností se s tímto nástrojem naučit efektivně pracovat, a to chvilku trvá. Je na vás, jestli tomu ten čas dáte, nebo budete preferovat grafické rozhraní.

5.7.2 Grafické rozhraní

V unixových systémech tvoří grafické rozhraní tzv. X-Window System. Je to software s architekturou klient/server a využívá síťové rozhraní (je tedy možné přihlásit se vzdáleně k jinému systému, ale tento systém stejně dobře funguje i na počítači odpojeném od sítě). Serverem je X-Server (nebo familiérně Xka) a poskytuje ty nejzákladnější funkce (zobrazovací funkce, polohovací zařízení, klávesnice). Samotná grafická prostředí a okenní manažeři vystupují v roli klientů, kteří se připojí k "serveru" a využívají jeho funkce.

5.7.3 Grafický desktop

Grafický desktop je velmi pravděpodobně to, o co budete stát nejvíce, plně vybavené grafické prostředí spravující okna i plochu, tvořící komplexní prostředí. Jejich výhodou je množství funkcí, nevýhodou pak těžkopádnost a náročnost na systémové zdroje (paměť, procesor). Příklady grafických desktopů jsou KDE a Gnome.

5.7.4 Okenní manažer (window manager)

Kromě grafických desktopů existují i odlehčená prostředí, přesněji správci oken. Ty se třeba nestarají o plochu a pozadí, spravují základní operace s okny a nabízí jednoduchá menu. Jejich konfigurace je zpravidla otázkou editace konfiguračních souborů. Jejich výhodou je lehkost, svižnost, nenáročnost na systémové zdroje. Taková prostředí poběží dobře i na starších počítačích. Jejich nevýhodou je naopak nedostatek funkcí a někdy obtížná konfigurace (ruční úprava konfiguračních souborů). Příklady takových prostředí mohou být Fluxbox, IceWM, twm, apod.

5.8 Běh systému

5.8.1 Spuštění systému

Po zapnutí počítače se spustí program obsažený na čipu v základní desce s názvem BIOS. Ten se podívá, co za hardware se vlastně v počítači nachází, provede základní diagnostiku, najde zařízení označené pro zavádění systému (zpravidla je to "primary master" pevný disk) a předá řízení zavaděči (bootloader).

Prvním 512 bytům na začátku pevného disku se říká MBR (Master Boot Record). Z nich prvních 446 bytů tvoří zavaděč (bootloader), dalších 64 bytů tvoří tabulka rozdělení disku (partition table) a zbylé dva byty tvoří tzv. magic number. Zavaděč se může vejít do oněch 448 bytů (např. LILO), nebo nemusí, a v takovém případě se zavaděč pouze odkazuje na nějaké místo na pevném disku, kde se nachází hlavní zaváděcí program (GRUB).

Zavaděč natáhne do paměti linuxový kernel (jádro operačního systému), a ten spustí. Kernel si osahá hardware, připojí kořenový oddíl a spustí program /sbin/init, který zjistí, do které úrovně běhu má nastartovat (z /etc/inittab) a to pak také provede.

5.8.2 Úrovně běhu (runlevely) a startovací skripty

V adresáři `/etc/rcX.d` (kde *X* je číslo runlevelu) se nachází symbolické odkazy na skripty služeb v `/etc/init.d`, které jsou postupně spouštěny podle pořadí.

GNU/Linux má celkem šest úrovní běhu, do kterých lze nabootovat nebo je při běhu měnit příkazem init cislo_runlevelu. Úroveň běhu specifikuje, které služby se mají zavést, tedy co všechno má být funkční. Ve všech distribucích platí, že úroveň běhu 0 se rovná vypnutí počítače, zatímco úroveň běhu 6 se rovná restartu počítače.

Jednička představuje jednouživatelský režim bez sítě pro provádění záchranných operací. Úroveň běhu můžete specifikovat i jako parametr jádru v rámci zavaděče. Parametr 1 nebo single by vás měl systém nastartovat do úrovně běhu 1. Úrovně běhu 2-5 bývají v různých distribucích různě definované.

5.9 Síť a servery

5.9.1 Síť

Práce v síťovém prostředí je pro unixové systémy přirozeností, dokonce řada programů (včetně X-Serveru) využívá síť, a to i na počítači, který není k žádné připojen. K tomu se využívá místní smyčka (tzv. local loopback), IP adresa je `127.0.0.1`, a je to reprezentace místního počítače (toho vašeho nepřipojeného). Místní smyčka je pro fungování mnoha programů nezbytná, naopak bez připojení ke klasické síti se GNU/Linux bez problémů obejde.

5.9.2 Druhy připojení

Nejběžnější připojení je přes ethernet, tedy síťová karta ve vašem počítači a kabel vedoucí třeba k síťové přípojce, speciálnímu modemu nebo jinému počítači. Identifi-

kovaná síťová karta bude zprovozněna jako rozhraní `ethX`, kde *X* je číslo síťové karty (první síťová karta je tedy `eth0`, další `eth1`, atd.).

Připojení přes modem běží pomocí point-to-point protokolu (PPP), rozhraní má symboliku `pppX`, kde *X* je číslo zprovozněného modemu (první zprovozněný modem bude `ppp0`). Velmi dobrý grafický klient k tomuto typu připojení je `kppp`.

Připojení přes wifi mívá symboliku `wlanX` (první rozhraní je `wlan0`) a nastavuje se příkazem iwconfig.

5.9.3 Zasíťování

Pokud je na síti DHCP server (server, který odpovídá na otázku vašeho počítače "kdo jsem?"), systém jej najde a nastaví síť podle něj (v přívětivých distribucích automat nebo klikátko, jinde příkaz **dhclient**). Bez DHCP serveru je potřeba nastavit síť ručně. Tady opět pomohou grafická klikátka nebo příkaz **ifconfig** či novější **ip**.

K tomu raději dodám trošku teorie. Každý počítač v síti musí být jednoznačně identifikován, a to IP adresou (ve tvaru `xxx.xxx.xxx.xxx`, např. `192.168.0.1`). Aby se od sebe odlišily různé sítě, používá se tzv. síťová maska (ve tvaru `xxx.xxx.xxx.xxx`, např. `255.255.255.0`). To ale nestačí. Je potřeba vědět, na jaký počítač (resp. na jakou IP adresu) se mají posílat pakety, které nepatří do žádné ze známých sítí, což je brána (tzv. gateway). A pořád ještě nám něco chybí. Asi bychom rádi do prohlížeče zadávali adresy ve tvaru `www.google.com` a nikoliv `66.249.93.104`. K tomu potřebujeme nějaký server, kterého by se náš počítač vždy zeptal, kterou IP adresu má daná URL, potřebujeme jmenný server (nameserver). Ty se zapisují do `/etc/resolv.conf`.

5.9.4 Servery

Servery jsou služby poskytované daným počítačem ostatním počítačům v síti. V GNU/Linuxu máte na výběr z mnoha serverů zajišťujících nejrůznější služby, namátkou webový server, poštovní server, FTP server, databázový server a řada dalších. Servery se ovládají pomocí skriptů v `/etc/init.d` nebo pomocí příslušných grafických nástrojů.

Server funguje tak, že si zabere příslušný port (číslo od 1 do 65535), a na něm naslouchá. Na tento port zašle klient požadavek o spojení a začíná komunikace. Pokud si nepřejeme, aby měl k serverům přístup každý počítač v síti, musíme je chránit firewallem.

5.9.5 Firewall

Komunikace mezi počítači prostřednictvím sítě probíhá za pomoci paketů, kterými se přenáší informace a požadavky mezi počítači. Teď už víme, že pakety směřují na určité porty, kde naslouchají patřičné servery. Firewall je de facto paketový filtr, umožňuje definovat jednoduchá i složitá pravidla, jak s pakety zacházet. Chceme-li servery chránit, musíme vytvořit filtr, který zahodí všechny pakety z počítačů mimo místní síť. GNU/Linux má velmi komplexní paketový filtr, Netfilter.

Dlužno dodat, že v přívětivých distribucích je nastavení firewallu otázkou několika kliknutí. Navíc existují grafické nástroje, které uživateli usnadňují tvorbu firewallu (KMyFirewall, Fwbuilder, Guarddog, apod.), ale i grafické nástavby fungující jako osobní firewall (Firestarter), které bych doporučil začátečníkům, kteří obdrželi distribuci bez klikátka nastavující firewall.

Firewall je přímo součástí linuxového jádra, přičemž utilita, kterou se firewall nastavuje, je iptables. Ručnímu stavění firewallu pomocí **iptables** se věnují některé články v odkazech.

5.10 Zdroje a odkazy

- ABC Linuxu, čtenáři, Učebnice GNU/Linuxu, Složení OS
- ABC Linuxu, čtenáři, Učebnice GNU/Linuxu, Přístupová práva
- ABC Linuxu, čtenáři, Učebnice GNU/Linuxu, Procesy
- ABC Linuxu, čtenáři, Učebnice GNU/Linuxu, X-Window
- ABC Linuxu, čtenáři, Učebnice GNU/Linuxu, Příkazová řádka
- Wikipédie (anglická), Shared library
- Slovník ABC Linuxu, filesystém
- ABC Linuxu, čtenáři, Učebnice GNU/Linuxu, Souborový systém
- Matej Gagyi, DevFS vs. udev
- Binh Nguyen, Linux Filesystem Hierarchy
- ABC Linuxu, Rastislav Stanik, Signály
- Johanka Spoustová, Pohádky z příkazové řádky
- Linux Command, http://www.linuxcommand.org/
- Seriál ABC Linuxu, Soukromá síť
- Seriál ABC Linuxu, Domácí síť
- ABC Linuxu, Zdeněk Štěpánek, Jak na WiFi kartu v Linuxu
- Seriál ABC Linuxu, Stavíme bezdrátovou síť
- InstallFest: Síťová zařízení a jak na ně (video, 2006)
- Seriál na Rootu, Vše o iptables

Kapitola 6

Správa GNU/Linuxu

6.1 Úvod do správy systému

Nepovažuji čtenáře za člověka, kterému dělá potíž kliknout na ikonu či najít určitou funkci v neznámém programu. Nebudu se tu tedy zabývat tím, kam v kterém programu kliknout, ostatně linuxové distribuce jsou, nejenom co se klikátek týče, značně odlišné. Dokumentace by pak nebyla o GNU/Linuxu, ale o některé z distribucí, a navíc by velmi rychle zastarávala. Místo orientace na klikátka se zaměřím na to, jak to funguje a jak věci "správně" dělat.

Samozřejmě na druhou stranu plně chápu, že se uživateli chce pracovat v pohodlí grafického rozhraní, jehož pohodlí je pro začátečníka klíčové. Stejně tak vím velmi dobře, že příkazová řádka je mocným nástrojem, který ocení nejenom nadšenci. Nebudu diskriminovat ani pohodlného uživatele, ani zkoumavého počítačového nadšence. Ukážu, jak věci realizovat v příkazové řádce, ale také představím adekvátní grafické programy sloužící k tomutéž.

6.1.1 Možnosti správy GNU/Linuxu

Jak už jsem naznačil, distribuce zaměřené na běžné uživatele obsahují grafické nástroje pro správu systému. Pomocí těchto nástrojů je možné realizovat většinu nastavení systému v pohodlí grafického rozhraní. První věc, kterou byste měli učinit, je tyto nástroje alespoň trochu prozkoumat, podívat se, co všechno nabízí. Měli byste také najít správce balíčků, pomocí kterého můžete snadno a rychle nainstalovat další software.

Každá linuxová distribuce se dá spravovat i z příkazové řádky. Konfiguraci tvoří obyčejné textové soubory, které se dají otevřít jakýmkoliv editorem. S pomocí nástrojů příkazové řádky je možné realizovat mnohé. Ovšem, je potřeba do toho investovat více času. Je na vás, který způsob si vyberete. Tento "ruční" způsob správy systému většinou umožňuje více, ale je složitější. Grafické nástroje umožňují méně, ale jsou snadní k použití.

6.1.2 Dokumentace je důležitá

Bez dokumentace to opravdu nejde, a to zejména, pokud přecházíte z jiného operačního systému, a jste navyklí na jiný styl správy systému. Primárním zdrojem informací by pro vás měla být příručka k distribuci, kterou používáte. Tu nemůže dokumentace na těchto stránkách v žádném případě nahradit. Každá distribuce má totiž svoje charakteristiky a specifika, které vám žádná obecná dokumentace nesdělí. Není ani na škodu projít další dokumentaci, a to zejména, pokud něčemu nerozumíte nebo se chcete něco dozvědět. Práce s dokumentací je podstatným prvkem správy systému.

6.1.3 Rozdělení dokumentace

Dokumentaci GNU/Linuxu můžeme rozdělit do následujících kategorií:

- manuálové stránky a programová dokumentace
- systémová dokumentace
- distribuční dokumentace
- on-line dokumentace
- zdrojové kódy (ideální pro programátory)

6.1.4 Manuálové stránky a programová dokumentace

Manuálové stránky a programovou dokumentaci byste měli najít minimálně v částečně počeštěné verzi ve většině přívětivých distribucí. Přeložené bývají dokumenty k nejčastěji používanému softwaru, zbytek dokumentace je v angličtině. K manuálovým stránkám se dostanete různými způsoby. Nejpohodlnější cestou v menu KDE je *Centrum nápovědy* spustitelné ručně příkazem **khelpcenter**, které shromažďuje jak část programové dokumentace, tak unixové manuálové stránky. Prostředí Gnome má k dispozici program **gnome-help**, který však integruje méně dokumentace. Z příkazové řádky je možné dostat se k dokumentaci pomocí příkazu **man**, který má syntax **man program**. Dokonce i tento příkaz sám o sobě má manuálovou stránku - **man man**.

Klasické manuálové stránky se dělí podle sekcí:

1. Uživatelské příkazy
2. Systémová volání
3. Knihovní funkce
4. Speciální soubory
5. Datové formáty
6. Hry
7. Různé

8. Správa systému

9. Jádro

Sekci je možné specifikovat `man číslo_sekce program`, takže v situaci, kdy má program více manuálových stránek, můžete tu, kterou chcete, specifikovat. Například `man 7 ip`. Ne vždy ale víme, který příkaz (resp. program) se na co hodí, a k tomu existuje jakýsi index popisků jednotlivých programů, ve kterém je možné vyhledávat buď pomocí `man -k hledaný_výraz` nebo pomocí ekvivalentního příkazu `apropos hledaný_výraz`. Je možné specifikovat i číslo sekce, kterou chcete prohledat (a odfiltrovat tak popisy knihovních funkcí) pomocí přepínače `-s`, takže například `apropos -s 1 mp3`.

Pokud známe příkaz, ale nevíme, co přesně dělá, můžeme vyvolat onen krátký popisek příkazem `whatis`, například `whatis ip`. Pokud má stejný program či výraz více manuálových stránek v různých sekcích (například zrovna zmíněný příklad), vypíše nám příkaz `whatis` popisky ze všech sekcí.

Vyhledávat lze však i fulltextově, tedy v obsahu manuálových stránek, příkazem `man -K hledaný_výraz` (unixové systémy rozlišují velká a malá písmena, proto se argument `-k` liší od argumentu `-K`). Pozor, takové vyhledávání může trvat velmi dlouho a je určitě vhodné specifikovat hledané sekce parametrem `-S` (lze jich specifkovat více, oddělovač je čárka). Například `man -S 8,9 -K memory`.

Kromě manuálových stránek existují ještě stránky programu `info`, které některé projekty upřednostňují. Ty podporují hypertextové odkazy.

6.1.5 Struktura manuálových stránek

Manuálové stránky mají svou předepsanou strukturu, která vám pomůže se v nich velmi rychle orientovat:

- jméno a krátký popisek
- stručný přehled
- popis
- příklady
- autor
- copyright
- související odkazy

Ne každá manuálová stránka musí být takto vybavena a dlužno dodat, že ani tento seznam není zcela kompletní. Podstatnými částmi je stručný přehled, který vás rychle obeznámí se syntaxí daného příkazu, popis, který tuto část rozvede do patřičné hloubky, někde nechybí ani příklady použití (velmi užitečná věc, mimochodem), samozřejmě nelze zapomenout na autora a copyright, ale bývá vhodné si všímat i souvisejících odkazů, které vám řeknou, kam ještě nahlédnout (příbuzná či přímo související témata).

6.1.6 Programová dokumentace

Ne všechna dokumentace je ve formě manuálových stránek, část této dokumentace (někdy podstatná část) se nachází v adresáři `/usr/share/doc`. Mohou tam být distribučně specifické záležitosti (třeba jak nastavit postgresql server v Debianu) nebo zbytky dokumentace z původního zdrojového balíčku (`README`, `INSTALL`, přehled změn, ap.).

6.1.7 Systémová dokumentace

Systémová dokumentace je veškerá dokumentace vztahující se ke GNU/Linuxu obecně. Nebývá součástí linuxových distribucí, nachází se v knihovnách, knihkupectvích, na webu, popřípadě v podobě audiovizuálních dat. Tato dokumentace vám pomůže ozřejmit řadu záležitostí, zejména funkčních principů, popřípadě praktických rad a postupů, které si jen těžko osvojíte studiem programové dokumentace.

Kromě zmíněných zdrojů existuje The Linux Documentation Project[1], který si klade za cíl sumarizovat širokou škálu poznatků o GNU/Linuxu. Hodit se mohou i odkazy v závěru této dokumentace.

6.1.8 Distribuční dokumentace

Každá linuxová distribuce má svá specifika. Informace o nich naleznete v dokumentaci vydávané přímo k vaší distribuci, ať už v podobě tištěné či elektronické. Jejich znalost je klíčová pro každého uživatele (resp. správce) dané distribuce a může vám nejen objasnit celou řadu záležitostí, ale především vám poskytne návody, jak si práci s distribucí usnadnit a jak předejít eventuelním problémům, popřípadě jak je řešit.

6.1.9 Úskalí dokumentace

Dokumentace GNU/Linuxu má i svá úskalí. Tím nejzřejmějším je její ohromující množství. To je důvodem, proč je třeba naučit se v dokumentaci vyhledávat (resp. filtrovat přebytečná data). Dalším problémem může být charakter dokumentace. Na tomto faktu se podepisuje skutečnost, že programovou dokumentaci píší programátoři, kteří mají tendenci být buď příliš odborní, nebo málo podrobní (tvorba dokumentace je zpravidla nejméně příjemnou činností při vývoji programu). Nedostatek dokumentace existuje spíše u okrajového softwaru, rozhodně ne u klíčového, ale přílišná odbornost nebo komplexnost dokumentace může působit problémy. Na druhou stranu, i špatná dokumentace je lepší než žádná.

Dokumentace má také tendenci zastarávat a pokud si uvědomíme rychlost, s jakou GNU/Linux roste, dostane tato záležitost zcela nový rozměr. Pokud si budete listovat nějakou knihou o GNU/Linuxu nebo budete-li číst nějaký starší článek či dokument na webu, mějte na paměti, že čas mohl od doby napsání dokumentu postoupit natolik, že se okolnosti změnily. Ne všechno se ale mění, jsou záležitosti, které platily před deseti lety a budou platit i za dalších deset.

[1] http://www.tldp.org/

6.2 Správa softwaru

K instalaci, odinstalaci a aktualizaci softwarového vybavení slouží v GNU/Linuxu primárně *správce balíčků* (package manager). K pochopení jeho funkce si nejprve vysvětleme základní pojmy. *Balíček* může být knihovna, datové soubory k programu, program samotný, dokumentace k programu nebo jiná komponenta. Jeden celý program tedy může být rozdělen na více balíčků. Smyslem takového členění je umožnit velmi precizně kontrolovat, co se do systému dostane a co ne, a značně usnadnit aktualizaci.

Balíčky najdeme v *repositářích*, což jsou v podstatě skladiště balíčků. Mohou to být adresáře, internetové zdroje či instalační média. Balíčky mají mezi sebou *závislostní vazby*, které určují, co vše je potřeba mít nainstalované, aby fungoval náš program.

Ukažme si to na příkladu. Řekněme, že chci nainstalovat program "míchačka". Tento program ale pro svou funkci vyžaduje komponentu "beton" a další program s názvem "dělník". Dělník zase vyžaduje komponentu "helma". Abych tedy mohl používat program míchačka, potřebuji beton, dělníka i helmu.

Repositáře si může uživatel sám volit a měnit, přidávat nové a ubírat stávající. Informace o nainstalovaných balíčcích, balíčcích v repositářích a jejich závislostních vazbách si ukládá správce balíčků ve své databázi. Pro práci s balíčky (a jejich stažení, instalaci, odinstalaci, aktualizaci, ap.) má správce balíčků nástroje. Tyto nástroje lze používat ručně.

6.2.1 Teorie funkce správce balíčků

Chce-li uživatel nainstalovat, odstranit či aktualizovat balíčky, použije za tímto účelem správce balíčků. Správce balíčků se podívá do databáze, vyřeší závislostní vazby a vyžádá si potvrzení od uživatele, je-li s vyřešením závislostí spokojen. Následně stáhne pomocí svých nástrojů sám všechny potřebné balíčky a ty nainstaluje, balíčky označené pro odinstalaci ze systému odstraní. Aktualizace databáze správce balíčků se děje buď automaticky před provedením jakékoliv operace (tak činí např. Yum), nebo ručně, kdy si o to uživatel požádá (tak činí např Apt).

6.2.2 Virtuální balíčky (meta-balíčky, dummy packages)

Virtuální balíček je speciální typ balíčku, který neobsahuje žádnou komponenetu, pouze závislostní informace. Možností jeho využití je celá řada, přičemž nejběžnější je seskupování balíčků, které na sobě přímo nezávisí, ale tvoří určitý celek (například distribuci grafického prostředí se základní sadou softwaru pro desktop). Instalací takového balíčku pak díky závislostem nainstaluji snadno daný celek, aniž bych musel vybírat desítky programů zvlášť, neboť ty na sobě přímo nezávisí. Z uživatelského hlediska je nejpodstatnější, že nevadí, když jej odinstalujete.

Ukažme si to celé na příkladu. V distribuci Ubuntu je virtuální balíček `ubuntu-desktop`, který sám neobsahuje žádnou funkcionalitu, ale závisí na všech komponenetách, které tvoří základ distribuce Ubuntu (systémové komponenety, prostředí Gnome a řada uživatelských aplikací). Pokud se rozhodnu, kupříkladu, odstranit program Evolution, na kterém mj. onen virtuální balíček závisí, zahlásí mi správce balíčků,

že hodlá odstranit i onen virtuální balíček, `ubuntu-desktop`. Jelikož tento balíček neposkytuje žádnou funkcionalitu, nemusím se obávat a nechám jej odstranit.

Jiným příkladem jsou běžnější virtuální balíčky, třeba balíček `kde`. Ten závisí na všech oficiálních součástech prostředí KDE, i takových, které nejsou k jeho běhu nutné. Pokud chci tedy nainstalovat standardní distribuci prostředí KDE, stačí mi nainstalovat tento virtuální balíček (správce balíčků mi pak nabídne všechny jeho závislosti, které právě tvoří standardní distribuci prostředí KDE). A opět, pokud některou komponenetu nezávislou pro běh samotného KDE budu chtít odinstalovat, nabídne mi správce balíčků jeho odstranění. Tady to na první pohled vypadá hrozně (chce odinstalovat `kde`), ale vzhledem k tomu, že je to pouze virtuální balíček, nehrozí žádné riziko, takže to mohu bez obav povolit. Naopak, pokud by mi nabídl odstranění mnoha balíčků začínající na "kde" včetně "`kdebase-bin`", pak je mi jasné, že se jedná o klíčovou komponenetu KDE a nic odinstalovávat nebudu.

6.2.3 Uživatelský pohled na správce balíčků

Jediné, co zbývá na uživateli, je vybírat, které programy chce mít nainstalované a které ne, popřípadě, ze kterých repositářů se má tahat software. Vše ostatní zařídí správce balíčků sám, včetně vyřešení závislostí a stažení ze sítě.

Správce balíčků lze ovládat pomocí příkazové řádky nebo přes adekvátní grafické nástavby. My si ukážeme na příkladu práci s obojím. Pozor, distribuce využívají různé správce balíčků, a proto nelze zaručit, že bude vše úplně stejné ve vámi zvolené distribuci. Tyto rozdíly by však mělo odstranit prostudování distribuční dokumentace.

6.2.4 Příklad: Řádkový Apt

Ukažme si na příkladech, jak vypadá práce se správcem balíčků, třeba na správci Apt, který je typický pro distribuce založené na Debianu (Ubuntu, Linspire, Xandros,apod.). Práce s ním je poměrně jednoduchá. Zkusíme pomocí něj nainstalovat program Filelight. Nejprve aktualizujeme databázi:

```
apt-get update
```

Nyní nainstalujeme program Filelight:

```
apt-get install filelight
```

A nyní ho zkusíme odinstalovat:

```
apt-get remove filelight
```

Aktualizace veškerého softwaru se provede příkazem:

```
apt-get upgrade
```

Někdy se člověku může hodit mít možnost požadovaný balíček vyhledat. Zkusíme si vyhledat nějaký program na ripování DVD:

```
apt-cache search dvd rip
```

Řekněme, že jedna z možností, balíček dvdrip, se nám zamlouvá. Mohli bychom si vyžádat popisek balíčku:

```
apt-cache show dvdrip
```

Více informací ke správci balíčků Apt naleznete v příslušné dokumentaci.

6.2.5 Příklad: Grafická nástavba Apt: Synaptic

Grafické nástavby správců balíčků v distribucích se chovají velmi podobně. Někde bývají rozděleny do více modulů (jeden pro instalaci, druhý pro odinstalaci, třetí pro aktualizaci), jinde je všechno pohromadě. V našem příkladu, programu Synaptic, je všechno pohromadě. Je možné současně označit balíčky pro instalaci, aktualizaci i odinstalaci, a nechat správce balíčků, ať všechno udělá najednou. K dispozici jsou i některé pokročilé funkce jako upřednostnění specifické verze balíčku (pomocí této funkce je možný downgrade balíčku). Samozřejmostí jsou integrované vyhledávací funkce a filtry.

Po označení všech požadovaných balíčků stačí použít tlačítko "realizovat", a veškeré změny se začnou provádět. Aktualizovat databázi repositářů je možné pomocí tlačítka "aktualizovat". Tlačítko "aktualizovat vše" automaticky vybírá všechny kandidáty pro aktualizaci, aby se s tím člověk nemusel namáhat ručně (ekvivalent příkazu `apt-get upgrade`).

6.2.6 Instalace softwaru bez správce balíčků

Tento postup důrazně nedoporučuji, pokud máte možnost použít správce balíčků a příslušné repositáře. V případě, že nutně potřebujete nějaký software, který v repositářích není, máte dvě možnosti, jak jej do systému přesto nainstalovat, i když tak přijdete o možnost balíček automaticky aktualizovat.

6.2.6.1 Instalace balíčku pomocí nízkoúrovňového nástroje

Jak už jsme si řekli, správce balíčků používá řadu nástrojů, z nichž jeden je určen na instalaci, odstranění a aktualizaci balíčku. Jeho nevýhodou je skutečnost, že neumí vyřešit závislosti, i když je kontroluje a zpravidla nedovolí nainstalovat balíček s nesplněnými závislostmi.

Nejprve je třeba nalézt balíček určený pro vaši distribuci. Měl by být opravdu pro vaši distribuci, a ne pro jinou, byť používá stejný formát balíčku (např. rpm). Stejně tak by měl být daný balíček určen přímo pro danou verzi vaší distribuce. Pokud nebude k dispozici balíček pro vaši distribuci, stáhněte si zdrojový kód a pokuste se o kompilaci (postup viz níže), instalaci balíčku sestaveného pro jinou distribuci či jinou verzi distribuce důrazně nedoporučuji.

Jakmile máte patřičný balíček, zjistěte si, jaké má závislosti a ty vyřešte pomocí správce balíčků (nainstalujte je). Nepokoušejte se je stahovat ručně, je to zbytečné. Snažte se počet instalovaných balíčků mimo repositáře minimalizovat. Až budete mít závislosti vyřešené, nainstalujte balíček pomocí nízkoúrovňového nástroje (`dpkg` v

distribucích založených na Debianu, `rpm` v distribucích vycházejících z Red Hatu/-Fedory), ale v žádném případě nevypínejte kontrolu závislostí. Nebudou-li závislosti splněné, nainstalujte raději program ze zdrojového balíčku (postup následuje).

6.2.6.2 Kompilace balíčku ze zdrojových kódů

Pro kompilaci potřebujete adekvátní vývojové nástroje, které nainstalujte pomocí správce balíčků. Patří mezi ně kompilátor gcc a g++, včetně jejich závislostí. Na stránkách projektu, který hodláte kompilovat, si najděte informace o jeho závislostech. Ty pak nainstalujte opět pomocí správce balíčků. Pro kompilaci je potřeba mít vývojové balíčky jednotlivých závislostí. Ty poznáte tak, že mají přídomek `-dev` nebo `-devel`, kupříkladu webový server apache má v mé distribuci (Ubuntu) vývojový balíček s názvem `apache-dev`.

Po stažení zdrojáků je rozbalte. Můžete použít mocné menu pravého tlačítka, souborový manažer či příkazovou řádku. V příkazové řádce doporučuji souborový manažer `mc`, který umí s archívy zacházet jako s adresáři. Můžete samozřejmě použít klasický způsob rozbalení tar.gz či tar.bz2 balíčků, který si tu teď ukážeme na příkladu balíčku hry Rocks'n'diamonds. Balíčky tar.gz nebo tgz lze rozbalit takto:

```
tar xzvf rocksndiamonds-3.1.2.tar.gz
```

Připomeňme, že v příkazové řádce lze použít klávesu tabulátor k doplnění zbytku názvu souboru, takže v tomto případě nemusím vypisovat celý název, ale postačí jen napsat "rock" a zbytek nechat doplnit tabulátorem. Balíček tar.bz2 by se rozbalil takto:

```
tar xjvf rocksndiamonds-3.1.2.tar.bz2
```

Balíček ve formátu zip by šel rozbalit takto:

```
unzip rocksndiamonds-3.1.2.zip
```

Jakmile je zdroj rozbalen, budete už nutně potřebovat příkazovou řádku. Dostaňte se do hlavního adresáře zdrojových souborů, pomocí příkazu cd, v mém případě třeba takto:

```
cd rocksndiamonds-3.1.2
```

Opět připomínám, že lze použít tabulátor k doplnění jména souboru nebo adresáře. Jakmile jsme v hlavním adresáři, zapíšeme:

```
./configure
```

Případné chybové hlášky se mohou týkat potřebných závislostí, které najdeme a doinstalujeme pomocí správce balíčků. Postup opakujeme, dokud nedostaneme žádnou chybovou hlášku. Pak provedeme příkaz:

```
make
```

Tento příkaz program zkompiluje. K instalaci můžeme použít příkaz `make install`, samozřejmě jako uživatel root, jehož oprávnění můžeme v příkazové řádce získat pomocí příkazu `su` nebo `sudo`. Já vám však ukážu jiný postup, kterým usnadníte

možnost balíček zase odinstalovat, a tím je program `checkinstall`. Ten si nainstalujte pomocí správce balíčků (měl by být snad v každé slušné distribuci) a použijte jej místo make install následujícím způsobem. Získejte oprávnění uživatele root a spusťte `checkinstall`:

```
su -c checkinstall
```

Například v distribuci Ubuntu, kde heslo uživatele root není zadáno, můžete použít mechanismus sudo, takto:

```
sudo checkinstall
```

Checkinstall vám vytvoří balíček, který pak nainstalujete nízkoúrovňovým nástrojem, v distribucích založených na balíčcích rpm třeba takto (stále s oprávněními uživatele root):

```
rpm -ivh rocksndiamonds-3.1.2-i386.rpm
```

V distribucích založených na balíčcích deb to bude následovně:

```
dpkg -i rocksndiamonds-3.1.2-i386.deb
```

Nebo, v případě distribuce Ubuntu, pomocí mechanismu sudo:

```
sudo dpkg -i rocksndiamonds-3.1.2-i386.deb
```

Následně se zbavte oprávnění uživatele root (příkaz "logout" nebo klávesová zkratka CTRL+D, v distribuci Ubuntu netřeba, mechanismus sudo propůjčí oprávnění uživatele root pouze přes sudo prováděným příkazům, takže máte stále uživatelská oprávnění).

6.2.7 Osobní doporučení

Jak jste si všimli, správa softwaru s použitím správce balíčků je velmi jednoduchá. Navíc, oficiální repositáře obsahují balíčky prověřené, otestované a digitálně podepsané. Jejich funkčnost a bezpečnost by tedy měla být zaručena. Občas se sice stane, že některý málo používaný balíček nefunguje (nebyl dostatečně otestován), ale tyto případy jsou výjimečné. Naopak správa softwaru bez použití správce balíčků je poměrně složitá, a funkčnost či bezpečnost daných balíčků vám nikdo nezaručí. Výjimku tvoří komerční software, který v repozitářích není (přirozeně), ale lze po zaplacení stáhnout příslušný balíček a nainstalovat jej (příkladem může být Cedega).

Mým doporučením je, abyste v každém případě preferovali správce balíčků. Je to snadné a v rámci každé slušné distribuce byste měli mít softwaru v repozitářích více než dost, abyste nemuseli balíčky instalovat ručně kdo ví odkud. Řada distribucí mívá neoficiální repositáře, kde naleznete hromadu dalšího softwaru, včetně toho, který z nějakých důvodů (např. licenčních) nemohl být umístěn do oficiálních repositářů (multimediální kodeky s patentově chráněnými technologiemi, apod.). Tyto neoficiální repositáře bývá vhodné lokalizovat a přidat do správce balíčků (informace o nich bývají v distribuční dokumentaci, FAQ, wiki, eventuelně na fanouškovských stránkách či v příslušných diskusních fórech).

Pokud vám nějaký program citelně chybí, a není ani v oficiálních a ani v neoficiálních repositářích, ještě je možné jej najít ve specializovaných repositářích. Ty bývají velmi malé, třeba pouze pro daný program, ale v našem případě postačí. Samozřejmě doporučuji obezřetnost a nedoporučuji takový repositář přidávat, pokud na vás působí nevěrohodně.

Budete-li chtít instalovat stažený balíček pomocí nízkoúrovňového nástroje, který neřeší závislosti (jenom na ně upozorňuje), ujistěte se, že je z důvěryhodného zdroje, a hlavně nevypínejte kontrolu závislostí. A pokud budete instalovat program ze zdrojových kódů, používejte `checkinstall`.

6.3 Správa hardware

O hardware se stará samotné linuxové jádro, spolu s démonem hotplug nebo udevem, podle toho, který máte nainstalovaný. Obojí se stará o načtení příslušného modulu do jádra, když zjistí, že se dané zařízení připojilo.

V současné době bere velká část výrobců HW v úvahu GNU/Linux. Část těch, kteří tyto ohledy neberou, je naštěstí schopna sestavovat takový HW, pro který je možné linuxové ovladače vytvořit. Problémovou skupinou jsou ti, kteří šijí horkou jehlou, popřípadě se příliš drží proprietárního modelu vývoje a tají specifikace svého HW, čímž znemožňují vytvoření ovladačů linuxovými vývojáři a pokud sami takové ovladače nevytvoří, HW nebude pod GNU/Linuxem funkční.

Z hlediska uživatele můžeme vyčlenit tři kategorie HW, dle kompatibility s GNU/-Linuxem:

- HW s ovladačem v jádře
- HW s ovladačem mimo jádro, ale dostupným
- HW bez ovladače

Ideálním stavem je první bod. Je-li k dispozici svobodný ovladač přímo v jádře, nemáme sebemenší problém, HW je nalezen a zprovozněn okamžitě bez nutnosti zásahu. Tento stav bychom měli preferovat, vybíráme-li hardware pro GNU/Linux. Bod číslo dvě znamená problém, ovladač sice existuje, ale postup zprovoznění může být značně složitý a za určitých okolností nemusí fungovat s novějšími jádry. Některé distribuce mohou rozšiřovat podporu pro tato zařízení, ale vzhledem ke konceptu svobodného softwaru to není zvykem, je-li ovladač nesvobodný.

U třetí kategorie není o čem mluvit, to je prostě smůla, pokud se k nějakému takovému kusu HW dostaneme. Možnosti řešení jsou mizivé. Je možné si počkat, doufaje v pozdější nápravu, je možné napsat rozhořčený e-mail výrobci či si napsat vlastní ovladač, ovšem ne každý je zrovna programátor. Nebývá od věci poslat výrobci e-mail s tím, že vzhledem k jeho politice podpory jednoho operačního systému s jeho produkty nadobro končíme.

6.3.1 Zprovoznění nefunkčního hardwaru

Není-li hardware zprovozněn, nezbývá než se podívat, jak to vlastně s podporou daného hardwaru je. Za tímto účelem pomůže pět zdrojů informací:

- uživatelská příručka
- vyhledávače: Google, Jyxo
- stránky věnované podpoře hardwaru v GNU/Linuxu
- diskusní fóra a e-mailové konference
- distribuční portál

Zprovoznění nejběžnějšího hardwaru bude nepochybně již obsaženo v distribuční příručce nebo adekvátních on-line zdrojích. Typicky je to zprovoznění 3D akcelerace. Pokud máte exotičtější zařízení, nezbývá než hledat v jiných zdrojích.

Vyhledávače představují způsob, jak velmi rychle najít požadovanou informaci. Prostě jako vyhledávací výraz napíšete název hardwaru a přidáte klíčové slovo "linux" nebo název distribuce, kterou používáte. S trochou štěstí hned natrefíte na návod, jak zařízení zprovoznit.

Stránky s informacemi o podpoře hardwaru v GNU/Linuxu umí být také cenným zdrojem informací, protože leckdy obsahují i příslušné návody, nebo alespoň nějaké tipy. Odkazy na některé z těchto webů naleznete níže.

Diskusní fóra a e-mailové konference bývá také dobré prohledat, zejména ty, které jsou určeny pro vaši distribuci. Je velmi pravděpodobné, že s daným kouskem hardwaru měl už někdo jiný problém, a vy se tak můžete rychle dostat k návodu nebo příslušným dokazům.

Pokud má vaše distribuce rozsáhlejší portál, je dobré se na něj podívat a zjistit, neobsahuje-li informace vztažené k hardwaru, nebo alespoň diskusní fóra či e-mailové konference věnované dané distribuci.

Pak už zbývá jenom prokousat se nalezenými informacemi a zkusit patřičné postupy aplikovat. Většinou se jedná o kompilaci modulu pro jádro, který si stáhnete od někud z Internetu. Kompilace modulu už v dešní době není takový problém, stačí patřičné vývojové nástroje, vývojové balíčky pro jádro (*kernel-headers*, *kernel-source*, apod.), a pak postupovat podle přiloženého návodu.

6.3.2 Výběr vhodného hardwaru

V případě, že byste si rádi pořídili hardware tak, abyste měli jistotu, že bude v GNU/Linuxu podporován, máte několik možností. Můžete si najít prodejce, který vám s výběrem hardwaru pro GNU/Linux pomůže. Ne každý má ale takové štěstí, a proto nám nakonec nezbyde než si informace vyhledat sami. Google a další vyhledávače jsou opět nedocenitelným pomocníkem. Existují ale také weby zabývající se mapováním podpory hardwaru v GNU/Linuxu. A tam bývá dobré nakouknout nejdříve.

Českým uživatelům se bude určitě hodit odkaz na web ABC Linuxu[2], kde naleznete uživatelskou databázi kompatibility hardwaru. Důležité je, že často obsahuje i návody

[2] http://www.abclinuxu.cz/hardware

ke zprovoznění daného kousku. A pokud vaše hledání neuspěje, hned po ruce máte diskusní fórum, kde můžete hledat dále.

6.3.3 Seznamy kompatibilního hardwaru a další odkazy

- Tiskárny
  - http://www.linuxprinting.org
- Scannery
  - http://www.sane-project.org
- Zvukové karty
  - http://www.alsa-project.org
- Základní desky
  - http://www.linux-tested.com
- Digitální fotoaparáty
  - http://www.gphoto.org
  - http://www.teaser.fr/~hfiguiere/linux/digicam.html
- Wifi karty
  - http://www.linux-wlan.org
- Hardware obecně
  - http://www.linux-drivers.org/#lhw
  - http://www.linuxhardware.org/
  - http://www.tldp.org/HOWTO/Hardware-HOWTO/

6.4 Informace o systému

GNU/Linux nabízí mnoho možností, jak se dostat k přesným informacím o tom, co se stalo nebo co se momentálně děje.

6.4.1 Logy

Logy jsou záznamy o proběhlých událostech. Jsou uložené v adresáři `/var/log`. Složení tohoto adresáře bývá distribučně specifické, nicméně při pohledu na názvy souborů vám rychle dojde, k čemu každý slouží. Hlavní log je zpravidla uložen v souboru `messages`, doplňující, detailnější log pak v souboru `syslog` (v některých distribucích není). To je také první místo, kam byste se měli podívat.

Jednotlivé služby mají svoje vlastní logy, takže třeba X-server má svůj vlastní log `Xorg`. Pokud nefunguje grafické prostředí, je tohle místo, kam byste se měli podívat.

Jelikož probíhající události logy zaplňují, provádí se jejich rotace (aby se časem nezaplnil disk). Po určité době (nebo při překročení určité velikosti) se log zpravidla zkomprimuje a uloží do stejného adresáře do souboru se stejným názvem zakončeným číslem (tj. třeba `messages.1`). Takhle se postupuje dále do určitého čísla, a pak následuje výmaz.

Jak číst logy? Logy jsou obyčejné textové soubory (lze je tedy otevřít jakýmkoliv textovým prohlížečem nebo editorem), ve kterých je na každé řádce zaznamenána jedna událost. Na začátku řádku je časový údaj značící, kdy událost proběhla. V příkazové řádce můžete snadno použít filtrační systémy, které vám umožní zobrazit přesně ty informace, které potřebujete (**grep**, **sort**, **head**, **tail**, apod.). Existují ale i prohlížeče logů v grafickém prostředí, v rámci prostředí KDE je to ksystemlog, pro Gnome existuje gnome-system-log.

Z logů tedy vyčtete, co se stalo a kdy se to stalo. Tyto informace můžete poměrně snadno využít k lokalizaci a průzkumu eventuálního problému. Kupříkladu, díky logům jsem mohl snadno odhalit, že za přerušením spojení nestojí vada mého modemu, ale poskytovatel připojení, který mne po 24 hodinách prostě odstřihnul.

Snadný přístup k nedávným událostem zaznamenaným linuxovým jádrem umožňuje program **dmesg**, po jehož zadání se jednotlivé události vypíší.

Tvorbu logů zajišťuje zpravidla démon syslog, který má konfigurační soubor v `/etc/syslog.conf`. Jeho úpravou lze velmi precizně řídit, které informace se budou zapisovat do kterého logu. Výchozí nastavení ale většinou bývá zcela postačující.

6.4.2 Informace o hardwaru počítače

Grafické prostředí nabízí mj. program hardinfo, který vám přehledně zpracuje informace o zjištěném hardwaru. To ovšem umí i ovládací centra distribucí Mandrake či SUSE. Je dobré vědět, že existuje program **lspci** (existují i další, třeba **lsusb**) z balíku `pciutils`, který sice pracuje v příkazové řádce, ale který vám vytáhne informace o zjištěném hardwaru v podobě, která je ideální k operaci kopíruj a vlož. Můžete samozřejmě také přímo použít rozhraní jádra v `/proc`, konkrétně v následujících souborech:

- `/proc/cpuinfo` - informace o procesoru
- `/proc/meminfo` - zaplnění paměti
- `/proc/mounts` - připojené souborové systémy
- `/proc/partitions` - zjištěné oddíly pevných disků

- `/proc/pci` - PCI zařízení
- `/proc/swaps` - odkládací oddíl(y)
- `/proc/version` - verze jádra (dostupné také příkazem **uname -a**)

Expertům přijdou vhod ještě následující soubory:

- `/proc/cmdline` - parametry předané jádru
- `/proc/devices` - bloková a znaková zařízení
- `/proc/interrupts` -
- - HW přerušení
- `/proc/iomem` - rozdělení paměti
- `/proc/ioports` - I/O porty

6.4.3 Monitory a utility

Tyto nástroje slouží ke sledování určitých hodnot, událostí a procesů při běhu systému. V grafickém prostředí oceníte zejména následující nástroje:

- `gkrellm` - komplexní monitorovací software
- `gnome-system-monitor` - systémový monitor pro GNOME
- `ksysguard` - systémový monitor pro KDE
- `qps` - sofistikovaná aplikace pro monitorování a správu procesů

Pro příkazovou řádku existuje celá řada nejrůznějších utilit sloužících ke sledování stavu systému:

- `top` - monitoruje procesy, jejich využití procesoru a paměti
- `free` - zobrazí stav obsazení paměti
- `ps` - slouží k vypsání spuštěných procesů a informací o nich, kompletní výpis: `ps aux`
- `uptime` - zobrazí dobu běhu systému a zatížení měřené v uplynulé minutě, 5 a 15 minutách
- `lsof` - umožňuje zjistit, které soubory jsou obsazeny kterými procesy (hodí se třeba v případě, že chcete odpojit nějaké zařízení a jste častování hláškou, že zařízení je využíváno)
- `du-h` - vypíše, kolik místa zabírají jednotlivé adresáře
- `df-h` - vypíše, kolik místa je obsazeno a volno na jednotlivých zařízeních

Informace vstažené k síťování:

- `ping` - ping na nějaký známý server pomůže zjistit, je-li připojení k Internetu funkční
- `ifconfig` - vypíše informace o nastavení síťových rozhraní
- `iwconfig` - vypíše informace o nastavení wifi rozhraní
- `route` - vypíše routovací tabulku
- `traceroute` - zjistí, jakou trasou prochází pakety k zadanému cíli
- `netstat` - vypíše aktivní síťová spojení, oblíbené použití:

```
netstat -tupan
```

- `lsof-i` - vypíše aktivní síťová spojení

6.4.4 Analýza běhu programu

Může se stát, že vám nefunguje nějaký program. Pak se hodí několik šikovných nástrojů. Podotýkám, že tohle je už záležitost pro minimálně začínající programátory a velmi pokročilé uživatele.

Základním nástrojem pro analýzu běhu programu je řádkový **strace**, který spustíte s parametrem v podobě programu, který chcete analyzovat. Strace vám pak začne poskytovat informace o systémových voláních a signálech analyzovaného programu. Analýzou průběhu operací můžete zjistit, na čem přesně program krachne.

Pokud máte problém s během programu, může tak být v extrémních případech (nepoužívali jste správce balíčků k instalaci softwaru) kvůli chybějící závislosti. Závislosti programů (alespoň na knihovnách) získáme pomocí příkazu **ldd**, jehož parametrem je spouštěcí soubor daného programu. Ten můžete najít třeba pomocí programu **locate** nebo **find**.

6.4.5 Kým je používané zařízení?

Zabezpečení integrity dat je mj. důvodem zamykání mechanik. Pokud však si však nějaké zařízení zarezervuje nějaký neposlušný program a vy netušíte, který by to mohl být, použijte příkaz lsof na daném zařízení, třeba takto:

```
lsof /dev/hda
```

Můžete k tomu potřebovat práva uživatele root. Příkaz vypíše název a PID programů, které dané zařízení používají. Ty pak můžete využít jako parametry programu **kill** (viz *Správa procesů* níže).

6.5 Správa procesů

Proces je, jak víme, běžící program. S programy většinou komunikujeme přímo, interaktivně. Výjimku tvoří démoni, programy pracující neinteraktivně, na pozadí. Některé z nich jsou systémové služby, některé z nich jsou síťové servery. Každý proces zabírá určité systémové zdroje (paměť, přístup k zařízení). Proces je v systému identifikován číslem procesu (PID) a běží s právy uživatele, který jej spustil.

6.5.1 Programy pro správu procesů

Tyto programy již známe ze sekce *Info o systému*. V grafickém prostředí jsou to **ksysguard**, **gnome-system-monitor** či **qps**. Pro příkazovou řádku to jsou programy **top** či **htop**, **ps** a **kill**.

6.5.2 Zabíjení procesů a signály

Pokud proces běží správně, nemá uživatel většinou důvod do jeho běhu nějak zasahovat. Ale někdy se proces zblázní a začne působit problémy, ať už vytížením procesoru, zabráním velkého množství paměti či pouhým zamrznutím. V takovém případě se hodí vědět, jak ho zabít, tedy násilně ukončit. V grafickém prostředí k tomu slouží program **xkill**, který je v prostředí KDE implicitně namapován na klávesovou zkratku Ctrl-Alt-Esc. Lze k tomu samozřejmě použít i výše zmíněné programy. V příkazové řádce lze použít program **kill**, který de facto slouží k zasílání signálů procesům.

Signál program zachytí a zachová se podle něj. Nejdůležitější jsou signály *SIGTERM* a *SIGKILL*. První požadá program o ukončení, druhý ho bez milosti sestřelí. To je důvod, proč doporučuji programu nejprve zaslat signál *SIGTERM*, a teprve pak *SIGKILL*. V případě, že program zareaguje na *SIGTERM*, může se pokusit ještě nouzově ukončit a uložit rozpracovaná data. Použití programu **kill** je samozřejmě popsáno v jeho manuálové stránce, ovšem přesto si dovolím malou ukázku v kombinaci s programem **ps**. Řekněme, že nám zamrznul program gmpc, a my se jej pokoušíme ukončit. Nejprve bychom měli zjistit jeho PID:

```
ps aux | grep "gmpc"
```

Co jsme udělali? Spustili program **ps** s parametry `aux`, které nám vypíšou všechny relevantní informace o všech procesech. My ale nechceme informace o všech procesech, a proto jsme výstup programu **ps** přesměrovali na vstup filtrovacího programu **grep**, který "propustí" pouze řádky obsahující *gmpc*. Výsledkem je výpis podobný tomuto:

```
michal 18406 0.0 1.5  83992 16216 ? S 00:59 0:05 gmpc
```

Hned první číslo zleva je námi hledané PID, které použijeme jako parametr programu **kill**. Pokud nezadáme číslo signálu, je zvolen *SIGTERM*.

```
kill 18406
```

Program se však stále neukončil, a proto mu posíláme devátý signál, tedy *SIGKILL*:

```
kill -9 18406
```

Zkusím uvést ještě jeden užitečný program, a tím je **killall**, který umožňuje poslat signál procesu nebo procesům podle jejich jména a nikoliv podle PID. Pomocí něj by stačilo zapsat:

```
killall gmpc
```

Ale pozor, tento program pošle signál všem procesům, které se jmenují tak, jak to specifikujete. Můžete takto nechtěně zabít i program, který nechcete. To je důvod, proč jej uvádím na konec. Onen první způsob je sice zdlouhavý, ale poměrně jistý.

O signálech jako takových se dozvíte více v odkazech a samozřejmě v adekvátní manuálové stránce:

```
man signal
```

6.5.3 Slušnost (priorita) procesů

V unixových systémech je třeba velmi precizně řídit priority jednotlivých procesů. K tomu se používá číselná hodnota, která ovlivňuje, jak moc mu jádro dává přednost před ostatními procesy. V jistých systémech se toto číslo nazývá "priorita" a platí, že čím vyšší číslo, tím vyšší priorita. V unixových systémech je to ale přesně naopak. Toto číslo udává slušnost (niceness), tedy čím vyšší číslo, tím méně jej jádro upřednostňuje před ostatními. V GNU/Linuxu je slušnost udávána v celých číslech v rozmezí `-20` až `19`, přičemž záporná čísla jsou vyhrazena pouze pro nastavení uživatelem root. Je jasné, že běžně smíte měnit slušnost pouze těch procesů, které vám patří. Jste-li root, můžete vše.

Slušnost procesu můžeme změnit v grafickém prostředí pomocí výše zmíněných nástrojů, nebo můžeme použít příkazovou řádku a programy nice a renice. Program nice spouští program s danou prioritou, program renice mění prioritu běžícího procesu. Použití si demonstrujeme na příkladě:

```
nice -n 19 slušňák
```

Výše zmíněný příkaz spuští program *slušňák* s maximální slušností. Při běhu procesu ale zjistíme, že by bylo dobré jeho slušnost snížit na 5. Použijeme tedy příkaz renice, ovšem nejprve musíme zjistit PID daného procesu (viz výše). Dejme tomu, že slušňák běží pod PID 7432:

```
renice 5 7432
```

Interaktivní manažery procesů, ať již pro příkazovou řádku (např. **htop**) či pro grafické prostředí (**ksysguard**, **qtop**, atd.), umožňují provádět změny "slušnosti" procesů velmi pohodlně.

6.5.4 ulimit: Omezování procesů

V GNU/Linuxu existuje mechanismus na omezování procesů a uživatelů. Je možné omezit v rámci procesu množství paměti a vytížení procesoru, v rámci uživatele množství spuštěných procesů, počet přihlášení a řadu dalších záležitostí. Vše se nastavuje v `/etc/security/limits.conf` nebo příkazem **ulimit**. Vypsáním následujícího příkazu můžete zjistit, jak jste omezeni:

```
ulimit -a
```

Více informací se dozvíte v manuálových stránkách interpretu Bash:

```
man bash
```

6.6 Správa serverů

Servery jsou procesy běžící na pozadí (démoni), které naslouchají na určitém portu a poskytují službu jiným počítačům v síti. Typickým příkladem může být webový server Apache. Linuxové distribuce nabízejí kompletní vybavení pro server, stačí se jenom podívat do správce balíčků.

Servery se spouští automaticky po startu systému. Za běhu se dají ukončit, spustit či restartovat pomocí skriptů zpravidla v `/etc/init.d`. Skripty se mohou jmenovat jinak, ale základ práce bývá velmi podobný. Ukážeme si to na příkladu webového serveru Apache. Předpokládejme, že server běží, a chceme jej ukončit. Zapíšeme:

```
/etc/init.d/apache stop
```

Pustíme jej opět takto:

```
/etc/init.d/apache start
```

Pokud máme živý, produkční server, pak vypnout a zapnout službu nemusí být ideální způsob, jak vyřešit drobnou úpravu konfigurace. Proto se nám hodí, třeba právě u serveru Apache, parametr reload či jeho násilnější varianta force-reload:

```
/etc/init.d/apache reload
```

Ne každá služba má však takové možnosti. Někdy nám tedy nezbyde nic jiného než službu restartovat (což je pořád lepší než vypnout a zapnout):

```
/etc/init.d/apache restart
```

V případě produkčních serverů se tyto úkony odehrávají zpravidla v pozdních nočních hodinách, kdy je počet aktivních uživatelů serveru minimální.

6.6.1 Konfigurace serverů

Většina serverů má nějaké počáteční nastavení, které umožní jejich běh, aniž bychom je museli nastavovat. Přívětivé distribuce mohou mít nastavovací klikátka pro některé

ze serverů, ale většinou budeme servery konfigurovat klasickým způsobem, přes konfigurační soubory v `/etc` podle dostupné dokumentace, kterou nalezneme mj. v `/usr/share/doc`, v distribuční dokumentaci, na stránkách projektů, apod.

6.6.2 Servery, o kterých je dobré vědět

Následuje popis některých serverů, o kterých by měl každý uživatel GNU/Linuxu vědět.

6.6.2.1 SSH (OpenSSH)

Server SSH, tedy přesněji OpenSSH, umožňuje zabezpečené (tj. šifrované) přihlášení k shellu na serveru. Můžete se tak přihlásit k příkazové řádce vzdáleného počítače a provádět s ním v podstatě cokoliv. Dokonce, je-li v konfiguraci (`/etc/ssh/sshd_config`) povolen *X11Forwarding*, je možné spouštět i vzdálené grafické aplikace.

SSH lze však využít k mnoha dalším věcem. Je možné s jeho pomocí obalit jiné nešifrované protokoly (třeba *rsync*), a zvýšit tak podstatně jejich bezpečnost. Prostřednictvím SSH je také možné vytvářet šifrované tunely mezi počítači.

6.6.2.2 X Server

Grafické rozhraní GNU/Linuxu je postaveno na architektuře klient/server, kde serverem je právě X Server a klientem jsou pak jednotlivá grafická prostředí nebo okenní manažery.

6.6.2.3 CUPS

Cups je tiskový server, který nejspíše běží i ve vaší distribuci a obsluhuje vaše případné tiskárny. Tiskárny a tisk se spravuje buď přes nějaké speciální grafické klikátko, nebo přes Gnome aplikaci **gnome-cups-manager**, nebo klasicky přes webové rozhraní http://localhost:631/. Server jako takový má konfigurační soubory umístěné v `/etc/cups`.

6.6.3 Bezpečnost

Je nad slunce jasné, že kterákoliv služba přístupná z Internetu je potenciálně riziková. I proto je důležité si pořádně projít dokumentaci a nastavit server správně. Stejně tak je důležité pravidelně aktualizovat. Pokud nechcete, aby byly služby přístupné z Internetu, nastavte adekvátně příslušné služby (zakažte jim "poslouchat" na síti) nebo firewall.

6.7 Správa paměti

GNU/Linux využívá jak fyzickou operační paměť typu RAM, tak odkládací prostor na pevném disku. Strategií tohoto systému je maximálním možným způsobem využít volné RAM. Diskové vyrovnávací paměti (cache) se mohou dynamicky roztáhnout a

zabrat celou volnou RAM, je-li třeba. Přirozeně, potřebuje-li některý program další paměť, disková cache se zase zmenší.

Odkládací prostor (swap) v GNU/Linuxu zpravidla existuje v podobě samotného diskového oddílu. Je sice možné jako swap použít i obyčejný soubor, ale oddíl plní svou funkci mnohem lépe (rychleji). Ačkoliv se může zdát používání swapu v dnešní době jako zbytečné, jelikož mají počítače dostatek volné RAM, není tomu tak. Démon *kswapd* zjišťuje aktivitu procesů, a je-li proces dlouho neaktivní, přesune jeho data z RAM do swapu, čímž zvýší dostupnou RAM pro ostatní procesy a diskové cache.

6.7.1 Správa swapu

Jelikož se jedná o prokročilé téma, už si nevystačíme s grafickým prostředím, ale budeme muset pracovat s příkazovou řádkou. Swap můžete za běhu systému odstavit, přidat další či měnit jeho prioritu. Swap se vytvoří na blokovém zařízení (třeba `/dev/hda3`) takto:

```
mkswap /dev/hda3
```

Zařadí se do používání takto:

```
swapon /dev/hda3
```

A vyřadí se takto:

```
swapoff /dev/hda3
```

6.7.2 Swap v souboru

Je samozřejmě možné vytvořit swap v souboru. Pochopitelně to má smysl pouze, pokud víte, co děláte. Jste-li začátečníci, tak se této kapitolce obloukem vyhněte.

K vytvoření swapu ze souboru lze použít modul loop, který umí ze souboru udělat blokové zařízení. Nejprve vytvoříme vhodný soubor:

```
dd if=/dev/zero of=soubor_swap bs=1M count=512
```

Tento příkaz slouží ke kopírování dat z jednoho souboru do druhého. Jako vstupní soubor (parametr if) použijeme generátor nul `/dev/zero`, jako výstupní soubor (parametr of) použijeme náš budoucí soubor se swapem. Nastavíme velikost bloku pro čtení a zápis na 1MB a určíme, že se překopíruje 512 bloků, tedy 512 MB. Nyní zavedeme příslušný modul:

```
modprobe loop
```

Použijeme program **losetup** k propojení blokového zařízení `/dev/loop0` s naším souborem:

```
losetup /dev/loop0 soubor_swap
```

Následně vytvoříme swap na daném blokovém zařízení:

```
mkswap /dev/loop0
```

A zařadíme jej do používání:

```
swapon /dev/loop0
```

Vyřadit jej z používání můžeme pomocí:

```
swapoff /dev/loop0
```

Zrušit "spojení" mezi blokovým zařízením `/dev/loop0` a naším souborem můžeme příkazem:

```
losetup -d /dev/loop0
```

6.7.3 Když dojde paměť

Jak se ale systém zachová, když paměť opravdu dojde, a nebude místo ani v RAM, ale ani ve swapu? Linuxové jádro používá techniku zvanou "memory overcommiting", která programům dovolí říci si o více paměti než je k dispozici. Je to dáno tím, že si programy mnohdy říkají o více paměti, než pak skutečně využijí. Jenomže, co se stane, když paměť dojde? Pak nastupuje nefalšovaný zabiják, *oom-killer*, rutina, která se pokusí odhadnout, který program je nejméně důležitý, a ten sestřelí.

Ono je ale velice těžké přesně odhadnout, který program je vhodné sestřelit a který ne. Takže se často stává, že zabiják sestřelí proces, který potřebujeme zachovat, zatímco ten, který situaci způsobil, zůstane nedotčen. Toto chování můžeme ovlivnit. Kromě možnosti přeprogramovat danou rutinu je možné použít rozhraní jádra v souboru `/proc/sys/vm/overcommit_memory` (povolení memory overcommiting, hodnota 1 povolí, hodnota 0 zakáže). *Overcommit memory* tedy zakážeme pomocí:

```
echo "0" > /proc/sys/vm/overcommit_memory
```

Nebo elegantněji:

```
sysctl -w vm.overcommit_memory=0
```

Permanentní nastavení můžeme provést v souboru `/etc/sysctl.conf`, kam zapíšeme následující rádku:

```
vm.overcommit_memory=0
```

6.8 Konfigurační soubory

GNU/Linux uchovává svou konfiguraci v adresáři `/etc`, v podobě obyčejných textových konfiguračních souborů. K většině těchto souborů existují manuálové stránky, takže není problém si patřičnou dokumentaci vyhledat, když je to potřeba. Struktura

Adresář	Obsah
`/etc/X11`	konfigurace grafického rozhraní (hlavním konfiguračním souborem je `xorg.conf` nebo `XF86Config-4`)
`/etc/init.d`	skripty pro obsluhu démonů (mívají parametry `start`, `stop` a `restart`)
`/etc/cron.*`	skripty nebo symbolické odkazy na skripty, které se mají pouštět pomocí cronu (daily - *denně*, *hourly* - každou hodinu, *monthly* - měsíčně, *weekly* - týdně)
`/etc/cups`	konfigurace tiskového serveru CUPS (hlavním konfiguračním souborem je `cupsd.conf`)
`/etc/ppp`	konfigurace služeb point-to-point protokolu (připojení přes modem)
`/etc/rc*d`	symbolické odkazy na skripty, které se mají zavést v jednotlivých úrovních běhu systému (runlevelech)
`/etc/skel`	"kostra", která se překopíruje do domovského adresáře právě založeného uživatele

Tabulka 6.1: Adresáře v `/etc`

Soubor	Obsah
`/etc/fstab`	konfigurace souborových systémů, které se mají připojit při startu systému
`/etc/hostname`	název počítače
`/etc/hosts.allow`	seznam počítačů, kterým je povolen přístup v rámci TCP-wrapperu
`/etc/hosts.deny`	seznam počítačů, kterým je odepřen přístup v rámci TCP-wrapperu
`/etc/inittab`	popisuje procesy, které se mají spustit při startu systému
`/etc/modules`	seznam modulů, které mají být zavedeny při startu systému
`/etc/motd`	motiv dne, hláška, která se vypíše uživateli na obrazovku po přihlášení do textového režimu
`/etc/resolv.conf`	velmi důležitý soubor pro funkci sítě, je to seznam nameserverů, které se mají použít
`/etc/sudoers`	seznam uživatelů, kteří mají oprávnění používat příkaz sudo
`/etc/passwd`	seznam uživatelů s jejich UID, GID, uživatelským jménem, adresářem a přihlašovacím shellem
`/etc/shadow`	zašifrovaná (nebo zahashovaná) hesla uživatelů a několik dalších podstatných nastavení o heslech
`/etc/group`	seznam skupin a uživatelé do nich patřící

Tabulka 6.2: Soubory v `/etc`

adresáře `/etc` bývá odlišná v jednotlivých distribucích, nicméně je tu řada společných bodů, které si probereme. Specifika struktury adresáře `/etc` by měla obsahovat dokumentace ke zvolené distribuci.

Soubory `hosts.allow` a `hosts.deny` se řídí jistá knihovna, kterou využívá řada síťových služeb (např. *OpenSSH*, *nfs*, apod.). V těchto souborech je možné specifikovat, které počítače smí ke kterým ze služeb přistupovat. Ano, hádáte správně, tohle lze řešit i v rámci firewallu.

Možná doplním ještě jeden velmi dobrý praktický trik, pokud budete hledat konfigurační soubor, o kterém víte, že obsahuje určitou hodnotu, použijte funkci *Najít soubor* v Midnight Commanderu nebo jinde, kde specifikujete to, co by měl obsahovat. Když jsem začínal, tento postup se mi velice osvědčil.

6.8.1 Práce s konfiguračními soubory

Ještě než začneme pracovat s nějakým konfiguračním souborem, měli bychom se podívat na jeho dokumentaci. Ta se může ukrývat buď v manuálových stránkách, nebo může být k dispozici přímo v daném konfiguračním souboru ve formě komentářů. Komentář je uvozen křížkem (#) a to, co za ním následuje, nebere program při procházení konfiguračním souborem v potaz. Komentářů můžeme využít i při editaci konfiguračního souboru a k dokumentaci změn.

Před úpravou konfiguračního souboru je dobré si vytvořit jeho zálohu, pro každý případ. Práci s komentáři si ilustrujeme na následujících dvou příkladech. Máme následující řádku v souboru /etc/fstab:

```
/dev/sda2 /media/sda2 ext3 defaults 0 2
```

My ale nechceme, aby se /dev/sda2 mountovalo při startu počítače. Ovšem nejsme si až tak jisti, že to někdy nebudeme znovu potřebovat, a proto řádku nesmažeme, pouze zakomentujeme:

```
#/dev/sda2 /media/sda2 ext3 defaults 0 2
```

Uvažme situaci, že chceme pouze změnit parametry připojení souborového systému, ale nevíme jistě, co to udělá. Proto si původní řádku zálohujeme a na nové budeme experimentovat:

```
#/dev/sda2 /media/sda2 ext3 defaults 0 2
/dev/sda2 /media/sda2 ext3 ro 0 0
```

Nezřídka se nám může hodit opatřit jistou změnu komentářem, třeba abychom ji opět našli, a po čase opět pochopili, proč jsme to vlastně tenkrát dělali:

```
#2006-08-22 (Michal): Pryc s tim, na /dev/sda2 je ted Gentoo
#/dev/sda2 /media/sda2 ext3 defaults 0 2
```

6.8.2 Uživatelský profil

Dosud jsme probírali konfiguraci systému jako takového, ale kam se ukládají nastavení programů, které spustíte? Ta se ukládají do vašeho domovského adresáře, zpravidla

jako skryté soubory (v unixových systémech jsou skryté soubory ty, které obsahují tečku na začátku svého jména). I tady platí, že se převážně jedná o textové soubory, které lze upravovat, zálohovat či přenášet na jiné unixové systémy.

V extrémně nepravděpodobném případě se může stát, že si nějaký špatně napsaný program uloží konfiguraci, která způsobí, že spadne při příštím spuštění. V takovém případě postačí najít příslušný konfigurační soubor a buď jej odstranit, nebo někam přesunout, program si ho vytvoří automaticky při dalším spuštění.

6.9 Správa filesystémů

6.9.1 Organizace dat na disku

Prvních 512 bytů na pevném disku tvoří MBR, kde se nachází zavaděč operačního systému (446 bytů), následuje tabulka rozdělení disku (partition table), kde jsou po 16 bytech uloženy informace o každém ze čtyřech primárních oddílů. MBR je zakončeno 2 byty tvořící tzv. magic number.

Pevný disk je rozdělen na jednotlivé oddíly (partitions). Omezení v podobě 4 primárních oddílů se obchází tak, že se jeden z primárních oddílů nastaví jako rozšířený, a v jeho útrobách pak mohou vznikat další, rozšířené oddíly. Podstatné je, že v každém oddílu může být jiný souborový systém.

Mezi důsledky této šedé teorie patří několik důležitých praktických poznatků:

- tabulku rozdělení disku, zavaděč nebo celou MBR lze zálohovat a podle potřeby obnovit samotnou změnou tabulky rozdělení disku nedojde ke ztrátě dat (souborové systémy zůstanou netknuté) a je možné je zachránit obnovením původní tabulky rozdělení disku (třeba programem **gpart** či **testdisk**)

- změnu tabulky rozdělení disku lze realizovat za běhu systému (GNU/Linuxu), ale po její změně je před příslušnými operacemi (vytváření souborových systémů v nových oddílech, apod.) třeba restartovat systém

6.9.2 Filesystém

Filesystém, alias souborový systém, je metoda ukládání dat na pevný disk tak, abychom byli schopni tato data opět přečíst zpět. Zdá se to možná banální, ale je to extrémně důležité. Díky souborovým systémům můžeme ukládat svoje data do souborů v adresářové struktuře, a hlavně, opět je přečíst. Ovšem souborové systémy jsou v dnešní době přeci jen složitější. Důraz je kladen i na bezpečnost uložení dat, rychlost přístupu k datům a na řadu dalších faktorů.

6.9.3 Žurnál

Jednou z podstatných vlastností moderních souborových systémů je systém pro zajištění konzistence dat v případě výpadku. Žurnál je, zjednodušeně, oblast, kam se zapisují informace o právě prováděných transakcích (zápis dat do souboru, apod.). V případě výpadku se ze žurnálu zjistí, které operace proběhly a které nikoliv. Tak lze

velmi snadno dostat souborový systém opět do konzistentního stavu, a to bez nutnosti provádění složitých a zdlouhavých prověřování.

Mezi žurnálovací souborové systémy patří v GNU/Linuxu *ext3*, *reiserfs*, *jfs*, *xfs*. Naopak *ext2* žurnálovací není, a proto jej doporučuji nepoužívat. Říkám to zejména proto, že ve starších dokumentacích bývá *ext2* často zmiňován. Dnes už však není důvod nepoužít *ext3*. Zejména pak proto, že *ext2* a *ext3* jsou mezi sebou kompatibilní a jeden lze přeměnit na druhý a zpět (zprovozněním nebo zrušením žurnálu pomocí *tune2fs*).

6.9.4 Adresářový strom

V unixových systémech jsou filesystémy připojovány do jednotné adresářové struktury. Na disku můžeme mít tedy teoreticky řadu oddílů, přičemž na každém bude jiný souborový systém. Ale z hlediska uživatele budou Linuxové souborové systémy jsou založeny na unixových souborových systémech.

6.9.5 Mountování

Pokud chceme připojit nějaký souborový systém do adresářového stromu, použijeme k tomu příkaz mount nebo adekvátní grafické nástavby (viz dokumentace k distribuci). Toto budeme pochopitelně realizovat pouze v situaci, kdy nedojde k automatickému připojení, jak bývá obvyklé pro řadu přívětivých distribucí. Příkaz mount má celkem jednoduchou syntaxi:

```
mount -t filesystem /dev/zarizeni /mnt/adresar
```

Zpravidla můžeme volbu `-t` (specifikace typu souborového systému) s parametrem určujícím souborový systém vypustit, mount se pokusí typ souborového systému určit sám. Stačí tedy použít zkrácenou syntax:

```
mount /dev/zarizeni /mnt/adresar
```

Tento příkaz připojí blokové zařízení `/dev/zarizeni` na adresář `/mnt/adresar`. Z praktického hlediska, pokud budu chtít připojit souborový systém (*ext3*) na třetím (primárním) oddíle primárního IDE pevného disku na sekundárním IDE řadiči do adresáře `/home/repository`, použiji příkaz:

```
mount -t ext3 /dev/hdc3 /home/repository
```

Můžeme také použít zkrácenou verzi s autodetekcí souborového systému:

```
mount /dev/hdc3 /home/repository
```

Někdy se můžou hodit i určité volby, které jsou k dispozici pod parametrem `-o`, třeba následující příkaz připojí stejný souborový systém pouze pro čtení:

```
mount -t ext3 -o ro /dev/hdc3 /home/repository
```

Další možnosti viz manuálová stránka k příkazu mount.

```
man mount
```

Souborový systém můžeme odpojit pomocí příkazu umount, takto:

```
umount /dev/hdc3
```

Pokud na souborovém systému, který chceme odpojit, provádí nějaký program nějakou činnost, nepůjde odpojit. Nejjednodušším řešením takové situace je daný program ukončit nebo v něm přejít do jiného adresáře. Pokud nevíme, který program zrovna na souborový systém přistupuje, použijeme příkaz **lsof**, který nám to zjistí:

```
lsof /dev/hdc3
```

Nebo můžeme riskovat a nechat systém násilně ukončit všechny programy pracující s daným zařízením (užívejte velmi opatrně):

```
kill -9 $(lsof -t /dev/zarizeni)
```

Souborový systém lze odpojit i násilím, ale to je dobré provádět pouze, pokud jsou všechny osatatní možnosti vyčerpány:

```
umount -f /dev/hdc3
```

Ještě lze použít parametr `-l`, který souborový systém okamžitě vyřadí z hierarchie, ale reference na něj odstraní později:

```
umount -l /dev/hdc3
```

6.9.6 `/etc/fstab`

Chceme-li připojovat nějaký souborový systém při každém startu, upravíme konfigurační soubor `/etc/fstab`. Jeho syntax je v podstatě velice jednoduchá, na každém řádku je záznam pro jeden souborový systém s následujícími parametry:

- soubor zařízení (např. "/dev/hda1")
- přípojný bod (např. "/mnt/hda1")
- souborový systém
- volby (oddělené čárkou)
- číslo používané programem dump (nula tady zcela postačí)
- číslo označující pořadí při kontrole filesystému

Typická řádka by tedy mohla vypadat nějak takto:

```
/dev/sda1 /media/card1 vfat ro,user,noauto 0 0
```

Výše zmíněná řádka by zajistila připojení zařízení `/dev/sda1` obsahující souborový systém typu vfat na přípojný bod `/media/card1` s volbami `ro` (připojení pouze pro čtení), `user` (povolí uživatelům připojovat souborový systém) a `noauto` (nepřipojí souborový systém hned po spuštění počítače. Prvního čísla bych si nevšímal a dal bych na jeho místo nulu (pokud nepoužíváte k zálohování program **dump**). Druhé číslo označuje pořadí při provádění kontroly souborového systému pomocí programu **fsck**. Nula značí vyjmutí z kontroly při startu počítače, jedničku by měl mít pouze kořenový souborový systém, pro ostatní je dvojka nebo nula.

6.9.7 Obnovení smazaného souboru

V zásadě, dostat smazaná data z linuxových oddílů je velmi těžké, až zhola nemožné. Existuje sice několik neoficiálních nástrojů, ale těmi lze napáchat více škody než užitku (je důrazně doporučena záloha oddílu před jejich použitím). Proto nemohu než doporučit důležitá data zálohovat a při mazání si dávat opravdu pozor.

Pokud si smažete omylem opravdu extrémně cenná data, jejichž zálohu nemáte k dispozici, můžete se pokusit data obnovit sami (viz dále) nebo obrátit na specializované společnosti zabývající se záchranou dat.

6.9.8 Oprava partition table

Pokud je poškozená tabulka rozdělní disku, nebo se nám podařilo ji upravit a následně zjistit, že jsme zapomněli zálohovat důležitá data, není ještě všechno ztraceno. Můžeme použít programy jako **gpart**[3] nebo **testdisk**[4], které jsou schopné poškozenou tabulku rozdělení disku rekonstruovat.

6.9.9 Oprava poškozeného filesystému

Je-li možné souborový systém, který chceme prověřovat, odpojit, učiníme tak. Pokud to možné není, musíme jej přepojit do režimu pouze pro čtení. Chceme-li to provést se souborovým systémem, který je používán, nezbyde než ukončit všechny procesy využívající daný souborový systém. Pokud chceme za běhu zkontrolovat kořenový oddíl, musíme přejít do jednouživatelského režimu pomocí příkazu `init 1`, následně souborový systém přemountujeme jenom pro čtení. Za předpokladu, že tím naším filesystémem je `/dev/hda1`, zapsali bychom:

```
mount -o remount,ro /dev/hda1
```

A teprve teď bychom se mohli pustit do kontroly souborového systému.

```
fsck /dev/hda1
```

Je-li souborový systém vážně poškozen, pak jej raději před pokusy o nápravu zálohujeme, třeba pomocí programu **dd**:

```
dd if=/dev/hda1 of=/mnt/sitovy_disk/zaloha_hda1
```

[3] http://www.stud.uni-hannover.de/user/76201/gpart/
[4] http://www.cgsecurity.org/wiki/TestDisk

Je-li poškozeno samotné zařízení, na kterém se souborový systém nachází, pak přidáme parametr `noerror` a specifikujeme rozumnou velikost bloku (v tomto případě 512 bytů):

```
dd if=/dev/hda1 of=/mnt/sitovy_disk/zaloha_hda1 noerror bs ↩
    =512
```

Máme-li připravený obraz poškozeného oddílu, můžeme na jeho kopii provádět záchranné operace. Můžeme se pokusit souborový systém opravit:

```
losetup /dev/loop0 /mnt/sitovy_disk/zaloha_hda1_kopie
```

Tento příkaz asociuje zařízení `/dev/loop0` s kopií našeho poškozeného souborového systému. To způsobí, že k němu budeme moci přistupovat jako k normálnímu blokovému zařízení. Toho využijeme a ihned na něm spustíme **fsck**:

```
fsck /dev/loop0
```

Fsck se pokusí opravit daný souborový systém. Když skončí, můžeme se pokusit opravený souborový systém připojit, tentokrát specifikujeme i souborový systém:

```
mount -t ext3 /dev/loop0 /mnt/zachrana
```

Pokud se nám oprava nepovede, jelikož je poškození souborového systému příliš veliké, zkusíme třeba program Foremost[5], který projde náš obraz a extrahuje z něj známé typy souborů. Můžeme samozřejmě zvolit i další software. V nejzazší nouzi se ještě můžeme obrátit na specializované společnosti zabývající se záchranou dat.

6.10 Správa jádra

Srdcem každého linuxového systému je kernel, jádro operačního systému. Linuxové jádro je modulární a široce konfigurovatelné. Modularitou mám na mysli LKM (Loadable Kernel Modules), vlastnost umožňující přímo za běhu vkládat nebo odebírat z kernelu jednotlivé moduly (modul může být ovladač hardwaru nebo jiná funkcionalita). Moduly jako takové naleznete v adresáři `/lib/modules/verze_kernelu` a lze je spravovat pomocí následujících příkazů:

- `lsmod` - vypíše zavedené moduly
- `modprobe` - zavede určitý modul
- `insmod` - umožňuje zavést modul ze souboru
- `rmmod` - odstraní modul z kernelu
- `depmod` - generuje závislosti mezi moduly

[5] http://foremost.sourceforge.net/

Více se dozvíte v manuálových stránkách jednotlivých příkazů. Obrazy jader (ano, kernelů můžete mít více a při startu systému se rozhodnout, který použijete) se nachází v adresáři `/boot`, zpravidla v podobě souborů začínajících na `vmlinuz`. Jsou zde uloženy i iniciální ramdisky, a to v podobě souborů zpravidla začínajících na `initrd`, patřících vždy konkrétnímu obrazu jádra. Samozřejmě ani kernel, ani inicální ramdisk se nemusí jmenovat tak, jak bylo uvedeno. Jejich zavádění je potom zajišťováno zavaděčem (LILO nebo GRUB). Konfiguraci jádra obecně lze rozdělit do dvou variant:

- úprava parametrů zkompilovaného jádra
- kompilace vlastního jádra

První ze jmenovaných případů se týká možností komunikovat s jádrem, předávat mu určité parametry (při bootu) nebo upravovat jeho funkcionalitu (při běhu). Parametry se jádru předávají před jeho nabootováním prostřednictvím zavaděče (LILO nebo GRUB). Funkce běžícího jádra se dají ovlivnit prostřednictvím souborového systému `/proc` nebo `/sys`.

Kompilace vlastního kernelu je možnost, jak si jádro přizpůsobit svým požadavkům a své architektuře. Optimalizací pro danou architekturu lze o něco zvýšit výkon systému. Kernel lze ale i opatchovat a implementovat do něj podporu pro dosud neoficální funkcionalitu, popřípadě touto cestou zajistit zprovoznění určitého hardwaru.

6.10.1 Typy jader

Existují dva typy kernelů, kernel vanilla, tedy čistý, neopatchovaný, oficiální kernel, který můžete stáhnout z kernel.org, a distribuční kernel, tedy ten, který za vás již sestavil správce vaší distribuce. Distribuční jádra bývají různě patchované a bývají nastavené tak, že se spustí takřka na jakémkoliv hardwaru. Používají k tomu právě LKM a možnost zavést moduly podle toho, jaký hardware je v počítači nalezen. To značně zvyšuje robustnost systému, který je pak možné přenášet mezi počítači s minimálními změnami.

Měl bych dodat, že k hlídání změn v hardwaru při běhu systému (např. připojení USB zařízení) se používá speciální démon - hotplug (momentálně již ve většině distribucí nahrazen systémem udev). Ten hlídá, co se změní a adekvátně se přizpůsobí (zavede patřičné moduly).

Poněkud podrobnější pohled do linuxového kernelu nabízí dokument *Kernel Hacking HOWTO*[6] (doporučuji spíše programátorům).

6.10.2 Kompilace kernelu

Předně je třeba říci, že kompilaci kernelu většinou nebudete potřebovat provádět. Doby, kdy to bylo nutné, jsou již dávno pryč. Dnes máte v linuxových distribucích k dispozici modulární, předem nastavená a vyladěná jádra, která jsou navíc v případě objevení nějakého problému patchována a vy si prostřednictvím balíčkovacího systému můžete stáhnout aktualizaci.

[6] http://www.kernelhacking.org/docs/kernelhacking-HOWTO/

Kompilace kernelu je vhodná, pokud toužíte po optimalizaci jádra přímo pro váš počítač (ať již optimalizace pro použitý procesor, nebo pro použitý hardware - zvolení jenom těch modulů, a vlastností, které chcete), popřípadě v oněch velmi vzácných případech, kdy distribuční jádra nemají něco, co vy nutně potřebujete.

Pro kompilaci kernelu potřebujete kompilátor (gcc), eventuelně některé další nástroje (něco pro snadné vytváření initrd, třeba initrd-tools). Další ingrediencí jsou zdrojové soubory jádra. Distribuční jádro má zdroj v balíčku kernel-source (může se ale jmenovat i jinak), vanilla jádro si můžete stáhnout z www.kernel.org.

Postup kompilace je následující:

- příprava zdrojáků
  - získání rootovských oprávnění
  - rozbalení archívu se zdrojáky jádra
  - otevření hlavního adresáře se zdrojáky (např. `kernel-source-2.6.16`)
  - (aplikace eventuelních patchů)
  - příprava pro kompilaci - zapsání příkazů:

```
make mrproper
```

```
make clean
```

  - lze použít existující nastavení jádra jako základ pro nastavování nového:

```
cp /boot/config-verze_jadra ./.config
```

```
make oldconfig
```

- Kompilace některého konfiguračního programu (vyberte si jednu z možností):
  - `make config` (otázka/odpověď v textovém režimu)
  - `make menuconfig` (menu v textovém režimu)
  - `make xconfig` (grafický konfigurátor)
  - `make gconfig` (grafický konfigurátor)
- konfigurace jádra (viz níže)
- kompilace jádra
  - jádro řady 2.4:

```
make dep
```

```
make bzImage
```

```
make modules
```

  - jádro řady 2.6:

```
make
```

```
make modules_install
```

- zkopírování obrazu jádra do `/boot`:

```
cp arch/i386/boot/bzImage /boot/vmlinuz-x.y.z.a
```

- Úprava konfigurace zavaděče (u Lila pak nezapomeňte provést jako root příkaz `lilo`, u Grubu nemusíte dělat nic)
- reboot a otestování, zda-li jádro funguje

Více informací naleznete v odkazech v závěru kapitoly. Nejnáročnější procedurou je vlastní konfigurace jádra, tedy výběr toho, co bude do kernelu zakompilováno a co ne. Je třeba si dávat pozor, abyste do jádra zahrnuli podporu všeho důležitého (grafika, myš, klávesnice, apod.). Pokud nepoužíváte iniciální ramdisk (viz níže), pak musíte zajistit, aby ovladače řadiče a souborových systémů byly v jádře zakompilovány na pevno (ne jako moduly).

Pokud se vám to napoprvé nepodaří, nezoufejte a zkuste konfiguraci poopravit (nepoužívejte v takovém případě příkaz make mrproper, jelikož ten vám vymaže uloženou konfiguraci). Ke zjištění toho, co všechno musíte do jádra zahrnout, vám mohou sloužit příkazy jako **lsmod**, který vypíše zavedené jaderné moduly, popřípadě **lspci** či **dmesg**. Hodí se také dostupné informace v `/proc`.

6.10.3 Kompilace jaderného modulu

Může se stát, že budete potřebovat ručně zkompilovat nějaký jaderný modul pro vaše jádro, třeba s proprietárním ovladačem k některému kousku hardware, který díky licenci nemůže být součástí jádra. Samozřejmě předpokládám, že máte k dispozici nějaký podrobný návod, který vám řekne, co a jak nastavit.

Kromě standardní kompilační sady (kompilátor **gcc**) budete potřebovat i hlavičkové soubory jádra. Pokud používáte distribuční jádro, pak je zpravidla najdete v balíčku, jehož název bude vypadat podobně jako `kernel-headers-x.y.z.a` (kde `x.y.z.a` je verze jádra). Pokud jste si zkompilovali vlastní jádro, pak je máte k dispozici v adresáři se zkompilovaným jádrem (zpravidla někde v `/usr/src`, ale to byste měli vědět, když jste si ho zkompilovali sami).

Jeden problém, který se zde může vyskytnout, je situace, kdy budete jaderný modul kompilovat pomocí jiné verze kompilátoru než samotné jádro. V takovém případě nemusí jaderný modul fungovat. Pokud fungovat nebude, a nepodaří se vám jej zavést ani pomocí `insmod -f soubor_s_modulem`, zřejmě vám nezbyde než zkompilovat vlastní jádro. Distributoři na toto však zpravidla pamatují a většinou to lze vyřešit aktualizací balíčku s jádrem.

6.10.4 Iniciální ramdisk

Jak jsme si říkali, moduly jsou umístěny v adresáři `/lib/modules`. Pokud máte jako modul zakompilovanou podporu souborového systému (nebo zařízení), na kterém se tento adresář nachází, pak za normálních okolností kernel nenabootuje a zpanikaří. Neumí totiž řešit situaci "Co bylo dříve, slepice nebo vejce?". K tomu, aby se mohl dostat k modulům potřebuje "vidět" záznamové zařízení a "rozumět" souborovému systému, na kterém se moduly nachází, ale pokud jsou ovladače (drivery) onoho souborového systému nebo daného zařízení zakompilovány jako moduly, pak se k nim nedostane a nemá šanci to přeusmívat.

K řešení tohoto problému slouží initrd, což je malý, zkomprimovaný souborový systém, který se zavede za kernel do paměti, ten jen pak rozbalí, připojí na kořenový souborový systém a postupuje podle jeho startovacího skriptu. Tento souborový systém obsahuje mj. i moduly potřebné k zavedení systému. Vytváří se příkazem `mkinitrd`. U novějších distribucí to je `mkinitcpio`.

6.11 Zdroje a odkazy

6.11.1 Správa softwaru

- LinuxSoft.cz, František Huček, Přecházíme z Windows - instalace software v Linuxu
- ABC Linuxu, Vlastimil Ott, Na co se často ptáme: Balíčkovací systémy
- ABC Linuxu, Stanislav Valasek, Zdroje balíčkov pre Ubuntu
- ABC Linuxu, Martin Fiala, Balíčkovací systém Mandrake Linuxu
- LinuxExpres, Anička Bernáthová, Instalace softwaru v openSUSE
- Petr Krčmář, Nebojíme se kompilace
- Ondřej Krčmář, Balíčkovací systém Gentoo Linuxu

6.11.2 Procesy

- Čtenáři ABC Linuxu, Procesy
- Rastislav Stanik, Signály
- Matouš Jan Fialka, htop: top na druhou

6.11.3 Filesystémy

- Čtenáři ABC Linuxu, Souborový systém
- Wikipédie (anglická), Filesystem

- Wikipédie (anglická), Master Boot Record
- Root.cz, Petr Krčmář, Na co se často ptáme: fstab
- ABC Linuxu, Leoš Literák, FAQ, Smazal jsem důležitá data, jak je mohu obnovit?
- ABC Linuxu, Leoš Literák, FAQ, Jak připojit FAT oddíl?

6.11.4 Kernel

- Root.cz, Michal Ludvig, Jak funguje initramdisk
- Wikipédie (anglická), Linux kernel
- Wikipédie (anglická), Linux kernel oops
- Wikipédie (česká), Linuxové jádro
- Ivan Bowman, Conceptual Architecture of the Linux Kernel
- kernelnewbies.org, Linux kernel API
- Knoppix cheat codes

6.11.5 Ostatní

- ABC Linuxu, seznam článků z rubriky sítě
- ABC Linuxu, Luk, Memory overcommitting - ráj nebo peklo?
- LinuxMM, OOM Killer
- Wikipédie, Wikibooks, Guide to Unix - Files

Kapitola 7

Základy práce s příkazovou řádkou

Proč pracovat s příkazovou řádkou? Grafická prostředí a grafické aplikace mohou být příjemné a pohodlné, ovšem jen do té doby, dokud po nich chceme to, co programátory napadlo. Ve chvíli, kdy chceme něco extra, může se z příjemného pobytu v grafickém prostředí stát noční můra. To, co bychom v příkazové řádce vyřešili jedním vhodně sestaveným příkazem, můžeme v grafickém prostředí realizovat hodiny a hodiny.

Práce s příkazovou řádkou není sice úplně jednoduchá záležitost, ale jakmile se s ní naučíte pracovat, získáte opravdu mocného spojence. Uživatelé bez příslušných znalostí (nebo uživatelé neunixových operačních systémů) vás pak možná začnou považovat za opravdového mága.

Co všechno lze s pomocí příkazové řádky realizovat? Mnohé. Potřebujete zjistit, co za objemné soubory máte ve svém domovském adresáři? Tento příkaz vypíše všechny soubory větší než 100MB ve vašem domovském adresáři.

```
find ~/ -size +100M
```

Máte *jpg* soubory rozeseté v komplexní adresářové struktuře a chcete je všechny přesunout do jednoho adresáře? Můžete to dělat ručně, nebo použít příkaz:

```
find ~/ -name "*.jpg" -type f -exec mv {} /mnt/skladiste \;
```

Chcete si zálohovat MBR na disku *sda*?

```
dd if=/dev/sda of=zaloha.mbr bs=512 count=1
```

Chcete bezpečně vymazat pevný disk *sda*, aby ste si mohli být jisti, že z něj nepůjdou obnovit vaše soukormá data?

```
shred /dev/sda
```

Naučit se efektivně využívat příkazovou řádku není jednoduché, navíc je třeba to vhodně spojit se znalostí fungování počítače a vnitřností GNU/Linuxu. Kupříkladu,

výše zmíněný příklad se zálohou MBR vyžaduje znalost symboliky zařízení v GNU-/Linuxu a unixové filosofie *vše je soubor*. Celý pevný disk je v našem příkladu reprezentován souborem `/dev/sda`, se kterým lze zacházet jako se kterýmkoliv jiným souborem, tj. číst z něj, zapisovat do něj, apod. Víme-li tohle a víme-li, že MBR se nachází v prvních 512 bytech pevného disku, můžeme použít program *dd*, který nám obsah MBR zapíše do souboru.

Jistě, na řadu úkonů jsou k dispozici různé speciální uživatelsky přívětivé nástroje (zejména pro systém Windows je typická existence tisíců takových jednoúčelových nástrojů, zpravidla proprietárních, komerčních a nezřídka s porcí malware navíc). Nepochybně existují nějaká uživatelsky přívětivá řešení pro zálohu MBR či bezpečné vymazání pevného disku, nevyžadující znalost příkazové řádky ani fungování počítače. Nikdy však nebudou existovat nástroje na splnění všech vašich požadavků, zejména těch komplexních.

Výhoda schopnosti efektivně využívat příkazovou řádku tkví v tom, že jste schopni sestavit jednoduché i komplexní řešení vašich požadavků, a to přesně na míru. Do této schopnosti je však třeba investovat nějaký čas.

7.1 Úvod

Rozlišujme mezi textovým režimem, terminálem, příkazovou řádkou a interpretem příkazové řádky. *Textovým režimem* označuji implicitní prostředí GNU/Linuxu mimo grafické rozhraní, tj. mimo X server. Přívětivé distribuce se obvykle snaží textový řežim zakrýt, avšak i když systém nastartuje do grafického rozhraní, aniž byste zahlédli textový režim, můžete se do něj snadno přepnout prostřednictvím Ctrl-Alt a některé z funkčních kláves (zpravidla F1 až F6). Pomocí této klávesové zkratky se dostanete z grafického prostředí do textového režimu, na některý z virtuálních terminálů (podle toho, jakou funkční klávesu jste použili). V textovém režimu se mezi virtuálními terminály přepínáte pomocí *Altu* a funkční klávesy. Na jednom virtuálním terminálu běží grafické rozhraní (zpravidla na sedmém). Přepnete-li se tedy pomocí Alt-F7 na sedmý terminál, ocitnete se zpět v grafickém prostředí.

Terminálem [1] budeme označovat program, prostřednictvím kterého je možné pracovat s příkazovou řádkou. V grafickém prostředí máte k dispozici celou řadu terminálů, třeba Konsole nebo gnome-terminal. Tyto programy přirozeně podporují ovládání myší, což je dobré zejména pro označování a kopírování textu.

Interpret příkazové řádky nebo *shell* je program, který interpretuje vaše příkazy a vykonává je. V GNU/Linuxu máte na výběr z mnoha shellů, přičemž výchozí je *bash*. Na něj se zaměříme.

Nuže, nyní již víme, jak se dostat k příkazové řádce. Buď přihlášením na některý z virtuálních terminálů v textovém režimu nebo prostřednictvím nějakého terminálu v grafickém prostředí.

[1] http://en.wikipedia.org/wiki/Computer_terminal

7.2 Základy práce s řádkou

Nejprve bych rád probral několik záležitostí, které značně usnadňují práci. Tou první je program s názvem *Midnight Commander*, jehož instalaci doporučuji provést, i když pobyt v příkazové řádce neplánujete. Midnight Commander, který se spouští příkazem `mc`, je diskový manažer podobný Norton Commanderu (nebo Total Commanderu, chcete-li). Mnoho funkcí, které budeme realizovat pomocí příkazů, lze provést pomocí tohoto programu mnohem snadněji.

Pokud budete pracovat v shellu, hodí se vědět o *historii příkazů*, ve které se můžete pohybovat pomocí šipek nahoru a dolů. Můžete tak listovat mezi dříve spouštěnými příkazy a ty upravovat. Přirozeně, když fungují šipky nahoru a dolů, fungují i šipky vlevo a vpravo, pomocí kterých se můžete pohybovat v textu aktuálně zadávaného příkazu a libovolně jej upravovat. Samozřejmě fungují i klávesy jako *Delete* nebo *Backspace*.

Další podstatnou věcí, o které je dobré vědět, je vlastnost zvaná *doplňování*, v Bashi se realizuje pomocí tabulátoru. Když potřebujete doplnit jméno souboru nebo cestu k němu, můžete si tak značně pomoci. Zadáte prvních pár písmen a stisknete tabulátor. Bash doplní zbytek. V případě, že vámi zadnému fragmentu odpovídá více souborů, opakovaný stisk tabulátoru je zobrazí. Po doplnění dostatečného počtu znaků a stisknutí tabulátoru se doplní konkrétní soubor.

7.3 Jdeme na průzkum

Jak víme, v GNU/Linuxu je vše uloženo v jisté hierarchické adresářové struktuře, která začíná kořenovým adresářem a rozvětvuje se do jednotlivých podadresářů. Pokud se dostaneme k příkazové řádce, zpravidla se ocitneme ve svém domovském adresáři. První věcí, kterou se naučíme, je pohyb po adresářové struktuře.

Abychom mohli vyrazit na průzkum, musíme si s sebou vzít příslušné vybavení, modul GPS umožňující zjistit kdekoliv naši pozici (příkaz **pwd**), svítilnu k prohledávání momentálního okolí (příkaz **ls**) a dopravní prostředek (příkaz **cd**). Posledním předpokladem je schopnost navigace, tedy povědomí o adresářové struktuře a znalost tzv. absolutních a relativních cest.

To není nic komplikovaného. Kupříkladu, mějme soubor `/var/log/messages` (povšimněte si, že jednotlivé položky v cestě se oddělují lomítkem). Tato cesta začíná lomítkem (kořenovým adresářem), a proto se jí říká absolutní cesta. Pokud se však nacházíme v adresáři `/var`, je relativní cesta k tomuto souboru `log/messages`. Relativní cesta je cesta k souboru z aktuálního adresáře. Jsme-li tedy ve `/var`, nemusíme vypisovat absolutní cestu, můžeme použít cestu relativní.

Každý adresář má vždy alespoň dvě položky, odkaz sám na sebe (tečku) a odkaz na předchozí adresář (dvě tečky). Pokud se tedy nacházíme v adresáři `/var/tmp`, můžeme se k příslušnému souboru dostat přes odkaz na předchozí adresář, tedy takto - `../log/messages`. Pokud bychom se nacházeli v adresáři `/var/tmp/mike`, měla by relativní cesta o jeden skok na předchozí adresář více - `../../log/messages`.

Nyní se již můžeme pustit do vlastního průzkumu. Začneme tím, že zjistíme, kde jsme:

```
[michal@griffon ~]$ pwd
```

```
home/michal
```

Jak vidíme, nacházíme se v adresáři `/home/michal`. Nyní se rozhlédneme:

```
[michal@griffon ~]$ ls
linuxbook.xml Desktop
```

Teď už víme, že se v našem domovském adresáři nacházejí tyto položky. A na závěr se necháme přepravit do adresáře `/tmp` prostřednictvím relativní cesty:

```
[michal@griffon ~]$ cd ../../tmp
```

Přirozeně jsme mohli použít absolutní cestu, což by bylo i rychlejší:

```
[michal@griffon ~]$ cd /tmp
```

7.3.1 Příkaz `ls` podrobněji

Program `ls` má některé zajímavé parametry, které nám umožní získat více informací. Volby unixových programů začínají vždy pomlčkou a mívají dvě formy, krátkou (např. `-a`) a dlouhou (např. `--all-files`). Tyto formy jsou zaměnitelné, použijte tu, která se vám lépe píše nebo pamatuje. Volby mohou mít i parametry, které jsou u krátké formy zpravidla specifikovány po jedné mezeře (např. `-f soubor.txt`, zatímco u dlouhé mohou být odděleny rovnítkem (např. `--file=soubor.txt`. Různé programy mají různé parametry a volby, o kterých se dozvíte v manuálových stránkách (příkaz `man`). Parametry a volby se oddělují mezerou. Názorněji to uvidíme na příkladech.

Parametrem programu `ls` je adresář, jehož obsah chcete vypsat. Pokud jej nespecifikujeme, vypíše se aktuální adresář. Pokud bychom tedy chtěli vypsat obsah adresáře `/bin`, zapsali bychom:

```
[michal@griffon ~]$ ls /bin
```

Tou nejpoužívanější volbou pro program `ls` je `-l`, která způsobí, že se vše vypíše v tzv. dlouhém formátu, kde získáte přehled o vlastníkovi a skupině příslušné danému souboru, jeho velikost a přístupová práva. Vyzkoušejme si to:

```
[michal@griffon ~]$ ls -l
drwx------ 16 michal users  4096 2007-10-05 22:03 Desktop
-rw-r--r--  1 michal users 12578 2007-10-05 10:22 linuxbook. ↩
    xml
```

Jak vidíme, ve výpisu máme k dispozici u jednotlivých položek přístupová práva, jméno vlastníka a skupiny, velikost, datum a čas poslední změny. Vidíme i typ jednotlivých položek (první znak na řádce), u položky `Desktop` vidíme, že se jedná o adresář (`d` jako *directory*). Normální soubor je označen pomlčkou, `b` by označovalo blokové zařízení, `c` znakové zařízení a `l` symbolický link.

Poslední důležitou volbu, kterou proberu, je `-a`. Tato volba zařídí vypsání všech souborů, i skrytých (tj. soubory začínající tečkou). Jednotlivé volby (resp. pouze ty,

které neočekávají parametr) lze v případě `ls` a řady dalších unixových utilit kupit za jedinou pomlčku, takto:

```
[michal@griffon ~]$ ls -la
```

Jednotlivé volby jde samozřejmě rozepsat:

```
[michal@griffon ~]$ ls -l -a
```

To je ale zbytečná práce navíc, když to jde výše uvedeným způsobem.

7.4 Zkoumáme podrobněji

V unixových systémech se typ souboru determinuje jinak než pomocí přípon, jak je tomu v některých jiných systémech. Soubory mají pouze jméno, přičemž přípona je jeho součástí, pokud je. Samozřejmě být nemusí. Typ souboru můžete zjistit pomocí příkazu `file`:

```
[michal@griffon ~]$ file linuxbook.xml
linuxbook.xml: XML 1.0 document text
```

Podrobnější informace k určitému souboru můžete zjistit pomocí příkazu `stat`:

```
[michal@griffon ~]$ stat linuxbook.xml
  File: 'linuxbook.xml'
  Size: 281722      Blocks: 560        IO Block: 4096    ↩
     regular file
Device: fe02h/65026d  Inode: 2387440     Links: 1
Access: (0666/-rw-rw-rw-)  Uid: (    0/    root)   Gid: (     ↩
    0/    root)
Access: 2007-10-16 20:02:57.000000000 +0200
Modify: 2007-10-16 20:02:57.000000000 +0200
Change: 2007-10-16 20:02:57.000000000 +0200
```

Pokud se jedná o textový soubor, budeme jej pravděpodobně chtít prohlédnout. Můžeme tak učinit příkazem `cat`, který slouží ke spojování obsahu více souborů, nebo lépe, a sice příkazem `more`, který zobrazí příslušný soubor tak, že jeho výpis neprobleskne obrazovkou, nebo ještě lépe, pomocí příkazu `less`, kde můžeme souborem pohodlně listovat či dokonce vyhledávat.

Less se ovládá následovně. Šipkami nebo pomocí Page Up a Page Down se pohybujeme v textu, výraz k vyhledání zadáváme pomocí lomítka, na další výskyt hledaného výrazu přejdeme pomocí klávesy `n` a prohlížení ukončíme klávesou `q`. Tento program bývá také využíván k prohlížení manuálových stránek prostřednictvím programu `man`.

7.5 Editory

Jelikož v GNU/Linuxu je nastavení uložené v konfiguračních souborech, což jsou obyčejné textové soubory, lze předpokládat, že se vám bude velice hodit znalost textových

editorů pro příkazovou řádku. Mezi editory patří `vi`, `emacs`, `nano`, `pico` či `mcedit`, který, jak název napovídá, je součástí dříve zmiňovaného *Midnight Commanderu*.

Editory `vi` a `emacs` bývají pro začátečníky obtížné, zejména `vi` je pověstný tím, že mnozí neznalí ani nenajdou způsob, jak jej ukončit. Ostatní výše zmíněné editory se začátečníkům ovládají snadněji, a proto mohu jen doporučit si některý z nich opatřit, pokud jej nainstalovaný nemáte.

Pro strýčka příhodu jsem samozřejmě připravil rychlokurz používání editoru `vi`, který je ve většině unixových systémů a linuxových distribucích.

7.5.1 Rychlokurz `vi`

Editor Vi má několik režimů práce. Po spuštění se ocitnete v tzv. normálním režimu. Do něj se kdykoliv dostanete stisknutím klávesy *Esc* dvakrát. V normálním režimu sice nemůžete editovat text, ale můžete se po textu pohybovat, buď pomocí šipek, nebo pomocí kláves `h` (vlevo), `j` (dolů), `k` (nahoru), `l` (vpravo).

Z normálního režimu se můžete dostat do příkazového režimu pomocí dvojtečky. Příkazový režim je důležitý, neboť pomocí něj můžete editor opustit či soubor uložit (či obojí). Vi se ukončuje pomocí příkazu `q`, soubor se ukládá pomocí příkazu `w`. Uložit soubor a ukončit editor můžete pomocí příkazu `wq`. Pokud jste provedli nějaké změny a chcete editor ukončit bez uložení změn, Vi vás nepustí. Musíte mu explicitně říci, že se má provést daný příkaz bez ohledu na cokoliv dalšího, což provedete přidáním vykřičníku. Příkaz pro ukončení editoru bez uložení změn je tedy `q!`.

Do editačního režimu se dostanete pomocí klávesy *Insert* nebo `i` z normálního režimu.

7.6 Zástupné znaky, speciální znaky

Ještě před tím, než budeme pokračovat, měli bychom si vysvětlit zástupné znaky, speciální znaky a expanzi. Pokud bychom měli soubor s názvem `nejaky text.txt` a chtěli ho zobrazit pomocí programu `less` a zapsali následující příkaz, došlo by k něčemu neočekávanému:

```
less nejaky text.txt
nejaky: není souborem ani adresářem
text.txt: není souborem ani adresářem
```

Problém je ve speciálním znaku, kterým je v tomto případě mezera. Slouží jako oddělovač parametrů, takže náš program `less` interpretoval příslušný soubor jako dva soubory, které samozřejmě neexistují, a `less` je tedy není schopen zobrazit.

Mezi *speciální znaky* patří kromě mezery křížek (označuje komentář), zástupné znaky (hvězdička, otazník, hranaté závorky, tilda), vykřičník (negátor výrazu), zpětné lomítko (neutralizuje jeden bezprostředně následující speciální znak), uvozovky (neutralizují speciální znaky v textu mezi uvozovkami), středník (odděluje více příkazů na jednom řádku), lomítko (oddělovač prvků cesty), špičaté (složené) závorky (oddělují blok příkazů a jsou prostředkem expanze), kulaté závorky (oddělují skupinu příkazů),

většítko a menšítko (přesměrování výstupu a vstupu), roura, paragraf (spustí příkaz na pozadí, znak dolaru (uvozuje proměnnou) a některé další.

Máme dvě možnosti. Buď neutralizujeme význam speciálního znaku (mezery) pomocí zpětného lomítka, nebo jméno souboru uzavřeme do uvozovek. Obojí bude fungovat:

```
less nejaky\ text.txt
```

```
less "nejaky text.txt"
```

Pokud bychom v názvu souboru měli samotné uvozovky, mohli bychom jejich vliv neutralizovat buď pomocí zpětného lomítka, nebo pomocí druhých uvozovek, takto:

```
less 'soubor s "textem".txt'
```

```
less soubor\ s\ \"textem\".txt
```

Pokud bychom měli soubor se zpětným lomítkem v názvu, zapsali bychom dvě zpětná lomítka za sebou, což by Bash interpretoval jako jedno normální zpětné lomítko bez speciálního významu.

7.6.1 Zástupné znaky

Pokud chceme nějakému příkazu předat více než jeden soubor, můžeme použít zástupné znaky, abychom je všechny nemuseli explicitně specifikovat. Řekněme, že chceme prolistovat všechny soubory začínající na `ukol` a končící na `txt`. V tom případě můžeme použít speciální znak v podobě hvězdičky, která zastupuje libovolný počet znaků. Výsledný příkaz tedy bude vypadat takto:

```
less ukol*txt
```

Dalším zástupným znakem je otazník, který zastupuje právě jeden libovolný znak. Pokud tedy chceme prolistovat všechny soubory, které začínají jedním libovolným znakem a pokračují `_ukol.txt`, provedeme následující příkaz:

```
less ?_ukol.txt
```

Pomocí hranatých závorek můžeme specifikovat, které znaky se smí na daném místě použít. Následující příkaz tedy zobrazí soubory `a_ukol.txt` a `b_ukol.txt`, pokud existují:

```
ls [ab]_ukol.txt
```

Můžeme samozřejmě specifikovat i rozsahy znaků a dokonce daný výraz negovat. Zkusíme tedy zobrazit všechny soubory začínající jedním velkým písmenem nebo jedním číslem:

```
ls [A-Z0-9]*
```

Nyní zobrazíme všechny soubory, které nezačínají číslem:

```
ls [!0-9]*
```

Měli bychom si ještě ozřejmit jednu důležitou věc. Pokud použijete jakýkoliv speciální znak jako parametr nějakého příkazu, Bash interpretuje jeho význam a příkazu předá výsledek. Ujasněme si to na příkladu. Mějme adresář s následujícími soubory:

```
abc_ukol.txt
a_ukol.txt
b_ukol.txt
c_ukol.txt
5_ukol.txt
```

Nyní chceme zobrazit obsah všech souborů začínající na `a`, takže použijeme příkaz:

```
ls a*
```

Bash nejprve vyřeší speciální znaky ve výrazu a programu `ls` předá výsledek. Program `ls` tedy obdrží seznam souborů v aktuálním adresáři, které začínají na `a`. Je to úplně stejné, jako kdybychom zapsali:

```
ls abc_ukol.txt a_ukol.txt
```

Speciální znaky a jejich význam si můžeme otestovat pomocí "ozvěny" v podobě programu `echo`. Ten vypíše na obrazovku všechno, co jsme mu předali jako parametry. Takže pokud bychom v tomto adresáři zapsali:

```
echo a*
```

Pak bychom obdrželi:

```
abc_ukol.txt a_ukol.txt
```

Program `echo` má ještě jeden zajímavý parametr, a tím je `-n`. Použití tohoto parametru způsobí, že `echo` neukončí výpis znakem konce řádku. To se nám bude hodit později.

Jiným zajímavým zástupným znakem je tilda (~), která supluje cestu k vašemu domovskému adresáři. Pokud se nacházíte někde úplně mimo, můžete se do svého domovského adresáře snadno navrátit příkazem:

```
cd ~
```

Bash si pamatuje, kde jste byli naposled a není problém se vrátit do předchozího adresáře:

```
cd -
```

Tím bychom mohli probírání zástupných znaků ukončit.

7.7 Práce se soubory

Opět připomínám, že mnoho operací se dá snado provést pomocí *Midnight Commanderu*. Nicméně někdy se určitě hodí vědět, jak běžné operace se soubory provádět "ručně". A pokud nahlédnete do manuálových stránek příslušných příkazů, možná objevíte i nové netušené možnosti, jak snadno provádět jisté náročnější operace.

Příkazy, které si představíme, jsou `cp` (kopírování), `mv` (přesun), `rm` (mazání souborů), `rmdir` (mazání adresářů), `touch` (vytvoření prázdného souboru), `mkdir` (vytvoření adresáře) a `ln` (vytvoření linku).

Zkusme si na jednoduchém příkladu ukázat základní práci se soubory:

```
[michal@griffon ~]$ ls
[michal@griffon ~]$ touch soubor
[michal@griffon ~]$ mkdir Adresar
[michal@griffon ~]$ ls
Adresar   soubor
[michal@griffon ~]$ cp soubor Adresar
[michal@griffon ~]$ ls
Adresar   soubor
[michal@griffon ~]$ ls Adresar
soubor
[michal@griffon ~]$ rm soubor
[michal@griffon ~]$ ls
Adresar
[michal@griffon ~]$ mv Adresar/soubor .
[michal@griffon ~]$ ls Adresar
[michal@griffon ~]$ ls
Adresar   soubor
[michal@griffon ~]$ rmdir Adresar
[michal@griffon ~]$ ls
soubor
```

Přibližme si teď jednotlivé příkazy podrobněji.

7.7.1 cp

Základní syntax příkazu `cp` jsme již probrali:

```
cp soubor adresář
```

Souborů může být samozřejmě více:

```
cp soubor1 soubor2 soubor3 adresář
```

Můžeme zkopírovat jeden soubor do druhého (tím přepíšeme jeho obsah):

```
cp soubor1 soubor2
```

Můžeme zkopírovat ale i celý adresář včetně všech podadresářů a souborů, které obsahuje. K tomu použijeme volbu `-R`:

```
cp -R adresář1 adresář2
```

O dalších možnostech programu `cp` se dozvíte pročtením jeho manuálové stránky.

Abychom si demonstrovali jeden zástupný znak, zkusíme zkopírovat nějaký soubor do aktuálního adresáře:

```
cp /var/tmp/soubor .
```

Tečka je odkaz na aktuální adresář.

7.7.2 mv

Program `mv` slouží k provádění přesunů nebo přejmenování. Má jasnou syntax:

```
mv zdroj1 zdroj2 zdroj3 cíl
```

Pokud specifikujete pouze jeden zdroj a zadaný cíl neexistuje, provede se přejmenování (platí samozřejmě pro soubory i adresáře):

```
mv staré_jméno nové_jméno
```

Pokud je cíl soubor, bude přepsán obsahem zdrojového souboru:

```
mv soubor1 soubor2
```

Je-li cílem adresář, všechny zdrojové soubory či adresáře se do něj přesunou:

```
mv soubor1 adresář1 cílový_adresář
```

7.7.3 rm

Program `rm` slouží k mazání. Bez dalších parametrů maže pouze soubory:

```
rm soubor1 soubor2
```

Pokud chceme vymazat adresář i se všemi jeho podadresáři, použijeme parametr `-r`:

```
rm -r adresář_k_vymazání
```

S programem `rm` a jeho volbou `-r` lze napáchat mnoho škody. Pokud chceme vymazat adresář `/tmp/práce`, ale překlepneme se třeba takto:

```
rm -r / tmp/práce
```

Začně program `rm` mazat celý adresářový strom, a až skončí, zkusí smazat `tmp/-práce`. Z tohoto důvodu si dávejte veliký pozor, budete-li mít oprávnění roota, protože tímto způsobem si spolehlivě zničíte celý systém včetně všech dat.

7.7.4 `mkdir`, `touch`, `ln`

Program `mkdir` umí vytvořit adresář:

```
mkdir nový_adresář
```

Program `touch` se "dotkne" specifikovaného souboru. Pokud tento soubor existuje, upraví se mu čas poslední změny a přístupu na aktuální čas. Pokud neexistuje, bude vytvořen s nulovým obsahem:

```
touch soubor
```

Program `ln` slouží k vytváření symbolických nebo pevných odkazů. Následující příkaz vytvoří symbolický odkaz s názvem `odkaz`, který bude ukazovat na soubor `data`:

```
ln -s data odkaz
```

Pokud bychom chtěli vytvořit pevný odkaz, vynecháme volbu `-s`:

```
ln data pevný_odkaz
```

Pevný odkaz je de facto další jméno pro daný soubor. Podmínkou jeho vytvoření je, že se zdrojový soubor i budoucí odkaz nachází na stejném souborovém systému. Po vytvoření pevného odkazu není mezi zdrojem a obrazem žádný rozdíl, obě jména ukazují na jeden a ten samý soubor. Vymazat tedy můžete kterýkoliv. Po vymazání posledního jména ukazujícího na daný soubor bude tento soubor vymazán. Pevných odkazů může být samozřejmě více.

Souborový systém realizuje tuto záležitost jednoduše. Po vytvoření pevného odkazu se zvýší číslo označující počet odkazů na daný soubor (po vytvoření souboru má příslušný soubor jeden odkaz). Ve chvíli, kdy toto číslo klesne na nulu, soubor se opravdu vymaže.

7.7.5 `find`, `locate`, `which`

Pokud potřebujete nějaký soubor najít, nepochybně se vám hodí znát příkaz `find` a `locate`. Příkaz `locate` prohledává databázi, která se aktualizuje zpravidla jednou denně. V této databázi jsou indexovány jednotlivé soubory. Výhodou tohoto postupu je, že vyhledávání je velice rychlé (prohledává se databáze a nikoliv vlastní adresářový strom). Nevýhodou je neaktuálnost. Použití programu je velice jednoduché:

```
locate vyraz
```

Můžete samozřejmě použít příslušné zástupné znaky. Pokud hledáte pouze umístění nějakého spustitelného programu, pomůže vám `which`:

```
which bash
```

Program `find` prohledává zadanou adresářovou strukturu přímo, bez pomoci nějaké databáze. Je to velmi komplexní program s řadou voleb. My si ukážeme opět jen to nejběžnější použití:

```
find /tmp -name "michal*.txt"
```

Takto bychom hledali soubor s názvem `michal[cokoliv].txt` v adresáři `/tmp`. Hledat můžeme ale i pomocí jiných kritérií:

```
find /tmp -type d
```

Tento příkaz pro změnu hledá soubory podle typu. Typ `d` označuje adresář. Příkaz tedy vypíše všechny adresáře v `/tmp`.

Pomocí `find` lze hledat soubory podle nejrůznějších kritérií včetně doby založení či poslední změny. Více informací viz manuálová stránka programu `find`:

```
man find
```

7.8 Přesměrování vstupu a výstupu, roury, filtry

Existují dva podstatné prostředky výstupu, `stdout` (standardní výstup) a `stderr` (chybový výstup). Smysl tohoto dělení je v tom, že lze kterýkoliv z nich (nebo oba) přesměrovat jinam. Pro vstup je určen `stdin` (standardní vstup). Vstup a výstup se běžně tiskne na obrazovku, ale lze jej přesměrovat jinam pomocí většítka a menšítka. Jak to všechno funguje v praxi si nyní ukážeme.

Řekněme, že bychom chtěli výpis obsahu adresáře uložit do souboru. V tom případě bychom rádi přesměrovali standardní výstup do souboru, což provedeme takto:

```
[michal@griffon ~]$ ls -l > vypis.txt
```

Pokud použijeme dvě většítka za sebou, soubor `vypis.txt` se nepřepíše, výpis se připojí na konec souboru:

```
[michal@griffon ~]$ ls -l >> vypis.txt
```

Stejně jako výstup lze přesměrovat i vstup, a příslušnému programu je tak možné poslat data ke zpracování z nějakého souboru (místo z klávesnice). Zkusme třeba tohle:

```
[michal@griffon ~]$ sort < vypis.txt
```

Program `sort` umí setřídit řádky podle určitých kritérií. Tento příkaz způsobil, že program `sort` zpracoval řádky v souboru `vypis.txt` a setřídil je výchozím způsobem.

Vstup i výstup lze přesměrovat najednou:

```
[michal@griffon ~]$ sort < vypis.txt > setrideno.txt
```

Tímto příkazem jsme setřídili řádky souboru `vypis.txt` a výsledek uložili do souboru `setrideno.txt`.

Dosud jsme pracovali pouze se standardním vstupem a výstupem, ale co chybový výstup (stderr)? Ten můžeme přesměrovat takto:

```
[michal@griffon ~]$ cp work/* /tmp 2&> chyby.txt
```

Tento příkaz zkopíruje obsah podadresáře `work` do adresáře `/tmp` a veškeré chybové hlášky uloží do souboru `chyby.txt`.

7.8.1 Roury

Nyní se dostáváme k tmelu, který umožňuje propojit jednotlivé unixové utility do složitějších funkčních celků. Roura je de facto pojítkem mezi dvěma programy v tom smyslu, že standardní výstup jednoho přesměruje na standardní vstup druhého. Nejtypičtější a nejjednodušší příklad by byl použití programu `less`, abychom si mohli prohlédnout delší výpis programu `ls`. To by se realizovalo takto:

```
[michal@griffon ~]$ ls -l | less
```

Známe-li z dřívějška program `sort`, můžeme jej napojit na výpis programu `du`, který vypisuje velikosti souborů (a adresářů). S vhodnými parametry získáme setříděný výpis (od největšího po nejmenší) velikosti domovských adresářů uživatelů:

```
du -s /home/* | sort -nr
```

Ještě jednou, co tedy roura způsobí? Způsobí to, že výpis programu `du` (tedy jeho standardní výstup) pošle (na standardní vstup) programu `sort`. Ten příslušný výpis zpracuje (setřídí) a pošle opět na standardní výstup, v tomto případě už na naši obrazovku.

Zkusme další příklad. Program `head` nebo `tail` nám poslouží, abychom z nějakého výpisu získali začátek nebo konec. V tomto případě bychom si mohli nechat vypsat pouze pět adresářů s největším obsahem:

```
du -s /home/* | sort -nr | head -n 5
```

Nyní zkusme nějaký komplexnější příkaz:

```
dd if=/dev/hda | gzip -c | ssh mike@base "dd of=~/dump"
```

Tenhle příkaz umožňuje poslat obsah blokového zařízení `hda` přes síť zabezpečeným (šifrovaným) kanálem a uložit jeho obsah do souboru `dump` v domovském adresáři uživatele `mike` na vzdáleném počítači `base`. Před tím, než se data pošlou přes síť se v tomto případě zkomprimují pomocí programu `gzip`.

Tímto příkladem se snažím ukázat, jak je v unixových systémech všechno pěkně propojeno. Jednotlivé unixové nástroje dělají jen jednu věc, ale dělají ji dobře. Tyto nástroje lze pak propojit prostřednictvím příkazové řádky a kombinovat do různých funkčních celků. V unixových systémech také platí, že všechno je soubor, tedy i pevný disk[2]. Unixové nástroje umí velmi dobře pracovat se soubory. Dvě vlastnosti unixových systémů, které se vhodně doplňují a umožňují "snadno" realizovat to, k čemu je v jiných operačních systémech potřeba speciální (a zpravidla komerční) software.

[2] Zařízení `hda` reprezentuje zařízení připojené jako master na prvním IDE řadiči. Podle toho, co je k tomuto řadiči připojeno se může jednat o pevný disk nebo optickou mechaniku.

7.8.2 Filtry

Filtry jsou jednoúčelové programy, které nějakým způsobem upravují data ze standardního vstupu a upravená data posílají na standardní výstup. Mohou je třídit, upravovat nebo propouštět jen určité řádky nebo vzory.

Filtr	Činnost
`sort`	setřídí řádky podle zadaných kritérií
`uniq`	ze setříděných řádek odstraní duplicity
`grep`	propouští pouze řádky obsahující určitý vzor či výraz
`head`	zachytí pouze několik prvních řádků výstupu
`tail`	zachytí pouze několik posledních řádků výstupu
`tr`	"překládá" znaky; může třeba provést konverzi velkých písmen na malá, apod.
`sed`	tzv. "stream editor", umožňuje různě upravovat jím procházející text
`awk`	programovací jazyk zaměřený na filtrování
`cut`	umožňuje "vyříznout" kusy řádek
`wc`	vrací počet slov (nebo počet řádek, použijeme-li parametr `-l`)

Tabulka 7.1: Nejtypičtější filtry

Pokud bychom měli na výstupu nějakého programu řadu duplicitních řádek, hodí se nám program `uniq`, který duplicity odstraní. Tomuto programu musíme ale nejprve pomoci a duplicitní řádky dát k sobě. K tomu použijeme nám známý program `sort`:

```
ps auh | cut -f 1 -d\  | sort | uniq
```

Tento příkaz vypíše všechny uživatele, pod jejichž právy běží některé procesy. Jako základ jsme použili program `ps` k vypsání všech procesů. Z jeho výstupu se nám hodí pouze první slovo na každé řádce. To představuje uživatele, pod jehož právy běží každý proces. Pomocí filtru `cut` jsme "vyřízli" pouze první pole, jako oddělovač jsme specifikovali mezeru. Všimněte si zpětného lomítka, který neutralizoval vliv mezery jako oddělovače parametrů. Následně jsme výpis setřídili programem `sort` a odstranili duplicity pomocí programu `uniq`.

Použití programu `cut` je vhodné použít tam, kde jsou jasně dané oddělovací znaky. Třeba v souboru `/etc/passwd` slouží jako oddělovač dvojtečka. Můžeme tedy snadno získat výchozí interpret příkazové řádky (shell) uživatele `michal`:

```
grep michal /etc/passwd | cut -f 7 -d:
```

Naopak tam, kde je výpis nějakým způsobem formátovaný a nelze se tedy spolehnout na oddělovací znaky, je vhodnější využít program `awk` následujícím způsobem:

```
ps aux | awk '{ print $2 }'
```

Tento příkaz vypíše všechny PID běžících procesů. Ty se nachází ve druhém sloupci výpisu programu `ps`, proto ta dvojka. Kdybychom chtěli jiný sloupec, nahradili bychom toto číslo adekvátním pořadovým číslem zvoleného sloupce.

Dlužno dodat, že program `ps` umožňuje velmi přesně specifikovat výstupní formát svého výpisu, a proto není tento trik vyloženě nutné znát. Na stranu druhou, je snadnější použít tento zápis, pokud si ho pamatujete, než pátrat v manuálové stránce programu `ps`, jak vhodně formátovat jeho výstup, aby se na něj dal použít program `cut`.

Program `sed` umožňuje provádět sofistikovanější úpravy filtrovaného textu. Asi nejznámější je konstrukce pro nahrazení nějakého řetězce jiným:

```
echo "Jdeme tam." | sed s/Jdeme/Nejdeme/
```

To je samozřejmě velmi primitivní příklad. Můžeme zkusit něco trošku složitějšího:

```
grep michal /etc/passwd | sed s/bash/zsh/
```

Tento příkaz upraví řádku se záznamem uživatele `michal` v `/etc/passwd` tak, že změní jeho výchozí shell z Bashe na zsh.

K programům `awk` a `sed` a jejich použití se dají najít i celé knihy, které do podrobna rozebírají jejich možnosti. Ze své praxe však nejčastěji používám výše uvedené konstrukce.

Program `tr` umožňuje nahradit nějakou skupinu znaků jinou skupinou. Může třeba převést všechna malá písmena v nějakém výpisu na velká:

```
[michal@griffon ~]$ echo "Ahoj" | tr [a-z] [A-Z]
AHOJ
```

Vpomeňme na zástupné znaky a význam hranatých závorek. Je možné provádět různé "překlady", ale to už nechám na vaší fantazii.

Program `wc` (zkrácenina od "word count") umí počítat slova nebo řádky v nějakém výpisu. Takže můžeme třeba velice snadno zjistit počet uživatelů v systému:

```
wc -l /etc/passwd
```

A ani jsme nemuseli použít žádnou rouru. Program `wc` ale samozřejmě umí pracovat i jako součást roury:

```
grep bash /etc/passwd | wc -l
```

Jak správně tušíte, příkaz vypíše počet uživatelů, kteří mají jako výchozí shell nastavený bash.

7.8.2.1 Příkaz místo parametru

Někdy se dostanete do situace, kdy budete chtít jako parametr zadat výstup nějakého příkazu. Kupříkladu, budete chtít zabít všechny procesy uživatele `brian`. Program `ps` vám poskytne potřebné informace o PID jednotlivých procesů, které pak budete chtít předat programu `kill`. Ten je však očekává jako parametr. Co s tím?

```
kill -9 `ps au | grep "brian" | awk '{print $1}'`
```

To, co uzavřete mezi znaky ` se provede nejdříve a jejich obsah se nahradí za výstup příslušného výrazu. V tomto případě se tedy nejprve provede příkaz `ps`, od něhož získáme příslušná PID, která se pak vloží na místo, kde je očekává program `kill`. Tento zápis je plně ekvivalentní s tímto:

```
kill -9 $(ps au | grep "brian" | awk '{print $1}')
```

7.9 Regulární výrazy

Jak už víme z dřívějška, `grep` umí různě filtrovat řádky podle vzorů. A tím vzorem může být kromě slova i regulární výraz. Pomocí něj můžeme velmi přesně specifikovat to, co se má filtrovat.

Předpokládám, že zástupné znaky již znáte, včetně možnosti na určitém místě specifikovat rozsah znaků, které se mají vybrat, pomocí hranatých závorek. A třeba právě to je také regulární výraz.

Něco takového už by nám mělo být jasné:

```
[Aa]hoj
```

Tomuto výrazu vyhoví jak `ahoj`, tak `Ahoj`. V hranatých závorkách jsme specifikovali, že na prvním místě může být velké nebo malé `A`. V regulárních výrazech máme ale k dispozici další zástupné a speciální znaky (viz tabulka). Cokoliv mimo těchto zástupných znaků bude interpretováno jako text, který se musí vyskytovat v příslušném řetězci, aby ten odpovídal regulárnímu výrazu.

Znak(y)	**Význam**
`.`	tečka zastupuje jeden libovolný znak
`[]`	hranaté závorky zastupují jeden znak, který musí odpovídat některému ze znaků uvnitř hranatých závorek
`^`	začátek řádku
`$`	konec řádku
`?`	předchozí znak je nepovinný a smí se nacházet v daném výrazu nejvýše jednou
`*`	libovolný počet opakování předchozího znaku (včetně nulového opakování, tj. situaci, kdy se znak na daném místě nenachází)
`+`	předchozí znak se musí vyskytovat alespoň jednou nebo vícekrát
`{n}`	předchozí znak se musí vyskytovat `n` krát
`{n,}`	předchozí znak se musí vyskytovat `n` nebo více krát
`{,m}`	předchozí znak se musí vyskytovat nejvýše `m` krát
`{n,m}`	předchozí znak se musí vyskytovat nejméně `n` krát a nejvíce `m` krát
`|`	roura je logickou spojkou nebo, tj. umožňuje specifikovat více regulárních výrazů, přičemž filtrovaný řetězec bude propuštěn, pokud vyhoví kterémukoliv z nich

Tabulka 7.2: Významy speciálních znaků v regulárních výrazech

Jakkoliv složitě to vypadá, je to vlastně docela jednoduché. Podívejme se na několik příkladů, na kterých si vše objasníme. Začneme tímto výrazem:

```
^[A-Z]hoj$
```

Znaky ^ a $ označují začátek a konec řádku. Tento regulární výraz tedy propustí pouze ty řádky, které začínají jedním velkým písmenem, pokračují textem hoj a tím končí.

```
^[0-9A-Za-z]{5}$
```

Tomuto regulárnímu výrazu odpoví jakákoliv řádka, na které je pouze pět znaků, a to z alfanumerické množiny (velká a malá písmena, čísla).

```
michal|jana
```

Tomuto regulárnímu výrazu odpoví jakýkoliv řetězec, ve kterém se nachází michal nebo jana. Znakem roury jsme spojili dva regulární výrazy do jednoho, kterému odpovídá jakýkoliv řetězec, který vyhovuje kterémukoliv z těchto dvou regulárních výrazů.

```
[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
```

Tento regulární výraz propustí jakýkoliv řetězec obsahující IP adresu. Přesněji jakýkoliv řetězec, který obsahuje jedno až tři čísla, tečku (zpětným lomítkem jsme neutralizovali její vliv jako zástupný znak pro jeden libovolný znak), jedno až tři čísla, další tečku, jedno až tři čísla, tečku a jedno až tři čísla.

Dlužno dodat, že tomuto výrazu bude odpovídat i nesmyslná IP adresa jako třeba tato:

```
945.847.654.425
```

Pokračujme dál:

```
ah.*oj
```

Tento regulární výraz bude odpovídat všem řetězcům, kde se nachází ah, libovolný počet libovolných znaků a oj.

7.9.1 Grep a regulární výrazy

Regulární výrazy v grepu se zadávají malinko jinak, než jsme si ukázali. Respektive, abychom mohli pracovat s regulárními výrazy tak, jak jsme na ně zvyklí, museli bychom použít takovýto příkaz:

```
echo -e "miii\nmooo" | grep -E "mi{3}|mo{3}"
```

Ten by pak vrátil následující:

```
miii
mooo
```

Tohle si opět podrobně vysvětlíme. Volbou `-e` programu `echo` jsme povolili interpretaci speciálních znaků uvozených zpětným lomítkem. Takovým znakem je třeba `\n`, který se pak interpretuje jako odentrování (tj. znak nového řádku). Tento zápis tedy vytvoří dva řádky, jeden s obsahem `miii` a druhý s obsahem `mooo`.

Regulární výrazy jsou spojeny rourou, takže vyhoví obě řádky. Jedna vyhoví prvnímu výrazu, druhá druhému. Grepu jsme museli parametrem `-E` říci, aby daný výraz interpretoval jako rozšířený regulární výraz. To je takový, na který jsme si dosud zvykli. Bez tohoto parametru interpretuje `grep` (a také třeba `sed` a další programy) pouze "základní" regulární výrazy. To jsou takové, kde místo speciálních znaků regulárních výrazů použijete jejich ekvivalenty se zpětným lomítkem, jako byste chtěli neutralizovat jejich význam.

Výše zmíněný výraz bychom pro standardní grep tedy museli přepsat do této podoby:

```
mi\{3\}\|mo\{3\}
```

Vypadá to sice hrozivě, ale pokud víte, že před každý ze speciálních znaků regulárních výrazů napíšete zpětné lomítko, není to až takový problém.

7.10 Správa úloh v shellu

Unixové systémy jsou víceúlohové, a jako takové nabízejí prostředky, jak realizovat více úloh současně. Kupříkladu, pokud budeme chtít pustit nějaký časově náročný program, ale budeme chtít dále pracovat s příkazovou řádkou, prostě ho pustíme na pozadí:

```
dd if=/dev/urandom of=whitenoise &
```

Ampersand na konci příkazu způsobí, že se daný příkaz začne provádět na pozadí. Pokud už nějaký program běží a vy jej chcete přesunout na pozadí, jde to také, i když je to o něco komplikovanější. Stiskem Ctrl-Z průběh daného příkazu (programu) zastavíte. Tak se dostanete k příkazové řádce, ale v tuto chvíli ještě daná úloha na pozadí neběží. Je jen pozastavená. Jednotlivé úlohy si můžete nechat vypsat pomocí příkazu `jobs`.

Číslo u každé úlohy je poměrně důležité, protože vám umožní danou úlohu převést do pozadí nebo opět do popředí. Pokud má naše pozastavená úloha číslo jedna, provedeme toto:

```
bg 1
```

Teoreticky jsme nemuseli číslo specifikovat, neboť bez parametru se převede do pozadí naposledy pozastavená úloha. Ale pokud budete mít takových úloh více, tomuto se nevyhnete.

Kteroukoliv úlohu můžete převést opět do popředí pomocí příkazu `fg`:

```
fg 1
```

V tomto případě bychom na pozadí běžící (nebo pozastavenou) úlohu č. 1 převedli na popředí.

Práci s více shelly na jednom terminálu umožňuje velmi snadno program `screen`. Jeho výhodou je to, že po jeho ukončení zůstanou všechny úlohy a shelly běžet, takže se k rozdělané práci můžete snadno vrátit na jiném terminálu.

7.11 Správa systému v shellu

Velkou část příkazů pro správu systému jsem již probral v jiných kapitolách. Máme tu nicméně pár restů. Nejprve si řekneme, jak získat oprávnění uživatele root nebo jiného uživatele. Abychom se "stali" jiným uživatelem, použijeme program `su` s parametrem v podobě uživatelského jména toho, jehož oprávnění chceme získat:

```
su michal
```

Budeme vyzváni k zadání hesla uživatele `michal` a pokud jej zapíšeme správně, získáme veškerá jeho oprávnění.

Pokud na konec příkazu umístíme pomlčku, získáme nejenom práva příslušného uživatele, ale i příslušná nastavení, která se hodí zejména v případě uživatele root. Ten má totiž v proměnné `PATH` uloženy adresáře `/sbin` a `/usr/sbin`. Pokud bychom pouze získali oprávnění uživatele root bez této úpravy, budeme muset upravit proměnnou `PATH` nebo specifikovat cestu ke všem programům v těchto adresářích, budeme-li je chtít použít.

```
su -
```

Pokud použijeme program `su` bez parametru v podobě uživatele, zvolí se výchozí uživatel, kterým je root.

Programem `su` můžeme spustit třeba pouze jediný příkaz:

```
su -c "ifconfig eth0 10.0.0.1 netmask 255.255.255.0"
```

Tento příkaz provede změnu nastavení síťové karty `eth0` na zmíněnou adresu a síťovou masku. K tomu je třeba oprávnění uživatele root, která získáme pomocí programu `su`.

Jiný mechanismus představuje program `sudo`, jehož nastavení je v `/etc/sudoers`. Tento program umožňuje získat příslušná oprávnění na jediný příkaz, avšak postačí k tomu naše heslo, tedy heslo uživatele. Sudo si navíc po zadání hesla pamatuje, že jste prověřená osoba a následné spuštění `sudo` už nebude vyžadovat žádné heslo. Pokud `sudo` nepoužijete během pěti minut, program opět zapomene, že jste prověřená osoba a o heslo požádá.

Příkladem použití může být třeba ekvivalent našeho příkazu výše:

```
sudo ifconfig eth0 10.0.0.1 netmask 255.255.255.0
```

V grafickém prostředí je vhodné pro spouštění "okenních" aplikací s jinými právy používat programy jako `kdesu`, `gksu` či `gksudo`.

7.11.1 Změny oprávnění

Ke změně oprávnění uživatelů slouží příkazy `chmod`, `chown` a `chgrp`. První zmíněný slouží ke změně práv, druhý ke změně vlastníka, třetí ke změně skupiny. Vše se samozřejmě týká souborů (resp. adresářů). Používají se poměrně snadno:

```
chown michal soubor.txt
```

Tento příkaz změnil vlastníka souboru `soubor.txt` na uživatele `michal`. Pokud bychom chtěli změnit skupinu příslušného souboru na `users`, provedeme:

```
chgrp users soubor.txt
```

Obojí můžeme provést i rychleji, jedním příkazem:

```
chown michal:users soubor.txt
```

Změny oprávnění pomocí programu `chmod` můžeme zadávat různě. Můžeme použít oktální číselnou reprezentaci nebo můžeme použít symbolický zápis.

Pokud si vzpomínáte na zápis oprávnění `rwxrwxrwx`, tak tam se první tři práva týkají vlastníka souboru, další člena skupiny a třetí ostatních uživatelů, přičemž `r` označuje právo čtení (v případě adresáře právo zobrazit jeho obsah), `w` označuje právo zápisu (v případě adresáře právo vytváření a mazání souborů) a `x` právo spustit soubor (v případě adresáře právo se do něj přepnout).

Oktální číselná reprezentace vychází z osmičkové soustavy a tvoří ji tři čísla, kde první reprezentuje práva vlastníka, druhé skupiny a třetí ostatních uživatelů. Jednotlivým oprávněním jsou přiřazeny hodnoty (r=4, w=2, x=1), které se sčítají. Tudíž, práva `rwxrw-r--` bude možné označit jako:

```
chmod 754 soubor.txt
```

Symbolický zápis pak rozeznává práva `r`, `w`, `x`, jak už je známe, a subjekty vlastník (`u`, user), skupina (`g`, group), ostatní (`o`, other) a všichni (`a`, all). Jednotlivá práva lze přidávat pomocí znaku `+`, odebírat pomocí znaku `-` a nastavovat pomocí znaku `=`. Následující příkaz tedy přiřadí všem subjektům možnost soubor číst:

```
chmod a+r soubor.txt
```

Následující příkaz přidá uživateli a skupině právo číst a zapisovat:

```
chmod ug+rw soubor.txt
```

Tento příkaz nastaví oprávnění příslušného souboru tak, že vlastníkovi nastaví právo číst a zapisovat, skupině pouze právo číst a ostatním uživatelům žádná práva nepřidělí:

```
chmod u=rw,g=r,o= soubor.txt
```

Jak vidíme, zápis pomocí čísel je poněkud rychlejší.

7.12 Vzdálený přístup k shellu

Jsou v zásadě dvě možnosti, jak se připojit k vzdálené příkazové řádce na jiném unixovém stroji. Tou první je hodně zastaralá a nebezpečná metoda - `telnet`. Telnet je nebezpečný v tom, že komunikace mezi počítači není šifrovaná, hesla putují v nezměněné podobě a kdokoliv, kdo by poslouchal na cestě, může velmi snadno k těmto informacím přijít. Naštěstí je tato metoda tak zastaralá, že se s ní velmi pravděpodobně nesetkáte.

Mnohem lepší metodou je použít `SSH` (Secured SHell). Na počítači, kam se chceme přihlásit, musí běžet SSH server (v prostředí GNU/Linuxu to bude velmi pravděpodobně OpenSSH) a firewall nesmí blokovat port `22`, na kterém SSH běží.

SSH klient bývá běžně přítomný snad v drtivé většině linuxových distribucí, ne-li skoro ve všech. Jsou k dispozici i klienti pro Windows[3]. SSH klient má velice jednoduchou syntax:

```
ssh uzivatel@pocitac
```

Je možné použít i trošku delší variantu:

```
ssh -l uzivatel pocitac
```

Při prvním spuštění se nás klient zeptá, jestli akceptujeme identitu serveru danou otiskem (fingerprint) jeho veřejného klíče. Nebývá na škodu si jej ověřit. Klient pak při každém dalším přihlášení kontroluje identitu serveru a odmítne se přihlásit, pokud otisk nesouhlasí. Zabrání tak tzv. man in the middle útoku[4].

Po zadání hesla se dostaneme k příkazové řádce vzdáleného počítače a s vhodně nakonfigurovaným serverem můžeme spouštět i grafické aplikace. Takto můžeme velmi snadno spravovat počítač na dálku.

Součástí balení SSH klienta je i program `scp`, který umožňuje mezi počítači kopírovat soubory (pomocí SSH):

```
scp soubor uzivatel@server:/cilovy/adresar
```

Zápis je možné samozřejmě obrátit:

```
scp uzivatel@server:/zdrojovy/soubor.txt .
```

Přes SSH lze kopírovat i v pohodlí grafického prostředí. Zkuste třeba v *Konqueroru* nebo *Krusaderu* zapsat adresu:

```
fish://uzivatel@pocitac
```

K dispozici jsou i scp klienti pro Windows[5].

[3] http://en.wikipedia.org/wiki/PuTTY
[4] http://en.wikipedia.org/wiki/Man_in_the_middle_attack
[5] http://en.wikipedia.org/wiki/Winscp

7.13 Zdroje a odkazy

- William Shotts, Jr., web http://linuxcommand.org/
- Čtenáři ABC Linuxu, Učebnice GNU/Linuxu: Příkazová řádka
- Johanka Spoustová, Pohádky z příkazové řádky
- ABC Linuxu, Bash (seriál)
- Machtelt Garrels, Bash guide for beginners
- Mendel Cooper, Advanced Bash-scripting guide

Kapitola 8

Psaní shellových skriptů

8.1 Hello world

V programátorských příručkách bývá zvykem začínat primitivním programem *hello world*. Psaní shellových skriptů je programování hodně podobné, a proto si zkusíme na primitivním programu ukázat náležitosti shellového skriptu:

```
#!/bin/bash

echo "Hello world."
```

Jednoduché, že? První řádka označuje, který interpret se má použít ke "spuštění" příslušného souboru. Na rozdíl od programovacích jazyků s kompilátory, které produkují spustitelný strojový kód, potřebují shellové skripty interpret, tedy program, který daný skript vykoná.

Takto opatřený soubor můžeme učinit spustitelným:

```
chmod a+x skript
```

Pak postačí jej spustit:

```
./skript
```

Všimněte si, že programy v aktuálním adresáři spouštíme tak, že se na aktuální adresář odkážeme v cestě. V unixových systémech totiž aktuální adresář není součástí cesty, která se prohledává při zadání příkazu. Je to vhodné bezpečnostní opatření pro případ, že někdo vytvoří škodlivý skript se stejným jménem jako některý z často používaných příkazů.

Výchozí cesta většinou obsahuje adresáře jako `/bin` či `/usr/bin`, kde se nachází spustitelné soubory všech nainstalovaných uživatelských programů. V GNU/Linuxu si tedy nemusíte dělat starosti s tím, kam se vám nainstaloval třeba Firefox, protože jeho spustitelný soubor se nachází v cestě, která se po zadání příkazu bude prohledávat. Není tedy problém spustit Firefox přímo z příkazového řádku:

```
[michal@griffon ~]$ firefox
```

Skripty lze samozřejmě spouštět "ručně", pomocí interpretu, takto:

```
bash skript
```

8.2 Druhy příkazů

Dosud jsme nerozlišovali typ použitých příkazů. Vlastní příkaz může být název spustitelného souboru, který tvoří nějaký program, může to být ale i vestavěná funkce shellu nebo třeba alias. Příkazy jako `ls`, `grep`, `cat` či `ssh` jsou názvy spustitelných souborů, tedy programy. Naopak třeba příkaz `cd` je vestavěnou funkcí shellu. Není to žádný konkrétní program. Podobně jako třeba `echo`. Této znalosti můžeme využít, pokud budeme v prostředí, kde máme sice shell, ale už ne jednotlivé programy jako `ls`. Program `ls` můžeme nahradit právě použitím vestavěné funkce shellu `echo`:

```
echo *
```

Poslední kategorií jsou aliasy. Ty můžete specifikovat vedle dalších nastavení ve svém `~/.bashrc`. Alias je jiné jméno pro nějaký příkaz. Můžete si tak namapovat nějaký komplikovanější příkaz na snadno zapamatovatelný alias:

```
alias la='ls -la'
```

Pokud by se vám stýskalo po logických označeních jednotek, můžete si pomoci třeba takto:

```
alias c:='cd /mnt/windows'
```

Pokud nevíte, co je daný příkaz zač, můžete zkusit vestavěnou funkci shellu `type`:

```
type prikaz
```

8.3 Proměnné

Proměnné slouží k uložení určitých hodnot či řetězců. Začneme proměnnými prostředí. O jedné jsme už několikrát mluvili, a sice o cestě, která se prohledává při zadání nějakého příkazu, který shell nerozpozná jako vestavěnou funkci. Proměnná se jmenuje `PATH` a její obsah můžeme zobrazit takto:

```
echo $PATH
```

Všechny proměnné prostředí můžeme vypsat pomocí příkazu `set`:

```
set
```

Další zajímavou proměnnou prostředí je `PS1`, která určuje tvar výzvy (prompt). Výzvu už jsme tu mnohokrát měli, vypadá třeba takto:

```
[michal@griffon ~]$
```

Mnoho informací o proměnných prostředí naleznete v manuálu bashe:

```
man bash
```

Proměnné nemusí ale být jenom proměnné prostředí. Můžete si je tvořit sami, což se zejména ve skriptech velice hodí. Proměnnou vytvoříte pomocí znaku =:

```
PROMENNA="Moje proměnná."
```

Vlevo je název proměnné, vpravo za = je její hodnota. Tento zápis netoleruje mezery. Pokud byste = oddělili mezerami, nebude to fungovat.

Hodnotu proměnné můžete dosadit kamkoliv, pokud před název proměnné zapíšete znak dolaru:

```
echo $PROMENNA
```

Ukažme si malý skriptík s proměnnou:

```
#!/bin/bash

LOG="~/log.txt"
cp soubor /tmp 2&>$LOG
```

Omlouvám se za takto primitivní příklad. Ano, jak tušíte, tento skript překopíruje soubor `soubor` do adresáře `/tmp` a veškeré případné chybové hlášky zapíše do souboru, který je specifikován v proměnné `LOG`. Pokud budete tvořit složitější skript, vyplácí se nějaké statické hodnoty (třeba adresáře, soubory, apod.) specifikovat v proměnných. Pak, když budete potřebovat hodnoty změnit, nemusíte procházet celý soubor, změníte pouze hodnotu v proměnné.

Do proměnných můžete přirozeně ukládat i výstup nějakého příkazu:

```
PROMENNA=$(ps au | grep "bash")
```

8.4 Matematika

Základní matematiku (celočíselné operace) provádí následující syntax:

```
$((vyraz))
```

Za `vyraz` stačí dosadit nějakou matematickou operaci, třeba:

```
$((4+5*(5-20/5)))
```

Operace vyžadující plovoucí desetinnou čárku můžete provádět pomocí programu `bc`. Jeho použití je ale trošku složitější:

```
echo "scale=2; 4*4-6*(1/25)" | bc
```

Proměnná `scale` nastavuje počet desetinných míst. Přirozeně, na místě pro jednotlivá čísla můžete uvádět proměnné a výsledek ukládat do jiných proměnných.

8.5 Funkce

Pokud často používáte nějaký sled příkazů, hodí se jej umístit do funkce, kterou lze pak spouštět jediným příkazem. Třeba takto:

```
#!/bin/bash

function nase_funkce {
   echo "Kopiruji soubor ..."
   cp soubor.txt /tmp
   du -h /tmp/soubor
}

nase_funkce
```

Tohle je opět značně primitivní příklad, ale svůj účel plní. Vidíme, jak se funkce vytváří i jak se používá. Funkcím lze předávat parametry:

```
#!/bin/bash

function funkce {
   echo "Parametr funkce c. 1: $1"
   echo "Parametr funkce c. 2: $2"
}

funkce parametr1 parametr2
```

O parametrech funkcí i skriptů se ale dozvíme více později.

8.6 Podmínky

Průběh skriptu může někdy narazit na situaci, kde bude třeba o něčem rozhodnout. Za tímto účelem máme k dispozici podmínku:

```
if [ podminka ]; then
  prikaz
fi
```

To je nejjednodušší možná konstrukce podmínky. Pokud je podmínka `podminka` pravdivá, provede se `prikaz`. Je však možné konstruovat i složitější podmínky, třeba jako je tato:

```
if [ vyraz1 ]; then
  prikaz1
elif [ vyraz2 ]; then
  prikaz2
else
  prikaz3
fi
```

V tomto případě se provede `prikaz1`, jestliže je podmínka `vyraz1` pravdivá. Pokud není, ale je pravdivá podmínka `vyraz2`, provede se `prikaz2`. Pokud ani tato podmínka pravdivá, provede se `prikaz3`. Ještě než zkusíme nějaký reálný příklad, povíme si více o podmínkách.

8.6.1 Výrazy (podmínky)

Podmínka je nějaký výraz, který nabývá určitých logických hodnot (pravda či nepravda). V zásadě, všechny podmínky interpretuje program `test`, takže když se podíváte do jeho manuálové stránky, objevíte přehledný výpis všeho, co lze testovat. Já se pokusím o stručný vytah.

Výraz	**Význam**
`(výraz)`	podmínka je pravdivá, je-li výraz pravdivý
`! výraz`	podmínka je pravdivá, pokud výraz není pravdivý (negace)
`výraz1 -a výraz2`	podmínka je pravdivá, pokud jsou oba výrazy pravdivé (logická spojka AND)
`výraz1 -o výraz2`	podmínka je pravdivá, pokud je alespoň jeden z výrazů pravdivý (logická spojka OR)

Tabulka 8.1: Podmínky obecně

Výraz	**Význam**
`-n řetězec`	řetězec je neprázdný
`-z řetězec`	řetězec je prázdný
`řetězec1 = řetězec2`	řetězce jsou shodné
`řetězec1 != řetězec2`	řetězce jsou různé

Tabulka 8.2: Podmínky vztažené k řetězcům

Výraz	**Význam**
`číslo1 -eq číslo2`	číslo1 = číslo2 (equals)
`číslo1 -ge číslo2`	číslo1 $\geq$ číslo2 (greater or equal)
`číslo1 -gt číslo2`	číslo1 > číslo2 (greater than)
`číslo1 -le číslo2`	číslo1 $\leq$ číslo2 (lower or equal)
`číslo1 -lt číslo2`	číslo1 < číslo2 (lower than)
`číslo1 -ne číslo2`	číslo1 $\neq$ číslo2 (not equal)

Tabulka 8.3: Podmínky vztažené k (celým) číslům

Seznam podmínek není úplný (v případě souborů), ale jak už jsem podotýkal, `man test` je vhodné si projít. Příklady různých výrazů si ukážeme v následující sekci.

Výraz	Význam
`-e soubor`	soubor existuje
`-f soubor`	soubor existuje a je to soubor
`-d soubor`	soubor existuje a je to adresář
`-h soubor`	soubor existuje a je to symbolický link
`-b soubor`	soubor existuje a je to blokové zařízení
`-c soubor`	soubor existuje a je to znakové zařízení
`-p soubor`	soubor existuje a je to pojmenovaná roura
`-s soubor`	soubor existuje a má nenulovou velikost
`-S soubor`	soubor existuje a je to socket
`soubor1 -nt soubor2`	soubor1 je novější (newer than) soubor2
`soubor1 -ot soubor2`	soubor1 je starší (older than) soubor2

Tabulka 8.4: Některé podmínky vztažené k souborům

8.6.2 Příklady různých podmínek

Pro začátek zkusme něco jednoduchého:

```
if [ `grep -c ^michal /etc/passwd` -eq 0 ]; then
  echo "Uživatel 'michal' neexistuje."
else
  echo "Uživatel 'michal' existuje."
fi
```

Příkaz `grep -c ^michal /etc/passwd` vypíše počet řádků souboru `/etc/passwd`, které začínají na `michal`. Pokud je toto číslo nula, je jasné, že žádný uživatel `michal` v systému není. Jiným příkladem může být:

```
if [ -f /var/run/mysqld.pid ]; then
  echo "MySQL server pravděpodobně běží."
else
  echo "MySQL server neběží."
fi
```

Podmínka testuje existenci souboru, který by měl být vytvořen po startu MySQL serveru (obsahuje PID běžící instance `mysqld`) a vymazán při jeho ukončení. Jistější by bylo použití programu `ps`, ale to by byl jednak příklad podobný předchozímu, a pak bych nemohl poukázat na to, že v GNU/Linuxu a shellu lze jednotlivé úlohy řešit různým způsobem, přičemž každá z variant může vést k cíli.

```
if [ -z "$PROMENNA" ]; then
  echo "Proměnná je prázdná."
else
  echo "Proměnná obsahuje '$PROMENNA'"
fi
```

Tato podmínka testuje, je-li proměnná `PROMENNA` prázdná, přičemž pokud není, vypíše, co obsahuje.

8.6.3 Jak řešit chyby

Řekněme, že máme následující skript:

```
#!/bin/bash

PROMENNA="dve slova"

if [ -z $PROMENNA ]; then
  echo "Promenna je prazdna."
else
  echo "Promenna obsahuje $PROMENNA"
  fi
```

Pokud ho spustíme, dostaneme následující hlášku:

```
skript: line 5: [: dve: binary operator expected
```

Problém nastal na řádce 5 (tj. řádce z podmínkou). Jelikož proměnná, kterou testujeme, obsahuje dvě slova oddělená mezerou a po dosazení proměnné do podmínky při běhu skriptu se tato mezera interpretuje jako oddělovač, otestuje se pouze řetězec "`dve`" a řetězec "`slova`" se Bash pokusí mylně interpretovat jako další část podmínky:

```
if [ -z dve slova ]; then
```

Tento problém lze odstranit uzavřením proměnné do uvozovek:

```
if [ -z "$PROMENNA" ]; then
```

Pak, po dosazení proměnné se Bash dopracuje k výrazu, kde "uvidí" pouze jediný řetězec:

```
if [ -z "dve slova" ]; then
```

Uzavření proměnné do uvozovek pomůže třeba i v situaci, kdy je příslušná proměnná prázdná. Další problém může nastat velmi snadno v případě, kdy je proměnná nesprávného typu. I když jsou v Bashi všechny proměnné považovány za řetězce, v některých případech může Bash očekávat pouze určité znaky. Třeba v situaci, kdy testujeme nějakou číselnou hodnotu, očekává Bash řetězec složený pouze z čísel:

```
#!/bin/bash

CISLO="ahoj"

if [ "$CISLO" -eq 5 ]; then
   echo "Cislo je rovne 5."
else
   echo "Cislo neni rovne 5."
fi
```

V tomto případě dostaneme hlášku:

```
skript: line 5: [: ahoj: integer expression expected
```

Tou se vám Bash snaží naznačit, že tam, kde očekával číslo, číslo není. V tomto případě můžeme před touto podmínkou testovat příslušnou proměnnou pomocí regulárních výrazů, zda-li obsahuje pouze čísla. Většinou ale postačí, pokud si tuto záležitost pouze sami ohlídáte.

Další velmi typickou chybou je zapomenutí nějaké párové značky, třeba uvozovek:

```
#!/bin/bash
PROMENNA="ahoj

if [ -z "$PROMENNA" ]; then
   echo "Promenna je prazdna."
else
   echo "$PROMENNA"
fi
```

Po spuštění tohoto skriptu nám Bash zahlásí:

```
./skript: line 7: unexpected EOF while looking for matching  ↩
    `"'
./skript: line 9: syntax error: unexpected end of file
```

V tomto případě nezbyde než problém najít a odstranit. V delších skriptech vám pomohou editory s barevným zvýrazněním syntaxe.

Ne vždycky ale bude chyba zcela zřejmá. Může se stát, že nevíme přesně, kde chyba nastala. Pak můžeme použít parametr `-x`, buď takto:

```
bash -x skript
```

Nebo umístíme do záhlaví skriptu lehce pozměněnou úvodní řádku:

```
#!/bin/bash -x
```

Tak nám Bash bude vypisovat průběh skriptu. Můžeme takto označit třeba jen některou část skriptu:

```
#!/bin/bash

CISLO="ahoj"

set -x

if [ "$CISLO" -eq 5 ]; then
   echo "Cislo je rovne 5."
else
   echo "Cislo neni rovne 5."
fi

set +x

[...]
```

Jiným trikem je třeba zakomentování nějakého podezřelého úseku:

```
#!/bin/bash

CISLO="ahoj"

if [ "$CISLO" -eq 5 ]; then
   echo "Cislo je rovne 5."
#else
#   echo "Cislo neni rovne 5."
fi
```

Pokud pracujeme s nějakým nebezpečným programem, můžeme při ladění skriptu nechat použití inkriminovaného programu nechat vypisovat:

```
#!/bin/bash

cp /var/log/messages /tmp
echo rm /tmp/messages
```

Což se bude hodit ještě více, pokud danému nebezpečnému programu budete předhazovat nějakou proměnnou, kde s jistotou nevíte, co se v ní bude v danou chvíli nacházet.

8.7 Větvení a cykly

Pokud se rozhodujeme mezi dvěma možnostmi, je konstrukce `if` optimální. Pokud se naopak potřebujeme rozhodovat mezi více možnostmi podle obsahu proměnné, můžeme použít konstrukci `case`:

```
case $promenna in
   1 ) echo "Promenna je rovna 1" ;;
   2 ) echo "Promenna je rovna 2" ;;
   3 ) echo "Promenna je rovna 3" ;;
   * ) echo "Promenna neni rovna 1, 2 ani 3"
esac
```

Můžeme použít i složitější výrazy se zástupnými znaky:

```
case $promenna in
   [a-z] | [A-Z] ) echo "Jedno male nebo velke pismeno" ;;
   [0-9] ) echo "Jednociferne cislo" ;;
   * ) echo "Neco jineho"
esac
```

Někdy potřebujeme určitou činnost opakovat. A to je úloha pro cykly. Ty můžeme rozdělit do dvou kategorií. Cykly provádějící určitý počet opakování (cyklus `for`) a cykly založené na podmínce (cykly `while` a `until`).

8.7.1 Cyklus `for`

Syntaxe cyklu `for` vypadá takto:

```
for promenna in 1 2 3 4 5; do
   echo "$promenna"
done
```

Cyklus pracuje tak, že každý z prvků za `in` vždy postupně dosadí do proměnné `promenna`, provede kód uvnitř svého těla a tento postup opakuje do té doby, dokud nevyčerpá všechny prvky. V tomto případě jsme za prvky dosadili čísla od 1 do 5. Výstup tohoto cyklu by vypadal takto:

```
1
2
3
4
5
```

Pokud chceme použít určitou sekvenci čísel, nemusíme se namáhat s jejich vyplňováním. Postačí použít příkaz `seq`:

```
for i in `seq 1 50`; do
   echo $i
done
```

Tento úsek kódu vypíše čísla od 1 do 50. Cyklus `for` však umožňuje, aby cyklem procházely i zcela jiné hodnoty. Kupříkladu, následující skript vypíše obsah aktuálního adresáře:

```
for i in *; do
   echo $i
done
```

8.7.2 Cykly `while` a `until`

Cykly `while` a `until` pracují tak, že opakují určitý postup, dokud je splněna určitá podmínka (cyklus `while`) nebo dokud určitá podmínka splněna není (cyklus `until`. Syntaxe je prakticky shodná:

```
cislo=4
until [ $cislo -eq 5 ]; do
  echo "$cislo"
  cislo=$((cislo+1))
done

while [ $cislo -gt 0 ]; do
  echo "$cislo"
  cislo=$((cislo-1))
done
```

Tento program vypíše:

```
4
5
4
3
2
1
```

První cyklus se opakuje tak dlouho, dokud nenastane jeho podmínka, tj. dokud proměnná `cislo` nabyde hodnoty 5. Druhý cyklus se opakuje tak dlouho, dokud je jeho podmínka splněná, tj dokud je proměnná `cislo` větší než 0.

Pomocí cyklu `while` můžeme třeba zpracovávat po řádcích nějaký soubor:

```
cat soubor | while read radka; do
  echo $radka
done
```

8.7.3 `break` a `continue`

Někdy můžeme mít zájem běh nějakého cyklu ukončit úplně nebo nedokončit iteraci a provést další opakování. K tomu slouží příkazy `break` a `continue`:

```
#!/bin/bash

for i in 1 2 3 4 5 6; do
   if [ $i -eq 3 ]; then
      continue
   elif [ $i -eq 5 ]; then
      break
   fi
   echo $i
done
```

Výstupem tohoto skriptu bude:

```
1
2
4
```

Proč? Když se do proměnné `i` dostala trojka, použili jsme příkaz `continue`, který přeskočil zbytek těla konstrukce `for` a přistoupil k další hodnotě, tedy ke čtyřce. V momentě, kdy se do proměnné `i` dostala pětka, provedl se příkaz `break`, který ukončil provádění celého cyklu.

8.8 Parametry a uživatelský vstup

Skriptům nebo funkcím můžeme předávat parametry. Ty jsou pak k dispozici buď jako pole všech parametrů reprezentované proměnnou `$@` nebo jako jednotlivé parametry v proměnných `$1`, `$2`, atd. Proměnná `$0` obsahuje jméno souboru s vlastním skriptem.

Vše si ozřejmíme na následujícím příkladě. Mějme skript:

```
#!/bin/bash

echo "Soubor se skriptem: $0"
echo "Prvni parametr: $1"
echo "Druhy parametr: $2"
echo "Treti parametr: $3"
echo "Pole vsech parametru: $@"
```

Tento skript spustíme následujícím způsobem:

```
./skript jedna dve tri
```

Následně obdržíme tento výpis:

```
Soubor se skriptem: ./skript
Prvni parametr: jedna
Druhy parametr: dve
Treti parametr: tri
Pole vsech parametru: jedna dve tri
```

Někdy můžeme dát přednost interaktivnímu programu, který požádá uživatele o zadání nějaké hodnoty. V takové situaci můžeme použít příkaz `read`, který uloží uživatelský vstup do proměnné. Ukažme si to na příkladu:

```
#!/bin/bash

while true; do
  echo -n "Zadejte polomer (Ctrl-C ukonci skript): "
  read POLOMER
  echo -n "Obsah kruhu s polomerem $POLOMER je"
  echo "scale=3; 3.14*$POLOMER^2" | bc
done
```

Tento program využívá mnohé z toho, co jsme se naučili už dříve. Umožňuje spočítat obsah kruhu. Cyklus `while` má v tomto případě místo podmínky dosazený program, který navrací pravdivostní hodnotu 1 nebo `true` (pravda). To znamená, že cyklus se bude opakovat, dokud uživatel násilně neukončí skript pomocí Ctrl-C.

Dosavadních znalostí můžeme využít i pro vytvoření lepšího řešení:

```
#!/bin/bash

function obsah {
  echo -n "Obsah kruhu s polomerem $1 je "
  echo "scale=3; 3.14*$1^2" | bc
}

if [ -z "$1" ]; then
  while true; do
    echo -n "Zadejte polomer (Ctrl-C ukonci skript): "
    read POLOMER
```

```
    obsah $POLOMER
  done
else
  obsah "$1"
fi
```

Tento skript využívá funkci, která provádí vlastní výpočet, dále podmínku, která hlídá, zda-li uživatel zadal skriptu při spuštění parametr nebo ne. Pokud ano, provede výpočet na hodnotě získané z parametru, pokud uživatel parametr nezadal, nabídne mu "interaktivní" režim, kdy může hodnotu zadat sám. Interaktivní režim pak pracuje stejně jako příklad výše.

8.9 Ošetření chybových stavů a reakce na signály

Pokud budeme pracovat s běžnými příkazy, může se stát, že některý z nich nebude schopen příslušnou operaci provést a selže:

```
#!/bin/bash

cp soubor adresar
rm soubor
```

Tento skript sice nic smysluplného nedělá, ale můžeme si na něm ukázat, o co jde. Pokud totiž první příkaz (`cp`) selže, kopie původního souboru se nevytvoří, a pokud bude skript pokračovat dál (což bude), smaže zdrojový soubor. V tomto případě by se tedy hodilo zjistit, jestli první příkaz proběhl úspěšně, a pokud ne, zastavit průběh skriptu.

Každý správně napsaný unixový program po sobě zanechá tzv. konečný stav (exit status). To je číslo, které nám pomůže zjistit, jestli program úspěšně skončil nebo jestli skončil s chybou. Nula představuje bezchybný konec, nenulové číslo poukazuje na chybu při práci programu. Hodnota konečného stavu naposledy spuštěného programu se uchovává v proměnné `$?`. Toho můžeme využít a skript adekvátně upravit:

```
#!/bin/bash

cp soubor adresar
if [ $? -gt 0 ]; then
  echo "Soubor se nepodarilo prekopirovat."
  exit 1
fi
rm soubor
```

Na tomto příkladě jsme si ukázali dvě věci. Jednak zpracování chybového stavu a jednak příkaz `exit`, který okamžitě ukončí běh skriptu. Jako parametr lze tomuto příkazu předat hodnotu, kterou má vrátit jako konečný stav. V tomto případě jsme zvolili nenulovou hodnotu, protože náš program v této situaci skončil s chybou.

Chybové stavy můžeme ošetřovat v rámci jednoho příkazu takto:

```
cp soubor adresar && echo "Soubor se zkopiroval."
```

Dva ampersandy za sebou fungují jako speciální oddělovač příkazů. Příkaz za nimi se provede pouze v případě, že příkaz před nimi bude korektně ukončen. Opačnou možnost nabízí dvě roury:

```
cp soubor adresar || echo "Nastala chyba."
```

V tomto případě se příkaz po dvou rourách provede pouze v situaci, kdy příkaz před nimi vrátí nenulový konečný stav.

8.9.1 Reakce na signály

Skripty mohou reagovat určitým způsobem na signály, které jim pošleme třeba příkazem `kill`. Tato znalost se nám hodí třeba proto, abychom mohli po sobě uklidit, když někdo požádá o násilné ukončení našeho skriptu.

K tomu, abychom mohli reagovat na určité signály, slouží příkaz `trap`. Syntaxe je následující:

```
trap "příkaz" signály
```

Příkladem tedy může být výmaz dočasného souboru při jakémkoliv násilném ukončení skriptu:

```
trap "rm $TEMP; exit" SIGHUP SIGINT SIGTERM
```

Pokud náš skript opatříme touto řádkou, v případě, že dojde k násilnému ukončení běhu skriptu, nezůstane po něm žádné "smetí".

8.9.2 nohup

Pokud provádímě nějaký příkaz na pozadí a odhlásíme se, pošle se všem programům či skriptům, které jsme nechali pracovat na pozadí, signál `SIGHUP`. Ten zpravidla programy zpracují tak, že se ukončí. Pokud tedy chceme spustit nějaký příkaz (program) na pozadí tak, aby se při našem odhlášení neukončil, použijeme k tomu program `nohup`:

```
nohup prikaz &
```

8.10 Zdroje a odkazy

- William Shotts, Jr., web http://linuxcommand.org/
- Čtenáři ABC Linuxu, Učebnice GNU/Linuxu: Příkazová řádka
- Johanka Spoustová, Pohádky z příkazové řádky
- ABC Linuxu, Bash (seriál)
- Machtelt Garrels, Bash guide for beginners
- Mendel Cooper, Advanced Bash-scripting guide

Kapitola 9

Jak a kde hledat pomoc

Každý se někdy dostane do problémů. Otevřenost GNU/Linuxu je při řešení problémů velikou výhodou, i když předpokladem pro její užití je jistá úroveň znalostí. Čím více svůj systém znáte, tím rychleji jste schopni problém vyřešit. Čím více problémů vyřešíte, tím více poznáte svůj systém.

9.1 Řešení problémů vlastními silami

Pokud nemáte po ruce nějakého známého, který ochotně řeší vaše problémy s GNU/Linuxem, nemáte předplacenou placenou podporu, očekávejte, že každý problém budete řešit převážně sami. Chtějte jej řešit sami. Pokud vám někdo předloží sadu pokynů, jak problém vyřešit, a vy je bezmyšlenkovitě vykonáte, nic si z toho neodnesete. Využijte raději těchto příležitostí, abyste svůj systém poznali a získali tak zkušenosti, které vám později nepochybně pomohou.

9.1.1 Hrubý postup řešení

Každý problém bychom se měli pokusit nejprve lokalizovat, tedy zjistit, kde přesně k problému dochází. GNU/Linux je modulární systém sestávající se z mnoha komponent, tudíž by bylo vhodné izolovat tu komponentu, kde k problému dochází. Je problém s linuxovým jádrem, X-Serverem, nějakým programem, hardwarem či něčím jiným?

Jedná-li se o problém se stabilitou, je prvním podezřelým hardware. Zkontrolujte zejména operační paměť, jestli není některý modul vadný (program memtest, nechte jej běžet dostatečně dlouho, tj. minimálně několik hodin). Zkontrolujte i teplotu základní desky, procesoru, pevných disků, apod. Pokud jste přetaktovali procesor, zkuste navrátit tovární nastavení procesoru. Zkuste si pohrát s nastavením jádra a vypnout některé vlastnosti, které na špatném hardwaru mohou způsobit problémy pomocí následujících parametrů jádra:

- `vga=normal`
- `noapic`

- `nolapic`
- `noacpi`
- `noapm`
- `nodma`
- `noscsi`
- `nousb`
- `nopcmcia`
- `nofirewire`

Máte-li SATA nebo SCSI disky, pak vynechte parametr `noscsi`, jinak vám systém nejspíše nenabootuje.

Druhým podezřelým je pak samotné linuxové jádro. Zkontrolujte, jestli nebyla vydána aktualizovaná verze. Je-li důvodné podezření, že se o problém s jádrem jedná, můžete zkusit i zkompilovat nové vanilla jádro.

Třetím podezřelým je nějaký neoficiální jaderný modul. Já osobně jsem měl problémy s jednou verzí ovladače Nvidia (v roce 2003, od té doby jsem problémy nezaznamenal). V tomto případě doporučuji zvolit jinou verzi ovladače, novější, je-li k dispozici, popřípadě starší.

Při eventuelních potížích je třeba rozlišit problém s grafickým rozhraním a s jádrem operačního systému. Z jiných operačních systémů můžete být zvyklí předpokládat, že v případě zamrznutí grafického rozrhraní zamrzl systém. Tady tomu tak být nemusí (jádro a grafické prostředí jsou odděleny), a je potřeba tyto případy odlišit. I když, při některém opravdu ošklivém pádu grafického prostředí (konkrétně X-Serveru) může dojít k zablokování klávesnice a myši (i když zbytek systému funguje). Pokud máte jiný počítač a přístup přes SSH, můžete se zkusit přihlásit z něj a situaci zachránit.

9.1.2 Systémové informace

Jak získat nejrůznější informace o systému jsem už probíral. Získáním informací o systému, popřípadě přímo o vašem problému, vám v jeho řešení nepochybně pomůže.

9.1.3 Dokumentace

Vždy upřednostňujte informační zdroje vstažené k vaší distribuci. Její příručku a webové stránky se vždy vyplatí projít, zejména pak poznámky k verzi vaší distribuce, kde naleznete informace o známých problémech a jejich řešení. Nezapomínejte ani na manuálové stránky a další off-line informační zdroje.

9.1.4 Vyhledávače

V drtivé většině případů lze nalézt řešení problému přímo pomocí vyhledávačů. Máte-li k dispozici chybovou hlášku, zadejte ji do Googlu nebo jiného vyhledávače. Pokud jste problém lokalizovali, zadejte jeho klíčová slova do vyhledávače. S jistou pravděpodobností se dostanete přímo k řešení daného problému. Vyhledávací služby lze použít i k vyhledání dokumentace k dané komponentě nebo k samotné distribuci.

9.1.5 Prostředky k řešení problému

Pokud se vám podařilo problém úspěšně lokalizovat, zbývá ho vyřešit. Řešení samozřejmě silně závisí na povaze problému. Někdy bude stačit nějaká operace balíčkovacího systému (instalace nového balíčku, reinstalace či odstranění existujícího, upgrade), jindy nezbyde než se ponořit do konfiguračních souborů v adresáři `/etc`.

Ty mají naštěstí podobu obyčejných textových souborů a dají se editovat jakýmkoliv textovým editorem, klidně v příkazové řádce. O změnách těchto souborů si raději veďte nějakou evidenci, využívejte komentáře (zpravidla uvozené znakem #) k uložení podstatných informací (že jste provedli změnu, kdy a kde jste ji provedli a co tam bylo původně).

9.2 Pokládání dotazů

Položit dotaz do diskusního fóra, e-mailové konference nebo zaslat žádost o placenou podporu se zdá být jako velmi jednoduchá záležitost, ale v praxi se ukazuje být mnohem komplikovanější. Následující průvodce vám pomůže položit správný dotaz správným způsobem, a maximalizovat tak pravděpodobnost, že vám diskutující pomohou problém vyřešit.

Předně je třeba uvážit, kam dotaz pokládáte. Pokud se jedná o placenou podporu, na odpověď máte nárok. Jedná-li se však o komunitní fórum, kde pomáhají dobrovolníci ve svém volném čase, není dobré uplatňovat stejně rázný a náročný přístup jako v případě placené podpory. Pokud se budete chovat arogantně, nepřátelsky či pohrdavě, spolehlivě odradíte diskutující od snahy vám pomoci. Ale to je jasné.

Prvním krokem při řešení problému je pokus jej vyřešit vlastními silami. I když ho sami třeba nevyřešíte, alespoň se zorientujete a získáte informace, které značně usnadní práci těm, kteří vám budou pomáhat.

Druhým krokem je prohledání všech relevantních informačních zdrojů, zejména pak využití služeb vyhledávačů. Vyhledávat doporučuji i v příslušných diskusních fórech. Váš problém velmi pravděpodobně již řešil někdo jiný a vyhledávače jsou nejrychlejší cestou, jak se dostat k jeho řešení.

9.2.1 Kam položit dotaz

Musíte zvolit správné fórum či konferenci, tedy takové, kde je největší pravděpodobnost, že na váš dotaz někdo (správně) odpoví. Distribučně specifické problémy je vhodné řešit na fóru dané distribuce (což je také hlavní kandidát na místo, kam položit dotaz). Problém s konkrétním programem lze řešit přímo s jeho vývojáři (resp. v

oficiálním fóru). Obecné dotazy lze řešit v obecných fórech na linuxových portálech[1], kde bývá největší množství dobrovolníků.

Před položením dotazu nezapomeňte fórum (či archív e-mailové konference) prohledat, jestli se podobný problém již neřešil. Pokud položíte dotaz, který byl mnohokrát (a úspěšně) řešen, vystavíte se jisté vlně nevole.

9.2.2 Formulace dotazu

Dotaz by měl řešit *jeden* problém, ne více. Pokud uvedete více problémů, riskujete buď uzamčení a výmaz takového vlákna administrátory, nebo situaci, kdy vám diskutující pomohou vyřešit pouze jeden problém a těch dalších si nevšimnou. Pokud máte více problémů, rozepište je do více dotazů.

Vhodnou či nevhodnou formulací dotazu zvýšíte nebo snížíte šanci na odpověď. Správná formulace je polovina úspěchu. Jak na to?

Předmět dotazu by měl být stručný, jasný a popisný. Měli byste jej formulovat tak, aby ti, kteří budou procházet seznamem předmětů jednotlivých dotazů, dokázali váš dotaz jednoznačně zařadit. Při problémech s HW uveďte pokud možno jeho identifkátor (tj. např. LG 4163B) jako součást předmětu. Stejně tak v případě problémů s konkrétní komponentou (X-server, Apache, atd.) ji v předmětu zmiňte.

Mezi odstrašující příklady předmětů patří:

- *‚mám problém, potřebuji pomoc‘* - to, že máte problém a potřebujete pomoc, je jasné
- *‚poradte, specha!!!!!!¡* - vyvarujte se použití vykřičníků i pokusů o urgenci (potřebujete-li pomoc rychle, bude vhodné využít nějaké IM služby typu IRC nebo Jabber, diskusní fóra jsou běh na delší trať)
- *‚64-bit‘* - příliš obecné, buďte co nejkonkrétnější
- *‚Píše mi to‘* - vyvarujte se také využití části první věty textu příspěvku pro zaplnění předmětu
- *‚restart‘* - nejasné
- *‚APACHE NESTARTUJE, SKONČÍ S CHYBOVOU HLÁŠKOU‘* - netiketa považuje kapitálky za ekvivalent křičení, takže, prosím, nekřičte

Předmět by měl vypadat nějak takto:

- *‚LG 4163B se sekne při pálení‘*
- *‚převod rpm balíčku na deb‘*
- *‚Courier-Maildrop s podporou mysql na Debianu‘*

[1] http://www.abclinuxu.cz/, http://www.root.cz/, apod.

Text dotazu by měl stručně a jasně popsat problém a jeho příznaky. Měl by obsahovat všechny informace podstatné pro řešení problému (HW konfigurace, použitá distribuce a její verze) a také vaše vlastní kroky, které jste podnikli k řešení problému. Pokud jste postupovali podle nějakého návodu, přiložte odkaz na něj. Pokud se snažíte něčeho dosáhnout, a máte problém s prostředkem zvoleným k dosažení tohoto cíle, nezapomeňte uvést i samotný cíl. Může vám být doporučen jiný prostředek, se kterým svého cíle dosáhnete.

9.2.3 Po položení dotazu

Stav svého dotazu průběžně sledujte, doplňujte informace podle požadavků a pokud se v řešení problému dostanete dále, co nejdříve o tom spravte diskutující.

9.2.4 Čeho se vyvarovat

Především jakéhokoliv neslušného, provokativního či arogantního jednání. Pokud přispíváte do veřejných fór, nežádejte o "zaslání řešení na e-mail", je to sobecké. Řešení vašeho problému probrané ve fóru může pomoci dalším lidem, kteří se dostanou do podobné situace a prohledají příslušné fórum.

9.2.5 Interpretace odpovědi

Zřídka dostanete přesně popsaný postup, jak problém vyřešit. Spíše očekávejte radu nebo odkaz na nějaký dokument, kde je váš problém popsán. Pokud dostanete radu typu RTFM (přečtěte si manuál) nebo STFW (prohlejdete web), nejspíše jste podcenili řešení problému vlastními silami a neprohledali jste příslušné informační zdroje. Z tohoto důvodu doporučuji vámi podniklé kroky v textu dotazu uvést.

Může se stát, že odpověď nedostanete, nejspíše proto, že nikdo odpověď nezná. V takovém případě dotaz neopakujte, ale vyhledejte jiné fórum, kde je větší šance, že váš dotaz bude zopdovězen.

Dodatek A

Odkazy

A.1 Portály

- ABC Linuxu, http://www.abclinuxu.cz/ (česky)
- LinuxSoft, http://www.linuxsoft.cz (česky)
- Root, http://www.root.cz (česky)
- LinuxOS.sk, http://www.linuxos.sk/ (slovensky)
- Penguin, http://www.penguin.cz/ (česky)
- Polishlinux, http://polishlinux.org/ (anglicky, polsky)
- Desktop Linux, http://www.desktoplinux.com/ (anglicky)

A.2 Zpravodajské weby, časopisy

- Linux Expres http://www.linuxexpres.cz/ (česky)
- Linux Zone http://www.linuxzone.cz/ (česky)
- Linux Journal http://www.linuxjournal.com/
- Slashdot http://www.slashdot.org/ (anglicky)

A.3 Dokumentační weby

- Úvod do systému Linux, http://wraith.iglu.cz/usl/usl.html (česky)
- Učebnice Linuxu, http://www.abclinuxu.cz/ucebnice (česky)

- Otázky uživatelů, kteří se začínají zajímat o Linux, http://bbs.cvut.cz/~covex/-linux/newbie.html (česky)
- Dokumentační projekt, http://www.tldp.org/ (anglicky)
- Really Linux, http://www.reallylinux.com/ (anglicky)

A.4 Distribuční weby

- Distrowatch, http://www.distrowatch.cz/ (seznam distribucí)
- Live distribuce
  - Slax, http://www.slax.cz/ (česky), http://www.slax.org/ (anglicky)
  - Knoppix, http://www.knoppix.org/ (anglicky), http://www.knoppix.com/- (německy)
- Klasické distribuce
  - Ubuntu, http://www.ubuntu.cz/ (česky), http://www.ubuntulinux.org/ (anglicky)
  - Mandriva, http://www.mandrivalinux.cz/ (česky), http://www.mandrivalinux.com/- (anglicky)
  - SUSE, http://suseportal.cz/ (český portál), http://www.novell.com/linux/-suse/ (anglicky), http://en.opensuse.org/Documentation (dokumentační web, anglicky)
  - PCLinuxOS, http://pclinuxos.cz/ (česky), http://www.pclinuxos.com/ (anglicky)
  - Fedora, http://fedora.cz/ (česky), http://fedoraproject.org/ (anglicky)
  - Debian, http://www.debian.cz/ (česky), http://www.debian.org/ (anglicky)
  - Slackware, http://www.slackware.cz/ (česky), http://www.slackware.org/- (anglicky)
  - Gentoo, http://www.gentoo.cz/ (česky), http://www.gentoo.org/ (anglicky)
  - Arch Linux, http://www.archlinux.cz/ (česky), http://www.archlinux.org/- (anglicky)

A.5 Vyhledávání

- Fulltextové vyhledávače
  - http://google.cz/
  - http://google.cz/linux
  - http://www.jyxo.cz/

  - http://www.alenka.cz/

- Vyhledávání v diskusních skupinách
  - http://usenet.jyxo.cz/
  - http://groups.google.com/

- Vyhledávání ve zdrojových kódech
  - http://www.google.com/codesearch
  - http://krugle.com/

A.6 Software

- Obecné katalogy a seznamy softwaru
  - Tabulka ekvivalentních aplikací, http://proc.linux.cz/ekvivalenty.html (česky)
  - LinuxSoft, http://www.linuxsoft.cz/ (česky)
  - ABC Linuxu, http://www.abclinuxu.cz/software (česky)
  - Freshmeat, http://freshmeat.net/ (anglicky)
  - SourceForge, http://sourceforge.net/ (anglicky)

- Počítačové hry (vše anglicky)
  - http://happypenguin.org/ (zejména FOSS hry)
  - http://tuxgames.com/ (komerční hry)
  - http://www.linuxgamepublishing.com/ (komerční hry)

A.7 Repositáře

- Debian
  - http://www.debian-multimedia.org/ - multimédia

- Fedora
  - http://rpm.livna.org/ - multimédia a proprietární ovladače pro grafické karty

- Mandriva
  - http://easyurpmi.zarb.org/?language=cz - snadné přidávání repositářů

- Víceúčelové
  - http://seerofsouls.com/ - repositář pro Ubuntu a Mandrivu
  - http://packman.links2linux.org/ - repositář pro Ubuntu, Debian a (Open)SUSE

A.8 Balíčky

- Vyhledávání balíčků
  - http://www.tuxfinder.com/ (obecné)
  - http://www.debian.org/distrib/packages (Debian)
  - http://packages.ubuntu.com/ (Ubuntu)
  - http://www.linuxpackages.net/ (Slackware)
  - http://gentoo-portage.com/Browse (Gentoo)
- Vyhledávání balíčků (RPM)
  - http://rpm.pbone.net/
  - http://rpmseek.com/
  - http://rpmfind.net/

A.9 Ostatní

- Propagační weby
  - http://proc.linux.cz/ (česky)
  - http://www.getgnulinux.org/ (anglicky)

Dodatek B

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration,

to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page"

means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from

which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version

as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/ .

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title

page:

> Copyright (C) YEAR YOUR NAME.
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.