

Ubuntu Debian správce serveru díl I

Libor Jelínek



Pro všechny správce jiných operačních systémů i
začátečníky, kteří přecházejí na Ubuntu nebo Debian


Vítá vás příručka Ubuntu a Debian správce serveru I.!

Vítáme budoucí i současné administrátory v příručce „Ubuntu Debian správce serveru I.“! Tato publikace může být skvělým doplňkem k našim [kurzům Linuxu](#), ale je koncipována jako zcela samostatná. Nabízíme ji zdarma všem návštěvníkům a samozřejmě našim studentům. Budeme rádi, když vám pomůže naučit se Ubuntu, Debian nebo dokonce i jinou distribuci Linuxu.

Tip

Líbí se vám tato knížka? Přijďte na [školení od autorů této příručky](#) na [Vacademy.cz](#)!

Licenční ujednání

Copyright © 2017 Virtage Software. Tato příručka je publikována pod  licencí [CC BY-NC-SA 4.0](#) (Uved'te původ-Neužívejte dílo komerčně-Zachovejte licenci 4.0 Mezinárodní). Tato licence dovoluje text sdílet a distribuovat v jakémkoli formátu nebo médiu *kromě použití pro výdělečné účely*. Text můžete upravovat a pozměňovat, pokud zachováte stejnou licenci. Vystavitel licence může tyto podmínky v budoucnu upravovat. Při sdílení a šíření je nutné uvést původ např. URL odkazem.

1. Úvod

Ačkoli mluvíme o Linuxu, ve skutečnosti myslíme určitou distribuci Linuxu. Mezi desítkami či spíše stovkami distribucí Linuxu je již mnoho posledních let oprávněně nejrozšířenější Ubuntu, které vychází z distribuce Debian. Ačkoli informace uvedené v příručce lze použít i pro jiné distribuce, než tyto dvě, konkrétní příklady a informace v této příručce jsme ověřovali v

- Ubuntu 16.04 LTS Server (Xenial) z dubna 2016
- Ubuntu 14.04 LTS Server (Trusty) z dubna 2014

Edice *Ubuntu Server* standardně neobsahuje grafické rozhraní, ale neliší se možnostmi od edice *Ubuntu Desktop*. Instalátor Ubuntu Server je rovněž textový, ale také velmi podobný tomu grafickému.

Většina informací v příručce předpokládá přítomnost pouze textového prostředí. Čas od času zmíníme i grafickou alternativu.

Vzhledem k blízké příbuznosti mezi Ubuntu a Debian je příručka téměř bez výjimky **platná i pro distribuci Debian**. Těch několik rozdílů mezi těmito systémy vždy zdůrazníme. Kromě Ubuntu a Debianu informace platí samozřejmě i pro všechny deriváty těchto operačních systémů jako Kubuntu, Xubuntu, Lubuntu, Mint a další.

Bohužel vzhledem k někdy značným rozdílům mezi rodinami Ubuntu/Debian a ostatními distribucemi jako RHEL (Red Hat Enterprise Linux), Fedora a SUSE (openSUSE, SUSE Linux Enterprise) nemusí být informace vždy správné i pro tyto systémy.

Důležité

Většina kapitol je však natolik „univerzálně linuxová“, že platí pro prakticky jakoukoli současnou linuxovou distribuci. Snad jediná vyloženě specifická kapitola pro Ubuntu/Debian je [Instalace a správa programů](#).

1.1. Historie Linuxu

V této části si povíme některé sice netechnické, ale i tak důležité informace, které z vás udělají lepšího správce. Není výjimečné, že správce Linuxu je současně i nadšeným fanouškem a propagátorem těchto systému a proto by měl vědět některé podstatné údaje z historie Linuxu, Ubuntu a Debianu.

Linux vznikl v roce 1991 jako osobní projekt finského vysokoškoláka Linuse Torvaldse. Mělo se jednat o svobodný operační systém unixového typu. Projekt vyvolal obrovský zájem a současný úspěch a vliv Linuxu je asi nezpochybnitelný. Maskotem Linuxu je tučňák.

Ovšem Linux staví na mnohem starších kořenech, protože historie unixových OS sahá několik desítek let zpátky. Za první unix se považuje projekt z roku 1969 vyvinutý AT&T Bell Labs. Značného rozšíření dosáhl po roce 1976, kdy byl nabídnut zdarma univerzitám. Řada dnešních komerčních unixů (HP-UX, Solaris, IRIX) vychází právě z AT&T Unixu.

Dalším významným unixem byl BSD Unix (Berkeley Software Distribution) z roku 1977, který má dodnes několik populárních svobodných potomků (FreeBSD, OpenBSD, NetBSD). Maskotem BSD unixů je čertík.

Linux přímo nesdílí kód s AT&T ani BSD Unixem a svými vlastnostmi je někde uprostřed.

1.2. Distribuce Linuxu

Úplně striktně vzato pod pojmem Linux myslíme „jen“ samotné jádro operačního systému (*kernel*). To sice zajišťuje klíčové funkce jako obsluha I/O (disky, soubory), síťový subsystém, ovladače ap., ale samo o sobě netvoří funkční celek.

Jádro Linuxu může díky licenci GPL kdokoli použít a „přibalit“ k němu další software, určit způsob konfigurace, přizpůsobit použití pro určitý účel (desktop, server, multimédia, bezpečnost), atd. Výsledku se pak říká linuxová distribuce.

V současné době existují desítky distribucí, dokonce možná stovky, pokud počítáme i jejich deriváty (odvozené distribuce). Tato skutečnost je výhodou i nevýhodou. Linuxový svět je velkou velmi čilou líní technologických inovací, ale na druhou stranu si velká rozmanitost a svoboda vybírá svou daň v podobě menší „uhlazenosti“ a přehlednosti a někdy i stability kódu.

1.2.1. Ubuntu

Dlouhodobě nejrozšířenější distribucí je Ubuntu, na které se tento kurz především zaměřuje. Důvodem masivního úspěchu je snadnost použití, perfektní podpora HW (většina HW funguje bez instalace dodatečných ovladačů) a spolupráce s řadou výrobců (nejznámější partnerství je s Dell). S rozšířeností souvisí i komunita, která čítá tisíce nadšených dobrovolníků, kteří o Ubuntu píší a diskutují, a tak je pravděpodobnost, že jste první s určitým problémem, téměř nulová.

Verzování

První verze Ubuntu byla vydána v říjnu 2004 a nejmenovala se 1.0, ale 4.10, tedy ROK.MĚSÍC. Toto verzovací schéma používá Ubuntu dodnes. Snadno tak víme, že např. Ubuntu 12.04 bylo vydáno v dubnu 2012.

Ubuntu je vydáváno v půlročních cyklech vždy v dubnu (verze X.04) a říjnu (X.10).

Jednou za dva roky je vydávána verze LTS (Long Term Support) s prodlouženou podporou na 5 let. LTS verze jsou vhodné pro nasazení na servery a větší množství desktopů.

Běžná verze má podporu 9 měsíců a proto se méně hodí na server, ale pro desktop je určitě správnou volbou. Zejména, chcete-li zkusit novinky a mít vždy to nejnovější.

Canonical a Mark Shuttleworth

Ubuntu z velké části vyvíjí společnost Canonical, kterou založil v roce 2004 Mark Shuttleworth. Mark byl známou osobní již dříve, a to díky založení společnosti Thawte (SSL certifikáty), kterou později výhodně prodal. Nejznámější je však Mark jako historicky druhý vesmírný turista, který se podíval na Měsíc.

Slovo Ubuntu znamená v jazyce jihoafrického kmene Zulu „lidskost ostatním“ nebo jen „lidskost“. Název vybral zakladatel Mark Shuttleworth, který je původem právě z Jihoafrické republiky.

Deriváty

Samotné Ubuntu má řadu derivátů, které se liší svou specializací. Nejčastěji nahrazují jiné grafické prostředí místo výchozího Gnome/Unity. Mezi nejznámější patří např.

- Kubuntu – Ubuntu s KDE grafickým prostředím
- Xubuntu – Ubuntu s Xfce grafickým prostředím vhodné pro méně výkonné počítače
- Lubuntu – Ubuntu s LXDE grafickým prostředím vhodné pro nejslabší počítače
- Edubuntu – Ubuntu sestavené s programy pro školy

Ubuntu Server

Každá verze Ubuntu je k dispozici také v „edici“ Server. Ubuntu Server vychází z běžného Ubuntu (někdy zvané Desktop).

Má k dispozici stejné APT repozitáře, ale odlišnost spočívá zvláště v nepřítomnosti grafického prostředí a tím souvisejícího GUI software. Také se trochu jinak instaluje, avšak jádro používá stejné jako desktopové edice.

Ubuntu Desktop lze použít i pro serverové nasazení, ale Server edice bude fungovat rychleji, je štíhlejší a sami postupně přijdete na to, že grafické prostředí ke správě nepotřebujete.

Další varianty Ubuntu

Popularita Ubuntu je obrovská, a tak najdete oficiální i komunitní varianty Ubuntu téměř pro jakékoli zařízení.

- Ubuntu TV – Ubuntu určené pro výrobce set-top-boxů a tzv. smart televizorů.
- Ubuntu Touch – varianta Ubuntu pro mobilní telefony a tablety, která umí nahradit Android na vašem mobilu
- Chromebook, PlayStation, Xbox – pro všechny tento HW najdete deriváty Ubuntu, kterými můžete nahradit původní operační systém.

1.2.2. Debian

Debian a Ubuntu mají mnoho společného. Ubuntu vzniklo právě jako odštěpek (fork) od Debianu. Debian je však přísně nekomerční, vyvíjený výhradně komunitou GNU okolo Free Software Foundation a nadšenci. Dokonce jako jediná distribuce se oficiálně pyšní přídomek „GNU/Linux“.

Také název Debian má svůj romantický původ. Původní zakladatel Ian Murdock, tehdy student americké Purdue University, pojmenoval projekt podle kombinace jména své přítelkyně Debra a svého Ian.

Debian je znám svou konzervativností. Verze nejsou vydávány tak často jako Ubuntu a neobsahují vždy nejnovější verze software, ale na druhou stranu je Debian považován za velmi spolehlivý a odladěný systém.

Jako jediná masivně rozšířená distribuce za sebou nemá velkou firmu. Téměř celý systém je skutečně vyvíjen komunitou nadšených a talentovaných lidí.

Ubuntu využívá systém balíčků a software původně navržený pro Debian, ale poslední dobou najdeme i obrácený směr symbiózy - technologie z Ubuntu se dostávají do Debianu.

1.2.3. Red Hat a Fedora

Red Hat býval v 90. letech synonymem pro Linux. Nyní má společnost Red Hat placený produkt Red Hat Enterprise Linux (RHEL) a volně přístupnou Fedoru s nejistou koncepcí, budoucností a menší uživatelskou komunitou. Tímto rozdělením a především zplatněním dost ztratil na svém rozšíření.

Na druhou stranu se cílení RHEL na velké podniky finálně vyplácí. Red Hat je silná a prosperující firma. Mimo velké podniky je jeho nasazení, vzhledem k vysokým licenčním poplatkům, málo časté.

1.2.4. SUSE a openSUSE

Distribuce SUSE má také dlouhou historii a těší se oblíbenosti rovněž ve velkých podnicích. Původně malá německá firma stojící za SUSE byla koupena společností Novell, ale ta byla sama v roce 2011 prodána Attachmate Group a SUSE se nyní vyvíjí v rámci této firmy jako samostatná obchodní jednotka.

SUSE se před relativně dlouhou dobou rozhodla pro podobný krok jako Red Hat a rozdělila se na placený SUSE Linux Enterprise (SLE) a volný openSUSE.

SUSE je pravděpodobně třetí a poslední rozšířenější verzí Linux, o které běžně uslyšíte. Často je považována za „nejuhlazenější“ Linux vůbec.

1.2.5. Která distribuce je ta nejlepší?

Doufáme, že zde opravdu nečekáte odpověď :-). Ta totiž závisí na tom, jak chcete Linux používat, jaké máte znalosti Linuxu a jaká další kritéria posuzujete (stabilita, rozšířenost, úroveň podpory).

Myslíme si však, že Ubuntu má pozici nejrozšířenější linuxové distribuce oprávněně. A to vzhledem k poměru ceny (zadarmo), funkčnosti a snadnosti použití.

Líbí se nám, že na rozdíl od Red Hat a SUSE můžeme vytvořit jakoukoli infrastrukturu bez ohledu na náš rozpočet. Oproti Debianu považujeme za výhodu větší komunitu, lepší podporu hardware a pro kritické nasazení můžeme přikoupit [podporu od výrobce Ubuntu Advantage](#).

1.3. Knihy a weby

Na závěr úvodní části bychom vás chtěli upozornit na některé knihy a weby, které považujeme za vhodný zdroj dalších informací.

- *knihy a web Linux: dokumentační projekt* – [anglický web projektu](#) nebo [česká kniha](#) z těchto návodů. Tato kniha byla dříve k zakoupení u nakladatelství [CPress](#), ale již není v nabídce.
- *knihy Linux kompletní příručka administrátora* od CPress.
- *magazín Root* - <http://www.root.cz>
- *magazín AbcLinuxu* - <http://www.abclinuxu.cz>
- *magazín Linuxsoft* - <http://linuxsoft.cz>
- *weby Ubuntu.cz* a zejm. *wiki.ubuntu.cz* – <http://www.ubuntu.cz>, <http://wiki.ubuntu.cz>

2. Instalace

Všechny následující postupy můžete uplatnit samozřejmě rovnou na váš počítač, ale doporučujeme instalaci a první kroky v Ubuntu zkusit v prostředí virtuálního počítače. Můžete použít volně dostupný virtualizační software [Virtual Box](#), který existuje pro Linux, Mac i Windows. Případně jemu podobný (ale placený) VMware. Nástroj Microsoft Virtual PC nedoporučujeme vzhledem ke špatné podpoře Linuxových hostů.

Poznámka

Obrázky uvedené u následujících odstavců jsou z Ubuntu 16.04 LTS Server, ale kroky instalátoru jsou téměř identické s grafickou verzí setupu pro Ubuntu Desktop.

2.1. Výběr architektury

Pod architekturou myslíme typ procesoru. V dnešních PC a serverech se můžete setkat s procesorovými architekturami, které jsou v Ubuntu označovány následujícími zkratkami.

Architektura	Popis
i386	32bitové CPU starších počítačů
amd64	64bitové CPU dnešních počítačů. Jsou takto označeny CPU od AMD i Intelu. Rovněž počítače s UEFI firmwarem (přeinstalovaný Windows 8 nebo 10)
amd64+mac	Ubuntu můžete provozovat místo OS X na Mac počítači nebo laptopu

Mezi další podporované CPU architektury patří např. ARM procesory.

Tip

Jak zjistit „bitovost“ mého CPU?

V dnešní době je téměř jisté, že váš počítač podporuje 64bitové instrukce. Častou chybou je plést bitovost OS a CPU, protože můžete mít 64bitový CPU, ale nainstalován 32bitový OS.

Chcete-li si být jisti, že jakou architekturu má CPU, proveďte z Ubuntu nebo jiného Linuxu příkaz `lscpu`. Je-li v řádku `Architecture` hodnota `x86_64` je váš CPU 64bitový.

Pro Windows si stáhněte např. program [CPU-Z](#).

2.2. Kde stáhnout

Doporučujeme instalátor stáhnout z českých stránek [Ubuntu.cz](#). Možná ještě lepší je <http://releases.ubuntu.cz/>, kde najdete všechny starší i aktuální verze a všechny jejich deriváty a edice na jednom místě. Další možností je stažení prostřednictvím BitTorrentu.

Instalátor je vždy v podobě ISO souboru, tedy bitové kopie disku. Jméno souboru obsahuje verzi, edici a označení architektury. Např. `+ubuntu-14.04-server-amd64.iso+` je Ubuntu 14.04 LTS Server pro 64bitové počítače.

Tip

Pokud si nejste jisti jakou architekturu zvolit, vyberte si 64bitovou.

2.3. Instalační média

CD/DVD

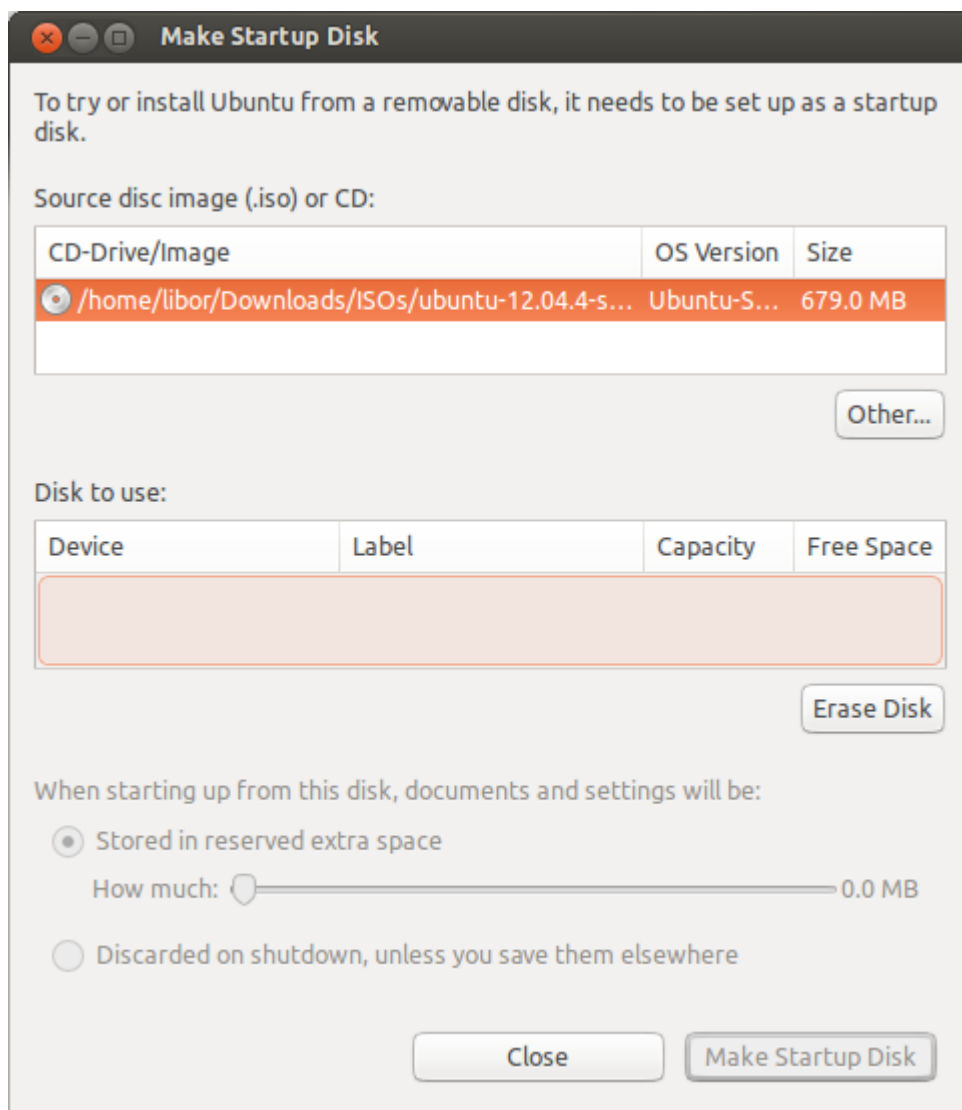
Asi vás překvapí malá velikost ISO souboru (hlavně pokud jste zvyklí na instalátory Windows a Mac OS X). Pro Ubuntu Desktop dlouho platilo, že to bylo 700 MB, aby

se ISO dal vypálit na staré dobré CD. Dnes už je velikost větší a použít proto musíme DVD(+/-)R. Např. Ubuntu Server 16.04 má kolem 800 MB.

USB flash

Kromě této tradiční metody můžete provést instalaci z USB flash disku. Přenos ISO souboru na USB i samotná instalace je mnohonásobně rychlejší (a tišší :-)), než instalace z CD/DVD.

Z již existující instalace Ubuntu použijte program *Startup Disk Creator* (z příkazové řádky `usb-creator-gtk`).



Pro Windows a Mac můžete zkusit třeba program [Unetbootin](#). Ten umí „vypálit“ na USB již dříve stažené ISO a taktéž sám ISO stáhnout i vypálit.

Síťová instalace (Netboot)

Neobvyklou metodou je instalace prostřednictvím sítě zvaná Netboot. Více informací najdete např. na <https://help.ubuntu.com/community/Installation/Netboot>.

Wubi

Wubi nebo *Windows Ubuntu Installer* nabízí možnost, jak nainstalovat Ubuntu na stávající disk Windows bez nutnosti rozdělovat oddíly. Jinými slovy je to výborná možnost jak zkusit Ubuntu přímo na PC nevirtualizovaně.

Wubi nainstaluje Ubuntu do (z pohledu Windows) jediného obrovského souboru na vybraném existujícím disku Windows. Při každém startu si budete moci vybrat, zda načíst běžné Windows nebo Ubuntu.

2.4. Upgrade

Pokud už máte nainstalováno Ubuntu nemusíte ho smazat a provádět čistou instalaci. Pro server nebo dlouho používaný desktop je to sice možnost jak „vyčistit“ počítač, ale na druhou stranu Ubuntu ani zdaleka netrpí tolik známou bolestí Windows, kdy se starší instalace stává pomalejší a pomalejší.

Rozhodnete-li se pro upgrade na vyšší verzi Ubuntu použijte příkaz `do-release-upgrade`. Některé učebnice vás budou navádět k `apt-get disk-upgrade`, které sice funguje ve všech distribucích založených na Debianu, ale neumí ošetřit změny konfigurace systému mezi vydáními.

Důležité

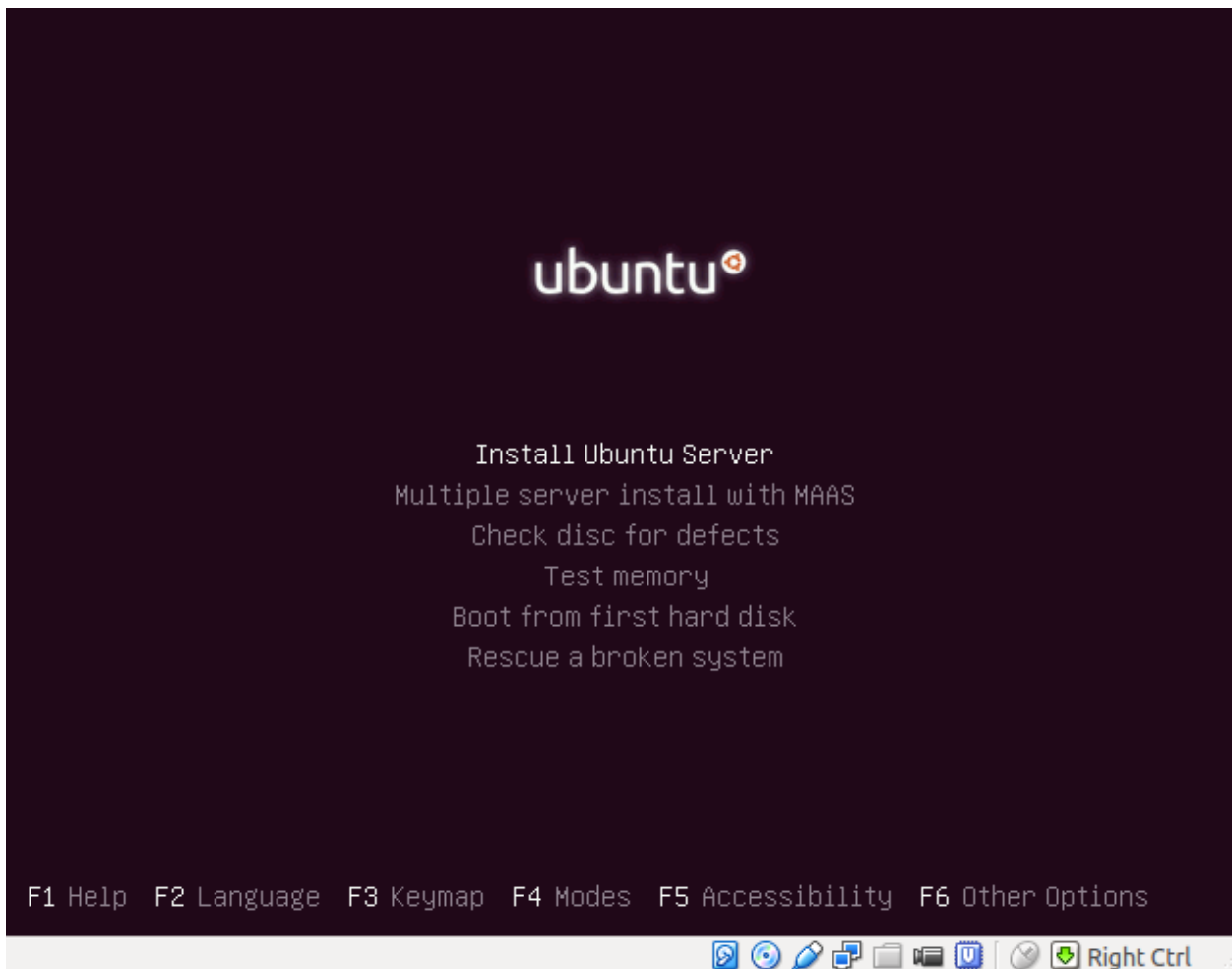
Rozlišujte mezi pojmy *upgrade* a *update*. Povýšení celého OS (tj. všech jeho balíčků) je *upgrade*, aktualizaci jednotlivých programů se říká *update*. Více také [Update vs. upgrade](#).

2.5. Další možnosti ISO disku

Ať už provádíte instalaci z CD/DVD nebo USB flash, ISO disk umožňuje kromě samotné instalace, také další užitečné volby:

- možnost zkontrolovat paměť RAM počítače

- možnost zkontrolovat stažený ISO soubor na případné chyby
- nastartovat čistý funkční systém z ISO a zachránit ten současný na disku



2.6. Regionální nastavení

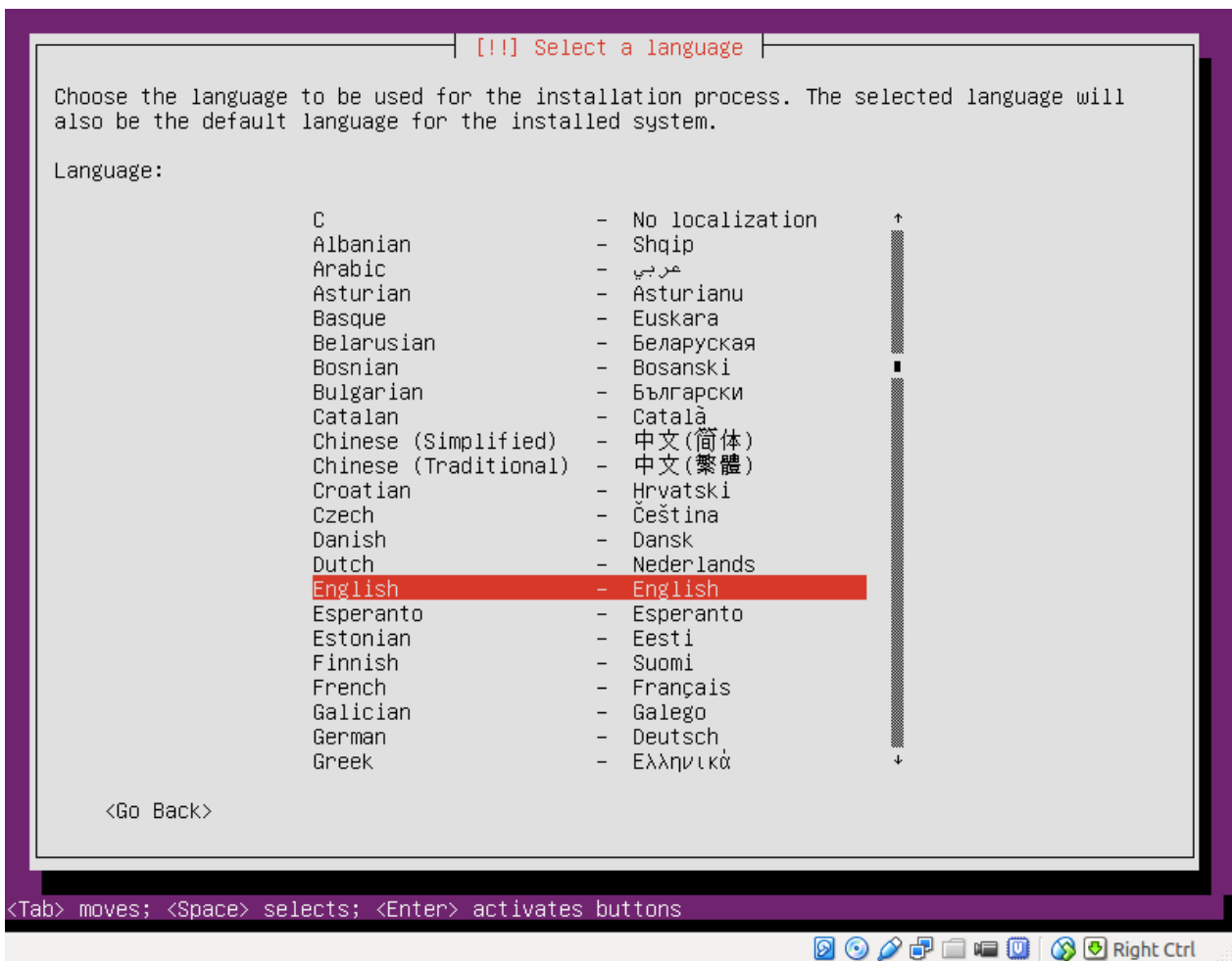
K nejjednodušším otázkám instalátoru patří volba jazyka, klávesnice, místního nastavení a časové zóny.

Language

Amharic	Français	Македонски	Tamil
Arabic	Gaeilge	Malayalam	தமிழ்
Asturiano	Galego	Marathi	Thai
Беларуская	Gujarati	Burmese	Tagalog
Български	עברית	Nepali	Türkçe
Bengali	Hindi	Nederlands	Uyghur
Tibetan	Hrvatski	Norsk bokmål	Українська
Bosanski	Magyar	Norsk nynorsk	Tiếng Việt
Català	Bahasa Indonesia	Punjabi (Gurmukhi)	中文(简体)
Čeština	Íslenska	Polski	中文(繁體)
Dansk	Italiano	Português do Brasil	
Deutsch	日本語	Português	
Dzongkha	தமிழ்	Română	
Ελληνικά	Қазақ	Русский	
English	Khmer	Sámegillii	
Esperanto	ಕನ್ನಡ	සිංහල	
Español	한국어	Slovenčina	
Eesti	Kurdî	Slovenščina	
Euskara	Lao	Shqip	
عەرەبى	Lietuviškai	Српски	
Suomi	Latviski	Svenska	

F1 Help F2 Language F3 Keymap F4 Modes F5 Accessibility F6 Other Options

 Right Ctrl



Pro server doporučujeme angličtinu kvůli možnosti snadného hledání znění hlášky či chyby na internetu. Pokud spravuje server více osob nebo se pohybujete v mezinárodním prostředí, určitě volte i anglickou klávesnici (US keyboard).

Pro desktopový počítač si můžete vybrat beze všeho češtinu/slovenštinu, je-li vám příjemnější, než angličtina.

Pro Českou republiky jako umístění musíme nejprve zvolit *Other* a pak *Europe*, v následném seznamu teprve najdeme *Czech Republic*.

Klávesnici můžete zvolit buď detekcí nebo vybráním ze seznamu. Při detekci jste vyzváni zadat určité neobvyklé znaky, podle kterých instalátor pozná, jaké rozvržení klávesnice očekáváte.

Všechna tato nastavení se dají kdykoli změnit.

2.7. Uživatelé

Během instalace se založí kromě účtu Superuživatel root, další účet pro naši běžnou práci. Pečlivě si proto zvolte a zapamatujte vaše uživatelské jméno a heslo.

2.8. Spouštění více operačních systémů

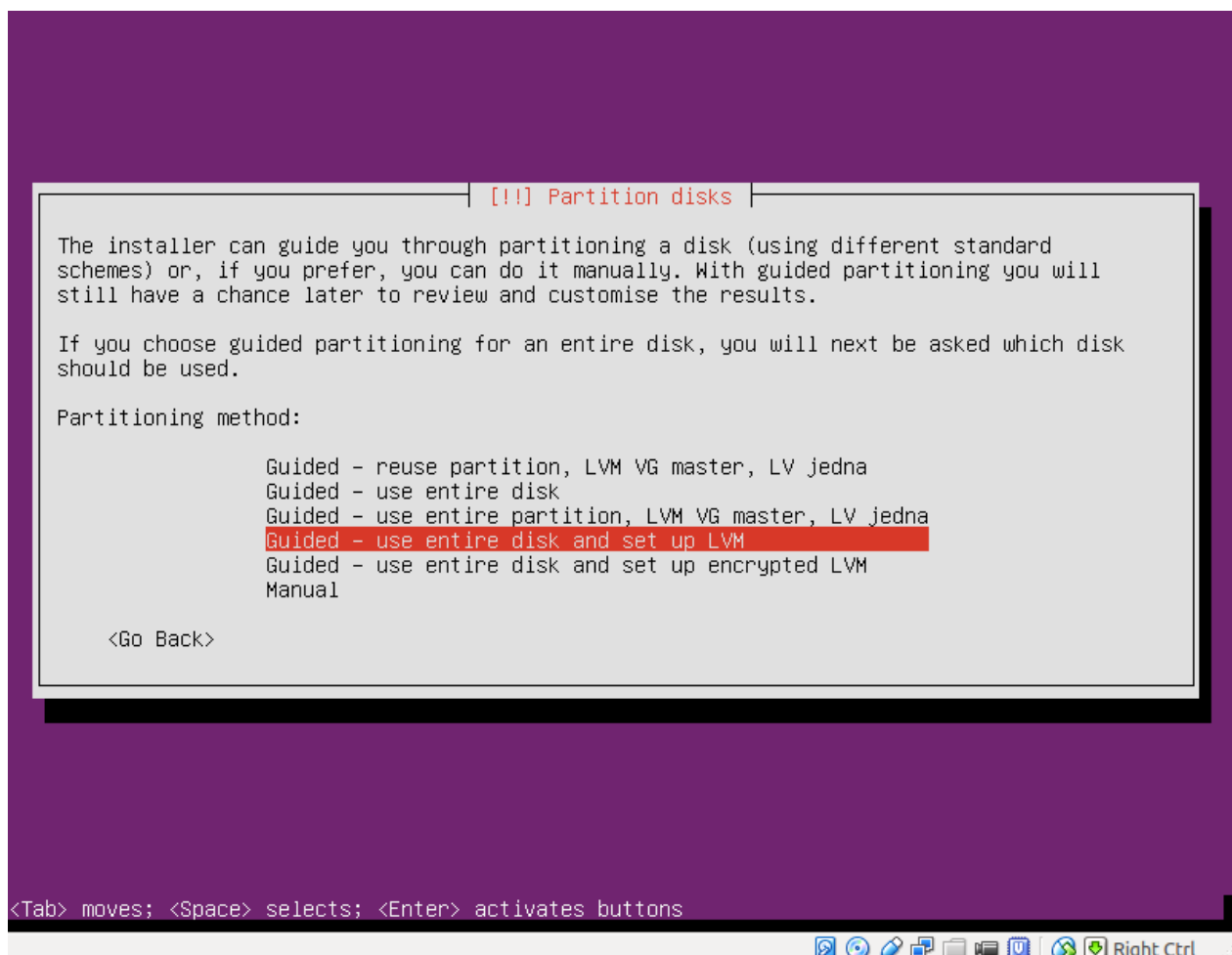
Instalátor Ubuntu se chová velmi zdvořile pokud na discích detekuje jiný operační systém. Nabídne vám:

- nainstalovat Ubuntu společně se stávajícím OS, přičemž si po startu PC zvolíte, které chcete spustit
- smazat stávající OS a použít celý disk jen pro Ubuntu

Pokud chcete provozovat Ubuntu a Windows zároveň, doporučujeme nejprve nainstalovat Windows např. na polovinu disku a teprve poté nainstalovat Ubuntu. Windows (myslíme si, že záměrně) jiný OS nerozpozná a velmi rád poškodí jeho oddíl nebo MBR (zruší možnost výběru OS po startu).

2.9. Příprava disku a oddílů

Instalátor nabízí možnost manuálního rozvržení disků nebo „automatické“ (volby *Guided*). Ve vlastní praxi vždy provádíme rozdělení disků manuálně např. podle následujících zásad.



2.9.1. Oddíly

Pro instalaci každého Ubuntu (a každého jiného Linuxu) potřebujete nejméně dva, spíše tři oddíly (partition).

- oddíl pro samotný OS naformátovaný na standardní linuxový ext4 nebo jiný podporovaný filesystem (ReiserFS, XFS, JFS ap.)
- swapovací oddíl, který není záměrně nijak naformátován
- ideálně další oddíl pro domovské složky uživatelů, aby jste mohli např. přeinstalovat OS, ale přitom zachovat veškeré soubory a nastavení uživatelů
- v závislosti na účelu serveru nebo v rámci ladění výkonu ještě další oddíly pro např. odkládání logů, dočasné soubory, transakční log databáze ap.

Velikost swap oddílu

O té „správné“ velikosti swap oddílu se vedou letité spory a najdete řadu protichůdných rad. Někdo nastavuje velikost stejnou jako RAM, někdo 1,5x velikosti RAM, někdo 2x velikosti RAM. Instalátor Ubuntu standardně volí o něco málo větší swap oddíl, než RAM.

Každý oddíl kromě swapovacího je připojen na nějakou složku v hierarchii souborového systému, které se říká *přípojný bod (mount point)*.

Příklad rozvržení disku pro server

souborový systém	přípojný bod	velikost	popis
ext4	/	alespoň 5 GB	samotný OS
ext4	/home/	alespoň 1 GB	domovské složky uživatelů
-	-	např. 1,5 násobek RAM	swap oddíl
ext4	/var/	podle účelu	ve /var/ jsou „data aplikací“ např. webové stránky, logy, soubory databáze ap.

Příklad rozvržení disku pro desktop

souborový systém	přípojný bod	velikost	popis
ext4	/	alespoň 10 GB	samotný OS
ext4	/home/	alespoň 1 GB	domovské složky uživatelů
-	-	např. 1,5 násobek RAM	swap oddíl

2.9.2. LVM a tradiční oddíly

Možná víte, že tradiční oddíly můžou být v rámci jednoho fyzického disku pouze čtyři. Oddíly jsou dvou typů - *primární (primary)* a *rozšířené (extended)*. V rámci rozšířeného oddílu můžete vytvořit další tzv. logické oddíly a tím limit čtyř oddílů překonat.

Jiným modernějším přístupem je *LVM neboli Logical Volume Management* (někdy uváděno jako *Linux Volume Management*), který kromě rušení limitu 4 oddílů nabízí řadu dalších výhod, např.:

- vytvářet logické svazky napříč více fyzickými disky
- přesouvat svazky mezi fyzickými disky
- za běhu zvětšovat a zmenšovat velikost oddílu bez ztráty dat
- za běhu vytvářet snímky celých souborových systémů
- za běhu vyměňovat fyzické disky
- oddíl za běhu šifrovat (on-the-fly)

Nevýhodou je, že se musíme naučit používat nové nástroje místo tradičních programů.

2.10. Package tasks (groups)

V předposledním kroku instalace máme možnost nainstalovat skupiny programů podle účelu serveru jako např. OpenSSH, Tomcat server, MySQL ap.

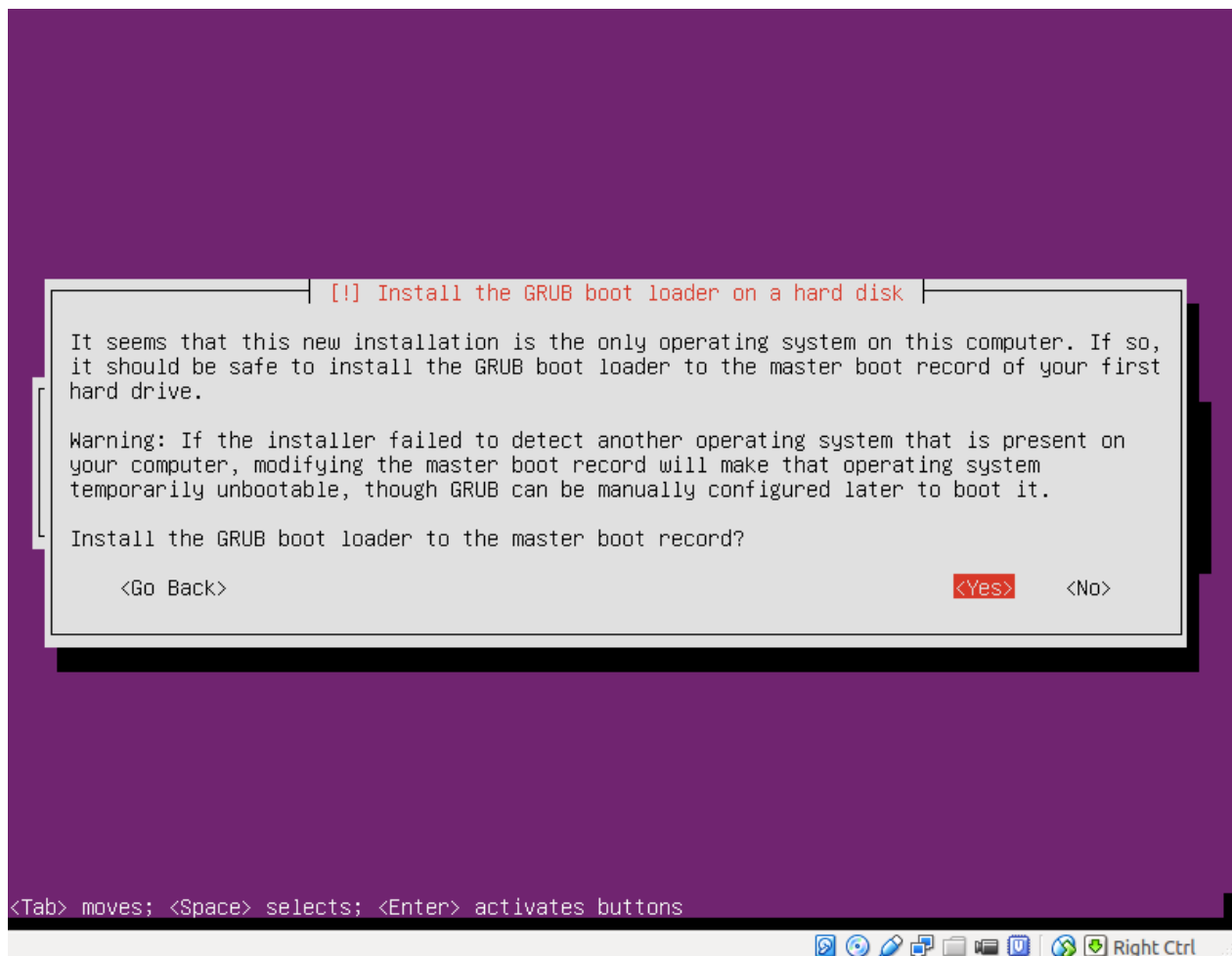
Při prvním seznamování s Ubuntu doporučujeme této možnosti nevyužívat a raději si potřebný software nainstalovat manuálně krok za krokem, abysme se naučili postup instalace a konfigurace těchto programů.

2.11. GRUB

Poslední otázka instalátoru směřuje k instalaci spouštěče GRUB. Je to modernější náhrada LILO (Linux LOader), kterou možná znají někteří „pamětníci“ Linuxů z konce 90. let.

GRUB je program, který se spustí jako úplně první po startu PC a umožňuje nám vybrat si jaký OS chceme načíst. Může to být Ubuntu a Windows, více verzí kernelu Linuxu a Windows ap.

Pokud počítač neobsahuje dosud žádný boot manager, pak využijte možnosti instalátoru nahrání GRUB do MBR. Pokud např. GRUB již máte (protože provozujete dva Linuxy), pak to není nutné.



3. Příkazová řádka

V této kapitole se seznámíme s příkazovou řádkou Linuxu, která je hlavním nástrojem a pracovním prostředím každého administrátora. Zvládnutí a navyknutí si na příkazovou řádku je nutná podmínka bez, které nemůžeme Linux efektivně používat a ovládat.

Zcela mylná je představa, že příkazová řádka v době grafických nebo webových rozhraní je něco překonaného nebo zbytečného. Důvodů, proč je příkazová řádka stále základní nejdůležitější pomocník správce je mnoho, např.:

- z příkazů můžete vytvořit skript a automatizovat úkoly jako zálohování, kontrola logů ap.
- mnohem rychleji pracujete s klávesnicí, než při přesouvání ruky od myši ke klávesnici a zpátky
- rychlost textového prostředí je mnohonásobně vyšší
- vzdálený přístup z dovolené nebo služební cesty zvládne i nejpomalejší mobilní připojení

Tip

Terminál se po chvilce nečinnosti ztmaví. Jakoukoli klávesou jej opět „rozsvítí“.

3.1. Přihlášení

Po dokončení instalace a restartu se můžeme konečně přihlásit jménem a heslem zvoleným při instalaci.

Důležité

Heslo zadávané při přihlašování se *záměrně při psaní nezobrazuje* podobně jako ve většině případech v Linuxu, kdy máte napsat heslo. To však nemá vliv na editaci - stále můžete používat např. Backspace ap.

Po přihlášení vás Ubuntu uvítá informací o aktuálním čase, stavu počítače a aktualizovatelných balíčcích.

```
Ubuntu 14.04 LTS srv-foo tty1
Hint: Num Lock on

srv-foo login: joe
Password:
Last login: Mon Apr 28 17:32:49 CEST 2014 on tty1
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic i686)

* Documentation:  https://help.ubuntu.com/

System information as of Mon Apr 28 17:32:49 CEST 2014

System load:  0.0                Processes:            68
Usage of /:   10.7% of 6.99GB    Users logged in:    0
Memory usage: 6%                IP address for eth0: 10.0.2.15
Swap usage:   0%

Graph this data and manage this system at:
  https://landscape.canonical.com/

0 packages can be updated.
0 updates are security updates.

joe@srv-foo:~$ _
```

3.2. Klávesové zkratky

Na tomto místě se musíme zastavit a naučit se symboliku zapisování klávesových zkratk a některých speciálních kláves.

Speciální znaky

Klávesa nebo zkratka	Význam
<code>^</code>	<code>Ctrl</code>
<code>^C</code>	<code>Ctrl+C</code>
<code>^C, m</code>	stisk <code>Ctrl+C</code> , uvolnění, vzápětí rychle následované <code>M</code>
<code>M</code>	tzv. Meta-klávesa, na dnešních klávesnicích <code>Alt</code>
<code>M-A</code>	<code>Alt+A</code>
<code>CR</code>	klávesa Return, na dnešních klávesnicích <code>Enter</code>

Klávesa nebo zkratka	Význam
Super	na většině klávesnic odpovídá klávese Win (s logem Windows)
AltGr	pravý Alt

3.3. Speciální znaky

Na tomto místě se musíme zastavit a naučit se číst speciální znaky na které možná nejste zvyklí, ale v příkazové řádce Linuxu mají velmi důležitý význam a často se používají.

Důležité

Následující znaky se naučte bezchybně psát, budete je opravdu používat.

Varování

Zkratky jako Alt+038 (číslo psané na numblocku) ap. v Linuxu nefungují.

Speciální znaky

Znak	Anglický a český název	Česká linuxová klávesnice	Častý význam
#	hash, mřížka	AltGr+X	začátek komentáře
~	tilda, vlnovka	AltGr+Shift+`	domovská složka uživatele
&	ampersand	AltGr+C	operátor AND (A SOUČASNĚ) nebo „poslat do pozadí“
@	at-character, zavináč	AltGr+V	

Znak	Anglický a český název	Česká linuxová klávesnice	Častý význam
^	wedge, stříška	AltGr+6	
\$	dolar	AltGr+; nebo AltGr+4	proměnné prostředí začínají znakem \$
\	backtick, zpětný (obrácený) apostrof	AltGr+`	příkazová uvozovka, řetězec uzavřený v ++ se provede jako příkaz
'	single quotes, jednoduché uvozovky	Alt+'	v řetězci se nerozbalují systémové proměnné
"	double quotes, dvojité uvozovky	Shift+;	v řetězci se rozbalují systémové proměnné
	pipe, svislítko	AltGr+W	operátor roura propojující STDOUT a STDIN dvou programů
>	větší, než	AltGr+>	
<	menší, než	AltGr+<	

~	!	@	~	#	^	\$	~	%	°	^	&	`	*	.	(')	"	-	..	+		⌘	←	
Tab	Q	\	W		E	€	R	T	Y	U	I	O	P	{	÷	}	×								
Caps Lock	A	S	đ	D	Đ	F	[G]	H	J	K	†	L	Ł	:	\$."	ß					Enter	
Shift	Z	X	#	C	&	V	@	B	{	N	}	M	<	<	>	>	?	*					Shift		
Ctrl		Alt											Alt	●									Ctrl		

Tip

Pokud vám česká klávesnice nevyhovuje, můžete si vybrat anglickou nebo jinou příkazem `sudo dpkg-reconfigure keyboard-configuration` (nutné zadat své heslo).

3.4. Bash prompt

V místě blikajícího kurzoru je *prompt* neboli *výzva příkazového řádku*, kde můžeme psát naše příkazy. Tím úplně prvním, co provedeme je „obarvení“ promptu, aby byl přehlednější. Napište následující text (bez počátečního \$ a mezery) a pak odešlete **Enter**:

```
$ nano .bashrc
```

Důležité

Od této chvíle dál bude cokoli, co máte napsat na prompt, začínat znakem dolar. Ten však ale nepíšete - jen reprezentuje, že „zde“ je prompt.

Otevřete soubor `.bashrc` v textovém editoru nano. Tento soubor je jedním z konfiguračních souborů *příkazového procesoru Bash* (též *Bash shell*), jak se správně jmenuje prostředí ve kterém od této chvíle budeme pracovat.

V některých komerčních Unixech nebo BSD můžete narazit i na jiné příkazové procesory jako KSH (Korn SHell), CSH (C SHell) ap. Těmito poněkud exotickými variantami se nebudeme zabývat, protože BASH je de facto standardem všech moderních linuxových distribucí.

Přibližně uprostřed souboru `.bashrc` najdete zakomentovaný řádek začínající znakem mřížka (`#`):

```
#force_color_prompt=yes
```

Odkomentujte řádek (vymažte znak `#`), stiskněte `Ctrl-X` pro odchod z editoru a odpovězte `Y` (Yes) pro uložení změn.

Příkazem `exit` se odhlaste, znovu přihlaste a prompt je nyní barevně rozdělen na části např.

```
joe@srv-foo:~$ _
```


kde

- `joe` je vaše uživatelské jméno
- `srv-foo` název počítače
- `~` aktuální pracovní složka (tilda je domovská složka)
- `$` indikuje, že odtud můžete psát příkazy

3.4.1. Pohyb a ovládání na promptu

Šipky nahoru a dolů

Pomocí kurzorových šipek nahoru a dolů se můžete pohybovat v historii použitých příkazů.

Příkaz history

Příkaz `history` vypíše standardně posledních 50 příkazů. V tomto seznamu se tedy pohybujete šipkami nahoru a dolů.

Klávesová zkratka `Ctrl+C`

Pošle tzv. signál přerušování, který ukončuje aktuální činnost nebo program. Jsou však výjimky, které na tuto kl. zkratku nereagují a ukončují se nejčastěji např. `Q`, příkazem `bye`, `exit` ap.

Klávesová zkratka `Ctrl+D` (EOF)

Znak EOF neboli End of file (konec souboru) se používá v několika málo posledních programech jako `mail` nebo `at` a znamená „ukončuji zadání, teď pracuj ty“.

Klávesová zkratka `Ctrl+Z`

Pošle aktuálnímu programu signál k uspání, tedy pozastavení činnosti. Obnovit program můžete příkazem `fg` (foreground, jdi do popředí). Seznam takto zmražených programů zobrazíte příkazem `jobs`.

Doplňování na `Tab`

Prompt je velmi inteligentní. Napište pár znaků, stiskněte **Tab**, a Bash zkusí doplnit název souboru, složky, programu, a u některých programů dokonce i parametry programu.

Klávesové zkratky **Ctrl+Alt+F1** až **Ctrl+Alt+F7**

Standardně můžeme pracovat až v 7 terminálech současně a mezi nimi pomocí těchto zkratk přepínat. 1. až 6. jsou vždy textové. 7. terminál je GUI, je-li nainstalováno, nebo startovací obrazovka v případě textového systému.

3.5. Parametry programů a příkazů

Než se naučíme několik základních programů a příkazů bez kterých se nedá obejít, bude užitečné se seznámit se symbolickým zápisem parametrů. Naučíte se tak správně číst jaké parametry program nabízí, jak se dají kombinovat, které jsou volitelné atp.

Mezi názvem programu a parametry ovlivňující jeho chování musí být vždy mezera. Třeba předchozí příkaz `nano .bashrc` je volání textového editoru nano s parametrem `.bashrc`.

V manuálových stránkách a dokumentaci programů narazíte na ustálený symbolický zápis parametrů ze kterých vyčteme přesný způsob použití.

Poznámka

Jako „prefix parametrů“ se obvykle nikdy nepoužívá `/` (např. `/h`), ale vždy `-` (např. `-h`) nebo nic. V Linuxu `/` znamená oddělovač složek v cestě.

Podívejme se na několik příkladů:

man [-C file]

Volitelný parametr `-C`, který musíte společně uvést s názvem souboru místo `file`.

find [path...]

Libovolně opakovatelný a současně volitelný parametr `path`.

apropos [-e|-w|-r]

Logická podmínka NEBO je symbolizována svíslítkem. Můžete si vybrat buď jen `-e`, `-w`, `-r`, nebo žádný, protože celá skupina parametrů je volitelná.

-l, --long

Zkrácený a dlouhý název parametru. Můžete si vybrat podobu, kterou si pamatujete nebo vám vyhovuje. Je identické `ls -l` a `ls --long`.

3.6. Manuálové stránky a nápověda

Je pevným pravidlem, že každý program má i svojí manuálovou stránku, kde najdete kompletní dokumentaci použití, parametrů a konfigurace.

Zobrazení manuálové stránky – man

Zobrazení man stránky je velmi jednoduché:

```
man [kapitola] <program | soubor>
```

např. `man nano`. Kapitola se většinou neuvádí.

Prohlížečem manuálu je ve skutečnosti program `less`, proto si nyní jen řekneme, že prohlížeč `less`

- ukončíte stiskem `Q`
- vyhledáváte na stránce stiskem `/`, a zapsáním výrazu do stavové řádky a `Enter`.
- mezi výsledky hledání s posouváte `n` pro vpřed a `N` pro zpět.

Tip

Manuálové stránky mají dokonce i konfigurační soubory. Zajímá vás jakou syntaxi má např. soubor `/etc/fstab`? Napište `man fstab`.

Vyhledávání v manuálových stránkách – apropos

Nemůžete si vzpomenou, jak se některý program jmenuje? Program apropos umí vyhledat zadaný výraz (resp. regulární výraz) v názvech a popisu man stránek.

Např.:

```
apropos passwd
```

najde všechny výskyty slova „find“ v man stránkách a samozřejmě najde i nápovědu pro program jmenující se find:

```
chgpaswd (8)          - update group passwords in batch mode
chpaswd (8)           - update passwords in batch mode
Crypt::PasswdMD5 (3pm) - Provides interoperable MD5-based crypt() functions
fgetpwent_r (3)       - get passwd file entry reentrantly
getpwent_r (3)        - get passwd file entry reentrantly
gpaswd (1)            - administer /etc/group and /etc/gshadow
grub-mkpasswd-pbkdf2 (1) - generate hashed password for GRUB
lppaswd (1)           - add, change, or delete digest passwords.
mkpasswd (1)          - Overfeatured front end to crypt(3)
pam_localuser (8)     - require users to be listed in /etc/passwd
passwd (1)            - change user password <1>
passwd (1ssl)         - compute password hashes
passwd (5)            - the password file <1>
passwd2des (3)        - RFS password encryption
smbpasswd (5)         - The Samba encrypted password file
smbpasswd (8)         - change a user's SMB password
SSL_CTX_set_default_passwd_cb (3ssl) - set passwd callback for encrypted PEM ..
SSL_CTX_set_default_passwd_cb_userdata (3ssl) - set passwd callback for encry..
update-passwd (8)     - safely update /etc/passwd, /etc/shadow and /etc/group
```

Všimněte si čísla v závorce za názvem stránky - např. `passwd (1)` a `passwd (5)`. Manuálové stránky jsou členěny na kapitoly a proto někdy může být stejná stránka v různých kapitolách. Seznam kapitol najdete na `man man`.

Chcete-li tedy např. zjistit informace o příkazu `passwd` z kapitoly 1, použijete `man passwd` nebo `man 1 passwd`. Naopak o stejně pojmenovaném konfiguračním souboru se dozvíte z `man 5 passwd`.

Nápověda pro příkazy – help

Některé programy jsou ve skutečnosti *zabudované příkazy (builtin commands)* Bashe. Patří mezi ně ty nezákladější, jako `cd`, `exit`, `fg`, `jobs`, `echo`, `set` ap. Pro ně neexistuje manuálová stránka, ale trochu jednodušší systém nápovědy `help`:

```
help <zabudovaný-příkaz>
```

např. `help cd` apod.

Tip

Není potřeba vědět, co je program a co příkaz. Zapamatujte si zkrátka, že pokud `man <něco>` neexistuje, zkuste `help <něco>`.

Tip

Pro zvědavé existuje zabudovaný příkaz `type`, který poví, zda je parametr program, příkaz nebo alias. Zkuste si např. `type echo` nebo `type nano`.

3.7. Přihlášení, odhlášení

exit

Příkaz `exit` už znáte. Ukončí vaše běžící programy a odhlásí vás.

logout

`Logout` je podobný, ale neumožní vás odhlásit, běží-li na pozadí nějaké programy.

3.8. Vypnutí a restart PC

`sudo shutdown -h now`

Příkaz shutdown vypíná nebo restartuje PC. Protože tato operace by ovlivnila jiné přihlášené uživatele a může ji provést jen administrátor, musíme celý program předat jako parametr programu sudo.

sudo reboot

Provede restart.

Poznámka

Detailní informace o tomto tématu najdete v [Ukončení práce s PC](#).

3.9. Zobrazení a editace souborů

3.9.1. Editory nano a vim

nano

Pravděpodobně nejjednodušším editorem pro textové prostředí je nano. Jeho název je narážkou na předchůdce program pico. Najdete ho v každé instalaci Ubuntu nastavený jako výchozí editor.

```
GNU nano 2.2.6      File: /home/libor/.bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
  *i*) ;;
  *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
[ Read 119 lines ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text    ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Ovládání:

- uložení – **Ctrl+O**
- hledání – **Ctrl+W**, zadejte výraz, opakujte **Ctrl+W** pro další výskyty
- ukončení – **Ctrl+X**, budete vyzváni k uložení, odpovězte **y** pro ano, **n** pro ne
- jednorázové zobrazení čísla řádku/sloupce – **Alt+C**

Důležité parametry:

- **-c** – zobrazit číslo řádku a sloupce v zápatí obrazovky, zobrazit číslo řádku na začátku nano neumí
- **-\$** – zalamovat dlouhé řádky (wrap). Protože \$ znamená proměnnou shellu, **musíme parametr uvést vždy jako poslední!**

vim a emacs

Mezi další tradiční editory v Linuxu a Unixu patří vim (vi iMproved) a emacs, ale jejich ovládání rozhodně není ani snadné, ani intuitivní.

Zájemce o Emacs odkazujeme internet.

Vim někdy bývá výchozím editorem, proto si řekneme alespoň, jak se vim ukončí. Pustíte-li vim např. **vim .bashrc**, ukončíte ho **:**, **x**, a **Enter**.

3.9.2. Prohlížeč cat a less

cat

Cat je jedním z nejprostších programů vůbec. Umí jen vypsat obsah souboru a skončit:

```
cat <soubor>
```

např. `cat /etc/hostname` vypíše název počítače v tomto souboru.

Užitečnou volbou může být `-n`, `--number` zobrazující u vypisovaných řádků jejich číslo:

```
$ cat -n /etc/hostname
1      srv-foo
```

less

Prohlížeč neboli pager less (méně) je opět slovní hříčkou na starší program more (více). Kdykoli použijete [man stránky](#) čtete si je v programu less. Vyplatí se proto, naučit se, less, výborně ovládat.

Příklad použití:

```
less [parametry] <cesta/k/souboru>
```

Ovládání:

- zalamovat dlouhé řádky – `-`, `Shift+S`, `Enter`
- vyhledávání a skok na první výskyt – `/`, hledaný výraz, `Enter`
- další výskyt hledaného textu – `n`
- předchozí výskyt hledaného textu – `N`
- skok na konec souboru – `Shift+G`

Důležité parametry:

- `-N`, `--LINE-NUMBERS` – zobrazení čísla řádku
- `-S`, `--chop-long-lines` – nezalamovat dlouhé řádky (protože defaultně zalamuje)

3.9.3. Začátky a konce – head a tail

Program head zobrazí standardně prvních 10 řádků souboru, tail posledních 10.

Porovnejte výstupy:

```
$ head .bashrc
$ tail .bashrc
```

Tail má velmi užitečný parametr, který se vyplatí si zapamatovat a to `-f`, kdy tail neskončí a zobrazuje „ocásek“ souboru, tak jak v něm postupně přibývají řádky. Tento parametr je velmi často používán např. pro „živé“ sledování nových záznamů v log souboru ap.

Vyzkoušejte si zajímavý příklad na `tail -f`:

1. Na prvním terminálu spusťte `strings /dev/urandom > ~/random.txt`
2. Chvilku nechte běžet
3. Přepněte se např. na druhý terminál (`Ctrl+Alt+F2`) a napište `tail -f ~/random.txt`
4. Střídejte po chvilkách první a druhý terminál.

Zatím jsme nevysvětlili znaky jako `>`, `~` nebo co je `/dev/random`, ale z příkladu sami jistě odůvodíte, že první příkaz zapisuje náhodné znaky do souboru `random.txt`.

3.10. Vyčištění obrazovky - reset a clear

clear

Clear je obdoba `cls` z MS-DOSu a smaže obsah obrazovky.

reset

„Drsnější“ clear, který kompletně resetuje obrazovku. Vhodné, když se vám terminál tzv. „zbláznil“ a místo znaků zobrazuje „kliky-háky“.

```
libor@libor-Inspiron-5423 ~$ cat /dev/random
0k
XU
) ^
] <ISP"ěušěi | > - C 4 1 {z - X m 43t > G
N = ^ ! *
  5 [ _ { M A
```

3.11. Pohyb na disku – cd, pwd, ls

cd

Příkaz `cd` (*change directory*) asi nebude nutné příliš představovat. Jeho funkcí je změnit aktuální *pracovní složku* (*working directory*).

Pro skok do nadřazeného adresáře slouží `cd` *mezera* a dvě tečky:

```
$ cd ..
```

Varování

Začátečníci často zkouší `cd..` (bez mezery před `..`). To skončí chybou `neexistující program cd..`

Nezáleží na tom, jestli je cesta relativní nebo absolutní:

```
$ cd /home/joe
$ cd ../../var/local
$ cd /etc/init.d/
```

Tip

`cd -` skočí do předcházejícího adresáře.

pwd

Pokud není prompt nakonfigurován zobrazovat aktuální složku jako v Ubuntu, můžete použít příkaz `pwd` neboli *print working directory*..

```
$ pwd
/home/joe/
```

```
joe@srv-foo:~$ _
```

ls

Program `ls` (*list*) vypisuje soubory a podadresáře aktuální nebo zadané složky. Stejný příkaz v MS-DOSu byl `dir`, možnosti `ls` jsou však mnohem větší.

Bez parametrů vypíše `ls` abecedně seřazený obsah ve sloupcích.

Vyzkoušejte a zapamatujte si následující tři klíčové parametry `ls`:

- `-l`, `--long` – dlouhý výpis neboli do tabulky se sloupci oprávnění, vlastník, skupina, velikost a samozřejmě název
- `-a`, `--all` – zobrazení i skrytých souborů (tečkových souborů, dot-files)
- `-h`, `--human-readable` – velikost souboru v násobcích bajtů (např. 1K, 234M, 2G ap.)

Na `ls` je vhodné se naučit se kombinovat parametry. Např. parametr `-h` má smysl jen s `-l`, kdy je zobrazována velikost:

```
$ ls -lh
```

Na pořadí parametrů většinou nezáleží (musíte ale posoudit význam parametrů vždy případ od případu). Pokud chcete zobrazit dlouhý výpis, skryté soubory a „lidské velikosti“ budou následující příkazy stejné:

```
$ ls -lha
$ ls -lah
$ ls -hal
$ ls -hla
$ ls -alh
$ ls -ahl
```

```
$ ls -lha
total 14M
drw-r--r--  6 libor libor 4,0K Dec 26 14:28 .
drw-r--r-- 83 libor libor 4,0K Dec 26 14:28 ..
-rw-r--r--  1 libor libor 1,2M Dec 26 14:28 certificate1.ott
-rw-r--r--  1 libor libor  16M Feb 28 18:40 backup.tar.gz
drwxr-xr-x  3 libor libor 4,0K May 23 12:05 gedit-plugins
-rw-r--r--  1 libor libor  38K Jan 23 10:41 random.txt
```

typ	oprávnění	odkazů	vlastník	skupina	velikost	čas posl. modifikace	název
drw	r--r--	6	libor	libor	4,0K	Dec 26 14:28	.
drw	r--r--	83	libor	libor	4,0K	Dec 26 14:28	..
-rw	r--r--	1	libor	libor	1,2M	Dec 26 14:28	certificate1.ott
-rw	r--r--	1	libor	libor	16M	Feb 28 18:40	backup.tar.gz
drwx	r-xr-x	3	libor	libor	4,0K	May 23 12:05	gedit-plugins
-rw	r--r--	1	libor	libor	38K	Jan 23 10:41	random.txt

Binární předpony

Jednotky, které ls používá při volbě `-h` nejsou kB, MB, GB ap.! Prefixy k, M, G jsou násobky tisíců, kdežto v IT se tradičně používají násobky 1024. Správné označování násobků 1024 je kiB, MiB, GiB ap., které se čtou [kilobí], [megabí], [gigabí] ap. Těmto předponám se říká **binární předpony**. Pokud výslovně potřebujete násobky 1000 (SI násobky), použijte parametr `--si`.

3.12. Vyhledávání - grep

Posledním elementárním programem pro běžnou práci je grep, který umí vyhledávat v obsahu buď standardního vstupu (STDIN) nebo v obsahu souborů.

Poznámka

Vysvětlit grep bez znalostí přesměrování a rour popisovaných v sekci o [přesměrování](#) je velmi obtížné. Proto si text zde přečtete, ale vraťte se k němu po prostudování mechanismu přesměrování.

Hledání v STDIN

Použití bude pro nás až do následující kapitoly trochu záhadné:

```
<příkaz> | grep <hledaný-výraz>
```

znamená, že se výstup STDOUT příkazu pošle (znak roura |) do vstupu STDIN programu grep, který vypíše jen řádky vyhovující hledanému výrazu. Např.:

```
cat /etc/passwd | grep root
```

vypíše řádky v `/etc/passwd` souboru obsahující slovo `root`.

Hledání v obsahu souborů – `grep -r`

Druhé použití grep je pro hledání v obsahu souborů:

```
$ grep -r <výraz>
```

Hledání bez ohledu na velikost písmen – parametr `grep -i`

Obě předchozí funkce jsou skvělé, ale často nám nezáleží na velikosti písmen hledaného výrazu (hledanýVýraz, HledanýVýraz, HLEDANÝVÝRAZ, nebo další kombinace). Parametr `-i`, `--ignore-case` vypíná citlovost na velikost písmen:

```
$ <příkaz> | grep -i <výraz>  
$ grep -ri <výraz>
```

3.13. Přesměrování vstupu a výstupu

Každý program žije zcela izolovaně od ostatních programů ve svém vlastním paměťovém prostoru. Jedinou možností spolupráce (výměny dat) mezi programy je používat zařízení jako je síťová karta, soubory ap. Programy mají však k dispozici ještě tzv. standardní vstup a dva standardní výstupy. Tyto komunikační vstupy/výstupy (V/V, nebo anglicky input/output (I/O)) jako uživatel snadno přesměrujeme jinam nebo navzájem propojíme.

- **standardní vstup (stdin nebo STDIN)** – na STDIN je standardně připojena klávesnice. STDIN můžeme přesměrovat např. na soubor a tak „simulovat“ stisky z klávesnice.
- **standardní výstup (stdout nebo STDOUT)** – první ze dvou výstupů je „běžný“ výstup určený pro ne-chybové hlášky, informace ap. Standardně je STDOUT posílán na obrazovku.
- **standardní chybový výstup (stderr nebo STDERR)** – druhý chybový výstup by měl být určen jen pro reportování chybových hlášek. Standardně je STDERR taktéž posílán na obrazovku.

Nejčastěji přesměrováváme standardní V/V mezi souborem a obrazovkou, ale vzhledem k faktu, že v Linuxu je vše soubor, můžeme přesměrování provést na/z sériového portu, pevného disku ap.

3.13.1. Operátory

Pro ovlivnění standardních V/V slouží tzv. operátory přesměrování. Následující tabulka uvádí ty nejpoužívanější.

Nejdůležitější operátory přesměrování

Operátor	Směr	Funkce
> nebo 1>	STDOUT → soubor	Přesměrování STDOUT a vytvoření/přepsání existujícího souboru
>>	STDOUT → soubor	Přesměrování STDOUT a vytvoření/připojení na konec souboru
<	soubor → STDIN	Přesměrování STDIN z klávesnice na soubor
2>	STDERR → soubor	Přesměrování STDERR do souboru
2>&1 nebo &>	STDERR → STDOUT	Přesměrování STDERR na STDOUT
1>&2	STDOUT → STDERR	Přesměrování STDOUT na STDERR
	STDOUT → STDIN	Přesměrování STDOUT na STDIN následujícího programu
2>&1	STDOUT+STDERR → STDIN	Spojí STDOUT a STDERR a přesměruje na STDIN následujícího programu

3.13.2. Příklady na přesměrování

Přesměrovat, přepsat

Přesměrování jsme již viděli ve [starsším příkladu](#), který nyní dovedeme vysvětlit:

```
$ strings /dev/urandom > random.txt
```

Program strings je vhodný hlavně pro vývojáře. Hledá v binárních souborech tisknutelné znaky. Speciální soubor zařízení (device file) `/dev/urandom` obsahující nekonečně dlouhou sekvenci náhodných čísel je takto filtrován jen na tisknutelné znaky. Výstup STDOUT, jinak směřující na obrazovku, je přesměrován operátorem `>` do souboru `random.txt`.

Připojit, nepřepsat

Změnou z `>` na `>>` dosáhneme, že je obsah k souboru připojen (append), nikoli přepsán:

```
$ cat /dev/random >> random.txt
```

Přesměrování STDERR

Přesměrovat pouze chybový výstup můžeme s `2>`:

```
$ grep -blah 2> stderr.txt
$ cat stderr.txt
Usage: grep [OPTION]... PATTERN [FILE]...
Try 'grep --help' for more information.
```

Spojení STDERR a STDOUT

Často chceme uchovat běžný výstup i ten chybový v jediném souboru. Tradiční a trochu krkolomné vyjádření je s `2>&1`. Nejprve přesměrujeme STDOUT programu do souboru a na závěr STDERR programu napojíme na STDOUT, který byl již přesměrován do souboru:

```
$ program > vystup.log 2>&1
```

Jiným a přehlednějším způsobem, jak spojit STDERR a STDOUT a přesměrovat do souboru je `&>`:

```
$ program &> vystup.log
```

Roura

Roura neboli znak | (svislítko, pipe) kombinuje předchozí operátory přesměrování STDIN < a STDOUT >. Spojuje STDOUT na STDOUT následujícího programu napřímo bez nutnosti použití souboru jako „mezičlánek“.

Tento druh přesměrování jsme také již viděli použitý v ukázce na [grep -i](#), kdy jsme propojili STDOUT příkazu (běžně napojený na obrazovku) na STDIN grepu (běžně napojený na klávesnici):

```
$ <příkaz> | grep -i <výraz>
```

Musíme držet na paměti, že roura spojuje STDOUT na STDIN následujícího programu. **Výstup na STDERR prvního programu do roury nevstupuje.**

Program ls můžeme požádat o výpis více souborů/složek zadaných jako parametry - např. /home/ a /var/, ale u druhé složky uděláme překlep v názvu. Ls vypíše obsah první složky na STDOUT, ale druhá neexistuje a chybu vypíše na STDERR. Např. cat napojený na rouru proto nikdy chybu neobdrží a čísluje jen získaný STDOUT:

```
$ ls /home/ /war/ | cat -n
ls: cannot access /war: No such file or directory
1 /home:
2 jell
3 lost+found
```

Aby jsme do roury poslali STDOUT i STDERR, musíme použít další operátor přesměrování spojující tyto dva proudy:

```
$ ls /home /war/ 2>&1 | cat -n
1 ls: cannot access /war/: No such file or directory
2 /home:
3 jell
4 lost+found
```

Kombinace operátorů

Operátory (nejčastěji rouru) na sebe můžeme vzájemně napojovat. Např.:

```
$ apropos find | grep -i path | cat -n
1 glob (3) - find pathnames matching a pattern, free memory f
2 globfree (3) - find pathnames matching a pattern, free memory f
3 XtFindFile (3) - search for a file using substitutions in the pat
```


Apropos vyhledá „find“ v manuálových stránkách. Výstup je poslán grepu, který vyfiltruje jen řádky se slovem „path“. I jeho výstup je předán cat číslující řádky.

3.14. Proměnné prostředí

Jak jsme již několikrát zmínili, Bash je ve skutečnosti docela propracovaný programovací jazyk. Pro běžnou práci na příkazové řádce Bashe to skoro nepoznáme, kromě proměnných prostředí, která se podobají proměnným v běžných programovacích jazycích.

Proměnná prostředí (environment variable) umožňují na číselnou nebo textovou hodnotu odkazovat jménem proměnné. Přítomnost nebo hodnota proměnné také může sloužit ke konfiguraci programu podobně, jako parametry.

Nastavení a zrušení

Proměnnou nastavíte jednoduše:

```
$ jmeno=Joe
$ vek=29
```

Proměnné začínají \$ a rozbalují se (expandují se) na hodnoty a můžete vytvářet kombinace jako:

```
$ dohromady="$jmeno je $vek let stary"
```

Alternativní syntaxe `${<proměnná>}` je vhodná, když by měl Bash problém rozlišit, kde začíná nebo končí název proměnné. Kdybychom chtěli vypsát „Joe je 29letý“, nemůžeme napsat

```
$ dohromady="$jmeno je $vekletý"
```

protože Bash bude hledat neexistující proměnnou `$vekletý`. Správně tedy bude:

```
$ dohromady="$jmeno je ${vek}letý"
```

Aby proměnnou prostředí viděl nejen interpret Bash sám, ale i programy interpretem spuštěné, musíme proměnnou exportovat:

```
# Dříve vytvořená proměnná
$ export dohromady

# Vytvoření a export v jednom kroku
$ export mesto=Praha
```

Jestli proměnná nebude již potřeba, pak můžeme jen nastavit „prázdnou“ hodnotu nebo ji úplně zrušit (nebude již ve výpisu proměnných `env`):

```
$ mesto=
$ unset mesto
```

Vypsání

Hodnotu můžeme vypsát příkazem `echo`:

```
$ echo $dohromady
Joe je 29 let stary
```

Nebo vypsát všechny pomocí `env` (výstup bývá na několik obrazovek proto ještě `less`):

```
$ env | less
```

Nejdůležitější proměnné prostředí

- `HOME` – Absolutní cesta k domovské složce aktuálního uživatele. Např. `/home/jekyll`.
- `USER` – Uživatelské jméno aktuálního uživatele. Např. `jekyll`.
- `HOSTNAME` – Jméno počítače. Např. `nb-jekyll`.
- `PATH` – Vyhledávací cesta (viz dále).

Vyhledávací cesta `PATH`

Proměnná `PATH` neboli *vyhledávací cesta* je seznam složek, kde bude Bash hledat programy. Pokud program v žádné složce tohoto seznamu nenajde, uvidíte `<program>: command not found`.

V čerstvé instalaci Ubuntu bude `PATH` obsahovat přibližně tyto dvojtečkou oddělené složky:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/lo
```

Všimněte si, že `PATH` neobsahuje aktuální složku (`.`), tj. i když je v aktuální složce spuštěný program, musíte se na něj explicitně odkázat `./<program>`:

```
$ ls
program-raz-dva  soubor.txt  ...
$ program-raz-dva
program-raz-dva: command not found
$ ./program-raz-dva
#program-raz-dva spuštěn
```

Výchozí cesta je nastavena startovacími skripty během bootu OS. Můžete ji ale kdykoli později modifikovat. Přidání na konec `PATH`:

```
$ PATH=$PATH:/moje/cesta
```

nebo na začátek:

```
$ PATH=/moje/cesta:$PATH
```

Nyní tedy víme, proč spustíme program nano odkudkoli na disku. Ve které složce ve vyhledávací cestě se však nachází, zjistíme pomocí `which` (který):

```
$ which nano
/usr/bin/nano
```

3.15. Tři druhy uvozovek

Viděli jsme, jak dochází k expanzi proměnných prostředí na jejich hodnoty. To však platí jen při neuvedení uvozovek nebo při dvojitéch uvozovkách.

- `"` (dvojitě uvozovky) – Řetězec mezi dvojitými uvozovkami se prohledává na proměnné, které se rozbalí na hodnotu.
- `'` (jednoduché uvozovky) – Řetězec mezi jednoduchými uvozovkami je interpretován tak, jak je - tzn. nedochází k expanzi proměnných a ignorují se speciální znaky jako `\n`, `\t` ap.
- `\`` (obrácený apostrof) – Řetězec mezi obrácenými apostrofy se provede jako příkaz. Jeho výstup se stane výslednou hodnotou řetězce.

Neuvést uvozovky nebo uvést dvojitě je stejné. Následující zápisy mají tedy stejný efekt:

```
$ echo $PATH
$ echo "$PATH"
```

Předchozí příklady napsané v jednoduchých uvozovkách tedy nevypíší hodnotu proměnné, ale vytisknou se tak, jak jsou:

```
$ echo '$PATH'
$PATH
```

Obrácené apostrofy jsou velmi zajímavou funkcí Bashe. Např. pro vypsání dnešního data můžete použít:

```
$ echo Ahoj $USER! Dnes je `date`.
Ahoj lisa! Dnes je St dub 30 13:51:40 CEST 2014.
```

3.16. Tilda ~

Tildu bychom mohli zařadit mezi proměnnou prostředí. Chová se jako proměnná i když jí technicky není. Její hodnotou je absolutní cesta k domovskému adresáři. Proto jsou následující příkazy identické:

```
$ cd $HOME
$ cd ~
```

Stejně jako tato dvojice:

```
$ cp /var/log/kern.log $HOME/tmp/
$ cp /var/log/kern.log ~/tmp/
```

3.17. Obrácené lomítko (backslash)

Znak (obrácené lomítko) má tři odlišné významy, které se musíme naučit.

Důležité

Obrácené lomítko (`\`) nikdy neslouží jako oddělovač cesty, kterým je v Linuxu vždy běžné lomítko (`/`) (forward slash).

Řídící nebo formátovací znaky

Kdykoli chcete vypsát třeba znak „tabelátor“ nebo „nový řádek“ použijte `\t`, resp. `\n`:

```
$ echo -e "ahoj\tjak\nse\tmas"
ahoj      jak
se        mas
```

Další méně používané řídicí znaky najdete např. v manuálové stránce programu `echo` (`man echo`).

Pokračování příkazu na další řádce

Velmi dlouhé příkazy na příkazové řádce nebo skriptu můžeme pro přehlednost rozdělit na více řádků pomocí `\`:

```
$ echo \
    velmi dlouhý \
    příkaz \
    na více řádků
velmi dlouhý příkaz na více řádků
```

Neinterpretovat znak

Některé znaky, jako např. právě zpětné lomítko, mají speciální význam pro vyhledávání (žolíky) nebo proměnné prostředí (`$`). Když ale opravdu chceme jen vypsát znaky jako `~` nebo `$` musíme je tzv. *escapovat* (napsat *escape sekvenci*) přidáním před znak:

```
$ echo Znaky se speciálním významem jsou např. \~, \\, \$
Znaky se speciálním významem jsou např. ~, \, $
```

3.18. Žolíky (wildcards)

Poslední základní dovedností jsou tzv. *žolíky* (*wildcards*, též *globbing patterns*) umožňující postihnout skupinu souborů nebo složek vyhovující určitému pravidlu.

Nejdůležitější žolíky v Bashi

Znaky	Popis	Příklad
*	jakýkoli počet znaků (včetně žádného)	Jinými slovy nula nebo znaků. Pro „f*d“ bude vyhovovat „find“, „fond“, ale i jen „fd“
?	jakýkoli <i>jeden</i> znak	Pro „f?nd“ bude vyhovovat „find“, „fInd“, „fond“, ap.
[]	rozsah	Pro „hd[a-e]“ bude vyhovovat „hda“, „hdb“, „hdc“, a „hde“.
[!]	vyloučit z rozsahu	Podobné jako [], ale slouží k vyloučení znaků v hranatých závorkách z vyhledávání. Pro „mujsoubor[!9]“ bude vyhovovat „mujsoubor1“, „mujsoubor2“, „mujsoubor3“ atd., ale nebude vyhovovat „mujsoubor9“.
{ }	výčet	Pro „{mama,tata}“ bude vyhovovat „mama“ nebo „tata“.
\	escape znak	Chceme-li hledat znak , musíme ho „ochránit“ zdvojením na .

Např. vypsát všechny soubory a složky začínající textem „pa“ ve složce `/etc/`:

```
$ ls /etc/pa*
```

Žolíky lze opakovat:

```
$ ls /dev/sd[a-z][0-9]
```

Rovněž lze žolíky kombinovat. Třeba, pokud chcete smazat všechny jpg, png a pdf soubory:

```
$ rm {*.jpg,*.png,*.doc}
```

4. Uživatelé a skupiny

4.1. Superuživatel root

Neomezenou mocí vládne superuživatel s uživatelským jménem root. Je to obdoba účtu „admin“, „Administrator“ nebo „su“ v jiných OS a softwarech.

Upozornění

Možná by vás napadlo v rámci zvýšení bezpečnosti nebo „přizpůsobení“ přejmenovat účet root na „admin“ nebo vytvořit nového superuživatele. To je sice technicky proveditelné, ale rozhodně vám to nedoporučujeme. Vytvoříte spíše nové zranitelnosti a kromě toho vás bude proklínat každý, kdo po vás bude muset server spravovat.

Root smí provádět jakoukoli operaci s jakýmkoli souborem nebo procesem kromě nesmyslných operací, jako třeba spuštění souboru bez spustitelného bitu.

Důležité

Chtěli bychom vyzdvihnout slova *jakoukoli* a *jakýkoli*. Root v Linuxu je skutečně neomezený. Ne jako ve Windows, kdy jste sice administrátor, ale přesto nemůžete úplně svobodně některé operace se systémem provádět a musíte využívat různé triky přes Shift, runas ap.

Ovšem i přes svoji privilegovanost je to pořád účet jako každý jiný a mohli byste se chtít na něj přihlásit a pracovat. Z důvodu naprosté neomezenosti tohoto účtu je to však nerozumné. Představte si, že se překlepnete a protože jste root, smažete důležité soubory nebo znefunkčíte systém. Některé distribuce jako Ubuntu vám to dokonce vůbec neumožní.

4.1.1. su – změna identity

Ve všech systémech kromě Ubuntu je výše uvedený problém vyřešen programem `su`. Když ho spustíme bez parametrů

```
$ su
```

vyzve nás k zadání hesla roota a spustí shell (příkazovou řádku) s oprávněními roota.

Upozornění

V Debianu se na účet roota přepnout můžete, ale i tak to nedoporučujeme.

Můžeme přepínat i na jiné identity, než roota.

```
$ su <uživatel>
```

a zadáte heslo tohoto uživatele. Nebo se můžeme přepnout nejprve na roota `su` a pak na konkrétního uživatele `su <uživatel>` a nebudeme muset zadávat jeho heslo (root se může přepnout i bez zadávání hesla daného uživatele).

4.1.2. sudo – vylepšený su v Ubuntu

Jak jsme si řekli, v Ubuntu se na roota nepřihlásíte (alespoň ve výchozím stavu). To ale neznamená, že Ubuntu roota nemá. Jak tedy v Ubuntu provést úkoly vyžadující superuživatele?

Ubuntu uživatel root sice existuje, ale nemá žádné platné heslo. Tedy se na něj nikdy nemůžeme přihlásit či přepnout. Namísto „přepínání“ se na roota (nebo na jiný účet) si jen „vypůjčíme“ oprávnění tohoto účtu. Sudo je velmi bohatě konfigurovatelné v souboru `/etc/sudoers`, kde najdete

- seznam uživatelů, kteří smí sudo používat
- jaké programy smí spouštět (dokonce i jaké parametry těmto programům mohou zadat)
- na jakých hostitelích smí tyto programy provádět

- pod jakým účtem se mají tyto příkazy provádět (nejčastěji root)

Upozornění

Příkaz sudo nás žádá o naše vlastní heslo, nikoli roota (resp. uživatele pod kterým se má program provést).

Výhody řešení přes sudo:

- existuje log, kdo, kdy a jaký příkaz spustil
- rootovské úkoly můžou provádět i uživatelé bez root oprávnění
- root oprávnění můžete uživateli odebrat bez změny root hesla
- skutečné heslo roota zná jen jeden či dva „vyvolení“

4.1.2.1. sudo -i

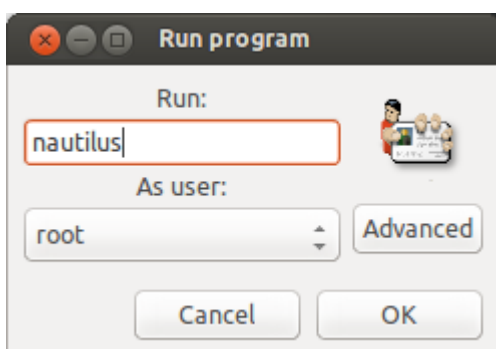
Tímto parametrem otevřeme root konzoli, abysme nemuseli stále opakovat před každým příkazem „sudo“. Nevýhodou však je, že příkazy zadané v rámci nové konzole nejsou zalogovány.

4.1.2.2. gksudo

Sudo se hodí pro spuštění s jinými oprávněními pouze pro textové (CLI) programy. Když chcete spustit jako root (nebo jiný uživatel) grafickou aplikaci (GUI), je vhodné použít variantu gksudo:

```
$ gksudo <program>
```

Nebo bez parametru `gksudo` můžete příkaz i identitu vybrat graficky.



Gksudo standardní součástí Ubuntu od verze 13.04. Doinstalovat gksudo můžeme příkazem:

```
$ sudo apt-get install gksu
```

Poznámka

Jestli náhodou používáte Kubuntu, tedy Ubuntu s KDE místo Gnome/Unity, pak se příkaz jmenuje `kdesu`.

4.1.2.3. Protokol `/var/log/auth.log`

V tomto souboru najdeme zmíněnou historii příkazů provedených prostřednictvím `sudo/gksudo`.

Zobrazit např. posledních 10 záznamů můžeme pomocí:

```
$ tail /var/log/auth.log
```

[Více o tail.](#)

4.1.2.4. Nevýhody `sudo`

Nevýhoda `sudo` přístupu je, že prolomení bezpečnosti „běžného“ účtu s oprávněním provádět `sudo` může mít stejný dopad jako prolomení účtu samotného `roota`. Dělat se s ním nedá nic kromě navádění uživatelů `sudo` k ochraně účtu stejně jako by se jednalo o účet superuživatele.

Druhou nevýhodou může být obtížné [přesměrování výstupu](#) výstupu „sudovaného“ programu pomocí operátorů např. `>`, `>>`, `|`. Např. když zkusíte vymazat `error.log` pomocí „černé díry“ `/dev/null`:

```
$ sudo cat /dev/null > /var/log/apache2/error.log
```

narazíme na *Permission denied*. Ale jak to, když se má příkaz provádět s právy `roota`?! Důvod je v tom, že se jen část před operátor `>` provede jako `root`. Přesměrování

totiž spustí nový subshell, který již není „sudován“. Celý příkaz musíme proto přepsat, aby se spustil naráz:

```
$ sudo sh -c 'cat /dev/null > /var/log/apache2/error.log'
```

4.1.2.5. Přidání do sudoers

Pro přidání mezi „sudoery“ stačí uživatele přidat do skupiny sudo:

```
$ sudo adduser <uživatel> sudo
```

Protože je členství ve skupině zjišťováno jen při přihlašování, musí se uživatel odhlásit a přihlásit znovu, aby se oprávnění projevilo.

Tento postup místo přímé editace souboru `/etc/sudoers` funguje díky tomu, že je v tomto souboru již nastaveno, že všichni uživatelé skupiny sudo mohou provádět jakýkoli příkaz a na jakémkoli počítači.

Maximální povolení pro skupinu sudo v /etc/sudoers

```
# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL
```

Pro jakoukoli jinou konfiguraci, než jen pouhé přidání mezi sudoery, je třeba editovat soubor `/etc/sudoers`. Nikdy to ale nedělejte na přímo otevřením v editoru, ale pomocí programu `visudo`. Ten otevře konfigurační soubor v textovém editoru a zabezpečí, že jste jediný, kdo ho bude v daný moment upravovat, a zejm. při ukončení editoru zkontroluje správnost syntaxe.

4.2. Uživatelé

Ihned po dokončení instalace existují v systému dva účty „pro lidi“ a dokonce až několik desítek účtů pro démony (služby běžící na pozadí) (v závilosti na vybraném software během instalace).

Sami jistě správně odhadnete, že prvním lidským účtem je root (se kterým nemůžeme běžně pracovat) a váš vlastní účet pro běžnou práci vytvořený během instalace.

Na další účty sloužící démonům se nelze přihlásit. Mají velmi omezená práva specifická pro jakého démona mají sloužit. V Linuxu běžně každý démon běží pod přísně vyhrazeným a omezeným účtem.

Důležité

Pamatujte, že téměř jakékoli změny uživatele, skupiny, členství ve skupinách a z nich vyplývajících oprávnění se projeví až při opětovném přihlášení.

4.2.1. Soubory `/etc/passwd` a `/etc/shadow`

Soubor `/etc/passwd` je klíčovým konfiguračním souborem pro správu uživatelů. Obsahem je seznam uživatelů systému:

- uživatelské jméno (username),
- heslo,
- user ID (UID) číslo,
- group ID (GID) číslo skupiny uživatele,
- dodatečné údaje (občanské jméno, číslo kanceláře, ap.),
- přihlašovací shell

Tip

UID 0 je rezervování pro roota. UID 1-999 by měly být vyhrazeny pro démony. Teprve od UID 1000 by se mělo jedna o „lidské“ účty.

Jednotlivé záznamy na řádcích jsou oddělené dvojtečkou.

Ukázka `/etc/passwd` (zkráceno)

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
```

```
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
...
joe:x:1000:1000:Joe Smith,,,:/home/joe:/bin/bash
lisa:x:1001:1001:Lisa Simpson,,,:/home/lisa:/bin/bash
```

Možná jste zvedli obočí při informaci, že se v tomto souboru nachází heslo. To je a není pravda. V dřevních dobách Unixu, zde opravdu bylo heslo v čitelné podobě, ale později začalo být ukládáno zašifrovaně.

Stínová hesla

V naší ukázce mají všichni uživatelé v poli hesla „x“, což znamená, že jsou používána tzv. *stínová hesla (shadow passwords)*, kdy je heslo uloženo v odděleném souboru `/etc/shadow` [1].

Na rozdíl od `/etc/passwd` není tento soubor veřejně dostupný:

```
$ ls -l /etc/passwd /etc/shadow
-rw-r--r-- 1 root root 1989 dub 14 19:53 /etc/passwd
-rw-r----- 1 root shadow 1325 dub 14 19:53 /etc/shadow
```

Ukázka `/etc/shadow` (zkráceno)

```
root:!:16106:0:99999:7:::
daemon*:15994:0:99999:7:::
bin*:15994:0:99999:7:::
sys*:15994:0:99999:7:::
sync*:15994:0:99999:7:::
games*:15994:0:99999:7:::
man*:15994:0:99999:7:::
lp*:15994:0:99999:7:::
mail*:15994:0:99999:7:::
news*:15994:0:99999:7:::
...
joe:$6$xhD/uG2q2dcqDG5M$2J3kIis1IU9PXaFI7WRhBBKRfEgFc2ERP.kv3LE0sTzxx/NK9ANAjn5
lisa:$6$ZR/Ld5EzJMaEpP8LA$h8FWBii.Zz/ZRWnwnCWg.BsvxpPZS0Z2fQj7DS9LAW0QyuN8TCYLF
```

Heslo uživatele je zde uloženo jako *hash (otisk, digest)*, tedy z hash řetězce nelze nijak odvodit jaké bylo původní heslo. Hash je znovu vypočten při přihlašování a porovnán s hodnotou v `/etc/shadow`.

Poznámka

Konkrétní hešovací algoritmus se liší. Tradiční DES a MD5 postupně nahrazuje např. bezpečnější SHA-512, které používá i Ubuntu.

Pokud zde není hash, ale `!` nebo `*`, pak je znemožněno se tímto účtem přihlásit. Vy-
křičník před hashem znamená zamčený účet (taktéž se není možné přihlásit).

Je-li zde prázdný řetězec, považuje se to za prázdné heslo (bez heslo), ale ne
všechny aplikace jsou na variantu prázdného hesla připraveny.

Další pole souboru `/etc/shadow` jsou (rovněž oddělené dvojtečkou):

- datum poslední změny hesla
- minimální počet dní mezi změnami hesla
- maximální počet dní mezi změnami hesla
- počet dní předem varování, že si musíme heslo změnit
- počet dní po vypršení hesla, po kterých bude účet zablokován
- datum vypršení hesla
- poslední pole je rezervováno pro budoucí využití

Datумы se uvádějí jako počet dní od 1. ledna 1970.

4.2.2. Vytvoření uživatele – `adduser`

Na základě předchozích znalostí bysme uměli umět založit uživatele manuálně, ale
jistější a pohodlnější způsob je použít `adduser <uživatel>`, postupně odpově-
dět na otázky programu a nakonec potvrdit správnost `Y` (Yes):

```
$ sudo adduser lisa
Adding user `lisa' ...
Adding new group `lisa' (1002) ...
Adding new user `lisa' (1002) with group `lisa' ...
Creating home directory `/home/lisa' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
Changing the user information for lisa
Enter the new value, or press ENTER for the default
Full Name []:
Room Number []:
Work Phone []:
Home Phone []:
```

```
Other []:  
Is the information correct? [Y/n] y
```

Důležité

V jiných systémech stejnou práci obstarává `useradd`. I když tento skript také existuje v Ubuntu/Debian, preferujeme vždy `adduser`.

4.2.3. Přejmenování uživatele

Když dojde k nutnosti přejmenovat uživatele (např. z důvodu svatby) použijeme jednu z mnoha funkcí programu `usermod`:

```
$ sudo usermod -l <nový-username> <současný-username>
```

Pozor na to, že domovská složka zůstane stále stejná. Musíme ji přejmenovat ručně a upravit v `/etc/passwd`.

4.2.4. Zjištění identity uživatele – `id`, `whoami`

Aktuálního uživatele a jeho skupiny ve kterých je členem zjistíte pomocí `id`.

Pouze jméno aktuálního uživatele zjistíte pomocí `whoami`.

4.2.5. Vymazání uživatele – `deluser`

Smazání je opět možné provést ručně, ale rychlejším způsobem je využít `deluser <uživatel>`:

```
$ sudo deluser lisa
```

Skript vymaže uživatele (z `/etc/passwd` a `/etc/shadow`), ale **nevymaže domovskou složku, mail pool a soubory vlastněné uživatelem**.. Připojte parametr

- `--remove-home` pro odstranění domovské složky
- `--remove-all-files` pro odstranění všech souborů vlastněných uživatelem (tedy i domovské složky)

Soubory uživatele je před odstraněním vhodné zazálohovat do [zkomprimovaného archívu](#) parametrem `--backup`, který vytvoří v aktuální složce soubor `<uživatelské_jméno>.tar.gz`.

Důležité

V jiných systémech stejnou práci zařizuje `userdel`. I když tento skript existuje také v Ubuntu/Debian, zde preferujeme vždy `deluser`.

Smazání uživatele a SSH přístup

Vymazání uživatele nebo jen jeho uzamčení se nevztahuje na další způsoby přihlašování, zejm. SSH! Pokud má uživatel přístup přes SSH s veřejným klíčem, může se nadále přihlásit. Proto

1. smažeme jeho domovskou složku nebo jen soubor `~/.ssh/authorized_keys`.
2. ukončíme všechny existující SSH spojení daného uživatele.

Lepším řešením je však omezit přístup přes SSH na skupinu např. `sshlogin`, přidat tuto skupinu pomocí `AllowGroups` v `/etc/ssh/sshd_config`. Následně postačí jen uživatele ze skupiny `sshlogin` odebrat a restartovat SSH démona neboli:

```
$ sudo adduser <username> sshlogin
$ sudo service ssh restart
```

4.3. Skupiny

4.3.1. Soubor `/etc/group`

Obsahem souboru `/etc/group` je seznam skupin a seznam členů těchto skupin.

Jednotlivá pole oddělená dvojtečkou mají tento význam:

- název skupiny
- heslo skupiny

- group ID (GID) číslo skupiny
- členové skupiny oddělené čárkou

Varování

Členové skupiny jsou odděleni opravdu jen čárkou („joe,lisa“), nikoli čárkou a mezerou („joe, lisa“)!

Ukázka /etc/group (zkráceno)

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:joe
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:joe,lisa
news:x:9:
...
joe:x:1000
lisa:x:1001:
```

Poznámka

Skupina může mít heslo (když se přepínáme na skupinu programem `newgrp`), ale využití této možnosti v praxi a tím pádem i hesla skupiny je téměř nulové. Pro hesla skupin existuje mechanismus podobný stínovým heslům v souboru `/etc/gshadow`, kde je většinou nastaveno `*`, tj. „nelze se přihlásit“.

4.3.2. Primární a sekundární členství

Možná jste si při kontrole souborů `/etc/passwd` a `/etc/group` po vytvoření uživatele povšimli, že zde byla vytvořena skupina se stejným názvem, jako je název uživatele. Tato skupina se jmenuje *primární (či osobní) skupina* a právě založený uživatel je jejím jediným členem.

Uživatel má svou primární skupinu uvedenou přímo v `/etc/passwd` jako číselné GID (může zde být uvedena jen jediná skupina pomocí GID). Čerstvě po vytvoření uživatele nemá tato skupina žádné další členy v `/etc/groups`.

Naproti tomu členové uvedení v `/etc/groups` mohou nazývat takovou skupinu jako *sekundární* (*secondary* nebo *supplementary*).

`/etc/passwd:`

```
...  
lisa:x:1002:1002:Lisa Simpson,,,:/home/lisa:/bin/bash  
...
```

`/etc/group:`

```
...  
lisa:x:1002:  
marketing:x:1003:joe,lisa,mark,tom  
...
```

GID

Primární a sekundární skupina z pohledu konfiguračních souborů.

Poznámka

Z pohledu oprávnění a jakýchkoli dalších efektů jsou oba druhy členství prakticky rovnocenné. Jediný rozdíl je, že nově vytvářené soubory a složky dostávají přiřazenou vaši primární skupinu jako skupinu souboru.

4.3.3. Vytvoření skupiny

Pro vytvoření skupiny slouží příkaz `addgroup <skupina>`:

```
$ sudo addgroup marketing
```

4.3.4. Přidání uživatele do skupiny

```
$ sudo adduser <username> <skupina>  
$ sudo adduser lisa marketing
```

4.3.5. Výpis členství ve skupinách

Aktuálního uživatele:

```
$ groups
joe adm mail sudo lpadmin sambashare marketing
```

Jiného uživatele:

```
$ groups lisa
joe adm mail sudo lpadmin sambashare marketing management
```

4.3.6. Vymazání skupiny

```
$ sudo deluser --group marketing
```

nebo identický výsledek bude mít

```
$ sudo delgroup marketing
```

Varování

Nelze odstranit primární skupinu existujícího uživatele!

4.3.7. Odstranění uživatele ze skupiny

```
$ sudo deluser <uživatel> <skupina>
$ sudo deluser lisa management
```

4.4. Externí databáze - LDAP, NIS

Všechny doposud popsané informace platily pouze lokálně, tj. účty a skupiny existují výhradně na místním počítači a není způsob, jak by mohl stejný účet nebo skupina existovat či se ověřovat se stejným heslem a sdílet oprávnění na více počítačích stejné sítě.

K dosažení skutečně sdíleného účtování je potřeba začít používat systémy jako LDAP nebo NIS, které přesouvají konfiguraci na externí centrální databáze a jednotlivé stroje jsou klienty tohoto centrálního serveru.

Vzhledem ke značné rozsáhlosti a faktu, že většina studentů bude spravovat maximálně jednotky instalací se nebudeme LDAP a NIS zabývat.

4.5. Správa hesel

4.5.1. Nastavení hesla

Pomocí programu passwd si uživatel sám sobě nebo správce uživateli může nastavit heslo. Pokud odpovídá politice hesel (viz dále), bude přijato.

Uživatel sám sobě:

```
$ passwd
```

Správce jinému uživateli:

```
$ sudo passwd <uživatel>
```

4.5.2. Zamčení účtu

S hesly souvisí i možnost uzamknout účet. Je to mírnější varianta vymazání účtu. Veškeré soubory, nastavení a oprávnění zůstávají, ale uživatel se ke svému účtu nemůže přihlásit.

Zamčení `-l` (lock):

```
$ sudo passwd -l <účet>
```

Odemčení `-u` (unlock):

```
$ sudo passwd -u <účet>
```

4.5.3. Politika hesel

Správná hesla by měla odpovídat zabezpečenému systému. Můžete proto vyžadovat určitou minimální délku, složitost, neopakování hesel, nebo časově omezená hesla (expiry).

PAM

Toto všechno je možné pomocí tzv. PAM (PAssword Management) modulu, který je konfigurován pomocí souborů v `/etc/pam.d/` složce. Zmíněná minimální délka hesla se např. nastavuje v souboru `/etc/pam.d/common-password`.

Expirace hesla

Aktuální dobu do vypršení hesla zjistíte `sudo chage -l <uživatel>`. Bez parametru `sudo chage <uživatel>` budete postupně dotazováni na jednotlivá nastavení..

Poznámky

[1] Není nám známá distribuce, která by používala starý systém ukládání hesel.

5. Složky a soubory

Nejprve si projdeme tím, jak vypadá organizace linuxového disku, nejdůležitější složky, soubory a jaký je jejich význam. Ve druhé, veskrze praktické části, budeme soubory a složky vytvářet, mazat, archivovat, přesouvat ap.

5.1. Hlavní odlišnosti Linuxu

Podívejme se na ty nejzásadnější odlišnosti souborů v Linuxu.

Skryté soubory

Linux nezná skryté soubory. Existuje pouze konvence, že soubory nebo složky s tečkou na začátku (např. již dobře známý `.bashrc`) jsou považovány za skryté a většina programů je standardně při práci se soubory nezobrazuje. Těmto „skrytým“ souborům se proto říká *dot-files (tečkové soubory)*.

Přípona neurčuje typ

Ve většině případů je přípona považována jen za součást názvu souboru a neovlivňuje, jak se se souborem bude zacházet. Typ je určen na základě obsahu, nikoli jména. Pro většinu typů souborů se zkontroluje několik prvních bajtů (hlavička souboru). Tomuto mechanismu se říká *magic pattern* a sami můžete zkusit, jak dobře tato detekce funguje příkazem `file`.

Neexistují jednotky

I když je souborový systém rozdělen na více disků nebo oddílů, z pohledu uživatele existuje jediná stromově tvořená hierarchie. Neexistují tedy jednotky, resp. písmena jednotek. Disky či oddíly jsou namapovány přímo na určitou složku.

Rozlišuje se velikost písmen

Toto je asi nejznámější vlastnost i mezi lidmi, kteří se s Linuxem doposud nesešli. Soubor `dovolená.jpg` je zcela jiným souborem, než `dovolená.JPG`. Obecně se dá říct, že v Linuxu se velikost písmen rozlišuje nejen u názvu souborů a složek, ale v podstatě všude jako např. v konfiguračních hodnotách, klíčích, skriptech apod.

Oddělovač cesty je /

Jako oddělovač diskové cesty slouží běžné lomítko / (dopředné lomítko, forward slash), nikoli \ (zpětné lomítko, backward slash).

Složka je druh souboru

Mluvíme-li proto o souboru, informace vždy platí i pro složky. Podobně mohou mít i složky přípony (např. /etc/init.d/).

5.2. Prohlídka stromu složek

Všechny složky jsou různě hluboko zanořené podsložky jediného *kořenu (root)* se jménem /. Jak jsme již zmínili jako uživatel se nezajímáme, zda jsou tyto podsložky kořene na stejném nebo na různých oddílech a discích.

Hierarchie a význam složek vychází z unixové tradice a je popsána normou [Filesystem Hierarchy Standard \(FHS\)](#). Tento standard dodržuje nejen Ubuntu a Debian, ale i všechny velké hlavní distribuce.

Poznámka

Názvy složek uvádíme vždy s ukončující /, takže vždy bezpečně rozlišíte složku/ od souboru. Doporučujeme i vám dodržovat tuto konvenci.

Nejdůležitější složky stromu jsou následující:

- / – kořen, root
 - /bin/ – binárky klíčových programů jako ls, mount, rm, ...
 - /boot/ – soubory jádra, bootloader, konfigurační soubory
 - /dev/ – virtuální složka zařízení
 - /etc/ – konfigurace společná pro všechny uživatele
 - /home/ – domovské složky uživatelů kromě roota
 - /lib/ – základní sdílené knihovny a moduly jádra

- `/media/` – přípojný místo pro *externí a odjímatelné (removable) zařízení* jako externí USB pevné disky, CD mechaniky ap. V Ubuntu Desktop zde najdete vložené CD/DVD disky.
- `/mnt/` – přípojný místo pro *dočasně připojená* zařízení jako síťové disky ap. Zda zařízení připojit sem nebo do `/media/` je spíše konvence. Pro ruční připojování bývá tradičnější tato složka.
- `/opt/` – software, který instalujete manuálně mimo systém balíčků.
- `/proc/` – virtuální složka se systémovými informacemi zejm. o procesech
- `/root/` – root má tuto domovskou složku přímo v rootu / odděleně od `/home/`
- `/sbin/` – důležité správcovské programy převážně určené výhradně pro roota jako fsck, halt, reboot, ifconfig ap.
- `/sys/` – virtuální složka pro zjištění nebo nastavení informací o jádře
- `/tmp/` – místo pro dočasné soubory a složky
- `/usr/` – uživatelské programy, neboli většinou software, instalovaný dodatečně, po skončení instalace.
 - `/bin/` – v případě GUI systému jsou zde převážně grafické programy.
 - `/lib/` – sdílené knihovny uživatelských programů
 - `/share/doc/` – dokumentace programů (README, návody ap.) a manuálové stránky. V době před
 Googlem se hledali informace právě zde :-)
- `/var/` – data aplikací. Asi nejdůležitější složka pro zálohy. Jsou zde data všech instalovaných aplikací - soubory databázového serveru, HTML a skripty webových aplikací ap.
 - `/log/` – logy aplikací a systému (jádra)
 - `/www/` – Apache web server složka webových aplikací
 - ... - další složky podle instalovaných aplikací
- `/srv/` – plánováno jako datové adresáře služeb jako FTP, HTTP, ale není téměř využíváno a složka je většinou prázdná

5.3. Relativní a absolutní cesta

Napíšete-li např. `cd acc/2014/` může to znamenat skok do `/home/joe/acc/2014/`, `/home/lisa/Documents/acc/2014/`, `/var/backups/acc/2014/` atd. v závislosti na tom jaká byla aktuální složka v době provádění `cd`. Takto napsaná cesta bez počátečního `/` je **relativní**, tedy vyhodnocuje se podle místa, kde jste byli na začátku.

V relativní cestě můžeme použít symbolické složky `..` pro nadřazenou složku a `.` pro aktuální složku. Např. cesta `../.. /acc/2014/` vede do `/acc/2014/` ve složce o dvě výš, než aktuální. Aktuální složku `.` již znáte při spouštění programů např. `./muj-skript`.

Naproti tomu `cd /var/log/apache/` je vždy jednoznačné a nezáleží na aktuální složce v době volání `cd`. Cesta s počátečním `/` je **absolutní** a platí vždy odkudkoliv.

Který způsob psaní cesty zvolit závisí na situaci. Měli byste však absolutní i relativní cesty bezpečně ovládat a podle kontextu využívat jeden nebo druhý způsob určování diskové cesty.

5.4. Práce se soubory

5.4.1. Volné místo

`df` (disk free), `-h` jako human-readable jednotky (kiB, MiB, GiB):

```
df -h
```

5.4.2. Velikost složek

`du` (directory utilization), `-h` human-readable jednotky, `-b` velikost na disku místo pouhé velikosti. Program vypíše velikosti aktuální složky. Výstup může být na více obrazovek, proto je vhodné jej stránkovat v `less`:

```
du -bh | less
```

5.4.3. Vytvoření složky

`mkdir` (make dir):

```
$ mkdir nova-slozka
```

Obdobou v MS-DOSu byl příkaz `md`.

Užitečnou volbou je `-p`, kdy příkaz vytvoří i neexistující složky:

```
$ mkdir -p /neexistujici/slozky/budou/vytvoreny
```

5.4.4. Vytvoření souboru

Soubory jsou vytvářeny převážně prostřednictvím aplikací. Čas od času se však hodí vytvořit prázdný soubor:

```
$ touch novy-soubor
```

Použití `touch` (dotkni se) na vytváření prázdných souborů je trochu „zneužití“ tohoto programu, jehož původním účelem bylo aktualizovat čas poslední modifikace souboru. Využíváme vlastnosti `touch` tím, že program, neexistuje-li již soubor s tímto názvem, jej vytvoří.

5.4.5. Vymazání složky

`rmdir` maže složky, ale bohužel jen prázdné:

```
$ rmdir prazdna-slozka
```

Proto se používá univerzálnější `rm`, který projde rekurzivně (`-r`, `--recursive`) obsah a násilím vymaže i neprázdné složky (`-f`, `--force`):

```
$ rm -rf neprazdna-slozka/
```

5.4.6. Vymazání souboru

```
$ rm soubor
```

5.4.7. Kopírování

`Cp` kopíruje standardně jen soubory a jen v přímé podúrovni (ne v podadresářích). Pokud nám to stačí, pak:

```
$ cp odkud kam
```

Pro kopírování adresářů a podadresářů slouží volba `-r`, `-R`, `--recursive` (můžeme si vybrat parametr, který se vám líbí nejvíce):

```
$ cp -r nejaka/slozka/ do/jine/slozky
```

5.4.8. Přesun a přejmenování

Operace přesun a přejmenování jsou z technického pohledu identické. Příkaz `mv` (move) tedy můžeme použít pro oba druhy změny:

```
$ mv soucasny-nazev novy-nazev  
$ mv soubor ../zaloza/
```

5.5. Odkazy (linky)

Odkazy (links) mohli unixům ostatní systémy dlouhou dobu jen tiše závidět. S odkazem pracujete jako by se jednalo o originální soubor nebo složku. Díky tomu můžete vytvářet iluzi, že se stejný soubor vyskytuje na více místech. Změna je tak nutná jen v originálu.

Linux rozlišuje dva druhy odkazů:

- operace nad **pevným odkazem (hard link)** se chovají jako by byly učiněny nad originálem. Smazání pevného odkazu znamená smazání originálu samotného.
- jako prevence nechtěného smazání originálu se proto mnohem častěji používají **symbolické odkazy (symlinks nebo soft links)**, kdy odkaz i originál existují víceméně nezávisle. Musíme sami zajistit, aby se při přejmenování, přesunutí nebo smazání originálu nestaly *neplatnými odkazy*, které nikam nesměřují.

Vytváření odkazů obstarává program `ln`. Bez parametru vytváří pevné odkazy:

```
$ ln original odkaz
```

S parametrem `-s` bude odkaz symbolický:

```
$ ln -s original symbolicky-odkaz
```

Tip

Cestu k originálu i odkazu doporučujeme uvádět absolutně.

Další výhodou symlinků oproti pevným odkazům je, že symlink může být na jiném zařízení (diskovém oddílu), než originál na který odkazuje.

Pevné i symbolické linky uvidíte ve výpisu `ls -l` jako šipky na originál:

```
$ ls -l /etc/rc6.d/
total 4
lrwxrwxrwx 1 root root 13 úno  5 14:43 K01t1p -> ../init.d/t1p
lrwxrwxrwx 1 root root 17 úno 23 13:52 K09apache2 -> ../init.d/apache2
lrwxrwxrwx 1 root root 29 úno  5 12:12 K10unattended-upgrades -> ../init.d/unattended-upgrades
lrwxrwxrwx 1 root root 18 úno  5 13:20 K20flexibee -> ../init.d/flexibee
lrwxrwxrwx 1 root root 20 úno  5 12:12 K20kerneloops -> ../init.d/kerneloops
lrwxrwxrwx 1 root root 27 úno  5 12:12 K20speech-dispatcher -> ../init.d/speech-dispatcher
...
```

5.6. Vyhledávání

Na vyhledávání z příkazové řádky v Linuxu existují tři hlavní nástroje.

5.6.1. find

Program `find` je jedním z nejsložitějších vůbec a množství voleb je doslova dech beroucí. `find` dovede vyhledávat na základě rozličných kritérií jako: datum modifikace, vlastník, hloubka vnoření, velikost větší, než atd. S vyhovujícími soubory umí kromě vypsaní provádět i změny jako přejmenování, vymazání atd. atd..

My zredukujeme bohaté možnosti `find` na hledání souboru nebo složky podle jména. Obecná syntaxe `find` pro tento případ je:

```
find <kde> [-type <f|d>] -name <výraz>
```

Jako `kde` uveďte místo začátku vyhledávání nebo prostě aktuální složku (`.`). Vynecháte-li `-type` úplně nebo uvedete `-file f` bude se hledat mezi běžnými soubory. Pro hledání mezi složkami slouží `-file d`. Hledaný výraz může být prostý („výkazy2014.ods“ přesně) nebo obsahovat hvězdičky (vše vyhovující „výkazy*.ods“ jako „výkazy2014.ods“, „výkazy2013.ods“, ale i jen „výkazy.ods“).

Hledání v aktuální složce souboru „chybejici“:

```
$ find . -name "chybejici"
```

Hledání složky obsahující výraz „2013“ kdekoli (root /):

```
$ find / -type d -name "*2013*"
```

5.6.2. locate

Když porovnáte rychlost hledání souborů nebo složek pomocí find a s program locate, tak zjistíte, že locate hledá prakticky okamžitě:

```
$ locate "2013"  
/home/lisa/Documents/acc/2013/  
/home/lisa/Documents/payroll2013.ods  
...
```

Jak je to možné? Locate nevyhledává soubory na disku, ale v průběžně vytvářené databázi. Tento index je zpravidla aktualizován jednou denně. Locate tedy nenajde nedávno vytvořené soubory.

5.6.3. grep -r

Poslední možnost hledání vlastně již znáte. Program grep s volbou `-r` (rekurzivně) slouží pro hledání *ne souborů, ale v obsahu souborů*. Volání můžete doplnit parametrem `-i`, aby grep nerozlišoval velikost písmen.

Hledání v aktuální složce:

```
$ grep -ri "výraz"
```

Hledání v zadané složce:

```
$ grep -ri "výraz" cesta/kde/hledat
```

Připravte se, že hledání v obsahu může trvat velmi dlouho.

5.7. Archívy a komprimace

Na začátek vysvětleme, jaký je rozdíl mezi archivací a komprimací (kompresí).

- **Archivace** je uložení více souborů a složek do jediného souboru pro snadnější manipulaci.
- **Komprimace (kompresa)** je uložení více souborů a složek do jednoho nebo více souborů s cílem menší velikostí.

Běžným programem pro kompresi, resp. dekompresi je **gzip** a **gunzip**. Soubory mají většinou příponu `.gz` nebo `.gzip`.

Varování

Navzdory podobnému jménu nemají gzip/gunzip nic společného s komprimáčním formátem ZIP (algoritmus PK-ZIP) a známým programem WinZip. Ale pracovat se ZIP soubory můžete i v Linuxu pomocí programů *zip* a *unzip*.

Méně se můžete setkat s komprimovanými soubory `.bz2`, které mají lepší kompresní poměr, než `.gz`, ale nejsou tak rozšířené. K vytváření a rozbalení bysme použili programy **bzip2** a **bunzip2**.

Tradičním unixovým programem pro archivaci je **tar** (tape archiver), který dnes samozřejmě používáme s běžnými soubory na disku místo s páskovými mechanikami. Obvyklou příponou je `.tar`. Tar však umí v jednom kroku soubory zkomprimovat i zaarchivovat (a obráceně). Takové soubory mají příponu `.tar.gz`, `.tgz` pro tar+gz, resp. `.tar.bz2` pro tar+bzip2.

Časté volby pro tar jsou:

- `v` (verbose) -- činnost vypisovat na obrazovku
- `z` -- použít komprimaci/dekomprimaci gzip
- `f` -- přijímat vstup ze souboru, nikoli z STDIN

Komprimace

```
$ gzip velky-soubor
```

Dekomprimace

```
$ gunzip velky-soubor.gz
```

Vytvoření archívu

Syntaxe:

```
$ tar cvf <archiv>.tar [soubor | slozka]...
```

Např.:

```
$ tar cvf archiv.tar soubor1 soubor2 slozka1 slozka2 slozka3/podslozka1
```

Pokud potřebujeme vytvořit zkomprimovaný archiv, pak přidáme parametr `-z` (gzip):

```
$ tar cvfz archiv.tar.gz soubor1 soubor2 slozka1 slozka2 slozka3/podslozka1
```

Vypsání obsahu archívu

```
$ tar tvf archiv.tar | less
$ tar tvf archiv.tar.gz | less
```

Rozbalení archívu

Tar archiv (ne tar+gzip) rozbalíme do aktuální složky pomocí

```
$ tar xvf archiv.tar
```

Jedná-li se o zkomprimovaný archiv přidáme parametr `+-z+` (unzip)

```
$ tar xvfz archiv.tar.gz
```

5.8. Midnight Commander (mc)

Poté, co jsme se trápili s příkazy pro práci se soubory, archivaci, komprimaci a vyhledáváním, se budete možná zlobit, že si představíme mc jako poslední program této kapitoly.

Midnight Commander (mc) je souborový manažer vycházející ze slavného Norton Commanderu. V Ubuntu není standardně a proto si ho nainstalujte a pak spusťte pomocí mc:

```
$ sudo apt-get install mc
$ mc
```

Left	File	Command	Options	Right			
<-	...ebrowser/data/icons/16x16	-	.[^]>	<-	...ource-code-browser-master	-	.[^]>
'n	Name	Size	Modify time	'n	Name	Size	Modify time
/..	UP--DIR		zář 9 2011	/..	UP--DIR		kvě 2 22:50
missing~age.png		576	zář 9 2011	/sourceco~browser		4096	dub 19 18:52
source-class.png		830	zář 9 2011	.gitignore		6	zář 9 2011
source-c~ser.png		830	zář 9 2011	README.markdown		4261	zář 9 2011
source-d~ine.png		424	zář 9 2011	echo.txt		10	dub 23 11:53
source-e~tor.png		424	zář 9 2011	ping.txt		15490	dub 23 12:12
source-field.png		424	zář 9 2011	random.txt		41	dub 23 11:50
source-f~ion.png		604	zář 9 2011	sourceco~r_patch		1441	dub 19 18:48
source-macro.png		424	zář 9 2011	sourceco~.plugin		308	dub 19 18:52
source-m~ber.png		604	zář 9 2011				
source-m~hod.png		604	zář 9 2011				
source-n~ace.png		679	zář 9 2011				
source-p~rty.png		472	zář 9 2011				
source-s~uct.png		783	zář 9 2011				
source-table.png		693	zář 9 2011				
UP--DIR				.gitignore			
219G/443G (49%)				219G/443G (49%)			
Hint: Want to see your *~ backup files? Set it in the Configuration dialog.							
er/data/icons/16x16\$							
1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDr 10Quit							

Spása jménem Midnight Commander

6. Souborová oprávnění

Linux a Unix jsou od počátku víceuživatelské systémy. Bylo tedy třeba od sebe jednotlivé uživatele a jejich soubory „oddělit“ a „ochránit“ před ostatními. Systém souborových oprávnění na základě identity uživatele a jeho členství ve skupinách je tím hlavním prostředkem.

6.1. Sady oprávnění

Každý soubor nebo složka má přiřazeno **právě jednoho vlastníka** a **právě jednu skupinu**. Oprávnění se samostatně nastavují pro

- *vlastníka* – konkrétní uživatel
- *skupinu* – uživatelé, kteří jsou členy konkrétní skupiny
- *svět* – neboli všichni ostatní, kteří nejsou uživatelem ani členy skupiny

6.2. Druhy práv

Soubor nebo složka má prvotně nastaveného vlastníka a skupinu na uživatele, který soubor vytvořil, a jeho primární skupinu. Základní oprávnění nastavitelná pro každou tuto sadu uživatelů jsou

- *právo číst (read, r)*
- *právo zapisovat (write, w)*
- *právo spustit (execute, x)*

Upozornění

Pozor na to, že práva se chovají jinak pro soubor a jinak pro složku!

Souborová práva

Právo	Pro soubor	Pro složku
číst (r)	zobrazit soubor	zobrazit obsah složky (příkaz ls)

Právo	Pro soubor	Pro složku
zapisovat (w)	editovat soubor	vytvořit, přejmenovávat, přesouvat a mazat soubory ve složce
spustit (x)	spustit soubor	vstoupit do složky nebo skrze projít (příkaz cd)

Vlastní skupina a oprávnění souboru jsou vidět v již důvěrně známém výpisu `ls -l`.

```
$ ls -lha
total 14M
drw-r--r-- 6 libor libor 4,0K Dec 26 14:28 .
drw-r--r-- 83 libor libor 4,0K Dec 26 14:28 ..
-rw-r--r-- 1 libor libor 1,2M Dec 26 14:28 certificate1.ott
-rw-r--r-- 1 libor libor 16M Feb 28 18:40 backup.tar.gz
drwxr-xr-x 3 libor libor 4,0K May 23 12:05 gedit-plugins
-rw-r--r-- 1 libor libor 38K Jan 23 10:41 random.txt
```

The diagram below maps the columns of the `ls -l` output to their respective components:

- typ: drw, -rw, drwx, -rw
- oprávnění: r--r--
- odkazů: 6, 83, 1, 1, 3, 1
- vlastník: libor
- skupina: libor
- velikost: 4,0K, 4,0K, 1,2M, 16M, 4,0K, 38K
- čas posl. modifikace: Dec 26 14:28, Dec 26 14:28, Dec 26 14:28, Feb 28 18:40, May 23 12:05, Jan 23 10:41
- název: ., .., certificate1.ott, backup.tar.gz, gedit-plugins, random.txt

Výstup `ls -l`

Oprávnění je zapsáno symbolicky jako tři trojice pro vlastníka, skupinu a svět. Celkem tedy 9 znaků. Zápis je vždy v pořadí rwx. Není-li právo přítomné je uvedena místo práva pomlčka, např. `r--`, `rw-` ap.

Poblázněná práva

Ne všechny kombinace práv souborů a složek mají smysl. Hlavně práva složky dovedou s výslednými možnostmi práce zamíchat.

Např. právo spustit lze nastavit na jakýkoli soubor, ale samozřejmě to má smysl jen na soubor, který je spustitelným programem. V Linuxu spustitelné programy většinou nemají žádnou příponu `.exe` ap. Spustitelné jsou ovšem často skripty, které mohou používat přípony jako `.sh` pro Bash, `.py` pro jazyk Python atd.

Další příklad bláznivého výsledku je pro složku nastavit jen `r` nebo jen `x` – nemůže-li do složky vstoupit (`x`), nemůže vypsát ani její obsah (`r`) a naopak. Výsledkem je, že i když jste vlastník vidíte při pokusu o `cd` nebo `ls` chybu *Permission denied*.

Oprávnění pro složku „přebíjí“ oprávnění na soubory. Pokud na složku máme oprávnění zapisovat můžeme zde smazat soubory i když k souborům právo zápisu nemáme. Podobně nemůžeme soubor modifikovat (třeba přejmenovat nebo editovat) i když na něj máme právo zápisu, ale nemáme ho na složku souboru.

Typické oprávnění pro složky je `rw-r-xr-x` (číselně /viz později/ vyjádřeno 755), tedy vlastníku umožňující cokoli, lidem ze skupiny a světu číst složku a vstoupit.

Typické oprávnění pro soubory je `rw-r--r--` (číselně 644), tedy vlastníku umožňující číst a editovat, skupině a světu jen soubor číst.

6.3. Číselné vyjádření oprávnění

Zkušenější správci kromě symbolického zápisu oprávnění používají i *číselné (oktalové) vyjádření* [1] jako např. 644, 755, 777 ap. Výhodou tohoto způsobu je, že místo zápisu a čtení 6 znaků potřebujeme jen 3 číslice.

Převod ze symbolických práv na číselná spočívá jen v součtu hodnot jednotlivých práv.

Hodnoty symbolických práv osmičkově

Právo	Oktalová hodnota
<code>rwx-</code>	<code>4210</code>

Nejlépe to pochopíte z příkladů:

<code>rw-</code>	<code>r-</code>	<code>-</code>
<code>4 + 2 + 0</code>	<code>4 + 0 + 0</code>	<code>0 + 0 + 0</code>
<code>6</code>	<code>4</code>	<code>0</code>
640 osmičkově		

nebo

<code>rwX</code>	<code>r-X</code>	<code>r-X</code>
------------------	------------------	------------------

4 + 2 + 1	4 + 0 + 1	4 + 0 + 1
7	5	5
755 osmičkově		

6.4. SUID a SGID

Kromě základních práv rwx bychom měli znát

- **právo SUID** (set user ID upon execution, nastav uživatelé po spuštění) – symbolicky „s“
- **právo SGID** (set group ID upon execution, nastavit skupina po spuštění) – symbolicky „S“

Jakýkoli program, který spustíme, přebírá naše vlastní oprávnění. Nemůže tedy provést to, co my sami nemůžeme. Je-li však na souboru programu nastaveno SUID nebo SGID právo, pak je spuštěn s právy vlastníka (SUID) nebo skupiny (SGID).

Např. program `passwd` může použít i běžný uživatel, který si chce změnit heslo sám sobě. Jak už víte, je heslo uloženo jako hash hodnota v `/etc/shadow`. Ne-superuživatel však právo zápisu (editace) k tomuto souboru nemá (a nesmí mít). Tedy přestože `passwd` spouští omezený uživatel, sám program `passwd` má práva roota, protože root je vlastníkem a na `passwd` je nastaveno SUID. (Všimněte si písmena „s“ napozici „x“.)

```
$ ls -l `which passwd`
-rwsr-xr-x 1 root root 47032 čec 26 2013 /usr/bin/passwd
```

Tyto speciální práva mají nastaveny i další programy `ping`, `crontab` ap.

SUID/SGID na složkách

Nastavení SGID na složku má zásadně odlišný význam, než na souboru:

1. Soubory vytvořené ve SGID složce dědí skupinu této složky, nikoli primární skupinu aktuálního uživatele.
2. Podsložky vytvořené ve SGID složce mají nastaveno také SGID právo.

Pokus o nastavení SUID na složku nemá význam a je ignorován.

6.5. Sticky bit

Poslední a ještě „exotičtější“ právo je sticky bit, který by snad šlo přeložit pro zasmání jako „lepivý bit“. Symbolicky označovaný písmenem „t“ pro složky, nebo „T“ pro soubory (na ty však nemá žádný vliv - viz dále).

Poznámka

Význam sticky bitu, který dále popíšeme není původním smyslem tohoto oprávnění, tak jak platil(i) v Unixech HP-UX, UnixWare ap. Použití v dnešním Linuxu je úplně odlišné.

Je-li sticky bit nastaven na složce, pak zde jen vlastník složky (a root) může přejmenovat, přesouvat a mazat soubory či podsložky.

Sticky bit je v Linuxu nastaven na složku `/tmp/`, takže více uživatelů může do této složky ukládat své dočasné soubory a přitom nemohou smazat soubory cizích uživatelů. (Všimněte si písmena „t“ na pozici „x“.)

```
$ ls -ld /tmp/
drwxrwxrwt 18 root root 12288 kvě  4 18:17 /tmp
```

Pokus o nastavení sticky bitu na soubor nemá žádný význam a je ignorován.

6.6. Nastavení oprávnění – chmod

Příkazem chmod (change mode) modifikujeme oprávnění souborů a složek.

Symbolicky zadávané oprávnění

Obecná syntaxe pro symbolický zápis vypadá:

```
chmod <sada> (+ | - | =) <práva> <soubor/nebo/složka>
```

Kde `<sada>` může být

- `u` pro vlastníka (user)

- **g** pro skupinu (group)
- **o** pro svět (others)
- **a** pro „všichni“ (all)

Chceme-li nastavit práva např. pro vlastníka i skupiny, můžeme jako sadu uvést **ug** (user + group) ap.

Poznámka

Chmod bohužel nepoužívá terminologii „owner“, „group“, „world“. Nepleťe si tedy **o** s vlastníkem (owner), který je pro chmod trochu podivně jako **u** (user).

Následuje operátor

- **+** pro přičtení práv k současným
- **-** pro odebrání práv od současných
- **=** pro nastavení stanovených práv

Práva zapisujeme již známým způsobem zkratkami **r** pro čtení, **w** pro zápis, a **x** pro spuštění.

Příklady:

```
# Skupině přidat zápis
$ chmod g+w soubor.txt

# Vlastníkovi odebrat spuštění
$ chmod u-x soubor.txt

# Světu přidat čtení a odebrat spuštění
$ chmod o+r-x soubor.txt

# Všem nastavit právo čtení a spuštění
$ chmod a=rx soubor.txt

# Vlastníkovi a světu nastavit právo čtení a spuštění
$ chmod uo=rx soubor.txt
```

Oktalově zadávané oprávnění

Pro numerický oktalový zápis je syntaxe prostá:

```
$ chmod 644 soubor.txt
```

Rekurzivně

Pro obě podoby zadání oprávnění se často připojuje parametr `-R`, `--recursive`, tedy oprávnění změní na zadané složce, všech podsložkách a souborech v ní:

```
$ chmod -R 400 slozka/
```

Poznámka

Pozor na záměnu `-R` a `-r`. Velké „R“ muselo být zvoleno, protože „r“ je interpretováno jako právo read.

Rekurzivně pracující `chmod` nastavuje zadané oprávnění na dceřiné soubory i složky. To ale není vždy to, co chceme. Pokud potřebujeme nastavit rekurzivně, ale jen na soubory nebo jen na složky, musíme si vypomoci s `find` nebo možnostmi `Bashe`.

```
# Nastavení všem souborům 644
find . -type f -exec chmod 644 "{}" \;

# Nastavení všem složkám 755
find . -type d -exec chmod 755 "{}" \;

# Nebo pomocí Bashe totéž pro složky
chmod 755 $(find /path/to/base/dir -type d)

# soubory
chmod 644 $(find /path/to/base/dir -type f)
```

6.7. Změna vlastníka – `chown`

Syntaxe `chown` (change owner) je snadná. `Chown` taktéž dovede pracovat rekurzivně s `-R`:

```
$ chown -R lisa /home/lisa/
```

Protože je často měněn vlastník i skupina, má chown speciální syntaxi <vlastník>:<skupina> pro změnu v jednom kroce:

```
$ chown -R lisa:marketing /var/share/public/marketing
```

6.8. Změna skupiny – chgrp

Chgrp (change group) má stejnou syntaxi i chown a rovněž podporuje volbu `-R` pro rekurzivní operaci:

```
$ chgrp -R marketing /var/share/public/marketing/
```

6.9. Výchozí oprávnění – umask

Možná vás už napadlo, jaká oprávnění budou mít vytvářené soubory, když „nějaká“ mít musí a vy jste je nemodifikovali (a většině případů ani asi nebudete). Obecná odpověď neexistuje, protože záleží na programu, kterým složky a soubory zakládáme, ale seznámíme se s *většinou* platnými konvencemi.

Tradiční linuxové programy při vytváření souborů nastavují `rw-rw-rw-` („dábělské“ 666), a pro složky `rwxrwxrwx` („andělské“ 777). Takto vytvořené soubory jsou čitelné a zapisovatelné pro všechny a složky plně přístupné taktéž pro všechny. Když vytvoříme soubor, tak ale tato oprávnění (naštěstí) nemá:

```
$ touch soubor
$ ls -l soubor
-rw-r--r-- 1 libor libor      0 kvě  4 20:42 soubor
```

Hodnota umask

Tradiční linuxové programy ctí hodnotu označovanou **umask** (user file creation mode mask), která se od těchto „plných“ práv odečte. Hodnota umask se nastavuje stejnojmenným příkazem umask, ale specifikujeme práva, která mít nově vytvářené soubory a složky mít *nemají*.

Umask bez parametrů hodnotu vypíše:

```
$ umask
0022
```


Hodnotu `umask` získáme odečtením požadovaného oprávnění od 777, ale `umask` podporuje i symbolické zadávání i zobrazování ve stejném formátu jako `chmod`.

Pozor, že výstup `umask -S` jsou práva, která *mají být, nikoli nesmí být* nastavena:

```
$ umask -S
u=rwx,g=rX,o=rX
```

Nastavení hodnoty `umask`

Při nastavování nového masky `umask` si musíme uvědomit, že platí až od této chvíle a pro procesy spuštěné ve stejném shellu. Při restartu nebo odhlášení se ztrácí a proto se `umask` nastavuje ve startovacích skriptech

- např. `~/ .bashrc` konkrétního uživatele
- nebo `/etc/login.defs` a `/etc/environment` pro všechny uživatele

Pravidla určení výchozího oprávnění a vlastnictví

- nově vytvořený objekt patří uživateli, který ho vytvořil a primární skupině tohoto uživatele
- nově vytvořený objekt má implicitně oprávnění určená příkazem `umask`
- oprávnění může měnit vlastník objektu nebo správce systému (`root`)
- vlastníka může měnit pouze `root`, v některých případech i majitel (za speciálních podmínek)
- skupinu může měnit `root`, v některých případech i majitel (za speciálních podmínek)

Operační systém nezasahuje do zapsaných údajů, pokud nemusí. Proto při přejmenování nedojde k ovlivnění oprávnění ani vlastníka či skupiny. Naopak při kopírování patří kopie tomu, kdo si ji vytvořil. Při přesunu záleží na tom, jestli je potřeba vytvořit nový i-uzel (`inode`) (při přesunu mezi různými souborovými systémy jde vlastně o kopírování s následným smazáním originálu) nebo nikoli (jde vlastně o variantu přejmenování).

([Zdroj Wikipedia](#))

6.10. POSIX a ACL oprávnění

Právě vysvětlená oprávnění jsou označována někdy jako *tradiční* nebo *POSIX* oprávnění. Vznikala v době, kdy autoři Unixu stáli před problémem vyřešit souborová oprávnění co nejjednoduššeji, protože výkonnostní a paměťové limity počítačů byly velmi nízké.

Přesto tento POSIX model oprávnění téměř vždy vyhovuje, a proto je hojně využíván dodnes.

Nicméně jako alternativní k tomuto tradičnímu unixovému přístupu existují tzv. *ACL (Access Control List, seznam oprávnění) modely*, které jsou podobné např. oprávněním známým z Windows. Na diskovou položku nastavujeme seznam oprávnění pro neomezený počet uživatelů a skupin jednotlivě.

Vzhledem k velmi malému použití ACL oprávnění v praxi se tímto modelem nebudeme hlouběji zabývat.

Poznámky

[1] Neboli v osmičkové soustavě. Pozor na překlep: oktalové, nikoli oktanové :-)

7. Instalace a správa programů

V této kapitole se zaměříme na možnosti instalace a správy aplikací v Linuxu, a odlišnosti Linuxu a ostatních OS.

Důležité

Protože velkou pozornost budeme věnovat balíčkovacímu systému DEB bude tato kapitola jako jediná téměř výhradně specifická pro systémy Debian a Ubuntu.

7.1. Kompilace

Kompilace byla tradiční a dlouho jediná možnost, jak získat pro náš počítač nový software.

Kompilace neboli překlad je převod zdrojových kódů (většinou v jazyce C nebo C++) do *strojového kódu* vašeho procesoru. Jistě víte, že Linux sám i většina programů pro Linux je open-source, tedy volně šiřitelná včetně zdrojových kódů. Ze stránek aplikace proto stáhnete „zdrojáky“ převážně jako .tar.gz nebo je získáte přímo ze systému verzování kódu (VCS) jako Git, Subversion, CVS ap.

Proč nekompilovat

Kompilace je však ta nejhorší možnost a měli byste se jí vyhnout, kdykoli můžete. Kompilace není zrovna snadná ani pro zkušeného správce. Často skončíte záludnými chybějícími závislostmi na externí knihovny nebo chybovými hláškami, o kterých ani Google příliš neslyšel.

Kompilace je také časově náročná operace, která může trvat jednotky i desítky minut.

Poslední hlavní nevýhodou kompilace je, že nemáme žádnou skutečnou možnost programy aktualizovat a odinstalovat (nevíme, jaké soubory tvoří program). Všechno závisí na „slušnosti“ programu samotného.

Příprava

Jedná-li se o C/C++ program (většinou), nainstalujte nejprve kompilátor, linker, make builder ap.:

```
$ sudo apt-get install build-essential checkinstall
```

Možná budete potřebovat i verzovacího klienta (na 99,9% to bude CVS, SVN, Mercurial nebo dnes nejpoblárnější Git):

```
$ sudo apt-get install cvs subversion mercurial git-core
```

Stručný postup pro C/C++

Když už kompilovat musíme, popišme, alespoň velmi stručně jak na to. Většina open-source projektů používá GNU auto-tools se kterým má instalace ze zdrojáků tři kroky

1. „*configure*“ (příkaz `./configure [--volby...]`) – ověření, že jsou dostupné všechny potřebné závislosti, konfigurace přes kompilace a výsledného programu. Můžete zkusit `./configure --help` pro zjištění všech sestavovacích a instalačních voleb.
2. „*make*“ (příkaz `make`) – samotná kompilace
3. „*make install*“ (příkaz `checkinstall`) – dnes je posledním krokem spíše *checkinstall*, ale `make install` je tak hluboko zakořeněn, že této fázi budeme takto říkat. Checkinstall vytváří DEB balíček (viz dále).

Více informací najdete např. na <https://help.ubuntu.com/community/Compiling-Software>.

7.2. DEB balíčky

Na všechny bolesti kompilace existuje lék v podobě DEB balíčků. Jsou to soubory s příponou `.deb` obsahující zkomprimované

- již zkompilevané spustitelné soubory (binárky)
- konfigurační soubory
- administrativní údaje (licence, autor, web programu)
- systémové požadavky (architektura, jazyk ap.)
- závislosti na jiných balíčcích

Autor (správce) DEB balíčku si dal tu práci s kompilací pro naši architekturu za nás, odzkoušel funkčnost programu, případně přizpůsobil pro specifika distribuce.

7.2.1. Výhody DEB balíčků

Další výhody balíčkovacích systému jako DEB jsou

- atomické operace – jestliže se instalace nepovede, nemělo by dojít k ovlivnění systému, balíček můžete odstranit nebo instalaci opakovat.
- deklarace závislostí – balíček říká „potřebuji tento a tamten balíček“ a bez něj nám nedovolí instalaci (program by beztak nefungoval). Není tedy zmatek v tom, kdo určitou knihovnu vlastně potřebuje, v jaké verzi atp.
- skripty – DEB balíček může spouštět v různých okamžicích instalace skripty, takže někdy instalace může být mnohem více, než jen pouhé kopírování souborů z balíčku na disk.
- snadné aktualizace – balíčky jsou verzované. Když se pokusíte instalovat program novější verze, než máte, provede se jen aktualizace.
- seznam aplikací – víte, co jste si nainstalovali.
- odinstalace – při odinstalaci jsou odstraněny všechny soubory, které balíček na váš počítač přidal. Systém je po odebrání balíčku většinou v prakticky identickém stavu (včetně volného místa) jako byl před instalací.

Celý Ubuntu a Debian je vlastně udržován jako soustava několika tisícovek balíčků. Dokonce i jádro se distribuuje v podobě DEB balíčku. Již při instalaci se nekopírují soubory „jen tak“, ale probíhá instalace z příslušných balíčků.

7.2.2. Update vs. upgrade

Měli bychom rozlišovat mezi těmito zdánlivě stejnými termíny. Zvýšení verze jednoho programu, resp. balíčku nazýváme *update*. Update všech balíčků je *upgrade*, tj. vlastně celého operačního systému.

7.2.3. Nevýhody DEB balíčků

Nevýhodou DEB balíčků (ale i konkurenčních [RPM](#)) je, že nemáte vždy nejnovější verze programů. Zkrátka může nějaký čas trvat, než správce DEB balíčku příslušné aplikace vytvoří a otestuje aktualizaci.

Některé programy bohužel jako balíčky nejsou dostupné a tak občas nezbyvá, než „stará špatná“ kompilace.

7.2.4. Nástroj dpkg

Pro manipulaci s DEB balíčky staženými z internetu ap. slouží nástroj dpkg. Důležité volby jsou zejm.

- `-i`, `--install` pro instalaci, resp. update balíčku
- `-r`, `--remove` pro odinstalaci balíčku
- `-P`, `--purge` pro odinstalaci balíčku včetně konfiguračních souborů. Vhodné, když víte, že určitě už nebudete program nikdy provozovat.
- `-l`, `--list` výpis všech nainstalovaných balíčků. Výpis lze omezit na balíčky obsahující jen určitý výraz, např. `-l nano`.
- `-L`, `--listfiles` velmi užitečný parametr, která vám řekne, jaké soubory určitý balíček obsahuje (kde budou uloženy po instalaci)
- `--dry-run` běh „na sucho“ neboli jen simuluj, že se operace provádí. Dobré pro vyzkoušení, zda by instalace/odinstalace proběhla v pořádku. *Tento parametr musíte pochopitelně umístit před jakýkoli jiný, aby operace byla opravdu jen „jako“.*

Příklad instalace a odinstalace (s ponecháním konf. souborů):

```
$ sudo dpkg -i super-aplikace.deb
```

7.2.5. Úprava konfigurace – dpkg-reconfigure

Pokud se např. instalátor ptal na heslo správce aplikace a vy jste ho zapoměli, můžete opětovně spustit instalaci a zadat nové heslo příkazem:

```
$ sudo dpkg-reconfigure <balíček>
```

Možná si vzpomenete, že jsme už jednou dpkg-reconfigure použili ke změně rozložení klávesnice v textovém prostředí:

```
$ sudo dpkg-reconfigure keyboard-configuration
```

7.3. Repozitáře balíčků

Repozitář je úložiště a katalog stovek až stovek tisíc balíčků, kterých může balíčkový systém znát desítky. Jeden repozitář slouží např. na publikování výhradně bezpečnostní aktualizace OS, další pro komerční nebo jinak licencovaný software, další

hry ap. Existují i repozitáře o jednom balíčku. Ve větších organizacích se může vyplatit vnitřní repozitář s programy používanými v organizaci.

V repozitářích bývá více variant stejného balíčku stejné verze pro všechny podporované procesorové architektury, jazyky, balíček se zdrojovými kódy ap.

Instalace jednotlivých balíčků stažených z internetu nástrojem dpkg můžeme proto spíše považovat nízkouúrovňovou operaci a za základ pro řešení repozitářů balíčků.

System repozitářů zjednodušuje vyhledávání a instalaci balíčků. Požadovaný program vyhledává ve známých repozitářích. Pokud balíček závisí na dalších balíčcích, tak je zkusí rovněž najít a stáhnout v dostupných repozitářích.

Program na správu repozitářů také sám kontroluje, zda není v repozitářích novější verze softwaru, než máme nainstalován a případně nabídne jeho update.

Poznámka

Ano, tento princip je velmi podobný Google Play, App Store ap., ale v Linuxu existují tyto „obchody“ již desítky let.

Repozitáře jsou zkrátka prvním místem, kde hledat nové programy.

7.4. Správce repozitářů APT

Jedním z nejvyspělejších systémů pro správu repozitářů je APT - Advanced Packaging Tool. APT vznikl původně v Debianu a používá ho tedy Ubuntu, ale existují i porty pro distribuce mimo Debian a dokonce i pro RPM balíčky.

APT je ve skutečnosti několik programů `apt-<něco>`.

7.4.1. apt-get

Základním příkazem správce APT je `apt - get`. U všech variant příkazu můžete zadat jeden nebo více balíčků oddělených mezerou.

Instalace programu:

```
$ sudo apt-get install balíček [balíček2...]
```

Odstranění:

```
$ sudo apt-get remove balíček [balíček2...]
```

Odstranění včetně konfiguračních souborů:

```
$ sudo apt-get purge balíček [balíček2...]
```

Pro povýšení na novější verzi nebo po přidání nového repozitáře do `/etc/apt/sources.list` (viz dále) musíme obnovit lokální cache podle skutečného stavu repozitářů. Příkazy proto budou dva - obnova cache volbou `update`, a pak samotná instalace:

```
$ sudo apt-get update  
$ sudo apt-get install balíček [balíček2...]
```

Důležité

Obnova cache (`apt-get update`) je jen dotaz, zda neexistují aktualizace balíčků, které máme a detekce úplně nově přidaných balíčků v repozitářích. Až druhý příkaz `apt-get install` provede skutečnou aktualizaci, resp. instalaci pro nový balíček.

7.4.2. apt-key

Správa klíčů používaných pro ověřování autenticity balíčků. Jen balíčky ověřené těmito klíči jsou považovány za důvěryhodné.

7.4.3. apt-cache

Dotazování nad APT cachí balíčků.

7.5. Repozitář pod lupou

Repozitář je místo v lokální síti nebo internetu dostupné pod URL, která je známá programům pro správu repozitářů jako APT (viz dále).

Repozitáře jsou konfigurovány textovými soubory v `/etc/apt/`, z nichž nejdůležitější je `/etc/apt/sources.list` obsahující údaje o

- typu balíčku (je vždy `deb` nebo `deb-src` pro zdrojové balíčky)
- URL HTTP nebo FTP v internetu či intranetu, ale i na CD-ROM. URL většinou směřuje na lokální obraz (mirror) download serveru pro váš stát
- označení pro jakou verzi vaší distribuce jsou balíčky určeny

Soubor `/etc/apt/sources.list` (zkráceno a vynechány komentáře)

```
deb http://cz.archive.ubuntu.com/ubuntu/ saucy main restricted
deb-src http://cz.archive.ubuntu.com/ubuntu/ saucy main restricted

deb http://cz.archive.ubuntu.com/ubuntu/ saucy-updates main restricted
deb-src http://cz.archive.ubuntu.com/ubuntu/ saucy-updates main restricted

deb http://cz.archive.ubuntu.com/ubuntu/ saucy universe
deb-src http://cz.archive.ubuntu.com/ubuntu/ saucy universe
deb http://cz.archive.ubuntu.com/ubuntu/ saucy-updates universe
deb-src http://cz.archive.ubuntu.com/ubuntu/ saucy-updates universe
...
```

Klidně si některou URL otevřete ve webovém prohlížeči a podívejte se jak vypadá „formát“ DEB repozitáře.

Připomínáme, že jakmile upravíte `sources.list`, musíte provést `sudo apt-get update` pro obnovu informací o balíčcích v cache.

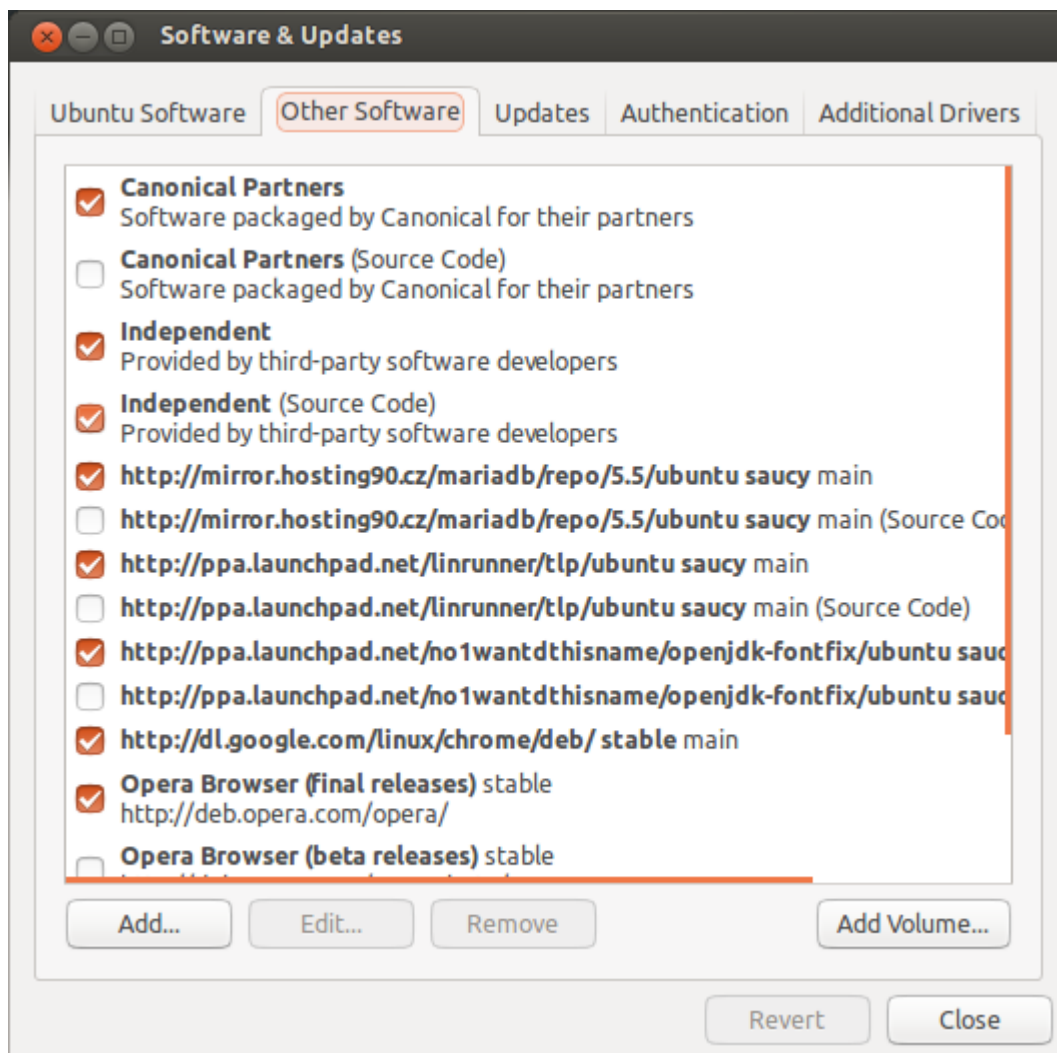
Tip

Jednoduše procházet a vyhledávat můžete standardní repozitáře distribuce také přes web na <http://packages.ubuntu.com>, resp. <http://packages.debian.org>.

7.6. Další programy pro repozitáře

7.6.1. Software & Updates

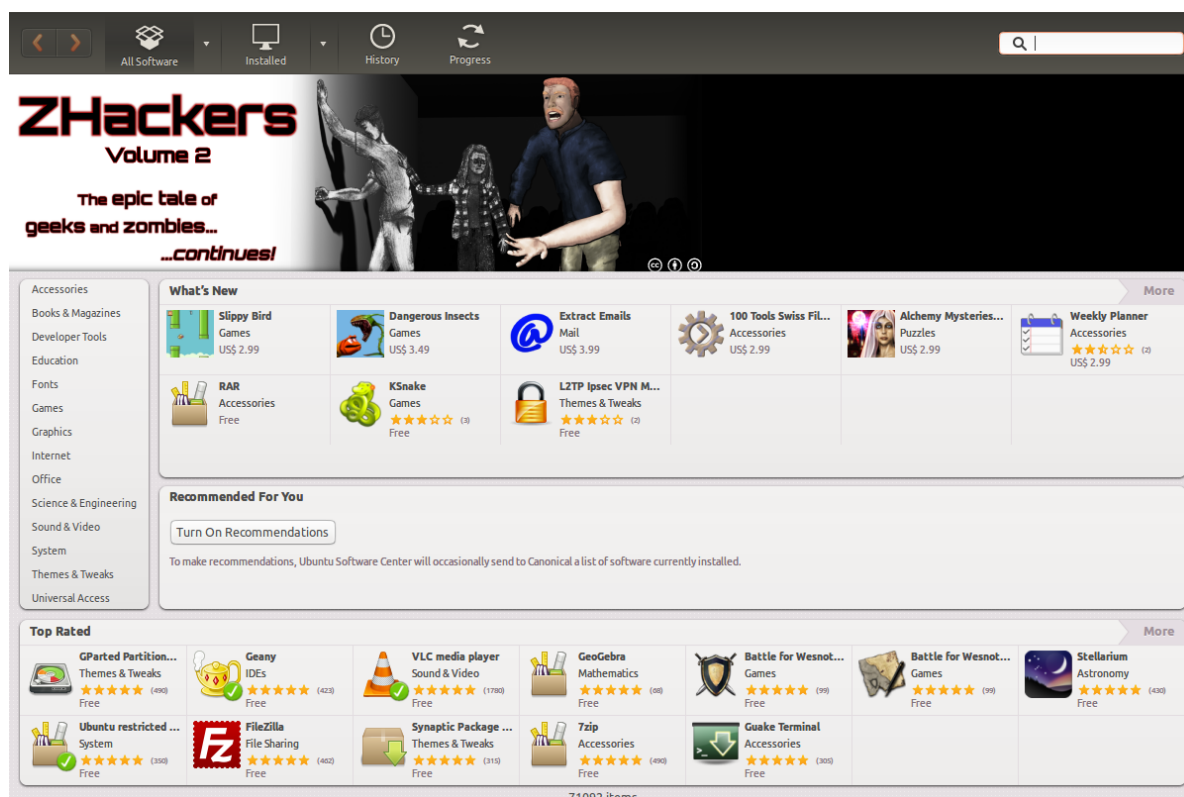
Repozitáře v případě Ubuntu Desktop můžete spravovat i graficky v nástroji Software & Updates.



Správce repozitářů Software & Updates

7.6.2. Ubuntu Software Center

Grafické Ubuntu Desktop obsahuje repozitářového klienta Ubuntu Software Center.



Ubuntu Software Center

7.6.3. Aptitude

Pro textové rozhraní ještě doporučujeme doinstalovat Aptitude, který by se dal přirovnat k Ubuntu Software Center a control panelu Software & Updates:

```
$ sudo apt-get install aptitude
$ aptitude
```

Při řešení problémů závislostí (chybějící, kolidující) je dokonce Aptitude chytřejší, než standardní apt-get.

7.7. RPM balíčky

Kromě výše probraných balíčků a repozitářů DEB pro rodiny Debian a Ubuntu Linuxu, bysme měli vědět o existenci balíčků RPM (Redhat Package Manager) původně vytvořených pro Redhat Linux, ale dnes používaných i v dalších distribucích (Fedora, SUSE).

Všechny popsané výhody, nevýhody a princip je velmi podobný DEB systému. Narážíte-li na program pro který existuje jen RPM balíček je možné jej jako *nouzové řešení* převést programem alien na DEB. Alien samozřejmě nekontroluje obsah, ale jen převádí formát z RPM na DEB. Zda bude tento balíček skutečně fungovat není jisté.

```
# Instalace konvertoru alien
$ sudo apt-get install alien

# Převod
$ alien balicek.rpm
```

Jako nízkoúrovňový ekvivalent dpkg pro RPM systémy slouží program yum:

```
$ yum install super-aplikace
```

8. Procesy a démoni

V této závěrečné kapitole kurzu Ubuntu a Debian Linuxu se naučíme trochu symbolicky počítač vypnout. Dále prozkoumáme procesy a jak se spravují. Posledním tématem budou démoni, neboli procesy běžící na pozadí, kteří plní nějakou službu.

8.1. Ukončení práce s PC

Poznámka

Vypnout nebo restartovat PC může v textovém prostředí jen superuživatel. Je to z důvodu, že Linux/Unix je od prvopočátku víceuživatelský systém a taková operace samozřejmě ovlivní i ostatní právě přihlášené uživatele.

Ubuntu nabízí více možností, jak ukončit práci s počítačem.

8.1.1. shutdown

Program shutdown připraví systém na bezpečné vypnutí nebo restart.

```
shutdown (-r | -h) <čas> [zpráva]
```

Všichni přihlášení uživatelé mohou obdržet na obrazovku hlášku *zpráva* (je-li uvedena) o chystaném vypnutí (-h) nebo restartu (-r) a 5 minut před *čas* je zabráněno novým přihlášením.

Čas je možné specifikovat jako

- konkrétní údaj *HH:MM*
- formou „od teď za N minut“ pomocí výrazu *+N*, kde N je počet minut
- nebo nejčastěji „teď hned“ slovem *now*

Příklady:

```
$ sudo shutdown 19:30 "Vážení uživatelé, restart v 19:30"  
$ sudo shutdown -r +5 "Vážené uživatelé, restart za 5 minut"  
$ sudo shutdown now
```

Poznámka

Volba *-h* zastaví OS a *pokusí se* vypnout napájení. Ve výjimečných případech, kdy není podporována správa napájení (většinou jen prehistorické PC), musíte fyzické vypnutí od elektrické energie provést sami.

8.1.2. reboot

Provede stejnou operaci jako `shutdown -r now`, ale je zapotřebí méně písmenek :-)

8.1.3. halt

Provede zastavení OS - nevypíná napájení počítače. Narozdíl do `shutdown -h` skutečně jen zastaví běh Linuxu. Využití v praxi neznáme.

8.1.4. poweroff

Provede stejnou operaci jako `shutdown -h now`, ale s méně písmenky :-)

8.2. Procesy

Proces je právě běžící program. Spustíte-li vy, jádro nebo jiný proces program, založí se nový proces.

Každý proces běží izolovaně od ostatních. Má vyhrazen vlastní paměťový prostor a čas procesoru. Komunikovat mezi procesy nebo uživatelem a procesy může probíhat jen pomocí V/V prostředků (disk, soubor, síť), standardních V/V (STDIN, STDOUT, STDERR), nebo tzv. signálů.

Někdy je proces třeba pozastavit, ukončit nebo změnit prioritu procesu. Jako super-uživatel můžete takto řídit procesy kohokoli, uživatelé mohou jen své vlastní.

8.2.1. Identifikace procesu

Mezi desítkami či stovkami procesů je třeba se nějak orientovat.

PID

Jádro i správce každý proces identifikují pomocí jednoznačného *PID (process ID)*. Většina dále popisovaných programů pro řízení procesů požaduje právě PID jako argument.

PPID

Pouze již existující proces může vytvořit nový proces, tj. podproces. Rodič tohoto podprocesu je označen jako *PPID (parent PID)*. Když hledáte, kdo vytváří „zblázněné“ procesy, stačí se podívat na jejich PPID hodnotu a hned znáte viníka.

UID a EUID

UID (User ID) majitele (kdo proces spustil) a efektivní UID.

8.3. Sledování procesů

8.3.1. ps – základní sledování

Tradičním programem pro sledování (výpis) procesů je ps. Velkou většinu jeho parametrů nebudeme nikdy potřebovat.

Poznámka

ps je ukázkou dokonce tzv. „trisexuálního“ chování. Možná si pamatujete, že např. find poskytoval BSD a GNU syntaxi. Protože ps byl programem používaným od prvopočátku a každý *nix systém si ho trochu přizpůsobil, je výsledkem sada parametrů obvyklých v UNIXu, BSD a GNU (Linuxu).

Bez parametrů ps vypíše procesy na aktuálním terminálu (TTY):

```
$ ps
PID TTY          TIME CMD
4354 pts/5        00:00:00 bash
10811 pts/5        00:00:00 ps
```

Druhé běžné použití je `ps aux`, kdy se vypíše seznam procesů a několik základních informací. (Kompletní popis sloupců hledejte v manuálové stránce.)

- USER – majitel procesu
- PID
- %CPU – vytížení CPU
- %MEM – využití paměti
- TTY – na kterém terminálu běží
- STAT – stav procesu (R=běží, S=spí ap.)
- START – od kdy běží
- COMMAND – příkaz, jakým byl proces vytvořen. Pozor na to, že program může sám sobě tento údaj

změnit a hodnota tedy nemusí přesně odpovídat. Pokud je uvedená hodnota v hranatých závorkách ([]) nejde o příkaz, ale démon jádra ([viz dále](#)).

Ukázka výpisu ps (zkráceno)

```
$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  27212  3048 ?        Ss   09:52   0:01 /sbin/init
root         2  0.0  0.0     0     0 ?        S    09:52   0:00 [kthreadd]
root         3  0.0  0.0     0     0 ?        S    09:52   0:00 [ksoftirqd/0]
root         5  0.0  0.0     0     0 ?        S<   09:52   0:00 [kworker/0:0H]
root         7  0.0  0.0     0     0 ?        S    09:52   0:00 [migration/0]
root         8  0.0  0.0     0     0 ?        S    09:52   0:00 [rcu_bh]
root         9  0.0  0.0     0     0 ?        S    09:52   0:00 [rcuob/0]
root        10  0.0  0.0     0     0 ?        S    09:52   0:00 [rcuob/1]
syslog    1086  0.0  0.0 247468  1528 ?        Sl   09:53   0:00 rsyslogd -c5
root     1092  0.0  0.0 272224  4848 ?        Sl   09:53   0:00 /usr/lib/polic
avahi    1120  0.0  0.0  32348  1728 ?        S    09:53   0:00 avahi-daemon:
avahi    1121  0.0  0.0  32228   468 ?        S    09:53   0:00 avahi-daemon:
```

8.3.2. pstree – ps ve stromě

Proces může vyvolat nový proces atd. Tyto stromové vazby zobrazíte příkazem `ps tree`. Více o řízení procesů v druhé části kurzu.

Ukázka výpisu pstree (zkráceno)



8.3.3. top – vylepšené sledování

Top je taktéž „vypisovač“ procesů, ale průběžně obnovovaný po 10 sekundách s neaktivnějšími nahoře. Kromě toho nabízí celkovou statistiku systému a přímo v programu akceptuje klávesové zkratky, kterými umí procesům posílat signály a měnit nice value.

Ukázka obrazovky top

```

$ top
top - 13:33:07 up 3:40, 2 users, load average: 1,12, 1,09, 0,91
Tasks: 266 total, 1 running, 265 sleeping, 0 stopped, 0 zombie
%Cpu(s): 12,1 us, 5,6 sy, 0,0 ni, 80,4 id, 1,8 wa, 0,0 hi, 0,0 si, 0,0 st
KiB Mem: 5976480 total, 5080600 used, 895880 free, 346596 buffers
KiB Swap: 16383996 total, 0 used, 16383996 free, 2230204 cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 1515 root        20   0   447m 149m 111m  S   12,6   2,6   8:17.78 Xorg
 3622 libor      20   0 1367m  91m  34m  S   11,3   1,6   8:56.65 compiz
 8972 libor      20   0   530m  30m  20m  S    8,6   0,5   2:04.34 gnome-system-mo
3534 libor      9  -11   551m 7612 5236  S    6,6   0,1   5:55.14 pulseaudio
3657 libor      20   0 1021m  48m  21m  S    6,6   0,8   3:36.88 radiotray
4346 libor      20   0   647m  19m  13m  S    6,3   0,3   0:10.88 gnome-terminal
5073 libor      20   0 2317m 258m 214m  S    6,0   4,4  12:07.25 VirtualBox
3656 libor      20   0   968m 157m  50m  S    1,3   2,7   5:11.86 chrome
7690 libor      20   0 2271m  91m  42m  S    1,0   1,6   4:49.84 vlc
1995 mysql      20   0   869m  97m 8948  S    0,7   1,7   0:20.27 mysqld
5044 libor      20   0   621m  10m 7472  S    0,7   0,2   1:22.84 VBoxSVC
  
```

Nejdůležitější klávesové zkratky pro top:

- **Page Down** a **Page Up** – o stránku dolů nebo nahoru
- **Shift+N** – třídění procesů podle PID
- **Shift+A** – třídění procesů podle PID od konce
- **Shift+P** – třídění procesů podle zatížení CPU (odhalení zaseknutých procesů)
- **Shift+M** – třídění procesů podle objemu zabrané paměti (odhalení viníků swapování)
- **Shift+T** – třídění procesů podle spotřebovaného strojového času (odhalení procesů nejvíce zatěžujících systém)
- **Shift+A** – třídění procesů podle PID od konce
- **M** – zapnutí nebo vypnutí informací o paměti
- **T** – zapnutí nebo vypnutí souhrnných informací o systému
- **K** – zabít právě vybraný proces
- **H** – nápověda
- **Q** – ukončení programu

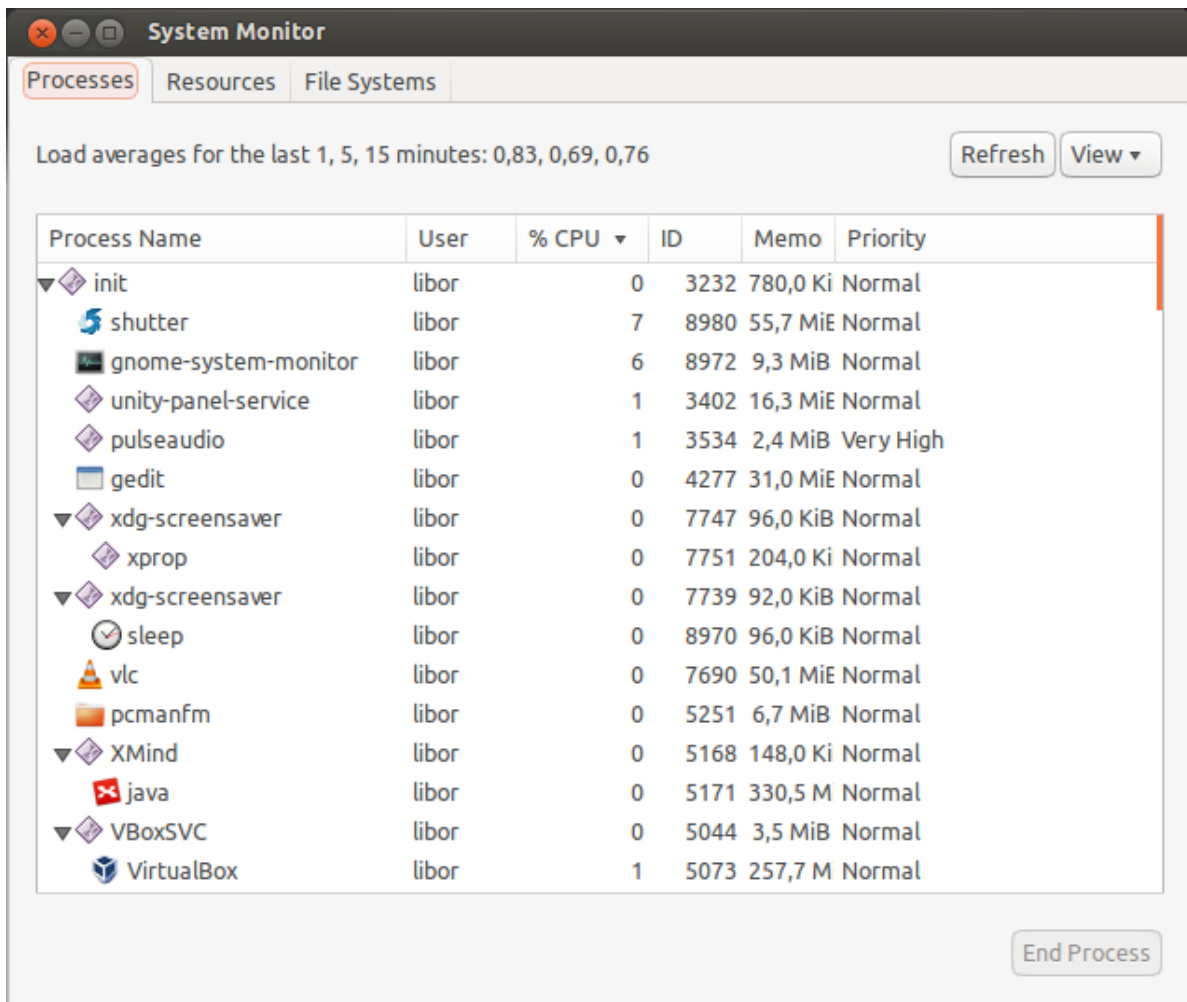
Tip

Populární nástroj, který je ale zapotřebí doinstalovat, je např. Glances. Jistě i sami

najdete sami několik dalších správců procesů.

8.3.4. Gnome System Monitor

Pokud jste v GUI, nejkomfortnější je správce procesů v Gnome.



Grafický správce procesů z Gnome

8.4. Signály

Signály jsou jednou z mála možností komunikace proces-proces, uživatel-proces a jádro-proces. Reakce na zasláný signál je záležitostí rozhodnutí programu. Většinu signálů je možné dokonce programem blokovat nebo ignorovat.

Celkem existuje asi 30 různých signálů, ale mezi ty nejčastěji používané, které můžeme procesu my, jiný proces nebo jádro poslat, patří: (v závorce číslo signálu)

- **KILL (9) – ukončit (zabít)**. Tento signál ve skutečnosti proces ani nedostane, protože je zabit přímo operačním systémem.
- **INT (2) – žádost na přerušení (interrupt)**. Již dobře známá klávesová zkratka **Ctrl+C** pošle právě tento signál běžícímu procesu. Protože je to jen žádost o přerušení, ne každý program se touto klávesovou zkratkou ukončí.

- **TERM (15) – ukončení (terminate)**. Slušný program by se měl na tento signál korektně ukončit.
- **STOP (17) – žádost na ukončení (termination)**. Žádost programu na úplné zastavení. Program by se měl korektně ukončit.
- **TSTP (18) – měkké přerušování (terminal stop)**. Signál vyslaný z terminálu procesu po stisku `Ctrl-Z`, tedy žádost o pozastavení běhu až do obdržení signálu `CONT`. Program může žádost ignorovat.
- **CONT (19) – pokračování (continuation)**. Proces pozastavený signálem `TSTP` pokračuje v činnosti. Tento signál využívá příkaz `fg`.
- **HUP (1) – žádost o restart**. Je obvykle programy vyhodnocen jako žádost o restart. Může mít však jiný význam podle OS nebo programu.
- **USR1 (30) a USR2 (31) – uživatelsky definované signály**. Nemají žádný „obvyklý“ význam. Většina programů je ignoruje.

Poznámka

Přehled všech signálů najdete `man 7 signal`, `kill -l` nebo `trap -l`.

8.4.1. kill – posílač signálu

```
kill [-<signál>] <PID>
```

Přes svůj název program `kill` bez parametru standardně posílá signál `TERM`, nikoli `KILL`. Programy ukončené `kill` jsou tedy vyzvány, aby se korektně ukončili, ne surově „zavražděny“:

```
$ sudo kill 6901
```

S parametrem umí `kill` poslat jakýkoli signál. Např. zmíněný `KILL`, `HUP` ap.:

```
$ sudo kill -KILL 6901
$ sudo kill -HUP 6901
...
```

Důležité

Signál KILL by měl zabít proces v jakémkoli stavu, ale výjimečně se proces dostane do neovladatelného stavu. Pokud např. proces čeká v mrtvém zámku (deadlock) na V/V operaci nebo zařízení, pak pomůže jen starý dobrý restart počítače.

8.4.2. killall – vylepšený posílač signálu

```
killall [-<signál>] (<PID> | <program>)...
```

Killall je vylepšená verze programu kill, která navíc:

- dovede poslat signál více procesům (opět standardně posílá TERM, ale signál se dá určit parametrem)
- umí proces zabít nejen pomocí PID, ale i názvem programu

Příklady:

```
$ sudo killall nano
$ sudo killall -TSTP nano
$ sudo killall 6956 5056 1005
$ sudo killall -HUP 6956 5056 1005
```

8.5. Niceness – priorita procesu

Niceness (ohleduplnost) neboli priorita určuje, jak ohleduplný nebo naopak bezohledný má proces být ve vztahu k ostatním procesům. Vyjádřením ohleduplnosti číselně je *nice value (hodnota ohleduplnosti)*, která může nabývat hodnot -20 (nejbezohlednější) až +19 (nejohleduplnější).

Důležité

Vyšší nice value (hodnota ohleduplnosti) = menší priorita

Tato hodnota je pouze doporučením pro jádro, nikoli příkazem a určuje výhradně prioritu času CPU. Jinými slovy, zvýšení priority nutně nezaručuje rychlejší odezvy,

protože při obrovských výkonech dnešních CPU jsou limitem spíše diskové operace nebo RAM.

8.5.1. nice

Rovnou spustit program s upravenou niceness můžeme pomocí příkazu nice, např.:

```
$ nice -n 10 /můj/program
```

Varování

Problém s nice je, že právě uvedený příklad nenastaví nice value 10, ale přičte 10 k aktuální niceness! Aktuální niceness zjistíte zavoláním `nice` bez parametrů (obvykle 0).

8.5.2. renice

Ohleduplnost již běžícího procesu nastavíte programem renice s parametrem nice value a PID. Pokud jste sudoer a proces není váš, pak s pomocí sudo:

```
$ sudo renice 10 8920  
$ sudo renice -10 8920
```

V případě renice již nastavujeme zadanou prioritu, jen nepřičítáme k aktuální hodnotě nice value.

8.6. Démoni

Démon (daemon) je relativně normální proces, který běží na pozadí (většinou) po celou dobu běhu počítače. Jakýkoli proces běžící od startu počítače by se dal nazvat démonem. Démoni však poskytují nějakou službu nebo provádí dlouhodobý úkol, který by měl být dostupný neustále. Démon běží bez ohledu na to, zda je k počítači někdo přihlášen. Poněkud nesprávně se někdy démonům říká služby.

Poznámka

V někom možná slovo démon asociuje zlé demony, ale démon znamená jen duch,

nespecifikováno, zda hodný nebo zlý.

Init systémy

Proces s PID 1 je tzv. init systém. PID 1 je spuštěn jádrem a všechny další procesy jsou potomky tohoto init procesu. Tradiční init systém se nazýval **SysV init** (nebo **System V**) a všechny pozdější init systémy podporují jeho tzv. SysV init skripty. SysV init skripty najdete v `/etc/init.d/`, ty se nashutují během bootování OS.

Ubuntu nějakou dobu využívalo vlastní init systém zvaný **Upstart** se službami konfigurovanými ve vlastní složce `/etc/init/`. Ve verzi Ubuntu 15.04 byl Upstart nahrazen init systémem **Systemd**. Debian Upstart nepoužíval nikdy a tuto složku standardně nemá.

Detailní informace o init systémech v Ubuntu a Debianu jsou v pokračování tohoto kurzu. Zde o nich budeme mluvit jen stručně.

8.6.1. Spouštěč service

Spouštěč service sloužil k ovládání SysV nebo Upstart služeb. I když bychom měli v systemd systémech preferovat příkaz `systemctl`, díky zpětné kompatibilitě systemd nám `service` funguje nadále pro všechny tři možné „formáty“ služeb:

- staričké tradiční System V init skripty
- Upstart joby
- systemd services (služby)

Nemusíme tedy vědět, jak je služba definována a vždy se můžeme spolehnout na obecnou syntaxi:

```
service <démon> <příkaz> [volby]
```

Démon odpovídá jménu souboru skriptu v `/etc/init.d/` nebo Upstart jobu v `/etc/init/`. Pokud existuje démon stejného jména v obou složkách, přednost má

Upstart. Nejvyšší „prioritu“ má systemd - pokud existuje sytemd service stejného jména, použije se.

Podporované příkazy jsou obvykle minimálně `start`, `stop`, `status`, ale i `restart` a další, podle „chytrosti“ a druhu démona:

```
$ sudo service sshd start
$ sudo service sshd stop
$ sudo service apache2 restart
```

Poznámka

System V init skripty nejsou konfigurační soubory jako v Upstart a Systemd, ale „obyčejné“ spustitelné skripty (nejčastěji v Bashi). Příkazy pro service jsou vlastně parametry příkazové řádky těmto skriptům. Šlo by tedy místo `sudo service rsync restart` zavolat `sudo /etc/init.d/rsync restart`, ale preferovaný způsob je `service`.

Zajímavostí, kterou si musíme dobře uvědomit je, že démon není připojen k žádnému terminálu nebo standardním V/V (STDIN, STDOUT, STDERR). Nevidíme tedy, co vypisuje (pokud neloguje), ani ho nemůže řídit jinak, než V/V zařízením, proměnnými prostředí, signály a příkazy pro service.

Výpis aktivních démonů a jejich stavu zjistíte (zkráceno):

```
$ service --status-all
[ + ] acpid
[ + ] anacron
[ + ] apache2
[ - ] apparmor
[ ? ] apport
[ + ] avahi-daemon
[ ? ] binfmt-support
[ + ] bluetooth
[ - ] brltty
[ + ] console-font
[ + ] console-setup
...
```


Význam symbolů:

- `+` démon běží
- `-` démon neběží
- `?` neznámý stav (démon nepodporuje příkaz status)

(Pokud chcete vidět i PID, zkuste spíše *initctl list*.)

Případně dotaz na stav konkrétního démona (odpovídá démon sám, výpisy se liší):

```
$ sudo service acpid status
acpid start/running, process 1343
```

Některé demony byste také mohli poznat v ps nebo top, podle toho, že se někdy jmenují *<něco>d* (sshd, httpd ap.), ale nelze na to spoléhat.