Operační systémy - cvičení

GNU/Linux

Kniha pro elektronické čtečky
verze pro PC

Autor: Ivo Kostecký
Odborná vedoucí: Ing. Lenka Závodná

Operační systém GNU/Linux

Spuštění OS

O systému

Linux, jehož domovská stránka je na univerzitě v Helsinkách prozradí, že jde o implementaci specifikace POSIX s rozšířeními BSD a System V. Jinými slovy: jde o Unix implementovaný úplně od začátku. Otcem celého projektu je Linus Torvalds, student helsinské univerzity. Původně si chtěl jenom zkusit, jak funguje chráněný režim procesoru i386, a dnes vede vývoj verze 2.1 plně funkčního operačního systému, který už byl přenesen i na další platformy.

Síťový operační systém Linux je stále více vyhledáván pro svoji stabilitu a bezpečnost, volné šíření, možnosti konfigurace. Umožňuje vlastní modifikace systému a syntaxe je příbuzná moderním programovacím a skriptovacím nástrojům jako je například jazyk C, Java, PHP a SQL.

Výhodou Linuxu je kromě velmi příznivého poměru cena/výkon, univerzálnost operačního systému velice dobře pracujícího v síťovém prostředí. Podpora TCP/IP, IPX, AppleTalk a dalších umožňuje nasazovat počítače vybavené Linuxem jako síťové servery. Není problém se sdílením tiskáren, disků a dalších periferií. Množství hardwaru podporovaného Linuxem je obdivuhodné, zvláště když si uvědomíme, že většina ovladačů nepochází od výrobců hardwarových zařízení, ale od nezávislých programátorů. Možnost simultánní práce více uživatelů na jednom počítači je samozřejmou vlastností všech Unixů, stejně jako ochrana dat pomocí přístupových práv.

Po nabootování běží v systému jen jádro (kernel). Jádro pak samo spustí první uživatelský program, kterému vyhradí potřebné prostředky a předá mu řízení. Běh programu je pak jádrem usměrňován jedině, když se musí obsloužit i jiné programy, nebo když je nutné obsloužit periferii. Program může po jádru vyžadovat nějakou systémovou službu, o kterou se přihlásí systémovým voláním.

Běžícím aplikačním programům se říká procesy. Proces se skládá z identifikačních údajů v jádře, kódu v operační paměti a z dalších dat programu. Procesy mohou pomocí systémového volání vytvářet další procesy a stanovovat, jakým programem budou obsluhovány. Každý proces a datový soubor má svého majitele, kterým může být kterýkoliv registrovaný uživatel.

První proces po startu systému se nazývá vždy init a jeho majitelem je root. Proces init pak vytváří řadu dalších procesů k poskytování komplexních systémových služeb, které běží po celou dobu života systému - tzv. démony (např. démon cron, který spouští aplikace podle naplánování). Většina démonů obsluhuje síťové služby, "vyčkává" na síťovém portu až přijde síťový paket a předává ho jádru. Jádro požadavek "přeloží" a vrátí ho zpět démonu, který si ho sám vyhodnotí a určitým způsobem se pak zachová (může třeba navázat komunikaci s jinou aplikací). Démon většinou pro provedení instrukcí vytvoří pomocí systémového volání nový proces a sám pak znovu vyčkává na další instrukce.

Uživatelské prostředí

Uživatel má dvě možnosti jak celý systém ovládat a dávat příkazy – CLI nebo GUI. CLI je zkratka pro *Command Line Interface* – jinými slovy ovládání přes příkazovou řádku. GUI je zkratka *Graphical User Interface* – tudíž tolik oblíbené grafické rozhraní ve kterém klikáme na ikony a pracujeme s okny. GNOME, KDE a Xfce jsou názvy

uživateli oblíbených GUI. Linux, s GUI a samotnými aplikacemi můžeme označit jako linuxovou distribuci pro snadnou instalaci.

Každá distribuce nainstaluje své hlavní grafické prostředí (někdy máte při instalaci možnost výběru). Tudíž pokud si vyberete např. distribuci Ubuntu, vyberete si tím i grafické prostředí GNOME. Naopak distribuce Kubuntu používá grafické prostředí KDE. Dalo by se říci, že se jedná o Ubuntu s grafickým prostředím KDE.

GNOME v distribuci Fedora 10

KDE v distribuci OpenSuSE 11

Xfce v distribuci Mint 10

Základním pracovním nástrojem uživatele UNIXu je shell, program sloužící ke spouštění dalších programů. Shell se aktivuje vždy, když se uživatel přihlásí k unixovskému stroji na konzoli, po síti nebo prostřednictvím terminálu. Používá se k zadání povelů pomocí klávesnice. V grafickém režimu lze spouštět programy pomocí příslušných ikon. Složitější textové příkazy shellu je vhodné napsat do souboru a ten pak spouštět jako kterýkoliv jiný program (program se spouští prostým zadáním jeho jména). Činnost programů lze ovlivnit úpravou jejich konfiguračních souborů.

Aplikace

Existuje velké množství programů určených pro Linux a mezi nimi najdete i známé softwarové kousky, které možná používáte i ve Windows. Jde o software, který jeho vývojáři vyvíjejí jak pro operační systém Linux, tak i pro operační systémy Windows či další. Těmito kousky jsou např. internetové prohlížeče Mozilla Firefox a Opera, emailový klient Mozilla Thunderbird, kancelářský balík OpenOffice.org, video a audio přehrávač VLC Media Player, grafický editor GIMP, komunikátor Skype nebo některé hry.
Pokud nejste vázáni na nějaký profesionální (placený) program či hru, tedy už nyní si vystačíte s tzv. free (zdarma) programy a hrami, jistě nebudete mít - ohledně softwaru - žádný problém s přechodem k Linuxu. Stejně jako celý operační systém Linux, tak i drtivá většina softwaru pro tento operačním systém je zdarma a ve většině případů i česky.

Pokud vyžadujete nějaký konkrétní software či hru na Windows, kterého se nemůžete vzdát, je tu alternativa "emulátoru" Wine. Tento program simuluje prostředí operačního systému Windows, díky němuž je umožněna instalace programů a jejich spuštění. Wine nemusí být vždy schopen každý program a především hru bez

problémů nainstalovat a spustit. Na stránkách Wine si můžete vyhledat informace o konkrétním programu určený pro Windows. Pokud aplikaci v seznamu najdete, pak u ní bude napsáno, ve které distribuci byla aplikace vyzkoušena, s jakou verzí programu Wine a hodnocení instalace a běhu programu od nejlepší Platinum, Gold, Silver až po nejhorší Garbage.

Dále existují aplikace vycházející z programu Wine: CrossOver Linux, CrossOver Games a Cedega.

Pokud by vám z nějakého důvodu nevyhovovala linuxová alternativa, ani "emulátor" Wine a placené programy postavené na jeho základě, jsou tu dvě další možnosti, jak se dostat k Linuxu a nepřijít o nepostradatelné programy z Windows.

- Instalace distribuce Linuxu vedle vašeho stávajícího systému.
- Virtualizace celého operačního systému v Linuxu.

Je možné stáhnout program ze stránek a pak ho nainstalovat, ale proč to dělat složitě, když to jde jednoduše. Snad ve všech běžně používaných distribucích najdete jeden program nazývající se **správce balíků**. Tento program vám zobrazí seznam s tisíci aplikacemi, utilitkami a ovladači, které si jednoduše vyberete a necháte stisknutím jednoho tlačítka nainstalovat. O ostatní se správce balíků postará sám. Stáhne program, vše potřebné z internetu a nainstaluje. Jedinou podmínkou pro takto snadnou instalaci software je nutné připojení k internetu.

Než se pustíte do bezhlavého hledání a instalací softwaru, uvědomte si, jakou linuxovou distribuci používáte. Jedná se především o grafické prostředí distribuce - GNOME a KDE. Tato dvě velká grafická prostředí mají několik "svých" typických programů, které je samozřejmě možné nainstalovat a používat také v druhém grafickém prostředí. Musíte však počítat s tím, že vzhled programu nemusí do druhého grafického prostředí příliš zapadat.

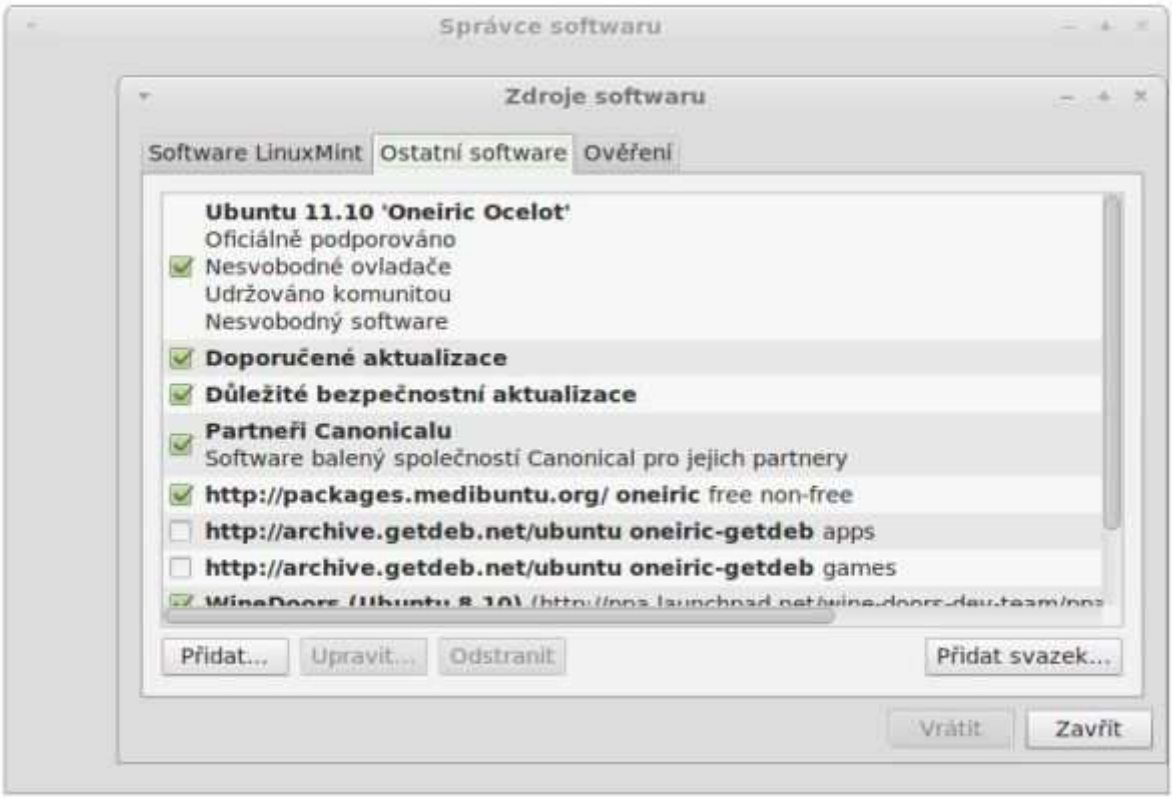
Jedná se především o programy, jako je např. kecálek Pidgin (GNOME) a Kopete (KDE), vypalovací program Brasero (GNOME) a K3b (KDE) či hudební přehrávače Rhythmbox (GNOME) a Amarok (KDE). Osobně doporučuji používat software vyvíjený pro dané grafické prostředí. Samozřejmě je to jen doporučení, pokud se rozhodnete používat aplikaci z KDE v GNOME nebo naopak, nebude v tom žádný problém.

Jak poznáte, který software je určen pro to či ono grafické prostředí? Pokud instalaci provedete pomocí správce balíků, pak bude při výběru programu ve správci balíků zobrazen popis aplikace. Uvidíte v popisu, zda je aplikace vyvíjena pro konkrétní grafické prostředí, nebo zda je "neutrální".

Repozitáře

Odkud se software prostřednictvím správce balíků stahuje? Jedná se o již zmiňované repozitáře. Jedná se o úložiště, kam je vkládán software ve formě balíků, včetně všech částí linuxové distribuce. Počet programů se pohybuje ve stovkách až tisících. Repozitáře obsahují veškerý software, od programů pro kancelářskou práci, přes grafické editory až po linuxové hry. V Linuxu si prostřednictvím správců balíků seznam programů z těchto serverů zobrazíte. Stažení vybraného programu a instalaci provede správce balíků za vás.

Pokud není vámi požadovaný software ve správci balíků, pak je možné přidat tzv. repozitáře třetích stran. Jedná se o úložiště softwaru, který nemusí být prověřené výrobcem systému. Také se může jednat o "neoficiální" repozitář, kde jsou uloženy aplikace, které z licenčních důvodů nemohou být v oficiálních repozitářích. Většina velkých distribucí má nástroj pro přidávání takových repozitářů ve svých ovládacích centrech. V některých případech není možné mít počítač připojený k internetu. Stejně jako repozitáře třetích stran je možné si tedy přidat jako repozitář DVD / CD distribuce.

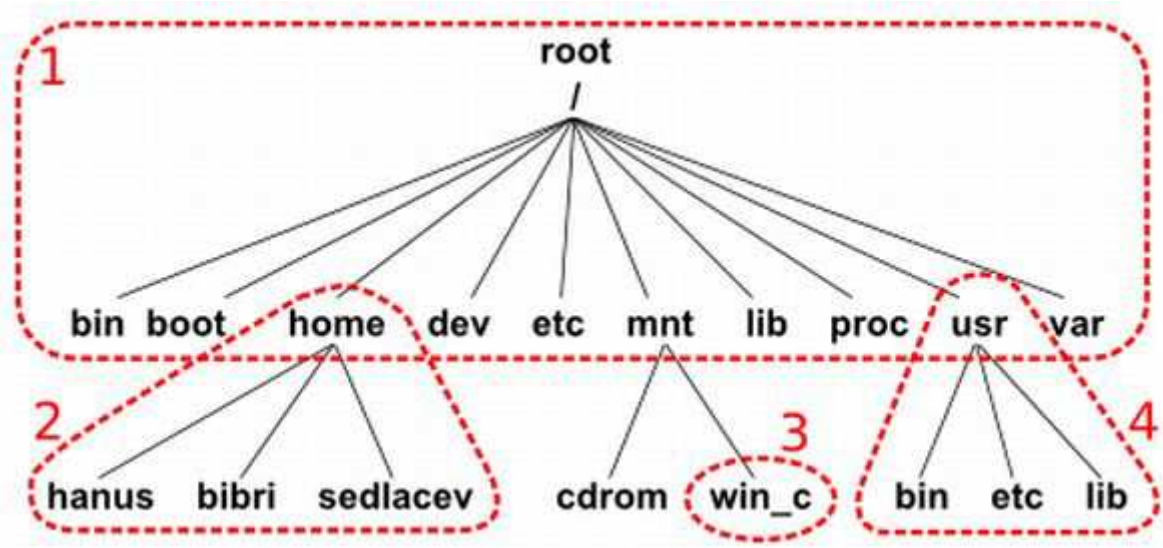


Konfigurace

Adresářová struktura

Celá adresářová struktura v Linuxu začíná tzv. kořenovým adresářem, který se označuje symbolem /. Často se také používá anglický výraz root (stejně se označuje i superuživatel-administrátor). Jeho obsah můžete vidět na obrázku. Nacházejí se zde hlavní systémové adresáře.

Pro lepší představu si můžeme adresářovou strukturu znázornit pomocí schématu. Jak jinak vše vychází z kořene. Oblasti 1-4 vymezují diskové oddíly (partitions). Kořen se svými podadresáři má vyhrazen samostatný diskový oddíl. Další diskový oddíl bývá obvykle vyhrazen pro adresář /home (do něj se ukládají data uživatelů, viz dále), často se vyhrazují diskové oddíly pro adresáře /usr, /var, /boot, popř. další.



Připojování disků a oddílů

Diskové oddíly jsou připojovány jako určité adresáře – pevné disky obvykle při startu systému a výměnná média za běhu. Předpis, kde se jaký oddíl má připojit, je uložen v souboru *fstab* v adresáři */etc/;* např. zápis */dev/hda6 / ext3* znamená, že 6. oddíl primárního IDE disku (hda6) se připojí jako kořen (/) na souborovém systému ext3. A v případě */dev/hda8 /home xfs* bude 8. oddíl téhož disku připojen do adresáře */home* se souborovým systémem *XFS*.

Diskové oddíly, na nichž jsou nainstalovány nelinuxové operační systémy, jsou obvykle připojovány do adresáře */mnt* jako adresáře označené příslušným názvem (např. windows, win_c, win_d apod.). Do stejného adresáře jsou připojovány i výměnná média – CD a DVD-ROM mechaniky, vypalovačky, disketové jednotky (adresáře se pak mohou jmenovat cdrom, dvd, floppy atp.). Do jakého konkrétního adresáře se dané výměnné médium či oddíl jiného OS připojí, je rovněž uloženo v souboru fstab.

Adresář */home* je vyhrazen pro domovské adresáře uživatelů a bývá vhodné vyhradit mu samostatný oddíl na disku již při instalaci. Důvod je praktický – jakmile je potřeba upgradovat či přeinstalovat systém (má svůj diskový oddíl), data uživatelů zůstanou nedotčena. V /home jsou podadresáře shodné s přihlašovacími jmény uživatelů. Jeden uživatel obvykle nemá právo zápisu do adresářů jiných uživatelů (lze zakázat i čtení aj.).

Každý soubor a adresář (jde vlastně o speciální soubor) má v adresářové struktuře svou unikátní absolutní cestu, pomocí které jsou jednoznačně identifikovány. **Absolutní cesta** vždy začíná kořenem a končí názvem konkrétního souboru či adresáře. Tak kupříkladu */home/kindle/Dokumenty/clanek.txt* jednoznačně určuje soubor *clanek.txt* v adresáři Dokumenty uživatele *kindle*.

Oproti absolutní cestě známe ještě **relativní cestu**. To je poloha adresáře či souboru oproti místu (adresáři), ve kterém se právě nacházíme. Důsledkem je, že určení cílového souboru není jednoznačné, ale záleží na aktuální poloze v adresářové struktuře. Relativní cestu od absolutní rozeznáme tak, že nezačíná symbolem /.

Pokud jsme tedy v adresáři */home/kindle/Video*, pak se k našemu článku dostaneme pomocí absolutní cesty */home/kindle/Dokumenty/clanek.txt*, nebo pomocí relativní cesty *../Dokumenty/clanek.txt*.

Existují určité speciální symboly pro usnadnění práce s cestami. Symbol ~ nahrazuje cestu do domovského adresáře přihlášeného uživatele, takže místo absolutní cesty bychom mohli napsat *~/Dokumenty/clanek.txt* (pokud budete ale jako uživatel *bibri*, bude mít ~ význam */home/bibri/*).

V relativní cestě lze použít symbol .. (dvě tečky) pro označení nadřízeného adresáře a symbol . (jedna tečka) pro označení aktuálního adresáře. Takže *./clanek.txt* označuje soubor *clanek.txt* v aktuálním adresáři a *../../clanek.txt* soubor clanek.txt o dva adresáře výše.

Skryté soubory a adresáře se v Linuxu jednoduše vytvářejí tak, že jejich jméno začnete tečkou. Pokud tedy soubor *clanek.txt* přejmenujete na *.clanek.txt*, stane se skrytým a pokud je správce souborů nastaven tak, aby nezobrazoval skryté soubory, pak již tento soubor neuvidíte. Stejné platí pro adresáře.

Systémové adresáře

V linuxovém systému existuje množství adresářů, které vznikají bez přímého přičinění běžných uživatelů; jde tedy o adresáře nacházející se jinde než v adresáři */home*. Zmíníme se zde alespoň o těch nejdůležitějších a nejzajímavějších:

- **/bin** – Adresář obsahuje několik málo základních programů potřebných pro start systému, jež mohou být využity i běžnými uživateli (po startu).
- **/sbin** – Obdobně jako adresář /bin, ale zde umístěné programy nejsou určeny pro běžné uživatele, nýbrž administrátora (superuživatele root).
- **/boot** – Zde jsou uloženy soubory zavaděče a jádra potřebné při spouštění systému.
- **/dev** – Soubory zařízení; jde o speciální soubory, které umožňují uživatelskou komunikaci se zařízeními systému (např. připojené diskové oddíly, sériové porty, zvuk, obraz aj.).
- **/etc** – Konfigurační soubory globální, systémové; další konfigurační soubory jednotlivých aplikací naleznete v domovských adresářích uživatelů.

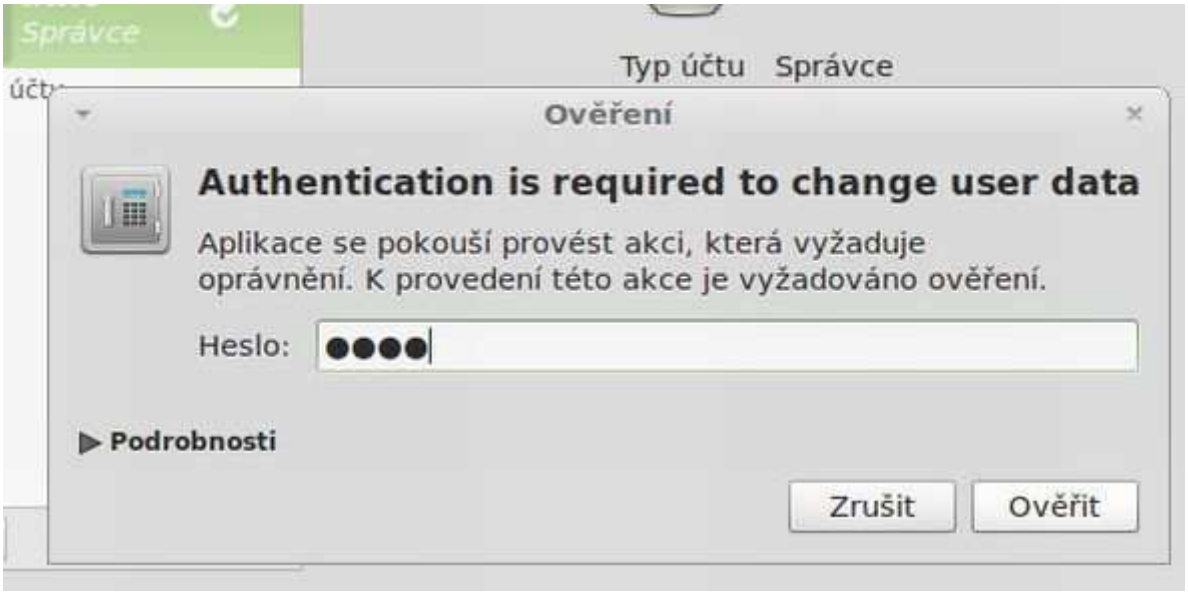
- **/home** – Domovské adresáře jednotlivých uživatelů obsahující uživatelská data a uživatelskou konfiguraci.

- **/lib** – Sdílené knihovny vyžadované programy v kořenovém adresáři.
- **/mnt**, **/media** – Adresář určený pro dočasně připojované systémy jako disketová jednotka, CD-ROM aj.
- **/opt** – Instalují se sem některé nestandardní součásti systému, např. OpenOffice.org aj.
- **/proc** – Poskytuje informace o systému. Obsahuje pseudosouborový systém, který nereprezentuje strukturu dat na disku, ale jde o strukturu vytvořenou v paměti umožňující přístup k informacím o procesech a nastavení systému a jádra.
- **/root** – Domovský adresář administrátora root; obvykle není přístupný ostatním uživatelům.
- **/tmp** – Prostor vyhrazený běžícím programům pro ukládání dočasných souborů.
- **/usr** – Tento adresář bývá velmi objemný, protože jsou do něj instalovány všechny aplikace; Najdeme zde *bin, sbin, etc, lib*; zvláštní význam mají adresáře */usr/share*, kam jsou umisťovány aplikacemi sdílené soubory, a */usr/local*, kde jsou instalovány aplikace "mimo" distribuci (např. při kompilaci).
- **/var** – Obsahuje data měněná za normálního běhu systému; jsou zde kupříkladu adresáře pro logování, tiskové a poštovní fronty, dlouhodobější tmp adresář aj.

Superuživatel

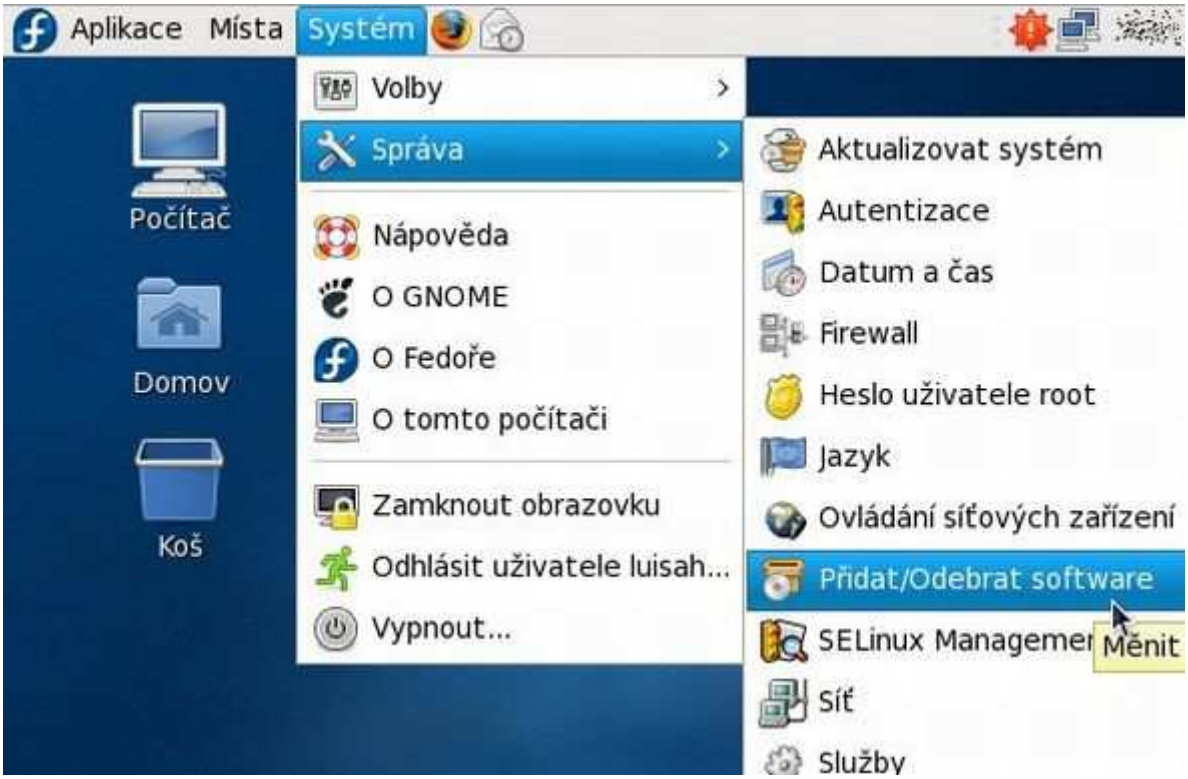
Pokud chce běžný uživatel provést úkon (odstranění, úpravu souborů, složky), který může mít vliv na správný chod systému, musí jej provést jako superuživatel root (správce). Systém se tím chrání před neodborným zásahem běžného uživatele.

Pokud se stanete rootem, můžete dělat v systému cokoliv. Rootem by se měl stát vždy správce systému, to znamená upravovat a nastavovat systém by měl vždy jen člověk, který ví, co dělá. Tato významná bezpečnostní složka v Linuxu odděluje samotný systém, jeho soubory a adresáře od uživatele a jemu přidělenou domovskou složkou. Oddělena je pomocí správcovského hesla neboli hesla roota, které jste zadávali při instalaci.



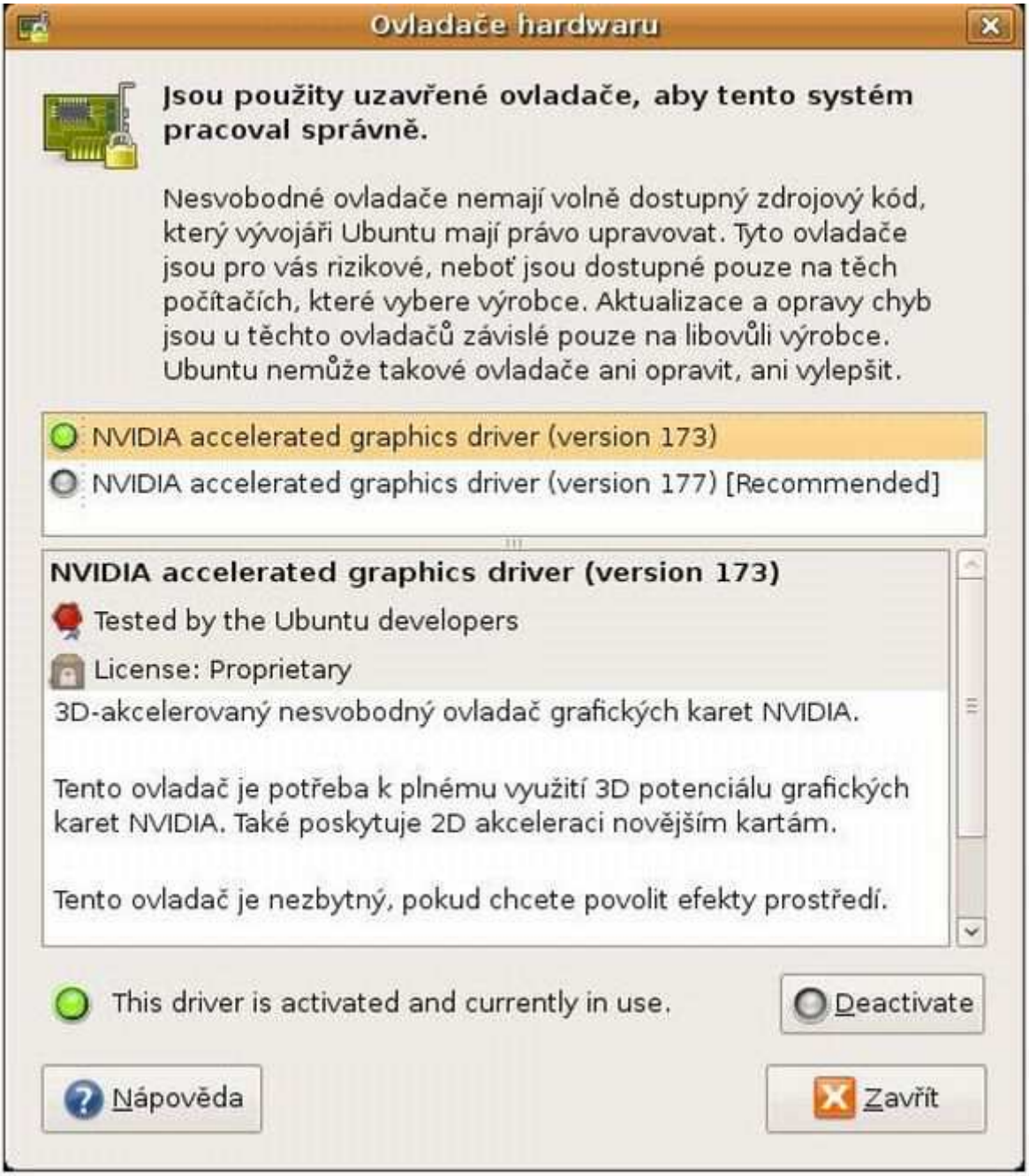
Nastavování

Některé distribuce obsahují typická ovládací centra, ve kterých najdete nejrůznější možnosti nastavení systému. S typickými ovládacími centry, které znáte např. z Windows, se setkáte především v distribucích jako Mandriva Linux, OpenSuSE. Distribuce Ubuntu, Fedora, především s grafickým prostředím GNOME, taková klasická ovládací centra neobsahují. Veškerá nastavení bývají v hlavní nabídce.



Ovladače

Po instalaci distribuce a prvním spuštění systému je v některých distribucích systémem oznámena možnost instalace nesvobodných ovladačů, v jiných se instalace provádí manuálně (pomocí Správce balíků). *Nesvobodné ovladače jsou drivery, které jsou vyvíjeny přímo výrobcem a které nemohou vývojáři distribucí nijak ovlivňovat. Toto se týká především grafických karet NVIDIA a ATI.*


Ovladače hardwaru
Jsou použity uzavřené ovladače, aby tento systém pracoval správně.
Nesvobodné ovladače nemají volně dostupný zdrojový kód, který vývojáři Ubuntu mají právo upravovat. Tyto ovladače jsou pro vás rizikové, neboť jsou dostupné pouze na těch počítačích, které vybere výrobce. Aktualizace a opravy chyb jsou u těchto ovladačů závislé pouze na libovůli výrobce. Ubuntu nemůže takové ovladače ani opravit, ani vylepšit.
NVIDIA accelerated graphics driver (version 173)
NVIDIA accelerated graphics driver (version 177) [Recommended]
NVIDIA accelerated graphics driver (version 173)
Tested by the Ubuntu developers
License: Proprietary
3D-akcelerovaný nesvobodný ovladač grafických karet NVIDIA.
Tento ovladač je potřeba k plnému využití 3D potenciálu grafických karet NVIDIA. Také poskytuje 2D akceleraci novějším kartám.
Tento ovladač je nezbytný, pokud chcete povolit efekty prostředí.
This driver is activated and currently in use.
Deactivate
Nápověda
Zavřít

Připojení k linuxovému serveru

Server

Popis

Jsou v zásadě dva způsoby, jak se lze něco udělat na vzdáleném linuxovém počítači podle toho, jakým protokolem smím přistupovat:

- přes FTP
- přes SSH

Práce přes FTP je velice omezená a prakticky se tím dají pouze kopírovat soubory a nastavovat jim práva. Přístupem přes SSH se dá plnohodnotně na vzdáleném serveru pracovat se vším všudy, samozřejmě podle toho, jaká mám přidělena práva. To, jako metodou můžu přistupovat, určuje správce serveru. Standardní webhostingy podporují pouze FTP. V následujících úkolech budeme pracovat přes SSH.

K připojení přes SSH potřebujete znát:

- adresu počítače
- své uživatelské jméno
- své heslo
- výjimečně port

Adresa počítače může být jak IP adresa (jak lokální, tak veřejná), tak doménové jméno (SSH si jej přeloží).

Jestliže jsem obdržel výše zmínění informace, potřebuji na připojení program, který se jmenuje SSH klient. Existuje jich více, nejčastěji uváděné jsou tyto:

- Putty
- řádkový program SSH v prostředí CygWin

Připojení

PuTTY je klient protokolů SSH, Telnet, rlogin a holého TCP. Dříve byl dostupný jen pro Windows v současnosti je dostupný i pro různé UNIXové platformy. Původním autorem je Simon Tatham z Cambridge. PuTTY je svobodný software, šířený pod licencí MIT.

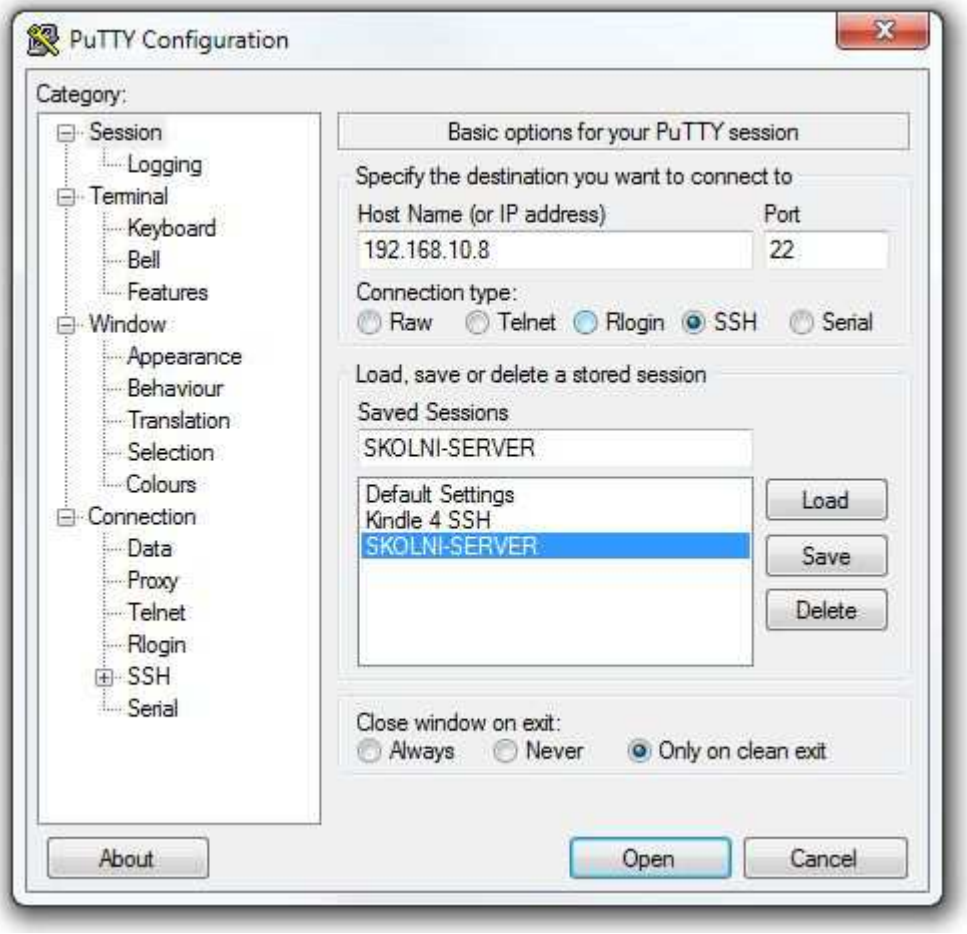
SSH je v informatice označení pro zabezpečený komunikační protokol v počítačových sítích, které používají TCP/IP. SSH byl navržen jako náhrada za telnet a další nezabezpečené vzdálené shelly (rlogin, rsh apod.), které posílají heslo v nezabezpečené formě a umožňují tak jeho odposlechnutí při přenosu pomocí počítačové sítě. Šifrování přenášených dat, které SSH poskytuje, slouží k zabezpečení dat při přenosu přes nedůvěryhodnou síť, jako je například Internet.

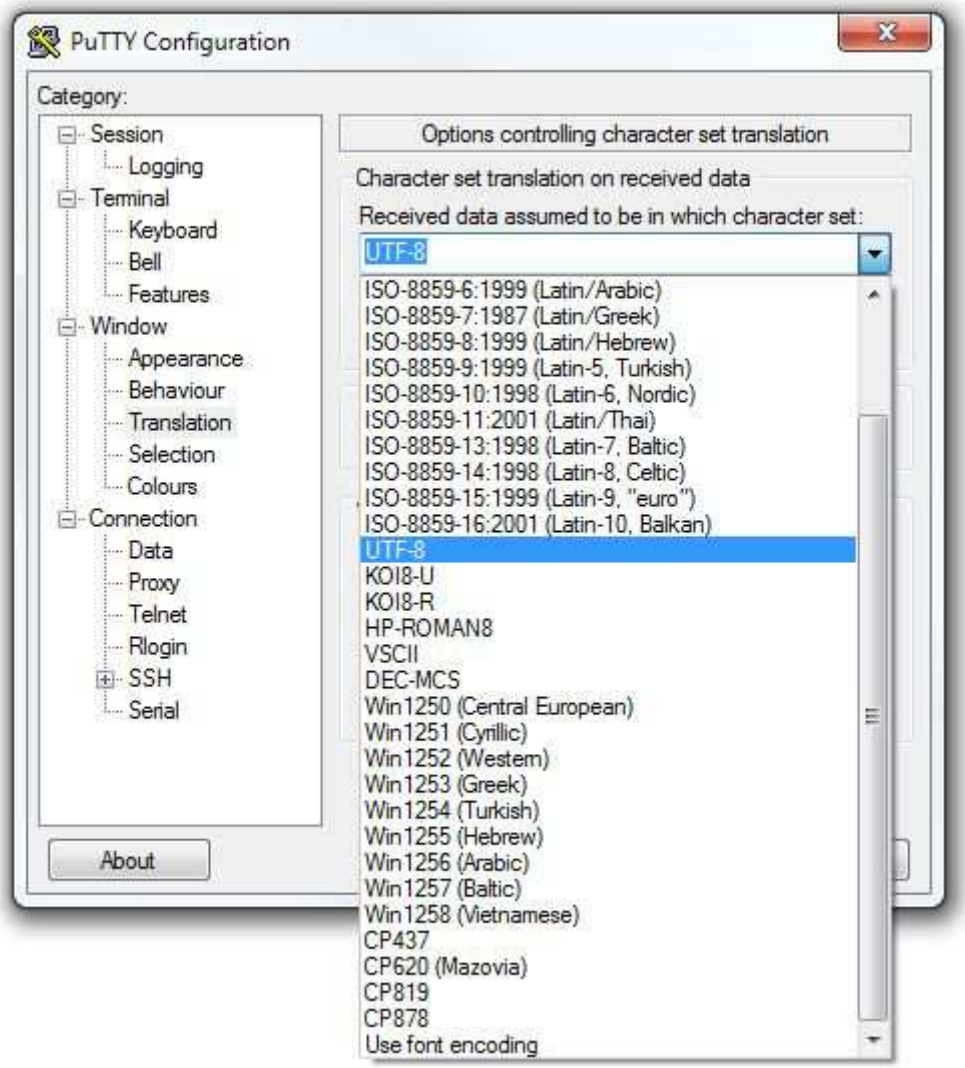
Připojení ve škole

Pod OS Linux spustit terminál a pomocí příkazu:

```
ssh loginname @192.168.10.8
```

Z OS Windows přes Putty.exe, položka 'Session', připojit se na IP 192.168.10.8, typ SSH a port 22.





Změna hesla

Po přihlášení je možné si změnit heslo k účtu. To se provádí příkazem:

```
passwd jmeno_uzivatele nove_heslo
```

Jméno uživatele není nutné zadávat pokud jde o vlastní účet na kterém jsme v dané chvíli přihlášení. A přidám také poznámku, že v Linuxu se nevypisují napsané znaky na obrazovku ani pod skrytými znaky v podobě teček i když se heslo zapisuje!

Příkazová řádka BASH

Práce se soubory a adresáři

Příkazová řádka

Bourne Again Shell

Shell je program, který v unixových operačních systémech vytváří základní textové rozhraní mezi uživatelem a operačním systémem.

Prompt

Prompt je několik znaků na začátku řádku, za kterými stojí kurzor. Toto uskupení se nazývá výzva (anglicky prompt). Před kurzorem se mohou zobrazovat různé informace od jména přihlášeného uživatele, přes jméno počítače, aktuální adresář, čas až po verzi Bashe. Když se v Bashi zobrazí prompt, je možné zadávat příkazy. Obecně to, co zadáme na řádku jako první, je chápáno jako příkaz.

Zadáme příkaz, pokud neodpovídá žádnému internímu příkazu, začne Bash hledat spustitelný soubor (program) stejného jména na disku. Některé příkazy můžeme použít jen takto samostatně, většina jich ale vyžaduje parametry.

Parametr je prostředek k tomu, aby uživatel svůj požadavek upřesnil (např. s čím a jak má příkaz pracovat).

Více argumentů od sebe oddělujeme mezerou.

```
ls /home
```

```
ls /home /var/spool
```

Vypíše obsahy adresářů */home* a */var/spool*

Argumenty nemusíme zadávat jen jeden argument. Více argumentů od sebe oddělujeme mezerou. Před argumenty lze většinou uvést ještě různé volby, které mění nebo rozšiřují funkcionalitu příkazu.

Voleb můžeme uvést více současně. Volby v krátké (jednoznakové) podobě:

```
ls -l -a
```

Zkrácený zápis:

```
ls -la
```

Volby dlouhé:

```
ls --format=long –all
```

Manuálové stránky

Komu nestačí nápověda získaná pomocí --help, může pro většinu příkazů využít manuálové stránky. Jejich vyvolání je snadné. Stará se o ně program man, jehož parametrem je jméno manuálové stránky, což je zpravidla zároveň jméno programu. Po zadání man ls tak dostaneme manuálovou stránku programu ls. V záhlaví vidíme stručnou definici programu, následuje podrobná syntaxe. I zde platí, že parametry uvedené v hranatých závorkách nejsou povinné.

Program man, parametr - jméno manuálové stránky, jméno programu.

```
man ls
```

Obsahuje:

- definici programu, podrobnou syntaxi
- podrobný popis daného programu
- odkazy na jinou dokumentaci ke zkoumanému problému
- přehled voleb
- odkazy na ostatní manuálové stránky vztahující se k tématu

Pro zobrazení se používá příkaz *less*. Po manuálové stránce se pohybujeme šipkami, klávesami [PgUp] a [PgDn], ukončení [q], nápověda [h].

Nápověda k příkazu

```
--help
```

Echo

Echo znamená v angličtině ozvěnu. Příkaz vypíše na standardní výstup zadané argumenty, oddělí je mezerou a zakončí znakem nový řádek.
Pokud bychom chtěli, aby echo chápalo více slov jako jeden argument, můžeme je uzavřít do uvozovek ("") nebo apostrofů ('). Uvozovky totiž dovolují Bashi provádět speciální znaky (umožňuje jejich expanzi). Co je v apostrofech, bude vypsáno přesně tak, jak to zadáme.

Přepínače echo

- -n
    - zamezí přidání znaku nový řádek na konec výstupu
    - z více příkazů echo lze poskládat jeden řádek výstupu.
- -e
    - echo bude rozumět speciálním skupinám znaků, které jsou uvozeny zpětným lomítkem.

- Speciální skupiny znaků
    - \b - zpětné mazání
    - \a - výstraha (zvonek)
    - \f - nová stránka (FF)
    - \n - nový řádek

Clear

Tento příkaz vymaže obrazovku terminálu.

Find

Příkaz rekurzivně prochází strom podadresářů, který začíná zadanou cestou a hledá v něm soubory splňující zadané podmínky. Implicitní cestou je pracovní adresář.

```
find [cesta...] [výraz]
```

```
find . -name retezec
```

Hledá soubor zadaného názvu. Najde-li jej, vypíše jej i s cestou. Pokud ne, nevypíše nic.

```
find /home/jirka /home/lada -size +10000c -atime +5 -ok rm {}\;
```

Hledá v adresářích */home/lada* a */home/jirka*, smaže všechny soubory větší než 10000 bajtů a poslední přístup k nim byl před více než pěti dny.

Expr

Expr slouží k vyhodnocení zadaného matematického výrazu (expression, tj. anglicky výraz). Každý operand a operátor musí být uveden jako oddělený argument.

```
expr 5 + 10
```

```
15
```

Práce s adresáři

Příkaz **pwd** zobrazí cestu a název právě aktuálního adresáře. Pokud jsme v domovském adresáři uživatele bfu, příkaz pwd zobrazí:

```
pwd
```

```
/home/bfu
```

- Oddělovačem jmen adresářů je znak `/`
- Rozlišujeme cestu absolutní a relativní
- Každý adresář obsahuje položky `.` (tento adresář) a `..` (nadřazený adresář)

Nový adresář vytvoříme příkazem **mkdir**. Takto se vytvoří nový podadresář texty v aktuálním adresáři:

```
mkdir texty
```

Již existující, prázdný adresář lze smazat příkazem **rmdir** adresář.

Pro změnu aktuálního adresáře použijeme příkaz **cd**. Lze použít absolutní i relativní cestu, takže do podadresáře texty aktuálního adresáře bfu lze přejít z tohoto adresáře jednou z těchto možností:

```
cd texty

cd /home/bfu/texty

cd ./texty
```

Příkaz cd bez parametru způsobí přechod do domovského adresáře uživatele.

Výpis souborů v adresáři, získáme příkazem **ls**. Bez parametrů zobrazí stručný seznam názvů souborů v adresáři. Při použití parametru **-l** získáme detailní výpis, přidáme-li ještě parametr **-a**, ve výpisu se objeví i skryté soubory. Lze samozřejmě uvést specifikaci souborů které se mají zobrazit, včetně cesty. Pokud není uvedena, **ls** vypíše všechny soubory v aktuálním adresáři. Příklady:

```
ls p*.txt     #vypíše soubory které začínají písmenem p

ls p??        #vypíše soubory, které začínají p a za ním jsou

              #právě 2 znaky (např. p01, p02)

ls /dev       #vypíše obsah adresáře /dev

ls -l -R      #vypíše adresář včetně podadresářů
```

- `-a` jsou vypsány všechny soubory i ty, které jsou normálně neviditelné, tedy ty jejichž jméno začíná tečkou.
- `-l` dlouhý formát výpisu (obsahuje práva přístupu, vlastníka, velikost, čas poslední změny atd.)
- `-d` jsou vypsány informace o adresáři, nikoli o jeho obsahu. (k výpisu práv pouze o adresáři: `ls -dl adresar`)
- `-p` adresáře jsou označovány připojením znaku / za jejich jméno.
- `-t` soubory jsou na výstupu tříděny podle času poslední modifikace (nejnovější je poslední).
- `-u` soubory jsou na výstupu tříděny podle času posledního přístupu.

K této skupině příkazů snad ještě můžeme připojit příkaz **du**, který vypíše zaplnění diskového prostoru.

Práce se soubory

- Jméno souboru smí být dlouhé nejvýše **255 znaků**
- Na tuto délku je zkracováno bez varování
- Může obsahovat libovolné znaky. Komplikace způsobují řídicí znaky shellu (?, *, [,], \, všechny druhy apostrofů a uvozovek);
- V kořenovém adresáři nesmí být soubor jména `/`
- Rozlišují se malá a velká písmena
- Zvláštní význam mají jména souborů začínající znakem "." (tečka), např. `.profile`.
- S každým souborem jsou spjaty tři časové údaje. První časový údaj obsahuje informaci o době vytvoření souboru, druhý o době jeho poslední modifikace a třetí o době, kdy byl naposledy zpřístupněn.

Soubor lze zkopírovat příkazem **cp**. Lze kopírovat do souboru jiného jména, nebo soubor pod stejným jménem do jiného adresáře. Například:

```
cp /mnt/floppy/01.jpg /home/bfu  #takto zkopírujeme soubor
                                 #z diskety do adresáře uživatele bfu
cp 01.jpg kopie.jpg              #a nakopírovaný soubor si pak uložíme jako
kopii
```

Zvláště pokud kopírujeme hromadně více souborů, bude užitečný parametr **-i** , při jehož použití bude nutné potvrdit každý přepisovaný, již existující soubor. Jestliže ve výstupním adresáři již existuje soubor 01.jpg, pak po příkazu

```
cp -i /mnt/floppy/*.jpg
```

se zobrazí dotaz:

```
cp ovewrite '01.jpg'?
```

Aby se soubor přepsal, je třeba ho potvrdit stiskem klávesy *y*. Pro zachování starší verze přepisovaného souboru lze použít parametr **-b**, který vytvoří archivní kopii přepisovaného souboru.

Obsah adresáře, včetně podadresářů lze zkopírovat s pomocí parametru **-r**.

```
cp -r . /tmp
```

Podobnou syntax má příkaz **mv**, který soubor přenese do jiného adresáře, užívá se i pro přejmenování souboru.

```
mv 01.jpg nejlepsi.jpg    #obrazek přejmenuje

mv 01.jpg obrazky         #přenese do podadresáře obrazky (musí existovat)
```

Existující soubor lze smazat příkazem **rm**. I zde lze pro bezpečnější rušení více souborů použít parametr **-i**, který si vynutí potvrzování každého souboru samostatně.

```
rm /home/bfu/01.jpg
```

Pro rychlé prohlížení souborů na obrazovce můžeme použít příkazy **cat** a **more**.

Příkaz more je výhodný v tom že stránkuje, po zaplnění obrazovky výpis zastaví a čeká na pokyn k dalšímu posunu stiskem klávesy. Příkaz **less** je pak alternativou k příkazu more. Příkaz cat naproti tomu můžeme využít k rychlému vytvoření krátkého textového souboru:

```
cat >poznamka.txt

 1.radek

 2.radek
```

po posledním řádku stisknout Ctrl+D

Archivace

Tar

tar = tape archiver

program pro archivaci souborů, který je schopen sloučit (spakovat) velké množství souborů do jediného archivního souboru, přičemž zachovává veškeré informace o souborech (čas vytvoření, přístupová práva, atd.)

Použití:

```
tar functionoptions files
```

Parametr function může být:

- `-c` : pro vytvoření nového archivního souboru
- `-x` : pro extrakci souborů z archivního souboru
- `-t` : pro výpis obsahu archivního souboru
- `-r` : pro přidání souborů na konec archivního souboru
- `-u` : pro aktualizaci souborů, které jsou novější než soubory v archivním souboru

- `-d` : pro porovnání souborů v archivním souboru se soubory v souborovém systému
- `-z` : předpoklad, že soubor je (nebo bude) komprimován programem gzip

Nejčastěji používané volby options jsou tyto:

- `-v` : pro výpis podrobných informací v průběhu pakování a rozpakování
- `-k` : pro zachování existujících souborů při rozpakování, tj. žádný existující soubor nebude přepsán souborem z archivního souboru
- `-f` : filename pro specifikaci jména archivního souboru

Spakování:

```
tar cf moje_sbaleno moje_skripty
```

Rozpakování:

```
tar xvf moje_sbaleno
```

Zip a Unzip

```
zip -r zazipovano.zip /home/kindle/
```

```
unzip zazipovano.zip
```

Filtry

Cat a Tac

Filtry jsou příkazy pro práci s textovými soubory. Ty na vstupu, podle zadaných parametrů zpracují text nebo textový soubor. Zpracovaný text odešlou na standardní výstupní zařízení – obrazovku (terminál).

Význam:

- Pro efektivní práci s konfiguračními soubory*, protože většina konfiguračních souborů v Linuxu je v textové podobě*
- Umožňují filtrovat výstupy příkazů při vytváření „kolon“ (spojování příkazů pomocí roury/pipe)

Příkaz **cat** posílá vstup na standardní výstup – začíná od prvního řádku, narozdíl od příkazu **tac**, který posílá vstup na standardní výstup, ale začíná od posledního řádku. Použití:

- Vypíše obsah souboru po řádcích na obrazovku

- Přeměňuje obsah souborů soubor1, 2 a 3 do jednoho souboru (spojování souborů)

```
cat  soubor1 soubor2 soubor3 > soubor
```

- Vytvoří soubor do kterého ukládá vstup z klávesnice, vstup je ukončen sekvencí [CTRL-C]

```
cat  > soubor
   Petr
   Karel
   Jana
  Ctrl-C
```

Head a Tail

Příkaz **head** vypisuje prvních *n*-řádků ze začátku daného souboru. Defaultně (bez parametru) vypíše prvních 10 řádků.

```
head soubor
head –počet_řádků soubor
```

Příkaz **tail** vypíše *n*-řádků od konce souboru. V základu, tedy bez parametru, vypíše posledních 10 řádků.

```
tail soubor
tail –n počet_řádků soubor
```

Použití: pro čtení logů, parametr **–f** způsobí , že po přečtení *n*-řádků ze souboru se příkaz neukončí a vypisuje dále nově příchozí data.

```
tail -f /var/log/messages
```

Grep

Vyhledá a vypíše ze vstupu / vstupního souboru všechny řádky, které obsahují zadaný řetězec.

```
grep "Kája  Mařík" text.txt
```

Vypíše všechny řádky z *text.txt*, které obsahují řetězec *Kája Mařík*

```
ls | grep  "te"
```

Vypíše všechny soubory v aktuálním adresáři, které obsahují řetězec *te*. Používá řadu voleb, některé z nich:

- -i potlačuje citlivost na velká/malá písmena
- -w vypíše řádky, které obsahují řetězec jako slovo *(před a za je mezera)*
- -v vypíše všechny řádky, které řetězec neobsahují

```
grep –v  \#  /etc/inetd.conf
```

Příkaz vypíše všechny nezakomentované řádky v zadaném konfiguračním souboru (# označuje ve skriptech řádky komentáře) a (\ potlačuje speciální význam některých znaků, znak se bude brát jako obyčejný řetězec).

Cut

Slouží k extrahování polí nebo znaků z daného souboru či výstupu.

- -c (character specifier) slouží k určení znaků, které se mají vypisovat - za volbu uvádíme pořadová čísla znaků
- -f (field specifier) určení polí, která se mají vypisovat - za volbu uvádíme pořadová čísla polí
- -d*znak* (delimiter) - uvedený oddělovací znak umožní identifikovat pole

```
cut -d: -f1,3 /etc/passwd
```

Ze souboru *passwd* vybere z každého řádku: pole první (login name) a pole třetí (UID).

Paste

Tento příkaz umožňuje spojovat více souborů do jednoho a to tak, že spojí soubory do sloupců – vždy řádek jednoho s řádkem dalšího.

```
cat > s1    cat > s2    paste s1 s2
 modrá       barva       barva  modrá
 rychlé      auto        auto rychlé
 studené     pivo        pivo studené
 [CTRL-C]    [CTRL-C]
```

Sort

Setřídí vstup / vstupní soubor a to po řádcích podle ASCII tabulky. Obsahuje řadu přepínačů:

- -k pole - určuje podle kterého pole se bude třídit
- -f nerozlišuje se mezi malými a velkými písmeny
- -n číselné třídění

- -r reverzní třídění
- -b jsou ignorovány mezery a tabulátory na začátku řádku

```
wc -l * | sort -r
```

Setřídí soubory podle klesajícího počtu řádků.

nl

Očísluje standartní vstup a na standartní výstup vypíše výsledek. Použití:

```
ls | nl
```

wc

Vypíše počet znaků, slov a řádků standartního vstupu na standartní výstup. `wc -l` zobrazí počet řádků standartního vstupu. `wc -l`wc program sám o sobě vypíše tři čísla, které vypočítal ze standartního vstupu. Jedná se (zleva doprava) o počet řádků, počet slov, počet znaků. Použití:

```
ls | wc
```

Diff

```
diff [volby] soubor1 soubor2
```

Při porovnávání textových souborů příkaz **diff** nalezne změny v libovolném znaku na řádku, tedy i např. přidání jedné mezery. Program se snaží nalézt změněné řádky, vložené části textu, vymazané části textu. Jestliže chceme porovnat obsah dvou souborů bez ohledu na počet mezer nebo tabulátorů mezi slovy, použijeme přepínač **-b** (blanks).
Ve výpisu je nejdříve údaj o typu a rozsahu nalezených změn, poté následuje výpis řádek z obou souborů: znak < označuje řádky prvého souboru, znak > označuje řádky druhého souboru. Údaj o typu a rozsahu změn má tuto strukturu m x n, kde:

- m - označuje čísla řádek prvního souboru
- n - označuje čísla řádek druhého souboru
- x - může nabývat hodnot:
    - a - přidání řádků k prvnímu souboru
    - d - zrušení řádků z druhého souboru
    - c - změna znaků na řádcích

Např. zápis 8,20 c 8,22 označuje že na řádcích 8 až 20 z prvního souboru byly provedeny změny, které jsou zapsány na řádcích 8 až 22 druhého souboru (současně byly dva řádky vloženy).

Cmp

```
cmp [volby] soubor1 soubor2
```

Porovnává dva soubory a vypíše číslo řádku, na kterém najde první rozdíl a číslo znaku, který se první na tomto řádku liší.

```
comm  soubor1  soubor2

cmp   soubor1  soubor2

diff  soubor1  soubor2
```

tr

Příkaz nahradí znaky z prvního řetězce odpovídajícími znaky z druhého řetězce. V příkazu tr nelze zadat jméno vstupního souboru na příkazové řádce - nutno pomocí přesměrování.

```
tr a y < konec text
```

Nahradí ve vstupním souboru text všechny znaky *a* znakem *y*

Alias

= přezdívka alias interní příkaz Bashe. Příkaz zapsaný bez parametrů vyvolá seznam přidělených přezdívek. Vytvoření:

```
alias přezdívka= 'příkaz'

alias ll=ls –la|less

unalias přezdívka
```

Příkazem **ln** vytvoříme odkaz, tedy jiný další název kterým se na soubor nebo adresář můžeme odkazovat. Odkaz samozřejmě může být uložen ve zcela jiném adresáři, což se dá využít například pro rychlejší přístup k prohlížení soubor logu */var/log/boot.log*.

```
ln -s /var/log/boot.log log
```

A pak si ho lze prohlížet přímo pomocí more log.

Častější a užitečnější budou případy, kdy program vyžaduje potřebné knihovny jinde než jsou nainstalovány. V tom případě se místo kopírování jen vytvoří odkaz na patřičný soubor. Zde byl použit parametr **-s**, který označuje běžnější, **symbolický odkaz**, který se odkazuje na jméno původního souboru, na rozdíl od druhého typu odkazu, pevného, který je přiřazen přímo inodu, tj. umístění souboru na disku.

Uživatelské účty v Linuxu

Uživatelské účty

Organizace

Základním rysem každého unixového systému je podpora více uživatelů s možností jemného nastavení práv. Každý, kdo do systému přistupuje, musí mít vlastní unikátní uživatelské jméno a libovolné heslo. Dále by měl mít vlastní domovský adresář, umístěný v adresáři */home*. Veškeré informace o uživateli a jeho nastaveních jsou zapsány v souboru */etc/passwd*. Práva na editaci tohoto souboru má z bezpečnostních důvodů pouze uživatel root, avšak právo na čtení musí mít kdokoliv, neboť informace z tohoto souboru využívá mnoho programů, což předpokládá čitelnost souboru pro všechny uživatele.

Právě z tohoto důvodu už soubor */etc/passwd* neobsahuje hesla. Ta jsou v šifrované podobě umístěna v souboru */etc/shadow*, k němuž může přistupovat pouze administrátor (root). Oddělení hesla od ostatních informací o uživateli se nazývá stínění.

Samotné jádro OS Linux považuje uživatele za pouhé číslo tzv. UID (User ID), od něj se pak odvozují samotné uživatelské účty, které se dále dělí na 3 typy:

- root - superuživatel
- user - standartní uživatel
- systémový - slouží ke spouštění systémových služeb, které z důvodu bezpečnosti nemohou běžet s právy roota

Konfigurační soubory

Základní **databází uživatelů** je v systému textový soubor */etc/passwd* (angl. password file), v němž jsou uvedeny platná uživatelská jména a další k nim přidružené informace. Každému uživateli odpovídá v souboru jeden záznam - řádek, který je rozdělen na sedm polí, jejichž oddělovačem je dvojtečka. Význam jednotlivých položek je následující:

1. Uživatelské jméno
2. Heslo - bylo nahrazeno stínovým heslem uloženým v */etc/shadow*
3. UID (Identifikační číslo uživatele) To je v systému unikátní - není tedy možné, aby stejné UID měli dva různí uživatelé
4. GID (Identifikační číslo skupiny) specifikuje primární skupinu, do níž uživatel náleží (každý musí být alespoň v jedné kvůli právům)

5. Nepovinná (skutečné jméno uživatele, popis účtu, atd.)
6. Domovský adresář
7. Příkazový interpret (nebo program), který se spustí po přihlášení

Každý uživatel systému má k souboru */etc/passwd* přístup (může jej číst). Může tedy například zjistit přihlašovací jména ostatních uživatelů. (To ale neznamená, že jsou všem přístupná i hesla!)

Zobrazení souboru passwd provedeme příkazem:

```
cat /etc/passwd
```

V souboru passwd jsou místo hesel jen znaky "x", protože samotná hesla jsou uložena v bezpečnějším souboru "shadow", tedy stínovém **souboru hesel**. Tento soubor může číst pouze superuživatel!

Zobrazení informací v něm provedeme příkazem:

```
su root
```

```
cat /etc/shadow
```

1. Identifikační jméno
2. Zašifrované heslo (u služeb je nahrazeno "*" nebo "!")
3. Počet dnů od 1.1.1970 do změny hesla
4. Počet dnů do povolení změny hesla
5. Počet dnů, po kterých si musí uživatel své heslo změnit
6. Počet dnů před skončením platnosti hesla - uživatel bude varován
7. Počet dnů od skončení platnosti hesla do zablokování účtu (nemusí být vyplněno)
8. Počet dnů od 1.1.1970 do zablokování účtu
9. Rezervovaný prostor pro budoucí použití.

Jak funguje šifrování hesla a proč nejde jen tak dešifrovat? Je to prosté: Ve chvíli, kdy si uživatel zvolí své přístupové heslo, je pro něj určena také hodnota zvaná *salt*. Jde o náhodně vygenerované číslo, s jehož pomocí se zakóduje heslo, které si uživatel zvolil. Jak salt, tak i samotné kódované heslo, jsou uloženy v souboru */etc/shadow*.
Při přihlašování se nejprve načte hodnota salt a s její pomocí se zakóduje heslo, které uživatel zadal při tomto přihlašování. Pokud výsledný řetězec odpovídá zápisu v */etc/shadow*, je uživatel oprávněn ke vstupu do systému. To vysvětluje, proč stejné heslo u více uživatelských účtů může mít v zašifrované formě zcela odlišnou podobu.

Jak už jsme si řekli, uživatelské skupiny slouží k diferenciaci uživatelských pravomocí. Například určíme, že několik uživatelů má přístup na Internet - přiřadíme je tedy do skupiny Internet. Dalším chceme zakázat hraní her - vyřadíme je tedy ze skupiny Games. Všichni uživatelé dané skupiny tedy sdílí přístupová práva, která nastavíme této skupině.

Každý uživatel systému musí náležet alespoň do jedné uživatelské skupiny. Může ale také patřit do více skupin. Tímto způsobem je možné snadno kombinovat práva tak, aby byla na míru ušita konkrétnímu uživateli a jeho potřebám. Veškeré informace o skupinách, jejich nastavení a členech, jsou uloženy v souboru */etc/group*.
Tento soubor obsahuje:

1. identifikační jméno skupiny
2. heslo skupiny (většinou se nepoužívá nebo se užívá gshadow)
3. GID (identifikační číslo skupiny)
4. seznam členů

1. Příklad:

```
useradd pokus
```

Co se udělá v souboru passwd:

- UID je o 1 větší
- GID: Vytvoří se skupina se stejným jménem jako user a nastaví se jako základní
- Domovský adresář: */home*
- Uživateli bude zpřístupněn interpret */bin/bash*
- Vytvoří se mu emailová schránka ve */var/spool/mail*

Co se udělá v souboru grub:

- Vytvoří se nová skupina se stejným jménem, ale je prázdná

Domovský adresář:

```
ls -la /home/pokus
```

- skryté soubory zkopírovány z */etc/skel/*
- *skryté soubory začínají tečkou*

Nastavení hesla:

```
passwd pokus
```

Zrušení účtu:

```
userdel pokus
```

Tímto příkazem jsou smazány záznamy z konfiguračních souborů, ale soubory, kterých byl tento uživatel vlastník zůstanou!

Zrušení účtu včetně souborů:

```
userdel -r pokus
```

Přístupová práva

Využití

Soubory a adresáře mohou být před nežádoucím přístupem chráněny nastavením přístupových práv. Pro každý soubor lze nastavit přístupová práva pro čtení, zápis nebo vykonání souboru pro vlastníka (User) souboru, skupinu (Group) uživatelů a všechny ostatní (Others).
Vlastníkem souboru je ten, kdo soubor vytvořil. Obyčejně vlastník může soubor číst i do něj zapisovat, ostatní uživatelé pak mohou soubor pouze číst.

Přístupová práva:

Název	*Kód*	*Význam pro adresáře*	*Význam pro soubory*
Supervisor	S	poskytuje všechna práva k adresáři i ke všem jeho souborům	Poskytuje k souborů všechna práva
Read	R	Umožňuje v adresáři otvírat soubory, číst je a spouštět	Umožňuje soubor otvírat, číst a spouštět
Write	W	Umožňuje v adresáři otvírat soubory a zapisovat do nich	Umožňuje soubor otvírat a zapisovat do něj
Create	C	Umožňuje v adresáři vytvářet nové podadresáře a soubory	Umožňuje soubor po jeho logickém zrušení obnovit
Erase	E	Umožňuje zrušit adresáři jeho podadresáře a soubory	Umožňuje zrušit soubor
Modify	M	Umožňuje změnu atributů a přejmenování jak adresáře, tak jeho podadresářů a souborů	Umožňuje změnu atributů a přejmenování souboru
File Scan	F	Umožňuje zobrazit adresář jeho soubory	Umožňuje zobrazit soubor
Access Control	A	Umožňuje přidělovat adresáři jeho podadresářům souborům pověřence, definovat jejich přístupová práva a modifikovat jejich filtry děděných práv	Umožňuje přidělovat souboru pověřence, definovat jejich přístupová práva a modifikovat filtr děděných práv souboru

Ke zjištění přístupových práv použijeme příkaz:

```
ls -l
```

Zobrazí se tabulka:

drwx------	1	own	grp	24	Apr	4	22:21	adresar
-rw-r--r--	1	own	grp	930	Apr	4	22:24	soubor1
-rw-r--r--	1	own	grp	7925	Apr	4	22:26	soubor2
-rw-r--r--	1	own	grp	37	Apr	4	22:28	soubor3

Význam jednotlivých sloupců:

- Typ souboru

    - '-' : obyčejný soubor
    - 'd' : adresář
    - 'l' : symbolický odkaz
    - 'c' : znakové zařízení
    - 'b' : blokové zařízení
    - 'p' : je pojmenovaná roura

  Následují oprávnění pro vlastníka, skupinu a ostatní uživatel (vždy tři oprávnění). Pokud není některé oprávnění přiděleno, objeví se při výpisu pomlčka (-).

- počet odkazů na i-uzel (inode)
- vlastník objektu
- skupina, které objekt patří
- velikost objektu
- datum a čas poslední změny objektu (3 sloupce)
- název objektu

Jak porozumět zápisu: **trwxrwxrwx**

- `-rwx------` jsou práva pro vlastníka
- `----rwx---` jsou práva skupiny
- `-------rwx` jsou práva ostatních
- `t---------` označuje typ souboru

Písmeno	**Práva**	**Hodnota**
r	čtení	4
w	zápis	2
x	spouštění	1

Přístupová práva **pro soubor** se definují:

- **r** soubor je povoleno číst.
- **w** do souboru je povoleno zapisovat.
- **x** soubor je povoleno spustit (provést).

Přístupová práva **pro adresáře** mají tyto významy:

- **r** adresář je povoleno vypsat
- **w** do adresáře je povoleno zapisovat; tj. lze vytvářet a rušit soubory.
- **x** do adresáře je možné vstoupit; tj. adresář může být argumentem příkazu cd.

Z těchto údajů tedy vyplývá:

- Číst soubor mohu, pokud mám právo vstupu do adresáře (x) a právo čtení souboru (r).
- Zapisovat do souboru mohu, pokud mám právo vstupu do adresáře (x) a právo zápisu do souboru (w).
- Spustit soubor mohu, pokud mám právo vstupu do adresáře (x) a právo provedení souboru (x).

Různé kombinace nastavení přístupových práv:

Kombinace	Ekvivalent	Popis
-rw-------	600	vlastník má práva pro čtení a zápis (nastavení většiny souborů)
-rw-r--r--	644	vlastník má práva pro čtení a zápis, skupina a ostatní mohou pouze číst.
-rw-rw-rw	666	každý může číst a zapisovat (tato kombinace není doporučena, jelikož může být soubor kdykoli a kýmkoli změněn)
-rwx------	700	vlastník má práva pro čtení, zápis a spuštění (nejlepší kombinace pro programy, které chce vlastník spouštět)
-rwxr-xr-x	755	vlastník má práva pro čtení, zápis a spouštění (všichni uživatelé mohou soubor číst a spouštět)
-rwxrwxrwx	777	všichni mají práva pro čtení, zápis a spouštění
-rwx--x--x	711	vlastník má práva pro čtení, zápis a spouštění, ostatní mohou program pouze spouštět (užitečné pro programy, které mohou spouštět všichni, ale nemohou je okopírovat)
drwx------	700	pouze vlastník může číst a zapisovat do tohoto adresáře
drwxr-xr-x	755	tento adresář může být změněn pouze vlastníkem, všichni ostatní však mohou zjistit jeho obsah.
drwx--x--x	711	vhodná kombinace, chcete-li aby mohli obsah adresáře číst všichni uživatelé, soubor byl pro příkaz *ls* neviditelný. Omezíte tím přístup pouze na osoby, které znají obsah adresáře.

Nastavení

Změní přístupová práva jednoho nebo více souborů. Pouze vlastník souboru nebo superuživatel může měnit přístupová práva. Práva mohou být vypsána jako číslo nebo výraz ve tvaru kategorie, operátor, právo.

Přístupová práva lze nastavit:

- Absolutně - pomocí oktálového čísla
- Symbolicky

Absolutně

Výsledná práva jsou tvořena součtem číslic

```
   user  group others
r  0400  0040  0004
w  0200  0020  0002
x  0100  0010  0001
```

```
chmod 444 soubor
```

Všem je nastaveno pouze právo pro čtení.

Symbolicky

Kategorie

- u uživatel
- g skupina
- o ostatní
- a všichni *(implicitní hodnota)*

Operátory

- + přidá práva
- - odebere práva
- = přiřadí práva *(nespecifikovaná práva odebere)*

Práva

- r právo čtení
- w právo zápisu
- x možnost spustit program

- s propůjčení práv vlastníka (nebo skupiny) souboru aktuálnímu uživateli
- t zamezení mazání souborů v adresáři uživateli nebo jinými vlastníky
- u aktuální práva vlastníka souboru
- g aktuální práva skupiny
- o aktuální práva ostatních
- l povinné zamykání souboru při přístupu (mandatory locking)

Příklad:

```
chmod u+x soubor
```

přidání práva *provádění* vlastníku souboru

```
chmod 644 soubor
```

nastaví rw-r--r--

chown, chgrp

Příkaz chown (change owner) slouží k přenášení vlastnických páv mezi uživateli. Tento příkaz smí použít pouze superuživatel.

```
chown username filename
```

Příkazem chgrp (change group) lze měnit nastavení oprávnění k souboru pro skupinu.

```
chgrp groupname filename
```

Odkazy

Při vytvořeni souboru se v adresáři vytvoří ukazatel na soubor. Spojuje jméno souboru s místem na disku, kde je soubor uložený. Odkaz se používá ke sdílení souborů.

Pevný

Pevný odkaz je jméno souboru, jeden soubor muže mít několik pevných odkazů - jmen. Každý soubor je reprezentován alespoň jedním odkazem v adresáři, kde byl vytvořen - viz sloupec popisující počet odkazů v `ls -l`. Je to prostě jiné jméno pro soubor (který je na disku pouze jednou). Ve výpisu vidíme, že soubory běžné soubory i pevné odkazy mají ve druhém sloupci dvojku. Což je počet pevných odkazů. Pevné odkazy nejsou svázány se jménem souboru (ale s číslem inodu). Nejde dělat pevné odkazy mezi různými oddíly disku a na adresáře.

Další odkazy na už existující soubor se vytvářejí pomoci příkazu `ln`, používá se stejně jako `cp`.

```
vi bob
```

```
ln bob /home/vas_home_adr/texty/bobek
```

V adresáři *work* vytvoří soubor *bob* a v adresáři *texty* k němu odkaz *bobek*.

Symbolický

Symbolický odkaz je malý soubor, který obsahuje cestu k odkazovanému, tj. sdílenému souboru.
Tento druh odkazu se dá se přirovnat k hypertextovému linku na webu. Ve výpisu vidíme, na který soubor odkaz ukazuje. Odkaz je svázán se jménem souboru, tudíž jeho přejmenování odkaz zneplatní. Můžeme vytvářet odkazy na adresáře. Při výpisu pomocí `ls` je prvním znakem přístupových práv písmeno s.
Vytvoření symbolického odkazu:

```
ln -s zdroj cíl
```

Práce s textovými soubory

Editor Vi

Obsluha

Spuštění:

```
vi soubor (soubor nemusí existovat)
```

Základním příkazem pro ukončení je **:q**. To projde pouze v případě, že jste v textu neprovedli žádné změny. Jinak skončíte s chybovým hlášením:

```
No write since last change (use ! to override)
```

Říká, že poslední změny v souboru nemáte uloženy. Vás další postup závisí na tom, zda je chcete uložit nebo ne.

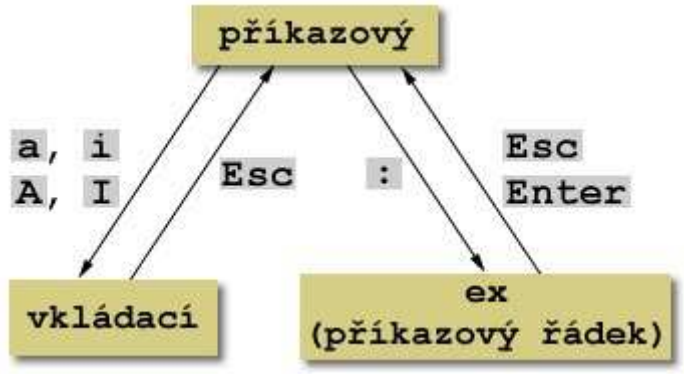
Ukončení s uložením změn:

Klasický postup velí nejprve uložit změny příkazem **:w** a poté ukončit editor pomocí **:q**. Oba příkazy můžete sloučit do jednoho společného **:wq**. Jelikož je tato varianta poměrně častá, byla pro ni vytvořena ještě jedna zkratka. Pokud v normálním režimu zadáte ZZ (dvě velká "**Z**"), provede se "inteligentní ukončení". Pokud byl soubor změněn, uloží se. Poté editor ukončí svou činnost.

Ukončení bez ukládání změn:

Jestliže změny nechcete uložit použijte příkaz **:q!**. Obdobně lze vykřičníkem přehlušit protesty i v několika dalších situacích.

Tři režimy práce



Příkazový mód:

Po otevření souboru je editor nastaven v příkazovém módu. V rámci příkazového módu můžete provádět následující operace:

- Spustit vkládací mód
- Aplikovat editovací příkazy
- Posouvat kurzor na jakoukoliv pozici v souboru
- Vyvolat příkazy editoru ex
- Spustit interpret příkazů
- Uložit nebo zrušit aktuální verzi souboru

Vkládací mód:

V rámci vkládacího režimu je možné přidávat nový text do souboru. K opuštění vkládacího režimu je potřeba **stisknout klávesu ESC**.

Následující příkazy vyvolají vkládací mód:

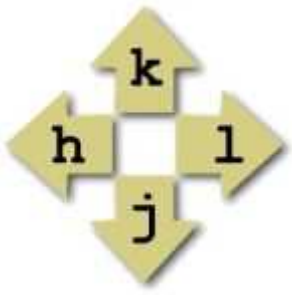
- `a` : přidání textu za pozici kurzoru
- `A` : přidání textu na konec řádku
- `c` : záměna textu
- `C` : zamění text do konci řádku za vložený text
- `i` : vložení textu před pozici kurzoru
- `I` : vložení textu na začátek řádku
- `o` : vytvoření nového řádku pod řádkem aktuálním
- `O` : vytvoření nového řádku nad řádkem aktuálním
- `R` : zapne přepisování znaků
- `s` : záměna jednoho znaku
- `S` : záměna celého řádku

Režim příkazového řádku

Slouží k zadávání příkazů na příkazovém řádku. V něm se realizují nejkomplikovanější operace, jako je vyhledání a náhrada řetězce znaků, práce se soubory a podobně. Režim ex je zahájen znakem **:**. Bývá zvykem jména příkazů pro příkazový řádek zahajovat dvojtečkou, aby se tato skutečnost zdůraznila. Režim ex je jednorázový. Po provedení příkazu (který odešlete klávesou Enter) se editor vrátí do normálního režimu. Pokud chcete dosáhnout téhož, ale bez provedení příkazu, použijte jako obvykle [Esc].

Ovládání

Základní pohyby kurzoru



Samozřejmě fungují i klasické kurzorové klávesy. Pro zmíněnou obludnost existují dva rozumné důvody:

1. Editor vi, který vznikal v době, kdy kurzorové klávesy nebyly.
2. Lidé, píšící všemi deseti, nemusejí přestěhovávat ruce ze základní klávesnice a zoufale se tak brzdit.

Druhým nejčastějším pohybem je posun o jednu obrazovku. Ten ve vim zajistí klávesové kombinace **[Ctrl-F]** (směrem dolů) a **[Ctrl-B]** (vzhůru). V běžných implementacích můžete pro tyto pohyby používat i obvyklé klávesy [PageDown] a [PageUp]. Vi se snaží při posunu na sousední řádek udržovat kurzor ve stejném sloupci. Je-li některý řádek kratší, posune kurzor na jeho poslední znak. Pokud se následující řádek opět prodlouží, vrátí se do původního sloupce.

Opakování příkazů

Téměř všechny příkazy vi lze provádět opakovaně. Pokud nejprve napíšete číslo n a po něm určitý příkaz, bude proveden n-krát. Například pohyb o 20 znaků doleva by zajistil příkaz 20h. Opakovat lze skutečně prakticky všechno, včetně komplikovaných věcí.
Další věci pro opakování:

- `12x` smaže dvanáct znaků
- `10k` posune kurzor o deset řádků nahoru
- `3dd` smaže tři řádky
- `3J` spojí tři řádky
- `db` smaže text od pozice kurzoru do začátku slova
- `6dw` smaže text od kurzoru do konce šestého slova
- `25,30d` smaže řádky 25 až 30 včetně
- `gdd` smaže vše od aktuálního řádku včetně k začátku

Logické poskoky

Pro posuny o slovo slouží následující skupina příkazů:

- `w, W` na začátek následujícího slova
- `e, E` na konec tohoto slova
- `b, B` na nejbližší předchozí začátek slova
- `)(, B` přesune kurzor na začátek nejbližší následující (předchozí) věty

Copy and paste

Veškeré likvidační operace mají jednu zajímavou vlastnost: vymazaný text se automaticky ukládá do registru, z nějž jej následně můžete vložit na jiné místo. O vložení se postará klávesa p (vkládá za aktuální pozici) nebo P (před aktuální pozici).

- x - kopírovat
- y - vyjmout
- p vložit

Odvolávání a vracení

- u - odvolání poslední změny (undo)
- Ctrl-r (redo)
- U - odvolá všechny změny, provedené na naposledy změněném řádku

Hledání

Příkaz k vyhledání řetězce znaků můžete ve vim zadat dvěma základními způsoby.

- / - hledání směrem vpřed
- ? - hledání směrem vzad

Je-li nastaveno `:set ignorecase`, vi je nebude rozlišovat. Pokud naopak použijete `:set noignorecase`, budou malá a velká písmena považována za odlišná (to je implicitní stav). Při nastavení `:set smartcase` se rozlišování malých/velkých písmen snaží chovat inteligentně. Zapíšete-li hledaný řetězec malými písmeny, hledá malá i velká. Pokud použijete nějaká velká písmena, hledá řetězec přesně v tom tvaru, ve kterém jste jej zadali.

Hledání je totiž pohyb se vším, co k němu náleží. Můžete ho proto kombinovat s modifikátory, které byly popsány v kapitole o editaci textu. Takže například d/ahoj vymaže vše od stávající pozice kurzoru až k nejbližšímu následujícímu výskytu řetězce "ahoj".

Editor Nano

```
 .d8888b.  888b    888 888     888
d88P  Y88b 8888b   888 888     888
888    888 88888b  888 888     888
888        888Y88b 888 888     888
888  88888 888 Y88b888 888     888
888    888 888  Y88888 888     888
Y88b  d88P 888   Y8888 Y88b. .d88P
 "Y8888P88 888    Y888  "Y88888P"

88888b.   8888b.  88888b.   .d88b.
888 "88b     "88b 888 "88b d88""88b
888  888 .d888888 888  888 888  888
888  888 888  888 888  888 Y88..88P
888  888 "Y888888 888  888  "Y88P"
```

Obsluha

Otevírání a vytváření souborů

Otevírání souborů i jejich vytváření je s nanem snadné, stačí jen napsat:

```
nano nazev_souboru
```

Nano nemá více režimů, to znamená, že můžete ihned začít psát a vkládat text. Jestliže editujete konfigurační soubor jako třeba */etc/fstab*, použijte parametr **-w** (zakázání zalamování slov).

```
nano -w /etc/fstab
```

Je opravdu velmi důležité, abyste při otevírání konfiguračních souborů použili tento přepínač. Neučiníte-li tak, může se stát, že systém již nabootuje nebo bude mít jiné problémy.

Ukládání a ukončení

Chcete-li uložit provedené změny, zmáčkněte [Ctrl+O]. Chcete-li editor ukončit, použijte [Ctrl+X]. Budete-li ukončovat editor s otevřeným změněným soubor, nano se Vás zeptá, zda jej budete chtít uložit. Pokud změny nechcete zachovat, jenom zmáčkněte N, pro jejich uložení Y. V tom případě budete dotázáni na jméno souboru; napište jej a zmáčkněte Enter. Pokud jste omylem potvrdili, že chcete uložit soubor, avšak rozmysleli jste si to, můžete proces při žádosti o jméno souboru přerušit stiskem [Ctrl+C].

Vyjmutí a vkládání

Pro vyjmutí jedné řádky použijte [Ctrl+K] (držte CTRL a zároveň zmáčkněte k) — řádek zmizí. Pro vložení najeďte kurzorem tam, kam má text přijít, a zmáčkněte [Ctrl+U] — řádka byla vložena. Potřebujete-li přesunout celý odstavec, jednoduše pomocí [Ctrl+K] vyjměte řádek po řádce a pak je jedním [Ctrl+U] vložte na potřebné místo. Jak vidíte, celý odstavec se posunul tam, kam jste chtěli.

Hledání textu

Hledání textu je snadné, stačí zmáčknout [Ctrl+W] (od anglického "WhereIs"), napsat hledaný řetězec a zmáčknout Enter. Chcete-li hledat další výskyt, zmáčkněte Alt+W. Poznámka: V nápovědě nano je klávesa Ctrl značena jako stříška (^), takže například [Ctrl+W] je zapsáno jako ^W. Klávesa Alt je značena jako M (od "Meta"), tudíž M-W znamená Alt+W.

Přesměrování

Přesměrování výstupu

Standardní výstup a standardní chybový výstup (zobrazený na terminálu) lze přesměrovat jinam, nejčastěji do souboru:

```
ls > seznam
```

Vznikne soubor *seznam* s výpisem obsahu adresáře. Pozor, pokud už soubor seznam existuje bude přepsán!

Zobrazení souboru provedete například příkazem `cat`.

Význam znaků <, >:

- > soubor - standardní výstup je přesměrován do zadaného souboru
- >> soubor - přesměruje standardní výstup do souboru. Jestliže soubor existuje, je výstup přidáván na konec souboru
- < soubor - standardní vstup je nahrazen obsahem souboru

Pokud na konec příkazu přidáte řetězec >>*název_souboru*, výstup příkazu se připojí k souboru se zadaným názvem místo toho, aby se soubor přepsal. Symbol >> se označuje jako operátor připojení výstupu.

Chcete-li například připojit soubor *soubor2* k souboru *soubor1*, zadejte:

```
cat soubor2 >> soubor1
```

Poznámka: Jestliže soubor *soubor1* neexistuje, je vytvořen.

Zřetězení textových souborů

Sloučení různých souborů do jednoho souboru se označuje jako zřetězení. Na příkazový řádek zadejte například:

```
cat soubor1 soubor2 soubor3 > soubor4
```

Vytvoří se soubor *soubor4*, který se skládá z obsahu souborů *soubor1*, *soubor2* a *soubor3* připojených v daném pořadí. Pozor, příkaz cat nejdříve vytvoří výstupní soubor, což je nutné vzít na vědomí v případě složitějších řetězení.

Přesměrování vstupu

Pokud na konec příkazu přidáte řetězec < název_souboru, vstup příkazu se čte ze souboru se zadaným názvem. Symbol < se označuje jako operátor přesměrování vstupu.
Poznámka: Vstup lze přesměrovat pouze pro ty příkazy, které standardně přebírají vstup z klávesnice.

```
mail tomas < dopis1
```

Odešle soubor *dopis1* jako zprávu uživateli *tomas*.

Černá díra

Soubor */dev/null* je speciální soubor. Tento soubor má jedinečnou vlastnost, že je vždy prázdný. Všechna data, která odešlete do souboru */dev/null*, se zahazují. Tato vlastnost je užitečná, když spouštíte program nebo příkaz, který generuje výstup, který chcete ignorovat.

Kromě standardního vstupu a výstupu příkazy často vytvářejí jiné druhy výstupu, například chybové a stavové zprávy známé jako diagnostický výstup. Podobně jako standardní výstup se i standardní chybový výstup vypisuje na obrazovku, pokud není přesměrován.

Máte například program nazvaný *mujprog*, který přijímá vstup z obrazovky a za běhu generuje zprávy, které byste chtěli ignorovat. Chcete-li číst vstup ze souboru *mujskript* a vyřadit standardní výstup zpráv, zadejte:

```
mujprog < mujskript 2> /dev/null
```

V tomto případě používá program *mujprog* soubor *mujskript* jako vstup a veškerý standardní **chybový** výstup se zahodí.

Kolony

Roura - pipe
Nástroj pro spojování procesů do řetězce pomocí vzájemného propojení standardních proudů tak, že je vždy výstup je nasměrován přímo do vstupu . Jde o jednosměrnou komunikaci.

- je oddělovačem příkazů
- zařídí to, že se standardní výstup předchozího příkazu napojí na standardní vstup následujícího příkazu

V příkladu uvedeném níže příkaz ls vypisuje obsah adresáře a less je program, který umožňuje prohlížet obsah svého vstupu po jednotlivých stránkách, vracet se a vyhledávat. Kolona příkazů v příkladu tedy umožní prohlížet celý výpis postupně, i když se nevejde na jednu obrazovku terminálu.

```
ls -l | less
```

Procesy

Spouštění programů

Proces je v UNIXu každý spuštěný program, nahraný do paměti. Každý proces má svoje jednoznačné identifikační číslo PID, vlastníka, prioritu a rodiče.

Spouštění aplikací v Bash

Když zadáme jméno spustitelného souboru v DOSu, ten se dívá, zda toto jméno existuje v aktuálním adresáři. Pokud ano, tak jej spustí. V Linuxu je třeba zadat jméno spustitelného souboru včetně cesty k němu i když máme soubor v aktuálním adresáři!
Stačí použít relativní cestu, tj. *./jmeno_souboru*.

Jinak Bash prohledává adresáře uvedené v *PATH*. Proměnná *PATH* obsahuje seznam adresářů ve kterých jsou spustitelné soubory (většinou jsou to adresáře bin).

Pokud chceme aby byl program nebo skript spustitelný, je vhodné použít konstrukci:

```
chmod +x jmeno_souboru
```

Kdy souboru přidáme právo X, čímž umožníme soubor spouštět (eXecute). V dalším kroku pak program spustíme. Takto se uvádějí v chod především skripty, tento postup lze však použít i pro další programy.

```
./jmeno_souboru
```

Process status

PID je unikátním číslem, náležící právě jednomu procesu a které jádro přiděluje vzestupně každému spuštěnému procesu. *Číslo 1 má vždy proces init, který je v systému spuštěn jako první.* Pro zjišťování PID kteréhokoli z běžících procesů (tedy nejen úloh spuštěných v rámci Bashe) slouží program **ps**. Když jej spustíme bez přepínačů, uvidíme jen aktuální Bash, seznam spuštěných úloh a vlastní běžící ps.

Ps vypíše informace o běžících procesech. Nevyžaduje znak "-" před zadáváním voleb.
Příkaz `ps` bez parametrů vypíše pouze procesy uživatele, který tento příkaz spustil a pouze ty procesy, které byly spuštěny ze stejného terminálu jako samotný příkaz `ps`. Chceme-li vypsat všechny procesy běžící v systému, tak použijeme následující parametry:

- `-a` : jsou vypsány informace o všech běžících procesech
- `-e` : ve výpisu je také zahrnuto prostředí, ve kterém program běží

- `-j` : vypíše procesy i s uvedením PID rodičovského procesu
- `-l` : dlouhý formát výpisu
- `-u` : vypíše také jméno uživatele a čas spuštění programu
- `-x` : jsou vypsány také procesy, které nemají řídící terminál
- `-w` : dlouhé řádky nejsou ořezávány

Stavy při výpisu procesů pomocí ps:

- `S` : proces usnul - čeká až na něj přijde řada a bude mu přidělen procesor
- `W` : paměťový prostor vyhrazený danému procesu byl kompletně uložen na disk (odswapován)
- `R` : proces je právě zpracováván procesorem
- `T` : proces byl pozastaven
- `D` : proces je v nepřerušitelném spánku (v tomto stavu jsou většinou procesy svázané s I/O operacemi)
- `Z` : proces, jehož rodičovský proces již ukončil svoji činnost (třeba díky nějaké závažné chybě a nechal po sobě sirotka. Rodičem Zombie procesu se stává Init).
- `L` : proces má uzamknuté stránky v paměti - platí obvykle pro procesy pracující v reálném čase
- `+` : proces je ve skupině procesů běžících na popředí
- `<` : proces s vysokou prioritou
- `N` : proces s nízkou prioritou
- `s` : is a session leader

pstree

- `-a` : zobrazení argumentů, s nimiž byly procesy spuštěny
- `-c` : zabrání shlukování stejných procesů, tj. vypíšou se i ty, které se několikrát za sebou opakují
- `-n` : seřadí procesy podle PID (číselně), a ne podle jména
- `-p` : za každým procesem zobrazí v závorce jeho PID

Top

Dalším příkazem, který vypisuje aktuálně běžící procesy a spoustu dalších důležitých systémových informací je příkaz `top`. Tento příkaz na rozdíl od předchozího příkazu `ps` nevypisuje pouze výpis aktuálního stavu procesů v systému, ale dokáže tento výpis dynamicky po určitých časových intervalech měnit. Díky tomu můžeme v reálném čase sledovat změny stavu procesů, jejich "boj" o procesor, aktuální velikost paměti, které procesy alokují a spoustu dalších užitečných informací. Defaultní časový interval změny výpisu je nastaven na 5 sekund, lze jej však změnit pomocí parametru `-d`.

Chod programu můžeme ovlivnit klávesami:

- [Shift+N] - třídění procesů podle PID
- [Shift+P] - třídění procesů podle zatížení CPU (odhalení zaseknutých procesů)
- [Shift+M] - třídění procesů podle zabraní v paměti (odhalení viníků swapování)
- [Shift+T] - třídění procesů podle strojového času (odhalení procesů nejvíce zatěžujících systém)
- [Q] - ukončení programu

Po zjištění viníka našich problémů nemusíme top opouštět. Po stisknutí [K] budeme vyzváni k zadání PID, kterému chceme poslat balíček. Po odentrování správného PID nám bude nabídnuto zadání signálu. Nezadáme-li žádný, bude odeslán SIGTERM (15), což je požadavek k ukončení, jeho násilná varianta, jak už bylo uvedeno je signál s číslem 9.

Spouštění na pozadí

Aby bylo možné co nejlépe využít víceúlohového prostředí (multitaskingu), zavedly interpretry možnost přepnout program spuštěný v textovém prostředí na pozadí. Spustit nebo přepnout program na pozadí znamená, že v průběhu jeho běhu máme k dispozici prompt a můžeme zadávat další příkazy. Spustíme-li prostřednictvím Bashe nějakou aplikaci v grafickém prostředí, neukáže se nám prompt dříve, než tato aplikace skončí.

Řekněme, že jsem z terminálu spustil *gkrellm* (panel pro monitorování systému), který teď blokuje prompt. Aplikace běží a kurzor v Bashi stojí nebo bliká bez promptu na začátku nového řádku. Takto běžící program můžeme z terminálu zastavit kombinací kláves [Ctrl+z]. Objeví se hláška Stopped. Od tohoto okamžiku sice máme zase prompt, ale program je zamrzlý. Aby mohl ve své činnosti pokračovat, musíme mu povolit činnost na pozadí příkazem **bg**. Po odentrování se ihned rozběhne.

Existuje ale i způsob, jak spustit grafickou aplikaci na pozadí přímo. Dělá se to přidáním znaku **ampersand (&)** za příkaz spouštějící aplikaci. Tak např. XMMS spouštíme z příkazové řádky zadáním:

```
xmms &
```

Po odentrování dostaneme zpátky prompt a také okno spouštěné aplikace.

Řízení více souběžných procesů

Z výše zmíněného vyplývá, že programů na pozadí můžeme spustit více souběžně. Jak se v nich potom ale vyznat, když jim chceme posílat různé signály? Jednoduše, neboť Bash si tyto úlohy čísluje. Seznam běžících nebo stojících úloh získáme příkazem **jobs**.

V seznamu je na každém řádku jedna úloha. V hranaté závorce je uvedeno číslo úlohy, následuje status (Running/Stopped) a zakončuje příkaz, kterým jsme úlohu spustili. Velmi důležité je zde číslo úlohy, pomocí něhož můžeme jednotlivé z nich oslovovat. Chceme-li např. vrátit z pozadí do popředí úlohu číslo 2, zadáme:

```
fg %2
```

Bash vypíše jméno příslušné úlohy a začne ji provádět na popředí, tj. nevrátí nám prompt. Ten bychom získali postupem [Ctrl+Z] a `bg %2`.

Nyní se podíváme ještě na pár parametrů příkazu `jobs`. Přepínače lze samozřejmě kombinovat.

- jobs -l, zobrazí nám mezi číslem úlohy a jeho stavem ještě PID
- jobs -r vypíše pouze běžící úlohy
- jobs -s pak pouze ty zastavené

Jak dostat úlohu pod správu Job Control?

1. Spustíme-li úlohu na pozadí (&), vypíše se identifikace:

```
yes > /dev/null &
```

```
[1] 1234
```

číslo úlohy je *1* a má top-level proces *1234*.

2. Běží-li úloha normálně spuštěná na popředí a chceme s ní nějak naložit, stiskneme [CTRL-Z] a shell pošle procesu signál STOP. Vypíše se:

```
yes > /dev/null
```

```
[2]+ Stopped yes > /dev/null
```

```
jobs
```

Vypíše seznam řízených úloh.

'+' znamená current job, '-' znamená previous job.

```
[1]- Running yes > /dev/null &
```

```
[2]+ Stopped yes > /dev/null
```

3. Spuštění stoplého procesu:

```
fg %2
```

4. Zabití zastaveného procesu:

```
kill %1
```

Ukončení procesu

Teď si ukážeme, jak ukončit nějakou úlohu na pozadí. Lze to samozřejmě udělat tak, že ji nejprve přepneme do popředí a potom ukončíme klávesovou kombinací [Ctrl+C]. Existuje ale i jiný způsob. Seznamme se s legendárním příkazem **kill**. Abychom určili adresáta, použijeme číslo úlohy (třeba kill %3) nebo číslo procesu (PID - např. kill 7190), tím pošleme požadavek na ukončení. Pokud program nereaguje vůbec, můžeme jej násilně ukončit signálem 9 takto:

```
kill -9 %2
```

Signály

Signál je systémem generovaná událost, kterou používá na základě reakce na podmínku a dává tím procesu šanci provést odezvu. Používají se při chybách v programech. Vyvolává je shell nebo terminal driver nebo si je procesy posílají mezi sebou.

Signál	Hodnota	Akce	Poznámka
SIGHUP	1	A	"Hangup" - při zavěšení na řídicím terminálu nebo ukončení řídicího procesu.
SIGINT	2	A	"Interrupt" - přerušení z klávesnice.
SIGQUIT	3	A	"Quit" - ukončení z klávesnice.
SIGILL	4	A	"Illegal Instruction" - neplatná instrukce.
SIGABRT	6	C	"Abort" - ukončení funkcí abort(3)
SIGFPE	8	C	"Floating point exception" - přetečení v pohyblivé řádové čárce.
SIGKILL	9	AEF	"Kill" - signál pro nepodmíněné ukončení procesu.
SIGSEGV	11	C	Odkaz na nepřípustnou adresu v paměti.
SIGPIPE	13	A	"Broken pipe" - pokus o zápis do roury, kterou nemá žádný proces otevřenou pro čtení.
SIGALRM	14	A	Signál od časovače, nastaveného funkcí alarm(1)
SIGTERM	15	A	"Termination" - signál ukončení
SIGUSR1	30,10,16	A	Signál 1 definovaný uživatelem
SIGUSR2	31,12,17	A	Signál 2 definovaný uživatelem
SIGCHLD	20,17,18	B	Zastavení nebo ukončení dětského procesu
SIGCONT	19,18,25		Pokračování po zastavení
SIGSTOP	17,19,23	DEF	Zastavení procesu
SIGTSTP	18,20,24	D	Zastavení znakem "Stop" z terminálu
SIGTTIN	21,21,26	D	čtení z terminálu v procesu běžícím na pozadí
SIGTTOU	22,22,27	D	zápis na terminál v procesu běžícím na pozadí

Následují ostatní signály:

Signál	Hodnota	Akce	Poznámka
SIGTRAP	5	CG	Přerušení při ladění (trasování,breakpoint)
SIGIOT	6	CG	IOT - synonymum signálu SIGABRT
SIGEMT	7,-,7	G	
SIGBUS	10,7,10	AG	"Bus error" - pokus o přístup mimo mapovanou paměť
SIGSYS	12,-,12	G	Nepřípustný parametr syst. volání (SVID)
SIGSTKFLT	-,16,-	AG	Chyba zásobníku koprocesoru
SIGURG	16,23,21	BG	Soket přijal data s příznakem Urgent (4.2 BSD)
SIGIO	23,29,22	AG	Lze pokračovat ve vstupu/výstupu (4.2 BSD)
SIGPOLL		AG	Synonymum SIGIO (Systém V)
SIGCLD	-,-,18	G	Synonymum SIGCHLD
SIGXCPU	24,24,30	AG	Překročen limit času CPU (4.2 BSD)
SIGXFSZ	25,25,31	AG	Překročen limit velikosti souboru (4.2 BSD)
SIGVTALRM	26,26,28	AG	Virtuální časovač (4.2 BSD)
SIGPROF	27,27,29	AG	Časovač používaný při profilování
SIGPWR	29,30,19	AG	Výpadek napájení (Systém V)
SIGINFO	29,-,-	G	Synonymum SIGPWR
SIGLOST	-,-,-	AG	Zámek souboru byl ztracen
SIGWINCH	28,28,20	BG	Změna velikosti okna (4.3 BSD, Sun)
SIGUNUSED	-,31,-	AG	Nepoužívaný signál

Příznaky ve sloupci "Akce" mají následující význam:

- A : signál standardně ukončí proces
- B : signál je standardně ignorován
- C : signál standardně způsobí výpis paměti procesu (core dump)
- D : signál standardně pozastaví provádění procesu
- E : signál nemůže být zachycen
- F : signál nemůže být ignorován
- G : signál není definován normou POSIX.1.

Plánování

At

Aplikace, která se má spouštět plánovaně nesmí mít interaktivní rozhraní. Pokud by totiž program potřeboval komunikovat s uživatelem, ztrácelo by předpřipravené spouštění smysl. K **jednorázovému** spouštění procesů se používá příkaz **at**. Nástroj se používá k ukládání příkazů, které mají být provedeny jednorázově později, v předem definovaný čas. Vykonání úlohy má na starosti démon. Jeho spuštění se provede příkazem:

```
/etc/init.d/atd start
```

Parametrem příkazu at je plánovaný čas vykonání. Pomocí přepínače *-f* lze zadat i soubor, který bude v určený okamžik proveden. Pokud soubor neuvedeme, objeví se prompt *at*>, kam příkazy zadáme. Zadávání ukončíme stiskem [Ctrl+D]. Předpokládejme, že je 16:00, do příkazového řádku zapište:

```
at 16:45
```

warning: commands will be executed using /bin/sh
at>

```
play /usr/share/sounds/KDE_Startup.wav
```

Po odentrování každého řádku přidáváme další příklady. Zadávání ukončíme již zmíněnou zkratkou [Ctrl+D].
Jako potvrzení úlohy proběhne hláška:

```
job 1 at Mon Feb 6 16:45:00 2012
```

Dále se hodí využít flexibilnost příkazu, jako ukazuje následující příklad:

```
spsei@debian:~$ at now + 2 hours
warning: commands will be executed using /bin/sh
at> shutdown -h now
at> <EOT>
job 2 at Mon Feb 6 18:50:00 2012
```

Se stavbou *at now + ...* můžete, kromě hodiny (hour) / (hours), použít i minuty (minutes), dny (days), týdny (weeks) i roky (years).

- HH:MM - konkrétní čas
- noon - 12:00
- teatime - 16:00
- MMM DD - název_měsíce den
- MM/DD/YY - měsíc, den, rok
- today - dnes

Pro zobrazení naplánovaných úloh se používá příkaz:

```
at -l
```

Níže uvedeným příkazem spolu s číslem úlohy (zjištěné pomocí přepínače -l) úlohu zrušíte

```
atrm 1
```

Právo na používání nástroje at lze regulovat:

- */etc/at.allow* obsahuje jména uživatelů, kteří smějí používat at
- */etc/at.deny*, obsahuje jména uživatelů se zakázaným přístupem.

Crontab

Cron je unixová utilita, která umožňuje automatické spouštění úkolů (scriptů) v pravidelných intervalech. To je velmi užitečné například při zálohování dat, pravidelné údržbě systému nebo kontrole funkčnosti vzdálených zařízení. **Crontab** je soubor, který obsahuje soupis příkazů, které jsou spouštěny v určeném čase a to na samostatném řádku. Příklad crontabu může vypadat například takto:

```
# Delej neco kazdou lichou hodinu kazdy sudy den:
* 1-23/2 */2 * * root /usr/local/sbin/delej_tady_neco.sh
# Zalohovani dat 1 x tydne v noci z patku na sobotu (v 0:42):
42 0 * * 6 root /usr/local/sbin/zal_data_tyden.sh
```

Skripty

Úvod

Skriptovací jazyky stejně jako kompilované programovací jazyky začínají psaním zdrojového kódu, ale již nejsou kompilovány a spuštěny. Místo toho interpreter jazyka čte instrukce ze zdrojového kódu a podle obsahu tyto instrukce vykonává. V případě bash spouští programy, které jste napsali do zdrojového kódu. Bohužel, protože interpreter čte každé instrukce jednotlivě a jejich zpracování není rychlé (program je částečně překládán za běhu), jsou tedy skriptovací jazyky výrazně pomalejší než kompilované programy. Hlavní výhoda skriptovacích jazyků spočívá v jejich přenositelnosti na jiné platformy či operační systémy, jejich jednoduchosti a také velikosti. Skriptovacím jazykem je právě například bash. Příklady dalších interpretovaných (skriptovacích) jazyků jsou JavaScript, Perl, Lisp, Prolog a další.

Proměnné

Proměnné v systému lze rozdělit do tří skupin.

- vnitřní proměnné shellu - o jejich naplnění hodnotami, se stará shell sám
- **uživatelské proměnné** - mohou být definovány uživatelem podle potřeby
- proměnné speciálního významu uchovávající informace o běžících procesech a předávaných argumentech

Proměnné jsou definovány svými jmény. Hodnotu proměnné přiřadíme znaménkem "rovná se ".

```
jmeno="Lenka"
```

Proměnnou zrušíme příkazem unset.

```
unset jmeno_promenne
```

Hodnotu proměnné zobrazíme pomocí echo.

```
echo "Zadali jste: $jmeno"
```

Změna hodnoty:

- Přepsáním hodnoty z Lenka na Petr: jmeno= "Petr"
- Přidáním hodnoty: jmeno= "$jmeno Novak"

Spojování řetězců

Dvě proměnné: *jmeno="Bart"* a *prijmeni="Simpson"* můžeme spojit: *cele="$jmeno $prijmeni"*.

Systémové proměnné

- PATH = seznam adresářů se spustitelným soubory
- BASH_VERSION = verze interpretru Bash , pole, které má 6 položek
- GROUPS = seznam skupin, jichž je současný uživatel členem
- HISTSIZE = počet zadaných příkazů, které si Bash pamatuje
- HOME = domovský adresář
- HOSTNAME = jméno počítače
- PS1 - syntaxe promptu
- MAIL = soubor s lokální schránkou
- OSTYPE = typ operačního systému
- PWD = aktuální pracovní adresář
- RANDOM = náhodné číslo do 0 do 32767
- SECONDS = počet sekund od startu shellu
- SHELL = určuje výchozí interpretr
- USER = jméno uživatele
- PS1 = hodnota v prompt

První skripty

Hello world. První řádka programu říká Linuxu, co se má použít za interpret.

```
#!/bin/bash

echo "Hello, world!"
```

Spuštění skriptu

Předpokládejme, že se náš skript jmenuje *hello.sh* (přípona sh není podmínkou, většina skriptů žádnou příponu nemá). Podíváme se jaká má nastavena práva. Zjistíme, že chybí právo pro spuštění. Musíme tedy nastavit právo pro spuštění. Zápis chmod může vypadat takto:

```
chmod 700 ./hello.sh, nebo
chmod +x ./hello.sh
```

Následně skript můžeme spustit takto:

```
./hello.sh
Hello, world!
```

Do programu je také možné přidat komentář. Děje se tak pomocí znaku #. Začíná-li řádka právě tímto znakem, bere se tato řádka jako poznámka. Tedy tato řádka se nevykonává. Jedinou výjimku tvoří právě první řádka skriptu, ve které je uveden interpreter, který provede skript.

2. script

Script, který přesune všechny soubory do adresáře, a pak tento adresář smaže i s tím, co obsahuje. Poté se tento adresář znovu vytvoří:

```
#!/bin/bash
mkdir trash
mv * trash
rm -rf trash
mkdir trash
echo "Vsechny soubory jsou smazany!"
```

Aritmetika

BASH také podobně jako většina programovacích jazyků dovede provádět matematické výrazy. Výsledek dostaneme pomocí konstrukce: `expr 3 + 3` (pozor na prázdné znaky)

Možné matematické operace:

- `+` : sčítání
- `-` : odčítání
- `*` : násobení
- `/` : dělení
- `%` : dělení modulo *(zbytek po dělení)*

```
#!/bin/bash
x=8
y=4
z=$[$x + $y]
echo "Soucet $x a $y je $z"
```

Bash dovede pracovat **pouze s čísly celými**. Nelze tedy používat desetinná čísla.

Uživatelský vstup

V bashi můžete vytvořit script, který bude interaktivní, tedy reagovat na vstup uživatele (popř. na vstup čtený ze souboru). Příkaz uživatelského vstupu je příkaz `read`. Příkaz vyžaduje jako parametr proměnnou.

```
#!/bin/bash
echo -n "Zadej sve jmeno: "
read jmeno
echo "Zdravim te $jmeno!"
```

Řídící struktury

Větvení

If

Syntax:

```
if seznam; then seznam; [ elif seznam; then seznam; ] ...
[ else seznam; ] fi
```

Provede se ’if seznam;’. Pokud jeho návratový kód je nulový (OK), provede se ’then seznam;’. Jinak se provede ’elif seznam;’ (...) nebo ’else seznam;’. Návratový kód je kód posledního provedeného procesu nebo 0, pokud se neprovedl žádný příkaz.

```
if test -f /etc/foo
then #Soubor existuje. Zkopíruj tedy soubor
cp /etc/foo .
echo "Hotovo!"
else
# Soubor neexistuje. Vypis tedy chybu
echo "Soubor neexistuje!"
exit
fi
```

Switch

Struktura `case` je velmi podobná struktuře `if`. V základu struktura case je určena pro vykonání jednoho z několika kusů kódu a to podle toho, jakou hodnotu má proměnná uvedená za příkazem case. Zde je příklad struktury case:

```
#!/bin/bash
x=5;     # inicializuje x na hodnotu 5
         # nyní se bude testovat hodnota x:
case $x in
  0) echo "Hodnota x je 0."
  ;;
  5) echo "Hodnota x je 5."
  ;;
  9) echo "Hodnota x je 9."
  ;;
  *) echo "Hodnota x neni ani 0, ani 5, ani 9."
Esac
```

Cykly

Syntax:

```
while seznam; do seznam; done
```

```
until seznam; do seznam; done
```

Seznam ’do seznam;’ se provádí tak dlouho, dokud je návratový kód

- ’while seznam;’ nulový
- ’until seznam;’ nenulový

Struktura `for` se používá, chcete-li během cyklu použít různé hodnoty pro jednu proměnnou, a to tak, že při každém průběhu cyklu bude mít proměnná jinou hodnotu. Například chcete napsat skript, který desetkrát vytiskne každou sekundu tečku (.) lze to provést takto:

```
#!/bin/bash
echo -n "Zjistuji chyby v systemu"
for dots in 1 2 3 4 5 6 7 8 9 10; do
  echo -n "."
sleep 1;
done
echo "System is clear."
```

Zdroje

Seznam

1. VYCHODIL, Vilém. *Operační systém Linux*: *příručka českého uživatele*. 1. vyd. Brno: Computer Press, 2003, 260 s. ISBN 80-722-6333-1.
2. MIKŠÍČEK, Lukáš. PENGUIN.CZ. *OS Linux - 1. část: Seznámení* [online]. [cit. 2012-03-20]. http://www.penguin.cz/noviny/chip/Linux-001.html
3. MITVSEHOTOVO.CZ. *Ubuntu*: *zlo(m) ve vybavení pracovních stanic?* [online]. [cit. 2012-03-20]. http://www.mitvsehotovo.cz/2011/10/ubuntu-zlom-ve-vybaveni-pracovnich-stanic/
4. PRUVODCE-LINUXEM.CZ. Průvodce Linuxem 5: Vybíráme distribuci GNU/Linuxu [online]. [cit. 2012-03-20]. http://www.pruvodce-linuxem.cz/vybirame-distribuci-gnu-linuxu
5. BERAN, Radek. *Operační systém UNIX* [online]. [cit. 2012-03-20]. http://www.beranr.webzdarma.cz/unix.html
6. JANOVSKÝ, Dušan. JAKPSATWEB.CZ. *Jak z Windows pracovat na vzdáleném Linuxu* [online]. [cit. 2012-03-20]. http://www.jakpsatweb.cz/server/ssh-klient.html
7. MILAR, Bohdan. *Seriál o Bashi » Bash 1*: *první část zajímavého a čtivého seriálu o BASHi* [online]. [cit. 2012-03-20]. http://www.linuxexpres.cz/praxe/bash-1-dil-1
8. DQ BLOG. *Crontab v praxi* [online]. [cit. 2012-03-20]. http://blog.dq.cz/informacni-technologie/unix-linux/crontab-v-praxi/
9. ŠÍPOŠ, Juraj. *Naplánujte si úlohy pomocou at* [online]. [cit. 2012-03-20]. http://www.linuxexpres.cz/praxe/naplanujte-si-ulohy-pomocou-at
10. KYSELA, Martin. *Sedláme Linux*: *5. díl: uživatelé a skupiny* [online]. [cit. 2012-03-20]. http://www.zive.cz/Clanky/Sedlame-Linux-5-dil-uzivatele-a-skupiny/sc-3-a-112712/default.aspx
11. JANKŮ, Dominik. *Linux*: *příkazový řádek, díl 2.* [online]. [cit. 2012-03-20]. http://www.postreh.com/phprs/view.php?cisloclanku=2007081501
12. NUC.ELF.STUBA.SK. *Linux documentation Project (CS)*: *Průvodce správce operačního systému Linux* [online]. [cit. 2012-03-20]. http://www.nuc.elf.stuba.sk/lit/ldp/02/020-09.htm

13. MAREK, Libor. *Seminář Linuxu* [online]. [cit. 2012-03-20]. http://www.cmsps.cz/~marlib/g7/
14. NANO-EDITOR.ORG. *Úvod do nano*: *Základy editoru nano* [online]. [cit. 2012-03-20]. http://www.gentoo.org/doc/cs/nano-basics-guide.xml