Linux

Daniel J. Barrett

Kapesní přehled

Linux Kapesní přehled popisuje nejdůležitější a nejužitečnější součásti každodenní práce s Linuxem **stručně, přesně a přehledně** – tak, abyste požadované informace byli schopni vždy rychle najít a ihned uplatnit v praxi. Do obvykle české autorské edice zařazujme tentokrát překlad informacemi nabité a brilantně uspořádané příručky z dílny nakladatelství O'Reilly. Sestavil ji **Daniel J. Barret**, autor uznávaných knih *SSH Kompletní průvodce*, *Bandité na informační dálnici* a *Linux Security Cookbook*. Představuje příkazy a nástroje Linuxu v podstatně srozumitelnější a ucelenější podobě (navíc v češtině) a se zaručenější správností, než bývá obvyklé například v manuálových nebo informačních stránkách.

U každého příkazu vidíte jeho přesný syntaktický zápis, význam, možné parametry, umístění na disku i název balíčku RPM, z něhož byl nainstalován. Zabývá se všemi základními oblastmi:

- Přihlašování a odhlašování
- Disky a souborový systém
- Shell
- Instalace softwaru
- Operace se soubory a adresáři
- Práce s texty
- Tisk
- Řízení procesů
- Uživatelské účty
- Informace o počítači
- Síťová připojení
- Procházení webu
- Elektronická pošta a interaktivní komunikace
- Matematika a výpočty
- Grafika, zvuk a video
- Programování v shell skriptu

Zkušenějším uživatelům Linuxu poslouží jako pohotová referenční příručka, kterou dosud nejspíš postrádali při pátrání po potřebném příkazu či jeho správném zápisu. Samostatnější začátečníci v Linuxu, kteří nepotřebují objasňovat základní části operačního systému, v něm najdou ještě předtím spolehlivého průvodce, který je povede po správné cestě při poznávání možností tohoto systému.

Jste-li pravidelným uživatelem Linuxu – ať už v roli programátora, administrátora či „pouze" běžného uživatele – aspiruje *Linux Kapesní přehled* stát se nejpoužívanější knížkou ve vaší knihovničce!

Zařazení publikace:

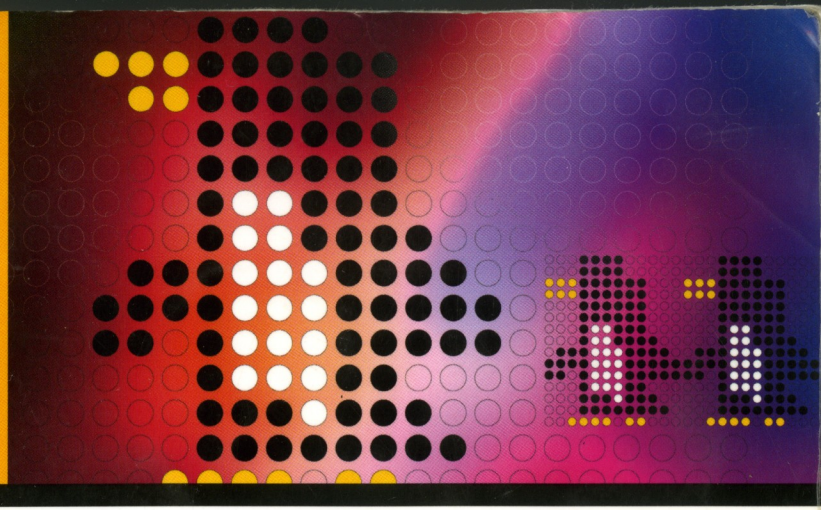
	začátečník	pokročilý	odborník
uživatel		■	■
manažer			■
administrátor		■	■
programátor		■	

ISBN 80-251-0838-4
PRODEJNÍ KÓD K1239
doporučená cena:
169 Kč/249 Kč

Computer Press®, a.s.
nám. 28. dubna 48
635 00 Brno

Objednávejte na:
http://knihy.cpress.cz
distribuce@cpress.cz
Bezplatná tel. linka: **800 555 513**

9 788025 108383



Linux

Daniel J. Barrett

Kapesní přehled

Linuxové příkazy stručně a přesně

Každodenní rádce uživatele, administrátora i programátora

Pro Fedoru i ostatní distribuce

knihy.cpress.cz

Daniel J. Barrett

Linux
Kapesní přehled

Computer Press, a.s.
Brno
2006

Linux
Kapesní přehled

Daniel J. Barrett

Vydavatelství a nakladatelství Computer Press, a.s.,
nám. 28. dubna 48, 635 00 Brno, knihy.cpress.cz

ISBN 80-251-0838-4

Prodejní kód: K1239

Překlad: Lubomír Ptáček
Jazyková korektura: Josef Novák
Vnitřní úprava: Jiří Matoušek, **Sazba:** Vladimír Ludva
Obálka: Martin Sodomka
Komentář na zadní straně obálky: Ivo Magera
Technická spolupráce: Tomáš Zeiner, Pavel Kynický
Odpovědný redaktor: Miroslav Hausknecht
Technický redaktor: Jiří Matoušek
Produkce: Petr Baláš

CP Books, a.s., nám. 28. dubna 48, 635 00 Brno
tel.: 546 122 111, fax: 546 122 112
Objednávejte na: **knihy.cpress.cz, distribuce@cpress.cz**
Bezplatná telefonní linka: **800 555 513**

Obsah

Co v knize naleznete? 7
Co je Linux? 7
Co je Fedora Linux? 8
Co je příkaz? 8
Uživatel a superuživatel 9
Jak číst tuto knihu 9
Vstup a výstup 10
Standardní záhlaví 10
Standardní symboly 11
Užitečný příkaz echo 11
Nápověda 11
Úvodní seznámení s Fedorou 13
K čemu je shell 14
Jak se spouští 14
Přihlášení, odhlášení a ukončení práce systému 14
Systém souborů 15
Domovský adresář 17
Systémové adresáře 17
Cesta, 1. část: kategorie 18
Cesta, 2. část, působnost 19
Cesta, 3. část, aplikace 20
Adresáře operačního systému 20
Ochrana souborů 21
Shell 22
Shell versus programy 22
Některé důležité funkce 23
Hvězdičková konvence 23
Složené závorky 23

Vlnovka („tilda“) 24
Shellové proměnné 24
Vyhledávací cesta 25
Synonyma (alias) 25
Přesměrování vstupů/výstupů 26
Roury 26
Kombinované příkazy 26
Citace 27
Zrušení funkčního významu 27
Editace příkazového řádku 27
Historie příkazů 28
Doplňování jmen souborů 28
Řízení dávek 28
Zrušení prováděného příkazu 30
Ukončení shellu 31
Nastavení shellu 31
Instalace softwaru 31
Soubory tar.gz a tar.bz2 34
Základní operace se soubory 34
Operace s adresáři 37
Zobrazování souborů 39
Vytváření a editace souborů 46
Implicitní editor 47
Atributy souborů 50
Umístění souborů 57
Práce s texty 63
Výkonnější nástroje pro práci s texty 71

Komprimace a balení souborů 72
Porovnávání souborů 75
Disky a souborový systém 80
Rozvrhování a formátování disků 83
Zálohování a vzdálená paměť 84
Výpis souborů 89
Kontrola pravopisu 90
Prohlížení procesů 91
Řízení procesů 95
Uživatelé a jejich okolí 97
Práce s uživatelskými účty 100
Superuživatel 104
Skupiny 104
Základní informace o počítači 106
Umístění počítače 108
Síťová připojení 111
Elektronická pošta 114
Brouzdání po síti 117
Usenet News 121
Okamžité posílání zpráv 123
Výstup na obrazovku 124
Matematika a výpočty 128
Datum a čas 131
Plánování prací 134
Grafika a spořiče obrazovky 138
Audio a video 140

Programování v shell skriptu 142
Mezery a konce řádků 143
Proměnné 143
Vstupy a výstupy 144
Booleovské proměnné a návratové kódy 144
Podmínky 146
Cykly 148
Break a continue 150
Vytváření a spouštění shell skriptů 151
Parametry na příkazovém řádku 152
Ukončení s návratovým kódem 153
Když skript nestačí 153
Závěrečné slovo 154
Poděkování 154
Rejstřík 155

Linux do kapsy

Vítejte v Linuxu! Začátečníkům tato kniha poslouží jako obecný úvod do všech Linuxů, i když konkrétně se v ní popisuje Fedora Linux. Taktéž lze příručku použít k vyhledávání nejčastěji používaných příkazů. Máte-li už s Linuxem nějaké zkušenosti, není třeba se zdržovat úvodem.

Co v knize naleznete?

Kniha je pouze příručkou Linuxu, nikoli jeho úplným manuálem. Obsahuje to nejdůležitější, co je nutné pro práci s Linuxem a co ji usnadňuje. Naopak neobsahuje úplný seznam příkazů se všemi volbami (jestliže jsme vynechali právě váš oblíbený příkaz, omlouváme se) a taktéž se nebudeme zabývat zbytečnými podrobnostmi v samotném operačním systému. Naší snahou je podat vše krátce, příjemně a jen to nejpodstatnější.

Zaměříme se na příkazy, což jsou ta otravná slůvka, která se píší do příkazového řádku, aby se Linuxu řeklo, co má dělat, jako ls (výpis souboru), grep (hledání textu v souboru), xmms (přehrávání zvukového souboru) a df (zjištění volného prostoru na disku). Krátce se zmíníme i o grafických příkazech GNOME a KDE, jejichž úplný popis by sám vystačil na celou knihu.

Materiál je seřazen tak, aby na sebe logicky navazoval. Například chceme-li se naučit zjišťovat obsah souboru, jsou popsány všechny k tomu určené příkazy: cat pro krátké textové soubory, less pro delší soubory, od pro binární soubory, ghostview pro postscriptové soubory atd. Poté si vysvětlíme, jak se příkazy používají s případnými volbami.

Předpokládáme, že máte v Linuxu založen účet a umíte se do systému přihlásit pomocí uživatelského jména a hesla. Pokud ne, poraďte se s vaším správcem systému, anebo, pracujete-li se svým vlastním systémem, použijte účet vytvořený při instalaci systému.

Co je Linux?

Linux je oblíbený počítačový operační systém s volně dostupným zdrojovým kódem, který konkuruje systémům Microsoft Windows a Apple Macintosh. Skládá se ze čtyř hlavních částí:

Jádro systému (kernel)

Základní operační systém pro práci se soubory, disky, sítěmi a ostatními nezbytnými komponentami.

Programy

Tisíce nejrůznějších programů pro práci se soubory, editaci textů, matematické výpočty, sazbu textů, audio, video, programování, tvorbu internetových stránek, kódování, vypalování CD ... zkrátka co si vzpomenete.

Shell

Uživatelské rozhraní pro zadávání příkazů, jejich provádění a zobrazování výsledků těchto příkazů. Existuje v mnoha variantách: Bourne shell, Korn shell, C shell atd. V této knize popisujeme bash neboli Bourne Again Shell, který je nejčastěji implicitním v uživatelských účtech. Všechny tyto shelly mají podobné základní funkce.

X

Grafický systém pro práci s okny, nabídkami (menu), ikonami, s myší a s dalšími grafickými prvky systému. V X je vybudován komplexní grafický systém, přičemž nejoblíbenějšími jsou KDE a GNOME. V celé této knize budeme mluvit o programech, které si při spuštění otvírají svoje vlastní X okno.

Co je Fedora Linux?

Fedora Linux je jedna konkrétní distribuce (občas se distribuci říká „distro") vytvořená firmou Red Hat, Inc. jako tzv. Fedora projekt (viz *http://fedora.redhat.com*). Dříve se tato distribuce nazývala Red Hat Linux.* Popis v této knize odpovídá první oficiální verzi nazývané Fedora Core 1 (listopad 2003). Zaměříme se na programy, jež jsou součástí systému, a na shell s krátkými odbočkami k X a k jádru systému, pokud to bude nutné.

Co je příkaz?

Linuxový příkaz se typicky skládá z jména programu, voleb a parametrů a zapisuje se v shellu. Jméno programu vlastně odkazuje na program uložený někde na disku, shell jej podle tohoto jména nalezne a spustí. Volby jsou nejčastěji uvozeny krátkou pomlčkou (spojovník, znaménko mínus) a upřesňují konkrétní funkci programu, pomocí parametrů se pak obvykle zadávají vstupy a výstupy. Například příkaz pro zjištění počtu řádků v souboru:

```
$ wc -l můjsoubor
```

se skládá z jména programu (`wc`, word count), volby (`-l`), která říká, že se mají počítat řádky a parametru (myfile), kterým se zadává vstupní soubor. (Znak dolaru $ je promptem shellu neboli znakem, jímž shell signalizuje, že čeká na zadání příkazu). Samostatných voleb lze zadat více:

```
$ můjprogram -a -b -c můjsoubor
```

Zadání samostatných voleb

anebo lze použít kombinované zadání s jedinou pomlčkou:

```
$ můjprogram -abc můjsoubor
```

Totéž jako -a -b -c

i když některé programy se mohou chovat jinak a kombinaci voleb neumožňují.

* Red Hat se nyní zaměřuje na distribuce s označením Enterprise Linux, což jsou špičkové komerční aplikace. Naše kniha je ovšem použitelná i pro takové distribuce.

Příkazy ovšem mohou být složitější a neslouží jen ke spuštění jediného programu:

- Může se jimi spustit více programů současně buď sekvenčně (postupně jeden za druhým) nebo formou roury („pipeline"), tj. tak, že výstup jednoho programu je vstupem pro jiný program.
- Volby nejsou standardizované. Táž volba (např. `-l`) může mít v různých programech odlišný význam: v příkazu `wc -l` znamená „spočítat textové řádky", zatímco v `ls -l` to znamená „delší řádky na výstupu". Naopak, u dvou různých programů mohou dvě různé volby mít stejný význam, např. `-q` i `-s` znamená „tichý chod".
- Ani parametry nejsou standardizované. Jsou jimi nejčastěji vstupní a výstupní soubory, mohou mít i jiný význam: adresáře, regulární výrazy apod.
- V uživatelském rozhraní v Linuxu – *shell* – je možno používat i programovací jazyk. Je tedy možno zadat např. nejen „spusť tento program", nýbrž i „jestliže je dnes úterý, spusť tento program, jinak zadej provádění jiného příkazu, a to šestkrát za sebou pro každý soubor s příponou *.txt*".

Uživatel a superuživatel

Linux je multiuživatelský operační systém. Na daném počítači je každý uživatel identifikován jedinečným uživatelským jménem jako „novák" nebo „trpaslík" a má (víceméně) soukromý přístup k výpočetní kapacitě systému. Kromě toho existuje ještě zvláštní uživatel jménem root, který může dělat v celém systému úplně vše. Přístup běžného uživatele je omezen: může sice spouštět většinu programů v systému, avšak měnit může pouze svoje vlastní soubory. Naopak, superuživatel může vytvářet, měnit i rušit libovolné soubory a může spouštět libovolný program.

Některé příkazy uvedené v této příručce může zadávat pouze superuživatel. V tomto případě se pro odlišení jako shellovský prompt užívá znak # (dvojitý kříž):

```
# příkaz je zde
```

zatímco u normálního uživatele se používá prompt $ (dolar):

```
$ příkaz je zde
```

Superuživatelem se lze stát bez odhlášení a nového přihlášení; stačí zadat příkaz su (viz popis „Superuživatel" na str. 104) a poté zadat správné heslo.

```
$ su -l
Password: ********
#
```

Jak číst tuto knihu

Příkaz je nejdříve popsán v obecném tvaru. Například obecný tvar příkazu `wc` (čítač slov) vypadá takto:

```
wc [volby] [soubory]
```

což znamená, že by se nejdřív mělo napsat „wc“, potom případné volby a nakonec jména souborů. Hranaté závorky nejsou součástí příkazu: ty pouze označují volitelný obsah; slova zapsaná kurzivou zase znamenají, že se na jejich místo zadávají uživatelské hodnoty, například jména souborů. Svislá čára mezi volbami nebo parametry (většinou společně se závorkami):

```
ls (soubor | adresář)
```

znamená výběr jedné z možností. Parametrem příkazu ls může tedy být buď soubor, nebo adresář.

Vstup a výstup

Většina linuxových programů čte data ze standardního vstupu, což je obvykle klávesnice, a výstupní data posílá na standardní výstup, což je většinou monitor. Dále, chyby se vypisují na standardním chybovém výstupu, což je opět nejčastěji monitor, avšak není totožný se standardním výstupem*. Později si ukážeme, jak lze přesměrovat standardní výstup, vstup a chybový výstup do nebo ze souborů a front. Bylo by ovšem nyní vhodné si některé výrazy upřesnit. Řekne-li se, že příkaz „čte“, míní se tím, že čte ze standardního vstupu, pokud to nezměníme. A když se příkazem „vypisuje“, míní se tím opět na standardní výstup, a nemusí to být právě tiskárna.

Standardní záhlaví

Každý popis příkazu začíná standardním záhlavím:

ls [*volby*] [*soubory*] **coreutils**

/bin **stdin stdout -file -opt -help -version**

Záhlaví obsahuje jméno příkazu (ls) a případné další součásti příkazu, adresář, ve kterém je umístěn (/bin), RPM balík, z něhož se příkaz instaluje (coreutils), a šest atributů příkazu v černé sazbě (podporované) nebo v šedé sazbě (nepodporované):

stdin

Příkaz čte ze standardního vstupu, tedy implicitně z klávesnice

stdout

Příkaz zapisuje do standardního výstupu, tedy implicitně na monitor

-file

Pomlčka na místě jména vstupního souboru znamená čtení ze standardního vstupu; podobně je-li pomlčka na místě výstupního souboru, zapisuje se na standardní výstup. Např. následující příkaz wc (počet slov) čte ze souborů *file1* a *file2*, pak ze standardního vstupu, pak z *file3*:

```
$ wc soub1 soub2 - soub3
```

* Například přesměruje-li se standardní výstup do souboru, chybová hlášení se i nadále vypisují na monitor.

--opt

Dvě pomlčky na místě volby „--“ znamenají „konec voleb“: cokoli se nachází za nimi, už není volba. Používá se to např. při práci se souborem, jehož jméno začíná pomlčkou, která by se jinak (chybně) považovala za volbu. Např. jmenuje-li se soubor *-foo*, příkaz `wc -foo` je chybný, protože *-foo* se bude považovat za (chybnou) volbu. Správně se musí zadat `wc -- -foo`. Pokud možnost zadat „--“ v příkazu neexistuje, lze před jméno souboru předřadit cestu „./“, takže pomlčka už není na prvním místě:

```
$ wc ./-foo
```

--help

Volbou `--help` se vypíše nápověda se správným použitím příkazu a příkaz se ukončí.

Standardní symboly

Některé klávesy se v textu označují speciálními symboly. Podobně jako ve většině dokumentů o Linuxu, i zde se znak ^ („stříška“) používá k označení stisknutí a přidržení klávesy Ctrl (control). Např. `^D` (čte se „kontrol D“) znamená „stiskni a přidrž klávesu Ctrl a současně stiskni klávesu D“. Stisknutí klávesy Esc (escape) se značí ESC. Stisknutí kláves „enter“ a „mezera“ není třeba vysvětlovat.

Užitečný příkaz echo

V mnoha příkladech se vypisují informace na monitoru pomocí příkazu `echo`. Formálně je tento příkaz popsán na str. 124. `echo` je jeden z nejjednodušších příkazů: zpracuje se tak, že se pouze vypíší parametry na standardní výstup.

```
$ echo Náš pes má blechy
Náš pes má blechy
$ echo Jmenuji se $user          shellovská proměnná USER
Jmenuji se novak
```

Nápověda

Podrobnější informace, které nenaleznete v této knize, lze získat několika způsoby:

Pomocí příkazu man

Příkazem man se zobrazí stránka manuálu, na které je popsán daný program. Například dokumentaci o příkazu `ls` získáme příkazem:

```
$ man ls
```

Stránky manuálu vztahující se k určitému tématu se zobrazí zadáním volby `-k` následované klíčovým slovem.

Pomocí příkazu info

Příkaz info je rozšířený, hypertextový systém nápovědy a zahrnuje mnoho linuxovských programů.

```
$ info ls
```

Pokud se k danému programu nenalezne žádná jiná dokumentace, zobrazí se příslušné stránky z manuálu. Chceme-li zjistit, jaká dokumentace je k dispozici, zadá se pouze `info`. Návod k použití příkazu info získáme pomocí příkazu `info info`.

Příkaz --help (pokud existuje)

Řada linuxových příkazů reaguje na volbu `--help` výpisem krátké zprávy s nápovědou. Tuto volbu je nutno vyzkoušet:

```
$ ls --help
```

Prohlížení adresáře /usr/share/doc

Tento adresář obsahuje dokumentaci k většině programů, obvykle označenou jménem programu a verzí. Například soubory textového editoru Emacs, verze 21.3, jsou uloženy v */usr/share/doc/emacs-21.3*.

Nápověda ke GNOME a KDE

Nápověda ke GNOME a KDE je k dispozici v položce Help v hlavním menu.

Webové stránky o Fedoře

Oficiální www stránky jsou na *http://fedora.redhat.com*. Neoficiální odpovědi na často kladené otázky (FAQ) se přesunuly na *http://fedora.artoo.net*. Navíc pochopitelně existují www stránky této knihy:

http://www.oreilly.com/catalog/linuxpg/

Diskusní skupiny

Linuxu je věnováno mnoho diskusních skupin, např. *comp.oslinux.misc* nebo *comp.oslinux.questions*. Informace zaměřené na Red Hat Linux jsou na *alt.os.linux.redhat*, *comp.os.linux.redhat*, *linux.redhat* a *alt.os.linux.redhat.misc*. Diskusní skupiny lze nalézt také na stránkách Google Groups, *http://groups.google.com*, jež jsou nejlepším zdrojem informací při potížích všeho druhu.

Google

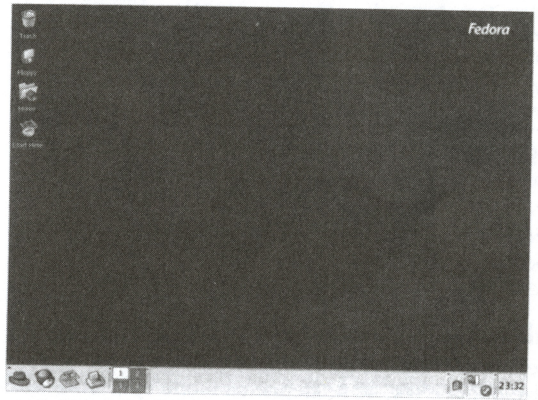
Další dokumentaci a návody lze nalézt na *http://google.com* (to jen pro ty, kteří dosud žili na pustém ostrově).

Úvodní seznámení s Fedorou

Když se přihlásíte do linuxového systému Fedora (či kteréhokoli jiného), na monitoru vás uvítá nejspíš pracovní plocha* podobná té, která je na obrázku 1. Najdete na ní:

- Lištu na spodním okraji, která obsahuje:
  - Ikonu RedHat vlevo dole. Klepne-li se na ni, aktualizuje se hlavní menu programů
  - Ikony různých programů, např. webového prohlížeče Mozilla, poštovního programu Evolution a aplikace Print Manager pro konfiguraci tiskáren
  - Přepínač pracovních ploch (čtvereček se čtyřmi okénky), který umožňuje přechody mezi různými plochami
  - Modré znaménko, které značí, že systémový software je aktualizovaný, anebo znaménko červené, když tomu tak není
  - Hodiny
- Další ikony na pracovní ploše, jako např. popelnici na odstraněné soubory, disketovou mechaniku anebo domovský adresář na ukládání uživatelských souborů.

Fedora poskytuje několik podobně vyhlížejících rozhraní, na obrázku je tvar GNOME nebo KDE**. Rozdíl lze zjistit klepnutím na ikonu RedHat, zobrazí se hlavní menu a zadá se help. V nápovědním okně je zřetelně uvedeno, zda jde o GNOME či KDE.



Obrázek 1. *Grafické pozadí Fedory*

* Pokud se nepřihlašujete po síti. V takovém případě se zobrazí jenom příkazový řádek s promptem.

** Rozhraní závisí na konkrétní konfiguraci. GNOME a KDE jsou snadno konfigurovatelné, přičemž Fedora obsahuje ještě třetí rozhraní, twm, s odlišným vzhledem. Jiné Linuxy mají ještě další rozhraní.

K čemu je shell

Nejdříve je třeba si prohlédnout ikony a menu v GNOME a KDE. Někteří uživatelé vystačí při práci s Linuxem pouze s grafickým rozhraním. Různé distribuce, Fedoru nevyjímaje, mají ovládání rozhraní tak jednoduché, že jím lze editovat soubory, číst elektronickou poštu a pohodlně prohlížet Internet.

Nicméně, skutečná síla Linuxu se skrývá „za scénou". K využití všech jeho možností je třeba co nejlépe zvládnout shell. Zpočátku může být používání shellu obtížnější než práce s ikonami a s menu, ale jakmile se shell naučíte, zjistíte, že je to snadný a velmi mocný prostředek. Podstatná část této knihy je věnována právě příkazům zadávaným právě prostřednictvím shellu.

Jak se spouští

Shell se v grafických rozhraních (GNOME, KDE a jiných) spouští otevřením *okna shellu*. Pomocí programů např. `xterm`, `gnome -terminal`, `konsole` nebo `uxterm`. Všechny tyto programy udělají jedinou věc: otevřou okno a spustí v něm shell, který čeká na vstup. Pomocí tří implicitních fedorových rozhraní se shell spouští takto:

Rozhraní	Co udělat ...	... aby se spustil shellový program
GNOME	Menu: System Tools: terminal *nebo na pracovní ploše:* Pravé tlačítko myši: Open Terminal	`gnome -terminal`
KDE	Menu: System Tools: terminal *nebo na pracovní ploše:* Pravé tlačítko myši: Open Terminal	`konsole`
twm	*Na pracovní ploše:* Pravé tlačítko myši: XTerm	`xterm`

Nesmí se směšovat program pro vytvoření okna (např. `konsole`) se shellem, který v něm běží. Okno je jenom rámec – i když s pozoruhodnými vlastnostmi – zatímco shell je to, čím se zadávají a provádějí příkazy.

Pokud uživatel nepoužívá grafické rozhraní – řekněme, že je přihlášen ze vzdáleného terminálu po síti nebo i přímo na připojeném terminálu – shell se spustí ihned po přihlášení, aniž by bylo třeba otevírat shellové okno.

Přihlášení, odhlášení a ukončení práce systému

Předpokládejme, že víte, jak se v Linuxu přihlásit k určitému účtu. Z programů GNOME a KDE se lze odhlásit tak, že se klepne na ikonu RedHat na liště a z hlavního menu se vybere Logout. Na vzdáleném terminálu stačí ukončit shell (příkazem exit nebo logout).

V Linuxu by se práce nikdy neměla ukončovat síťovým vypínačem nebo vytažením šňůry ze zásuvky – chce to jemnější metodu. V GNOME se práce ukončí pomocí Logout → Shutdown. V KDE je třeba se nejdříve odhlásit a pak se na odhlašovací obrazovce klepne na ikonu Shutdown. Systém ze shellu musí příkazem shutdown ukončit superuživatel takto:

shutdown [*volby*] *time* [*zpráva*] **SysVinit**

/bin **stdin** **stdout** **-file** **-opt** **-help** **-version**

Příkazem shutdown se ukončí nebo zrestartuje systém; může jej spustit pouze superuživatel. Takto vypadá příkaz, kterým se po 10 minutách ukončí práce systému a na všech terminálech, kde je někdo přihlášen, se vypíše zpráva „plánovaná údržba":

```
# shutdown -h +10 "plánovaná údržba"
```

Time může být počet minut se znaménkem +, anebo absolutní časový údaj jako 16:25, anebo slovo *now* značící okamžité ukončení.

Pokud není uvedena žádná volba, příkazem `shutdown` se systém nastaví na speciální jednouživatelský režim údržby, v němž je přihlášena na systémové konzole jen jedna osoba a ostatní nepodstatné funkce se neprovádějí. Tento režim se ukončí buď dalším příkazem shutdown, jímž se systém buď úplně ukončí, nebo znovu odstartuje, anebo se systém příkazem `^D` vrátí zpět do multiuživatelského režimu.

Užitečné volby

`-r`	Restart systému
`-h`	Ukončení systému
`-k`	Vtípek: pouze se vypíše všem uživatelům zpráva jako při ukončení, avšak ve skutečnosti se systém neukončí
`-c`	Zrušení probíhajího ukončení (bez parametru *time*)
`-f`	Při restartu se přeskočí kontrola souborů, kterou provádí program fsck (viz „Disky a souborový systém" na str. 80).
`-F`	Při restartu se provede kontrola souborů

Další podrobnosti o příkazu shutdown, jednouživatelském režimu a stavech systému jsou uvedeny v manuálu pod init a inittab.

Systém souborů

Pro práci s Linuxem je nutno znát soubory v Linuxu a jejich strukturu. Soubory jsou uloženy v *adresářích*. Můžeme si je představit jako složky v systémech Windows anebo Macintosh. Jsou uspořádány hierarchicky do *stromů*: adresář může obsahovat jiný adresář nazývaný *podadresář*, který znovu může

obsahovat jiné soubory a podadresáře atd. až do nekonečna. Adresář na vrcholu se nazývá kořenový adresář a označuje se lomítkem (/)*.

Syntaxe odkazů na soubory a adresáře je tvořena jmény a lomítky a nazývá se *cesta*. Například cesta:

```
/jeden/dva/tri/ctyri
```

se odkazuje na kořenový adresář /, který obsahuje adresář zvaný *jeden*, ten zase obsahuje adresář jménem *dva*, ten obsahuje adresář *tri* a nakonec v něm je obsažen adresář nebo soubor s názvem *ctyri*. Začíná-li cesta kořenovým adresářem, mluvíme o tzv. *absolutní* cestě, pokud ne, jde o *relativní* cestu. Více o tom za okamžik.

Shell „pracuje" v určitém adresáři (v abstraktním slova smyslu), který se nazývá *pracovní* neboli *běžný adresář* a příkazy, které se mu zadávají, používají relativní (znovu to slovo) odkazy vzhledem k tomuto pracovnímu adresáři. Přesněji, použije-li se relativní cesta v shellu, chápe se jako relativní v daném pracovním adresáři. Např. pracuje-li shell v adresáři */jeden/dva/tri* a spustí se příkaz odkazující se na soubor *mujsoubor*, jde ve skutečnosti o soubor */jeden/dva/tri/mujsoubor*. Podobně, relativní cesta *a/b/c* znamená ve skutečnosti cestu */jeden/dva/tri/a/b/c*.

Existují dva speciální adresáře . (tečka) a .. (dvě tečky za sebou). Prvním se označuje běžný adresář, druhým rodičovský adresář, tedy adresář o jednu úroveň vyšší. V adresáři */jeden/dva/tri* tedy . znamená tento adresář a .. znamená adresář */jeden/dva*.

Běžný adresář shellu lze změnit příkazem cd:

```
$ cd /jeden/dva/tri
```

Přesně řečeno, tímto příkazem se nastaví pracovní adresář na hodnotu */jeden/dva/tri*. To je absolutní změna (neboť adresář začíná znakem „/"); lze pochopitelně použít i relativní adresování:

```
$ cd d                  Nastav podadresář d
$ cd ../mujadresar      Jdi o úroveň výš (rodič) a pak do adresáře mujadresar
```

Jména souborů mohou obsahovat nejrůznější znaky: malá a velká písmena**, čísla, mezery, pomlčky, podtržítka a další, (nikoli ovšem lomítko „/", které slouží jako oddělovač adresářů). Obecně se ovšem nedoporučuje používání mezer, hvězdiček, závorek a dalších znaků, které mají nějakou funkci v shellu. Takové znaky se musejí dávat do úvozovek, čímž si uživatel zbytečně komplikuje situaci (viz „Citace" na str. 27).

* V Linuxu vycházejí *všechny* soubory a adresáře z kořenového adresáře. V tom se Linux liší od systémů Windows a DOS, kde se jednotlivá zařízení označují písmeny.

** V Linuxu se v názvech souborů a adresářů malá a velká písmena rozlišují

Domovský adresář

Uživatelské soubory se často ukládají do adresáře */home* (obyčejný uživatel) nebo */root* (superuživatel). Typický domovský adresář je */home/uzivatelskejmeno*: */home/novak*, */home/prochazka* atd. Existuje několik způsobů, jak je možno se odkázat na domovský adresář.

cd

Příkazem `cd` bez parametru se lze vrátit (tj. nastavit pracovní adresář shellu) do domovského adresáře

HOME proměnná

Proměnná `HOME` (viz „Shellové proměnné" na str. 24) obsahuje jméno domovského adresáře.

```
$ echo $HOME      Vypíše se parametr příkazu echo
/home/novak
```

~

Vlnovka se v adresáři expanduje shellem do jména domovského adresáře.

```
$ echo ~
/home/novak
```

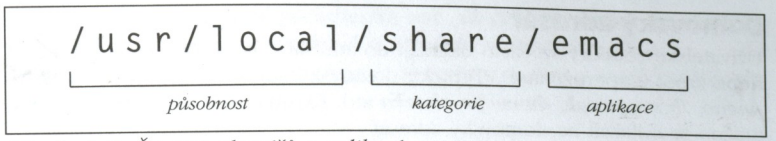
Pokud za vlnovkou následuje uživatelské jméno (např. ~novak), dosadí se místo ní domovský adresář:

```
$ cd ~novak
$ pwd             Příkaz pro výpis pracovního adresáře
/home/novak
```

Systémové adresáře

Typický Linux má desítky tisíc systémových adresářů. V nich jsou uloženy soubory operačního systému, aplikací, dokumentace a vše možné až na uživatelské soubory (které „přežívají" v adresáři */home*).

Nejste-li správcem systému, systémové adresáře poctíte svojí návštěvou jen velice zřídka. Pokud ovšem o nich něco málo budete vědět, můžete pochopit nebo uhodnout, k čemu jsou. Jejich jména se často skládají ze tří částí, které budeme nazývat působnost, kategorie a aplikace (to nejsou standardně užívané termíny, ale pomohou k lepšímu porozumění). Například adresář */usr/local/share/emacs*, který obsahuje lokální data textového editoru Emacs, má působnost */usr/local* (lokálně nainstalované systémové soubory), kategorii *share* (data a dokumentace jsou specifické pro daný program) a aplikací je *emacs* (textový editor) uvedený na obrázku 2.



Obrázek 2. Řazení adresářů a aplikací

Cesta, 1. část: kategorie

Kategorie určuje typy souborů nalezené v adresáři. Např. je-li kategorie *bin*, adresář zcela jistě obsahuje programy. Zde je seznam možných kategorií:

Kategorie programů

bin Programy (obvykle binární soubory)

sbin Programy (obvykle binární soubory), které by měl spouštět superuživatel, root

lib Knihovny kódů pro programy

libexec Programy volané jinými programy, nikoli nutně uživatelskými; jinak řečeno „knihovna spustitelných programů"

Kategorie dokumentace

doc Dokumentace

info Dokumentace pro zabudovaný systém nápovědy Emacs

man Dokumentační soubory (stránky manuálu) zobrazované programem man; soubory bývají komprimované nebo zalomené pomocí programu man

share Speciální soubory, např. příklady nebo pokyny k instalaci

Kategorie konfigurace

etc Konfigurační soubory systému (a jiné blíže neurčené soubory)

init.d Konfigurační soubory pro natažení Linuxu; taktéž *rc1.d, rc2.d, ...*
rc.d

Kategorie programování

include Hlavičkové soubory

src Zdrojové soubory

Kategorie webové soubory

cgi-bin Skripty/programy, které běží na www stránkách

html www stránky

public_html www stránky, nejčastěji v uživatelském domovském adresáři

www www stránky

Kategorie výpisů

fonts Fonty (kdo by to byl řekl!)

X11 Soubory systému X window

Kategorie hardware

dev Soubory realizující disková a jiná hardwarová rozhraní

mnt Adresáře pro přístup na disk

misc

Kategorie prováděných souborů

var Pomocné soubory pro zajištění činnosti počítače

lock Uzamykací soubory vytvořené běžícími programy, jejichž prostřednictvím sdělují ostatním programům „jsem v běhu" kvůli synchronizaci akcí s jinými programy nebo s opakovaným spouštěním sebe sama

log Soubory sloužící ke sledování důležité systémové události; obsahují chybová hlášení, varování a zprávy

mail Schránky pro příchozí poštu

run Soubory PID obsahující ID běžících procesů; často se používají ke sledování nebo ukončení procesů

spool Soubory, které jsou ve frontě nebo se právě přenášejí, jako např. odchozí pošta, tisky a plánované procesy

tmp Dočasné soubory pro programy i pro uživatele

proc Stav operačního systému: viz „Adresáře operačního systému" na str. 20

Cesta, 2. část, působnost

Působnost cesty se vztahuje k jejímu vrcholu a vyjadřuje se jí účel podřízené adresářové struktury. Několik příkladů nejčastějších působností:

/ Systémové soubory Linuxu (kořenové soubory, „root")

/usr Další systémové soubory Linuxu

/usr/games Hry (opět překvapení!)

/usr/kerberos Soubory náležející autentizačnímu systému Kerberos

/usr/local Lokální systémové soubory určené pro soukromé účely nebo pro daný počítač

/usr/X11R6 Soubory náležející systému X window

Taktéž pro kategorie jako *lib* (knihovny) může mít Linux adresáře */lib*, */ust/lib*, */usr/local/lib*, */usr/games/lib* a */usr/X11R6/lib*. Mohou existovat i adresáře s jinou působností, které patří správci systému: */my-company/lib*, */my-division/lib* atd.

Mezi / a */usr* ve skutečnosti nevede jasná hranice, pouze intuitivně / má „nižší úroveň" a je blíže k operačnímu systému. Adresář */bin* tedy obsahuje základní programy jako `ls` a `cat`, zatímco v */usr/bin* je množství aplikací patřící ke konkrétní linuxové distribuci a v */usr/local/bin* jsou další programy dle výběru správce systému. Pro využívání adresářů neexistují pevná pravidla a umísťování souborů se řídí spíše zvyky.

Cesta, 3. část, aplikace

Aplikační částí cesty bývá obvykle jméno programu. V rámci působnosti a kategorie (řekněme */usr/local/doc*) mohou mít programy své vlastní podadresáře (např. */usr/local/doc/ mujprogram*), kam si ukládají pomocné soubory.

Adresáře operačního systému

/boot

Soubory sloužící ke startování systému. Obsahují mimo jiné kernel, obvykle s názvem */boot/vmlinuz*.

/lost+found

Poškozené soubory, které se systém pokusil obnovit.

/proc

Seznam běžících procesů (pro pokročilé uživatele)

Soubory v */proc* umožňují sledování činnosti kernelu a mají speciální vlastnosti. Jeví se, jako kdyby měly nulovou velikost, byly určeny jen pro čtení a byly právě vytvořeny.

```
$ ls -l /proc/version
-r--r--r--   1 root   root   0 Oct 3 22:55
   /proc/version
```

Jejich obsah ovšem obsahuje informace o kernelu:

```
$ cat /proc/version
Linux version 2.4.22-1.2115.nptl
```

Tyto soubory většinou slouží programům k jejich činnosti. Máte-li zájem o další podrobnosti, zde je pár příkladů:

/proc/ioports	Seznam vstupních a výstupních zařízení
/proc/version	Verze operačního systému. Stejnou informaci lze získat příkazem uname.
/proc/uptime	Doba činnosti systému od jeho spuštění v sekundách. Příkazem uptime získáme tento údaj v čitelnější podobě.
/proc/nnn	Pokud je nnn celé kladné číslo, systém poskytne informace o běžícím procesu s ID nnn.
/proc/self	Informace o právě běžícím uživatelském procesu; je to vlastně průběžně aktualizovaný ukazatel na soubor /proc/nnn.

Zadáme-li příkaz `ls -l /proc/self` několikrát za sebou, můžeme sledovat, jak se údaje o procesu průběžně mění.

Ochrana souborů

V systému může existovat množství uživatelských účtů. Soukromí a bezpečnost jsou v systému zajištěny tak, že uživatel má přístupové právo pouze k některým souborům. Mechanismus přístupových práv lze charkterizovat dvěma otázkami:

Kdo má přístupové právo? Každý soubor a adresář má *vlastníka*, který s ním může libovolně nakládat. Uživatel, který soubor vytvořil, bývá obvykle i jeho vlastníkem, avšak vztah mezi souborem a uživatelem může být i složitější.

Dále, povolení k přístupu k souboru lze stanovit pro určitou skupinu uživatelů. Skupiny definuje správce, tento postup je popsán v části „Skupiny" na str. 104.

Konečně, každý soubor nebo adresář může být přístupný *všem uživatelům*, kteří mají účet v daném systému. Množina všech uživatelů se také někdy nazývá svět (*world*) nebo ostatní (*others*).

Jaké druhy přístupových práv existují? Každý uživatel označený jako vlastník, skupina nebo svět může mít oprávnění ze souboru číst, do souboru zapisovat, modifikovat jej, anebo ho spouštět. Povolení platí analogicky i pro adresáře, tj. v adresáři lze soubory číst, psát (tedy soubory vytvářet a rušit) nebo spouštět.

Vlastnictví k souborům a přístupová práva lze zjistit příkazem:

```
$ ls -l jmeno_souboru
```

V případě adresáře pak příkazem:

```
$ ls -ld jmeno_adresare
```

Přístupová práva jsou určena prvními 10 znaky na výstupu. Výstupní řetězec obsahuje písmena `r` (číst), `w` (psát), `x` (spustit) a i jiná písmena. Například:

```
drwxr-x---
```

Význam písmen a znaků v řetězci:

Umístění	Význam
1	Typ souboru: `-` = soubor, `d` = adresář, `l` = symbolický odkaz, `p` = pojmenovaná roura, `c` = zařízení pro přenos znaků, `b` = zařízení pro přenos bloků
2–4	Přístupová práva vlastníka: čtení, zápis, spuštění
5–7	Přístupová práva skupiny: čtení, zápis, spuštění
8–10	Přístupová práva ostatních: čtení, zápis, spuštění

Příkaz `ls` je podrobněji popsán v „Základních operacích se soubory“ na str. 34. Změnit vlastníka, skupinu nebo přístupová práva lze pomocí příkazů `chown`, `chgrp` a `chmod`, jež jsou popsány v části „Atributy souborů“ na str. 50.

Shell

Chceme-li v Linuxu zadávat příkazy, musíme je nejdříve někde zapsat. To „někde“ se nazývá shell, a je to uživatelské rozhraní pro zadávání linuxových příkazů: příkaz se zadá, stiskne se enter a shell spustí program (nebo programy), který daný příkaz provede. Spouštění shellu je popsáno v „Úvodním seznámení s Fedorou“ na str. 13.

Např. chceme-li se podívat, kdo je přihlášen v systému, zadá se příkaz:

```
$ who
novak     :0     Sep 23   20:44
prochazka         pts/0    Sep 15   13:51
hausknecht        pts/1    Sep 22   21:15
hausknecht        pts/2    Sep 22   21:18
```

(Znaménko $ je promptem shellu označujícím, že shell je připraven provést příkaz). Jediným příkazem lze spustit i několik nezávislých programů v témže okamžiku, anebo dokonce vyvolat jejich součinnost. Následujícím příkazem se přesměruje výstup programu `who`, který se stane vstupem programu `wc`, jenž počítá řádky textu v souboru; výsledkem je počet řádků na výstupu příkazu `who`:

```
$ who | wc -l
4
```

což je vlastně údaj o tom, kolik je v daném okamžiku připojených uživatelů*. Svislá čára se nazývá *roura* a spojuje who a wc.

Shell sám je linuxovým programem, jakých je v systému celá řada. Zaměříme se na Bash („Bourne-Again Shell“), který je uložen v adresáři */bin/bash*. Ve Fedoře je v tomto adresáři uložen implicitně.

Shell versus programy

Zadá-li se příkaz, může se spustit linuxový program (např. who), anebo se provede příkaz přímo zabudovaný v shellu. Při zadávání příkazu to můžeme rozlišit pomocí příkazu type:

```
$ type who
who is /usr/bin/who
$ type cd
cd is a shell builtin
```

Je užitečné vědět, které příkazy provádí shell a které systém. V následujících odstavcích se budeme věnovat shellu.

* Ve skutečnosti je to počet interaktivně spuštěných shellů. Má-li jich jeden uživatel spuštěných několik, jako uživatel hausknecht v našem příkladu, vypíší se na výstupu příkazu who každý na jednom řádku.

Některé důležité funkce

Shell umí mnohem víc než jenom provádět příkazy. Disponuje prostředky, které práci podstatně zjednodušují. Lze používat např. hvězdičkovou konvenci („wildcards“) pro zadávání jmen souborů, přesměrování vstupu a výstupu příkazu do souborů, roury na předávání výtsupů příkazů na vstup jiných příkazů, alternativní jména neboli synonyma („aliasy“) pro rychlejší zadávání příkazů, proměnné pro ukládání hodnot v shellu atd. O těchto možnostech se zmíníme jen stručně, kompletní dokumentaci lze získat pomocí příkazu info bash.

Hvězdičková konvence

Pomocí hvězdičkové konvence se lze zkráceně odkazovat na podobná jména souborů. Např. `a*` znamená všechny soubory začínající písmenem (malé) „a“. Hvězdička se nahradí znaky, jež společně s písmenem „a“ tvoří skutečná jména existujících souborů. Zadá-li se:

```
$ ls a*
```

naleznou se všechny soubory v pracovním adresáři začínající písmenem a, jako kdyby se zadal příkaz:

```
$ ls auto andula apetit
```

přičemž těmito parametry se nahradí zástupný znak * ještě před provedením příkazu `ls`, takže samotný příkaz vlastně o hvězdičkové konvenci nic neví.

Zástupný znak	Význam
*	Libovolná množina znaků kromě úvodní mezery
?	Libovolný znak
`[množina]`	Libovolný znak z uvedené množiny, nejčastěji z posloupnosti znaků, např. všechny krátké samohlásky `[aeiouyAEIOUY]`, nebo interval naznačený pomlčkou `[A-Z]`, tj. všechna velká písmena
`[^množina]` `[!množina]`	Libovolný znak, který není z uvedené množiny (dle pravidel uvedených v předchozím případě)

Pokud má zadaná množina obsahovat pomlčku (-), je nutno ji uvést na začátku nebo na konci posloupnosti. Hranatá závorka může být pouze na začátku posloupnosti, naopak znaky ^ a ! nesmějí být na první pozici.

Složené závorky

Podobně jako při hvězdičkové konvenci lze zadávat vícenásobné parametry i pomocí složených závorek. Výraz (čárka je oddělovačem):

```
{a,b,cc,ddd}
```

expanduje do:

```
a b cc dddd
```

Rozdíl mezi hvězdičkovou konvencí a složenými závorkami spočívá v tom, že v druhém případě se pracuje s libovolnými řetězci, zatímco hvězdička se týká jenom jmen souborů. Například `bambo{X,Y,ZZZ}rák` se nahradí:

```
$ echo bambo{X,Y,ZZZ}rák
bamboXrák bamboYrák bamboZZZrák
```

bez ohledu na to, jaké soubory jsou v běžném adresáři.

Vlnovka („tilda")

Samostatně stojící vlnovka (~) nebo vlnovka na začátku slova je rovněž speciálním znakem

~	Vlastní domovský adresář
~novak	Domovský adresář uživatele novaka

Shellové proměnné

Proměnné a jejich hodnoty se definují přiřazením:

```
$ MOJEPROM=3
```

Na proměnné se odkazujeme tak, že před její název předřadíme $ (znak dolaru):

```
$ echo $MOJEPROM
3
```

Některé standardní proměnné jsou v rámci shellu předdefinované.

Proměnná	Význam
DISPLAY	Jméno výstupu X window
HOME	Jméno domovského adresáře
LOGNAME	Jméno přihlášeného
MAIL	Cesta schránky příchozí pošty
OLDPWD	Předchozí adresář v shellu
PATH	Cesta pro vyhledávání: adresáře jsou odděleny dvojtečkou
PWD	Běžný adresář
SHELL	Cesta k shellu, tj. */bin/bash*
TERM	Typ terminálu, tj. xterm nebo vt100
USER	Jméno přihlášeného

Proměnné shellu se zobrazí příkazem:

```
$ printenv
```

Působnost proměnné (tj. kde ji programy „vidí") je implicitně omezena na ten shell, v němž je definovaná. Jiným programům, které spouští shell („subshelly") ji lze zpřístupnit pomocí příkazu export:

```
$ export MOJEPROM
```

anebo zkráceně včetně přiřazení hodnoty:

```
$ export MOJEPROM=3
```


Proměnná se nyní nazývá *vnější proměnná*, neboť je přístupná i jiným programům v rámci daného shellu. Má-li se určitá hodnota zpřístupnit jinému programu pouze jednorázově, přiřazení se provede na daném příkazovém řádku:

```
$ echo $HOME
/home/novak
$ HOME=/home/mařenka echo "Můj adresář je $HOME"
Můj adresář je mařenka
$ echo HOME
/home/novak                    Původní hodnota proměnné se nezmění
```

Vyhledávací cesta

Velice důležitá je proměnná PATH, jíž se shellu sděluje, kde se mají vyhledávat programy. Zadá-li se nějaký příkaz, třeba:

```
$ who
```

shell jej musí někde najít. Řídí se přitom hodnotou proměnné PATH, což je posloupnost adresářů oddělených dvojtečkami:

```
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/
novak/bin
```

a příkaz hledá ve všech uvedených adresářích. Pokud jej najde (řekněme v */usr/bin/who*), příkaz se spustí. V opačném případě se vypíše hlášení:

```
bash: who: command not found
```

Modifikací proměnné PATH je možné k seznamu prohledávaných cest dočasně přidat další adresáře, například */usr/sbin*:

```
$ PATH=$PATH:/usr/sbin
$ echo $PATH
/usr/local/bin:/bin:/usr/bin/usr/X11R6/bin:/home/
novak/bin:/usr/sbin
```

Má-li být tato změna trvalá, je třeba modifikovat proměnnou PATH ve startovacím souboru ~/bash_profile, viz „Nastavení shellu" na str. 31 a poté se odhlásit a znovu přihlásit.

Synonyma (alias)

Z důvodu úspory času je možno si příkazem alias definovat vhodné zkratky dlouhých příkazů. Například příkazem:

```
$ alias ll='ls -l'
```

se definuje nový příkaz ll, kterým se spustí ls -l:

```
$ ll
total 436
-rw-r--r--      1 novak    3584    Oct 11   14:59 soubor1
-rwxr-xr-x      1 novak      72     Aug 6   23:04 soubor2
...
```

Synonyma je vhodné definovat v souboru ~/.bashrc (viz „Nastavení shellu" na str. 31), aby byla k dispozici ihned po přihlášení do systému. Výpis všech synonym se provede příkazem alias. Pokud se synonymy nevystačíte (nemají totiž parametry ani větvení), je třeba použít další funkce shellu popsané v části „Programování v shell skriptu" na str. 142.

Přesměrování vstupů/výstupů

V shellu je možno přesměrovat standardní vstup, standardní výstup a standardní chybový výstup do a ze souborů. Jinými slovy, každý příkaz, který čte ze standardního vstupu, může ve skutečnosti číst ze souboru pomocí přesměrování operátorem <:

```
$ mujprikaz < vst_soubor
```

Podobně každý příkaz, který zapisuje na standardní výstup, může místo toho zapisovat do souboru:

```
$ mujprikaz > vyst_soubor        Vytvoř/přepiš výstupní soubor
$ mujprikaz >> vyst_soubor       Připoj k výstupnímu souboru
```

Příkaz zapisující na chybový výstup může také být přesměrován do souboru:

```
$ mujprikaz > vyst_soubor 2> chybovy_soubor
                                 Různé soubory
$ mujprikaz > vyst_soubor 2>&1   Jediný soubor
```

Roury

Pomocí shellu lze pomocí operátoru shellu roura (|) přesměrovat standardní výstup jednoho příkazu do standardního vstupu jiného příkazu. Například příkazem:

```
$ who | sort
```

se pošle výstup příkazu `who` třídicímu programu, takže se vypíše seznam připojených uživatelů v abecedním pořadí.

Kombinované příkazy

Je také možno zadat několik příkazů na jednom řádku, v takovém případě se oddělují středníkem:

```
$ prikaz1 ; prikaz2 ; prikaz3
```

Pokud chceme, aby se provádění příkazů na jednom řádku ihned ukončilo, dojde-li u některého z nich k chybě, jako oddělovač se použijí znaky &&:

```
$ prikaz1 && prikaz2 && prikaz3
```

Naopak, má-li se provádění ukončit, jakmile se jeden příkaz provede, příkazy se oddělí znaky || („nebo"):

```
$ prikaz1 || prikaz2 || prikaz3
```


Citace

Normálně se mezery používají k oddělování slov na příkazovém řádku. Chceme-li ovšem zadat slovo obsahující mezeru (např. jméno souboru), musí se všechny znaky tohoto slova uzavřít mezi jednoduché nebo dvojité uvozovky, aby je shell považoval za celek. Jednoduché uvozovky berou obsah tak, jak je, zatímco ve dvojitých uvozovkách lze k vytvoření posloupnosti znaků využívat shellové konstrukce (např. proměnné):

```
$ echo 'Proměnná HOME má hodnotu $HOME'
Proměnná HOME má hodnotu $HOME'
$ echo "Proměnná HOME má hodnotu $HOME"
Proměnná HOME má hodnotu /home/novak
```

Obrácené uvozovky způsobí, že se obsah toho, co je mezi nimi, považuje za příkaz; obsah se pak nahradí standardním výstupem příkazu:

```
$ /usr/bin/whoami
novak
$ echo Jmenuji se `/usr/bin/whoami`
Jmenuji se novak
```

Zrušení funkčního významu

Z funkčního znaku (např. *) se stane normální znak, když se před něj napíše obrácené lomítko „\":

```
$ echo a*          Zástupný znak, výběr jmen souborů začínajících na „a"
adam  andula  angrešt
$ echo a\*                          Normální znak
a*
$ echo "Bydlím v $HOME"             Zde dolar označuje proměnnou
Bydlím v /home/novak
$ echo "Bydlím v \$HOME"           Normální znak dolar
```

Lze také zrušit funkční význam řídicích znaků (tabulátor, nový řádek, ^D atd.) a používat je na příkazovém řádku jako normální znaky, pokud se před ně zadá ^V. To je užitečné např. pro tabulátory (^I), které by shell jinak použil k dokončení jména souboru (viz „Doplňování jmen souborů" na str. 28).

```
$ echo "Mezi zde^V^Ia zde je tabulátor"
Mezi zde                a zde je tabulátor
```

Editace příkazového řádku

Řádek, který se právě tvoří, lze editovat podobným způsobem jako v textových editorech `emacs` nebo `vi` (viz „Vytváření a editace souborů" na str. 46). Editace pomocí `emacs` se povolí zadáním příkazu (resp. umístěním do *~/.bash_profile*, aby byla platnost trvalá):

```
$ set -o emacs
```

resp. pro `vi`:

```
$ set -o vi
```

Klávesa v emacs	Klávesa ve vi (po ESC)	Význam
^P nebo šipka nahoru	k	Předchozí příkazový řádek
^N nebo šipka dolů	j	Následující příkazový řádek
^F nebo šipka vpravo	l	Vpřed o jeden znak
^B nebo šipka vlevo	h	Vzad o jeden znak
^A	0	Začátek řádku
^E	$	Konec řádku
^D	x	Vynech následující znak
^U	^U	Smaž celý řádek

Historie příkazů

Dříve provedené příkazy (proto *historie*) lze znovu zobrazit, změnit a znovu je provést. Některé užitečné příkazy pro opakování příkazů jsou uvedeny v následující tabulce:

Příkaz	Význam
`history`	Výpis historie
`history` *N*	Výpis posledních *N* příkazů z historie
`history -c`	Vymazání historie
`!!`	Předchozí příkaz
`!`*N*	Příkaz č. *N* z historie
`!-`*N*	Příkaz č. *N* od konce

`!$`	Poslední parametr z předchozího příkazu; tato možnost je vhodná např. pro ověření existence souborů před jejich vymazáním:
	`$ ls a*`
	`$ rm !$`

`!*`	Všechny parametry z předchozího příkazu

Doplňování jmen souborů

Stiskne-li se klávesa TAB uprostřed zadávání jména souboru, jméno souboru se automaticky doplní. Existuje-li více jmen souborů se shodným začátkem, ozve se pípnutí. Stiskne-li se TAB ihned znovu, vypíší se alternativy:

```
$ cd /usr/bin
$ ls un<TAB><TAB>
```

Řízení dávek

jobs	Výpis všech dávek
&	Spuštění dávky v pozadí
^Z	Pozastavení běžné dávky (v popředí)
suspend	Pozastavení shellu
fg	Pokračuj v pozastavené dávce v popředí
bg	Pozastav dávku spuštěnou v pozadí

Ve všech linuxových shellech existuje možnost *řízení dávek*: schopnost spouštět programy v pozadí (současné spouštění několika procesů „za scénou") a v popředí (spuštění aktivního procesu v shellu). Dávkou se rozumí jakákoli posloupnost shellových programů. I příkazy zadávané interaktivně zpracovává shell jako dávku. Řízení dávek je procesem na vyšší úrovni než linuxové procesy samotné; Linux o řízení dávek nic neví, jde pouze o vlastnost shellu. Uveďme si několik důležitých pojmů z oblasti řízení dávek:

dávka v popředí

Spouští se v shellu po výpisu shellového promptu. Do ukončení dávky nelze zadat další příkaz.

dávka v pozadí

Spouští se v shellu, avšak s výpisem promptu se nečeká na její dokončení. Je možno ihned zadat další příkaz (v témže shellu).

pozastavení

Dočasné zastavení dávky v popředí

pokračování

Opětovné spuštění pozastavené dávky

jobs

Příkazem `jobs` se vypíše seznam spuštěných dávek v daném shellu:

```
$ jobs
[1]- Running                 emacs mujsoubor &
[2]+ Stopped                 su
```

Celé číslo vlevo je číslo dávky, znaménko plus značí implicitní dávku vypsanou příkazy `fg` (popředí) a `bg` (pozadí).

&

Umísťuje se na konec příkazového řádku a znamená spuštění příkazu na pozadí.

```
$ emacs mujsoubor &
[2] 28090
```

Shell vypíše číslo dávky (2) a ID procesu (28090).

^Z

Tímto povelem se zastaví běžící dávka v popředí, stav dávky se zapamatuje.

```
$ mujbiprogram
^Z
[1]+ Stopped                 mujbiprogram
$
```

Nyní lze příkazem `bg` spustit program v pozadí nebo příkazem `fg` pokračovat v popředí.

suspend

Příkazem suspend se pozastaví běžící shell, pokud je to možné, jako kdyby se zadal povel `^Z` přímo shellu. Jestliže se například spustil příkaz `su`, lze se tímto příkazem vrátit do původního shellu.

```
$ whoami
novak
$ su -l
Password: ********
# whoami
root
# suspend
[1]+ Stopped
$ whoami                          su
novak
```

bg [%číslo_dávky]

Příkazem `bg` se pozastavená dávka spustí v pozadí. Bez parametru se příkaz vztahuje k poslední pozastavené dávce. Chceme-li pozastavit některou určitou dávku, v parametru se uvede znak % a její číslo:

```
$ bg %2
```

Některé dávky nemohou být spuštěny na pozadí, například ty, které čekají na vstup. Pokud k tomu přece jen dojde, dávka se pozastaví a vypíše se hlášení:

```
[2]+ Stopped    Příkazový řádek je zde
```

Nyní lze příkazem `fg` pozastavenou dávku znovu spustit a pokračovat.

fg [%číslo_dávky]

Příkazem `fg` se pozastavená dávka nebo dávka z pozadí přenese do popředí. Bez parametru se příkaz vztahuje k poslední dávce pozastavené nebo spuštěné v pozadí. Chceme-li přenést do popředí některou určitou dávku, v parametru se uvede znak % a její číslo:

```
$ fg %2
```

Zrušení prováděného příkazu

Zadá-li se příkaz z shellu běžícího v popředí a chceme-li jej ihned zrušit, zadá se povel `^C`. Shell tento povel chápe jako „ihned ukonči příkaz v popředí". Pokud se například vypisuje dlouhý soubor (např. příkazem `cat`), zastaví se povelem `^C`:

```
$ cat velky_soubor
Toto je veliký soubor s mnoha řádky. Blablabla
blablabla blablabla blablabla ^C
$
```


Program běžící v pozadí se pozastaví tak, že se nejdříve přenese do popředí příkazem `fg` a zadá se povel `^C`, anebo se použije příkaz `kill` (viz „Řízení procesů" na str. 95).

Obecně vzato, povel `^C` není příliš vhodným způsobem, jak ukončit program. Může-li se program ukončit sám, měl by to udělat. Povelem `^C` se totiž program ukončí okamžitě, aniž by měl možnost po sobě „zamést stopy", tedy uvést sebe a svoje okolí do určitého (např. původního) stavu. Okamžitým ukončením programu v popředí se může shell dostat do nedefinovaného stavu, může se stát, že neodpovídá a nereaguje na stisk kláves. Pokud se to stane:

1. Zadejte `^]`, vypíše se shellový prompt. Tento povel by měl zafungovat, i když shell nereaguje na enter.
2. Na klávesnici zadejte slovo `reset` (i když se znaky neobjevují na monitoru) a znovu zadejte `^]`. Tím by se měl shell uvést do normálního stavu.

Povel `^C` se provede pouze v shellu, v jiném než shellovém okně nemá žádný význam. Navíc, některé programy tento povel úmyslně ignorují, např. textový editor `emacs`.

Ukončení shellu

Shell se ukončí příkazem exit nebo povelem `^D`.*

```
$ exit
```

Nastavení shellu

Funkce a chování všech shellů se nastavuje v souboru *.bash_profile* a *.bashrc* v domovském adresáři. Tento soubor se provádí pokaždé, když se uživatel přihlásí *(~/.bash_profile)*, anebo když se spustí shell *(~/.bashrc)*. Může nastavovat proměnné a synonyma, spouštět programy, tisknout horoskop nebo cokoli se uživateli zlíbí.

Tyto dva soubory jsou příklady shellových skriptů: spustitelné programy obsahující příkazy shellu. O těchto atributech se podrobněji zmíníme v „Programování v shell skriptu" na str. 142.

Instalace softwaru

Občas je nutné přidat do Linuxu další software. Ve Fedoře i v ostatních distribucích Linuxu se to obvykle dělá tímto způsobem:

**.rpm soubory*

Soubory RPM (Red Hat Package Manager). Instalují se buď ručně programem `rpm` nebo automaticky pomocí `up2date`.

* Control-D znamená „konec souboru" pro všechny programy, které čtou data ze standardního vstupu. V tomto případě je tím programem shell a po přečtení tohoto znaku se ukončí.

**.tar.gz soubory, *.tar.Z soubory, a *.tar.bz2 soubory*

Komprimované soubory tar. Jsou vytvořeny programem tar a komprimovány buď `gzip` (*.gz*), `compress` (*.Z*), anebo `bzip2` (*.bz2*).

Nový software musí obvykle instalovat superuživatel, před instalací je tedy třeba zadat příkaz su (anebo ekvivalentní). Například:

```
$ su -l
Password: ********
# rpm -ivh muj_balik.rpm
...atd. ...
```

Nový software naleznete nejčastěji na linuxových cédéčkách nebo na www stránkách jako např.:

http://freshmeat.net/
http://freshrpms.net/
http://rpmfind.net/
http://sourceforge.net/

up2date [volby][balíky] up2date

/usr/bin **stdin stdout -file –opt –help –version**

`up2date` je nejjednodušší způsob, jak udržet systém Fedora ... nuže, *up-to-date*. Superuživatel zadá pouze:

```
# up2date
```

a odpovídá na prompty. K dispozici je grafické rozhraní, avšak program lze spustit i v řádkovém režimu:

```
# up2date -l
```

čímž se vypíší všechny balíky RPM (pokud existují), které jsou v systému. Balíky se stáhnou na počítač příkazem:

```
# up2date -d balíky
```

a právě stažené balíky se nainstalují příkazem:

```
# up2date -i balíky
```

`up2date` stahuje RPM balíky ze serverů Red Hat resp. z Fedory po Internetu, takže se může stát, že při prvním spuštění se budete muset zaregistrovat.

Některé Linuxy dávají přednost jiným programům než up2date, jako např. `yum` (*http://linux.duke.edu/projects/yum*) a `apt` (*http://ayo.freshrpms.net*).

rpm [volby][soubory] rpm

/bin **stdin stdout -file –opt –help –version**

Dáváte-li přednost ruční instalaci RPM balíků, je nutno použít program `rpm`, který ostatně používá i nástroj na automatickou instalaci `up2dat. rpm` však

nejen instaluje nový software, nýbrž i ověřuje, zda jsou splněny podmínky k jeho činnosti. Potřebuje-li například balík *superzboží* ke své činnosti jiný balík s názvem *dalšízboží*, který není nainstalován, nenainstaluje se ani *superzboží*. Software se nainstaluje až poté, co systém úspěšně projde kontrolou `rpm`.

Soubory RPM mají obvykle tvar *jmeno-verze.architektura.rpm*. Například jméno balíku *emacs-20.7-17.i386.rpm* říká, že jde o program `emacs`, verze 20.7-17, počítač osazený procesorem i386 (Intel 80386 a vyšší). Je třeba počítat s tím, že `rpm` někdy požaduje celé jméno souboru (jako *emacs-20.7-17.i386.rpm*), někdy jen jméno balíku (*emacs*).

Pro manipulaci s balíky RPM se obvykle používají příkazy:

`rpm -q jméno_balíku`

Zjistí se, zda soubor *jméno_balíku* je v systému nainstalován a v jaké verzi. Například: `rpm -q textutils`. Pokud neznáte jméno balíku, (problém, zda bylo dřív vejce nebo slepice), vypište si jména všech balíků a `grepem` vyhledejte podobná jména:

```
$ rpm -qa | grep -i podobná_jména
```

`rpm -ql jméno_balíku`

Výpis souborů, které obsahuje tento nainstalovaný balík. Zkuste i `rpm -ql emacs`

`rpm -qi jméno_balíku`

Obecná informace o balíku

`rpm -qlp jméno_balíku`

Výpis obsahu souboru RPM, který nemusí ještě být nainstalován

`rpm -qa`

Výpis všech nainstalovaných balíků RPM. Užitečné jako roura pro `grep` na určení jména balíku:

```
rpm -qa | grep -i emacs
```

`rpm -gf jmén_ souboru`

Výpis balíku, ze kterého se nainstaloval daný soubor do systému

```
$ rpm -qf /usr/bin/who
sh-utils-2.0-11
```

`rpm -ivh balík1.rpm balík2.rpm ...`

Instalace balíků, které zatím v systému nejsou

`rpm- Fvh balík1.rpm balík2.rpm ...`

Aktualizace balíků, které už jsou v systému

`rpm -e jména balíků`

Vymazání balíků ze systému. V tomto případě se nezadávají čísla verzí, jen jména balíků. Např. když se instaluje balík GNU Emacs *emacs-20.7-17.i386.rpm*, měl by se nejdříve odinstalovat pomocí `rpm -e emacs`, nikoli `rpm -e emacs-20.7-17.rpm`.

Soubory tar.gz a tar.bz2

Sbalené soubory, jejichž jméno končí *tar.gz* a *tar.bz2*, obvykle obsahují zdrojový kód, který se před instalací musí zkompilovat.

1. Vypíše se obsah balíku, po jednom souboru na řádek. Zkontrolujte, aby vyextrahované soubory nepřepsaly některé důležité soubory v systému, ať už náhodou či ve zlém úmyslu:

```
$ tar tvzf balík.tar.gz | less   pro soubory gzip
$ tar tvjf balík.tar.bz2 | less  pro soubory bzip2
```

2. Pokud nikoli, vyextrahujte soubory do nového adresáře:

```
$ mkdir nový_adresář
$ cd nový_adresář
$ tar xvzf balík.tar.gz          pro soubory gzip
$ tar xvjf balík.tar.bz2         pro soubory bzip2
```

3. Vyhledejte extrahovaný soubor jménem *INSTALL* nebo *README*. V nich zjistíte, jak se software nainstaluje:

```
$ cd nový_adresář
$ less INSTALL
```

4. Obvykle je v těchto souborech uvedeno, že se má spustit skript jménem `configure` v běžném adresáři, pak se spustí `run make` a nakonec `run make install`. Ověřte si volby, které lze zadat v příkazu `configure script`:

```
$ ./configure --help
```

Pak nainstalujte software:

```
$ ./configure --help
$ make
$ su -l
Password: ********
# make install
```

Základní operace se soubory

ls	Výpis jmen souborů v daném adresáři
cp	Kopírování souboru
mv	Přejmenování souboru („move")
rm	Zrušení souboru
ln	Vytvoření ukazatelů na soubory (alternativní jména)

Jednou z prvních věcí, které je nutno se naučit v Linuxu, je manipulace se soubory: kopírování, přejmenování, zrušení atd.

ls [volby][soubory]

coreutils

/bin stdin stdout -file **-opt** **-help** **-version**

Příkazem `ls` se zobrazí atributy souborů a adresářů. Lze vypisovat jména souborů z běžného adresáře:

```
$ ls
```

ze zvolených adresářů:

```
$ ls adr1 adr2 adr3
```

nebo zvolených souborů:

```
$ ls soubor1 soubor2 soubor3
```

Nejdůležitějšími volbami jsou `-a` a `-l`. Implicitně se příkazem `ls` nezobrazují soubory začínající tečkou; volbou `-a` se zobrazí i ony. Pomocí volby `-l` získáme úplný výpis:

```
-rw-r--r--  1 novak uživatelé  149 Oct 28 2002     moje.data
```

s tímto významem (zleva doprava): přístupová práva k souboru (-rw-r--r--), vlastník (novak) skupina (uživatelé), velikost (149 bajtů), poslední modifikace (28. říjen 2002) a jméno. Další podrobnosti jsou uvedeny v části „Ochrana souborů" na str. 21.

Užitečné volby

`-a`	Výpis všech souborů včetně těch, které začínají tečkou
`-l`	Dlouhý výpis vč. atributů souboru. Volbou `-h` („human-readable") se velikost vypisuje v kilobajtech, megabajtech a gigabajtech místo v bajtech.
`-F`	Výpis včetně značek s indikací typu. K adresářům se přidává „/", k spustitelným souborům „*", k symbolickým odkazům „@", k pojmenovaným rourám „|" a k socketům „=". Jsou to ovšem jen optické pomůcky a nejsou součástí jména souboru.
`-i`	Předřadit inode před jména souborů
`-s`	Předřadit velikost souboru v blocích, což je užitečné např. pro třídění souborů podle velikosti: `$ ls -s | sort -n`
`-R`	Výpis adresáře odzadu
`-d`	Výpis samotného adresáře bez obsahu

cp [volby] soubory (soubor\adresář) coreutils

/bin stdin stdout -file -opt -help -version

Příkazem cp se kopíruje soubor:

```
$ cp soubor soubor2
```

anebo se kopíruje více souborů do adresáře

```
$ cp soubor1 soubor2 soubor3 soubor4 adresář
```

Pomocí volby `-a` nebo `-R` lze adresáře kopírovat rekurzívně.

Užitečné volby

`-p` Kopíruje nejen obsah souborů, ale i přístupová práva, časové razítko, a má-li uživatel k tomu povolení, i vlastník a skupina

`-a` Rekurzívní kopírování adresáře. Zachovávají se speciální soubory, přístupová práva, symbolické odkazy a tvrdé odkazy. Tento parametr je kombinací parametrů `-R` (rekurzívní kopírování včetně speciálních souborů), `-p` (přístupová práva) a `-d` (odkazy).

`-i` Interaktivní režim. Zeptá se, zda smí přepsat cílový soubor.

`-f` Pokud cílový soubor existuje, nepodmíněně se přepíše.

mv [volby] zdroj cíl coreutils

/bin stdin stdout -file -opt -help -version

Pomocí příkazu mv se přejmenovává soubor:

```
$ mv soubor1 soubor2
```

anebo se přesouvají soubory a adresáře do jiného (tzv. cílového) adresáře:

```
$ mv soubor1 soubor2 adresář3 adresář4 cílový_adresář
```

Užitečné volby

`-i` Interaktivní režim. Zeptá se, zda smí přepsat cílový soubor

`-f` Pokud cílový soubor existuje, nepodmíněně se přepíše.

rm [volby] soubory | adresáře coreutils

/bin stdin stdout -file -opt -help -version

Příkazem `rm` se ruší soubory

```
$ rm soubor1 soubor2 soubor3
```

anebo se rekurzívně ruší adresáře:

```
$ rm -r adresář1 adresář2
```

Užitečné volby

`-i` Interaktivní režim. Zeptá se, zda smí přepsat cílový soubor .

`-f` Nepodmíněné zrušení souboru. Všechna chybová hlášení a varování se ignorují.

`-r` Rekurzívní odstranění adresáře i s obsahem. Musí se používat opatrně, zejména s volbou `-f`.

ln [volby] zdroj cíl coreutils

/bin stdin stdout -file -opt -help -version

Příkazem `ln` („link“) se vytvoří odkaz na jiný soubor. Existují dva druhy odkazů. Symbolický odkaz se odkazuje na jiný soubor pomocí cesty, podobně jako „shortcut“ ve Windows anebo „alias“ v Macintoshi.

```
$ ln -s mujsoubor měkký_odkaz
```

Smaže-li se původní soubor, odkaz se stane neplatným (zůstane „viset ve vzduchu“), neboť bude ukazovat na neexistující cestu. Naopak *tvrdý odkaz* je pouze jiným jménem fyzického souboru na disku (technicky řečeno ukazuje na stejný inode) a smazáním původního souboru se tento link nenaruší.

```
$ ln mujsoubor tvrdý_odkaz
```

Symbolické odkazy mohou ukazovat do jiných oblastí, neboť obsahují jen cestu; tvrdé odkazy nikoli, protože inod na jednom disku nemá na jiném disku smysl. Symbolické odkazy se také mohou odkazovat na adresáře, což tvrdé odkazy nemohou … pokud ovšem nejste superuživatel s možností použití volby `-d`.

Užitečné volby

`-s` Vytvoření symbolického odkazu. Implicitní je tvrdý odkaz.

`-i` Interaktivní režim. Zeptá se, zda smí přepsat cílový soubor.

`-f` Nepodmíněné vytvoření odkazu. Pokud cílový soubor existuje, nepodmíněně se přepíše.

`-d` Vytvoření tvrdého odkazu na adresář. Tuto volbu smí použít jen superuživatel.

Kam ukazuje symbolický odkaz, můžeme zjistit například pomocí těchto příkazů:

```
$ readlink jméno_odkazu
$ ls -l jméno_odkazu
```

Operace s adresáři

cd	Změň běžný adresář
pwd	Vypiš jméno běžného adresáře, tj. kde se uživatel momentálně nachází
basename	Vypiš koncovou část cesty
dirname	Odstraň koncovou část cesty
mkdir	Vytvoř adresář
rmdir	Zruš prázdný adresář
rm -m	Zruš neprázdný adresář i s jeho obsahem

Struktura adresáře v Linuxu je popsaná v „Systému souborů“ na str. 15. Nyní se zmíníme o příkazech, jimiž se vytvářejí, modifikují a ruší adresáře, anebo se s nimi manipuluje.

cd [adresář]

bash

součást shellu stdin stdout -file -opt -help -version

Příkazem `cd` („change directory“) se nastaví běžný pracovní adresář. Příkazem bez parametru se nastaví domovský adresář.

pwd

bash

součást shellu stdin **stdout** -file -opt -help -version

Tímto příkazem se vypíše absolutní cesta běžného pracovního adresáře.

```
$ pwd
/users/novak/mujadresar
```

basename *cesta*

coreutils

/bin stdin **stdout** -file -opt **-help** **-version**

Příkazem `basename` se vypíše poslední komponenta cesty; například:

```
$ basename /users/novak/mujadresar
mujadresar
```

dirname cesta

coreutils

/usr/bin stdin **stdout** -file -opt **-help** **-version**

Příkazem `dirname` se odstraní poslední komponenta cesty; například:

```
$ dirname /users/novak/mujadresar
/users/novak
```

Tímto příkazem se nezmění běžný pracovní adresář.

mkdir [*volby*] *adresáře*

coreutils

/bin stdin stdout -file **-opt** **-help** **-version**

Příkazem `mkdir` se vytvoří jeden nebo více adresářů.

```
$ mkdir d1 d2 d3
```

Užitečné volby

`-p` Pokud se zadá cesta (nikoli tedy jen jméno adresáře), automaticky se vytvoří i uvedené rodičovské adresáře: příkazem `mkdir -p /raz/dva/tri` se vytvoří nejdříve adresáře */raz* a */raz/dva*, a pak teprve adresář */raz/dva/tri*.

`-m režim` Vytvoř adresář s uvedenými přístupovými právy:

```
$ mkdir 0755 mujadresar
```

Implicitně se práva nastaví podle shellové proměnné `umask`. Viz příkaz `chmod` v „Atributech souborů“ na str. 50 a „Ochraně souborů“ na str. 21.

rmdir [*volby*] *adresáře*

coreutils

/bin stdin stdout -file **-opt** **-help** **-version**

Příkazem `rmdir` se odstraní jeden nebo několik prázdných adresářů. Neprázdný adresář včetně jeho obsahu se odstraní (musí se opatrně!) pomocí `rm -r adresář`. Interaktivně se mohou adresáře mazat pomocí `rm -ri adresář`, nebo lze mazat s ignorováním všech chybových hlášení a bez potvrzování pomocí `rm -rf adresář`.

Užitečné volby

`-p` Pokud se zadá cesta (nikoli tedy jen jméno adresáře), automaticky se vymažou i uvedené rodičovské adresáře, které by jinak musely být prázdné. Příkazem `rmdir -p /raz/dva/tri` se vymaže nejen adresář */raz/dva/tri*, ale i adresáře */raz* a */raz/dva*.

Zobrazování souborů

cat	Zobraz celý soubor současně
less	Zobrazuj soubor po stránkách
head	Zobraz první řádek souboru
tail	Zobraz poslední řádek souboru
nl	Při zobrazování čísluj řádky
od	Zobraz obsah v oktalovém (nebo jiném) tvaru
xxd	Zobraz obsah hexadecimálně
gv	Zobraz postscriptový soubor nebo soubor PDF
xdvi	Zobraz soubor TeX

cat [*volby*] [*soubory*]

coreutils

/bin **stdin** **stdout** **-file** **-opt** **-help** **-version**

Nejjednodušším prohlížečem je příkaz `cat`, kterým se soubory jeden za druhým pouze kopírují na standardní výstup. Velké soubory budou bez zastavování rolovat po monitoru a pokud bychom je chtěli prohlížet podorobněji, musíme použít příkaz `less`. Příkaz `cat` je spíše vhodný pro odeslání souboru do roury. `cat` může taktéž výstup poněkud upravit, umí zobrazit netisknutelné znaky, číslovat řádky (k tomuto účelu je ovšem vhodnější `nl`) a odstraňovat mezery.

more | pg binární od

Užitečné volby

`-T`	Tabulátor vypisuj jako `^I`.
`-E`	Nový řádek vypisuj jako `$`.
`-v`	Ostatní netisknutelné znaky vypisuj v čitelné podobě.
`-n`	Čísluj řádky.
`-b`	Čísluj jen neprázdné řádky.
`-s`	Posloupnost prázdných řádků nahraď jediným prázdným řádkem.

less [*volby*] [*soubory*]

less

/usr/bin stdin stdout* -file –opt –help –version

Příkaz `less` se používá k prohlížení textu po stránkách resp. po obrazovkách. Hodí se zejména k prohlížení textových souborů nebo jako poslední příkaz v rouře s dlouhým výstupem.

```
$ příkaz1 | příkaz2 | příkaz3 | příkaz4 | less
```

Je-li příkaz `less` v činnosti, lze si pomocí klávesy `h` vypsat kompletní nápovědu k tomuto příkazu. Některé užitečné klávesy k ovládání výpisu jsou uvedeny zde:

Klávesa	Význam
`h`, `H`	Nápověda
mezera, `f`, `^V`, `^F`	Dopředu o jednu stránku
enter	Dopředu o jeden řádek
`b`, `^B`, `ESC -b`	Dozadu o jednu stránku
`/`	Vyhledávací režim. Za lomítko se zadá regulární výraz a stiskne se enter. `less` pak hledá první řádek se shodným obsahem.
`?`	Totéž co `/`, ale směrem nazpět.
`n`	Hledej dál (podle posledně nastaveného parametru)
`N`	Hledej směrem zpět (podle posledně nastaveného parametru)
`V`	Edituj stávající soubor implicitním editorem (hodnota vnější proměnné `VISUAL`, anebo, není-li definovaná, `EDITOR`, anebo, není-li definovaná, `vi`).

Klávesa	Význam
`<`	Přechod na začátek souboru.
`>`	Přechod na konec souboru
`:n`	Přechod na další soubor
`:p`	Přechod na předchozí soubor

* I když příkaz `less` lze umístit do roury, anebo jej lze přesměrovat do souboru, mnoho rozumných důvodů k tomu neexistuje.

`less` má nepřeberné množství možností, uvádíme jen některé. Je ovšem dobré si je alespoň projít v manuálu.

Užitečné volby

`-c`	Smazání monitoru před zobrazením další stránky
`-m`	Vypisuje se „upovídaný“ prompt, který udává, jaká část se vypsala (v procentech).
`-N`	Číslování vypisovaných řádků
`-r`	Řídicí znaky se nenahrazují viditelnými znaky
`-s`	Posloupnost prázdných řádků nahraď jediným prázdným řádkem
`-S`	Zobrazí se jen ta část řádku, která se vejde na monitor.

head [*volby*] [*soubory*]

coreutils

/usr/bin stdin stdout -file –opt –help –version

Příkazem `head` se vypíše prvních 10 řádků souborů: vhodné pro zjišťování obsahu souborů.

```
$ head mujsoubor
$ head * | less Nahlédni do všech souborů v běžném adresáři
```

Užitečné volby

`-N`	Výpis prvních *N* řádků (nikoli tedy 10)
`-n N`	
`-c N`	Výpis prvních *N* bajtů souboru
`-q`	Tichý režim: zpracovává-li se více než jeden soubor, nevypisuje se záhlaví každého z nich. Normálně se vypisuje záhlaví se jménem souboru.

tail [*volby*] [*soubory*]

coreutils

/usr/bin stdin stdout -file –opt –help –version

Příkazem `tail` se vypíše 10 posledních řádků v souboru, a umí toho ještě víc.

```
$ tail mujsoubor
```

Užitečné volby

`-N`	Výpis posledních *N* řádků (nikoli tedy 10)
`-n N`	
`+N`	Výpis všech řádků kromě prvních *N*.
`-c N`	Výpis posledních *N* bajtů souboru
`-f`	Ponech soubor otevřený a kdykoli k němu přibudou řádky, vypiš je. Příkaz je mimořádně užitečný. Zadá-li se volba retry a soubor ještě neexistuje, čeká se na jeho vznik.

`-q` Tichý režim: zpracovává-li se více než jeden soubor, nevypisuje se záhlaví každého z nich. Normálně se vypisuje záhlaví se jménem souboru.

nl [*volby*] [*soubory*] coreutils

/usr/bin **stdin stdout -file -opt -help -version**

Příkazem `nl` se soubor kopíruje na standardní výstup a přidávají se čísla řádků. Je to pohodlnější způsob než složité zadávání číslování řádků v příkazu `cat` (volby `-n` a `-b`). `nl` lze používat dvojím způsobem: na obyčejné textové soubory a na speciální textové soubory s definovaným záhlavím a zápatím.

Užitečné volby

`-b [a | t | n | pR]` Očíslovat všechny řádky (`a`), neprázdné řádky (`t`), nečíslovat (`n`), nebo číslovat jen řádky obsahující regulární výraz *R*. (Implicitně=*a*)

`-v N` Číslování začít od celého čísla *N* (Implicitně=1)

`-i N` Inkrementace číslování je *N* pro každý řádek, lze tedy používat např. jen lichá (`-i2`) nebo sudá (`-v2 -i2`) čísla.

`-n [ln | rn | rz]` Zarovnat čísla řádků vlevo (`ln`), vpravo (`rn`) nebo vpravo s úvodními nulami (`rz`). (Implicitně=`ln`)

`-w N` Zadej počet míst číslování = *N* (Implicitně=6)

`-s S` Mezi číslo řádku a text vlož řetězec *S* (Implicitně=`TAB`)

Navíc, `nl` umí šílenou věc: rozdělit textový soubor na virtuální stránky, přičemž každá z nich má záhlaví, tělo a zápatí s jiným schématem číslování. K tomu je ale potřeba přímo do souboru vsunout oddělovací řetězce: `\:\:\:` (začátek záhlaví), `\:\:` (začátek těla) a `\:` (začátek zápatí). Každý z nich musí být na samostatném řádku. Pak lze tuto možnost využít (viz manuál) a tisknout soubory s kudrlinkami v záhlaví a v zápatí.

od [*volby*] [*soubory*] coreutils

/usr/bin **stdin stdout -file -opt -help -version**

Binární soubor si lze prohlížet pomocí příkazu `od` („Octal Dump"). Jeden nebo více souborů se kopíruje na standardní výstup, data se vypisují v ASCII, oktalově, desítkově, hexadecimálně nebo v pohyblivé řádové tečce v různých velikostech (bajty, krátce, dlouze), např. v příkazu:

```
$ od -w8 /usr/bin/who
0000000 042577 043114 000401 000001
0000010 000000 000000 000000 000000
0000020 000002 000003 000001 000000
0000030 106240 004004 000064 000000
...
```


se zobrazí byty v binárním souboru */usr/bin/who* oktalově, osm bajtů na řádek. Sloupec vlevo udává vzdálenost každého řádku od začátku, taktéž oktalově.

Užitečné volby

`-N B` Vypiš hexadecimálně (předřadí se Ox nebo OX) pouze prvních *B* bajtů každého souboru, zadáno dekadicky, do 512-bajtových bloků (`b`), kilobajtů (`k`) nebo megabajtů (`m`). Implicitně se vypisuje celý soubor.

`-j B` Výpis všech souborů od bytu *B*+1; formáty stejné jako u `-N` (Implicitně=0).

`-w [B]` Výpis *B* bajtů na řádek; formáty stejné jako u `-N`. `-w` bez parametru je stejné jako `w32` (Implicitně=16).

`-s [B]` Seskupit každý řádek do posloupnosti o *B* bajtech oddělených mezerou. Formáty stejné jako u `-N`. `-s` bez parametru je rovno `-s3` (Implicitně=2)

`-A (d | o | x | n)` Vzdálenosti od počátku v levém sloupci vypiš dekadicky (`d`), oktalově (`o`) nebo hexadecimálně (`h`), anebo vůbec. (Implicitně=`o`)

`-t (a | c) [z]` Výpis ve znakovém tvaru, nealfanumerické znaky se vypisují jako posloupnost uvozená lomítkem (`a`), nebo jménem znaku (`c`). z viz níže

`-t(d | o | u | x) [SIZE[z]]` Výpis v celočíselném tvaru, a to oktalově (`o`), dekadicky se znaménkem (`d`), dekadicky bez znaménka (`u`), hexadecimálně (`x`). (Binární výstup se zadá jako `xxd`). *SIZE* značí počet bajtů na celé číslo; může to být kladné celé číslo nebo jeden ze znaků C, S, I a L zadávající datový typ znak, krátké, celé nebo dlouhé. `z` viz níže

`-t f[SIZE[z]]` Výpis v pohyblivé řádové tečce. *SIZE* je počet znaků na celé číslo. Může to být kladné celé číslo nebo jeden ze znaků F, D nebo L udávající datový typ pohyblivá řádová tečka, dvojitý integer a dlouhý integer. `z` viz níže. Vynechá-li se parametr `-t`, implictně se nastaví na `2`. Připojí-li se k parametru `-t` písm. `z`, vypisuje se vpravo ještě jeden sloupec s tisknutelnými znaky na každém řádku, podobně jako u implicitního výstupu `xxd`.

xxd [*volby*] [*soubory*] vim-common

/usr/bin stdin stdout -file -opt -help -version

Příkaz `xxd` je podobný příkazu `od`. Pořizuje se jím hexadecimální nebo binární výpis souboru v různých tvarech. Umí i opačný postup: převod hexadecimálního výstupu na původní data Například příkazem:

```
$ xxd /usr/bin/who
0000000:7f45 4c46 0101 0100 0000 0000 0000 0000 .ELF............
0000010:0200 0300 0100 0000 a08c 0408 3400 0000 ............4...
0000020:6824 0000 0000 0000 3400 2000 0600 2800 h$......4. ...(.
0000030:1900 1800 0600 0000 3400 0000 3480 0408 ........4...4...
...
```

se pořídí hexadecimální výpis binárního souboru */usr/ bin/who*, 16 bajtů na řádek. Levý sloupec udává vzdálenost řádku od počátku souboru, osm dalších sloupců obsahuje data a v posledním sloupci jsou zobrazeny tisknutelné znaky na daném řádku, pokud existují.

Příkazem `xxd` se implicitně pořizuje výpis ve třech sloupcích: vzdálenost od počátku souboru, data v hexadecimálním tvaru a data v textovém tvaru (pouze tisknutelné znaky).

Užitečné volby

`-l` *N*	Výpis prvních *N* bajtů (Implicitně celý soubor)
`-s` *N* `-s -`*N*	Začátek výpisu od jiného než prvního bajtu. Přeskočí se prvních *N* bajtů (první případ), anebo se začne *N* bajtů od konce (druhý případ, pro *N* záporné). Existuje i možnost `+`*N*, viz manuál.
`-c` *N*	Výpis *N* bajtů na řádek (Implictně=16)
`-g` *N*	Bajty na řádku se zobrazují po skupinách o *N* bajtech oddělených mezerami, podobně jako u od `-s`. (Implicitně=2)
`-b`	Binární výstup místo znakového.
`-u`	Zobrazují se horní znaky
`-p`	Prostý hexadecimální výpis, 60 bajtů na řádek.
`-i`	Výpis ve tvaru datové struktury jazyka C. Při čtení ze souboru se vypisují znaky bez znaménka a celá čísla bez znaménka udávající délku pole. Při čtení ze standardního vstupu se vypisují pouze hexadecimální obsahy bajtů oddělené čárkami.
`-r`	Reverzní operace: konverze hexadecimálního výpisu do souboru. Vstupem je implicitní hexadecimální tvar anebo (zadá-li se parametr `-p`) tvar odpovídající tomuto parametru. Kdybyste se někdy nudili, zkuste některý z těchto výpisů a zpětnou konverzi prohnat rourou, na výstupu by se měl objevit soubor v původním tvaru:

```
$ xxd mujsoubor | xxd -r
$ xxd -p mujsoubor | xxd -r -p
```


gv [*volby*] *soubor* gv

/usr/X11R6/bin stdin stdout -file -opt -help -version

Příkazem `gv` („GhostView") se vypisuje soubor PDF nebo Adobe PostScript v X window. Spustit se může buď jako `gv` nebo jako `ghostview`. Činnost je jednoduchá: klepnutím na požadované číslo strany se přejde na tuto stranu atd. Stačí si s příkazem trochu pohrát.

GhostView je čistě linuxový postscriptový prohlížeč, avšak některé jiné volně dostupné prohlížeče PDF obsahují acroread (*http://www.adobe.com/*) a xpdf (*http://www.foolabs.com/xpdf/*).

Užitečné volby

`-page` *P*	Začni na straně *P*. (Implicitně=1)

`-monochrome`
`-grayscale`
`-color`
	Režim monitoru – výběr z uvedených možností.

`-portrait`
`-landscape`
`-seascape`
`-upsidedown`
	Orientace stránky. Bez parametru se nastaví automaticky.

`-scale N`	Nastavení měřítka (zvětšení) na monitoru. *N* může být kladné (zvětšení) nebo záporné (zmenšení).
`-watch` `-nowatch`	Automatické načtení (nebo nenačtení) postscriptového souboru, když se změní jeho obsah.

xdvi [*volby*] *soubor* tetex-xdvi

/usr/bin stdin stdout -file -opt -help -version

Výstupem systému na zpracování dokumentů TeX jsou binární soubory ve formátu nazývaném DVI, příponou je *.dvi*. Prohlížeč xdvi vypisuje soubor DVI v prostředí X window. Jinou možností je pomocí příkazu `dvips` převést soubor DVI do postscriptu a výpis provést příkazem ghostview (`gv`):

```
$ dvips -o mujsoubor.ps mujsoubor.dvi
$ gv mujsoubor.ps
```

Při výpisu souboru má `xdvi` na pravé straně lištu s tlačítky s obvyklými funkcemi jako Next pro přechod na další stranu apod. (Lištu lze skrýt, spustí-li se

xdvi s parametrem -expert.). V souboru se můžeme pohybovat také pomocí kláves:

Klávesa	Význam
q	Konec.
n, mezera, enter, pagedown	Jdi na další stranu. S parametrem *N* přeskok *N* stran.
p, backspace, delete, pageup	Jdi na předchozí stranu. S parametrem *N* přeskok *N* stran zpět.
<	Přechod na první stranu.
>	Přechod na poslední stranu.
^L	Stranu vypiš znovu.
R	Znovu přečti soubor DVI a oznam příp. změnu v souboru.
Tlačítko myši	Zvětši obdélník pod kurzorem myši.

xdvi má celou řadu příkazů, kterými si uživatel může nastavit barvy, geometrii, zvětšení a celkové chování systému.

Vytváření a editace souborů

emacs	Textový editor Free Software Foundation
vim	Textový editor, rozšíření unixového vi
umask	Nastavení implicitního režimu pro nové soubory a adresáře
soffice	Sada pro editaci dokumentů Microsoft Office (Word, Excel, PowerPoint)
abiword	Editace dokumentů Microsoft Word
gnumeric	Tabulky Excelu

Pokročilejší užívání Linuxu vyžaduje zvládnutí některého z textových editorů. Nejužívanějšími jsou emacs od Free Software Foundation a vim, který je odvozen od unixového vi. Úplný popis těchto editorů přesahuje možnosti této knihy, avšak k oběma existuje on-line popis a nejčastěji užívané příkazy jsou uvedeny v tabulce 1. Editace souboru se spouští příkazy:

```
$ emacs myfile
$ vim myfile
```

Pokud *myfile* neexistuje, automaticky se vytvoří. Taktéž lze snadno vytvořit prázdný soubor (pro pozdější editaci) buď pomocí příkazu touch (viz „Atributy souborů“ na str. 50):

```
$ touch newfile
```

anebo zapsáním dat do nového souboru přesměrováním výstupu programu (viz „Přesměrování vstupů/výstupů“ na str. 26):

```
$ echo anything at all > newfile
```

V případě sdílení souborů s MS Windows jsou k dispozici linuxové programy pro editaci souborů MS Word, Excel a PowerPoint.

Implicitní editor

Linuxové programy si spouští editor samy, implicitním editorem je vim. Například program pro zasílání zpráv (e-mail) si může spustit editor pro vytvoření nové zprávy, lze jej také spustit zadáním písmene „v“. Pokud ovšem vim nemá být implicitním editorem, stačí nastavit globální proměnné VISUAL a EDITOR dle vlastního výběru, např.:

```
$ EDITOR=emacs
$ VISUAL=emacs
$ export EDITOR VISUAL
```

Volitelné

Je nutno nastavit obě proměnné, neboť různé programy pracují s různými proměnnými. Permanentně lze nastavit proměnné EDITOR a VISUAL ve starovacím souboru *~/.bash_profile*. Implicitní editor může používat každý program, jehož parametrem může být jméno souboru.

Bez ohledu na nastavení těchto proměnných by správci systémů měli znát alespoň příkazy editorů vim a emacs pro případ, když je systém aplikuje na kritický soubor.

emacs [*volby*] [*soubory*] **emacs**

/usr/bin **stdin stdout -file -opt -help -version**

emacs je mimořádně mocný nástroj s množstvím příkazů a má v sobě zabudovaný programovací jazyk na vytváření vlastních edičních postupů. Popis emacs se spouští příkazem:

```
$ emacs
```

a zadá se ^h t.

Většina příkazů emacs se zadává pomocí klávesy control (např. ^F), nebo meta klávesami, přičemž většinou jsou to escape nebo alt. V dokumentaci emacs se meta klávesa označuje symboly M- (např. M-F znamená „současně stisknout meta klávesu a F“), tento způsob značení budeme používat i v tomto textu. Nejdůležitější funkční klávesy jsou uvedeny v tab. 1.

vim [*volby*] [*soubory*] **vim-enhanced**

/usr/bin **stdin stdout -file -opt -help -version**

vim je rozšířenou verzí původního standardního unixového editoru vi. Popis vim se spouští příkazem:

```
$ vimtutor
```

vim je editor orientovaný na práci s režimy. Pracuje ve dvou režimech, *insert* (vkládání) a *normal*. Režim insert se používá při běžném zadávání textu, zatímco normální režim pracuje s příkazy jako „vynech řádek“ nebo „kopíruj“. Základní klávesové příkazy normálního režimu jsou uvedeny v tab. 1.

Tabulka 1. Základní klávesové příkazy editorů emacs a vim

Činnost	emacs	vim
Spustit editor v běžném okně	`$ emacs -nw[soubor]`	`$ vim [soubor]`
Spustit editor v novém X window	`$ emacs [soubor]`	`$ gvim [soubor]`
Psát text	*text*	`i` *text* `ESC`
Uložit & ukončit	`^x^s`, pak `^x^c`	`:wq`
Ukončit bez uložení	`^x^c` Zadej „no", mají-li se uložit buffery	`:q!`
Uložit	`^x^s`	`:w`
Uložit jako	`^x^w`	`: jméno_souboru`
Zrušit poslední opravu	`^_`	`u`
Pozastavit editor (nikoli v X)	`^z`	`^z`
Přechod do režimu editování	(N/A)	`ESC`
Přechod do příkazového režimu	`M-x`	`:`
Přerušit prováděný příkaz	`^g`	`ESC`
Posun vpřed	`^f` nebo šipka vpravo	`l` nebo šipka vpravo
Posun zpět	`^b` nebo šipka vlevo	`h` nebo šipka vlevo
Posun nahoru	`^p` nebo šipka nahoru	`k` nebo šipka nahoru
Posun dolů	`^n` nebo šipka dolů	`k` nebo šipka dolů
Jdi na další slovo	`M-f`	`w`
Jdi na předchozí slovo	`M-b`	`b`
Jdi na začátek řádku	`^a`	`0`
Jdi na konec řádku	`^e`	`$`
Posun dolů o jednu obrazovku	`^v`	`^f`
Posun nahoru o jednu obrazovku	`M-v`	`^b`
Přechod na začátek bufferu	`M-<`	`gg`
Přechod na konec bufferu	`M->`	`G`
Vynechání následujícího znaku	`^d`	`x`
Vynechání předchozího znaku	`BACKSPACE`	`X`
Vynechání následujícího slova	`M-d`	`de`
Vynechání předchozího slova	`M-BACKSPACE`	`db`
Vynechání běžného řádku	`^a^k^k`	`dd`
Vynechání znaků až do konce řádku	`^k`	`d$`
Definice bloku (stisknutím klávesy se označí začátek bloku, pak se kurzor přesune na konec bloku)	`^mezera`	`v`
Odstranění bloku	`^w`	`d`
Kopírování bloku	`M-w`	`y`
Přidání bloku	`^y`	`p`
Nápověda	`^h`	`:help`
Přechod do uživ. manuálu	`^h i`	`:help`

umask [*volby*] [*maska*]

bash

součást shellu — stdin **stdout** -file -opt -help -version

Příkazem `umask` se nastaví nebo vytiskne implicitní režim pro vytváření souborů a adresářů: zda jsou určeny pro čtení, zápis, a/nebo proveditelné uživatelem, skupinou uživatelů nebo všemi (viz také „Ochrana souborů" na str. 21 a příkaz `chmod` v „Atributech" na str. 50)

```
$ umask
0002
$ umask -S
u=rwx, g=rwx, o=rx
```

Nejdříve technická poznámka. Hodnota `umask` je maska, tj. binární hodnota zkombinovaná (pomocí binární operace `XOR`) s 0666 v případě souborů a s 0777 v případě adresářů. Tím vznikne implicitní režim ochrany. Např. `0002 XOR 0666` dá 0664, nebo `0002 XOR 0777` dá pro adresáře režim 0775.

Pokud toto vysvětlení nestačí, zde je jednoduchý recept. Pomocí masky 0022 si nastavíte veškerá práva, zatímco všichni ostatní budou mít právo jenom na čtení/spuštění:

```
$ umask 0022
$ touch newfile && mkdir dir
$ ls -ld newfile dir
-rw-r--r--  1 novak novak     0 Nov 11 12:25 newfile
drwxr-xr-x  2 novak novak  4096 Nov 11 12:25 dir
```

Pomocí masky 0002 si lze nastavit vlastní práva a práva své skupiny, všichni ostatní budou mít právo číst/spouštět:

```
$ umask 0002
$ touch newfile && mkdir dir
$ ls -ld newfile dir
-rw-rw-r--  1 novak novak     0 Nov 11 12:26 newfile
drwxrwxr-x  2 novak novak  4096 Nov 11 12:26 dir
```

Použitím masky 0077 se nastaví vlastní práva, zatímco ostatním se nenastaví práva žádná:

```
$ umask 0077
$ touch newfile && mkdir dir
$ ls -ld newfile dir
-rw-------  1 novak novak     0 Nov 11 12:26 newfile
drwx------  2 novak novak  4096 Nov 11 12:26 dir
```

soffice [*soubory*] — openoffice.org

/usr/lib/openoffice/programs — stdin stdout -file -opt -help -version

OpenOffice.org je rozsáhlý integrovaný balík kancelářského softwaru, jímž lze editovat soubory Wordu a Excelu a PowerPointu. Žadá se pouze:

```
$ soffice
```

a můžeme pracovat. Jediným programem lze editovat všechny tři typy souborů*. Program je značně rozsáhlý a potřebuje ke své činnosti spoustu paměti a prostoru na disku.

Program OpenOffice umí zpracovávat i kresby (příkaz `sdraw`), faxy (`sfax`), adresní štítky atd. Další informace o tomto programu jsou uvedeny buď na *http://www.openoffice.org/* nebo v nápovědě k programu soffice.

abiword [*volby*] [*soubory*] — abiword

/usr/bin — stdin stdout -file -opt -help -version

`abiword` je jiný program sloužící k editaci souborů Wordu. Je menší a rychlejší než `soffice`, má méně funkcí, avšak pro běžnou práci bohatě stačí. Soubory uvedené v parametrech musí existovat, `abiword` si je nevytváří.

gnumeric [*volby*] [*soubory*] — gnumeric

/usr/bin — stdin stdout -file -opt -help -version

`gnumeric` je program pro práci s tabulkami, kterým lze editovat dokumenty Excelu. Je rychlý, má velký rozsah funkcí a práce s ním je podobná práci s Excelem. Soubory uvedené v parametrech musí existovat, `gnumeric` si je nevytváří.

Atributy souborů

stat	Výpis atributů souborů a adresářů
wc	Spočítat bajty, slova a řádky v souboru
du	Výpis zaplnění disku soubory a adresáři
file	Identifikace typu souboru
touch	Změna časové známky v souboru a v adresáři
chown	Změna vlastníka souboru a adresáře
chgrp	Změna skupiny vlastníků souboru a adresáře
chmod	Změna přístupových práv souboru a adresáře
chattr	Změna rozšířených atributů souboru a adresáře
lsattr	Výpis rozšířených atributů (EA) souboru a adresáře

* Ve skutečnosti se `soffice` skládá ze tří různých programů, a to Writer (příkaz `swriter`) na zpracování textů, Calc (příkaz `scalc`) na zpracování tabulek a Impress (příkaz `simpress`) na prezentaci. Tyto programy se mohou spouštět i samostatně.

Obsah linuxového souboru tvoří jen část uchovávaných údajů. Součástí každého souboru a adresáře jsou jeho atributy, tj. vlastník souboru, velikost, přístupová práva a další informace. Některé z nich (viz „Základní operace se soubory“ na str. 34) lze vypsat příkazem `ls -l`, další atributy souborů lze zjišťovat pomocí jiných příkazů.

stat [*volby*] *soubory* — coreutils

/usr/bin — stdin **stdout** -file **-opt -help -version**

Příkazem `stat` se vypisují důležité vlastnosti souborů (implicitně), nebo systémů souborů (volba `-f`). Informace o souborech vypadají takto:

```
$ stat mujsoubor
File: "mujsoubor"
Size: 1264        Blocks: 8      Regular file
Access (0644/-rw-r--r--)    UID: (  600/novak)  Gid:
(  620/users)
Device: 30a      Imode: 99492          Links: 1
Access: Fri Aug 29 00:16:12 2003
Modify: Wed Jul 23 23:09:41 2003
Change: Wed Jul 23 23:11:48 2003
```

a obsahují jméno souboru, velikost v bajtech (1264), velikost v blocích (8), typ souboru (Regular File), přístupová práva v oktalovém tvaru (0644), přístupová práva ve tvaru ls -l (-rw-r--r--), ID vlastníka (600), jméno vlastníka (novak), ID skupiny (620), jméno skupiny (users), typ zařízení (30a), číslo uzlu (99492), počet tvrdých odkazů (1) a časové známky posledního přístupu, poslední modifikace a poslední změny stavu. Informace o systému souborů vypadají takto:

```
$ stat -f mujsoubor
File: "mujsoubor"
ID: bffff358 ffffffff   Namelen: 255  Type: EXT2
Blocks: Total: 2016068  Free: 876122  Available:
773709 Size: 4096
Inodes: Total: 1026144  Free: 912372
```

a obsahují jméno souboru (*mujsoubor*), ID systému souborů, (bffff358, ffffffff), maximální délku jména souboru v daném systému (255 bajtů), typ systému (EXT2), počet všech, volných a dostupných bloků v systému (2016068, 876122, 773709), velikost bloku v daném systému (4096) a počet všech a volných uzlů (1026144 a 912372).

Pomocí volby `-t` získáme stejné údaje, avšak jsou všechny na jednom řádku a bez hlavičky. Tento tvar je vhodný k dalšímu zpracování (shellovým skriptem apod.):

```
$ stat -t mujsoubor
mujsoubor 1264 8 81a4 500 500 30a 99492 1 44 1e 1062130572
1059016181 1059016308
```

```
$ stat -tf mujsoubor
mujsoubor bffff358 ffffffff 255 ef53 2016068 875984 773571 4096 102
6144 912372
```

Užitečné volby

`-l`	Výpis symbolických odkazů vč. popisu souboru, na který ukazují
`-f`	Popis systému souborů obsahujícího daný soubor
`-t`	Zhuštěný režim: výpis na jediný řádek

wc [*volby*] [*soubory*] — coreutils

/usr/bin — **stdin stdout -file -opt -help -version**

Příkazem `wc` („word count“) se vypíše počet bajtů, slov a řádků v textovém souboru

```
$ wc musjoubor
      24      62     428 mujsoubor
```

Soubor má 24 řádků, 62 slov oddělených mezerami a 428 bajtů

Užitečné volby

`-l`	Vypiš pouze počet řádků
`-w`	Vypiš pouze počet slov
`-c`	Vypiš pouze počet bajtů (znaků)
`-L`	V každém souboru najdi nejdelší řádek a vypiš jeho délku

du [*volby*] [*soubory*] — coreutils

/usr/bin — **stdin stdout -file -opt -help -version**

Příkazem `du` („disk usage“) se zjistí velikost diskového prostoru obsazeného soubory a adresáři. Implicitně se zjišťuje velikost běžného adresáře a všech jeho podadresářů, vypíše se velikost každého z nich a na konci se vypíše celkový součet.

```
$ du
8       ./Poznamky
36      ./Pošta
340     ./Soubory/moje
40      ./Soubory/bobovy
416     ./Soubory
216     ./PC
2404    .
```

Lze také stanovovat velikost souborů:

```
$ du mujsoubor mujsoubor2
4       ./mujsoubor
16      ./mujsoubor2
```


Užitečné volby

`-b -k -m`	Údaje o využití disku v bajtech (`-b`), kilobajtech (`-k`) nebo megabajtech (`-m`).
`-B N`	Výpis v blocích o délce, jeden blok=*N* bajtů (Implicitně=1024)
`-h -H`	„Srozumitelný“ výpis s výběrem nejvhodnější jednotky. Jsou-li např. velikosti dvou adresářů 1 gigabajt a 25 kilobajtů, vypíše se 1G a 25K. Při volbě `-h` se používá mocnina 1024, zatímco při `-H` mocnina 1000.
`-c`	Výpis součtu na posledním řádku. Tento údaj se u adresářů vypisuje implicitně, u souborů je třeba uvést `-c`.
`-L`	Výpis velikosti souboru, na nějž směřuje symbolický odkaz
`-s`	Výpis pouze celkové velikosti

file [*volby*] *soubory* — file

/usr/bin — **stdin stdout -file -opt -help -version**

Příkaz `file` slouží k výpisu typu souboru:

```
$ file /etc/hosts/usr/bin/who dopis.doc
/etc/hosts:  ASCII text
/usr/bin/who:  ELF 32-bit LSB executable, Intel 80386 …
dopis.doc:  Microsoft Office Document
```

Na rozdíl od jiných operačních systémů Linux nerozlišuje typy souborů, takže příkaz `file` si typ souboru zjišťuje sám podle jeho obsahu a dalších znaků.

Užitečné volby

`-b`	Vynech jména souborů (levý sloupec)
`-i`	Místo obvyklého výstupu se u souborů jako „text/ plain“ nebo „audio/mpeg“ vypisuje typ MIME
`-f soubor_jmen`	Z uvedeného souboru se čtou jména souborů (po jednom na řádku, vypisuje se jejich typ) a zpracovávají se obvyklým způsobem.
`-L`	Zjišťuje se typ souboru, na nějž směřuje symbolický odkaz.
`-z`	Je-li soubor komprimovaný, (viz „Komprimace a balení souborů“ na str. 72), zjišťuje se typ souboru v rozbaleném stavu a nevypisuje se „komprimovaný soubor“.

touch [*volby*] *soubory* — coreutils

/bin — **stdin stdout -file -opt -help -version**

Příkazem `touch` se změní hodnota dvou časových známek v souboru: modifikace (kdy se naposledy změnil obsah souboru) a přístup (kdy se ze souboru naposledy četlo).

```
$ touch mujsoubor
```

Hodnotu těchto časových známek lze nastavit libovolně, např.:

```
$ touch -d "November 18 1975" mujsoubor
```

Pokud soubor neexistuje, tímto příkazem se vytvoří. Takto jednoduše tedy lze vytvořit soubor.

Užitečné volby

`-a`	Změnit pouze poslední přístup
`-m`	Změnit pouze modifikaci
`-c`	Pokud soubor neexistuje, nevytvářej jej (normálně se vytvoří)
`-d` *timestamp*	Nastav časové známky souboru. Jsou povoleny všechny možné tvary zadání, od „12/28/2001 3pm" přes „28-May" (doplní se běžný rok a půlnoc), „next tuesday 13:59" až po „0" (dnešní půlnoc). Možnosti je třeba si vyzkoušet a ověřit pomocí příkazu stat. Kompletní popis je v info touch.
`-t` *timestamp*	Méně inteligentní způsob zadávání časové známky souboru ve tvaru `[[CC]YY]MMDDhhmm[.ss]`, kde *CC* je dvoumístné století, *YY* je dvoumístný rok, *MM* je dvoumístný měsíc, *DD* je dvoumístný den, *hh* je dvoumístná hodina, *mm* je dvoumístná minuta a *ss* je dvoumístná vteřina. Například `-t 20030812150047` znamená 12. srpna 2003 v 15:00:47.

chown [*volby*] *spec_uzivatele soubory* — coreutils

/bin — stdin stdout -file **–opt –help –version**

Příkaz pro změnu vlastníka („change owner") souboru a adresáře.

```
$ chown novak mujsoubor mujsoubor2 mujadresar
```

Parametrem *spec_uzivatele* může být kterákoli z možností:

- Uživatelské jméno (anebo uživatelské ID). Nastaví se uživatel.
- Uživatelské jméno (anebo uživatelské ID) následované dvojtečkou a jménem skupiny (nebo číslem ID skupiny). Nastaví se uživatel a skupina.
- Uživatelské jméno (anebo uživatelské ID) následované dvojtečkou. Skupina se nastaví na skupinu, do níž uživatel patří.
- Jméno skupiny (anebo číslo ID skupiny) s dvojtečkou na začátku. Nastaví se pouze skupina.
- `--reference=`*soubor* nastaví se stejný uživatel a skupina jako u uvedeného souboru

Užitečné volby

`--dereference`	Činnost se vztahuje k souboru, na který ukazuje symbolický odkaz
`-R`	Rekurzívní změna vlastníka uvnitř adresářové struktury

chgrp [*volby*] *spec_skupiny soubory* — coreutils

/bin — stdin stdout -file **–opt –help –version**

Příkaz pro změnu skupiny („change group") souboru a adresáře.

```
$ chgrp novak mujsoubor mujsoubor2 mujadresar
```

Parametrem *spec_skupiny* může být kterákoli z množností:

- Jméno skupiny (anebo číslo ID skupiny)
- `--reference=`*soubor* nastaví se stejná skupina jako u uvedeného souboru

Viz „Skupiny" na str. 104.

Užitečné volby

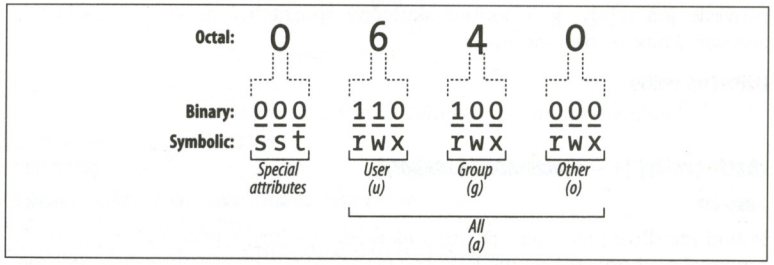
`--dereference`	Činnost se vztahuje k souboru, na který ukazuje symbolický odkaz
`-R`	Rekurzívní změna vlastníka uvnitř adresářové struktury

chmod [*volby*] *práva soubory* — coreutils

/bin — stdin stdout -file **–opt –help –version**

Příkazem `chmod` („change mode") se nastavují přístupová práva souborů a adresářů. Ne každý soubor je přístupný každému (nejsou to totiž Windows 95) a chmod je nástrojem pro správu těchto práv. Typickými právy jsou čtení, zápis a spuštění a mohou svědčit pouze vlastníkovi, skupině anebo všem. Parametr se může zadávat třemi různými způsoby:

- `--reference=`*soubor*. Nastaví se stejná přístupová práva, jako má *soubor*.
- Oktalové číslo s nejvýše čtyřmi číslicemi, jímž se zadávají *absolutní* práva po jednotlivých bitech. Význam první číslice zleva si popíšeme později, druhou, třetí a čtvrtou číslicí se zadávají práva vlastníka, skupiny a ostatních. Viz příklad na obr. 3, kde jsou popsána práva určená číslem 0640.
- Jeden nebo více řetězců oddělených čárkami, jimiž se zadávají absolutní nebo relativní přístupová práva (tj. relativní vzhledem k existujícím právům).



Obrázek 3. *Význam jednotlivých bitů pro přístupová práva*

Ve třetím tvaru se každý řetězec skládá ze tří částí: volitelný *rozsah*, *příkaz* a *přístupová práva*.

Rozsah

`u` je uživatel, `g` je skupina, `o` jsou ostatní uživatelé, kteří nejsou ve skupině, `a` jsou všichni uživatelé. Implicitní je `a`.

Příkaz

`+` znamená přidat práva, `-` znamená ubrat práva, `=` znamená nastavit práva absolutně bez ohledu na stávající nastavení

Práva

`r` jako číst, `w` jako psát/modifikovat, `x` jako spustit (v případě adresářů znamená `x` povolení k příkazu `cd` v daném adresáři), `X` jako podmíněné spuštění (vysvětlíme později), `u` je duplikace uživatelských práv, `g` je duplikace skupinových práv, `o` je duplikace práv ostatních uživatelů, `s` je setuid nebo `setgid` a `t` je „sticky" bit (příznak rezidentnosti v operační paměti).

Například `ug+rw` je přidání práv číst a psát vlastníkovi a skupině, `a-x` (nebo jen `-x`) znamená všem zrušit právo soubor spustit a příkazem `u=r` se vlastníkovi nejdříve zruší všechna práva a poté se nastaví jen na čtení. Řetězce lze kombinovat pomocí čárky jako oddělovače, např. `ug+rw,a-x`.

Setuid a setgid se vztahují ke spustitelným souborům (programům a skriptům). Předpokládejme, že máme spustitelný soubor *F*, jehož vlastníkem je „novak" a skupinu „pratele". Má-li soubor *F* setuid („set user ID") nastavený, pak každý, kdo spustí *F*, „se stane" uživatelem novak se všemi jeho právy a privilegii po celou dobu běhu programu. Tedy, má-li F setgid („set group ID") nastavený, každý, kdo spustí program F, se stane členem skupiny pratele po celou dobu běhu programu.. Jak si jistě každý umí představit, setuid a setgid může narušit bezpečnost systému, takže příkaz chmod by se neměl zadávat, dokud uživatel zcela bezpečně neví, *co* dělá. Jediné chybné provedení příkazu `chmod +s` může učinit systém zcela bezbranným proti napadení.

Podmíněné právo ke spuštění (`X`) znamená totéž co `x` s tím rozdílem, že se provede jen tehdy, je-li soubor skutečně spustitelný nebo je-li souborem adresář. Jinak se nestane nic.

Užitečné volby

`-R` Rekurzivní změna vlastníka uvnitř adresářové struktury

chattr [*volby*] [+ - =]*vlastnosti* [*soubory*] e2fsprongs

/usr/bin **stdin stdout -file –opt –help –version**

Pokud jste dříve pracovali s jinými unixovými systémy, možná vás překvapí, že linuxové soubory mohou mít ještě jiné vlastnosti než přístupová práva. Je-li soubor v systému souborů ext2 nebo ext3 (což je ve Fedoře implicitní), lze tyto rozšířené vlastnosti nastavit pomocí příkazu `chattr` a vypsat příkazem `lsattr`.

Podobně jako u `chmod`, atributy lze přidávat (+) nebo ubírat (-) relativně i absolutně (=).

Atribut	Význam
a	Pouze rozšíření: tento soubor lze pouze rozšiřovat, ale nelze ho jinak měnit
A	Přístupy k tomuto souboru nemají vliv na změnu časové známky
c	Komprimace: data se při zápisu automaticky komprimují a při čtení rozbalují
d	Nepořizují se záložní kopie tohoto souboru (příkaz du, viz „Zálohování a vzdálená paměť" na str. 84)
i	Soubor nelze změnit ani vymazat
j	Žurnální data (pouze systém souborů ext3)
s	Bezpečné vymazání: po vymazání souboru se data přepíší nulami
S	Synchronizovaná aktualizace: změny se ihned zapisují na disk, jako by se po uložení zadalo `sync` (viz „Disky a souborový systém" na str. 80)
u	Soubor nelze vymazat

Užitečné volby

`-R` Rekurzivní zpracování adresářů

lsattr [*volby*] [*soubory*] e2fsprongs

/usr/bin **stdin stdout -file –opt –help –version**

Nastaví-li se rozšířené vlastnosti příkazem `chattr`, lze si je vypsat příkazem `lsattr` („list attributes"). Na výstupu se objevují stejná písmena jako v `chattr`; např. tento soubor nelze změnit a nelze jej smazat:

```
$ lsattr mujsoubor
-u--i--- mujsoubor
```

Užitečné volby

`-R` Rekurzivní zpracování adresářů

`-a` Výpis všech souborů vč. souborů začínajících tečkou

`-d` Výpíše se pouze adresář, nikoli jeho obsah

Není-li uveden soubor, vypíší se vlastnosti všech souborů v běžném adresáři.

Umístění souborů

find	Lokalizuje soubory v adresářové struktuře
slocate	Vytvoř index souborů a hledej index řetězce
which	Vyhledání spustitelných souborů v cestě (příkaz)
type	Vyhledání spustitelných souborů v cestě (bash)
whereis	Vyhledání spustitelných souborů, dokumentace a zdrojových souborů

Linux je schopen snadno spravovat desítky nebo i stovky tisíc souborů. Otázkou je, jakým způsobem se jednotlivé soubory hledají. Především by měly být v adresářích nějak smysluplně uspořádány, ale existuje i řada jiných možností vyhledávání. Záleží na tom, co vlastně hledáme.

Každý soubor lze nalézt „hrubou silou" pomocí příkazu `find`, který prochází adresářovou strukturou soubor po souboru. Příkaz **slocate** je rychlejší, avšak soubory musí být nejdříve indexovány. (Fedora provádí indexaci souborů implicitně jako tzv. „noční prácí").

Programy v shellové adresářové cestě se vyhledávají pomocí příkazů `which` a `type`. Příkaz `type` je zabudován do bash shellu (a tedy přístupný, jen když běží `bash`), zatímco `which` je program (obvykle jako */usr/bin/which*); `type` je rychlejší a umí rozpoznat shellová synonyma (alias)*. Naopak, `whereis` hledá v zadané množině adresářů, nikoli ve vyhledávací cestě.

find [*adresáře*] [*výraz*]

findutils

/usr/bin stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `find` se v jednom nebo více adresářích (a rekurzívně i v jejich podadresářích) hledají soubory vyhovující zadaným kritériím. Je to velice mocný příkaz s více než 50 volbami, i když, bohužel, s poněkud nezvyklou syntaxí. Některé jednoduché příklady, jimiž se prohledává celý systém souborů počínaje kořenovým adresářem:

Nalezení určitého souboru jménem *mujsoubor*:

```
$ find / -type f -name mujsoubor -print
```

Výpis všech jmen adresářů:

```
$ find / -type d -print
```

Užitečné volby

`-name` *vzorek*
`-path` *vzorek*
`-lname` *vzorek*
`-iname` *vzorek*
`-ipath` *vzorek*
`-ilname` *vzorek*

Jméno (`-name`), cesta (`-path`) a symbolický odkaz (`-lname`) musí souhlasit se shellovým vzorkem, který může obsahovat i zástupné znaky *, ? a []. Cesta je relativní vzhledem k prohledávanému adresářovému stromu. Volby `-iname`, `-ipath` a `-ilname` jsou shodné s `-name`, `-path` a `-lname`, avšak nerozlišuje se mezi dolními a horními znaky.

`-regex` *regvyraz*

Cesta (relativní vzhledem k prohledávanému adresářovému stromu) musí souhlasit se zadaným regulárním výrazem

* Shell `tcsh` používá k detekci synonym pomocí `which` jisté „triky".

`-type f|d|l|b|c|p|s`

Vyhledání pouze textových souborů (`f`), adresářů (`d`), symbolických odkazů (`l`), zařízení přenášejících bloky (`b`), znaky (`c`), pojmenovaných rour (`p`) nebo socketů (`s`)

`-atime` *N*
`-ctime` *N*
`-mtime` *N*

Poslední přístup k (`-atime`), poslední modifikace (`-mtime`) nebo poslední změna stavu (`-ctime`) souboru přesně před *N*`*24` hodinami. *+N* znamená „větší než *N*", *-N* znamená „menší než *N*".

`-amin` *N*
`-cmin` *N*
`-mmin` *N*

Poslední přístup k (`-atime`), poslední modifikace (`-mtime`) nebo poslední změna stavu (`-ctime`) souboru přesně před *N* minutami. *+N* znamená „větší než *N*", *-N* znamená „menší než *N*".

`-anewer` *jiný_soubor*
`-cnewer` *jiný_soubor*
`-newer` *jiný_soubor*

Poslední přístup k (`-atime`), poslední modifikace (`-mtime`) nebo poslední změna stavu (`-ctime`) souboru dříve než soubor *jiný_soubor*.

`-maxdepth` *N*
`-mindepth` *N*

Soubor leží nejméně (`-mindepth`) nebo nejvýše (`-maxdepth`) v *N*-té úrovni adresářového stromu.

`-follow`

Odstranění symbolického odkazu

`-depth`

Hledání od konce (rekurzívní hledání); nejdříve se prohledává obsah (podřízeného) adresáře, pak teprve adresář samotný

`-xdev`

Hledá se pouze v systému souborů, nikoli v zařízeních

`-size` *N*`[bckw]`

Soubory o velikosti *N* zadané v blocích (`b`), jednobajtových znacích (`c`), kilobajtech (`k`) nebo dvoubajtových slovech (`w`). *+N* znamená „větší než *N*", *-N* znamená „menší než *N*".

`-empty`

Soubor má nulovou velikost a může to být jak běžný soubor, tak i adresář

`-user` *jméno*
`-group` *jméno*

Pouze soubory uvedeného vlastníka nebo skupiny

`-perm` *režim*
`-perm -`*režim*
`-perm +`*režim*

Přístupová práva odpovídající parametru režim. *-mode* znamená *všechny* bity (jimiž jsou nastavena přístupová práva), *+mode* *některé* bity.

Pomocí následujících operátorů se mohou části výrazů slučovat a negovat:

výraz1 -a výraz2

And. (Implicitní hodnota, když dva výrazy stojí vedle sebe; „-a" je nepovinné.

výraz1 -o výraz2

Nebo

! výraz

-not výraz

Negace výrazu

(výraz)

Priorita zadaná závorkami (v algebraickém smyslu). Obsah závorek se vyhodnocuje nejdříve. Zrušit tento význam závorky v shellu lze pomocí „\".

výraz1, výraz2

Čárka má stejný význam jako v jazyce C. Oba výrazy se vyhodnotí, vrátí se hodnota druhého z nich.

Příkaz `find` pak provádí vyhledávání dle takto zadaných kritérií.

Užitečné volby

`-print`	Prostý výpis vyhledávací cesty k souboru, relativní vzhledem k prohledávanému adresáři.
`-printf` *řetězec*	Výpis řetězce zadaného dle syntaxe knihovní funkce `printf ()` jazyka C. Viz manuál.
`-print0`	Stejný jako `-print` s tím rozdílem, že se řádky ukončují znakem `0` (v ASCII). Vhodný např. pro výstup do roury, obsahují-li jména souborů mezery. Příjemcem musí být pochopitelně program, který umí takovýto výstup číst a rozebrat, např. `xargs -0`.
`-ok` *cmd* ; `-exec` *cmd* ;	Spuštění shellového příkazu `cmd`. Funkce všech metaznaků musí být vyrušeny včetně (povinného) středníku na konci, neboť nejsou na příkazovém řádku prováděny. Taktéž znaky „{}" (musí být uzávorkovány nebo uvozeny znakem „\") se chápou jako cesta k hledanému souboru. Tvar `-ok` upozorní uživatele na spuštění shellového příkazu, zatímco `-exec` nikoli.
`-ls`	Povede se `ls -dils`

Příkaz `find` je vhodný v kombinaci s `xargs`, jehož vstupem je seznam souborů a provádí s nimi příkazy (viz popis `xargs` v manuálu). V následujícím příkazu je uvedeno, jak lze v běžné adresářové struktuře nalézt soubory obsahující slovo „myxomatóza":

```
$ find . -print0 \ xargs -0 grep myxomatóza
```


slocate [*volby*] slocate

/usr/bin **stdin** **stdout** **-file** **–opt** **–help** **–version**

Příkaz `slocate` („secure locate") je určen k rychlému vyhledávání souborů indexovou metodou. Je vhodný zejména pro případ, kdy se opakovaně prohledává velké množství souborů, jejichž adresářová struktura se příliš nemění. K vyhledání jednoho souboru anebo ke složitějším způsobům hledání se hodí spíše příkaz `find`.

Fedora vytváří indexy automaticky jednou za den, uživatel však může spustit indexaci sám (indexy se uloží např. do */tmp/mujindex*):

```
$ slocate -u -o /tmp/mujindex
```

Indexy daného adresáře a všech podadresářů se vytvoří příkazem:

```
$ slocate -U adresář -o /tmp/mujindex
```

a řetězec v indexech se nalezne pomocí příkazu:

```
$ slocate -d /tmp/mujindex řetězec
```

A proč bezpečné (secure)? V průběhu prohledávání se nevypisují jména souborů, k nimž uživatel nemá přístupová práva. Vytvoří-li indexy chráněných souborů superuživatel, normální uživatel je sice může používat k běžnému hledání, avšak soubory, ke kterým nemá přístup, se mu nezobrazí.

Volby pro indexování

`-u`	Vytvoření indexů od kořenového adresáře dolů
`-U` *adresář*	Vytvoření indexů od *adresáře* dolů
`-l (0 | 1)`	Vypnutí / zapnutí bezpečného vyhledávání. Implicitní je 1.
`-e` *adresáře*	Vyloučení jednoho nebo více adresářů z vyhledávání. Cesty se oddělují čárkami
`-o` *výstsoubor*	Indexy se zapisují do *výstsoubor*

Volby pro vyhledávání

`-d` *index*	Zadání indexového souboru
`-i`	Nerozlišuj horní / dolní znaky
`-r` *regvýraz*	Jména souborů se zadávají pomocí regulárních výrazů

which [*soubor*] — which

/usr/bin — stdin **stdout** -file **–opt** **–help** **–version**

Tímto příkazem se v cestě hledá spustitelný soubor. Spustí-li se program zadáním jeho jména:

```
$ who
```

příkazem `which` se zjistí, kde je uložen:

```
$ which who
/usr/bin/who
```

Lze také nalézt program `which` samotný:

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

Pokud se v dané cestě vyskytne několik programů stejného jména (např. */usr/bin/who* a */usr/local/bin/who*), vypíše se první z nich.

type [*volby*] *příkazy* — bash

součást shellu — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `type` se podobně jako příkazem `which` vyhledává spustitelný soubor v shellové cestě:

```
$ type grep who
grep is /bin/grep
who is /usr/bin/who
```

`Type` je ovšem zabudovaný příkaz, zatímco `which` je programem na disku:

```
$ type which type rm if
which is /usr/bin/which
type is a shell builtin
rm is aliased to `/bin/rm -i'
if is a shell keyword
```

`Type`, jako zabudovaný příkaz, je rychlejší než `which`; lze jej však použít, pouze je-li spuštěn `bash`.

whereis [*volby*] *soubory* — bash

/usr/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkaz `whereis` zkouší nalézt dané soubory pomocí prohledávání zakódovaného seznamu adresářů. Umí vyhledat spustitelné soubory, dokumentaci a zdrojový kód. Chování tohoto příkazu nemusí vždy být korektní, neboť seznam adresářů nemusí obsahovat právě ten adresář, který bychom potřebovali.

Užitečné volby

`-b`
`-m`
`-s`
	Výpis pouze spustitelných (`-b`), manuálových (`-m`) nebo zdrojových souborů (`-s`).

`-B` *adresáře...* `-f` *soubory...* `-M` *adresáře...* `-f` *soubory...* `-S` *adresáře...* `-f` *soubory...*	Hledání spustitelných (`-B`), manuálových (`-M`) nebo zdrojových souborů (`-S`) jen v uvedených adresářích. Seznam adresářů musí být od seznamu souborů oddělen volbou `-f`.

Práce s texty

grep	Vyhledávání řádků, které odpovídají regulárnímu výrazu
cut	Extrakce sloupců ze souboru
paste	Přidání souborů
tr	Překlad znaků
sort	Třídění řádků textu dle různých kritérií
uniq	Hledání identických řádků
tee	Kopírování souboru se současným výpisem na standardním výstupu

Jednou z nejsilnějších stránek Linuxu je práce s texty tj. formování textových souborů (nebo standardního vstupu) do požadovaného tvaru pomocí transformací. Každý program, který čte ze standardního vstupu a píše na standardní výstup, spadá do této kategorie, avšak zmíníme se jen o těch nejčastěji používaných a nejvýkonnějších.

grep [*volby*] *vzorek* [*soubory*] — grep

/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkaz `grep` je jedním z nejmocnějších a nejpoužívanějších příkazů z celého arzenálu Linuxu. Jeho účel je prostý: zadá se mu jeden nebo více souborů a vypíší se všechny řádky těch souborů, které se shodují s uvedeným regulárním výrazem. Například, když soubor obsahuje tyto řádky:

```
Jedna dvě, Honza jde, nese pytel mouky
máma se raduje, že bude péct vdolky
Film v jedenáct.
```

a chceme najít všechny řádky, které obsahují slovo „vdolky“, dostaneme:

```
$ grep vdolky mujsoubor
máma se raduje, že bude péct vdolky
```

`grep` používá dva druhy regulárních výrazů, které se nazývají *základní* a *rozšířený*. Oba jsou dostatečně mocné, ale liší se od sebe. Který z nich komu vyhovuje, se zjistí až delším užíváním. Základní syntaxe je uvedena v tabulkách 2 a 3.

Užitečné volby

`-v`	Výpis jen těch řádků, které se neshodují s regulárním výrazem
`-l`	Výpis pouze jmen souborů, které obsahují shodné řádky, nikoli řádky samotné
`-L`	Výpis pouze jmen souborů, které neobsahují shodné řádky, nikoli řádky samotné
`-c`	Výpis počtu shodných řádků
`-n`	Výpis číslovaných řádků (původní číslo v rámci souboru)
`-b`	Před každým řádkem na výstupu se vypisuje vzdálenost řádku od počátku souboru v bajtech.
`-i`	Nerozlišovat mezi horními a dolními znaky
`-w`	Vyhledávání celých slov (která se shodují s regulárním výrazem).
`-x`	Vyhledávání celých řádků. Má přednost před `-w`.
`-A N`	Po nalezení každé shody se vypíše *N* následujících řádků
`-B N`	Po nalezení každé shody se vypíše *N* předchozích řádků
`-C N`	Po nalezení každé shody se vypíše *N* předchozích i následujících řádků
`-r`	Rekurzívní prohledávání všech souborů v adresáři a všech subadresářích.
`-E`	Použití rozšířených regulárních výrazů. Viz egrep.
`-F`	Místo regulárních výrazů se používá seznam pevných řetězců.

egrep [*volby*] *vzorek* [*soubory*] **grep**

/bin **stdin stdout -file -opt -help -version**

Příkaz `egrep` je shodný s příkazem `grep`, pouze používá jiný („rozšířený") jazyk v regulárních výrazech. Totéž, jako když zadáme `grep -E`.

Tabulka 2. *Základní a rozšířené regulární výrazy příkazu grep*

Regulární výraz	Význam
`.`	Každý znak
`[...]`	Hledej znak v tomto seznamu
`[^...]`	Hledej znak, který není v tomto seznamu
`(...)`	Seskupení
`^`	Začátek řádku
`$`	Konec řádku
`\<`	Začátek slova
`\>`	Konec slova
`[:alnum:]`	Alfanumerický znak
`[:alpha:]`	Alfabetický znak

Regulární výraz	Význam
`[:cntrl:]`	Kontrolní znak
`[:digit:]`	Číslice
`[:graph:]`	Grafický znak
`[:lower:]`	Spodní znak
`[:print:]`	Tisknutelný znak
`[:punct:]`	Interpunkční znak
`[:space:]`	Mezera
`[:upper:]`	Horní znak
`[:xdigit:]`	Hexadecimální číslice
*	0-násobné anebo vícenásobné opakování regulárního výrazu
`\c`	`c` jako znak, i když `c` je v regulárním výraze funkčním znakem. Např. hvězdička se zadává jako `\*`, opačné lomítko se zadává jako `\\`. Alternativně je možno funkční znak uzavřít do hranatých závorek: [*] nebo [\].

Tabulka 3. *Rozdíly mezi základním a rozšířeným regulárním výrazem*

Základní	Rozšířený	Význam
`|`	`|`	Nebo
`\+`	`+`	Jedno nebo více opakování regulárního výrazu
`\?`	`?`	Žádný nebo jeden výskyt regulárního výrazu
`\{n\}`	`{n}`	Přesně n opakování regulárního výrazu
`\{n,\}`	`{n,}`	`n` nebo více opakování regulárního výrazu
`\{n,m\}`	`{n,m}`	Opakování regulárního výrazu od `n` po `m` (včetně), `n < m`

fgrep [*volby*] [*pevné_řetězce*] [*soubory*] **grep**

/bin **stdin stdout -file -opt -help -version**

Příkaz `fgrep` je shodný s příkazem `grep` s tím rozdílem, že místo regulárních výrazů se zadává seznam pevných řetězců oddělených znakem „nový řádek". Stejnou funkci má příkaz `grep -F`. Například vyhledání řetězců jedna, dvě a tři v souboru `mujsoubor` se zadá:

```
$ fgrep 'jedna    Za řetězce se píše znak „nový řádek".
dvě
tři' mujsoubor
```

`fgrep` se pužívá obvykle s dolním znakem `-f`, jímž se čtou vzorky ze souboru. Například máme-li slovník jako soubor s řetězci na každém řádku:

```
$ cat můj_slovník
abatyše
```

```
abbé
abdikovat
...
```

můžete tyto řetězce v množině vstupních souborů pohodlně vyhledat pomocí

```
$ fgrep -f můj_slovník *
```

Příkaz `fgrep` je vhodný i pro hledání nealfanumerických znaků jako * nebo {, neboť se zde nepovažují za metaznaky.

cut -(b|c|f)*rozsah* [*volby*] [*soubory*]

coreutils

/usr/bin **stdin stdout -file -opt -help -version**

Příkazem `cut` se extrahují sloupce textů ze souborů. „Sloupec" se definuje buď jako vzdálenost od počátku ve znacích (tj. devatenáctý znak každého řádku):

```
$ cut -c19 mujsoubor
```

nebo jako vzdálenost v bajtech (což je totéž jako ve znacích, pokud se ovšem nepracuje s jazykem, který má vícebajtové znaky):

```
$ cut -b19 mujsoubor
```

anebo jako ohraničená pole (tj. páté pole, pole se oddělují čárkou):

```
$ cut -d, -f5 mujsoubor
```

Může se vypisovat více než jeden sloupec: lze zadat rozsah (3–16), posloupnost oddělenou čárkami (3,4,5,6,8,16) nebo obojí: (3,4,8–16). Zadá-li se rozsah bez prvního čísla (–16), dosadí se na jeho místo 1 (1–16); vynechá-li se poslední číslo, míní se tím rozsah „do konce řádku".

Užitečné volby

`-d C` — Oddělovačem polí (`-f`) na *vstupu* je znak `C`. Implicitně je to tabulátor.

`--output-delimiter=C` — Oddělovačem polí (`-f`) na *výstupu* je znak `C`. Implicitně je to tabulátor.

`-s` — Potlačení tisku řádků, které neobsahují oddělovač.

paste [*volby*] [*soubory*]

coreutils

/usr/bin **stdin stdout -file -opt -help -version**

Příkaz `paste` je opačným příkazem ke `cut`: Zadané soubory považuje za svislý sloupec, který přidává do standardního výstupu.

```
$ cat písmena
A
B
C
```



```
$ cat čísla
1
2
3
4
5
paste čísla písmena
1       A
2       B
3       C
4
5
paste písmena čísla
A       1
B       2
C       3
4
5
```

Užitečné volby

`-d oddělovače` — Souborem *oddělovače* se zadají oddělovače mezi sloupci; implicitně je to tabulátor. Lze zadat jediný oddělovač (`-d:`) nebo seznam oddělovačů (`-dxyz`), které se na každém řádku berou postupně (prvním oddělovačem je `x`, druhým `y`, pak `z`, pak znovu `x` atd.)

`-s` — Transpozice řádků a sloupců na výstupu:

```
$ paste -s písmena čísla
A  B  C
1  2  3  4  5
```

tr [*volby*] *znak_množ1* [*znak_množ2*]

coreutils

/usr/bin **stdin stdout -file -opt -help -version**

Příkazem `tr` se provádí jednoduchá náhrada jedné znakové množiny jinou. Chceme-li např. nahradit všechny samohlásky hvězdičkou, zadáme:

```
$ cat mujsoubor
Toto je můj kouzelný souboreček.
$ cat mujsoubor | tr aáeéěiíoóuúůyýAÁEÉĚIÍOÓUÚŮYÝ '*'
T*t* j* m*j k**z*ln* s**b*r*č*k.
```

anebo vynechat všechny samohlásky:

```
$ cat mujsoubor | tr aáeéěiíoóuúůyýAÁEÉĚIÍOÓUÚŮYÝ
Tt j mj kzln sbrčk.
```

anebo nahradit malá písmena velkými:

```
$ cat mujsoubor | tr 'a-z' 'A-Z'
TOTO JE MŮJ KOUZELNÝ SOUBOREČEK.
```

Příkaz `tr` nahradí první znak ze *znak_množ1* prvním znakem ze *znak_množ2*, druhý druhým, třetí třetím atd. Je-li délka *znak_množ1* rovna *N*, ze *znak_množ2* se vezme jen prvních *N* znaků. (Je-li *znak_množ1* delší než *znak_množ2*, viz volba `-t`).

Znakové množiny mohou vypadat například takto:

Tvar	Význam
`ABCD`	Posloupnost znaků A, B, C, D.
`A-B`	Rozsah znaků od A po B
`[x*y]`	y-krát opakovat znak x
`[:class:]`	Třídy znaků (`[:alnum:]`,`[:digit:]`) stejné jako v příkazu `grep`

Znaky v příkazu tr lze zadávat i pomocí zpětného lomítka: „\a“ (^G = zvonek) „\b“ (^H = zpět) „\f“ (^L = posun řádku) „\n“ (^] = nový řádek) „\r“ (^M = návrat) „\t“ (^I = tabulátor) „\v“ (^K = vertikální tabulátor), těmto znakům rozumí příkaz printf. (viz „Výstup na obrazovku“ na str. 124). Podobně notace \nnn znamená znak s oktalovou hodnotou nnn.

Příkaz `tr` je vhodný pro rychlou a jednoduchou náhradu; složitější nahrazování znaků se provádí pomocí příkazů `sed`, `awk` a `perl`.

Užitečné volby

`-d` Na výstupu vynech znaky uvedené v *znak_množ1*

`-s` Stejné znaky (ze *znak_množ1*) stojící vedle sebe se nahradí znakem jediným. Například příkazem `tr -s aeiouyAEIOUY` se opakované samohlásky nahradí jen jedinou: ze slova `straaaaana` se stane `strana`.

`-c` Pracuje se všemi znaky, které nejsou uvedeny v *znak_množ1*.

`-t` Je-li *znak_množ1* delší než *znak_množ2*, *znak_množ1* se zkrátí na stejnou délku jako má *znak_množ2*. Není-li uvedena volba `-t`, poslední znak ze *znak_množ2* se zopakuje tolikrát, až se dosáhne stejné délky jako má *znak_množ1*.

sort [*volby*] [*soubory*]

coreutils

/bin **stdin stdout -file -opt -help -version**

Příkazem `sort` se vypisují textové řádky v abecedním pořádku anebo setříděné podle jiného (zadaného) pravidla. Soubory se zřetězí, výsledek se setřídí a vypíše.

```
$ cat mujsoubor
def
xyz
abc
$ sort mujsoubor
abc
sef
xyz
```


Užitečné volby

`-f` Nerozlišuje se mezi horními a dolními znaky

`-n` Numerické třídění (9 je před 10), nikoli abecední (10 by bylo před 9)

`-g` Jiný algoritmus numerického třídění. Bere se v úvahu exponenciální tvar čísel (7.4e3 znamená „7,4 krát deset na třetí“, tedy 7400).

`-u` Ignorování duplicitních řádků. (Použije-li se společně s volbou `-c`, kterou se zjišťuje, zda je soubor setříděný, a v souboru existují shodné řádky, příkaz se neprovede).

`-c` Netřídí se, pouze se zjistí, zda je vstup setříděný. Pokud ano, nevypíše se nic, jestiže není, vypíše se chybové hlášení.

`-b` Ignorují se levostranné mezery.

`-r` Reverzní výstup: třídí se od největšího k nejmenšímu.

`-t X` Použij `X` jako oddělovač polí ve volbě `-k`

`-k F1[.C1][,F2[.C2]]` Výběr klíčů pro třídění

Klíč je část řádku, podle níž se třídí. Například se může třídit podle pátého znaku v řádku. Normálně by se řádky setřídily:

```
aaaaz
bbbby
```

Třídí-li se však podle 5. znaku, pořadí třídění je opačné. Syntaxe zadávání klíčů:

Položka	Význam	Implicitní hodnota
`F1`	Počáteční pole	Neexistuje, musí se uvést
`C1`	Počáteční umístění uvnitř prvního pole	1
`F2`	Koncové pole	Poslední pole
`C2`	Počáteční umístění uvnitř posledního pole	1

V případě `sort -k1.5` se bude třídit podle prvního pole, začátek klíče je na pátém znaku; `sort -k2.8,5` znamená „od osmého znaku druhého pole k prvnímu znaku pátého pole“.

Vícenásobné klíče se zadávají pomocí opakování volby `-k` a budou se aplikovat jeden po druhém.

uniq [*volby*] [*soubory*] coreutils

/usr/bin stdin stdout -file –opt –help –version

Příkazem `uniq` se zpracovávají za sebou jdoucí duplicitní řádky. Máme-li například soubor:

```
$ cat mujsoubor
a
b
b
c
b
```

příkazem `uniq` se rozpoznají a zpracují (zadaným způsobem) dvě po sobě jdoucí b, nikoli však třetí b.

```
$ uniq mujsoubor
a
b
c
b
```

Příkaz `uniq` se často používá po setřídění souboru:

```
$ sort mujsoubor | uniq
a
b
c
```

a zůstane jen jediný řádek s `b`. Duplicitní řádky lze také třeba jen spočítat:

```
$ sort mujsoubor | uniq -c
      1 a
      3 b
      1 c
```

Užitečné volby

`-c`	Spočítej sousedící shodné řádky
`-i`	Ignoruj horní / dolní
`-u`	Vypiš pouze unikátní řádky
`-d`	Vypiš pouze duplicitní řádky
`-s` *N*	Při vyhledávání duplicity ignoruj prvních *N* znaků na řádku
`-f` *N*	Při vyhledávání duplicity ignoruj prvních *N* polí oddělených mezerami
`-w` *N*	Při vyhledávání duplicity vezmi v úvahu jen prvních *N* znaků na řádku. Použije-li se příkaz s volbami `-s` nebo `-f`, příkaz bude nejdříve ignorovat zadaný počet znaků nebo polí, a pak bude uvažovat *N* znaků.

tee [*volby*] [*soubory*] coreutils

/usr/bin stdin stdout -file –opt –help –version

Podobně jako v případě `cat`, příkazem `tee` se standardní vstup beze změny kopíruje na standardní výstup. Současně však se může vstup kopírovat do jednoho nebo více souborů a tento příkaz se obvykle používá „uprostřed" rour, když je třeba předávat data jinému příkazu v rouře:

```
$ who | tee původní_who | sort
```

Tímto příkazem se setříděný výstup `who` vypíše na standardním výstupu, ale také se původní (nesetříděný) obsah uloží do souboru *původní_who*.

Užitečné volby

`-a`	Soubor na výstupu se nepřepisuje, nýbrž rozšiřuje
`-i`	Nereaguje se na přerušení

Výkonnější nástroje pro práci s texty

V oblasti zpracování textů jsme se zatím dotkli jen špičky ledovce. V Linuxu lze komplexně zpracovávat data pomocí stovek nejrůznějších filtrů, avšak s jejich vzrůstající silou narůstá i křivka osvojování znalostí, což překračuje rámec stručného průvodce. Uvedeme pouze několik příkladů pro další inspiraci:

awk

`awk` je jazyk na vyhledávání podle vzorků. Vyhledávání lze zadávat pomocí regulárních výrazů a další akce se mohou provádět na základě výsledků hledání. Zde je několk jednoduchých příkladů na dané téma:

Vypiš každé druhé a čtvrté slovo ze souboru *mujsoubor*:

```
$ awk '{print $2, $4}' mujsoubor
```

Výpis všech řádků, které jsou kratší než 60 znaků:

```
$ awk '{length($0) < 60}' mujsoubor
```

sed

Podobně jako v `awk`, sed je nástroj na hledání podle vzorků. Syntaxe vychází z `vim` a z řádkového editoru `ed`. Zde je několik jednoduchých příkladů:

Výpis souboru obsahujícího řetězec „red", který se současně změní na „hat":

```
$ sed 's/red/hat/g' mujsoubor
```

Výpis souboru, v kterém se smaže prvních deset řádků:

```
$ sed '1,10d' mujsoubor
```

m4

m4 je jazyk na zpracování maker. Vyhledává klíčová slova v souboru a nahrazuje je zadanými hodnotami. Vezměme třeba soubor:

```
$ cat mujsoubor
Jmenuji se JMÉNO a je mi VĚK let
ifelse(ZÁVORKA,yes,No matter where you go… there you are)
```

a jak proběhne substituce maker JMÉNO, VĚK a ZÁVORKA:

```
$ m4 -DJMÉNO=Andulka mujsoubor
Jmenuji se Andulka a je mi VĚK let
$ m4 -DNAME=Andulka -DVĚK=25 mujsoubor
Jmenuji se Andulka a je mi 25 let
No matter where you go… there you are
```

perl, python

Perl a Python jsou plnohodnotné skriptové jazyky k tvorbě rozsáhlých aplikací

Komprimace a balení souborů

gzip	Komprimace pomocí GNU Zip
gunzip	Dekomprimace pomocí GNU Zip
compress	Komprimace pomocí klasického unixového příkazu
uncompress	Dekomprimace pomocí klasického unixového příkazu
zcat	Komprimace / dekomprimace standardních V/V (gzip nebo compress)
bzip2	Komprimace pomocí BZip
bunzip2	Dekomprimace pomocí BZip
zip	Komprimace do windowsového tvaru (zip)
unzip	Dekomprimace z windowsového tvaru (zip)
uuencode	Konverze do tvaru uuencode
uudecode	Konverze z tvaru uuencode

Linux umí komprimovat a dekomprimovat soubory do/z mnoha tvarů. Nejpoužívanějším tvarem je GNU Zip (gzip), jehož komprimované soubory mají příponu .gz. Ostatní běžně používané tvary jsou klasický unixový komprimační program (přípona .Z), bzip2 (bz2), a Zip z Windows (zip).

Příslušné technologie umožňují konverzi binárních souborů do textových tvarů, takže je lze přenášet například jako elektronickou zprávu. To se nyní děje automaticky s přílohami a nástroji MIME, zmíníme se nicméně i o starších způsobech kódování uuencode a uudecode, jež se dosud používají.

Pokud se setkáte s tvary, které zde nejsou popsány, jako např. macintoshovské hqx/sit, Arc, Zoo a další, informace o nich lze nalézt např. na *http://www.faqs.org/faqs/compression-faq/part1section-2.html* a *http://www-106.ibm.com/ developerworks/library/l-lw-comp.html.*

gzip [*volby*] [*soubory*] — gzip

/bin — **stdin stdout -file –opt –help –version**

gzip a gunzip komprimují a dekomprimují soubory do/z tvaru GNU.

Příklady příkazů

gzip *soubor*	Komprimace *souboru* do *souboru.gz*. Původní soubor se smaže.
gzip -c *soubor*	Komprimovaná data na standardním výstupu
cat *soubor* | gzip	Komprimovaná data z roury
gunzip *soubor.gz*	Dekomprimace *souboru.gz* do *souboru*. Původní (komprimovaný) soubor se smaže.
gunzip -c *soubor.gz*	Dekomprimovaná data na standardním výstupu
cat *soubor.gz* | gunzip	Dekomprimovaná data z roury
zcat *soubor.z*	Dekomprimovaná data na standardním výstupu

Komprimované tarové soubory: příklady příkazů

tar czf *mujsoubor.tar.gz jméno_adr*	Komprimace adresáře *jméno_adr*
tar tzf *mujsoubor.tar.gz*	Výpis obsahu
tar xzf *mujsoubor.tar.gz*	Rozbalení

Zadáme-li v taru volbu v, vypisují se jména zpracovávaných souborů.

compress [*volby*] [*soubory*] — ncompress

/usr/bin — **stdin stdout -file –opt –help –version**

compress a uncompres komprimují a dekomprimují soubory do/z standardního unixového tvaru (Lempel Ziv). Komprimované soubory mají příponu .Z.

Příklady příkazů

compress *soubor*	Komprimace *souboru* do *souboru.Z*. Původní soubor se smaže.
compress -c *soubor*	Komprimovaná data na standardním výstupu
cat *soubor* | compress	Komprimovaná data z roury
uncompress *soubor.Z*	Dekomprimace *souboru.Z* do *souboru*. Původní (komprimovaný) soubor se smaže.
uncompress -c *soubor.Z*	Dekomprimovaná data na standardním výstupu
cat *soubor.Z* | uncompress	Dekomprimovaná data z roury

`zcat soubor.Z`	Dekomprimovaná data na standardním výstupu

Komprimované tarové soubory: příklady příkazů

`tar cZf mujsoubor.tar.Z jméno_adr`	Komprimace adresáře *jméno_adr*
`tar tZf mujsoubor.tar.Z`	Výpis obsahu
`tar xZf mujsoubor.tar.Z`	Rozbalení

Zadáme-li v taru volbu `v`, vypisují se jména zpracovávaných souborů.

bzip2 [*volby*] [*soubory*] — bzip2

/usr/bin — **stdin stdout -file -opt -help -version**

`bzip2` a `bunzip2` komprimují a dekomprimují soubory do/z Burrowsova-Wheelerova tvaru. Komprimované soubory mají příponu *bz2*.

Příklady příkazů

`bzip2 soubor`	Komprimace *souboru* do *souboru.bz2*. Původní soubor se smaže.
`bzip2 -c soubor`	Komprimovaná data na standardním výstupu
`cat soubor | bzip2`	Komprimovaná data z roury
`bunzip2 soubor.bz2`	Dekomprimace *souboru.bz2* do *souboru*. Původní (komprimovaný) soubor se smaže.
`bunzip2 -c soubor.bz2`	Dekomprimovaná data na standardním výstupu
`cat soubor.bz2 | bunzip2`	Dekomprimovaná data z roury
`bzcat soubor.bz2`	Dekomprimovaná data na standardním výstupu

Komprimované tarové soubory: příklady příkazů

`tar cjf mujsoubor.tar.bz2 jméno_adr`	Komprimace adresáře *jméno_adr*
`tar tjf mujsoubor.tar.bz2`	Výpis obsahu
`tar xjf mujsoubor.tar.bz2`	Rozbalení

Zadáme-li v taru volbu `v`, vypisují se jména zpracovávaných souborů.

zip [*volby*] [*soubory*] — zip

/usr/bin — **stdin stdout -file -opt -help -version**

Příkazy `zip` a `unzip` komprimují a dekomprimují soubory do/z windowsového formátu Zip. Komprimované soubory mají příponu .zip. Na rozdíl od `gzip`, `compress` a `bzip2` tento příkaz nemaže původní soubor(y).

`zip mujsoubor.zip soubor1 soubor2 soubor3...`	*Sbalit*
`zip mujsoubor.zip jmeno_adr`	*Sbalit rekurzívně*
`unzip -l mujsoubor.zip`	*Výpis obsahu*
`unzip mujsoubor.zip`	*Rozbalit*

uuencode [volby] novysoub vstsoub — sharutils

/usr/bin — **stdin stdout -file -opt -help -version**

Před vznikem příloh k e-mailům a MIME bylo obtížné přenášet binární soubory. Musely se nejdříve zakódovat („zauukódovat") do ASCII znaků, a pak vypadaly například takto:

```
begin 644 mujsoubor
M(R'N8F%:S[P<F]F:6QE"B,@4G5N<R!F:7)5="!W:85N(8Q9V=I;F<@:6X@
M=6YD97(@1TY/344*"G1R87@)PH@(1E<WO@+6X@(B134TA?04=%E1
...
end
```

Když příjemce takový soubor obdržel, musel jej zase dekódovat („uudekódovat"), aby získal původní obsah.

Konverze souboru *mujsoubor* do zakódovaného tvaru se provádí příkazem:

```
$ uuencode novysoub mujsoubor > mujsoubor.uu
```

První parametr *novysoub* je soubor, který se vytvoří při dekódování a objeví se na prvním řádku zakódovaného výstupu:

```
begin 644 mujsoubor
M(R'N8F%:S[P<F]F:6QE"B,@4G5N<R!F:7)5="!W:85N(8Q9V=I;F<@:6X@
```

Zakódovaný soubor *mujsoubor.uu* se rozkóduje (a současně se vytvoří *novysoub*) příkazem:

```
$ uudecode mujsoubor.uu
```

Porovnávání souborů

diff	**Porovnávání souborů nebo adresářů po řádcích**
comm	**Porovnávání setříděných souborů po řádcích**
cmp	**Porovnávání souborů po bajtech**
md5sum	**Výpočet kontrolního součtu zadaných souborů (MD5)**

V Linuxu se soubory porovnávají třemi způsoby:

- Po řádcích (`diff`, `diff3`, `sdiff`, `comm`). Vhodné pro textové soubory.
- Po bajtech (`cmp`), používá se u binárních souborů
- Porovnáním kontrolního součtu (`md5sum`, `sum`, `cksum`)

Všechny tyto programy jsou orientovány na práci s texty. Nástroj pro grafické porovnávání naleznete např. na *http://xxdiff.sourceforg.net*, příkaz `xxdiff`.

diff [*volby*] *soub1 soub2* — diffutils

/usr/bin — stdin **stdout -file -opt -help -version**

Příkazem `diff` se porovnávají dva soubory nebo dva adresáře řádek po řádku. Při porovnávání textových souborů se vypisují podrobné zprávy o rozdílech. U binárních souborů se pouze vypíše, zda se soubory od sebe liší či nikoli. Pro všechny soubory platí, že v případě shody se nevypisuje žádná zpráva.

Obvyklý výstupní formát vypadá takto:

Indikace čísla řádku a druh odlišnosti
< Odpovídající část souboru1, pokud existuje

> Odpovídající část souboru2, pokud existuje

Začneme například souborem *soubA*

```
Povím vám moc pěknou pohádku.
Červená Karkulka si vzala košíček
a cupitala do lesa.
To je prozatím všechno.
```

Soubor *soubB* vytvoříme tak, že v *soubA* vynecháme první řádek, „Červená" na druhém řádku nahradíme „Modrá" a nakonec přidáme poslední řádek:

```
Modrá Karkulka si vzala košíček
a cupitala do lesa.
To je prozatím všechno.
linux rOOlz!
```

Příkaz `diff soubA soubB` vypíše:

```
1,2c1                          Z 1. a 2. řádku soubA se stal 1. řádek soubB
< Povím vám moc pěknou pohádku
< Červená Karkulka si vzala košíček
---                            oddělovač diff
> Modrá Karkulka si vzala košíček   1. řádek soubB
4a4                            Byl přidán 4. řádek
> linux rOOlz!                 Přidaný řádek
```

Znaky < a > označují *soubA* a *soubB*. Výstupní tvar je implicitní: existuje i mnoho dalších, některé lze použít jako vstup pro jiné příkazy. Je třeba je vyzkoušet.

Volba	Výstupní tvar
`-n`	Tvar RCS, jaký poskytuje `rcsdiff` (viz manuál `rcsdiff`)
`-c`	Kontextový formát diff, který používá příkaz `patch` (viz manuál `patch`)
`-D macro`	Preprocesorový tvar C, který používá #ifdef `macro` ... #else ... #endif
`-u`	Jednotný formát, který soubory slučuje a vypisuje – při vynechávání a + při přidávání.
`-y`	Paralelní porovnávání; příkazem -W se nastaví šířka dat na výstupu.
`-e`	Vytváří skript, kterým se soubA změní v soubB.
`-q`	Nevypisují se odlišnosti, pouze se zjistí, zda jsou soubory shodné.

Příkazem `diff` také lze porovnávat adresáře:

```
$ diff adr1 adr2
```

Příkaz pracuje tak, že se v těchto adresářích porovnávají soubory se stejným jménem a vypisují se všechny soubory, které se nacházejí jen v jednom z adresářů. Celá struktura adresářů se porovná pomocí volby `-r` (rekurzívní porovnávání):

```
$ diff -r adr1 adr2
```

a o rozdílech se vypisují poměrně obšírné zprávy.

Užitečné volby

`-b`	Ignorují se mezery
`-B`	Ignorují se prázdné řádky
`-i`	Ingnorují se znaky horní / dolní
`-r`	Při porovnávání adresářů se rekurzívně porovnávají i subadresáře

Program `diff` je pouze jedním ze skupiny programů na zjišťování rozdílů. Dalšími jsou např. `diff3`, který porovnává tři soubory současně a `sdiff`, který na základě rozdílů mezi dvěma soubory vytváří třetí soubor, jehož tvar si definuje uživatel.

comm [*volby*] *soub1 soub2* — coreutils

/usr/bin — stdin **stdout -file -opt -help -version**

Příkaz `comm` porovnává dva setříděné soubory a na výstupu vytváří tři sloupce oddělené tabulátory:

1. Všechny řádky, které se vyskytují v *soub1* a nevyskytují se v *soub2*.
2. Všechny řádky, které se vyskytují v *soub2* a nevyskytují se v *soub1*.
3. Všechny řádky, které se vyskytují v obou souborech.

Nechť *soub1* a *soub2* obsahují tyto řádky:

```
soub1:    soub2:
auto        boby
boby        cukr
cukr          dřevo
```

Výstupem příkazu `comm` je:

```
auto
        boby
        cukr
  dřevo
```

Užitečné volby

`-1`	Potlač sloupec 1
`-2`	Potlač sloupec 2
`-3`	Potlač sloupec 3

cmp [*volby*] *soub1 soub2* [*offset1*[*offset2*]] — diffutils

/usr/bin — **stdin stdout -file -opt -help -version**

Příkazem `cmp` se porovnávají dva soubory. Je-li jejich obsah shodný, nehlásí se nic. V opačném případě se vypíše umístění první neshody:

```
$ cmp mujsoubor tvujsoubor
mujsoubor tvujsoubor differ: char 494, line 47
```

Implicitně se nesděluje, v čem se liší, jenom kde se liší. Tento způsob je vhodný pro porovnávání binárních souborů, zatímco pro práci s textovými soubory lze spíše doporučit `diff`.

Normálně probíhá porovnávání od začátku obou souborů, ale může se zadat i porovnávání od jiného místa:

```
$ cmp mujsoubor tvujsoubor 10 20
```

Porovnávání začne od desátého znaku souboru *mujsoubor* a od dvacátého znaku ve *tvujsoubor*.

Užitečné volby

`-l` Dlouhý výstup: vypisují se všechny rozdíly, bajte po bajtu: `$ cmp -l mujsoubor tvujsoubor`

```
494 164 172
```

`-s` Tichý výstup: Nic se nevypisuje; program pouze vrátí příslušnou chybovou hodnotu a ukončí se: 0 když se soubor shoduje, 1 když se neshoduje (anebo jiné kódy, pokud se program ukončí z jiného důvodu)

md5sum *soubory* | -check *soubor* — coreutils

/usr/bin — **stdin stdout -file -opt -help -version**

Příkaz `md5sum` vypisuje 32bajtový kontrolní součet daného souboru vyčíslený algoritmem MD5 (technické detaily viz *http://www.faqs.org/rfcs/rfc1321.html*):

```
$ md5sum mujsoubor
dd3602df1cceb57966d085524c3980f
```

Je velmi nepravděpodobné, že dva různé soubory budou mít stejný kontrolní součet MD5, takže porovnání kontrolního součtu lze považovat za dostatečně spolehlivý způsob, jak zjistit, zda se dva soubory od sebe liší.

```
$ md5sum mujsoub1 > souc1
$ md5sum mujsoub2 > souc2
$ diff -q souc1 souc2
Files souc1 and souc2 differ
```

anebo, pokud se množina souborů změnila, pomocí `--check`

```
$ md5sum soub1 soub2 soub3 > mujsouc
$ md5sum --check mujsouc
soub1: OK
soub2: OK
soub3: OK
$ echo "nová data" > soub2
$ md5sum --check mujsouc
soub1: OK
soub2: FAILED
soub3: OK
md5sum: WARNING: 1 OF 3 computed checksums did NOT match
```

Podobné programy jsou `sum` a `cksum`, které používají k výpočtu kontrolního součtu jiné algoritmy. sum je kompatibilní s jinými unixovými systémy, zejména BSD Unix (implicitně) nebo Unix System V (volba `-s`) a vypočte se kontrolní součet CRC:

```
$ sum mujsoub
12410  3
$ sum -s mujsoub
47909 6 mujsoub
$ cksum mujsoub
1204834076 2863 mujsoub
```

První celé číslo je kontrolní součet, druhé čítač bloků. Avšak, jak lze vidět, tyto kontrolní součty jsou malá čísla, a jsou tedy nespolehlivé, neboť existuje vyšší pravděpodobnost, že by dva soubory mohly mít shodné kontrolní součty. Příkaz `md5sum` je zdaleka nejlepší.

Disky a souborový systém

df	Výpis volného prostoru na připojeném systému souborů
mount	Zpřístupnění diskové oblasti
umount	Odpojení (znepřístupnění) diskové oblasti
fsck	Hledání chyb v diskové oblasti
sync	Fyzický zápis vyrovnávacích pamětí na disk

Linux může pracovat s více disky a s diskovými oblastmi. Běžně se v této souvislosti používají pojmy jako disky, oblasti, souborové systémy, svazky, dokonce i adresáře. Pokusíme se pojmenovat věci přesněji.

Disk je hardwarové zařízení, jež může být rozděleno na *oblasti* (angl. *partitions*). Oblasti jsou v Linuxu tvořeny speciálními soubory, obvykle v adresáři */dev*. Například */dev/hda7* může být oblast na hlavním disku s řadičem IDE. Adresář */dev* běžně obsahuje tato zařízení:

hda	První sběrnice s řadičem IDE, hlavní zařízení; oblasti *hda1*, *hda2*, ...
hdb	První sběrnice s řadičem IDE, podřízené zařízení; oblasti *hdb1*, *hdb2*, ...
hdc	Druhá sběrnice s řadičem IDE, hlavní zařízení; oblasti *hdc1*, *hdc2*, ...
hdd	Druhá sběrnice s řadičem IDE, podřízené zařízení; oblasti *hdd1*, *hdd2*, ...
sda	První zařízení s řadičem SCSI; oblasti *sda1*, *sda2*, ...
sdb	Druhé zařízení s řadičem SCSI; oblasti *sdb1*, *sdb2*, ... Analogicky *sdc*, *sdd*
ht0	První pásková mechanika s řadičem IDE (dále *ht1*, *ht2*, ...) s autom. převíjením
nht0	První pásková mechanika s řadičem IDE (dále *nht1*, *nht2*, ...) bez autom. převíjení
st0	První pásková mechanika s řadičem SCSI (dále *st1*, *st2*, ...)
scd0	První mechanika CD-ROM s řadičem SCSI (dále *scd1*, *scd2*, ...)
fd0	První disketová mechanika (dále *fd1*, *fd2*, ...), obv. připojená k */mnt/floppy*

Do oblasti lze zapisovat soubory až poté, co se „naformátuje", tj. vytvoří se v ní souborový systém, který definuje způsob zápisu souborů do tohoto systému; například ext3 (linuxový žurnálový souborový systém, ve Fedoře implicitní) a vfat (souborový systém Microsoft Windows). Disk se formátuje při instalaci Linuxu.

Poté, co se vytvoří souborový systém, lze jej zpřístupnit připojením k prázdnému adresáři. Například připojí-li se windowsový systém k adresáři */mnt/win*, stane se součástí systémového adresářového stromu a lze v něm

vytvářet a zpracovávat soubory jako např. */mnt/win/mujsoubor*. Souborové systémy lze také odpojovat (znepřístupňovat) např. z důvodů údržby. Připojování pevných disků se děje automaticky při spouštění operačního systému.

df [*volby*][*disková zařízení* | *soubory* | *adresáře*] — coreutils

/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `df` („disk free") se zjišťuje velikost, využitý prostor a volný prostor v diskové oblasti. Zadá-li se soubor nebo adresář, df sdělí, na kterém diskovém zařízení je uložen. Neuvede-li se parametr, vypíše se zpráva o všech připojených souborových systémech.

```
$ df
Filesystem   1k-blocks    Used  Available Use% Mounted on
/dev/hda       1011928  225464     735060  24% /
/dev/hda9       521748  249148     246096  51%       /var
/dev/hda8      8064272 4088636    3565984  54% /usr
/dev/hda10     8064272 4586576    3068044  60% /home
```

Užitečné volby

`-k -m`	Výpis všech velikostí v kilobajtech (implicitně) nebo megabajtech.
`-B` *N*	Výpis velikostí v blocích definované velikosti, přičemž 1 blok = *N* bajtů (Implicitně = 1024).
`-h -H`	Výpis v „čitelné" podobě s automatickým výběrem nejvodnější jednotky. Mají-li např. dva disky 1 gigabajte resp. 25 kilobajtů volného prostoru, `df -h` vypíše 1G a 25K. Při volbě `-h` se berou násobky 1024, při `-H` násobky 1000.
`-l`	Výpis pouze lokálních souborových systémů, nikoli síťových.
`-T`	Vypíše se i typ soub. systému (ext2, vfat, atd.).
`-t` *type*	Výpis souborových systémů uvedeného typu.
`-x` *type*	Výpis souborových systémů kromě systémů uvedeného typu.
`-i`	Inodový režim. Výpis všech, použitých a volných inod ve všech souborových systémech, nikoli tedy diskových bloků.

mount [*volby*] *zařízení* | *adresář* — mount

/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `mount` se zpřístupní hardwarové paměťové zařízení. Nejčastěji jsou to disky (např. */dev/hda1*), které se po provedení tohoto příkazu stanou dostupnými prostřednictvím existujícího adresáře (např. */mnt/mujadr*):

```
# mkdir /mnt/mujadr
# mount /dev/hda1 /mnt/mujadr
```

```
# df /mnt/mujadr
Filesystem  1K-blocks  Used  Available  Use%  Mounted on
/dev/hda1   1011928   285744   674780   30%   /mnt/mujadr
```

Příkaz `mount` má množství voleb, zmíníme se jen o těch nejzákladnějších.

Příkaz `mount` se obvykle provádí tak, že se nejdříve nahlédne do souboru */etc/fstab* (tabulka souborového systému), aby se zjistilo, jak se má uvedený disk připojit. Zadá-li uživatel např. `mount /usr`, hledá se v tabulce řádek s „/usr", jenž může mít např. tento tvar:

```
/dev/hda8    /usr    /ext3    defaults    1    2
```

Zde se mimo jiné zjistí, že diskové zařízení */dev/hda8* by mělo být připojeno k */usr* jako linuxový souborový systém naformátovaný ve tvaru ext3*.

Příkaz `mount` obvykle zadává superuživatel, avšak běžná zařízení jako disketová nebo CD mechanika si uživatel připojuje sám.

```
$ mount /mnt/cdrom
$ mount /mnt/floppy
```

umount [*volby*] [*zařízení* | *adresář*] — mount

/bin — **stdin stdout -file –opt –help –version**

Příkaz `umount` je opakem mount: znepřístupní se jím disková oblast. Je-li například připojena CD mechanika, médium (optický disk, CD-ROM) z ní nelze vysunout, dokud se příkazem umount neodpojí:

```
$ umount /mnt/cdrom
```

Média, která lze vyjmout z mechaniky resp. jinak fyzicky odstranit, je třeba vždy nejdříve odpojit, aby se nepoškodil souborový systém. Všechna připojená zařízení se odpojí příkazem:

```
# umount -a
```

Souborový systém se nesmí odpojit, když je v provozu; ve skutečnosti by se takový příkaz odmítl z důvodů bezpečnosti.

fsck [*volby*] [*zařízení*] — e2fsprogs

/sbin — **stdin stdout -file –opt –help –version**

Příkazem `fsck` („filesystem check") se kontroluje disková oblast a opravují se v ní případné chyby. Tento příkaz probíhá autmaticky při spouštění systému; lze jej však zadat i „ručně". Obecně by před zadáním tohoto příkazu mělo být kontrolované zařízení odpojeno, aby s ním v průběhu kontroly nepracovaly jiné programy:

* Alternativně lze také zadat volbu `-t`, v níž se přímo uvede typ souborového systému, např. mount `-t ext3 /dev/hda1 /mnt/mujadr`. Viz manuál, příkaz `mount`.


```
# umount /dev/hda10
# fsck -f /dev/hda10
Pass 1: Checking inodes, blocks and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference count
Pass 5: Checking group summary information
/home 172/1281696 files (11.6% non-contiguous), 1405555/2562359
blocks
```

`fsck` je nejdůležitějším z mnoha kontrolních programů uložených v adresáři */sbin*, jejichž jméno začíná „fsck". Jsou určeny jen pro některé typy souborových systémů; lze si je vypsat příkazem:

```
$ ls /sbin/fsck.* | cut -d. -f2
```

Užitečné volby

`-A`	Kontrolují se postupně všechny disky z adresáře */etc/fstab*
`-N`	Neprovedou se žádné kontroly, vypíše se jen jejich popis
`-r`	Interaktivní opravovy chyb, před každou opravou se vypíše prompt
`-a`	Automatické opravy chyb (tato volba by se měla zadávat jen tehdy, je-li si uživatel stoprocentně jist tím, co činí; jinak může dojít k vážnému poškození souborového systému)

sync — coreutils

/bin — **stdin stdout -file –opt –help –version**

Příkazem `sync` se přenesou obsahy vyrovnávacích pamětí disků fyzicky na disk. Jádro systému většinou pracuje tak, že některá data, jež mají být uložena na disk (čtení, zápisy, změny v inodech a ostatní diskové operace), zůstávají v „bufferech" v operační paměti s tím, že budou zapsána na disk později. Příkazem sync se všechna taková data ihned přenesou na disk.

Obvykle nejsou žádné důvody k používání tohoto příkazu. Hrozí-li však například havárie systému (třeba výpadek proudu), neuškodí, když občas „vyženeme" data z paměti na disk.

Rozvrhování a formátování disků

Operace jako rozvrhování prostoru na disku a jejich formátování se většinou provádějí současně. Zde jsou programy, které se k těmto činnostem používají (viz příslušné manuály):

`parted, fdisk a sfdisk`	Rozdělení pevného disku na oblasti; funkce všech programů jsou shodné, liší se pouze v uživatelském rozhraní.
`mkfs`	Formátování pevného disku, tj. vytvoření souborového systému
`floppy`	Formátování diskety

Zálohování a vzdálená paměť

mt	Řízení magnetopáskové mechaniky
dump	Uložení obsahu diskové oblasti na magnetickou pásku
restore	Obnovení ze zálohovacího média
tar	Čtení / zápis archívu
cdrecord	Vypálení na CD
rsync	Kopírování souborů na V/V zařízení nebo jiný počítač

Soubory lze v Linuxu zálohovat několika způsoby:

- Zkopírují se na magnetickou pásku
- Vypálí se na CD
- Uloží se na jiný počítač

Zálohovací páska připojená přes řadič IDE je obvykle na */dev/ht0*, v případě řadiče SCSI (anebo IDE řadiče s emulací ide-scsi) na */dev/st0*. Obvykle se na přílušné zařízení vytvoří odkaz:

```
$ ln -s /dev/ht0 /dev/tape
```

Bylo by zbytečné uvádět všechny zálohovací příkazy. Někdo dává přednost příkazu cpio, někdo příkazu tar. Pro fyzické kopírování disků je neocenitelným pomocníkem program dd. Popis všech těchto příkazů naleznete v příslušných manuálech.

mt [-f *zařízení*] *příkaz* — mt -st

/bin — stdin **stdout** -file –opt –help –version

Příkazem `mt` („magnetic tape") se zadávají příkazy řadiči magnetické pásky jako např. převíjení, posun vpřed a vzad a napnutí pásky. Přehled nejčastěji používaných příkazů:

`status`	Stav řadiče
`rewind`	Převinutí pásky
`retension`	Napnutí pásky
`erase`	Vymazání pásky
`offline`	Odpojení pásky
`eod`	Posun vpřed na konec dat

Například:

```
$ mt -f /dev/tape rewind
```

Pásku je také možné posouvat po souborech a po záznamech, avšak spíše se používají programy, které z pásky čtou a zapisují na ni, jako např. tar a restore.

dump [*volby*] *oblast_nebo_soubory* — dump

/sbin — stdin **stdout** -file –opt –help –version

Příkazem `dump` se zapisuje celá disková oblast nebo vybrané soubory na záložní médium, například na pásku. Provádí se zálohování jak plné, tak postupné, kdy se automaticky zjišťuje, který soubor se od minulého zálohování změnil, a tudíž musí být znovu zazálohován. Soubory se obnovují příkazem `restore`.

K úplnému zálohování souborového systému (např. */usr*) na magnetickou pásku (třeba */dev/tape*) se použijí volby `-0` (nula) a `-u`:

```
# dump -0 -u -f /dev/tape /usr
```

Takové zálohování se nazývá zálohování nulté úrovně. Volbou `-u` se do souboru */etc/dumpdates* poznačí, že byl zálohován.

Postupné zálohování může mít 1 až 9 úrovní: při i-té úrovni se ukládají všechny nové soubory a soubory, které se změnily od `i-1`, tedy od minulého zálohování.

```
# dump -1 -u -f /dev/tape /usr
```

Zálohování by se nemělo spouštět na „živém" souborovém systému: je-li to možné, měl by se nejdříve odpojit.

restore [*volby*] [*soubory*] — dump

/sbin — stdin **stdout** -file –opt –help –version

Příkazem `restore` se čte ze zálohovacího média vytvořeného příkazem `dump`. Přečtená data mohou sloužit k obnově souborů, k porovnání se soubory na disku a k řadě dalších operací. Nejjednodušší způsob použití restore je s volbou `-i` (interaktivní), který umožní prohledávat magnetickou pásku jako souborový systém, umí na ní vybírat soubory a adresáře a obnovovat je na disku.

```
# restore -i -f /dev/tape
```

Po zadání tohoto příkazu se vypíše prompt a lze zadávat příkazy k obnově, viz níže:

`help`	Výpis nápovědy
`quit`	Konec činnosti programu bez obnovy souborů
`cd` *adr*	Změna pracovního adresáře v záloze (analogie shellového příkazu)
`ls`	Výpis souborů ze zálohy (analogie linuxovského příkazu)
`pwd`	Výpis jména pracovního adresáře (analogie shellového příkazu)
`add`	Přidání souborů nebo adresářů k „obnovovacímu seznamu", tedy seznamu souborů, které se mají obnovit. Příkazem `add` bez parametru se přidá pracovní adresář se všemi soubory.

`add jm_soub`	Přidání souboru *jm_soub* k obnovovacímu seznamu.
`add adr`	Přidání adresáře *adr* k obnovovacímu seznamu.
`delete`	Opak k add: odstranění souborů nebo adresářů z obnovovacího seznamu. Příkazem `delete` bez parametru se odstraní pracovní adresář se všemi soubory.
`delete jm_soub`	Odstranění souboru *jm_soub* z obnovovacího seznamu.
`delete adr`	Odstranění adresáře *adr* z obnovovacího seznamu.
`extract`	Obnovit všechny soubory z obnovovacího seznamu. (Tip: je-li záloha uložena na více než jedné pásce, doporučuje se začít od konce a obnovovat soubory pozpátku).

Příkaz restore pracuje i v neinteraktivních režimech:

`restore -x`	Obnova celé zálohy z pásky do existujícího souborového systému. (Do něho je třeba se nejprve dostat příkazem cd).
`restore -r`	Obnova celé zálohy z pásky do čerstvě naformátovaného disku. (Do jeho kořenového adresáře je třeba se nejprve dostat příkazem cd).
`restore -t`	Výpis obsahu zálohy
`restore -C`	Porovnání zálohy a (původního) souborového systému.

tar [*volby*] [*soubory*] — tar

/bin — **stdin stdout -file -opt -help -version**

Příkaz `tar` („tape archive“) toho umí mnohem víc, než jen číst a zapisovat soubory na pásku:

```
$ tar -cf /dev/tape mujsoubor1 mujsoubor2
```

Příkaz umí vytvářet a číst *tarové soubory*, které jsou standardním nástrojem k sbalování souborů v systémex Linux a Unix:

```
$ tar -czvf mujarchiv.tar gz mujadr     Vytvoření
$ tar -tzvf mujarchiv.tar gz            Výpis obsahu
$ tar -xzvf mujarchiv.tar gz            Extrakce
```

Pokud se na příkazovém řádku uvedou soubory, zpracují se pouze ony:

```
$ tar -xvf /dev/tape soub1 soub2 soub3
```

Jinak se zpracuje celý archiv.

Užitečné volby

`-c`	Vytvoření archivu. Vstupní soubory a adresáře se zadávají na příkazovém řádku.
`-r`	Přidání souborů k existujícímu archivu.
`-u`	Přidání nových / změněných souborů k existujícímu archivu.

`-A`	Přidání archívu (tj. tarového souboru) za konec jiného archivu: tj. `tar -A -f /dev/tape mujsoub.tar`.
`-t`	Výpis archivu.
`-x`	Extrakce souboru z archivu.
`-f soub`	Čtení / zápis archivu z / do daného souboru. Může to být zařízení (např. */dev/tape*) nebo obyčejný soubor, do kterého se zapíše záloha v tarovém tvaru.
`-d`	Porovnání archivu se souborovým systémem.
`-z`	Komprimace (při zápisu) nebo rozbalení (při čtení) dat pomocí `gzip`.
`-j`	Komprimace (při zápisu) nebo rozbalení (při čtení) dat pomocí `bzip2`.
`-Z`	Komprimace (při zápisu) nebo rozbalení (při čtení) dat pomocí compress.
`-b N`	Velikost bloku *N**512 bajtů.
`-v`	„Upovídaný“ režim: výpis zpráv o všem, co probíhá.
`-h`	Vezmi v úvahu symbolické odkazy.
`-l`	Nepřekračuj hranice souborového systému.
`-p`	Obnova souborů včetně vlastníka a přístupových práv.

cdrecord [*volby*] *stopy* — cdrecord

/usr/bin — **stdin stdout -file -opt -help -version**

Příkazem `cdrecord` se vypálí CD v mechanice s řadičem SCSI nebo IDE s emulátorem ide-scsi. Obsah adresáře se vypálí na CD, které lze číst v Linuxu, Windows a na Macintoshi*:

1. Najděte mechaniku pro zápis na CD („vypalovačku“) příkazem:

```
$ cdrecord --scanbus
...
0,0,0    0) *
0,1,0    1) *
0,2,0    2) *
0,3,0    3) 'YAMAHA  '  'CRW64165  '  '1.0d'  Removable
CD-ROM
...

The device in this case is 0,3,0.
```

2. Zjistěte rychlost zápisu mechaniky (CD-R nebo CD-RW). Předpokládejme, že rychlost je 6.

* Zejména na ISO9660 s rozšířením RockRidge. `mkisoft` umí vytvářet jiné formáty pro vypalování pomocí `cdrecord`.

3. Soubory, které chcete vypálit, uložte do adresáře např. *dir*. Uspořádejte je tak, jak je chcete mít na CD. Samotný adresář *dir* se nebude kopírovat, pouze jeho obsah.
4. Vypalte CD:

```
$ DEVICE="0,3,0"
$ SPEED´6
$ mkisofs -R -l dir > mujdisk.iso
$ cdrecord -v dev=${DEVICE} speed=${SPEED} -
```

anebo, je-li používaný systém dostatečně rychlý, lze totéž udělat jedinou rourou:

```
$ mkisoft -R -l dir \
| cdrecord -v dev=${DEVICE} speed=${SPEED} -
```

`cdrecord` umí také vypalovat hudební CD, avšak existují i uživatelsky příjemnější programy s grafickým rozhraním jako např. `xcdroast` (viz „Audio a video“, str. 140), který je vystavěn nad programem cdrecord.

rsync [*volby*] *zdroj cíl* rsync

/usr/bin **stdin stdout -file –opt –help –version**

Příkazem `rsync` se kopíruje množina souborů. Lze pořídit buď přesnou kopii včetně přístupových práv a dalších vlastností (tzv. *zrcadlení*), anebo se mohou zkopírovat jen data. Může běžet na síti i na samostatném počítači. `rsync` má široké užití a více než 50 voleb; uvedeme si jen pár příkladů, jež se vztahují k zálohování.

Vytvoření zrcadla adresáře D1 a jeho obsahu v jiném adresáři (D2) na jednom počítači:

```
$ rsync -a D1 D2
```

Pořízení zrcadla adresáře D1 na síťovém počítači *server.priklad.cz*, kde má uživatel účet se jménem novak, přenos se uskuteční pomocí připojení SSH, aby se znemožnil tajný odposlech:

```
$ rsync -a -e ssh D1 novak@server.priklad.cz;
```

Užitečné volby

`-o`	Zkopírovat vlastníka souborů (na vzdálené stanici bude pravděpodobně nutné být superuživatelem).
`-g`	Zkopírovat skupinu vlastníků souborů (na vzdálené stanici bude pravděpodobně nutné být superuživatelem).
`-p`	Zkopírovat přístupová práva.
`-t`	Zkopírovat časové známky.
`-r`	Rekurzívní kopírování adresářů, tj. včetně jejich obsahu

`-l`	Povolit kopírování symbolických odkazů (pouze superuživatel).
`-D`	Povolit kopírování zařízení (pouze superuživatel).
`-a`	Zrcadlo: kopírovat všechny vlastnosti původního souboru. Tato voba zahrnuje všechny volby `-Dgloprt`
`-v`	„Upovídaný“ režim: výpis zpráv o všem, co probíhá v průběhu kopírování. Zadá-li se navíc `--progress`, graficky se znázorňuje postup kopírování („žížala“).
`-e` *příkaz*	Specifikace jiného vzdáleného shellového programu, jako např. `ssh`, z důvodu bezpečnosti.

Výpis souborů

lpr	**Výpis souboru**
lpq	**Nahlédnutí do fronty na výpis**
lprm	**Odstranění výpisu z fronty**

V Linuxu existují dva způsoby výpisu: CUPS a LPRng; ve Fedoře se používá CUPS. V obou případech se výpis zadává stejnými příkazy: `lpr`, `lpq` a `lprm`, avšak některé volby se u obou způsobů liší. Uvedeme proto jen ty voby, které jsou shodné.

Instalace tiskárny ve Fedoře se provede příkazem:

```
$ redhat-config-printer
```

a instalace se provede podle instrukcí, které se vypisují.

lpr [*volby*] [*soubory*] cups

/usr/bin **stdin stdout -file –opt –help –version**

Příkazem `lpr` se odešle soubor na tiskárnu.

```
$ lpr -P mojetisk mujsoub
```

Užitečné volby

`-P` *jméno_tisk*	Odešli soubor na tiskárnu *jméno_tisk* nastavené příkazem `redhat-config-printer`
`-#` *N*	Tisk *N* kopií souboru
`-J` *jméno*	Jméno práce („job“), které se vytiskne na titulní stranu (je-li v systému nastaven tisk titulní strany)

lpq [*volby*] cups

/usr/bin **stdin stdout -file –opt –help –version**

Vypíší se všechny práce, které čekají ve frontě na tisk.

Užitečné volby

`-P jméno_tisk`	Výpis fronty na tiskárnu *jméno_tisk*
`-a`	Výpis front na všechny tiskárny
`-l`	Výpis tiskových informací v delším „upovídaném“ tvaru

lprm [*volby*] [*ID_prací*] — cups

/usr/bin — stdin **stdout** -file -opt -help -version

Příkazem `lprm` („line printer remove“) se zruší jedna nebo více tiskových prací (např. 61 a 78):

```
$ lprm -P printername 61 78
```

Nezadá-li se žádné ID, zruší se běžná práce. (Práce jiných uživatelů může rušit pouze superuživatel). Volbou `-P` se zadává číslo fronty, která danou práci obsahuje.

Kontrola pravopisu

look	Rychlá kontrola pravopisu slova
aspell	Interaktivní kontrola pravopisu
spell	Dávková kontrola pravopisu

Linux disponuje několika zabudovanými nástroji na kontrolu pravopisu. Je-li někdo zvyklý na grafické nástroje, mohou mu ty linuxové připadat primitivní, ale zase je lze používat například v rouře, což je mocný nástroj.

look [*volby*] *předpona* [*slovník*] — util-linux

/usr/bin — stdin **stdout** -file -opt -help -version

Příkazem `look` se na standardním výstupu vypisují slova začínající zadanou *předponou*. Slova jsou umístěna ve slovníku, což je soubor (implicitně */usr/share/dict/words*). Například příkazem `look zast` se vypíše:

```
zastavět
zastavit
zastávka
zastoupit
zastrašit
zastrčit
...
atd.
```

Lze použít i vlastní slovník – libovolný abecedně setříděný soubor.

Užitečné volby

`-f`	Ignorovat horní / dolní
`-t X`	Vezmi v úvahu pouze předložku po uvedený znak *X*. Například, `look -t t zast` vypíše všechna slova začínající předponou „zas“.

aspell [*volby*] *soubor* / *příkaz* — aspell

/usr/bin — stdin stdout -file -opt -help -version

`aspell` je mocný nástroj na kontrolu pravopisu s mnoha volbami. Některé z nich si uvedeme:

`aspell -c soubor`

Interaktivní kontrola všech slov na řádku s volitelnou možností oprav.

`aspell -l < soubor`

Výpis chybných slov ze souboru na standardní výstup

`aspell dump master`

Výpis hlavního slovníku na standardní výstup

`aspell help`

Výpis nápovědy. Viz *http://aspell.net*.

spell [*soubory*] — aspell

/usr/bin — stdin stdout -file -opt -help -version

Příkazem `spell` se vypíší všechna chybná slova v daném souboru. Totéž lze provést příkazem:

```
$ cat soubory | aspell -l | sort -u
```

Nezadá-li se žádný soubor, čte se ze standardního vstupu.

Prohlížení procesů

ps	Výpis procesu
uptime	Výpis doby od spuštění systému
w	Výpis aktivních procesů všech uživatelů
top	Interaktivní monitorování procesů využívajících zdroje
xload	Grafické monitorování systému v X window
free	Výpis volné paměti

Procesem nazýváme činnost systému, kterou vykonává podle nějakého programu. Každým spuštěním programu se odstartuje jeden nebo více procesů. Procesy lze sledovat a lze s nimi nakládat. Každý proces je označen číslem, které se nazývá PID („process ID“).

Proces není totéž, co práce (angl. „job“, viz „Řízení dávek“ na str. 28): proces je součástí systému, zatímco práce je pouze záležitostí shellu, který ji spouští. Spuštěný program se skládá z jednoho nebo více procesů, zatímco práce se skládá z jednoho nebo více programů spuštěných jako shellový příkaz.

ps [*volby*]

procps

/bin stdin **stdout** -file -opt -help -version

Příkazem ps se zobrazí informace o uživatelských běžících procesech, volitelně i o procesech ostatních uživatelů.

```
$ ps
PID TTY          TIME     CMD
4706 pts/2       00:00:01 bash
15007 pts/2      00:00:00 emacs
16729 pts/2      00:00:00 ps
```

Příkaz ps má více než 80 voleb; ukážeme si jen několik nejpoužívanějších. Pokud by se některá z nich jevila jako nelogická nebo nekonzistentní, je to proto, že příkaz ps je poplatný jiným Unixům a snaží se být s nimi kompatibilní.

Zobrazení vlastních procesů:

```
$ ps -ux
```

zobrazení procesů uživatele novaka:

```
$ ps -U novak
```

zobrazení procesů vykonávaných jedním programem:

```
$ ps -C jmeno_prog
```

procesy na terminále *N*:

```
$ ps -tN
```

procesy 1, 2 a 3505:

```
$ ps -p1, 2, 3505
```

všechny procesy, jejichž příkazová řádka je přizpůsobená šířce monitoru:

```
$ ps -ef
```

všechny procesy s plnou šířkou řádku:

```
$ ps -efww
```

a všechny procesy včetně jejich „potomků", kteří jsou zobrazeni odsazeně pod rodičovskými procesy

```
$ ps -efH
```

Informace o procesech lze uspořádat, když výstup příkazu ps zpracujeme např. grepem nebo jiným filtrovacím programem.

uptime

procps

/usr/bin stdin **stdout** -file -opt -help -version

Příkazem uptime zjistíme, jak dlouho systém běží od svého spuštění.

```
$ uptime
10:54pm up 8 days, 3:44, 3 users, load average: 0.89, 1.00, 2.15
```


Význam vypsaných údajů je (zleva doprava): denní čas (22:54), činnost systému bez přerušení (8 dní, 3 hodiny, 44 minuty), počet připojených uživatelů (3) a průměrné zatížení počítače ve třech časových intervalech: 1 minuta (0.89), 5 minut (1.00) a 15 minut (2.15). Průměrným zatížením se míní počet procesů připravených ke spuštění.

w [*jméno_uživ*]

procps

/usr/bin stdin **stdout** -file **-opt** -help -version

Příkazem w se zobrazí všechny běžící procesy všech přihlášených uživatelů, přesněji všech shellů, které mají přihlášení uživatelé spuštěny.

```
$ w
 10:51pm  up 8 days, 3:42, 8 users
 load average: 0.00, 0.00, 0.00
USER     TTY      FROM     LOGIN@   IDLE   JCPU   PCPU  WHAT
bures    pts/0    :0       Sat2pm  27:13m  0.07s  0.07s emacs
plch     pts/1    host1    6Sep03   2:33m  0.74s  0.21s bash
novak    pts/1    host2    6Sep03   0.00s 13.35s  0.04s w
```

První řádek je týž, jako se vypisuje příkazem uptime. Sloupce pak označují uživatelský terminál, způsob připojení (host nebo X displej, lze-li jej připojit), čas přihlášení, délku přihlášení a dvojí údaj o využití centrální jednotky (viz příkaz w v manuálu) a jméno běžícího procesu. Tyto informace pouze o jediném uživateli zjistíme, zadáme-li jeho jméno.

Nejstručnější výpis získáme příkazem w -hfs

Užitečné volby

-h	Potlačení tisku hlavičky
-f	Netiskne se sloupec FROM
-s	Netisknou se sloupce JCPU a PCPU

top [*volby*]

procps

/usr/bin stdin **stdout** -file -opt **-help** **-version**

Příkazem top lze sledovat nejaktivnější procesy s pravidelným obnovováním (např. každou vteřinu). Příkaz je orientovaný na displej, pracuje s ním interaktivně.

```
$ top
116 processes: 104 sleeping, 1 running, 0 zombie, 11 stopped
CPU states: 1.1% users, 0.5% system, 0.0% nice, 4.5% idle
Mem: 523812K av, 502328K used, 21484K free, 0K shrd, 160436K buff
Swap: 530104K av, 0K used, 530104K free, 115300K cached
PID    USER  PRI  NI  SIZE  RSS SHARE  STAT  %CPU  %MEM  TIME  COMMAND
26265  novak  10   0  1092 1092   840  R     4.7   0.2   0:00  top
1      root    0   0   540  540   472  S     0.0   0.1   0:07  init
2      root    0   0     0    0     0  SW    0.0   0.0   0:00  kflushed
...
```

Top lze ovládat některými klávesami i v průběhu činnosti; lze změnit rychlost aktualizace (s), skrýt prázdné procesy (i) a také lze procesy zastavit (k). Volbou h se zobrazí všechny volby, q je ukončení.

Užitečné volby

-n*N*	Provede se *N* aktualizací, pak se příkaz ukončí
-d*N*	Aktualizace displeje každých *N* vteřin
-p*N* -p*M* ...	Výpis pouze procesů s PID *N*, *M*, ..., nejvýše 20 procesů
-c	Výpis parametrů procesu
-b	Neinteraktivní výpis na standardní výstup, bez grafických efektů. Příkazem top -b -n1 > vystsoub se uloží „snímek" procesu do souboru.

xload — XFree86-tools

/usr/X11R6/bin — stdin stdout -file -opt -help -version

Grafické znázornění zátěže systému v závislosti na čase v X window. Na ose X je vynesen čas, na ose Y je zatížení procesoru.

Užitečné volby

-update *N*	Aktualizace displeje každých *N* vteřin (implicitně 10)
-scale *N*	Rozdělení osy Y na *N* úseků (implicitně 1). xload může při zvyšujícím se zatížení přidat ještě další úseky; *N* je nejmenší počet trvale viditelných úseků grafu
-hl *color*	Nastavení barvy čáry oddělující úseky
-label *X*	Nad graf se vypíše text *X* (implicitně = přihlašovací jméno)
-nolabel	Nad graf se nevypíše žádný text
-jumpscroll *N*	Když graf dosáhne pravého okraje obrazovky, odroluje se o *N* pixelů vlevo a pokračuje se v kreslení grafu (implicitní hodnota je polovina šířky obrazovky)

free [*volby*] — procps

/usr/bin — stdin **stdout** -file -opt -help -version

Příkazem free se vypíše využití paměti v kilobajtech:

```
$ free
             total       used       free     shared
buffers     cached
Mem:        523812     491944      31868          0
67856       199276
-/+ buffers cache:     224812     299000
Swap:       530104          0     530104
```


Jádro systému rezervuje pro rychlou vyrovnávací paměť co největší prostor, takže v předchozím příkladu je nejlepším odhadem pro volnou RAM 299000.

Užitečné volby

-s *N*	Trvalé spuštění, obnova displeje každých *N* vteřin.
-b -m	Výpis údajů v bajtech nebo megabajtech.
-t	Na posledním řádku budou součty.
-o	Nevypíše se řádek označený buffers/cache

Řízení procesů

kill	**Ukončení procesu (anebo zaslání signálu)**
nice	**Spuštění programu s určenou prioritou**
renice	**Změna pririty procesu v době běhu**

Spuštěný proces lze zastavit, restartovat, zrušit a lze změnit jeho prioritu. O některých operacích jsme se zmínili v „Řízení dávek" na str. 28. Nyní jejich popis doplníme o zrušení procesu a změnu priority.

kill [*volby*] [*id_procesů*] — bash

součást shellu — stdin **stdout** -file -opt -help -version

Příkazem kill se pošle procesu signál. Tím se může proces ukončit (implicitně), přerušit, odložit, a proces může havarovat. Tento příkaz je účinný, jen zadá-li jej vlastník procesu nebo superuživatel.

```
$ kill 13243
```

Pokud tento příkaz nefunguje (některé procesy se neukončí, i když tento signál obdrží), je nutno zadat volbu KILL:

```
$ kill -KILL 13243
```

a příkaz se určitě provede. Je to ovšem ukončení nestandardní, nemusí se uvolnit alokované prostředky, anebo může nastat jiná nekonzistence.

Pokud neznáte PID procesu, lze je zjistit například příkazem pidof:

```
$ /sbin/pidof emacs
```

anebo můžete spustit ps a podívat se na výstup.

K programu */bin/kill* je třeba ještě doplnit, že většina shellů má v sobě zabudovaný příkaz kill, avšak jejich chování mohou být odlišná. Všechny ovšem akceptují tvary:

```
$ kill -N PID
$ kill -NAME PID
```

kde *N* je číslo signálu a *NAME* je jméno signálu bez úvodního „SIG“ (tj. SIGHUP se pošle pomocí volby -HUP). Úplný přehled signálů, jež lze zadat příkazem kill, získáme pomocí kill -l, pochopitelně odpovídající příslušné verzi. Popis signálů nalezneme v man 7 signal.

nice [*-priorita*] *příkazový řádek* — coreutils

/bin — stdin **stdout** -file -opt -help -version

Pokud někdo chce spustit proces, který spotřebovává mnoho systémových prostředků, lze se vůči ostatním procesům (a uživatelům) zachovat ohleduplně a snížit prioritu tohoto programu. Příkaz nice slouží právě k tomuto účelu. Zde je příklad nastavení „velké práce“ na prioritu 7:

```
$ nice -7 sort soub_VelkáPráce > vyst_soub
```

Pokud se neuvede priorita, nastaví se automaticky na 10. Prioritu procesu lze zjistit příkazem nice bez parametru:

```
$ nice
0
```

Superuživatel může i zvyšovat prioritu (snížením jejího čísla):

```
$ nice --10
```

(Ano, opravdu to znamená „pomlčka mínus deset“). „Úroveň nice“ uživatelových prací lze zjistit pomocí ps, je uvedena ve sloupci označeném NI:

```
$ ps -o pid,user,args,nice
```

renice *priorita* [*volby*] *PID* — util-linux

/usr/bin — stdin **stdout** -file **-opt** -help -version

Zatímco příkazem nice můžeme zadat programu prioritu při spouštění, příkazem renice lze změnit prioritu již běžícího procesu. Příklad snížení priority (zvýšení „úrovně nice“) procesu s číslem 28734 o 5:

```
$ renice +5 -p 28734
```

Uživatelé obvykle snižují prioritu svých procesů, zatímco superuživatel ji může také zvyšovat. Platný rozsah je od -2 do 20, avšak záporné priority s vysokou absolutní hodnotou je nutno zadávat uvážlivě, mohly by totiž kolidovat se sytémovými procesy.

Užitečné volby

-p *pid*	Zadání ID procesu. Lze vynechat -p a zadat přímo PID (renice +5 28734)
-u *uživ_jm*	Příkaz se vztahuje na všechny procesy vlastněné uživatelem.

Uživatelé a jejich okolí

logname	Výpis přihlašovacího jména
whoami	Výpis běžného jména uživatele
id	Výpis uživatelského ID a příslušnost ke skupině
who	Výpis přihlášených uživatelů, dlouhý tvar
users	Výpis přihlášených uživatelů, krátký tvar
finger	Výpis informací o uživateli
last	Zjisti, kdo a kdy se přihlásil naposledy
printenv	Výpis okolí

Kdo jste? Bezpečně to ví jenom systém. Uvedená všehochuť programů vypoví o uživateli vše: jméno, čas přihlášení a vlastnosti okolí.

logname — coreutils

/usr/bin — stdin **stdout** -file -opt **-help** **-version**

Příkazem logname se vypíše přihlašovací jméno uživatele. Funkce vypadá triviálně, avšak využívá se v shellových skriptech.

```
$ logname
novak
```

whoami — coreutils

/usr/bin — stdin **stdout** -file -opt **-help** **-version**

Příkazem whoami se vypisuje jméno běžného uživatele. Může se od přihlašovacího jména (výstup logname) lišit, pokud použil příkaz su. Z následujícího příkladu je rozdíl mezi příkazy whoami a logname zřejmý:

```
$ logname
novak
$ whoami
novak

$ su
Password: ********
# logname
novak
# whoami
root
```

id [*volby*] [*uživ_jm*] — coreutils

/usr/bin — stdin **stdout** -file -opt -help -version

Každý uživatel má jedinečné *uživatelské číslo* UID a patří do skupiny, která má *číslo skupiny* GID. Příkazem id se tyto hodnoty vypíší společně s jejich jmény (uživatele a skupiny):

```
$ id
uid=500(novak) gid=500(novak)
groups=500(novak),6(disk),490(src),501(cdwrite)
```

Užitečné volby

`-u`	Výpis pouze UID
`-g`	Výpis pouze GID
`-G`	Výpis GID všech skupin, do nichž uživatel patří.
`-n`	Výpis pouze jména uživatele a skupiny, nikoli číselné ID. Užívá se v kombinaci s `-u`, `-g` nebo `-G`. Například `id -Gn` je totéž jako příkaz `groups`.
`-r`	Výpis skutečných hodnot místo hodnot okamžitých. Užívá se v kombinaci s `-u` a `-g`.

who [*volby*] [*jm_soub*] — coreutils

/usr/bin — stdin **stdout** -file --opt **--help** **--version**

Příkazem `who` se vypíší jména všech přihlášených uživatelů, po jednom na řádek:

```
$ who
novak  :0        Sep 6 17:09
plch             pts/1    Sep 6 17:10
bures  pts/2     Sep 8 20:58
bures  pts/4     Sep 3 05:11
```

Normálně získává who data ze souboru */var/run/utmp*. Parametrem *jm_soub* lze zadat jiný soubor, např. */var/ log/wtmp* pro dříve přihlášené nebo */var/log/btmp* pro chybně přihlášené*.

Užitečné volby

`-H`	Výpis hlavičky
`-l`	Při dálkovém připojení výpis jména účastníka
`-u`	Výpis délky doby běhu naprázdno na účastníkově terminálu.
`-T`	Indikace psaní na uživ. terminál (viz `mesg y` v „Okamžité posílání zpráv“ na str. 123). Znaménko plus znamená ano, znaménko mínus znamená ne, otazník znamená není známo.
`-m`	Výpis pouze informací týkajících se běžného terminálu
`-q`	Rychlý výpis pouze uživatelských jmen. Podobné příkazu `users`, navíc je uveden počet.

* Pokud je systém nakonfigurován tak, že se tyto příkazy smějí zadávat.

users [*jm_soub*] — coreutils

/usr/bin — stdin **stdout** -file --opt **--help** **--version**

Příkazem `users` se vypíše rychlý seznam přihlášených uživatelů. Má-li některý uživatel spuštěn, několik shellů, vypíší se všechny.

```
$ users
novak plch bures bures bures
```

Podobně, jako v případě who, users se bere implicitně z adresáře */var/log/utmp*, ale lze zadat jakýkoli jiný adresář.

finger [*volby*] [*uživ[@host]*]* — finger

/usr/bin — stdin **stdout** -file **--opt** **--help** **--version**

Příkazem `finger` se vypisují informace o uživatelích buď ve zkráceném tvaru:

```
Login     Name            Tty      Idle     Login Time
novak     Franta Novák    :0                Sep 6 17:09
plch      Jiří Plch       pts/1    24       Sep 6 17:10
bures     Pavel Bureš     pts/2              Sep 8 20:58
```

anebo v dlouhém tvaru:

```
$ finger novak
Login: novak              Name: Franta Novák
Directory: /home/novak    Shell: /bin/bash
On since Sat Sep 6 7:09 (EDT) on: 0
Last login Mon Sep 8 21:07 (EDT) on pts/6 from localhost
No mail
Project:
Boj za mír
Plan:
Nedůvěřuj prvnímu dojmu; je vždy pravdivý.
```

Parametrem příkazu `user` může být lokální uživatel nebo vzdálený uživatel ve tvaru *uživ@host*. Informace o vzdáleném uživateli se poskytnou pouze tehdy, je-li to povoleno v konfiguraci.

Užitečné volby

`-l`	Výpis v dlouhém tvaru
`-s`	Výpis v krátkém tvaru
`-p`	Nevypisuje se část Project a Plan, které se obvykle čtou z uživatelských souborů ~/.project a ~/.plan.

* Překladatelská poznámka: Anglický termín „host“ znamenající (hostitelský) počítač budeme z důvodů zkrácení používat zejména v parametrech příkazů.

last [*volby*] [*uživ*] [*ttys*]

SysVinit

/usr/bin stdin **stdout** -file **–opt** –help –version

Příkazem `last` se vypíše průběh přihlášování od nejpozději přihlášeného k nejdříve přihlášenému.

```
$ last
plch      pts/3    localhost    Sep 8 21:07 - 21:08 (00:01)
novak     pts/6    :0           Sep 8 20:25 - 20:56 (00:31)
plch      pts/4    mujhost      Sep 7 22:19 still logged in
...
```

Výstupní seznam lze zkrátit, když se zadají konkrétní jména nebo terminály.

Užitečné volby

`-N`	Výpis jen posledních N řádků výstupu, kde *N* je celé kladné.
`-i`	Výpis IP adres místo jmen
`-R`	Nevypisují se jména
`-x`	Vypisují se i zprávy o ukončení činnosti systému a změnách v režimu systému (tj. jednouživatelský a víceuživatelský).
`-f jm_soub`	Čte se z jiného souboru než */var/run/utmp*, podrobnosti viz popis příkazu who.

printenv [*vnější_proměnné*]

coreutils

/usr/bin stdin **stdout** -file **–opt** –help –version

Příkazem `printenv` se vypíší všechny vnější proměnné známé shellu a jejich hodnoty:

```
$ printenv
HOME=/home/novak
MAIL=/var/spool/mail/novak
NAME=Franta Novák
SHELL=/bin/bash
...
```

anebo jen požadované proměnné:

```
$ printenv HOME SHELL
/home/novak
/bin/bash
```

Práce s uživatelskými účty

useradd	Vytvoření nového účtu
userdel	Zrušení účtu
usermod	Změna účtu
passwd	Změna hesla
chfn	Změna uživ. osobních údajů
chsh	Změna uživ. shellu

Instalační program Fedory nabídne vytvoření dvou účtů, jeden pro superuživatele, druhý běžný uživatelský. Současně lze vytvořit i další účty.

Vytvoření uživatelských účtů je důležitá práce a neměla by být brána na lehkou váhu. Každý účet je potenciální cestou k proniknutí vetřelce do systému. Uživatelská hesla by měla být nesnadno odhadnutelná a měla by se pravidelně měnit.

useradd [*volby*] *uživ_jméno*

shadow-utils

/usr/bin stdin **stdout** -file **–opt** –help –version

Příkazem `useradd` přidává superuživatel nový uživatelský účet.

```
# useradd novak
```

V tomto příkazu se implicitní proměnné obvykle nepoužívají, (jedině `useradd -D` pro výpis účtů), takže byste si měli být jisti, že jste zadali všechny požadované volby. Například:

```
# useradd -d /home/novak -s /bin/bash -g users novak
```

Užitečné volby

`-d adr`	Nastavení domovského adresáře na *adr*
`-s shell`	Nastavení uživatelského shellu na *shell*
`-u uid`	Nastavení UID na *uid*. Dokud si nebudete jisti, jak se tento příkaz používá, je vhodné přijmout implicitní hodnotu.
`-g skup`	Úvodní nastavení uživatelské skupiny na *skup*, což může být buď číselný údaj, anebo jméno skupiny. Skupina musí existovat.
`-G skup1, skup2, ...`	Přidej uživatele do dalších existujících skupin *skup1*, *skup2* atd.
`-m`	Zkopírují se všechny soubory „kostry“ (skeletonového adresáře, */etc/skel*) do nově vytvořeného domovského adresáře. Kostra tradičně obsahuje minimální verze inicializačních souborů jako *~/.bash_profile* pro nové uživatele. Je-li žádoucí, aby se kostra nového uživatele zkopírovala do samostatného adresáře, zadá se volba `-k` (`-k váš_adresář`).

userdel [*-r*] *uživ_jméno*

shadow-utils

/usr/bin stdin **stdout** -file **–opt** –help –version

Příkazem `userdel` se zruší existující uživatel.

```
# userdel novak
```

Domovský adresář se ponechá, pokud není zadána volba `-r`. Rušení uživatelského účtu je nutno si důkladně promyslet a zvážit, zda by nebylo vhodnější jej pouze deaktivovat (příkazem `usermod -L`). Také je nutno se přesvědčit, zda jsou zazálohovány všechny uživatelské soubory, chceme-li je vymazat; nevíme, kdy je budeme potřebovat.

usermod [*volby*] *uživ_jméno* — shadow-utils

/usr/bin — stdin **stdout** -file **-opt** **-help** **-version**

Příkazem `usermod` se provádějí modifikace uživatelského účtu, např. změna domovského adresáře apod.:

```
# usermod -d /home/jiný novak
```

Užitečné volby

`-d` *adr*	Změna domovského adresáře uživatele na *adr*.
`-l` *uživ_jm*	Změna přihlašovacího jména na *uživ_jm*. Při této volbě je třeba důkladně zvážit, zda neexistuje jakákoli závislost systému na původním jménu. A zcela určitě by parametrem neměl být žádný systémový účet (root, daemon atd.).
`-s` *shell*	Změna přihlašovacího shellu na *shell*.
`-g` *skup*	Změna původní (implicitní) skupiny na *skup*. Lze ji zadat buď číselně (GID) nebo jménem a musí existovat.
`-G` *skup1, skup2, …*	Zařazení uživatele *pouze* do uvedených skupin *skup1*, *skup2*, …. Pokud předtím patřil do jiných, zde neuvedených skupin, příslušnost k nim bude zrušena.
`-L`	Deaktivace účtu, uživatel se nebude moci přihlásit.
`-U`	Aktivace účtu deaktivovaného volbou `-L`.

passwd [*volby*] [*uživ_jméno*] — shadow-utils

/usr/bin — stdin **stdout** -file **-opt** **-help** **-version**

Příkazem `passwd` se změní přihlašovací heslo, implicitně uživatelské:

```
# passwd
```

anebo heslo jiného uživatele:

```
# passwd novak
```

Příkaz `passwd` má několik voleb, většina z nich se vztahuje k expirační době. Používá se pouze v souvislosti s bezpečnostní politikou.

chfn [*volby*] [*uživ_jméno*] — shadow-utils

/usr/bin — stdin **stdout** -file **-opt** **-help** **-version**

Příkazem `chfn` („change finger“) se aktualizují osobní údaje vedené o uživateli v systému: skutečné jméno, telefon domů a do zaměstnání, adresa pracoviště, tedy údaje poskytované příkazem `finger`. Nezadá-li se parametr *uživ_jméno*, příkaz se vztahuje k osobním údajům uživatele, který příkaz zadal; s uvedením tohoto parametru (což může pouze root) pak k údajům daného uživatele. Nezadá-li se žádný parametr, příkaz `chfn` si nové údaje vyžádá sám:

```
$ chfn
Password: ********
Name [Franta Novák]: František Novák
Office [Dulánek 12]:
Office Phone [05-48 21 11 21 ]: 548 21 11 21
Home Phone []:
```

Užitečné volby

`-f` *jméno*	Změna celého jména na *jméno*.
`-h` *telefon*	Změna telefonu domů na *telefon*.
`-p` *telefon*	Změna telefonu do kanceláře na *telefon*.
`-o` *kancelář*	Změna adresy *kanceláře*.

chsh [*volby*] [*uživ_jméno*] — util-linux

/usr/bin — stdin **stdout** -file **-opt** **-help** **-version**

Příkazem `csh` („change shell“) se spustí přihlášení programem `shell`. Neuvede-li se *uživ_jméno*, vztahuje se příkaz k danému účtu; s parametrem *uživ_jméno* (zadat jej může jen root) se vztahuje k danému jménu. Příkaz bez parametru si vyžádá přihlašovací údaje sám.

```
$ chsh
Changing shell for novak
Password: ********
New shell [/bin/bash]: /bin/tcsh
```

Nový shell musí být uložen v */etc/shells*.

Užitečné volby

`-s` *shell*	Specifikace nového shellu.
`-l`	Výpis všech povolených shellů.

Superuživatel

Normální uživatel může většinou měnit pouze svoje soubory. Tzv. *superuživatel* neboli *root* má neomezený přístup k celému počítači a může na něm provádět naprosto vše. Superuživatel se nejdříve přihlásí jako normální uživatel, a pak zadá:

```
$ su -l
Password: ********
#
```

Musí se zadat heslo superuživatele a jako prompt se vypíše dvojitý křížek (#), čímž se potvrdí správné přihlášení. Svoji činnost superuživatel ukončí buď povelem `^D` nebo příkazem `exit`, poté se z něho opět stane normální uživatel.

Toto je nejjednodušší způsob, jak získat privilegovaná oprávnění k systému náležející superuživateli. Existují i jiné programy, jimiž se vykonávají funkce superuživatele, avšak spadají mimo rozsah této knihy.

Jestliže se v příkazu `su` zadá uživatelské jméno:

```
$ su -l plch
Password: ********
```

lze se stát oním zadaným uživatelem, pochopitelně známe-li heslo.

Užitečné volby

`-l`	Spustí se přihlášení do shellu.
`-m`	Uchování vnějších proměnných v novém shellu.
`-c` *příkaz*	V režimu jiného uživatele se provede pouze zadaný *příkaz* a skončí se. Pokud tuto volbu potřebujete často, viz příkaz sudo v manuálu.
`-s` *shell*	Spuštění shellu (*/bin/bash*).

Skupiny

groups	**Výpis uživatelovy příslušnosti ke skupině**
groupadd	**Vytvoření nové skupiny**
groupdel	**Zrušení skupiny**
groupmod	**Modifikace skupiny**

Skupina je množina uživatelských účtů, na niž se z hlediska systému pohlíží jako na jeden objekt, nad nímž lze provádět hromadné operace (např. povolit změny v souboru) tak, že platí pro všechny příslušníky skupiny. Například skupině *přátelé* můžeme dát povolení ke čtení ze souboru, zapisování do souboru a spouštění souboru */tmp/příklad*:

```
$ groups
users novak přátelé
$ chgrp přátelé /tmp/příklad
```



```
$ chmod 770 /tmp/příklad
$ ls -l /tmp/příklad
-rwxrwx--- 1 novak přátelé 2874 Oct 20 22:35 /tmp/příklad
```

Uživatelé se přidávají do skupiny tak, že se edituje soubor /etc/group*, a to s přístupovými právy superuživatele. Změna vlastnictví skupiny k souboru se provede příkazem chgrp, viz „Atributy souborů“ na str. 50.

groups [*uživ_jména*] — coreutils

/usr/bin — **stdin** **stdout** -file **–opt** **–help** **–version**

Příkazem `groups` se vypisují skupiny, do nichž patří daný uživatel nebo jiní uživatelé:

```
$ whoami
novak
$ groups
novak uživatelé
$ groups plch root
plch : plch uživatelé
root : root bin daemon sys adm disk wheel src
```

groupadd [*volby*] *skupina* — shadow-utils

/usr/bin — **stdin** **stdout** -file **–opt** **–help** **–version**

Příkazem `groupadd` se vytvoří nová skupina. Příkaz se většinou zadává s volbou `-f`, aby nedošlo k duplicitě:

```
# groupadd -f přátelé
```

Užitečné volby

`-g` *gid*	Zadání vlastního čísla skupiny (GID) – nepřiřazuje automaticky.
`-f`	Pokud už uvedená skupina existuje, pouze se to ohlásí a příkaz se ukončí.

groupdel *skupina* — shadow-utils

/usr/bin — **stdin** **stdout** -file **–opt** **–help** **–version**

Příkazem `groupadd` se zruší existující skupina.

```
# groupdel -f přátelé
```

Před zadáním tohoto příkazu je dobré zjistit všechny soubory, jež patří do zadané skupiny, aby se s nimi dalo pracovat i po zrušení skupiny:

```
# find / -group přátelé -print
```

protože příkazem `groupdel` se nezmění příslušnost souboru ke skupině. Pouze se odstraní daná skupina ze systému.

* Jiné systémy mohou ukládat seznamy členů skupin jinak.

groupmod [*volby*] *skupina* — shadow-utils

/usr/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `groupmod` se dané skupině změní jméno nebo GID.

```
# groupmod -n nové_jméno přátelé
```

Příkaz nemá žádný vliv na soubory patřící k dané skupině: prostě se změní GID nebo jméno skupiny v systému. Se změnou GID je nutno zacházet opatrně, mohlo by se totiž stát, že soubory bude vlastnit neexistující skupina.

Užitečné volby

`-g` *gid*	Změna čísla skupiny na *gid*.
`-n` *jméno*	Změna jména skupiny na *jméno*.

Základní informace o počítači

uname	**Výpis základních informací o systému**
hostname	**Výpis jména počítače**
dnsdomainname	**Totéž, co `hostname -d`**
domainname	**Totéž, co `hostname -y`**
nisdomainname	**Totéž, co `hostname -y`**
ypdomainname	**Totéž, co `hostname -y`**
ifconfig	**Nastavení a zobrazení informací o rozhraní sítě**

Každý linuxový počítač (neboli hostitelský počítač) má jméno, síťovou adresu (IP) a další vlastnosti. Ukážeme si, jak je lze zobrazit.

uname [*volby*] — coreutils

/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `uname` se vypíší základní informace o počítači:

```
$ uname -a
Linux server.example.com 2.4.18-27.8.0 #1 Fri Mar 14 06:45:49 EST 2003
i686 i686 i386 GNU/Linux
```

Výpis obsahuje jméno jádra (Linux), jméno počítače (server.example.com), verzi jádra (2.4.18-27.8.0 #1 Fri Mar 14 06:45:49 EST 2003), jméno hardwaru (i686), typ procesoru (i686), hardwarovou platformu (i386) a jméno operačního systému (GNU/Linux).

Užitečné volby

`-a`	Všechny informace.
`-s`	Pouze jméno jádra (implicitně).
`-n`	Pouze jméno počítače.
`-r`	Pouze verzi jádra.
`-m`	Pouze jméno hardwaru.

`-p`	Pouze typ procesoru.
`-i`	Pouze hardwarovou platformu.
`-o`	Pouze jméno operačního systému.

hostname [*volby*] [*jméno*] — net-tools

/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `hostname` se vypíše jméno počítače. Může být shodné s plným jménem hostitelského počítače, záleží na konkrétním nastavení:

```
$ hostname
myhost.example.com
```

anebo krátké jméno hostitelského počítače:

```
$ hostname
myhost
```

Superuživatel může nastavit jméno počítače:

```
# hostname pomeranč
```

Ovšem jména hostitelských počítačů jsou složité téma a v knize tohoto rozsahu pro ně není dostatek místa. Ctěnému běžnému uživateli si dovolíme poradit, aby raději počítačům přidělovat jména nezkoušel.

Užitečné volby

`-i`	IP adresa počítače
`-a`	Alias počítače
`-s`	Krátké jméno počítače
`-f`	Celé jméno počítače
`-d`	Jméno DNS domény
`-y`	Jméno NIS nebo YP domény
`-F` *hostfile*	Nastavení jména počítače dle souboru *hostfile*.

ifconfig *rozhraní* — net-tools

/sbin — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `ifconfig` se zobrazují a nastavují vlastnosti síťového rozhraní. Podrobnější popis přesahuje možnosti této knihy, pouze si ukážeme několik triků.

Informace o implicitním síťovém rozhraní (nazývaném *eth0*) si zobrazíme takto:

```
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:50:BA:48:4F:BA
inet    addr: 192.168.0.10  Bcast: 192.168.0.255  Mask: 255.255.255.0
```

```
UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
RX packets:824 errors:0 dropped:0 overruns:0 frame:0
TX packets:535 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:122801 (119.9 KiB)  TX bytes:67963 (66.3 KiB)
Interrupt:5 Base address:0x2000
```

Výpis obsahuje MAC adresu: 00:50:BA:48:4F:BA, IP adresu (192.168.0.10), síťovou masku (255.255.255.0) a řadu dalších informací. Všechna síťová rozhraní se zobrazí pomocí:

```
$ ifconfig -a
```

Další informace o práci se sítí jsou v manuálu, příkaz `ifconfig`.

Umístění počítače

host	**Hledání jmen, IP adres a informací o DNS**
whois	**Hledání registrace internetových domén**
ping	**Zjisti dosažitelnost vzdáleného počítače**
traceroute	**Najdi cestu po síti ke vzdálenému počítači**

Někdy je potřeba získat více informací o vzdálených počítačích. Kdo je vlastníkem? Jakou mají IP adresu? Kde se nacházejí?

host [*volby*] *jméno* [*server*] — bind-utils

/usr/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `host` se pomocí dotazu na DNS hledá jméno vzdáleného (hostitelského) počítače nebo jeho IP adresa.

```
$ host www.redhat.com
www.redhat.com has address 66.187.232.50
$ host 66.187.232.50
50.232.187.66.in-adr.arpa domain name pointer www.redhat.com
```

Lze nalézt i mnohem víc:

```
$ host -a www.redhat.com
Trying "www.redhat.com"
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 50419
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.redhat.com.      IN ANY

;; ANSWER SECTION:
www.redhat.com.          196   IN   A    66.187.232.50

;; AUTHORITY SECTION:
redhat.com.           90535   IN   NS   ns2.redhat.com.
```



```
redhat.com.           90535   IN   NS   ns3.redhat.com.
redhat.com.           90535   IN   NS   ns1.redhat.com.

;; ADDITIONAL SECTION
ns2.redhat.com.       143358  IN   A    66.187.224.210
ns3.redhat.com.       143358  IN   A    66.187.229.10
ns1.redhat.com.       143358  IN   A    66.187.233.210
```

Ovšem diskuse na téma nameserverů by byla hodně rozsáhlá a v naší knize pro ni není dostatek místa. „Serverovými" parametry můžete vybrat určitý nameserver. Zde je příklad dotazu na *comcast.net*.

```
$ host www.redhat.com ns01.jdc01.pa.comcast.net
Using domain server:
Name: ns01.jdc01.pa.comcast.net
Address: 66.45.25.71#53
Aliases:
www.redhat.com has address 66.187.232.50
```

Všechny možnosti se vypíší zadáním příkazu host.

Užitečné volby

`-a` Výpis všech dostupných informací

`-t` Výběr typu dotazu:`A`, `AXFR`, `CNAME`, `HINFO`, `KEY`, `MX`, `NS`, `PTR`, `SIG`, `SOA` atd.

```
$ host -t MX redhat.com
redhat.com mail is handled by 20 mx2.redhat.com
redhat.com mail is handled by 10 mx1.redhat.com
```

Pokud příkaz `host` nepracuje podle vašich představ, zkuste příkaz `dig`, což je další mocný nástroj na dotazy na DNS. Taktéž se lze setkat s příkazem `nsloo-kup`, dnes už poněkud demódé, avšak některé Linuxy a Unixy jej dosud používají.

whois [*volby*] *jméno_domény* — jwhois

/usr/bin — stdin **stdout** -file **–opt** **–help** **–version**

Příkazem `whois` se hledá registrace internetové domény:

```
$ whois redhat.com
Registrant:
Red Hat, Inc. (REDHAT-DOM)
  P.O. Box 13588
  Research Triangle Park, NC 27709
...
```

Počítejte ovšem s tím, že před nebo za požadovanou informací na vás vyskočí několik obrazovek různých právnických textů.

Užitečné volby

`-h registrant`	Nahlédnutí na registrovaný server. Např.: whois.networksolutions.com yahoo.com.
`-p port`	Dotaz na určitý TCP port (nikoli implicitní 43, kde je služba `whois`)

ping [*volby*] *host*

iputils

/bin **stdin stdout -file -opt -help -version**

Příkazem `ping` se zjišťuje dostupnost vzdáleného počítače. Na vzdálený počítač se pošlou malé pakety (přesně řečeno pakety ICMP) a čeká se na odpověď.

```
$ ping google.com
PING google.com (216.239.37.100) from 192.168.0.10 :
56(84) bytes OF data.
64 bytes from www.google.com (216.239.37.100): icmp_seq=0
ttl=49 time=32.390 msec
64 bytes from www.google.com (216.239.37.100): icmp_seq=1
ttl=49 time=24.208 msec
^C
--- google.com ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/mdev = 24.208/28.299/32.390/4.091 ms
```

Užitečné volby

`-c N`	Ping nejvýše *N* krát.
`-i N`	Mezi pingy čekej *N* vteřin (implicitně 1)
`-n`	Výpis IP adres počítačů, nikoli jmen.

traceroute [*volby*] *host* [*délka_paketu*]

traceroute

/bin **stdin stdout -file -opt -help -version**

Příkazem `traceroute` se vypíše cesta po síti z lokálního počítače ke vzdálenému počítači a doba přenosu paketu po této cestě.

```
$ traceroute yahoo.com
 1 server.example.com (192.168.0.20) 1.397 ms 1.973 ms 2.817 ms
 2 10.221.16.1 (10.221.16.1) 15.397 ms 15.973 ms 10.817 ms
 3 gbr2-p10.cb1ma.ip.att.net (12.123.40.190) 11.952 ms 11.720 ms
11.705 ms
 ...
 16 p6.www.dcn.yahoo.com (216.109.118.69) 24.757 ms 22.659 ms *
```

Každému počítači se pošlou tři „zkoušky" a oznámí se doba odpovědi. Pokud vzdálený počítač neodpoví do 5 vteřin, vypíše se hvězdička. Cesta ovšem může být blokována firewallem anebo není schopna zpracovat odpověď z jiných důvodů, v těchto případech se vypíší určité znaky.

Znaky	Význam
`!F`	Požaduje se fragmentace.
`!H`	Nedostupný počítač.
`!N`	Nedospupná síť.
`!P`	Nedostupný protokol.
`!S`	Selhání cesty.
`!X`	Komunikace není povolena.
`!N`	Neexistující kód N v ICMP.

Implicitní velikost paketu je 40 bajtů; ale lze ji změnit volitelným parametrem *délka_paketu* (např. `traceroute můjhost 120`).

Užitečné volby

`-n`	Numerický režim: vypisuje se IP adresa, nikoli jméno.
`-w N`	Změna časového limitu z 5 na *N* vteřin.

Síťová připojení

ssh	**Bezpečné připojení ke vzdálenému počítači anebo spuštění příslušných příkazů**
telnet	**Připojení ke vzdálenému počítači (nikoli bezpečné!)**
scp	**Bezpečné kopírování souborů do/ze vzdáleného počítače (dávkou)**
sftp	**Bezpečné kopírování souborů do/ze vzdáleného počítače (interaktivně)**
ftp	**Kopírování souborů do/ze vzdáleného počítače (interaktivně, nikoli bezpečně)**

Zřídit spojení mezi dvěma počítači pro vzdálená přihlášení a pro přenos souborů je v Linuxu velice jednoduché. Je ovšem třeba dbát na bezpečnost takového připojení.

ssh [*volby*] *host* [*příkaz*]

openssh-clients

/usr/bin **stdin stdout -file -opt -help -version**

Příkazem `ssh` („Secure Shell") se uživatel bezpečně přihlásí k existujícímu účtu na vzdáleném počítači:

```
$ ssh vzdaleny.priklad.cz
```

Anebo lze vzdálenému počítači zadat příkaz, aniž bychom byli přihlášeni:

```
$ ssh vzdaleny.priklad.cz who
```

Příkaz `ssh` dešifruje příchozí data ze vzdáleného počítače včetně uživatelského jména a hesla (které se musí při přihlašování na vzdálený počítač zadat). Protokol SSH podporuje i jiné způsoby autentizace, jako např. veřejné klíče a ID počítače. Podrobnosti k příkazu `ssh` jsou uvedeny v manuálu.

Užitečné volby

`-l uživ_jm` Buď se *uživ_jm* explicitně uvede, nebo se použije lokální uživatelské jméno. Taktéž je možno použít syntaxe *uživ_jm@host*:

```
$ ssh novak@server.priklad.cz
```

`-p port` Zadání jiného než implicitního (22) portu.

`-t` Alokace tty na vzdáleném systému; vhodný příkaz pro spuštění vzdáleného příkazu s interaktivním rozhraním, např. textový editor.

`-v` „Upovídaný" výstup, je užitečný pro ladění.

telnet [*volby*] *host* [*port*] — telnet

/usr/bin — stdin **stdout** -file -opt -help -version

Příkaz `telnet` slouží k přihlášení k existujícímu účtu na vzdálenému počítači.

```
$ telnet vzdaleny.priklad.cz
```

Používání tohoto příkazu se nedoporučuje, protože ve většině systémů je implementován jako nezabezpečené přihlášení, kterým se přenáší heslo jako obyčejný text v nezašifrované podobě a každý si ho může ukrást. Lépe je přihlašovat se příkazem `ssh`, který přenáší hesla i data zašifrovaná. Existují dvě výjimky:

- V kerberovském okolí se na obou stranách spojení použije rozšířeného („kerberovského") softwaru. Fedorový telnet umí pracovat s Kerberosem. Viz také *http://web.mit.edu/kerberos/*.
- Připojení k vzdálenému portu, pokud se neposílají citlivé informace. Např. se zjišťuje přítomnost webového serveru (port 80) na vzdáleném systému:

```
$ telnet vzdaleny.priklad.cz 80
Trying 192.168.55.21...
Connected to vzdaleny.priklad.cz (192.168.55.21).
Escape character is '^]'.
xxx                          Napiš nějakou blbost a stiskni enter.
<HTML><HEAD>  # Yep, it's a web server
<TITLE>400 Bad request</TITLE>
</HEAD<BODY>
<H1>Bad Request</H1>
Your browser sent a request that this server could not
understand.<P>
</BODY></HTML>
Connecting closed by foreign host.
```

Volby `telnet` raději ani nebudeme popisovat, aby náhodou někoho nenapadlo tento příkaz používat.

scp *lok_spec vzdal_spec* — openssh-clients

/usr/bin — stdin **stdout** -file -opt **-help** **-version**

Příkazem `scp` („secure copy") se dávkovým způsobem kopírují soubory a adresáře z jednoho počítače na druhý. (K interaktivnímu kopírování slouží příkaz `sftp`). Veškerá komunikace mezi dvěma počítači probíhá zašifrovaně.

```
$ scp mujsoubor vzdaleny.priklad.cz:newfile
$ scp -r mujadr vzdaleny.priklad.cz:
$ scp vzdaleny.priklad.cz:mujsoubor.
$ scp -r vzdaleny.priklad.cz:mujadr.
```

Alternativní uživatelské jméno na vzdáleném systému se zadá pomocí syntaxe *username@host*:

```
$ scp mujsoubor novak@vzdaleny.priklad.cz:
```

Užitečné volby

`-p` Kopírování souboru se všemi vlastnostmi (povolení, časová známka).

`-r` Rekurzívní kopírování adresáře a jeho obsahu.

`-v` „Upovídaný" výstup, vhodný při ladění.

sftp (*host* | *uživ_jm@host*) — openssh-clients

/usr/bin — stdin **stdout** -file **-opt** **-help** **-version**

Programem `sftp` se interaktivně kopírují soubory mezi dvěma počítači. (Na rozdíl od `scp`, kterým se soubory kopírují v dávce). Uživatelské rozhraní je velmi podobné příkazu `ftp`.

```
$ sftp vzdaleny.priklad.cz
Password: ********
sftp> cd mojesoubory
sftp> ls
README
soub1
soub2
soub3
sftp> get soub2
Fetching /home/novak/mojesoubory/soub2 to soub2
sftp> quit
```

Pokud se uživatelské jméno na vzdáleném systému liší od jména lokálního, zadá se parametr *uživ_jm@host*:

```
$ sftp novak@vzdaleny.priklad.cz
```

Příkaz	Význam
`help`	Výpis všech příkazů
`ls` `lls`	Výpis souborů v běžném vzdáleném (`ls`) nebo lokálním (`lls`) adresáři
`pwd` `lpwd`	Výpis vzdáleného (`pwd`) nebo lokálního (`lpwd`) pracovního adresáře.
`cd` *adr* `lcd` *adr*	Změna vzdáleného (`cd`) nebo lokálního (`lcd`) adresáře na *adr*.
`get` *soub1* [*soub2*]	Kopírování vzdáleného souboru *soub1* na lokální počítač, s volitelným přejmenováním na *soub2*.
`put` *soub1* [*soub2*]	Kopírování lokálního souboru *soub1* na vzdálený počítač, s volitelným přejmenováním na *soub2*.
`mget` *soub**	Kopírování více vzdálených souborů na lokální počítač s využitím zástupných znaků * a ?.
`mput` *soub**	Kopírování více lokálních souborů na vzdálený počítač s využitím zástupných znaků * a ?.
`quit`	Konec

ftp [*volby*] *host* ftp

/usr/bin **stdin stdout -file -opt -help -version**

Programem `ftp` („File Transfer Protocol") se nezabezpečeným způsobem kopírují soubory mezi počítači. Uživatelské jméno a heslo se po síti přenášejí nezašifrované. Je-li to jen trochu možné, měl by se vždy raději používat příkaz `sftp`.

Stejné příkazy, jež byly uvedeny u `sftp`, platí i pro `ftp`. Existuje nicméně několik příkazů, v nichž se neshodují.

Elektronická pošta

evolution	**Emailový klient s grafickým rozhraním (GUI)**
mutt	**Emailový klient orientovaný na texty**
mail	**Emailový klient minimálně orientovaný na texty**

Ve Fedoře je celá řada programů pro elektronickou poštu. Uvedeme si tři z nich, jež jsou určeny pro různé účely. Jiné mailovací programy obsahují `pine`, `RMAIL` a `vm` aplikace, jež jsou součástí `emacs` a `mozillového` Mail & News.

Průběh přenosu odchozích a příchozích zpráv lze sledovat v souboru */var/log/maillog*. Root může pro sledování odchozích zpráv, které čekají ve frontě na odeslání, využít služeb příkazu mailq.

evolution evolution

/usr/bin **stdin stdout -file -opt -help -version**

Ximianový program Evolution je grafický program, který vypadá jako Microsoft Outlook. V závislosti na nastavení systému je možno spustit Evolution buď z hlavního menu jako `internet : Evolution Email`, anebo jako normální příkaz shellu.

Nastavení účtu pro elektronickou poštu:

1. Vyberte Tools –> Settings...
2. V okně Settings, pokud jste ještě nastavení neprohlíželi, vyberte Add. Jinak vyberte příslušný účet a Edit.
3. V okně Account Editor vyberte tabulku Identity, vyplňte celé jméno a elektronickou adresu.
4. Vyberte tabulku Receiving Mail a typ serveru (IMAP, POP, lokální doručování atd.) a vyplňte příslušná pole vztahující se k vašemu serveru. V případě serverů POP nebo IMAP vyplňte jako mailovací server hostitelský počítač a username z vašeho ISP; v případě lokálního doručování se vyplní cesta k lokální poštovní schránce.
5. Vyberte tabulku Sending Mail a zvolte typ serveru pro odchozí zprávy: SMTP v případě vzdáleného serveru (program si vyžádá jeho jméno), anebo sendmail, je-li server lokální stroj.
6. Zbytek tabulek a ostatní volby závisí na vaší úvaze. Činnost Evolution Account Editoru se ukončí příkazem OK. Může se začít mailovat.

Inbox	Zobrazení obsahu schránky s elektronickou poštou.
New	Vytvoření nové zprávy
Send/Receive	Dotaz na novou zprávu
Reply	Odpověď na zprávu, pouze odesílateli
Reply to All	Odpověď na všechny adresy na řádcích To a CC
Forward	Přeposlání zprávy třetí osobě

Existuje mnohem více možností, je třeba je vyzkoušet.

mutt [*volby*] mutt

/usr/bin **stdin stdout -file -opt -help -version**

`mutt` je textově orientovaný poštovní program, který se spouští na obyčejném terminálu (nebo v terminálovém okně), lze jej používat jak lokálně (tj. v prostředí X window), tak i vzdáleně přes připojení SSH. Je velmi mocný, s mnoha příkazy a volbami. Spouští se:

```
$ mutt
```

Když se zobrazí hlavní okno, vypíší se všechny zprávy ve schránce, po jedné na řádek. Příkazy naleznete v tabulce 4.

Tabulka 4. *Příkaz mutt, povely na obrazovce*

Klávesa	Význam
Šipka nahoru	Posun na předchozí zprávu.
Šipka dolů	Posun na následující zprávu.
PageUp	Posun o jednu stránku nahoru.
PageDown	Posun o jednu stránku dolů.
Home	Posun na první zprávu.
End	Posun na poslední zprávu.
`m`	Vytvoření nové zprávy. Spustí se implicitní editor. Příkazem `y` se odešle, příkazem `q` se odloží.
`r`	Odpověď na běžnou zprávu. Pracuje jako `m`.
`f`	Přeposlání zprávy třetí osobě. Pracuje jako `m`.
`i`	Zobrazení obsahu schránky s elektronickou poštou.
`c`	Kopírování obsahu schránky do jiné schránky.
`d`	Vynechání běžné zprávy.

Tabulka 5. *Příkaz mutt při psaní zprávy*

Klávesa	Význam
`a`	Připojení souboru (přílohy) ke zprávě.
`c`	Nastav seznam CC.
`b`	Nastav seznam BCC.
`e`	Znovu opravit zprávu.
`r`	Editace pole Reply-To.
`s`	Editace řádku s předmětem zprávy (subject).
`y`	Odeslání zprávy.
`C`	Kopírování zprávy do souboru.
`q`	Odložení zprávy bez odeslání.

Další příkazy jsou uvedeny v tabulce 6.

Tabulka 6. *Ostatní příkazy mutt*

Klávesa	Význam
`?`	Výpis všech příkazů (mezerou se posouvají dolů, `q` se ukončí)
`^C`	Zrušení probíhajícího příkazu
`q`	Konec

Oficiální stránky mutt jsou na *http://www.mutt.org* a stručný popis je na *http://www.cs.utk.edu/~help/mail/mutt_starting.php*.

mail [*volby*] *příjemce* MailX

/bin **stdin stdout -file -opt -help -version**

Program `mail` (anebo `Mail`)* je rychlý a jednoduchý mailovací klient. Uživatelé většinou požadují výkonnější klienty, avšak pro rychlé použití je mail postačující.

```
$ mail novak@priklad.cz
Subject: můj předmět
Píši zprávu.
Ukončím ji samostatnou tečkou na řádku.
.
Cc: plch@priklad.cz
$
```

Rychlou zprávu lze pomocí jediného příkazu poslat takto:

```
$ echo "Nazdar kluci" | mail -s "subject" novak@priklad.cz
```

Jednoduché odeslání souboru může vypadat jako jeden z těchto příkladů:

```
$ mail -s "můj předmět" novak@priklad.cz < jm_soub
$ cat jm_soub | mail -s "můj předmět"
novak@priklad.cz
```

Všimněme si, jak snadno lze odeslat výstup roury jako e-mailovou zprávu.

Užitečné volby

`-s` *`předmět`*	Nastavení předmětu zprávy.
`-v`	„Upovídaná“ verze: výpis zpráv o odeslání.
`-c` *`adresy`*	Nastavení pole CC, adresy jsou odděleny čárkami.
`-b` *`adresy`*	Nastavení pole BCC, adresy jsou odděleny čárkami.

Brouzdání po síti

mozilla	**Plně funkční internetový prohlížeč**
lynx	**Textový internetový prohlížeč**
wget	**Stahování stránek ze sítě**
curl	**Stahování stránek ze sítě**

V Linuxu existuje několik možností, jak prohlížet Internet: tradiční prohlížeče, textové prohlížeče a nástroje na stahování stránek na vlastní počítač.

* Ve starších unixových systémech byly `mail` a `Mail` odlišné programy. V Linuxu jsou to shodné programy: */usr/bin/Mail* je symbolický odkaz na */bin/mail*.

mozilla [*volby*] [*URL*]

mozilla

/usr/bin stdin stdout -file -opt -help -version

Mozilla je jedním z nejpopulárnějších internetových prohlížečů; lze ji provozovat i na jiných systémech včetně X window. V pozadí se spustí příkazem:

```
$ mozilla &
```

Mozilla disponuje všemi funkcemi, které lze od takového nástroje očekávat (brouzdání, tlačítka *vpřed* i *vzad*, záložky, historii atd.), navíc nabízí prohlížení stránek v panelech („tabbed browsing"), potlačení vyskakovacích oken („pop-up windows") a mnoho dalších. Je také možné využívat kompletní mailovací nástroj a číst zpravodajství. Stručný návod k obsluze je uveden v nápovědě, kompletní popis všech funkcí lze nalézt na *http://www.mozilla.org*.

Jinými linuxovými prohlížeči jsou např. Firebird („převlečená" Mozilla) na *http://www.mozilla.org/products/firebird*), Netscape (taktéž klon Mozilly) na *http://www.netscape.com*, Opera (*http://www.opera.com*), Konquerer pro KDE (*http://www.konquerer.org*), Epiphany pro GNOME (*http://www.gnome.org*) a Galeon (rovněž založený na Mozille, *http://galeon.sourceforge.net*).

lynx [*volby*] [*URL*]

lynx

/usr/bin stdin stdout -file -opt -help -version

`lynx` je pouze textový prohlížeč, což dnes už je raritou, může se však hodit, když uživatele grafika nezajímá, anebo má pomalé připojení k Internetu.

```
$ lynx http://www.yahoo.com
```

Veškeré brouzdání se ovládá z klávesnice, nikoli myší. Stránky nebudou vypadat příliš pěkně, zejména tam, kde obsahují větší počet tabulek a rámečků, ale většinou se lze i v takových stránkách zorientovat.

Klávesa	Význam
?	Nápověda.
k	Výpis všech kláves a jejich funkcí.
^G	Zrušení probíhajícího příkazu.
q	Konec.
enter	„Kliknutí" na odkaz nebo ukončení pole ve formuláři.
šipka vlevo	Zpět na předchozí stranu.
šipka vpravo	Vpřed na následující stranu nebo „kliknutí" na odkaz.
g	Přechod na URL (vypíše se prompt).
p	Uložení, tisk nebo odeslání běžné strany.
mezera	Posun dolů.
b	Posun nahoru.
šipka dolů	Přechod na následující odkaz nebo pole.

Klávesa	Význam
šipka nahoru	Přechod na předchozí odkaz nebo pole.
^A	Přechod na začátek strany.
^E	Přechod na konec strany.
m	Návrat na hlavní / domovskou stranu.
/	Hledání textu na stránce.
a	Založení běžné stránky.
v	Výpis seznamu záložek.
r	Odstranění záložky.
=	Zobrazení vlastností běžné stránky a odkazu.
\	Zobrazení HTML (a návrat zpět při dalším zadání).

`lynx` má více než 100 voleb na příkazovém řádku, podrobnosti lze nalézt na manuálových stránkách.

Užitečné volby

`-dump`	Výpis stránek na standardní výstup a ukončení (srv. se `-source`)
`-source`	Výpis HTML na standardní výstup a ukončení (srv. s příkazy `wget` a `curl`)
`-emacskeys`	Nastavení klávesnice přibližně dle editoru `emacs`.
`-vikeys`	Nastavení klávesnice přibližně dle editoru `vim` (nebo `vi`).
`-homepage=`*URL*	Nastavení domovské stránky na *URL*.
`-color -nocolor`	Zapnutí / vypnutí barevného textového režimu.

wget [*volby*] [*URL*]

wget

/usr/bin stdin stdout -file -opt -help -version

Příkazem `wget` se stáhne obsah URL do souboru nebo na standardní výstup. Je to velmi užitečná funkce pro získávání obsahu celých struktur stránek do libovolné hloubky. Jak by vypadalo stažení domovské stránky Yahoo:

```
$ wget http://www.yahoo.com
--23:19:51--  http://www.yahoo.com/
              => `index.html'
Resolving www.yahoo.com... done.
Connecting to www.yahoo.com[216.109.118.66]:80...
connected.
http request sent, awaiting response... 200 OK
Length: unspecified [text/html]

                [ <=>                                 ] 31,434
220.84K/s

23:19:51 (220.84 KB/s) - `index.html' saved [31434]
```

Stránka se stáhne do souboru *index.html* v běžném adresáři, `wget` navíc umí zopakovat stahování v případě jakéhokoli přerušení přenosu, např. kvůli chybě v síti: pouze se zadá `wget -c` se stejným URL a pokračuje se tam, kde se přestalo.

Jiným podobným příkazem je `curl`, který na rozdíl od `wget` vypisuje implicitně na standardní výstup (`wget` píše do souboru).

```
$ curl http://www.yahoo.com > mojestrana.html
```

Příkaz `wget` má více než 70 voleb, zmíníme se jen o některých. (`curl` má jiné volby – viz manuál).

Užitečné volby

`-i` *jm_soub*	Čtení URL ze souboru, jedno po druhém.
`-O` *jm_soub*	Zápis všech stažených HTML do souboru, stránky se připojují jedna za druhou.
`-c`	Režim pokračování: bylo-li předchozí čtení přerušeno a uložila se jen část stránky, pokračuje se v místě přerušení. Jestliže se tedy stáhlo například 100 K ze 150 K souboru, volbou -c se spustí přečtení jen zbývajících 50 K, které se připojí k původnímu souboru. Pozor, pokud se mezitím obsah vzdáleného souboru změnil, wget to nezjistí. Tato volba by se měla používat jen tehdy, máme-li jistotu, že změna nenastala.
`-t` *N*	Akci zkus *N*-krát. Pro *N*=0 bez omezení.
`--progress=dot` `--progress=bar`	Stahování se znázorňuje vypisováním teček nebo lomítek.
`--spider`	Nestahuj, pouze zjisti, zda jsou k dispozici stránky ke stažení.
`-nd`	Všechny stránky se uloží do jednoho adresáře, a to i v případě, že vzdáleně jsou uloženy ve složitější adresářové struktuře.
`-r`	Rekurzivní získávání stránek: všechny adresáře i podadresáře.
`-l` *N*	Získání stránek do *N*-té úrovně (implicitně je hloubka 5).
`-k`	Ve všech souborech se upraví odkazy tak, aby stránky šly prohlížet lokálně.
`-p`	Stažení všech souborů, které jsou nutné k úplnému zobrazení stránek, tj. styly, obrázky atd.

`-L`	Sledování pouze relativních odkazů v rámci stránky, nikoli absolutních.
`-A` *vzor1,vzor2, vzor3,...*	Přijímací režim: stahují se pouze stránky, jejichž jména odpovídají zadání. Lze používat zástupné znaky jako v shellu.
`-R` *vzor1,vzor2, vzor3,...*	Odmítací režim: stahují se pouze stránky, jejichž jméno neodpovídá zadání.
`-I` *vzor1,vzor2, vzor3,...*	Inkluzivní režim: stahují se pouze soubory z těch adresářů, jejichž jména odpovídají zadání.
`-X`	Exkluzivní režim: stahují se pouze soubory z těch adresářů, jejichž jména *neodpovídají* zadání.

Usenet News

Usenet News je jedna z nejstarších komunit na internetu. Obsahuje desítky tisíc diskusních fór, *newsgroups*, jejichž prostřednictvím si lidé vyměňují zprávy. Fedora nabízí nástroj na čtení zpráv `slrn`, avšak na síti je k dispozici množství jiných nástrojů (`rn`, `trn`, `tin` atd.). Usenet News umí číst i Mozilla: stačí vybrat v menu Mail & Newsgroups. Usenet News lze také najít diskusních skupinách Goolu: *http://groups.google.com*.

Přístup na Usenet se uskuteční tak, že je třeba se napojit na příslušný server (např. *novinky.priklad.cz*). Záznam o napojení na skupiny se společně s přečtenými články automaticky uloží do souboru v domovském adresáři. Jsou to buď *~/.newsrc* nebo *~/.jnewsrc*, záleží na konfiguraci.

slrn [*volby*] **slrn**

/usr/bin **stdin stdout -file -opt -help -version**

`slrn` je nástroj na čtení Usenet News. Chceme-li jej používat, musíme nejdřív nastavit jméno daného serveru v shellové proměnné `NINTPSERVER`:

```
$ export NINTPSERVER=novinky.priklad.cz
```

Pak se musí vytvořit soubor pro uložení seznamu diskusních skupin (pokud jsme jej ovšem nevytvořili už dříve):

```
$ slrn --create
a může se začít se čtením:
$ slrn
```

Po spuštění se zobrazí stránka s diskusními skupinami, do kterých je uživatel přihlášen. V tabulce 7 je uvedeno ovládání programu pomocí kláves:

Tabulka 7. Ovládání slrn pomocí kláves

Klávesa	Význam
q	Konec.
(šipka) dolů	Další skupina.
(šipka) nahoru	Předchozí skupina.
enter	Číst vybranou skupinu.
p	Odeslání nového příspěvku do vybrané skupiny.
a	Přidání skupiny (musíme znát jméno).
u	Odhlášení ze skupiny (provede se po ukončení příkazu). Volbou s je možno se znovu přihlásit.

Zadáme-li enter (čtení skupiny), zobrazí se stránka skupiny obsahující seznam probíhajících diskusí („nitě" diskusí) v dané skupině. V tabulce 8 jsou uvedeny některé užitečné příkazy pro práci s touto stránkou:

Tabulka 8. Užitečné příkazy slrn pro práci s diskusemi

Klávesa	Význam
q	Konec.
(šipka) dolů	Další diskuse.
(šipka) nahoru	Předchozí diskuse.
enter	Čtení vybrané diskuse.
c	Označení všech diskusí. Zruší se pomocí ESCAPE.

V tabulce 9 jsou uvedeny některé příkazy pro práci s textem při čtení příspěvků.

Tabulka 9. Užitečné příkazy pro práci s textem při čtení příspěvků

Klávesa	Význam
q	Konec čtení, návrat na stránku skupin.
mezera	Další strana příspěvku.
b	Předchozí strana příspěvku.
r	Odpověď autorovi příspěvku e-mailem.
f	Odeslání prohlíženého příspěvku.
p	Odeslání nového příspěvku.
o	Uložení příspěvku do souboru.
n	Přechod na další nepřečtený příspěvek.
P	Přechod na předchozí nepřečtený příspěvek.

Kdykoli lze zadat ? (otazník), vypíše se stránka s nápovědou. `slrn` má velké množství všech možných příkazů a voleb a lze jej nakonfigurovat ze souboru *~/.slrnc*. Popisujeme jen ty základní; podrobnosti viz */usr/share/doc/slrn** a *http://www.slrn.org*.

Okamžité posílání zpráv

gaim	Okamžité poslání zprávy a IRC klient
talk	Linuxový / unixový program pro komunikaci
write	Zaslání zprávy na terminál
mesg	Zákaz mluvení, pouze psaní
tty	Výpis jména terminálu

V Linuxu existuje řada možností, jak posílat zprávy jiným uživatelům, ať už na stejném počítači nebo po síti. K dispozici jsou programy od historicky nejstarších `talk` a `write`, jež komunikují prostřednictvím linuxových terminálů, až po moderní klienty pro posílání zpráv typu `gaim`.

gaim [*volby*]

gaim

/usr/bin — stdin stdout -file -opt **–help –version**

`gaim` je klient pro okamžité posílání zpráv (instant messaging, IM) pracující s mnoha protokoly včetně AOL, MSN, Yahoo a dalšími. Je rovněž klientem IRC („International Relay Chat"), pracuje v prostředí X window.

```
$ gaim &
```

Pokud ještě nevyužíváte účet s některou ze služeb IM, je nutno si nejdříve takový účet vytvořit; k tomuto účelu doporučujeme navštívit například *http://www.aim.com* a vytvořit si IM účet s protokolem AOL. Poté stačí klepnout na tlačítko pro spuštění programu gaim, v přihlašovacím okně zadat jméno a heslo a jste připojeni.

Užitečné volby

`-u screenname`	Nastavení implicitního účtu na *screenname*.
`-l`	Automatické přihlášení po spuštění programu gaim (heslo je uloženo).
`-w [zpráva]`	Nastavení „nepřítomnosti" s volitelným zasláním zprávy.

talk [*uživatel*[*@hostí*]] [*tty*]

talk

/usr/bin — stdin stdout -file -opt **–help –version**

Program `talk` předběhl dobu o několik desetiletí: vytváří mezi dvěma uživateli spojení s možností vzájemné komunikace bez ohledu na to, zda jsou oba připojeni k témuž počítači či k různým počítačům. Jde o standardní linuxovou i unixovou aplikaci (kterou převzaly i jiné platformy), jež běží ve standardním textovém okně, např. `xterm`. Na horizontálně rozdělené obrazovce je vidět jak partnerův, tak i vlastní text.

```
$ talk kamarad@priklad.cz
```

Je-li *kamarád* připojen vícekrát, lze určit terminál, na který se má program talk napojit.

write [*uživatel*] [*tty*] util-linux

/usr/bin **stdin stdout -file -opt -help -version**

Program `write` je jednodušší než `talk`: pracuje pouze na jednom linuxovém stroji.

```
$ write novak
Čau, jak se máš?
Ahoj.
^D
```

Povelem `^D` se spojení ukončí. `write` je rovněž vhodný pro zasílání jednorázových zpráv do rour:

```
$ echo 'Nazdar!' | write novak
```

mesg [y / n] SysVinit

/usr/bin **stdin stdout -file -opt -help -version**

Programem `mesg` se nastavuje, jestli se pomocí programů `talk` nebo `write` může jiný uživatel napojit na můj terminál. `mesg y` znamená ano, `mesg n` znamená ne. Příkazem `mesg` bez parametrů se zjišťuje stav (ano či ne)*. S novějšími programy jako `gaim` příkaz `mesg` nespolupracuje.

```
$ mesg
is n
$ mesg y
```

tty coreutils

/usr/bin **stdin stdout -file -opt -help -version**

Programem `tty` se vypíše jméno terminálu patřícího běžnému shellu.

```
$ tty
/dev/pts/4
```

Výstup na obrazovku

echo	**Výpis jednoduchého textu na standardní výstup.**
printf	**Výpis formátovaného textu na standardní výstup.**
yes	**Výpis opakovaného textu na standardní výstup.**
seq	**Výpis posloupnosti čísel na standardní výstup.**
clear	**Vymazání obrazovky nebo okna.**

Linux nabízí několik možností psaní na standardním výstupu, např. když chce uživatel mluvit sám k sobě:

```
$ echo ahoj všem
ahoj všem
```

* Momentálně Fedora příkaz `mesg y` odmítá a vypisuje se chybová zpráva „tty device is not owned by group tty". Předpokládáme, že tato chyba bude záhy odstraněna.

Každý z příkazů má jinou sílu a hodí se k jinému účelu, nicméně všechny jsou neocenitelné pro poznání Linuxu, ladění programů a psaní shellových skriptů (viz „Programování v shell skriptu" na str. 142).

echo [*volby*] *řetězce* bash

součást shellu **stdin stdout -file -opt -help -version**

Příkazem `echo` se pouze vypíše parametr:

```
$ echo To je ale legrace
To je ale legrace
```

Bohužel, existuje více variant příkazu echo s poněkud odlišným chováním. Program echo je v */bin/echo*, avšak obvykle se použije stejnojmenný příkaz zabudovaný v shellu. Které `echo` se vlastně používá, lze zjistit pomocí příkazu `echo` bez parametrů.

Užitečné volby

`-n`	Vypisovaný řádek není ukončen znakem „nový řádek".
`-e`	Rozpoznávání a interpretace escape sekvencí. Zkuste např. `echo 'hello\a'` a `echo -e 'hello\a'`. V prvním případě se jen parametr doslova opíše, v druhém to navíc pípne.
`-E`	Zrušení rozpoznávání a interpretace escape sekvencí; opak k `-e`.

Přehled escape sekvencí:

`\a`	Výstraha (pípnutí)
`\b`	Posun zpět
`\c`	Vypisovaný řádek není ukončen znakem „nový řádek" (totéž, co `-n`)
`\f`	Posun „papíru" (formuláře)
`\n`	Posun řádku (nový řádek)
`\r`	Návrat „vozíku" (návrat na začátek řádku)
`\t`	Horizontální tabulátor
`\v`	Vertikální tabulátor
`\\`	Opačné lomítko
`\'`	Jednoduchá uvozovka
`\"`	Dvojitá uvozovka
`\nnn`	Znak, jehož oktalová hodnota je *nnn*

printf *formátovací_řetězec [parametry]* bash

součást shellu stdin **stdout** -file **-opt** -help -version

Příkaz `printf` je rozšířený příkaz `echo`: vypisuje formátované řetězce na standardní výstup. Pracuje se s ním podobně jako s příkazem `printf()` v programovacím jazyce C, v němž se formátovací řetězec aplikuje na posloupnost parametrů, čímž se vytvoří formátovaný výstup. Například:

```
$ printf "Uživatel %s má %d let.\n" pepík 29
Uživatel pepík má 29 let.
```

První parametr je formátovací řetězec, který v našem případě obsahuje dvě specifikace formátů, `%s` a `%d`. Následujícími parametry pepík a 29 se dosadí do formátovacího řetězce a výsledek se vypíše. Specifikace formátu může být efektní v případě čísel s pohyblivou řádovou tečkou:

```
$ printf  "Dělá to %0.2f Kč, pane.\n" 3
Dělá to 3.0 Kč, pane.
```

V Linuxu existují dva příkazy printf: jeden je v bash shellu, druhý v adresáři */usr/bin/printf*. Jsou identické s jedinou výjimkou: v bash shellu existuje specifikace formátu `%q`, kterou se zadává výpis obráceného lomítka („\"), takže jeho výstup se může bezpečně použít jako vstup do shellu. Všiměme si rozdílu:

```
$ printf "Toto je uvozovka: %s\ n" "\""
Toto je uvozovka: "
$ printf "Toto je uvozovka: %q\ n" "\""
Toto je uvozovka: \"
```

Je na uživateli, aby si ověřil, zda počet specifikací formátů je roven počtu parametrů zadaných `printf`. Pokud je jich víc, přebytečné se ignorují, je-li jich méně, dosazují se implicitní hodnoty (0 u číselného formátu, " " v případě řetězce). Doporučujeme nicméně považovat nestejný počet parametrů za chybu, i když `printf` nic nehlásí. Později se takové chyby, např. v shell skriptech, velice obtížně hledají.

Specifikace formátů jsou detailně popsány na manuálových stránkách funkce printf jazyka C (viz man 3 printf). Zde jsou některé z nich:

`%d`	Dekadické celé číslo
`%ld`	Dlouhé dekadické celé číslo
`%o`	Oktalové celé číslo
`%x`	Hexadecimální celé číslo
`%f`	Pohyblivá řádová tečka
`%lf`	Pohyblivá řádová tečka, dvojitá přesnost
`%c`	Samostatný znak
`%s`	Řetězec

`%q`	Řetězec s shellovými metaznaky
`%%`	Znak %

Za úvodním znakem `%` může následovat číselný výraz udávající minimální délku výstupu. Např. „`%5d`" znamená výpis dekadického čísla do pětiznakového pole a „`%6.2f`" znamená číslo v pohyblivé řádové tečce dlouhé 6 znaků se dvěma místy za desetinnou tečkou. Zde jsou příklady číselných výrazů:

n	Minimální délka *n*.
0*n*	Minimální délka *n*, vypisují se levostranné nuly.
n.m	Minimální délka *n*, *m* míst za desetinnou tečkou.

Příkaz `printf` taktéž interpretuje escape sekvence jako např. „`\n`" (výpis nového řádku) nebo „`\a`" (zvonek, pípnutí). Podrobnější popis je uveden v příkazu echo.

yes [*řetězec*] coreutils

/usr/bin stdin **stdout** -file -opt **-help** **-version**

Příkazem `yes` se spustí nekonečný výpis daného řetězce (nebo implicitně znaku „y"), po jednom na každý řádek.

```
$ yes znovu
znovu
znovu
znovu
...
```

I když na první pohled se může zdát takový příkaz nesmyslný, je velice vhodný pro změnu charakteru interaktivního příkazu na příkaz dávkový. Chcete se zbavit obtížné zprávy „Are you SURE you want to do that?"? Výstup z yes se nasměruje do roury tak, aby odpovídal na všechny tyto dotazy:

```
$ yes | můj_interaktivní příkaz
```

Jakmile skončí *můj_interaktivní příkaz*, ukončí se i příkaz `yes`.

seq [*volby*] *specifikace* coreutils

/usr/bin stdin **stdout** -file -opt **-help** **-version**

Příkazem `seq` se vypíše posloupnost celých nebo reálných čísel, kterou lze využít v rouře nebo v jiných programech. Existují tři možné specifikace parametrů:

Jedno číslo: horní mez

```
$ seq 3
1
2
3
```

Dvě čísla: horní a dolní mez

```
$ seq 5 2
5
4
3
2
```

Tři čísla: dolní mez, inkrement, horní mez

```
$ seq 1 .3 2
1
1.3
1.6
1.9
```

Užitečné volby

`-w` Výpis levostranných nul (zarovnání řádků):

```
$ seq -w 8 10
08
09
10
```

`-f form_řetězec` Formátování výstupních řetězců jako u příkazu `printf` pomocí `%g` (implicitně), `%e` nebo `%f`:

```
$ seq -f '**%g**' 3
**1**
**2**
**3**
```

`-s řetězec` Řetězec se použije jako oddělovač mezi čísly. Implicitně se vkládá nový řádek, tedy každé číslo se vypíše na zvláštní řádek.

```
$ seq -s ':' 10
1:2:3:4:5:6:7:8:9:10
```

clear — ncurses

/usr/bin stdin **stdout** -file -opt -help -version

Tímto příkazem se vymaže obrazovka nebo shellové okno.

Matematika a výpočty

xcalc	Zobrazit kalkulačku
expr	Jednoduchý výpočet na řádku
dc	Textová kalkulačka

Potřebujete kalkulačku? V Linuxu je k dispozici nejen obyčejná grafická kalkulačka, nýbrž i příkaz pro matematické výpočty.

xcalc [*volby*] — XFree86-tools

/usr/X11R6/bin stdin stdout -file -opt -help -version

Příkazem `xcalc` se zobrazí jednoduchá kalkulačka v X window. Implicitně se zobrazí tradiční kalkulačka; volbou `-rpn` si lze zvolit kalkulačku s reverzní polskou notací.

expr *výraz* — coreutils

/usr/bin stdin **stdout** -file -opt **-help** **-version**

Příkazem `expr` se provádí vyhodnocování matematických (a jiných) výrazů:

```
$ expr 7 + 3
10
$ expr '(' 7 + 3 ')' '*' 14          Speciální znaky musí
140                                  být v uvozovkách
$ expr length ABCDEFGH
7
$ expr 15 '>' 16
0
```

Parametry se oddělují mezerami. Všimněme si, že speciální znaky musí být buď v uvozovkách, nebo se zadávají jako escape sekvence. Ve výrazech lze používat závorky. Seznam operátorů příkazu `expr` je uveden v tabulce 10.

Tabulka 10. *Operátory příkazu expr*

Operátor	Numerická operace	Operace s řetězcem
+	Součet	
-	Rozdíl	
*	Násobení	
/	Celočíselné dělení	
%	Zbytek (modulo)	
<	Menší než	Dříve v abecedě
<=	Menší nebo rovno	Dříve v abecedě nebo stejně
>	Větší než	Později v abecedě
>=	Větší nebo rovno	Později v abecedě nebo stejně
=	Rovnost	Rovnost
!=	Nerovnost	Nerovnost
|	Booleovské „nebo"	Booleovské „nebo"
&	Booleovské „a"	Booleovské „a"
`s : regvyr`		Souhlasí regulární výraz `regvyr` s řetězcem `s`?
`substr s p n`		Výpis `n` znaků řetězce `s` od pozice `p` (`p`=1 znamená první znak)

Operátor	Numerická operace	Operace s řetězcem
`index` *s znaky*		Výsledkem je poloha prvního znaku v řetězci *s* shodného s některým znakem z řetězce *znaky*. 0 znamená, že se takový znak nenašel. Jde o stejnou funkci jako `index()` v jazyce C.

V booleovských výrazech jsou 0 a prázdný řetězec nepravdivé hodnoty, všechny ostatní jsou pravdivé.

Příkaz `expr` není příliš efektivní. Na výpočet složitějších výrazů je vhodnější použít např. jazyk Perl.

dc [*volby*] [*soubory*]

coreutils

/usr/bin **stdin stdout -file –opt –help –version**

Příkazem `dc` („desk calculator") se spustí kalkulačka pracující na bázi zásobníků s reverzní polskou notací, kterou se čtou výrazy ze standardního vstupu a výsledek se vypisuje na standardní výstup. Kdo umí pracovat s RPN kalkulačkou Hewlett-Packard, snáze pochopí syntaxi kalulačky `dc`. Pokud používáte jen klasickou kalkulačku, může se vám zdát práce s `dc` nepochopitelná. Zmíníme se jen o několika základních příkazech.

Operace se zásobníkem a výpočty:

`q` Konec `dc`.

`f` Výpis celého zásobníku.

`c` Vymazání celého zásobníku.

`p` Výpis hodnoty uložené na vrcholu zásobníku.

`P` Odstranění hodnoty z vrcholu zásobníku.

nk Nastavení přesnosti následujících operací na n desetinných míst (implicitní hodnota je 0: celočíselné operace)

Dvě hodnoty na vrcholu zásobníku se zpracují tak, že se vezmou z vrcholu zásobníku, provede se s nimi požadovaná operace a výsledek se uloží na vrchol zásobníku místo nich:

`+` Součet.

`-` Rozdíl.

`*` Násobení.

`/` Dělení.

`%` Zbytek.

`^` Mocnina (základem mocniny je druhá hodnota shora, exponentem je hodnota na vrcholu zásobníku)

Hodnota na vrcholu zásobníku se zpracuje tak, že se vezme z vrcholu zásobníku, provede se s ní požadovaná operace a výsledek se uloží na vrchol zásobníku místo ní:

`v` Druhá odmocnina

Příklady:

```
$ dc
4 5 + p          Výpis součtu čísel 4 a 5
9
2 3 ^ p          Výpis 2 na třetí
8
10 * p           Vrchol zásobníku krát 10, výpis výsledku
80
f                Výpis zásobníku
80
9
+p               Operace se dvěma hodnotami na vrcholu zásobníku,
89               výpis součtu
```

Datum a čas

xclock	**Grafické zobrazení času**
cal	**Kalendář**
date	**Výpis nebo nastavení data a času**
ntpdate	**Nastavení času podle vzdáleného časového serveru**

Potřebujete znát nebo nastavit datum? Přesný čas? Vyzkoušejte tyto příkazy.

xclock [*volby*]

XFree86-tools

/usr/X11R6/bin **stdin stdout -file –opt –help –version**

Příkazem `xclock` se zobrazí hodiny v X window. Existují i programy s jiným grafickým zobrazením, např. `oclock` (kruhový tvar), `t3d` (třídimenzionální skákající míče, uložené mimo cestu na */usr/X11R6/lib/xscreensaver/t3d*) a hodiny, které se ukládají na lištu v GNOME a KDE.

Užitečné volby

`-analog` Analogové hodiny s ručičkami.

`-digital [-brief]` Digitální hodiny s úplným datem a časem; zadá-li se `-brief`, ukáže se jen čas

`-update N` Aktualizace zobrazovaného času každých *N* vteřin.

Seznam všech prací lze zobrazit příkazem `atq` („at queue"):

```
$ atq
559 2003-09-14 07:00 a novak
```

Práce zadaná příkazem `at` se zruší příkazem `atrm` („at remove") s uvedením jejího čísla:

```
$ atrm 559
```

Užitečné volby

`-f` *jm_souboru* — Příkaz se přečte z uvedeného souboru, nikoli ze standardního vstupu.

`-c` *číslo_práce* — Výpis příkazů job na standardní výstup.

crontab [*volby*] *soubor*

vixie-cron

/usr/bin **stdin stdout -file -opt -help -version**

Příkazem `crontab` se podobně jako příkazem `at` spouštějí práce v určeném čase. Pomocí `crontab` však lze spouštět práce opakovaně, jako např. „tento příkaz spouštěj každé druhé úterý v měsíci o půlnoci". K tomuto účelu je třeba příkazu `crontab` připravit speciální „crontabový" soubor:

```
$ crontab -e
```

který se automaticky nainstaluje do systémového adresáře *(/var/spool/cron)*. V systému se každou minutu spouští proces `cron`, kterým se prohlížejí „crontabové" soubory a provádějí se naplánované práce.

`$ crontab -e`

Editace crontabového souboru implicitním editorem (`$EDITOR`)

`$ crontab -l`

Výpis crontabového souboru na standardní výstup

`$ crontab -r`

Zrušení crontabového souboru

`$ crontab mujsoubor`

mujsoubor se nainstaluje jako crontabový soubor

Superuživatel může pomocí volby `-u` *uživ_jmeno* pracovat s crontabovými soubory jiných uživatelů.

V crontabovém souboru jsou jednotlivé práce uváděny na samostatných řádcích. (Prázdné řádky a řádky s komentáři uvozené znakem „#" se ignorují). Každý řádek se skládá ze šesti polí oddělených mezerami. V prvních pěti polích se uvádí čas spouštění práce, poslední pole obsahuje příkaz samotný.

Minuty v hodině

Celá čísla od 0 do 59. Může zde být samostatné číslo (30), posloupnost čísel oddělených čárkami (0,15,30,45), rozsah (20-30), posloupnost rozsahů (0-15,50-59) anebo hvězdička s významem „všechno". Taktéž lze zadat „každý *n*-tý časový úsek" s příponou `/n`; např. `*/12` a `0-59/12` znamená `0,12,24,36,48` (tj. každých 12 minut).

Hodiny v dni

Stejná syntaxe jako v případě minut.

Dny v měsíci

Celá čísla od 1 do 31; rovněž lze zadávat posloupnosti, rozsah, posloupnost rozsahů a hvězdičku.

Měsíce v roce

Celá čísla od 1 do 12; rovněž lze zadávat posloupnosti, rozsah, posloupnost rozsahů a hvězdičku. Navíc lze používat třípísmenové zkratky (`jan`, `feb`, `mar`, ...), nikoli však v rozsazích nebo v posloupnostech.

Dny v týdnu

Celá čísla od 0 (neděle) do 6 (sobota); rovněž lze zadávat posloupnosti, rozsah, posloupnost rozsahů a hvězdičku. Navíc lze používat třípísmenové zkratky (sun, mon, tue, ...), nikoli však v rozsazích nebo v posloupnostech.

Příkazy

Libovolný shellový příkaz, který bude proveden v prostředí přihlášeného, lze se tedy odkazovat na vnější proměnné jako `$HOME`. Obecně platí pravidlo, že se mohou používat jen absolutní cesty k příkazům (tj. */usr/bin/who*, nikoli `who`).

V tabulce 11 je uvedeno několik příkladů zadávání času.

Tabulka 11. *Příklady zadávání času pro crontab*

`* * * * *`	**Každou minutu**
`45 * * * *`	**Vždy 45 minut po celé hodině (1:45, 2:45 atd.)**
`45 9 * * *`	**Každý den v 9:45**
`45 9 8 * *`	**Osmého každý měsíc v 9:45**
`45 9 8 12 *`	**Každého 8. prosince v 9:45**
`45 9 8 dec *`	**Každého 8. prosince v 9:45**
`45 9 * * 6`	**Každou sobotu v 9:45**
`45 9 * * sat`	**Každou sobotu v 9:45**
`45 9 * 12 6`	**Každou sobotu v prosinci v 9:45**
`45 9 8 12 6`	**Každého 8. prosince a každou sobotu v 9:45**

Má-li prováděný příkaz nějaký výstup, `cron` jej pošle e-mailem.

Grafika a spořiče obrazovky

eog	Zobrazení grafických souborů
gqview	Zobrazení grafických souborů a předvádění diapozitivů
ksnapshot	Uložení obrazovky
gimp	Editace grafických souborů
gnuplot	Tvorba grafů a rýsování
xscreensaver	Spuštění spořiče obrazovky

Pro prohlížení a editaci grafiky existuje v Linuxu nepřeberné množství všech možných nástrojů a voleb. Nebudeme se o nich příliš rozepisovat, pokusíme se jen trochu nařuknout zájem čtenářů. Stačí, aby o nich něco věděli; ostatní už si zjistí sami.

eog [*volby*] [*soubory*] eog

/usr/bin **stdin stdout -file -opt -help -version**

Příkazem `eog` („Eye of Gnome") se zobrazí grafické soubory v různých formátech. Lze zobrazovat jeden soubor samostatně, anebo dva a více souborů:

```
$ eog soub1.jpg soub2.gif soub3.pbm
```

Každý ze souborů se zobrazí v jiném okně.

Popisovat ostatní volby příkazu `eog` by bylo zabíháním do technických podrobností, které si ušetříme; omezíme se pouze na sdělení, že volby `eog` *existují* a lze je nalézt například v nápovědě (`eog --help`).

gqview [*volby*] [*soubor*] gqview

/usr/bin **stdin stdout -file -opt -help -version**

Příkazem `gqview` se zobrazí grafické soubory v různých formátech a lze se automaticky přepínat z jednoho obrázku na druhý jako při promítání diapozitivů. Implicitně se zobrazí jména všech grafických souborů v běžném adresáři a lze vybírat jména obrázků, které se mají zobrazit. K dispozici je i menu na obrazovce. Příkazem `^q` se činnost příkazu ukončí.

Užitečné volby

`-f` Zobrazení na celou obrazovku. (Přepínat mezi režimem celé obrazovky a okna lze klávesou `v`).

`-s` Promítání obrázků jako diapozitivy (Střídavé zapínání a vypínání klávesou `s`).

ksnapshot [*volby*] kdegraphics

/usr/bin **stdin stdout -file -opt -help -version**

Příkaz `ksnapshot` je univerzálním nástrojem na snímání obrazovky. Zadá se pouze:

```
$ ksnapshot
```


čímž se sejme obrazovka a zobrazí se jako miniatura. Poté je možno ji uložit jako grafický soubor anebo pořídit další snímek. Jediným oříškem je výběr jména výstupního souboru; to se zadává při ukládání souboru příslušného standardního typu: .jpg pro soubor typu JPEG, .bmp pro windowsovou bitovou mapu, .pbm pro formát PBM („portable bitmap"), .eps pro zapouzdřený („encapsulated") PostScript, .ico pro windowsovou ikonu atd. Seznam podporovaných formátů se zobrazí po klepnutí na tlačítko Save Snapshot, pak výběr Filter.

Podrobnější informace získáme v nápovědě v okně `ksnapshot`, anebo lze tento příkaz spustit v shellu s volbou `--help`.

gimp [*volby*] [*soubory*] gimp

/usr/bin **stdin stdout -file -opt -help -version**

GIMP („GNU Image Manipulating Program") je komplexní balík na zpracování grafických obrázků, plně konkurující Adobe Photoshopu. Funkce tohoto balíku jsou skutečně rozsáhlé a výsledky ohromující. Úplný popis viz *http://www. gimp.org*. Program se spouští:

```
$ gimp
```

Jestliže chcete pracovat s určitým souborem, stačí zadat:

```
$ gimp jm_souboru
```

Pokud by byl program GIMP zbytečně rozsáhlý, existuje jeho jednodušší verze `xv` na *http://www.trilon.com/xv*, která se spouští:

```
$ xv mujsoubor.jpg
```

a menu programu se spustí tak, že se pravým tlačítkem myši klepne na obrázek.

gnuplot [*volby*] [*soubory*] gnuplot

/usr/bin **stdin stdout -file -opt -help -version**

Programem `gnuplot` se metodou spojování bodů křivkami a úsečkami kreslí grafy, které se ukládají do souborů v různých formátech pro tisk na tiskárně a kreslení, např. jako PostScript. Pro ovládání `gnuplot` je třeba zvládnout jednoduchý, avšak efektivní programovací jazyk. Ukážeme si, jak se nakreslí graf funkce $y=x^2$ pro `x` od 1 do 10 a zobrazí se v X window na displeji terminálu:

```
$ gnuplot
gnuplot> plot [1:10] x**2
gnuplot> quit
```

Totéž, avšak s uložením do postscriptového souboru:

```
$ echo 'set terminal postscript; plot [1:10] x**2' | gnuplot > out-
put.ps
```

Podrobnosti viz *http://www.gnuplot.info*.

xscreensaver xscreensaver

/usr/X11R6/bin stdin stdout -file -opt **-help** -version

Systém `xscreensaver` je spořič obrazovky bohatý na funkce se stovkami různých animací. Běží v pozadí a lze jej ovládat mnoha způsoby:

Po určité době nečinnosti

Implicitně se z grafického rozhraní Fedory (KDE nebo GNOME) spustí `screensaver` automaticky po pěti minutách nečinnosti. Konfigurace se provede z hlavního menu. V KDE se spustí aplikace Control Center, vybere se Appearence & Themes, pak Screen Saver. Alternativně lze pravým tlačítkem myši klepnout na desktop, vybere se Configure Desktop, pak Screen Saver. V GNOME se vybere Preference/Scrensaver z hlavního menu.

Jako zámek obrazovky

Kdykoli se (v GNOME a KDE) zvolí hlavní menu a vybere se Lock Screen. Obrazovka zůstane zamčená, dokud není zadáno přihlašovací heslo.

Na příkazovém řádku

Spustí se `xscreensaver-demo`, vybere se animace. Pak se spouští `xscreensaver-command` dle potřeby:

```
$ xscreensaver-command -activate    Vymaž obrazovku
$ xscreensaver-command -next        Vyber další animaci
$ xscreensaver-command -prev        Vyber předchozí animaci
$ xscreensaver-command -cycle       Vyber náhodnou animaci
$ xscreensaver-command -lock        Uzamkni obrazovku
$ xscreensaver-command -exit        Konec
```

Audio a video

grip	CD přehrávač, stahování z CD a kódování MP3
xmms	Přehrávač audio souborů (MP3, WAV)
cdparanoia	Stahování audio z CD do souborů WAV
audacity	Eitace audio souborů
xcdroast	Vypalování CD s grafickým rozhraním

Zvuku se v Linuxu daří velice dobře. Většina programů z této kategorie má intuitivní ovládání, množství funkcí a kvalitní dokumentaci, není třeba se zabývat podrobným popisem. Pouze si uvedeme stručný přehled toho, co je k dispozici. Adresy s vyčerpávajícím popisem této tematiky jsou například *http://linux-sound.org/* a *http://www.xdt.com/ ar/linux-snd.*

Fedora nemá vlastní videopřehrávače, je však možné si je stáhnout z Internetu a nainstalovat. Nejpopulárnější jsou `mpg123` (*http://www.mpg123.de/*), `smpeg` (*http://www.lokigames.com/development/*) a `mplayer` (*http://www.mplayerhq.hu*).

grip [*volby*] grip

/usr/bin stdin stdout -file -opt **-help** **-version**

`grip` je audiopřehrávač a nástroj na ukládání zvukových stop z CD do souborů WAV, příp. konverzi do MP3 („audio ripper"). Má rozsáhlou nápovědu a intuitivní ovládání.

cdparanoia [*volby*] *rozsah* [*výst_soub*] cdparanoia

/usr/bin stdin stdout -file **-opt** **-help** **-version**

Příkazem `cdparanoia` se čtou audio data z CD a ukládají se do souborů WAV (případně jiných formátů: viz manuál). Obvyklé použití vypadá např. takto:

`$ cdparanoia N`

Přenes stopu *N* do souboru.

`$ cdparanoia -B`

Přenes všechny stopy z CD do samostatných souborů.

`$ cdparanoia -B 2 -4`

Přenes stopy 2, 3 a 4 do samostatných souborů.

`$ cdparanoia 2-4`

Přenes stopy 2, 3 a 4 do jediného souboru.

Máte-li potíže s CD mechanikou, zkuste zadat `cdparanoia -Qvs` („upovídané" hledání CD), program vám nabídne vodítko. Soubory WAV se do MP3 převádějí pomocí LAME (*http://lame.sourceforge.net/*) nebo NotLame (*http://www. idiap.ch/~sanders/not_lame/*).

xmms [*volby*] [*soubory*] xmms

/usr/bin stdin stdout -file -opt **-help** **-version**

`xmms` (X MultiMedia System) je vynikající grafický přehrávač audio souborů podporující MP3, WAV, Ogg Vorbis a další audio formáty*. Ovládání je tradiční, je možno vytvářet si přehrávací seznamy a k dispozici je řada dalších funkcí. Nejlepší je začít tak, že se program spustí bez parametru:

```
$ xmms
```

anebo zadat audio soubory na příkazovém řádku:

```
$ xmms soub1.mp3 soub2.wav soub3.ogg
```

Zde jsou některé užitečné možnosti.

* Fedora má `xmms` nepodporující formát MP3; aktuální doplněnou verzi naleznete na *http://www.xmms.org/*.

Akce	Význam
Klepnutí pravým tlačítkem na lištu	Zobrazení hlavního menu
Klepnutí na tlačítko PL	Zobrazení přehrávacího seznamu (tlačítkem add lze přidávat další položky)
Klepnutí na tlačítko EQ	Zobrazení grafického ekvalizéru
Dvojité kleputí na stopu v přehrávacím seznamu	Přehrání stopy
Dvojité kleputí na přehrávací seznam	Zobrazení menu seznamu

audacity [*soubory*]

audacity

není součástí Fedory stdin stdout -file -opt -help -version

`audacity` je grafický editor audio souborů, kterým lze upravovat soubory WAV, MP3 a Ogg. Lze odnímat a přidávat zvukové záznamy, filtrovat je, přidávat zvukové efekty (echo, zvýrazňování basů, přehrávání pozpátku) apod. audacity není součástí Fedory, ale vřele se doporučuje stáhnout si jej z *http://audacity.sourceforge.net*.

xcdroast [*volby*]

xcdroast

/usr/bin stdin stdout -file -opt -help -version

`xcdroast` je program na vypalování CD s grafickým ovládáním. Podporuje pouze SCSI CD mechaniky. Kdo používá mechaniku IDE, musí si pořídit emulátor *ide-scsi*. Popis příslušného postupu, sahá daleko za možnosti této útlé příručky, ale v zásadě, pokud se na výstupu příkazu:

```
$ cdrecord -scanbus
```

objeví CD mechanika, je vše v pořádku. V opačném případě nahlédněte do „CD-Writing Howto" na *http://www.tldp.org/ HOWTO/CD-Writing-HOWTO.html*. Navíc, nežli použijete `xcdroast`, nahlédněte na *http://www.xcdroast.org* a přečtěte si dokumentaci, zejména *často kladené otázky* (FAQ), protože nastavení může být složité. Pak zadejte:

```
$ xcdroast
```

Klepněte na Setup a zkontrolujte nastavení programu. Poté zadejte Save Configuration a pomocí OK se vrátíte na hlavní obrazovku. Tam zvolte Duplicate CD nebo Create CD a pokračujte. Některé systémy jsou zkonfigurovány tak, že CD může vypalovat pouze superuživatel.

Programování v shell skriptu

Už jsme se zmínili o tom, že v shellu (`bash`) je zabudovaný programovací jazyk. Když uživateli nestačí běžné příkazy shellu, může si vytvářet programy

(*shell skripty*). Jako jiné dobré programovací jazyky, shell má proměnné, podmíněné větvení (if-then-else), cykly, vstupy a výstupy atd. O shellových skriptech byly napsány stohy knih, zde jsme schopni se zmínit jen o tom nejzákladnějším, abychom se pokusili vzbudit váš zájem. Úplnou dokumentaci získáte pomocí `info bash`.

Mezery a konce řádků

Shellové skripty jsou velice háklivé na mezery a konce řádků. Klíčovými slovy tohoto programovacího jazyka jsou skutečné příkazy zpracovávané shellem a parametry musí být odděleny mezerami. Podobně, konec řádku uprostřed příkazu se chápe jako nekompletní příkaz. Je vhodné dodržovat zde uvedené zásady.

Proměnné

O proměnných jsme hovořili už dříve:

```
$ MOJEPROM=6
$ echo $MOJEPROM
6
```

Hodnoty, kterých mohou proměnné nabývat, jsou řetězce, avšak jsou-li numerické, budou se považovat za čísla, ukáže-li se, že je to vhodné.

```
$ CISLO="10"
$ expr $CISLO + 5
15
```

Hodnota proměnné v shell skriptu by měla být uzavřena v (dvojitých) uvozovkách, předejde se tím některým chybám v době běhu programu („runtime errors"). Nedefinovaná proměnná nebo proměnná obsahující mezery, pokud nejsou uzavřeny v závorkách, se vyhodnotí chybně, což povede k chybné funkci skriptu.

```
$ JMENO_SOUB="Můj dokument"                    Mezera ve jméně
$ ls $JMENO_SOUB                               Zkus je vypsat
ls: Můj: No such file or directory             Doprč...! ls viděl
                                               2 parametry
ls: dokument: No such file or directory
$ ls -l "$JMENO_SOUB"                          Výpis je správný
Můj dokument                                   ls viděl jen 1 parametr
```

Sousedí-li jméno proměnné s řetězcem, je vhodné je uzavřít do složených závorek, aby nedošlo k neočekávaným akcím:

```
$ HAT="brambor"
$ echo "Množné číslo od $HAT je $HATy"
Množné číslo od brambor je                     Proměnná „HATy" neexistuje
$ echo"Množné číslo od $HAT je ${HAT}y"
Množné číslo od brambor je brambory            Tak jsme to chtěli!
```

Vstupy a výstupy

Výstup ve skriptu se realizuje pomocí příkazů `echo` a `printf`, které jsou popsané ve „Výstupu na obrazovku" na str. 124.

```
$ echo "Všechny zdravím"
Všechny zdravím
$ printf  "Je mi %d roků\n" `expr 20 + 20`
Je mi 40 roků
```

Vstup se provádí příkazem `read`, kterým se přečte jeden řádek ze standardního vstupu a uloží se do proměnné:

```
$ read jméno
Mařenka Nováková <ENTER>
$ echo "Přečetl jsem jméno $jméno"
Přečetl jsem jméno Mařenka Nováková
```

Booleovské proměnné a návratové kódy

Ještě než si popíšeme podmínky a cykly, zmíníme se o booleovských hodnotách true/false. V shellu je hodnota 0 true neboli úspěch, vše ostatní je false resp. neúspěch.

Navíc, každý linuxový příkaz vrací celočíselnou hodnotu nazývanou *návratový kód* nebo také *stav* po ukončení příkazu. Tento stav je uložen ve speciální proměnné `$?`:

```
$ cat mujsoubor
Jmenuji se Franta Novák a
Fedoru mám opravdu rád
$ grep Novák mujsoubor
Jmenuji se Franta Novák a          Nalezla se shoda ...
$ echo $?
0                                  ... takže návratový kód je „úspěch".
$ grep hraboš mujsoubor
$ echo $?                          Nenalezla se shoda ...
1                                  ... takže návratový kód je „neúspěch"
```

Návratové kódy příkazů jsou uvedeny v příslušných manuálech.

test a „["

Příkaz `test` (zabudovaný v shellu) vyhodnotí jednoduchý booleovský výraz obsahující čísla a řetězce, nastaví návratový kód na 0 (true) nebo 1 (false):

```
$ test 10 -lt 5          Je 10 menší než 5?
$ echo $?
1                        Ne, není
$ test -n "ahoj"         Má řetězec „ahoj" nenulovou délku?
$ echo $?
0                        Ano, má
```


Seznam nejčastěji testovaných parametrů ohledně zjišťování vlastností celých čísel, řetězců a souborů naleznete v tabulce 12.

Příkaz `test` má jedno neobvyklé synonymum, a to „[" neboli levá hranatá závorka, jež slouží jako zkratka pro podmíněné větvení. Pokud se tato zkratka použije, musí být za posledním parametrem naopak „]" (pravá hranatá závorka) znamenající ukončení testu. Následující test je totožný s testem předchozím:

```
$ [ 10 -lt 5 ]
$ echo $?
1
$ [ -n "ahoj" ]
$ echo $?
0
```

Je třeba si uvědomit, že „[" je příkaz jako každý jiný a seznam jeho parametrů musí být také oddělen mezerami. takže pokud omylem zapomenete na mezeru:

```
$ [ 5 -lt 4]              Mezi 4 a ] chybí mezera
bash: [: missing ']'
```

`test` si myslí, že posledním parametrem je řetězec „4]" a upozorní na chybějící hranatou závorku.

Tabulka 12. *Parametry obvykle používané příkazem test*

Testování souborů	
`-d` *jméno*	Soubor *jméno* je adresář
`-f` *jméno*	Soubor *jméno* je regulérní soubor
`-L` *jméno*	Soubor *jméno* je symbolický odkaz
`-r` *jméno*	Soubor *jméno* existuje a lze jej číst
`-w` *jméno*	Soubor *jméno* existuje a lze do něj zapisovat
`-x` *jméno*	Soubor *jméno* existuje a je spustitelný
`-s` *jméno*	Soubor *jméno* existuje a má nenulovou velikost
`f1 -nt f2`	Soubor *f1* je novější než soubor *f2*
`f1 -ot f2`	Soubor *f1* je starší než soubor *f2*
Testování řetězců	
s1 `=` *s2*	Řetězce *s1* a *s2* jsou shodné
s1 `!=` *s2*	Řetězce *s1* a *s2* nejsou shodné
`-z` *s1*	Řetězec *s1* má nulovou délku
`-n` *s1*	Řetězec *s1* má nenulovou délku
Testování čísel	
`a -eq b`	Celá čísla *a* a *b* jsou stejná
`a -ne b`	Celá čísla *a* a *b* nejsou stejná

`a -gt b`	Celé číslo *a* je větší než celé číslo *b*
`a -ge b`	Celé číslo *a* je větší nebo rovno celému číslu *b*
`a -lt b`	Celé číslo *a* je menší než celé číslo *b*
`a -le b`	Celé číslo *a* je menší nebo rovno celému číslu *b*

Kombinování a negace testů

`t1 -a t2`	A: Oba testy t1 a t2 mají hodnotu true
`t1 -o t2`	Nebo: Alespoň jeden z testů t1 nebo t2 je pravdivý
`! váš_test`	Negace: tj. *váš_test* má hodnotu false
`\( váš_test \)`	Závorky lze užívat v algebraickém smyslu

true a false

`bash` má zabudované příkazy `true` a `false`, jimiž lze nastavit výstupní kód na 0 resp. 1.

```
$ true
$ echo $?
0
$ false
$ echo $?
1
```

K významu těchto příkazů se vrátíme u výkladu podmíněných cyklů.

Podmínky

Příkazem `if` se vybírá mezi alternativami, z nichž každá může sama být složitým testem. Nejprve jednoduchý příkaz `if-then`:

```
if příkaz                    Výstupní kód příkazu je 0
then
  tělo
fi
```

Například:

```
if  [ `whoami` = "root" ]
then
  echo "Jsi superuživatel"
fi
```

Dále příkaz `if-then-else`:

```
if příkaz
then
  tělo1
else
  tělo2
fi
```


Například:

```
if  [ `whoami` = "root" ]
then
  echo "Jsi superuživatel"
else
  echo "Jsi normální frajer"
fi
```

Zobecněním je konstrukce `if-then-elif-else`, která může obsahovat libovolný počet testů.

```
if příkaz1
then
  tělo1
elif příkaz2
then
  tělo2
elif …
  …
else
  těloN
fi
```

Například:

```
if  [ `whoami` = "root" ]
then
  echo "Jsi superuživatel"
elif
if  [ "$USER" = "root" ]
  echo "Mohl bys být superuživatel"
elif [ "$uplatek" -gt 10000 ]
  echo "Dej bakšiš a budeš superuživatel"
else
  echo "Jsi jen normální frajer"
fi
```

Příkaz `case` otestuje hodnotu a podle ní větví výpočet:

```
echo 'Co bys chtěl dělat?'
read answer
case "$answer" in
  jíst)
       echo "Tak si dej buřta"
       ;;
  spát)
       echo "Tak dobrou noc"
       ;;
  *)
```

```
        echo "Tomu moc nerozumím"
        echo "Ale těším se na zítřek"
        ;;
esac
```

Obecný tvar je:

```
case řetězec in
     vyr1)
        tělo1
        ;;
     vyr2)
        tělo2
        ;;
     ...
     vyrN
        těloN
        ;;
     *)
        tělo else
        ;;
esac
```

kde *řetězec* je libovolná hodnota, obvykle proměnná jako `$mojeprom` a *vyr1* až *vyrN* jsou vzory (podrobnosti viz příkaz `info bash reserved case`) a na konci * místo posledního „else“. Každá skupina příkazů se musí ukončit ;; (viz příklad):

```
case $písmeno in
     X)
        echo "$písmeno je X"
        ;;
[aeiouy])
        echo "$písmeno je malá, krátká neměkčená samohláska"
        ;;
[0-9])
        echo "$písmeno je číslice, co je to za blbost"
        ;;
     *)
        echo "Tohle teda nepoznám"
        ;;
esac
```

Cykly

Příkazem pro cyklus `while` se opakuje množina příkazů tak dlouho, dokud je podmínka true (splněna).

```
while příkaz      Dokud je výstupní kód 0
do
  tělo
done
```


Příklad skriptu *můjskript*:

```
i=0
while [ $i -lt 3 ]
do
  echo "ještě"
  i=`expr $i + 1`
done

$ ./můjskript
0
1
2
```

Příkazem pro cyklus `while` se opakuje množina příkazů tak dlouho, dokud podmínka nenabude hodnoty true:

```
until příkaz
do
  tělo
done
```

Například:

```
i=0
until [ $i -gt 3 ]
do
  echo "ještě"
  i=`expr $i + 1`
done
$ ./můjskript
0
1
2
```

Příkazem pro cyklus `for` se iteruje podle hodnot uvedených v seznamu:

```
for proměnná in seznam
do
  tělo
done
```

Například:

```
for jméno in Tom Kuba Frantík
do
  echo "$jméno je můj kamarád"
done
$ ./můjskript
Tom je můj kamarád
Kuba je můj kamarád
Frantík je můj kamarád
```

Příkaz pro cyklus `for` je vhodný pro zpracování seznamů souborů, např. všech souborů určitého typu v běžném adresáři:

```
for soubor in *.doc
do
  echo "$soubor je hnusný wordovský soubor"
done
```

Nekonečný cyklus lze naprogramovat pomocí příkazu `while` s podmínkou `true` nebo `until` s podmínkou `false`:

```
while true
do
  echo "navždy"
done

until false
do
  echo "znovu navždy"
done
```

Tyto cykly se mohou ukončit pomocí `break` nebo `exit`.

Break a continue

Příkaz `break` vyskočí z nejbližšího uzavřeného cyklu. Následující skript nazvěme můjskript:

```
for jméno in Tom Kuba Frantík
do
  echo $jméno
  echo "znovu"
done
echo "hotovo"

$ ./můjskript
Tom
znovu
Kuba
znovu
Frantík
znovu
hotovo
```

A nyní s příkazem `break`:

```
for jméno in Tom Kuba Frantík
do
  echo $jméno
  if [ "$jméno" = "Kuba" ]
  then
    break
  fi
  echo "znovu"
done
echo "hotovo"

$ ./můjskript
Tom
znovu
Kuba                    Vyskytl se break
hotovo
```


Příkazem `continue` se v cyklu přejde na další iteraci.

```
for jméno in Tom Kuba Frantík
do
  echo $jméno
  if [ "$jméno" = "Kuba" ]
  then
    continue
  fi
  echo "znovu"
done
echo "hotovo"

$ ./můjskript
Tom
znovu
Kuba                    Vyskytl se continue
Frantík
znovu
hotovo
```

V příkazech `break` a `continue` se může použít i číselný parametr (`break` *N*, `continue` *N*) pro řízení vnořených cyklů (tj. vyskočit z *N*-té úrovně vnořeného cyklu), ale tyto způsoby v psaní skriptů vedou ke špagetovému kódu, který nelze doporučit.

Vytváření a spouštění shell skriptů

Shellový skript se nejsnáze vytvoří tak, že se příkazy `bash` uloží do souboru. Spustit jej lze trojím způsobem:

Vytvoř #!/bin/bash a zadej, aby byl soubor spustitelný.

Toto je jeden z nejobvyklejších způsobů, jak spouštět skripty. Přidej řádek:

```
#!/bin/bash
```

na začátek skriptového souboru. Musí to být první řádek souboru a zarovnaný doleva. Poté se k vlastnostem souboru přidá příznak spustitelnosti:

```
$ chmod +x můjskript
```

Volitelně jej lze přesunout do adresáře na vyhledávací cestě. Pak jej lze spustit jako kterýkoli jiný příkaz:

```
$ můjskript
```

je-li skript v běžném adresáři, avšak běžný adresář „.“ není ve vyhledávací cestě a musí se proto předřadit „./“, takže se najde soubor:

```
$ ./můjskript.
```

Běžný adresář není ve vyhledávací cestě z bezpečnostních důvodů.

Předej bash

`bash` iterpretuje parametr jako jméno skriptu, spustí jej.

```
$ bash můjskript
```

Spusť běžný shell s „.“

Předchozí metodou se spustí skript jako nezávislá entita, která nemá vliv na běžný shell.* Pokud si ovšem přejete, aby skript naopak změny v běžném shellu provedl (nastavení proměnných, změna adresáře atd.), lze jej spustit pomocí „.“ příkazem:

```
$ . můjskript
```

Parametry na příkazovém řádku

Shell skript pracuje s parametry na řádku podobně jako jiné linuxové příkazy. (Ve skutečnosti některé linuxové příkazy *jsou* skripty.) Uvnitř shell skriptu se na ně lze odkazovat jako na `$1`, `$2`, `$3` atd.

```
$ cat můjskript
#!/bin/bash
echo "Jmenuji se $1 a pocházím z $2"

$ ./můjskript Franta
Jmenuji se Franta a pocházím z
```

Skript si může otestovat počet zadaných parametrů pomocí `$#`:

```
if [ $# -lt 2 ]
then
  echo "chyba $0: musí se zadat dva parametry"
else
  echo "Jmenuji se $1 a pocházím z $2"
fi
```

* Ve skutečnosti skript běží v jiném shellu (*subshell* neboli *potomek*), který zdědí atributy po původním shellu, přičemž v původním shellu se tyto atributy nezmění.

Speciální hodnota `$0` obsahuje jméno daného skriptu a je k dispozici nejen pro hlášení chyb:

```
$ ./můskript Franta
./můjskript chyba: musí se zadat dva parametry
```

Pokud parametry chybí, jejich postupné vyžádání lze provést pomocí příkazu `for` s využitím speciální proměnné `$@`, v níž jsou uloženy všechny parametry:

```
for par in $@
do
  echo "Našel jsem parametr $par"
done
```

Ukončení s návratovým kódem

Příkazem `exit` se skript ukončí a shellu se předá návratový kód. Při úspěšném ukončení je návratový kód 0, při chybě 1 (nebo jiná hodnota různá od nuly). Pokud skript není ukončen příkazem `exit`, návratový kód se automaticky nastaví na 0.

```
if [ $# -lt 2 ]
then
  echo "Chyba: musí se zadat dva parametry"
  exit 1
else
  echo "Jmenuji se $1 a pocházím z $2"
fi
exit 0

$ ./můskript Franta
./můjskript chyba: musí se zadat dva parametry
$ echo $?
1
```

Když skript nestačí

Shellový skript je většinou dostatečně výkonným nástrojem, avšak v Linuxu existují mnohem mocnější prostředky pro psaní skriptů, podobně jako je tomu u programovacích jazyků. Zde je přehled některých z nich:

Jazyk	Program	Popis
Perl	`perl`	`man perl` *http://www.perl.com/*
Python	`python`	`man python` *http://www.python.org/*
C, C++	`gcc`	`man gcc` *http://www.gnu.org/software/gcc*

Jazyk	Program	Popis
Java	`javac`[a]	*http://java.sun.com/*
FORTRAN	`g77`	`man g77` *http://www. gnu.org/software/fortran/fortran.html*
Ada	`gnat`	`info gnat` *http://www. gnu.org/software/gnat/gnat.html*

Závěrečné slovo

I když by se mohlo zdát, že jsme popsali mnoho příkazů a možností Linuxu, ve skutečnosti jsme se pohybovali pouze po povrchu. Fedora i ostatní distribuce disponují *tisíci* nejrůznějších programů. Doporučujeme pokračovat ve studiu Linuxu, zkoušet nové věci a seznamovat se s dalšími funkcemi vaší distribuce. Mnoho štěstí!

Poděkování

Srdečné díky mému vydavateli Miku Loukidesovi, produkčnímu týmu nakladatelství O'Reilly, technickým redaktorům (Ronu Bellomovi, Wesleymu Crossmanovi, Davidu Debonnairovi, Timu Greerovi, Jacobu Heiderovi a Eriku van Oorschotovi), Alexi Schowtkovi a Robertu Dulaneymu z VistaPrintu a mé kouzelné rodině, Lise a Sophii.

[a] Není součástí Fedory ani většiny jiných distribucí Linuxu

Rejstřík

!!, 28
&, 29
&&, 26
. (tečka), 16
.. (dvě tečky za sebou), 16
/ (lomítko), 16
; (středník), 26
[(levá hranatá závorka), 145
\ (obrácené lomítko), 27
^Z, 29
| (roura), 26
||, 26
~ (vlnovka), 17, 24

A

abiword, 50
absolutní cesta, 16
Ada, 154
adresář, 15, 37
– domovský, 17
– systémový, 17
alias, 25
aspell, 91
at, 135
atributy souborů, 50
audacity, 142
audio, 140
awk, 71

B

balení souborů, 72
basename, 38
běžný adresář, 16
bg, 30
bin, 18
booleovské proměnné, 144
/boot, 20
break, 150
brouzdání po síti, 117
bzip2, 32, 74

C

C a C++, 153
^C, 30
cal, 132
case, 147
cat, 39
cd, 16, 38
cdparanoia, 141
cdrecord, 87
cesta, 16
– absolutní, 16
– relativní, 16
cgi-bin, 18
citace, 27
clear, 128
cmp, 78
comm, 77
compress, 32, 73
continue, 150–151
cp, 36
crontab, 136
cut, 66
cykly, 148

Č

čas, 131

D

date, 132
datum, 131

cal [*volby*] [*měsíc*[*rok*]]

util-linux

/usr/bin stdin **stdout** -file --opt --help --version

Příkazem `cal` se vypíše kalendář – implicitně běžný měsíc:

```
$ cal
    September 2003
Su    Mo    Tu    We    Th    Fr    Sa
                1     2     3     4     5     6
 7     8     9    10    11    12    13
14    15    16    17    18    19    20
21    22    23    24    25    26    27
28    29    30
```

Jiný kalendář je možno vytisknout, zadá-li se měsíc a rok čtyřmi ciframi: `cal 8 2002`. Vynechá-li se měsíc (`cal 2002`), vypíše se celý rok.

Užitečné volby

`-y` Výpis kalendáře na běžný rok.

`-m` Pracovní kalendář; týden začíná pondělím.

`-j` Číslování dní v roce; 1. září 2003 je 244. den, 2. září 245. atd.

date [*volby*] [*tvar*]

coreutils

/bin stdin **stdout** -file --opt **--help** **--version**

Příkazem `date` se vypisuje datum a čas. Implicitně se vypisuje systémové datum a čas v místním časovém pásmu.

```
$ date
Sun Sep 28 21:01:31 EDT 2003
```

Lze změnit výstupní tvar; zadá se formátovací řetězec uvozený znakem `%`.

```
$ date '+%D'
09/28/03
$ date '+The time is  %l:%M %p on a beautiful %A in %B'
The time is 9:01 PM on a beautiful Sunday in September
```

Tvar	Význam	Příklad
Úplné datum a úplný čas:		
`%c`	Úplné datum, úplný čas, dvanáctihodinové dělení	Sun 28 Sep 2003, 09:01:25 PM EDT
`%D`	Číselné datum, dvoumístný rok	09/28/03
`%x`	Číselné datum, čtyřmístný rok	09/28/2003
`%T`	Čas, dělení na 24 hodin	21:01:25
`%X`	Čas, dělení na 12 hodin	09:01:25 PM

Slova:		
`%a`	Den v týdnu (zkratka)	Sun
`%A`	Den v týdnu (celé jméno)	Sunday
`%b`	Jméno měsíce (zkratka)	Sep
`%B`	Jméno měsíce (celé)	September
`%Z`	Časové pásmo	EDT, MET
`%p`	AM nebo PM	PM
Čísla:		
`%w`	Den v týdnu (0-6, 0=neděle)	0
`%u`	Den v týdnu (1-7, 1=pondělí)	7
`%d`	Den v měsíci, levostr. nuly	02
`%e`	Den v měsíci, levostr. mezery	2
`%j`	Den v roce, levostr. nuly	005
`%m`	Číslo měsíce, levostr. nula	09
`%y`	Rok, 2 místa	03
`%Y`	Rok, 4 místa	2003
`%M`	Minuty, levostr. nuly	09
`%S`	Vteřiny, levostr. nuly	05
`%l`	Hodina, dělení na 12 hodin, levostr. mezery	9
`%I`	Hodina, dělení na 12 hodin, levostr. nuly	09
`%k`	Hodina, dělení na 24 hodin, levostr. mezery	9
`%H`	Hodina, dělení na 24 hodin, levostr. nuly	09
`%N`	Nanosekundy	737418000
`%s`	Počet vteřin od vzniku Linuxu: půlnoc 1. ledna 1970	1068583983
Ostatní:		
`%n`	Nový řádek	
`%t`	Tabulátor	
`%%`	Znak %	%

Volby, kterými lze zobrazit datum a čas ještě jiným způsobem.

Užitečné volby

`-d` *`datum_nebo_řetězec`* Zobrazení *`datum_nebo_řetězec`* dle vlastního návrhu

`-r` *`jméno_souboru`* Zobrazení poslední změny časové známky souboru dle vlastního návrhu

`-s` *`datum_nebo_řetězec`* Nastavení systémového data a/nebo času; pouze superuživatel.

ntpdate *časový_server* ntp

/usr/sbin stdin **stdout** -file **-opt** -help -version

Příkazem `ntpdate` se nastaví běžný čas systému podle časového serveru v síti. Tento příkaz lze zadat jen z rootu.

```
# /usr/sbin/ntpdate timeserver.někde.cz
7 Sep 21:01:25 ntpdate[2399]: step time server 178.99.1.8
offset 0.51 sec
```

K trvalému udržování systémového času podle časového serveru se používá daemon `ntpd`; viz *http://www.ntp.org*. Neznáte-li místní časový server, zadejte Googlu „public ntp time server".

Plánování prací

sleep	Čekání zadaný počet vteřin; neděje se nic
watch	Spouštění programu v zadaných intervalech
at	Naplánování práce na určitý čas
crontab	Naplánování práce na určité časy

Na plánování programů v určitém čase nebo v určitých intervalech existuje v Linuxu několik více či méně dokonalých nástrojů.

sleep *časový_údaj* coreutils

/bin stdin stdout -file -opt **-help** **-version**

Příkazem `sleep` se nastaví čekání po zvolený čas. Časovým údajem může být buď celé číslo (počet vteřin) anebo celé číslo s písmenem `s` (taktéž vteřiny), `m` (minuty), `h` (hodiny) nebo `d` (dny).

```
$ sleep 5m                    5 minut nic nedělej
```

Tento příkaz je například vhodný pro pozdržení jiných příkazů:

```
$ sleep 10 && echo 'Uplynulo deset vteřin.'
(desetivteřinová přestávka)
Uplynulo deset vteřin.
```

watch [*volby*] *příkaz* procps

/usr/bin stdin **stdout** -file **-opt** **-help** **-version**

Programem `watch` se spouští daný příkaz v pravidelných intervalech; implicitní hodnota je každé dvě vteřiny. Příkaz se předá shellu (speciální znaky tedy musí být v závorkách nebo uvedeny escape sekvencemi) a výsledky se zobrazí v režimu celé obrazovky, takže je možno pozorovat výstup a dívat se, co se změnilo. Například `watch -n 60 date` spouští příkaz `date` jednou za minutu, vzniknou tedy takové levnější hodinky. Povelem `^C` se příkaz ukončí.

Užitečné volby

`-n` *vteřiny*	Nastavení času mezi spouštěním ve vteřinách.
`-d`	Zvýraznění změn na výstupu – upozornění na změny mezi dvěma spuštěními.

at [*volby*] *časový_údaj* at

/usr/bin stdin **stdout** -file **-opt** **-help** **-version**

Příkazem `at` se v určeném čase jednorázově spustí shellový příkaz:

```
$ at 7am next sunday
at> echo Nezapomeň jít nakoupit | Pošta novak
at> lpr $HOME/seznam
at> ^D
<EOT>
job 559 at 2003-09-14 21:30
```

Specifikace časového údaje v příkazu `at` je velice flexibilní. Lze specifikovat:

- Čas a datum (nikoli naopak)
- Pouze datum (předpokládá se momentální čas)
- Pouze čas (kdy nejdříve nastane, dnes nebo zítra)
- Speciální slovo jako `now`, `midnight` nebo `teatime` (16:00)
- Cokoli z předchozího, navíc s posunem, např. „+3 days"

Datum lze zadávat v mnoha podobách: `december 25 2003`, `25 december 2003`, `december 25`, `25 december`, `12/25/2003`, `25.12.2003`, `20031225`, `today`, `thursday`, `next thursday`, `next month`, `next year` apod. Jména měsíců lze zkrátit na 3 znaky (`jan`, `feb`, `mar`, ...). Čas rovněž může mít mnoho podob: `8pm`, `8 pm`, `8:00pm`, `8:00 pm`, `20:00` a `2000` znamenají totéž. Posun se uvádí znaménkem plus nebo minus, za kterým následuje mezera a časový údaj: `+ 3 seconds`, `+ 2 weeks`, `- 1 hour` atd*.

Pokud část časového údaje nebo data chybí, doplní se ze systémového času a data. Takže „next year" znamená odteďka za rok, „thursday" znamená nejbližší čtvrtek ve stejný čas, „december 25" znamená nejbližší 25. prosinec a „4:30pm" je nejbližších budoucích 16 hodin 30 minut.

Příkaz se vyhodnocuje až při vlastním provádění, do té doby se zástupné znaky, proměnné a ostatní shellové konstrukce ponechávají beze změny. Taktéž okolí se ukládá v rámci každé práce (viz `printenv`), takže příkaz se provede, jakoby byl uživatel přihlášen. Naopak synonyma (alias) v pracích spouštěných příkazem at k dispozici nejsou, v příkazech by se tedy neměla používat.

* Přesná syntaxe je uvedena v */usr/share/doc/at-*/timespec*.