INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Příkazy v LINUXu

Zde je úmluva o označení klávesových zkratek. Stisk klávesy **CONTROL** na klávesnicích, někdy značené **CTRL,** budeme označovat C-. V systému má speciální význam klávesa **META**. Stisku **META** odpovídá stisk klávesy *levý* **ALT** nebo stisk a uvolnění klávesy **ESC.** Stisk klávesy **META** budeme označovat M-. Po sobě jdoucí stisky kláves budeme oddělovat mezerami, například C-x i znamená nejprve držet **CTRL** a stisknout x, uvolnit **CTRL** a poté stisknout i. Ačkoliv se tato notace může zdát na první pohled exotická, výrazně zjednodušuje zápis klávesových zkratek. Nebude-li to na úkor přehlednosti, bude se v dalším textu používat. V ukázkových příkladech je použit strojopis pro výstupy operačního systému a programů. Tučný strojopis je použit pro zdůraznění vstupů uživatele. Dále je dobré upozornit na fakt, že systém Unix striktně *rozlišuje velká a písmena,* jednak na úrovni systémových příkazů, ale i u jmen souborů, uživatelských jmen a klávesových zkratek. Tato vlastnost se nazývá *case-sensitivity.* Citlivost na velká a malá písmena již dále nebudeme zdůrazňovat, je ale nezbytné mít ji neustále na paměti.

V následujícím textu jsou stručně popsány základní kroky uživatele v systému. Předpokládejme, že se uživatel *Jan Novák* úspěšně přihlásil do systému na počítači phoenix pod uživatelským jménem novakj. Uživatel po přihlášení obdrží zhruba následující výpis

```
login: novakj
Password:
Last login: Mon Jun 11 11:35:28 2001 from titan on pts/3
Linux phoenix 2.4.5 #1 SMP Mon May 28 08:38:07 CEST 2001 I686
You have new mail.
$
```

První řádek uživatele informuje o jeho posledním přihlášení do systému. V našem případě se uživatel naposledy hlásil 11. června 2001 z počítače titan. Druhý řádek je verze 2.4.5 s podporou symetrického multiprocesingu (zkratka SMP). Z informací v druhém řádku lze rovněž vyčíst, kdy bylo jádro sestaveno a na jaké platformě systém funguje, v našem případě se jedná o Intel i686. Za druhým řádkem se mohou objevit dodatečné informace od správce systému. Jedná se zpravidla o důležité informace zveřejněné pro všechny uživatele. Například během reinstalace systémových programů se nemusí všechny programy chovat korektně. Správce systému může na tuto skutečnost upozornit pomocí veřejného souboru *„Message Of The Day"*, jehož obsah se vypíše každému uživateli po přihlášení do systému. Na třetím řádku

systém upozorňuje uživatele, že má ve své poštovní schránce nepřečtenou poštu. Pokud je poštovní schránka prázdná, systém tuto skutečnost buď nebude nijak ohlašovat, nebo vypíše hlášení „No mail". Jakmile se uživatel přihlásí, automaticky se spustí uživatelův *přihlašovací příkazový interpret - login shell.* Informace o přihlašovacím příkazovém interpretu jsou přímo součástí uživatelského účtu a uživatel je nemůže svévolně měnit. Interpret příkazů čeká na posloupnost příkazů od uživatele, tuto skutečnost ohlašuje *prompt.* Prompt je *posloupnost symbolů*, kterou vypisuje příkazový interpret. Jeho konkrétní tvar si může uživatel dle libosti upravit, v zásadě ale platí, že prompt je ukončen znakem „$“ pokud jde o běžného uživatele, nebo „#", pokud jde o superuživatele root. V naší ukázce je prompt tvořen pouze znakem „$", za nímž následuje kurzor. Tvar promptu lze upravit. Z uživatelského hlediska je například praktické umístit si do prompt informaci o aktuálním adresáři. Zadávání příkazů v shellu je přímočaré. Každý příkaz je zpracován shellem poté, co uživatel ukončí jeho vkládání klávesou **ENTER**. Jednotlivé příkazové interprety se liší mírou pohodlí, kterou poskytují uživateli při editaci příkazů. *GNU Bash* je velmi vyspělý, obsahuje například historii příkazů, kterou lze jednoduše prohledávat. Základní klávesové zkratky pro ovládání shellu GNU Bash jsou v následující tabulce.

klávesa	*význam*
C-a	skoč na začátek řádku
C-e	skoč na konec řádku
C-d	smaž znak na pozici kurzoru
C-u	smaž znaky od začátku řádku do pozice kurzoru
C-k	smaž znaky od pozice kurzoru do konce řádku
C-t	zaměň poslední dva znaky a posuň se vpravo
C-r	hledej v historii příkazů
C-_	krok zpět – undo
C-y	vlož smazaný text na pozici kurzoru
C-l	vyčisti obrazovku
M-<	najdi poslední příkaz
M->	najdi první příkaz
M-l	převeď slovo na malá písmena
M-u	převeď slovo na velká písmena

Veškeré příkazy zadávané v shellu musí být opět chápány jako *case-sensitive.* Jakmile je zadán příkaz pro ukončení příkazového interpretu, sezení uživatele je ukončeno. Příkazový interpret lze ukončit stiskem C-d na *prázdném řádku,* stejný efekt má příkaz exit. Interpret GNU Bash má rovněž příkaz logout, ten na rozdíl od příkazu exit ukončuje pouze přihlašovací shell. Uživatel totiž může ze svého přihlašovacího shellu spustit další shell. Tento „vnořený" shell lze ukončit příkazem exit nebo

klávesou C-d, ale nelze jej ukončit příkazem logout. Systém by při pokusu použít logout ve vnořeném shellu vypsal chybové hlášení

$ logout
bash: logout: not login shell: use 'exit'

Během práce se uživatel může setkat se zdánlivě netypickým chováním interaktivních programů, například klientů na čtení pošty nebo textových editorů. Problémem může být *nastavení emulace terminálu.* Se špatným nastavením emulace terminálu se lze setkat například při přihlášení na vzdálený systém nepodporující používaný typ terminálu. Nastavením proměnné prostředí TERM lze vybrat přijatelný terminál. Například pomocí příkazu

$ export TERM="vt100"

je nastavena proměnná TERM na hodnotu vtl00, což je označení pro standardní terminál, který by neměl dělat potíže v žádném systému. Proměnnou TERM používá systémové knihovna *ncurses* starající se o výstup na terminál. Součástí knihovny je i databáze terminálů. Uživatel by měl vždy nastavit některý z terminálů dostupných v databázi systému.

Na začátku bylo konstatováno, že Unix je *tichý.* Tato vlastnost se projevuje nejvíc při práci s příkazovým interpretem. Pokud příkaz nebo program neplní informativní účel, chová se implicitně tiše. Pouze v případě chyby vypíše hlášení. Stejně tak se chová i sám interpret příkazů.

Mezi nejčastěji používané příkazy shellu patří *příkaz spuštění programu.* Pomocí něj může uživatel spouštět další externí programy a využívat jejich služeb. Syntaxe příkazu spuštění programu je jednoduchá. Příkaz sestává ze slov oddělených mezerami. Řečeno přesně, oddělovačem slov je *whitespace* - alespoň jednoprvková sekvence mezer anebo znaků tabulátor. První slovo v příkazu se musí shodovat buďto se jménem *vestavěného příkazu interpretu,* nebo se musí shodovat se jménem *spustitelného souboru* umístěného na cestě. V prvním případě je vykonán vnitřní příkaz shellu. V druhém případě je spuštěn externí program daného jména. Pokud by první slovo příkazu nesplňovalo žádnou z předchozích podmínek, shell by vypsal chybové hlášení.

Je-li v těle příkazu přítomno více slov, označujeme je jako argument. Pod pojmem přepínač (parametr) je zpravidla myšlen speciální argument začínající znakem „-„. Přepínače mají zvláštní význam - mění chování spuštěných programů. Nejprve ukažme spuštění programu bez dodatečných argumentů.

$ who

```
pes      pts/0    Jun  11   11:21  (bouda.upol.cz)
jezekm   pts/1    Jun   7   18:14  (thunder.upol.cz)
pes      pts/3    Jun  11   11:59  (bouda.upol.cz)
novakj   pts/5    Jun  11   16:15  (titan.inf.upol.cz)
polakr   ttyp0    Jun  11   11:34  (158.194.92.55)
```

Jelikož byl program who spuštěn bez argumentů, provedl výpis přihlášených uživatelů v systému. Výpis sestává z informací o uživatelském jménu, čísle terminálu, času přihlášení a doménovém jménu uzlu, z něhož se uživatel přihlásil. Program who lze spustit i s argumenty

```
$ who am i
phoenix!novakj   pts/5   Jun 11   16:15   (titan.inf.upol.cz)
```

V příkladu byly předány dva argumenty - „am" a „i". Podotkněme, že tyto argumenty nejsou samostatné programy, jejich zpracování je plně v režii spuštěného programu. V tomto případě je to program who. Předchozí volání programu who slouží k identifikaci uživatele, který je přihlášen v systému na konkrétním terminálu. Pokud používáte jeden účet a jednoho hostitele, může se vám tento program zdát nadbytečný. Na druhou stranu při současném používání více účtů na různých hostitelských počítačích může předchozí volání who usnadnit orientaci ve změti terminálů.

Jméno přihlášeného uživatele lze vypsat i jinak, například pomocí programu whoami. Jak již, ale z názvu programu vyplývá, jedná se o zcela jiný program, než je who.

```
$ whoami
novakj
```

Program whoami nevypisuje počítač, na kterém probíhá sezení ani další podrobné údaje. Na tomto příkladu je dobré uvědomit si, že who a whoami jsou dva různé programy, které spolu nijak nesouvisejí. I když se nám volání „who am i" a „whoami“ mohou zdát podobná.

Detailní popis uživatelů přihlášených v systému lze získat pomocí programu w

```
$ w
4:26pm  up 10 days,  7:53,  6 users,  load avg:  0.00,  0.00,  0.00
USER   TTY     FROM              LOGIN®   IDLE   JCPU   PCPU   WHAT
pes    pts/0   bouda.upol.cz     11:21am  10:45  0.10s  0.00s  bash
jezekm pts/1   thunder.upol.cz   Thu 6pm  2:54m  0.12s  0.12s  bash
```

Na prvním řádku výpisu jsou údaje o běhu systému, tyto informace lze získat i pomocí programu uptime. Na dalších řádcích jsou informace o uživatelích. Zajímavý údaj je IDLE reprezentující dobu od poslední odezvy uživatele. Údaje JCPU, PCPU udávají čas běhu procesů uživatele. Druhý z údajů zahrnuje i procesy běžící na pozadí. Poslední sloupec výpisu reprezentuje poslední spuštěný program.

Informativním program v systému Unix je date. Program date vypisuje systémový čas. Čas je v Unixu reprezentován počtem sekund, které uběhly od půlnoci 1. ledna 1970. Program date bez argumentů vypisuje čas v klasické podobě včetně informací o časovém pásmu. Česko spadá do pásma CET - *Central European Time.* Počítače s operačním systémem Unix si zpravidla uchovávají světový čas GMT

- *Greenwich Mean Time,* jiná časová pásma vypočítávají podle databáze časových pásem, kterou mají systémy k dispozici. Pomocí programu date lze vypisovat časové údaje v rozličných formátech. Následující příklady ukazují možné použití date pro výpis údajů o čase, poslední z ukázek je vnitřní unixová reprezentace.

```
$ date
Tue  Jun  12  09:59:02  CEST 2001
$ date +%d.%m.%Y
12. 06. 2001
$ date  +%s
992332681
```

Poslední základní potřebou každého uživatele, je změna přístupového hesla. Na úvod podotkněme, že pro změnu hesla lze použít dva programy, passwd a yppasswd. Druhý z nich se používá v systémech využívajících sdílení uživatelských účtů pomocí služby NIS. Při změně hesla pomocí passwd a yppasswd jde z uživatelského hlediska o týž proces. Nejprve je uživatel dotázán na dosavadní heslo. Poté musí dvakrát zadat nové heslo, tím se předchází nechtěným překlepům. Během změny hesla se samozřejmě nevypisují žádné znaky na obrazovku

```
$ passwd
Changing  password  for  novakj
Old  password:
New password:
Retype  new  password:
Password  changed.
```

Systémová dokumentace

Z počátku se může uživatel při práci s příkazovým interpretem cítit bezradný. Systém navíc obsahuje velké množství obslužných programů a dalších nástrojů. V systému jsou naštěstí téměř všechny dostupné nástroje systematicky dokumentovány. Uživatelům je k dispozici hned několik prohlížečů dokumentace. Základní informace o obslužných programech, ale třeba i o funkcích programovacích jazyků či o voláních jádra lze získat pomocí *manuálových stránek.* Jde o elektronickou podobu fulltextové nápovědy, kterou je možné prohlížet pomocí programů apropos , whatis a man. Alternativou k manuálovým stránkám je hypertextový dokumentační systém GNU Info.

Celou dokumentaci k zadané frázi lze prohlížet pomocí programu **man**.

```
$ man man
```

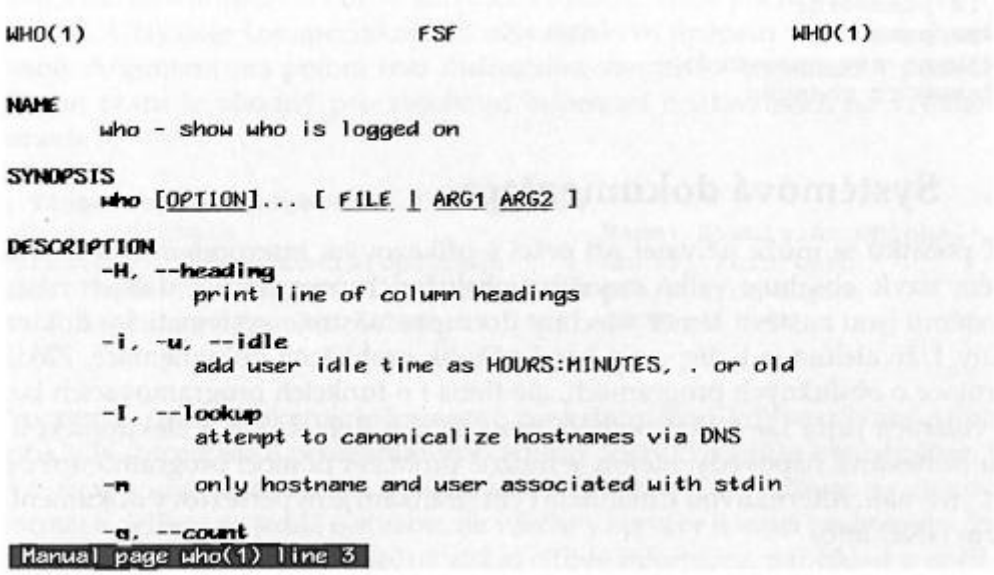
Zobrazí se celá nápověda.

```
$ man printf
Reformatting printf(1), please wait...
```

• • • *zde bude stránka zobrazena* • • •

$ man 3 printf
Reformatting printf(3), please wait...
• • • *zde bude stránka zobrazena* • • •

V prvním případě se po dokončení formátování zobrazí stránka ze sekce 1, vždy se přednostně zobrazují stránky ze sekcí s menším číslem. V druhém případě si uživatel explicitně vynutí zobrazení stránky ze sekce 3. Na následujícím obrázku je ukázka části manuálové stránky k programu who.



Manuálové stránky jsou velmi jednoduché a vesměs se drží základní struktury. Téměř každá stránka je rozdělena do částí oddělujících logické celky dokumentu. Manuálové stránky mají definovány *jméno* a *stručný popis.* Další oddíly jsou volitelné a do jisté míry souvisejí s dokumentovaným programem, službou, příkazem nebo funkcí. Například u knihovních funkcí programovacího jazyka C je zvykem uvádět oddíl „CONFORMING TO" - oddíl uvádí standardy zahrnující definici dané funkce.

Hypertextová dokumentace Info

Klasické manuálové stránky bohužel kromě odkazů v oddílu „SEE ALSO" neobsahují žádné odkazy na související dokumenty ani jednotně nedefinují odkazy v rámci jednoho dokumentu. Tato vlastnost nevadila v době jejich vzniku, kdy s počítači pracovali pouze skuteční odborníci a dokumentace k systémovým nástrojům nebyla nikterak rozsáhlá. V současnosti však pro systémy odvozené z Unixu existuje nesčíslně softwarových balíků a jejich dokumentace si vyžádala nový přístup.

V rámci projektu GNU byl vyvinut hypertextový dokumentační systém GNU Info. Dokumenty pro GNU Info jsou začleněny do hierarchie s jedním počátečním dokumentem - *hlavní stránkou dokumentace.* Hlavní stránka slouží v dokumentaci jako „rozcestník", jsou v ní uvedeny odkazy na další dokumenty. Dokumenty na nižších úrovních hierarchie mohou opět obsahovat odkazy buďto v rámci vlastního dokumentu, nebo na ostatní dokumenty.

Systém nápovědy GNU Info je možné prohlížet například pomocí programu **info**. Ve většině případů je program spouštěn bez dodatečných argumentů. Program info pracuje v celoobrazovkovém režimu a po svém startu zobrazí hlavní stránku nápovědy. Předpokládejme, že uživatel spustil program **info**. Na obrazovce může vidět například následující výpis.

```
* Info: (info).      Documentation browsing system.
* Emacs: (emacs).    The extensible self-documenting text editor.
* Gnus: (gnus).      The news reader Gnus.
* CVS: (cvs).        Concurrent Versions System
* GCC: (gcc).        The GNU Compiler Collection.
```

V dokumentech pro GNU Info jsou znakem hvězdička „ * " uvozeny odkazy na dokumenty. Hlavní stránka dokumentace je zpravidla tématicky rozdělena a odkazy na další dokumenty jsou zde uvedeny v přehledných seznamech. Samotné odkazy jsou doplněny i krátkým popisem, jak je tomu i v předchozí ukázce. Pokud si chce uživatel přečíst některý z dokumentů, musí následovat příslušný odkaz. Uživatel se například může kurzorovými šipkami přesunout na odkaz k dokumentu a stisknout klávesu ENTER . Ovládání programu info je v některých případech dost nezvyklé. Následující tabulka shrnuje základní klávesové zkratky.

klávesa	*význam*
q	ukončení prohlížeče
u	přechod na nadřazený uzel nápovědy
p	přechod na předcházející uzel nápovědy
n	přechod na následující uzel nápovědy
m	najdi položku v menu podle jména
C-h	nápověda
C-s	hledej dopředu
C-r	hledej dozadu

Mezi nejdůležitější klávesy patří právě u, p a n. Jedním stiskem klávesy u se uživatel dostane o úroveň výš. Pokud si v našem případě uživatel začne číst dokumentaci k programu GNU Emacs a rozhodne se přečíst si dokumentaci k překladači GCC, musí se nejprve vrátit na hlavní stránku dokumentace. Tak lze učinit právě stisk klávesy u. Pokud se uživatel v dokumentaci „ztratí", vždy se pomocí několika stiknutí klávesy u dostane na hlavní stránku dokumentace. Klávesy pak slouží k pohybu mezi částmi dokumentu pouze *v rámci jedné úrovně.*

$ cal

Vypíše se kalendář – aktuální měsíc.

$ cal m rok

Vypíše se kalendář – měsíc a rok (např. cal 8 2011)

$ cal rok

Vypíše se kalendář na rok (např. cal 2011)

$ clear

Příkazem se vymaže obrazovka nebo shellové okno.

$ history

Zobrazí výpis historie příkazů.

$ history *n*

Zobrazí výpis posledních *n* příkazů z historie.

$ history -c

Vymaže historii.

$ free

Vypíše se využití operační paměti v kB.

$ df

Vypíše se velikost, využitý prostor a volný prostor v diskové oblasti. Zadá-li se soubor nebo adresář, příkaz sdělí, na kterém diskovém zařízení je uložen. Jinak se vypíše zpráva o všech připojených souborových systémech.

- Práce s adresáři a soubory

$ pwd

Příkazem se vypíše absolutní cesta běžného pracovního adresáře.

$ cd

Příkazem se mění pracovní (běžný) adresář.

$ cd adresář

Příkazem se nastaví pracovní (běžný) adresář.

$ cd

Příkazem se nastaví domovský adresář.

$ cd ..

Příkazem se nastaví nadřízený adresář.

$ cd /

Příkazem se nastaví kořenový adresář.

$ mkdir

Vytvoření adresáře.

$ mkdir adresář1 adresář2 adresář3

Příkazem se vytvoří jeden nebo více adresářů.

$ rmdir

Vymazání prázdného adresáře.

$ rmdir adresář

Příkazem se vymaže jeden nebo několik prázdných adresářů.

$ rm –r

Vymazání neprázdného adresáře.

$ rm –r adresář

Příkazem se vymaže neprázdný adresář včetně jeho obsahu.

$ ls

Příkazem se zobrazí jména souborů z běžného adresáře.

$ ls –l

Příkazem se zobrazí jména atributy souborů z běžného adresáře (dlouhý, úplný výpis).

$ ls -a

Příkazem se zobrazí i skryté soubory z běžného adresáře (začínají tečkou).

$ du

Příkazem se zjistí velikost diskového prostoru obsazeného soubory a adresáři.

$ du

Příkazem se zobrazí velikost všech adresářů a podadresářů.

$ du soubor

Příkazem se zobrazí velikost souboru.

$ cp

Příkazem se kopíruje soubor.

$ cp soubor1 soubor2

Příkazem se kopíruje soubor1 do souboru2.

$ mv

Příkazem se přejmenuje soubor anebo se přesouvají soubory do jiného adresáře.

$ mv soubor1 soubor2

$ mv soubor1 nový_ adresář

$ rm

Příkazem se ruší soubor.

$ rm –i soubor

Příkaz se interaktivně dotáže, zda se smí smazat soubor.

$ cat

Příkazem se vypíše obsah souboru najednou.

$ cat –n soubor

Příkaz vypíše obsah souboru a očísluje řádky.

$ cat > soubor

Příkaz vytvoří nový soubor, text zadáváme z klávesnice, ukončíme Ctrl-D.

$ cat >> soubor

Příkaz připojí na konec souboru text zadaný z klávesnice.

$ less

Příkazem se vypíše obsah textového souboru po obrazovkách.

$ more

Příkazem se vypíše obsah textového souboru po obrazovkách.
(ENTER – posun o řádek, mezerník – posun o obrazovku, q – ukončení výpisu)

$ touch

Příkazem se změní hodnota dvou časových známek v souboru (modifikace a přístup).

$ touch soubor

Pokud soubor neexistuje, vytvoří se prázdný soubor.

$ file

Příkaz zjistí typ souboru.

$ file *

Zjistí typ všech souborů v pracovním adresáři.

$ find

Příkaz, s jehož pomocí můžeme bez problémů vyhledat v systému námi požadovaný soubor nebo více souborů, které odpovídají specifikovaným požadavkům.

$ find / -name passwd -print

Po zadání příkazu ve specifikovaném tvaru začne interpret prohlížet celý systém a vyhledávat v něm zadaný soubor. Jakmile jej objeví, parametr -print způsobí, že tuto informaci vypíše i s plnou cestou k jeho umístění.

$ head

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

$ head soubor -*N*

Výpis prvních *N* řádků v souboru.

$ head soubor1 soubor2 -q

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

$ tail

Příkazem tai1 se vypíše 10 posledních řádků v souboru.

$ tail soubor -*N*

Výpis posledních *N* řádků v souboru.

$ sort

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

$ mount

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

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

$ umount

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

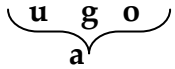
Příkazem fsck („filesystem check") se kontroluje disková oblast a opravují se v ní případné chyby. Tento příkaz probíhá automaticky při spouštění systému; lze jej však zadat i „ručně". Obecně by před zadáním tohoto příkazu mělo být kontrolované zařízení odpojeno, aby s ním v průběhu kontroly nepracovaly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

```
# 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
```

- **Přístupová práva**

Přístupová práva jsou v objektu uložena v 9 bitech.

Ve výpisu **ls –l:**
- rwx r-x r-x 1 novák student ... lze oktalově zapsat **7 5 5**



vlastník (označuje se **u** jako 'user') Vlastníkem je ten, jehož UID je zapsáno v i-uzlu.
skupina (označuje se **g** jako 'group')
ostatní (označuje se **o** jako 'others')
všichni (označuje se **a** jako 'all')

Přístupová práva souboru:
r - Soubor je povoleno číst (read).
w - Do souboru je povoleno zapisovat (write).
x - Soubor je povoleno spustit (provést) – execute.

Přístupová práva adresáře (d):
r - Adresář je povoleno vypsat; nikoli však zpřístupnit soubory v něm odkazované.
w - Do adresáře je povoleno zapisovat; tj. lze vytvářet a rušit soubory.
x - Do adresáře je možné vstoupit; tj. adresář může být argumentem příkazu **cd** a lze zpřístupnit i-uzly souborů, na které se adresář odkazuje.

$ chmod

Nastavení přístupových práv vlastníkem souboru.
1) absolutně – oktalově
    $ chmod 777 soubor
Všichni uživatelé mají všechna přístupová práva.
    $ chmod 664 soubor
Vlastník a skupina mají právo čtení a zápisu, ostatní mají pouze právo čtení.
2) symbolicky - '+' přidávají se práva, '–' ubírají se práva, '=' absolutní nastavení.
    $ chmod ug+rw soubor
Vlastníkovi a skupině bylo přidáno právo čtení a zápisu.

$ umask

Příkazem se nastaví implicitně přístupová práva pro nově vytvářené soubory a adresáře. Přístupová práva se nastavují inverzně.

$ umask

Zobrazí implicitní nastavení přístupových práv.

$ umask 022

Nastaví veškerá práva pro vlastníka souboru, ostatní uživatelé budou mít právo číst/spouštět.

$ chown

Příkaz pro změnu vlastníka souboru a adresáře.

$ chown soubor

Příkaz v některých verzích nelze provést.

$ chgrp

Příkaz pro změnu vlastníka souboru a adresáře. Příkaz v některých verzích nelze provést.

- Zálohování a komprimace souborů

$ gzip

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

compress a uncompres komprimují a dekomprimují soubory do/z standardního unixového tvaru. 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 souboru.

$ bzip2

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 souboru.

$ zip

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 rekurzivně
unzip -l mujsoubor.zip                       Výpis obsahu
unzip mujsoubor.zip                          Rozbalit
```

- Příkazy pro práci s procesy

$ ps

```
PID   TTY  TIME    CMD
14611 pts/O 00:00:00 bash
15694 pts/O 00:00:00 ps
```

Předchozí výpis není nijak překvapující. Prvním z procesů je aktivní interpret příkazů, druhý proces je samotný program ps. První sloupec označuje PID procesů. Druhý sloupec reprezentuje aktuální terminál. Třetí sloupec reprezentuje celkový čas běhu procesu. V předchozím případu jsou časy procesů nulové, protože ps proběhne okamžitě a bash tráví většinu života čekáním na uživatelský vstup. Podrobnější informace o době života procesu lze zjistit například pomocí příkazu „**time“**. Poslední sloupec je shellovský příkaz, který způsobil vytvoření procesu.

```
drimalkp 12925 12311 0 13:18 pts/4  00:00:00 ssh pocal
hropkop  16760 16747 0 15:56 pts/2  00:00:00 talk novakj
novakj   16764 14731 0 15:56 pts/6  00:00:00 talk hropkop
novakj   19081 17762 0 17:10 pts/3  00:00:00 ps -a -f
```

Ve výpisu uživatel nalezne kromě již uvedených údajů sloupec STIME -- systémový čas vzniku procesu. Za údajem PID je uveden PPID - *Parent Process Identification,* to jest PID rodičovského procesu. Údaj C reprezentuje procentuální vytížení procesoru.

$ ps -a

```
  PID TTY          TIME CMD
12925 pts/4    00:00:00 ssh
16624 pts/1    00:00:00 bash
16629 pts/1    00:00:00 mutt
16760 pts/2    00:00:00 talk
16764 pts/6    00:00:00 talk
17616 pts/0    00:00:00 ps
```

Přepínač -a zobrazí informace o všech aktivních procesech řízených nějakým terminálem.

$ ps -u novakj

```
  PID TTY          TIME CMD
14611 pts/0    00:00:00 bash
14731 pts/6    00:00:00 bash
16764 pts/6    00:00:00 talk
17617 pts/0    00:00:00 ps
```

Přepínač -u zobrazí informace o všech aktivních procesech daného uživatele.

$ ps -a -1

```
  F S   UID   PID  PPID C PRI  SZ  WCHAN TTY        TIME CMD
000 S   903 12925 12311 0  69 608 13e25e pts/4 00:00:00 ssh
000 S  1382 16760 16747 0  69 400 13e25e pts/2 00:00:00 talk
000 S  1045 16764 14731 0  69 400 13e25e pts/6 00:00:00 talk
000 R  1045 19080 17762 0  79 616      - pts/3 00:00:00 ps
```

Dlouhý výpis navíc zobrazuje ve sloupci F masku procesu a ve sloupci S *stav procesu*. Zobrazované stavy mohou nabývat hodnot S - *spící proces*, R - *běžící proces* a T - *pozastavený proces.* Dále jsou zobrazeny informace o počtu bloků zabraných obrazem programu - SZ. Sloupec WCHAN obsahuje informace o akci, na kterou proces čeká. U běžícího procesu je tato položka prázdná. Je-li proces ve stavu spánku, pole je vyplněno hodnotou akce.

$ ps -ax

Příkaz vypíše informace o všech procesech.

$ ps -u novakj -o pid,ppid,s,cmd

```
  PID  PPID S CMD
14501 14500 S -bash
14702 14701 S -bash
14710 14702 S talk jezekm
14812 14501 R ps -u novakj -o pid,ppid,s,cmd
```

Za argument - o lze psát seznam sloupců oddělených čárkami. Seznam nesmí obsahovat mezery. Názvy jednotlivých sloupců lze najít v manuálové stránce programu ps. Příkaz ps má více než 80 voleb.

Zobrazení vlastních procesů

$ ps -ux

Dalším programem zobrazujícím procesy je top,

$ top

Program top je interaktivní. Po svém spuštění top běží, dokud není zastaven uživatelem. Program top pracuje v celoobrazovkovém režimu a periodicky zobrazuje informace o běžících procesech. Běh programu lze ukončit klávesou q, po stisku h je zobrazena nápověda programu top. Výpis běžících procesů lze třídit podle *spotřeby paměti, vytížení procesoru, stáří procesu* a podobně.

$ uptime

Příkazem zjistíme, jak dlouho systém běží od svého spuštění.

$ kill

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.

Úplný přehled signálů, jež lze zadat příkazem ki11, získáme pomocí

$ kill -1

$ killall

Program killall rovněž zasílá signály, ale procesy nejsou identifikovány svým PID, nýbrž svým jménem. To má své výhody i nevýhody. Neopatrné použití může vést k nechtěnému ukončení více procesů se stejným jménem. Na druhou stranu tento program poskytuje vyšší uživatelský komfort.

$ nice

Příkazem nice můžeme zadat programu prioritu při spouštění.

Prioritu procesu lze zjistit příkazem nice bez parametru:

$ nice

Příklad nastavení „velké práce" na prioritu 7:

$ nice -7 sort soub_VelkePrace > vyst_soub

Pokud se neuvede priorita, nastaví se automaticky na 10.

Superuživatel může i zvyšovat prioritu (snížením jejího čísla):

$ nice --10

$ renice

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 systémovými procesy.

$ watch

Spouštění daného příkazu v pravidelných intervalech (implicitní hodnota je každé dvě vteřiny).

$ watch příkaz

$ watch –n 60 date

Spouští příkaz date jednou za minutu (^C – ukončení příkazu).

$ at

Příkazem se v určeném čase jednorázově spustí shellový příkaz (ukončení zadávání příkazu ^D).

$ at čas datum –f soubor

V určeném čase se spustí příkazy uložené v souboru.

Následující tabulka obsahuje příklady specifikace času pro program at.

specifikace	*význam*
9:45 26.3.2009	dne 26. 3. 2009 v 9:45 hodin
tomorrow +1 hour	zítra o hodinu později
+2 weeks	za dva týdny
8:30 +3 days	za tři dny v 8:30

Po spuštění programu s korektním časovým intervalem je uživatel vyzván k zadání příkazů. Po zadání posledního příkazu musí uživatel ukončit vstup klávesou C-d. Viz příklad.

```
$ at 11:30 26.4.2002
warning: commands will be executed using /bin/sh
at> ls -la ~
at> CTRL-D
job 21 at 2002-04-26 11:30
```

V ukázce je vidět, že program at používá vlastní prompt „at>". Důvodem je odlišení programu at od shellu. Na posledním řádku program at oznamuje, že příkaz ls -la ~ bude proveden 26. 4. 2002 v 11:30 hodin. Pokud má program nějaký výstup, jako v tomto případě, bude výsledný výstup zaslán uživateli poštou. Kromě informace o čase obsahuje poslední řádek i *identifikátor úlohy*. Identifikátor úlohy je číslo. V žádném případě jej ale nelze chápat jako PID. Proces v době naplánování ještě neběží, je pouze vytvořen záznam o jeho spuštění. Záznam je trvale uložen na disku. O spuštění procesu se stará daemon.

$ atq

Příkaz zobrazí seznam všech naplánovaných procesů.

```
559 2003-09-14 07:00 a novak
```

$ atrm číslo procesu v seznamu

Příkaz odstraní proces ze seznamu.

```
$ atrm 559
```

$ crontab

Příkazem se v určeném čase opakovaně spustí shellový příkaz. K tomuto účelu se musí 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.

$ 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_ jméno* 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. Muže zde být samostatné číslo (30), posloupnost čísel oddělených čárkami (0,15,30,45). rozsah (20-30), posloupnost rozsahu (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 ve 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, raon, 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 $H0ME. 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.
Příklad:

Předpokládejme, že uživatel novak j nastavil plánovač na následující hodnoty:

```
$ cat ~/cron-my.tab
# moje akce
5 0 * * *                       ~/prvni
0,10,20,30,40,50 14 1 * *       ~/druhy
15 4 * * sun                    ~/treti
```

První řádek v ukázce začíná znakem hash „#" a jde tedy o komentář. Druhý řádek definuje spuštění -/prvni každý den, pět minut po půlnoci. Na třetím řádku definováno spuštění -/druhy každý první den v měsíci, ve 2 hodiny odpoledne každých 10 minut. Konečně na posledním řádku je definováno spuštění -/treti každou neděli, patnáct minut po čtvrté hodině ranní.

$ bg

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

$ fg

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
```

- Příkazy pro práci v síti

$ **uname**

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 (seiver.example.com), verzi jádra (2.4.I8:27.8.0 #1 Fri Mar 14 06:45:49 EST 2003), jméno hardwaru (i686), typ procesoru (i686), hardwarovou platformu (¡386) 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**

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
myhcst
```

Superuživatel může nastavit jméno počítače:

```
# hostname pomeranč
```

Ovšem jména hostitelských počítačů jsou složité téma a běžný uživatel by raději počítačům přidělovat jména neměl zkoušet.

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.*

$ logname

Příkazem logname se vypíše přihlašovací jméno uživatele. Funkce vypadá triviálně, avšak využívá se v shellových skriptech.

```
$ logname
novak
```

$ id

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)
```

$ users

Příkazem users se vypíše rychlý seznam přihlášených uživatelů. Má-li některý uživatel spuštěn, několik shellů, vypíší se všechny.

```
$ users
novak plch bures bures bures
```

$ finger

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      Jiři 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.

$ last

Příkazem last se vypíše průběh přihlašování 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.

$ ifconfig

Příkazem ifconfig se zobrazují a nastavují vlastnosti síťového rozhraní.
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

$ host

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
```

$ host -a

Výpis všech dostupných informací.

$ whois

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.

$ ping

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
```

$ ping google.com -n

Výpis IP adres počítačů, nikoli jmen.

$ traceroute

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.cblma.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 mujhost 120).

$ traceroute yahoo.com -n

Numerický režim: vypisuje se IP adresa, nikoli jméno.

$ ssh

Příkazem ssh („Secure Shell") se uživatel bezpečně přihlásí k existujícímu účtu na vzdáleném počítači:

$ ssh vzdaleny.prik1ad.cz

Anebo lze vzdálenému počítači zadat příkaz, aniž bychom byli přihlášení:

$ 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.

$ telnet

Příkaz telnet slouží k přihlášení k existujícímu účtu na vzdáleném počítači.

$ telnet vzdaleny.prik1ad.cz

Používání příkazu se nedoporučuje, protože se jedná o nezabezpečené přihlášení.

$ scp

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:
```

$ sftp

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

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.

$ mutt

mutt je textově orientovaný poštovní program, který se spouští na obyčejném terminálu (nebo v terminálovém okně), lze jej používat jak lokálně, 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.

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	Kopirováni obsahu schránky do jiné schránky.
d	Vynechání běžné zprávy.

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.

Ostatní příkazy mutt

Klávesa	Význam
?	Výpis všech příkazů (mezerou se posouvají dolů, q se ukončí)
^C	Zrušení probíhajícího příkazu
q	Konec

$ mail

Program mail (anebo Mai1) 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.
Ukončim ji samostatnou tečkou na řádku.
.
Cc: plch@priklac.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
```

Všimněte si, jak snadno lze odeslat výstup roury jako e-mailovou zprávu.

$ mozilla

Mozilla je jedním z nejpopulárnějších internetových prohlížečů; lze ji provozovat i na jiných systémech včetně X-window. V pozadí se spustí příkazem:

$ mozilla &

$ wget

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.

$ talk

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

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]

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.

$ mesg

Příkazem mesg bez parametrů se zjišťuje stav (ano či ne).
S novějšími programy příkaz mesg nespolupracuje.

```
$ mesg
is n
$ mesg y
```

$ tty

Programem tty se vypíše jméno terminálu patřícího běžnému shellu.

```
$ tty
/dev/pts/4
```

$ echo

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.

Použité zdroje informací:

[1] SKOČOVSKÝ,L. Principy a problémy operačního systemu UNIX. 1.vyd. Brno, Science, 1993. 288s. ISBN 80-901475-0-X.
[2] SIEVER,E. a kol.Linux v kostce: Pohotová referenční příručka. 1.vyd. Praha, Computer Press, 1999. 560s. ISBN 80-7226-227-0.
[3] MCCARTY,B. Učíme se RedHat Linux. 1.vyd. Praha, Computer Press, 2000. 292s. ISBN 80-7226-227-7.
[4] SOBELL,M.G. Mistrovství v Linuxu. 1.vyd. Brno, Computer Press, 2007. 878s. ISBN 978-80-251-1726-2.
[5] BRANDEJS,M. Linux: Praktický průvodce. 2.vyd. Brno, Konvoj, 2003. 312s. ISBN 80-7302-050-5.
[6] LASSER,J. Rozumíme Unixu. 1.vyd. Brno, Computer Press, 2002. 252s. ISBN 80-7226-706-X.
[7] VYCHODIL,V. Operační system Linux: Příručka českého uživatele. 1.vyd. Brno, Computer Press, 2003. 260s. ISBN 80-7226-333-1.
[8] KYSELA,M. Přecházíme na Linux. 1.vyd. Brno, Computer Press, 2003. 190s. ISBN 80-7226-844-9.

[9] KYSELA,M. a kol. 333 tipů a triků pro Linux. 1.vyd. Brno, CP Books, 2005. 168s. ISBN 80-722-6866-X.
[10] BARRETT,D.J. Linux: Kapesní přehled. 1.vyd. Brno, Computer Press, 2006. 160s. ISBN 80-251-0838-4.