Príkazový riadok a jeho príkazy

Obsah

Súborový systém....1
- Zmena pracovného adresára....1
- Výpis obsahu adresára....2
- Práca s adresármi....4
- Práca so súbormi....7
- Práca so súborovým systémom....8
- Pripojenie a odpojenie súborového systému....9
- Odkazy....9
- Nastavenie vlastníka a skupiny....10
- Zmena módu prístupu....11
- Riadenie prístupu pomocou ACL....12
- Zdieľanie diskového priestoru....13

Systém procesov....13
- Informácie o procesoch v systéme....13

Základy príkazové interpretu....18
- Hľadanie súborov....18
- Oddeľovanie príkazov....20
- Presmerovanie vstupu a výstupu....21
- Zreťazenie obslužných programov pomocou „rúry“....22
- Používanie neúplných mien súborov....23
- Premenné....24
- Premenné špeciálneho významu....25
- Oddeľovače podmieneného vykonávania príkazov....25
- Podmienené príkazy....25
- Cykly....26
- Základné programy (ping, tracepath)....31
- Zisťovanie IP adries a symbolických mien....32
- Synchronizácia času....32
- Zdieľanie diskového priestoru....33
- Posielanie správ užívateľovi....33
- Vzdialené prihlasovanie....33
- Prenos dát pomocou FTP....34
- Služba Secure Shell (ssh)....36
- Komunikácia....37

Súborový systém

Vypracoval: Branislav Škula

Zmena pracovného adresára

pwd

- Zobrazí aktuálny pracovný adresár. Na začiatku práce v systéme je užívateľov pracovný adresár nastavený na domovský adresár. Príkaz pwd nemá žiadne ďalšie prepínače ani argumenty.

cd

- Tento program zmení aktuálny pracovný adresár (z anglického jazyka: **c**hange **d**irectory znamená: zmeniť adresár).
- Ak sa chceme prepnúť do adresára, ktorý neexistuje, tak nám terminál vyhodí hlásenie o chybe: „No such file or directory.“
- Z významového hľadiska sú si príkazy **cd /var, cd lib** a jeden príkaz **cd /var/lib** totožné. Druhý príkaz je úspornejší ako ten prvý.

```
root@ASUS-PC:/# cd /home/
globalmenu/                TAB          kimbl/
gnome-globalmenu-0.4-svn964.tar.gz
root@ASUS-PC:/# cd /home/kimbl
root@ASUS-PC:/home/kimbl# pwd
/home/kimbl
root@ASUS-PC:/home/kimbl#
```

- Pri prechádzaní priečinkami môžeme použiť klávesu TAB, ktorá nám zobrazí všetky možné priečinky a súbory, do ktorých sa môžeme prepnúť. Klávesa TAB je veľmi mocná, výrazne zjednodušuje prácu v shellu.
- Ak by sme opäť chceli ísť o adresár vyššie, zadali by sme príkaz **cd.. .**

Výpis obsahu adresára

ls

- Príkaz ls vypíše obsah adresára (z angličtiny **list**). Program má mnoho argumentov špecifikujúcich čo a ako sa má vypísať. Ak je program ls použitý bez argumentov, vypíše zoznam mien súborov v pracovnom adresáre.

```
root@ASUS-PC:/home/kimbl# ls
ahoj     Documents  flower2.gif  Pictures   Templates  wifi
Desktop  Examples   Music        Public     Videos     workspace
root@ASUS-PC:/home/kimbl#
```

- Program ls môže taktiež zobraziť aj skryté súbory a priečinky použitím prepínača **-a**
  ls –a

- použitím prepínača **-F** sa nám zobrazí aj druh jednotlivých súborov
  ls -aF.

```
root@ASUS-PC:/home/kimbl# ls -aF
./                .gconfd/            Public/
../               .gimp-2.4/          .pulse/
.adobe/           .gksu.lock          .pulse-cookie
ahoj/             .gnome/             .purple/
.appletviewer     .gnome2/            .recently-used.xbel
.bash_history     .gnome2_private/    .screenlets/
.bash_logout      .gnupg/             .ssh/
.bashrc           .gstreamer-0.10/    .sudo_as_admin_successful
.bluefish/        .gtk-bookmarks      Templates/
.bogofilter/      .gtkrc-1.2-gnome2   .themes/
.cache/           .gvfs/              .thumbnails/
.compiz/          .ICEauthority       .update-manager-core/
.config/          .icons/             .update-notifier/
Desktop/          .java/              Videos/
.dmrc             .local/             .w3m/
Documents/        .macromedia/        .wapi/
.eclipse/         .metacity/          wifi/
.esd_auth         .mozilla/           .windows-label
.evolution/       Music/              .wine/
Examples@         .nautilus/          workspace/
```

- Pre detailnejší výpis súborov použijeme prepínač **-l**. Zobrazí sa nám tabuľka kde sú zobrazené práva, vlastník, skupina, veľkosť a dátum modifikácie
- Prepínač **-r** nám umožní vypísanie súborov a priečinkov v opačnom poradí.
- Prepínač **-S** nám umožní vypísanie súborov a priečinkov podľa veľkosti.
- Prepínač **-t** nám umožní vypísanie súbor a priečinkov podľa času poslednej modifikácie súboru alebo priečinku.
- Prepínač **-1** nám umožní vypísať súbory a priečinky do jedného stĺpca.

```
root@ASUS-PC:/home/kimbl# ls -1
ahoj
Desktop
Documents
Examples
flower2.gif
Music
Pictures
Public
Templates
Videos
wifi
workspace
```

- Prepínač **-R** nám umožní vypísať súbory vo všetkých podpriečinkoch rekurzívne.
- Prepínač **-n** je skoro to isté ako **-l** ale naviac vypíše UID aj GID.

```
root@ASUS-PC:/home/kimbl# ls -n
total 64
drwxr-xr-x 2    0    0  4096 2008-11-27 11:38 ahoj
drwxr-xr-x 2 1000 1000  4096 2008-11-29 19:17 Desktop
drwxr-xr-x 2 1000 1000  4096 2008-11-03 17:14 Documents
```

- Za príkaz **ls {Prepínače}** tiež možno napísať cestu, ktorý priečinok alebo súbor sa má vypísať.

tree

- Program **tree** nám umožní vypísať hierarchiu v stromovej forme. Program tree nie je štandardnou súčasťou všetkých implementácií Linuxu. Je potrebné ho doinštalovať.

```
root@ASUS-PC:/home/kimbl# tree
.
|-- Desktop
|   |-- Amy_Macdonald_-_This_Is_The_Life_-_02_-_This_Is_The_Life.mp3
|   |-- MouseClickExample.class
|   |-- MouseClickExample.java
|   |-- applet.html
|   |-- applet.html~
|   |-- index.html
|   |-- index.html~
|   |-- linux.doc
|   |-- styl.css
|   `-- styl.css~
|-- Documents
|-- Examples -> /usr/share/example-content
```

- Prepínač **-d** nám umožní vypísať iba adresáre.

```
root@ASUS-PC:/home/kimbl# tree -d
.
|-- Desktop
|-- Documents
|-- Examples -> /usr/share/example-content
|-- Music
|-- Pictures
|-- Public
|-- Templates
|-- Videos
|-- ahoj
|-- wifi
|   `-- aircrack-2.41
|       |-- linux
|       |   `-- patch
|       |-- test
|       `-- win32
|           |-- airdecap
|           |-- airodump
|           `-- wzcook
```

- Prepínač **-f** nám umožní vypísať plnú cestu každého súboru.
- Prepínač **-D** nám vypíše ku každému súboru aj dátum poslednej modifikácie.
- Prepínač **-u** nám vypíše meno vlastníka
- Prepínač **-r** nám vypíše hierarchiu súborov v opačnom poradí

Práca s adresármi

mkdir

- Program **mkdir** nám vytvorí nový adresár, ak taký neexistuje (z angličtiny **m**a**k**e **dir**ectory znamená: vytvoriť adresár). Program berie ako argument cestu. Program predpokladá, že cesta bez posledného elementu je cestou k adresáru.

```
root@ASUS-PC:/home/kimbl# ls
Desktop    Examples     Music     Public     Videos  workspace
Documents  flower2.gif  Pictures  Templates  wifi
root@ASUS-PC:/home/kimbl# mkdir NOVE
root@ASUS-PC:/home/kimbl# ls
Desktop    Examples     Music  Pictures  Templates  wifi
Documents  flower2.gif  NOVE   Public    Videos     workspace
root@ASUS-PC:/home/kimbl#
```

- Prepínač **-p** vytvorí aj všetky podadresáre v zadanej ceste ak neexistujú.
- Prepínač **-v** vypíše aj text s informáciou o vytváraní priečinka.

rmdir

- Program vymaže prázdny adresár (z angličtiny **r**e**m**ove **dir**ectory znamená: vymazať adresár). Prázdnym adresárom sa myslí adresár obsahujúci iba odkazy „ .“ a „ ..“, to znamená, že nesmie obsahovať žiadne dátové súbory, ďalšie podadresáre a tak ďalej. Program rmdir má takú istú syntax ako mkdir.

du

- Program **du** nám zistí veľkosť adresára. Programu **du** sa dá tiež predať ako argument aj cesta k adresáru alebo súboru, ktorého veľkosť chceme zistiť.

```
root@ASUS-PC:/# du /home/kimbl/Desktop
8280    /home/kimbl/Desktop
```

- Prepínač **-h** pre každú hodnotu automaticky volí aj jednotku tak, aby bol výsledok dobre. čitateľný.
- Prepínač **-b** nám vypíše výsledok v B.
- Prepínač **-k** nám vypíše výsledok v kB.
- Prepínač **-m** nám vypíše výsledok v MB.
- Prepínač **-s** nám vypíše celkový súčet veľkostí.

```
root@ASUS-PC:/# du -b /home/kimbl/Desktop/
8411238 /home/kimbl/Desktop/
root@ASUS-PC:/# du -k /home/kimbl/Desktop/
8280    /home/kimbl/Desktop/
root@ASUS-PC:/# du -m /home/kimbl/Desktop/
9       /home/kimbl/Desktop/
root@ASUS-PC:/#
```

- Ak je treba výpis obmedziť len na určitú hĺbku, je možné použiť prepínač --max-depth= <hodnota>, kde < hodnota > je maximálna hĺbka výpisu. Ak je hodnota nulová, potom bude vypísaná len veľkosť aktuálneho adresára.

file

- Program nám zistí typ súboru. Cesta k súboru je programu **file** predaná ako argument. Databáza programu **file** je veľká a je veľká šanca, že správne identifikuje typ súboru.

```
root@ASUS-PC:/# file /home/kimbl/Desktop/picture3.png
/home/kimbl/Desktop/picture3.png: PNG image data, 1280 x 800, 8-bit/color RGBA,
non-interlaced
root@ASUS-PC:/# file /home/kimbl/Desktop/picture3.png
/home/kimbl/Desktop/picture3.png: PNG image data, 1280 x 800, 8-bit/color RGBA, non-interlaced
root@ASUS-PC:/# file /home/kimbl/Desktop/styl.css
/home/kimbl/Desktop/styl.css: ASCII text
```

cat

- Program nám umožní vypísať obsah textového súboru (z angličtiny concatenate files and print on the standard output). Program uskutoční zreťazenie súborov, ktoré sú mu predané ako argumenty. Nasledujúci príklad demonštruje zreťazenie dvoch súborov:

```
root@ASUS-PC:/home/kimbl# cat new\ file.txt faktoria.scm
dnes sa mám dobre, dúfam
(define (f n)
  (if (= n 0)
     1
     (* n (f (- n 1)))))
```

- Prepínač **-n** nám do výstupu vypíše aj čísla riadkov.

more

- Program zobrazí textový súbor a za každou stránkou čaká na stlačenie medzerníka alebo klávesy q. Pokiaľ je stlačená klávesa medzerník, pokračuje sa ďalšou stranou, klávesa q ukončí prehliadanie súboru.
- Prepínač **-num**, kde za num sa dosadí číslo, špecifikuje, koľko riadkov sa má vypísať

```
root@ASUS-PC:/home/kimbl# more -2 faktoria.scm
(define (f n)
  (if (= n 0)
     1
     (* n (f (- n 1)))))
root@ASUS-PC:/home/kimbl# more -1 faktoria.scm
(define (f n)
root@ASUS-PC:/home/kimbl# more -1 faktoria.scm
(define (f n)
  (if (= n 0)
     1
     (* n (f (- n 1)))))
```

- Prepínač **-d** vypíše informáciu: "[Press space to continue, ’q’ to quit.]" a vypíše "[Press ’h’ for instructions.]".

```
root@ASUS-PC:/home/kimbl# more -1d faktoria.scm
(define (f n)
  (if (= n 0)
     1
--More--(58%)[Press space to continue, 'q' to quit.]
```

less

- Program umožňuje vracať sa v texte späť a prehľadať súbor pomocou “regulárnych výrazov”

most

- Program vie skoro to isté ako program less, najviac je schopný prehliadať aj binárne súbory a podporuje aj farby. Ak sú v textovom súbore prítomné špeciálne sekvencie nastavujúce chovanie terminálu, napríklad farby, program **most** na ne dokáže správne reagovať.

head

- Program head (súbor) vypíše prvých 10 riadkov zo súboru.
- Prepínač **-n** kde za n je dosadené číslo vypíše prvých n riadkov zadaného súboru.
- Prepínač **-c** vypíše prvých c bytov zadaného súboru.

```
root@ASUS-PC:/home/kimbl# head -c 54  faktoria.scm
(define (f n)
  (if (= n 0)
     1
     (* n (f (- n 1
```

tail

- Program funguje skoro tak isto ako program **head,** štandardne vypíše posledných 10 riadkov súboru.
- Prepínač **-n** kde za n je dosadené číslo vypíše posledných n riadkov zadaného súboru.

```
root@ASUS-PC:/home/kimbl# tail -1 faktoria.scm
     (* n (f (- n 1)))))
```

- Prepínač **-c** vypíše posledných c bytov zadaného súboru

wc

- Pomocou programu wc je možné zisťovať podrobnejšie informácie o dĺžke súboru. Program štandardne vypíše počet bajtov, slov a riadkov v súbore.
- Prepínač **-c** nám vypíše počet bytov.
- Prepínač **-m** nám vypíše počet znakov.
- Prepínač **-l** nám vypíše počet riadkov.

```
root@ASUS-PC:/home/kimbl# wc faktoria.scm
 4 14 60 faktoria.scm
```

Práca so súbormi

cp

- Program cp uskutoční skopírovanie súboru (z angličtiny **co**p**y** znamená: kopírovať). Program berie ako argumenty súbory, nutné je udať aspoň dva súbory. Posledný predaný argument sa chápe ako cieľ. Cieľom môže byť cesta k súboru alebo adresáru. Ostatné predané argumenty sú chápané ako zdroje. Argumenty pre kopírovanie sa uvádzajú vždy v poradí „čo – kam“.
- Prepínač **-R**, **-r** nám umožní kopírovať priečinky rekurzívne.
- Prepínač **-a** uskutočňuje surovú kópiu, to znamená, že sa nesnaží nahradzovať

symbolické odkazy, zachováva taktiež špeciálne súbory a tak ďalej.

- Prepínač **-p** nám umožní kopírovať nielen obsah súboru ale aj prístupové práva, časové razítko, vlastníka a skupinu.
- Prepínač **-i** nás vyzve, či smie prepísať cieľový súbor.
- Prepínač **-f** bezpodmienečne prepíše cieľový súbor, pokiaľ existuje.

```
root@ASUS-PC:/home# cp kimbl/faktoria.scm kimbl/ahoj/
root@ASUS-PC:/home# ls kimbl/ahoj/
faktoria.scm
```

mv

- Program mv uskutočňuje presuny súborov adresárov do cieľového priečinka.(z angličtiny: **m**ove znamená: presunúť.**)**
- Prepínač **-i** nás vyzve, či smie prepísať cieľový súbor.
- Prepínač **-f** bezpodmienečne prepíše cieľový súbor, pokiaľ existuje.

```
root@ASUS-PC:/home# mv kimbl/ahoj/faktoria.scm kimbl/Public/
root@ASUS-PC:/home#
root@ASUS-PC:/home# ls kimbl/Public/
faktoria.scm
root@ASUS-PC:/home#
```

rm

- Program rm uskutočňuje vymazávanie súborov (z angličtiny **rem**ove znamená: vymazať). Opäť implicitne nejde mazať adresáre. Vymazanie opäť prebieha bez pýtania.
- Prepínač **-r** nám umožní mazať adresáre aj s obsahom.
- Prepínač **-i** nás vyzve, či smie vymazať cieľový súbor.
- Prepínač **-f** umožňuje bezpodmienečné odstránenie súboru, všetky chybové hlásenia a varovania sa ignorujú.

```
root@ASUS-PC:/home/kimbl# ls -l Public/
total 4
-rw-r--r-- 1 root root 60 2008-11-26 09:28 faktoria.scm
root@ASUS-PC:/home/kimbl# rm -i Public/faktoria.scm
rm: remove regular file `Public/faktoria.scm'? y
root@ASUS-PC:/home/kimbl# ls -l Public/
total 0
root@ASUS-PC:/home/kimbl#
```

Práca so súborovým systémom

mount, umount

- Pripojenie súborového systému sa uskutočňuje pomocou príkazu **mount,** odpojenie pomocou príkazu **umount**. Tieto programy môže až na výnimky používať iba super užívateľ. Pri pripojovaní súborového systému je nutné uviesť špeciálny súbor odpovedajúci blokovému zariadeniu. Ďalej je potreba špecifikovať cieľový adresár – tzv. mount point. Dôležité je tiež i uvedenie príslušného typu reálneho súborového systému, ktorý sa na blokovom zariadení nachádza. V systéme Linux sú v súčasnosti najpoužívanejšie Ext2, Ext3, ReiserFS a XFS. Ak je program **mount** spustený bez argumentov, dôjde k vypísaniu informácií o pripojených systémoch.

```
root@ASUS-PC:/home/kimbl# mount
/dev/sda6 on / type ext3 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
/sys on /sys type sysfs (rw,noexec,nosuid,nodev)
varrun on /var/run type tmpfs (rw,noexec,nosuid,nodev,mode=0755)
varlock on /var/lock type tmpfs (rw,noexec,nosuid,nodev,mode=1777)
udev on /dev type tmpfs (rw,mode=0755)
devshm on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
lrm on /lib/modules/2.6.24-21-generic/volatile type tmpfs (rw)
/dev/sda5 on /media/sda5 type fuseblk (rw,nosuid,nodev,noatime,allow_other,default_permissions,blksize=4096)
securityfs on /sys/kernel/security type securityfs (rw)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noexec,nosuid,nodev)
/dev/sda1 on /media/disk type fuseblk (rw,nosuid,nodev,noatime,allow_other,blksize=4096)
```

mkfs

- Program vytvorí súborový systém určeného typu, zvyčajne na hard diskov.
- Prínač **-c** skontroluje najprv disk pred tým ako urobí súborový systém.
-

Pripojenie a odpojenie súborového systému

S vybranými blokovými zariadeniami môžu manipulovať aj radový užívatelia. Jedná sa predovšetkým o disketové mechaniky a podobne. V systéme existuje súbor /etc/fstab obsahujúci informácie o súborovom systéme a ich prípojných adresároch. V súboru môžeme okrem iného nájsť napríklad riadky nasledujúceho tvaru:

```
/dev/sda5      /media/sda2     ntfs    defaults,nls=utf8,umask=007,gid=46 0
1
/dev/scd0      /media/cdrom0   udf, iso9660 user, noauto     0       0
```

Odkazy

Pevný odkaz – *hard link* sa dá vytvoriť iba na existujúci súbor. Počet referencií na súbor sa z každým odkazom zväčší o jeden. Ak je pôvodný súbor zmazaný a existuje naňho nejaká referencia, obsah súboru je zachovaný, iba sa počet referencií zmenší o jeden. Pevný odkaz je možné vytvoriť len v rámci jedného reálneho súborového systému. Vo väčšine systémov nie je možné vytvoriť pevný odkaz na adresár.

Symbolický link – *symblic link* je v podstate obyčajný súbor obsahujúci cestu k cieľovému súboru. Cesta je v symbolickom odkaze uchovávaná ako reťazec znakov a môže byť absolútna alebo relatívna. Symbolický odkaz je možné zaviesť aj na neexistujúce súbory alebo súbory ležiace na inom súborovom systéme. V neposlednej rade je možné symbolický link vytvoriť aj na adresár. Pokiaľ je zdrojový súbor zmazaný alebo presunutý, potom bude symbolický odkaz ukazovať na neexistujúci súbor. Takýto odkaz sa nazýva „narušený“ - broken link.

ln

- Program **ln** vytvára symbolické aj pevné odkazy medzi súbormi. Program ln má syntax <prepínač> <zdroj1> <zdroj2> ...<ciel>. Pritom cieľ môže byť vynechaný za predpokladu, že zdroj je práve iba jeden.
- Prepínač **-s** umožňuje vytvárať symbolické odkazy.

- Prepínač **-f** umožňuje vytvoriť odkaz aj v prípade, že jeho meno koliduje s už existujúcim súborom v systéme.

symlinks

- Program slúži na urobenie poriadku medzi symbolickými odkazmi. Súbor sa bežne nenachádza v OS, je treba ho doinštalovať. Jedná sa o nástroj vytvorený pre zjednodušenie údržby symbolických odkazov. Argumenty programu sú prehľadávané adresáre. Pokiaľ je program spustený v upovedanom režime, vypíše informácie o všetkých symbolických odkazoch. V opačnom prípade informuje iba o problematických odkazoch.
- Prepínač -d zabezpečí odstránenie všetkých narušených odkazov.
- Prepínač -c prevedie všetky absolútne odkazy v rámci jedného súborového systému na relatívne.
- Prepínač -v vypíše všetky symbolické linky.

```
root@ASUS-PC:/home/kimbl# symlinks -v /home/kimbl/
absolute: /home/kimbl/Examples -> /usr/share/example-content
```

Nastavenie vlastníka a skupiny

Pokiaľ užívateľ potrebuje získať informácie o svojom účte, môže ich vyčítať priamo zo súboru ***/etc/passwd.***

whoami

- Vypíše username (prihlasovacie meno) pod ktorým ste zalogovaní

```
root@ASUS-PC:/home/kimbl# whoami
root
```

groups

- Program slúži na vypísanie skupín, do ktorých patrí užívateľ. Ak nie je uvedené meno užívateľa ako argument, sú vypísané informácie o práve prihlásenom užívateľovi.

```
root@ASUS-PC:/home/kimbl# groups
root
```

chown

- Program chown slúži na zmenu vlastníka (z anglického jazyka: **ch**ange **own**er znamená: zmena vlastníka). Program chown je univerzálny, je ním možné zmeniť vlastníka súboru, skupinu alebo oboje zároveň. Zmena vlastníka súboru je vyhradená iba super užívateľovi root. U súborov však môže nastaviť iba skupiny, do ktorých sám patrí. Syntax je nasledujúca:

  chown <vlastník>.<skupina> <súbor>

- Prepínač -R spôsobí zmenu vlastníka vo všetkých podadresároch.

chgrp

- Program slúži na zmenu skupiny (z anglického jazyka: **ch**ange **gr**ou**p** znamená: zmena skupiny). Skupinu, môže meniť aj obyčajný užívateľ. Syntax je nasledujúca:

chgrp <skupina> <súbor>

- Prepínač -R spôsobí zmenu skupiny vo všetkých podadresároch.

Zmena módu prístupu

chmod

- Pomocou programu chmod je možné meniť prístupové práva (z anglického jazyka: **ch**ange file **mod**e znamená: zmena súborového módu). Program býva používaný dvojitým spôsobom. Prvý spôsob je tradičný, práva sú jednoznačne určené pomocou **módu prístupu** – čísla zapísané v oktánovej sústave. Druhý spôsob je **symbolický**, práva sú modifikované podľa vzorového reťazca symbolov. Syntax je nasledujúca:

  chmod <číslo><súbory>
  chmod <kto><operácia><právo><súbory>

- Medzi často používané kombinácie práv patrí 644 – vlastník má právo čítania a zápisu, skupina a ostatní majú iba právo na čítanie. 775 všetky tri skupiny majú naviac právo pre spustenie. 710 – vlastník bude mať plné práva, členovia skupiny môžu do adresára vstúpiť, ale nemôžu vypisovať jeho obsah a ostatní nemajú žiadne práva.

- Symbolický tvar prístupových práv pozostáva z troch po sebe idúcich prvkov a je ho možné chápať ako reťazec <kategória><operácia><právo>. Reťazec <kategória> určuje, aké z troch skupín sa zmeny týkajú. Reťazec <kategória> sa môže skladať z hodnôt: „u“ - vlastník, „g“ - skupina, „o“ – ostatní , „a“ - všetky predchádzajúce kategórie. Symbol <operácia> určuje, akým spôsobom sa bude s právami zachádzať. Môže nadobúdať jednu z hodnôt „+“ - právo bude pridané, „-“ - právo bude odobrané, „=“ - právo bude nastavené absolútne. Posledný reťazec je samotné <právo>. Znak „r“ označuje právo pre čítanie, „w“ označuje právo pre zápis a „x“ označuje právo spúšťania. Pomocou symbolického zápisu je možné prideliť aj špeciálne práva, symbol „s“ označuje SUID alebo SGID. Argument <právo> je možné vynechať iba vtedy, keď je <operácia> rovná „=“.

Kategória	Symbol	Význam	Operácia	Právo	symol
owner	u	pridaj	+	read	r
group	g	odober	-	write	w
others	o	nastav	=	execute	x
all	a			SUID/SGID	s
				sticky	t

- Prepínač -R spôsobí zmenu práv vo všetkých podadresároch. Ak je použitý, daná operácia je aplikovaná na každý súbor nájdený v ľubovoľnom podadresáre.

```
-rw-rw-rw- 1 kimbl kimbl 15325 2008-11-07 20:40 flower2.gif
root@ASUS-PC:/home/kimbl# chmod a=r flower2.gif
root@ASUS-PC:/home/kimbl# ls -l flower2.gif
-r--r--r-- 1 kimbl kimbl 15325 2008-11-07 20:40 flower2.gif
```

umask

- Program umask ustanoví práva prístupu do adresárov a súborov danej partície, čo užívatelia nemajú povolené a čo majú povolené. Možnosti sú v opačnom pomere ako u príkazu chmod. Ak nezadáme žiadny parameter tak sa vypíše momentálne platná maska odoberaných práv v osmičkovom tvare (**022**).

```
root@ASUS-PC:/home/kimbl# umask
0022
```

- Prepínač -S umožní vypísať práva v symbolickom tvare.

```
root@ASUS-PC:/home/kimbl# umask -S
u=rwx,g=rx,o=rx
```

- Hodnota umask= 026 znamená, práva 751 (doplnok do 777).
- Hodnota umask=077 znamená, práva 700. To znamená plné práva pre vlastníka. Ostatní majú všetko zakázané.
- Hodnota umask=000 povolí všetko všetkým.

umask	r	w	x
0	áno	áno	áno
1	áno	áno	nie
2	áno	nie	áno
3	áno	nie	nie
4	nie	áno	áno
5	nie	áno	nie
6	nie	nie	áno
7	nie	nie	nie

„r“- práva pre čítanie, **„w”** - práva pre zápis, **„x“**- práva pre spúšťanie

(syntax: umask = **xyz**, kde: **x** - vlastník, **y** – skupina, **z** - ostatní)

Riadenie prístupu pomocou ACL

getfacl

- Program getfacl zobrazí informácie ACL = zoznam riadení prístupu o súbore (z anglického jazyka: **get f**ile **a**ccess **c**ontrol **l**ists znamená: zoznam riadení prístupu).Ako argumenty sú programu getfacl predané názvy cesty k súborom. Pokiaľ je ako argument uvedení aj súbor ležiaci mimo súborový systém podporujúci ACL, getfacl sa chová identicky ako v prípade súboru so základnými ACL záznamami.

- Prvé tri riadky poskytujú informácie o mene súboru, vlastníkovi a skupine. Riadky sú chápané ako komentáre, pretože začínajú symbolom #.

```
root@ASUS-PC:/home/kimbl# getfacl flower2.gif
# file: flower2.gif
# owner: kimbl
# group: kimbl
user::r--
group::r--
other::r--
```

- Prepínačom **--skip-base** je možné potlačiť výpis základných ACL záznamov.
- Prepínačom **--omit-header** je možné potlačiť výpis informácií v hlavičke.

Zdieľanie diskového priestoru

df

- Program zobrazí informácie o dostupných pripojených súborových systémoch vrátane ich veľkosti a voľného miesta.

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda6              9361164   4496764   4388880  51% /
varrun                 1037780       104   1037676   1% /var/run
varlock                1037780         0   1037780   0% /var/lock
udev                   1037780        60   1037720   1% /dev
devshm                 1037780        12   1037768   1% /dev/shm
/dev/sda5             53102824  21389692  31713132  41% /media/sda5
/dev/sda1             34812820   7103996  27708824  21% /media/disk
```

- Prepínač **-f** vypíše veľkosti v ľudskej forme (na príklad:1K 234M 2G).
- Prepínač **-a** vypíše všetky súborové systémy (aj nulové).
- Prepínač **-T** vypíše aj typ súborového systému.

Systém procesov

Informácie o procesoch v systéme

ps

- Program ps nám slúži na získanie zoznamu procesov. Program ps má veľa argumentov. Použijeme ho bez argumentov, ps vypíše zoznam aktívnych procesov spojených s aktuálne používaným terminálom. Vo výpise nebudú zobrazené procesy spustené z iných terminálov.
- Prvým z procesov je príkaz su, druhým je aktívny interpret príkazov, tretí proces je samotný program **ps**. Prvý stĺpec označuje PID procesov, druhý stĺpec reprezentuje aktuálny terminál, tretí stĺpec reprezentuje celkový čas behu procesu.

```
root@ASUS-PC:/home/kimbl# ps
  PID TTY          TIME CMD
 8073 pts/1    00:00:00 su
 8074 pts/1    00:00:00 bash
 8094 pts/1    00:00:00 ps
```

- Prepínač **-a** zobrazí informácie o všetkých aktívnych procesoch riadených nejakým terminálom.

```
root@ASUS-PC:/home/kimbl# ps -a
  PID TTY          TIME CMD
 8073 pts/1    00:00:00 su
 8074 pts/1    00:00:00 bash
 8236 pts/0    00:00:00 su
 8241 pts/0    00:00:00 bash
 8257 pts/0    00:00:00 su
 8258 pts/0    00:00:00 bash
 8277 pts/0    00:00:00 su
 8278 pts/0    00:00:00 bash
 8298 pts/1    00:00:00 ps
```

- Prepínač **-e** zobrazí informácie o úplne všetkých procesoch, to znamená aj o procesoch, ktoré nie sú napojené na terminál.
- Prepínač **-l** zobrazí takzvaný „dlhý výpis.“
- Prepínačom **-f** je možné zobraziť podmnožinu „dlhého výpisu.“

```
root@ASUS-PC:/home/kimbl# ps -af
UID        PID  PPID  C STIME TTY          TIME CMD
root      8073  7962  0 12:33 pts/1    00:00:00 su
root      8074  8073  0 12:33 pts/1    00:00:00 bash
root      8236  7120  0 12:35 pts/0    00:00:00 su
root      8241  8236  0 12:35 pts/0    00:00:00 bash
root      8257  8241  0 12:35 pts/0    00:00:00 su
root      8258  8257  0 12:35 pts/0    00:00:00 bash
kimbl     8277  8258  0 12:35 pts/0    00:00:00 su kimbl
kimbl     8278  8277  0 12:35 pts/0    00:00:00 bash
root      8759  8074  0 12:42 pts/1    00:00:00 ps -af
root@ASUS-PC:/home/kimbl# ps -al
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0  8073  7962  0  80   0 -   917 -      pts/1    00:00:00 su
0 S     0  8074  8073  0  80   0 -  1038 -      pts/1    00:00:00 bash
4 S     0  8236  7120  0  80   0 -   917 -      pts/0    00:00:00 su
0 S     0  8241  8236  0  80   0 -  1036 -      pts/0    00:00:00 bash
4 S     0  8257  8241  0  80   0 -   917 -      pts/0    00:00:00 su
0 S     0  8258  8257  0  80   0 -  1037 -      pts/0    00:00:00 bash
4 S  1000  8277  8258  0  80   0 -   917 -      pts/0    00:00:00 su
0 S  1000  8278  8277  0  80   0 -  1420 -      pts/0    00:00:00 bash
4 R     0  8776  8074  0  80   0 -   607 -      pts/1    00:00:00 ps
```

- Nájdeme tu stĺpce ako sú STIME – systémový čas vzniku procesu. Za údajom PID je uvedený PPID - „parent process indetification“, to jesť PID rodičovského procesu. Údaj C reprezentuje percentové vyťaženie procesoru. Stĺpec F – maska procesu a stĺpec S – stav procesu (S – spiaci proces, R – bežiaci proces, T pozastavený proces).

SZ – zobrazuje informácie o počte blokov zabratých obrazom programu. Stĺpec WCHAN obsahuje informácie o akcii, na ktorú proces čaká. U bežiaceho procesu je táto položka vždy prázdna.

- Za prepínač **-o** je možné písať stĺpce, ktoré chceme aby sa vypísali. Oddeľujeme ich čiarkou a nesmú obsahovať medzery.

```
root@ASUS-PC:/home/kimbl# ps -o pid,wchan,ppid,s,cmd
  PID WCHAN   PPID S CMD
 8073 -       7962 S su
 8074 -       8073 S bash
 9913 -       8074 R ps -o pid,wchan,ppid,s,cmd
```

top

- Program nám zobrazí linuxácke procesy, pracuje v celoobrazovom režime a periodicky zobrazuje informácie o bežiacich procesoch. Beh programu je možné ukončiť klávesou „q“, po stlačení klávesy „h“ je zobrazená nápoveda / help programu.

```
top - 13:05:59 up 59 min,  3 users,  load average: 0.76, 0.60, 0.44
Tasks: 151 total,   1 running, 150 sleeping,   0 stopped,   0 zombie
Cpu(s): 17.1%us,  2.4%sy,  0.0%ni, 80.1%id,  0.0%wa,  0.0%hi,  0.5%si,  0.0%st
Mem:   2075560k total,  1064184k used,  1011376k free,    48940k buffers
Swap:   248968k total,        0k used,   248968k free,   537784k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6979 kimbl     20   0  140m  32m  14m S   25  1.6   5:41.75 totem
 5643 root      20   0  102m  54m  10m S    2  2.7   2:07.07 Xorg
 6300 kimbl     20   0 70308  35m  11m S    2  1.8   1:35.24 compiz.real
 6213 kimbl     20   0 29068 4628 3584 S    1  0.2   0:28.44 pulseaudio
 6483 kimbl     20   0 51416  19m  12m S    1  1.0   0:19.82 python
 6665 kimbl     20   0  246m 102m  26m S    1  5.1   3:28.23 firefox
 5124 messageb  20   0  2952 1436  796 S    1  0.1   0:03.82 dbus-daemon
 6327 kimbl     20   0 49432  12m 9792 S    1  0.6   0:02.26 nm-applet
 6433 kimbl     20   0 21856 9956 8368 S    1  0.5   0:24.30 multiload-apple
 7116 kimbl     20   0  101m  20m  12m S    1  1.0   0:02.62 gnome-terminal
 7059 kimbl     20   0  221m  83m  52m S    0  4.1   0:30.88 soffice.bin
    1 root      20   0  2844 1692  544 S    0  0.1   0:01.32 init
    2 root      15  -5     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S    0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S    0  0.0   0:00.06 ksoftirqd/0
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/0
    6 root      RT  -5     0    0    0 S    0  0.0   0:00.00 migration/1
```

bg

- Program **bg** slúži na presun pozastaveného procesu na pozadie. Jedná sa o príkaz vstavaný priamo v shelli. Argumentom pre príkaz bg je indetifikátor procesu v rámci shellu. Príkazu bg nie je možné predať ako argument PID. Ak nie je pre program bg uvedený žiadny argument, berie sa posledný použitý identifikátor procesu v rámci aktívneho shellu.

fg

- Príkaz **fg** slúži na presný opak ako bg, to znamená, že slúži na obnovenie chodu procesu, ktorý bol užívateľom pozastavený, z dôvodu ďalšej práce. Programu fg je ako argument predaný identifikátor procesu.

nice, renice

- Programom **nice** je možné meniť. Pomocou programu **renice** je možné meniť prioritu už bežiacemu procesu.
- Syntax programov je:

nice -n <priorita><príkaz>
renice <priorita> -p <PID>

- Argument *priorita* môže nadobúdať celočíselných hodnôt z intervalu -20 až 19 a predstavuje znevýhodnenie procesu. Ak je hodnota kladná, priorita procesu bude znížená, ak je záporná tak bude priorita zvýšená.

Signál	Číslo	význam
SIGHUP	1	Odpojenie terminálu
SIGINT	2	prerušenie z klávesnice
SIGQUIT	3	Koniec s uložením obrazu pamäte
SIGILL	4	Použitá inštrukcia neznáma pre procesor
SIGTRAP	5	Ladiace prerušenie
SIGABRT	6	Ukončenie z dôvodu vstupov / výstupov
SIGFPE	8	Chyba aritmetiky s pohyblivou desatinnou čiarkou
SIGKILL	9	Okamžité ukončenie procesu
SIGSEGV	11	Zlyhanie segmentácie
SIGPIPE	13	Pokus o zápis do rúry, ktorú nikto nečíta
SIGALRM	14	Vypršanie časového intervalu
SIGTERM	15	Ukončenie
SIGUSR1	16	Užívateľsky signál 1
SIGUSR2	17	Užívateľský signál 2
SIGCHLD	18	Zmena stavu synovského procesu
SIGSTOP	23	Pozastavenie procesu
SIGCONT	24	Pokračovanie činnosti

kill, killall

- Program posiela signál na “zabitie” procesu. Aj keď tomu tak nemusí byť. Bežný užívateľ môže posielať príkaz iba svojím procesom. Super užívateľ root môže zasielať signály ľubovoľným procesom.
- Spustenie programu kill má nasledujúcu syntax:

kill -<signál> <procesy>

- -signál je číslo alebo skratka signálu. Ak je argument <signál> uvedení, musí byť pred

ním uvedená pomlčka. Argument <procesy> predstavuje zoznam oddelený medzerami. Prvky zoznamu môžu byť PID jednotlivých procesov alebo čísla tvaru %<id>, kde id je identifikátor v rámci aktuálneho shellu.

- Program killall taktiež zasiela signály, ale procesy nie sú identifikované svojím PID, ale svojím menom. Toto má svoje výhody ale aj svoje nevýhody. Neopatrné použitie môže viesť k ukončeniu viacerých procesom pod tým istým menom.

nohup

- Program nohup berie ako argument príkaz spustenia programu. Program je spustený spôsobom, nohup naviac urobí premiestnenie jeho výstupu do súboru nohup.out.

```
root@ASUS-PC:/home/kimbl# nohup firefox
nohup: ignoring input and appending output to `nohup.out'
```

at

- Program **at** umožňuje plánovať spúšťanie procesov. Užívateľský program at umožňuje spúšťať príkazy dopredu zadanom čase. Program má nasledujúcu syntax:

at <čas> <dátum> +<prírastok>

- <čas> určuje čas dňa určeného argumentom <dátum>. Prírastok určuje časový posun vzhľadom na tento deň. Každý z parametrov je voliteľný, prítomný musí byť aspoň jeden z nich. Každý neuvedený údaj bude nahradený svojou implicitnou hodnotou. Čas môže byť špecifikovaný ako jednociferné, dvojciferné alebo štvorciferné číslo. Ak je čas jednociferný alebo dvojciferný jedná sa o hodiny, pokiaľ je štvorciferný. Ide o hodiny a minúty zapísané v 24 hodinovej notácii. Ak nie je čas uvedený, bude úloha spustená o polnoci.
- <dátum> je možné špecifikovať ako deň v týždni, alebo ako dátum v rámci roku. Deň je možné špecifikovať pomocou anglických skratiek. Napríklad ***Mon*** reprezentuje ***pondelok***, ***Jul 4*** reprezentuje ***4 júla***. Ak nie je dátum uvedený, program kalkuluje s aktuálnym dátumom.
- +<prírastok> sa zapisuje v tvare čísla nasledované jedným zo slov: minutes, hours, days, weeks, months, years. Význam je evidentný, k zadanému dátumu sa pripočíta daný počet minút, hodín, dní, týždňov, mesiacov alebo rokov. Ak nie je prírastok uvedený, bude uvedený časový okamžik absolútny.

```
root@ASUS-PC:/home/kimbl# at 13:00 4.12.2008
warning: commands will be executed using /bin/sh
at> ls -al ~
at> <EOT>
job 4 at Thu Dec  4 13:00:00 2008
```

- Zadávanie sa ukončí stlačením CTRL+D.

atd

- Program sa stará o spustenie procesu.

atq

- Program zobrazí všetky naplánované úlohy. Vypíše číslo úlohy, čas spustenia a meno užívateľa, ktorý úlohu naplánoval.

atrm

- Program vymaže špecifikovanú úlohu. Argumentom je číslo naplánovanej úlohy.

```
root@ASUS-PC:/home/kimbl# atq
4       Thu Dec  4 13:00:00 2008 a root
root@ASUS-PC:/home/kimbl# atrm 4
```

Základy príkazové interpretu

Vypracoval: Peter Veress

Hľadanie súborov

which

- Program **which** dokáže lokalizovať spustiteľný program podľa jeho mena a aktuálneho nastavenia prehľadávanej cesty. Ako argumenty sú programu **which** predané mená súborov. Program pre každý predaný argument uvedie absolútnu cestu k príslušnému spustiteľnému súboru. Keď cesta nie je nájdená, výstup bude prázdny.

```
root@ASUS-PC:/# which ls man xeyes ahoj
/bin/ls
/usr/bin/man
/usr/bin/xeyes
```

whereis

- Okrem spustiteľných súborov je taktiež možné hľadať aj zdrojové kódy a manuálové stránky. Syntax programu **whereis** je podobná ako u programu **which**. Ku každému heslu vypíše **whereis** zoznam spustiteľných súborov, manuálových stránok a zdrojových kódov, ak sú k dispozícii.

```
root@ASUS-PC:/# whereis print talk
print: /usr/bin/print /usr/share/man/man3/print.3ncurses.gz /usr/share/man/man1/print.1.gz
talk: /usr/bin/talk /usr/share/man/man1/talk.1.gz
```

find

- Program **find** je veľmi mocný. Program je možné použiť k hľadaniu súborov podľa ich mena, vlastníka, typu a podobne. Z jednotlivých podmienok prehľadávania je možné vzájomnou kombináciou vytvárať zložitejšie podmienky. Podľa zadaných podmienok prehľadáva špecifikovanú adresárovú štruktúru. Základná syntax programu je jednoduchá. Programu **find** je možné zadať *zoznam prehľadávaných ciest* a *prehľadávacie podmienky*. Zoznam ciest musí byť uvedený pred podmienkami, ale je možné ho aj vynechať. V tomto prípade prehľadávanie začína v domácom adresáre.
- Pár argumentov:

Argument	Význam
-links <n>	**Na súbor existuje práve n odkazov**
-mtime <n>	**Obsah súboru bol za uplynulých n dní zmenení**
-name <name>	**Meno súboru odpovedá zadanému vzoru**

-newer <subor>	**Súbor je novší ako <subor>**
-type <typ>	**Súbor je daného typu**
-user / -group	**Obmedzuje vlastníka alebo skupinu**
-size <n>	**Súbor veľkosti n**
-used <n>	**Súbor bol použitý pre n dňami**
-mmin <n>	**Dáta v súbore boli naposledny modifikované pred n minútami.**

- Size: a) 'b' - pre 512-bytové bloky (použité ako default ak nie je nič uvedené)

  b) 'c' - pre bytes

  c) 'w'- pre dvojbytové slovo

  d) 'k' - pre Kilobytes

  e) 'M' - pre Megabytes

  f) 'G' - pre Gigabytes

- Type: a) b - block (bufferovaný) speciálne

  b) c - znak (nebuferovaný) špeciálny

  c) d – priečinok

  d) p – meno „rúry" (FIFO)

  e) f – regulárny súbor

  f) l – symbolický link

```
root@ASUS-PC:/# find /var/log/ /tmp/ -not -user root -type d
/var/log/news
/tmp/keyring-rMGnqa
/tmp/svhi6.tmp
/tmp/seahorse-fr9guT
/tmp/orbit-kimbl
/tmp/screenlets
/tmp/pulse-kimbl
/tmp/plugtmp
/tmp/.exchange-kimbl
/tmp/Tracker-kimbl.6126
/tmp/Tracker-kimbl.6126/Attachments
/tmp/virtual-kimbl.3aA6EF
/tmp/.esd-1000
/tmp/gconfd-kimbl
/tmp/gconfd-kimbl/lock
```

- Program **find** umožňuje vytvárať aj disjunktné väzby pomocou prepínača **-or** a tiež je možné stanoviť prioritu pomocou zátvoriek.
- Program **find** môže s nájdenými súbormi robiť rôzne akcie. Ak je použitý argument **-exec**, program môže pre každý nájdený súbor vykonať zadaný príkaz. V príkazu je vždy reťazec „{}“ nahradený menom aktuálneho súboru. Príkaz uvedený za argumentom **-exec** musí byť ukončený „\;“

```
root@ASUS-PC:/# find /var/log/ /tmp/ -not -user root -type f -exec mv {} ~/Backup \;
```

Súbory sú nájdené a namiesto vypísania na obrazovku sú presunuté do adresára ~/Backup

```
root@ASUS-PC:/# ls ~/Backup
5952              messages.0
auth.log          messages.1.gz
auth.log.0        messages.2.gz
auth.log.1.gz     messages.3.gz
```

locate

- Program nám slúž i k prehľadávaniu databáze. Jeho syntax je opäť taká istá ako v predchádzajúcich prípadoch. Je však možné uvádzať aj neúplné názvy.

```
root@ASUS-PC:/home/kimbl# locate ahoj
/home/kimbl/.local/share/Trash/info/ahoj.3.trashinfo
/home/kimbl/.local/share/Trash/info/ahoj.trashinfo
```

Oddeľovanie príkazov

Na jeden riadok je možné uviesť aj viacej príkazov. Na to nám slúži „;“. Príkazy oddelené bodkočiarkou sú uskutočnené, ako keby boli napísané pod sebou.

Metaznak „&“ má v shelli vyššiu prioritu ako metaznak „;“. To znamená, ak sú napríklad uvedené príkazy oddelené bodkočiarkou, a posledný je ukončený ampersandom, príkazy budú spustené postupne a posledný príkaz z nich bude spustený na pozadí.

```
root@ASUS-PC:/home/kimbl# who am i; ls; uptime
kimbl    pts/0        2009-01-19 18:15 (:0.0)
Desktop    faktoria.scm   Music          Public     wifi
Documents  faktoria.scm~  new file.txt~  Templates  workspace
Examples   Mail           Pictures       Videos
 18:15:41 up  1:04,  2 users,  load average: 0.44, 0.59, 0.46
```

Obidva znaky „;“ a „&“ slúžia ako oddeľovače a používajú sa identickým spôsobom – ukončujú príkaz.

```
root@ASUS-PC:/home/kimbl# sleep 3; ls -l &
[1] 11149
root@ASUS-PC:/home/kimbl# total 44
drwxr-xr-x 3 kimbl kimbl 4096 2009-01-19 18:15 Desktop
drwxr-xr-x 2 kimbl kimbl 4096 2008-11-03 17:14 Documents

[1]+  Done                    ls --color=auto -l
```

Presmerovanie vstupu a výstupu

Pre presmerovanie vstupu a výstupu je možné použiť metaznaky „<“ „<<“ „>“ „>>“. Metaznaky musia byť použité spolu s definíciou vstupných prípadne výstupných súborov.

- Metaznak „<“ nám slúži k presmerovaniu vstupu, na svojej pravej strane očakáva cestu do vstupného súboru.
- Metaznaky „>“ a „>>“ slúžia k presmerovaniu výstupu. Obidva tieto metaznaky musia mať na pravej strane definovaný výstupný súbor. Na ľavej strane môže byť uvedený ďalší metaznak určujúci typ súboru.

Notácia	Význam
< <subor>	Štandardný vstup je presmerovaný zo súboru
> <subor>	Štandardný výstup je presmerovaný do súboru
2> <subor>	Štandardný chybový výstup je presmerovaný do súboru
&> <subor>	Obidva štandardné výstupy sú presmerované do súboru
>> <subor>	Štandardný výstup je pripojený na koniec súboru
2>> <subor>	Štandardný chybový výstup je pripojený na koniec súboru

```
root@ASUS-PC:/home/kimbl# cat ehoj
Ako sa dnes mas
root@ASUS-PC:/home/kimbl# find /etc/ssl -type f >>ehoj
root@ASUS-PC:/home/kimbl# cat ehoj
Ako sa dnes mas
/etc/ssl/private/ssl-cert-snakeoil.key
/etc/ssl/openssl.cnf
/etc/ssl/certs/ca-certificates.crt
/etc/ssl/certs/ssl-cert-snakeoil.pem
```

Presmerovania je možné využiť aj k primitívnej manipulácii so súbormi. Program **cat** uskutočňuje zreťazovanie súborov, ktoré sú mu predané ako argumenty. Ak nie sú uvedené žiadne argumenty, tak informácie číta zo štandardného vstupu.

```
root@ASUS-PC:/home/kimbl# cat > novy
tak, toto sa zapisuje do suboru
a tiez toto  CTRL + D
```

Zreťazenie obslužných programov pomocou „rúry“

Rúra funguje ako rad. Vznik rúry je podmienený volaním jadra pipe(), na užívateľskej úrovni je jej možné podmieniť metaznakom „|“. Syntax príkazu využívajúci rúru je:

<**producent**> | <**konzument**>

kde **producent** je príkaz pre spustenie programu, ktorého štandardný výstup je zapisovaný do rúry a **konzument** je príkaz pre spustenie programu, ktorý svoj štandardný vstup číta z rúry.

```
root@ASUS-PC:/home/kimbl# date; who | wc &
Mon Jan 19 18:55:23 CET 2009
[1] 13343
root@ASUS-PC:/home/kimbl#       2      10      90

[1]+  Done                    who | wc
```

Najprv bol spustený príkaz **date**. Potom bol rúrou pripojený výstup programu **who** na vstup programu **wc**. Táto kolóna dvoch procesov bola spustená na pozadí.

tee

- Program číta štandardní vstup a predáva ho na štandardní výstup podobne ako program **cat,** naviac môže dáta priebežne zapisovať do súboru na disku. Meno súboru je programu **tee** predané ako argument.

```
root@ASUS-PC:/home/kimbl# tee zaznam.txt
fdg
fdg
wqe
wqe
CTRL + D
root@ASUS-PC:/home/kimbl# cat zaznam.txt
fdg
wqe
```

mkfifo

- Príkaz nám slúži na vytvorenie pomenovanej rúry. Jediný povinný argument pre program **mkfifo** je cesta k novo vytvorenému špeciálnemu súboru. Cesta reprezentuje umiestnenie pomenovanej rúry v súborovom systéme. Program môže byť tiež spustený s argumentom --mode =práva. V tomto prípade sú prístupové práva k pomenovanej rúre nastavené na hodnotu **práva**. Keď je rúra vytvorená, je s ňou možné narábať ako s akýmkoľvek súborom.

```
root@ASUS-PC:/home/kimbl# mkfifo  /tmp/mojarura
root@ASUS-PC:/home/kimbl# find /usr/share -type d > /tmp/mojarura
```

Používanie neúplných mien súborov

echo

- Program echo vypíše predané argumenty na štandardný výstup a ukončí svoju činnosť. Pokiaľ sa medzi argumentami vyskytujú expadovateľné metaznaky, shell je pred predaným programu expanduje. Užívateľ môže pomocou programu **echo** sledovať, ako expanzia prebehla. Program je tiež možné použiť v systéme aj ako čiastočné nahradenie programu **ls**.
-

```
root@ASUS-PC:/home/kimbl# echo .*
. .. .adobe .appletviewer .bash_history .bash_logout .bashrc .bluefish .bogofilt
er .cache .compiz .config .dmrc .eclipse .esd_auth .evolution .fontconfig .gconf
```

V tomto prípade shell metaznak „*“expandoval na mená všetkých viditeľných súborov.

```
root@ASUS-PC:/home/kimbl# echo .*r?
.bash history .bashrc .dmrc .mcoprc .update-manager-core .xsession-errors
```

Vypísané boli skryté súbor, ktorých predposledné písmeno je r.

```
root@ASUS-PC:/home/kimbl# echo [a-h]*
ahoj Desktop Documents ehoj Examples faktoria.scm faktoria.scm
root@ASUS-PC:/home/kimbl# echo [a-h]a*
faktoria.scm faktoria.scm~
root@ASUS-PC:/home/kimbl# echo [^a-h]a*
Mail zaznam.txt
```

Najprv boli vypísané viditeľné súbory začínajúce písmenami „a“ až „d“. V druhom prípade bola podmienka ďalej zosilnená pevnou voľbou písmenka na druhej pozícii. V tretom prípade boli vypísané súbory ktoré nezačínajú písmenami „a“ až „d“.

Pokiaľ užívateľ volá program a ako argument mu predáva regulárny výraz, v niektorých prípadoch musí explicitne shellu oznámiť, aby neuskutočňoval expanziu metaznakov. To je možné uskutočniť niekoľkými spôsobmi. Jedným z nich je použitie metaznaku „\“.

```
root@ASUS-PC:/home/kimbl# echo *
ahoj Desktop Documents ehoj Examples faktoria.scm faktoria.scm~ Mail Music new f
ile.txt~ novy Pictures Public Templates Videos wifi workspace zaznam.txt
root@ASUS-PC:/home/kimbl# echo \*
*
```

Pomocou metaznaku je možné vytvárať súbory, v ktorých sú medzery. Medzera bola doteraz tiež chápaná ako metaznak, pretože ju shell interpretuje ako oddeľovač argumentov na príkazovom riadku.

```
root@ASUS-PC:/home/kimbl# cat jeden\ argument
cat: jeden argument: No such file or directory
root@ASUS-PC:/home/kimbl# cat viac argumentov
cat: viac: No such file or directory
cat: argumentov: No such file or directory
```

V prvom prípade vzniklo len jedno chybové hlásenie, v druhom nám už vznikli dve hlásenia.

Ďalšia možnosť ako potlačiť expanziu je uzavrieť postupnosť znakov medzi úvodzovky alebo apostrofy. Z pohľadu expanzie sa úvodzovky aj apostrofy chovajú identicky, rozdiel je v chovaní pri premenných.

Premenné

Premenné slúžia shellu na uchovávanie informácií meniacich sa behom práce alebo viazaných sa s prihlásením užívateľa. Premenné v systéme je možné rozdeliť do troch skupín. Prvú skupinu tvoria *vnútorné premenné shellu*, o ich inicializáciu sa stará sám shell. Druhou skupinou sú *užívateľské premenné*, tie môžu byť definované užívateľom podľa potreby. Do tretej skupiny patria *premenné špeciálneho významu*, ktoré uchovávajú informácie o bežiacich procesoch, predávaných argumentoch a podobne.

- **Priradenie:** Príkaz priradenia slúži na definovanie obsahu premennej. Vo väčšine prípadov ide o priradenie tvaru L = R. Výraz L na ľavej strane príkazu reprezentuje referenciu na premennú. V shelly je to názov premennej.

- Príkaz dereferencie sa v shellu robí znakom "$" a bezprostredne za ním nasleduje názov premennej.

```
root@ASUS-PC:/home/kimbl# prvy=obsah
root@ASUS-PC:/home/kimbl# echo $prvy
obsah
```

- Užívateľ najprv definoval premennú prvy s obsahom „obsah". Potom uskutočnil jej dereferenciu, vďaka príkazu echo bola vypísaná hodnota premennej prvy. Ďalej bola premenná odstránená príkazom „prázdneho priradenia".
-

$ premenná	Výraz je expandovaný na hodnotu premennej „premenná"
$ {premenná}	Výraz je expandovaný na hodnotu premennej. Zložené zátvorky slúžia k oddeleniu názvu premennej od jej okolia.
$ {premenná – výraz}	Ak je premenná definovaná, tak výsledkom je jej hodnota. V opačnom prípade je hodnota „výraz".
$ {premenná = výraz}	Ak je premenná definovaná, tak výsledkom je jej hodnota. V opačnom prípade je premenná definovaná na hodnotu „výraz" a potom je vrátená.
$ {premenná + výraz}	Ak je premenná definovaná, je vrátená hodnota „výraz". Ak nie je premenná definovaná, nie je vrátená žiadna hodnota.

```
root@ASUS-PC:/home/kimbl# prvy=ahoj
root@ASUS-PC:/home/kimbl# readonly prvy
root@ASUS-PC:/home/kimbl# prvy=bubu
bash: prvy: readonly variable
```

- Príkaz **readonly** nám slúži na nastavenie premennej "iba na čítanie". Príkaz **readonly** uvedený bez argumentov nám vypíše zoznam všetkých použitých premenných na čítanie.

Premenné špeciálneho významu

Medzi premenné špeciálneho významu patrí skupina troch premenných, ktoré nesú informácie o hodnote PID shellu, ďalej o hodnote PID posledného spusteného procesu na pozadí a o návratovej hodnote procesu. Všetky tri premenné sú iba na čítanie.

Premenná	**Význam**
$$	PID shellu
$!	PID posledného spusteného procesu na pozadí
$?	Návratová hodnota procesu

Oddeľovače podmieneného vykonávania príkazov

O návratovú hodnotu procesu sa opierajú ďalšie dva „oddeľovače" príkazov. Sú to znaky „&&", „||". Slúžia k jednoduchému podmienenému uskutočneniu príkazu. Ich syntax je nasledovná:

<príkaz1> && <príkaz2>
<príkaz1> || <príkaz2>

Oddeľovač && sa chová ako logický súčin, oddeľovač || sa chovajú ako logický súčet.

V systéme existujú dva programy s dopredu definovanou návratovou hodnotou, false a true. Program **false** končí vždy s chybou č. 1, program **true** je vždy úspešný, to znamená, že jeho návratová hodnota je 0. Program false sa chová neutrálne k oddeľovaču ||, program true sa chová neutrálne k programu &&.

Set

- Program **set** sa stará o výpis aktuálne definovaných premenných a ich hodnôt. Program sa spúšťa bez argumentov.

Podmienené príkazy

Podmienené príkazy nám slúžia k vykonávaniu podmienených príkazov. Medzi základné podmienené príkazy patrí: **if ... then ... else**.
Syntax v shelly je nasledovná:

if <výraz>; then <príkazy>; fi
if <výraz>; then <príkazy>; else <alternatíva>; fi
if <výraz>; then <príkazy>; <podmienka> <alternatíva>; fi

kde <podmienka> je ľubovoľná postupnosť alternatívnych podmienok tvaru elif <výraz>;

then <príkazy>;

Druhým podmieneným príkazom je príkaz **case**. Príkaz **case** nám umožňuje vykonať príkazy podmienené „zhodou vstupného slova“ s „požadovanými vzormi“.
Syntax v shelly je nasledovná:

case <slovo> in <vetva> esac

kde <vetva> tvorí neprázdnu postupnosť tvaru vzory príkazy.

Cykly

Príkazy cyklu nám slúžia na spracovanie postupnosti príkazov. Iterácia je zastavená, ak je dosiahnutý limit podmienky. Shell nám umožňuje písať cykly aj bez podmienky. K dispozícií sú nám príkazy: **for, while, until**. Príkazy majú nasledujúcu syntax:

for <premenná> in <zoznam>; do <príkazy>; done
while <výraz>; do <príkazy>; done
until <výraz>; do <príkazy>; done

Spracovanie textu

Dátové súbory sú v Linuxe tradične textové. Dôvodov je niekoľko: textové súbory je možné veľmi jednoducho spracovávať, každú textovú informáciu je možné interaktívne meniť pomocou „celoobrazových textových editorov“.

Základné nástroje

K základným nástrojom patria programy: **cat, head, tail, wc** a ďalšie. Programy čítajú vstupné dáta zo štandardného vstupu a výsledok predávajú na štandardný výstup.

Sort

- Program **sort** nám slúži k utriedeniu textových riadkov súboru. K triedeniu sú využité aj databázy. Program disponuje niekoľkými užitočnými prepínačmi:

Argument	**Význam**
-r	Opačné prehľadávanie
-b	Medzery na začiatku riadku sú ignorované
-c	Riadky sú triedené iba ak sú neutriedené
-f	Ignoruje veľké a malé písmená pri porovnávaní
-g	Dáta sú porovnávané ako čísla
-m	Spájanie už utriedených súborov
-o	Výsledok zapíše do súboru
-R	Triedenie podľa náhodného kľúča

Programu **sort** je tiež možné špecifikovať číslo stĺpca, podľa ktorého by mali byť riadky utriedené. Stĺpec je možné definovať argumentom tvaru + <číslo>. Stĺpce sú číslované od nuly.

```
root@ASUS-PC:/home/kimbl# sort -gr cisla
543
34
9
8
7
5
1
```

Uniq

- Program **uniq** štandardne vypíše všetky „unikáte riadky“ v súbore.

Argument	**Význam**
-d	Sú vypísané opakujúce sa riadky
-c	Sú vypísané počty opakovaní
-u	Sú vypísané iba unikátne riadky

- Program **uniq** číta vstup riadok po riadku a pre vynechanie duplicitných riadkov potrebuje poznať okrem aktuálne spracovávaného riadku iba posledný načítaný riadok.

Porovnávanie dvoch súborov

Cmp

- Program **cmp** nám slúži k porovnaniu obsahu neutriedených súborov. Program nie je obmedzený iba na textové súbory, jeho porovnávanie je univerzálne. Program porovnáva obsah súborov znak po znaku. Ak je nájdený rozdielny znak na tej istej pozícii, program ukončí svoju činnosť a nahlási pozíciu chyby.

```
root@ASUS-PC:/home/kimbl# cmp jablka.txt hrusky.txt
jablka.txt hrusky.txt differ: byte 15, line 1
```

Comm

- Program **comm** nám slúži k porovnaniu obsahu utriedených súborov. Štandardne zobrazuje dáta v troch stĺpcoch. Prvý stĺpec obsahuje riadky, ktoré sa nachádzajú iba „v prvom súbore“. Druhý stĺpec obsahuje riadky, ktoré sa nachádzajú iba „v druhom súbore“. Tretí stĺpec obsahuje dáta nachádzajúce sa v „obidvoch súboroch“ súčasne.

Argument	**Význam**
-1	Potlačenie prvého stĺpca
-2	Potlačenie druhého stĺpca
-3	Potlačenie tretieho stĺpca

Diff

- Program **diff** porovnáva súbory riadok po riadku.

Argument	**Význam**
-c	Zapínanie kontextového výstupného režimu
-r	Umožňuje zaznamenať rozdiely medzi súbormi vo všetkých podadresároch
-e	Vytvára rozdielový súbor pre editor **ed**

Cut

- Program **cut** slúži k výpisu zvolených stĺpcov. Stĺpce je možné definovať dvojitým spôsobom: ohraničené pevnou pozíciou okrajových znakov na riadku alebo pomocou polí oddeľovačov.

Syntax je nasledujúca:

cut –c <stĺpec> <súbor>
cut –d <oddeľovače> - f <pole><súbor>

Paste

- Program **paste** slúži k spojeniu viacej súborov. Program číta jednotlivé riadky zo vstupných súborov a na vstup ich predáva vypísané za sebou. Vlastný oddeľovač je možné definovať argumentom –d <znak>. Program podporuje aj sériový výpis stĺpcov.

Dd

- Program **dd** slúži ku kopírovaniu a konvertovanie súborov.

Argument	**Význam**
Bs = veľkosť	Nastavenie veľkosti jedného bloku
Count = počet	Prevedená kópia iba „počet“ blokov
If = súbor	Vstupné dáta číta zo súboru
Of = súbor	Výstupné dáta zapisuje do súboru
Seak= n	Pri zápise výstupu preskoč prvých „n“ blokov
Skip = n	Pri čítaní vstupu preskoč prvých „n“ blokov

Filtre

Grep, Egrep

- Programy **grep** a **egrep** slúžia na vypísanie riadkov zhodujúcich sa so vzorom. Po spustení sa program podľa svojho mena prepne buď do režimu grep alebo egrep. Program grep umožňuje používať pozitívny uzáver, voliteľný výraz, limitované opakovanie a aj špeciálne znaky ako sú +, ?, {}, (). Je však nutné ich používať so spätným lomítkom \.

Argument	**Význam**
-i	Nerozlišuje veľké a malé písmená
-w	Vypisovaný riadok musí zodpovedať zadanému vzoru
-v	Vypisované riadky nevyhovujú vzoru
-r	Rekurzívne sa prechádzajú adresáre
-l	Vypisujú sa mená súborov

Prúdové editory

Program **sed** je v podstate odvodený od riadkového editoru **ed**. Na rozdiel od neho spracováva jednotlivé riadky prúdovo. Používa sa najčastejšie k nahradzovaniu v texte podľa zadaného vzoru. Program má nasledujúcu syntax:

sed <program>
sed –f <súbor>

Ak sa uvedie argument „n“, program sa chová „ticho“.

Inštrukcie	**Význam**
d	Ukončí spracovávaný aktuálny riadok
n	Spracovávaný riadok sa vypíše na výstup
p	Ukončí spracovávanie aktuálneho riadku a vypíše ho na výstup
a, c, i	Inštrukcie spôsobia vloženie nového textu „za, miesto, pred“ aktuálnu pozíciu v súbore
r	Vloží obsah iného súboru
w	Ukončí spracovanie aktuálneho riadku a vypíše ho do súboru

Program **tr** je určený na preklad alebo vymazanie znakov. Je schopný uskutočňovať podobnú činnosť ako program **sed**. Program spracováva súbor znak po znaku.

Existuje aj program **awk**, ktorý spracováva text tiež prúdovo.

Archivácia a kompresia dát

Program **tar** je štandardný archivačný nástroj v Linuxe. Dokáže archivovať súbory prípadne aj adresáre vrátane svojho obsahu. Archív má podobu jedného súboru, v ktorom sú uložené dáta v nekomprimovanej forme.

Argument	**Význam**
-c	Vytvorenie nového archívu
-d	Porovnanie rozdielov v archíve so súborovým systémom
-r	Pridanie nových dát na koniec archívu
-t	Vypísanie obsahu archívu
-u	Pripojenie aktualizovaných súborov
-x	Extrahovanie dát
-z	Implicitne zapína kompresiu programom Gzip

Program **gzip** používa slovníkový algoritmus LZ77. **Bzip2** používa kombináciu Burrows-Wheelerova triedenia blokov a Huffmanovu kompresiu. Prepínačmi -1 až -9 určujeme mieru kompresie. Ak je použitý -1, tak kompresia je minimálna ale prebehne veľmi rýchlo. Ak je použitý -9, tak je použitá maximálna kompresia. pre dekompresiu existuje v obidvoch programoch prepínač –d. Prepínačom –t kontrolujeme integritu komprimovaných dát. Prepínačom –c vypisujeme výsledok na štandardný výstup.

Sieťové prostredie

Vypracoval: Norbert Špót

Cieľom tejto časti nebude vysvetliť čo sú to protokoly TCP/IP, aká je ich architektúra, ale vysvetliť na čo sú a ako sa používajú niektoré príkazy systému GNU/Linux (alebo aj UNIX).

Základné programy (ping, tracepath)

Program **ping** berie ako argument IP adresu sledovaného rozhrania. Každú sekundu pošle packety a meria odozvu.

```
root@spoti-desktop:/home/spoti# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=128 time=2.95 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=128 time=4.18 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=128 time=3.09 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=128 time=3.07 ms
64 bytes from 192.168.1.1: icmp_seq=5 ttl=128 time=2.77 ms

--- 192.168.1.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 2.771/3.216/4.187/0.501 ms
root@spoti-desktop:/home/spoti#
```

Po stlačení **Ctrl-C** sa vypíše štatistika a program sa ukončí. Počet odoslaných požiadaviek sa dá obmedziť pomocou prepínača **–c <počet>**. V tomto prípade nebude nutné program ukončiť stlačením **Ctrl-C**.

```
root@spoti-desktop:/home/spoti# ping 192.168.1.1 -c 4
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=128 time=6.78 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=128 time=2.89 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=128 time=3.16 ms
64 bytes from 192.168.1.1: icmp_seq=4 ttl=128 time=3.06 ms

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 2.897/3.976/6.782/1.624 ms
root@spoti-desktop:/home/spoti#
```

Na sledovanie približnej trasy packetov slúži napr. program **tracepath.** Ako argument dostáva IP adresu.

```
root@spoti-desktop:/home/spoti# tracepath 192.168.1.3
 1:  spoti-desktop.local (192.168.83.130)                  0.134ms pmtu 1500
 1:  192.168.83.2 (192.168.83.2)                           0.318ms
 2:  no reply
```

Hodnota za **pmtu** znamená maximálny veľkosť súvislého bloku. Ak je cez rozhranie prenášaný blok väčší ako 1500 bajtov tak sa rozdelí na menšie časti a tie sú potom prenášané samostatne.

traceroute – obdoba programu tracepath

Zisťovanie IP adries a symbolických mien

Väčšina klientov robí preklad mien a IP adries automaticky. Keď ale chce užívateľ spraviť preklad ručne môže k tomu požiť program **nslookup**.

Ako argument treba zadať buď symbolické meno, alebo IP adresu.

```
root@spoti-desktop:/home/spoti# nslookup www.ubuntu.com
Server:         192.168.83.2
Address:        192.168.83.2#53

Non-authoritative answer:
Name:   www.ubuntu.com
Address: 91.189.94.8

root@spoti-desktop:/home/spoti# nslookup 91.189.94.8
Server:         192.168.83.2
Address:        192.168.83.2#53

Non-authoritative answer:
8.94.189.91.in-addr.arpa        name = jujube.canonical.com.
```

V prvom prípade sme sa pýtali na IP adresu počítača s meno www.ubuntu.com. V druhom prípade pomocou IP adresy sme chceli zistiť meno príslušného počítača. Z príkladov vidíme, že mená počítačov sa nezhodujú. To nie je nič neobvyklého. Jeden počítač môže mať viac mien, dokonca i v rámci domény.

Program **host** je obdoba programu nslookup, len sa nedá používať v interaktívnom, príkazovom režime.

Synchronizácia času

Synchronizácia času počítačov v sieti nie je ani z ďaleka triviálna záležitosť. Predstavte si napríklad situáciu keď kvôli zlej synchronizácií dôjde pošta skôr ako bola odoslaná ☺. Systémový čas by nemal byť zmenený v skokoch ale plynule. Pre synchronizáciu času v sieti bol vytvorený protocol NTP (Network Time Protokol). Tento protokol sa snaží synchronizovať časové informácie serverov, ktoré sú zaradené do akýchsi úrovní. Servery triedy **stratum 1** sú aktualizované priamo s atómovými hodinami, počítače triedy **stratum 2** zas s týmito servermi atď.

Z užívateľského hľadiska je zaujímavý program **ntpdate** ktorý vypíše stav času na konkrétnom počítači. Program býva používaný správcom systému na jednorazové synchronizovanie času systému. Keď je ale použitý s prepínačom **–q** vypíše len časové údaje.

Program **ntpdate** patrí medzi systémové nástroje, preto bude vo väčšine systémov umiestnený v adresári /**usr/sbin** na ktorý užívatelia spravidla nemajú nastavenú cestu.

```
root@spoti-desktop:/home/spoti# ntpdate -q time.euro.apple.com
server 17.72.255.11, stratum 2, offset -0.002301, delay 0.10181
server 17.72.255.12, stratum 2, offset -0.002711, delay 0.09845
17 Jan 07:24:01 ntpdate[20687]: adjust time server 17.72.255.12 offset -0.002711
 sec
```

Zdieľanie diskového priestoru

Zdieľanie diskového priestoru sa uskutočňuje s jedným už známym príkazom, s použitím systému NFS (Network File System). Tento systém je už priamo zabudovaný v nových kerneloch Linuxu kvôli efektivite. NFS spravidla neposkytuje všetky dáta každému. V súbory **/etc/exports** je zoznam exportovaných adresárov, ktorý určuje ktorým staniciam budú súbory po sieti dostupné. NFS pritom neimplementuje vlastný súborový systém ale beží ako transparentná vrstva. Pripojenie prebehne podobne ako pri pripájaní lokálneho súborového systému len s použitím NFS.

mount –t nfs <čo> <kam>

```
spoti@spoti-desktop:~$ mount -t nfs pocitac.niekde.com:/home/user /home/spoti
```

Posielanie správ užívateľovi

K posielaniu správ užívateľovi poslúži štandardný unixový nástroj **mail**, ktorý nájdete v každom systéme.

mail –s <subjekt> -c <kópia> <adresy>

Prepínače **–s** a **–c** nie sú povinné, keď sa –**s** vynechá správa sa odošle bez subjektu. Program **mail** číta štandardný vstup kým sa nenapíše riadok s jedným znakom bodky „ . “ alebo užívateľ nestlačí **Ctrl+D**. Program **mail** sa zíde napr. v prípade keď spustíte nejaký proces na počítači ale potrebujete odísť. Pošlete správu užívateľom aby pri odchode počítač nevypínali. Táto správa sa im zobrazí pri prihlásení.

Vzdialené prihlasovanie

Dnes klasickou službou pre vzdialené prihlásenie je **Telnet**. Nejedná sa len o program, ale o systém komunikujúci spôsobom klient/server. Klientský program **telnet** sa pripája na počítač na ktorom by mal bežať na porte 23 špeciálny deamon **telnetd**. Program dostáva ako argument IP adresu alebo symbolické meno počítača. Ďalším nepovinným údajom môže byť číslo portu, štandardne sa pripája na 23. Program **telnet** sa dá použiť aj v interaktívnom režime klasickým spustením bez argumentov.

```
spoti@spoti-desktop:~$ telnet unix.dcs.fmph.uniba.sk 44000
Trying 158.195.18.141...
Connected to cvika.dcs.fmph.uniba.sk.
Escape character is '^]'.

Linux 2.4.25 (cvika) (6)


cvika login:
```

Po zadaní príkazu sa telnet pripojí na počítač ktorý vás vyzve na zadanie prihlasovacieho mena a hesla.

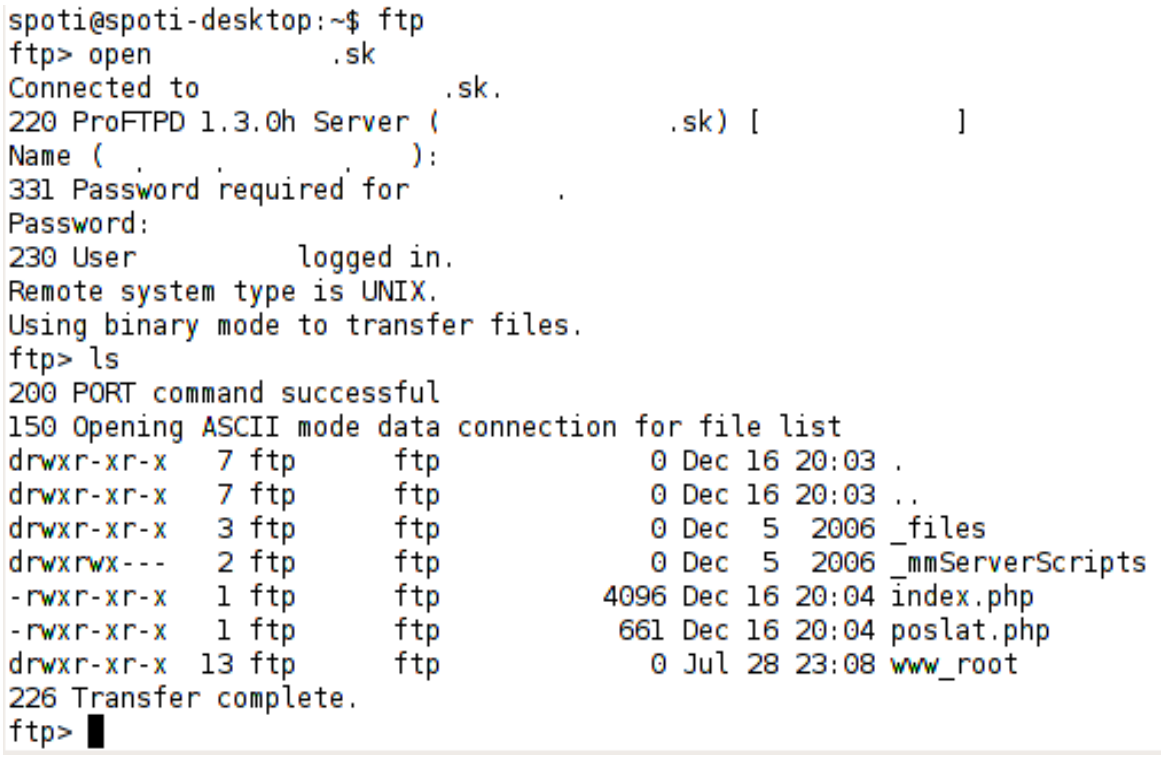
Escape character is '^]'

Toto znamená, že stlačením **Ctrl+],** (ctrl je ^) sa dostanete do príkazového režimu telnetu. Zadaním **close** sa odpojíte, **quit** zas vypne telnet. Príkazom **open** sa môžete pripojiť k serveri.

Prenos dát pomocou FTP

File **T**ransfer **P**rotocol je služba umožňujúca prenášať dáta medzi počítačmi. Prenosu sa opäť zúčastňujú dve strany – FTP server a klient používajúci špeciálny ftp klient. Základný klientský program **ftp** pracuje podobne ako telnet. Má vlastný príkazový režim slúžiaci na spojenie so vzdialeným serverom a prácu so súbormi. Základné príkazy, ktoré sa dajú používať s programom **ftp** sa nachádzajú v nasledujúcej tabuľke.

Príkaz	Význam
cd	zmena pracovného adresára na vzdialenom počítači
get	download jedného súboru
hash	po každom prenesenom bloku je zobrazený znak "#"
help	pomoc
lcd	zmena pracovného adresára na lokálnom počítači
ls	výpis súborov na vzdialenom počítači
mget	download viacerých súborov naraz
mput	upload viacerých súborov naraz
passive	prechod do passívneho režimu
prompt	zapnutie režimu "bez pýtania sa"
put	upload jedného súboru
pwd	vypíše pracovný adresár na vzdialenom počítači



Na obrázku vidíte príklad na pripojenie sa k serveri a výpisu súborov nachádzajúcich sa v koreňovom adresári. Ukončiť program okrem príkazu **quit** sa dá aj príkazom **bye** ☺.

Ďalšie užitočné príkazy:

open	– slúži k naviazaniu spojenia so serverom
close	– ukončuje spojenie so serverom
ascii	– nastavuje režim na ASCII – prenášanie textových súborov (neprenáša sa najvyšší bit)
bin	– nastavuje prenos do binárneho režimu (prenášanie súborov)
delete	– zmaže súbor na vzdialenom počítači
mdelete	– zmaže viacero súborov naraz
rm	– zmaže adresár na vzdialenom počítači
user	– dovoľuje zadať užívateľské meno
!<príkaz>	– dovoľuje zavolať príkaz na lokálnom počítači

Ďalšie príkazy zistíte po zadaní príkazu **help.** Väčšinou sú to štandardné unixové príkazy.

Služba Secure Shell (ssh)

Hlavnou nevýhodou služieb Telnet a FTP je ich nešifrovaný charakter. Heslá sú medzi klientmi prenášané nešifrovane v jednom packete. Na ceste môžu byť odchytené a zneužité potenciálnym útočníkom. V Unixu je v súčasnosti bežne dostupná služba **Secure Shell.** Táto služba je tiež rozdelená na serverovú a klientskú časť. Služba SSH sa neobmedzuje len na vzdialené prihlasovanie. Je to komplexná služba umožňujúca prenášanie dát a komunikáciu cez šifrovaný kanál. K vzdialenému prihláseniu sa používa program **ssh** ktorý nemá na rozdiel od telnetu príkazový režim a tak všetky príkazy treba zadať ako argumenty. Spravidla sa k serveri pripája nasledovným spôsobom:

```
ssh prihlasovacie_meno@server.com
```

Okrem autentifikácie využívajúcej heslo je možné používať aj iné mechanizmy. Bežná je napr. autentifikácia na základe tajného kľúča. Užívateľ si najprv vytvorí kľúč programom **ssh-keygen** a potom ju zaregistruje na vzdialenom počítači. Keď sa užívateľ chce prihlasovať len na základe vlastnenia kľúča tak netreba zadávať žiadnu passprhase.

```
spoti@spoti-desktop:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/spoti/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/spoti/.ssh/id_rsa.
Your public key has been saved in /home/spoti/.ssh/id_rsa.pub.
The key fingerprint is:
b1:c8:de:05:46:75:d8:65:55:16:df:4b:22:ef:1e:99 spoti@spoti-desktop
```

Verejný kľúč je uložený v súbore **~/.ssh/identity.pub** (alebo ~/.ssh/id_rsa.pub). Keď je vygenerovaný kľúč bez hesla stačí pripojiť na koniec súboru **~/.ssh/authorized_keys** (alebo ~/.ssh/known_hosts) na vzdialenom počítači obsah súboru **~/.ssh/identity.pub** (alebo ~/.ssh/id_rsa.pub).

Tento verejný kľúč môže vyzerať nasledovne:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEArC0b5XTdV6f54+MIrn8Oy2pO72VFaS
hm2VU0pK2rgx4fT3rZQZdFJHFplrolXNwGF3cXQRvtUc2OBbx2OZ+5INjTjwsf
j2JeZq/IKXx/9pHpr4PO8HD28JQBNBhSBR1giDr2W7rQGaMMKiSs8giMJPAlCra
o+9VeuZmelVaGmoTO2jtOaXv22oYLCuDdks1U6ADb1QJ7N8GgTi/tg6c9LCK
cn5tpyToadlHJ/xfDRLhjWkZ7xjn5EPCaN5n2v+4Pvn2AuMGO3CkcJVSc9gzRR
coapP9uGeV+dkHr/jADIJC8denRGSmwsowwYiL5ej2TV5IO51zdTKUSfmRfsR
8L7w== spoti@spoti-desktop
```

Komunikácia

V našom prípade pracujeme s príkazovým riadkom a so štandardnými unixovými nástrojmi. Takýto nástroj na komunikáciu unixové systémy štandardne obsahujú. Program sa volá **talk**. Pri komunikácií používajú obidve strany tento klientský program, ktorý prenáša písaný text veľmi efektne hneď ako to píšeme, po znakoch teda nie po riadkoch ako klasické instant messageingové programy.