Lupina

Preprosto rečeno je lupina program, ki uporabnikove ukaze posreduje operacijskemu sistemu, ki ukaze izvede. Včasih je bila lupina edini uporabniški vmesnik, kije bil na voljo v Unixu. Danes poleg 'ukazne vrstice' kot je lupina, obstajajo tudi grafični uporabniški vmesniki.

V Linuxu je privzeta lupina *bash* (Bourne Again SHell, izboljšana Bournova lupina `sh`, ki jo je napisal Steve Bourne). Poleg bash obstaja več dodatnih lupin, ki jih v tipičnem sistemu z Linuxom lahko uporabimo, npr.: *`ksh`*, *`tcsh`* `in` *`zsh`*.

Zagon terminala

Z lupini delamo v terminalskem oknu; terminal oziroma natančneje emulator terminala omogoča uporabniku interakcijo z lupino. V okolju KDE, lahko najdemo "konsole" in "terminal"; Linux lahko uporabljamo tudi brez grafičnega vmesnika preko konzole, npr. (CTRL+ALT+F1/F6, CTRL+ALT+F7)…

Ko poženemo terminal, nam ta ponudi najavko, ki je sestavljen iz imena uporabnika in imena računalnika, ki jima sledi znak $.

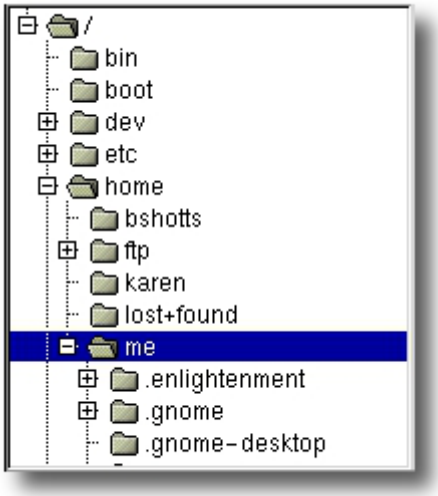
```
[me@linuxbox me]$
```

Uporaba miške

Čeprav v lupini delamo z ukazno vrstico, lahko vseeno uporabljamo miško. Uporabljamo jo za pomikanje s pomočjo drsnikov in kopiranje. Kopiramo tako, pritisnemo levo tipko in z miško potegnemo preko besedila, ki ga želimo kopirati. Besedilo bi moralo biti označeno. Besedilo prekopiramo s pritiskom na srednji gumb (če imamo miško z dvema gumboma, pritisnemo najprej levega in nato še desnega. Nato levi gumb spustimo- tako simuliramo pritisk na srednji gumb…)

Organizacija datotečnega sistema

Datoteke so organizirane v hierarhičnem sistemu direktorijev. To pomeni, da so organizirane v obliki drevesa direktorijev (imenikov, map…), ki vsebujejo datoteke in druge direktorije. Na vrhu drevesa je korenski direktorij (root), ki je označen z /. Če datotečni sistem pogledamo z grafičnim upravljalcem datotek, bi zgledal nekako tako:



Pomembna razlika glede na nekatere druge sisteme je, da v Linuxu ni pogonov, ki so običajno označeni s črkami npr. A:, C:… Vsak pogon ima svoj datotečni sistem, v Linuxu pa je datotečni sistem samo eden. Različni pogoni predstavljajo veje v drevesu direktorijev, drevo direktorijev pa je vedno samo eno.

pwd

Direktorij, v katerem se nahajamo, se imenuje delovni direktorij (*working directory*). Z ukazom *pwd* (print working directoriy) izvemu, v katerem direktoriju se trenutno nahajamo.

```
[me@linuxbox me]$ pwd
/home/me
```

Ko se prijavimo v Linux, se znajdemo v domačem direktoriju. Na večini sistemov je to. /home/uporabniško_ime, vendar lahko sistemski administrator domač direktorij določi tudi drugače.

Za izpis datotek v trenutnem direktoriju uporabljamo ukaz *ls*.

```
[me@linuxbox me]$ ls


Desktop     Xrootenv.0    linuxcmd
GNUstep     bin           nedit.rpm
GUILG00.GZ  hitni123.jpg  nsmail
```

cd

Ukaz *cd* uporabljamo za zamenjavo trenutnega direktorija. Ukazu `cd` kot parameter podamo *pot* do novega delovnega direktorija. Pot lahko podamo na dva načina: absolutno in relativno, glede na trenutni direktorij. Absolutna pot se začne s korenskim direktorijem, ki mu sledijo z znakom / ločeni direktoriji, preko katerih pridemo do podanega direktorija, npr. cd /usr/X11R6/bin. To pomeni, da je v korenskem direktoriju (ki ga predstavlja /) direktorij *usr*, v katerem je direktorij *X11R6*, ki vsebuje direktorij z imenom *bin*. Primer:

```
[me@linuxbox me]$ cd /usr/X11R6/bin

[me@linuxbox bin]$ pwd
/usr/X11R6/bin

[me@linuxbox bin]$ ls

Animate                 import                  xfwp
AnotherLevel            lbxproxy                xg3
Audio                   listres                 xgal
Auto                    lndir                   xgammon
Banner                  makedepend              xgc
Cascade                 makeg                   xgetfile
Clean                   mergelib                xgopher
Form                    mkdirhier               xhexagons
Ident                   mkfontdir               xhost
Pager                   mkxauth                 xieperf
Pager_noxpm             mogrify                 xinit
RunWM                   montage                 xiterm
RunWM.AfterStep         mtv                     xjewel
RunWM.Fvwm95            mtvp                    xkbbell
RunWM.MWM               nxterm                  xkbcomp
```

Relativna pot se začenja v delovnem direktoriju. Relativni položaj označujemo z dvema posebnima simboloma "." in "..".

"." se nanaša na delovni direktorij, ".." pa predstavlja direktorij na nivoju nad delovnim direktorijem. Primer:

```
[me@linuxbox me]$ cd /usr/X11R6/bin
[me@linuxbox bin]$ pwd
/usr/X11R6/bin
```

Če želimo spremeniti delovni direktorij v /usr/X11R6, lahko to dosežemo na dva načina:

```
[me@linuxbox bin]$ cd /usr/X11R6
[me@linuxbox X11R6]$ pwd
/usr/X11R6
```

Ali s pomočjo relativne poti:

```
[me@linuxbox bin]$ cd ..
[me@linuxbox X11R6]$ pwd
/usr/X11R6
```

Podobno se lahko iz /usr/X11R6 pomaknemo v /usr/X11R6/bin na dva načina. Najprej z uporabo absolutne poti:

```
[me@linuxbox X11R6]$ cd /usr/X11R6/bin
[me@linuxbox bin]$ pwd
/usr/X11R6/bin
```

Ali s pomočjo relativne poti:

```
[me@linuxbox X11R6]$ cd ./bin
[me@linuxbox bin]$ pwd
/usr/X11R6/bin
```

V skoraj vseh primerih lahko "./" na začetku relativne poti izpustimo. Če bi natipkali:

```
[me@linuxbox X11R6]$ cd bin
```

bi dosegli isti rezultat. Na splošno, če določene poti (./) ne napišemo, se smatra, da gre za delovni direktorij.

Nekaj bližnjic

Če natipkamo samo `cd` (brez parametra - poti), `pridemo v domač direktorij.`

Podobno s `cd ~`*`uporabniško_ime`*. Pridemo v domači direktorij določenega uporabnika..

Imena datotek

1. Datoteke, katerih imena se začenjajo s piko, so skrite. To pomeni samo, da jih ukaz *ls* ne bo izpisal, razen če uporabimo stikalo `ls -a`. Različne aplikacije uporabljajo skrite datoteke in direktorije za shranjevanje nastavitev.

2. Pri imenih datotek Linux razlikuje velike in male črke.
3. V Linux ni koncepta končnic datotek. Vsebina oziroma namen datotek je določena na drugačen način.
4. Čeprav Linux podpira dolga imena datotek, ki lahko vsebujejo tudi presledke, uporaba presledkov ni priporočljiva. Bolje je uporabiti pomišljaj, piko ali podčrtaj.

ls

Z ukazom ls izpišemo vsebino trenutnega direktorija. Uporabljamo jo lahko na več načinov, npr.:

Primeru uporabe ukaza ls

Ukaz	***Rezultat***
`ls`	Seznam datotek in poddirektorijev v trenutnem direktoriju
`ls /bin`	Izpis vsebine direktorija /bin
`ls -l`	Izpis vsebine trenutnega direktorija v dolgem formatu
`ls -l /etc /bin`	Izpis vsebine direktorijev /etc in /bin v dolgem formatu
`ls -la ..`	Izpis vseh datotek in poddirektorijev (tudi skritih – katerih ime se začne s .) v direktoriju za en nivo višje v dolgem formatu

Primeri nakazujejo koncept večine ukazov v lupini:

```
command -options arguments
```

kjer je *command* ime ukaza, *-options* pa ena ali več opcij, ki prilagodijo delovanje ukaza, in *arguments* ena ali več "stvari"nad katerimi ukaz deluje.

V primeru `ls`, je `ls` ime ukaza, ki mu lahko dodamo eno ali več opcij, kot sta `-a` in `-l`, in dals lahko deluje nad enim ali več direktoriji..

Podrobnejši opis dolgega formata izpisa

Če z uporabimo opcijo `-l`, z ukazom ls pridemo do množice informacij. Izpis je naslednji:

```
-rw-------   1 bshotts  bshotts       576 Apr 17  1998 weather.txt
drwxr-xr-x   6 bshotts  bshotts      1024 Oct  9  1999 web_page
-rw-rw-r--   1 bshotts  bshotts    276480 Feb 11 20:41 web_site.tar
-rw-------   1 bshotts  bshotts      5743 Dec 16  1998 xmas_file.txt

----------     -------  -------  -------- ------------ -------------
    |             |        |         |         |             |
    |             |        |         |         |       Ime datoteke
    |             |        |         |         |
    |             |        |         |         +---   Čas spremembe
    |             |        |         |
```

```
    |             |        |         +------------- Velikost v bajtih
    |             |        |
    |             |        +-----------------------        Skupina
    |             |
    |             +-------------------------------        Lastnik
    |
    +---------------------------------------------       Dovoljenja
```

Ime datoteke
: Ime datoteke ali direktorija.

Čas spremembe
: Čas, ko je bila datoteka zadnjič spremenjena. Če je od zadnje spremembe poteklo več kot pol leta, se izpišeta samo datum in leto spremembe.

Velikost
: Velikost datoteke v bajtih.

Skupina
: Ime skupine, ki ima dovoljenja za delo z datoteko, poleg lastnika.

Lastnik
: Ime uporabnika, ki je lastnik datoteke.

Dovoljenja
: Predstavitev dovoljenj za dostopanje do datoteke. Prva črka označuje tip datoteke. "-" predstavlja navadno datoteko, če je prva črka "d", gre za direktorij. Naslednji trije znaki predstavljajo dovoljenja za branje, pisanje in izvajanje datoteke. Ta dovoljenja se nanašajo na lastnika datoteke.Naslednji trije znaki so dovoljenja, ki se nanašajo na skupino, zadnji trije pa so dovoljenja za vse ostale uporabnike.

less

`je program za pregledovanje datotek z besedilom`. Poženemo ga preprosto kot

```
less text_file
```

Možnosti less

Ko ga poženemo, `less` prikaže datoteko z besedilo stran za stranjo. Med stranmi se pomikamo s tipkama Page Up in Page Down. Pregledovanje končamo s "q". `less` sprejme še nekaj ukazov, našteti so najpogosteje uporabljani:

Ukazi programa less

Ukaz	***Rezultat***
Page Up ali b	Pomik za eno stran navzgor

Page Down ali presledek	Pomik za eno stran navzdol
G	Pojdi na konec datoteke
1G	Pojdi na začetek datoteke
/*characters*	Išči prvo naslednjo pojavitev niza, določenega s *characters*
n	Ponovi iskanje
q	Izhod iz less

file

S tem ukazom izvemo, kakšna je vsebina datotek oziroma za kakšno vrsto datoteke gre. Primer:

```
file name_of_file
```

Sprehod po datotečnem sistemu

V spodnji tabeli je seznam zanimivih direktorijev s kratkim opisom, kaj je v njih. Po direktorijih se sprehodimo z ukazom *cd*, vsebino izpišemo z *ls*. Če nas katera datoteka zanima, sz ukazom *file* ugotovimo, za kakšno vrsto datoteke gre in če gre za datoteko z besedilom, jo izpišemo z ukazom *less*.

Direktoriji in njihova vsebina

Direktorij	***Opis***
`/`	Korenski direktorij, začetek datotečnega sistema. Največkrat so v njem samo poddirektorji .
`/boot`	V njem se nagaja jedro operacijskega sistema, običajno v datoteki

	`vmlinuz`.

`/etc`	Direktorij `/etc` vsebuje konfiguracijske datoteke za operacijski sistem in druge programe. Večina datotek je tekstovnih. Posebej zanimive:
	`/etc/passwd`
	Datoteka `passwd` vsebuje pomembne podatke o uporabnikih. Z njo so uporabniki definirani.
	`/etc/rc.d`
	Vsebuje skripte, ki se poženejo ob zagonu sistema

`/bin, /usr/bin`	Ta direktorija vsebujeta večino programov, ki so na voljo. V `/bin` so programi, ki so bistveni za delovanje sistema, v `/usr/bin` pa so uporabniški programi.
`/sbin, /usr/sbin`	Vsebujeta programe za sistemsko administracijo. Te programe lahko poganja administrator, 'superuser'.

`/usr`	Vsebuje mnogo stvari za podporo uporabniškim aplikacijam. Nekaj pomembnejših:
	`/usr/X11`
	Sistem X Windows
	`/usr/dict`
	Slovarji za preverjanje črkovanja. Linux ima črkovalnik, glej *look* in *ispell*.
	`/usr/doc`
	Razna dokumentacija v različnih formatih.
	`/usr/man`
	Strani priročnika man.
	`/usr/src`
	Izvorna koda, npr izvorna koda jedra Linuxa.

`/usr/local`	`/usr/local` in poddirektorije se uporablja za namestitev progamske opreme za lokalno uporabo. V resnici to pomeni zgolj, da tu nameščena programska oprema ni del distribucije Linuxa, ampak smo jo namestili naknadno…. KO nameščamo programe, bi jih morali namestiti v direktorije pod `/usr/local`. Najpogosteje je to `/usr/local/bin`.

`/var`	Vsebuje datoteke, ki se med delovanjem sistema spreminjajo:
	`/var/log`
	Vsebuje datoteke log. Te se dopolnjujejo, ko sistem deluje.
	`/var/spool`
	Vsebujejo datoteke, ki čakajo na določen proces, npr. sporočila elektronske pošte, opravila tiskalnika….

/lib	Knjižnice (podobne DLLjem v Microsoftovih Oknih).
/home	Mesto, kamor uporabniki shanjujejo svoje delo. Običajno je to edino mesto, kamor uporabniki smejo zapisovati podatke.
/root	Domači direktorij administratorja.
/tmp	/tmp je direktorij za zapisovanje začasnih datotek.
/dev	/dev je poseben direktorij, v resnici ne vsebuje datotek v običajnem pomenu. V resnici vsebuje naprave, ki so v sistemu na voljo. Do teh naprav dostopamo enako kot do datotek.
/proc	Tudi direktorij /proc je posebenl. Tudi ta ne vsebuje datotek. Pravzaprav v resnici sploh ne obstaja, je navidezen. Preko njega lahko 'gledamo' jedro operacijskega sistema. Npr. v njem je skupina oštevilčenih 'datotek' ki predstavljajo procese, ki se izvajajo v sistemu. Preko drugih lahko vidimo konfiguracijo sistema. Npr. v /proc/cpuinfo so podatki o centralni procesni enoti…
/mnt	/mnt je običajen direktorij, ki pa se uporablja na poseben način. Nanj lahko priklopimo (mount) različne naprave, kot so CD-ROM, gibki disk…

Delo z datotekami

Osnovni ukazi za delo z datotekami in direktoriji so naslednji:

- cp – kopiranje datotek in direktorijev
- mv – premikanje in preimenovanje datotek in direktorijev
- rm – brisanje datotek in direktorijev
- mkdir - ustvarjanje direktorijev

Pravzaprav lahko vse našteto opravimo z grafičnim upravljalcem datotek. Zakaj potem uporabljati te ukaze?

Odgovor je v fleksibilnosti. Zahtevnejše naloge lažje kot z upravljavcem datotek dosežemo z ukazi. Npr. Kako kopiratti vse datoteke tipa HTML iz enega v drug direktorij, vendar samo tiste, ki v ciljnem direktoriju ne obstajajo ali so novejše od tistih v ciljnem direktoriju? Z upravljavcem datotek težko. Uporabimo:

```
[me@linuxbox me]$ cp -u *.html destination
```

Posebni znaki

Ker v lupini veliko delamo z imeni datotek in direktorijev, nam lupina omogoča načine za hitro sklicevanje na skupine datotek. To naredimo s pomočjo posebnih znakov. Ti nam omogočajo podajanje imen datotek, ki ustrezajo določenemu vzorcu. Posebni znaki so našteti v tabeli:

Povzetek posebnih znakov in njihov pomen

Znak	***Pomen***
*	Ustreza kateremukoli zaporedju znakov, tudi praznemu.
?	Ustreza kateremukoli posameznemu znaku.
[characters]	Ustreza kateremukoli znaku izmed naštetih. Naštejemo lahko tudi območja znakov, npr. [A-Z] predstavlja velike črke.
[!characters]	Ustreza kateremukoli znaku, ki **ni** med naštetimi.

Z uporabo posebnih znakov lahko opišemo razmeroma zapletene kriterije za izbiro datotek. Naštetih je nekaj primerov:

Primeri uporabe posebnih znakov pri imenih datotek

Vzorec	***Ujemanje***
*	Vse datoteke
*g**	Datoteke, ki se začnejo z "g"
b.txt*	Datoteke, ki se začnejo z "g". in končajo z ".txt"

Data???	Vse datoteke, ki se začenjajo z nizom "Data", ki mu sledijo še natančno trije znaki
*[abc]**	Datoteke, ki se začnejo na "a", "b" ali "c"
*[A-Z]**	Vse datoteke, ki se začnejo z veliko črko.
BACKUP.[0-9][0-9][0-9]	Datoteke, ki se začnejo z nizom "BACKUP.", ki mu sledijo natančno tri cifre.
*[!a-z]**	Vse datoteke, ki se ne začnejo z malo črko.

Posebne znake lahko uporabljamo v povezavi z vsemi ukazi, ki kot argumente sprejemajo imena datotek.

cp

uporabljamo za kopiranje datotek in direktorijev. V osnovni obliki `cp` kopira eno datoteko:

```
[me@linuxbox me]$ cp file1 file2
```

Lahko ga uporabimo za kopiranje več datotek v drug direktorij:

```
[me@linuxbox me]$ cp file1 file2 file3 directory
```

Še nekaj uporabnih primerov uproabe ukaza `cp` in njegovih opcij:

Primeri uporabe cp

Ukaz	***Rezultat***
cp file1 file2	Kopira datoteko *file1* v *file2*. Če datoteka *file2* ne obstaja, jo cp ustvari; sicer, datoteko *file2* prepiše z vsebino *file1*.
cp -i file1 file	Enako kot v prejšnjem primeru; uporaba "-i" (interactive) pomeni

2	naslednje: če *file2* že obstaja, cp uporabnika vpraša, če jo želi prepisati z vsebino of *file1*.
cp file1 dir1	Kopiraj datoteko *file1* (v datoteko z imenom *file1*) v direktoriju *dir1*.
cp -R dir1 dir 2	Kopiraj vsebino direktorija *dir1*. Če direktorij *dir2* ne obstaja, ga cp ustvari. Sicer ustvari direktorij z imenom *dir1* znotraj direktorija *dir2*.

mv

uporabljamo na dva načina. prvi je premikanje ene ali več datotek v drug direktorij, drugi pa je preimenovanje datotek oziroma direktorijev. Za preimenovanje datoteke ga uporabljamo tako:

```
[me@linuxbox me]$ mv filename1 filename2
```

In za premikanje datotek v drug direktorij:

```
[me@linuxbox me]$ mv file1 file2 file3 directory
```

Primeri uporabe:

Primeri uporabe mv

Ukaz	***Rezultat***
mv file1 file2	Če *file2* ne obstaja, to pomeni preimenovanje *file1* v *file2*. Če datoteka *file2* obstaja, se njena vsebina nadomesti z vsebino *file1*.
mv -i file1 file2	Podobno kot prejšnji primer; uporaba "-i" (interactive) pomeni naslednje: če *file2* že obstaja, mv uporabnika vpraša, če jo želi prepisati z vsebino *file1*.
mv file1 file2 file3 dir1	Premakne *file1*, *file2*, *file3* v direktorij *dir1*. *dir1* mora obstajati, sicer mv javi napako.

mv dir1 dir2	Če direktorij *dir2* ne obstaja, se *dir1* preimenuje v *dir2*. Če *dir2* obstaja, se v njem ustvari direktorij *dir1*.

rm

Ukaz `rm` briše datoteke ali direktorije.

```
[me@linuxbox me]$ rm file
```

Podobno lahko pobrišemo tudi direktorij:

```
[me@linuxbox me]$ rm -r directory
```

Primeri uporabe:

Primeri uporabe rm

Ukaz	***Rezultat***
rm file1 file2	Pobriši datoteki *file1* in *file2*.
rm -i file1 file2	Podobno, le da nas rm za vsako datoteko vpraša, če jo želimo pobrisati.
rm -r dir1 dir2	Pobriše direktorija *dir1* in *dir2* skupaj z njuno vsebino.

mkdir

`mkdir` uporabljamo za ustvarjanje direktorijev. Uporaba je preprosta:

```
[me@linuxbox me]$ mkdir directory
```

Preusmeritev vhoda in izhoda

Večina ukazov, npr. `ls` izpisuje rezultate na ekran. Vendar to ni nujno tako. Izhod ukazov je možno preusmeriti v datoteke, naprave, in celo v vhod drugih ukazov.

Standardni izhod

Večina programov ukazne vrstice prikazuje rezultate tako, da jih pošlje na standardni izhod. Privzeto to pomeni izpis na ekran. Če želimo standardni izhod preusmeriti v datoteko, uporabimo znak ">" na naslednji način:

```
[me@linuxbox me]$ ls > file_list.txt
```

Izvrši se `ls`, rezultat pa se zapiše v datoteko z imenom file_list.txt. Ker je bil izhod `ls` preusmerjen v datoteko, rezultatov ne izpiše na ekranu.

Vsakič, ko ponovimo zgornji ukaz, se datoteka file_list.txt prepiše (v celoti) z izhodom ukaza `ls`. Če želimo rezultat dodati k prejšnji vsebini file_list.txt, uporabimo ">>":

```
[me@linuxbox me]$ ls >> file_list.txt
```

Sedaj se rezultat doda na konec datoteke, datoteka je torej z vsako uporabo ukaza daljša. Če datoteka, kateri želimo rezultate dodati še ne obstaja, se datoteka ustvari.

Izhod za napake

Poleg standardnega izhoda obstaja tudi poseben izhod, namenjen izpisovanju napak, do katerih pride pri izvajanju programov. Privzeto se napake izpisujejo na ekran. Tudi ta izhod lahko preusmerimo v datoteko z uporabo "2>" in "2>>".

Standardni vhod

Veliko ukazov sprejema vhodne podatke s standardnega vhoda. Privzeto je standardni vhod tipkovnica, vendar ga lahko, podobno kot standardni izhod, preusmerimo. Za preusmeritev standardnega vhoda uporabimo znak "<" na naslednji način:

```
[me@linuxbox me]$ sort < file_list.txt
```

V tem primeru smo uporabili ukaz sort tako, da deluje nad vsebino datoteke file_list.txt. Rezultat se izpiše na ekran, saj standardnega izhoda nismo preusmerili. Če želimo preusmeriti tudi tega, to storimo tako:

```
[me@linuxbox me]$ sort < file_list.txt > sorted_file_list.txt
```

Vrstni red preusmeritev ni pomemben. Edina omejitev je, da znaka za preusmeritev ("<" in ">") pišemo za vsemi ostalimi opcijami in argumenti ukaza.

Cevi

Preusmerjanje vhoda in izhoda v večini primerov uporabljamo za povezovanje večih ukazov; izhod nekega ukaza preusmerimo v vhod drugega. To naredimo s pomočjo cevi. Namesto zaporedja ukazov

```
[me@linuxbox me]$ ls >> file_list.txt
[me@linuxbox me]$ sort < file_list.txt > sorted_file_list.txt
```

lahko uporabimo cev in izhod ls preusmerimo v vhod sort brez 'vmesne' datoteke file_list.txt:

```
[me@linuxbox me]$ ls | sort > sorted_file_list.txt
```

Zelo uporabno za izpisovanje vsebine direktorija z mnogo datotekami je naslednje:

```
[me@linuxbox me]$ ls -l | less
```

V tem primeru rezultat `ls` preusmerimo v `less`. Z uporabo `"| less"` lahko pregledujemo rezultate vseh ukazov.

Še nekaj pogostih primerov uporabe cevi:

Primeri ukazov v povezavi s cevmi

Ukaz	***Delovanje***
`ls -lt | head`	Izpiše 10 najnovejših datotek v trenutnem direktoriju.
`du | sort -nr`	Izpiše seznam direktorijev in prostor, ki ga zasedajo, urejeno od največjega proti najmanjšemu.
`find . -type f -print | wc -l`	Izpiše število datotek v direktoriju in njegovih poddirektorijih.

Filtri

So programi, ki jih pogosto uporabljamo v povezavi s cevmi. Filtri nad standardnim vhodom izvedejo določene operacije in rezultat pošljejo na standardni izhod. Predstavljajo močno orodje za obdelavo rezultatov drugih ukazov. V nadaljevanju je naštetih nekaj pogosto uporabljanih filtrov.

Pogosti filtri

Filter	***Delovanje***
sort	Uredi standardni vhod in urejen rezultat izpiše na standardni izhod.
uniq	V urejenem toku podatkov s standardnega vhoda odstrani duplikate. Najpogosteje se uporablja skupaj s filtrom sort.
grep	Pregleda vse vrstice podatkov, ki jih sprejme s standardnega vhoda in izpiše tiste, ki ustrezajo določenemu vzorcu.
fmt	Bere standardni vhod in na standardni izhod izpiše oblikovan (fomatiran) tekst.
head	Izpiše prvih nekaj vrstic, ki jih sprejme s standardnega vhoda. Uporabno, če želimo priti do glave datoteke.
tail	Izpiše nekaj zadnjih vrstic, ki jih sprejme s standardnega vhoda. Uporabno, če želimo priti do zadnjih zapisov datoteki.
tr	Zamenjuje posamezne znake. Lahko ga uporabljamo za spremembo velike/male črke ali npr. zamenjavo znakov za konec vrstice (datoteke v DOSu uporabljajo drugačne znake za konec vrstice kot tiste v Uniu). Uporabno tudi za pretvorbe šumnikov med različnimi standardi…
sed	Stream editor. Za zahtevnejše zamenjave kot tr.
awk	Programski jezik za konstruiranje različnih filtrov. Zelo močno orodje

Primer uporabe cevi

Tiskanje datotek iz ukazne vrstice z ukazom lpr, ki podatke s standardnega vhoda pošlje na tiskalnik. Za oblikovanje izpisa uporabimo cevi in filtre:

```
cat poorly_formatted_report.txt | fmt | pr | lpr

cat unsorted_list_with_dupes.txt | sort | uniq | pr | lpr
```

V prvem primeru smo uporabili `cat`, ki datoteko izpiše na standardni izhod, ki ga preko cevi preusmerimo v standardni vhod filtra `fmt`. `fmt` oblikuje besedilo v odstavke itd. in rezultat pošlje filtru `pr`. `pr` besedilo razporedi na strani in ga pošlje ukazu `lpr`. `lpr` končni rezultat izpiše na tiskalniku.

V drugem primeru začnemo z neurejenim seznamom, ki vsebuje tudi duplikate. Najprej `cat` seznam pošlje filtru `sort`, ki ga uredi in pošlje `uniq`, ki odstrani duplikate. Nato `pr` seznam razdeli na strani, `lpr` pa seznam natisne.

Pravice

Unix je večuporabniški operacijski system. Isti računalnik lahko hkrati uporablja več uporabnikov. Pravice so mehanizem, ki omogoča zaščito med različnimi uporabniki. Vsak uporabnik npr. ne sme imeti pravice do zaustavitve sistema, saj bi s tem onemogočil delo ostalim uporabnikom. Prav tako ne bi bilo dobro, če bi uporabniki lahko drug drugemu neovirano spreminjali in brisali datoteke. Osnovni ukazi za določanje pravic so naslednji:

- `chmod` – določi pravice za delo z datotekami
- `su` – začasno postani 'superuser'
- `chown` – zamenjaj lastnika datoteke
- `chgrp` – zamenjaj skupino, kateri pripada datoteka

Zaščita datotek

Vsaka datoteka in direktorij v Unixu ima attribute, ki določajo pravice, ki jih imajo nad datoteko oziroma direktorijem lastnik datoteke, skupina, kateri datoteka pripada in vsi ostali. Vsakemu od naštetih določimo pravico do branja datoteke, do pisanje v datoteko in pravico do izvajanja (poganjanja kot program) datoteke.

Pravice za posamezno datoteko in direktorij si lahko ogledamo z ukazom ls -l:

```
[me@linuxbox me]$ ls -l some_file


-rw-rw-r-- 1 me   me   1097374 Sep 26 18:48 some_file
```

Iz izpisa lahko razberemo naslednje:

- Lastnik datoteke "some_file"je uporabnik "me"

- Uporabnik "me" lahko datoteko bere in v njo piše
- Datoteka pripada skupini "me"
- Vsi člani skupine "me" prav tako lahko berejo in pišejo datoteko
- Vsi ostali lahko datoteko samo berejo

Podobno si lahko ogledamo še datoteko bash:

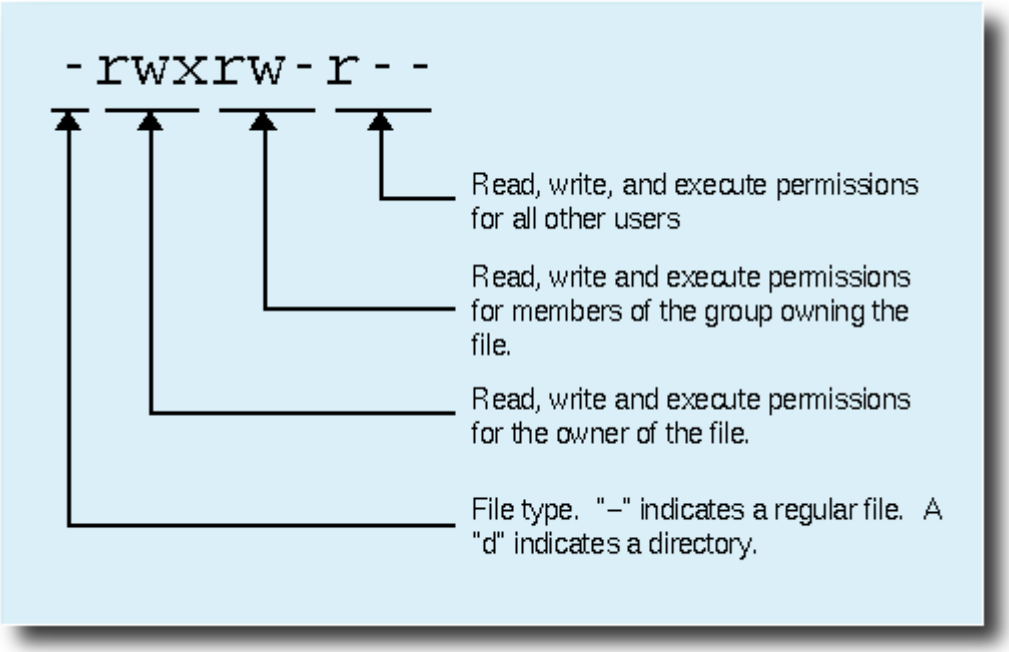
```
[me@linuxbox me]$ ls -l /bin/bash


-rwxr-xr-x 1 root root  316848 Feb 27  2000 /bin/bash
```

Razberemo lahko naslednje:

- Lastnik "/bin/bash" je uporabnik "root" – 'superuser'
- "root" lahko datoteko bere, vanjo piše in jo izvaja (poganja)
- Datoteka pripada skupini "root"
- Člani skupine "root" lahko datoteko berejo, vanjo pišejo in jo izvajajo
- Vsi ostali lahko datoteko berejo in jo izvajajo

Na sliki je prikazano, kako se interpretira izpis s pravicami do datoteke. V izpisu je znak, ki določa tip datoteke, sledijo pa mu tri skupine znakov (r)ead (w)rite in e(x)ecute, ki določajo pravice do branja pisanja in izvajanja datoteke za lastnika, skupino in vse ostale.



chmod

Z ukazom chmod določimo pravice za datoteko oziroma direktorij. Pri uporabi moramo navesti želene pravice in datoteko oziroma datoteke, za katere želimo te pravice določiti. Pravice lahko navedemo na dva načina. Prvi je oblike

chmod ugoa+-rwx file

prvi parameter določa komu (u)ser, (g)roup, (o) others, (a)ll želimo (+)dodati ali odvzeti (-) pravice (r)ead, (w)rite, e(x)ecute. V prvem delu navedemo eno ali več črk, ki določajo komu bomo pravice spreminjali, sledi +/- za dodajanje oziroma odvzemanje pravic in ena ali več črk, ki določi pravice, ki jih bomo spremenili. Primer

chmod ug+x file

pomeni, da smo u-uporabniku in g-skupini dali pravico do izvajanja datoteke file. Pravice za lastnika, skupino in vse ostale lahko podamo tudi numerično.

Pravice si predstavljamo kot zaporedje bitov (tako jih vidi tudi sistem):

```
rwx rwx rwx = 111 111 111
rw- rw- rw- = 110 110 110
rwx --- --- = 111 000 000

itd...

rwx = 111 dvojiško = 7
rw- = 110 dvojiško = 6
r-x = 101 dvojiško = 5
r-- = 100 dvojiško = 4
```

Če si pravice za vsakega (lastnik, skupina, in ostale) predstavljamo kot eno cifro, je to zelo prikladen način za določanje pravic. Če želimo datoteki `file` določiti pravice read in write za lastnika, ostalim pa ne želimo omogočiti dostopa do datoteke, to storimo tako:

```
[me@linuxbox me]$ chmod 600 file
```

Zaščita direktorijev

Direktorij je v resici samo posebna vrsta datoteke, zato nastavitve pravic delujejo na enak način. To ne velja samo za pravico do izvajanja, ki določa pravico do izpisovanja direktorija. Kot primer je naštetih nekaj pogostih nastavitev pravic za direktorije:

Vrednot	*Pomen*
777	*(rwxrwxrwx)* Ni omejitev. Vsak lahko izpiše vsebino, ustvarja nove datoteke in briše datoteke v direktoriju.

755	*(rwxr-xr-x)* Lastnik direktorija ima neomejene pravice. Vsi ostali lahko izpisujejo vsebino direktorija, vendar ne morejo ustvarjati novih datotek v njem oziroma jih brisati. To je običajna nastavitev za direktorije, ki jih želimo deliti z drugii uporabniki.
700	*(rwx------)* Lastnik direktorija ima neomejene pravice. Vsi ostali nimajo nobenih pravic. Uporabno za direktorije, ki jih želimo skriti pred ostalimi

Kako začasno postati 'superuser'

Običajno smo iz varnostnih razlogov v sistem prijavljeni kot navaden uprabnik brez administratorskih pravic. Včasih imamo zato premalo pravic, da bi opravili določeno administracijo in želimo začasno postati 'superuser', da lahko administracijo opravimo. To naredim z ukazom su (superuser). Vnesti moramo samo še ustrezno geslo.

```
[me@linuxbox me]$ su
Password:
[root@linuxbox me]#
```

P izvedbi ukaza `su`, smo prijavljeni kot 'superuser'. Ko končamo z opravili, ki jih moramo opraviti kot 'superuser' sejo zaključimo z ukazom exit in zopet smo prijavljeni kot navadni uporabnik.

Sprememba lastnika datoteke

Lastnika datoteka je mogoče zamenjati z ukazom `chown`. Primer: lastnik datoteke `some_file` je uporabnik "me", želimo, da lastnik te datoteke postane uporabnik "you":

```
[me@linuxbox me]$ su
Password:
[root@linuxbox me]# chown you some_file
[root@linuxbox me]# exit
[me@linuxbox me]$
```

Za spremembo lastnika datoteke moramo biti prijavljeni kot 'superuser'. `chown` na enak način kot z datotekami deluje tudi z direktoriji.

Sprememba skupine, kateri pripada datoteka

Skupino, ki je lastnik določene datoteke, zamenjamo z ukazom `chgrp`. Ukaz uporabljamo podobno kot chown:

```
[me@linuxbox me]$ chgrp new_group some_file
```

Za izvajanje chgrp moramo biti lastnik datoteke oziroma direktorija.

Nadzor nad opravili

Linux je večopravilni (multitasking) sistem.V nadaljevanju je opisanih nekaj ukazov za nadzor nad opravili iz ukazne vrstice. Kot v vseh večopravilnih operacijskih sistemih, v Linuxu hkrati teče več procesov. V resnici gre za hitro preklapljanje med procesi, naenkrat se izvaja le en proces. Ukazi za nadzor nad procesi so naslednji:

- `ps` – izpis procesov, ki se trenutno izvajajo
- `kill` – pošlji signal enemu ali več procesom (običajno za 'ubijanje' procesa)
- `jobs` – drug način za izpis lastnih procesov uporabnika
- `bg` – pomakni proces v ozadje
- `fg` – pomakni proces v ospredje

Primer

Večino programov, ki tečejo v okolju X Windows, je mogoče pognati iz ukazne vrstice. Za zgled poženimo programček za merjenje obremenjenosti sistema xload:

```
[me@linuxbox me]$ xload
```

Pojavi se okno programa `xload`, v katerem se izrisuje graf obremenjenosti sistema. V ukazni vrstici se odzivnik ne pojavi – lupina čaka, da se xload zaključi. Če okno xload zapremo, se odzivnik zopet pojavi.

Izvajanje programa v ozadju

Tokrat bomo `xload` pognali v ozadju. Odzivnik se bo v tem primeru pojavil takoj. Program v ozadju poženemo z uporabo znaka "&":

```
[me@linuxbox me]$ xload &
[1] 1223

[me@linuxbox me]$
```

Kaj pa če pozabimo uporabiti "&"? V tem primeru lahko proces ustavimo, to naredimo tako, da v oknu xload pritisnemo tipko control-z. Proces še vedno obstaja, vendar se ne izvaja (idle). Ustavljen proces zopet poženemo z ukazom bg:

```
[me@linuxbox me]$ xload
[2]+ Stopped xload

[me@linuxbox me]$ bg
[2]+ xload &
```

Izpis uporabnikovih procesov

Xload se sedaj izvaja v ozadju. Večkrat želimo ugotoviti, katere procese smo pognali. To lahko naredimo z uporabo ukazov jobs in ps.

```
[me@linuxbox me]$ jobs
[1]+ Running xload &

[me@linuxbox me]$ ps
PID TTY TIME CMD
1211 pts/4 00:00:00 bash
1246 pts/4 00:00:00 xload
1247 pts/4 00:00:00 ps

[me@linuxbox me]$
```

Ubijanje procesov

Procese pokončamo z ukazom `kill`. Poskusimo pokončati xload, ki teče v ozadju. Če želimo pokončati proces, potrebujemo ali številko opravila ali njegov PID (process id). Uporabimo ukaz jobs (številka pravila) ali ps (PID):

```
[me@linuxbox me]$ xload &
[1] 1292

[me@linuxbox me]$ jobs
[1]+ Running xload &

[me@linuxbox me]$ kill %1

[me@linuxbox me]$ xload &
[2] 1293
[1] Terminated xload

[me@linuxbox me]$ ps
PID TTY TIME CMD
1280 pts/5 00:00:00 bash
1293 pts/5 00:00:00 xload
1294 pts/5 00:00:00 ps

[me@linuxbox me]$ kill 1293
[2]+ Terminated xload

[me@linuxbox me]$
```

Več o ukazu kill

Kil običajno uporabljamo za ubijanje procesov, vendar je njegov namen nekoliko širši. V resnici s tem ukazom procesom pošiljamo *signale* (največkrat za ubijanje procesa). Kako proces reagira na signale, je (za večino signalov) odvisno od procesa samega. Pravilno napisani programi, ko dobijo določen signal, nanj primerno reagirajo. Npr. urejevalnik

besedil lahko ob signalu, ki označuje, da se uporabnik odjavlja iz sistema, shrani besedilo, preden se zaključi. Z ukazom kill lahko procesom torej pošljemo različne signale. Če natipkamo:

```
kill -l
```

dobimo seznam vseh signalov, ki jih kill podpira. Signalov je precej, posebej uporabni so naslednji:

Št. signala	*Ime*	*Opis*
1	*SIGHUP*	Signal 'Hang up'. Programi lahko ta signal upoštevajo ali pa ne.
15	*SIGTERM*	Signal 'Termination signal'. S tem signalo zahtevamo, da se proces zaključi. Program lahko na ta signal reagira, lahko pa tudi ne (npr. če se program zacikla). Ta signal pošljemo procesu, če pritisnemo control-c. To je tudi privzeti signal, ki ga pošlje ukaz kill (če ob uporabi kill ne določimo signala).
9	*SIGKILL*	Signal 'Kill signal'. Ta povzroči, da se proces nemudoma zaključi, proces zaključi jedro Linuxa – torej zaključi proces tudi, če program ne deluje pravilno.

Predpostavimo, da želimo zaključiti program, ki se je zaciklal in ne deluje pravilno:

1. Uporabimo ukaz `ps` , da dobimo PID ustreznega procesa.
2. Izvedemo ukaz `kill`, ki mu podamo dobljen PID.
3. Če se proces ne zaključi (ne reagira na signal), pošiljamo vedno 'strožje' signale, dokler e proces ne odzove.

```
[me@linuxbox me]$ ps x
PID TTY STAT TIME COMMAND
2931 pts/5 SN 0:00 netscape

[me@linuxbox me]$ kill -SIGTERM 2931

[me@linuxbox me]$ kill -SIGKILL 2931
```

Ker je SIGTERM privzeti signal, običajno v praksi ravnamo tako:

```
[me@linuxbox me]$ kill 2931
```

Če se proces ne zaključi, ga v to prisilimo s signalom SIGKILL (podamo lahko njegovo številko):

```
[me@linuxbox me]$ kill -9 2931
```