Poglavje 3

Učbenik Linuxa

3.1 Uvod

Če ste novinec v Unixu in Linuxu, vas morda malce straši obsežnost in očitna kompleksnost sistema pred vami. To poglavje ne gre v podrobnosti niti ne pokriva naprednih tem. Namesto tega želimo, da stojite na trdnih tleh in da vaš sistem teče.

Tukaj ne predpostavljamo posebnega predznanja, razen morda, da ste kolikor toliko domači z osebnimi računalniškimi sistemi in MS-DOS-om. Vendar tudi če niste uporabnik MS-DOS-a, bi morali biti sposobni razumeti vse od tukaj. Na prvi pogled je Linux precej podoben MS-DOS-u – konec koncev, deli MS-DOS-a so se zgledovali po operacijskem sistemu CP/M, ki se je zgledoval po Unixu. Vendar le najbolj površinske lastnosti Linuxa spominjajo na MS-DOS. Tudi če ste popoln novinec v svetu osebnih računalnikov, vam naj bi ta učbenik pomagal.

In, preden začnemo: *Ne bojte se eksperimentiranja*. Sistem vas ne bo ugriznil. Ničesar ne morete uničiti, če delate na sistemu. Linux ima vgrajene varnostne zapore, da prepreči »navadnim« uporabnikom poškodbo datotek, ki so nujno potrebne za sistem. Celo tedaj je najhujše, kar se vam lahko zgodi, da pobrišete nekaj ali vse vaše datoteke in boste morali ponovno namestiti sistem. Torej na tej točki nimate česa izgubiti.

3.2 Osnovni pojmi Linuxa

Linux je večopravilni, večuporabniški operacijski sistem, kar pomeni, da lahko več ljudi hkrati poganja več različnih aplikacij na enem samem računalniku. To se razlikuje od MS-DOS-a, kjer sistem hkrati uporablja le ena oseba. Pod Linuxom se morate, kot dokaz vaše istovetnosti, **prijaviti**, kar vključuje vnos vašega **uporabniškega imena** (imena, pod katerim vas sistem identificira) in vnos **gesla**, ki je vaš osebni ključ za prijavo v vaš račun. Ker le vi poznate vaše geslo, se nihče drug ne more prijaviti v sistem pod vašim uporabniškim imenom.

Na tradicionalnih sistemih Unix vam sistemski upravitelj določi uporabniško ime in začetno geslo, ko vam odpre račun na sistemu. Vendar ker ste na vašem Linuxu *vi* sistemski upravitelj, morate usposobiti vaš lastni račun, preden se lahko prijavite vanj. Za nadaljnje razprave bomo uporabljali izmišljeno uporabniško ime, »`larry`«.

Nadalje, vsakemu sistemu je prirejeno **ime gostitelja**. To ime gostitelja določa ime vašega stroja, mu daje značaj in šarm. Ime gostitelja se uporablja za identifikacijo posa-

meznih strojev na mreži, a tudi če vaš stroj ni omrežen, mora imeti ime gostitelja. Za naše nadaljnje primere naj bo gostiteljsko ime »`mousehouse`«.

3.2.1 Izdelava računa

Preden lahko uporabljate na novo nameščen sistem Linux, morate usposobiti uporabniški račun zase. Navadno ni dobro uporabljati računa `root` za normalno uporabo; račun `root` rezervirajte za poganjanje privilegiranih ukazov in za vzdrževanje sistema, kot je opisano spodaj.

Za izdelavo svojega računa se prijavite kot `root` in uporabite ukaz `useradd` ali `adduser`. Glejte razdelek 4.7 za informacije o tem postopku.

3.2.2 Prijava v sistem

Ob prijavi boste videli pozornik, podoben naslednjemu:

```
mousehouse login:
```

Vnesite vaše uporabniško ime in pritisnite `Enter`. Naš junak `larry` bi napisal:

```
mousehouse login:  larry
Password:
```

Potem vnesite svoje geslo. Znaki, ki jih boste vnašali, se ne bodo videli na zaslonu, zato tipkajte previdno. Če se zatipkate pri vnosu gesla, boste videli sporočilo o napačni prijavi

```
Login incorrect
```

in boste morali ponovno poskusiti postopek prijave.

Ko ste enkrat pravilno vnesli uporabniško ime in geslo, ste uradno prijavljeni na sistem in lahko se pričnete klatiti po njem.

3.2.3 Navidezne konzole

Sistemska **konzola** je sestavljena iz monitorja in tipkovnice, priključene neposredno na sistem. (Ker je Linux večuporabniški operacijski sistem, imate lahko tudi druge terminale, ki so na vaš sistem priključeni prek serijskih vrat, a ti niso konzole.) Linux, kot nekatere druge različice Unixa, ponuja dostop do **navideznih konzol** (angl. virtual consoles, VCs), ki vam omogočajo, da imate hkrati na konzoli več kot eno prijavno sejo.

Za demonstracijo tega se najprej prijavite v vaš sistem. Potem pritisnite `Alt-F2`. Spet bi morali videti prijavni pozornik `login:`. Gledate drugo navidezno konzolo (VC). Za preklop na prvo VC, pritisnite `Alt-F1`. *Voila!* Vrnili ste se v prvo prijavno sejo.

Novo-nameščeni sistem Linux vam verjetno dovoljuje dostop le do prvega pol ducata (ali kaj takega) navideznih konzol, s pritiskom kombinacij od `Alt-F1` do `Alt-F4` oziroma kolikor VC je pač nastavljenih na vašem sistemu. Možno je omogočiti do 12 VC – po eno za vsako funkcijsko tipko na vaši tipkovnici. Kot vidite, je uporaba VC lahko zelo zmogljiva, saj lahko delate v precej različnih sejah hkrati.

Medtem ko je uporaba VC včasih omejujoča (konec koncev, naenkrat lahko gledate le eno VC), vam naj bi vendarle dala občutek večopravilnostnih zmožnosti Linuxa. Medtem ko počnete eno stvar na prvi VC, lahko preklopite na drugo VC in delate na nečem drugem.

3.2.4 Ukazne lupine in ukazi

Pri večini vaših odkrivanj sveta Linuxa se boste pogovarjali s sistemom skozi **ukazno lupino** (angl. shell), programom, ki bere ukaze, ki jih vpišete, in jih prevede v navodila operacijskemu sistemu. To se lahko primerja s programom `COMMAND.COM` pod MS-DOS-om, ki opravlja pravzaprav enako stvar. Ukazna lupina je le en vmesnik do Linuxa. Obstaja več različnih vmesnikov – na primer grafični vmesnik X Window System, ki vam omogoča poganjanje ukazov z uporabo miške in tipkovnice.

Takoj ko se prijavite v sistem, ta zažene ukazno lupino in začnete lahko z vnosom ukazov. Tukaj je kratek primer. Larry se prijavi in čaka v **pozorniku** (angl. prompt) ukazne lupine.

```
mousehouse login:  larry
Password:  larryjevo geslo
Welcome to Mousehouse!


/home/larry$
```

Zadnja vrstica tega teksta je pozornik ukazne lupine, ki nakazuje, da je ukazna lupina pripravljena sprejemati ukaze. (Več o tem, kaj pomeni sam pozornik, kasneje.) Poskusimo naročiti sistemu, da naredi kaj zanimivega:

```
/home/larry$ make love
make:  *** No rule to make target 'love'.  Stop.
/home/larry$
```

No, kot se izkaže, je `make` ime pravega programa na sistemu, in ukazna lupina je izvedla ta program, ko smo ji tako ukazali. (Žal je bil sistem neprijazen.)

To nas privede do žgočega vprašanja: Kaj je ukaz? Kaj se zgodi, ko vpišete »`make love`«? Prva beseda v ukazni vrstici, »`make`«, je ime ukaza, ki naj se izvede. Vse drugo v ukazni vrstici šteje za argument temu ukazu. Primer:

```
/home/larry$ cp foo bar
```

Ime tega ukaza je »`cp`«, argumenta sta »`foo`« in »`bar`«.

Ko vnesete ukaz, naredi ukazna lupina precej stvari. Najprej preveri ukaz ter pogleda ali je vgrajen v ukazno lupino. (Se pravi, ali je to ukaz, ki ga zna izvesti sama lupina. Obstaja vrsta takih ukazov in vanje se bomo poglobili pozneje.) Lupina tudi preveri, ali je ukaz vzdevek (angl. alias) ali nadomestno ime za drug ukaz. Če noben od teh pogojev ni izpolnjen, ukazna lupina na disku poišče program z danim imenom. Če je pri iskanju uspešna, lupina požene program in mu pošlje argumente, podane v ukazni vrstici.

V našem primeru ukazna lupina išče program, imenovan `make`, in ga požene z argumentom `love`. `Make` je program, ki se pogosto uporablja pri prevajanju velikih programov in vzame za argumente imena »cilja« (angl. target) za prevod. V primeru ukaza »`make love`« smo naročili programu `make`, naj prevede cilj `love`. Ker `make` ne more najti cilja s tem imenom, se pritoži s šegavim sporočilom o napaki in nas postavi v ukazno vrstico.

Kaj se zgodi, če vpišemo ukaz v ukazno lupino in lupina ne more najti programa z določenim imenom? No, poskusimo lahko naslednje:

```
/home/larry$ eat dirt
eat:  command not found
/home/larry$
```

Precej preprosto, če ukazna lupina ne more najti programa z imenom, danim v ukazni vrstici (tukaj »`eat`«), izpiše sporočilo o napaki. Pogosto boste videli to sporočilo o napaki, če boste zatipkali ukaz (na primer, če ste napisali »`mkae love`« namesto »`make love`«).

3.2.5 Odjava iz sistema

Preden se še bolj zatopimo, vam moramo povedati, kako se lahko odjavite iz sistema. V ukazni vrstici uporabite ukaz

```
/home/larry$ exit
```

za odjavo. Obstajajo tudi drugi načini odjave iz sistema, a ta je najbolj otročje lahek.

3.2.6 Sprememba vašega gesla

Vedeti morate tudi, kako spremeniti svoje geslo. Ukaz `passwd` vas vpraša za vaše staro geslo in za novo geslo (angl. old & new password). Prosi vas tudi, da ponovno vnesete novo geslo za preverbo. Pazite, da ne boste pozabili svojega gesla – če ga, boste morali prositi sistemskega upravitelja, da ga spet nastavi. (Če ste vi ta sistemski upravitelj, glejte stran 158.)

3.2.7 Datoteke in imeniki

Pod večino operacijskih sistemov (vključno z Linuxom) obstaja pojem **datoteke**, ki je le skupek informacij pod nekim imenom (imenovanim **ime datoteke**). Primeri datotek so lahko vaš seminar iz zgodovine, e-poštno sporočilo ali pravi program, ki se lahko izvede. V bistvu se vse, kar se posname na disk, posname kot posamična datoteka.

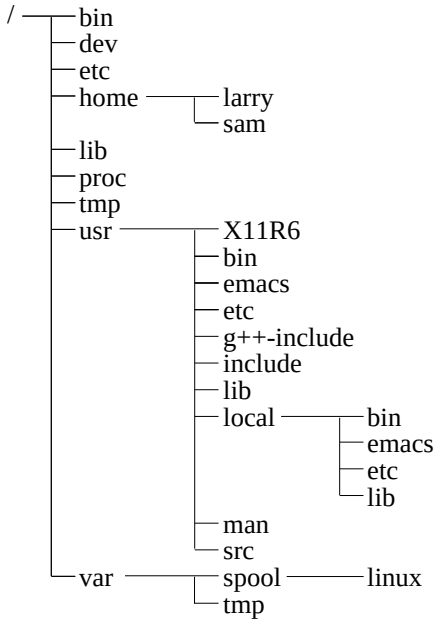
Datoteke (angl. files) se prepoznavajo po njihovih imenih datotek (angl. file names). Na primer, datoteka z vašim seminarjem iz zgodovine se lahko posname z imenom datoteke `zgodovinski-seminar`. Ta imena navadno določajo datoteko in njeno vsebino v neki obliki, ki ima za vas pomen. Za imena datotek ni standardnega formata, kot je pod MS-DOS-om in nekaterimi drugimi operacijskimi sistemi; v splošnem lahko ime datoteke vsebuje katerikoli znak (razen znaka »`/`« – glejte razpravo o imenih poti spodaj) in je po dolžini omejeno na 256 znakov.

S pojmom datotek pride tudi pojem imenikov. **Imenik** je zbirka datotek. Predstavljate si ga lahko kot »mapo«, ki vsebuje mnoge različne datoteke. Imenikom so prirejena imena, pod katerimi jih lahko istovetite. Nadalje, imeniki so vzdrževani v drevesu podobni strukturi; se pravi, imeniki lahko vsebujejo druge imenike.

Na datoteko se lahko torej sklicujete z njenim **imenom poti** (angl. path name), ki jo sestavlja ime datoteke, pred katerim je ime imenika, ki vsebuje datoteko. Na primer, denimo, da ima Larry imenik z imenom `papers`, ki vsebuje tri datoteke: `history-final`, `english-lit` in `masters-thesis`. Vsaka od teh treh datotek vsebuje informacije o treh Larryjevih projektih. Za sklicevanje na datoteko `english-lit` lahko Larry določi ime datoteke kot v:

```
papers/english-lit
```

Kot vidite, imenik in ime datoteke ločuje ena sama poševnica (`/`). To je razlog, da sama imena datotek ne morejo vsebovati znaka `/`. Uporabnikom MS-DOS-a se bo zdelo to pravilo domače, čeprav se v svetu MS-DOS-a namesto tega uporablja poševnica vznak (\).



Slika 3.1: Tipično (okrnjeno) drevo imenikov Linuxa

Kot smo že omenili, se imeniki tudi lahko gnezdijo drug v drugega. Na primer, denimo, da obstaja še en imenik znotraj `papers`, imenovan `notes`. Imenik `notes` vsebuje datoteke `math-notes` in `cheat-sheet`. Ime poti do datoteke `cheat-sheet` bi bilo

```
papers/notes/cheat-sheet
```

Torej je ime poti res kot pot do datoteke. Imenik, ki vsebuje dani podimenik, je znan kot **starševski imenik**. Tukaj je imenik `papers` starševski imenik imenika `notes`.

3.2.8 Drevo imenikov

Mnogi sistemi Linux uporabljajo standardni načrt za datoteke, tako da se sistemski viri in programi lahko zlahka poiščejo. Ta načrt tvori drevo imenikov, ki se začenja z imenikom »`/`«, znanim tudi kot »korenski imenik«. Neposredno pod `/` so pomembni podimeniki: `/bin`, `/etc`, `/dev`, `/usr` in drugi. Ti imeniki spet vsebujejo druge imenike, ki vsebujejo nastavitvene datoteke sistema, programe in tako naprej.

Vsak uporabnik ima svoj **domači imenik**, ki je imenik, namenjen uporabnikovem shranjevanju svojih datotek. V zgornjih primerih so vse Larryjeve datoteke (kot `cheat-sheet` in `history-final`) shranjene v njegovem domačem imeniku. Navadno so uporabniški domači imeniki vsebovani pod imenikom `/home` in so poimenovani po uporabniku, ki je lastnik tega imenika. Larryjev domači imenik je `/home/larry`.

Diagram na strani 107 kaže primer drevesa imenikov, ki naj bi vam dal predstavo, kako je organizirano drevo imenikov na vašem sistemu.

3.2.9 Trenutni delovni imenik

Za ukaze, ki jih vnašate, se vedno predpostavlja, da so relativni glede na vaš **trenutni delovni imenik**. Delovni imenik si lahko zamišljate kot imenik, v katerem se trenutno »nahajate«. Ko se prvič prijavite, je vaš delovni imenik nastavljen na domači imenik – v našem primeru `/home/larry`. Kadar se sklicujete na datoteko, se lahko sklicujete nanjo v odvisnosti od vašega trenutnega delovnega imenika, namesto da določate polno ime datoteke.

Tukaj je primer. Larry ima imenik `papers`, in imenik `papers` vsebuje datoteko `history-final`. Če želi Larry pogledati vsebino te datoteke, lahko uporabi ukaz

```
/home/larry$ more /home/larry/papers/history-final
```

Ukaz `more` preprosto izpiše datoteko, po en zaslon naenkrat. A ker je Larryjev trenutni imenik `/home/larry`, se lahko namesto tega sklicuje na datoteko *relativno* glede na svojo trenutno lokacijo, z uporabo ukaza

```
/home/larry$ more papers/history-final
```

Če imena datoteke (kot `papers/final`) ne začnete z znakom `/`, se sklicujete na datoteko v izrazih, relativnih glede na vaš trenutni delovni imenik. To je znano kot **relativno ime poti**.

Po drugi strani, če začnete ime datotek z znakom `/`, ga sistem tolmači kot polno ime poti – se pravi, ime poti, ki vključuje celotno pot do datoteke, začenši od korenskega imenika, `/`. To je znano kot **absolutno** ali **polno ime poti**.

3.2.10 Sklicevanje na domače imenike

Pod ukaznima lupinama `tcsh` in `bash`[1] lahko določite vaš domači imenik z znakom za tildo (`~`). Na primer, ukaz

```
/home/larry$ more ~/papers/history-final
```

je ekvivalenten ukazu

```
/home/larry$ more /home/larry/papers/history-final
```

Ukazna lupina nadomesti znak `~` z imenom vašega domačega imenika.

Z znakom za tildo lahko določate tudi domače imenike drugih uporabnikov. Ime poti `~karl/letters` ukazna lupina tolmači kot `/home/karl/letters` (če je `/home/karl` domači imenik uporabnika `karl`). Uporaba tilde je preprosto bližnjica; ni imenika, imenovanega `~` – to je le skladenjski posladek, ki ga priskrbi ukazna lupina.

3.3 Prvi koraki v Linuxu

Preden začnemo, je pomembno vedeti, da Linux razlikuje med velikimi in malimi črkami pri vseh imenih datotek in ukazov (za razliko od operacijskih sistemov, kot je MS-DOS). Na primer, ukaz `make` je zelo različen od `Make` ali `MAKE`. Isto velja za imena datotek in imenikov.

[1] `tcsh` in `bash` sta dve *ukazni lupini*, ki tečeta na Linuxu. Ukazna lupina je program, ki bere uporabniške ukaze in jih izvaja; večina sistemov Linux omogoča `tcsh` ali `bash` za račune novih uporabnikov.

3.3.1 Premikanje naokrog

Zdaj ko se znate prijaviti in veste, kako se sklicujete na datoteke z uporabo imen poti, si oglejmo še, kako lahko spremenite svoj trenutni delovni imenik.

Ukaz za premikanje po strukturi imenikov se imenuje `cd` (iz angleškega izraza »change directory«, sprememba imenika). Mnogo pogosto uporabljanih ukazov Linuxa je dolgih dve ali tri črke. Uporaba ukaza `cd` je

```
cd imenik
```

kjer je *imenik* ime imenika, za katerega želite, da postane vaš trenutni delovni imenik.

Kot je bilo že omenjeno, ob prijavi začnete v svojem domačem imeniku. Če bi Larry želel preklopiti v podimenik `papers`, bi uporabil ukaz

```
/home/larry$ cd papers
/home/larry/papers$
```

Kot lahko vidite, se Larryjev pozornik spreminja, da odraža njegov trenutni delovni imenik (tako, da ve, kje je). Zdaj, ko je v imeniku `papers`, lahko pogleda svoj zaključni spis iz zgodovine z ukazom

```
/home/larry/papers$ more history-final
```

Zdaj je Larry obtičal v podimeniku `papers`. Za premik nazaj v naslednji višji (ali starševski) imenik, uporabite ukaz

```
/home/larry/papers$ cd  ..
/home/larry$
```

(Pazite na presledek med »`cd`« in »`..`«.) Vsak imenik ima vnos, imenovan »`..`«, ki se nanaša na njegov starševski imenik. Podobno, vsak imenik ima vnos, imenovan »`.`«, ki se nanaša na njega. Torej nas ukaz

```
/home/larry/papers$ cd  .
```

pusti v istem imeniku.

Z ukazom `cd` lahko uporabljate tudi absolutna imena poti. Za `cd` v Karlov domači imenik lahko uporabimo ukaz

```
/home/larry/papers$ cd /home/karl
/home/karl$
```

Tudi uporaba `cd` brez argumentov nas bo vrnila v naš lastni domači imenik.

```
/home/karl$ cd
/home/larry$
```

3.3.2 Ogled vsebine imenikov

Zdaj ko veste, kako se premikati po imenikih, si morda mislite, »Pa kaj?«. Premikanje po imenikih je samo po sebi precej neuporabno, zato uvedimo nov ukaz, `ls`. Ukaz `ls` izpiše seznam datotek in imenikov, privzeto iz vašega trenutnega imenika. Na primer:

```
/home/larry$ ls
Mail
letters
papers
/home/larry$
```

Tukaj vidimo, da ima Larry tri vnose v svojem trenutnem imeniku: `Mail`, `letters` in `papers`. To nam ne pove veliko – so to imeniki ali datoteke? Lahko uporabimo izbiro `-F` ukaza `ls` za podrobnejše informacije.

```
/home/larry$ ls -F
Mail/
letters/
papers/
/home/larry$
```

Iz znaka `/`, ki je pripet vsakemu imenu datoteke, vemo, da so ti trije vnosi pravzaprav podimeniki.

Uporaba `ls -F` lahko v izpisu tudi pripne »`*`« na konec imena datoteke, kar pomeni, da je datoteka **izvedljiva** oziroma program, ki se lahko požene. Če ni ničesar pripetega imenu datoteke ob uporabi `ls -F`, je datoteka »navadna stara datoteka«, se pravi, ni niti imenik niti izvedljiva datoteka.

V splošnem vsak ukaz Unixa lahko vzame številne izbire poleg drugih argumentov. Te izbire se navadno začenjajo z znakom »`-`«, kot smo videli zgoraj pri izbiri `-F`. Izbira `-F` pove ukazu `ls`, naj izda več informacij o tipu datotek – v tem primeru izpiše `/` za vsakim imenom imenika.

Če podate ukazu `ls` ime imenika, bo sistem izpisal vsebino tega imenika.

```
/home/larry$ ls -F papers
english-lit
history-final
masters-thesis
notes/
/home/larry$
```

Za bolj zanimiv izpis poglejmo, kaj je v sistemskem imeniku `/etc`.

```
/home/larry$ ls /etc
Images          group           pac             rpcinfo
adm             inet            passwd          securetty
bcheckrc        init            printcap        services
brc             init.d          profile         shells
brc             initrunlvl      psdatabase      startcons
csh.cshrc       inittab         rc              swapoff
csh.login       inittab.old     rc.new          swapon
default         issue           rc0.d           syslog.conf
disktab         lilo            rc1.d           syslog.pid
fdprm           lpc             rc2.d           syslogd.reload
fstab           magic           rc3.d           termcap
ftpaccess       motd            rc4.d           umount
```

```
ftpusers        mount           rc5.d           update
getty           mtab            rmt             utmp
gettydefs       mtools          rpc             wtmp
/home/larry$
```

Če ste uporabnik MS-DOS-a, boste opazili, da so imena datotek lahko daljša od 8 znakov in lahko vsebujejo pike na kateremkoli položaju. V imenu datoteke lahko uporabljate celo več kot eno piko.

Premaknimo se na vrh drevesa imenikov in potem navzdol do drugega imenika, z ukazi

```
/home/larry$ cd ..
/home$ cd ..
/$ cd usr
/usr$ cd bin
/usr/bin$
```

V imenike se lahko premikate tudi v enem koraku kot v ukazu `cd /usr/bin`.

Poskusite se premikati po različnih imenikih in uporabljajte `ls` in `cd`. V nekaterih primerih trčite ob slabo znamenje v obliki sporočila o napaki zavrnitve dostopa (»`Permission denied`«). To je le varnost Unixa, ki se oglaša: če želite uporabljati ukaza `ls` ali `cd`, morate imeti za to potrebna dovoljenja. O tem bomo spregovorili več od strani 126 naprej.

3.3.3 Ustvarjanje novih imenikov

Čas je, da se naučimo, kako ustvariti imenike. To vključuje uporabo ukaza `mkdir`. Poskusite naslednje:

```
/home/larry$ mkdir foo
/home/larry$ ls -F
Mail/
foo/
letters/
papers/
/home/larry$ cd foo
/home/larry/foo$ ls
/home/larry/foo$
```

Čestitamo! Naredili ste nov imenik in se premaknili vanj. Ker v tem novem imeniku še ni datotek, se naučimo, kako prepišete datoteke z enega mesta na drugo.

3.3.4 Prepisovanje datotek

Za prepisovanje datotek uporabljajte ukaz `cp`, kot je prikazano tukaj:

```
/home/larry/foo$ cp /etc/termcap  .
/home/larry/foo$ cp /etc/shells  .
/home/larry/foo$ ls -F
shells     termcap
/home/larry/foo$ cp shells bells
/home/larry/foo$ ls -F
bells     shells     termcap
/home/larry/foo$
```

Ukaz cp prepiše datoteke, navedene v ukazni vrstici, v datoteko ali imenik, določen z zadnjim argumentom. Tu uporabljamo ».« za sklicevanje na trenutni imenik.

3.3.5 Premikanje datotek

Ukaz mv premika datoteke, namesto da bi jih prepisal. Skladnja je zelo premočrtna:

```
/home/larry/foo$ mv termcap sells
/home/larry/foo$ ls -F
bells     sells     shells
/home/larry/foo$
```

S tem smo datoteko termcap preimenovali v sells. Ukaz mv lahko uporabljate tudi za premikanje datoteke v popolnoma nov imenik.

◇ **Pozor!** Ukaza mv in cp bosta prepisala ciljno datoteko z istim imenom, ne da bi vas vprašala. Bodite previdni pri premikanju datoteke v drugi imenik. Morda je tam že datoteka z istim imenom, ki jo boste prepisali!

3.3.6 Brisanje datotek in imenikov

Zdaj se vam pri uporabi ukaza ls razvija grda rima. Za brisanje datoteke uporabite ukaz rm (iz angleškega izraza »remove«, odstrani) kot je prikazano tukaj:

```
/home/larry/foo$ rm bells sells
/home/larry/foo$ ls -F
shells
/home/larry/foo$
```

Ostalo nam ni nič drugega kot shells, a ne bomo se pritoževali. Vedite, da vas ukaz rm, privzeto, ne bo vprašal, ali naj izbriše datoteko – zato bodite previdni.

Ukazu rm je soroden ukaz rmdir. Ta ukaz izbriše imenik, a le, če je ta prazen. Če imenik vsebuje kakšno datoteko ali podimenik, ga rmdir ne bo pobrisal, ampak se bo pritožil.

3.3.7 Ogled datotek

Ukaza more in cat se uporabljata za ogled vsebine datotek. Ukaz more izpiše datoteko po en zaslon naenkrat, medtem ko cat izpiše vso datoteko naenkrat.

Za ogled datoteke shells uporabite ukaz

```
/home/larry/foo$ more shells
```

V primeru, da vas zanima, kaj vsebuje datoteka shells: to je seznam veljavnih programov za ukazno lupino na vašem sistemu. Na večini sistemov to vključuje /bin/sh, /bin/bash in /bin/csh. O teh različnih vrstah ukaznih lupin bomo spregovorili kasneje.

Med uporabo more pritisnite Space (tipko presledek) za izpis naslednje strani s tekstom in b za izpis prejšnje strani. V pripomočku more so dostopni tudi drugi ukazi, ta dva sta le osnova. Pritisk na q bo zaključil more.

Zapustite more in poskusite cat /etc/termcap. Besedilo bo verjetno letelo prehitro, da bi ga sploh lahko brali. Ime »cat« pravzaprav izhaja iz angleške besede »concatenate«, ki pomeni »povezati«, »spenjati« in res opisuje uporabo tega programa. Ukaz cat se lahko

uporablja za spenjanje vsebine različnih datotek in shranjevanje rezultata v drugo datoteko. To bo na vrsti ponovno v razdelku 3.14.1.

3.3.8 Dobivanje pomoči na zvezi

Skoraj vsak sistem Unix, vključno z Linuxom, priskrbi pripomoček, znan kot **strani priročnika** (angl. manual pages). Te strani priročnika vsebujejo dokumentacijo, dostopno na zvezi, za sistemske ukaze, vire, nastavitvene datoteke in tako naprej.

Ukaz, ki se uporablja za dostop do strani priročnika, se imenuje `man`. Če bi radi izvedeli več o drugih izbirah ukaza `ls`, lahko napišete

```
/home/larry$ man ls
```

in prikazana bo stran priročnika o ukazu `ls`.

Žal je večina strani priročnika napisana za tiste, ki vsaj približno že vedo kaj počne ukaz ali vir. Zato strani priročnika navadno vsebujejo le tehnične podrobnosti o ukazu, brez veliko razlage. Vendar so strani priročnika neprecenljiv vir za osvežitev vašega spomina, če pozabite skladnjo nekega ukaza. Strani priročnika vam bodo povedale tudi o ukazih, ki jih ne pokrivamo v tej knjigi.

Priporočam, da uporabite `man` za ukaze, ki smo jih že spoznali, in tako vsakič, ko uvedemo nov ukaz. Nekateri od teh ukazov ne bodo imeli strani v priročniku, iz različnih vzrokov. Najprej, strani priročnika morda še niso bile napisane. (Dokumentacijski projekt za Linux je odgovoren tudi za strani priročnika v Linuxu. Sčasoma pridobivamo večino strani priročnika, dostopnih za sistem.) Drugič, ukaz je lahko notranji ukaz ukazne lupine (razložen na strani 105) ali vzdevek, ki ne bo imel svoje strani v priročniku. En primer je že `cd`, ki je notranji ukaz lupine. Ukazna lupina pravzaprav sama izvede `cd` – ni posebnega programa, ki bi izvedel ta ukaz.

3.4 Dostop do datotek MS-DOS-a

Če zaradi kakršnegakoli sprevrženega in bizarnega razloga želite dostopati do datotek MS-DOS-a, se to v Linuxu naredi zlahka.

Običajen način za dostop do datotek za MS-DOS je priklop dosovske particije ali diskete pod Linuxom, kar vam dovoljuje dostop do datotek neposredno skozi datotečni sistem. Na primer, če imate disketo za MS-DOS v `/dev/fd0`, jo bo ukaz

```
# mount -t msdos /dev/fd0 /mnt
```

priklopil pod `/mnt`. Glejte razdelek 4.9.4 za več informacij o priklopu disket.

Priklopite lahko tudi particije za MS-DOS na vašem trdem disku ter potem dostopate do njih. Če je particija z MS-DOS-om `/dev/hda1`, jo priklopi ukaz

```
# mount -t msdos /dev/hda1 /mnt
```

Poskrbite, da boste particijo odklopili z `umount`, ko jo nehate uporabljati. Particija za MS-DOS se lahko samodejno priklopi, če vključite ustrezen vnos v datoteko `/etc/fstab`. Glejte razdelek 4.5 za podrobnosti. Naslednja vrstica v `/etc/fstab` bo priklopila particijo za MS-DOS `/dev/hda1` v imenik `/dos`.

```
/dev/hda1     /dos     msdos      defaults
```

Priklopite lahko tudi datotečne sisteme VFAT, ki jih uporablja Windows 95:

```
# mount -t vfat /dev/hda1 /mnt
```

To dovoljuje dostop do dolgih imen datotek sistema Windows 95. To se nanaša le na particije, ki imajo zares shranjena dolga imena datotek. Tega ne morete uporabiti, da bi priklopili navadni datotečni sistem FAT16 in uporabljali dolga imena datotek.

Za dostop do datotek za MS-DOS se lahko uporablja tudi programje Mtools. Ukazi `mcd`, `mdir` in `mcopy` se vsi obnašajo kot njihovi ekvivalenti v MS-DOS-u. Če namestite Mtools, bi morale biti dostopne tudi strani priročnika za te ukaze.

Dostopanje do datotek za MS-DOS je ena stvar; poganjanje programov za MS-DOS je druga. Za Linux se razvija program MS-DOS Emulator, emulacija MS-DOS-a; dostopen je na široko in vključen v večino distribucij. Vzamete ga lahko s številnih lokacij, vključno z različnimi mesti za prenos datotek Linuxa po FTP, naštetimi v dodatku B. Za MS-DOS Emulator poročajo, da je dovolj zmogljiv, da z njim tečejo v Linuxu številne aplikacije, vključno z WordPerfectom. Vendar sta Linux in MS-DOS nadvse različna operacijska sistema. Zmogljivost vsakega emulatorja za MS-DOS pod Unixom je omejena. Poleg tega se razvija tudi emulacija Microsoft Windows, ki teče pod grafičnim sistemom X Window.

3.5 Povzetek osnovnih ukazov Unixa

Ta razdelek predstavlja nekatere od najbolj uporabnih osnovnih ukazov sistema Unix, vključno s tistimi, ki so pokriti v prejšnjih razdelkih.

Upoštevajte, da se izbire navadno začnejo z znakom »`-`«, in v večini primerov lahko določite več kot eno izbiro z enim samim znakom »`-`«. Na primer, namesto da bi uporabljali ukaz `ls -l -F`, lahko uporabljate `ls -lF`.

Namesto da bi vam izpisali vsako izbiro vsakega ukaza, vam predstavljamo le v tem času uporabne ali pomembne ukaze. Pravzaprav ima večina teh ukazov izbire, ki jih ne boste nikoli uporabljali. Strani priročnika o vsakem posameznem ukazu si lahko ogledate z uporabo `man`, ki izpiše seznam vseh dostopnih izbir.

Upoštevajte tudi, da veliko ukazov vzame kot argument seznam datotek ali imenikov, označenih v tej tabeli kot »*datoteka*$_1$ ... *datoteka*$_N$«. Na primer, ukaz `cp` vzame za argument seznam datotek za prepis, ki mu sledi ciljna datoteka ali imenik. Ko prepisujete več kot eno datoteko, mora biti cilj imenik.

`cd` — Spremeni trenutni delovni imenik.
Skladnja: `cd` *imenik*
Kjer je *imenik* ime imenika, v katerega želite vstopiti. (»`.`« se nanaša na trenutni imenik, »`..`« na starševski imenik. Če ni določen noben imenik, se privzame vaš domači imenik.)
Primer: `cd ../foo` nastavi vaš trenutni imenik za en nivo navzgor in spet za en nivo navzdol v `foo`.

`ls` — Prikaže seznam datotek in imenikov.
Skladnja: `ls` *datoteke*
Kjer so *datoteke* imena datotek ali imenikov za izpis. Najpogosteje uporabljani izbiri sta `-F` (za prikaz tipa datoteke) in `-l` (za »dolgi« izpis, ki vključuje velikost datoteke, lastnika, dovoljenja za dostop do datoteke in

tako naprej).
Primer: `ls -lF /home/larry` izpiše vsebino imenika `/home/larry`.

`cp` Prepiše eno ali več datotek v drugo datoteko ali v imenik.
Skladnja: `cp` *datoteke cilj*
Kjer *datoteke* našteva datoteke za prepis in je *cilj* ciljna datoteka ali imenik.
Primer: `cp ../frog joe` prepiše datoteko `../frog` v datoteko ali imenik `joe`.

`mv` Premakne eno ali več datotek v drugo datoteko ali imenik. Ta ukaz je ekvivalenten prepisu, ki mu sledi izbris izvorne datoteke. To lahko uporabljate za preimenovanje datotek kot pri ukazu `RENAME` v MS-DOS-u.
Skladnja: `mv` *datoteke cilj*
Kjer *datoteke* našteva datoteke za premik in je *cilj* ciljna datoteka ali imenik.
Primer: `mv ../frog joe` premakne datoteko `../frog` v datoteko ali imenik `joe`.

`rm` Izbriše datoteke. Upoštevajte, da datotek ne morete več dobiti nazaj, ko jih enkrat v Unixu izbrišete (za razliko od MS-DOS-a, kjer lahko navadno »odbrišete« datoteko).
Skladnja: `rm` *datoteke*
Kjer *datoteke* opisujejo imena datotek za brisanje.
Izbira `-i` vas vpraša za potrditev pred izbrisom datoteke.
Primer: `rm -i /home/larry/joe /home/larry/frog` pobriše datoteki `joe` in `frog` v imeniku `/home/larry`.

`mkdir` Ustvari nove imenike.
Skladnja: `mkdir` *imeniki*
Kjer so *imeniki* imeniki, ki naj se ustvarijo.
Primer: `mkdir /home/larry/test` ustvari imenik `test` v imeniku `/home/larry`.

`rmdir` Izbriše prazne imenike. Ko uporabljate `rmdir`, trenutni delovni imenik ne sme biti znotraj imenika, ki ga izbrisujete.
Skladnja: `rmdir` *imeniki*
Kjer *imeniki* definira imenike za izbris.
Primer: `rmdir /home/larry/papers` izbriše imenik `/home/larry/papers`, če je prazen.

`man` Prikaže stran priročnika za dani ukaz ali vir (se pravi, vsak sistemski pripomoček, ki ni ukaz, kot je knjižnična funkcija).
Skladnja: `man` *ukaz*
Kjer je *ukaz* ime ukaza ali vira, za katerega bi radi dobili pomoč.
Primer: `man ls` izpiše pomoč o ukazu `ls`.

`more` Prikaže vsebino poimenovanih datotek, po en zaslon naenkrat.
Skladnja: `more` *datoteke*

Kjer *datoteke* naštevajo seznam datotek za prikaz.
Primer: `more papers/history-final` prikaže datoteko `papers/history-final`.

`cat` Uradno se uporablja za združevanje datotek, a `cat` se uporablja tudi za prikaz vsebine datoteke na zaslonu.
Skladnja: `cat` *datoteke*
Kjer so v seznamu *datoteke* naštete datoteke, katerih vsebina naj se združi in prikaže.
Primer: `cat letters/from-mdw` prikaže vsebino datoteke `letters/from-mdw`.

`echo` Izpiše dane argumente na zaslon.
Skladnja: `echo` *argumenti*
Kjer so *argumenti* argumenti, ki naj se izpišejo.
Primer: `echo "Zdravo, svet"` izpiše niz »`Zdravo, svet`«.

`grep` Izpiše vsako vrstico v eni ali več datotekah, ki ustreza danemu vzorcu.
Skladnja: `grep` *vzorec datoteke*
Kjer je *vzorec* vzorec regularnih izrazov in *datoteke* našteva seznam datotek za iskanje.
Primer: `grep loomer /etc/hosts` izpiše vse vrstice v datoteki `/etc/hosts`, ki vsebujejo vzorec »`loomer`«.

3.6 Raziskovanje datotečnega sistema

Datotečni sistem je zbirka datotek in hierarhije imenikov na sistemu. Prišel je čas, da vas pospremimo skozi datotečni sistem.

Zdaj imate veščine in znanje, da razumete datotečni sistem Linuxa, in imate tudi zemljevid (poglejte na diagram na strani 107).

Najprej pojdite v korenski imenik (`cd /`) in vnesite `ls -F` za izpis njegove vsebine. Verjetno boste videli naslednje imenike[2]: `bin`, `dev`, `etc`, `home`, `install`, `lib`, `mnt`, `proc`, `root`, `tmp`, `user`, `usr` in `var`.

Zdaj si oglejmo vsakega od teh imenikov.

`/bin` `/bin` je kratko za »binarne« ali izvedljive datoteke, kjer počiva veliko programov, nujnih za sistem. Uporabite `ls -F /bin` za izpis tamkajšnjih datotek. Če pogledate na dobljeni spisek, boste lahko videli nekatere ukaze, ki jih prepoznate, kot so `cp`, `ls` in `mv`. Ti so pravi programi za te ukaze. Ko uporabite na primer ukaz `cp`, s tem poženete program `/bin/cp`.

Z uporabo `ls -F` boste videli, da ima večina (če ne vse) datotek v imeniku `/bin` za imenom zvezdico (»`*`«). To pomeni, da so te datoteke izvedljive, kot je bilo opisano na strani 109.

`/dev` »Datoteke« v imeniku `/dev` so **datoteke naprav** (angl. device files) –

[2] Lahko vidite tudi druge in morda ne boste videli vseh. Vsaka izdaja Linuxa se razlikuje v nekaterih pogledih.

prek njih se dostopa do sistemskih naprav in virov, kot so diskovni pogoni, modemi in pomnilnik. Kot lahko vaš sistem bere podatke iz datoteke, lahko bere tudi vhod iz miške z dostopom do datoteke naprave `/dev/mouse`.

Datoteke, katerih imena se začenjajo s `fd`, so disketniške naprave. `fd0` je prvi disketnik in `fd1` je drugi. Morda ste opazili, da je naštetih več disketnih naprav kot le dve zgoraj omenjeni: te predstavljajo različne tipe disket. Na primer, `fd1H1440` dostopa do 3,5-disket visoke gostote v prvem disketniku.

Sledi seznam nekaterih najpogosteje uporabljanih datotek naprav. Četudi morda nimate vseh fizičnih naprav, naštetih spodaj, imate verjetno v imeniku `/dev` vseeno gonilnik zanjo.

- `/dev/console` se nanaša na sistemsko konzolo – se pravi na monitor, priključen neposredno na vaš sistem.
- Različne naprave `/dev/ttyS` in `/dev/cua` se uporabljajo za dostop do zaporednih vrat. `/dev/ttyS0` se nanaša na »`COM1`« v MS-DOS-u. Naprave `/dev/cua` so naprave za »klic ven« (angl. callout) in se uporabljajo z modemom.
- Prek naprav z imeni, ki se začenjajo na `hd` (iz angl. hard drives), dostopamo do trdih diskov. `/dev/hda` se nanaša na *celoten* prvi trdi disk, medtem ko se `/dev/hda1` nanaša na prvo *particijo* na `/dev/hda`.
- Naprave, katerih imena se začenjajo s `sd`, so pogoni SCSI (angl. SCSI drive). Če imate trdi disk SCSI, boste do njega dostopali kot `/dev/sda` namesto `/dev/hda`. Do tračnih enot SCSI dostopamo prek naprav `st` in do CD-ROM-ov SCSI prek naprav `sr`.
- Prek naprav z imeni, ki se začenjajo na `lp`, dostopamo do vzporednih vrat. `/dev/lp0` je isto kot »`LPT1`« v svetu MS-DOS-a.
- `/dev/null` se uporablja kot »črna luknja« – podatki, poslani na to napravo, so za vedno izgubljeni. Zakaj je to uporabno? No, če želite ukiniti izhod ukaza, ki se prikazuje na vašem zaslonu, lahko pošljete ta izhod na ničelno napravo `/dev/null`. Več o tem bomo govorili pozneje.
- Naprave, katerih imena so `/dev/tty`, ki mu sledi številka, se nanašajo na »navidezne konzole« (angl. virtual console, VC) vašega sistema (do njih dostopate s pritiskom `Alt-F1`, `Alt-F2` in tako naprej), pa tudi na čisto prave terminale, priključene na zaporedna vrata. `/dev/tty1` se nanaša na prvo navidezno konzolo, `/dev/tty2` na drugo in tako naprej.
- Naprave, katerih imena se začenjajo z `/dev/pty`, so **psevdoterminali**, ki se uporabljajo, da ponudijo »terminal« oddaljenim prijavnim sejam. Na primer, če je vaš stroj na omrežju, bodo prihajajoče prijave z uporabo `telnet` uporabljale eno od naprav `/dev/pty`.

`/etc` `/etc` vsebuje številne različne nastavitvene datoteke sistema. Te vključujejo `/etc/passwd` (bazo podatkov o uporabnikih), `/etc/rc` (sistemski inicializacijski skript) in tako naprej.

`/sbin` `/sbin` vsebuje nujne sistemske binarne datoteke, ki se uporabljajo pri upravljanju sistema.

`/home` `/home` vsebuje domače imenike uporabnikov. Na primer, `/home/larry` je domači imenik uporabnika »`larry`«. Na sveže nameščenem sistemu morda ne bo nobenih uporabnikov v tem imeniku.

`/lib` `/lib` vsebuje **slike deljenih knjižnic**. To so datoteke, ki vsebujejo kodo, skupno več programom. Namesto da vsak program uporablja svoj izvod teh podprogramov, so vsi shranjeni na enem samem skupnem mestu, v imeniku `/lib`. To naredi izvedljive datoteke manjše in varčuje s prostorom na vašem sistemu.

`/proc` `/proc` podpira »navidezni datotečni sistem« (angl. virtual file system, VFS), kjer so datoteke shranjene v pomnilniku, ne na disku. Te »datoteke« se nanašajo na različne **procese**, ki tečejo na sistemu, in vam omogočajo pridobitev informacij o programih in procesih, ki se v vsakem času izvajajo na sistemu. Več o tem bomo razložili na strani 131 in naslednjih straneh.

`/tmp` Mnogi programi shranjujejo začasne informacije v kakšno datoteko, ki se pobriše, ko program preneha z izvajanjem. Standardno mesto za te datoteke je v imeniku `/tmp`.

`/usr` `/usr` je zelo pomemben imenik, ki vsebuje podimenike z nekaterimi najpomembnejšimi in najuporabnejšimi programi in nastavitvenimi datotekami, uporabljanimi na sistemu.

Različni imeniki, opisani zgoraj, so nujni za delovanje sistema, a večina postavk v imeniku `/usr` je izbirnih. Vendar je sistem uporaben in zanimiv prav zaradi teh izbirnih delov. Brez imenika `/usr` bi imeli dolgočasen sistem, ki bi podpiral le programe, kot sta `cp` in `ls`. `/usr` vsebuje večino večjih programskih paketov in namestitvene datoteke, ki jih spremljajo.

`/usr/X11R6` `/usr/X11R6` vsebuje grafični vmesnik X Window System, če ste ga namestili. X Window System je veliko, zmogljivo grafično okolje, ki ponuja veliko število grafičnih pripomočkov in programov, prikazanih v »oknih« na vašem zaslonu. Če ste navajeni na okolja Microsoft Windows ali Macintosh, se vam bo tudi okenski sistem X Window zdel domač. Imenik `/usr/X11R6` vsebuje vse izvedljive datoteke za sistem X Window, nastavitvene datoteke in podporne datoteke. To je podrobneje pokrito v poglavju 5.

`/usr/bin` `/usr/bin` je pravo skladišče programja na vsakem sistemu Linux. Vsebuje večino izvedljivih datotek za programe, ki jih ne najdete na drugih

mestih, kot `/bin`.

`/usr/etc` Kakor `/etc` vsebuje različne nujno potrebne sistemske programe in namestitvene datoteke, tako `/usr/etc` vsebuje različne pripomočke in datoteke, ki v splošnem niso nujni za sistem.

`/usr/include` `/usr/include` vsebuje **glave** za prevajalnik programskega jezika C. Te datoteke (večina od njih se konča s `.h` kot »glava« (angl. header)) določajo imena podatkovnih struktur, podprogramov in konstant, ki se uporabljajo pri pisanju programov v C-ju. Datoteke v imeniku `/usr/include/sys` se v splošnem uporabljajo za programiranje na sistemskem nivoju Unixa. Če ste domači v programskem jeziku C, boste tukaj našli datoteke z glavami kot `stdio.h`, ki deklarirajo funkcije, kot je `printf()`.

`/usr/g++-include`
`/usr/g++-include` vsebuje vključne datoteke za prevajalnik za C++ (precej podobno kot `/usr/include`).

`/usr/lib` `/usr/lib` vsebuje »štoraste« in »statične« ekvivalente datotek, najdenih v `/lib`. Ko prevajate program, se program »povezuje« s knjižnicami, najdenimi v `/usr/lib`, ki potem naročijo programu, naj pogleda v `/lib`, ko potrebuje pravo kodo. Poleg tega tudi različni drugi programi shranjujejo nastavitvene datoteke v imeniku `/usr/lib`.

`/usr/local` `/usr/local` je precej podoben `/usr` – vsebuje različne programe in datoteke, ki niso nujne za sistem, a naredijo sistem zabaven in vznemirljiv. V splošnem so programi v `/usr/local` specializirani za vaš sistem – posledično se `/usr/local` zelo razlikuje pri posameznih sistemih Linux.

`/usr/man` Ta imenik vsebuje strani priročnika. V njem sta dva podimenika za vsak »razdelek« (angl. section) strani priročnika (uporabite ukaz `man man` za podrobnosti). Na primer, `/usr/man/man1` vsebuje izvorno kodo (se pravi, neformatirani izvirnik) za strani priročnika v razdelku 1, `/usr/man/cat1` pa vsebuje formatirane strani priročnika za razdelek 1.

`/usr/src` `/usr/src` vsebuje izvorno kodo (neprevedena navodila) za različne programe na vašem sistemu. Najpomembnejši imenik tukaj je `/usr/src/linux`, ki vsebuje izvorno kodo jedra za Linux.

`/var` `/var` vsebuje imenike, katerih velikost se pogosto spreminja ali se sčasoma veča. Večina od teh imenikov je včasih ležala v `/usr`, a ker skušajo tisti, ki podpirajo Linux, ta imenik obdržati relativno nespremenljiv, so se imeniki, ki se pogosto spreminjajo, preselili v `/var`. Nekatere distribucije Linuxa vzdržujejo svoje baze podatkov o programskih paketih v imenikih pod `/var`.

`/var/log` `/var/log` vsebuje različne datoteke, ki zanimajo sistemskega upravitelja, posebej dnevnike sistemske aktivnosti, ki zapisujejo napake ali probleme sistema. Druge datoteke beležijo prijave v sistem kot tudi neuspele

poskuse prijav. To bo pokrito v poglavju 4.

`/var/spool` `/var/spool` vsebuje datoteke, ki v vrsti čakajo na obdelavo. Te so »navite« (angl. spooled) na neki program. Na primer, če imate tiskalnik, se datoteke, ki čakajo na tiskanje, začasno shranjujejo v imeniku `/var/spool/lp`. Če je vaš stroj povezan v omrežje, se prihajajoča pošta shranjuje v imeniku `/var/spool/mail`, dokler je ne preberete ali pobrišete, odhajajoči ali prihajajoči novičarski članki so v imeniku `/var/spool/news` in tako naprej.

3.7 Tipi ukaznih lupin

Kot je bilo že omenjeno, je Linux večopravilni, večuporabniški operacijski sistem. Večopravilnost je *zelo* uporabna, in ko jo enkrat razumete, jo boste uporabljali vseskozi. Ne bo dolgo, ko boste poganjali programe v ozadju, preklapljali med opravili in povezovali programe s cevovodi, da boste dosegli zapletene rezultate z enim ukazom.

Veliko lastnosti, ki jih pokrivamo v tem razdelku, ponuja sama ukazna lupina. Bodite previdni, da ne boste zamešali Linuxa (pravega operacijskega sistema) z ukazno lupino – lupina je le vmesnik do spodaj ležečega sistema. Ukazna lupina priskrbi funkcionalnost poleg samega Linuxa.

Ukazna lupina ni le tolmač za interaktivne ukaze, ki jih vpisujete v pozornik, ampak ima vgrajen tudi zmogljiv programski jezik. Omogoča vam pisanje **lupinskih skriptov** (angl. shell scripts) za »pakiranje« različnih lupinskih ukazov v skupno datoteko. Če poznate MS-DOS, boste prepoznali podobnost s »paketnimi datotekami«. Lupinski skripti so zelo zmogljivo orodje, ki vam omogoča avtomatizacijo in razširitev vaše uporabe Linuxa. Glejte stran 141 za več informacij.

V svetu Linuxa obstajajo različni tipi ukaznih lupin. Dva glavna tipa sta »Bournova lupina« in »lupina C« (C shell). Bournova lupina uporablja ukazno skladnjo, podobno originalni ukazni lupini na zgodnjih sistemih Unix, kot je System III. Ime Bournove lupine na večini sistemov Linux je `/bin/sh` (kjer `sh` pomeni »shell«, ukazno lupino). Lupina C uporablja drugačno skladnjo, nekako podobno programskem jeziku C, in je na večini sistemov Linux imenovana `/bin/csh`.

Pod Linuxom so dostopne različne variacije teh ukaznih lupin. Dve najpogosteje uporabljani sta ukazna lupina Bourne Again Shell ali »Bash« (`/bin/bash`) in »Tcsh« (`/bin/tcsh`). `bash` je oblika Bournove lupine, ki vključuje veliko naprednih značilnosti, ki jih najdemo v lupini C. Ker `bash` podpira nadmnožico skladnje Bournove lupine, bo lupinski skript, napisan v standardni Bournovi lupini, deloval tudi z lupino `bash`. Če raje uporabljate skladnjo lupine C, Linux podpira `tcsh`, ki je razširjena različica izvorne lupine C.

Tip uporabljane ukazne lupine, za katerega se odločite, je predvsem religiozno vprašanje. Nekaterim je ljubša skladnja Bournove lupine z naprednimi odlikami `bash`-a, drugim je ljubša bolj strukturirana skladnja lupine C. Kar se tiče navadnih ukazov, kot sta `cp` in `ls`, ukazna lupina, ki jo uporabljate, ni pomembna – skladnja je enaka. Le kadar začnete pisati lupinske skripte ali uporabljati napredne značilnosti lupine, igrajo vlogo razlike med različnimi tipi ukaznih lupin.

Ko razlagamo značilnosti različnih ukaznih lupin, bomo opazili različnosti med Bour-

novo in lupino C. Vendar so za potrebe tega priročnika te razlike večinoma minimalne. (Če ste na tej točki zares radovedni, preberite strani priročnika o lupinah `bash` in `tcsh`).

3.8 Džokerji

Glavna značilnost večine ukaznih lupin Linuxa je zmožnost sklicevanja na več kot eno datoteko z uporabo posebnih znakov. Ti **džokerji** (angl. wildcards) vam omogočajo sklicevanje na, denimo, vsa imena datotek, ki vsebujejo znak »`n`«.

Džoker »`*`« lahko zamenja katerikoli znak ali niz znakov v imenu datoteke. Ko v imenu datoteke uporabite znak »`*`«, ga ukazna lupina zamenja z vsemi mogočimi zamenjavami iz imen datotek, ki so v imeniku, v katerem se sklicujete.

Tukaj je preprost primer. Denimo, da ima Larry v svojem trenutnem imeniku datoteke `frog`, `joe` in `stuff`.

```
/home/larry$ ls
frog    joe    stuff
/home/larry$
```

Za določitev vseh datotek, katerih ime vsebuje črko »o«, uporabite ukaz

```
/home/larry$ ls *o*
frog    joe
/home/larry$
```

Kot lahko vidite, je vsaka pojavitev »`*`« zamenjana z vsemi nadomestitvami, ki ustrezajo džokerju iz imen datotek v trenutnem imeniku.

Uporaba samega znaka »`*`« preprosto ustreza vsem imenom datotek, saj vsi znaki ustrezajo džokerju.

```
/home/larry$ ls *
frog    joe    stuff
/home/larry$
```

Tukaj je še nekaj več primerov:

```
/home/larry$ ls f*
frog
/home/larry$ ls *ff
stuff
/home/larry$ ls *f*
frog    stuff
/home/larry$ ls s*f
stuff
/home/larry$
```

Proces spremembe »`*`« v seznam imen datotek se imenuje **razvitje džokerjev** (angl. wildcard expansion) in ga opravi ukazna lupina. To je pomembno: posamezen ukaz, kot je `ls`,*nikoli* ne vidi »`*`« v svojem seznamu parametrov. Ukazna lupina razvije džoker tako, da vključi vsa imena datotek, ki mu ustrezajo. Torej ukaz

```
/home/larry$ ls *o*
```

razvije ukazna lupina v ukaz

```
/home/larry$ ls frog joe
```

Pomembno opozorilo o džokerju »*«: ta *ne* ustreza imenom datotek, ki se začenjajo s piko (».«). Te datoteke veljajo za **skrite** datoteke – čeprav niso zares skrite, se ne prikažejo v običajnih izpisih programa `ls` in jih uporaba džokerja »*« ne zadeva.

Tukaj je primer. Omenili smo že, da vsak imenik vsebuje dva posebna vnosa: ».« se nanaša na trenutni imenik, in »..« na starševski imenik. Vendar ko uporabljate `ls`, se ta dva vnosa ne prikažeta.

```
/home/larry$ ls
frog     joe     stuff
/home/larry$
```

Če uporabite stikalo `-a` ukaza `ls`, pa lahko prikažete imena datotek, ki se začenjajo s piko. Poglejte:

```
/home/larry$ ls -a
.     ..     .bash_profile     .bashrc     frog      joe     stuff
/home/larry$
```

Seznam vsebuje dva posebna vnosa, ».« in »..«, kot tudi dve drugi »skriti« datoteki – `.bash_profile` in `.bashrc`. Ti dve datoteki sta začetni datoteki, ki ju uporablja `bash`, ko se `larry` prijavi v sistem. Opisani sta od strani 145 naprej.

Upoštevajte, da se ob uporabi džokerja »*« ne prikaže nobeno ime datoteke, ki se začne z ».«.

```
/home/larry$ ls *
frog     joe     stuff
/home/larry$
```

To je varnostna lastnost: če bi znak »*« ustrezal imenom datotek, ki se začnejo z ».«, bi tudi ustrezal imenom imenikov ».« in »..«. To je lahko pri uporabi nekaterih ukazov nevarno.

Še en džoker je »?«. Džoker »?« se lahko zamenja v en sam znak. Torej, »`ls ?`« prikaže vsa imena datotek z enim samim znakom. In »`ls termca?`« bi prikazalo »`termcap`« a *ne tudi* »`termcap.backup`«. Tukaj je še en primer:

```
/home/larry$ ls j?e
joe
/home/larry$ ls f??g
frog
/home/larry$ ls ????f
stuff
/home/larry$
```

Kot lahko vidite, vam džokerji dovoljujejo določitev veliko datotek hkrati. V pregledu ukazov, ki se začenja na strani 114, smo dejali, da lahko ukaza `cp` in `mv` pravzaprav prepisujeta ali premikata več kot eno datoteko naenkrat. Na primer,

```
/home/larry$ cp /etc/s* /home/larry
```

prepiše vse datoteke iz `/etc`, ki se začenjajo s »s«, v imenik `/home/larry`. Oblika uporabe ukaza `cp` je v resnici

`cp` *datoteke cilj*

kjer *datoteke* naštevajo ime datotek za prepis in je *cilj* ciljna datoteka ali imenik. `mv` ima identično skladnjo.

Če prepisujete ali premikate več kot eno datoteko, mora biti *cilj* imenik. Le eno samo datoteko lahko prepišete ali premaknete v drugo datoteko.

3.9 Vodovodne inštalacije Linuxa

3.9.1 Standardni vhod in standardni izhod

Veliko ukazov Linuxa bere vhod in podatke iz tega, kar se imenuje **standardni vhod** in pošilja svoj izhod na **standardni izhod** (pogosto sta ta dva pojma okrajšana kot `stdin` in `stdout`). Vaša ukazna lupina uredi zadeve tako, da je standardni vhod vaša tipkovnica, standardni izhod pa zaslon.

Tukaj je primer uporabe ukaza `cat`. Navadno bere `cat` podatke iz vseh datotek, določenih v ukazni vrstici, in pošlje te podatke neposredno na `stdout`. Torej uporaba ukaza

```
/home/larry/papers$ cat history-final masters-thesis
```

izpiše vsebino datoteke `history-final`, ki ji sledi vsebina datoteke `masters-thesis`.

A če ne določite imena datoteke, `cat` bere podatke iz `stdin` in jih pošilja na `stdout`. Tukaj je primer:

```
/home/larry/papers$ cat
Pozdravljeni.
Pozdravljeni.
Na svidenje.
Na svidenje.
Ctrl-D
/home/larry/papers$
```

Vsaka vrstica, ki ste jo vpisali, takoj odmeva nazaj v `cat`-u. Ko berete s standardnega vhoda, določite, da je vaš vnos »končan« tako, da pošljete signal za konec teksta, imenovan signal EOT (iz angl. end-of-text signal), v splošnem ga dobite s pritiskom `Ctrl-D`.

Tukaj je še en primer. Ukaz `sort` prebere vrstice teksta (spet iz standardnega vhoda, razen če določite eno ali več datotek) in pošlje urejen izhod na standardni izhod. Poskusite naslednje.

```
/home/larry/papers$ sort
korenje
banane
jabolka
Ctrl-D
banane
jabolka
korenje
/home/larry/papers$
```

Zdaj lahko svoj nakupovalni spisek uredimo po abecedi ... mar ni Linux uporaben?

3.9.2 Preusmeritev vhoda in izhoda

Zdaj denimo, da želite poslati izhod ukaza `sort` v datoteko, da boste shranili vaš nakupovalni spisek na disk. Ukazna lupina vam omogoča **preusmeritev** standardnega izhoda v datoteko z uporabo znaka »>«. Takole to deluje:

```
/home/larry/papers$ sort > nakupovalni-spisek
korenje
banane
jabolka
Ctrl-D
/home/larry/papers$
```

Kot lahko vidite, se rezultat ukaza `sort` ni prikazal, a je shranjen v datoteki, imenovani `nakupovalni-spisek`. Poglejmo vsebino te datoteke:

```
/home/larry/papers$ cat nakupovalni-spisek
banane
jabolka
korenje
/home/larry/papers$
```

Zdaj lahko uredite vaš nakupovalni spisek in ga tudi shranite! A denimo, da shranjujete neurejen, izvorni nakupovalni spisek v datoteko `stvari`. En način urejanja te informacije in shranjevanja v datoteko bi bil, da daste ukazu `sort` ime datoteke za branje namesto standardnega vhoda, in preusmerite standardni izhod, kot smo to naredili zgoraj. Torej takole:

```
/home/larry/papers$ sort stvari > nakupovalni-spisek
/home/larry/papers$ cat nakupovalni-spisek
banane
jabolka
korenje
/home/larry/papers$
```

Vendar obstaja tudi druga pot za dosego tega. Ne le, da lahko preusmerite standardni izhod, preusmerite lahko tudi standardni *vhod*, z uporabo znaka »<«.

```
/home/larry/papers$ sort < stvari
banane
jabolka
korenje
/home/larry/papers$
```

Tehnično je »`sort < stvari`« ekvivalenten »`sort stvari`«, a vam omogoča demonstracijo naslednje stvari: `sort < stvari` se obnaša, kot da bi bili podatki v datoteki `stvari` vtipkani na standardnem vhodu. Ukazna lupina ureja preusmerjanje. Ukazu `sort` ni dano ime datoteke (`stvari`), s katere naj bere; kar se tiče `sort`-a, še vedno bere s standardnega vhoda, kot če bi vnesli podatke z vašo tipkovnico.

To uvaja koncept **filtra**. Filter je program, ki bere podatke s standardnega vhoda, jih na nek način obdela in pošlje obdelane podatke na standardni izhod. Z uporabo preusmerjevanja (angl. redirection) se na standardni vhod in izhod lahko sklicujete iz datotek. Kot smo

omenili, `stdin` in `stdout` privzeto ustrezata tipkovnici in zaslonu, po vrsti. Pripomoček `sort` je preprost filter. Uredi vhodne podatke in pošlje rezultat na standardni izhod. Pripomoček `cat` je še preprostejši. Ničesar ne naredi z vhodnimi podatki, le izpiše, kar pač že dobi.

3.9.3 Uporaba cevi

Pokazali smo že, kako uporabljati `sort` kot filter. Vendar ti primeri predpostavljajo, da imate podatke nekje shranjene v datoteki ali ste pripravljeni sami vpisati vse podatke standardnega vhoda. Kaj, če podatki, ki jih želite urediti, pridejo kot izhod nekega drugega ukaza, kot je `ls`?

Izbira `-r` ukaza `sort` uredi podatke v obratnem abecednem vrstnem redu. Če želite našteti imena datotek v vašem trenutnem imeniku v obratnem vrstnem redu, je en način za to naslednji:

```
/home/larry/papers$ ls
english-list
history-final
masters-thesis
notes
```

Zdaj preusmerimo izhod ukaza `ls` v datoteko, imenovano `file-list`:

```
/home/larry/papers$ ls > file-list
/home/larry/papers$ sort -r file-list
notes
masters-thesis
history-final
english-list
/home/larry/papers$
```

Tukaj shranjujemo izhod ukaza `ls` v datoteko, in potem na tej datoteki poženemo `sort -r`. A to je nepripravno in uporablja začasno datoteko za shranjevanje podatkov iz `ls`.

Rešitev je uporaba **cevovoda** (angl. pipeline). To je lastnost ukazne lupine, ki poveže niz ukazov s »cevjo« (angl. pipe). Standardni izhod prvega ukaza se pošlje na `stdin` drugega ukaza. V tem primeru želimo poslati `stdout` ukaza `ls` na `stdin` ukaza `sort`. Uporabimo znak »|« za ustvarjanje cevi, kot sledi:

```
/home/larry/papers$ ls | sort -r
notes
masters-thesis
history-final
english-list
/home/larry/papers$
```

Ta ukaz je krajši in lažji za vnos.

Tukaj je še en uporaben primer, ukaz

```
/home/larry/papers$ ls /usr/bin
```

izpiše dolg seznam imen datotek, večina od njih preleti zaslon prehitro, da bi jih lahko prebrali. Zato uporabimo `more` za prikaz seznama datotek v `/usr/bin`.

```
/home/larry/papers$ ls /usr/bin | more
```

Zdaj se lahko pomikate po seznamu teh datotek po svoji volji.

A zabava se tu še ne konča! S cevmi lahko povežete skupaj tudi več kot dva ukaza. Ukaz `head` je filter, ki prikaže prve vrstice vhodnega toka (v tem primeru, vhod iz cevi). Če želite prikazati ime datoteke v trenutnem imeniku, zadnje po abecednem redu, uporabljajte ukaze, kot so naslednji:

```
/home/larry/papers$ ls | sort -r | head -1
notes
/home/larry/papers$
```

kjer `head -1` izpiše prvo vrstico vhoda, ki jo sprejme (v tem primeru tok podatkov iz `ls`, urejenih po obratnem abecednem redu).

3.9.4 Ne-uničevalna preusmeritev izhoda

Uporaba »>« za preusmeritev izhoda v datoteko je uničevalna; z drugimi besedami, ukaz

```
/home/larry/papers$ ls > file-list
```

prepiše vsebino datoteke `file-list`. Če namesto tega preusmerjate s simbolom »>>«, bo izhod pripet na (dodan na konec) poimenovano datoteko, namesto da bi jo prepisal. Na primer,

```
/home/larry/papers$ ls >> file-list
```

pripne izhod ukaza `ls` na datoteko `file-list`.

Zavedajte se, da so preusmerjanje in cevi lastnosti ukazne lupine – ki podpira uporabo »>«, »>>« in »|«. To nima nič opraviti s samimi ukazi.

3.10 Dovoljenja datotek

3.10.1 Koncepti dovoljenj datotek

Ker je na sistemu Linux tipično več kot en uporabnik, Linux ponuja mehanizem, znan kot **dovoljenja datotek**, ki ščiti vaše uporabniške datoteke pred brkljanjem drugih uporabnikov. Ta mehanizem omogoča datotekam in imenikom, da so »last« določenega uporabnika. Na primer, ker je Larry ustvaril datoteke v svojem domačem imeniku, je Larry njihov lastnik in ima do njih dostop.

Linux tudi dovoljuje delitev datotek med uporabniki in skupinami uporabnikov. Če Larry tako želi, lahko onemogoči dostop do svojih datotek, da do njih ne bo mogel dostopati noben drug uporabnik. Vendar je na večini sistemov privzeto dovoliti drugim uporabnikom branje svojih datotek, ne pa tudi njihovo spreminjanje ali brisanje.

Vsako datoteko si lasti določen uporabnik. Vendar so datoteke tudi last določene **skupine** (angl. group) uporabnikov sistema. Vsak uporabnik se ob stvaritvi uporabniškega računa dodeli vsaj v eno skupino. Sistemski upravitelj pa lahko podeli uporabniku dostop do več kot ene tovrstne skupine.

Skupine navadno določa vrsta uporabnikov, ki dostopajo do stroja. Na primer, na univerzitetnem sistemu Linux so lahko uporabniki razdeljeni v skupine `studenti`, `osebje`,

predavatelji ali gostje. Obstaja tudi nekaj sistemsko definiranih skupin (kot sta skupini bin in admin), ki jih uporablja sam sistem za nadzor dostopa do virov – zelo redko v ti sistemski skupini spadajo pravi uporabniki.

Dovoljenja so razdeljena v tri glavne oddelke: branje, pisanje in izvajanje. Ta dovoljenja so lahko dana trem razredom uporabnikov: lastniku datoteke, skupini, kateri pripada datoteka, in vsem uporabnikom, ne glede na skupino.

Dovoljenje za branje pusti uporabniku brati vsebino datoteke ali, v primeru imenikov, izpis vsebine imenika (z uporabo ls). Dovoljenje za pisanje dovoljuje uporabniku pisanje ali spreminjanje datoteke. Pri imenikih, dovoljenje za pisanje dovoljuje uporabniku ustvarjanje novih datotek ali brisanje starih datotek v tem imeniku. Končno, dovoljenje za izvajanje dovoljuje uporabniku pogon datoteke kot programa ali skripta ukazne lupine (če je datoteka program ali skript ukazne lupine). Pri imenikih omogoča posest dovoljenja za izvajanje uporabniku cd v ta imenik.

3.10.2 Razlaga dovoljenj datotek

Poglejmo primer, ki demonstrira dovoljenja datotek. Z uporabo ukaza ls z izbiro -l se prikaže »dolgi« izpis imena datoteke, vključno z dovoljenji za uporabo te datoteke.

```
/home/larry/foo$ ls -l stuff
-rw-r--r--   1 larry    users         505 Mar 13 19:05 stuff
/home/larry/foo$
```

Prvo polje v izpisu predstavlja dovoljenja za uporabo datoteke. Tretje polje vsebuje lastnika datoteke (to je larry) in četrto polje je skupina, kateri datoteka pripada (users). Zadnje polje je očitno ime datoteke (stuff). Druga polja bomo obravnavali pozneje.

To datoteko si lasti larry, in pripada skupini users. Niz -rw-r--r-- po vrsti našteva dovoljenja, dana lastniku datoteke, skupini datoteke in vsem ostalim.

Prvi znak niza dovoljenj (»-«) predstavlja tip datoteke. Znak »-« pomeni, da je to običajna datoteka (za razliko od imenika ali gonilnika naprave). Naslednji trije znaki (»rw-«) predstavljajo dovoljenja, podeljena lastniku datoteke larry. Črka »r« pomeni »branje« (angl. **r**ead) in črka »w« pomeni »pisanje« (angl. **w**rite). Torej, larry ima dovoljenja za branje in pisanje datoteke stuff.

Kot smo omenili, poleg dovoljenj za branje in pisanje obstaja tudi dovoljenje za »izvajanje« (angl. e**x**ecute) – predstavlja ga črka »x«. Vendar je tukaj namesto »x« naštet »-«, torej Larry nima dovoljenja za poganjanje te datoteke. To je v redu, saj datoteka stuff ni program kakršnekoli vrste. Seveda, ker je Larry lastnik te datoteke, si lahko podeli dovoljenja za izvajanje datoteke stuff, če tako želi. (O tem bomo spregovorili v kratkem.)

Naslednji trije znaki, (»r--«), predstavljajo dovoljenja skupine do uporabe datoteke. Skupina, ki si lasti to datoteko, se imenuje users. Ker se tukaj pojavlja le »r«, lahko vsak uporabnik, ki pripada skupini users, bere to datoteko.

Zadnji trije znaki, tudi (»r--«), predstavljajo dovoljenja, podeljena vsem uporabnikom sistema (razen lastnika datoteke in uporabnikom skupine users). Spet, ker je prisoten le »r«, lahko drugi uporabniki berejo datoteko, a ne morejo vanjo pisati ali je izvajati.

Tukaj je nekaj drugih primerov dovoljenj:

-rwxr-xr-x Lastnik datoteke lahko bere, piše in izvaja datoteko. Uporabniki v skupini datoteke in vsi drugi uporabniki lahko berejo in izvajajo datoteko.

`-rw-------` Lastnik te datoteke lahko bere in piše v datoteko. Do datoteke ne more dostopati noben drug uporabnik.

`-rwxrwxrwx` Vsi uporabniki lahko berejo, pišejo in izvajajo datoteko.

3.10.3 Odvisnosti dovoljenj

Dovoljenja za dostop do datoteke so tudi odvisna od dovoljenj za dostop do imenika, v katerem je datoteka. Na primer, če so dovoljenja datoteke nastavljena na `-rwxrwxrwx`, drugi uporabniki ne morejo dostopati do datoteke, razen če imajo bralni ali izvajalni dostop do imenika, v katerem je datoteka. Na primer, če bi Larry želel omejiti dostop do vseh svojih datotek, bi lahko nastavil dovoljenja svojega domačega imenika `/home/larry` na `-rwx------`. Na ta način noben drug uporabnik nima dostopa do njegovega imenika in datotek in imenikov v njem. Larryju ni treba skrbeti o posameznih dovoljenjih za vsako od njegovih datotek.

Z drugimi besedami, če želite sploh dostopati do datoteke, morate imeti izvajalni dostop do vseh imenikov po poti datoteke in dovoljenje za branje (ali izvajanje) same datoteke.

Uporabniki na sistemu Linux so, navadno, zelo radodarni s svojimi datotekami. Običajen nabor dovoljenj za dostop do datotek je `-rw-r--r--`, kar omogoča branje datoteke drugim uporabnikom, a brez vsakih sprememb. Običajen nabor dovoljenj za imenike je `-rwxr-xr-x`, kar omogoča drugim uporabnikom sprehod po imenikih, a brez ustvarjanja ali brisanja datotek v njih.

Vendar veliko uporabnikov želi obdržati druge uporabnike čimdlje od njihovih datotek. Nastavitve dovoljenj datoteke na `-rw-------` bo preprečevala vsem drugim uporabnikom dostop do datoteke. Podobno, nastavitev dovoljenj imenika na `-rwx------` prepreči vstop drugim uporabnikom v ta imenik.

3.10.4 Spreminjanje dovoljenj

Ukaz `chmod` se uporablja za nastavitev dovoljenj za dostop do datoteke. Dovoljenja lahko spreminja le lastnik datoteke. Skladnja ukaza `chmod` je

`chmod {a,u,g,o}{+,-}{r,w,x}` *imena_datotek*

Na kratko, podajte eno ali več od črk »a« (vsi, angl. **a**ll), »u« (uporabnik, angl. **u**ser), »g« (skupina, angl. **g**roup) ali »o« (ostali, angl. **o**ther). Potem določite, ali naj se pravice dodajo (`+`) ali odvzamejo (`-`). Končno, določite še eno ali več dovoljenj za branje (angl. **r**ead), pisanje (angl. **w**rite) in izvajanje (angl. e**x**ecute). Nekateri primeri legalnih ukazov so:

`chmod a+r stuff`
: To podeli vsem uporabnikom dovoljenje za branje te datoteke.

`chmod +r stuff`
: Kot zgoraj – če ni določen noben znak od `a`, `u`, `g` ali `o`, se privzame `a`.

`chmod og-x stuff`
: Prekliče dovoljenje za izvajanje vsem uporabnikom razen lastniku.

`chmod u+rwx stuff`
: Dovoli uporabniku datoteke branje, pisanje in izvajanje datoteke.

`chmod o-rwx stuff`

Prekliče dovoljenja za branje, pisanje in izvajanje vsem uporabnikom razen lastniku in uporabnikom v skupini, v katere lasti je datoteka.

3.11 Upravljanje s povezavami datotek

Povezave vam omogočijo, da daste eni datoteki več kot eno ime. Sistem navadno prepozna datoteke po njihovem **številu inode**, ki je le enoličen sistemski razpoznavni znak te datoteke. Imenik je navadno seznam števil inode z njihovimi ustreznimi imeni datotek. Vsako ime datoteke v imeniku je **povezava** (angl. link) do določene informacije inode.

3.11.1 Trde povezave

Ukaz `ln` se uporablja za ustvarjanje več povezav do ene datoteke. Denimo, da imate v imeniku datoteko, imenovano `foo`. Z uporabo `ls -i` lahko pogledate število inode za to datoteko.

```
/home/larry$ ls -i foo
22192 foo
/home/larry$
```

Tukaj ima `foo` v datotečnem sistemu prirejeno število inode 22192. Takole lahko ustvarite še eno povezavo, po imenu `bar`, do `foo`:

```
/home/larry$ ln foo bar
```

Z uporabo `ls -i` boste videli, da imata ti dve datoteki isto število inode.

```
/home/larry$ ls -i foo bar
22192 bar   22192 foo
/home/larry$
```

Zdaj lahko do iste datoteke dostopate prek imena `foo` ali pa `bar`. Če naredite spremembe v datoteki `foo`, se te spremembe pojavijo tudi v datoteki `bar`. Za vse namene sta `foo` in `bar` ista datoteka.

Tovrstne povezave so znane kot **trde povezave** (angl. hard links), saj ustvarijo neposredno povezavo na inode. Upoštevajte, da lahko trdo povežete datoteki le, če ležita na istem datotečnem sistemu; simbolne povezave (glejte spodaj) nimajo te omejitve.

Ko pobrišete datoteko z `rm`, pravzaprav pobrišete le eno povezavo do nje. Če uporabite ukaz

```
/home/larry$ rm foo
```

se pobriše le povezava, imenovana `foo`, `bar` bo še vedno obstajala. Datoteka je zares odstranjena s sistema le, ko ni več povezav do nje. Navadno imajo datoteke le po eno povezavo, zato uporaba ukaza `rm` pobriše datoteko. Vendar če ima datoteka več povezav do nje, bo uporaba `rm` pobrisala le eno povezavo; za brisanje datoteke morate pobrisati vse povezave do nje.

Ukaz `ls -l` prikaže število povezav do datoteke (med ostalimi informacijami).

```
/home/larry$ ls -l foo bar
-rw-r--r--   2 root     root          12 Aug  5 16:51 bar
-rw-r--r--   2 root     root          12 Aug  5 16:50 foo
/home/larry$
```

Drugi stolpec v izpisu, »2«, prikazuje število povezav do datoteke.

Kot se izkaže, je imenik pravzaprav le datoteka, ki vsebuje informacije o zvezah med povezavami in inodi. Vsak imenik vsebuje tudi vsaj dve trdi povezavi: »`.`« (povezavo, ki kaže nanj) in »`..`« (povezavo, ki kaže na starševski imenik). Povezava »`..`« korenskega imenika (`/`) kaže le nazaj na `/`. (Z drugimi besedami, starševski imenik korenskega imenika je sam korenski imenik.)

3.11.2 Simbolne povezave

Simbolne povezave (angl. symbolical links, ali symlinks) so še ena vrsta povezave, ki je drugačna od trdih povezav. Simbolna povezava vam omogoča določitev drugega imena, a ne poveže datoteke z inode.

Ukaz `ln -s` ustvari simbolno povezavo do datoteke. Na primer, če uporabljate ukaz

```
/home/larry$ ln -s foo bar
```

boste ustvarili simbolno povezavo, imenovano `bar`, ki kaže na datoteko `foo`. Če uporabite `ls -i`, boste videli, da imata ti dve datoteki zares različni števili inode.

```
/home/larry$ ls -i foo bar
22195 bar   22192 foo
/home/larry$
```

Vendar z uporabo `ls -l` vidimo, da je datoteka `bar` simbolna povezava, ki kaže na `foo`.

```
/home/larry$ ls -l foo bar
lrwxrwxrwx   1 root     root           3 Aug  5 16:51 bar -> foo
-rw-r--r--   1 root     root          12 Aug  5 16:50 foo
/home/larry$
```

Dovoljenja datoteke se pri simbolni povezavi ne uporabljajo (vedno so oblike `rwxrwxrwx`). Namesto tega se uporabljajo dovoljenja cilja simbolne povezave (v našem primeru datoteke `foo`).

Funkcionalno so trde in simbolne povezave podobne, a obstajajo razlike. Ena od njih je, da lahko ustvarite simbolno povezavo do datoteke, ki ne obstaja; kar ne velja za trde povezave. Simbolne povezave obdela jedro drugače kot trde, kar je le tehnična, a včasih pomembna, razlika. Simbolne povezave so nam v pomoč, ker določajo datoteko, na katero kažejo; pri trdih povezavah ni preprostega načina za ugotovitev, katere datoteke so povezane na ista števila inode.

Povezave se uporabljajo v sistemu Linux na veliko mestih. Simbolne povezave so posebej pomembne pri slikah deljenih knjižnic v imeniku `/lib`. Glejte stran 175 za več informacij.

3.12 Nadzor opravil

3.12.1 Opravila in procesi

Nadzor opravil (angl. job control) je lastnost, ki jo ponuja mnogo ukaznih lupin (vključno z `bash` in `tcsh`), ki vam omogoča nadzor več tekočih ukazov ali **opravil** naenkrat. Preden se potopimo globlje, moramo spregovoriti o **procesih**.

Vsakič, ko poženete program, začnete to, kar se imenuje proces. Ukaz `ps` izpiše spisek procesov, ki trenutno tečejo, kot je prikazano tukaj:

```
/home/larry$ ps
  PID TT STAT  TIME COMMAND
   24  3 S     0:03 (bash)
  161  3 R     0:00 ps
/home/larry$
```

Številka `PID` v prvem stolpcu je **identifikacijska številka procesa**, unikatna številka, dodeljena vsakemu tekočemu procesu. Zadnji stolpec, `COMMAND`, je ime tekočega ukaza. Tukaj gledamo le procesa, ki ju trenutno poganja sam Larry. (Na sistemu obstajajo tudi drugi tekoči procesi – »`ps aux`« jih vse izpiše.) To sta `bash` (Larryjeva ukazna lupina) in sam ukaz `ps`. Kot vidite, teče `bash` vzporedno z ukazom `ps`. `bash` je izvedel ukaz `ps`, ko ga je Larry vtipkal. Ko `ps` preneha delovati (po izpisu tabele procesov), se nadzor vrne procesu `bash`, ki izpiše pozornik, pripravljen na naslednji ukaz.

Tekoči proces se tudi imenuje *opravilo* (angl. job). Izraza *proces* in *opravilo* sta zamenljiva. Vendar se o procesu navadno govori kot o »opravilu«, ko se uporablja v povezavi z **nadzorom opravil** – značilnosti ukazne lupine, da vam omogoča preklapljanje med različnimi neodvisnimi opravili.

V večini primerov uporabniki poganjajo le eno opravilo hkrati – katerikoli ukaz so pač zadnji vnesli v ukazno lupino. Vendar pa z uporabo nadzora opravil lahko hkrati poganjate več opravil in po potrebi preklapljate med njimi.

Kako je lahko to uporabno? Denimo, da urejate tekstovno datoteko in bi radi prekinili vaše urejanje in naredili nekaj drugega. Z nadzorom opravil lahko začasno odložite urejevalnik, se vrnete v pozornik ukazne lupine in začnete delati na nečem drugem. Ko ste opravili, lahko preklopite nazaj v urejevalnik in ste tam, kjer ste bili, kot da sploh ne bi zapustili urejevalnika. Obstaja mnogo praktičnih uporab nadzora opravil.

3.12.2 Ospredje in ozadje

Opravila so lahko v **ospredju** (angl. foreground) ali v **ozadju** (angl. background). Hkrati je lahko v ospredju le eno opravilo. Opravilo v ospredju je opravilo, s katerim vzajemno delujete – sprejema vhod s tipkovnice in pošilja izhod na zaslon (razen seveda, če ste preusmerili vhod ali izhod, kot je opisano od strani 124 naprej). Po drugi strani, opravila v ozadju ne sprejemajo vhoda s terminala – v splošnem tečejo tiho, brez potrebe po interakciji.

Nekatera opravila potrebujejo dolgo časa, da se končajo, in ne počnejo ničesar zanimivega, dokler tečejo. Prevajanje programov je takšno opravilo kot tudi komprimiranje velike datoteke. Ni razloga, da bi posedali naokrog in se dolgočasili, dokler ta opravila ne končajo svojih nalog; preprosto jih poženite v ozadju. Dokler opravila tečejo v ozadju, lahko po mili volji poganjate druge programe.

Opravila so lahko **odložena**. Odloženo opravilo (angl. suspended job) je opravilo, ki je le začasno ustavljeno. Ko odložite opravilo, mu lahko poveste, naj nadaljuje v ospredju ali ozadju, če je to potrebno. Nadaljevanje odloženega opravila nikakor ne spremeni stanja opravila – opravilo nadaljuje s tekom tam, kjer je ostalo.

Odložitev opravila ni enaka prekinitvi opravila. Ko **prekinete** (angl. interrupt) tekoči proces (s pritiskom prekinitvene tipke, ki je navadno Ctrl-C)[3], se proces ubije, za vedno. Ko se opravilo ubije, ni upanja za njegovo nadaljevanje. Ponovno boste morali pognati ukaz. Nekateri programi pa prestrežejo prekinitev tako, da pritisk Ctrl-C ne ubije opravila takoj. To je zato, da se programu omogoči vse potrebne čistilne operacije, preden se konča. Pravzaprav vam nekateri programi sploh ne dovoljujejo, da jih ubijete s prekinitvijo.

3.12.3 Spravljanje v ozadje in ubijanje opravil

Začnimo s preprostim zgledom. Ukaz `yes` je navidezno neuporaben ukaz, ki pošilja neskončen tok črk `y` na standardni izhod. (To je celo uporabno. Če s cevjo povežete izhod ukaza `yes` do drugega ukaza, ki vprašuje niz vprašanj tipa da/ne (angl. yes/no), bo tok črk `y` potrdil vsa vprašanja.)

Preizkusite:

```
/home/larry$ yes
y
y
y
y
y
```

Izpis ipsilonov se bo nadaljeval *ad infinitum*. Proces lahko ubijete s pritiskom na prekinitveno tipko, ki je navadno Ctrl-C.

Da nam ne bo treba spet shajati z mučnim tokom ipsilonov, preusmerimo standardni izhod programa `yes` na napravo `/dev/null`. Kot se spomnite, `/dev/null` deluje kot »črna luknja« za podatke. Vsi podatki, poslani vanjo, izginejo. To je zelo učinkovita metoda za utišanje sicer zgovornega programa.

```
/home/larry$ yes > /dev/null
```

Ah, veliko bolje. Nič se ne izpiše, a pozornik ukazne lupine se ne prikaže nazaj. To je zato, ker `yes` še vedno teče, in pošilja te nesmiselne ipsilone na `/dev/null`. Spet za ubijanje opravila pritisnite prekinitveno tipko.

Denimo, da želite, da ukaz `yes` nadaljuje s tekom, a želite dobiti pozornik ukazne lupine nazaj, da boste lahko delali na drugih zadevah. Ukaz `yes` lahko postavite v ozadje, in mu s tem omogočite tek brez potrebe za vzajemno delovanje.

En način za postavljanje procesa v ozadje je, da pripnemo znak »&« na konec ukaza.

```
/home/larry$ yes > /dev/null &
[1] 164
/home/larry$
```

Kot lahko vidite, se je pozornik ukazne lupine vrnil. A kaj pomeni ta »`[1] 164`«? In ali ukaz `yes` zares teče?

[3] Prekinitveno tipko lahko nastavite z ukazom `stty`.

Oznaka »[1]« predstavlja **številko opravila** za trenutni proces yes. Ukazna lupina dodeli številko opravila vsakemu tekočemu opravilu. Ker je yes edino opravilo, ki teče, mu je dodeljena številka opravila 1. Oznaka »164« je ID procesa ali PID, številka, ki jo opravilu dodeli sistem. Katerokoli od teh dveh številk lahko uporabite za sklicevanje na opravilo, kot boste videli pozneje.

Zdaj vam v ozadju teče proces yes, ki nenehno pošilja tok črk y na /dev/null. Za preverbo statusa tega procesa uporabite notranji lupinski ukaz jobs.

```
/home/larry$ jobs
[1]+  Running                 yes >/dev/null  &
/home/larry$
```

Tukaj je, dovolj prepričljivo. Uporabili bi lahko tudi ukaz ps, opisan zgoraj, za preverbo statusa opravila.

Za pokončanje opravila uporabite ukaz kill. Ta ukaz vzame za svoj argument bodisi številko opravila, bodisi številko ID procesa. To je bilo opravilo številka 1, zato uporaba ukaza

```
/home/larry$ kill %1
```

pobije opravilo. Ko se sklicujete na opravilo s številko opravila, morate pred številko dodati znak za odstotek (»%«).

Zdaj ko ste pobili opravilo, spet uporabite ukaz jobs za njegovo preverbo:

```
/home/larry$ jobs
[1]+  Terminated              yes >/dev/null /home/larry$
```

Opravilo je zares mrtvo, in če spet uporabite ukaz jobs, se ne bo izpisalo nič.

Opravilo lahko pobijete tudi z uporabo številke ID procesa (PID), ki je prikazana poleg ID opravila, ko zaženete opravilo. V našem primeru je ID procesa 164, zato je ukaz

```
/home/larry$ kill 164
```

ekvivalenten ukazu

```
/home/larry$ kill %1
```

Kadar se sklicujete na opravilo z njegovo številko ID procesa, navedete številko PID brez znaka za odstotek.

3.12.4 Ustavljanje in ponovno zaganjanje opravil

Obstaja tudi drug način za postavitev opravila v ozadje. Opravilo lahko zaženete kot običajno (v ospredju), **ustavite** opravilo in ga ponovno zaženete v ozadju.

Najprej, poženite proces yes v ospredju, kot ste storili že prej:

```
/home/larry$ yes > /dev/null
```

Spet, ker yes teče v ospredju, ne bi smeli dobiti nazaj pozornika.

Zdaj, namesto da prekinete opravilo s Ctrl-C, **odložite** opravilo. Odložitev opravila ne pobije: le začasno ustavi opravilo, dokler ga ponovno ne poženete. Za to pritisnite tipko za odložitev, ki je navadno Ctrl-Z.

```
/home/larry$ yes > /dev/null
Ctrl-Z
[1]+  Stopped                 yes >/dev/null
/home/larry$
```

Ko je opravilo odloženo, preprosto ne teče. Za opravilo se ne uporablja nič procesorskega časa. Vendar lahko ponovno zaženete opravilo, kar povzroči, da opravilo spet teče, kot da se ni nikoli nič zgodilo. Opravilo bo nadaljevalo s tekom tam, kjer je končalo.

Za ponovni zagon opravila v ospredju, uporabite ukaz `fg` (iz angl. »foreground«, ospredje).

```
/home/larry$ fg
yes >/dev/null
```

Ukazna lupina spet izpiše ime ukaza, tako da se zavedate, katero opravilo ste pravkar postavili v ospredje. Spet ustavite opravilo s `Ctrl-Z`. Tokrat uporabite `bg` in postavite opravilo v ozadje. To povzroča, da ukaz teče, kot da bi ga zagnali z ukazom z »&«, kot smo videli v zadnjem razdelku.

```
/home/larry$ bg
[1]+ yes >/dev/null &
/home/larry$
```

In spet imate nazaj svoj pozornik. Ukaz `jobs` naj bi poročal, da `yes` zares teče, in opravilo lahko pobijete z ukazom `kill`, kot smo že naredili.

Kako lahko ponovno ustavite to opravilo? Uporaba `Ctrl-Z` ne bo delovala, saj je opravilo v ozadju. Odgovor je, da postavite opravilo v ospredje s `fg` in ga potem ustavite. Kot se izkaže, lahko uporabljate `fg` na ustavljenih opravilih ali opravilih v ozadju.

Obstaja velika razlika med opravilom v ozadju in opravilom, ki je ustavljeno. Ustavljeno opravilo ne teče – ne uporablja procesorskega časa in ne opravlja nobenega dela (opravilo še vedno zaseda sistemski pomnilnik, čeprav je morda izmenjano na disk). Opravilo v ozadju teče in uporablja pomnilnik, kot tudi zaključuje neko nalogo, medtem ko vi delate kaj drugega.

Včasih opravilo v ozadju poskuša na vašem terminalu prikazati tekst, kar je lahko nadležno, če poskušate delati kaj drugega. Na primer, če ste uporabili ukaz

```
/home/larry$ yes &
```

brez preusmeritve `stdout` na `/dev/null`, bo na zaslonu prikazovan tok črk `y`, ne da bi ga lahko prekinili. (Za prekinitev opravil v ozadju ne morete uporabiti `Ctrl-C`.) Za ustavitev neskončnih ipsilonov uporabite ukaz `fg`, da prenesete opravilo v ospredje, potem uporabite `Ctrl-C`, da ga pobijete.

Še eno sporočilo. Ukaza `fg` in `bg` navadno prizadeneta le opravilo, ki je bilo zadnje ustavljeno (prikazano z znakom »+« poleg številke opravila, ko uporabite ukaz `jobs`). Če hkrati poganjate več opravil, lahko postavljate opravila v ospredje ali ozadje tako, da podate številko opravila kot argument ukazu `fg` ali `bg`, kot v primeru

```
/home/larry$ fg %2
```

(za postavitev opravila številka 2 v ospredje), ali

```
/home/larry$ bg %3
```

(za postavitev opravila številka 3 v ozadje). Z ukazoma fg in bg ne morete uporabljati številk ID procesov (PID).

Nadalje, uporaba same številke opravila, kot v

```
/home/larry$ %2
```

je ekvivalentna uporabi

```
/home/larry$ fg %2
```

Spomnite se le, da je uporaba nadzora opravil lastnost ukazne lupine. Ukazi fg, bg in jobs so notranji ukazi ukazne lupine. Če iz nekega razloga uporabljate lupino, ki ne podpira nadzora opravil, ne pričakujte, da bodo ti ukazi dostopni.

Poleg tega obstajajo nekateri vidiki nadzora opravil, ki se razlikujejo med ukaznima lupinama bash in tcsh. Nekatere ukazne lupine sploh ne ponujajo nadzora opravil – večina ukaznih lupin, dostopnih za Linux, pa ga ponuja.

3.13 Uporaba urejevalnika vi

Urejevalnik besedil (angl. text editor) je program, ki se uporablja za urejanje datotek, ki so sestavljene iz besedila: pismo, program v C-ju ali sistemska nastavitvena datoteka. Medtem ko je za Linux na voljo veliko takšnih urejevalnikov, je edini urejevalnik, ki ga boste zajamčeno našli na vsakem sistemu Unix ali Linux, urejevalnik vi – tako imenovani »vizualni urejevalnik« (angl. visual editor). Urejevalnik vi ni najpreprostejši urejevalnik za uporabo niti uporaba ni zelo samoumevna. A ker je vi tako pogost v svetu Unixa/Linuxa in včasih nujno potreben, si zasluži tukajšnji opis.

Izbira vašega urejevalnika je predvsem vprašanje osebnega okusa in stila. Veliko uporabnikov ima najraje baročni, samoumevni in zmogljivi emacs – urejevalnik z več zmožnostmi kot katerikoli drugi posamični program v svetu Unixa. Na primer, Emacs ima vgrajeno lastno narečje programskega jezika LISP in ima mnogo razširitev (ena od njih je program umetne inteligence, podoben Elizi). Vendar ker je Emacs s svojo podporo relativno velik, morda na nekaterih sistemih ne bo nameščen. Urejevalnik vi, po drugi strani, je majhen in zmogljiv, a zapleten za uporabo. A ko ga enkrat obvladate, je vi pravzaprav zelo enostaven.

Ta razdelek predstavlja uvod v vi – ne bomo razlagali vseh njegovih značilnosti, le tiste, ki jih potrebujete, da začnete. Pogledate lahko stran referenčnega priročnika o vi, če bi se radi naučili več o značilnostih tega urejevalnika. Ali pa preberete knjigo *Learning the vi Editor* založbe O'Reilly and Associates ali *VI Tutorial* založbe Specialized Systems Consultants (SSC) Inc. Glejte dodatek A za informacije.

3.13.1 Pojmi

Ko uporabljate vi, ste v kateremkoli času v enem od treh načinov delovanja. Ti načini se imenujejo *ukazni način* (angl. command mode), *način za vstavljanje* (angl. insert mode) in *način zadnje vrstice* (angl. last line mode).

Ko poženete vi, ste v *ukaznem načinu*. Ta način vam omogoča uporabo ukazov za urejanje datotek ali spremembo v druga dva načina. Na primer, če v ukaznem načinu pritisnete [x], pobrišete znak pod kazalcem. Tipke s puščicami med urejanjem premikajo

kazalec po datoteki. V splošnem so ukazi, uporabljani v ukaznem načinu, dolgi en ali dva znaka.

Besedilo zares vrivate ali urejate v *načinu za vstavljanje*. Ko boste uporabljali `vi`, boste verjetno preživeli večino vašega časa v tem načinu. Način za vstavljanje poženete z uporabo ukaza, kot je `i` (iz angl. »insert«, vstavi) iz ukaznega načina. Dokler ste v načinu za vstavljanje, lahko vstavljate besedilo v spis na trenutnem mestu kazalca. Za konec načina za vstavljanje in vrnitev v ukazni način, pritisnite `Esc`.

Način zadnje vrstice je poseben način, ki daje določene razširjene ukaze urejevalniku `vi`. Ko vpisujete te ukaze, se prikazujejo v zadnji vrstici zaslona (odtod ime). Na primer, ko vpisujete v ukaznem načinu »`:`«, skočite v način zadnje vrstice in lahko uporabljate ukaze, kot sta »`wq`« (iz angl. write & quit, za zapis datoteke in izhod iz `vi`), ali »`q!`« (za izhod iz `vi` brez shranjevanja sprememb). Način zadnje vrstice se v splošnem uporablja za ukaze v `vi`, ki so daljši kot en znak. V načinu zadnje vrstice lahko vnesete enovrstični ukaz in pritisnete `Enter` za njegovo izvajanje.

3.13.2 Zagon `vi`

Najboljši način za spoznavanje teh pojmov je, da zaženete `vi` in pričnete urejati datoteko. Primeri »zaslonov« spodaj kažejo le nekaj vrstic besedila, kot da bi bil zaslon visok le šest vrstic, namesto štiriindvajset.

Skladnja za klic `vi` je

```
vi datoteka
```

kjer je *datoteka* ime datoteke za ureditev.

Poženite `vi` z vnosom

```
/home/larry$ vi test
```

za urejanje datoteke `test`. Videti bi morali nekaj kot

```
~
~
~
~
~
~
"test" [New file]
```

Stolpec znakov »˜« nakazuje, da ste na koncu datoteke. Podčrtaj »_« predstavlja kazalec.

3.13.3 Vstavljanje besedila

Program `vi` je zdaj v ukaznem načinu. Vnesite besedilo v datoteko s pritiskom `i`, kar postavi urejevalnik v način za vstavljanje, in začnite tipkati. Morda boste trenutno imeli še probleme z vnosom šumnikov, odvisno od vaše distribucije in tega, kako ste nastavili tipkovnico. Če imate tovrstne težave, pišite »`CSZ`« namesto »ČŠŽ«. Ureditev vnosa slovenskih znakov je podrobno opisana v spisu Slovenian HOWTO, glejte dodatek A za informacije o tem, kje ga dobite.

```
Zdaj je čas za vse dobre može, da pridejo pomagat zabavi.
~
~
~
~
~
```

Vpišite toliko vrstic, kot želite (po vsaki pritisnite Enter). Napake lahko popravite z Backspace.

Za konec načina za vstavljanje in vrnitev v ukazni način pritisnite Esc.

V ukaznem načinu lahko uporabljate tipke s puščicami za premikanje po datoteki. (Če imate le eno vrstico besedila, bo poskus uporabe tipk s puščicama navzgor ali navzdol verjetno povzročil le, da vam bo vi zapiskal.)

Poleg ukaza i obstaja še precej drugih načinov za vstavljanje besedila. Ukaz a vstavi besedilo, začenši za trenutnim položajem kazalca, namesto na trenutnem položaju kazalca. Na primer, uporabite levo puščico za premik kazalca med besedi »dobre« in »može«.

```
Zdaj je čas za vse dobre_može, da pridejo pomagat zabavi.
~
~
~
~
~
```

Pritisnite a za začetek načina za vstavljanje, napišite »figa« in potem pritisnite Esc za vrnitev v ukazni način.

```
Zdaj je čas za vse dobre figamože, da pridejo pomagat zabavi.
~
~
~
~
~
```

Za začetek vstavljanja besedila v naslednjo vrstico uporabite ukaz o. Pritisnite o in vnesite še vrstico ali dve:

```
Zdaj je čas za vse dobre figamože, da pridejo pomagat zabavi.
Potem bomo šli ven na pizzo in pivo.
~
~
~
~
```

3.13.4 Brisanje besedila

Ukaz x v ukaznem načinu pobriše znak pod kazalcem. Če petkrat pritisnete x, boste končali z:

```
Zdaj je čas za vse dobre figamože, da pridejo pomagat zabavi.
Potem bomo šli ven na pizzo in
~
~
~
~
```

Zdaj pritisnite [a] in vstavite nekaj besedila, nato pa [Esc]:

```
Zdaj je čas za vse dobre figamože, da pridejo pomagat zabavi.
Potem bomo šli ven na pizzo in dietno kokto.
~
~
~
~
```

Celotne vrstice lahko pobrišete z ukazom dd (se pravi, dvakrat zaporedoma pritisnite [d]). Če je kazalec na drugi vrstici in vpišete dd, boste videli:

```
Zdaj je čas za vse dobre figamože, da pridejo pomagat zabavi.
~
~
~
~
~
```

Za izbris besede, na kateri je kazalec, uporabite ukaz dw. Postavite kazalec na besedo »dobre«, in vtipkajte dw.

```
Zdaj je čas za vse figamože, da pridejo pomagat zabavi.
~
~
~
~
~
```

3.13.5 Spreminjanje besedila

Razdelke besedila lahko nadomestite z uporabo ukaza R. Postavite kazalec na prvo črko v »zabavi«, pritisnite [R], in vpišite besedo »lačnim«.

```
Zdaj je čas za vse figamože, da pridejo pomagat lačnim.
~
~
~
~
~
```

Uporaba R za urejevanje besedila je podobna ukazoma i in a, toda R prepiše besedilo, namesto, da bi ga vstavljal.

Ukaz r nadomesti en sam znak pod kazalcem. Na primer, postavite kazalec na začetek besede »Zdaj«, in pritisnite r, ki mu sledi K. Videli boste:

```
Kdaj je čas za vse figamože, da pridejo pomagat lačnim.
~
~
~
~
~
```

Ukaz »~« spremeni velikost črk pod kazalcem iz velikih v male in obratno. Na primer, če postavite kazalec na »d« v »Kdaj«, zgoraj, in zaporedoma pritiskate [~], boste končali z:

```
KDAJ JE ČAS ZA VSE FIGAMOŽE, DA PRIDEJO POMAGAT LAČNIM.
~
~
~
~
~
```

Če se velikost slovenskih črk »čšž« pri tem ohrani, kot v zgornjem primeru, jih popravite na roke, denimo z ukazom `r`. V tem primeru boste najbrž želeli prikrojiti svoj sistem po navodilih iz spisa *Slovenian HOWTO* (glejte dodatek A) – ključna sta predvsem izvoz in nastavitev okoljske spremenljivke, ki določa uporabo slovenske abecede (`export LC_CTYPE=sl_SI` v lupini `bash`).

3.13.6 Ukazi za premikanje kazalca

Veste že, kako uporabljati tipke s puščicami za premikanje po spisu. Poleg tega lahko uporabljate ukaze `h`, `j`, `k` in `l` za premikanje kazalca levo, dol, gor in desno, po vrsti. To je pripravno, če (zaradi nekega vzroka) vaše tipke s puščicami ne delujejo pravilno.

Ukaz `w` premakne kazalec na začetek naslednje besede; ukaz `b` ga premakne na začetek prejšnje besede.

Ukaz `0` (to je tipka z ničlo) premakne kazalec na začetek trenutne vrstice, in ukaz `$` ga premakne na konec vrstice.

Ko urejate velike datoteke, se boste želeli pomikati po datoteki po en zaslon hkrati navzgor ali navzdol. Pritisk `Ctrl-F` premakne kazalec en zaslon naprej, `Ctrl-B` ga premakne zaslon nazaj.

Za premik kazalca na konec datoteke pritisnite `G`. Lahko ga tudi premaknete na poljubno vrstico; na primer, vnos ukaza `10G` bi premaknil kazalec na vrstico 10 v datoteki. Za premik na začetek datoteke uporabite `1G`.

Premikalne ukaze lahko združujete z drugimi ukazi, kot so tisti za brisanje besedila. Na primer, ukaz `d$` pobriše vse od kazalca do konca vrstice, `dG` pobriše vse od kazalca do konca datoteke in tako najprej.

3.13.7 Shranjevanje datotek in zapuščanje `vi`

Urejevalnik `vi` zapustite, ne da bi naredili spremembe v datoteki, z uporabo ukaza `:q!`. Ko pritisnete »`:`«, se kazalec premakne v zadnjo vrstico zaslona in boste v načinu zadnje vrstice.

```
KDAJ JE ČAS ZA VSE FIGAMOŽE, DA PRIDEJO POMAGAT LAČNIM.
~
~
~
~
~
:
```

V načinu zadnje vrstice so dostopni nekateri razširjeni ukazi. Eden od njih je `q!`, ki zapusti `vi` brez shranjevanja. Ukaz `:wq` shrani datoteko in potem zapusti `vi`. Ukaz `ZZ` (iz ukaznega načina, brez »`:`«) je ekvivalenten ukazu `:wq`. Če datoteka ni bila spremenjena od zadnjega shranjevanja, le zapusti urejevalnik in ohrani čas spremembe datoteke. Ne pozabite, da morate po ukazu, vnešenem v načinu zadnje vrstice, pritisniti `Enter`.

Za shranitev datoteke brez zapuščanja `vi` uporabite `:w`.

3.13.8 Urejanje druge datoteke

Za urejanje druge datoteke uporabite ukaz `:e`. Na primer, za prenehanje urejevanja datoteke `test` in urejevanje datoteke `foo` uporabite ukaz

```
KDAJ JE ČAS ZA VSE FIGAMOŽE, DA PRIDEJO POMAGAT LAČNIM.
~
~
~
~
~
:e foo_
```

Če uporabite `:e`, ne da bi najprej shranili datoteko, boste dobili sporočilo o tovrstni napaki

```
No write since last change (":edit!" overrides)
```

kar pomeni, da `vi` noče urejevati nove datoteke, dokler ne shranite prvotne. Na tej točki lahko uporabite `:w` za shranitev prvotne datoteke in potem uporabite `:e`, ali pa lahko uporabite ukaz

```
KDAJ JE ČAS ZA VSE FIGAMOŽE, DA PRIDEJO POMAGAT LAČNIM.
~
~
~
~
~
:e!  foo_
```

Klicaj »!« pove urejevalniku `vi`, da to res mislite – urejevanje nove datoteke, ne da bi shranili spremembe v prvotno.

3.13.9 Vključevanje drugih datotek

Če uporabite ukaz `:r`, lahko v trenutno datoteko vključite vsebino druge datoteke. Na primer, ukaz

```
:r foo.txt
```

vstavi vsebino datoteke `foo.txt` v besedilo na mestu kazalca.

3.13.10 Poganjanje lupinskih ukazov

V urejevalniku `vi` lahko poganjate tudi ukaze ukazne lupine. Ukaz `:r!` deluje kot `:r`, a namesto da bi prebral datoteko, vstavi izhod danega ukaza v vmesni pomnilnik na trenutnem mestu kazalca. Na primer, če uporabite ukaz

```
:r!  ls -F
```

boste končali z

```
KDAJ JE ČAS ZA VSE FIGAMOŽE, DA PRIDEJO POMAGAT LAČNIM.
pisma/
razno/
spisi/
~
~
```

Lahko se tudi »preselite« v ukazno lupino iz urejevalnika `vi`, z drugimi besedami, poženete ukaz iz njega in se vrnete v urejevalnik, ko ste končali. Na primer, če uporabite ukaz

```
:! ls -F
```

se bo ukaz `ls -F` pognal in bo rezultat prikazan na zaslonu, a ne bo vstavljen v datoteko, ki jo urejate. Če uporabite ukaz

```
:shell
```

bo `vi` zagnal izvod ukazne lupine in vam omogočil, da začasno postavite `vi` »na čakanje«, dokler izvajate druge ukaze. Za vrnitev v `vi` se le odjavite iz ukazne lupine (z uporabo ukaza `exit`).

3.13.11 Iskanje pomoči o `vi`

Urejevalnik `vi` ne ponuja dosti na področju interaktivne pomoči (večina programov za Linux ne), a vedno lahko preberete stran priročnika za `vi` (z ukazom `man vi`). Urejevalnik `vi` je vizualni vmesnik za urejevalnik `ex`; ki ureja mnoge ukaze načina zadnje vrstice v `vi`. Zato poleg branja strani priročnika o `vi` poglejte še `ex`.

3.14 Prilagoditev vašega okolja

Ukazna lupina ponuja veliko mehanizmov za prilagoditev vašega delovnega okolja. Kot smo omenili zgoraj, je ukazna lupina več kot le tolmač posamičnih ukazov – je tudi tolmač za zmogljiv programski jezik. Čeprav je pisanje skriptov ukazne lupine široko področje, bi vas radi seznanili z nekaterimi načini za poenostavitev vašega dela na sistemu Linux brez uporabe teh naprednih odlik ukazne lupine.

Kot je bilo že omenjeno, različne ukazne lupine uporabljajo različno skladnjo, ko izvajajo skripte ukazne lupine. Na primer, Tcsh uporablja C-ju podobno skladnjo, medtem, ko Bournove lupine uporabljajo drugo vrsto skladnje. V tem razdelku ne bomo opazili velikih razlik med tema dvema, a bomo privzeli, da se skripti ukazne lupine izvajajo z uporabo skladnje Bournove lupine.

3.14.1 Skripti ukazne lupine

Denimo, da pogosto uporabljate vrsto ukazov in bi radi prihranili čas tako, da jih uvrstite skupaj v en sam »ukaz«. Na primer, trije ukazi

```
/home/larry$ cat poglavje1 poglavje2 poglavje3 > knjiga
/home/larry$ wc -l knjiga
/home/larry$ lp knjiga
```

združijo datoteke `poglavje1`, `poglavje2`, in `poglavje3` in postavijo rezultat v datoteko `knjiga`. Drugi ukaz prikaže prešteto število vrstic v datoteki `knjiga`, tretji ukaz `lp knjiga` pa natisne datoteko `knjiga`.

Namesto da pišete vse te ukaze, jih lahko poberete skupaj v **lupinski skript**. Lupinski skript, uporabljan za tek vseh teh ukazov, bi se lahko glasil takole:

```
#!/bin/sh
# Lupinski skript za ustvarjanje in natis knjige
cat poglavje1 poglavje2 poglavje3 > knjiga
wc -l knjiga
lp knjiga
```

Lupinski skripti so le preproste tekstovne datoteke; ustvarite jih lahko s katerimkoli urejevalnikom, kot je `emacs` ali `vi`, ki je opisan od strani 135 naprej.

Poglejmo ta lupinski skript. Prva vrstica, »`#!/bin/sh`«, identificira datoteko kot lupinski skript in naroči ukazni lupini, kako naj izvede skript. Naroči ukazni lupini, naj prepusti skript v izvajanje `/bin/sh`, kjer je `/bin/sh` sam program za ukazno lupino. Zakaj je to pomembno? Na večini sistemih Linux je `/bin/sh` ukazna lupina Bournovega tipa, kot je `bash`. Prisila lupinskega skripta, da teče z uporabo `/bin/sh`, zagotavlja, da se bo pognal pod ukazno lupino Bournove skladnje (namesto pod lupino C). To bo povzročilo, da bo vaš skript tekel z uporabo Bournove skladnje, tudi če uporabljate `tcsh` (ali drugo podobno ukazno lupino) kot vašo prijavno lupino.

Druga vrstica je **komentar**. Komentar se začne z znakom »#« in se nadaljuje do konca vrstice. Ukazna lupina ignorira komentarje – pogosto se jih uporablja, da predstavijo lupinski skript programerju in ga naredijo lažje umljivega.

Ostanek vrstic v skriptu so preprosto ukazi, kot bi jih vpisali neposredno v ukazno lupino. Ukazna lupina tako prebere vsako vrstico skripta in požene to vrstico, kot da bi jo vpisali v pozornik ukazne lupine.

Za lupinske skripte so pomembna dovoljenja. Če ustvarite skript ukazne lupine, poskrbite, da imate dovoljenja za izvajanje skripta, če bi ga radi poganjali. Ko ustvarjate tekstovne datoteke, privzeta dovoljenja navadno ne vključujejo dovoljenja za izvajanje, in tega morate navesti eksplicitno. Glejte razpravo o dovoljenjih za dostop do datotek na strani 126 za podrobnosti. Na kratko, če bi bil zgornji skript shranjen v datoteki, imenovani `narediknjigo`, bi lahko uporabili ukaz

```
/home/larry$ chmod u+x narediknjigo
```

da bi si dali dovoljenje za izvajanje lupinskega skripta `narediknjigo`.

Za poganjanje vseh ukazov v skriptu lahko tedaj uporabite ukaz

```
/home/larry$ ./narediknjigo
```

3.14.2 Spremenljivke ukazne lupine in okolje

Ukazna lupina vam dovoljuje definiranje **spremenljivk** kot večina programskih jezikov. Spremenljivka je le kos podatkov, ki mu je prirejeno ime.

◇ Ukazna lupina `tcsh`, kot tudi druge ukazne lupine tipa C, uporabljajo drugačen mehanizem za nastavitev spremenljivk od tukaj opisanega. Ta razprava predpostavi uporabo Bournove ukazne lupine, kot je `bash`. Glejte stran priročnika o `tcsh` za podrobnosti.

Ko spremenljivki določite vrednost (z uporabo operatorja »=«), lahko dostopate do spremenljivke s predpono »$« njenemu imenu, kot je pokazano spodaj.

```
/home/larry$ foo="zdravo tam"
```

Spremenljivki `foo` je dana vrednost »`zdravo tam`«. Na to vrednost se lahko potem sklicujete z imenom spremenljivke, ki mu spredaj dodate znak »$«. Na primer, ukaz

```
/home/larry$ echo $foo
zdravo tam
/home/larry$
```

doseže enak rezultat, kot

```
/home/larry$ echo "zdravo tam"
zdravo tam
/home/larry$
```

Te spremenljivke so notranje ukazni lupini, kar pomeni, da do njih lahko dostopa le lupina. To je lahko uporabno v lupinskih skriptih; če si morate, na primer, zabeležiti ime datoteke, ga lahko shranite v spremenljivko kot zgoraj. Uporaba ukaza `set` prikaže seznam vseh definiranih lupinskih spremenljivk.

Ukazna lupina pa vam dovoljuje **izvoz** spremenljivk v **okolje**. Okolje (angl. environment) je nabor spremenljivk, ki so dostopne vsem ukazom, ki jih izvajate. Ko enkrat definirate spremenljivko znotraj ukazne lupine, njen izvoz naredi, da postane tudi ta spremenljivka del okolja. Za izvoz spremenljivke v okolje uporabite ukaz `export`.

◇ Spet, tukaj razlikujemo med `bash` in `tcsh`. Če uporabljate `tcsh`, se za nastavitev okoljskih spremenljivk uporablja druga skladnja (uporablja se ukaz `setenv`). Glejte stran priročnika o ukazni lupini `tcsh` za več informacij.

Okolje je za sistem Unix zelo pomembno. Dovoljuje vam nastavitev nekaterih ukazov le z nastavitvijo spremenljivk, ki jih ti ukazi poznajo.

Tukaj je kratek primer. Okoljska spremenljivka `PAGER` se uporablja v ukazu `man` in določa ukaz, ki naj se uporablja za prikaz strani referenčnega priročnika, po eno stran naenkrat. Če nastavite spremenljivko `PAGER` na ime ukaza, uporablja namesto (privzetega) ukaza `more` novi ukaz za prikaz strani priročnika.

Nastavite `PAGER` na »`cat`«. To povzroča, da se izhod priročnika `man` prikaže naenkrat na zaslonu, brez premora med stranmi.

```
/home/larry$ PAGER=cat
```

Zdaj izvozimo `PAGER` v okolje.

```
/home/larry$ export PAGER
```

Poskusite ukaz `man ls`. Stran priročnika bi morala zleteti mimo vašega zaslona, ne da bi čakala na vas.

Če zdaj nastavimo `PAGER` na »`more`«, se bo za prikaz strani priročnika uporabljal ukaz `more`.

```
/home/larry$ PAGER=more
```

Upoštevajte, da ni več treba uporabiti ukaza `export`, ko smo spremenili vrednost spremenljivke `PAGER`. Spremenljivko moramo izvoziti le enkrat; od takrat naprej se bodo vse njene spremembe samodejno prenesle v okolje.

Pogosto je nize nujno navesti med narekovaji, ali naj se prepreči ukazni lupini, da bi upoštevala njihove različne znake kot posebne. Na primer, niz boste morali zapisati med

narekovajema, če želite lupini preprečiti tolmačenje posebnega pomena znakov, kot so »*«, »?« ali presledek. Obstaja veliko drugih znakov, ki jih je treba zaščititi pred tolmačenjem. Podrobna razlaga in opis citiranja je opisana v knjigi *Bourne Shell Tutorial* založbe SSC.

Strani priročnika za določen ukaz vam povedo, če ukaz uporablja okoljske spremenljivke. Na primer, stran priročnika o ukazu `man` razlaga, da se za določitev izpisovalnika uporablja spremenljivka `PAGER`.

Nekateri ukazi delijo okoljske spremenljivke. Na primer, mnogi ukazi uporabljajo okoljsko spremenljivko `EDITOR` za določitev privzetega urejevalnika za uporabo, ko je ta potrebna.

Okolje se uporablja tudi za zapis sledi pomembnih informacij o vaših prijavnih sejah. Primer je okoljska spremenljivka `HOME`, ki vsebuje ime vašega domačega imenika.

```
/home/larry/papers$ echo $HOME
/home/larry
```

Druga zanimiva okoljska spremenljivka je `PS1`, ki definira glavni pozornik ukazne lupine. Na primer,

```
/home/larry$ PS1="Vaš ukaz, prosim:  "
Vaš ukaz, prosim:
```

Za nastavitev pozornika na prvotno vrednost (ki vsebuje trenutni delovni imenik, ki mu sledi znak »$«), uporabite

```
Vaš ukaz, prosim:  PS1="\w\$ "
/home/larry$
```

Stran priročnika o `bash` opisuje skladnjo, potrebno za nastavitev pozornika.

Okoljska spremenljivka `PATH` Ko uporabite ukaz `ls`, kako ukazna lupina najde samo izvajalno datoteko programa `ls`? Pravzaprav `ls` leži v imeniku `/bin` na večini sistemov. Ukazna lupina uporablja okoljsko spremenljivko `PATH` za določitev izvajalnih datotek za ukaze, ki jih vpisujete.

Na primer, vaša spremenljivka `PATH` je lahko nastavljena na

```
/bin:/usr/bin:/usr/local/bin:.
```

To je seznam imenikov, ki naj jih ukazna lupina poišče. Imeniki so ločeni z »:«. Ko uporabljate ukaz `ls`, ukazna lupina najprej pogleda za `/bin/ls`, potem `/usr/bin/ls`, in tako naprej.

Upoštevajte, da `PATH` nima ničesar opraviti z iskanjem običajnih datotek. Na primer, če uporabite ukaz

```
/home/larry$ cp foo bar
```

ukazna lupine ne uporablja `PATH` za iskanje datotek `foo` in `bar` – za ti imeni datotek se predpostavlja, da sta popolni. Ukazna lupina uporablja `PATH` samo, da najde izvedljivo datoteko `cp`.

To vam prihrani čas, in pomeni, da si vam ni treba zapomniti, kje se shranjujejo vsi izvajalni ukazi. Na veliko sistemih so izvedljive datoteke razmetane po veliko mestih, kot so imeniki `/usr/bin`, `/bin` ali `/usr/local/bin`. Namesto da podate polno ime poti ukaza

(kot je `/usr/bin/cp`), lahko nastavite spremenljivko `PATH` na seznam imenikov, za katere želite, da jih ukazna lupina samodejno preišče.

Upoštevajte, da `PATH` vsebuje »`.`«, kar je trenutni delovni imenik. To vam omogoča ustvarjanje lupinskega skripta ali programa in njegovo poganjanje iz trenutnega imenika, ne da bi ga bilo treba določiti neposredno (kot `./narediknjigo`). Če imenik ni v vaši poti `PATH`, potem ga ukazna lupina ne bo preiskovala za ukaze, ki se lahko izvedejo; to vključuje tudi trenutni imenik.

3.14.3 Inicializacijski skripti ukazne lupine

Poleg skriptov ukazne lupine, ki jih naredite sami, obstajajo še številni skripti, ki jih za določene namene uporablja sama ukazna lupina. Najpomembnejši od teh so **inicializacijski skripti**, ki so skripti, ki jih ukazna lupina izvede, ko se prijavite.

Inicializacijski skripti sami so skripti ukazne lupine, vendar inicializirajo vaše okolje s samodejno izvedbo ukazov ob vaši prijavi. Če vedno uporabljate ukaz `mail` za preverbo vaše pošte, ko se prijavite v sistem, lahko ta ukaz položite v inicializacijski skript, da se bo izvedel samodejno.

Obe ukazni lupini, `bash` in `tcsh`, ločita med **prijavno ukazno lupino** (angl. login shell) in drugimi priklici ukazne lupine. Prijavna lupina je lupina, ki se izvede, ko se prijavite v sistem. Navadno je to edina ukazna lupina, ki jo uporabljate. Vendar če se »spravite v lupino« drugega programa, kot je `vi`, zaženete drugi primerek ukazne lupine, ki ni vaša prijavna ukazna lupina. Poleg tega, če zaženete lupinski skript, samodejno začnete novo pojavitev lupine, ki izvaja skript.

Inicializacijske datoteke, ki jih uporablja `bash`, so: `/etc/profile` (nastavi jo sistemski upravitelj in se izvede pri vseh uporabnikih lupine `bash` v času prijave), `$HOME/.bash_profile` (izvede jo prijavna seja `bash`-a), in `$HOME/.bashrc` (izvedejo jo neprijavne pojavitve `bash`-a). Če datoteka `.bash_profile` ni prisotna, se namesto nje uporablja `.profile`.

`tcsh` uporablja naslednje inicializacijske skripte: `/etc/csh.login` (ob prijavi se izvede vsem uporabnikom lupine `tcsh`), `$HOME/.tcshrc` (izvede se ob prijavi in ob novih pojavitvah `tcsh`-ja), in `$HOME/.login` (izvede se v času prijave, takoj za `.tcshrc`). Če `.tcshrc` ne obstaja, se uporablja `.cshrc`.

Popoln vodnik po programiranju ukaznih lupin bi presegal obseg te knjige. Glejte strani priročnika o ukazni lupini `bash` ali `tcsh` za nadaljnje učenje o prilagoditvi vašega okolja Linux.

3.15 Torej želite poskusiti po svoje?

To poglavje bi vam moralo dati dovolj informacij za osnovno uporabo Linuxa. Strani referenčnega priročnika so nepogrešljivo orodje za dodatno učenje o Linuxu. Sprva se lahko zdijo zmedene, a če se prekopljete pod površino, predstavljajo pravi zaklad informacij.

Predlagamo tudi, da najprej preberete dobro splošno referenčno knjigo o Linuxu. Linux ima več odlik, kot se sprva zdi nevajenemu očesu. Žal je večina od njih izven dosega te knjige. Druge priporočene knjige o Linuxu so naštete v dodatku A.