1.

Uvod

Sadržaj

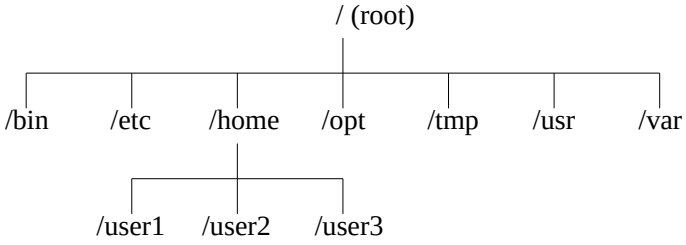
UNIX je višekorisnički, višezadaćni operacijski sustav.

1.1 Sustav datoteka

Datotečni sustav predstavlja skup pravila za organizaciju i upravljanje datotekama i direktorijima pohranjenim na čvrstom disku ili drugim oblicima trajne memorije. Unix datotečni susav organiziran je u obliku stabla (slika 1.1, u čijem se ishodištu nalazi korjenski (**root**) direktorij. Iz korjenskog direktorija, stablo se grana na datoteke i direktorije, koji u sebi sadrže druge datoteke i

direktorije. Broj grana i konfguracija stabla su proizvoljni, iako većina unix sustava dijeli sličnu organizacijsku strukturu datotečnog sustava.

Osnovne razlike u odnosu na *Windows* datotečni sustav:



Slika 1.1: UNIX datotečni sustav

- Unix svoj datotečni sustav ne dijeli prema fizičkoj (ili logičkoj) podjeli (dakle ovdje ne postoje oznake a: za disketnu jedinicu ili c: d: za particije), nego sve direktorije dodaje na osnovni s oznakom / kojeg nazivamo korjenski ili **root** direktorij. Različite fizičke ili logičke jedinice za pohranu podataka mogu se nalaziti uključne u bilo kojem dijelu datotečnog stabla, ne nužno na prvoj razini dijeljenja stabla. Ovo je za korisnika transparentno, tj. ne vidi se razlika kada se sa jednog uređaja ili particije prijeđe na neku drugu.

- Unix datoteke nemaju ekstenziju. Ovo ne znači da datoteka u svom imenu ne može imati dio (nastavak) koji označava njezinu primarnu namjenu, ali ovaj nastavak je za operativni sustav sastavni dio imena i ne određuje način pristupa datoteci. Naprimjer, ukoliko se datoteka zove `dat1.txt`, kratica `txt` je sastavni dio imena. Ova datoteka može biti tekstualna, ali isto tako i izvršna, ili čak direktorij. Način pristupa datoteci određuje se preko sustava prava pristupa *(permissions)*. Posljedica ovoga je i da oznaka `*.*` koja bi u windowsima označavala sve datoteke u ovom slučaju označava sve datoteke koje u imenu imaju točku.

1.1.1 Tipovi datoteka

- **Obične datoteke**
  Koriste se za spremanje informacija raznog tipa (tekst, slike, programi, itd.)

- **Direktoriji**
  Točke grananja datotečnog stabla, sadrže druge datoteke (uključujući druge direktorije)

- **Simbolički linkovi**
  Simbolički link je datoteka koja pokazuje na neku drugu datoteku

- **Specijalne datoteke**
  Točke u datotečnom sustavu koje predstavljaju hardwerske uređaje (nazivaju se i *device files*). Omogućavaju korisničkim programima da sa device driverom komuniciraju putem standardnog ulazno/izlaznog sučelja. Razlikujemo tzv. *character special files* i *block special files*.

- **Sockets**
  Koriste se za komunikaciju među procesima, bilo lokalno ili putem mreže.

- **FIFOs**
  Slično kao socket, služi za lokalnu komunikaciju među procesima

1.1.2 Struktura direktorija

- Svaki direktorij može sadržavati datoteke i druge direktorije (hijerarhijska struktura), tj. datoteke tipa direktorij.

- Svaki direktorij već u trenutku kreiranja sadrži dvije datoteke:

  – `.` - pokazivač na radni direktorij, tj. na direktorij u kojem se trenutno nalazimo

  – `..` - pokazivač na roditeljski direktorij, direktorij jednu raznu više u hijerarhijskoj strukturi datotečnog stabla.

  Ova dva unosa nije moguće izbrisati.

- Kod navigacije datotečnim sustavom koristimo se apsolutnim ili relativnim načinom zadavanja putanje. Apsolutna putanja ili `PATH` uvijek započinje korjenskim `(root)` direktorijem, npr:

  `/home/dkrst/nastava/unix`

  Kod zadavanja putanje relativnim načinom, trenutni radni direktorij koristi se kao početna točka u odnosu na koju zadajemo putanju, npr.
  `nastava/unix`

  ili (ekvivalentno)

```
./nastava/unix
```

1.1.3 Prava pristupa

Sustavom prava pristupa moguće je precizno definirati ovlasti svakog pojedinog korisnika na UNIX računalu. Pri definiranju ovlasti, prava se dijele na tri razine, tj. skupine korisnika na koja se dana prava odnose.

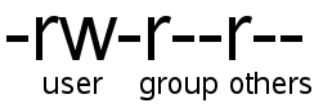
- **user** - vlasnik datoteke
  Svaka datoteka na UNIX sustavu ima točno jednog vlasnika. Obično se radi o korisniku koji je datoteku i stvorio, ali se vlasništvo može prenositi i na druge korisnike. Vlasnik u pravilu ima najveću razinu prava na datoteci, a može mijenjati prava pristupa za sve ostale korisnike.

- **group** - grupa vlasnika
  Pored vlasnika, svaka datoteka ima i grupu korisnika kojoj pripada. Grupu obično čine korisnici koji imaju neko zajedničko obilježje. Mijenjanjem ovlasti za grupu moguće je istovremeno namiještati razinu pristupa za više korisnika.

- **others** - ostali korisnici na sustavu
  Odnosi se na sve ostale korisnike sustava koji nisu u grupi koja ima vlasništvo nad datotekom, niti su vlasnik datoteke osobno. U pravilu najniža razina prava pristupa.

Prava pristupa definiraju se zasebno za korisnika koji je vlasnik datoteke, za grupu korisnika kojima datoteka pripada, i za sve ostale korisnike na sisemu. Mijenjanje prava pristupa dozvoljeno je samo vlasniku datoteke i *root* korisniku. Root korisnik, koji se još naziva i *superuser*, ima i pravo promjene vlasništva nad datotekom.

Prava pristupa za datoteke



čitanje	**r**
pisanje	**w**
izvršavanje	**x**

- Pravo čitanja datoteke omogućava korisnicima na koje se odnosi pregled sadržaja datoteke.

- Pravo pisanja omogućava korisnicima na koje se odnosi izmjenu sadržaja datoteke.

- Pravo izvršavanja omogućava korisnicima na koje se odnosi pokretanje izvršne datoteke. Za razliku od nekih drugih operativnih sistema, kod UNIX-a tip datoteke ne određuje ekstenzija, pa izvršnu datoteku nije moguće prepoznati po imenu. Datoteke izvršnog koda, bilo u binarnom *(strojni kod)*, ili u tekstualnom *(shell skripte)* obliku najčešće je moguće prepoznati po oznaci **x** u pravima pristupa, koja označava da se navedena datoteka može učitati u memoriju računala i izvršiti kao program. Nužan preduvjet za pokretanje datoteke izvršnog koda je i pravo čitanja datoteke (**r**) za korisnika koji program želi pokrenuti.

Prava pristupa za direktorije

Pravo čitanja (**r**), pisanja (**w**) i izvršavanja (**x**) imaju nešto izmjenjeno značenje kada se odnose na datoteke tipa *direktorij*. Ovo se najviše odnosi na pravo izvršavanja, pošto je jasno da direktorij, koji je skup drugih datoteka i direktorija, nije moguće učitati u memoriju računala i izvršiti kao program. Da bi jednostavnije pojasnili smisao prava pristupa za direktorije, potrebno je promisliti o načinu na koji operativni sustav upravlja sustavom direktorija.

Direktorij ne može biti fizički dio diska računala rezerviran za datoteke i direktorije koje se u njemu nalaze iz jednostavnog razloga što nije moguće unaprijed znati veličinu podataka koji će u taj direktorij biti pohranjeni. Sa svakom izmjenom bilo koje datoteke u direktoriju, brisanjem ili dodavanjem novih datoteka, količina podataka pohranjenih u direktoriju se mijenja, pa bi bilo potrebno promijeniti veličinu prostora na disku rezerviranu za promatrani direktorij.

Zbog opisanih razloga, svaki direktorij je u osnovi lista zapisa o datotekama koji se u njemu nalaze, pohranjena u formatu koji operativnom sustavu omogućava efikasno pretraživanje i pristup traženim podacima. Na ovako zamišljeni direktorij možemo primjeniti pravo čitanja, pisanja i izvršavanja na slijedeći način:

- Pravo čitanja omogućava otvaranje datoteke i pregled njenog sadržaja. Upravo ovo nam omogućava i pravo čitanja direktorija. Dakle, ukoliko na direktoriju ima pravo čitanja (**r**), korisnik može pregledati sadržaj direktorija, točnije saznati koje se datoteke u direktoriju nalaze. Iako pravo čitanja daje korisniku mogućnost gledanja sadržaja direktorija, ovo još ne znači da mu je dozvoljeno sadržaj i "izvršiti", tj. neku od datoteka iz direktorija i otvoriti i vidjeti njezin sadržaj.

- Kao i kod običnih datoteka, pravo pisanja omogućava korisniku izmjenu sadržaja direktorija. Ovo uključuje brisanje postojećih i dodavanje novih datoteka u direktoriju.

- Pravo izvršavanja za direktorije omogućava korisniku "izvršavanje" sadržaja direktorija, tj. otvaranja datoteke koja se nalazi u direktoriju. Naravno, da bi korisnik mogao pogledati sadržaj datoteke koja se nalazi u direktoriju, mora imati i pravo čitanja na traženu datoteku, koje je definirano za svaku datoteku zasebno.

Promjena prava pristupa

Prava pristupa za datoteku mogu se promijeniti naredbom `chmod`:

```
chmod [opcije] prava datoteka
```

pri čemi je prava moguće definirati apsolutno ili simbolički.

Apsolutna prava pristupa:

_	r	w	x	r	w	x	r	w	x
	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1
	4	2	1	4	2	1	4	2	1

Apsolutna prava pristupa računaju se pojedinačno za korisnika, grupu i ostale zbrajanjem tri binarne znamenke koje određuju prava.

Primjer:

`chmod 644 dat1.txt`
na datoteci `dat1.txt` postavlja slijedeća prava:

- korisnik dobija pravo čitanja i pisanja (`r w _`)

- grupa dobija pravo čitanja (`r _ _`)
- ostali dobivaju pravo čitanja (`r _ _`)

Simbolička prava pristupa:

Promjena prava se definira korištenjem simboličkih oznaka skupine korisnika na koje se prava odnose, znakom za davanje ili oduzimanje prava (`+/-`), te oznakama prava koja se mijenjaju (`r, w, x`).

Oznake korisnika na koje se odnosi promjena:

u	vlasnik datoteke (*user*)
g	grupa vlasnika (*group*)
o	ostali korisnici (*others*)
a	odnosi se na sve korisnike (*all*), zamjenjuje kombinaciju **ugo**

Primjeri:

`chmod u+x dat1.txt`
daje vlasniku pravo izvršavanja na datoteci `dat1.txt`

`chmod go+rx dat1.txt`
daje grupi i ostalim korisnicima pravo čitanja i izvršavanja na datoteci `dat1.txt`

`chmod a-x dat1.txt`
oduzima pravo izvršavanja svima (vlasniku, grupi i ostalim korisnicima).

Važno je uočiti razliku kod primjene simboličkog i apsolutnog načina zadavanja prava. Ukoliko prava zadajemo simbolički, točno je definirano koja se prava i za koju skupinu korisnika mijenjaju. Prava koja nisu navedena ostaju ista nakon izvršavanja naredbe. Tako, na primjer, ukoliko grupi vlasnika ukidamo pravo čitanja datoteke, preostala prava se neće promjeniti. Nadalje, ukoliko prethodno grupa vlasnika nije imala pravo čitanja, naredba neće imati nikakav efekt.
Nasuprot tome, ukoliko prava zadajemo apsolutnim načinom, uvijek je potrebno definirati sva prava (za vlasnika, grupu i ostale) koja želimo postaviti na ciljanoj datoteci. Bez obzira na prethodne postavke, nakon izvršavanja naredbe na datoteci će biti postavljena prava točno onako kako su u naredbi navedena.

1.2 Naredbena ljuska - SHELL

UNIX ljuska (**Shell**) je interpreter naredbi putem kojeg korisnik komunicira sa pokrenutim programima, odnosno s jezgrom operacijskog sustava. Potrebno je uočiti da ljuska nije dio jezgre operacijskog sustava *(kernela)*, nego korisnički program koji se koristi za interakciju sa sustavom. U osnovi, *shell* je prevoditelj naredbi[1] koje korisnik zadaje. Spremnost za prihvaćanje naredbi korisnika ljuska signalizira znakom za unos (uobičajeno `>, %, $`, može biti proizvoljna kombinacija znakova).
Nakon što korisnik unese naredbu, ljuska analizira unos i, ukoliko je to mogoće, prevodi ga u odgovarajuću akciju. Ovo najčešće podrazumjeva pokretanje korisničkog programa sa diska računala, nakon čega pokrenuti program preuzima ulaze i izlaze, tj. komunikaciju sa korisnikom putem terminala. Ipak, komunikacija sa ljuskom je i dalje omogućena putem kontrolnih signala. Korisnički program je iz ljuske moguće pokrenuti i *"u pozadini"*, na način da ljuska zadrži kontrolu nad ulazima i izlazima terminala na kojem korisnik radi.

Pokretanje programa u pozadini:
```
/> naredba &
```

Iako postoji više različitih ljuski, razlikujemo dva osnovna tipa: **sh** (Bourne Shell) i **csh** (C Shell). Svaki korisnik može promijeniti naredbenu ljusku korištenjem naredbe `chsh`

1.2.1 Osnovne naredbe

Zadavanjem naredbi iz ljuske možemo pokretati korisničke programe, ali i koristiti naredbe ljuske. Osnovna razlika leži u činjenici da svakom korisničkom programu uvijek odgovara izvršna datoteka, koja se pokretanjem učitava u memoriju i izvršava. Nasuprot tome, pokretanjem ugrađenih naredbi ljuske *(Shell Built-in Commands)* zapravo aktiviramo funkcije ljuske. S aspekta prosječnog korisnika ova razlika nema veliko značenje. Ipak, kod korištenja naredbi koje nisu dio ljuske, svejedno je iz koje je ljuske naredba pokrenuta. S druge strane, kada pozovemo neku od ugrađenih naredbi ljuske, izvršavaju se rutine koje su implementirane u ljusci, što znači da detalji procesa ovise o izboru ljuske. Druga posljedica ovog je da pokretanje programa koji nije dio ljuske znači i

[1] Command Line Interpreter - prevoditelj naredbenog retka. Naredbe koje korisnik unosi analiziraju se i na osnovi zadanih ulaza poduzimaju se odgovarajuće akcije. Ukoliko unos korisnika nije moguće prevesti, ispisuje se poruka o greški

pokretanje novog procesa.

`pwd` - ispis trenutnog radnog direktorija:

```
/> pwd
/home/dkrst/nastava/unix/work
```

`cd` - promjena tekućeg radnog direktorija. Putanju do novog radnog direktorija moguće je zadati na apsolutni ili relativni način:

```
/> cd /home/dkrst/nastava/unix/vjezbe
```

ili (uz isti efekt)

```
/> cd ../vjezbe
```

Naredba `cd` je ugrađena naredba ljuske (nema pripadajuću izvršnu datoteku, već se aktiviraju rutine koje su integralni dio ljuske).

`ls` - Pregled sadržaja direktorija:

```
/> ls
pr1.tgz primjeri1 Uvod.ppt
```

`ls -al` - Opcije: `'a'` prikaz svih datoteka, uključujući datoteke čije ime započinje sa točkom; `'l'` prikaz u obliku liste sa više detalja

```
/> ls -al
total 56
drwxr-xr-x   3 dkrst users   128 2006-03-11 11:24 .
drwxr-xr-x  11 dkrst users   352 2006-03-11 12:38 ..
-rwxr-xr-x   1 dkrst users  1120 2006-03-08 13:24 pr1.tgz
drwxr-xr-x   2 dkrst users   216 2006-03-11 11:24 primjeri1
-rwxr-xr-x   1 dkrst users 50176 2006-03-08 13:24 Uvod.ppt
```

`cat` - Ispis sadržaja jedne ili više datoteka. Lista datoteka čiji sadržaj želimo pregledati zadaje se u nizu iza naredbe `cat`. Datoteke će biti ispisani onim redosljedom kojim su zadane u naredbenom retku, na način da se završetkom jedne datoteke odmah nastavi sadržaj druge. Cjelokupan sadržaj ispisuje se odjednom, tj. ne postoji način postepenog pregleda ispisa.

`more` - Postupan pregled sadržaja datoteke stranicu po stranicu.

`less` - Slično kao i `more`. Uz postupan ispis sadržaja datoteke, korisnik ima mogućnost vraćanja unatrag kroz sadržaj datoteke

`mkdir` - Stvaranje direktorija.

```
/> mkdir ./test1
/> mkdir /home/dkrst/nastava/unix/work/test2
/> ls -al
total 4
drwxr-xr-x   3 dkrst users  96 2006-03-11 13:03 .
drwxr-xr-x  11 dkrst users 352 2006-03-11 12:38 ..
-rw-r--r--   1 dkrst users  33 2006-03-11 12:58 dat1.txt
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test1
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test2
```

`cp` - Kopiranje datoteka

```
/> cp dat1.txt dat2
/> ls -al
total 8
drwxr-xr-x   4 dkrst users 144 2006-03-11 16:12 .
drwxr-xr-x  11 dkrst users 352 2006-03-11 12:38 ..
-rw-r--r--   1 dkrst users  33 2006-03-11 12:58 dat1.txt
-rw-r--r--   1 dkrst users  33 2006-03-11 16:12 dat2
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test1
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test2
```

`mv` - Prmještanje ili preimenovanje datoteka

```
/> mv dat2 dat2.txt
/> ls -al
total 8
drwxr-xr-x   4 dkrst users 144 2006-03-11 16:14 .
drwxr-xr-x  11 dkrst users 352 2006-03-11 12:38 ..
-rw-r--r--   1 dkrst users  33 2006-03-11 12:58 dat1.txt
```

```
-rw-r--r--   1 dkrst users  33 2006-03-11 16:12 dat2.txt
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test1
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test2
```

`rm` - Brisanje datoteke (ili više njih)

`/> rm test1/*` - brisanje svih datoteka u direktoriju `test1`

`rmdir` - Brisanje direktorija

`/> rmdir test1` - brisanje direktorija `test1`

Direktorij je moguće obrisati samo ukoliko ne sadrži ni jednu datoteku

`echo` Ispis na standardni izlaz

```
/> echo Dobar dan
Dobar dan
```

Naredba `echo` je ugrađena naredba ljuske.

`eval` - Izvršavanje argumenta kao naredbe.

```
/> eval pwd
/home/dkrst/nastava/unix/work
```

Često se koristi u kombinaciji sa varijablama kod izrade *shell skripti*:

```
/> set a = pwd
/> eval $a
/home/dkrst/nastava/unix/work
```

Naredba `eval` je ugrađena naredba ljuske.

`exec` - pokretanje naredbe (zamijeni ljusku sa naredbom). Za razliku od naredbe `eval` koja izvršava naredbu, nakon koje ljuska nastvalja sa normalnim radom, ni jedna naredba nakon `exec` neće biti izvršena, ukoliko ne dođe do greške prilikom izvođenja naredbe `exec`

Naredba `exec` je ugrađena naredba ljuske.

`jobs` - Prikazuje listu svih zaustavljenih i pozadinskih procesa.

`bg` - Pokretanje procesa u pozadini (*background*)

`fg` - Vraća proces iz pozadine

`man` - Pomoć *(manual pager)* za bilo koju UNIX naredbu:

```
/> man ls
```

Pomoć za naredbu `ls`

Naredba `man` jedna je od najkorisnijih UNIX naredbi. Korištenjem ove naredbe moguće je dobiti pomoć i cjelokupnu listu opcija za bilo koju drugu naredbu.

1.3 Ulaz i izlaz naredbi

Interakcija UNIX programa i naredbi pokrenutih iz ljuske, i korisnika sustava omogućena je putem tri datoteke koje su otvorene u trenutku pokretanja programa. Svaki pokrenuti program ima:

- **stdin (0)** - standardni ulaz *(standard input)*
  Na ovom *file descriptoru* [2] otvorena je datoteka putem koje korisnik može upravljati pokrenutim programom. Datoteka je otvorena za čitanja, tj. pokrenuti program iz ove datoteke čita znakove.
- **stdout (1)** - standardni izlaz za rezultate *(standard output)*
  Ova datoteka otvorena je za pisanje, a služi za ispis rezultata obrade UNIX programa.
- **stderr (2)** - standardni izlaz za greške *(standard error)*
  Datoteka otvorena za pisanje, služi za ispis obavijesti o greškama koje se mogu pojaviti za vrijeme izvršavanja programa.

Kod pokretanja programa u uobičajenom slučaju, sve tri spomenute datoteke povezane su na korisničku konzolu, tj. na terminal iz kojeg je pokrenut program. Znakovi koje korisnik unosi sa tipkovnice pojavljuju se na `stdin` file

[2] UNIX program svaku otvorenu datoteku referencira putem file descriptora. File descriptor je cijeli nenegativni broj, a novo otvorenoj datoteci pridjeljuje se najniži slobodni file descriptor.

descriptoru, onim redom kojim ih korisnik unosi. Poruke koje program ispisuje u `stdout` i `stderr` pojavljuju se u istom treminalu, onim redom kojim su poslane od strane programa.
Bilo koju od tri navedene datoteke moguće je zatvoriti i na slobodnom file descriptoru otvoriti drugu datoteku. Na ovaj način moguće je preusmjeriti standardni ulaz i izlaze programa u neku od datoteka na disku računala, ali i na druge tipove datoteka, npr. otvoreni *socket* na mreži. Pokrenuti program pristupa otvorenoj datoteci na isti način, bez obzira radi li se o uobičajenom ulazu ili izlazu u terminalu iz kojeg je program pokrenut, ili nekom drugom tipu datoteke.

1.3.1 Preusmjeravanje ulaza i izlaza naredbi

`>`	standardni izlaz naredbe upiši u stvorenu datoteku
`>>`	standardni izlaz naredbe dodaj na kraj postojeće datoteke
`>&`	standardni izlaz za greške upiši u stvorenu datoteku
`>>&`	standardni izlaz za greške dodaj na kraj postojeće datoteke
`<`	standardni ulaz naredbe čitaj iz datoteke

```
/> ls -al > out.txt
/> cat out.txt
total 8
drwxr-xr-x   4 dkrst users 168 2006-03-11 18:04 .
drwxr-xr-x  11 dkrst users 352 2006-03-11 12:38 ..
-rw-r--r--   1 dkrst users  33 2006-03-11 12:58 dat1.txt
-rw-r--r--   1 dkrst users  33 2006-03-11 16:12 dat2.txt
-rw-r--r--   1 dkrst users   0 2006-03-11 18:04 out.txt
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test1
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test2
```

Izlaz naredbe `ls` preusmjeren je u datoteku `out.txt`. Naredbom cat out.txt gledamo sadržaj stvorene datoteke.

```
/> cp dat1.txt / >>& out.txt
/> cat out.txt
total 8
```

```
drwxr-xr-x   4 dkrst users 168 2006-03-11 18:27 .
drwxr-xr-x  11 dkrst users 352 2006-03-11 12:38 ..
-rw-r--r--   1 dkrst users  33 2006-03-11 12:58 dat1.txt
-rw-r--r--   1 dkrst users  33 2006-03-11 16:12 dat2.txt
-rw-r--r--   1 dkrst users   0 2006-03-11 18:27 out.txt
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test1
drwxr-xr-x   2 dkrst users  48 2006-03-11 13:05 test2
cp: cannot create regular file ‘/dat1.txt’: Permission denied
```

Standardni izlaz za greške naredbe `cp dat1.txt /` preusmjeren je na kraj postojeće datoteke`out.txt`. Izvršavanje naredbe rezultira greškom pošto običan korisnik nema pravo pisanja u root (oznaka `/`) direktorij. Obavijest o grešci dodana je na postojeći sadržaj datoteke (posljednja linija ispisa naredbe `cat out.txt`).

1.3.2 Ulančavanje naredbi

Pored preusmjeravanja ulaza i izlaza naredbi na datoteku, moguće je izlaz jedne naredbe povezati na standardni ulaz druge. Na ovaj način, rezultat obrade jedne naredbe pojavljuje se kao ulaz iduće.

`|`	poveži `stdout` prethodne naredbe na`stdin` slijedeće
`|&`	poveži `stdout` i `stderr` prethodne naredbe na `stdin` slijedeće

```
/> cat /etc/passwd | sort
aalagic:x:23722:111:Alen Alagic,student:/home/aalagic:/bin/csh
aalilovi:x:24575:111:Andjelko Alilovic,student:/home/aalilovi:/bin/csh
aaljinov:x:23476:111:Ante Aljinovic,student:/home/aaljinov:/bin/csh
aancic:x:24659:111:Ante Ancic,student:/home/aancic:/bin/csh
aarapovi:x:24397:111:Ante Arapovic,student:/home/aarapovi:/bin/csh
aarmanda:x:22378:111:Ante Armanda,student:/home/aarmanda:/bin/csh
ababin:x:24108:111:Andrej Babin,student:/home/ababin:/bin/csh
abalic:x:21059:111:Ante Balic,student:/home/abalic:/bin/csh
abandur:x:23723:111:Ana Bandur,student:/home/abandur:/bin/csh
abatinic:x:21108:111:Ante Batinica,student:/home/abatinic:/bin/csh
abelak:x:24394:111:Ante Belak,student:/home/abelak:/bin/csh
...
```

Standardni izlaz naredbe `cat /etc/passwd` preusmjeren je na ulaz naredbe `sort` (slaganje po abecedi). Krajnji rezultat je abecedni popis korisnika na

sustavu.

1.4 UNIX procesi

U trenutku kada se program sa diska učita u memoriju računala, nastaje proces koji odgovara pokrenutom programu[3]. Pokrenutom procesu pridjeljuju se resursi, kreira se radno okruženje (*Environment*) procesa, definiraju se ovlasti i prava pristupa za proces, i prodjeljuju mu se odgovarajuće identifikacijske oznake. Proces je način na koji UNIX sustav upravlja pokrenutim programima, a čine ga skup resursa i parametri procesa:

- **PID** - *Process ID*
  Jedinstveni identifikacijski broj procesa. Ovo je osnovni parametar po kojem operativni sistem identificira svaki pojedini proces.
- **PPID** - *Parrent Process ID*
  Proces ID *"roditeljskog"* procesa, tj. onog procesa iz kojeg je proces pokrenut. Ukoliko roditeljski proces završi prije pokrenutog procesa, proces nasljeđuje **init** (proces ID 1), tj. *PPID* procesa postaje 1.
- **UID** - *User ID*
  Identifikacijski broj korisnika koji je vlasnik procesa. Kao i kod datoteka, svaki proces ima svog vlasnika. Na temelju prava vlasnika procesa određuju se prava i ovlasti procesa u memoriji. Na ovaj način osigurano je sigurno i efikasno istovremeno izvršavanje više procesa više različitih korisnika sa precizno definiranim ovlastima svakog procesa u memoriji računala.
- **GID** - *Group ID*
  ID grupe vlasnika procesa.
- **EUID, EGID** - *Effective UID, Effective GID*
  Pored stvarnog vlasnika i grupe, proces može imati i efektivnog vlasnika i grupu kojoj pripada. Koristi se za dodatno definiranje ovlasti.

[3]Novi proces može nastati samo iz već postojećeg. U principu, svaki proces pokreće se slijedećom sekvencom naredbi: naredbom `fork` kopira se postojeći proces u memoriji računala, nakon čega se memorijska slika kopiranog procesa zamjenjuje programom sa diska korištenjem naredbe `exec`

1.5 Shell skripte

Pored interaktivnog načina rada, ljuskom je moguće upravljati korištenjem skripti. Niz naredbi zapisan u tekstualnoj datoteci koje se mogu izvršiti u zadanoj ljusci na ovaj način postaju izvršna datoteka, odnosno program koji možemo pokrenuti.

Prilikom izrade *Shell* skripti potrebno je voditi računa o tipu ljuske za koji je skripta namijenjena. Iako je način pozivanja programa u osnovi jednak za sve ljuske, detalji kao što su definiranje varijabli, zadavanje i provjeravanje uvjeta i interne naredbe ljuske razlikuju se za razne tipove ljuski. Ljusku za koju je namijenjena skripta zadajemo u prvom retku same skripte:

`#!/bin/sh`	Skripta pisana za *Bourne Shell*
`#!/usr/bin/csh`	Skripta pisana za *C Shell*

Retkom `#!/bin/sh` zadan je interpreter naredbi (ljuska) u kojoj će se izvršavati naredbe koje slijede. Iako se ova sintaksa najčešće koristi za zadavanje ljuske u kojoj će se interpretirati naredbe, ova sintaksa se može koristiti za kreiranje skripti za bilo koji interaktivni korisnički program. Na primjer, stvaranjem tekstualne datoteke koja započinje sa `#!/usr/bin/octave` moguće je napraviti skriptu za programski paket **Octave**. Prilikom izvođenja skripte, interpretira se naredba po naredba. Da bi napisana skripta postala izvršni program, potrebno je na datoteku u kojoj se skripta nalazi dati pravo izvršavanja naredbom `chmod`.

Shell skripte koriste se za razne zadaće, od automatiziranja jednostvanih radnji, do kompleksnih skripti koje rješavaju složene programerske probleme. Područje programiranja za ljusku opsežna je cjelina i izlazi iz okvira ove knjige. Ipak, svaki ozbiljniji unix korisnik trebao bi poznavati osnovne principe pisanja shell skripti. Čak i ako nemamo ambicije pisanja svojih skripti, zbog velikog broja problema koji su riješeni upravo na ovaj način, velika je vjerovatnost da ćemo se u radu sresti sa nekom vrstom skripte, a temeljna znanja poslužit će nam barem za razumijevanje njezinog sadržaja.

U sljedećem pasusu dani su osnovne shell skripti. Objašnjeno je korištenje varijabli, argumenata naredbenog retka i naredbi kontrole toka, te značenje posebnih znakova i navodnika *(quoting)*. Izloženo je potkrijepljeno primjerima. Objašnjenja i primjeri dani su paralelno za `sh` i `csh` ljusku.

1.5.1 Varijable

Zadavanje varijabli

sh	csh
`a=3`	`set a=3`
`b=5`	`set b=5`

Čitanje sadržaja varijabli

sh	csh
`echo a=$a b=$b`	`echo a=$a b=$b`
`[ $a = $b ]`	`test $a = $b`
`echo $?`	`echo $status`

Naredba `test` *koristi se za provjeru tipova datoteka i usporedbu vrijednosti.*

Argumenti naredbenog retka primaju se kroz varijable `$1, $2, ...`.
Ljuska `csh` dozvoljava i sintaksu sličnu `C` jeziku, pa se argumenti mogu čitati kroz varijable `$argv[1], $argv[2], ...`

sh	csh

`echo $1 $2`	`echo $1 $2`
	ili
	`echo $argv[1] $argv[2]`

1.5.2 Quoting - mijenjanje značenja znakova

Naredbeni jezik ljuske koristi nekoliko posebnih znakova (`<, >, |, $, ...`) za izvođenje određenih operacija, ili mijenjanje značenja drugih znakova. Postoje slučajevi u kojima korisnik želi upotrijebiti te znakove kao dio običnog teksta. Tablica 1.1 prikazuje načine uklanjanja posebnih znakova iz naredbene linije.

znak	značenje
\	isključuje funkciju sljedećeg posebnog znaka
'...'	Isključuje funkciju posebnih znakova u nizu znakova koji se nalazi između navodnika
"..."	Isključuje funkciju posebnih znakova u nizu znakova koji se nalazi između navodnika, osim znakova $, ", ', \
‘naredba‘	Izvršava naredbu i zamjenjuje string sa standardnim izlazom pokrenute naredbe

Tablica 1.1: Quoting - mijenjanje značenja posebnim znakovima

1.5.3 Naredbe kontrole toka

Naredba **if** - uvjetno izvršavanje koda

sh	csh
`if [ uvjet1 ]; then`	`if (uvjet1) then`
`  naredbe`	`  naredbe`
`elif [ uvjet2 ]; then`	`else if (uvjet2) then`
`  naredbe`	`  naredbe`
`else`	`else`
`  naredbe`	`  naredbe`
`fi`	`endif`

switch - case - uvjetno izvršavanje koda

sh	csh
`case` *`var`* `in`	`switch (`*`var`*`)`
`  `*`vrijednost1`*` )`	`  case `*`vrijednost1`
`    naredbe`	`    naredbe`
`    ;;`	`    breaksw`
`  `*`vrijednost2`*` )`	`  case `*`vrijednost2`
`    naredbe`	`    naredbe`
`    ;;`	`    breaksw`
`  * )`	`  default:`
`    naredbe`	`    naredbe`
`esac`	`endsw`

while petlja - ponavljanje niza naredbi

sh	csh

`while [uvjet]; do`
`  naredbe`
`done`
	`while (uvjet)`
	`  naredbe`
	`end`

for petlja - izvršavanje niza naredbi za sve elemente liste

sh	csh

`for` *`var`* `in` *`lista`*`; do`
`  naredbe`
`done`
	`foreach` *`var`* `(`*`lista`*`)`
	`  naredbe`
	`end`

1.5.4 Primjeri

Dana je skripta koja čita dva argumenta naredbenog retka. Ispituje se jesu li zadani argumenti jednaki i ispisuje se poruka.

sh	csh
`#!/bin/sh`	`#!/usr/bin/csh`

`if [ $1 = $2 ]; then`
`  echo jednaki`
`else`
`  echo nisu jednaki`
`fi`
	`if ($1 == $2) then`
	`  echo jednaki`
	`else`
	`  echo nisu jednaki`
	`endif`

Dana je skripta koja odbrojava unazad prema nuli. Odbrojavanje počinje od broja kojeg korisnik zada kao argument naredbenog retka.

sh	csh
`#!/bin/sh`	`#!/usr/bin/csh`
`a=$1`	`set a=$1`

`while [$a -gt 0 ]; do`
`  echo $a`
`  a=‘expr $a - 1‘`
`done`
	`while ($a > 0)`
	`  echo $a`
	`  set a=‘expr $a - 1‘`
	`end`

Naredba `expr` *koristi se za izvođenje izraza. Na ovaj način mogu se provesti aritmetičke operacije na varijablama. U ovom primjeru koriste se backquotes*

`(set a=‘expr $a - 1‘)` - *varijabla* `a` *zamijenjuje se standardnim izlazom naredbe.*

Dana je skripta koja izlistava sadržaj direktorija na način da su satoteke poredane po vremenu zadnje izmjene (`ls -t`). Direktorij se zadaje kao argument naredbenog retka.

sh	csh
`#!/bin/sh`	`#!/usr/bin/csh`
`a=‘ls -t $1‘`	`set a=‘ls -t $1‘`

`for i in $a; do`
`  echo $i`
`done`
	`foreach i ( $a )`
	`  echo $i`
	`end`

Skripta iz prethodnog primjera je dorađena na način da korisniku ostavlja mogućnost izbora načina prikaza datoteka. Datoteke mogu biti prikazane po vremenu pristupa, ili po veličini.

sh	csh
`#!/bin/sh`	`#!/usr/bin/csh`

`case $1 in`
`  size )`
`  a=‘ls -S $2‘`
`  ;;`
`  time )`
`  a=‘ls -t $2‘`
`  ;;`
`  default`
`  a=‘ls $2‘`
`esac`
	`switch ( $1 )`
	`  case size`
	`  set a=‘ls -S $2‘`
	`  breaksw`
	`  case time`
	`  set a=‘ls -t $2‘`
	`  breaksw`
	`  default`
	`  set a=‘ls $2‘`
	`endsw`

`for i in $a; do`
`  echo $i`
`done`
	`foreach i ( $a )`
	`  echo $i`
	`end`