



Ljuska operacijskog sustava



Ljuska operacijskog sustava

- fleksibilno (tekstualno) sučelje prema operacijskom sustavu
- prije grafičkih okolina za operacijske sustave – osnovni način pokretanja operacija
 - ne samo Unix – MS-DOS, CPM, VMS, ...
 - kreiranje mapa u datotečnom sustavu, pregledavanje sadržaja mapa
 - pokretanje programa
 - razni alati koji se pokreću iz komandne linije (naredbenog retka)
- i danas svi operacijski sustavi nude komandnu liniju
- umjesto traženja neke naredbe ili opcije u menijima, iskusan korisnik može ih brže pokrenuti iz komandne linije
 - u slučaju velikog broja naredbi, te bogatog skupa opcija – može biti znatno brže
- niz naredbi može se zapisati u tekstualnu datoteku iz koje se zatim pokreću (*batch obrada*)



Unix ljuska

- ljuska (*shell*) je program koji posreduje između korisnika i operacijskog sustava (UNIX)
- nije dio jezgre operacijskog sustava već korisnički program – koristi pozive prema ljusci OS-a
- ljuska naprosto prihvaća naredbe koje korisnik unosi u naredbenom retku (*command line*) i prevodi ih u pozive jezgre OS-a
- “svatko” može napisati svoju ljusku
- standardni program – kompatibilnost
- jedna od najvažnijih značajki Unix-a: velika zbirka programa – preko 200 osnovnih naredbi (alata) standardno se isporučuje
- možda još važnije – lakoća kojom se te naredbe mogu kombinirati za obavljanje sofisticiranih zadataka – tu je ljuska presudna
 - podržava programske konstrukte koji omogućavaju ispitivanje uvjeta, formiranje programskih petlji, korištenje varijabli za pohranu vrijednosti



Unix ljuska (2)

- standardne ljuske koje se isporučuju s Unix/Linux sustavima izvedene su iz *sh* ljuske razvijene u AT&T Bell Labs (Stephen Bourne, , 1979.)
 - *csh* ljuska (Bill Joy, Berkeley, 1979.)
 - *ksh* ljuska
 - *bash* ljuska
- IEEE je razvio standarde temeljene na Bourne ljusci i ostalim novijim izvedbama (Shell and Utilities volume of IEEE Std 1003.1-2001, poznat kao *POSIX* standard)
- opis i primjeri u ovom predmetu bit će temeljeni na *bash* ljusci (*Bourne Again Shell*)
 - najraširenija ljuska
 - u svim standardnim distribucijama Linux-a i Unix-a
 - Cygwin – port Unix alata na MS Windows



Unix ljuska (3)

- kada se prijavimo na sustav (*logiramo*), pokreće se (neka) ljuska i moguća je interakcija s OS-om putem naredbenog retka
- ako želimo (privremeno) promijeniti ljusku, možemo je pokrenuti kao program:

```
$ bash
```
- napuštanje ljuske:

```
$ exit
```
- ako poželimo promijeniti ljusku koju uobičajeno koristimo (*login shell*), možemo upotrijebiti naredbu `chsh` (*change shell*) (ne radi pod Cygwin-om):

```
$ chsh -s bash
```
- mogućnost uređivanja naredbe, ponavljanja prethodne unesene naredbe (`strelice`), automatskog nadopunjavanja (`Tab`), pretraživanja liste prethodnih naredbi (`ctrl-R`)



Lektira

- prof. Mario Žagar, *UNIX i kako ga koristiti*
(<http://www.fer.hr/rasip/knjiznica>)
- prof. Mario Žagar, *UNIX i kako ga iskoristiti*
(<http://www.fer.hr/rasip/knjiznica>)



Sustav pomoći

- više podataka o nekoj Unix naredbi može se dobiti navođenjem parametra `--help`:

```
$ ls --help
```

```
$ ls --help | more
```

 za prikaz stranicu po stranicu

- prikaz dokumentacije (*manpages*):

```
$ man ls
```

- unutar stranice možemo koristiti pretraživanje
 - : /trazeni_tekst (može se navesti i regularni izraz)
 - : n za traženje sljedeće pojave
 - : N za traženje prethodne pojave
 - : q izlaz

- još neki alati za dobivanje dodatnih informacija:

```
info, apropos, whatis
```



Datoteke i kazala

- Unix koristi *hijerarhijski* datotečni sustav
- nakon pristupanja sustavu, korisnik se nalazi u svom osobnom kazalu datoteka (*home directory*)
- unutar svog kazala korisnik može stvarati nove datoteke, kao i oblikovati nova kazala → rezultat je stablasta struktura, s korijenom “/” (*root directory*)
- stvaranje novog kazala:
`$ mkdir naziv_kazala`
- prikaz sadržaja tekućeg kazala:
`$ ls`
`$ ls -F, $ ls -l, $ ls -al`
- promjena tekućeg kazala:
`$ cd naziv_kazala`



Datoteke i kazala (2)

- prikaz tekućeg kazala:

```
$ pwd  
/home/korisnik/SkriptniJezici/ciklus1
```

- premještanje u kazalo prve više razine hijerarhije:

```
$ cd ..  
$ pwd  
/home/korisnik/SkriptniJezici
```

- korištenje oznake za osobno kazalo (~):

```
$ cd ~/SkriptniJezici/ciklus1  
$ pwd  
/home/korisnik/SkriptniJezici/ciklus1
```

- brisanje datoteke:

```
$ rm ime_datoteke
```

- brisanje kazala:

```
$ rmdir ime_kazala
```



Datoteke i kazala (3)

- rekurzivno brisanje datoteka (brišu se podkazala i njihovi sadržaji – pažljivo, proučiti dokumentaciju !):

```
$ rm -r ime_kazala
```

- kopiranje datoteke:

```
$ cp datoteka1 datoteka2
```

- preimenovanje (premještanje) datoteke:

```
$ mv ime_prije ime_poslije
```

- primjena naredbi na druga kazala:

```
$ mv ime_prije
```

```
~/SkriptniJezici/ciklus1/ime_poslije
```

- posebne oznake kazala:

tekuće kazalo (*current directory*): “.”

kazalo prve više razine hijerarhije: “..”



Uređivanje i prikaz datoteka

- primjena uređivača teksta (*editor*):
vi, emacs, joe, nedit, nano,...
- primjenom naredbe `cat`:
`cat > ime_datoteke`
unos teksta, završetak s `ctrl-D`
- prikaz datoteke:
`cat ime_datoteke`
`cat -n ime_datoteke`
- prikaz datoteke ekran po ekran (listanje s “blank”, izlaz s “q”):
`more ime_datoteke`
- inačica s mogućnošću listanja gore/dolje:
`less ime_datoteke`



Preusmjeravanje (redirekcija)

- ponekad je prikladno preusmjeriti izlaz (ispis) nekog programa u datoteku:
`ls > ime_datoteke`
- prije izvršavanja naredbe (programa) njen ulaz odnosno izlaz može biti preusmjeren korištenjem posebne notacije koju interpretira ljuska
- uobičajeno – program ispisuje podatke na ekran (`stdout`, deskriptor datoteke je 1, može se izostaviti)
- dijagnostički ispis (poruke o pogreškama) – obično se ispisuje u `stderr` (ako nije preusmjeren, pojavljuje se na ekranu; deskriptor datoteke je 2)
`program 2> ime_datoteke`
- zajedničko preusmjeravanje `stdout` i `stderr`
`program &> ime_datoteke`
- preusmjeravanje `stderr` na `stdout`: `2>&1`
`gcc x.c 2>&1 | less`



Preusmjeravanje (2)

- nadodavanje (*append*) na postojeću datoteku:
`ls >> ime_datoteke`
- standardni ulaz s tipkovnice `stdin`
- preusmjeravanje standardnog ulaza (deskriptor datoteke je 0, može se izostaviti):
`program < ime_datoteke`
- cjevovodi (*pipelines*) – izlaz jednog programa prosljeđuje se na ulaz drugog :
`ls | more`
- *filter* – naredba (ili program) čiji ulaz se čita sa `stdin` a izlaz se prosljeđuje na `stdout`
primjer: ispisati dio sadržaja neke (tekstne) datoteke i to od 21. do 35. retka i to zapisati u novu datoteku:
`$ tail -n +21 < lista.txt | head -n15 > kratka.txt`



Uvjetno izvođenje naredbi

- nizovi ulančanih naredbi (povezanih cjevovodom) mogu se kombinirati tako da se njihovo izvođenje međusobno uvjetuje
 - `<prvi niz naredbi> && <drugi niz naredbi>`
drugi niz naredbi izvršava se samo ako je naredba (naredbe) ispred operatora znakova `&&` vratila izlazni status "0"
`$ who | grep -s "mario" && echo "Mario je tu"`
 - `<prvi niz naredbi> || <drugi niz naredbi>`
drugi niz naredbi izvršava se samo ako je naredba (naredbe) ispred operatora znakova `||` vratila izlazni status različit od "0"
`$ who | grep -s "mario" || echo "Mario nije tu"`



Variable ljske

- postavljanje vrijednosti varijabli:
`X="abcd"` ili `X=abcd`
- `bash` je osjetljiv na razmak prije i poslije znaka jednakosti, pa će sljedeće izazvati pogrešku (točnije, ljska će pokušati pozvati program S):
`S = "abcd"`
- znakovi navodnika nisu nužni, osim ako želimo pridružiti varijabli tekst s razmacima, npr.:
`S="idemo na rucak"` a ne `S=idemo na rucak`
- varijabla se koristi predznačena znakom "\$":
`echo $S` a ne `echo S`
- imena varijabli sastoje se od slova, brojeva i znaka "_", razlikuju se velika i mala slova
- prikaz svih varijabli ljske naredbom `set`



Variable okoline

- posebna vrsta varijabli ljuske su varijable okoline (*environment variables*)
- varijable okoline koje postavimo u ljusci nasljeđuju se u programima pokrenutim iz ljuske, odnosno pod-ljuskama
- postavljanje varijable okoline (zapravo vrijednosti varijable i atributa `export`):
`export varijabla=vrijednost`
- ispis (eksportiranih) varijabli okoline:
`export -p` ili `env`
- ugrađene (*built-in*) varijable okoline – varijable sustava (*system variables*)



Variable sustava

- varijabla `$PATH` – izuzetno važna
 - sadrži listu kazala (odvojenih dvotočkom) u kojima se traže izvršne datoteke kada se ljusti zada naredba
 - redoslijed kazala u listi određuje redoslijed pretraživanje

```
$ echo $PATH  
/usr/local/bin:/usr/bin:/bin
```

- radno kazalo određeno je varijablom `$PWD`, promjenu radnog kazala možemo napraviti i ovako (umjesto `cd ciklus1`):
`PWD=$PWD"/ciklus1"`
- varijabla `$TERM` – tip terminala



Variable sustava (2)

- varijabla `$HOME` određuje matično kazalo korisnika
- `$USER` sadrži korisničko ime
- `$SHELL` sadrži ime pokrenute ljuske (uključujući put)
- `$PS1` pohranjuje oznaku koja se ispisuje na početku naredbenog retka

- primjer podešavanja oznake:

```
Green="\[\033[32m\]"
```

```
Brown="\[\033[33m\]"
```

```
White="\[\033[0m\]"
```

```
export PS1="$Green\u@\h:$Brown\w$White\n\$ "
```

- značenje nekih oznaka:

`\h` : ime računala, do prve točke

`\n` : novi red

`\u` : ime korisnika

`\w` : radno kazalo

`\[...\]` : sekvenca znakova za upravljanje terminalom



Pokretanje ljuske

- ljuska se može pokrenuti u interaktivnom i neinteraktivnom načinu rada
- interaktivni rad: korisnik zadaje naredbe i dobiva rezultate, mjesto unosa označeno je odzivnim znakom (*prompt*)
- neinteraktivni rad: ljuska izvodi naredbu danu kao argument ili tumači naredbe iz liste naredbi (skripta)
- prilikom pokretanja ljuske kao interaktivne *login* ljuske (ili neinteraktivne s opcijom `--login`), najprije čita i izvršava naredbe iz datoteka (ako postoje i čitljive su):

```
/etc/profile
```

```
~/ .bash_profile
```

```
~/ .bash_login
```

```
~/ .profile
```

- prilikom napuštanja *login* ljuske, čitaju se i izvršavaju naredbe iz datoteke (ako postoji):

```
~/ .bash_logout
```



Pokretanje ljuske (2)

- pri pokretanju interaktivne ljuske koja nije *login*, čitaju se i izvršavaju naredbe iz datoteke (ako postoji):
`~/ .bashrc`
- pri pokretanju neinteraktivne ljuske (npr. za izvršavanje neke skripte), koristi se varijabla okoline `BASH_ENV` za određivanje datoteke koja će se izvršiti



Koraci obrade naredbenog retka

- Ijuska obavlja sljedeći niz operacija:
 1. Ijuska čita sadržaj ulaznog retka
 2. ulazna linija razlaže se u niz simbola (*tokens*): riječi i operatora
 3. niz simbola se parsira – prepoznaju se jednostavne i složene naredbe
 4. obavlja se širenje (*expansion*) pojedinih dijelova svake naredbe (riječi) – rezultat je lista imena putanja, te polja koja predstavljaju naredbe i argumente
 5. obavlja se preusmjeravanje, te uklanjaju operatori preusmjeravanja i njihovi operandi iz liste parametara
 6. pokreće se funkcija, ugrađena naredba, izvršna datoteka ili skripta, pri čemu se predaju i imena argumenata u obliku pozicijskih parametara označenih s 1 do n, i ime naredbe ili skripte u obliku parametra 0
 7. Ijuska čeka da naredba završi, te prikuplja njen izlazni status



Navodnici i uloga u tumačenju naredbi (1)

- neki znakovi imaju posebno značenje u naredbama koje ljuska tumači:
| & ; < > () \$ ` \ " ' <space> <tab> <newline>
- kako bi se znak tumačio doslovce (bez specijalnog značenja), potrebno ga je predznačiti (*escape*) znakom “\”
- niz znakova uokviren jednostrukim navodnicima (*single-quotes*) ne tumači se u ljusci, već se samo prosljeđuje kao niz:
\$ echo 'Tekuce kazalo je \$PWD'
Tekuce kazalo je \$PWD
- dvostruki navodnici (*double-quotes*) uokviruju niz kao cjelinu, ali pritom se obavlja širenje (*expansion*) argumenata označenih posebnim znakovima ljuske:
\$ echo "Tekuce kazalo je \$PWD"
Tekuce kazalo je /home/kalfa/Skriptni/ciklus1



Navodnici i uloga u tumačenju naredbi (2)

- obrnuti jednostruki navodnici (*backquotes*) – sadržaj unutar tih navodnika izvodi se kao UNIX naredba, na to mjesto se upisuje izlaz koji je naredba generirala

```
$ Datum=`date`
```

```
$ echo $Datum
```

```
Fri Feb 15 23:23:11 CEST 2008
```

```
$ echo "Danas je `date +%A, %d.%m.%Y`"
```

```
Danas je Sunday, 02.03.2008
```



Širenja u naredbenom retku

- širenje (*ekspanzija*) se obavlja nakon dijeljenja naredbenog retka u riječi
- razlikuje se sedam vrsta širenja, obavljaju se navedenim redoslijedom
 - širenje vitičastih zagrada
 - širenje znaka ~
 - širenje parametara i varijabli
 - zamjena naredbi
 - aritmetičko širenje
 - dijeljenje riječi
 - širenje imena datoteka (*pathname expansion*)



Širenje vitičastih zagrada

- širenje vitičastih zagrada (*brace expansion*) – mehanizam kojim se mogu generirati proizvoljni znakovni nizovi
- uzorci koje treba proširiti tipično se sastoje od prefiksa, slijeda znakovnih nizova odvojenih zarezom ili izraza za generiranje sekvence, navedenih unutar vitičastih zagrada, te sufiksa:

`a{d,c,b}e` širi se u `'ade ace abe'`

`$ echo ab{c,d,e,f}oooo`

`abcoooo abdoooo abeoooo abfoooo`

- izrazi za generiranje sekvenci su oblika `{x..y}`, gdje su `x` i `y` brojevi (šire se numerički) ili znakovi (šire se leksikografski):

`$ echo ab{1..4}ij`

`ab1ij ab2ij ab3ij ab4ij`

- širenje vitičastih zagrada obavlja se prije svih ostalih ekspanzija, znakovi za ostale ekspanzije se čuvaju

- primjer:

`$ mkdir projekt/{old,new,dist,bugs}`



Širenje oznake matičnog kazala

- ako riječ počinje s “~” (tilda) svi znakovi do prvog znaka “/” ili kraja riječi smatraju se tilda-prefiksom
- ako tildu slijedi riječ bez navodnika, ona se smatra mogućim korisničkim imenom (ako korisnik tog imena ne postoji, ekspanzija se ne obavlja):

```
$ cp ~francek/bin/*.c .  
$ echo ~francek  
/home/francek
```

- ako je tilda-prefiks samo tilda, ona se zamjenjuje varijablom \$HOME, odnosno matičnim kazalom korisnika koji je pokrenuo ljusku:

```
ls ~/scripts/*.pl  
$ echo ~  
/home/Korisnik
```



Širenje varijabli i parametara

- znak “\$” uvodi širenje varijabli i parametara, zamjenu naredbe ili aritmetičku ekspanziju
- ime ili oznaka varijable ili parametra može se uokviriti vitičastim zagradama kako bi se izbjegla moguća pogrešna interpretacija

```
$ ab=1234
$ echo ${ab}c
1234c
```
- postoje različite varijante širenja varijabli:
 - korištenje pretpostavljene (*default*) vrijednosti ako varijabla nije postavljena ili je prazna

```
${varijabla:-word}
```
 - postavljanje pretpostavljene vrijednosti ako varijabla nije postavljena ili je prazna

```
${varijabla:=word}
```
 - za više detalja pogledati `man`



Zamjena naredbe

- zamjena naredbe (*command substitution*) omogućuje da se navedeni tekst protumači i izvede kao naredba, te se zatim zamijeni generiranim izlazom izvršene naredbe

- koriste se dva oblika:

`$(naredba)`

``naredba``

```
$ echo "Danas je $(date "+%A, %d.%m.%Y") "
```

```
Danas je Sunday, 02.03.2008
```

- zamjene naredbi mogu se gnijezditi
 - ako se koristi notacija s jednostrukim obrnutim navodnicima, unutarnji navodnici se moraju predznačiti s “\”



Širenje aritmetičkih izraza

- aritmetička ekspanzija omogućuje evaluaciju aritmetičkih izraza i zamjenu rezultatom
- format za ovu vrstu širenja:

```
$ ( (expression) )  
$ echo $ ( (3*7+15/3) )  
26
```
- članovi izraza podvrgavaju se svim prethodnim širenjima

```
$ a=5  
echo $ ( ( $a*3%4 ) )  
3
```
- aritmetičke ekspanzije mogu se gnijezditi
- za neispravne izraze ispisuje se obavijest o pogrešci



Širenje imena datoteka

- nakon dijeljenja naredbenog retka na riječi, u svakoj riječi traže se znakovi *, ? i [
 - ako se pronađe neki od tih znakova, riječ se smatra uzorkom (*pattern*) i zamjenjuje se abecedno poredanom listom imena datoteka koje odgovaraju uzorku
 - u uzorku svaki znak (osim posebnih) predstavlja sam sebe, specijalni znakovi koji se žele doslovno podudarati moraju biti predznačeni s “\”
- značenja specijalnih znakova:
 - * predstavlja proizvoljan (pod)niz znakova, uključujući i prazan niz
 - ? predstavlja jedan proizvoljan znak
 - [. . .] podudara se s jednim znakom iz skupa znakova navedenih unutar zagrada



Širenje imena datoteka (2)

- može se zadati raspon znakova `[a-c]`
- ako je prvi znak unutar zagrada “^” ili “!” onda se radi o negaciji skupa navedenih znakova – podudara se s bilo kojim znakom koji nije naveden u skupu
- ako je potrebno podudarati znak “-”, treba ga navesti kao prvog ili zadnjeg unutar zagrada
- znak “[” može se podudarati ako se navede kao zadnji unutar zagrada
- standard POSIX.2 definira i oznake za posebne klase znakova, koje se označavaju u formatu `[:klasa:]`, pri čemu su predviđene oznake za klase:
alnum alpha ascii blank cntrl digit graph lower
print punct space upper word xdigit
- primjer:

```
$ ls [[:alpha:]]* [[:digit:]]*  
skriptni_P2.pdf skriptni_P2.ps skriptni_P2.tex
```



Složeniji Unix alati

- u Unix/Linux distribucijama standardno se isporučuje niz vrlo korisnih alata
- brojanje riječi, redaka, znakova: `wc`
- ispis redaka teksta u kojima se pronalazi zadani uzorak: `grep`
- manipulacija tekstnim tokom (*stream*) `sed`
- sortiranje (`sort`), manipulacija ponovljenim retcima (`uniq`), izdvajanje dijelova retka (`cut`)
- traženje datoteka: `find` i `locate`



Brojanje riječi, znakova i redaka

- koristimo naredbu `wc` (*word count*)
- broji riječi, linije i znakove u datoteci ili na standardnom ulazu
- oblik naredbe:
`wc [opcije] [<datoteka>]`
- opcije:
 - `w` : ispisuje se samo broj riječi
 - `l` : ispisuje se samo broj redaka
 - `c` : ispisuje se samo broj znakova
- ispisuju se redom broj redaka, riječi i znakova
- primjeri:
`$ wc /etc/passwd`
`$ wc -l /etc/passwd`
- ako se ne navede ime datoteke, obrađuje se standardni ulaz:
`$ ls -l /usr/include | wc -l`



Pretraživanje tekstnih datoteka

- koristimo naredbu `grep`
- traži se uzorak u datoteci ili u podacima sa standardnog ulaza, redak po redak
- redak u kojem se pronađe zadani uzorak ispisuje se na standardni izlaz
- mnoštvo opcija kojima se određuje ponašanje naredbe
- oblik naredbe:
`grep [opcije] trazen_i_uzorak [<datoteka>]`
- neke često korištene opcije:
 - v : ispisuju se linije koje ne sadrže zadani uzorak
 - i : pretraživanje bez razlikovanja malih i velikih slova
 - c : ne ispisuju se retci teksta, već samo broj redaka u kojima je uzorak pronađen
 - E : `trazen_i_uzorak` predstavlja regularni izraz s proširenom sintaksom (*extended*)
 - q : nema ispisa, izlazni status označava je li uzorak pronađen



Pretraživanje tekstnih datoteka (2)

● primjeri korištenja:

```
$ grep "<cv.h>" *.c
```

```
$ grep Korisnik /etc/passwd
```

● grep i regularni izrazi

- regularni izraz je uzorak koji *opisuje* skup znakovnih nizova
- regularni izrazi grade se od jednostavnijih regularnih izraza primjenom posebnih operatora
- `grep` prihvaća dvije sintakse regularnih izraza: “osnovnu” i “proširenu” (opcija `-E`)
- osnovni građevni blok je regularni izraz koji predstavlja pojedinačni znak
- većina znakova (npr. sva slova i znamenke) predstavljaju sami sebe
- posebni znakovi (metaznakovi) se mogu tumačiti doslovno tako da ih se predznači s “\”



Regularni izrazi

- uglate zagrade “[]” određuju listu znakova s kojima se može podudarati pojedinačni znak na odgovarajućem mjestu traženog uzorka
 - ako je prvi znak unutar zagrada “^” lista se tumači kao negacija – znak se može podudarati sa svim znakovima osim onima navedenim unutar zagrada
 - primjer:
[0123456789] podudara se s bilo kojom znamenkom
[^0123456789] na tom mjestu ne smije stajati znamenka
 - unutar zagrada može se definirati raspon – označava se s dva znaka između kojih je znak “–”
 - kod zadavanja raspona treba pripaziti, jer rezultat ovisi o postavkama sustava: [a–d] može odgovarati [abcd] ali i [aBbCcDd] ovisno o varijablama okoline !

Regularni izrazi (2)

- za neke uobičajene klase znakova definirane su simboličke oznake (pažnja – uglate zagrade *su dio* simboličke oznake klase):
`[:alnum:], [:alpha:], [:cntrl:], [:digit:],`
`[:graph:], [:lower:], [:print:], [:punct:],`
`[:space:], [:upper:], [:xdigit:]`
- unutar liste većina metaznakova gubi svoje specijalno značenje
 - da bi se u listu uključio znak “`]`” mora biti naveden na početku liste
 - da bi se u listu uključio znak “`^`” ne smije stajati na početku liste
 - da bi se u listu uključio znak “`-`” mora biti naveden na kraju liste
- točka “`.`” se podudara s bilo kojim pojedinačnim znakom
- oznaka “`\w`” je sinonim za `[:alnum:]`, a “`\W`” sinonim za `[^ :alnum:]`



Regularni izrazi (3)

- znak “^” označava *početak retka*
- znak “\$” označava *kraj retka*
`grep '\.eps$' *.txt`
- oznake “\<” i “\>” označavaju početak odnosno kraj riječi
“\<pra” pronalazi “pravednost” ali ne “naprasno”
“ba\>” pronalazi “grba” ali ne “banalno”
- oznaka “\b” odgovara granici riječi
“\bpra” pronalazi “pravednost” ali ne “naprasno”
“ba\b” pronalazi “grba” ali ne “banalno”
- oznaka “\B” odgovara svemu *osim* granici riječi
“\Bpra” pronalazi “naprasno” ali ne “pravednost”
“ba\B” pronalazi “banalno” ali ne “grba”

Regularni izrazi (4)

- nakon regularnog izraza može se navesti jedan od nekoliko operatora ponavljanja:
 - ? : prethodni izraz se pojavljuje najviše jednom
 - * : prethodni izraz se pojavljuje 0 ili više puta
 - + : prethodni izraz se pojavljuje jednom ili više puta
 - { n } : prethodni izraz se pojavljuje točno n puta
 - { n, } : prethodni izraz se pojavljuje n ili više puta
 - { n, m } : prethodni izraz se pojavljuje barem n ali najviše m puta
- dva regularna izraza mogu se nadovezati – složeni izraz podudara se s nizovima koji se sastoje od dva podniza koji odgovaraju nadovezanim regularnim izrazima (ulančavanje)
- dva regularna izraza mogu se povezati operatorom “|” (alternacija) – složeni izraz podudara se s nizovima koji se podudaraju s bar jednim od dva povezana regularna izraza



Regularni izrazi (5)

- ponavljanje se obavlja prije ulančavanja, a tek nakon toga dolazi na red alternacija
- zagradama se može izmijeniti redoslijed primjene
- povezivanje unazad (*backreference*) označava se s “\n”, gdje *n* predstavlja jednu dekadsku znamenku – odgovara podnizu koji je prethodno podudaren s *n*-tim podizrazom uokvirenim zagradama

```
$ grep '\([0-9]\{2\}\)\.\1' rodjendani.txt
```

pero 03.03.1981.
ivica 12.12.1977.
- “osnovna” sintaksa regularnih izraza zahtijeva da metaznakovi budu predznačeni: \?, \+, \{, \|, \(\, \), a proširena ne:

```
$ grep -E '([0-9]{2})\.\1' rodjendani.txt
```
- koristit ćemo proširenu sintaksu



Sortiranje teksta

- koristimo naredbu `sort`
- naredba se može primjenjivati elementarno – zadani ulazni tekst se slaže (redak po redak) po (engleskoj) abecedi i zapisuje na standardni izlaz

```
$ sort imena.txt > imena_sortirano.txt
```
- mnoštvo opcija kojima se podešava ponašanje naredbe
- oblik naredbe:

```
sort [opcije] [<datoteka>]
```



Sortiranje teksta (2)

- neke često korištene opcije:
 - u : jednaki retci se ne ispisuju dvaput
 - r : sortiranje obrnutim redoslijedom
 - o : navodi se ime izlazne datoteke
korisno: `sort imena.txt -o imena.txt`
jer `sort imena.txt > imena.txt` ne funkcioniра !
 - n : numeričko sortiranje
 - k N,M : sortiranje se obavlja na temelju teksta od *N*-tog do *M*-tog polja, uobičajeni graničnici su praznina i `tab`
 - t : navodi se drugačiji graničnik
- primjer – numeričko sortiranje po trećem polju, pri čemu je graničnik znak “:”
`$ sort -n -k 3 -t : /etc/passwd`



Pronalaženje ponovljenih redaka

- koristimo naredbu `uniq`
- pronalaze se ponovljeni retci teksta, pri čemu se ponavljanjem smatra *uzastopno* ponavljanje – potrebno je prethodno sortirati tekst:

```
$ sort imena.txt | uniq > imena_2.txt
```

ista funkcija kao `sort -u imena.txt > imena_2.txt`

- opcije omogućavaju i druge namjene
- oblik naredbe:

```
uniq [opcije] [<in_file> [<out_file>]]
```



Pronalaženje ponovljenih redaka (2)

- neke često korištene opcije:
 - d : ispisuju se samo ponovljeni retci
 - c : ispisuje se i broj ponavljanja
 - f N : iz usporedbe se isključuje prvih N polja
 - s N : iz usporedbe se isključuje prvih N znakova
 - i : ne uzima se u obzir razlika između velikih i malih slova
- primjeri:
 - provjera postoje li duplikati u datoteci /etc/passwd :
`$ sort /etc/passwd | uniq -d`
 - ispis broja ponavljanja višestrukih redaka teksta:
`$ sort imena.txt | uniq -c`



Manipulacija tekстом

- koristimo naredbu `sed` (*stream editor*)
- namijenjena filtriranju i mijenjanju teksta
- složen alat, upoznat ćemo se samo s nekoliko osnovnih funkcija
- oblik naredbe:
`sed [opcije] [-e upute] [-f upute_dat] [<datoteka>] ...`
- značenje opcija:
 - `-n` : ne ispisuje se izlazni rezultat
 - `-r` : koriste se regularni izrazi proširene sintakse
 - `-e upute` : naredbe za `sed`, ako je samo jedna naredba i nema opcije `-f`, `-e` se može izostaviti
 - `-f upute_dat` : upute se čitaju iz datoteke `upute_dat`
 - `<datoteka>` : datoteka s podacima



Manipulacija tekstem (2)

- nekoliko primjera:

- zamjena teksta:

```
$ sed 's/Unix/UNIX/' intro.txt
```

```
$ sed 's/Unix/UNIX/' intro.txt > intro2.txt
```

- Napomena: zamjenjuje se samo prva pojava navedenog niza u pojedinom retku ! Ako želimo zamijeniti sve podnizove koji odgovaraju navedenom uzorku (regularni izraz), treba navesti opciju **g** (globalno):

```
$ sed 's/Unix/UNIX/g' intro.txt > intro2.txt
```

- brisanje pronađenog uzorka (npr. 3 zadnja znaka u svakom retku):

```
$ sed 's/...$//' datoteka.txt
```

- ispis samo redaka u kojima se pronađe zadani uzorak:

```
$ sed -n '/UNIX/p' intro.txt
```

- brisanje redaka u kojima se pronađe zadani uzorak:

```
$ sed '/UNIX/d' intro.txt
```



Traženje datoteka

- koristimo naredbu `find`
- složena naredba s mnoštvom opcija
- oblik naredbe:
`find [opcije] [<direktoriji>] <uvjeti>`
`find [opcije] [<direktoriji>] <uvjeti>`
- češće korišteni uvjeti:
 - `-name <ime>` : traži se datoteka zadanog imena
 - `-type <tip>` : traži se datoteka zadanog tipa
tipično: `d` – kazalo, `f` – obična datoteka, `l` – simbolički link
 - `-size <velicina>` : traži se datoteka zadane veličine
 - `-print` : ispisuje se ime pronađene datoteke
 - `-mtime n` : traže se datoteke mijenjane prije $n \cdot 24h$
 - `-regex izraz` : traže se datoteke čije osnovno ime odgovara zadanom regularnom izrazu



Traženje datoteka (2)

● primjeri:

- tražimo datoteku `passwd` u kazalu `/etc`
`$ find /etc -name passwd`
- tražimo datoteke u kazalu `/etc` koje su veće (“+”) od 1MB
`$ find /etc -size +1M`
- tražimo datoteke u kazalu `/etc` koje su manje (“-”) od 1kB
`$ find /etc -size -1k`
- tražimo datoteke u svom matičnom kazalu koje su mijenjane danas, pritom ispisujemo podatke o nađenim datotekama `-ls`
`$ find ~/ -mtime 0 -ls`
- tražimo datoteke u svom matičnom kazalu koje su mijenjane u zadnja 3 dana
`$ find ~/ -mtime -3`

Traženje datoteka (3)

● primjeri:

- tražimo datoteke u svom matičnom kazalu čije ime, uključujući punu stazu, odgovara regularnom izrazu

```
$ find ~/ -regex '.*skr.*P2.*\.pdf'
```

- tražimo sve direktorije unutar /etc

```
$ find /etc -type d
```

- tražimo sve datoteke unutar /etc

```
$ find /etc -type f
```

- tražimo sva kazala i datoteke veće od 1MB unutar /etc, te ispisujemo podatke o njima (-o znači lll, zagradama je potrebno naznačiti grupiranje)

```
$ find /etc \( -type d -o -size +1M \) -ls
```

- pogrešno korištenje – obavlja se širenje imena datoteka:

```
$ find . -name *.c -print
```

- ovo ipak možemo postići korištenjem jednostrukih navodnika (nema širenja):

```
$ find . -name '*.c' -print
```

Traženje datoteka (4)

- umjesto naredbe `find` često je prikladno koristiti naredbu `locate`
 - brža je i manje opterećuje računalo
 - za pretragu koristi unaprijed izgrađenu internu bazu podataka
 - ako na korištenom sustavu nije već podešeno automatsko periodično obnavljanje baze, potrebno je to učiniti naredbom `updatedb`

```
$ updatedb --prunepaths='/cygdrive'
```

- oblik naredbe:

```
locate [<opcije>] <uzorak>
```

<uzorak> može biti običan niz znakova, može sadržavati metaznakove koji se koriste pri širenju imena datoteka (staviti uzorak u jednostruke navodnike da ljuska ne bi obavila širenje!), a može biti i regularni izraz (opcija `-r`)

- češće korištene opcije:

`-r` : uzorak je regularni izraz

`-i` : zanemaruje se razlika između malih i velikih slova



Pisanje i pokretanje skripti

- skripte su obične tekstovne datoteke, za njihovo upisivanje i uređivanje koristimo tekst editor
- primjer skripte:

```
$ cat hello.sh  
echo "Hello world!"
```
- pokretanje skripte – naredbom `source` ili `.`

```
$ source hello.sh  
$ . hello.sh
```
- skriptu izvršava ljuska u kojoj smo je pokrenuli
- sve promjene u varijablama okoline ostaju sačuvane



Pisanje i pokretanje skripti (2)

- drugi način pokretanja skripte – proglasiti datoteku izvršnom i pokretati je kao program

```
$ chmod u+x hello.sh
$ ./hello.sh
```
- pokreće se `bash` (pod)ljuska u kojoj se skripta izvršava
 - po završetku izvršavanja, promjene varijabli okoline se gube
- u *prvom retku* skripte može se navesti interpreter koji skriptu izvršava, korištenjem oznake “#!”

```
#!/bin/sh
```



Prenošenje parametara

- ulazni podaci mogu se skripti predati kao varijable ljustice ako se skripta pokreće u istoj ljustici:

```
$ a=3
```

```
$ cat > ispisi.sh  
echo $a
```

```
$ source ispisi.sh
```

- ako se skripta pokreće u podljustici, na raspolaganju su joj samo varijable okoline (eksportirane varijable):

```
$ chmod u+x ispisi.sh  
$ ./ispisi.sh
```

```
$ export a  
$ ./ispisi.sh
```



Prenošenje parametara (2)

- uobičajen je prijenos parametara kao argumenata u naredbenom retku
 - pozicijski parametri
 - pristupa im se s $\$n$, pri čemu je n redni broj parametra
 - $\$0$ sadrži ime skripte ako je skripta pokrenuta u podljusci
 - ako je skripta pokrenuta u tekućoj ljusci (`source`) onda je parametar $\$0$ ime ljuske
- primjer skripte s parametrima:

```
$ cat parametri.sh
#!/bin/bash
echo $0
echo $1
echo $2
```
- pokretanje skripte:

```
$ ./parametri.sh par1 par2
```



Prenošenje parametara (3)

- parametri s rednim brojem većim od 9 označavaju se vitičastim zagradama:
`${10}`, `${11}` itd.
- naredbom `shift` obavlja se promjena imena pozicijskih parametara: indeksi parametara počevši od `$2` umanjuju se za 1
 - ako se koristi oblik `shift n`, argument `${n+1}` postaje `$1`, `${n+2}` postaje `$2` itd.
- broj prenesenih pozicijskih parametara navedenih u naredbenom retku može se pročitati iz varijable `$#`
- lista svih argumenata predanih skripti može se dobiti s `$@`

```
$ cat parametri2.sh
echo "Broj parametara je: $#"
```

```
echo "Parametri su: $@"
echo $0; echo $1; echo $2; echo $3
```

```
$ ./parametri2.sh p1 p2 p3
```



Izlazni status

- svaka naredba (program, skripta) pri završetku izvođenja vraća status
 - status je broj koji obično označava je li se program uspješno izvršio
 - uobičajeno izlazni status 0 označava uspješno izvršavanje, dok vrijednost različita od 0 označava neuspješno izvršavanje (ili pogrešku)
 - npr. `grep` će vratiti izlazni status 0 ako je traženi uzorak barem jednom pronađen, a neku drugu vrijednost ako uzorak nije pronađen ili se dogodi neka pogreška (neispravno navedeni argumenti, nečitljiva datoteka)
 - izlazni status zadnje izvedene naredbe (ili skripte) pohranjuje se u varijablu “\$?”
`$ echo $?`
 - kada se izvodi cjevovod (*pipeline*), izlazni status vraća posljednja naredba u cjevovodu

Ispitivanje uvjeta

- naredba `if`
- najjednostavniji oblik:
`if` naredba_t
`then`
 blok_naredbi
`fi`
- naredba naredba_t se izvršava, te se ispituje njen izlazni status
– ukoliko je status 0, blok naredbi između `then` i `fi` se izvršava

```
korisnik=$1 # ulazni argument je korisnicko ime
if grep -q $korisnik /etc/passwd
then
    echo "Korisnik $korisnik postoji."
fi
```

Ispitivanje uvjeta (2)

● složeniji oblici naredbe `if`:

● **if – then – else** oblik

```
if naredba_t
```

```
then
```

```
    blok_naredbi1 #ako je uvjet ispunjen
```

```
else
```

```
    blok_naredbi2 #ako uvjet nije ispunjen
```

```
fi
```

● naredba **elif**

```
if naredba_1 ; then
```

```
    blok_naredbi1
```

```
elif naredba_2 ; then
```

```
    blok_naredbi2
```

```
else
```

```
    blok_naredbi3
```

```
fi
```



Naredba `test`

- naredba `test` najčešće s koristi za ispitivanje uvjeta u naredbi `if`
- njen opći oblik je:
`test` `izraz`
 - `izraz` predstavlja uvjete koji se ispituju
 - naredba `test` evaluira `izraz` i ako je njegova vrijednost *istina*, izlazni status je 0
 - ako je `izraz` *neistinit* vraća se izlazni status različit od 0
 - primjer – ispitivanje jednakosti znakovnih nizova:
`test "$name" = Vedrana`
 - napomena: znak “=” mora biti odvojen od operanada !
 - dobro je varijable ljske koje se prosljeđuju kao argumenti naredbi `test` uokviriti dvostrukim navodnicima, jer ako je varijabla neinicijalizirana (prazna), širenjem parametara će “nestati” – navodnici će osigurati da i u tom slučaju naredba “vidi” argument



Naredba test (2)

● primjer:

```
dan=$(date +%A)
if test "$dan" = Sunday ; then
    echo "Danas je nedjelja."
fi
```

● ostali operatori za usporedbu znakovnih nizova:

```
string1 = string2 : istinito ako su nizovi jednaki
string1 != string2 : istinito ako su nizovi različiti
string : niz nije prazan
-n string : niz nije prazan (mora biti vidljiv naredbi test)
-z string : niz je prazan (mora biti vidljiv naredbi test)
```

● primjeri:

```
$ test "" ; echo $?
$ test "a" ; echo $?
$ test -n "a" ; echo $?
$ test -z "" ; echo $?
```



Naredba `test` (3)

- alternativni oblik naredbe `test`
 - čitljiviji, posebno u naredbi `if`
 - oblik naredbe istovjetne `s test izraz:`
`[izraz]`
 - nužno je odvojiti zagrade od izraza
 - primjer:

```
dan=$(date +%A)
if [ "$dan" = Tuesday ] ; then
    echo "Danas je utorak."
fi
```



Naredba test (4)

- operatori za usporedbu cijelih brojeva:

`int1 -eq int2` : istinito ako su brojevi jednaki

`int1 -ge int2` : istinito ako je `int1` \geq `int2`

`int1 -gt int2` : istinito ako je `int1` $>$ `int2`

`int1 -le int2` : istinito ako je `int1` \leq `int2`

`int1 -lt int2` : istinito ako je `int1` $<$ `int2`

`int1 -ne int2` : istinito ako je `int1` \neq `int2`

- primjeri:

```
$ x1="005"
```

```
$ [ "$x1" = 5 ] ; echo $? # usporedba nizova
```

```
$ [ "$x1" -eq 5 ] ; echo $? # usporedba brojeva
```

```
$ [ "$x1" -lt 15 ] ; echo $? # usporedba brojeva
```



Naredba `test` (5)

- naredba `test` može se primijeniti i za ispitivanje svojstava datoteka
- operatori za ispitivanje datoteka:
 - `-d file : file` je kazalo
 - `-e file : file` postoji
 - `-f file : file` je obična datoteka
 - `-r file : čitanje datoteke` dozvoljeno
 - `-s file : duljina datoteke` veća od 0
 - `-w file : dozvoljeno pisanje u datoteku`
 - `-x file : datoteka` je izvršna
 - `-L file : file` je simbolički link



Naredba test (6)

● primjeri:

● postoji li (obična) datoteka navedenog imena ?

```
[ -f /users/steve/phonebook ]
```

● je li nam dozvoljeno čitanje navedene datoteke ?

```
[ -r /users/steve/phonebook ]
```

● ispitivanje je li datoteka u koju su zapisivane greške neprazna, ispis datoteke:

```
if [ -s $ERRFILE ] ; then
    echo "Pronadjene greske: "
    cat $ERRFILE
fi
```




Naredba test (7)

- operator logičke negacije (!) – negira se rezultat evaluacije izraza
`[! -r /users/steve/phonebook]`
- operator logičko I (-a) – kombiniranje izraza
`[-f "$mailfile" -a -r "$mailfile"]`
- operator logičko ILI (-o) – kombiniranje izraza
`[-n "$mailopt" -o -r $HOME/mailfile]`
- operator I ima prednost pred operatorom ILI
- za promjenu redoslijeda evaluacije izraza mogu se koristiti *zagrade*
 - zagrade je potrebno predznačiti "`\ (`", "`\)`"
 - zagrade moraju biti okružene prazninama
`[\ ("$count" -ge 0 \) -a \ ("$count" -lt 10 \)]`



Naredba `exit`

- naredba `exit` omogućuje trenutni završetak izvođenja skripte
- argument naredbe vraća se kao izlazni status : `exit n`
- ako se argument ne navede, vraća se status prethodno izvedene naredbe
- primjer:

```
# brisanje zapisa u adresaru
```

```
# provjera broja argumenata
```

```
if [ "$#" -ne 1 ]; then
```

```
    echo "Incorrect number of arguments."
```

```
    echo "Usage: rem name"
```

```
    exit 1
```

```
fi
```

```
grep -v "$1" phonebook > /tmp/phonebook
```

```
mv /tmp/phonebook phonebook
```



Naredba case

- naredba `case` omogućuje usporedbu jedne vrijednosti s nizom drugih vrijednosti i izvođenje bloka naredbi koji odgovara podudaranju vrijednosti

- oblik naredbe:

```
case value in
    pat_1) naredba_1
           naredba_2
           ...
           naredba_k;; #PAZI! ;;
    pat_2) blok_naredbi_2;;
    ...
    pat_n) blok_naredbi_n;;
esac
```



Naredba case (2)

● primjer:

```
char=$1
case "$char" in
    [0-9] ) echo "znamenka";;
    [a-z] ) echo "malo slovo";;
    [A-Z] ) echo "veliko slovo";;
    ?      ) echo "specijalni znak";;
    *      ) echo "molim upisite jedan znak";;
esac
```



Naredba for

● oblik naredbe:

```
for var in rijec_1 rijec_2 ... rijec_n
do
    naredba_1
    ...
    naredba_k
done
```

● primjer:

```
for i in 1 2 3
do
    echo $i
done
```



Naredba `for` (2)

- primjer – širenje imena datoteka:

```
for i in *.sh
do
    echo $i
done
```

- primjer – korištenje argumenata:

```
for arg in "$@"; do echo $arg; done
```

- *napomena*: oblik `for arg in "$@"` može se kraće napisati i kao `for arg` – podrazumijeva se lista argumenata !

```
for arg; do echo $arg; done
```

- primjer – učitavanje liste iz datoteke:

```
for file in $(cat lista.txt); do echo $file; done
```



Naredba while

- oblik naredbe:

```
while test_uvjet
do
    naredba_1
    ...
    naredba_k
done
```

- primjer:

```
i=1
while [ "$i" -le 5 ]
do
    echo $i
    i=$((i + 1))
done
```



Naredba while (2)

● primjer:

```
while [ -n "$1" ]  
do  
    echo $1  
    shift  
done
```

● primjer:

```
while [ "$#" -ne 0 ]  
do  
    echo $1  
    shift  
done
```




Naredba `until`

- oblik naredbe:

```
until test_uvjet
do
    naredba_1
    ...
    naredba_k
done
```

- primjer:

```
i=1
until [ "$i" -gt 5 ]
do
    echo $i
    i=$((i + 1))
done
```



Naredba until (2)

● primjer:

```
$ cat mon
# Ceka da se zadani korisnik prijavi na sustav

if [ "$#" -ne 1 ] ; then
    echo "Usage: mon user"
    exit 1
fi

user="$1"

# Provjera svake minute: je li se korisnik prijavio ?
until who | grep "^$user " > /dev/null ; do
    sleep 60
done

# Korisnik je docekan :-)
echo "$user has logged on"
```



Naredba break

- naredba `break` trenutno prekida izvršavanje petlje
- naredba `break n` trenutno prekida izvršavanje `n` unutrašnjih petlji
- primjer:

```
i=1
while true
do
    echo $i
    i=$((i + 1))
    if [ "$i" -gt 20 ]; then
        break
    fi
done
```



Naredba `continue`

- naredba `continue` izaziva preskakanje svih preostalih naredbi unutar petlje i nastavljjanje sljedećeg prolaza kroz petlju
- naredba `continue n` izaziva preskakanje preostalih naredbi unutar `n` unutrašnjih petlji uz normalan nastavak izvršavanja petlji
- primjer:

```
for file; do
    if [ ! -e "$file" ]; then
        echo "$file not found!"
        continue
    fi
    #
    # Process the file
    #
done
```



Redirekcija u petlji

- izlaz/ulaz cijele petlje može se preusmjeriti
- ulaz preusmjeren u petlju primjenjuje se na sve naredbe koje podatke čitaju sa standardnog ulaza
- izlaz preusmjeren iz petlje primjenjuje se na sve naredbe koje podatke zapisuju na standardni izlaz
- primjer:

```
for i in 1 2 3 4; do
    echo $i
done > izlaz.txt
```

- primjer:

```
for i in 1 2 3 4; do
    head -n 3 # cita iz "lista.txt"
    echo "prolaz: $i "
done < lista.txt
```



Naredba `getopts`

- prilikom navođenja argumenata koje prenosimo skripti ponekad bismo željeli navesti opcije u stilu Unix naredbi
(`head -n 3 lista.txt`)
- naredba `getopts` olakšava preuzimanje opcija iz naredbenog retka
- oblik naredbe:
`getopts` `opcije` `varijabla`
- naredba je prilagođena izvršavanju unutar petlje
 - svakim pozivom `getopts` ispituje sljedeći argument naredbenog retka i provjerava je li to valjana opcija – počinje li znakom "-" nakon kojeg slijedi slovo navedeno među opcijama
 - ako je riječ o ispravnoj opciji, `getopts` pohranjuje slovo u navedenu varijablu i vraća izlazni status 0
 - ako opcija zahtijeva dodatni argument, nakon odgovarajućeg slova navodi se ":", a argument se vraća u varijabli `$OPTARG`



Naredba `getopts` (2)

- ponašanje naredbe u slučaju pogrešne opcije ili kraja liste opcija
 - ako slovo nakon znaka "`-`" nije navedeno u opcijama, u navedenu varijablu se pohranjuje "`?`", ispisuje se poruka o grešci na `stderr` i vraća izlazni status 0
 - kada se stigne do kraja naredbenog retka ili kad sljedeći argument ne započinje znakom "`-`", `getopts` vraća izlazni status različit od 0
 - varijabla `$OPTARG` sadrži indeks sljedećeg argumenta koji treba obraditi – kad je završeno čitanje opcija, to je indeks prvog argumenta *nakon* opcija

- primjer – fragment kôda za učitavanje opcija neke skripte:

```
# postavljanje default vrijednosti
mailopt=FALSE
interval=60
```

```
# ... nastavak na sljedećem slajdu
```



Naredba getopts (3)

● primjer (nastavak)

```
while getopts mt: option; do
    case "$option" in
        m) mailopt=TRUE;;
        t) interval=$OPTARG;;
        \?) echo "Usage: mon [-m] [-t n] user"
            exit 1;;
    esac
done

# treba jos i procitati argument user
if [ "$OPTARG" -gt "$#" ] ; then
    echo "Missing user name!" ; exit 2
fi
#
shiftcount=$((OPTARG - 1))
shift $shiftcount
user=$1
```


Učitavanje i ispis

● naredba `read`:

`read` `lista_varijabli`

- ljuska čita redak sa standardnog ulaza, pohranjuje prvu riječ u prvu varijablu navedenu u listi, drugu riječ u drugu varijablu itd.
- ako je broj navedenih varijabli manji od broja riječi u retku, ostatak retka se pohranjuje u zadnjoj navedenoj varijabli:

`read x y`

učitava prvu riječ u `x` i ostatak retka u `y`

- učitavanje cijelog retka o navedenu varijablu:

`read mojRedak`

- `read` vraća izlazni status 0 osim kada stigne do kraja datoteke (EOF) – ako se podaci učitavaju s terminala, za označavanje kraja unosa treba utipkati `Ctrl+D`



Učitavanje i ispis (2)

- primjer: uzastopno učitavanje 2 broja i ispis njihovog zbroja:

```
while read n1 n2; do echo $((n1 + n2)); done
```

- primjer:

- imamo 2 datoteke s podacima o nekim korisnicima: prva sadrži specifične podatke jedne podgrupe korisnika, a druga opće podatke svih korisnika
- zapis o korisniku je pohranjen kao redak tekstne datoteke, a u obje datoteke koristi se jedinstveni identifikator korisnika sastavljen od 10 znamenaka
- želimo izvući podatke iz druge datoteke za sve korisnike iz prve datoteke

```
while read Redak; do
  mbr=$(echo $Redak|sed -r 's/.*([0-9]{10}).*/\1/')
  grep $mbr svi.txt >> G3.txt || echo "-- $mbr">> G3.txt
done < "popis_G3.csv"
```



Učitavanje i ispis (3)

- naredba `echo` je najčešće prikladna za većinu ispisa, no nekad je potreban *formatirani* ispis
- naredba `printf` slična je istoimenoj naredbi programskog jezika C:

printf "format" arg1 arg2 ...

- znakovi niza `format` ako nisu predznačeni znakom "%" prosljeđuju se na standardni izlaz
- znakovi predznačeni znakom "%" označavaju format u kojem se ispisuju navedeni argumenti
- značenja nekih oznaka:
 - %d : cijeli broj
 - %u : cijeli broj bez predznaka
 - %o : oktalni broj
 - %X : heksadekadski broj
 - %c : znak
 - %s : niz znakova



Učitavanje i ispis (4)

● primjeri:

```
$ printf "Oktalna vrijednost broja \
%d je %o\n" 20 20
Oktalna vrijednost broja 20 je 24
```

```
$ printf "Heksadekadska vrijednost broja \
%d je %X\n" 30 30
Heksadekadska vrijednost broja 30 je 1E
```

```
$ printf "Nepredznacena vrijednost broja \
%d je %u\n" -1000 -1000
Nepredznacena vrijednost broja -1000 \
je 18446744073709550616
```

● primjer:

```
while read broj1 broj2; do
    printf "%12d %12d\n" $broj1 $broj2
done
```