



Operativni sistemi 2

Vežbe 6 BASH Shell

Lista predmeta:

IR smer - 13e113os2@lists.etf.rs

SI Smer - 13s113os2@lists.etf.rs

Uvod

- Shell – tekstualni interfejs korisnik-OS.
- Čita i obrađuje liniju po liniju.
- Linija = niz riječi razmaknutih razmakom ili TAB-om.
- Elementi linije:
 - komande – prva reč u komandi,
 - argumenti,
 - opcije – obično počinju sa '-' i mogu imati svoje argumente.

Izbor bash školjke

- Komanda za provjeru podrazumjevanе školjke:
echo \$SHELL
ispis bi trebao da sadrži string "bash"
- Za promjenu shell-a:
chsh putanja_do_shella
- Najčešće putanje za bash shell:
 - /bin/bash
 - /usr/local/bin/bash

Fajl sistem u Linux-u

- Sve počinje od korjenog direktorijuma - /
- Korjeni direktorijum sadrži:
 - Regularne fajlove,
 - Izvršne fajlove,
 - Druge direktorijume - koji mogu da sadrže isto što i root.
- Putanje se mogu zadati:
 - apsolutno – počinju sa /
 - relativno – počinju nazivom direktorijuma ili sadrže samo naziv fajla.
- Direktorijumi se u putanji odvajaju kosom crtom '/'.

Radna putanja

- Radna putanja + relativna putanja = apsolutna putanja.
- Može se dobiti pokretanjem komande **pwd**.
- Menja se komandom:
cd putanja_do_novog_direktorijuma
 - putanja do novog direktorijuma može biti zadata relativno ili apsolutno
 - direktorijum iznad se označava sa ..
 - tekući direktorijum se označava sa .
 - poziv komande bez argumenata vraća korisnika u njegov home direktorijum.
 - prosleđivanjem znaka '-' se poništava poslednja promjena direktorijuma
=> dvostruka primjena nema efekta na tekući direktorijum.

Home direktorijum

- Svaki korisnik ima svoj polazni direktorijum.
- Svi direktorijumi korisnika su najčešće smješteni u jedan zajednički direktorijum:
 - /home – direktorijum u kojem su najčešće direktorijumi ostalih korisnika,
 - /home/OS2 – home direktorijum korisnika OS2
 - /home/OS1 – home direktorijum korisnika OS1
- Ako je ulogovan korisnik OS2:
 - /home/OS2/dir možemo zamjeniti sa ~/dir
 - /home/OS1/dir možemo zamjeniti sa ~/../OS1/dir
- Skraćeni zapis je posebno pogodan u slučaju da ne znamo gdje se nalaze home direktorijumi.

Listanje sadržaja direktorijuma

- **ls**
- Poziv bez opcija i argumenata ispisuje sve fajlove.
- Najčešće korišćene opcije:
 - -l – (long) prikazuje dodatne informacije o fajlu: prava pristupa, vrste fajl, vrijeme kreiranja...
 - -a – (all) lista i skrivene fajlove (fajlovi čiji naziv počinje tačkom).
 - -s – (size) štampa veličinu fajlova u blokovima
 - -h – (human-readable) prikazuje veličine u bajtovima
 - -R – (recursive) rekurzivno listanje poddirektorijuma
 - --help – za ostale opcije
- Ako se kao argument zada direktorijum, štampa se sadržaj zadatog direktorijuma.
- Ako se zada šablon, štampa se sve što može da se upari sa šablonom => ako se upari direktorijum, štampa se njegov sadržaj.

Listanje direktorijuma – opcija -l

```
$ls -l
total 0
drwxr-xr-x 2 stanislav stanislav 48 2007-05-26 12:07 dwiki
-rw-r--r-- 1 stanislav stanislav 0 2007-05-26 12:07 wiki
-rw-r--r-- 1 stanislav stanislav 0 2007-05-26 12:07 wiki1
^***** ^ ***** ^^^^^^^* ^^^^^^^^^^^^^^^^^ *****
```

- | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----|---|-----|---------------|-----|-------------------|-----|--|
| 1. | Tip fajla: | "d" | direktorijum, | "c" | <i>char dev</i> , | "b" | <i>block dev</i> , "-" običan fajl. |
| 2. | Dozvole: | "r" | čitanje, | "w" | pisanje, | "x" | izvršavanje, "-" ukinuta dozvola. Dozvole idu redom, po tri dozvole za vlasnika, vlasničku grupu i sve ostale. |
| 3. | Broj čvrstih veza ka tom fajlu. Svaki običan fajl ima makar jednu referencu na samog sebe, a direktorijum dve, sam direktorijum i "." fajl u njemu. | | | | | | |
| 4. | Vlasnik fajla. | | | | | | |
| 5. | Vlasnička grupa. | | | | | | |
| 6. | Veličina fajla. | | | | | | |
| 7. | Vreme poslednje promene. | | | | | | |
| 8. | Ime fajla. | | | | | | |

Džokeri i šabloni

- Menjaju više naziva fajlova i direktorijuma.
- ? – ekspanduje se u bilo koji znak.
- * – ekspanduje se u 0 ili više proizvoljnih znakova.
- **[skup_slova]** – ekspanduje se u jedan znak iz skupa.
- **[!skup_slova]** – ekspanduje se u jedan znak koji nije u skupu.
- Primjeri korišćenja:
 - **ls p*** - ispisuje sve fajlove u tekućem direktorijumu čiji naziv počinje slovom 'p',
 - **ls [a-c]*** - ispisuje sve fajlove čiji naziv počinje nekim od slova a, b ili c
- Napomena:
 - Zadati izraz se ekspanduje, ako je moguće, i komanda se poziva sa ekspandovanim vrednostima.
U suprotnom, prosleđuje se izraz, što može dovesti do greške.

Vitičaste zagrade

- Ekspanduju se u jedan od proizvoljno zadatih stringova odvojenih zapetama.
- $a\{b,c,d\}e$ se ekspanduje u:
abe ace ade
- $a\{b,c\{d..f\}\}g$ se ekspanduje:
abg acdg aceg acfg
- Može se koristiti i pri uparivanju naziva fajlova.

Kreiranje i brisanje direktorijuma

- Komanda za kreiranje:
mkdir naziv_direktorijuma
 - Ako se navede putanja, svi direktorijumi prije poslednjeg moraju postojati.
 - Direktorijum koji se kreira ne sme postojati.
- Komanda za brisanje:
rmdir naziv_direktorijuma
 - Direktorijum mora biti prazan.
 - U suprotnom koristiti **rm -r**.

Osnovne komande za rad sa fajlovima

- **cp** – kopiranje (ako postoji istoimeni fajl, briše ga!):
 - `cp stariFajl noviFajl`
 - `cp -r stariDir noviDir`
 - `cp fajl1 fajl2 fajl3 dir`
- **mv** – premještanje zadatog fajla na novu putanju:
 - `mv starolme novolme`
- **rm** – brisanje zadatog(ih) fajla(ova):
 - `rm fajl1 fajl2 fajl3`
 - `rm -r dir`
- Opcija `-r` se navodi za direktorijume kako bi se obradili rekurzivno.

Prava pristupa fajlovima

- Linuks definiše prava za 3 kategorije:
 - vlasnik
 - grupa
 - ostali
- Za svaku kategoriju definiše po tri bita rwx:
 - r – pravo čitanja
 - w – pravo upisa
 - x – pravo izvršavanja.
- Tekuća prava pristupa se dobijaju komandom ls -l:
- rw- r-- r-- ...(ostatak linije)
uuugggooo

Izmena prava pristupa

- Komanda:
chmod mod fajl1 {fajl2 ...}
- Nove dozvole se mogu zadati:
 - fiksno - tri oktalna broja, npr 641
 - relativno - navodi se onaj kome se mjenja pravo pristupa [ugoa], potom [+ -] u zavisnosti da li se pravo dodjeljuje ili oduzima i na kraju prava koja se dodjeljuju/oduzimaju [r][w][x]. Primjeri:
 - u+r # vlasnik dobija pravo čitanja
 - gu-w # grupi i vlasniku se oduzima pravo pristupa
 - u+r, o-w #vlasniku se dozvoljava čitanje i ostalima zabranjuje čitanje.

Traženje fajla

- Komanda find kojoj se prosleđuje:
 - putanja od koje se kreće u potragu, i
 - opis onoga što se traži.
- Najčešći oblici:
 - `find dir -name "naziv"` - traži fajl zadatog naziva (case sensitive),
 - `find dir -iname "naziv"` - kao prethodno, osim case insensitive,
 - `find dir -mtime n` - traži se fajl koji je modifikovan pre n dana (navodi se posle jedne od prethodne dve opcije).
Ispred n može doći
 - '-' - od poslednje izmjene proteklo je manje od n dana;
 - '+' - od poslednje izmjene proteklo je više od n dana.

Poređenje fajlova i direktorijuma

- Komanda:
diff [opcije] fajl1 fajl2
- Opcije mogu da budu:
 - -q - u slučaju razlikovanja, samo ispisati da se razlikuju (ne prikazivati razlike).
 - -s - kao prethodno, samo u slučaju poklapanja.
 - -r - ako se želi rekurzivno poređenje sadržaja direktorijuma.
 - -y - prikazati fajlove jedan pored drugog.
- Rezultat komande:
 - 0 - sadržaji su isti
 - 1 - sadržaji su različiti
 - 2 - došlo je do greške

Komunikacija sa programima

- Svaki program ima:
 - standardni ulaz
 - standardni izlaz
 - standardni izlaz za greške
- Preusmjeravanje standardnog ulaza/izlaza:
 - na neki fajl,
 - na standardni izlaz/ulaz nekog drugog programa.

Preusmjeravanje na fajlove

- Navodi se na kraju komande za pokretanje programa.
- Preusmjeravanje:
 - `> izlazni_fajl` – standardni izlaz se preusmjerava u fajl `izlazni_fajl`.
 - `< ulazni_fajl` – preusmjerava standardni ulaz.
 - `2> error_fajl` – preusmjerava standardni izlaz za greške.
 - `>> izlazni_fajl` – Koristi se kao `>`, osim što se vrši dopisivanje na kraj postojećeg fajla.
 - `2>&1` – preusmjerava standardni izlaz za greške na standardni izlaz.
 - `&> izlazni_fajl` – preusmjerava standardni izlaz i standardni izlaz za greške u `izlazni_fajl`

Nadovezivanje procesa cevima

- `prog1 <ulaz | prog2 | prog3 | ... | progN >izlaz`
- Standardni ulaz prog1 se preusmjerava na fajl ulaz.
- Standardni izlaz prog1 se kroz cev preusmjerava na standardni ulaz prog2.
- Standardni izlaz prog[i] se preusmjerava na standardni ulaz prog[i+1], za $i=1..N-1$.
- Standardni izlaz progN se preusmjerava u fajl izlaz.
- Programi se izvršavaju u paralelnim procesima.
- Primjer ispisa sadržaja direktorijuma ekran po ekran:

`ls | more`

Komande more i less

- Ako je izlaz predugačak da stane na ekran, preusmjeriti ga na komandu more.
 - Komanda more prikaže prvi ekran izlaza i zatim čeka:
 - Enter - pomjera ispis za jedan red,
 - Space - pomjera ispis za jedan ekran.
- Komanda less je slična komandi more, osim što omogućava kretanje unazad kroz izlaz.
 - strelicama se skroluje sadržaj;
 - taster q za izlazak.

Pokretanje poslova u pozadini

- Za pokretanje u pozadini iza komande navesti znak '&'.
- Po pokretanju ispisuje se linija nalik sledećoj:
[2] 1234
 - broj u uglastim zagradama služi za kasnije referisanje kreiranog procesa.
- Nakon prethodnog ispisa, dobija se odzivni znak.
- Kada se pokrenuta komanda završi, ispisuje se sledeća linija:
[2]+ Done

Poslovi u pozadini

- jobs - komanda za listanje svih poslova; primjer ispisa:
[1]- Running komanda1
[2]- Running komanda2
[3]+ Stopped komanda3
- Posao se vraća u fokus pozivom:
fg %broj_posla
 - %broj_posla se može izostaviti i tada se odnosi na posao sa '+'.
- Moguće je i tekući program poslati u pozadinu:
 - Ctrl+z – zaustavlja program,
 - bg – nastavlja prethodni program u pozadini.
- Važno:
 - Poslovi u pozadini ne bi trebalo da rade sa standardnim ulazom i izlazom => preusmjeriti ih u fajlove.
 - Ako proces nešto treba da pročita sa standardnog ulaza, suspenduje se dok ne dođe u fokus.

Povratna vrednost procesa i kombinovanje komandi

- Proces vraća:
 - `== 0` - ispravno završen posao,
 - `!= 0` - došlo je do greške.
- `komanda1 && komanda2`
 - komanda 2 se izvršava samo ako je komanda1 korektno izvršena.
- `komanda1 || komanda2`
 - komanda2 se izvršava samo ako je izvršavanje komande1 bilo neuspešno.
- `komanda1; komanda2; komanda3`
 - komande se izvršavaju redom, sekvencijalno.

Skripte i komentari

- bash omogućava izvršavanje niza komandi zapisanih u fajl - izvršavanje skripti.
- Fajl koji sadrži skriptu, u prvoj liniji treba da ima putanju i naziv interpretera za tu skriptu:
`#!/bin/bash`
- Može se izvršiti i direktnim pozivom i korišćenjem **source** komande.
- Niz znakova "#!" se naziva sha-bang.
- Druga namjena znaka # je početak komentara do kraja linije.
- Skripta može (treba) da vrati vrednost.
 - koristi se exit komanda;
 - podrazumjevano vraća 0.

Izvršavanje skriptova

- Prva mogućnost - korišćenjem komande source:
source naziv_skript_fajla
 - komande iz skripta se izvršavaju kao da su navedene umjesto poziva source komande
- Druga mogućnost - navođenjem naziva skript fajla:
./naziv_skript_fajla
 - ./ je često neophodno, jer bash podrazumjevano ne traži komandu u radnom direktorijumu.
 - Na ovaj način se pokreće subshell koji je novi proces i u okviru njega se izvršava zadati skript, dok pozivajući šel čeka blokiran dok se pozvani ne završi (ovo ne važi ako je skript pozvan na izvršavanje u pozadini).

Promenljive

- Definišu se implicitno, dodjelom nove vrednosti.
- Tip se određuje na osnovu vrednosti koju trenutno pamti.
- Sadržaj čuvaju kao string.
- Primjer dodjele vrednosti:
PROMJENLJIVA="Pocetni sadrzaj"
 - pre i posle = ne sme biti razmaka.
- Dohvatanje vrednosti:
 - `${PROMJENLJIVA}`
 - skraćeno: `$PROMJENLJIVA`
- Da bi se promjenljiva vidjela u subshell-u, mora se izvesti:
export naziv_promjenljiva
 - Ako promjenljiva nije već definisana, ovako se definiše.

Definisanje nizova

- Primjer:

```
NIZ[1]="Prvi clan niza"
```

```
NIZ[2]="Drugi clan niza"
```

```
echo "${NIZ[2]}"
```

```
echo "${NIZ[*]}"
```

- ekvivalentno: `echo "${NIZ[1]} ${NIZ[2]}"`
- string se formira od elemenata niza razdvojenih prvim znakom u IFS (internal field separator).

```
echo "${NIZ[@]}"
```

- ekvivalentno: `echo "${NIZ[1]}" "${NIZ[2]}"`
- od svakog elementa se formira string.

Primjer razlike upotrebe * i @

```
NIZ[1]=n1
```

```
NIZ[2]=n2
```

```
for i in "a${NIZ[@]}b"
```

```
do
```

```
  echo $i
```

```
done
```

```
Ispis:
```

```
an1
```

```
n2b
```

```
NIZ[1]=n1
```

```
NIZ[2]=n2
```

```
for i in "a${NIZ[*]}b"
```

```
do
```

```
  echo $i
```

```
done
```

```
Ispis:
```

```
an1 n2b
```

Često korišćene predefinisane promjenljive

- PWD - tekući direktorijum.
- IFS - znak kojim se razdvajaju reči.
- PATH - spisak putanja, odvojenih dvotačkama, na kojima se podrazumjevano traže programi za izvršavanje. Za uključivanje tekućeg direktorijuma treba dodati prazan string(dve uzastopne dvotačke).
- PS1 - primarni odzivni (*prompt*) string `\u@\h:\w\$:`
 - `\u` - korisnik
 - `\h` - host
 - `\w` - tekući direktorijum.
- PS2 - sekundarni odzivni string (nastavak prethodne linije).
- `$0, $1, $2, ..., $9, ${10}...` - parametri prosleđeni prilikom poziva skripta.
- `$#` - broj prosleđenih argumenata.
- `$?` - rezultat izvršavanja prethodne komande (rezultat `!=` ispis).

Ispis i navodnici

- Komanda: echo
- Ispis promjenljive:
 - echo \$PWD
 - echo "\$PWD"
- Pod dvostrukim navodnicima, promjenljiva će biti ekspanđovana i takva dalje biti prosleđena.
- Postoji razlika ukoliko promjenljiva sadrži razmake:
 - bez navodnika: uzastopni razmaci se spajaju u jedan,
 - sa navodnicima: zadržavaju se svi razmaci, onako kako su raspoređeni u ulaznom tekstu.

Navodnici

- `echo ej gde si tu sam`
`ej gde si tu sam`
- `echo "ej gde si tu sam"`
`ej gde si tu sam`
- `PROMENLJIVA=" ovo je uvrnuta promenljiva"`
`echo $PROMENLJIVA`
`ovo je uvrnuta promenljiva`
`echo "$PROMENLJIVA"`
`ovo je uvrnuta promenljiva`
- Za ispis '\$' i drugih specijalnih znakova koristi se \ ispred znaka.
- Jednostruki navodnici: sprečavaju ekspanziju promenljivih.

Korišćenje ispisa komande

- Ispis na standardni izlaz neke komande se može dobiti okruživanjem komande sa `$()` ili ```
- Primer: `VAR=`ls -la``
- Promenljiva `VAR` nakon izvršene naredbe ima vrednost onoga što bi komanda `ls -la` ispisala na standardni izlaz
- Ispis na standardni izlaz se ne vrši

Komanda exec

- Komanda exec pokreće kao komandu string koji joj se prosledi kao parametar
- Primer:
 - VAR="ls -l"
 - exec \$VAR
- Dati primer izvršava komandu ls -l i ispisuje njen rezultat na standardni izlaz

Uslovno izvršavanje

```
if komanda1
then
    komanda2
[elif komanda3
    komanda4 ...]
[else
    komanda5]
fi
```

- Komanda u uslovu može biti poziv bilo kog programa ili poziv ugrađene komande test.
- then, else i fi se moraju pisati u novom redu.

test

- Poređenje celih brojeva:

Operator	Značenje	Ekvivalent u C-u
• -lt	(less than)	<
• -gt	(greater than)	>
• -eq	(equal)	==
• -ge	(is greater than or equal to)	>=
• -le	(is less than or equal to)	<=
• -ne	(is not equal to)	!=

test

- Poređenje stringova:
 - -z string je prazan, to jest dužina mu je nula
 - -n string nije prazan, ima pozitivnu dužinu
 - = jednakost string-ova
 - != nejednakost string-ova
 - Važno: promjenljive navesti pod navodnicima ("\$...") i ostaviti razmak između operatora
- Logičke operacije:
 - -a Logičko i
 - -o Logičko ili
 - ! Negacija

test

- Fajlovi:
- -f Постоји фајл и упитању је обичан фајл
- -s Фајл није празан
- -r Фајл је читљив од стране корисника који извршава скрипту
- -w Можемо да уписујемо у фајл
- -x Можемо да извршавамо фајл
- -d Тип фајла је директоријум
- -h Фајл је симболички линк

Primjeri formiranja uslova

- `test -f $1 -a -x $1` - provjerava da li argument 1 predstavlja naziv fajla i da li je taj fajl izvršiv.
- Skraćeni oblik test komande:
`[-f $1 -a -x $1]`
- `[$# -gt 2 -a $# -lt 5]` - provjerava da li je broj prosleđenih parametara veći od 2 i manji od 5.
- `[$0 = $1]` - provjerava da li je prvi parametar kao string jednak nazivu fajla koji sadrži skriptu.

komanda let

- bash ne definiše operatore za rad sa celim brojevima.
- Komanda let prihvata i izračunava izraz (uključujući i dodjelu vrednosti).
- Standardni operatori: ++, --, +, -, !(логичка негација), ~(негација над битовима), ^^ (експонент), *, /, %, <<, >>, ==, !=, &, |, ^, &&, ||

Primjer korišćenja let komande

```
$ var=0
```

```
$ let var++
```

```
$ echo $var
```

```
1
```

```
$ ((var +=var)) #isto kao let var += var ili var=$((var+var))
```

```
$ echo $var
```

```
2
```

```
$ ((var=~1&131)) #primer rada sa bitovima
```

```
$ echo
```

```
$var
```

```
130
```


Primjer if naredbe

```
if test $0 = $1
```

```
then
```

```
echo "Isti."
```

```
else
```

```
echo "Razliciti."
```

```
fi
```

- Drugi način:

```
test $0 = $1 && echo Isti || echo Različiti
```

- ili:

```
[ $0 = $1 ] && echo Isti || echo Različiti
```

For petlja

- ```
for i in * #sve datoteke u dir-u
do
 echo Naziv datoteke je: $i
done
```

- ```
#!/bin/bash
```

```
IFS_old=$IFS
IFS=":"
n=0
for i in $PATH
do
    echo "Dir $n: $i"
    ((n++)) #ekvivalentno let "n++"
done
IFS=$IFS_old
```

while petlja

- while komanda1
do
 komanda2
done
- #!/bin/bash

```
while [ ! -z "$1" ] #mozemo da koristimo i 'until [ -z "$1" ]'  
do  
    echo -n "$1 " # -n = ispis bez prelaska u novi red  
    shift  
done  
echo  
exit 0
```

- shift vrši pomjeranje argumenata u parametrima u levo:
\$1 <--- \$2, \$2 <--- \$3, \$3 <--- \$4, itd. (\$0 ostaje neizmjenjen)

Case naredba

- ```
case rec in
 uzorak1)
 komanda1
 komanda2
 ;;
 uzorak2)
 komanda3
 ;;
 *) #poklapa se sa bilo kojim uzorkom
 komanda4
 ;;
esac
```

- Uparuje se prvi uzorak na koji se naiđe od početka.
- U svakom uzorku može biti i više različitih vrednosti razdvojenih uspravnom crtom - dovoljno je da se rec upari sa jednom vrednošću.
- Moguće koristiti i džokere.

# cat i echo

- cat
  - Spaja zadate fajlove na standardni izlaz.
  - Ako se pozove bez ulaznih fajlova, čita standardni ulaz.
    - Za kraj fajla koristi se Ctrl+D.
  - Opcija -n numeriče sve neprazne linije.
- echo
  - Ispisuje zadatu vrednost na standardni izlaz i prelazi u novi red.
  - -n - opcija koja isključuje prelazak u novi red.

# sort

- `sort {opcije...} {fajl1, ...}`
- Sortira linije iz nadovezanih ulaznih fajlova.
  - bez opcije sortira po alfabetu;
  - `-b` - preskače vodeće blanko znakove;
  - `-n` - sortira po brojevima na početku linije;  
u nekim slučajevima daje različit izlaz od varijante bez opcija; primjer:

| ulaza | bez -n | sa -n |
|-------|--------|-------|
| 20p   | 1"     | 1"    |
| 1"    | 20p    | 2k    |
| 2k    | 2k     | 20p   |

# wc i uniq

- wc - broji bajtove, reči i linije u ulaznom fajlu.
  - bez imena fajla koristi standardni ulaz (Ctrl+d EOF)
  - podrazumjevano broji reči;
  - -l - za brojanje linija;
  - -c - za brojanje bajtova.
- uniq - omogućava izdvajanje jedinstvenih i dupliranih linija iz fajla.
  - izbacuje duplirane linije (ostavlja jednu liniju, ostale odbacuje);
  - -d - izdvaja sve one linije koje se pojavljuju više puta;
  - -u - izdvaja samo one linije koje su unikatne;
  - proverava se radi samo lokalno (porede se susedne linije).

# grep

- `grep {opcija ...} šablon {fajl ...}`
  - ako nema fajlova, koristi standardni ulaz;
  - traži šablon u ulaznim fajlovima, ispisuje linije u kojima nađe šablon (ukoliko je deo linije uparen sa regularnim izrazom).
  - -i - ignorisanje veličine slova.
  - -v - Štampa one koji se ne uklapaju sa šablonom;
  - -n - štampa i broj linije u kojoj je nađen (redni broj prve linije fajla je 1).
  - -H - štampa i naziv fajla u kojem je šablon nađen.
- Šabloni - regularni izrazi.



# Regularni izrazi

- `.` Menja bilo koji karakter (kao `?`).
- `[...]` Menja bilo koji karakter iz navedenog skupa (`[abc]` menja karaktere `a`, `b` i `c`).
- `[^...]` Menja bilo koji karakter koji nije u skupu.
- `*` Menja prethodni karakter 0 ili više puta.
- `{n}` Menja prethodni karakter `n` puta.
- `{n,m}` Menja prethodni karakter najmanje `n` a najviše `m` puta.
- `{n,}` Ponavlja prethodni karakter `n` ili više puta.
- `^` Označava početak reda.
- `$` Označava kraj reda.
- `( \ )` Zagrade za grupisanje.
  - za ponavljanje regularnog podizraza;
  - za kasnije referisanje dela regularnog izraza.

# grep primeri

- `grep "$tekst$"`
  - izdvaja redove koji se završavaju istim tekstom kao što je sadržaj promjenljive tekst.
- `grep "^$1.*$2$"`
  - izdvaja linije čiji je početak jednak prvom parametru skripta i čiji je kraj jednak drugom prosleđenom parametru.

# tr

- *tr [-c] [-d] [-s] [string1] [string2]*
- Ulazni tok prosleđuje na izlaz uz zamenu zankova koji se pojavljuju u string1 odgovarajućim znakovima iz string2.
  - znakovi iz string1 i string2 se uparuju po redosledu, prvi sa prvim, drugi sa drugim itd.
- Opcije:
  - -c - umjesto string1 posmatra komplementaran skup;
  - -d - navodi se samo string1; brišu se znakovi koji se nalaze u string1;
  - -s - nakon standardne zamene sva uzastopna pojavljivanja bilo kog znaka iz string2 zamenjuje tim znakom (npr. aaa=>a).
- string2 kraći od string1 => string2 se proširuje poslednjim znakom.

# tr - primeri

- `echo "aaabacd" | tr "a-c" "o-q"`  
ooopoqd
- `echo "aaabacd" | tr "a-c" "o-z"`  
ooopoqd
- `echo "aaabacd" | tr "a-c" "o-p"`  
ooopopd
- `echo "aaabacd" | tr -d "a-c"`  
d
- `echo "aaabacd" | tr -cd "a-c"`  
aaabac
- `echo "aaabacpdd" | tr -s "a-c" "pqp"`  
pqpdd

# cut

- **cut** {*OPTION* ...} {*FILE* ...}
- Na standardni izlaz ispisuje zadate delove linija sa standardnog ulaza (ili fajlova).
- Opcije:
  - -c CHARACTER-LIST
    - ispisuje samo znakove na pozicijama zadate listom;
  - -f FIELD-LIST
    - ispisuje samo polja na pozicijama zadata listom;
  - -d znak - znak kojim se u slučaju opcije -f izdvajaju polja;
    - podrazumevano TAB (\t)
    - dva uzastopna delimitera => prazno polje između.
  - -s - ne štampati linije bez ijednog delimitera;
  - lista se navodi kao niz brojeva i/ili opsega u rastućem poretku.
  - znakovi i polja se numerišu počev od 1.

# cut - primer

- Ispisivanje prava pristupa i naziva fajlova korišćenjem `ls -l`:  
`ls -l | tr -s " " " " | cut -c 2- | cut -d\ -f 1,8`
- Primer ulazne linije:  
`-rwxr-xr-x 1 etf etf 44 2010-11-29 00:07 sk`
- dobija se izlaz:  
`rwxr-xr-x sk`

# Sed

- `sed 's/regexp/replacement/'`
- Uparuje deo ulazne linije sa zadatim regularnim izrazom;
  - u jednoj liniji samo jednom vrši uparivanje;
  - svaki upareni deo linije menja stringom za zamenu;
  - za uparivanje čitave linije koristiti oznake za početak i kraj reda.
- Neuparene linije i delovi linija se prepisuju na izlaz.
- Regularni izraz je isti kao kod grep.
- `&` - predstavlja upareni string – može se navesti u stringu za zamenu

```
echo "123 abc" | sed 's/[0-9]*/& &/'
```

Rezultat: 123 123 abc #abc je neupareno

# Sed – dohvatanje delova uparenog stringa

- Delovi izraza u zagradama mogu se referisati u stringu za zamenu - `\(... \)`      Zagrade za grupisanje.
- Referišu se po rednom broju zagrada od početka regularnog izraza navedenim iza `'\'` (prve zagrade imaju redni broj 1).
- Primer: `echo abcd123 | sed 's/\([a-z]*\).*\1/'`  
Rezultat: `abcd`
- `sed 's/\([a-z]*\) \([a-z]*\)\/\2 \1/'` #menja redosled dve reci
- Takođe `\1` se može naći i u šablonu, npr.  
`sed 's/\([a-z]*\) \1\1/'` - eliminiše duplikate



# sed - primeri

- Izdvajanje prava pristupa i naziva fajlova uz zamenu redosleda:

```
ls -l | sed 's/[a-z|-]\([a-z|-]*\) \. *\ \([^\]*\)/\2 \1/'
```

- Malo čitljivije - separator ':' umesto '/'

```
ls -l | sed 's:[a-z|-]\([a-z|-]*\) \. *\ \([^\]*\):\2 \1:'
```

[a-z|-] - bilo koji znak iz skupa a-z ili -

\ - predstavlja razmak

.\* - bilo koji znak 0 ili više puta (zamenjuje sve između stringova od interesa)

[^\ ] - bilo koji znak osim razmaka

# sed - primeri

- ako ls -l ispisuje:

```
drwxr-xr-x 3 etf etf 4096 2010-11-25 01:41 ff
```

```
-rw-r--r-- 1 etf etf 25 2010-11-29 21:28 reci.txt
```

```
-rwxr-xr-x 1 etf etf 198 2010-11-30 09:07 sk
```

- Rezultat je:

```
ff rwxr-xr-x
```

```
reci.txt rw-r--r--
```

```
sk rwxr-xr-x
```

# sed - primeri

- Modifikacija koja uparuje samo fajlove sa ekstenzijom :

```
ls -l | sed 's/[a-z\-\]\([a-z\-\]^*\)\ \.*\ \([^\]*\)\.[a-z]\{1,3\}\)/\2 \1/'
```

- ako ls -l ispisuje:

```
-rwxr-xr-x 1 notroot notroot 355 2011-12-05 17:16 skript
-rw-r--r-- 1 notroot notroot 14 2011-12-05 16:58 test1.txt
-rw-r--r-- 1 notroot notroot 20 2011-11-18 19:06 test2.txt
```

Rezultat je:

```
-rwxr-xr-x 1 notroot notroot 355 2011-12-05 17:16 skript
test1.txt rw-r--r--
test2.txt rw-r--r--
```

# sed - primeri

- Modifikacija koja uparuje samo nazive fajlova kreće od 4 znaka:

```
ls -l | sed 's/[a-z|-]\([a-z|-]*\) \. *\ \([^\]\{1,3\}\)\$/\2 \1/'
```

- ako ls -l ispisuje:

```
drwxr-xr-x 3 etf etf 4096 2010-11-25 01:41 ff
-rw-r--r-- 1 etf etf 25 2010-11-29 21:28 reci.txt
-rwxr-xr-x 1 etf etf 198 2010-11-30 09:07 sk
```

- Rezultat je:

```
ff rwxr-xr-x
-rw-r--r-- 1 etf etf 25 2010-11-29 21:28 reci.txt
sk rwxr-xr-x
```

# Zadatak 1

Napisati *bash shell* skriptu koja prihvata nazive dve datoteke. Prva datoteka sadrži reči koje se traže u drugoj datoteci. Potrebno je iz druge datoteke izdvojiti sve one redove koji počinju s nekom od reči iz prve datoteke i ispisati ih na standardnom izlazu.

```
#!/bin/bash
for i in $(cat $1)
do
 cat $2 | grep "^$i.*"
done
```

# Zadatak 2

Napisati bash shell skript koji kao parametre prihvata nazive programa koje treba da pokrene, tako da izlaz prvog preusmeri kroz neimenovanu cev na ulaz drugog, izlaz drugog na isti način na ulaz trećeg i tako sve do poslednjeg programa u listi, čiji izlaz ostaje standardni izlaz. Ukoliko se uoči da neki od programa iz bilo kog razloga nije moguće izvršiti, ispisati poruku o grešci i prekinuti izvršavanje. Grešku treba prijaviti i u slučaju da skriptu nije prosleđen nijedan argument.

```
#!/bin/bash
if test $# -eq 0
then
 echo "Nije prosledjen nijedan parametar"
 exit 1
fi
while test $# -gt 0
do
 if test -f $1 -a -x $1
 then
 if test -z $KOMANDA
 then
 KOMANDA="./$1"
 else
 KOMANDA="$KOMANDA | ./$1" #create command
string
 fi
 fi
done
```



```
else
 echo "Fajl $1 ne postoji ili nemate dozvolu za
izvrsavanje"
 exit 2
fi
shift
done
exec $KOMANDA # execute
```

# Zadatak 3

Napisati *Bash shell script* koji treba da za zadati spisak fajlova prebroji koliko od tih fajlova postoji u tekućem direktorijumu. Spisak fajlova se zadaje kao niz parametara skripta. Izlaz skripta treba da bude u formatu: <broj nadjenih fajlova>/<broj fajlova na spisku>.

```
#!/bin/bash
ukupno=0
postoji=0
while [$# -ge 1]
do
 let ukupno++
 if [-f $1]
 then
 let postoji++
 fi
 shift
done
echo "$postoji/$ukupno"
```

# Zadatak 4

Napisati *bash shell* skript koji treba da sortira spisak studenata po prezimenu i imenu. Spisak studenata je dat u tekstualnom fajlu čiji se naziv prosleđuje kao prvi parametar skripta. Sortiran spisak ispisati na standardni izlaz. U ulaznom fajlu, svaki student je zapisan u po jednom redu po sledećem formatu: redni broj, broj indeksa, ime, prezime i ocena. Sva polja su razdvojena razmakom. Na izlazu za svakog studenta zapisati po jedan zapis po sledećem formatu: redni broj, redni broj iz ulaznog fajla, broj indeksa, prezime, ime i ocena. Smatrati da je ulazni fajl ispravan ukoliko postoji. U slučaju pogrešnog broja parametara, nepostojanja ulaznog fajla ili nemogućnosti čitanja istog ispisati poruku o grešci i prekinuti izvršavanje skripta.

Primer ulaza: 1. 123/2011 Mika Peric 10

```
#!/bin/bash
if [$# -eq 1]
then
 if [-f $1 -a -r $1]
 then
 cat $1 |
 sed "s/\([0-9]*\.\)\ \([0-9]*\[/[0-9]*\)\ \([a-
z,A-Z]*\)\ \([a-z,A-Z]*\)\ \([0-9]*\)/\4 \3 \1 \2
\5/" | sort | #sort prezime ime ...
 grep -n ".*" | #print line num. 1:...
 sed "s/\([0-9]*\):\([a-z,A-Z]*\)\ \([a-z,A-Z]*\)\
\([0-9]*\.\)\ \([0-9]*\[/[0-9]*\)\ \([0-9]*\)/\1. \4
\5 \2 \3 \6/"
 else
 echo "Greska: ulazni fajl ili ne postoji ili mu
se ne moze pristupiti";
 fi
else
 echo "Pogresan broj parametara. $#";
fi
```

# Zadatak 5

Napisati *shell script* koji treba da pređe u direktorijum zadat kao prvi argument, pozove program zadat kao drugi argument prosledivši mu sve ostale argumente kao njegove argumente, ispiše da li se pokrenut program uspešno izvršio i na kraju se vrati u polazni direktorijum. Ako nije prosleđeno dovoljno parametara ispisati odgovarajuću poruku.

```
#!/bin/bash
if [$# -ge 2]
then
 tempdir=$PWD
 cd $1
 prog=$2
 shift
 shift
 exec $prog $@
 if $? #rezultat $prog $@
 then
 echo "Program uspesno izvršen"
 else
 echo "Program nije uspesno izvršen."
 fi
 cd $tempdir
else
 echo "Nije prosledjeno dovoljno parametara"
fi
```

# Zadatak 6

Napisati *shell script* koji treba da iz datoteke, zadate kao prvi parametar, izvuče one zapise o korisnicima čija šifra počinje onim što je zadato kao drugi parametar i koji su iz mesta zadanog trećim parametrom (izdvajaju se samo oni kod kojih su oba uslova zadovoljena). Ulazna datoteka sadrži zapise o korisnicima, i to u svakom redu za tačno jednog korisnika. Pojedinačne informacije u svakom redu su razdvojene tabulatorima. Podaci koji se vode o svakom korisniku su šifra (niz cifara), ime, ime oca, prezime, mesto i broj telefona. Telefon je zapisan u formatu *+<pozivni broj države><blanko znak><pozivni kod grada><blanko znak><broj telefona>*. U izlaznoj datoteci u svakom redu treba da budu informacije o korisnicima, ali u sledećem obliku i redosledu: početno slovo mesta za kojim sledi šifra (bez razmaka između), ime, prvo slovo očevog imena, prezime i broj telefona bez pozivnih brojeva. Prikazati kako se taj skript poziva ako želimo sve korisnike iz datoteke *ulaz.txt* čija šifra počinje sa 4 i koji su iz Beograda.



```
#!/bin/bash
```

```
cat $1|grep "^$2[0-9]*.*$3+[0-9]*\ [0-9]*\ [0-9]*"|
```

```
sed "s/\([0-9]*\) \t\(.*\)\ \t\(.\) .*\t\(.*\)\ \t\(.\) .*\t+[0-9]*\ [0-9]*\ \([0-9]*\)/\5\1 \2 \3 \4 \6/"
```

# Zadatak 7

Napisati *shell script* koji treba da spoji fajlove čiji su nazivi zadati u komandnoj liniji. Skript prihvata do 8 parametara, ali najmanje 3, od čega je prvi izlazni, a ostali ulazni. Može se desiti da neki od zadatah ulaznih fajlova ne postoje, ili da čitanje tih fajlova nije dozvoljeno, pa je potrebno izvršiti odgovarajuće provere. Spajanje se vrši samo u slučaju da postoje bar tri ulazna fajla koja se mogu spojiti. U slučaju da neki uslov nije ispunjen, odštampati odgovarajuću poruku.

```
#!/bin/bash
if [$# -gt 8 -o $# -lt 3]
then
 echo Prosledjeno je vise od 8 ili manje od 3
parametra.
else
 s=0
 izlaz=$1
 shift
 for i in $*
 do
 if [-f $i -a -r $i]
 then
 let "s++"
 fi
 done
```

```
if [$s -gt 2]
then
 ulaz=""
 while [! -z "$1"]
 do
 if [-f $1 -a -r $1]
 then
 ulaz="$ulaz $1"
 fi
 shift
 done
 cat $ulaz >$izlaz
else
 echo Broj fajlova je $s, sto je manje od 3.
fi
echo $*
fi
```