Operativni sistemi I

Vežbe 5

ALATI ZA SHELL PROGRAMIRANJE

Primer najjednostavnijeg skripta

Rad sa skriptovima se svodi na 3 koraka, i to:

1. Formiranje samog skripta pomoću tekst editora ili iz komandne linije redirekcijom standardnog ulaza pomoću komande **cat**. Skript **ss1.sh** (upisan iz **cat**-a u fajl **ss1.sh**)

```
  $ cat > ss1.sh
  #
  # ss1.sh: jednostavan shell program
  #
  clear
  echo "Hello, World!"
<CTRL-D>
```

2. Davanje korisnicima x (execute) dozvole nad datotekom

   ```
   $ chmod +x ss1.sh
   ```

   Pitanje: Kojim korisničkim kategorijama je data dozvola x?

3. Pokretanje

   $ ./ss1

   Skript briše ekran (komanda clear), a zatim na ekranu ispisuje poruku Hello, World! Sav tekst u liniji iza znaka # se smatra komentarom.

   Pitanje: Kako izbeći ./ ?

Prilikom pokretanja skripta može se specificirati komandni interpreter u kome će se program izvršavati. Potrebno je u prvu liniju skripta upisati sledeće:

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

Ukoliko se komandni interpreter ne specificira na ovaj način, program se izvršava u tekućem interpreteru. Skript se može pokrenuti i na drugi način, bez eksplicitne dodele **x** dozvole - dovoljno je pozvati komandni interpreter da izvrši shell program:

```
$ bash ss1.sh
```

ili

```
$ /bin/sh ss1.sh
```

Ukoliko se shell program ne nalazi u tekućem direktorijumu, potrebno je specificirati putanju do programa.

```
$ bash /home/jsmith/ss1.sh
```

Za razvoj i korišćenje shell skriptova preporučuje se sledeća procedura:

1. skript treba razviti na svom direktorijumu,
2. zatim ga istestirati:
   ```
   $ bash imeskripta
   ```
3. na kraju iskopirati u neki direktorijum koji je podrazumevano uključen u sistemsku putanju.

Program se može kopirati u bin poddirektorijum home direktorijuma autora. Ukoliko veći broj korisnika želi da koristi program datoteku treba kopirati u direktorijume **/bin** ili **/usr/bin** ili **/usr/local/bin** kojima mogu pristupati svi korisnici. Dodatno, korisnicima treba dati dozvolu execute da bi mogli da pokreću program pomoću imena datoteke.

Komande koje se mogu zadavati u skriptovima su:

- standardne Linux komande (poput **cp** ili **mv**)
- komande specifične za shell programiranje. Neke od komandi specifičnih za shell programiranje su gotovo kompletni programski jezici (na primer **awk**).

echo

Jedna od često koričćenih komandi je **echo** koja prikazuje tekst ili vrednost promenljive na ekranu. Sintaksa komande **echo** je:

```
$ echo [opcije] [string, promenljive...]
```

Opcije:

-n ova opcija ne prebacuje kursor u novi red,nakon izvršenja echo komande

-e omogućava interpretaciju sledećih karaktera u kombinaciji sa obrnutom kosom crtom:

\a upozorenje (alert bell)

\b povratak unazad (backspace)

\c ne prelaziti u novi red (suppress trailing new line)

\n novi red (new line)

\r povratak na početak reda (carriage return)

\t horizontalni tabulator (horizontal tab)

\\ obrnuta kosa crta (backslash)

Primer.

```
[milos@cluster1 ~]# echo -e "Petar\n\t\t Petrovic"
Petar
           Petrovic
```

Navodnici

Bash shell prepoznaje tri tipa navodnika, i to:

- **Dvostruki navodnici** - "Double Quotes". Sve što se nalazi u ovim navodnicima gubi originalno značenje (osim \ i $).

- **Jednostruki navodnici** - 'Single quotes'. Sve što je zatvoreno jednostrukim navodnicima ostaje nepromenjeno.
- **Obrnuti navodnici** - `Back quote`. Izraz zatvoren obrnutim navodnicima tretira se kao komanda koju treba izvršavati.

Primer.

```
$ echo "Danasnji datum : date" # tretira date kao string
Danasnji datum : date
$ echo "Danasnji datum : `date`" # tretira date kao komandu
Danasnji datum : Fri Apr 2 16:30:35 CEST 2004
```

Regularni izrazi i metakarakteri

Regulrni izrazi su sintaksički skup fraza koje reprezentuju šablone unutar teksta ili stringova. Regularni izrazi omogućavaju reprezentaciju različitih nizova karaktera vrlo malim skupom predefinisanih karaktera. Često sadrže i **metakaraktere** - karaktere koji reprezentuju drugu grupu karaktera i komandi.

Fajl koji će se koristiti u svrhu testiranja je recimo **/tmp/testfile** (obratiti pažnju na interpunkciju i mala i velika slova):

```
Juliet Capulet
The model identifier is DEn5c89zt.
Sarcastic was what he was.
No, he was just sarcastic.
Simplicity
The quick brown fox jumps over the lazy dog
It’s a Cello? Not a Violin?
This character is (*) is the splat in Unix.
activity
apricot
capulet
cat
celebration
corporation
cot
cut
cutting
dc9tg4
eclectic
housecat
persnickety
The punctuation and capitalization is important in this example.
simplicity
undiscriminating
Two made up words below:
c?t
C?*.t
cccot
cccccot
```

Metakarakteri

Metakarakteri su korisni u redukciji količine teksta koji se koristi sa komandama i za reprezentaciju tekstualnih grupa minimalnim skupom karaktera. Sledeći metakarakteri su u široj upotrebi:

- **. - Tačka. Reprezentuje jedan karakter.**

  Primer:

  Pronaći bilo koju pojavu slova **c** i slova **t** sa tačno jednim karakterom između.

  c.t

  Rezultati iz test fajla:

  ```
  Simplicity cut simplicity
  apricot cutting c?t
  cat dc9tg4 cccot
  ```

```
cot housecat cccccot
```

- **[] - uglaste zagrade. Rezultat odgovara bilo kojem karakteru unutar zagrada.**

  Primer:

  Pronaći bilo koju instancu slova c i slova t sa tačno jednim samoglasnikom između.

  c[aeiou]t

  Rezultati iz test fajla:

```
Simplicity cot simplicity
apricot cut ccot
cat housecat ccccot
```

- *** - zvezda. Reprezentuje nula ili više pojava bilo kojih karaktera.**

  Primer:

  Pronaći sve instance slova c i slova t sa nula ili više karaktera između njih.

  c*t

  Rezultati iz test fajla:

```
Juliet Capulet
The model identifier is DEn5c89zt.
Sarcastic, was what he was.
No, he was just sarcastic.
Simplicity
The quick brown fox jumps over the lazy dog
It's a Cello? Not a Violin?
This character is (*) is the splat in Unix.
activity
apricot
capulet
cat
celebration
corporation
cot
cut
cutting
dc9tg4
eclectic (also eclectic; same word so only one instance shows)
housecat
persnickety
The punctuation and capitalization is important in this example.
simplicity
undiscriminating
c?t
c?*.t
ccot
 ccccot
```

- **[^karakteri] - uglaste zagrade sa kapom između. Nijedan od navedenih karaktera se NE pojavljuje.**

  Primer:

  Pronaći sve pojave karaktera **c** i karaktera **5**, a da između njih ne stoji nikakav samoglasnik.

  c[^aeiou]t

Rezultati iz test fajla:

```
dc9tg4
c?t
```

- **^karakter - Odgovara sekveci samo ako je na početku linije.**

  Primer:

  Pronaći sve pojave stringa **ca** na početku linije.

  ^ca

  Rezultati iz test fajla:

  ```
  capulet
  cat
  ```

  Bez karaktera ^, sekvenca **ca** bi mogla da se nađe bilo gde u stringu:

  ```
  Sarcastic was what he was.
  No, he was just sarcastic.
  capulet
  cat
  housecat
  ```

- **^[karakter(i)] - Kapa ispred sekvence u uglastim zagradama. Odgovara bilo kom karakteru u uglastim zagradama, ali na početku linije.**

  Primer:

  Pronaći sve pojave slova **c** praćenog samoglasnikom i slovom **t** na početku linije.

  Rezultati iz test fajla:

  ```
  cat
  cot
  cut
  cutting
  ```

  Da je izostavljen karakter ^, sekvenca bi mogla biti bilo gde u liniji:

  ```
  Simplicity
  apricot
  cat
  cot
  cut
  cutting
  housecat
  simplicity
  cccot
  cccccot
  ```

- **$ - Znak dolara. Odgovara pojavi sekvence karaktera na kraju linije.**

  Primer:

  Pronaći linije koje se završavaju slovima c i t između kojih može biti bilo šta.

  c*t$

  Rezultati iz test fajla:

  ```
  Capulet cat c?t
  DEn5C89zt cot c?*.t
  ```

```
apricot cut cccot
capulet housecat ccccot
```

- **\ - Backslash. Anulira specijalno značenje karaktera koji ga neposredno sledi.**

  Primer:

  Pronaći sve pojave sekvence karaktera **c?t**.

  c\?t

  Rezultati iz test fajla:

  ```
  c?t
  ```

- **? - Upitnik. Reprezentuje nula ili jednu pojavu karaktera (ne treba ga mešati sa *, koji odgovara nula, jednom, ili više karaktera). Ne podržavaju ga svi UNIX programi.**

  Primer:

  Pronaći sve pojave karaktera **c** i karaktera **t** sa jednim ili nijednim karakterom između njih.

  c?t

  Rezultati iz test fajla:

  ```
  Simpli**cit**y
  ecle**ct**ic
  a**ct**ivity
  house**cat**
  apri**cot**
  The pun**ct**uation and capitalization is important in this example.
  cat
  simpli**cit**y
  cot
  c?t
  cut
  cc**cot**
  **cut**ting
  cccc**cot**
  d**c9t**g4
  ```

- **[a-z] - Potpuna engleska abeceda malim slovima. Odgovara bilo kojem slovu abecede.**

  Primer:

  Pronaći sve pojave karaktera **c** i **t** sa bilo kojim slovom između njih.

  c[a-z]t

  Rezultati iz test fajla:

  Simpli**cit**y cut simpli**cit**y

  apri**cot cut**ting cc**cot**

  cat house**cat** cccc**cot**

  cot

- **[0-9] - Odgovara bilo kojoj cifri.**

  Primer:

  Pronaći sve pojave karaktera **c** i **t** sa bilo kojom cifrom između njih.

c[0-9]t

Rezultati iz test fajla:

d**c9t**g4

- **[d-m7-9] - Odgovara jednom pojavljivanju bilo kog karaktera iz opsega d-m ili 7-9. Primer ilustuje grupisanje komandi.**

  Primer:

  Pronaći sve pojave slova c i slova t, sa jednim karakterom između koji može biti bilo koje slovo u opsegu od **c** do **t** ili cifra između 0 i 4.

  Rezultati iz test fajla:

  Simpli**cit**y dc**9t**g4

  apri**cot** cc**cot**

  cot cccc**cot**

  simpli**cit**y

Prošireni regularni izrazi (extended regular expressions)

Kod proširenih regularnih izraza koji se aktiviraju npr. navođenjem opcije -E grep komandi, a dostupni su i u programima, awk, emacs, vi, itd. Karakteristični su sledeći metakarakteri koji služe za označavanje ponavljanja karaktera ili podizraza (blokova):

- **(*karakteri*)** - označeni podizraz (blok).
- + - Izraz od jednog karaktera nakon kojeg sledi "+" sparuje jednu ili više kopija izraza. Na primer, "ab+c" sparuje "abc", "abbbc" itd. "[xyz]+" sparuje "x", "y", "zx", "zyx", i tako dalje
- **{x,y}** - Sparuje poslednji blok barem "x" i ne više od "y" puta. Na primjer, "a{3,5}" sparuje "aaa", "aaaa" ili "aaaaa"
- **{x}** - Prethodni karakter se pojavljuje tačno **x** puta
- **{x,}** - Prethodni karakter se pojavljuje najmanje **x** puta
- **{,y}** - Prethodni karakter se pojavljuje najviše **y** puta
- **?** - prethodni karakter je opcioni i pojavljuje se najviše jednom.

Složen primer:

fajl **proba.txt**:

```
www8.dobarsajt1.edu
www678.amu.ac.zu5l
www6.ailmit.net
www.euler.ni.ac.yu
mitcl.edu
www.core.amu.edu
www.znanje.edu
```

Izlaz komande:

```
$ grep -E "^(w{3}[0-9]*\.)?[a-z]{1}[a-z0-9]{1,7}((\.ac\.[a-z]{2})|(\.edu))$" proba.txt
mitcl.edu
```

```
www.znanje.edu
```

Objašnjenje:

Navedeni regularni izraz znači da string može, a ne mora (zbog "?") početi sekvencom "www"+bilo koje cifre nakon čega sledi tačka. Iza tačke mora biti slovo, a zatim sekvenca od najmanje 1, a najviše 7 alfanumerika (slova ili brojeva), dok na kraju mora biti domen ".ac.(dva bilo koja slova)" ili ".edu".

grep komanda

Grep je skraćenica od *global regular expression print*. Sintaksa grep komande je sledeća:

```
grep string_za_pretragu fajl_za_pretragu
```

Grep podržava najveći broj metakaraktera.

- Jednostavni grep se može iskoristiti za pretragu reči root u fajlovima direktorijuma /etc i njegovim poddirektorijumima:

  ```
  grep root /etc/*
  ```

- Grep takođe ima argument -v koji odgovara pretrazi svega što **ne sadrži** zadati string:

  ```
  grep -v root /etc/passwd
  ```

- Komanda grep se može koristiti i preko pajpa, kada se njen *stdin* zamenjuje izlazom iz neke druge komande:

  ```
  cat /etc/passwd | grep root
  ```

sort komanda

Korisna komanda za sortiranje izlaza neke komande ili fajla u specificiranom redosledu. Njene opcije su sledeće:

```
-d  Sorts via dictionary order, ignoring non-alphanumerics or blanks.
-f  Ignores case when sorting.
-g  Sorts by numerical value.
-M  Sorts by month (i.e., January before December).
-r  Provides the results in reverse order.
-m  Merges sorted files.
-u  Sorts, considering unique values only.
```

Primer:

1. Kreirati fajl /tmp/outoforder sa sledećim sadržajem:

```
Zebra
Quebec
hosts
Alpha
Romeo
juliet
unix
XRay
xray
Sierra
Charlie
horse
horse
```

```
horse
Bravo
1
11
2
23
```

2. Sortirati fajl u rečničkom redosledu:

```
sort -d /tmp/outoforder
```

Rezultat je:

```
1
11
2
23
Alpha
Bravo
Charlie
Quebec
Rome
Sierra
XRay
Zebra
horse
horse
horse
hosts
juliet
unix
xtra
```

3. Treba primetiti da se string **horse** pojavljuje više puta. Da bi se uklonile te dodatne pojave, koristiti opciju -u:

```
sort -du /tmp/outoforder
```

tee komanda

Komanda tee omogućava da se izlaz komande šalje na više lokacija odjednom. Na primer, ako se želi redirekcija u neki fajl, i istovremeni ispis na ekran, može se uraditi sledeće:

```
ps -ef | tee /tmp/troubleshooting_file
```

Da bi se, umesto da se fajl obriše, tekući izlaz na njega dodao, treba dodati opciju -a:

```
ps -ef | tee -a /tmp/troubleshooting_file
```

Mogu se specificirati i više fajlova kao argumenti **tee** komande, *output* procesa će ići u svaki od tih fajlova.

script komanda

Komanda script omogućava snimanje cele interaktivne sesije. Dakle, u fajl se snima sav stdin i stdout, tj. svaki otkucani karakter i izlaz svake komande. Dodavanje na log fajl vrši se navođenjem opcije **-a**.

```
script -a /tmp/script_session
```

Ako se opcija -a ne navede, ako fajl već postoji, biće obrisan. Ako se pak ne navede ime fajla, u

tekućem direktorijumu se pravi fajl po imenu **typescript**. Da bi se završilo snimanje, potrebno je otkucati **exit** za izlaz. Treba voditi računa **da se ne ostavi sesija komande script** u stanju izvršavanja jer fajl može veoma da naraste!

awk komanda

Awk je, u stvari, jednostavan programski jezik namenjen procesiranju teksta, tj. transformisanju teksta u formatirani *output*. Awk uzima sva ulaza:

- komandu, skup komandi ili komandni fajl koji sadrže instrukcije za poklapanje tekstualnih šablona i smernice za procesiranje i generisanje izlaza.
- podatke sa kojima se radi ili fajl sa podacima.

Prvi primer je:

```
$ awk '{ print $0 }' /etc/passwd
```

Rezultat izvršenja je nešto kao:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
…
```

U gornjem primeru AWK ne procesira nikakve podatke, već prosto čita sadržaj **/etc/passwd** fajla i štampa nefilterisan izlaz na *stdout,* isto kao **cat** komanda. Kada se pozove AWK, opskrbljen je sa dve informacije, a to je komanda za editovanje i podaci za editovanje. Dakle, u primeru je **/etc/passwd** reprezent ulaznih podataka, a komanda za uređivanje prosto štampa (**$0 je oznaka za celu liniju teksta**) fajl u istom redosledu.

Prava primena AWK-a leži u odvajanju traženih delova iz većeg skupa podataka. Na primer, iz fajla /etc/passwd može se dobiti daleko čitljiviji izlaz:

```
awk -F":" '{ print "username: " $1 "\t\t\t user id:" $3 }' /etc/passwd
```

Razultat bi bio nešto kao:

```
username: root          user id:0
username: bin           user id:1
username: sync          user id:5
username: shutdown      user id:6
username: halt          user id:7
username: mail          user id:8
username: nobody        user id:99
username: sshd          user id:74
username: apache        user id:48
username: webalizer     user id:67
username: ldap          user id:55
username: mysql         user id:27
username: pdw           user id:500
```

Podrazumevano, AWK koristi blanko za separator ulaznih podataka, ali se ovo ponašanje može promeniti opcijom **-F**. Ovde je kao separator iskorišćen karakter ":". **$1** sadrži tekst do prvog separatora, **$2** tekst do drugog separatora itd. **$0** je uvek cela linija.

AWK komanda za editovanje se uvek sastoji iz dva dela:

- **šabloni**
- **komande**

Šabloni se upoređuju sa linijama u fajlu, a ako šablon nije naveden, kao u gornjem primeru, sve linije dolaze u obzir.

Rad sa šablonima

Šabloni u AWK-u sastoje se od teksta i jednog ili više regularnih izraza između karaktera "/" (*slash*). Na primer:

```
# String example
/text pattern/
# Reg Ex example match any lowercase chars
/[a-z]/
```

Sledeća komanda:

```
$ awk -F":" '/^m/ { print "username: " $1 "\t\t\t user id:" $3 }' /etc/passwd
username: mail              user id:8
username: milos             user id:1000
username: mysql             user id:89
```

kao što se vidi, u obzir uzima samo linije sa početnim karakterom "m".

Komande

Uobičajene komande AWK-a su **=, print, printf, if, while, i for**. Ove instrukcije se ponašaju kao odgovarajuće instukcije bilo kog programskog jezika, omogućavajući dodelu vrednosti varijablama, štampanje izlaza i kontrolu toka.

Programiranje pomoću AWK-a

AWK komande koje se navode u komandnoj liniji mogu se snimiti i u fajl, na primer:

1. Tekst editorom kreirati fajl **print.awk** sa sledećim sadržajem:

   ```
   BEGIN {
   FS=":"
   }
   { printf "username: " $1 "\t\t\t user id: " $3 }
   ```

2. Izvršiti komandu na sledeći način:

   ```
   $ awk -f print.awk /etc/passwd
   username: root user id:0
   username: bin user id:1
   …
   ```

Pošto je AWK strukturirani jezik, fajl mora da sadrži određene blokove, i to:

1. **Komande koje se izvršavaju samo jednom na početku, blok počinje ključnom rečju BEGIN.** U gornjem primeru u tom delu se postavlja separator,
2. **Komande za poklapanje šablona koje se izvršavaju po jedanput za svaku liniju ulaza.** U gornjem primeru to je blok koji sadrži print komandu.
3. **Komande koje se izvršavaju samo jednom na kraju, blok počinje ključnom rečju END.**

U gornjem primeru taj deo je izostavljen, a mogao bi da glasi:

END {

printf “Zavrseno procesiranje fajla /etc/passwd”

}

FS (field separator) je jedna od nekoliko specijalnih varijabli AWK-a. Još neke su:

- **NF—** Variable for providing a count as to the number of words on a specific line.
- **NR—** Variable for the record being processed. That is, the value in NR is the current line in a file awk is working on.
- **FILENAME—**Variable for providing the name of the input file.
- **RS—**Variable for denoting what the separator for each line in a file is.

Primer 1.

```
$ cat /tmp/data
123abc
Wile E.
aabcc
Coyote
$ awk '/abc/ {print}' /tmp/data
123abc
aabcc
```

U ovom slučaj awk traži uzorak ‘abc’ u datoteci **/tmp/data**, a akcija koja se pri tom obavlja nad nađenim uzorcima je prikazivanje teksta na ekranu (**print**).

Primer 2.

Drugi primer je štampanje broja linija koje sadrže string “abc”.

```
$ awk '/abc/ {i=i+1} END {print i}' /tmp/data
2
```

Ukoliko se u datoteci traži više uzoraka i ukoliko se vrši više obrada, potrebno je prvo napraviti datoteku u kojoj su opisane akcije (na primer actionfile.awk). Prilikom zadavanja komande awk potrebno je zameniti tekst između navodnika, kojim su opisani uzorak i akcija, imenom datoteke: '-f actionfile.awk'.