VEŽBA 2

Pregled

I u shell skriptovi su po svim svojim karakteristikama zapravo programi. I najprostiji proračuni, esencijalni su deo svakog algoritma, isto kao i provere uslova na osnovu kojih se odlučuje kojim putem se algoritam nastavlja. Zato i shell pordžava aritmetičke proračune, provere uslova, uslovno izvršavanje delova programa, kao i upravljačke strukture poput petlji.

Aritmetički proračuni

Shell podržava dva načina obavljanja aritmetičkih proračuna. Jedan uključuje upotrebu komande *let*, a druga upotrebu komande *expr*.

Aritmetički proračuni korišćenjem *let*

Ova komanda podrazumeva dodelu, tj. dodela izračunate vrednosti nekoj promenljivoj je obavezna. Komanda ima argumente oblika *ime_prom=aritmeticki_izraz*, može ih biti više i u tom slučaju se svaki od njih uvažava (i izračunava). Razmaci unutar argumenta nisu dozvoljeni jer bi učinili da se argument interpretira kao više odvojenih argumenata. Ako je neophodno ubaciti razmake, argument treba ubaciti u navodnike što će sprečiti shell da razmake tumači kao znak za odvajanje argumenata. Aritmetički izraz sme da sadrži brojeve, matematičke operacije i promenljive. Svaki član izraza koji ne počinje brojem tumači se kao promenljiva, korišćenje znaka *$* uz promenljive nije potrebno, mada je dozvoljeno. Primer:

```
let x=(10*3)/5+8
let "y = 9 + (3*x)"
```

Aritmetički proračuni korišćenjem *expr*

Ova komanda rezultat proračuna šalje na standardni izlaz. Ukoliko standardni izlaz nije preusmeren u fajl ili cev (pipe), rezultat će biti ispisan na ekran. Pošto se skoro svaki rezultat proračuna dalje koristi u okviru programa, ispis na ekran obično nije ono što je potrebno. Iz tog razloga se komanda expr najčešće koristi unutar *$(sh_kom)* notacije koja čini da se celokupan standardni izlaz shell komande *sh_kom* sačuva kao vrednost unutar programa koja se po potrebi može dodeliti nekoj promenljivoj. Na primer:

```
expr 10 \* \( 3 + 7 \)
R=$( expr 500 - $Y \* 25 )
```

Prioriteti operacija se poštuju, a zagrade na uobičajeni način menjaju prioritete. Ispred operacije * i zagrada neophodno je navesti escape karakter da shell ne bi te karaktere interpretirao na uobičajeni način.

Provera uslova

Komanda *test* kao svoj status vraća vrednost 0 (*ok*) ili različito od 0 (*error*) u zavisnosti od toga da li je navedeni uslov koji se želi testirati tačan ili ne. Shell strukture *if* i petlje *while* u

zavisnosti od rezultata testa izvršavaju odgovarajuće delove skripta.

Skraćeno i praktičnije zapisivanje uslova postiže se navođenjem test izraza unutar uglastih zagrada, npr. `[ $x -gt 10 ]` što testira da li je vrednost promenljive `x` veća od 10. Najvažniji uslovi koji se mogu testirati na ovaj način su sledeći:

```
-e <FILE>      postoji li <FILE>?
-f <FILE>      <FILE> je običan fajl?
-d <FILE>      <FILE> je direktorijum?
-r <FILE>      može li se <FILE> čitati?
-w <FILE>      može li se u <FILE> pisati?
-x <FILE>      <FILE> je izvršni?
-s <FILE>      postoji li <FILE> dužine veće od 0?
-z <STRING>    <STRING> je prazan (nulte dužine)?
-n <STRING>    <STRING> nije prazan?
<STRING1> = <STRING2>    <STRING2> i <STRING2> su jednaki?
<STRING1> != <STRING2>   <STRING2> i <STRING2> nisu jednaki?
<STRING1> < <STRING2>    <STRING1> kraći od <STRING2>?
<INTEGER1> -eq <INTEGER2>     celi brojevi su jednaki?
<INTEGER1> -ne <INTEGER2>     celi brojevi nisu jednaki?
<INTEGER1> -le <INTEGER2>     <INTEGER1> manji od ili jednak
                                         <INTEGER2>?
<INTEGER1> -ge <INTEGER2>     <INTEGER1> veći od ili jednak
                                         <INTEGER2>?
<INTEGER1> -lt <INTEGER2>     <INTEGER1> manji od <INTEGER2>?
<INTEGER1> -gt <INTEGER2>     <INTEGER1> veći od <INTEGER2>?
```

Zadaci

1. Napisati skript koji omogućava sledeće: korisnik preko tastature unosi broj `x`, a odmah potom računar izračunava vrednost `3 * x + 5` i ispisuje je u sledećem redu na ekranu. Nakon toga očekuje se unos sledećeg broja. Postupak se ponavlja sve dok korisnik ne unese vrednost 0.

   Primer:

   ```
   Unesite broj X : 5
   (3 * X) + 5  je : 20
   Unesite broj X : 7
   (3 * X) + 5  je : 26
   Unesite broj X : 0
   Kraj!
   ```

2. Modifikovati prethodni skript tako da svoje ulazne podatke dobija sa standardnog ulaza, a standardni ulaz treba preusmeriti iz tekstualnog fajla sa po jednom numeričkom vrednošću po liniji.

   Primer ulaznog fajla `test2.dat`:

   ```
   5
   7
   19
   120
   ```

```
8
```

Primer poziva i rada skripta:

```
[user@max vezba_2]$ ./z2.sh < test.dat
(3 * 5) + 5  je : 20
(3 * 7) + 5  je : 26
(3 * 19) + 5  je : 62
(3 * 120) + 5  je : 365
(3 * 8) + 5  je : 29
Kraj!
```

3. Napisati skript koji kopira sve fajlove sa ekstenzijom *.mv* iz *izvornog* direktorijuma navedenog kao prvi parametar u *ciljni* direktorijum naveden kao drugi parametar. Ciljni i izvorni direktorijumi treba da postoje pre poziva skripta, a izvorni treba da ima i odgovarajući sadržaj (određeni broj fajlova sa *.mv* ekstenzijom). Svakom kopiranom fajlu treba dati ime *MV_<broj>.mv* gde *broj* počinje od 1, a uvećava se nakon svakog kopiranog fajla. U toku rada skript treba da prijavljuje šta i gde kopira.

Primer rada skripta:

```
[user@max vezba_2]$ ./z3.sh mvfrom mvto
Fajl mvfrom/rambo.mv kopiran u mvto/MV_1.mv
Fajl mvfrom/rocky.mv kopiran u mvto/MV_2.mv
```

4. Modifikovati prethodni skript tako da proverava broj parametara skripta (treba da ih bude 2) kao i postojanje navedenih direktorijuma (treba da postoje da bi se zadata operacija mogla izvršiti). Ukoliko se utvrdi bilo kakva greška ne kreće se u obavljanje operacija kopiranja nego se na ekran ispisuje odgovarajuće obaveštenje.