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 kao džoker (*engl. wildcard*) karakter.

Aritmetički zapis pogodan za shell programe (skriptove)

U programima se uobičajeno koristi notacija sa dvostrukim zagradama: `$(( arit. izraz ))`. Aritmetički izraz unutar zagrada prati sintaksu izraza koji se koristi u komandi `let` nakon znaka dodele (=) u slučaju da je ceo parametar stavljen unutar navodnika (videti drugi primer kod

objašnjenja komande `let`).

```
echo $(( 2*x + 80 ))  # x je promenljiva (ne mora imati $)
rez=$(( a+b+120 ))
```

Dejstvo ovog načina zapisivanja je da će rezultat proračuna zameniti celokupan izraz na mestu navođenja kao što bi se to desilo u slučaju zamene vrednosti neke promenljive.

U slučaju greške, rezultat će ostati prazan, a shell će ispisati tekstualnu poruku o grešci i nastaviti sa radom.

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?
-v <VAR>       proverava se da li je promenljiva <VAR> definisana
<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>?
```

Provera postojanja parametra skripta

Jedan jednostavan način za proveru konkretnog parametra je ispitivanje da li je string prazan:

```
if [ -n "$1" ]
then
# ok - parametar postoji
else
# greska - parametar je nulte duzine!
fi
```

Zadaci

1. Napisati shell-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.

   Primer:

   ```
   Unesite broj x : 5
   (3 * x) + 5  je : 20
   ```

2. Na osnovu prethodnog skripta napisati novi s tom razlikom što se broj ne očekuje kao unos preko tastature, nego kao prvi parametar skripta.

   Primer rada skripta:

   ```
   [user@max vezba_2]$ ./z2.sh 25
   (3 * x) + 5  je : 80
   ```

3. Na osnovu prethodnog skripta napisati novi takav da on proverava postojanje svog prvog parametra. Ukoliko on ne postoji, dati obaveštenje korisniku. Ukoliko postoji, rezultat se smešta u fajl `rezultat.txt`.

   Napomena: za upis rezultata u fajl koristiti preusmeravanje izlaza komande `echo` (>). Sadržaj fajla sa rezultatom najlakše se proverava komandom `cat rezultat.txt`.

   Primer rada skripta:

   ```
   [user@max vezba_2]$ ./z3.sh
   Greška: ne postoji prvi parametar!
   ```

4. Napisati skript koji izračunava i ispisuje vrednost kvadrata broja koji se čuva u promenljivoj `BROJ`. Taj skript se poziva iz drugog skripta koji omogućava unos vrednosti preko terminala, poziva skript za izračunavanje kvadrata, prihvata rezultat, množi ga sa 7 ako je manji od 50 ili sa 3 u suprotnom. Konačni rezultat se ispisuje na terminalu.

   Napomena: ne zaboraviti da se pozvani skript izvršava u novom procesu, pa je neophodno uraditi eksport promenljive `BROJ`. Takođe, prihvat vrednosti usmerene na standardni izlaz, obavlja se operacijom `$(...)`. Kvadrat broja `x` najlakše se računa kao `x*x`. Takođe, shell podržava i sintaksu `x**2`.