VEŽBA 3

Pregled

U prethodnoj vežbi, tema su bili aritmetički i logički proračuni, kao i korišćenje logičkih izraza za odlučivanje. U ovoj vežbi tema su petlje i uzorci koji se koriste za generisanje spiskova imena fajlova. Takođe će biti reči o obradi parametara skriptova.

Složeni logički izrazi

Logičko I

```
izraz_1 && izraz_2 && izraz_3
```

Izraz može biti logički izraz u uglastim zagradama ili proizvoljna komanda. Izrazi se evaluiraju, tj. komande se pozivaju redom s leva na desno. Prvi vraćeni netačan rezultat prekida niz (ostali izrazi se ne evaluiraju). Celokupan niz ima logičku vrednost poslednje evaluiranog izraza.

Logičko ILI

```
izraz_1 || izraz_2 || izraz_3
```

Ovde se niz prekida sa prvim evaluiranim izrazom koji je tačan. Takođe, celokupan niz izraza imaće logičku vrednost poslednjeg evaluiranog izraza.

Negacija

```
! izraz
```

Navođenje znaka `!` ispred logičkog izraza invertuje njegovu logičku vrednost.

Petlje

`while` i `until` petlja

Telo `while` petlja se ponavlja dok je navedeni uslov tačan. Uslov se proverava na početku svakog prolaska kroz petlju.

```
while <uslov>
do
  TELO PETLJE
done
```

Petlja `until` počinje ključnom reči `until` umesto `while`, inače je istog oblika. Uslov se interpretira suprotno, tj. petlje se izvršava sve dok navedeni uslov nije zadovoljen.

`for` petlja

U programskim jezicima, for petlja je obično brojačka petlja. U shell-u, ova petlja prolazi kroz spisak imena. Najčešće se koristi za obradu pojedinačnih fajlova čija imena su sadržana u spisku.

```
for p in <lista imena>
do
  TELO PETLJE, u svakom prolazu, $p sadrzi jedno od imena sa
liste
done
```

upravljačke struktura `case` i `select`

```
case word in
  pattern_1)
    spisak komandi
    ;;
  pattern_2)
    spisak komandi
    ;;
  ...
esac
```

Izvršiće se onaj spisak komandi kod kojeg se navedeni uzorak (*pattern*) uklapa sa izrazom (*word*) navedenim na početku. Uzorak sme sadržati i džoker karaktere po principu koji odgovara generisanju spiska fajlova (videti odeljak niže).

preusmerenje ulaza/izlaza za telo petlje

Petlja u shell-u ima status složenog izraza. Zbog toga je moguće za komande koje se nalaze unutar tela petlje dodeliti standardni ulaz ili izlaz koji se može razlikovati od onog koji važi za ceo skript. Npr. standardni ulaz može da se preusmeri iz fajla `input.txt` na sledeći način:

```
while [ $x -gt 10 ]
do
  # komande unutar tela petlje
done < input.txt
```

Generisanje spiskova pomoću džoker karaktera

Džoker (engl. widlcard) karakteri su sledeći

- `*` - menja proizvoljan broj proizvoljnih karaktera
- `?` - menja jedan proizvoljan karakter
- `[...]` - menja jedan karakter iz skupa navedenog između uglastih zagrada

Kad god shell u nekom izrazu prepozna neki od džoker karaktera, taj izraz se tumači kao šablon (*engl. pattern*). Shell analizira imena fajlova iz tekućeg (za relativne putanje) ili navedenog direktorijuma (za apsolutne putanje). Šablon će biti zamenjen listom imena fajlova u kom su imena razdvojena znakom praznog mesta. Ukoliko se u šablon ne uklapa ime ni jednog fajla, šablon će biti zamenjen praznim sadržajem (efektivno, biće ignorisan).

Na primer, u šablon sa džokerima oblika `?[AB]c*` uklapaju se sledeća imena:

```
qAcwww, wBcx.txt, sAc, sAcfile
```

Istovremeno, sledeća imena se ne uklapaju:

```
Acwww, qcwww, qABcmix
```

Obrada parametara skripta

Komanda `shift` pomera sve parametre skripta na levo, pri čemu se uvek prvi parametar gubi. To omogućuje da se implementira dosledan sistem analize parametara gde se uvek posmatra krajnje levi parametar i eventualno nekoliko susednih u odnosu na prvi.

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 ili karakter `x`.

   Napomena: ni u ovom, ni u narednim zadacima se ne treba posebno obazirati na ispravnost ulaznih podataka. Korisnika ništa ne sprečava da unese podatak koji se ne može protumačiti kao ceo broj ili drugi podatak očekivanog oblika. Rešenja treba da daju ispravne rezultate ako su zadati ispravni ulazni podaci.

   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. Ova verzija treba da obavlja proračun sve dok u ulaznom fajlu ima podataka, tj. dok se sa čitanjem brojeva ne stigne do kraja fajla.

   U svakom redu ulaznog fajla treba da se nalazi jedna brojna vrednost. Za unos koristiti komandu `read`. Po dostizanju kraja fajla (*end-of-file*) komanda `read` vraća logičku vrednost netačno (u suprotnom, vraća tačno).

   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 nalazi srednju vrednost svih svojih parametara (računanje treba da bude celobrojno, to će za rezultat imati izvesnu grešku u srednjoj vrednosti).

   Primer rada skripta:

```
$ ./z3.sh 2 15 87 51 34
broj parametara je 5
zbir parametara je 189
srednja vrednost je 37
```

4. Modifikovati prethodni skript tako da se brojevi čija se srednja vrednost računa čitaju iz niza fajlova unutar tekućeg direktorijuma koji imaju ekstenziju `dat` (ime se završava na `.dat`). Broj fajlova nije unapred ograničen. Svaki fajl sadrži samo cele brojeve i to jedan u svakom redu (kao u 2. zadatku). Izračunata srednja vrednost se potom ispisuje kao ceo broj ili broj sa 3 decimalna mesta u zavisnosti od izbora koji se napravi u meniju sledećeg oblika:

```
1) celobrojno
2) dec_3
```

   Za ispis menija i proveru izbora koristiti kombinaciju `select` i `case` strukture.

Pomoć za rešavanje zadataka

Shell nudi samo aritmetiku sa celim brojevima. Zato se ispis na tri decimalna mesta može obaviti na sledeći način:

```
k=$((suma/n))
d=$((suma%n))
dec3=$(( (1000*d)/n ))
echo "Srednja vrednost je $k.$dec3"
```

Suština rešenja je da se na osnovu ostatka pri deljenju odredi razlomljeni deo sa tačnošću od 3 decimalna mesta. Ovaj postupak se može uporediti sa postupkom koji se naziva rad sa *fiksnim zarezom*.