Promenljive:

env - izlista sve sistemske promenljive (Oblik je promenljiva = vrednost)

echo $USER - ispisivanje vrednosti sistemske promenljive

Dodela vrednosti(broj,string,druga promenljiva)

var1=10 - definisanje promenljive koja je živa dok je terminal otvoren

echo $var1 - ispisivanje vrednosti naše promenljive

var2="Neki tekst" - string mora pod navodnicima

(echo $var2)

var3=$var2 - var3 dobija vrednost promenljive var2

(echo $var3)

Aritmetika(zbir dva broja):

b1=10

b2=5

b3=b1+b2

echo $b3 (Ovde će se ispisati b1+b2 jer deo nakon = u gornjoj naredbi se tumači kao string)

b3=$b1+$b2(Ovde će se ispisati 10+5, zameniće se lepo vrednosti ali se neće izvršiti aritmetika već ispis stringa)

b3=$(expr $b1 + $b2) (ispravan način, mora da stoje ovako razmaci)

echo $b3 (ispisaće se 15)

b4=$[$b1 + $b2] (drugi način za pisanje izraza)

echo $b4

Uspešno izvršavanje komande i test komanda (echo $? i test)

echo $? - prikazuje da li se poslednja izvršena komanda uspešno izvršila (0-uspešno, >0 - neuspešno; pokazati na primeru ls -l i lss(ne postoji komanda))

var=123

test $var = 123; echo $? - proveravamo da li je var==123, echo$? štampa rezultat poslednje izvršene komande, komande razdvajamo sa ;

(pokazati i test $var = 124)

test $var = 123; <--> [$var = 123]; (drugi način zapisa)

Relaciona poređenja i poređenja na pripadnost

[abc '<' abd]; echo $? - mora < pod navodnike jer bi inače ono označavalo preusmeravanje ulaza tj izlaza

[-z "123"]; echo $? - -z ispituje da li string sadrži 0 karaktera, pošto ne sadrži dobijamo rezultat veći od 0 (isprobati sa "")

[-n "123"]; echo $? - -n ispistuje da li string sadrži bar jedan karakter

var=100

[$var -gt 90]; echo $? da li je var veće od 90 (greater than)

[$var -lt 90]; echo $? da li je var manja od 90 (less than)

[$var -eq 90]; echo $? da li je var jednaka 90 (equal)

U okviru uglastih zagrada možemo da vezujemo uslove (or, and...)

[$var -gt 101 -o $var -lt 110]; echo $? (var>101 or var<110 , za and se koristi -a)

mkdir direktorijum

touch f1.txt

ls -l

[-d deljeni]; echo $? da li je deljeni direktorijum

[-f deljeni]; echo $? da li je deljeni fajl

[-x f1.txt]; echo $? da li fajl može da se izvršava

---

*Osnove skripti, if, for, while i ifs

Za pisanje skripti koristićemo tekstualni editor emacs

emacs -nw skripta.sh - pokretanje emacs-a u samom terminalu (analogija sa emacs-om je npr notepad++), skriptu pišemo u fajlu skripta.sh

Da bi se fajl tumačio kao bash-shell skripta mora da počne sa #!/bin/bash

echo Zdravo svima!

Da bi izašli iz editora ctr+x+s (sačuvali ono što smo odradili), ctr+x+c (izlazak)

ls -l (nalazi se tu sad skripta naša)

./skripta.sh (pokretanje skripte ali moramo promeniti prava pristupa skripti)

chmod 777 skripta.sh

./skripta.sh (pokrećemo skriptu)

(emacs -nw skripta.sh)

Komentare pišemo iza znaka #

(#Ovo je komentar) (napišemo u tekstualnom editoru)

Primeri napisani u okviru jedne skripte:

1) if else

```
tmp=1
if [ $tmp -eq 1 ]; then
        echo Tmp je 1!
else
        echo Tmp nije 1!
fi (zatvaranje if-a kao kontrolne strukture)
```

if [-e skripta.sh -a -w skripta.sh -a -x skripta.sh]; then - da li postoji fajl skripta.sh u direktorijumu odakle se pokreće skripta, i da li u njega možemo da upisujemo i izvršavamo

```
        echo Skripta skripta.ch postoji, moze da se menja i izvršava
else
        echo Skriptu nije moguce koristiti!
fi
```

2)else if grana

```
tmp2=2
if [ $tmp2 -eq 1 ]; then
        echo tmp2 je 1!
elif [ $tmp2 -eq 2 ]; then
```

```
        echo tmp2 je 2!
fi
```

3) while

```
i=0
while [ $i -lt 5 ]; do
        let i++
        echo $i.
done
```

4) for(for petlja nije brojačka petlja kao u višim programskim jezicima, dok je i nešto idi uvećavaj i ili smanjuj, to je tekstualna petlja jer radi sa tekstom,

njen input je tekst, i onda ona prolazi kroz tekst, promenljiva koja se koristi za iteriranje kroz for petlju uzima po jednu reč iz teksta tj sve što se nalazi

izmedju belina (tab, enter i space)

```
for j in 6. 7. 8. 9. 10.; do (promenljiva j prolazi kroz tekst koji ide nakon in)
        echo $j
done
```

Mi for petlji može da prosledimo rezultat neke druge komande, a da onda for petlja prodje kroz taj tekst koja je dobila kao rezultat

```
for j in $(ls -h); do
        echo $j
done
```

(find . -maxdepth 1 -name "*.txt" -pretraži u tekućem direktorijumu na maksimalnoj dubini 1 sa imenom koje se završava sa .txt)

```
i=1
for j in $(find . -maxdepth 1 -name "*.txt"); do
```

```
        echo $i. $j
        let i++
done
```

5)IFS - sistemska promenljiva koja sadrži znakove koji predstavljaju delimitere na osnovu kojis for petlja razgraničava reči, podrazumevana vrednost je znak beline

(Van editora ovo odraditit: touch ff.txt, chmod 777 ff.txt, echo Prva recenica. >> ff.txt, cat ff.txt, echo Ovo je druga recenica. >> ff.txt, cat ff.txt)

```
OLD_IFS = $IFS
```

IFS=$'\n' (mora dolar da bi shell tumačio enter kao dodeljenu vrednost promenljivoj a ne kao da smo samo prešli u novi red

```
for j in $(cat ff.txt); do
        echo $j
done
IFS=$OLD_IFS
```

Dobra je praksa restaurirati staru vrednost IFS u slučaju da neki drugi korisnik želi nešto da radi u konzoli i slično

*Pokretanje skripte koja prima parametre iz komandne linije

(cd ./treci_cas/

emacs -nw zad1.sh)

```
(#!/bin/bash)
echo Ime: $1
echo Prezime: $2
echo Broj indeksa: $3
```

Želimo da od spolja kad neko bude pozvao skriptu, da nam zapravo prosledi 3 parametra

Vraćamo se u konzolu

(chmod u+x zad1.sh)

zad1.sh Bogdan Radosavljevic 2020/0109

Vraćamo se u skriptu

Dodajemo na početku echo Program: $0 - možemo da ispišemo i ime programa

Međutim ispisuje se kompletna putanja do skripte, da bi ispisali samo ime pisem `basename $0`

Možemo vršiti proveru da li su parametri prosleđeni valjani

Dodamo na početku if [-n "$1" -a -n "$2"] then - ovo -n pita da li se nešto nalazi unutar $1 tj $2 i ukoliko se nalazi vraća logičku istinu

Objedinićemo ime i prezime

echo Program: `basename $0`

echo Ime i prezime: $1

echo Broj indeksa: $2

else

    echo Nisu svi parametri uneti

zad1.sh Bogdan Radosavljevic 2020/0109

Međutim ovo neće raditi valjano jer se parametri razdvajaju belinama

Da bi Bogdan Radosavljević bio jedan parametar stavićemo ga pod ''

Pokazati i slučaj kad nisu svi parametri uneti zad1.sh Bogdan Radosavljevic

Da ne bi ispitivali za svaki od par dal je prazan možemo reći if [$# -eq 2] - $# je promenljiva koja čuva broj par prosleđen iz kom linije

Pored $# postoje još dve spec promenljive $* i $@ koje čuvaju sve parametre koje smo prosledili kao jedan string

echo Testiranje \$* i \$@ - escape-ujemo znake $ jer želimo ispis toga

echo $* - ovo sve tumači kao jedan definisani string

echo $@ - ovo tumači kao razdvojene parametre

Primer se može videti u for petlji

for param in "$*" do - ovo isto i za $@ uraditi

echo $param

done