Bash script

Su programsi jezik sam po sebi . On se interpetira on se ne kompajlira. On se jednostavno odmah izvršava. Izuzetno su korisne. Nisu komplikovane kao neki prog. jezici.

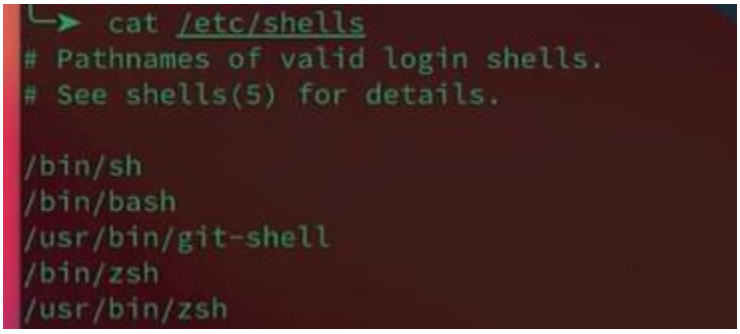
Običnu tekstualnu datoteku morate da je napravite izvršnu. Da bi se te skripte mogle izvrsavati

Veoma je bitno gde čuvate bash skriptu , trebate ih staviti u posebne direktorijume da bi se pravilno koristile.

Da krenemo od ovoga ako ukucamo

cat /etc/shells

dobićemo nazive svih terminala, ljuski koje imate u vašoj linux nandistribuciji.



Treba se proveriti koje direktorijume imamo sa kom ls.

Traži se direkorijum **bin .** *On služi da mi u njemu čuvamo naše skripre. Kao što možemo videti on se nalazi u home direktorijumu. Kada u njemu stavimo neku skriptu ona se izvršava.*

Ako ga ne nalazimo moramo ga napraviti. (mkdir bin).

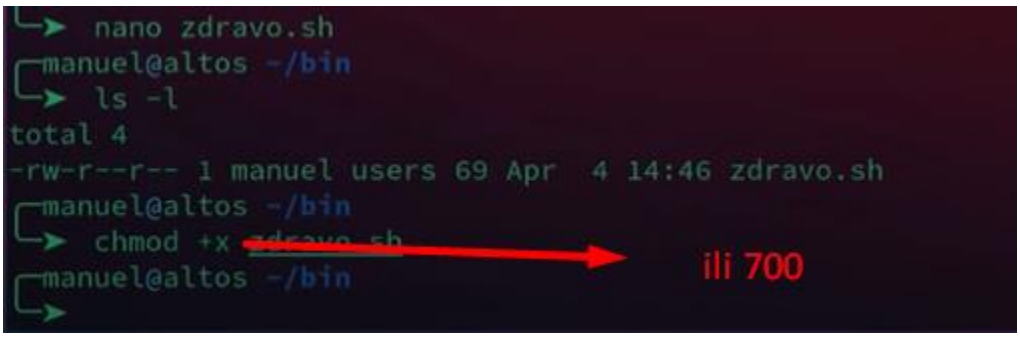
Kreirajmo jednu prostu komandu skriptu. Koristićemo tekstualni editor **nano**

nano zdravo.sh.

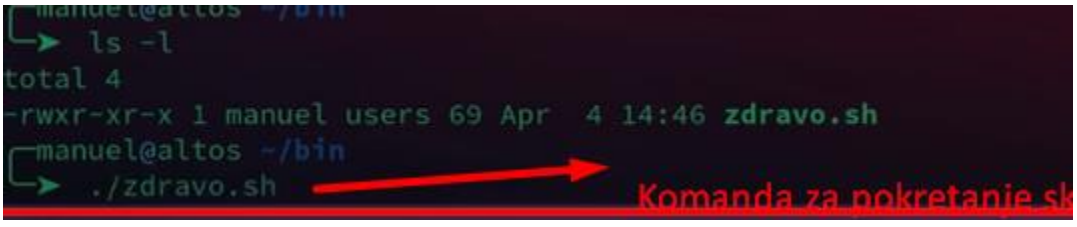
Imamo sada jedan tekstualni fajl **nano zdravo.sh**

Da bi znali da je bash skripta moraćemo pisati ovu prvu liniju koda

Ukucali smo neki tekst. Da bi ovo bilo nešto više od tekstualno fajla tj da bi ovo moglo da se izvrši mi moramo videti koja ovlašćenja ovaj fajl ima . To se postiže komandom **chmod. Bitno je naravno da ima ovlašćenje za izvršavanje tj +x .**Na taj način smo odredili nivo pristupa**.**



I da vidimo sada kao se to izvršava. Naravno potreban je i komanda za pokretanje skripte.



2.Kako da izvršavate BASH skripte koristeći samo ime skripte

Šta ako nećemo da pokrećemo kao u prethodnom primeru. Možda hoćemo samo navođenjem naziva skripte da se ona pokrene. Pogledajmo kako to može.

To najviše zavisi gde se naša skirpta nalazi. Ona ne moramo kucati **./nazivskripte**

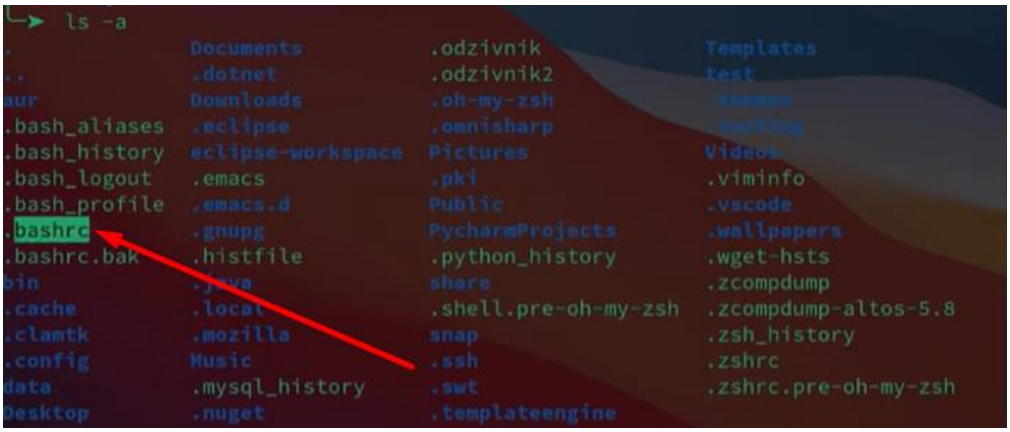
Ovde imamo problem da vidimo ko sve može videti našu skriptu i ko sa njom može raspolagati.

Kucanjem komande **echo $PATH** možemo videti sva moguća mesta, direktorijume gde će samim stavljanjem naše skripte ona u mogućnosti da se automatski izvrši. Obratite pažnju na ono što sam malopre rekao a to je da ako je stavimo u **usr/local/bin**

Ona biti na raspolaganju svim korisnicima koji koriste računar.

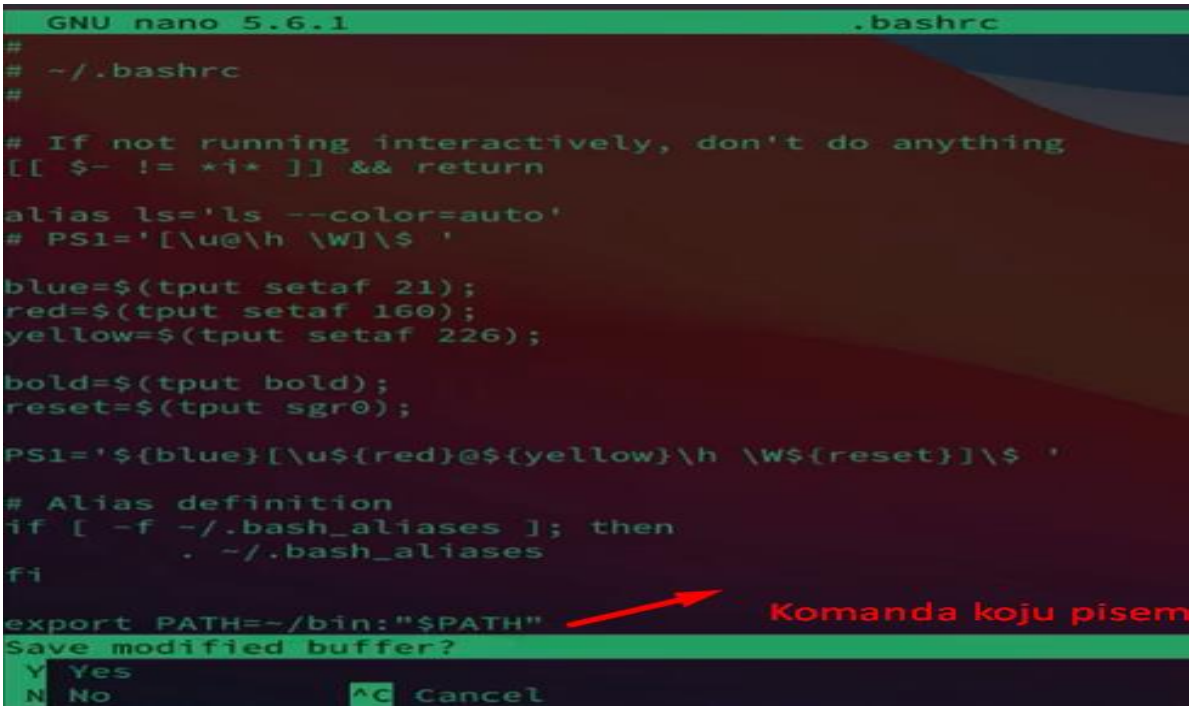
Ako želimo da samo mi možemo da vidimo i raspolažemo tom skriptom e zbog toga postoji direktorijum **bin**

Naš problem je što on nije među ovim gore navedenim lokacijama. Kako staviti naš dir bin među navedenim lokacijama.



U ovom našem direktorijum .bashrc moramo ukucati komandu.

Ućićeno u taj direktorijum sa **nano .bashrc** i na kraju možemo napisati sledeću komandu.



Šta se sada dešava.

Dovoljno je u dir bin kucati samo zdravo.sh i on će automatski izvršiti skriptu.

Možemo pokrenuti našu skriptu i bez kucanja .sh

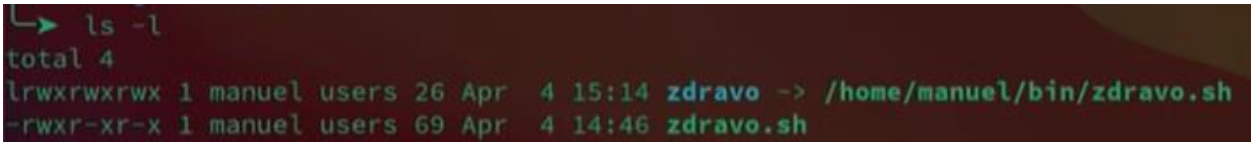
Postoje dva načina za to.

Prvi je da napravimo neki link sa **ln**

```
ln -s ~/bin/zdravo.sh zdravo
```

gde smo sa zdravo nazvali taj naš link.

To možemo proveriti sa komandom ls -l i kao što vidimo imamo link na dir **bin/zdravo.sh**



Ovo može samo ako smo u direktorijumu bin.

Kako uraditi to ako nismo u bin-u to je drugi načina. To se postiže pravljenjem **alias**-a

On se pravi ovako

```
alias zdravo="~/bin/zdravo.sh"
```

Izađivo sada u home direktorijum i probajmo .

On traje samo dok traje sesija.

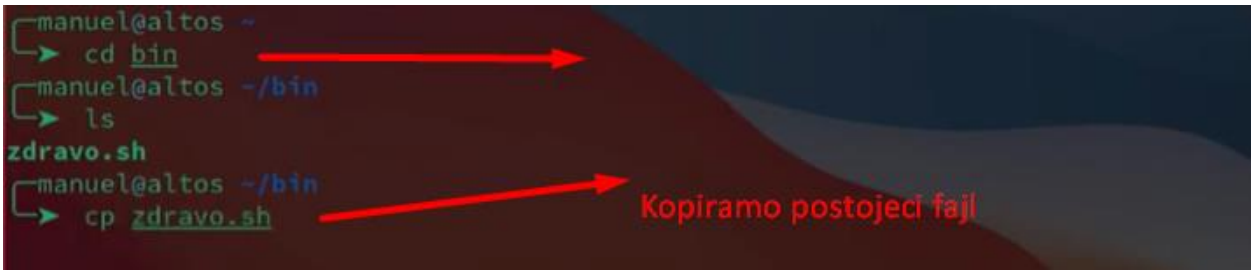
Moramo napraviti trajni alias da bi stalno radilo.

3.Redirekcija ulaza i izlaza u skriptama

Skripte ne služe samo za izvršavanje naredbe koje se kucuju u terminalu. Mnogi korisnici ih kucaju direktno u terminalu.

Mogu se koristiti za redirekciju na neke druge stvari.

Uđimo u naš direktoriju koji smo već formirali i tu ćemo kopirati već postojeći fajl

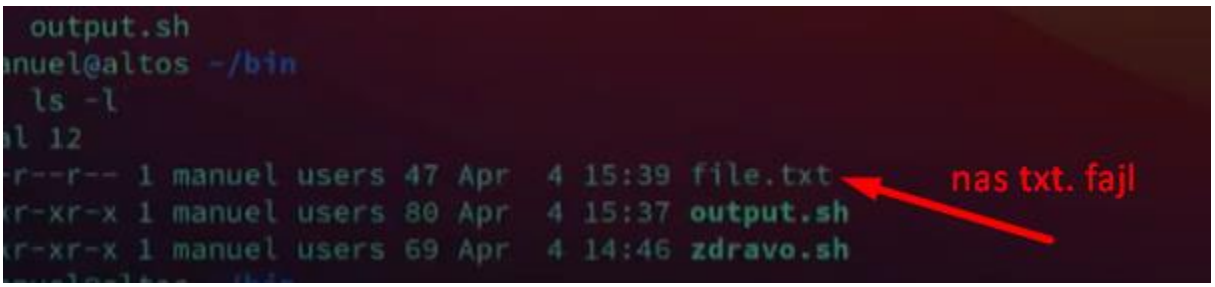


I on će kopirati našu skriptu

Ako pogledamo sta smo dobili neće se videti naš tektualni fajl, zato što još nije kreiran.

On će se kreirati kada pokrenemo skriptu .

Kucamo **output.sh** i videćemo da će se pojaviti naš tekstualni fajl.

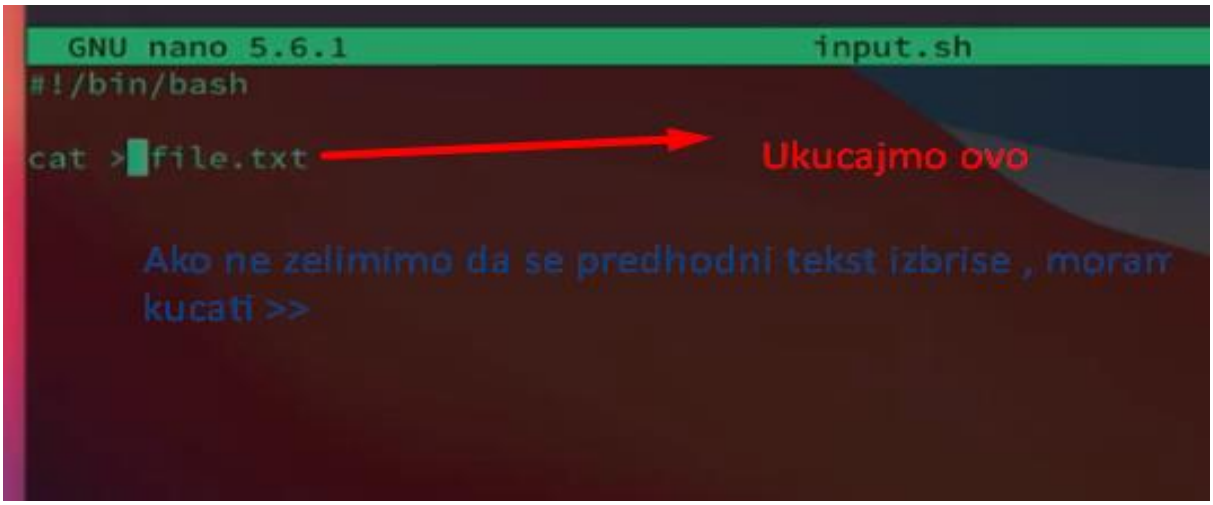


Videćemo da je on usmerio izlaz **echo** komande u txt. Fajl

Kreirajmo sada još jednu skriptu npr **input.sh**

```
cp output.sh input.sh
```

I hajde preko nano editora da je promenimo , tj njen sadržaj menjamo **nano input.sh**



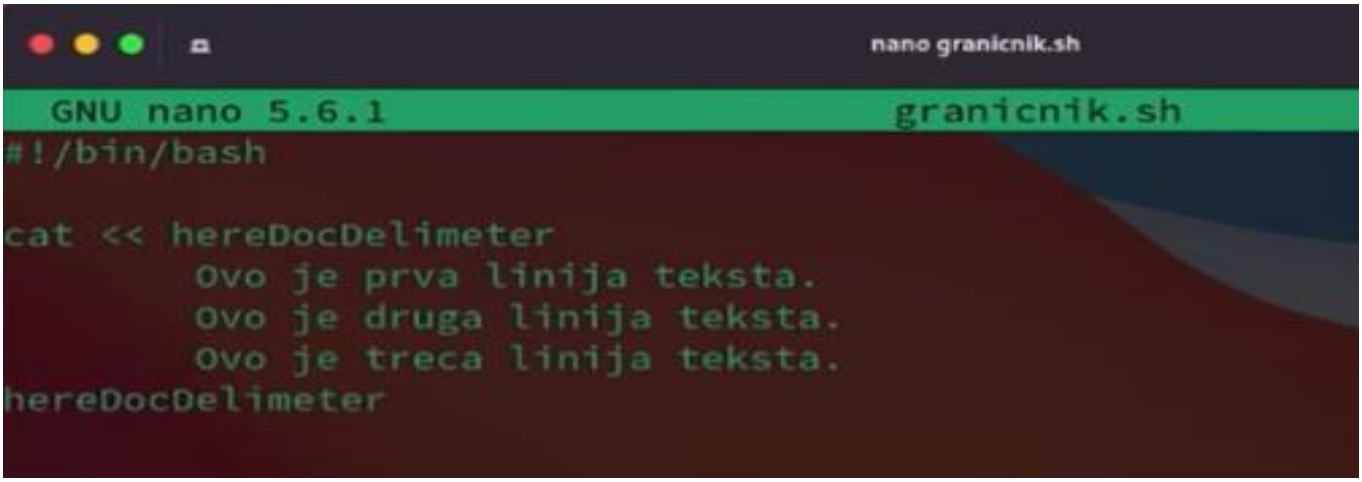
Sada da pogledamo našu skriptu. Uzmimo i kucajmo neki tekst

```
➜  input.sh
Ovo je neki moj input tekst.
```

Izlaz je sa ctrl d. Kada uđemo u naš txt fajl možemo videti upravo tekst koji smo upravo kucali.

4. Šta je Bash here Doc Delimeter - graničnik ?

Ako želimo da pisemo neki teskt koji se nalazi u više redova.
Da ne bi stalno za svaki red pisali echo možemo uraditi upravo sledeće
#!/binb/bash
cat << ovde kucamo nesto sto prestavlja naslov nase lekcije
tekst
tekst
tekst
…….
ovde kucamo nesto sto prestavlja naslov nase lekcije



I evo šta je izvršeno. Možemo zaključiti da on sve vidi kao neki tekst ovde.

```
djordje@Djordje80:~/bin$ delimetar.sh
        x=4
        y=5
        echo"zbir je dat"
        (( + ))
djordje@Djordje80:~/bin$
```

Imamo I jednolinijske kao i višelinijske komentare

```
#!/bin/bash

echo "Postoje dve vrste komentara:"

# Ovo je jednolinijski komentar

echo " 1. # jednolinijski komentar"

: ' Ovo je viselinijski komentar
    Ovo je viselinijski komentar
    Ovo je viselinijski komentar '
```

Kao što vidimo iz predhodne slike , višelinijski komentari, pocinju sa : ’ a zatvaraju se sa ’

6.Promenjive, konstante , funkcije

U bin direktorijumu kreirajmo jednu skriptu. Neka se zove systeminfo.sh

U bash skriptama je mnogo jednostavnije bar kada su u pitanju promenjive. Oni ih tretiraju kao znakova niz i mi ih ne moramo deklarisati.

Da vidimo u primeru.

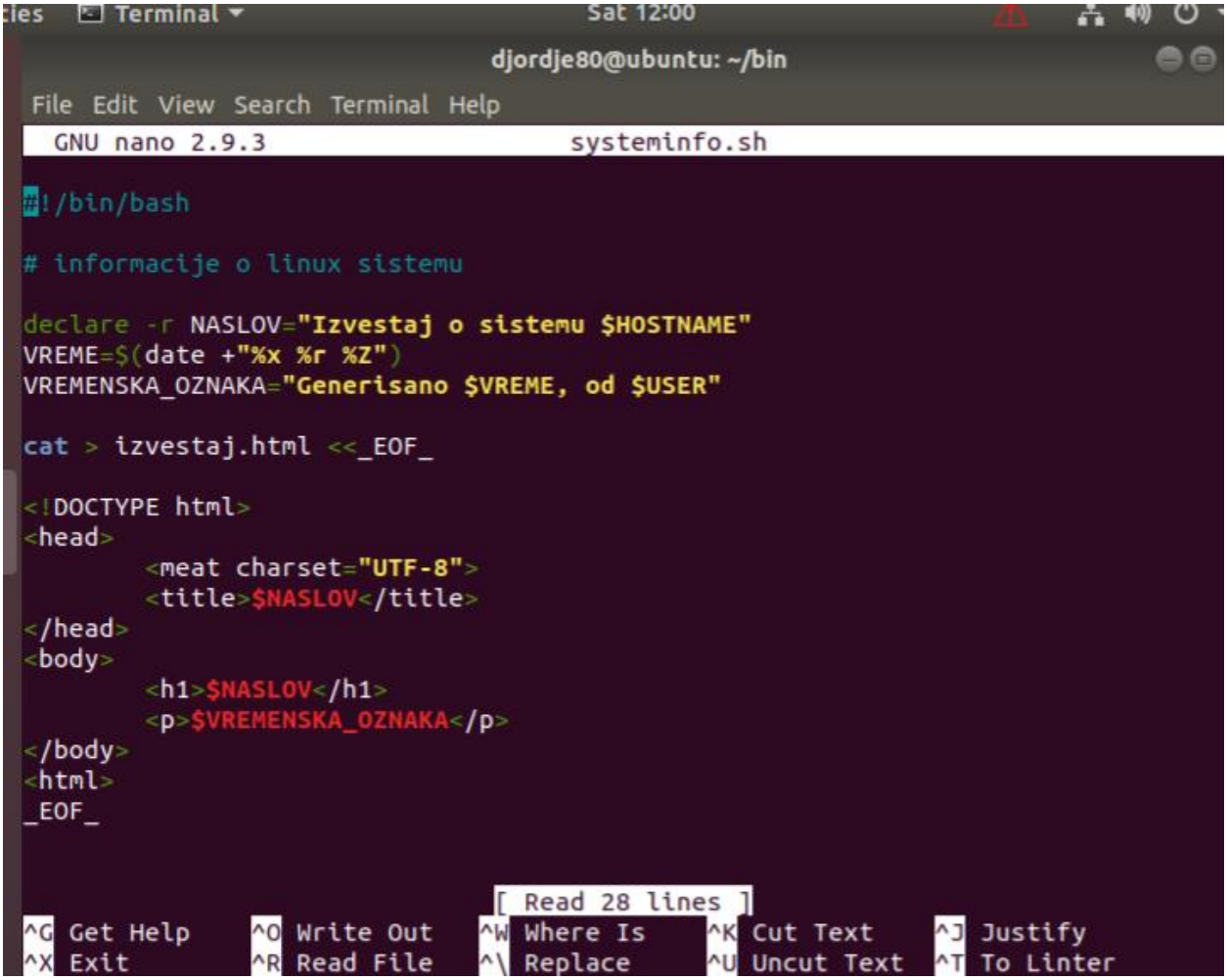
naslov="Izvestaj o sistemu $HOSTNAME"

ovo je promenjiva tipa string, napisana je malim slovima.

Ne mogu krenuti brojem , specijalnim karakterom, razmakom isl.

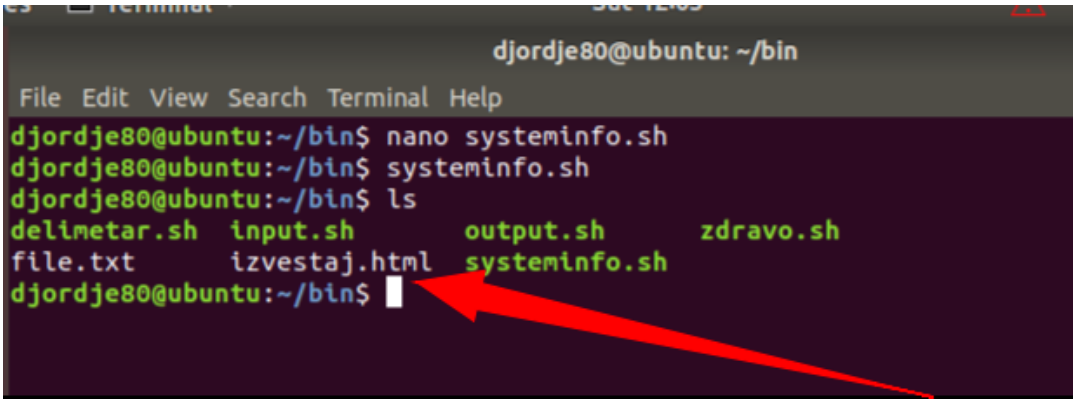
Konstante se pisu velikim slovima i to je vrednost koju ne menjamo .

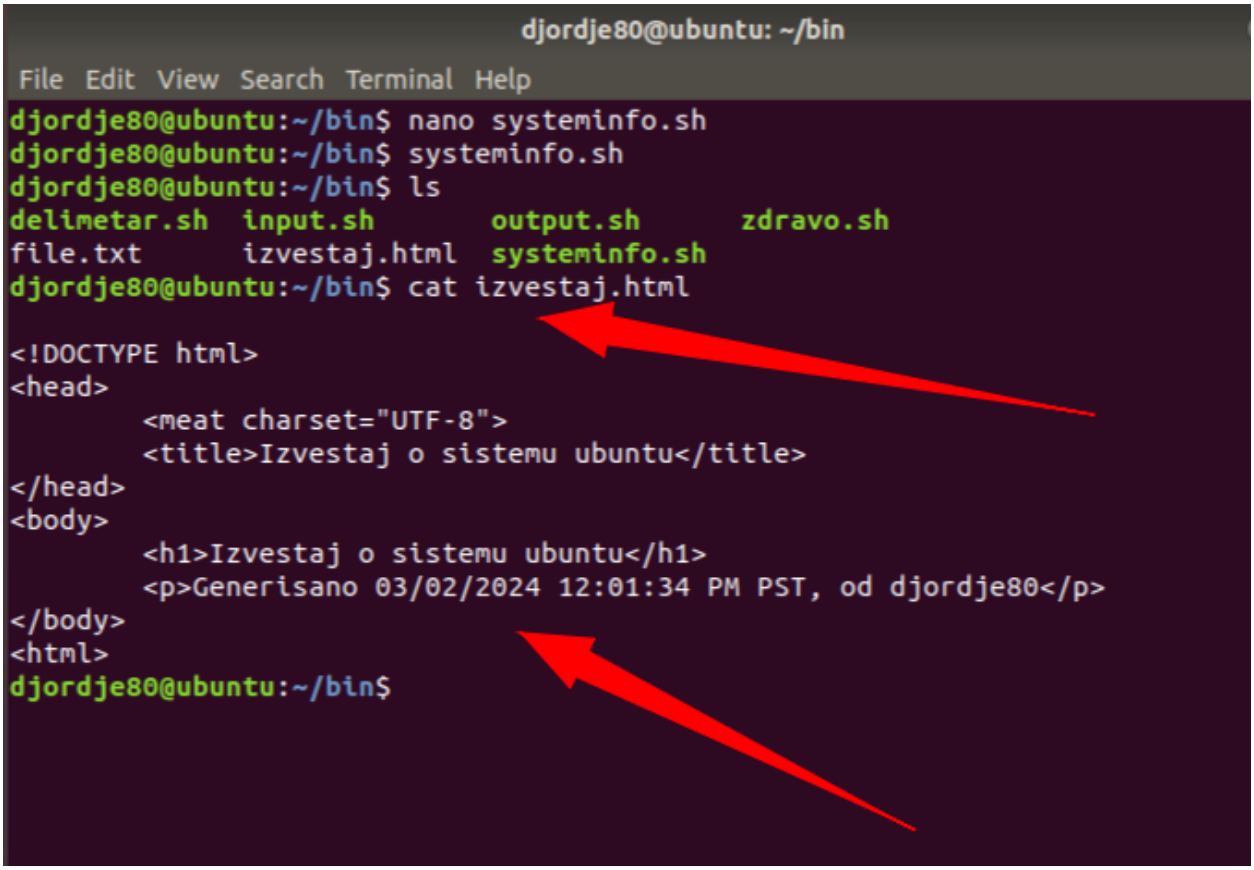
Varijable možemo menjati kada mi želimo.



I sada kada smo to odradili da pogledamo naše skripte i da pogledamo koji novi fajl smo dobili. To je izvestaj.html

Sa cat izvestaj.html možemo pogledati taj fajl





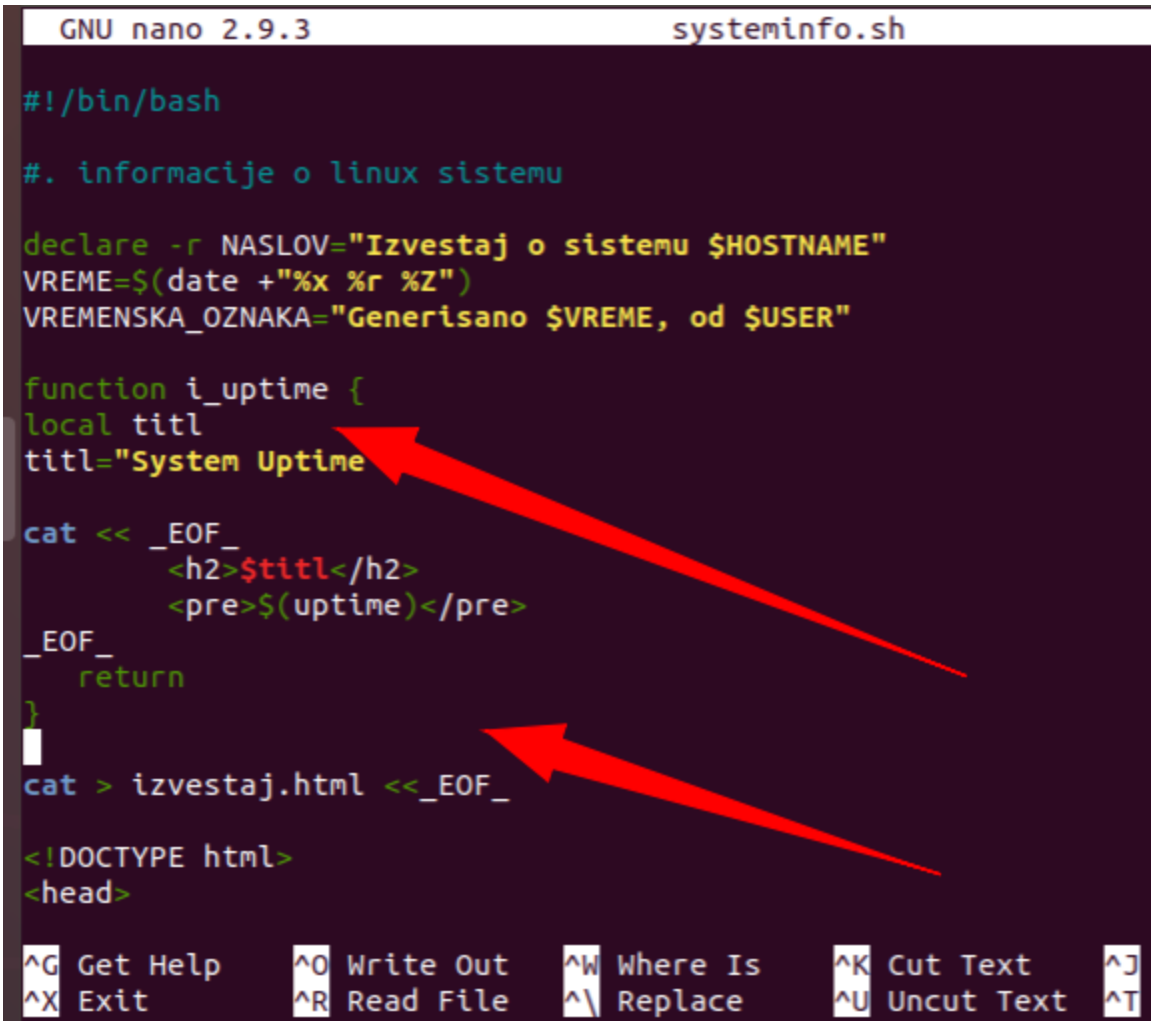
Šta vidimo sve ovde. Imamo izvestaj o mom sitemu ubuntu, genesrisano datum i vreme, moje korisničko ime.

Da se vratimo u našu program.

Funkcije su izdvojeni deloviti skup naredbi koji izvršavaju neki zadatak. Kada trebamo rešiti neki veliki zadatak i kada vidimo da će se kod ponavljati mi tu možemo koristiti funkciju.

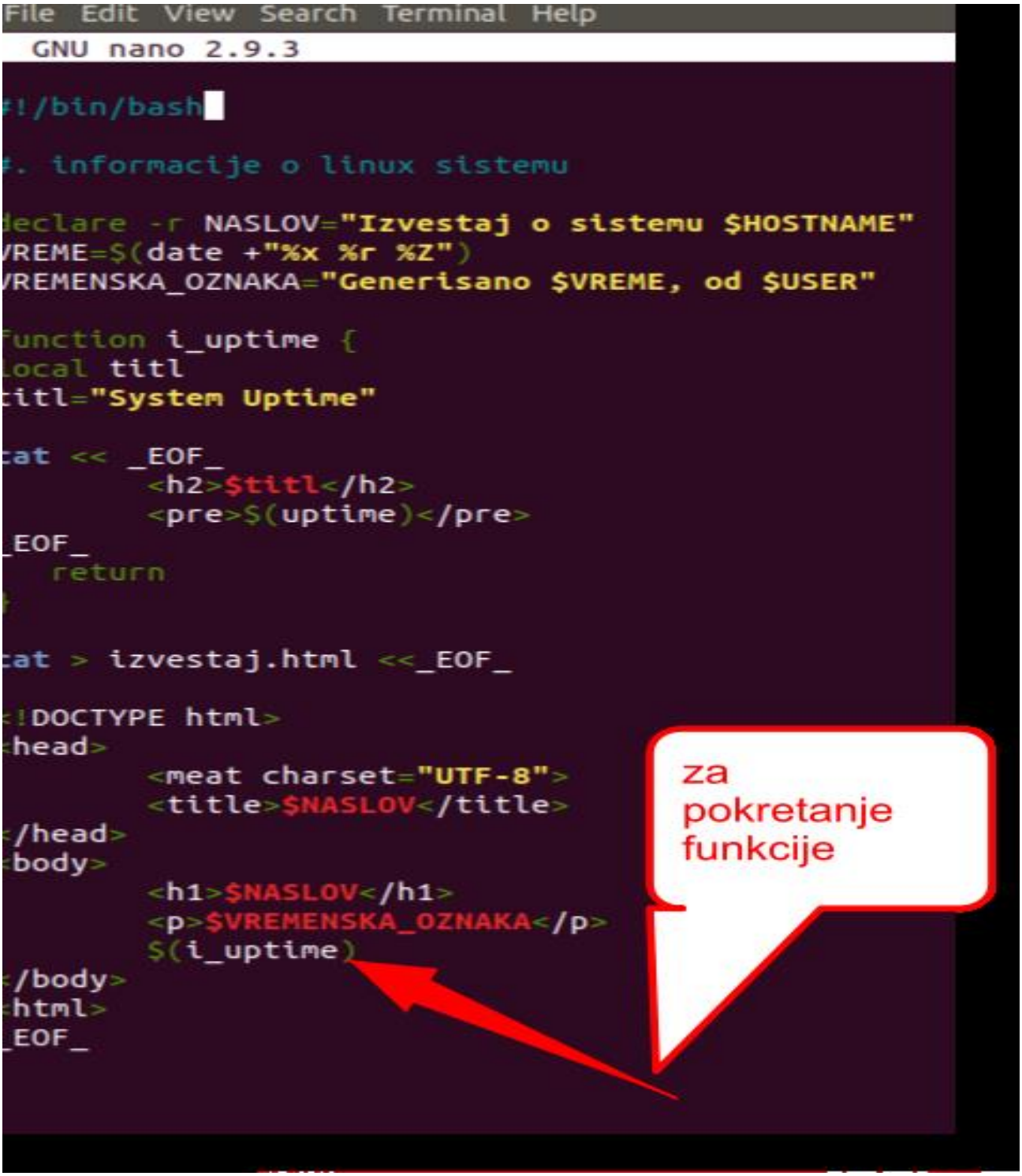
Da pogledamo u našem primeru.

U_kucajmo sada jednu funkciju sa lokalnom promenjivom titl



Funkcija se poziva u bash skripti na sledeci način.

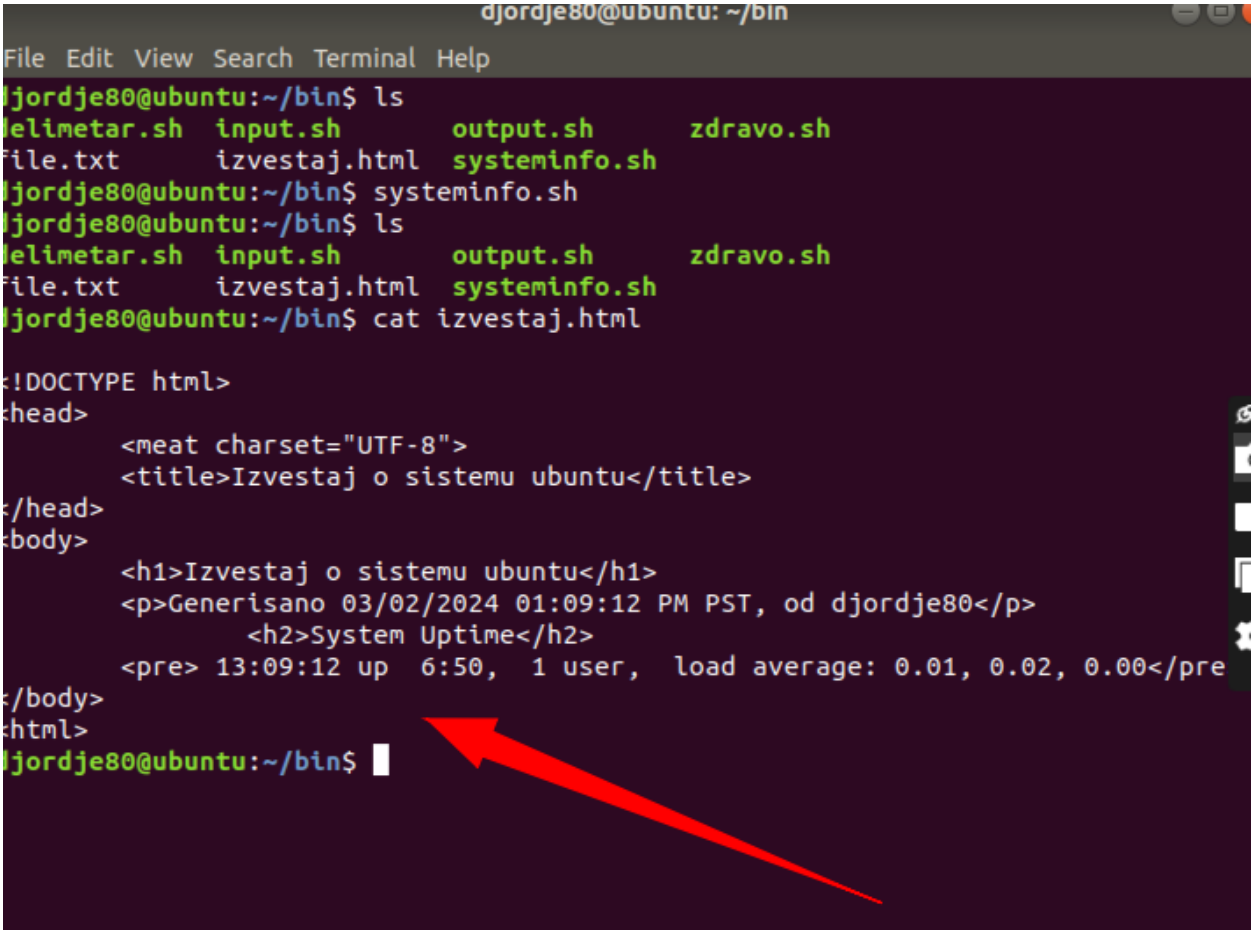
Kucaćemo naziv skripte ali je moramo staviti u zagradi, kod nas u primeru kucamo $(i_uptime)



Snimimo I pokrenimo našu skriptu

systeminfo.sh pa pogledajmo cat izvestaj.html

Šta nam je on pokazao, pokazao nam je prosečno vreme rada u sistemu

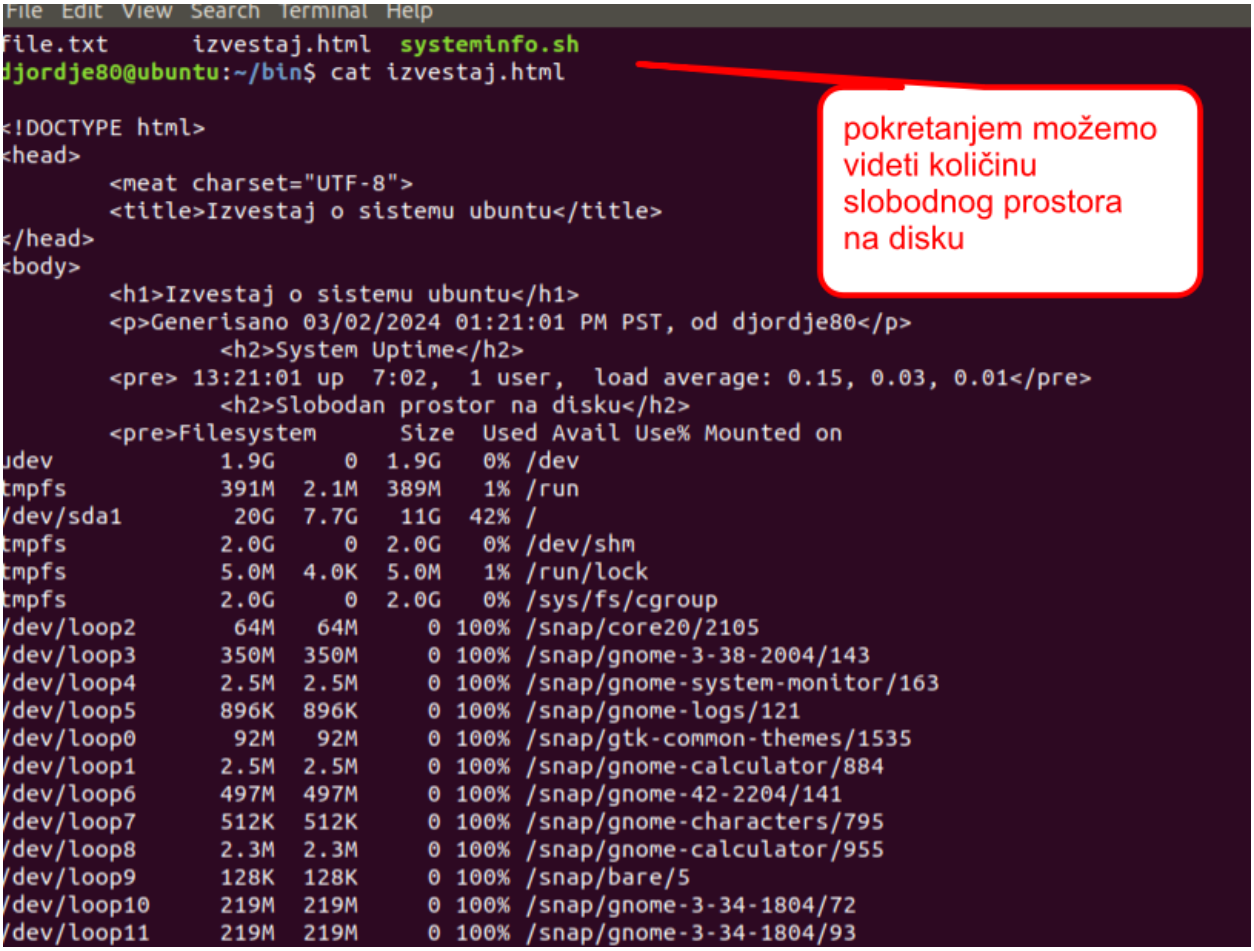


Ubacimo sada još jednu funkciju

Takođe moramo znati da je isto da pišemo function i_uptime ili i_uptime ()

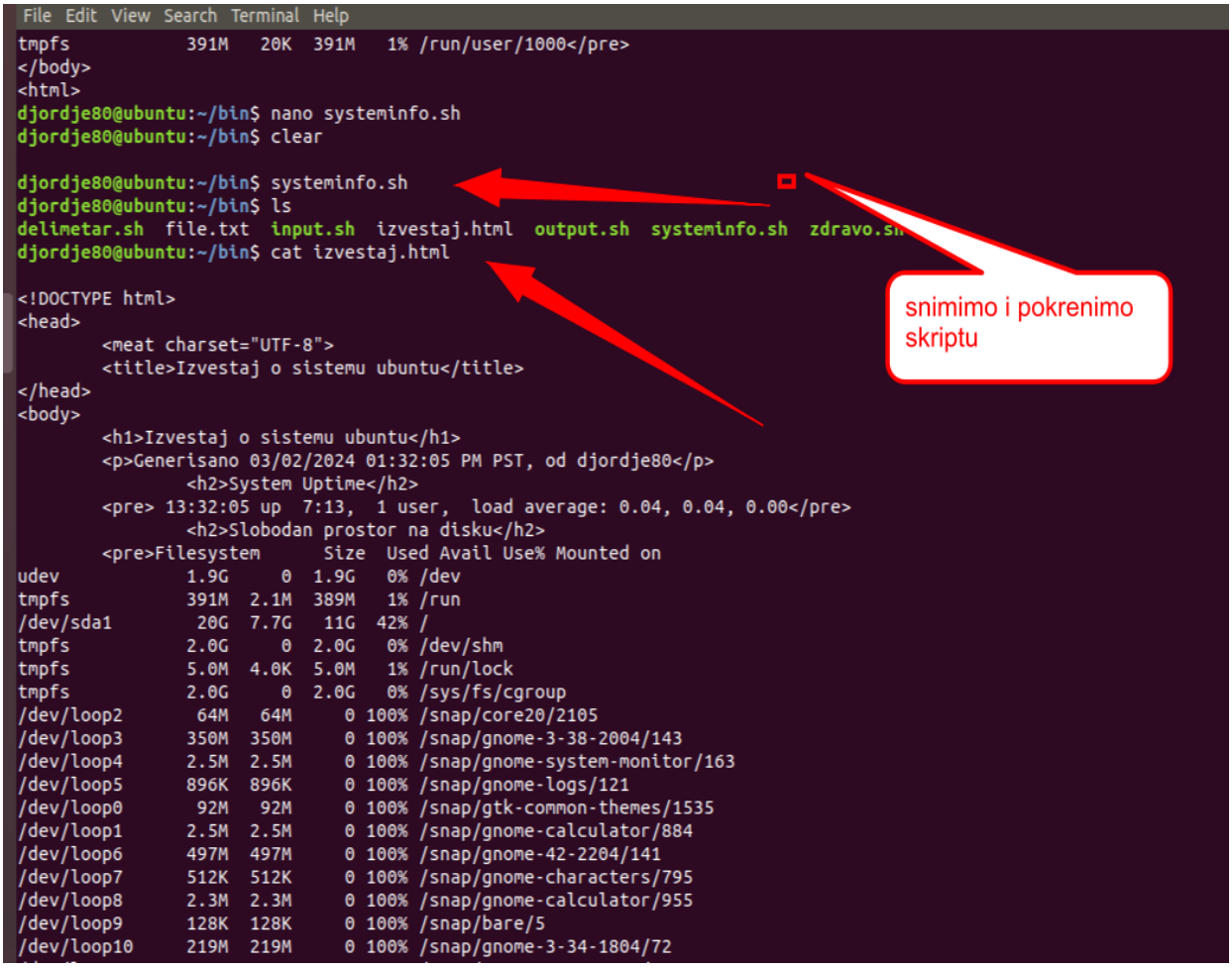
Sada napisimo funkciju koja određuje količinu slobodnog prostora na disku


File Edit View Search Terminal Help
GNU nano 2.9.3
#. informacije o linux sistemu
declare -r NASLOV="Izvestaj o sistemu $HOSTNAME"
VREME=$(date +"%x %r %Z")
VREMENSKA_OZNAKA="Generisano $VREME, od $USER"
function i_uptime {
local titl
titl="System Uptime"
cat << _EOF_
<h2>$titl</h2>
<pre>$(uptime)</pre>
EOF
return
}
i_disk_space() {
cat << _EOF_
<h2>Slobodan prostor na disku</h2>
<pre>$(df -h)</pre>
EOF
return
}
cat > izvestaj.html <<_EOF_
<!DOCTYPE html>
<head>
<meat charset="UTF-8">
<title>$NASLOV</title>
</head>
<body>
<h1>$NASLOV</h1>
$VREMENSKA_OZNAKA
$(i_uptime)
$(i_disk_space)
</body>
<html>
EOF



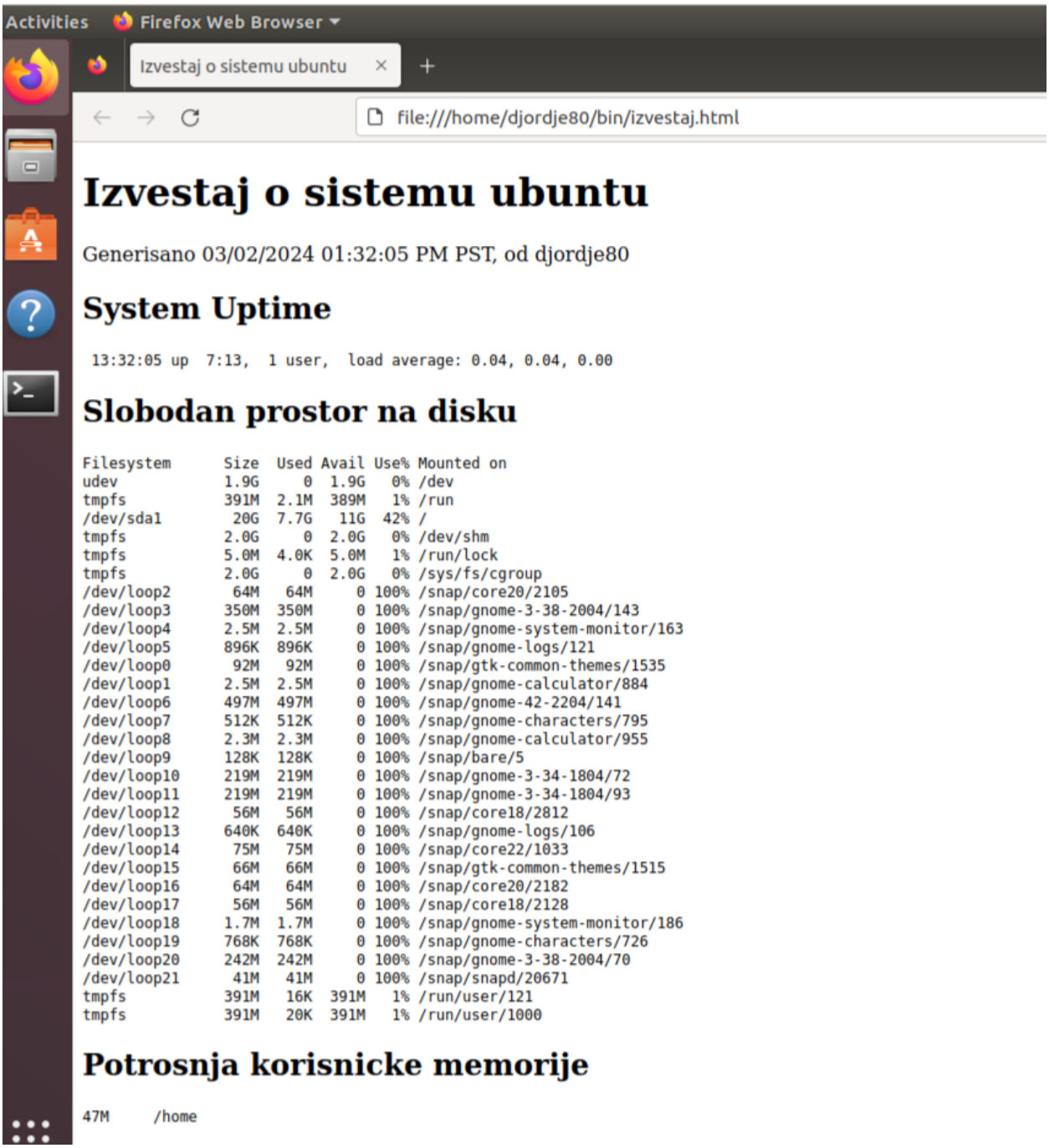
I evo još jedne funkcije za primer, pogledaćemo potrošnju korisničke memorije sada.

Sada je snimimo i pokrenemo .



U zavisnosti šta imate instalirano od browser-a kucajmo sad da vidimo sta smo uradili.

firefox izvestaj.html I dobiću mali izvestaj o informaciju o mom sistemu



Izvestaj o sistemu ubuntu

Generisano 03/02/2024 01:32:05 PM PST, od djordje80

System Uptime

```
 13:32:05 up  7:13,  1 user,  load average: 0.04, 0.04, 0.00
```

Slobodan prostor na disku

```
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           391M  2.1M  389M   1% /run
/dev/sda1        20G  7.7G   11G  42% /
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           2.0G     0  2.0G   0% /sys/fs/cgroup
/dev/loop2       64M   64M     0 100% /snap/core20/2105
/dev/loop3      350M  350M     0 100% /snap/gnome-3-38-2004/143
/dev/loop4      2.5M  2.5M     0 100% /snap/gnome-system-monitor/163
/dev/loop5      896K  896K     0 100% /snap/gnome-logs/121
/dev/loop0       92M   92M     0 100% /snap/gtk-common-themes/1535
/dev/loop1      2.5M  2.5M     0 100% /snap/gnome-calculator/884
/dev/loop6      497M  497M     0 100% /snap/gnome-42-2204/141
/dev/loop7      512K  512K     0 100% /snap/gnome-characters/795
/dev/loop8      2.3M  2.3M     0 100% /snap/gnome-calculator/955
/dev/loop9      128K  128K     0 100% /snap/bare/5
/dev/loop10     219M  219M     0 100% /snap/gnome-3-34-1804/72
/dev/loop11     219M  219M     0 100% /snap/gnome-3-34-1804/93
/dev/loop12      56M   56M     0 100% /snap/core18/2812
/dev/loop13     640K  640K     0 100% /snap/gnome-logs/106
/dev/loop14      75M   75M     0 100% /snap/core22/1033
/dev/loop15      66M   66M     0 100% /snap/gtk-common-themes/1515
/dev/loop16      64M   64M     0 100% /snap/core20/2182
/dev/loop17      56M   56M     0 100% /snap/core18/2128
/dev/loop18     1.7M  1.7M     0 100% /snap/gnome-system-monitor/186
/dev/loop19     768K  768K     0 100% /snap/gnome-characters/726
/dev/loop20     242M  242M     0 100% /snap/gnome-3-38-2004/70
/dev/loop21      41M   41M     0 100% /snap/snapd/20671
tmpfs           391M   16K  391M   1% /run/user/121
tmpfs           391M   20K  391M   1% /run/user/1000
```

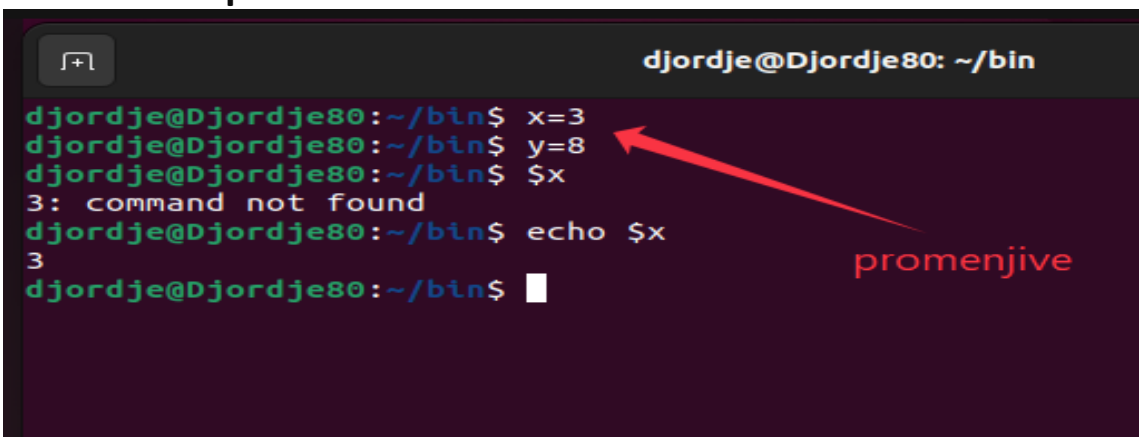
Potrosnja korisnicke memorije

```
47M     /home
```

7. Iskazi odlučivanja... if... elif... else... i operatori poređenja...

Iskazi odlučivanja se u zavisnosti od programskog jezika u kome radimo pišu različito.
Pokazaćemo i operatore poređenja zato što bez njih mi i ne možemo pisati iskaze.
Uđemo u naš dir gde su bash skripte.
Iskaz odlučivanja – (ako ispunjavaš ovaj uslov onda uradi ovu stvar … ili ako ispunjavaš ovaj uslov uradi ovo i ako ispunjava ovaj uslov uradio i ovo….)

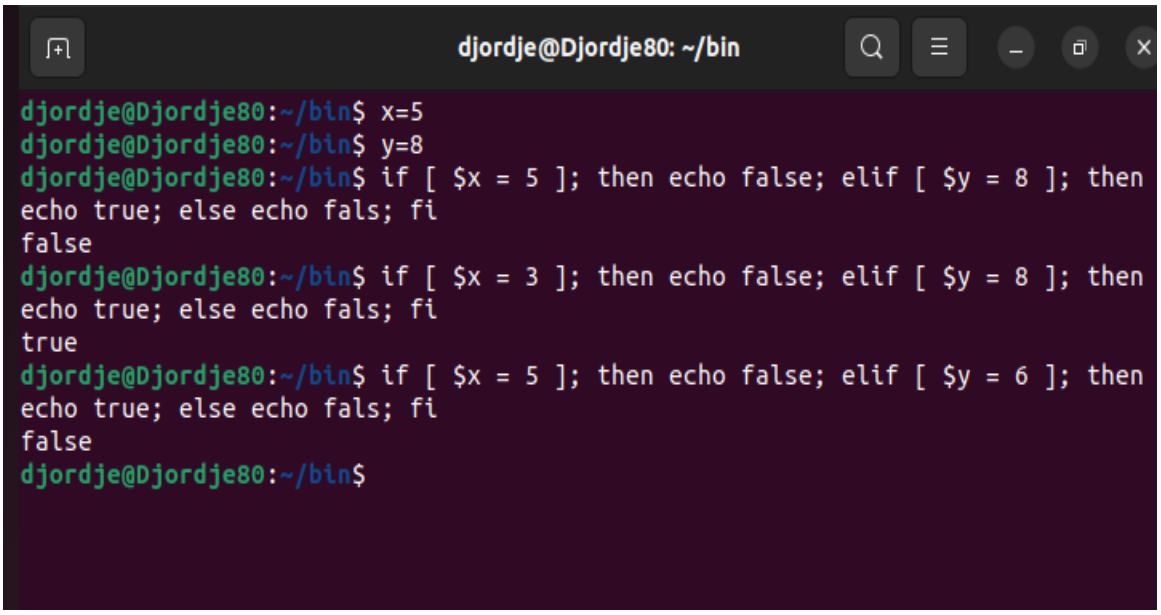
Da vidimo u primeru.




```
djordje@Djordje80:~/bin$ x=3
djordje@Djordje80:~/bin$ y=8
djordje@Djordje80:~/bin$ $x
3: command not found
djordje@Djordje80:~/bin$ echo $x
3
djordje@Djordje80:~/bin$
djordje@Djordje80:~/bin$ if [ $x=3 ]; then  echo true; fi
true
```

```
djordje@Djordje80:~/bin$ if [ $y = 58 ]; then echo  true ; else echo false; fi
false
djordje@Djordje80:~/bin$
```

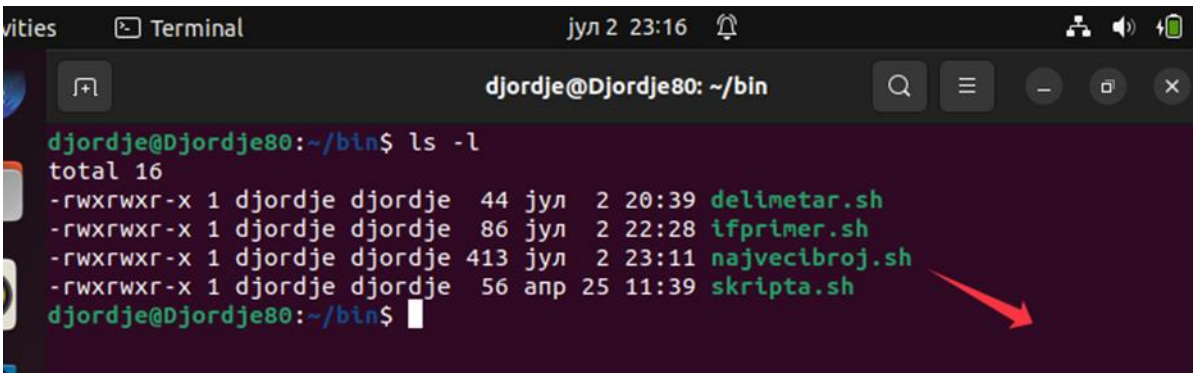
Obratite pažnju gde stavljate (;) takođe morate obraćati pažnju gde ima a gde ne razmaka u tekstu.

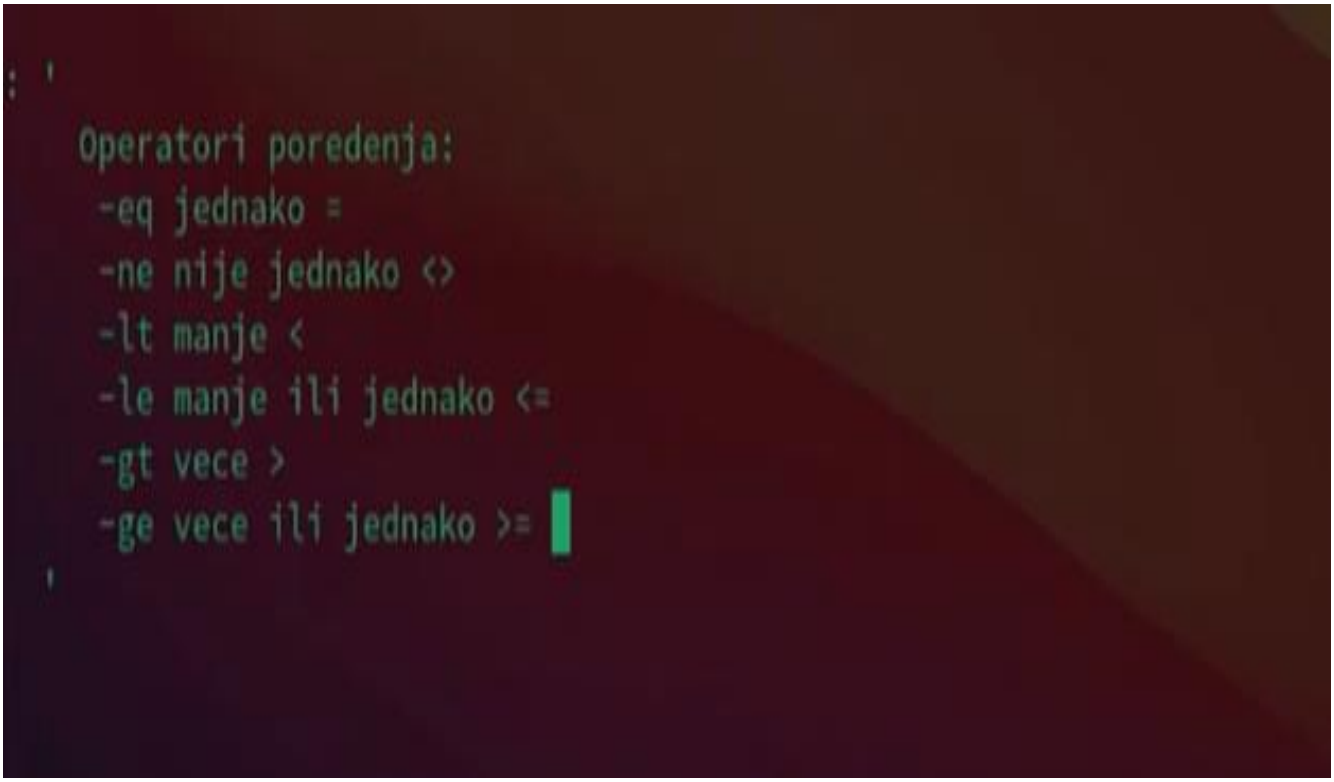


Imamo skriptu najvecibroj.sh Videcemo sta ona radi .

Od ponuđenih tri broja koji su integer određuje koji je broj najveći.

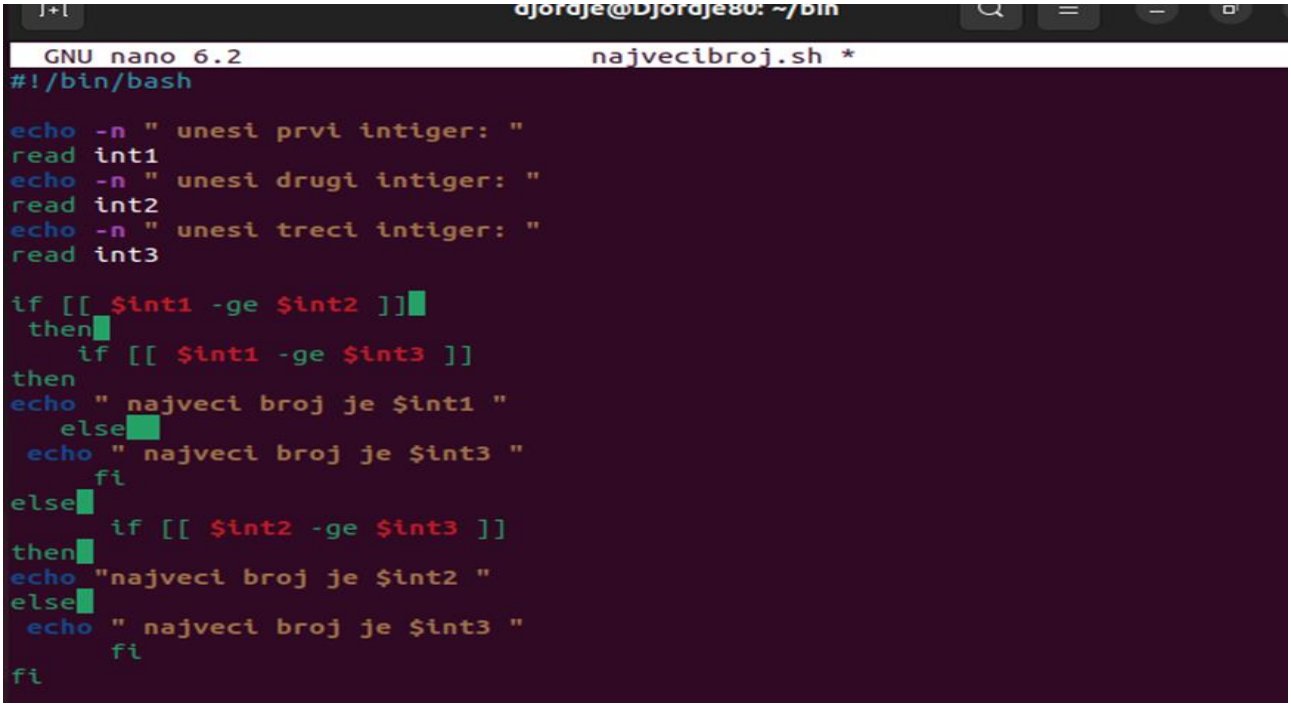
Kao sto možemo videte iz priloženog tu imamo if else naredbe





Naravno sa komandom chmod +x najvecibroj.sh napravio sam joj ovlasćenja da postane izvršna.

Nije više tekstualni fajl već postaje bash skripta.

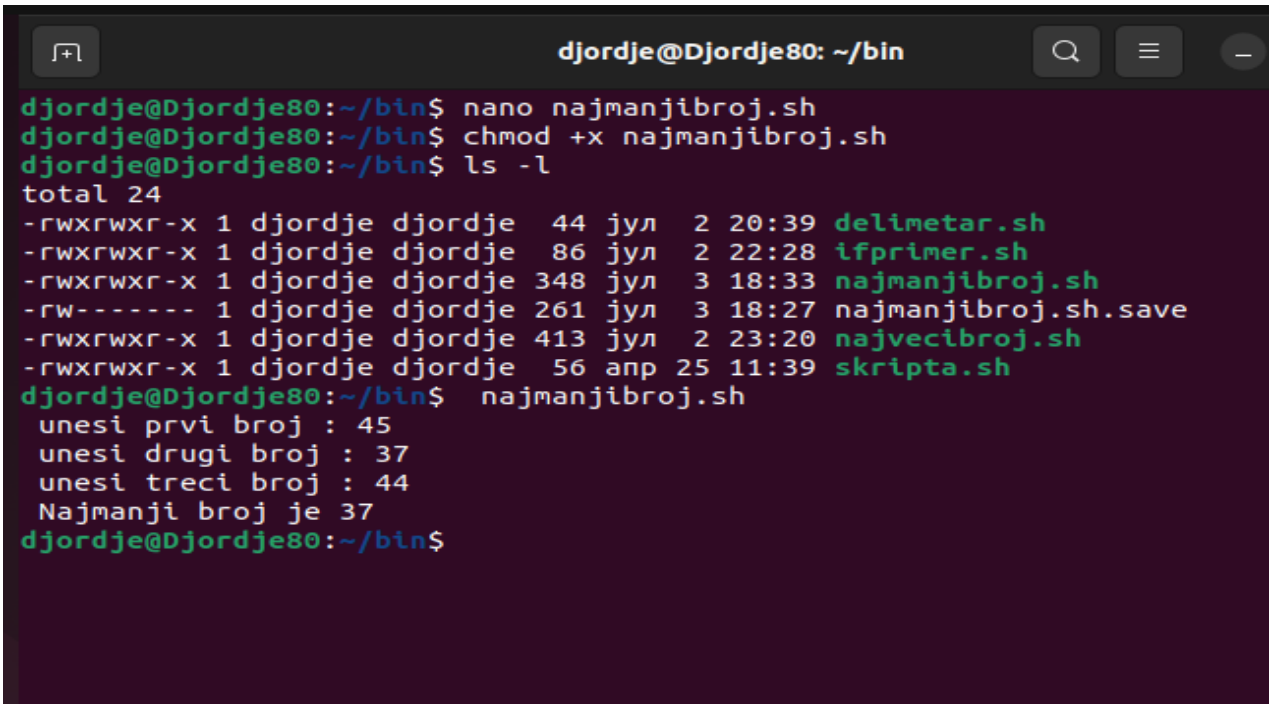


8. Iskazi odlučivanja... if... elif... else... i logičke operacije...

Napravimo skriptu da nađe koji je od ponuđenih brojeva najmanji broj.

Umesto operatora poređenja koristićemo logičke operacije što će nam I skratiti kod u pisanju.

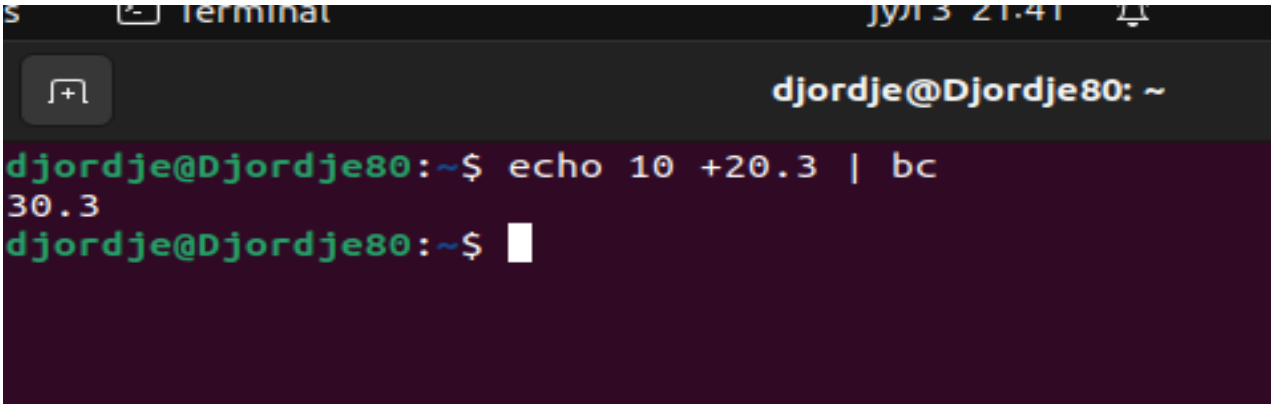

djordje@Djordje80: ~/bin
GNU nano 6.2 najmanjibroj.sh
#!/bin/bash
echo -n " unesi prvi broj : "
read int1
echo -n " unesi drugi broj : "
read int2
echo -n " unesi treci broj : "
read int3
if [[int1 -le int2]] && [[int1 -le int3]]
then echo " Najmanji broj je $int1 "
elif [[int2 -le int1]] && [[int2 -le int3]]
then echo " Najmanji broj je $int2 "
else echo " Najmanji broj je $init3 "
fi


djordje@Djordje80: ~/bin
djordje@Djordje80:~/bin$ nano najmanjibroj.sh
djordje@Djordje80:~/bin$ chmod +x najmanjibroj.sh
djordje@Djordje80:~/bin$ ls -l
total 24
-rwxrwxr-x 1 djordje djordje 44 jyn 2 20:39 delimetar.sh
-rwxrwxr-x 1 djordje djordje 86 jyn 2 22:28 ifprimer.sh
-rwxrwxr-x 1 djordje djordje 348 jyn 3 18:33 najmanjibroj.sh
-rw------- 1 djordje djordje 261 jyn 3 18:27 najmanjibroj.sh.save
-rwxrwxr-x 1 djordje djordje 413 jyn 2 23:20 najvecibroj.sh
-rwxrwxr-x 1 djordje djordje 56 anp 25 11:39 skripta.sh
djordje@Djordje80:~/bin$ najmanjibroj.sh
unesi prvi broj : 45
unesi drugi broj : 37
unesi treci broj : 44
Najmanji broj je 37
djordje@Djordje80:~/bin$

Kao što vidimo iz priloženog dosta je smanjen kod uz pomoć logičkih operacija

9. Iskazi odlučivanja... case... *) ... esac ... kodiramo kalkulator..

Poprilično je teško snaći se i koristiti aritmetičke operacije u bash skriptama.
Instaliraćemo prvo pomoć za to tzv. Basic kalkulator
Instaliraćemo tj kucamo# sudo pacman -S bc

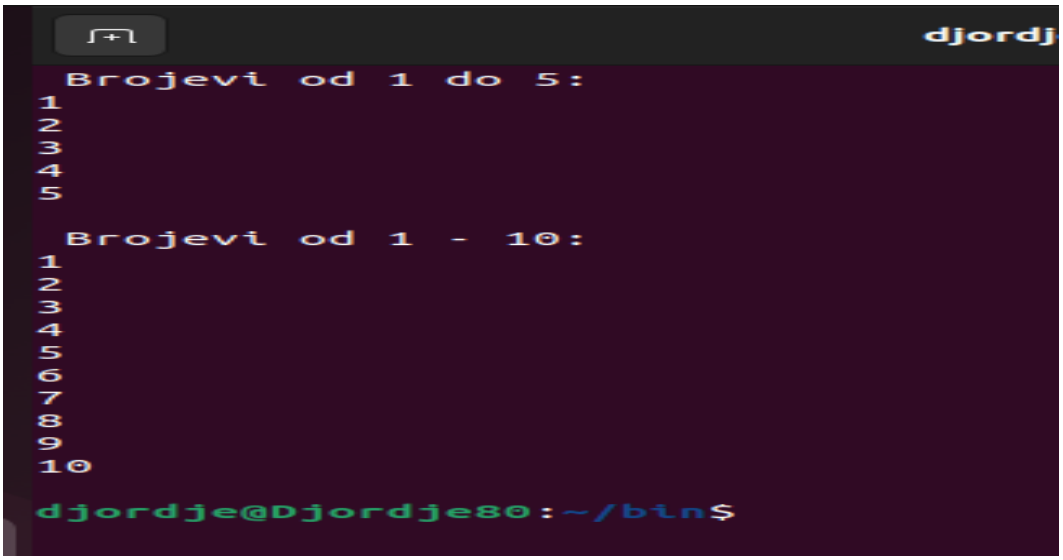


10. Kako da pišete for petlje u bash skriptama ?

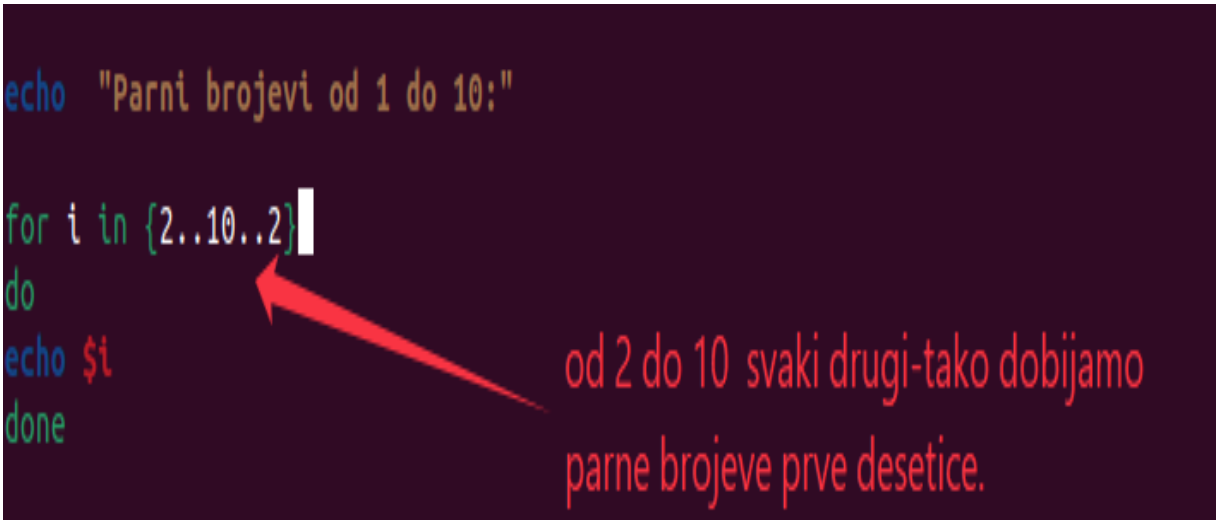
Napravimo jedan primer for petlje

On ce ispisivati

Kao sto vidite nasa naredba clear je prvo ocistila ekran
Ajmo to isto samo da se ispisuju brojevi do 10.
Necemo ih ponovo navoditi pojedinacno kao u predhodnom primeru.



Da vidimo isti primer samo kako zapisati parne brojeve do 10



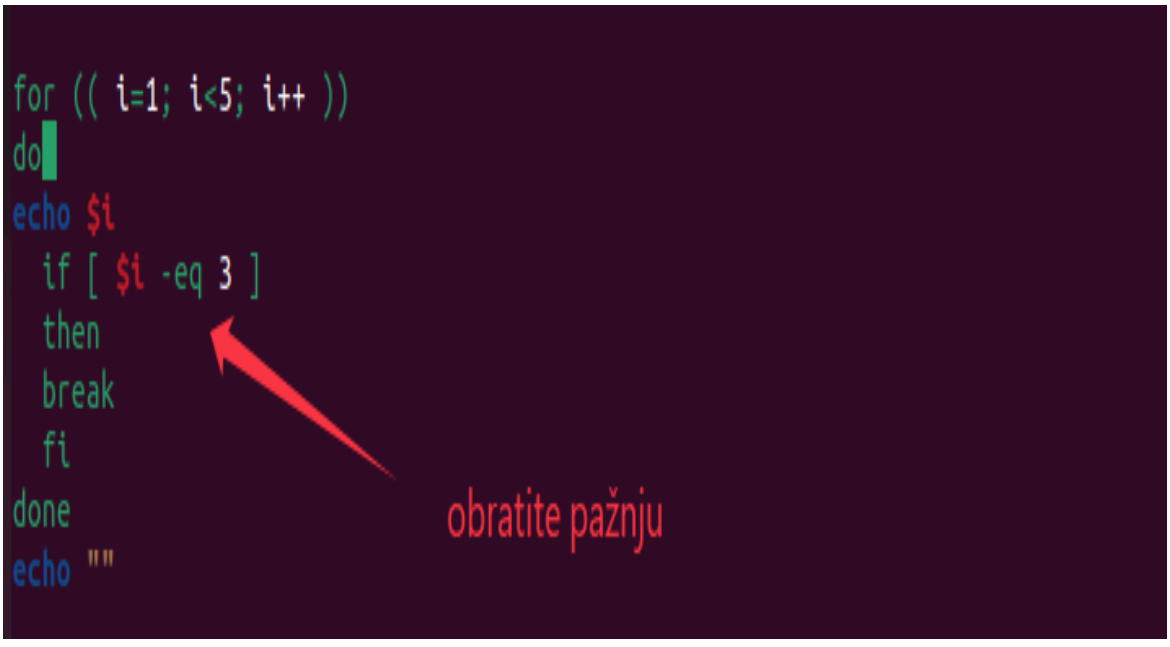
```
Parni brojevi od 1 do 10:
2
4
6
8
10

djordje@Djordje80:~/bin$
```

Da smo želeli neparne brojeve onda bi krenuli od jedinice a ne dvojke.

Da vidimo sledeći primer

Koristimo for komandu kao i if koju smo već ranije koristili.

Hoćemo da ispišemo samo prva tri broja u nizu od pet brojeva.



Ako hoćemo da promenimo malo tačnije neka se na trećem mestu nešto ispiše a onda neka se nastavi uslov.

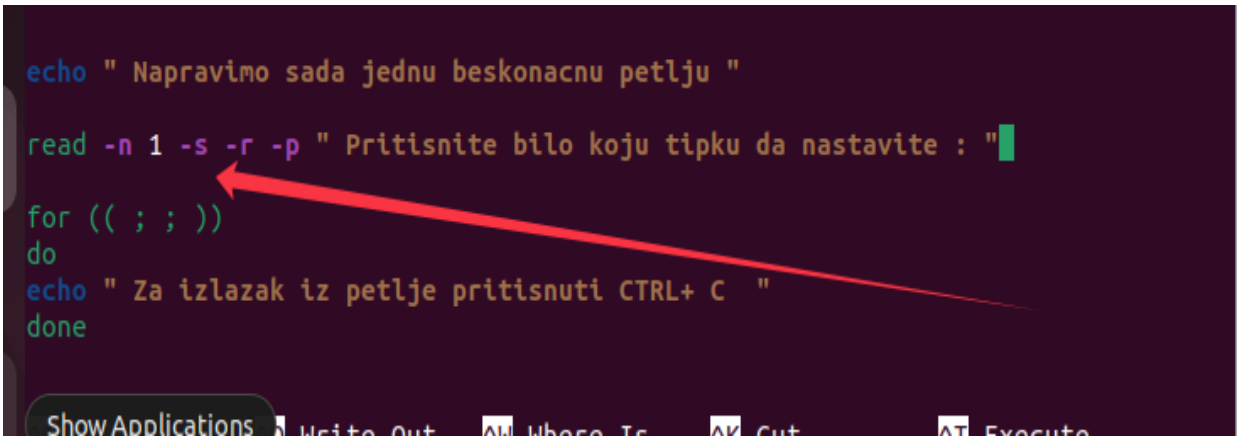
Bilo bi ovako nešto

```
for (( i=1; i<5; i++ ))
do
echo $i
  if [ $i -eq 3 ]
  then
        echo " Mala promena na trecem mestu  "
  continue
  fi
done
echo ""
```

Program bi ispisivao sledeću stvar

```
1
2
3
 Mala promena na trecem mestu
4
```

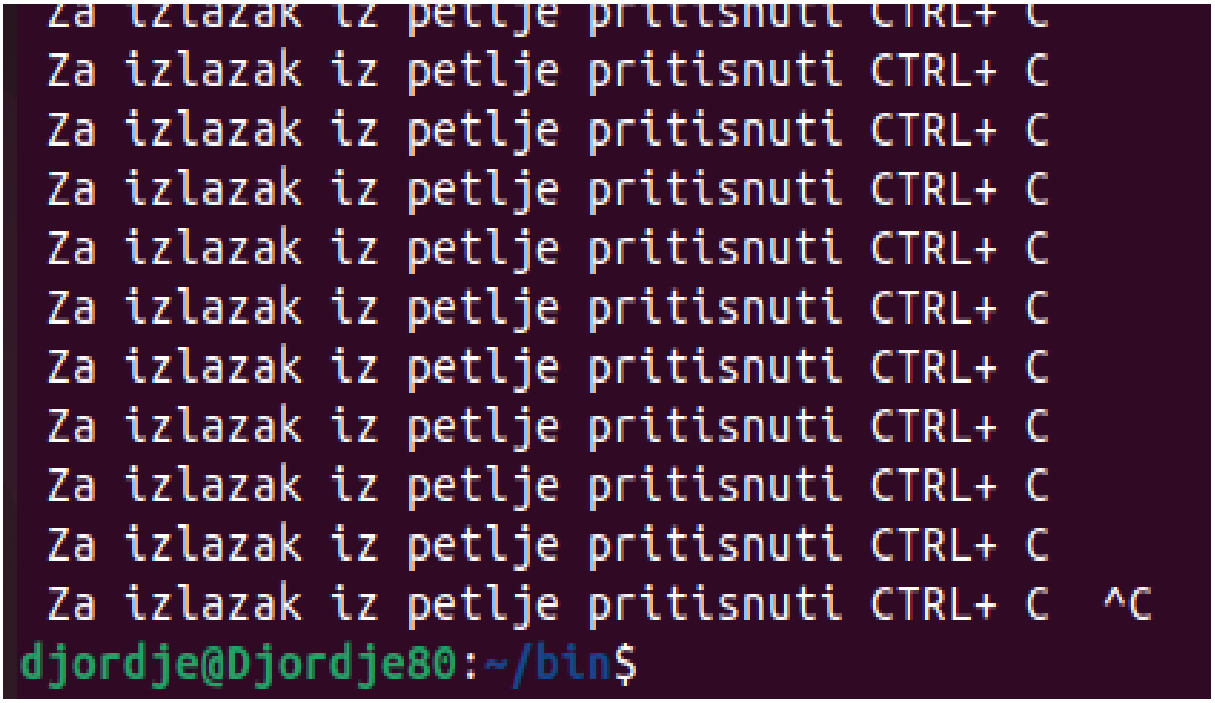
Da napravimo jednu beskonačnu petlju.



Moramo obratiti paznju na sve ove atribute. Da je samo -n onda moramo pritusnuti samo enter. Zato dodajemo ostale atribute 1 tipka, da bude sakriveno. Kada stavimo sve atribute

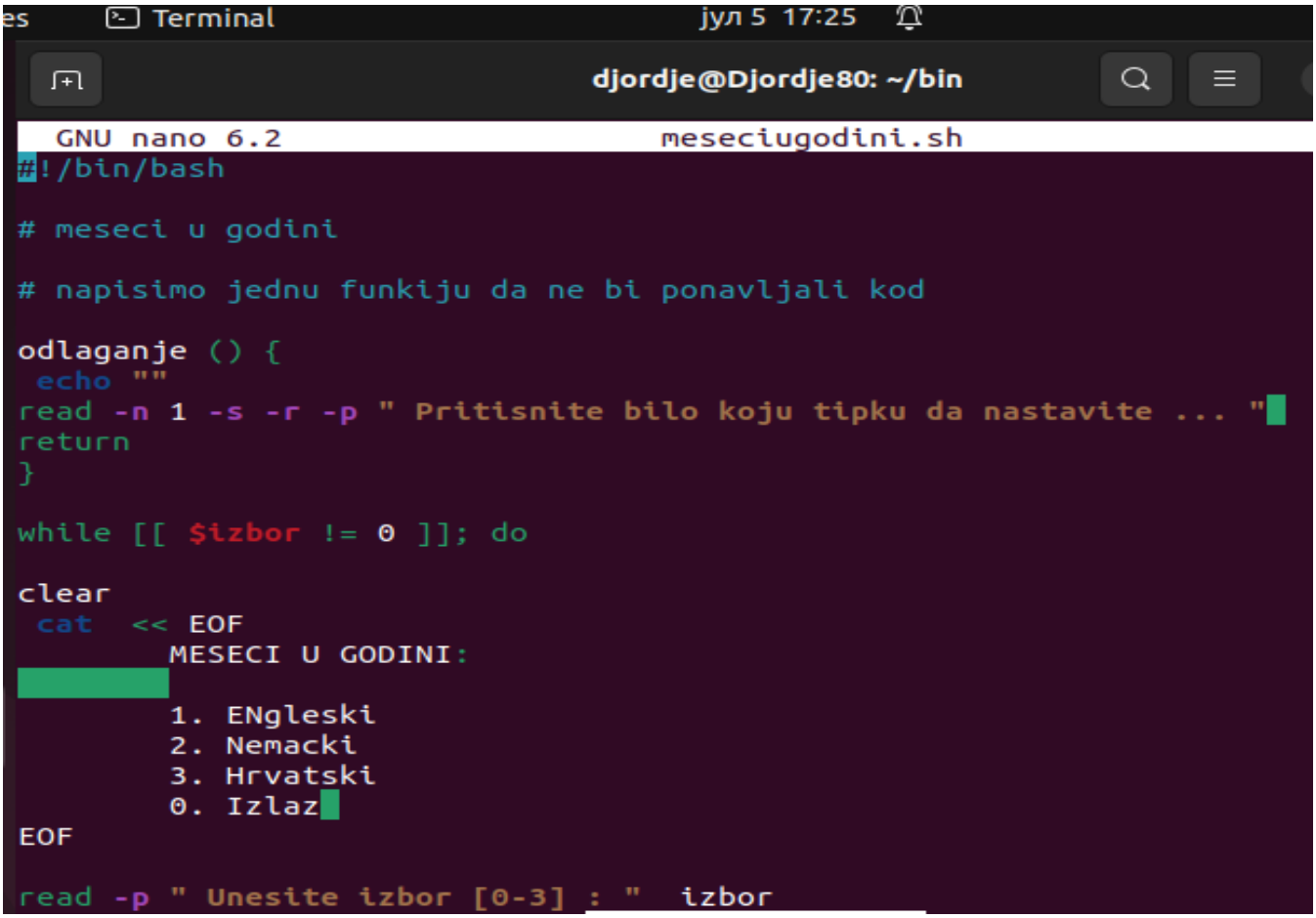
Onda se može uzeti u obzir da pritiskom na bilo koju tipku mi nastavljamo na beskonacnu petlju.

Ovako to izgleda dok ne pritisnemo naredbu za prekid .



11. Kako da pišete while petlju i pravite meni u bash skripti ?

Da bi smo videli kako se koristi while petlja koristićemo primer gde ćemo napisati mesece u godini na primer tri jezika.


es
Terminal
јул 5 17:25
djordje@Djordje80: ~/bin
GNU nano 6.2
meseciugodini.sh
#!/bin/bash
# meseci u godini
# napisimo jednu funkiju da ne bi ponavljali kod
odlaganje () {
echo ""
read -n 1 -s -r -p " Pritisnite bilo koju tipku da nastavite ... "
return
}
while [[$izbor != 0]]; do
clear
cat << EOF
MESECI U GODINI:
1. ENgleski
2. Nemacki
3. Hrvatski
0. Izlaz
EOF
read -p " Unesite izbor [0-3] : " izbor


if [[$izbor =~ ^[0-3]$]]; then
if [[$izbor == 1]]; then
clear
cat << EOF
Month in year :
1. January
2. February
3. March
4. April
5. May
6. June
7. July
8. Avgust
9. September
10. October
11. November
12. December
EOF
odlaganje
fi

```
      if  [[ $izbor == 2 ]]; then
                clear

        cat << EOF
                Monate in ein jahr :

                1. Januar
                2. Februar
                3. Marz
                4. April
                5. Mai
                6. Juni
                7. Juli
                8. Avgust
                9. September
               10. October
               11. November
               12. Dezember
EOF
  odlaganje
  fi
```

```
      if  [[ $izbor == 3 ]]; then
                clear

        cat << EOF
                Mjeseci u godini :

                1. Sijecanj
                2. Veljaca
                3. Ozuljak
                4. Travanj
                5. Svibanj
                6. Lipanj
                7. Srpanj
                8. Kolovoz
                9. Rujan
               10. Listopad
               11. Studeni
               12. Prosinac
EOF
  odlaganje
  fi

 else
  echo " Pogresan izbor    "
   odlaganje
```

```
fi

done
```

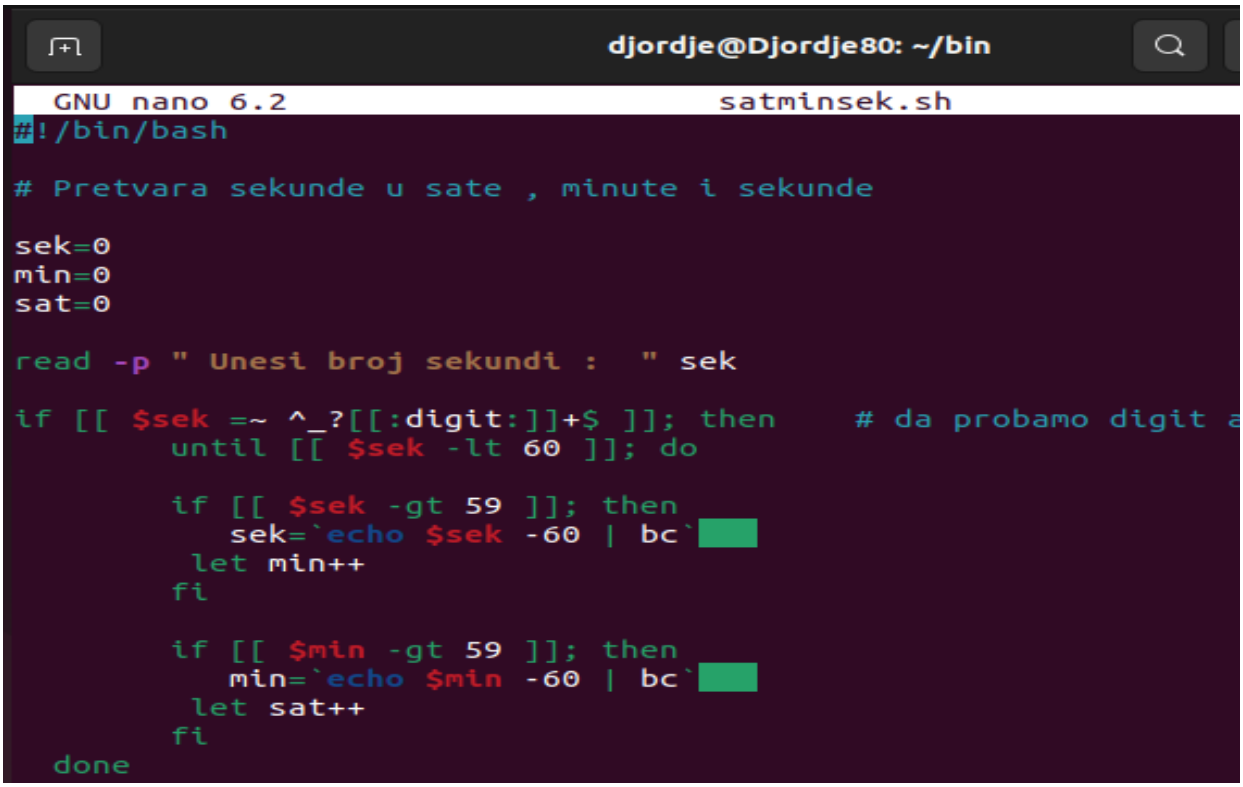
12. Kako da pišete until petlju u BASH skripti ?

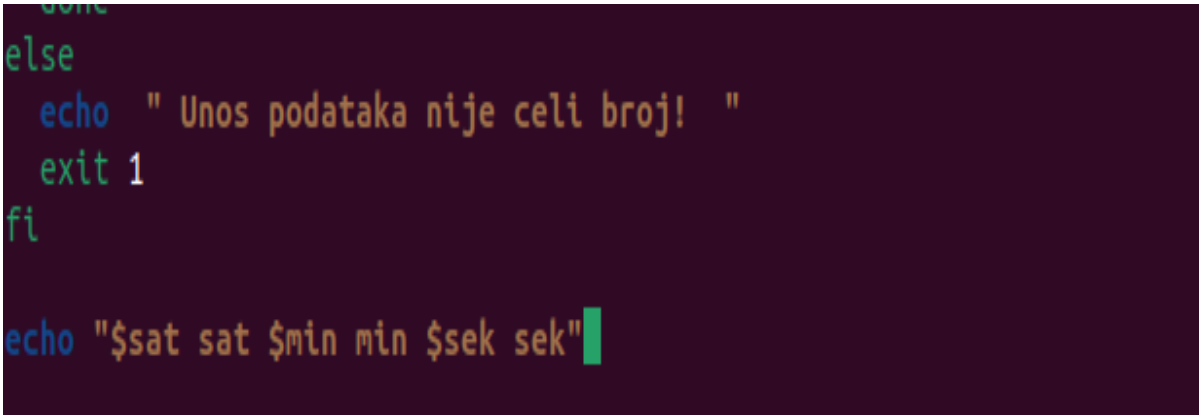
Koja je razlika. Petlja while se izvrsava sve dok je uslov ispunjen.

Dok petlja Until ona se izvrsava sve dok uslov nije ispunjen. Kada se on ispuni izlazi se iz skripte.

Ajmo da pogledamo neki primer.

Skripta koja pretvara recimo sekunde u sekunde minute i sate.



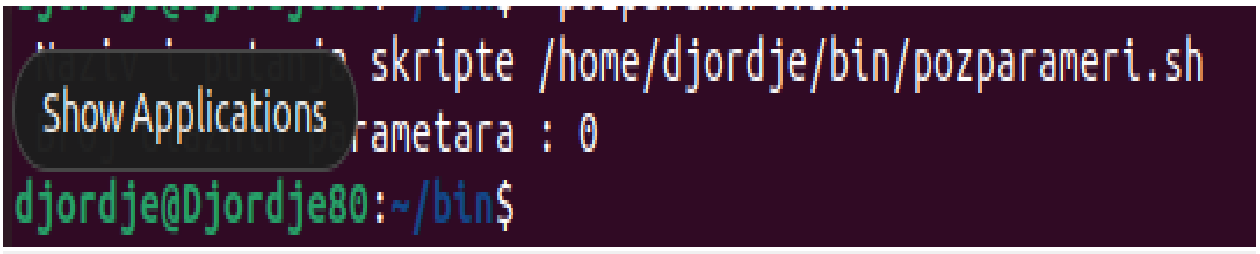


13. Pozicioni parametri i kako se koriste u BASH skriptama

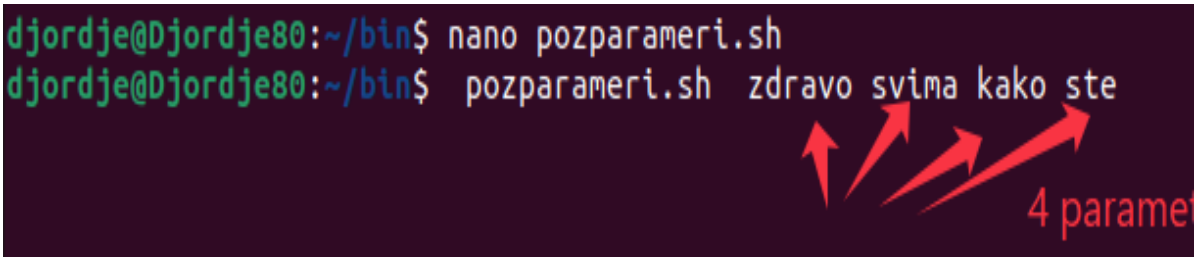
Pozicioni parametri sluze da uz pomoć skripti direktno sa komandne linije prihvatamo neke argumente, opcije koje se koriste u skriptama.

Onda naše skripte postaju puno bolje i korisnije.

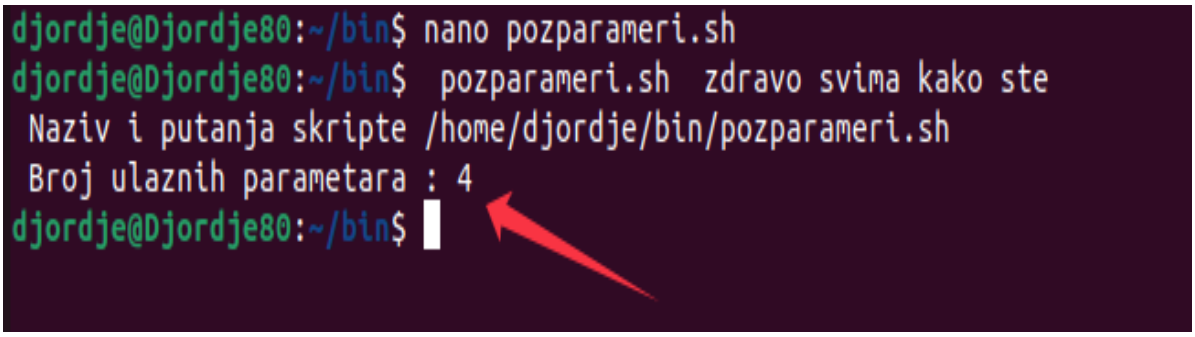
Da vidimo sta je on uradio

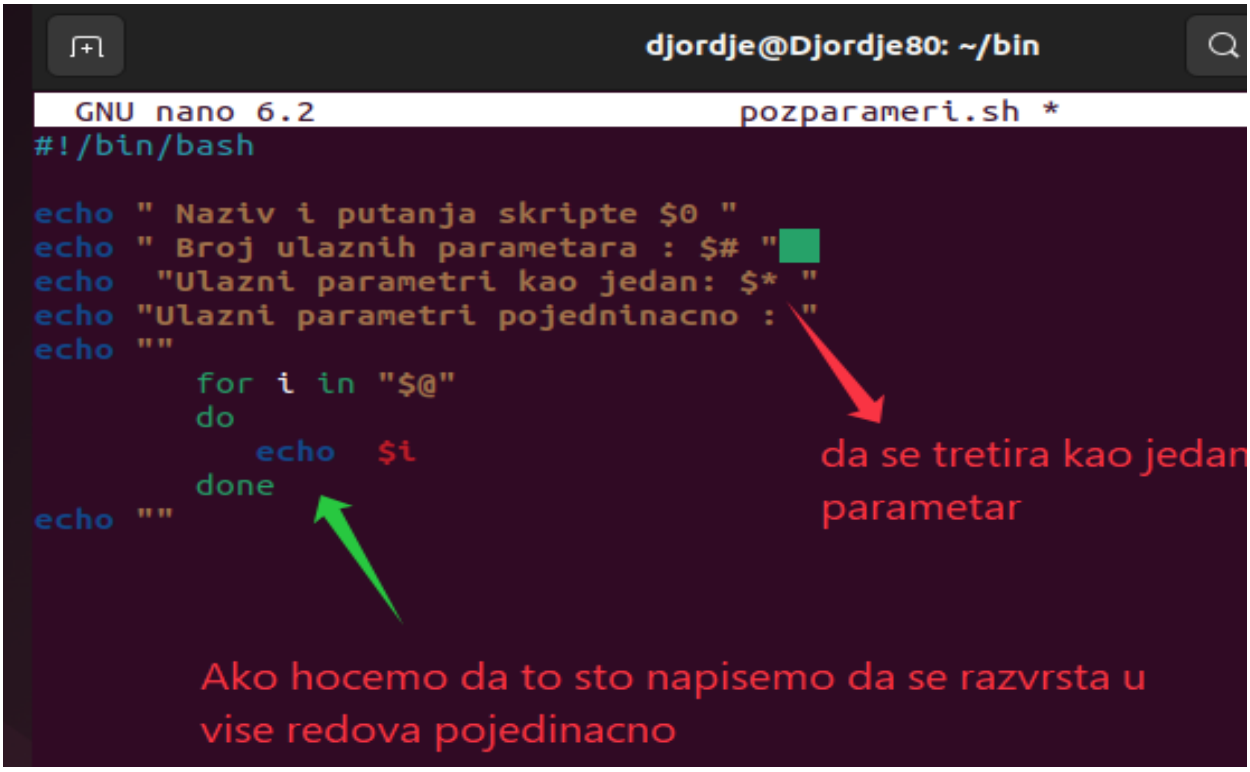


Kao sto vidimo nema ulaznih parametara pa je ta vrednost 0



Kao sto vidite uneti tekst je 4 parametara



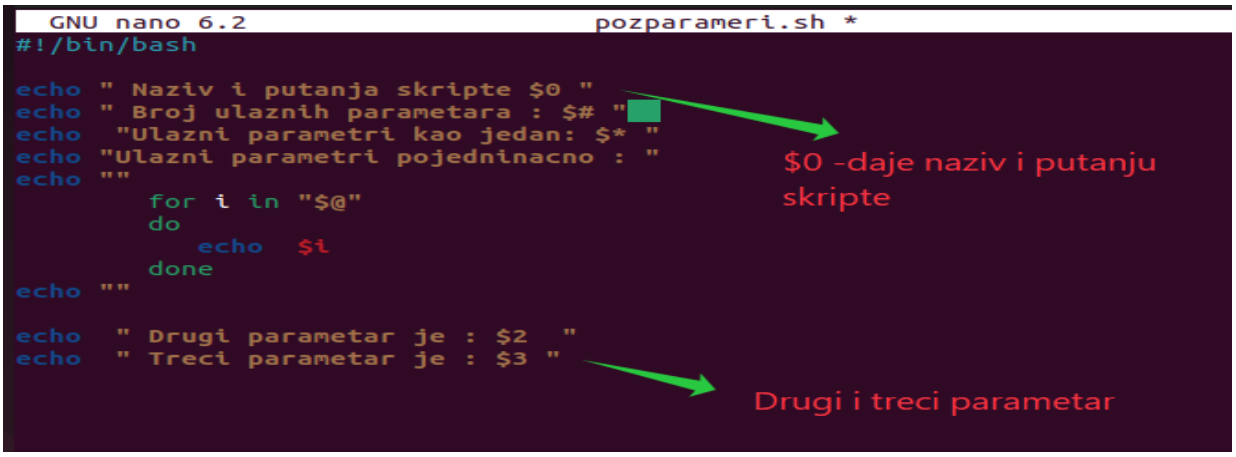


Dobićemo sledeće

```
djordje@Djordje80:~/bin$  pozparameri.sh Ucite programiranj
 Naziv i putanja skripte /home/djordje/bin/pozparameri.sh
 Broj ulaznih parametara : 4
Ulazni parametri kao jedan: Ucite programiranje trebace vam
Ulazni parametri pojedninacno :

Ucite
programiranje
trebace
vam

djordje@Djordje80:~/bin$
```

Ako želimo od naprimer 5 parametara treba nam samo drugi

Onda bi uradili sledeće.



Da vidimo primer za gore navedeno

```
djordje@Djordje80:~/bin$  pozparameri.sh Dobar dan svima ima li sta novo
 Naziv i putanja skripte /home/djordje/bin/pozparameri.sh
 Broj ulaznih parametara : 7
Ulazni parametri kao jedan: Dobar dan svima ima li sta novo
Ulazni parametri pojedninacno :

Dobar
dan
svima
ima
li
sta
novo

 Drugi parametar je : dan
 Treci parametar je : svima
djordje@Djordje80:~/bin$
```

Možemo pratiti kroz pozicione parametre i samu skriptu kako radi.

Svaki proces ima svoj broj tako da možemo i ovde to videti

Takođe ako svaki put kada zavrsimo bilo koju komandu, skriptu mi dobijemo određeni broj.

Ako je taj broj nula onda je ta skripta uspešno izvršena. Svaki drugi broj daje kao značenje neku grešku.

Pogledajmo u našem primeru navedeno.

```
echo "PID; $$ "    # daje nam  PID broj
echo "Exit code : $? " # aKO JE 0 ONDA JE SKRIPTA USPESNO IZVRSENA
                       # AKO JE BILO KOJI DRUGI BROJ NIJE
```

```
djordje@Djordje80:~/bin$  pozparameri.sh
 Naziv i putanja skripte /home/djordje/bin/pozparameri.sh
 Broj ulaznih parametara : 0
Ulazni parametri kao jedan:
Ulazni parametri pojedninacno :


 Drugi parametar je :
 Treci parametar je :

PID; 7968
Exit code : 0
djordje@Djordje80:~/bin$
```

14. Nizovi u BASH skriptama ..

Slobodno mozemo reci da su nizovi promenjive u koje mozemo staviti vise vrednosti.

U zavisnosti koliko smo mi odredili da budu elemenata.

Oni se dele na elemente,vrednosti i indekse.

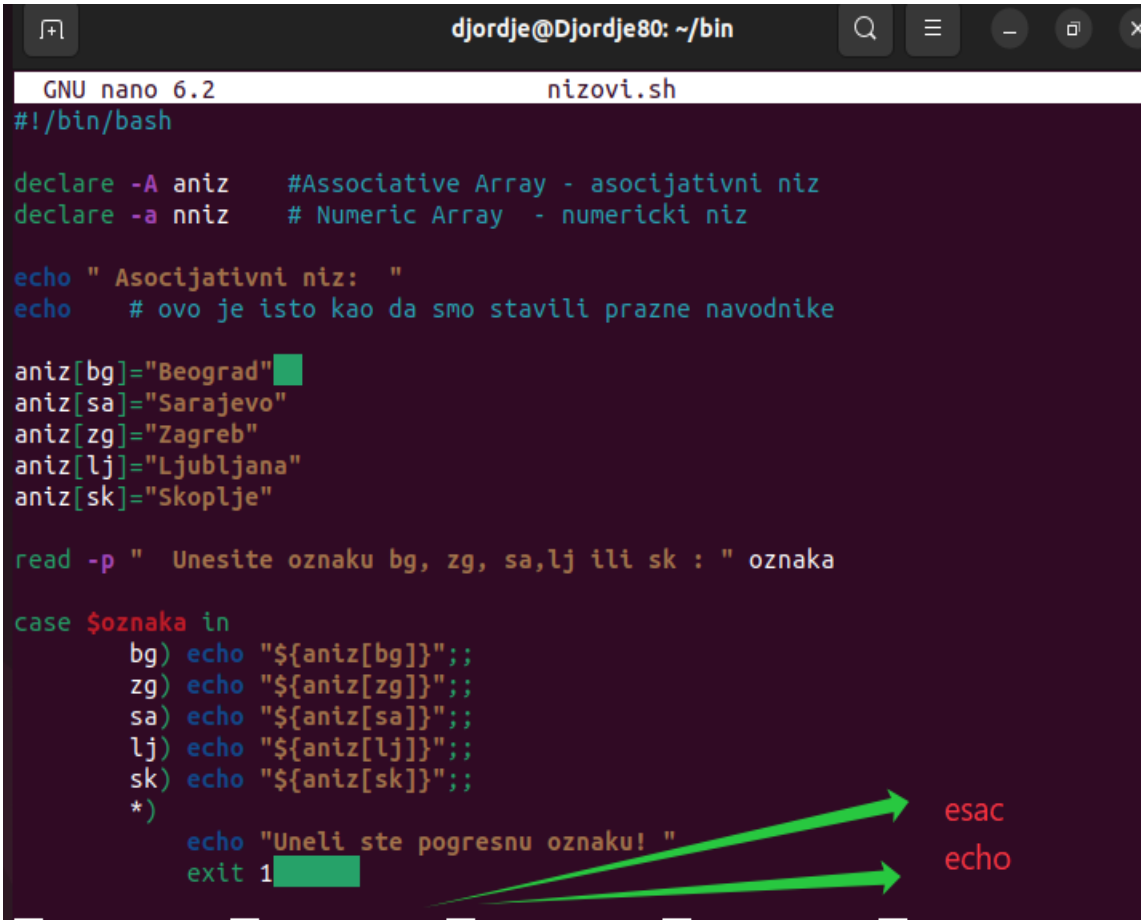
Indeksi su mesta gde se neka vrednost smesta/ Ovde su nizovi veoma ograniceni tj u bash skriptama mozemo praviti jednodimenzionalne nizove samo.

Delimo na asocijativne I numeričke. Drugacije se deklarišu I koriste dugacije skripte.

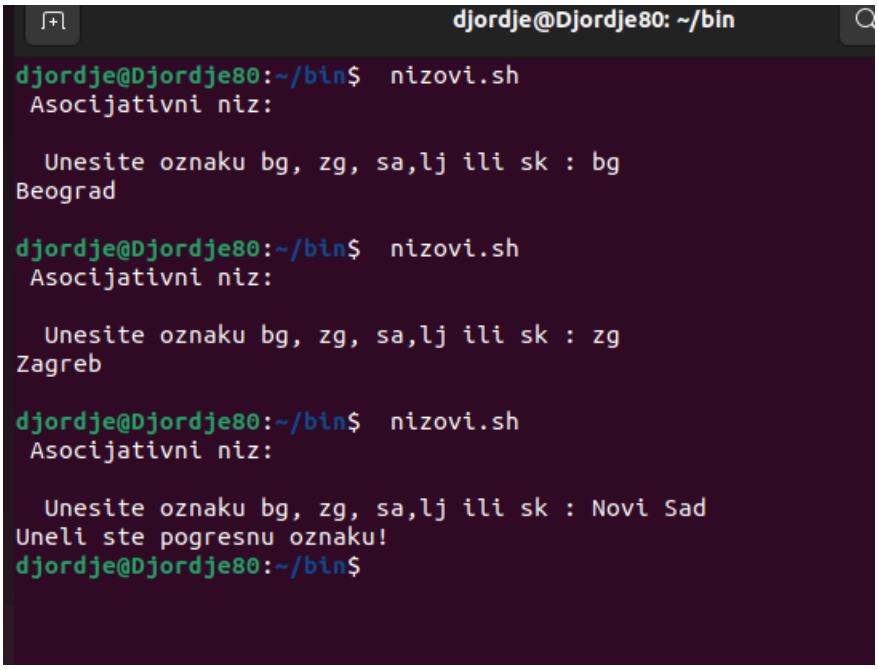
Asocijativan niz on znaci da od indeska ne koristimo broj već neke reči, ključeve

Svaki niz se sastoji od indeska I elemenata.

Da vidimo jedan primer asocijativnog niza



Program će uraditi



Da vidimo u istom primeru i numericki niz

```
echo "Numericki niz:"
echo

nniz[0]="Beograd"
nniz[1]="Sarajevo"
nniz[2]="Zagreb"
nniz[3]="Ljubljana"
nniz[4]="Skoplje"

echo "${nniz[@]} "
 # @ oznacava da nam se prikazuje celi niz
```

Pokazivaće sledeće

```
Numericki niz:

Beograd Sarajevo Zagreb Ljubljana Skoplje
djordje@Djordje80:~/bin$
```

Ako bi želeli da pokazuje grad ispod grada onda moramo napisati sledeće

```
echo "Numericki niz:"
echo

nniz[0]="Beograd"
nniz[1]="Sarajevo"
nniz[2]="Zagreb"
nniz[3]="Ljubljana"
nniz[4]="Skoplje"

 #  echo "${nniz[@]} "
 # @ oznacava da nam se prikazuje celi niz


        for  i in "${nniz[@]} "
        do
          echo $i
        done
        echo
```

Gde smo dobili ono što smo želeli.

```
djordje@Djordje80:~/bin$ nano nizovi.sh
djordje@Djordje80:~/bin$  nizovi.sh
 Asocijativni niz:

  Unesite oznaku bg, zg, sa,lj ili sk : bg
Beograd

Numericki niz:

Beograd
Sarajevo
Zagreb
Ljubljana
Skoplje
```