

Bash script

Su programski jezik sam po sebi . On se interpretira 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 izvršavati

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.

```
↳ cat /etc/shells
# Pathnames of valid login shells.
# See shells(5) for details.

/bin/sh
/bin/bash
/usr/bin/git-shell
/bin/zsh
/usr/bin/zsh
```

Treba se proveriti koje direktorijume imamo sa kom ls.

Traži se direktorijum **bin** . *On služi da mi u njemu čuvamo naše skripte. 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

```
GNU nano 5.6.1 zdravo.sh
#!/bin/bash
echo "Zdravo svete! Ovo je moj prvi komandni skript."
```

Komanda

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.

```
manuel@altos ~/bin
└─> nano zdravo.sh
└─> ls -l
total 4
-rw-r--r-- 1 manuel users 69 Apr  4 14:46 zdravo.sh
└─> chmod +x zdravo.sh
└─>
```

ili 700

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

```
manuel@altos ~/bin
└─> ls -l
total 4
-rwxr-xr-x 1 manuel users 69 Apr  4 14:46 zdravo.sh
└─> ./zdravo.sh
```

Komanda za pokretanje sk

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 skripta nalazi. Ona ne moramo kucati **./nazivskripte**

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



```
└─> echo $PATH
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/lib/jvm/default/bin:/usr/bin/site_p
erl:/usr/bin/vendor_perl:/usr/bin/core_perl:/var/lib/snapd/snap/bin
```

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 raspoložemo 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.

```
ls -a
.          Documents      .odzivnik      Templates
..         .dotnet           .odzivnik2     test
aur        Downloads      .oh-my-zsh     tmux
.bash_aliases  .eclipse       .omnisharp     tmux.conf
.bash_history  eclipse-workspace Pictures         Videos
.bash_logout  .emacs         .pki           .viminfo
.bash_profile  .emacs.d       Public         .vscode
.bashrc       .gnupg         PycharmProjects .wallpapers
.bashrc.bak   .histfile     .python_history .wget-hsts
bin          .java         share         .zcompdump
.cache       .local        .shell.pre-oh-my-zsh .zcompdump-altos-5.8
.clamtk      .mozilla     snap         .zsh_history
.config      Music        .ssh         .zshrc
.data       .mysql_history .swt         .zshrc.pre-oh-my-zsh
.Desktop    .nuget       .templateengine
```

U ovom našem direktorijum .bashrc moramo ukucati komandu.

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

```
GNU nano 5.6.1 .bashrc
#
# ~/.bashrc
#
# If not running interactively, don't do anything
[[ $- != *i* ]] && return

alias ls='ls --color=auto'
# PS1='[\u@\h \W]\$ '

blue=$(tput setaf 21);
red=$(tput setaf 160);
yellow=$(tput setaf 226);

bold=$(tput bold);
reset=$(tput sgr0);

PS1='${blue}[\u${red}@${yellow}\h \W${reset}]\$ '

# Alias definition
if [ -f ~/.bash_aliases ]; then
. ~/.bash_aliases
fi

export PATH=~/bin:"$PATH"
Save modified buffer?
Y Yes
N No ^C Cancel
```

Komanda koju pisem

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**

```
└─> ls -l
total 4
lrwxrwxrwx 1 manuel users 26 Apr  4 15:14 zdravo -> /home/manuel/bin/zdravo.sh
-rwxr-xr-x 1 manuel users 69 Apr  4 14:46 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 pravljjenjem **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

```
manuel@altos ~  
└─> cd bin  
manuel@altos ~/bin  
└─> ls  
zdravo.sh  
manuel@altos ~/bin  
└─> cp zdravo.sh
```

Kopiramo postojeći fajl

```
cp zdravo.sh output.sh
```

I on će kopirati našu skriptu

```
└─> ls  
output.sh zdravo.sh
```

kopirana je nova s

Ako pogledamo sta smo dobili neće se videti naš tekstualni 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.

```
output.sh  
manuel@altos ~/bin  
└─> ls -l  
total 12  
-r--r-- 1 manuel users 47 Apr  4 15:39 file.txt  
-r-xr-x 1 manuel users 80 Apr  4 15:37 output.sh  
-r-xr-x 1 manuel users 69 Apr  4 14:46 zdravo.sh
```

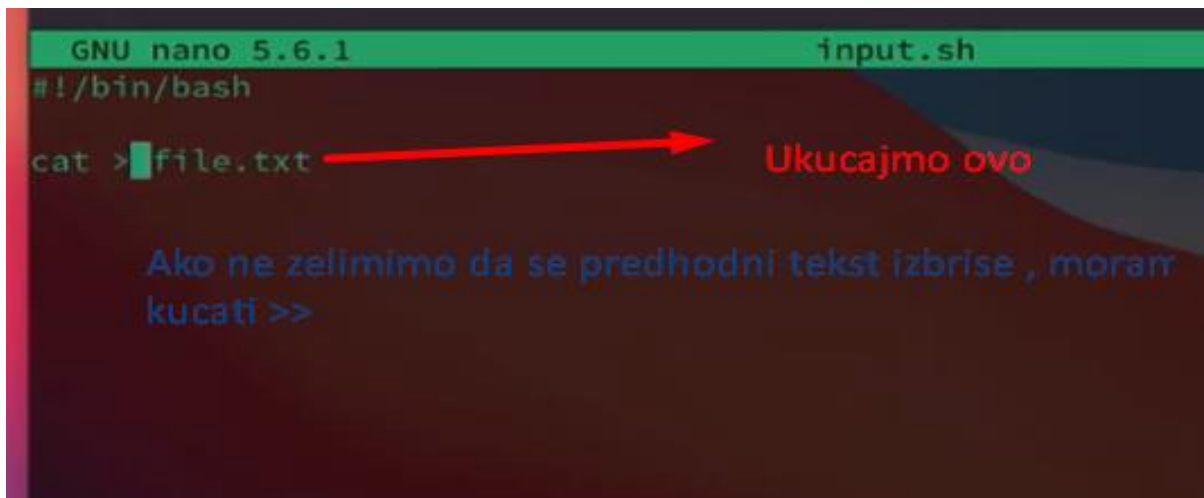
nas txt. fajl

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



```
GNU nano 5.6.1 input.sh
#!/bin/bash
cat > file.txt Ukucajmo ovo

Ako ne zelimimo da se predhodni tekst izbrise , moram
kucati >>
```

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 tekst koji se nalazi u više redova.

Da ne bi stalno za svaki red pisali echo možemo uraditi upravo sledeće

```
#!/bin/bash
```

```
cat << ovde kucamo nesto sto prestavlja naslov nase lekcije
```

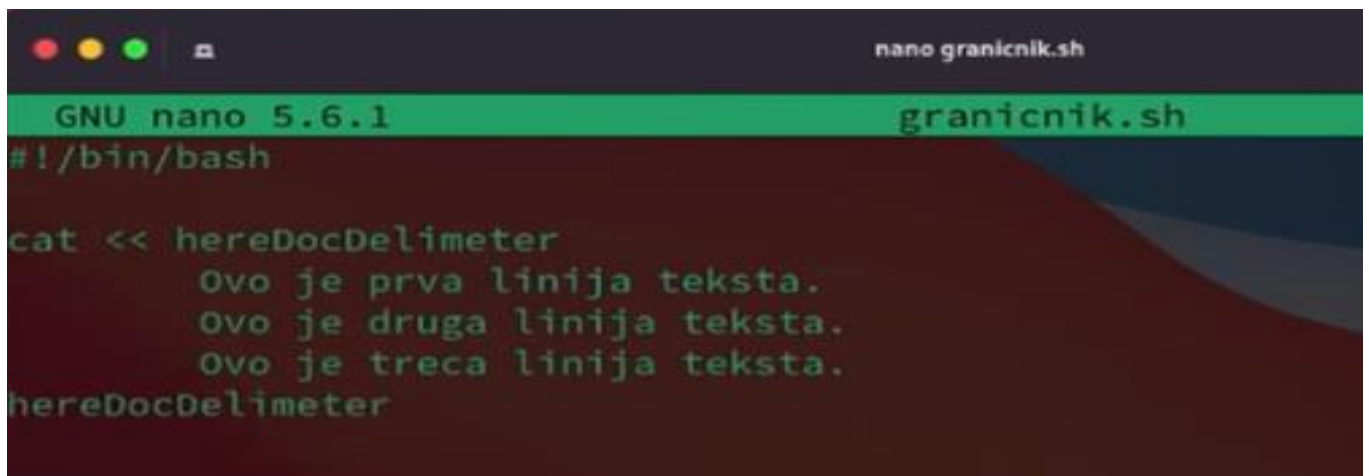
```
    tekst
```

```
    tekst
```

```
    tekst
```

.....

```
ovde kucamo nesto sto prestavlja naslov nase lekcije
```



```
nano granicnik.sh
GNU nano 5.6.1 granicnik.sh
#!/bin/bash
cat << hereDocDelimeter
    Ovo je prva linija teksta.
    Ovo je druga linija teksta.
    Ovo je treca linija teksta.
hereDocDelimeter
```



```
djordje@Djordje80: ~/bin
GNU nano 6.2 delimeter.sh
#!/bin/bash
cat << EOF
    x=4
    y=5
    echo "zbir je dat"
    $zbir(( $x+$y ))
EOF
```


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

```
djordje@Djordje80:~/bin$ delimiter.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 višelinijski komentar
  Ovo je višelinijski komentar
  Ovo je višelinijski 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 deklarirati.

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.

```
Terminal Sat 12:00
djordje80@ubuntu: ~/bin
File Edit View Search Terminal Help
GNU nano 2.9.3 systeminfo.sh

#!/bin/bash

# informacije o linux sistemu

declare -r NASLOV="Izvestaj o sistemu $HOSTNAME"
VREME=$(date +"%x %r %Z")
VREMENSKA_OZNAKA="Generisano $VREME, od $USER"

cat > izvestaj.html <<_EOF_

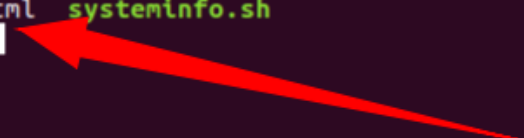
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>$NASLOV</title>
</head>
<body>
  <h1>$NASLOV</h1>
  <p>$VREMENSKA_OZNAKA</p>
</body>
<html>
_EOF_

[ Read 28 lines ]
^G Get Help      ^O Write Out     ^W Where Is     ^K Cut Text     ^J Justify
^X Exit          ^R Read File     ^\ Replace      ^U Uncut Text  ^T To Linter
```

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

```
Terminal Sat 12:00
djordje80@ubuntu: ~/bin
File Edit View Search Terminal Help
djordje80@ubuntu:~/bin$ nano systeminfo.sh
djordje80@ubuntu:~/bin$ systeminfo.sh
djordje80@ubuntu:~/bin$ ls
delimetar.sh  input.sh      output.sh     zdravo.sh
file.txt      izvestaj.html systeminfo.sh
djordje80@ubuntu:~/bin$
```



```

djordje80@ubuntu: ~/bin
File Edit View Search Terminal Help
djordje80@ubuntu:~/bin$ nano systeminfo.sh
djordje80@ubuntu:~/bin$ systeminfo.sh
djordje80@ubuntu:~/bin$ ls
delimetar.sh  input.sh          output.sh        zdravo.sh
file.txt      izvestaj.html    systeminfo.sh
djordje80@ubuntu:~/bin$ cat izvestaj.html
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>Izvestaj o sistemu ubuntu</title>
</head>
<body>
  <h1>Izvestaj o sistemu ubuntu</h1>
  <p>Generisano 03/02/2024 12:01:34 PM PST, od djordje80</p>
</body>
</html>
djordje80@ubuntu:~/bin$

```

Šta vidimo sve ovde. Imamo izvestaj o mom sistemu ubuntu, generisano 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

```
GNU nano 2.9.3                               systeminfo.sh

#!/bin/bash

#. informacije o linux sistemu

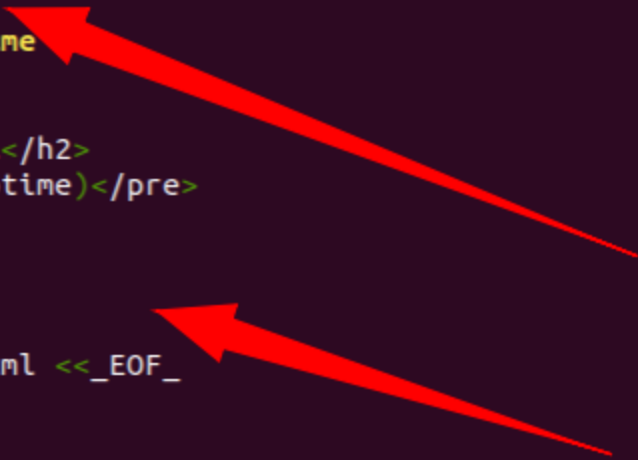
declare -r NASLOV="Izvestaj o sistemu $HOSTNAME"
VREME=$(date +"%x %r %Z")
VREMENSKA_OZNAKA="Generisano $VREME, od $USER"

function i_uptime {
local titl
titl="System Uptime

cat << _EOF_
  <h2>${titl}</h2>
  <pre>$(uptime)</pre>
_EOF_
  return
}

cat > izvestaj.html <<_EOF_

<!DOCTYPE html>
<head>
```



Funkcija se poziva u bash skripti na sledeci naćin.

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

```
File Edit View Search Terminal Help
GNU nano 2.9.3

#!/bin/bash

#. 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
}

cat > izvestaj.html <<_EOF_

<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>$NASLOV</title>
</head>
<body>
    <h1>$NASLOV</h1>
    <p>$VREMENSKA_OZNAKA</p>
    $(i_uptime)
</body>
</html>
_EOF_


```

za
pokretanje
funkcije

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

```
djordje80@ubuntu: ~/bin
File Edit View Search Terminal Help
djordje80@ubuntu:~/bin$ ls
delimetar.sh  input.sh      output.sh     zdravo.sh
file.txt      izvestaj.html systeminfo.sh
djordje80@ubuntu:~/bin$ systeminfo.sh
djordje80@ubuntu:~/bin$ ls
delimetar.sh  input.sh      output.sh     zdravo.sh
file.txt      izvestaj.html systeminfo.sh
djordje80@ubuntu:~/bin$ cat izvestaj.html
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>Izvestaj o sistemu ubuntu</title>
</head>
<body>
  <h1>Izvestaj o sistemu ubuntu</h1>
  <p>Generisano 03/02/2024 01:09:12 PM PST, od djordje80</p>
  <h2>System Uptime</h2>
  <pre> 13:09:12 up 6:50, 1 user, load average: 0.01, 0.02, 0.00</pre>
</body>
</html>
djordje80@ubuntu:~/bin$
```



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"

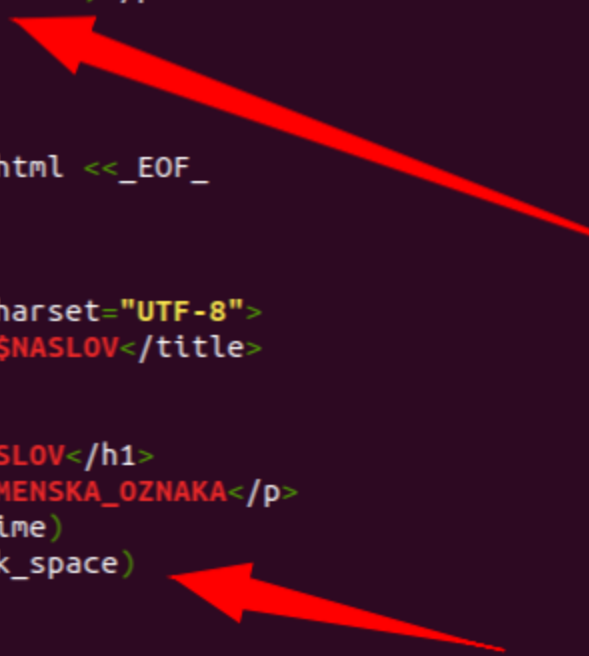
function i_uptime {
local titl
titl="System Uptime"

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>
    <meta charset="UTF-8">
    <title>$NASLOV</title>
</head>
<body>
    <h1>$NASLOV</h1>
    <p>$VREMENSKA_OZNAKA</p>
    $(i_uptime)
    $(i_disk_space)
</body>
<html>
_EOF_
```

Two red arrows originate from the right side of the terminal window. One arrow points to the line `$(df -h)` within the `i_disk_space` function call in the HTML output. The other arrow points to the line `$(i_uptime)` within the same function call.


```
File Edit View Search Terminal Help
File.txt   izvestaj.html   systeminfo.sh
djordje80@ubuntu:~/bin$ cat izvestaj.html

<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>Izvestaj o sistemu ubuntu</title>
</head>
<body>
  <h1>Izvestaj o sistemu ubuntu</h1>
  <p>Generisano 03/02/2024 01:21:01 PM PST, od djordje80</p>
  <h2>System Uptime</h2>
  <pre> 13:21:01 up 7:02, 1 user, load average: 0.15, 0.03, 0.01</pre>
  <h2>Slobodan prostor na disku</h2>
  <pre>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
```

pokretanjem možemo videti količinu slobodnog prostora na disku

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

File Edit View Search Terminal Help

GNU nano 2.9.3

```
        <pre>$(df -h)</pre>
_EOF_
    return
}
i_home_space() {
cat << _EOF_
    <h2>Potrosnja korisnicke memorije</h2>
    <pre>$(du -sh /home*)</pre>
_EOF_
    return
}
```

funkcija

```
cat > izvestaj.html << _EOF_

<!DOCTYPE html>
<head>
    <meta charset="UTF-8">
    <title>${NASLOV}</title>
</head>
<body>
    <h1>${NASLOV}</h1>
    <p>${VREMENSKA_OZNAKA}</p>
    ${i_uptime}
    ${i_disk_space}
    ${i_home_space}
</body>
</html>
_EOF_
```

za pokretanje

Sada je snimimo i pokrenemo .

```
File Edit View Search Terminal Help
tmpfs          391M   20K  391M   1% /run/user/1000</pre>
</body>
<html>
djordje80@ubuntu:~/bin$ nano systeminfo.sh
djordje80@ubuntu:~/bin$ clear

djordje80@ubuntu:~/bin$ systeminfo.sh
djordje80@ubuntu:~/bin$ ls
delimetar.sh file.txt input.sh izvestaj.html output.sh systeminfo.sh zdravo.sh
djordje80@ubuntu:~/bin$ cat izvestaj.html

<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <title>Izvestaj o sistemu ubuntu</title>
</head>
<body>
  <h1>Izvestaj o sistemu ubuntu</h1>
  <p>Generisano 03/02/2024 01:32:05 PM PST, od djordje80</p>
  <h2>System Uptime</h2>
  <pre>13:32:05 up 7:13, 1 user, load average: 0.04, 0.04, 0.00</pre>
  <h2>Slobodan prostor na disku</h2>
  <pre>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
```

snimimo i pokrenimo skriptu

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...

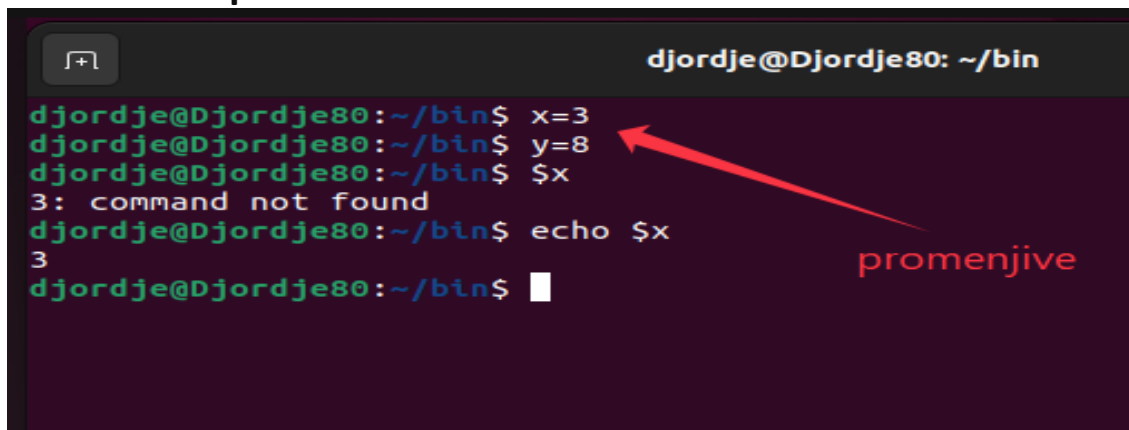
Iskazi odlučivanja se u zavisnosti od programskog jezika u kome radimo pišu različito.

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.

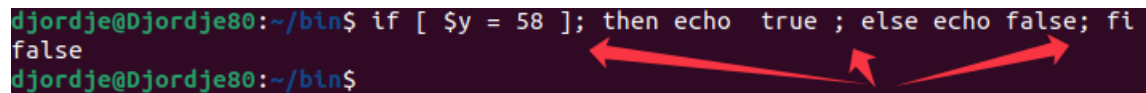


```
djordje@Djordje80: ~/bin
djordje@Djordje80:~/bin$ x=3
djordje@Djordje80:~/bin$ y=8
djordje@Djordje80:~/bin$ $x
3: command not found
djordje@Djordje80:~/bin$ echo $x
3
djordje@Djordje80:~/bin$
```

A red arrow points from the word "promenljive" to the variable expansion "\$x" in the terminal output.

```
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.

```
djordje@Djordje80: ~/bin
djordje@Djordje80:~/bin$ x=5
djordje@Djordje80:~/bin$ y=8
djordje@Djordje80:~/bin$ if [ $x = 5 ]; then echo false; elif [ $y = 8 ]; then
echo true; else echo fals; fi
false
djordje@Djordje80:~/bin$ if [ $x = 3 ]; then echo false; elif [ $y = 8 ]; then
echo true; else echo fals; fi
true
djordje@Djordje80:~/bin$ if [ $x = 5 ]; then echo false; elif [ $y = 6 ]; then
echo true; else echo fals; fi
false
djordje@Djordje80:~/bin$
```

Imamo skriptu najvećibroj.sh Videćemo 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

```
Terminal jyn 2 23:16 djordje@Djordje80: ~/bin
djordje@Djordje80:~/bin$ ls -l
total 16
-rwxrwxr-x 1 djordje djordje 44 jyn 2 20:39 delimeter.sh
-rwxrwxr-x 1 djordje djordje 86 jyn 2 22:28 ifprimer.sh
-rwxrwxr-x 1 djordje djordje 413 jyn 2 23:11 najveci broj.sh
-rwxrwxr-x 1 djordje djordje 56 anp 25 11:39 skripta.sh
djordje@Djordje80:~/bin$
```

```
Operatori poredenja:
-eq jednako =
-ne nije jednako <>
-lt manje <
-le manje ili jednako <=
-gt vece >
-ge vece ili jednako >=
```

Naravno sa komandom `chmod +x najveci broj.sh` napravio sam joj ovlasćenja da postane izvršna.

Nije više tekstualni fajl već postaje bash skripta.

```
+] | dJoraje@Djoraje80: ~/bin
GNU nano 6.2 najvecibroj.sh *
#!/bin/bash

echo -n " unesi prvi intiger: "
read int1
echo -n " unesi drugi intiger: "
read int2
echo -n " unesi treci intiger: "
read int3

if [[ $int1 -ge $int2 ]]
then
    if [[ $int1 -ge $int3 ]]
    then
        echo " najveci broj je $int1 "
    else
        echo " najveci broj je $int3 "
    fi
else
    if [[ $int2 -ge $int3 ]]
    then
        echo " najveci broj je $int2 "
    else
        echo " najveci broj je $int3 "
    fi
fi
```

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 $int3 "
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 jyl  2 20:39 delimeter.sh
-rwxrwxr-x 1 djordje djordje  86 jyl  2 22:28 ifprimer.sh
-rwxrwxr-x 1 djordje djordje 348 jyl  3 18:33 najmanjibroj.sh
-rw----- 1 djordje djordje 261 jyl  3 18:27 najmanjibroj.sh.save
-rwxrwxr-x 1 djordje djordje 413 jyl  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

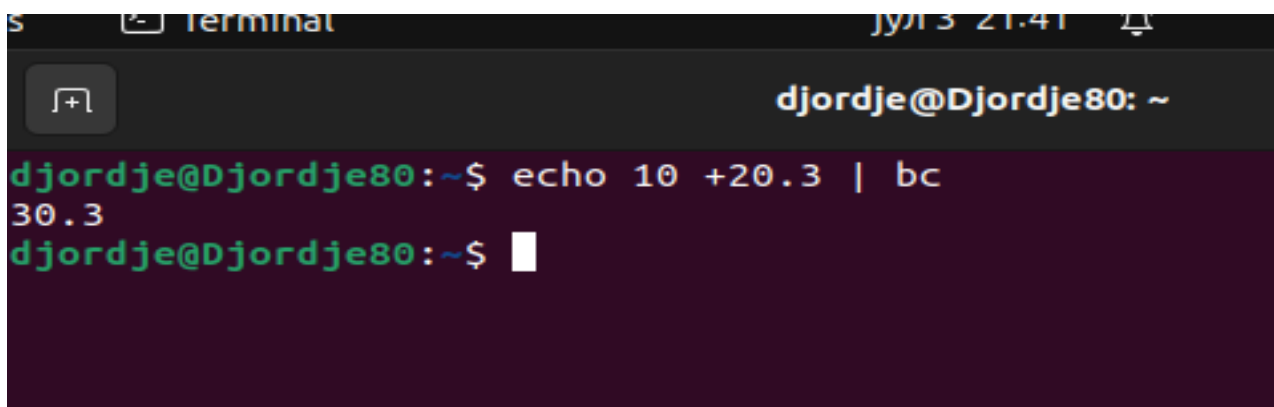
```
Logicki operatori:  
AND -a &&  
OR -o ||  
NOT ! !
```

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**



```
Terminal  
jy13 21:41  
djordje@Djordje80: ~  
djordje@Djordje80:~$ echo 10 +20.3 | bc  
30.3  
djordje@Djordje80:~$
```

10. Kako da pišete for petlje u bash skriptama ?

Napravimo jedan primer for petlje



```
GNU nano 6.2                               forloop.sh
#!/bin/bash

clear
echo " Brojevi od 1 do 5:"

for i in 1 2 3 4 5
do
    echo $i
done
echo "" # prazan red
```

On ce ispisivati

```

djordje@Djordje80: ~/bin
clear
Brojevi od 1 do 5:
1
2
3
4
5
djordje@Djordje80:~/bin$

```

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

```

GNU nano 6.2                               forloop.sh *
#!/bin/bash

clear
echo " Brojevi od 1 do 5:"

for i in 1 2 3 4 5
do
    echo $i
done
echo "" # prazan red

echo " Brojevi od 1 - 10: "

for i in {1..10}
do
    echo $i
done
echo ""

```

```

djordj
Brojevi od 1 do 5:
1
2
3
4
5

Brojevi od 1 - 10:
1
2
3
4
5
6
7
8
9
10

djordje@Djordje80:~/bin$
```

Da vidimo isti primer samo kako zapisati parne brojeve do 10

```

echo "Parni brojevi od 1 do 10:"
for i in {2..10..2}
do
echo $i
done
```

od 2 do 10 svaki drugi-tako dobijamo
parne brojeve prve desetice.

```
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.

```
for (( i=1; i<5; i++ ))  
do  
echo $i  
  if [ $i -eq 3 ]  
  then  
  break  
  fi  
done  
echo ""
```

obratite pažnju



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

Bilo bi ovako nešto

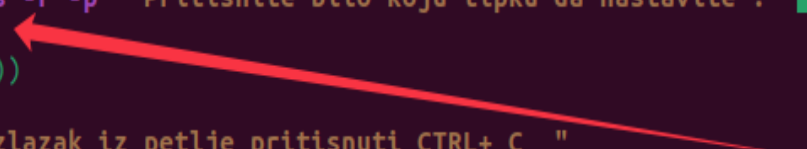
```
for (( i=1; i<5; i++ ))
do
echo $i
  if [ $i -eq 3 ]
  then
      echo " Mala promena na trecem mestu "
  continue
  fi
done
echo ""
```

Program bi ispisivao sledeću stvar

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

Da napravimo jednu beskonačnu petlju.

```
echo " Napravimo sada jednu beskonacnu petlju "
read -n 1 -s -r -p " Pritisnite bilo koju tipku da nastavite : "
for (( ; ; ))
do
echo " Za izlazak iz petlje pritisnuti CTRL+ C "
done
```



Show Applications

Write Out

Where To

Cut

Execute

Moramo obratiti paznju na sve ove atribute. Da je samo -n onda moramo pritisnuti 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 .

```
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C
Za izlazak iz petlje pritisnuti CTRL+ C ^C
djordje@Djordje80:~/bin$
```

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 jyl 5 17:25
djordje@Djordje80: ~/bin
GNU nano 6.2 meseciugodini.sh
#!/bin/bash

# meseci u godini
# napisimo jednu funkciju 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

```

```

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.

```
GNU nano 6.2          satminsek.sh
#!/bin/bash

# Pretvara sekunde u sate , minute i sekunde

sek=0
min=0
sat=0

read -p " Unesi broj sekundi : " sek

if [[ $sek =~ ^_?[[[:digit:]]+$ ]]; then      # da probamo digit a
    until [[ $sek -lt 60 ]]; do

        if [[ $sek -gt 59 ]]; then
            sek=`echo $sek -60 | bc`
            let min++
        fi

        if [[ $min -gt 59 ]]; then
            min=`echo $min -60 | bc`
            let sat++
        fi
    done
else
    echo " Unos podataka nije celi broj! "
    exit 1
fi

echo "$sat sat $min min $sek sek"
```

13. Pozicioni parametri i kako se koriste u BASH skriptama

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

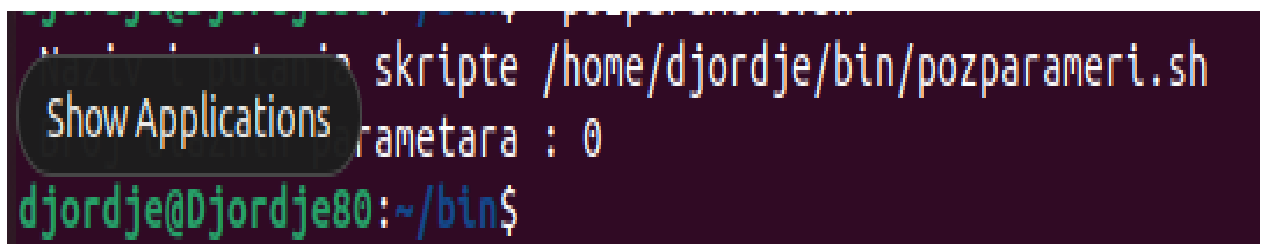
Onda naše skripte postaju puno bolje i korisnije.



```
GNU nano 6.2          pozparameri.sh *
#!/bin/bash

echo " Naziv i putanja skripte $0 "
echo " Broj ulaznih parametara : $# "
```


Da vidimo šta je on uradio



```
skripte /home/djordje/bin/pozparameri.sh
parametara : 0
djordje@Djordje80:~/bin$
```

Kao što vidimo nema ulaznih parametara pa je ta vrednost 0


```
djordje@Djordje80:~/bin$ nano pozparameri.sh
djordje@Djordje80:~/bin$ pozparameri.sh zdravo svima kako ste
```



4 paramet


Kao sto vidite uneti tekst je 4 parametara

```
djordje@Djordje80:~/bin$ nano pozparameri.sh
djordje@Djordje80:~/bin$ pozparameri.sh zdravo svima kako ste
Naziv i putanja skripte /home/djordje/bin/pozparameri.sh
Broj ulaznih parametara : 4
djordje@Djordje80:~/bin$
```



```
GNU nano 6.2 pozparameri.sh *
#!/bin/bash

echo " Naziv i putanja skripte $0 "
echo " Broj ulaznih parametara : $# "
echo "Ulazni parametri kao jedan: $* "
echo "Ulazni parametri pojednino : "
echo ""
for i in "$@"
do
    echo $i
done
echo ""
```



da se tretira kao jedan parametar

Ako hocemo da to sto napisemo da se razvrsta u vise redova pojednino

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.

```

GNU nano 6.2                                pozparameri.sh *
#!/bin/bash

echo " Naziv i putanja skripte $0 "
echo " Broj ulaznih parametara : $# "
echo "Ulazni parametri kao jedan: $*"
echo "Ulazni parametri pojedninacno : "
echo ""
    for i in "$@"
    do
        echo $i
    done
echo ""

echo " Drugi parametar je : $2 "
echo " Treci parametar je : $3 "

```

\$0 -daje naziv i putanju skripte

Drugi i treci parametar

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 završimo 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 USPEŠNO IZVRŠENA
# 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 pojedninačno :

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 elementa.

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

```
GNU nano 6.2                                nizovi.sh
#!/bin/bash

declare -A aniz    #Associative Array - asocijativni niz
declare -a nniz    # Numeric Array - numericki niz

echo " Asocijativni niz: "
echo  # ovo je isto kao da smo stavili prazne navodnike

aniz[bg]="Beograd"
aniz[sa]="Sarajevo"
aniz[zg]="Zagreb"
aniz[lj]="Ljubljana"
aniz[sk]="Skoplje"

read -p " Unesite oznaku bg, zg, sa,lj ili sk : " oznaka

case $oznaka in
  bg) echo "${aniz[bg]}";;
  zg) echo "${aniz[zg]}";;
  sa) echo "${aniz[sa]}";;
  lj) echo "${aniz[lj]}";;
  sk) echo "${aniz[sk]}";;
  *)
    echo "Uneli ste pogresnu oznaku! "
    exit 1
esac
echo
```


Program će uraditi

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

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

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

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

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

Unesite oznaku bg, zg, sa,lj ili sk : Novi Sad
Uneli ste pogresnu oznaku!
djordje@Djordje80:~/bin$
```

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
```