UNIVERSITETI “UKSHIN HOTI” - PRIZREN
FAKULTETI I SHKENCAVE KOMPJUTERIKE
TEKNOLOGJITË E INFORMACIONIT DHE TELEKOMUNIKIMI (TIT)

Ushtrimet laboratorike 7, 8 dhe 9

Lënda: Arkitektura e Kompjuterëve dhe Sistemet Operative

Ph. D. c. Arbër Beshiri

Objektivat e ushtrimeve laboratorike: të tregohen disa komanda dhe sintaksa e tyre për Linux (Ubuntu) dhe të zhvillohen e testohen detyra përmes Shell Script-ës së Linux (Ubuntu).

1. Përdorimi i shumë urdhërave (komandave) *if*

Në vazhdim është dhënë sintaksa kur në program kemi disa urdhëra *if* dhe një shembull për këtë rast:

```
if kushti
then
if kushti
then
        …
        …
        urdhëri
else
        …
        …
        udhëri
fi
else
        …
        …
        urdhëri
fi
```

```
$cat> prog9
#
# Përdorimi i komandës if…then…else përmes komandës read duke përdorur disa if
#
clear
echo “Jepni vlerat e variablave a, b, c: ”
read a b c
echo “Pra, vlerat e variablave janë: a=$a, b=$b, c=$c”
if [ $a -lt $b ]
then
echo "Numri a është më i vogël se numri b"
if [ $a –eq $b ]
then
echo “Numri a=$a dhe numri b=$b janë numra të barabartë”
else
if [ $a –le $c ]
echo “Numri a=$a nuk është më i vogël ose i barabartë me numrin c=$c”
fi
else
echo "Numri a është më i madh se numri b dhe c"
fi
```

- Shtypim **CTRL+D**, ruhet programi.
- Pastaj shtypim **chmod 777 prog9** për aktivizim të programit.
- Pastaj shtypim **./prog9** që të ekzekutohet programi.

Në dalje të Shell Script-ës fitohet rezultati:

```
Jepni vlerat e variablave a, b, c:
30 20 12
Pra, vlerat e variablave janë: a=30, b=20, c=12
Numri a është më i madh se numri b dhe c
```

2. Përdorimi shumënivelesh i *if...then...else* me shumë variabla

```
$ cat> prog10
#
# Përdorimi i if…elif...else me shumë variabla
#
```

```
clear
echo “Jepni vlerat e variblave: a, b, c, d, e, f”
read a b c d e f
echo “Vlerat e variablave janë: a=$a, b=$b, c=$c, d=$d, e=$e, f=$f”
if [ $a -gt $b -a $a -gt $c -a $a -gt $d ]; then
echo "Sipas formulës S1=c+d+f dhe P1=e*f fitohen këto rezultate: "
S1=`expr $c+$d+$f`
P1=`expr $e\*$f`
echo “Shuma është: S1=$S1, ndërsa prodhimi: P1=$P1”
elif [ $a -lt $b -a $c -lt $d -a $e -le $f ]
then
echo "Sipas formulës S2=a+b+c+e+f dhe P2=c*d fitohen këto rezultate: "
S2=`expr $a + $b + $c + $e + $f`
P2=`expr $c \* $d`
echo "Shuma është: S2=$S2 dhe prodhimi është: P2=$P2"
elif [ $a -lt $b -o $c -lt $d -o $e -le $f -o $c -ge $e ]
then
S3=`expr $e + $f`
echo "Sipas formulës së shumës S3=e+f fitohet ky rezultat: S3=$S3"
P3=`expr $S3 \* $d`
echo "Sipas formulës së prodhimit P3=S3*d fitohet ky rezultat: P3=$P3"
else
echo "Kujdes! A janë numra $a, $b, $c, $d, $e, $f? Ju lutemi jepni numrin!"
fi
```

- Shtypet **CTRL+D**, ruhet programi.
- Pastaj shtypim **chmod 777 prog10** për aktivizim të programit.
- Pastaj shtypim **./prog10** që të ekzekutohet programi.

3. Unazat (ciklet (loops))

Kompjuteri mund të përsëris disa herë instruksione të veçanta, derisa të përmbushet kushti i kërkuar. Grupi i instruksioneve që ekzekutohet në formë ciklike quhet unazë (cikël). Shell Script i Linux (Ubuntu) (bash) përkrah këto lloje ciklesh: urdhërin *for* dhe *while*.
Në fillim të çdo cikli:

a) duhet të inicializohet variabla e përdorur, pastaj fillon ekzekutimi i ciklit,
b) kushti vendoset në fillim të çdo iteracioni dhe
c) me deklarimin e vlerës së variablës së kushtit kompletohet trupi i ciklit (unazës).

3.1. Urdhëri *for*

Sintaksa e këtij urdhëri është:

for { emri i variablit } in { lista }
do
    ekzekutohen të gjitha procedurat në listë derisa lista të përfundoj (dhe përsëriten të gjitha deklarimet ndërmjet do dhe done)
done

3.1.1. Përdorimi i urdhërit *for...do...done* përmes urdhërit *for-i-in*

Në shembullin vijues është dhënë përdorimi e unazës *for...do...done* duke i dhënë vlerat paraprakisht sipas një rradhitje arbitrare përmes urdhërit *for-i-in*. Në këtë shembull vlerat jepen në fillim dhe prej këtyre vlerave realizohet përsëritja ciklike për grupin e caktuar të instruksioneve të cilat duhet të ekzekutohen.

```
$cat> prog11
#
# Përdorimi i urdhërave for...do...done
#
clear
for i in 1 2 3 4 5
do
echo "Mirë se vini $i herë"
done
```

- Shtypet **CTRL+D**, ruajm programin.
- Pastaj shtypim **chmod 777 prog11** për aktivizim të programit.
- Pastaj shtypim **./prog11** që të ekzekutohet programi.

Theksojmë që rezultati fitohet edhe nëse kombinojmë:

- **chmod –x prog11**, për aktivizim të programit.
- **./prog11** që të ekzekutohet programi.

Rezultati në dalje të terminalit do të jetë:

```
Mirë se vini 1 herë
Mirë se vini 2 herë
```

```
Mirë se vini 3 herë
Mirë se vini 4 herë
Mirë se vini 5 herë
```

3.1.2. Përdorimi i urdhërit *for...do...done* përmes urdhërit *if-i-in* për paraqitjen e tabelës së shumëzimit (shembull elaborues)

Në shembullin vijues do të paraqesim tabelën e shumëzimit për numra të ndryshëm të dhënë përmes komandës *read* dhe vargut përkatës të numrave, të cilët paraprakisht vendosen në program përmes urdhërit *for-i-in*.

```
$cat> prog12
#
# Përdorimi i urdhërave for...do...done
#
clear
echo "Jepni një numër të çfardoshëm të tipit integjer: "
read n
for i in 1 2 3 4 5 7 8 9 10
do
echo "$n * $i =`expr $i \* $n`"
done
```

- Shtypim **CTRL+D**, ruajm programin.
- Pastaj shtypim **chmod 777 prog12** për aktivizim të programit.
- Pastaj shtypim **./prog12** që të ekzekutojmë programin.

Rezultati në dalje do të jetë:

```
Jepni një numër të çfardoshëm të tipit integjer:
4
4 * 1 =4
4 * 2 =8
4 * 3 =12
4 * 4 =16
4 * 5 =20
4 * 7 =28
4 * 8 =32
4 * 9 =36
```

```
4 * 10 =40
```

3.1.3. Përdorimi i urdhërit (ciklit) *for...do...done* si rast iteracioni

Në shembullin në vazhdim do të paraqesim rastin e urdhërit *for* përmes iteracionit të caktuar.

```
$cat> prog13
#
# Përdorimi i urdhërave for...do...done si rast iteracioni
#
clear
echo "Jepni vlerën e n, duke treguar sa herë dëshirojmë të përsëritet cikli (loop)"
read n
for (( i = 0; i <= n; i++ ))
do
echo "Mirë se vini $i herë"
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog13** për aktivizim të programit.
- Pastaj, shtypim **./prog13** që të ekzekutohet programi.

Rezultati në dalje të Shell Script-ës do të jetë:

```
Jepni vlerën e n, duke treguar sa herë dëshirojmë të përsëritet cikli
5
Mirë se vini 0 herë
Mirë se vini 1 herë
Mirë se vini 2 herë
Mirë se vini 3 herë
Mirë se vini 4 herë
Mirë se vini 5 herë
```

3.1.4. Llogaritja e shumës përmes urdhërit *for...do...done*

```
$cat> prog14
#
# Llogaritja e shumës përmes urdhërit ciklik for...do...done
#
```

```
clear
echo "Jepni vlerën integjer për "n" si vlerë maksimale (nëse i=0; i<=n; i++): "
read n
echo "Ndërsa "i" ndryshon në hapin 1 dhe i<=$n"
echo "Në fillim S=0, ndërsa shuma llogaritet sipas formulës S=S+i: "
S=0
for (( i = 0 ; i <= n; i++ ))
do
S=`expr $S+$i`
done
echo "Shuma është: S=$S"
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog14** për aktivizim të programit.
- Pastaj, shtypim **./prog14** që të ekzekutohet programi.

Rezultati në dalje do të jetë:

```
Jepni vlerën integjer për n si vlerë maksimale (nëse i=0; i<=n; i++):
30
Në fillim S=0, ndërsa shuma llogaritet sipas formulës S=S+i:
Ndërsa “i" ndryshon në hapin 1 dhe i<=30
Shuma është: S=465
```

3.1.5. Llogaritja e prodhimit përmes urdhërit *for...do...done*

```
$cat> prog15
#
# Llogaritja e prodhimit përmes urdhërit unazor for...do...done
#
clear
echo "Jepni vlerën për "n" si vlerë maksimale, por jo më shumë se n=20 (nëse i=1; i<=n; i++): "
read n
echo "Në fillim P=1, ndërsa prodhimi llogaritet sipas formulës P=P*i dhe "i" përmes iteracionit
ndryshon deri në i=$n"
P=1
for (( i=1; i <= n; i++ ))
do
P=`expr $P \* $i`
echo "Për i=$i, prodhimi është: P=$P"
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog15** për aktivizim të programit.
- Pastaj shtypim **./prog15** që të ekzekutohet programi.

Rezultati në dalje do të jetë:

```
Në fillim P=1, ndërsa prodhimi llogaritet sipas formulës P=P*i dhe “i" përmes iteracionit
ndryshon deri në i=20
```

3.1.6. Llogaritja e prodhimit përmes urdhërit *for…do…done* duke përdorur kushtin *if* për plotësim të ndonjë veprimi

Në vijim është dhënë shembulli për llogaritjen e prodhimit përmes urdhërit *for…do…done* duke përdorur kushtin if për plotësim të ndonjë veprimi të caktuar.

```
$cat> prog16
#
# Llogaritja e prodhimit përmes urdhërit unazor: for...do...done
#
# Në Shell Script kemi rang të kufizuar të llogaritjeve dhe rezultateve…
clear
echo "Jepni vlerën për "n", por jo më shumë se n=20 (nëse i=1;i<=n; i++): "
read n
if test $n -le 20
then
echo "Në fillim P=1, ndërsa prodhimi llogaritet sipas formulës P=P*i dhe "i" është i=$n"
P=1
for (( i=1; i <= n; i++ ))
do
P=`expr $P \* $i`
echo "Për i=$i prodhimi është: P=$P"
done
else
echo "Për n=$n>20: Rezultati është jashta rangut. Ju lutemi jepni një vlerë për "n"=<20!"
fi
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog16** për aktivizim të programit.
- Pastaj shtypim **./prog16** që të ekzekutohet programi.

Rezultati në dalje do të jetë:

```
Jepni vlerën integjer për "n" si vlerë maksimale, por jo më shumë se n=20 (nëse i=1;i<=n; i++):
23
Për n=23>20: Rezultati është jashta rangut. Ju lutemi jepni një vlerë për "n"=<20!
```

3.1.7. Vendosja e ciklit të jashtëm dhe të brendshëm

Shembulli në vijim tregon vendosjen e unazës (ciklit) së jashtme dhe të brendshme.

```
$cat> prog17
#
#Vendosja e unazës (ciklit) së jashtme dhe të brendshme
#
clear
for (( i=1; i <= 9; i++ )) ### cikli i jashtëm ###
do
for (( j=1; j <= 9; j++ )) ### cikli i brendshëm ###
do
echo -n "$i"
done
echo " " #### vendos rresht të ri ###
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog17** për aktivizim të programit.
- Pastaj, shtypim **./prog17** që të ekzekutohet programi.

Rezultati në dalje do të jetë:

```
1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8
```

```
9 9 9 9 9 9 9 9 9
```

3.1.8. Përdorimi i urdhërit *for-in* për gjetjen e fajllave ekzekutues nga direktoriumi aktual

Në këtë shembull paraqitet programi për gjetjen e fajllave ekzekutues në kuadër të direktoriumit aktual.

```
$cat> prog18
#
#Përdorimi i urdhërit for-in për gjetjen e fajllave ekzekutues në kuadër të direktorimit aktual
#
clear
for i in *
do
if [ -f "$i" -a -x "$i" ]
then
echo "Fajlli ekzekutues $i "
fi
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog18** për aktivizim të programit.
- Pastaj, shtypim **./prog18** që të ekzekutohet programi.

Në rezultat paraqiten të gjithë fajllat e ekzekutuar në direktoriumin përkatës:

```
Fajlli ekzekutues x1
Fajlli ekzekutues x2
Fajlli ekzekutues x3
Fajlli ekzekutues x4
Fajlli ekzekutues x5
…
```

4. Skripta për testim të urdhërit *while* dhe komandë-rreshtit

```
$cat> prog19
#
#Skripta për testim të urdhërit while
```

```
#
# $# - është komandë për numrin e argumenteve
# Numri i argumenteve mund të ndryshoj
# $0 - kjo komandë jep emrin e programit aktual
clear
if [ $# -eq 0 ]
then
echo "Gabim, sepse në ekzekutim të programit mungon numri që duhet të shkruhet si argument në komandë-rreshtin. Shiko sintaksën!"
echo "Sintaksa është: $0 numri"
echo "Tek fjala "numri" shënojmë numrin me të cilin dëshirojmë të shumëzojmë ciklin prej i=1 deri i=10"
echo "Ky program shfrytëzohet për të shtypur tabelën e shumëzimit për numrin e dhënë"
exit 1
fi
n=$1
i=1
while [ $i -le 10 ]
do
echo "$n * $i = `expr $i \* $n`"
i=`expr $i + 1`
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog19** për aktivizim të programit.
- Nëse programi ekzekutohet përmes **./prog19**, në dalje duhe të kemi rezultatin: "Gabim, sepse në ekzekutim të programit mungon numri që duhet të shkruhet si argument në komandë-rreshtin. Shiko sintaksen!"
- Sintaksa është : **./prog19 numri**
- Tek fjala “numri” shënojmë numrin me të cilin dëshirojmë të shumëzojmë ciklin prej i=1 deri i=10.
- Prandaj, gjatë ekzekutimit mund të shënojmë: **./prog19 7**

```
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
```

```
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```

4.1.1. Skripta për aktivizimin dhe ekzekutimin e programit përmes komandë-rreshtit

Në shembullin në vazhdim është dhënë skripta për disa llogaritje, duke përdorur komandë-rreshtin. Pra, kur përdorim komandë-rreshtin të dhënat duhet të jepen në formë të argumenteve për realizimin e programit.

```
$cat> prog20
#
# Skripta për paraqitjen e ekzekutimit në Shell Script, përmes përdorimit të komandë-rreshtit
#
tot=`expr $1 + $2`
echo $tot
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog20** për aktivizim të programit.
- Për ekzekutimin e programit përdorim sintaksën: **./prog20 argumenti1 argumenti2.**
- Tek argumenti1 dhe argumenti2 shënojmë numrat e argumenteve përkatëse, ku në mënyrë të drejtëpërdrejtë do të mblidhen sipas formulës tot=`expr $1+$2`, që gjendet në skript.

Prandaj, si shembull jepet: **./prog20 3 4**
Në dalje fitohet rezultati:

```
7
```

Nëse gjatë ekzekutimit përdorim shprehjen si në vijim: **sh -x prog20 4 5**, në dalje fitojmë rezultat të këtillë:

```
+ expr 4 + 5
+ tot=9
+ echo 9
9
```

4.1.2. Realizimi i disa operacioneve duke përdorur komandë-rreshtin.

Në shembullin në vijim jepet skripta për llogaritje të disa operacioneve të caktuara.

```
$cat> prog21
#
# Skripta për realizimin e disa operacioneve aritmetike, duke përdorur komandë-rreshtin
#
echo "Të llogaritet shuma S1: "
S1=`expr $1 + $2 + $3`
echo "Shuma e fituar sipas formulës S1=\$1+\$2+\$3 është: S1=$S1"
echo "Të llogaritet prodhimi P1: "
P1=`expr $2 \* $3`
echo "Prodhimi sipas formulës P1=\$1\*\$3 është: P1=$P1"
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog21** për aktivizim të programit.
- Për ekzekutimin e programit përdorim sintaksën: **./prog21 argumenti1 argumenti2 argumenti3.**
- Tek argumenti1, argumenti2 dhe argumenti3 shënojmë numrat e argumenteve përkatëse, të cilat në mënyrë të drejtëpërdrejtë do të inkorporohen në programin e shënuar.
- Nëse japim vlerat e argumenteve: **./prog21 3 4 5**, në dalje fitojmë këtë rezultat:

```
Të llogaritet shuma S1:
Shuma e fituar sipas formulës S1=$1+$2+$3 është: S1=12
Të llogaritet prodhimi P1:
Prodhimi sipas formulës P1=$1 \* $3 është: P1=15
```

5. Urdhëri *case*

Urdhëri *case* është alternativë për deklarimin shumënivelësh të *if…then…else*. Ky urdhër mundëson përshtatjen e disa vlerave ndaj një variable.

Sintaksa:

```
case $emri i variablës in
        modeli 1) komanda
                …
                …
```

```
                komanda;;
        modeli 2) komanda
                …
                komanda;;
        modeli N) komanda
                …
                …
                komanda;;
            *)  komanda
                …
                …
                komanda;;
esac
```

$emri i variablës krahasohet me modelet derisa të realizohet veprimi. Pastaj, në skript ekzekutohen të gjitha komandat. Shënja standarde *) ekzekutohet nëse nuk është realizuar veprimi.

5.1. Shembull me urdhërin case

```
$cat> prog22
#
# Skripta për realizimin e urdhërit case
#
if [ -z $1 ]
then
rental="*** Makina e panjohur ***"
elif [ -n $1 ]
then
# Përndryshe realizoje argumentin e parë si rental
rental=$1
fi
case $rental in
"Makinax") echo "Për $rental 0.5 euro për km";;
"Makinay") echo "Për $rental 0.9 euro për km";;
"Makinaz") echo "Për $rental 1 euro për km";;
"Bicikleta") echo "Për $rental 20 cent për km";;
*) echo "Kërkojmë falje, por ne nuk kemi makinë me emër të tillë: $rental-";;
esac
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog22** për aktivizim të programit.

- Pastaj, shtypim **./prog22** që të ekzekutohet programi.

Rezultati:

```
Kërkojmë falje, por ne nuk kemi makinë me emër të tillë: *** Makina e panjohur ***
```

Nëse japim komandën: ./prog22 Makinax
Fitohet ky rezultat në dalje:

```
Për Makinax 0.5 euro për km
```

6. Urdhëri *select*

Sintaksa e urdhërit *select* është:

```
select emri_variabla in të dhënat
do
break
done
```

6.1. Përdorimi i urdhërit *select*

Në vazhdim është dhënë shembulli që mbështetet në të dhënat e përdoruesit për të vendosur se çfarë duhet të bëhet mëtej. Përmes këtij shembulli mundësohet krijimi i një mënye me përzgjedhje.

```
$cat> prog23
PS3="Zgjidh (1-5):"
echo "Zgjidh ngjyrën nga lista e ngjyrave: "
select Emri in E_kuqe E_gjelbërt E_kaltërt E_verdhë E_bardhë
do
break
done
echo "Ju keni zgjedhur ngjyrën $Emri"
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog23** për aktivizim të programit.
- Pastaj, shtypim **./prog23** që të ekzekutohet programi.

Rezultati:

```
Zgjidh ngjyrën nga lista e ngjyrave:
1)E_kuqe
2)E_gjelbërt
3)E_kaltërt
4)E_verdhë
5)E_bardhë
Zgjidh (1-5):3
Ju keni zgjedhur ngjyrën E_kaltërt
```

6.2. Krijimi dhe përdorimi i vargjeve në shell script (shembull elaborues)

```
$cat> prog24
#
#Krijimi dhe përdorimi i vargut
#
clear
vargu=(E_kuqe E_gjelbërt E_kaltërt E_verdhë E_bardhë)
ngjyrat=${#vargu[*]}
echo "Vargu ka $ngjyrat ngjyra. Ato janë: "
i=0
while [ $i -lt $ngjyrat ]; do
echo "$i: ${vargu[$i]}"
let i++
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog24** për aktivizim të programit.
- Pastaj, shtypim **./prog24** që të ekzekutohet programi.

Rezultati:

```
Vargu ka 5 ngjyra. Ato janë:
0: E_kuqe
1: E_gjelbërt
2: E_kaltërt
3: E_verdhë
4: E_bardhë
```

6.3. Zëvendësimi i një stringu me një tjetër - krijimi i vargut (shembull elaborues)

```
$cat> prog25
#
#Zëvëndësimi i një stringu me një tjetër - krijimi i vargut
#
clear
listë="artikulli1 artikulli2 artikulli3 artikulli4"
artikujt="Ky artikull është x\n\
Ndërsa x është një nga artikujt në listë \n"
for kriteri in $list; do
echo -e ${artikujt//x/$kriteri}
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog25** për aktivizim të programit.
- Pastaj, shtypim **./prog25** që të ekzekutohet programi.

Rezultati:

```
Ky artikull është artikulli1
Ndërsa artikulli1 është një nga artikujt në listë
Ky artikull është artikulli2
Ndërsa artikulli2 është një nga artikujt në listë
Ky artikull është artikulli3
Ndërsa artikulli3 është një nga artikujt në listë
Ky artikull është artikulli4
Ndërsa artikulli4 është një nga artikujt në listë
```

6.4. Përdorimi i urdhërit *select* dhe *case*

Përdorimi i kombinuar i urdhërit *select* dhe *case* për zgjedhjen e disa opsioneve është paraqitur si në shembullin vijues:

```
$cat> prog26
#
#
```

```
clear
opsionet=(
"Paraqiteni listën e direktoriumeve aktuale"
"Paraqiteni se kush është kyçur në sistem"
"Paraqiteni datën dhe kohën aktuale"
"Dilni nga sistemi"
)
select option in "${opsionet[@]}"; do
case "$REPLY" in
1) ls -l;;
2) who;;
3) date;;
4) break;;
esac
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog26** për aktivizim të programit.
- Pastaj, shtypim **./prog26** që të ekzekutohet programi.

Rezultati:

```
-rwxrwxrwx 1 user user 388 2018-04-24 11:11 x1
-rwxrwxrwx 1 user user 204 2018-04-24 11:19 x2
-rwxrwxrwx 1 user user 288 2011-04-24 11:33 x3
-rwxrwxrwx 1 user user 288 2011-04-24 11:33 x3_1
-rwxrwxrwx 1 user user 584 2011-04-24 11:49 x5
-rwxr-xr-x 1 user user 959 2018-05-04 05:53 yesnobox.sh
#? 2
user tty7 2018-05-11 11:03 (:0)
user pts/0 2018-05-11 05:08 (:0.0)
#? 3
Wed May 11 07:27:06 EDT 2018
#? 4
```

6.5. Paraqitja e një programi për përcaktimin e disa opsioneve të caktuara

Në shembullin në vazhdim është paraqitur një program që mundëson përzgjedhjen e opsioneve të radhitura.

```
$cat> prog27
#
# Në shembullin në vazhdim është paraqitur një program që mundëson përzgjedhjen e opsioneve
# të radhitura.
#
clear
echo -n "Ju lutem shënoni cilin opsion dëshironi të paraqiteni!"
read Emri
echo "Mëny për $Emri
1. Paraqiteni listën e direktoriumeve aktuale
2. Paraqiteni se kush është kyçur në sistem
3. Paraqiteni datën dhe kohën aktuale
4. Dilni nga sistemi"
read hyrja
hyrja1="1"
hyrja2="2"
hyrja3=$(date)
hyrja4=$(exit)
while [ "$hyrja" = "3" ]
do
echo "Data dhe koha aktuale: $hyrja3"
done
while [ "$hyrja" = "4" ]
do
echo "Mirupafshim $hyrja4"
done
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog27** për aktivizim të programit.
- Pastaj, shtypim **./prog27** që të ekzekutohet programi.

Rezultati:

```
Ju lutem shënoni cilin opsion dëshironi të paraqiteni!
```

7. Funksionet

Në krahasim me gjuhët “reale” programuese, programimi në *bash shell linux* ka implementim të kufizuar. Funksioni është nënprogram (subroutine) ose bllok kodi i cili implementon bashkësinë e operacioneve.

Sintaksa e funksioneve është dhënë në vazhdim:

```
Emri_funksionit ()
{
        komanda ...
}
```

Ose sintaksa e funksionit mund të jetë edhe kështu:

```
emri_funksionit (){
        komanda ...
}
```

7.1. Përdorimi i funksioneve

Në vazhdim është dhënë një shembull (elaborues) për përdorimin e funksioneve në Shell Script-en e Linux.

```
$cat> prog28
#
# Përdorimi i funksionit fnTungjatjetaTëGjithëve
#
clear
fnTungjatjetaTëGjithëve()
{
# Shtyp Tungjatjeta të gjithëve
echo "Tungjatjeta të gjithëve!"
}
# Thirrja e funksionit Tungjatjeta të gjithëve
fnTungjatjetaTëGjithëve
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog28** për aktivizim të programit.
- Pastaj, shtypim **./prog28** që të ekzekutohet programi.

Rezultati:

```
Tungjatjeta të gjithëve
```

7.2. Përdorimi i funksionit për shtypjen (printimin) e një mesazhi

```
$cat> prog29
#
# Përdorimi i funksionit për shtypjen e një mesazhi
#
clear
PrintoMesazhin() {
echo "Tungjatjeta, ky është funksioni PrintoMesazhin"
}
PrintoMesazhin
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog29** për aktivizim të programit.
- Pastaj, shtypim **./prog29** që të ekzekutohet programi.

Rezultati:

```
Tungjatjeta, ky është funksioni PrintoMesazhin
```

8. Kalimi i argumentit në funksion

Kalimi i argumentit në funksion realizohet lehtësishtë duke përdorur variablat $1, $2, …, $n që paraqesin argumentet në funksion. Në vazhdim është dhënë shembulli për kalimin e dy argumenteve në funksion dhe shtypjen (printimin) e tyre.

8.1. Le të japim funksionin me emrin fnArgumentiFunksionit, ashtu që në të, të kalojnë dy argumentet: "Tungjatjeta" "Të gjithëve".

```
$cat> prog30
#
# Përdorimi i funksionit dhe kalimi i argumenteve në funksion
# Emri i funksionit është fnArgumentiFunksionit
fnArgumentiFunksionit()
{
echo "Argumenti 1: " . $1
echo "Argumenti 2: " . $2
}
fnArgumentiFunksionit "Tungjatjeta" "Të gjithëve"
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog30** për aktivizim të programit.
- Pastaj, shtypim **./prog30** që të ekzekutohet programi.

Rezultati:

```
Argumenti 1: . Tungjatjeta
Argumenti 2: . Të gjithëve
```

8.2. Llogaritja e shumës përmes funksionit

Në vazhdim do të llogaritisim shumën e numrave përmes funksionit.

```
$cat> prog31
#
# Llogaritja e shumës përmes funksionit
# Funksioni si bllok kod mundëson realizimin e llogaritjes
#
clear
LlogaritjaShumës ()
{
shuma=$[$1 + $2]
}
echo "Jepni numrin e parë: "
read num1
echo "Jepni numrin e dytë: "
read num2
LlogaritjaShumës $num1 $num2
echo "Shuma është: $shuma"
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog31** për aktivizim të programit.
- Pastaj, shtypim **./prog31** që të ekzekutohet programi.

Vlerat merren si vlera arbitrare dhe nxjerrim rezultatin:

```
Jepni numrin e parë:
45
Jepni numrin e dytë:
58
```

```
Shuma është: 103
```

8.3. Përdorimi i funksionit për të shtuar përdorues dhe fjalëkalim

Në shembullin në vazhdim është paraqitur skripta përmes të cilës mund të shtojmë përdorues dhe fjalëkalim duke përdorur funksionin.

```
$cat> prog32
#
#
clear
Shtimi_përdoruesit()
{
PËRDORUESI=$1
FJALËKALIMI=$2
shift; shift;
# Bëjm zhvendosje dy herë, pjesa tjetër është komenti
KOMENTI=$@
echo “Përdoruesi i shtuar është $PËRDORUESI $KOMENTI”
echo “Fjalëkalimi i përdoruesit $PËRDORUESI është $FJALËKALIMI”
echo "Përdoruesi i shtuar $PËRDORUESI ($KOMENTI) ka fjalëkalimin $FJALËKALIMI"
}
###
# Trupi i skriptës fillon këtu
###
echo "Fillo me skriptën..."
Shtimi_përdoruesit x1 PËRDORUESI1
Shtimi_përdoruesit x2 PËRDORUESI2
Shtimi_përdoruesit x3 PËRDORUESI3
echo "Fundi i skriptës”
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog32** për aktivizim të programit.
- Pastaj, shtypim **./prog32** që të ekzekutohet programi.

Në dalje do të paraqitet ky rezultat:

```
Fillo me skriptën...
Përdoruesi i shtuar është x1
Fjalëkalimi i përdoruesit x1 është PËRDORUESI1
Përdoruesi i shtuar x1 () ka fjalëkalimin PËRDORUESI1
```

```
Përdoruesi i shtuar është x2
Fjalëkalimi i përdoruesit x2 është PËRDORUESI2
Përdoruesi i shtuar x2 () ka fjalëkalimin PËRDORUESI2
Përdoruesi i shtuar është x3
Fjalëkalimi i përdoruesit x3 është PËRDORUESI3
Përdoruesi i shtuar x3 () ka fjalëkalimin PËRDORUESI3
Fundi i skriptës
```

8.4. Përdorimi i funksionit ku si variabla ndihmëse janë variablat e shfrytëzuesit

Në këtë shembull do të paraqesim funksionin i cili përdor variabla të shfrytëzuesit të përcaktuara me vlera si stringje.

```
$cat> prog33
#
#
clear
Funksioni1()
{
echo "\$1 është $1"
echo "\$2 është $2"
# 1="Puna"
# Është sintaksë valide. Prandaj këtë mund të ndryshojmë
# Ndrysho $a:
a="Puna"
}
### Skripta kryesore fillon nga këtu
a=Puna
b=Veprimi
Funksioni1 $a $b
echo "a është $a"
echo "b është $b"
```

- Shtypim **CTRL+D**, ruhet programi.
- Shtypim **chmod 777 prog33** për aktivizim të programit.
- Pastaj, shtypim **./prog33** që të ekzekutohet programi.

Në dalje do të paraqitet ky rezultat:

```
$1 është Puna
```

```
$2 është Veprimi
a është Puna
b është Veprimi
```

8.5. Gjetja e fajllave përmes funksionit, sipas një shkronje të caktuar

Në këtë shembull është dhënë programi i cili mundëson gjetjen e disa fajllave përmes një shkronje të caktuar.

```
$cat> prog34
#
# Për dallim nga detyra paraprake rezultatet do të paraqiten në kolona
# Në këtë bllok kodi do të përdorim komandën “return”
#
a() {
R=0
ls -1 a*
[ "$?" == "1" ] && { R=1; }
return $R
}
a
r=$?
echo $r
```

Shfaqeni rezultatin…