C`eshte Kerneli?

Kerneli eshte zemra e sistemit Operativ, i cili sherben per te menaxhuar burimet e tij si ruajtjen e te dhenave, printimin e dokumentave, menaxhimin e memorjes etj.
Detyrat e tij jane:

1. Menaxhimi i pajisjeve I/O
2. Menaxhimi i proceseve
3. Menaxhimi i pajisjeve
4. Menaxhimi i filave
5. Menaxhimi i memorjes

C`eshte SHELL?

Kompjuteri kupton vetem gjuhen makine domethene 1 dhe 0, te quajtur gjuhe binare. Ne kompjuterat e meparshem, instruksionet jepeshin ne formatin binare, e cila eshte shume e veshtire per perdoruesin ta lexoje dhe ta shkruaje. Ditet e sotme eshte nje program special i quajtur SHELL. Shell-i pranon instruksione apo komanda ne gjuhen angleze(njeri) dhe e perkthen ne nje gjuhe te kuptueshme per makinen(dmth ne format binare).

Si punon Shell-i

Komanda e shtypur nga njeriu(ose skripti) -> Linux SHELL -> Konvertohet ne gjuhe binare nga SHELL-> Linux Kernel e kupton kerkesen

Nese si Shell do te sherbeje BASH atehere do te kishim:

```
$ls                    00101000
$date   -> BASH  ->    10101011  -> Linux Kernel
$time                  01010010
```

Shell-i eshte nje gjuhe interpretuese e komandave. Ai ekzekuton komandat e lexuara nga pajisjet standarte hyrese (psh tastiera) ose nga nje file.Linux-i mund te perdore keto shell-e: (Ne MS-DOS, emri i Shell-it eshte COMMAND.COM, i cili perdoret per te njejtat qellime, por nuk eshte aq i fuqishem sa ai ne Linux).

Emri i Shell-it	Zhvilluar nga...	Ku	Shenime
BASH(Bourne-Again Shell)	Brian Fox dhe Chet Ramey	Fondacioni Free Software	Shumica e Shell-eve ne LINUX
CSH(C Shell)	Bill Joy	Universiteti i Kalifornise(Per BSD)	Sintaksa e CSH eshte shume e ngjashme me gjuhen C
KSH(Korn Shell)	David Korn	AT&T Bell Labs	

Te gjithe Shell-et e mesiperme lexojne komanda nga perdoruesi (psh nepermjet tastieres apo mouse-it) dhe i tregojne sistemit operativ Linux c`fare deshiron perdoruesi. Nese shkruajme komanda nga tastiera, nderfaqja quhet Command Line dhe fillon me shenjen $ per ambjentin perdorues dhe me shenjen # per ambjentin administrator.
Per te gjetur llojin e shellit atehere, shtypni komanden:
$ echo $SHELL.
Per te hyre ne shell pergjithsisht shkohet te Tab-i:
“Application->Accessories->Terminal”

Komandat me te perdorshme ne Linux.
Komandat qe do te permendim jane per fillestaret ne Linux.Qellimi eshte qe me ane te ketyre komandave, te jeni sa me familjar me Shell-in dhe e dyta disa prej ketyre komandave te jene pjese e shkripteve. Nese duam informacion me te detajuar ne lidhje me nje komande atehere shtypni emrin e komandes se ndjekur nga opsioni help,komanda man ose info,psh:
$date --help
$man date
ose
$info date
Komanda e dyte dhe e trete do te na coje ne nje editor ndihmes qe tregon informacion ne lidhje me komanden e shtypur. Per te dale nga ky editor shtypni q (Per me teper do te shikojme te editoret ne Linux).
Ne menyre qe pamja te ndahet ne copa sipas madhesise se ekranit atehere me ane te nje pipe | dhe nje komande more do te mernim nje pamje me te

kendshme, psh:
$ls --help | more
Per te levizur nga nje faqe ne tjetren perdorim cdo taste(psh space).
Ne MS-DOS ndihma do te ishte:
c:\> dir /? ose
c:\> help dir

Disa komnada ne LINUX.

Qellimi	Sintaksa	Shembull
Per te shikuar daten	date	$date
Per te shikuar cili po perdor sistemin(meqe Linux eshte Multi-user).	who	$who
Per te shfaqur direktorine e punes	pwd	$pwd
Per te listuar emrat e filave ne direktorine korente	ls ose dirs	$ls
Per te krijuar nje file tekst. KUJDES: Per te dale duhet te shtypni tastat CTRL dhe D (***CTRL+D***)	cat > {emer file}	$cat > myfile
Per te shikuar permbajtjen e files	cat {emer file}	$cat myfile
Per te shfaqur te gjithe filen ne ekran, per nje cast	more {file name}	$more myfile
Per te zhvendosur(ose riemerton) nje file te nje direktorie ne nje vend tjeter	mv {file1} {file2}	$mv sales sales99
Per te krijuar kopjimin e shume filave me linqe te ndryshme. Pas kesaj komande si “fila e vjeter” ashtu dhe “fila e re” do ti referohen te njejtit emer	ln {oldfile} {newfile}	$ln page1 book1
Per te fshire nje file	rm file1	$rm myfile
Per te fshire te gjitha filat ne nje direktori,(perdoreni me kujdes)	rm -rf (emer direktorie)	$rm -rf oldfiles

Per te fshire nje direktori bosh	rmdir {emer direktorie}	$rmdir mydir
Per tu zhvendosur nga nje direktori ne tjetren	cd {emer direktorie}	$cd mydir
Per te shkuar nje direktori me siper ose ne rrenje	cd .. ose cd /	$cd ..
Per te pare te drejtat e filave te nje direktorie	ls -l {emer direktorie}	$ls -l dev
Per te ndryshuar te drejtat e aksesimit te filave u - useri i cili ka pronesi mbi file. g – Grupi i cili ka pronesi mbi file. o- Perdoruesit e klasifikuar si other a- Te gjithe perdoruesit sistem + Shton te drejta - Heq te drejta r- Lexon te drejtat w- Shkruan te drejtat x- Ekzekuton te drejtat	chmod {u|g|o|a} {+|-} {r|w|x} {emer file}	$chmod u+x,g+wx,o-r myscript
Per te lexuar emailin	mail	$mail
Per te shikuar me shume rreth personit korrent te loguar	who am i	$who am i
Per te dale nga logimi	logout ose ***CTRL+D***	$logout (Kujdes: Ka raste qe te kerkohet passwordi, kurse ne disa raste eshte i caktivizuar nga Administratori i Sistemit)
Per ti derguar email personit tjeter	mail {user-name}	$mail igli
Per te numeruar rreshtat, fjalet dhe	wc -l,	$wc -c myfile

karakteret ne nje file.	wc -w, wc -c	
Per te gjetur nje fjale ne filen tone	grep {fjalen qe do te kapim} {filename}	$grep fox myfile
Per te gjetur emrin e nje file apo te nje direktorie	find {emer file}	$find myfile
Per te edituar nje file.	gedit {emer file}	$gedit myfile
Per te edituar serish nje file	pico {emer file}	$pico myfile
Per te krijuar nje direktori	mkdir {emer direktorie}	$mkdir mydir
Per ti rradhitur filat sipas: -r Ne rendin e kundert -n Sipas rendit numerik -nr Ne rendin e kundert numerik	sort -r -n -nr {filename}	$sort myfile
Per te shfaqur rreshtin e fillimit ose te fundit te nje file.	tail -|+ {numri i rreshtit} {emri i files}	$tail +5 myfile
Per te krahasuar filat	cmp {file1} {file2} ose diff {file1} {file2}	$cmp myfile myfile.old
Per te printuar nje file	pr {emer file}	$pr emer file.
Per te hyre ne editorin vi	vi	$vi Kujdes: Per te

		dale nga editori mund te dilni me :q!, nese nuk doni ta ruani ate qe shkruajtet ose :wq nese deshironi ta ruani.

C`eshte nje proces ne Linux?

Nje proces eshte nje program (komande e dhene nga perdoruesi) per te kryer disa pune te caktuara. Ne linux kur fillon nje proces, ai mer nje numer te quajtur Process ID (PID). PID fillon nga 0 deri ne 65535.

Duke ditur qe linux eshte nje sistem multiperdorues dhe multitasking,kjo do te thote qe mund te ekzekutohen dy procese ne te njejten kohe,nese deshirojme, psh per te gjetur sa fila keni ju ne sistemin tuaj mund te perdorim komanden:

$ls -R | wc -l

Kjo komande kerkon shume kohe per te kerkuar te gjitha filat ne sistemin tuaj. Atehere mund te ekzekutojme kete komande ne Background, ose duke dhene komanden:

$ls -R | wc -l &

Simboli & ne fund te komandes i tregon shell-it te nis ekzekutimin e komandes $ls -R |wc -l dhe ta ekzekutojme ate ne background, duke filluar menjehere me komanden tjeter.

Komandat e Linux-it qe kane lidhje me procesin:

Qellimi	Komanda	Shembull
Per te pare procesin korent	ps	$ps
Per te ndaluar procesin	kill {PID}	$kill 1012
Per te mare informacion per te gjithe proceset ne ekzekutim	ps -ag	$ps -ag
Per te ndaluar te gjitha proceset me perjashtim te shell-it	kill 0	$kill 0
Per proceset background	linux – command &	$ls -R	wc -l &

(Shenja & perdoret per komandat dhe programet qe ekzekutohen ne background)		

Shenim:Ju mund te vrisni ato procese qe i keni krijuar vete. Nje administrator mund te vrase nga ne 95%-98% te proceseve. Disa procese nuk mund te vriten si psh VDU.

Redireksionet Input/Output

Pothuajse te gjitha komandat japin output ne ekran ose marrin input nga tastiera. Ne linux eshte e pamundur te dergohet output dhe te lexohet input te nje file pa ndermjetesimin e disa parametrave speciale. Psh $ls shfaq permbajtjen e nje direktorie ne ekran. Per te derguar te dhenat e mara nga kjo komande ne nje file atehere: $ls > filename. Dallojme tre lloje simbolesh redireksional:

1. > Simboli Redireksional

Sintaksa: Komanda Linux > filename

Per te cuar rezultatet e komandave ne linux ne nje file.Shenojme se nese fila ekziston rezultati do te mbivendoset te ajo file pa dhene asnje paralajmerim.

2. >> Simboli Redireksional

Sintaksa: Komanda Linux >> filename

Vendos rezultatin e komandes ne fund te files pa humbur informacionin qe ajo ka.

3. <Simboli Redireksional

Sintaksa: Komanda Linux < filename

Merr te dhenat nga nje file ne vend qe ti marre ato nga tastiera

Pipe

Nje pipe eshte nje metode qe sherben per te lidhur daljen e nje programi me hyrjen e nje programi tjeter, pa ndihmen e ndonje file temprorary.

Dalja e komandes I-re -> | -> Merr input nga komanda e pare

Nje pipe eshte nje magazine e perkohshme ku dalja e nje komande ruhet dhe me pas kalohet si hyrje per komanden e dyte.Pipet mund te perdoren per te

ekzekutuar me shume se dy komanda nga e njejta linje komandash.
Sintaksa command1 | command2

Disa komanda qe perdorin pipe	Domethenia e perdorimit te pipe
$ls | more	Dalja e komandes ls percaktohet si hyrje te komanda more. Keshtu dalja do te shfaqe nje faqe te plote per nje cast
$ who |sort	Dalja e komandes who percaktohet si hyrje ne komanden sort. Do te shfaqet lista e rradhitur me perdoruesit
$who | wc -l	Ketu do te numerohen perdoruesit qe jane loguar ne sistem
$ls -l | wc -l	Do te jepet numri i filave ne direktorine korrente
$who | grep raju	Do te shfaqet ne ekran ai perdorues me emrin raju, ne te kundert s`do te na shfaqet gje.

Filter
Nese nje komande ne linux pranon inputin e saj nga inputi standart dhe jep outputin e saj nga outputi standart, kjo gje do te njihet si filter. Nje filter kryen disa lloje procesimesh ne hyrje dhe jep nje informacion te caktuar ne dalje, psh nese kemi thirrur filen hotel.txt me 100 rreshta dhe nga kjo file na nevojitet vetem rreshtat nga 20 – 30 dhe me pas ta ruajme kete rezultat ne filen hlist atehere:
$tail +20 < hotel.txt | head -n30 > hlist
Ketu head eshte filter qe merr inputin e tij nga komanda tail (komanda tail i fillon selektimin nga rreshti me numrin 20 te files hotel.txt) dhe i kalon keto rreshta ne input te filtrit head, ku outputi eshte redirektuar te file hlist.

Prezantimi i programimit shell
Programi i Skripteve Shell eshte seria e komandave ne linux. Skriptet e shellit jane si filat batch ne MS-DOS, por jane me te fuqishem se to. Skriptet

ne shell mund te marrin input nga perdoruesi dhe ta shfaqin rezultatin ne ekran. Kjo ben te mundur automatizimin e disa detyrave perkatese.

Variablat ne Linux.

Shpesh here per te procesuar te dhenat,duhet ti ruajme ne RAM. Memorja RAM ndahet ne zona te vogla, ku cdo zone ka nje numer unik te quajtur lokaliteti(adresa) i memorjes. Programuesit mund te japin nje emer unik ne kete zone te quajtura variablat e memorjes, ose shkurt variablat. Ne linux ka dy lloje variablash:

1. Variablat Sistem – Te krijuara dhe te mirembajtura nga vete sistemi Linux. Keto tipe variablash shkruhen me germa kapitale
2. Variablat te percaktuar per perdoruesit (UDV) - Te krijuara dhe te projektuara per perdoruesit. Keto variabla shkruhen me germa te vogla.

Disa variabla sistem:

Ju mund te shikoni variablat sistem duke dhene komanden $set. Disa nga variablat me te rendesishem Sistem jane:

Variablat Sistem	Kuptimi
BASH=/bin/bash	Emri i shell-it tone
BASH_VERSION=1.14.7(1)	Emri i versionit te shellit tone
COLUMNS=80	Numri i kolonave per ekranin tone
HOME=/home/vivek	Direktoria jone home
LINES=25	Numri i rreshtave per ekranin tone
LOGNAME=student	Emri jone i logimit
OSTYPE=Linux	Tipi i sistemit tone operativ
PATH=/usr/bin:/sbin:/bin:/usr/sbin	Parametrat e pathit tone
PS1=[\u@\h\W]\$	Parametrat e promptit tone
PWD=/home/students/Common	Direktoria jone e punes
SHELL=/bin/bash	Emri jone i shellit
USERNAME=vivek	Emri i perdoruesit qe eshte aktualisht i loguar ne sistem

Shenojme qe disa nga parametrat ne kompjutera mund te jene te ndryshem. Ju mund te shfaqni shume nga variablat e mesiperm:
$echo $USERNAME
$echo $HOME
Kujdes: Mos i modifikoni varialbat e sistemit tuaj, sepse kjo mund te shkaktoje disa problem

Si te percaktojme variblat perdorues (UDV - User Define Variable)
Per te percaktuar variablat UDV:
emri i variblit=vlera
psh $no=10 # kjo eshte ok
$10=no # GABIM, vlera duhet te jete ne te djathte
psh per te percaktuar varialbin te quajtur “vech” me vleren BUS:
$vech=BUS
Per ti dhene variablit n vleren 10:
$n=10.
Rregullat per emertimin e variablave (Per UDV dhe Varialbat sistem)

1. Emrat e varialbave duhet te fillojne me karaktere alfanumerike ose me karakterin underscore (_), e ndjekur nga nje ose disa karaktere alfanumerike. psh nje variabel shell-i i rregull eshte:

HOME
SYSTEM_VERSION
vech
no

2. Mos vendosni hapsira ne te dy anet e barazimit, kur i jepni nje vlere te caktuar variablave. Psh ne deklarimin e varialbave te meposhtem nuk do te kete gabim:

$no=10
Por jane gabim:
no =10
no= 10
no = 10

3. Variablat jane case sensitive, sikurse emri i filave ne Linux. Psh...

$no=10
$No=11
$NO=20

nO=2
Te gjitha keto jane variabla te ndryshme. Per te shfaqur vleren 20 do te kemi: $echo $NO por nuk mund te shkruajme no ose No apo nO.

4. Ju mund te percaktoni variablin NULL si me poshte: (Variabli NULL nuk ka asnje vlere ne momentin e pare)

$vech=
$vech=””
Ketu nuk do te shfaqet gje ne ekran, pasi variablat nuk kane vlera. Per te shfaqur ne ekran vleren e variablit vech do te kishim: $echo $vech.

5. Nuk lejohen karakteret ?,* etj per te emertuar variablat.

Si te shfaqim apo te aksesojme vleren e UDV
Per te printuar apo aksesuar UDV do te perdorim sintaksen
Sintaksa:$emrivariablit.
psh per te shfaqur permbajtjen e variablit 'vech' do te kishim: $echo $vech
Ne ekran do te na shfaqet 'BUS' (nese vlera e ruajtur nga ky variabel ka qene vech=BUS. Kujdes nuk mund te shkruani $echo vech, pasi ne ekran do te shfaqet vech dhe jo BUS.

U1. Si mund ti japim variablit x vleren 10 dhe me pas ta shfaqim ne ekran.
$x=10
$echo $x

U2. Jepni variablit xn emrin RANI dhe shfaqeni ne ekran
$xn=RANI
$echo $xn

U3. Jepni ne ekran shumen e dy numrave, psh 6+3.
$echo 6+3
Ne ekran do te na shfaqet 6+3 jo shuma 9. Per te bere shumen matematike duhet te perdorim operatorin expr sipas sintakses:
expr op1 operator op2, ku op1 dhe op2 jane numra te plote dhe operator mund te jete +(Mbledhja), -(Zbritja),*(Shumezimi) dhe /(Pjestimi)
% sherben per te gjetur mbetjen psh 20/3=6 ku do te gjeje mbetjen e pjestimit 20/3 dhe do te jete 20%3=2

Atehere ne shprehjen tone do te kishim:
$expr 6 + 3
Ne ekran do te na shfaqet shuma 9
por shprehja $expr 6+3 nuk do te funksionoje pasi duhen hapsira para dhe pas operatorit +.

U4. Vendosni vlerat x=20 dhe y=5 dhe shfaqni ne ekran rezultatin e pjestimit tyre.
$x=20
$y=5
$expr $x /$ y
Ne ekran do te na shfaqet 20/4

U5. Modifikoni rastin e mesiperm duke perdorur variablin z
$x=20
$y=5
$z='expr x / y'
$echo $z

Si te shkruajme nje skript ne shell?
Le te shkruajme skriptin e pare ne ekran “Informacioni eshte fuqi”. Per te shkruajtur skriptin tuaj duhet te hapni editorin vi ose comanden cat e shoqeruar nga nje simbol redireksional (>).Keto komanda do te na cojne ne hapjen e nje teksti editor.

$cat > first
#
Ky eshte skripti i pare
#
clear
#Komanda clear do te sherbeje per te pasur ekranin sa me te paster
echo “ Informacioni eshte fuqi”

Ruajeni filen me ***CTRL*** + ***D*** per ta ruajtur. Tani skripti jone eshte gati. Per ta ekzekutuar $./first
Kjo komande mund te na sjelle error sepse nuk kemi vene te drejtat ne lidhje

me filen skript. Per ta rregulluar kete gje duhet:
$chmod +x first ose chmod 777 first, ku 777 do te jape te gjitha te drejtat per user-u, group-g, dhe other-0. (Dihet qe secili nga keto grupe ka 3 bit: r-read, w-write, x-execute, ku 7 ne formatin dhjetor eshte 111)
$./first
Fillimisht ekrani do te na pastrohet, me pas do te na shfaqet ne ekran fjala "Informacioni eshte fuqi" Per te shaqur mesazhin ne ekran ne perdorem komanden echo:
echo "Mesazhi"
echo " Mesazhi variabli1, variabli2,... variabliN"

Si te ekzekutojme nje skript ne Shell?
Sic thame per shkak te sigurise se filave, ne linux krijuesi i skriptit nuk mund ta ekzekutoje menjehere ate pa i dhene te drejtat ekzekutim. Per kete ju duhet te:
Perdorni komanden: chmod +x emri i skriptit tuaj ose chmod 777 emri i skriptit tuaj.
Nje forme tjeter mund te jete: $bash emri i skriptit
$/bin/sh emri i skriptit
Kujdes per ta ekzekutuar kete skript, ju duhet te jeni ne te njejten direktori ku eshte krijuar skripti, sepse nese ndodheni ne nje direktori tjeter, skripti nuk do t`ju ekzekutohet (per shkak te mos gjetjes se pathit). Per te pare direktorine tuaj korrente duhet te shtypni komanden $pwd (path work direktori). Psh nese skripti jone ndodhet ne /home/vivek/first. Nese levizim ne direktori te ndryshme dhe do te perpiqemi te ekzekutojme skriptin, ai nuk do te ekzekutohet. Per ta kaluar kete problem duhet qe te japim pathin absolut te skriptit tone, sa here qe do te duam ta ekzekutojme nga direktorite e tjera psh:
$/bin/sh /home/vivek/first
Ekzekutimi ne kete rast eshte pak i ngadalte dhe ju duhet te mbani mend te gjithe pathin.
Nje tjeter menyre eshte ne rast se ne do te dime qe te gjitha programet tona(ne formen e filave te ekzekutueshme) jane shenuar te ekzekutueshme ato mund te ekzekutohen direkt nga prompti nga cdo direktori(Per te shikuar ekzekutimin e programeve jepet komanda $ls -l /bin ose $ls -l /usr/bin).
Si ndodh kjo? Te gjitha filat e ekzekutueshme jane instaluar ne direktorine /bin dhe kjo direktori vendoset ne pathin e punes korente. Keshtu kur ne

shtypim emrin e cdo komande ne promptin $,ajo qe do te beje shelli eshte se fillimisht ai do te verifikoje komanden(e cila quhet si komande e brendshme, si pjese e vete Shell-it dhe eshte i afte te ekzekutoje pa pasur nevoje per fila te tjera te ekzekutueshme) dhe nese e gjen si komande te brendshme, shell-i do ta ekzekutoje ate, nese jo do te shikoje PATH-in,dhe do te perpiqet te gjej filen e ekzektueshme sipas komandes se dhene ne te gjitha direktorite e shkruajtura ne path. Nese e gjen e ekzekuton ate, ne te kundert jep mesazhin “bash:xxx : command not found”. Nje tjeter pyetje mund te jete, mund te ekzekutohet skripti njesoj si filat e ekzekutueshme. Pergjigja eshte Po, per kete qellim krijohet direktoria bin ne direktorine home dhe me pas kopjohet versioni i skriptit ne direktorine bin. Pas kesaj mund te ekzekutojme skriptin si fila te ekzekutueshme pa perdorur $ ./emer skripti. Pra, ndjekim keto hapa
#cd
$mkdir bin
$cp first ~/bin
$first
Le te shpjegojme komandat e mesiperme

Komanda	Shpjegimi
$cd	Shkojme ne direktorine home
$mkdir bin	Krijojme direktorine bin per te instaluar skriptin e shellit tone. Keshtu skripti mund te ekzekutohet si program i pavarur ose mund te aksesohet nga cdo direktori.
$cp first ~/bin	Kopjon skriptin first te direktoria bin
$first	Testojme nese skripti eshte duke u ekzekutuar apo jo (duhet te eci)

Ne skript komentet jepen me simbolin #. Keto komente injorohen nga shell-i, ato behen vetem per qellime shpjegimi.

Komandat qe kane lidhje me programimin Shell.

(1) echo [opsionet] [string,variablat...]

Shfaq tekstin ose vlerat e variablave ne ekran.

Opsionet:

-e Jep mundesine e interpretimit te karaktereve backslash ne stringe

\a alert (psh, zile)

\b backspace

\n rreshti i ri

\r enter

\t Tab-Horizontal

\\ backslash

psh. $echo -e “Informacioni eshte fuqi \a\t\n”

(2) Rreth shenjave te Kuotave “ '

Jane tre tipe:

Kuota dyshe “ - Cdo gje brenda “ “humbet domethenien e karaktereve me perjashtim te \ dhe $).

Kuota njeshe' – Cdo gje brenda ' ' mbetet e pandryshueshme

Kuota back` - Ekzekutojne komanden

psh $echo “Sot eshte data” Nuk do te na printoje mesazhin me daten e sotme, kurse $echo “Sot eshte 'date'” do te printohet data e sotme si: E marte Janar,.... Shiko `date` duke perdorur kuotat back ` `.

(3) Aritmetika ne Shell

Perdoret per te kryer operacionet aritmetike psh..

$expr 1 + 3

$expr 2 – 1

$expr 10 / 2

$expr 20 % 3 #lexohet si 20/3 mbetja 2

$expr 10 * 3 # Shumezimi eshte * ne vend te *

$echo `expr 6 + 3`

` ` Jane Kuotat back te cilat ndodhen te tasta me simbolin ~. Ne ekran do te na shfaqet vlera 9. Kujdes nese do te kishim $echo “expr 6 + 3” do te dale ne ekran expr 6 + 3.

(4) Procesimi i linjes se komandave

Le te vijojme me komanden (duke supozuar se fila shkolla nuk ekziston)

$ls shkolla

Do te na printohet nje mesazh i tille si:

shkolla: No such file or directory

ls eshte nje komande ne linjen e komandave kurse shkolla eshte argumenti perkates. Le te percaktojme komandat dhe argumentat per komandat ne vijim:
$ls foo
$cp y y.bak
$mv y.bak y.okay
$tail -10 myf
$mail raj
$sort -r -n myf
$date
$clear

Komandat	Nr i argumentave per komanden perkatese	Argumentat
ls	1	foo
cp	2	y dhe y.bak
mv	2	y.bak dhe y.okay
tail	2	-10 dhe myf
sort	3	-r -n dhe myf
date	0	
clear	0	
mail	1	raj

Pse kerkohen argumentat ne linjen e komandes?

Cdo komande duhet te kete argumentat e veta te lidhura ngushte me to sepse ne baze te argumentave percaktohet dhe ambjenti ku mund te operoje komanda psh rm {emer file}. Komanda rm do te fshije nje file por per te percaktuar se cila file ajo duhet te percaktohet si argument i kesaj komande.
Si i adresojme apo i aksesojme argumentat e linjes se komandave ne skripte? Le ta konkretizojme me nje shembull:
$cat > demostrimi
#/bin/sh
#Skripti i linjes komandave
echo " Nr total i argumentave ne linjen e komandave eshte $#"

```
echo "$0 eshte emri i skriptit"
echo "$1 eshte argumenti i pare"
echo "$2 eshte argumenti i dyte"
echo "Te gjitha argumentat jane: $*"
```

Skriptin e ruajme me tastat ***CTRL + D***

Per ta bere te ekzekutueshme duhet te ndjekim keto komanda:

```
$chmod +x demostrimi
$./demostrimi Hello World
$cp demostrimi ~/bin
$demo
```

Ne ekran do te na shfaqet:

```
 Nr total i argumentave ne linjen e komandave eshte 2
./demo eshte emri i skriptit
Hello eshte argumenti i pare
World eshte argumenti i dyte
Te gjitha argumentat jane :Hello World
```

Ky informacion do te merret per arsye se: Simboli $# do te jap nr e argumentave qe eshte i barabarte me 2 (Hello dhe World). $0 do te mare si argument emrin e skriptit, $1 do te mare si argument Hello dhe $2 do te mare si argument World. Kurse $* do te jap te gjithe argumentat dmth Hello World.

(5) Gjendja Exit

Ne linux nese ekzekutohet nje komande, ajo mund te ktheje dy vlera: 0 nese komanda eshte e sukseshme dhe nje numer te ndryshme nga 0 nese komanda si pasoj e disa gabimeve nuk eshte e sukseshme. Kjo vlere njihet si gjendja exit e komandes. Per te percaktuar gjendjen exit ne perdorim simbolin e $? psh nese shkruajme $echo $?, do te na shfaqet nje vlere e ndryshme nga 0, qe tregon se kemi nje gabim.

```
$ ls
a anaconda-ks.cfg bin demo demo~ Desktop first igli install.log
install.log.syslog mbox scsconfig.log scsrun.log
$echo $?
```

(6) if-then-fi sherben per te mare vendime ne baze te kushteve. Para se te fillojme me keto komanda kushtesh, le te permendim nje software i cili eshte free, qe quhet bc i cili do te sherbeje per te kryer disa

veprime artimetike dhe llogjike.

```
$ bc
bc 1.06
Copyright 1991-1994, 1997, 1998, 2000 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
```

Pra, pas kesaj komande, mund te kryejme disa veprime aritmetike si psh:

```
5 + 2
7
apo 5 > 2
1
```

Ku vlera 0 tregon FALSE, dhe 1 tregon TRUE.

Le te shikojme kushtin if i cili sherben per te mare nje vendim.Nese kushti eshte i vertete atehere ekzekutohet komanda 1

Sintaksa:

```
if kushti
then
komanda 1 nese kushti eshte i vertete ne te kundert gjendja exit kushti eshte 0
...
...
fi
```

Per krahasim ne mund te perdorim nje shprehje testimi. Nje shprehje eshte nje kombinim i vlerave, operatoreve relacional dhe operatoreve matematike (si >, <,<> etj) apo atyre matematike si (+ - /) etj.

Le te realizojme nje skript i cili do te therrase nje file si argument dhe do te shikoje permbajtjen e tij

```
cat > shiko_filen
#/bin/sh
#Realizojme nje skript per te shfaqur nje file
if cat $1
then
echo -e "\n\nFile $1, u gjet me sukses "
fi
```

Per ta ekzekutuar fillimisht i japim te drejten ekzekutim me ane te:

```
$chmod +x shiko_filen
$./shiko_filen shkolla
```

Emri i skriptit tone eshte shiko_file($0) dhe shkolla eshte argumenti ($1). Kjo do te na shfaqe permbajtjen e files shkolla(kete file duhet ta kemi krijuar me pare).

(7) Komanda e testimit [shprehje]

Komanda test perdoret per te shikuar nje shprehje nese eshte e vertete ose jo dhe nese eshte e vertete ai kthen vleren 0, ne te kundert kthen nje vlere te ndryshme nga 0. Le te provojme nje skript:

```
$cat > testo_numrin
#/bin/sh
#
# Skript per te shikuar nese argumenti eshte pozitiv
#
if test $1 -gt 0
then
echo ”$1 numri eshte pozitiv”
fi
```

Fillimisht $chmod +x testo_numrin

```
$testo_numrin 5
Ketu numri 5 eshte pozitiv
$testo_numrin -45
Nuk printohet gje
```

./testo_numrin do te sherbeje per te bere ekzekutimin

Nese do te te testonim dy shprehje atehere do te shkruanim:

```
$ cat testo_numrin
#/bin/sh
#
if test $1 -gt 0
then
echo "$1 nr i pare eshte pozitiv "
fi

if test $2 -lt 7
then
echo "$2 nr i dyte eshte me i vogel se 7"
fi
$ ./testo_numrin 6
```

6 nr i pare eshte pozitiv
./testo_numrin: line 8: test: -lt: unary operator expected (Sepse nuk kemi vene parametrin e dyte)
$ ./testo_numrin 6 5
6 nr i pare eshte pozitiv
5 nr i dyte eshte me i vogel se 7

Operatoret ne Skript	Kuptimi	Operacionet Aritmetike	Deklarimi test me komanden if	Deklarimi i shprehjes me komanden if
-eq	eshte e = me	5==6	if test 5 -eq 6	if expr [5 -eq 6]
-ne	nuk eshte e = me	5!=6	if test 5 -ne 6	if expr [5 -ne 6]
-lt	me e vogel se	5<6	if test 5 -lt 6	if expr [5 -lt 6]
-le	<=	5<=6	if test 5 -le 6	if expr [5 -le 6]
-gt	>	5>6	if test 5 -gt 6	if expr [5 -gt 6]
-ge	>=	5>=6	if test 5 -ge 6	if expr [5 -ge 6]

Shenim: == eshte barazim dhe != eshte e ndryshme

Operator	Kuptimi
string1 = string2	stringa1 eshte e = me stringen2
string1 != string2	stringa 1 e ndryshme nga stringa2
string1	stringa 1 nuk eshte definuar ose nuk eshte NULL
-n string1	Stringa 1 ekziston dhe nuk eshte NULL
-z string1	Stringa ekziston eshte NULL

Shell-i testohet per file dhe direktori

Test	Kuptimi
-s file	File e mbushur
-f file	Fila ekziston si file jo si direktori
-d dir	Eshte direktori dhe jo file.
-w file	Fila eshte e rishkrueshme
-r file	Fila eshte vetem e lexueshme
-x file	Fila eshte vetem e ekzekutueshme

Operatoret Logjike qe sherbejne per te testuar dy ose me shume kushte ne te njejten kohe

Operatori	Kuptimi
! shprehja	Not Logjik
shprehje1 -a shprehje2	And Logjik
shprehje1 -o shprehje2	Or Logjik

8).if....else...fi

Nese kushti eshte i vertete, atehere ekzekutohet komanda1, ne te kundert ekzekutohet komanda2.

Sintaksa:

if kushti
then
komanda1 nese kushti plotesohet, ose ne gjendjen exit nese kushti nuk plotesohet domethene eshte 0(zero)...
...
...
else
ekzekutohet komanda2 nese kushti nuk plotesohet, ose ne gjendjen exit nese kushti eshte >0 (jo zero)
...
...
...
fi

Le te shkruajme nje skript
$cat > isnump_n
#/bin/sh
#Skripti qe sherben per te testuar argumentin nese eshte pozitiv ose negativ
#
if [$# -eq 0]
then
echo “$0: Ju lutem jepni nje numer te plote”
exit 1
fi
if test $1 -gt 0
then
echo “$1 numri eshte pozitiv”
else
echo “$1 numri eshte negativ”
fi
Per ta ekzekutuar perdorim $chmod +x isnump_n dhe me pas
$./isnump_n {argumentat}

9).if-then-else me shume nivele

Sintaksa:
if kushti
then
kushti eshte 0 (true – 0) ekzekutohen te gjitha komandat siper elif
elif kushti1
kushti1 eshte 0 (true – 0) ekzekutohen te gjitha komandat siper elif
elif kushti2
kushti2 eshte 0 (true – 0) ekzekutohen te gjitha komandat siper elif
else
Asnje nga kushtet e mesiperme nuk eshte true (Te gjithe jane te ndryshme nga 0). Ekzekutohen te gjitha komandat siper fi
fi
Le te bejme nje skript i cili teston disa kushte:
$cat > testo
Ky skript teston nr e futur
if [$1 -gt 0]
then

```
echo “$1 numri eshte pozitiv”
elif [ $1 -lt 0 ]
then
echo “$1 numri eshte negativ”
elif [ $1 -eq 0 ]
then
echo “$1 numri eshte 0”
else
echo “OOOPS $1 nuk eshte numer, fut nje numer”
fi
```

Fillimisht sic dihet ekzekutojme skriptin: $ chmod 777 testo, me pas ./testo {numri}

10). Ciklet ne skriptet e shell-it

Kompjuteri mund te perseris instruksione te caktuara vazhdimisht, deri ne plotesimin e nje kushti. Nje grup instruksionesh i cili ekzekutohet ne menyre te perseritur quhet cikel (loop)

a) cikli for

Sintaksa: for {emer variabli } in { list }

do

ekzekuto cdo pike ne list deri ne fund (perserit te gjitha deklarimet midis do dhe done

done

Le te japim nje shembull:

Do te realizojme nje skript qe shfaq fjalen mireseerdhet 5 here

```
$cat > test_for
for i in 1 2 3 4 5
do
echo “Miresevini $i here”
done
```

Per ta ekzekutuar duhet te shtypni chmod 777 test_for ose chmod +x test_for dhe me pas e ekzekutojme me ./test_for

Le te perdorimi nje skript tjeter qe ka dhe kusht pervec ciklit

```
# Ne do te perdorim tabelen e shumezimit midis nje numri te futur nga tastiera :
cat > tabela_shumezimit
#Ky skript perllogarit shumezimin midis nje numri te futur si argument nga
```

linja e komandave dhe nr te mare me ciklin for

```
if [ $# -eq 0 ]
then
echo "Gabim, nr mungon ne linjen e komandave\n" cd
echo " Sintaksa eshte $0 numri"
exit 1
#exit 1 tregon se dalja do te behet si pasoj e nje gabimi
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
echo "$n * $i = `expr $i \* $n`"
done
```

Nje skript me tabelen aritmetike do te ishte:

```
# Ne do te perdorim tabelen arithmetike per disa numra te futur nga tastiera
$cat > tabela_arithmetike
if [ $# -eq 0 ]
then
echo "Gabim nr mungon \n"
echo " Sintaksa eshte $0 numri"
exit 1
fi
if [ $4 -eq 0 ]
then
echo " Numri i fundit duhet te jete me i madh se 0 "
exit 1
fi
n=$1
m=$2
k=$3
l=$4
for i in 0 1 2 3 4 5 6 7 8 9 10
do
echo "$n * $i = `expr $i \* $n`"
echo "$m + $i = `expr $i + $m`"
echo "$i - $k = `expr $i - $k`"
```

```
echo "$i / $l = `expr $i / $l`"
done
```

b) cikli while

Sintaksa:

```
while [ kushti ]
do
komanda 1
komanda 2
... komanda n
done
```

Cikli do te ekzekutohet per aq kohe sa te plotesohet kushti. Po ti referohemi shembullit te mesiperm duke perdorur ciklin while do te kemi:

```
$ cat > nt1
#skript qe teston deklarimin while
if [ $# -eq 0 ]
then
echo " Gabim, fut nje numer ne linjen e komandave"
echo " Sintaksa duhet te jete: $0 numer"
echo " Do te perdorim tabelen e shumezimit per nje numer te dhene"
exit 1
fi
n=$1
i=1
while [ $i -le 10 ]
do
echo " $n * $i = `expr $i \* $n`"
i=`expr $i + 1`
done
$ chmod 777 nt1
$ ./nt1 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

Le ti shpjegojme komandat e skriptit tone me nje komande:

n=$1	Vendos vleren e argumentit te linjes se komandes ne variablin n (Ketu eshte vendosur vlera 7)
i=1	Vendos variablin i ne 1
while [$i -le 10]	Ky eshte kushti per ciklin, nese vlera eshte me e vogel se 10, atehere shell-i ekzekuton komandat mes do dhe done
do	Fillon cikli
echo “$n * $i = `expr $i \* $n`”	Shfaqet tabela e shumezimit: 7 * 1 = 7 7 * 2 = 14 ..... 7 * 10 = 70. Ketu cdo vlere e variablit n shumezohet me i
i=`expr $i + 1`	Vlera e i-se rritet me 1
done	Cikli perfundon kur i arrin 10

11). Cikli case:

Ky deklarim eshte nje alternative e mire ne deklarimet shume niveleshe if-then-else-fi. Ai u jep mundesi te testoni disa vlera kundrejt nje variabli. Eshte i lehte per tu lexuar dhe shkruar.

Sintaksa:

case $emri i variablit in
modeli1) komanda ....
..... komanda; ;
modeli2) komanda....

.....komanda; ;
modeliN) komanda.....
.....komanda ; ;
* komanda
.....
.....
komanda ; ;
esac
$emri i variablit krahasohet kundrejt modeleve te ndryshme deri sa te gjendet nje rast. Shell-i me pas ekzekuton te gjitha komandat deri ne deklarimet mes dy ; ; . Default eshte *) dhe ekzekutohet ne raste kur asnje kusht nuk plotesohet. Le ta konkretojme me nje shembull.
Do te krijojme nje skript me emrin makina
$cat > makina
#Nese nuk eshte dhene emri i makines
psh -z $1 percaktohet dhe vlera e saj eshte NULL
Nese nuk ka asnje argument ne linjen e komandes
if [-z $1]
then
mer="asnje makine"
elif [-n $1]
then
#Ne te kundert bej argumentin e pare si mer
mer=$1
fi
case $mer in
"benz") echo " Makine e ngadalte";;
"golf") echo "Makine qe ecen deri ne 100 km/h";;
"bmw") echo "Makine qe ecen deri ne 200 km/h";;
"biciklete") echo "Bicikleta ecen deri ne 15 km/h";;
*) echo "Me vjen keq, nuk ka $mer per ty";;
esac
$ ./makina golf
Makine qe ecen deri ne 100 km/h
$./car ferrari
Me vjen keq, nuk ka ferrari per ty

Nje tjeter shembull do te jete:

```
$ cat > fakulteti
if [ -z $1 ]
then
pedagogu="mungon"
elif [ -n $1 ]
then
pedagogu=$1
fi
case $pedagogu in
"igli") echo "Igli eshte pedagog ne Sistemet Operative";;
"hakiku") echo "Hakiku eshte pedagog ne Gjuhet e Programimit";;
"evi") echo "Evi eshte pedagoge ne Algoritmike";;
"hergysi") echo "Hergysi eshte pedagog ne Elektronike";;
*) echo "Pedagogu $pedagogu";;
esac
$./fakulteti ev
Pedagogu ev
$./fakulteti evi
Evi eshte pedagoge ne Algoritmike
$ ./fakulteti
Pedagogu mungon
```

Nje shembull tjeter do te ishte po te perdornim komandat if else else if dhe fi, si ne shembullin e meposhtem:

```
cat > nr
#ky skript do te sherbeje per te testuar nje numer te futur nga tastiera
if [ -z $1 ]
then
test="s'ka numer"
elif [ -n $1 ]
then
test=$1
fi

if [ $1 -gt 0 ]
then
```

```
echo "Numri eshte pozitiv"
elif [ $1 -lt 0 ]
then
echo " Numri eshte negativ"
elif [ $1 -eq 0 ]
then
echo " Numri eshte zero"
else
echo "JU LUTEM fut nje numer sepse: $test"
fi
```

12). Deklarimi Read

Perdoret per te marre te dhena nga tastiera dhe per ta ruajtur ate ne variabel
Sintaksa read variabli1, variabli2,...variabliN

Le te realizojme nje shembull konkret:

```
$ cat > futboll
#Ky skript lexon nga tastiera dhe e shfaq ne ekran
read skuadra
echo "$skuadra jane skuadrat e zemres"
$ chmod +x futboll
$ ./futboll
Tirona Brazil
Tirona Brazil jane skuadrat e zemres
```

13). Shkurtimi i emertimit te files ose meta karakteret

Disa nga karakteret e shkurtuara jane: *,? ose [...]
Simboli * do te sherbeje per te dhene nje stringe ose nje grup karakteresh te caktuar
Psh $ls* do te tregoje te gjithe skedaret, $ls a* do te tregoje te gjithe skedaret emri i te cileve fillon me a, ose ls ut*.c do te tregoje te gjithe skedaret me prapashtesen c, por qe dy karakteret e para u fillojne me ut.
Simboli ? perdoret per nje karakter te vetem.
psh $ls ? do te shfaq vetem nje karakter te emrave te filave. Le te marrim nje shembull tjeter $ls fo?, kjo do te shfaq te gjithe filat emrat e te cileve jane me tre karaktere dhe i fillon me fo.
Simboli [...], kerkon kombinim me ato germa te futura brenda kllapave psh: $ls[abc]* , do te kerkoj te gjitha filat qe fillojne me karakteret a, b dhe c.

Nje forme tjeter do te ishte $ls[a-c]*, qe do te thote se do te kerkoje te gjithe skedaret emri i te cileve i fillon me a, b ose c.
Dhe dy format te tjera, por kesaj rradhe te perjashtimit jane: $ls[!a-c]* ose $ls[^a-c]*, qe tregon se do te shfaq te gjitha filat pervec atyre qe fillojne me germat a, b dhe c.

14). komanda1;komanda2

Kjo sherben per te ekzekutuar dy komanda brenda te njejtes linje komandash te ndara me ; psh: $date;who. Ketu do te na shfaqen te gjithe perdoruesit e futur sipas nje date

Skripte te avancuara

Per te hedhur programet e padeshirueshme linux ka nje file special te quajtur null e cila eshte e lokalizuar ne direktorine dev. Shembulli do te ishte:
$ls > /dev/null.
Direktoria dev permban te gjitha skedaret e pajisjeve. Skedaret ne kete direktori ne pergjithsi shfaqin pajisjet periferike si disqe, floppy disqe, karte zeri, rreshta printimi etj.

Variablat Shell lokal dhe global

Normalisht te gjithe variablat jane lokal. Variablat lokal mund te perdorin te njejtin shell, por nese ngarkohet nje kopje tjeter e shell-it (psh, duke shtypur

/bin/bash ne promptin $) atehere shelli i ri injoron te gjitha variablat e shellit te vjeter. Le te marrim nje shembull:

```
[root@localhost ~]# cd /
[root@localhost /]# cd home
[root@localhost home]# cd itafaj
[root@localhost itafaj]# echo $a

[root@localhost itafaj]# a=2
[root@localhost itafaj]# echo $a
2
[root@localhost itafaj]# export a
[root@localhost itafaj]# cd ..
[root@localhost home]# cd ..
[root@localhost /]# cd bin
[root@localhost bin]# echo $a
2
[root@localhost bin]# cd /
[root@localhost /]# cd mnt
[root@localhost mnt]# echo $a
2
[root@localhost mnt]#
```

Nese nuk do te perdornim komanden export atehere vlera e a-se sapo te iknim nga direktoria home do te na humbte.

Kushtet e ekzekutimit And(&&) dhe Or (||)
Komanda and ka sintaksen:
komanda1 && komanda2
Komanda 2 do te ekzekutohet atehere dhe vetem atehere kur komanda 1 te ktheje nje vlere 0.
Komanda or ka sintaksen:
komanda1 || komanda2
Komanda 2 do te ekzekutohet atehere dhe vetem atehere kur komanda 1 te ktheje nje vlere te ndryshme nga 0.
Ne mund ti perdorim te kombinuara te dy keto komanda ne kete menyre:
Nese komanda 1 ekzekutohet me sukses atehere shelli do te ekzekutoje

komanden 2 dhe nese komanda 1 nuk perfundon me sukses atehere ekzekutohet komanda 3. Le ta konkretizojme me nje shembull:

```
[root@localhost itafaj]# ls
date  emer  fshi  fshi~  matematika  permbajtja
[root@localhost itafaj]# rm fshi~ && echo "Fila u fshi" || echo "Fila nuk u gjet"
rm: remove regular file `fshi~'? y
Fila u fshi
[root@localhost itafaj]# rm kot && echo "Fila u fshi" || echo "Fila nuk u gjet"
rm: cannot lstat `kot': No such file or directory
Fila nuk u gjet
[root@localhost itafaj]#
```

Redireksionet I/O dhe deskriptoret e filave.
Sic e dime redirektoret jane perdorur per te derguar daljen e nje komande ne nje file ose anasjelltas per te lexuar nga nje file.

```
[root@localhost itafaj]# cat > myfil
[root@localhost itafaj]# cal
   December 2010
Su Mo Tu We Th Fr Sa


                1  2  3  4
    5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

[root@localhost itafaj]# cal > myfil
[root@localhost itafaj]# cat myfil
   December 2010
Su Mo Tu We Th Fr Sa
                1  2  3  4
```

```
      5  6  7  8  9 10 11
 12 13 14 15 16 17 18
 19 20 21 22 23 24 25
 26 27 28 29 30 31
```

[root@localhost itafaj]#

Nje tjeter komande eshte sort, le ta japim me nje shembull:
[root@localhost itafaj]# sort
5
3
7
8
1
Shtypim CTRL D dhe do te na shfaqet rradha e numrave te futur nga ne
1
3
5
7
8
Le ta japim me nje shembull tjeter:
[root@localhost itafaj]# sort
[root@localhost itafaj]# cat > nos
5
3
7
8
1
[root@localhost itafaj]# sort < nos
1
3
5
7
8
[root@localhost itafaj]#

Nje shumebull tjeter do te ishte renditja e emrave ne nje skedare sipas rendit alfabetik, psh:

[root@localhost itafaj]# cat rradhit
[root@localhost itafaj]# sort > rradhit
Igli Tafaj
Hakik Paci
Hergys Rexha
Evis Trendafili
“Dalim duke shtypur CTRL D”
[root@localhost itafaj]# cat rradhit
Evis Trendafili
Hakik Paci
Hergys Rexha
Igli Tafaj
[root@localhost itafaj]#

Ne linux sikurse ne gjuhen e programimit C, tastiera, ekrani etj trajtohen si skedare (fila). Me poshte po japim emrat dhe nje pershkrim te tyre me ane te nje tabele:

Fila Standart	Nr i deskriptorit	Perdorimi	Shembull
stdin	0	si standarti hyrjes	Tastiera
stdout	1	si standarti daljes	Monitori
stderr	2	si standarti i gabimeve	Ekrani

Default ne Linux cdo program ka tre fila te shoqeruar me te (kur hapni nje program keto tre fila automatikisht hapen nga shell-i). Perdorimi i dy fila-ve te pare (stdin dhe stdout) jane tashme te perdorur nga ne. Fila e fundit stderr perdoret nga programi per te printuar gabimet ne ekran.

Mund te bejme redirekt nga dalja e nje file deskriptor direkt ne nje file tjeter:
Sintaksa: Nr i files desktiptor > emri i files.
psh nese:
$rm gabim_emri_files111
rm: cannot lstat `gabim_emri_files111': No such file or directory
Nese shkruajme:

$rm gabim_emri_files111 > er
do te na shfaqet e njejta gje:
rm: cannot lstat `gabim_emri_files111': No such file or directory
Kjo gje ndodh pasi nuk mund te shfaqim fshirje e nje file me simbolin redirekt > . Zgjidhja do te ishte:
$ rm gabim_emri_files111 2>er
$ cat er
rm: cannot lstat `gabim_emri_files111': No such file or directory
Pra duhet te shtojme nje 2>, dmth duhet te kalojme nga formati redirekt ne direkt duke shtuar nje 2.
Le te japim nje shembull:

```
$ cat > demoscr
if [ $# -ne 2 ]
then
echo "Gabim numrat nuk jane futur"
echo "Perdor $0 numri1 numri2"
exit
fi
ans=`expr $1 + $2`
echo "Shuma eshte $ans"
```

Nese do ta ekzekutojme ./demoscr 5 4 , Ne ekran do te na shfaqet: Shuma eshte 9.
Nese do te ekzekutojme ./demoscr
Do te na shfaqet mesazhi: Gabim numrat nuk jane futur
Perdor ./demoscr numri1 numri2
$./demoscr > er1
$ cat er1
Gabim numrat nuk jane futur
Perdor ./demoscr numri1 numri2
Ketu per ekzekutimin e pare, skripti printon nje mesazh gabimi qe tregon se nuk jane futur dy numra.
Ne rastin e dyte, kemi bere redirekt te skriptit drejt files, kjo do te thote se do te na printohet mesazh gabimi ne stderr jo ne stdout. Per ta kaluar kete problem e zevendesojme deklarimet e mesiperme si vijon:
echo “Gabim : Numrat nuk jane futur” 1>&2
echo “Perdor: $0 numri1 numri2” 1>$2

Tani nese: $./demoscr > er1
Gabim numrat nuk jane futur
Perdor ./demoscr numri1 numri2
Ajo do te printoje mesazhin e gabimit ne stderr dhe jo ne stdout. 1>&2 ne fund te deklarimit echo, drejton stdout ne stderr
Pra sintaksa: from>&destinacion
$ ./demoscr > er2
Gabim numrat nuk jane futur
Perdor ./demoscr numri1 numri2
$./demoscr 2> er2
$ cat er2
Gabim numrat nuk jane futur
Perdor ./demoscr numri1 numri2

Funksionet
Funksioni eshte seria e instruksioneve apo komandave. Funksionet kryejne aktivitete specifike ne shell. Per te percaktuar funksionin perdorni:
Sintaksa:
emri-funksionit ()
{
komanda 1
komanda 2
...
komanda N
return
}
Ku emri i funksionit eshte emri i funksionit qe ekzekuton komandat perkatese. Deklarimi return do te mbylle funksionin. Psh:
$sayhello()
> {
> echo "Hello $LOGNAME, Kalofsh bukur"
> return
> }
Per ta ekzekutuar funksionin duhet te shtypim thjesht emrin e funksionit :
$sayhello
Hello user, Kalofsh bukur

Shenojme se pasi te bejme ristartim te kompjuterit, do ta humbim funksionin sayhello(). Per ta eleminuar kete gje duhet ta ruajme funksionin tone ne filen /etc/bashrc.Le te bejme nje shembull me funksionin today() i cili do te sherbeje per te shfaqur formatin e dates. Fillimisht duhet te logohemi si root:

```
$ su
password:  xxxxxx
```

hapni filen si:

```
#vi /etc/bashrc
```

Ne fund te files shtoni:

```
#funksioni today() per te printuar formatin e dates
today()
{
echo " Kjo eshte nje `date + "%A %d in %B of %Y (%r)"`
return
}
```

E ruajme filen dhe dalim. Pas ketyre modifikimeve fila do te duket:

```
#/etc/bashr
PS1="[\u@\h \W]\\$"
#funksioni today() per te printuar formatin e dates
today()
{
echo " Kjo eshte nje `date + "%A %d in %B of %Y (%r)"`
return
}
```

Per te ekzekutuar funksionin duhet dalim qe aty duke shtypur exit apo CTRL D dy here dhe me pas shtypni $today, qe do ta bej kete funksion te vlefshem te te gjithe perdoruesit ne sistemin tuaj. Nese keni deshire te shtoni funksione te vecanta te nje perdorues te caktuar atehere hap filen .bashrc ne direktorine home si me poshte:

```
#vi .bashrc
```

Ne fund te files shto:

```
saybuy()
{
echo "Blej $LOGNAME !"
echo " Shtyp nje taste per logout"
read
```

```
return
}
```

Ruajeni dhe dilni qe aty. Pas ketyre modifikimeve fila do t`ju duket:

```
#.bashrc
# User specific aliases and functions
#Source global definitions
if [ -f /etc/bashrc ]; then
./etc/bashr
fi
SayBuy()
{
echo “Blej $LOGNAME !”
echo “ Shtyp nje taste per logout”
read
return
}
```

Ndiqni te njejtat rregulla sic ndoqet me pare

Nderfaqja perdorues dhe utilitetet dialog

Skriptet ne shell duhet te ndermjetesojne me perdoruesin. Le te japim nje shembull:

```
cat > userint
#Skript qe demostron echo dhe lexon komanden per nderveprimet perdorues
#
echo "Emri juaj ju lutem:"
read emri
echo  "Mosha juaj ju lutem:"
read mosha
vt=`expr $mosha + 1`
echo "Pershendetje $emri, vitin tjeter ju do te jeni $vt vjec."
$./userint
Emri juaj ju lutem:
ola
Mosha juaj ju lutem:
28
Pershendetje ola, vitin tjeter ju do te jeni 29 vjec.
```

Le te realizojme nje skript tjeter qe shfaq nje menu:

```
cat > menuui
#Skript per te krijuar nje menu te thjeshte dhe te kryeje disa veprime ne lidhje me ate qe selektojme
#pikat e menuse
while :
do
clear
echo"_____"
echo "Menuja Kryesore"
echo "_____"
echo "[1] Shfaq sot daten/oren"
echo "[2] Shfaq skedaret ne direktorine korente"
echo "[3] Shfaq kalendarin"
echo "[4] Fillo me editorin per te shkuar nje leter"
echo "[5] Dil/Stop"
echo "==========="
echo -n " Zgjidh menune [1-5]:"
read llojin
case $llojin in
1)echo "Sot eshte `date`, Shtyp enter per te dale...";read;;
2)echo "Skedaret ne `pwd`"; ls -l ; echo "Shtyp enter per te dale...";read;;
3)cal ; echo "Shtyp enter per te dale ...";read;;
4)vi;;
5)exit 0;;
*)echo "Opps!!!, Ju lutem zgjidhni 1,2,3,4,ose 5";
echo "Shtyp enter ...";read;;
esac
done
```

Le te sqarojme rast per rast me nje tabele

Deklarimet	Shpjegimi
while :	Fillohet me nje cikel pafund. Ky cikel perfundon duke shtypur 5 (Psh Exit/Stop)
do	Fillon cikli

clear	Pastron ekranin
echo"_____" echo "Menuja Kryesore" echo "_____" echo "[1] Shfaq sot daten/oren" echo "[2] Shfaq skedaret ne direktorine korente" echo "[3] Shfaq kalendarin" echo "[4] Fillo me editorin per te shkuar nje leter" echo "[5] Dil/Stop" echo"=========="	Shfaq menune ne ekran
echo -n " Zgjidh menune [1-5]:"	Kerkohet perdoruesit te fus nje numer
read llojin	Lexohen pikat e menuse te dhene me numra nga useri
case $llojin in 1)echo "Sot eshte `date`, Shtyp enter per te dale...";read;; 2)echo "Skedaret ne `pwd`"; ls -l ; echo "Shtyp enter per te dale...";read;; 3)cal ; echo "Shtyp enter per te dale ...";read;; 4)vi;; 5)exit 0;; *)echo "Opps!!!, Ju lutem zgjidhni 1,2,3,4,ose 5"; echo "Shtyp enter ...";read;; esac	Kryej veprimin e duhur ne lidhje me menune e zgjedhur. Nese numrat e futur nuk jane nga 1 ne 5, atehere do te na shfaqet mesazh gabimi.
done	Ndalo ciklin, nese numri i shtypur eshte 5.

Pergjithsisht nderfaqja perdorues perfshin, menute, tipe te ndryshme kutish dialogu si: Kuti informative, kuti mesazhi, etj. Ne Shell-in e Linux nuk ka

mjete te mundshme per te krijuar nderfaqen perdorues. Gjithsesi ne Linux **Red Hat versioni 6** ka nje mjet te quajtur *dialog* e cila perdoret per te krijuar tipe te ndryshme kuti dialogu. Le te japim nje shembull:

```
dialog --title "Miresevini ne Linux" --backtitle "Tutorial me skriptet ne Linux" --infobox "Kjo eshte nje kuti dialogu me\ infobox,qe eshte perdorur per te treguar disa informacione ne ekran.Shtyp enter ... " 7 50; read
```

I japim te drejten e ekzekutimit chmod +x dia dhe me pas e ekzekutojme me ./dia
Pasi ta ekzekutojme do te shohim ne ekran nje kuti dialogu: “Miresevini ne Linux dhe me mesazhin “ Kjo eshte nje kuti dialogu...Shtyp enter.
Ketu 7 dhe 50 jane lartesia dhe gjeresia e kutise se dialogut.
Sintaksa:
dialog –title {title} –backtitle {backtitle} {Opsionet}
Ku opsionet e kutise mund te mare nje nga opsionet:
--yesno {text} {lartesia} {gjeresia}
--msgbox {text} {lartesia} {gjeresia}
--infobox {text} {lartesia} {gjeresia}
--inputbox {text} {lartesia } {gjeresia}
--textbox {file} {lartesia } {gjeresia}
--menu {text} {lartesia } {gjeresia} {menu} {height} {tag1} item}...

msgbox perdorin mjetet dialog
Le te japim nje shembull.

```
cat >dia2
dialog --title "Miresevini ne Linux" --backtitle "Tutorial me skriptet ne Linux" --msgbox "Kjo eshte nje kuti dialogu me\ msgbox,qe eshte perdorur per te treguar disa informacione ne ekran.Shtyp enter ... " 9 50; read
```

e ekzekutojme me chmod 777 dia2 dhe ne fund e ekzekutojme ./dia2
Le te marim dhe nje shembull tjeter:

```
cat>dia2
dialog --title "Kujdes : Fshi file" --backtitle " Skript ne Linux" --yesno "A\
```

```
deshiron ta fshini filen '/home/itafaj/desktop/notat'\" 7 60
sel=$?
case $sel in
0) echo " selekto nese deshiron te fshish filen";;
1) echo " selekto nese nuk deshiron te fshish filen";;
255) echo "Anullo duke shtypur tasten ESC";;
esac
```

I japim chmod 777 dia2
dhe ekzekutimi do te realizohet me ./dia2
Le te realizojme nje skript tjeter:

```
cat >smenu
# Si te krijojme nje menu te vogel duke perdorur kutite e dialogut
dialog --backtitle "Linux skript" --title "Menuja Kryesore" --menu "
Zhvendosu\
duke perdorur [UP] [DOWN] dhe [ENTER] ne Zgjidh" 15 50 3 \
Date/time "Tregon daten dhe oren"\
Calendar "Per te pare kalendarin"\
Editor "Per te filluar editorin vi" 2>/tmp/pikatemenuse.$$
pikatemenuse=`cat /tmp/pikatemenuse.$$`
opt=$?
case $pikatemenuse in
Date/time) date;;
Calendar) cal;;
Editor) vi;;
esac
rm -f /tmp/menuitem.$$
```

Dalim me CTRL D dhe me pas shtypim chmod 777 smenu dhe ne fund e ekzekutojme.

--opsionet e menuse	Kuptimi
Zhvendosu\ duke perdorur [UP] [DOWN] dhe [ENTER] ne Zgjidh" 15 50 3 \	Ky tekst shfaqet para menuse
15	Lartesia e kutise

50	Gjeresia e kutise
3	Lartesia e menuse
Date/time “ Tregoni daten dhe oren”	Fillimisht menuja therritet tag1 (psh Date/time) dhe pershkrimi per menu therritet si item1 (psh te shikojme Tregon daten dhe oren)
Calendar "Per te pare kalendarin"\	Fillimisht menuja therritet tag2 (psh Kalendar) dhe pershkrimi per menu therritet si item2 (psh te shikojme Kalendarin)
Editor "Per te filluar editorin vi"	Fillimisht menuja therritet tag3 (psh Editorin) dhe pershkrimi per menu therritet si item3 (psh te nisim editorin vi)
/tmp/pikatemenuse.$$	Dergojme pikat e zgjedhura ne menu (tag-et) te fila temprorary.

Sic do te kuptohet mjeti i dialogut ka me shume fuqi se operacionet ndervepruese me njeriun echo apo read. Problemi i vetem me mjetin dialog eshte se punon ngadale.

Komanda trap.
Le te marrim skriptin
$cat > testsign
ls -R /

E ruajme dhe e ekzekutojme. Do te shohim nese shtypim CTRL C, gjate kohes se ekzekutimit te skriptit, skripti do te perfundoje. CTRL C ketu punon si sinjal. Kur ky sinjal vepron, ai dergohet te te gjithe proceset korrente ne ekzekutim ne sistem.
Le te marrim nje skript tjeter:
#

```
# Perdorimi i sinjalit trap, versioni 1
#
Merr_input1()
{
rregj=0
clear
echo "Sheno takimet"
echo -n " Futni emrin e files database:"
read emri_files
if [ ! -f $emri_files ];then
echo "Me vjen keq por, $emri_files nuk ekziston, krijo $emri_files database"
echo "Fila database qe ruan takimet" > $emri_files
fi

echo "Fut te dhenen: `date`" > /tmp/inpu0.$$
#
#Percaktohet nje cikel pafund
#
while :
do
echo -n "Titulli i takimit:"
read tak
echo -n "Koha e takimit:"
read koh
echo -n "Shenimet:"
read shen
echo -n "Eshte e sakte e dhena (y/n)?"
read pergj

if [ $pergj = y -o $pergj = Y ];then
rregj=`expr $rregj + 1`
echo "$regj.$tak $koh $shen" >> /tmp/input0.$$
fi
echo -n "Shto nje takim tjeter (y/n)?"
read tjeter
if [ $tjeter = n -o $tjeter = N ];then
```

```
cat /tmp/input0.$$ >> $emri_files
rm -f /tmp/input0.$$
return # perfundimi i ciklit
fi
done
}
#
#Therrit funksionin perdorues : Merr_input1
Merr_input1
```

E ruajme, i japim te drejten ekzekutim me chmod +x dhe ne fund e ekzekutojme me ./testsign1
do te shofim:

```
Sheno takimet
Futni emrin e files database:shkolla1
Me vjen keq por, shkolla1 nuk ekziston, krijo shkolla1 database
Titulli i takimit:kot
Koha e takimit:13.30
Shenimet:Eshte per qef
Eshte e sakte e dhena (y/n)?y
Shto nje takim tjeter (y/n)?y
Titulli i takimit:shkolla2
Koha e takimit:sot
Shenimet:kot fare
Eshte e sakte e dhena (y/n)?n
Shto nje takim tjeter (y/n)?n
$
```

Pra ne fund dalim nga cikli
Problemi do te ekzistonte nese gjate ekzekutimit te ketij skripti,shtypim CTRL C, skripti do te mbyllet dhe fila temprorary ka shkuar ne direktorine /tmp. Psh:

```
$./testsign1
```

Pasi japim emrin e files database dhe pasi shtojme te pakten nje rekord takimi ne filen temprorary, shtypim CTRL C. Skripti jone do te mbyllet dhe do te largohet ne direktorine /tmp. Mund ta verifikojme kete duke shtypur:

$ls /tmp/input*
Skripti jone ka nevoje te dedektohet kur vjen ky sinjal. Per te arritur kete ne fillimisht duhet te dedektojme sinjalin duke perdorur komanden trap
Sintaksa: trap {komandat} {lista e numrit te sinjaleve}

Nr i sinjalit	Kur ndodh
0	Dalje nga shell-i
1	Hangup
2	Interrupt (CTRL C)
3	dalje
9	kill (nuk mund te kapet)

Per te kapur kete sinjale ne skriptin e mesiperm, vendosim deklarimin trap, para se te therrasim funskionin Merr_input1 si trap del_file2. Ketu komanda trap therret fshi_file() kur ndodh interrupti me numrin 2 (CTRL C). Le ta modifikojme skriptin e mesiperm:

```
#
# Perdorimi i sinjalit trap, per te fshire filen temprorary, versioni 2
#
fshi_file()
{
echo " ***Kur ndodh CTRL + C (do te na fshihet fila temprorary) ***"
rm -f /tmp/input0.$$
exit 1
}
Merr_input1()
{
rregj=0
clear
echo "Sheno takimet"
echo -n " Futni emrin e files database:"
read emri_files
if [ ! -f $emri_files ];then
echo "Me vjen keq por, $emri_files nuk ekziston, krijo $emri_files database"
echo "Fila database qe ruan takimet" > $emri_files
```

```
fi

echo "Fut te dhenen: `date`" > /tmp/inpu0.$$
#
#Percaktohet nje cikel pafund
#
while :
do
echo -n "Titulli i takimit:"
read tak
echo -n "Koha e takimit:"
read koh
echo -n "Shenimet:"
read shen
echo -n "Eshte e sakte e dhena (y/n)?"
read pergj

if [ $pergj = y -o $pergj = Y ];then
rregj=`expr $rregj + 1`
echo "$regj.$tak $koh $shen" >> /tmp/input0.$$
fi
echo -n "Shto nje takim tjeter (y/n)?"
read tjeter
if [ $tjeter = n -o $tjeter = N ];then
cat /tmp/input0.$$ >> $emri_files
rm -f /tmp/input0.$$
return # perfundimi i ciklit
fi
done
}
#
#Kur ndodh interrupti CTRL + C ai fillimisht therret fshi_file() dhe me pas
del
trap fshi_file 2
#Therret funksionin perdorues: Merr_input1
Merr_input1
```

Atehere per ta ekzekutuar do te shtypim ./testsign1
dhe do te kemi:
Sheno takimet
Futni emrin e files database:shkollaa
Me vjen keq por, shkollaa nuk ekziston, krijo shkollaa database
Titulli i takimit: ***Kur ndodh CTRL + C (do te na fshihet fila temprorary) ***
Pra ne pasi futem emrin e files me pas shtypem tastat CTRL C. Kjo taste automatikisht do te therrase interrupt 2 i cili do te fshi filen /tmp/input0.$$
Keshtu nese shtypim komanden:
$ ls /tmp/input*
Do te na shfaqet
ls: /tmp/input*: No such file or directory

Komanda getopts
Kjo komande perdoret per te verifikuar vlefshmerine e argumentit ne linjen e komandes te futur ne skript. Pergjithsisht perdoret ne ciklin while
Sintaksa: getopts {optstring} {variable1}
Kjo komande perdoret nga shell-i per te analizuar argumentet ne linjen e komandes, ku optstring permban opsionin e karaktereve te shenuara, psh nese nje karaktere ndiqet nga nje presje, opsioni duhet te kete nje argument, qe do te ndahet nga ai nga nje hapsire.Kur nje opsion kerkon nje argument, getopts vendos ate argument ne variablin OPTARG. Nje mesazh gabimi getopts shfaqet kur opsionet ilegale ose mungesa e opsioneve te argumentave jane ***encounter.*** Kur shikohet nje opsion ilegal, getopts vendos ? ne variablin 1.
Psh nese kemi nje skript te quajtur ani qe ka sintaksen:
ani -n -a -s -w -d
Opsionet: Keto jane argumenta opsionale
-n emri i kafshes
-a mosha e kafshes
-s seksi i kafshes
-w pesha e kafshes
-d vlera demo

Le te japim nje skript te thjeshte:

```
# Perdorimi i ani -n -a -s -w -d
# help_ani() Per te shfaqur help
help_ani()
{
echo "Perdor:$0 -n -a -s -w -d"
echo "Opsionet: Keto jane argumenta opsional"
echo " -n emri i kafshes"
echo " -a mosha e kafshes"
echo " -s seksi i kafshes"
echo " -w pesha e kafshes"
echo " id vlerat demo
exit 1
}
#Fillo proceduren kryesore
#Vendos vleren default per variablin
isdef=0
na=Moti
mosha="2 muaj"
sex=mashkull
pesha=3kg
#Nese nuk ka argument
if [ $# -lt 1 ]; then
help_ani
fi
while getopts n:a:s:w:d opt
do
case "$opt" in
n)na="$OPTARG";;
a)mosha="$OPTARG";;
s)seksi="$OPTARG";;
w)pesha="$OPTARG";;
d)isdef=1;;
\?)help_ani;;
esac
done
```

```
if [ $isdef -eq 0 ]
then
echo "Emri i kafshes: $na,Mosha:$mosha, Seksi:$seksi, Pesha: $pesha
(menyra e percaktimit perdorues)"
fi
```

I japim te drejten ekzekutim dhe ne fund e ekzekutojme
$./ani
Perdor:./ani -n -a -s -w -d
Opsionet: Keto jane argumenta opsional
-n emri i kafshes
-a mosha e kafshes
-s seksi i kafshes
-w pesha e kafshes
id vlerat demo
$./ani -n sedi -a 2 -s femer -w 2 kg
Emri i kafshes: sedi, Mosha: 2, Seksi: femer, Pesha: 2
$ ./ani 5
Emri i kafshes: Moti, Mosha: 2 muaj, Seksi: , Pesha: 3kg
$ ./ani -a 2
Emri i kafshes: Moti, Mosha: 2, Seksi: , Pesha: 3kg

Ushtrime me skripte ne shell:
U1. Shkruani nje skript qe do te mbledh 2 numra qe jane dhene si argumenta ne linjen e komandave. Nese nuk futen dy numra nxjerr nje mesazh gabimi
U2. Shkruani nje skript qe kerkon numrin me te madh nga tre te dhene. Keto tre numra jane futur si argumenta ne linjen e komandes. Shfaq nje gabim nese nuk eshte dhene argumenti
U3.