Unix :

- Unix is an operating system like windows, which is developed at Bell labs in 1969 as an interactive time sharing system.
- Ken Thomson & Dennis Ritchie are the two inventors of unix.
- In 1974 unix becomes the first OS (official) written in 'C' language
- Unix is evolve large freeware product with many extentions and new ideas.
- Unix and associated OS (linux) is using in every departments (university, industries, companies, individuals) as an operating system to perform and interact with system. an operations.
- Unix has many flavours some of them are in below.

  Novell – (latest version)
  HP-UX – 11.31
  IBM-AIX – 7.2
  SUNOS – 11.3
  MAC OS

- In 1985 Richard Stallman created a free S/w with help of Unix in order to spread S/w as freely and named it as Linux.
- The first version of Linux is released in 1991.
- Linux is a clone of unix and below are the differences.

Linux	Unix
– Linux is an open source S/w and free OS.	– unix is an OS and it is commercial
– Linux is a GUI bant	– Unix is a CUI
– Security around 60 – 100 Verily of security of viruses installed.	– 85 – 120 viruses installed till date.

Linux is also having different flavours

They are

	Versions
→ Redhat. 2	9.
→ Ubuntu	– 16.10
→ Sushi 4.	– 12.1
→ Fedora 3	23

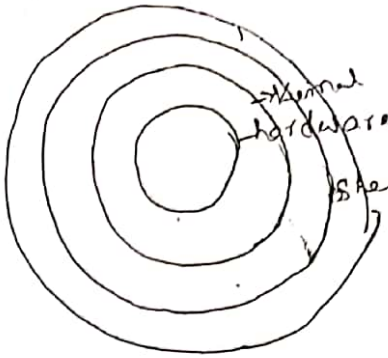
Scanned with CamScanner

Differences b/n windows & Unix.

windows	Unix
- It doesn't support multithreading	- It supports multithreading
- Less Secure	- More Secure
- It's a desktop oriented.	- It is command oriented.
- Easy to understand	- difficult to understand
It is a GUI	- It is a CUI
It is a Commercial	- It is also commercial but some flavours of unix are free.
No multiprocessing.	- Multiprocessing.
File system is in flat type	- It is hirarchical model.
It is a viruses exist in window we need to install external antivirus.	- No virus.
- less stable	- More stable
It runs all processes. when it starts	- It runs required process, starts only required processes.
- It can't support python and other programming codes.	- It supports all kind of codes which is developed by using any language.
- Easy to maintain and recovery	- Bit bit difficult to maintain and recovery.

Scanned with CamScanner

Architecture of Unix.



Unix Architechture is having 4 layers

1. Tools & Applications :-

In this layer we can able to run our programs and commands.

- putty is one of the tool to connect unix os which is installed in another machine.
- winscp, Core ftp are the GUI Tools to connect Unix os from our system. N Vs W which are used to transfer the files b/n windows to unix.

2). Shell :-

It is a command line interpriter which interprits with commands

- It acts as mediator b/n Kernal & user
- The commands are which are executing by the users will convert to kernal understandable format by shell then again it will convert to user understandable format, which are giving by kernal
- Below are the different types of shells available in unix.

Sh - Bourne shell

KSh - Korne shell

Bash - Bourne again shell

Csh - C-shell

Tcsh - Tenthmes faster than Cshell. (TENEX C Shell)

Scanned with CamScanner

- usually shell is a language to communicate with unix OS. These are inbill with unix OS. when we install unix OS these shells will be installed automatically.

'/etc/shell' is a file contains all available shells.

- There are some differences bln each shells the forst difference is exporting variables to the system.

** In ksh

Syntax:- Var-name = Value
Export Variable name.

Ex:- a=10
Export a.

ksh

Syn:- Export variable_name = Value

Ex:- Export a=10;

In csh

Syn:- Set env Variable_name = values

Ex:- Set env x=10;

→ echo $SHELL :-
is a command to display default shell which is assigned to you

3) kernal :-

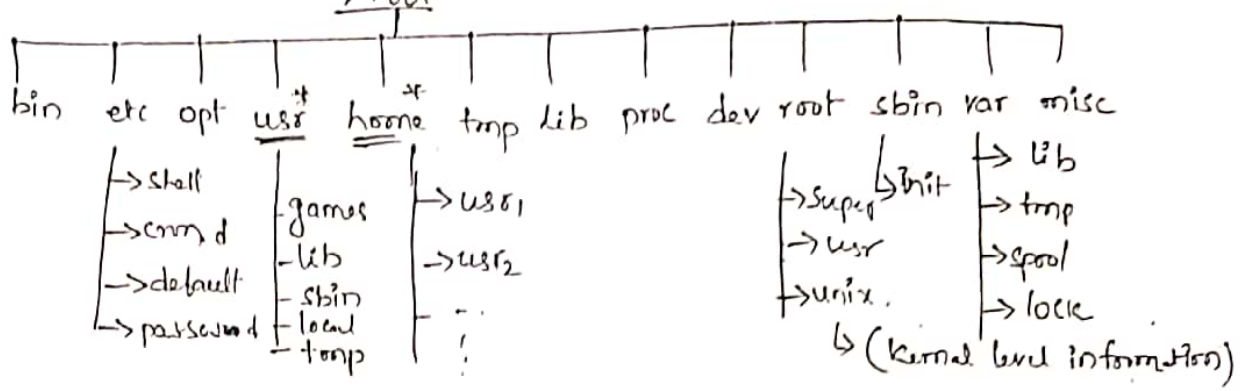
It is a hub of unix operating system. It controls all system resources, memory managment and process management etc.

:- It is a meadeator between shell and Hardware.

:- It performs the validation when user trying to login. If user provides valid credentials it allows to login else it won't allow to login.

4) Hardware :-
This layer contains actual code of unix.

Scanned with CamScanner



unix file system resembles upside down tree.

- '/' is the root file system. starting file system in unix.
- As per unix each and every thing treats as file

-> **/bin :**
It stands for binaries, contains binary or executable file's related to users

-> **/etc :**
It contains binary executable files related to system administrator. It also contains some of important files. like
shell, passwd, shadow, cron.d, etc...

-> **/dev :-**
It stands for device contains : file representation of external devises and sudo devices.

-> **/lib :->**
It contains all library files including C.

-> **/opt :-**
It contains locally installed sfw details.

-> **sbin :-**
stands for system binaries or superuser binaries. It contains fundamental utilities such as booting process, maintenance process and recovery process.

-> **/usr :-**
user file system. It contains some of executable files which are not available in bin along with other details related to users such as games, libraries, local & tmp.

-> prior to /home it maintains all user informa

-> **/home :-** It contains all users information who are exist in system

Scanned with CamScanner

/tmp :-
It contains tempary files.

/var :- define as variable. It contains system log files, spool information & mail information.

/root :-
It is a super user directory which contains some of the important programs related to administrator.

/proc :-

Only directory display :
only files : ls -lrt | grep ^d
: ls -lrt | grep ^-
/var/messages
/var/spool
/etc/ * release : Space grep - Head - tail - Cat

Print :- → sed -n '2,5p' (file name)
~~sed -n '3,8p' filename~~ > space

cat filename → file of [illegible]

Important files :-

/etc/shell.
/etc/passwd
/etc/shadow
/etc/cron.d
/etc/init
/proc/cpuinfo

/etc/shell :- It contains all shell information
/etc/passwd :- It contains all users information which are exist in 6 fields.
/etc/shadow :- It contains password level information of each user [illegible]
/etc/cron.d :- It contains scheduled Job's information
/etc/init :- It contains booting level information
/proc/cpuinfo :- It contain cpu details interms of size,
/var/messages :- It contains messages which are received from others
/var/spool :- it contains the spool level information
/var/log : it contain all system level lock details,
/etc/ * release :- " version of unix os:
L
-> directory → mkdir king1 → cd king1 → cat >> king1 → cd ..
→ files → touch king

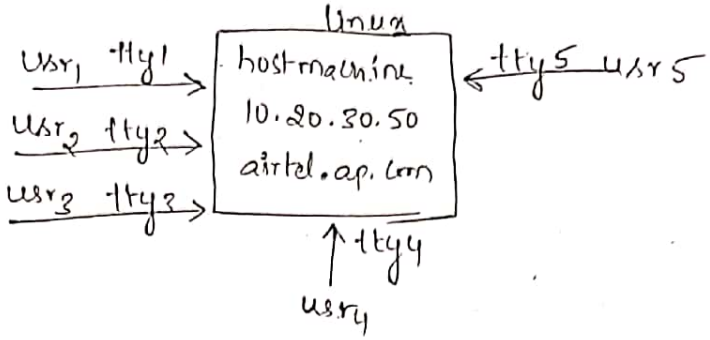
ctrl + D → worked and save

isq.
30 cases

Features of Unix :-

Below are the features of unix OS.

- Multi user capability :-



As part of multiuser capability multiple users can connect to host machine and they can perform their operations without of effecting one and each other.

Whenever we connect host machine from our system by using putty unix establish a path to perform the operations (usually we need to called it as tty (terminal type)).

putty :- (Communication)

It is a tool to connect unix os from our desktops.

Raw	Telnet	Rlogin	SSH	serial
23	23	513	22	9600

Important -

To connect unix os, from our system we required ip address / hostname of system along with valid account details.

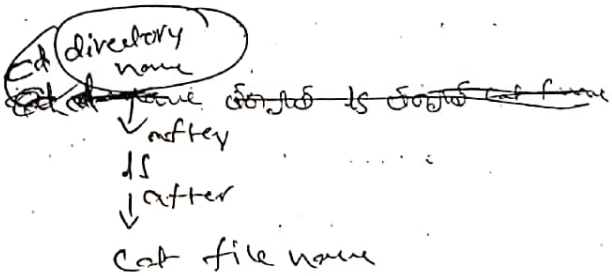
points to remember :-

→ Before connecting to system change the presence of keyboard (which is located under terminal from Ctrl+? to Ctrl+H.

By using above changes we can use navigation keys and backspace, delete buttons.

→ Change the lines of scrollback from 200 to another number(20,000) which located under window icon.

directory open close ~~edit~~ ~~save~~ ~~config~~ ~~ls~~ ~~config~~ ~~cat~~ ~~file~~



Scanned with CamScanner

Multi tasking capability :-

Multiple tasks can be perform in unix one is in foreground rest of them are in background.

- portability :-

We can execute any language program in unix with or without modifications.

Security :-

Unix Provide security in 3 levels

→ Authentication :- while entering it into the system unix validates wether he is valid or not.

→ File level :- ✓ As part of file level security we can set file permissions for groups & others as per the requirement.

✓ So unix prevents to perform the operations if user doesn't have proper previllage.

→ In unix users are categorised in 3 levels.

owners, groups, other

→ permissions are 3 types : . read, write, execute.

→ Encryption & Decryption :-

unix allows us to protect a file with specific key to prevent changes from others.

Communication :-

unix allows us to perform all mode of communications such as mail, chat, broadcast, etc.

Automation :-

We can able to automate predefined tasks and some system important activities as per our requirement.

Disadvantages :-

- unix is a case sensitive, and cud for each and every task we need to execute proper command.
- always we need to pay heigher attention when we are performing an activity
- small mistakes, cause higher consequences.
- we can't able to roll back the files once we delete.

Scanned with CamScanner

Portnumbers :-

- ftp (file transfer protocal): 21
  Ssh (Secure Shell
  Socket
- sftp (Secure file transfer protocal) } 22
  telnet : 23
  Smtp (Simple mail transfer protocal): 25
  DNS (Domain name Server) : 53
  http (hyper text-transfer protocal): 80
  Post office protocal : 110
  NNTP (Network News Transfer protocal): 119
  SNMP (simple network management protocal): 161
  https (hyper text-transfer protocal secure): 443
- Scp: (secure copy) ⇒ If you want ~~file~~ copy file from one server to another server.

ls -lrt — ①
ls -l — ②
ll — ③
} ಎಲ್ಲವೂ

Cat > raju → ಫೈಲನ್ನು ರಚಿಸಿಕೊಳ್ಳಲು
Cat >> raju → ಇರುವ ಫೈಲಿಗೆ ಮತ್ತಷ್ಟು Continues ಆಗಿ ಬರೆಯಲಿಕ್ಕೆ >> ಅನ್ನು ಉಪಯೋಗಿಸುತ್ತೇವೆ 'ctrl-d'

: Unix Commands :

✓ who / w :-

it displays no. of users who are connected to the system

who ↵

Name	line/tty	Time	Comment
Sharuf	TTy1	2016-04-28 10:30	(20:50)
mastan	TTy2	10:50 pm	(10:50)

w :-

the functionality is similar to who but it with provides system statistics along with extra fields such as cpu, from, hostname

w ↵

Name line/tty Time Comment cpu from hostname :-

options :-

-a : it list down active and inactive users

** -b : it provide system time & date of reboot.

-r : it shows run levels of system.

-T : it displays msg status of user. '+' means ready for communication.

-q : it count login names

✓ who am i / :- it displays details about current user

whoami

** ✓ uname :- it displays flavour of unix which is installed

uname ↵

Redhat...

options :-

-a : it displays all information about system such as kernelname, node name, kernel release, kernelversion, mission ...

-s : it displays only kernel.

-r : kernel release

-v : kernel version

-m : mission name -> 32 bit or 64 bit.

hostname :- It displays hostname of system.

options :-

-i : it extracts (prints) IP address of system

Scanned with CamScanner

✓ logname :- It is displays login name

✓ id :- it displays id of user.

id↵

✓ gid :- It displays group id

✓ history / fc -l :- It displays all the commands which are executed by user.

✓ man <Command name> :-

it provide details of command such as definition, options, history, examples etc..

* need to press 'q' to come out from manual entry pag.

ex:- man hostname
man logname
man id

✓ <Command_name> --help :-

it provides important options related to the command.

ex:- who hostname who --help
who pwd —

✓ exit :-

To come out from systems.

④

✓ uptime :- it is used to findout from how long system is upand running.

similar to who -b

✓ last :- It is used to list down the users who are connected to and existed from system.

last↵

File creation Commands:-

✓ touch :- It is used to create one or more empty files. (change files timestamps)

ls -lrt

$ touch file1 file2 file3 ... ↵

touch information.txt student.

consider if file is already exist it will change the timestamp from created date to today date.

Scanned with CamScanner

ex:- student 2016-09-01

Suppose student file is created above Now if issue the touch command it will change file timestamp to current timestamp.

options:-

-a : It will change access time.

-m : to change modify time

-t : allows us to change timestamp of file from todays date to desired date & time.

ex:- touch -t 1604101010 student.
yymmddHHmmss

above.

Operators in unix :-

> :- It is a creator in unix.

And it is a dangerous symbol because if the file is already exist it will override the contents of file.

Ensure that before mentioning filename we need to test it properly.

> filename <followed by path>
optional

>>student

>/hari/log/student.

>> :- (Append operator)

It appends the data to existing file

if file is not exist it will create and inserts the data into the file.

>> student

>>/hari/log/student.

< :- (Input operator)

It is a default operator for every command

It passes content of a file to specify command.

<<- (here document).

it is used to execute some set of commands to a specific command and come out to command prompt.

It should be followed by name.

Scanned with CamScanner

<Command> << !! (name of here document)

!!

Ex:- touch << Ram

hi
this
sharef
Ramr

o/p :-

Hi
this
Sharef

The default working directory in unix is /home/<logname>.
If login name is Ram. the default working directory for Ram is /home/Ram.

*** Ex:- Consider your system size is completely occupied in that case if you want to make it free space. we can able to get by using > symbol followed by file name.

> /home/log.txt.

Unix commands are categorized as internal & external.

Internal commands doesn't required memory to execute below are some of internal commands.

>	who	logname
>>	w	
x	whoami	hostname
<<	who am i	

external commands are required memory to execute.

ex:- cat, touch, rm, ---

✓ Cat:- it is used to perform below operations.

Concatenate Cat Command) 2 or more files and print on the standard output.

i. to displays the contents of file on the screen.

Cat Information.txt

Cat information.txt student.

ii. it is used to create new file and insert the data into that.

cat > class ↵

nanny

)

ctrl+c ctrl+D ctrl+z

Scanned with CamScanner

iii. It allows us to append the data to existing filename -

Cat >> filename . (existing file)

≡ } old data

≡ } new data

Ctrl+C | Ctrl+D | Ctrl+Z

↓ ↓

Without saving with last line -
last line

iv. it allows us to copy the content of one file to new a file

Cat class > sample.

Syn:- Cat <existing filename> > <new filename>

if <new filename> is already is exist it will override

/home/loginname

↓

all files are stored by default.

/

/home/hari

↳ class, information (x)

- Cat class > information

the content of class file, move to information file.

- Cat class information >> out.txt

the content of class & information move to out.txt.

- Cat class information >> out.txt. out.txt

the content of class, information move to out.txt and it will display the content of out.txt on screen if it exist.

Options:-

-n : to assign the line numbers to content.

⑤

Signals In Unix

0 - standard input (keyboard)

1 - standard output (screen)

2 - standard error (screen)

with the help of above signals we can able to capture I/P, O/P specific error to error files.

Scanned with CamScanner

1 > out.txt

2 > error.txt

2>&1 output_error.txt

Ex:- cat file.txt 1 > file.out 2 > /dev/null

ls *log* 1 > file.out 2 > /dev/null

*** cat *file* 2>&1 /dev/null

*** → Important command.

✓ /dev/null :- it is a nullable file so whatever the o/p or error that are redirecting to the file will be deleted automatically.

Special Characters :-

* :- It denotes all, It allows us to match one or more characters at a time

Ex:- cat *.txt (it will displays all files which are having extention of .txt).

? :- It denotes a single character, it is used to match single character it would be anything (digit, character, special character, number).

Ex:- cat Bangalore?.txt

[ ] :- To specify one or more characters at a time.

Ex:- cat [aeiou]*.txt.

It displays content of the file whose starting character should be a,e,i,o,u and 2nd characters is anything.

- :- To specify some range while searching or performing activities.

[aeiou]

[a-d]

[*-+]

[1-3]

[5-7] ending character should be greater than starting character

cat [7-8] [8-9] [0-9] [0-9] [0-9] [0-9] [0-9] . . .

[78] [89] [0123456789] . . .

above commands displays content of file whose first character either 7 or 8 and second is 8 or 9 the remaining 8 are any number

Scanned with CamScanner

! :- used to Search a negitive criteria.

! [aeiou] It should be starting.

^ :- (Cap / Carrot)

the usage of ^ is to specify a starting character while performing activity.

Ex:- ^-

^r

^d

cat ^d.txt.

$:- To specify ending character. (or) it is used to invoke value of Variable.

Ex:- a$ ~~echo~~ a=10 ; name = Hari

b$ echo $a echo $name.

\ :- It suppress the default behaviour of special characters.

Ex:-

cat *1.txt

cat *2.txt

it will display content of file whose name *1.txt instead of displaying the file content which are ending with *.txt

; :- It allows us to ~~display~~ (execute) one or more commands at a time.

Ex:- who ; w ; uname ; hostname ; ls ;

| :- It is used to transfer previous command o/p to next command i/p concurrently.

Ex:- Command1 | Command2...

the o/p of previous command should be valid i/p to the next command.

[' ']

Single quote:-

It is used to display user defined text on screen without any modification. (ctrl+8-9s); a=10; name= shareef;

Ex:- echo 'my name is $name';

my name is $name.

echo.— It is a print function in Unix it is used to display user defined like text on screen:

Ex:- echo 'my name is shareef'

" " :— It allows us to invoke values of variables and other operations also

a=10, name = shareef b= 30

echo " my name is : $name "↵

echo " value of a is : $a "↵

echo " value of a,b is : $a + $b "↵

40

` ` :— It allows us to execute command as part with echo and it allows us to assign the value of any variable of any command output

Ex:- echo " The system date and time is: `date` "

date_time = `date`;

login_name = `logname`;

echo " system details are : `uname` ";

mkdir:— It is used to create one or more directories at a time.

Ex:- mkdir dir_name1 dir_name2 ---↵

mkdir dir_name ↵ → make directories.

mkdir directory name/ sub directory name -

mkdir student class 245.

options:-

-m : to specify permissions.

Ex. mkdir -m 555 student ↵

-p : to create heararchy. file directory

EX :- mkdir -p Countries / India/ Apl kurnool

Scanned with CamScanner

✓ ls :- It is used to display files and directories from current directory or specific directory

Ex:- ls ↵

abc, abc1 abcc basha bash1 basha123 . . .

Options:-

-l :- It is used to get files and directories in long list format in ascending order. If there are any files starting with digits first it will display those files then it will display character files. Below are 9 fields displays with -l option:

1 2 3 4 5 6 7 8 9

file type permission, no. of linked to files, owner, group, size, date/time, file_name

- file
d directory
c character
b boot file
l linked files

→ File Types

= etc..

permissions :- rwx rwx rwx
owner group others

r - read
w - write
x - execute

Ex:- - rw- rw- w- 1 hari hari 20kb 2018-05-08 10:50 info--
d rwx - r - r - 1
l rw- rw- w- 2

-r :- To display the o/p in reverse order.

Ex:- ls -lr

*** -t :- It displays the o/p based on time stamp.

Ex:- ls -lrt

it displays files & directories in long list format in reverse order with respect to timestamp

note:- recently created files in down

-a :- To display all files including with . hidden

Hidden files :-

It contains sensitive information and will not be visible. To create hidden files we need to use . symbol before file / directory name.

Ex:- touch .Contacts.txt

Cat >.out.txt

-i :- To display inode numbers.

Inodes- It is address of file contains characteristics of files or directories. Unix identify uniquely each file or directory by using inode numbers. Inode numbers are unique for each attribute.

-h :- (human readable) :-

It displays file size in kbs, GB, TB ... instead of displaying in KB.

-R :- (Recursive mode)

It displays files or directories in recursive mode.

. → current directory information

.. → parent directory files.

whenever we create a directory in unix by default it creates 2 hidden directories. Contain

1. . : it displays current directory details

2 .. : " parent directory of current directory.

Ex:-

ls -lrt *.log

" " log*

" " *-*

" " [aeiou]*

" " !a*m

" " [7-9] [6-8] [8-9] [0-9] [0-9] [0-9] [0-9] ...

" " * *.txt

" " *Jan * 2006 *

Scanned with CamScanner

✓ cp :- It is used to perform below 2 operations

i) taking backing backup of a file

Ex:- cp <existing filename> <target file>

cp abc abc123.

ii) to copy existing files or directories to another location

(ctrl+c)

/home/master/information.txt

Ex:- cp /home/master/information.txt /Home/master/Shared/out.txt

If you copy the file in current directory then no need to mention full path.

note:- unix will create new inode number for newly created files by using cp

· options:-

-r :- recursively

If you want to perform an operations on directories use -r

Ex: cp -r <directoryname> <new directory name>

-p :- preserve the properties (carries properties of old file to new file)

✓ mv :- It is used to perform below 2 operations

i) Renaming the existing files - move/rename files & directories

Ex:- mv <existing filename> <new filename>

Ex:- mv abc abc2345

ii) moving the files from one location to another location

(ctrl+x)

Ex:- mv <filename> <path of destination>

✓ rm :- By using this we can remove the files or directories from current / specified location

Remove files or directories

Ex:- rm filename/ directory name

Scanned with CamScanner

Ex:- rm information.txt

it will ask user confirmation to delete or not.

Options:-

-i ; user input mode.

-r ; recursively. (mainly use for directories).

Ex:- rm -r <directory name>

-f : (forcibly)

it allows us to delete the file doesn't have previlages. (7)

✓ rmdir :- It is used to remove empty directories.

Syn:- rmdir <directory name>

Ex:- rmdir shareef_dir.

✓ *** grep :-

It define as Globally Search for Regular Expression and Printing.

it is allows us to search one or more than one string from one or more than one file.

Syntax :- grep <option> " string-name " <filename>.

Telephone.txt

name	phonenumber	pincode.
SHAREEF	1234	560103
mastan	1235	560104
mansoor	1236	560105
Shareef	1237	560106
Hussain	1238	560107
Farook	1239	560108
Shareef	1240	560109
Shariff	1241	560110
kareem	1242	560111

Ex:- grep " Shareef " Telephone.txt

o/p:- Shareef 1237 560106

Scanned with CamScanner

options:-

-i :- Ignore case.

It allows us to print the matched lines eventhough if it is small or upper case.

Ex:- grep -i "shareef" Telephone.txt

o/p:-

SHAREEF	1234	560103
shareef	1237	560106
Shareef	1240	560109

-c :- It provides no.of times lines given string repeat.

Ex:- grep -c "shareef" Telephone.txt

o/p:-1

grep -ic "shareef" Telephone.txt

o/p:-3.

-l :- It displays file names which contains a given string from specific or current directory.

Ex:- grep -l "shareef" Telephone.txt

o/p:- Telephone.txt.

-n :- It provides line number followed by row.

Ex:- grep -n "shareef" Telephone.txt

4 : shareef 1237 560106

-v :- verbose mode.

It is used to get non matched search strings.

It displays the o/p which are not matching to given string.

Ex:- grep -v "shareef" Telphone.txt

name phonenumber pincode.

mastan

mansoor

Hussin

farook

Sharif

Kareem

Scanned with CamScanner

-s :- to suppress error msgs

-R :- Recursively search for given string from specified or given / current directory

-w :- to match exact word.

-A :- Consider below is one of the file

It prints no. of specific lines after matching.

Ex:- grep -A +2 "hai" abc.txt

o/p: 1
2
3
4
4
6

abc.txt

hai
1
2
3
4
hai
3
4
4
6

-B :- it prints no. of lines before matched string

Ex:- grep -B +2 "hai" abc.txt

o/p:- 3
4
4
6

✓ egrep / grep -e :-

It allows us to search more than one string at a time

Ex:- egrep -i "Shareef | master | mansoor" Telephone.txt.

✓ fgrep :- file grep or fast grep. (f- file invoker)

it allows us to invoke grep file.

Ex:- fgrep <filename>

✓ Cut :- It is used to extract characters or fields from file.

Syntax:- cut <option> <range> filename.

options:-

-c :- to extract characters from file.

eg:- i) cut -c "1-2" Telephone.txt

ii) cut -c '1,$' Telephone.txt

↳ last character.

cut -c :'1-3, 5-7, 9-10.' Telephone.txt.

Scanned with CamScanner

-d :- delimeter

-f :- field separator. (to extract field)

Ex:- cut -f '3,5' Telephone.txt.

It extracts 3rd and 5th fields from Telephone.txt by default cut consider space as a field delimeter. if you want to extract the fields which are other than space we need to use option '-d'.

cut -d "|" -f '5,7' Telephone.txt.

✓ Sort :-

It is used to sort the data either ascending or descending order by default sort is ascending that is based on character.

Syntax :- Sort <option> <filename>

Ex:- Sort Telephone.txt.

Options:-

-r :- to do sorting on reverse order.

Ex:- Sort -r Telephone.txt

Above command will not make changes on existing file if you want sorted data we need to capture output to store other file. as shown in below.

Ex:- Sort -r Telephone.txt > abc123.txt.

-n :- * numeric sort

It allows us to sort the data on numbers

Ex:- Sort -nr Telephone.txt

-t :- tab delimitre

-k :- to specify key field.

Ex:- Sort -k3 -nr Telephone.txt

Sort -t "|" -k3 -nr Telephone.txt

-o :- It allows us to specify to store sorted data in specific file.

Ex:- Sort -o "out.txt" -k3 -nr Telephone.txt

Scanned with CamScanner

-u : it eliminates duplicate data

ex:- sort -u -nr Telephone.txt

-m :- To merge sorted file o/p to a single file.

ex:- sort -o "out.txt" -m -nr Telephone.txt abc123 abc.

(or)

sort -m -nr Telephone.txt abc123 abc > out.txt.

⇒ Important questions :-

write a syntax to extract only files from current directory?

ls -l | grep '^-' - files

write a syntax to extract only directories from current directory

ls -l | grep '^d' - directories

write a syntax display only hidden files

ls -d .* | grep

ls -la .*

Links :-

Unix allows us to make the relationship between files.

There are 2 types of links in unix.

ln is a command to establish the links b/n files

1) Hard link :-

Syntax : ln <existing filename> <link filename>

ex:- ln information.txt /home/mastan/sensitive/Contacts.txt

Hard link is a exact mirror of source file.

If you make a changes on either source or link file, changes will be reflect in another file.

Inode numbers are same for source and link files. So if you delete a source file link file will be remain and it can be useful.

We can't create hard link over the file system.

Scanned with CamScanner

Soft link / Symbolic link :-

Syntax :- ln -s <existing filename> <Link filename>

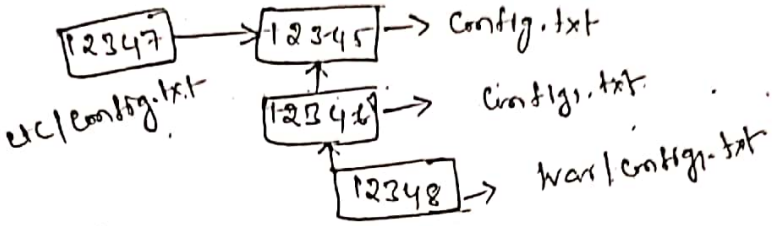
Ex :-

ln -s Config.txt Config1.txt

ln -s Config.txt /etc/Config1.txt

ln -s Config1.txt /var/Config1.txt

Consider 12345 is the inode of Config.txt



Soft link inode numbers are different for source and link files. If you delete a source file target file / link file will become unusable.

We can able to create across the unix.

Soft link

ls -i <filenames> provides inodes of each file.

If you want to get all linkfiles which are associated to that inode number issue below command

ls -i <inodenumber>

uniq :-

It allows us to extract either unique or duplicate values from file.

Syntax :- uniq <filename>

Options :-

-u :- to extract unique values

-d :- to extract duplicate values

simple.txt	o/p -u	o/p -d
1	1	1
1	2	Hi
2	3	
3	2	
2	3	
3	Hi	
Hi	Hello	
Hi	1	
Hello	Hello	
1		
Hello		

Scanned with CamScanner

Unix

note:- In file if lines are in not in order unique command will not work effectively so sort the data by using sort command and give input command as follow

Sort simple.txt.

✓Paste :-

it is used to combine file contents ~~vertically~~ Parallelly.

Syntax :- paste filename, filename2 - - —

Consider below are the file.

abc.txt

1	a
2	b
3	c
4	d

def.txt

5	e
6	f
7	g
8	h

Ex:- paste abc.txt def.txt

1	a	5	e
2	b	6	f
3	c	7	g
4	d	8	h

✓

option :-

-s :- it allows us to paste next file content after completion of first file content (serial)

Ex:- paste -s abc.txt def.txt

1a
2b
3c
4d
5e
6f
7g
8h

X

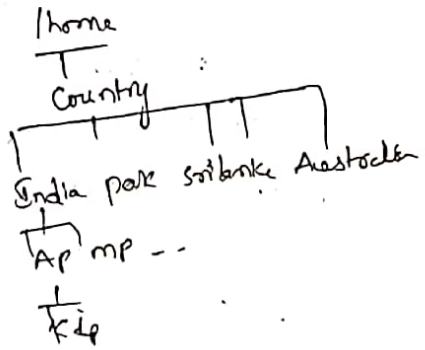
o/p 1 a 2 b 3 c 4 d
5 e 6 f 7 g 8 h ✓

pwd :- It provides path of current working directory from root.

Ex:- pwd↵

/home/mastan

cd /home/country/India/AP/Kadapa



Scanned with CamScanner

✓ cd :- It is used to change the directory

Syntax: cd <path of directory>

cd <directory>

cd country → from master it will moves to country

cd /

✓ cd .. ↵ :- to go back parent directory of current directry

cd ../../.. → it goes to country directry.

✓ cd ↵ :-

It goes to default working directry from current location.

✓ cd - / - :-

It goes to previous working directry.

✓ cd / :- it goes to root directory.

✓ wc :- (word count)

It is used to count no. of lines, charaters and words from file.

Syntax :- wc filename ;

wc abc1

no-of-lines no-of ch

8 12 25 abc.txt

8 :- no-of lines

12 :- no-of words

25 :- no-of character.

by default it provides 4 fields output which are in above.

Options :-

-l :- to count line numbers

-w :- to count words

-c :- to count characters.

Scanned with CamScanner

was to count no.of characters from specific string

```
echo "Bangalore" | wc -c
```

was to count no.of files in current directory.

```
ls -lit | grep '^-' | wc -l
```

was to count how many users are logged in to system

```
who | wc -l
```


tar :—

It is used to create an archive file, it allows us to keep more than one file into new archive file.

Syntax:- tar -<options> <file.tar> file1, file2 - - - - -

Ex:- tar -cvf

Options:—

- -c : create new tar file
- -v : verbose mode
- -f : file name (to specify)
- -t : to display list of the file name.
- -x : to extract files from tar.
- -a : to add new files to an existing tar file (append)
- -r : to remove specific entry or file from tar.

cp -rRL *../. it will copy all files from current to directory

notes—

all options should be associated with vf option.

=> write a syntax to create a tar file called test.tar and insert some existing files into it.

tar -cvf <test.tar> information.txt telephone.txt abc, -

- above syntax will place all files into test.tar.
- extension of tar file is '.tr'.

append :—

-rvf

tar (-arf) test.tar abc1.234 abc1234~

Scanned with CamScanner

write a syntax to list down the file it's contains by test.tar.

- tar -tvf test.tr.

⇒ extract the files:-

→ tar -xvf test.tr

→ Specific file to extract:-

→ tar -xvf <filename> test.tr. (vice versa)

→ remove the tar file :-

→ tar -rvf test.tr.

file:- tar -xvf abc.txt test.tr.
↓ filename.

✓ gzip :-

It is used to gzip existing file/tar file.

It is used to reduce the space of existing file as much as possible.

gzip <file name> / <tar filename>

gzip test.tr ↵

test.tr.gz.

> extention of gzip is '.gz'.

> it doesn't accept more than file and directory

options :-

-c : to copy zip file content to specified file name.

gzip -c information_1 > information.txt

information_1.gz

✓ gunzip :-

It is used to unzip the zip file.

gunzip file.gz.

gunzip test.tr.gz.

Scanned with CamScanner

zcat / zgrep :-

are the commands to see the contents and search the strings in zipped files.

zip :-

It is used to create an archive and zip it.

it can take more than one file as an input but compare to gzip command it will reduce less space.

zip file.tar filename1 filename2 ...
<tar filename> <file names>

Ex :- zip abc.txt information.txt context.txt

abc.txt.zip

information.txt.zip

Context.txt.zip

→ zip *↵

it zips all files in current directory.

unzip :-

It is used to unzip the zipped files.

Syntax :- unzip <zip filename>

Ex :- unzip information.txt.zip

Compress & pack :-

These commands are used to reduce the file sizes compare to compress pack will reduce more space.

extention of compress and pack is '.p'

Syntax :- Compress / pack :

Compress filename

pack filename

Scanned with CamScanner

uncompress / unpack :—

To bring back compress or pack files to original files.

Syntax :- uncompress <filenames>
: unpack <pack filenames>

File viewing Commands :-

Below are the commands will be helpfull

✓ head :— to see content of the files if it is too big.

It is used to display first lines / starting lines from file.

head <n> filename :

Ex :- head -5 information.txt

-n : to specify line numbers.

by default it prints first 10 lines. [head information.txt]

head -15 information.txt.

*** head -1 information.txt ; grep -i "Ram" information.txt

✓ file :-

tail :— It is used to list down last lines from file by default it prints last 10 lines.

Syntax :- tail <-n> filename :

Ex :- tail information.txt -- (10 lines)

first & last :—

** =
*

: head -1 filename ; here tail -1 filename ;

was to print middle lines or specific lines from file.

head -6 filename | tail -3 filename

file.t
1
2
3
4
5
6
7
8

Scanned with CamScanner

options :-

-f :

tail -f filename.

to specify filename. it is used to view the runlevel informati or dynamic information of file.

Ex:- tail -f log.txt.

Pg :-

it is used to view the file content page by page.

Pg filename↵

to goto next page, we need to press 'enter' if you want to comeout in between press 'q'.

more :- it is used to see the file content line by line

more file;]

Ex:- more information.txt.

if you want to go the next line we need to press 'enter'.

to comeout press 'q'. in between,

it shows percentage in left side bottom how much we completed.

it doesn't allows us to go back to previous lines.

less :-

The functionality and syntax is similar to more command. but it allows navigation keys.

(10)

File permissions :-

Permissions of file/ directory will play important role in unix.

In unix each file or directory are having 3 levels of permissions.

1. Owner - u
2. group - g
3. others - o

Scanned with CamScanner

Each file or directory should associated with any of previleges such as

read - r 4

write - w 2

execute - x 1

↓ Symbolic ↓ weightage.

The total weitage of permissions are '7'.

✓ Umask :-

It is used to set the default directory or file permissions.

The value (default) umask is 022. so when you create a file or directory permissions will be derived by using umask value.

Syntax :- umask ↵

o/p : 022

Even user can able to change umask value as needed as shown below

umask 222

umask 000

umask 444.

** default # file permissions in unix is 666

** default directory " " 777 :

When you create file/directory these default permissions will be substract from umask value and sets privileges to file or directory

umask 022

f 666 - 022 644 -rw-r--r

dir 777 - 022 755 rwx-rx-rx

umask 000
mkdir test
777-000 → 777
touch test
666-000 → 666

umask -777
mkdir
777 - 777 → 000
touch
666-777 → 000

umask 233
mkdir
777 - 233 - 544
touch
666 - 233 - 433

Scanned with CamScanner

Range of umask value is — 000 to 777.

chmod :-

It is used to change the permissions of file or directory.

Syntax :- chmod <permissions> <file/directory name>

There are 2 ways to change file permissions

1. absolute mode/ Numeric mode
2. Symbolic mode.

1. Absolute mode :-

In this mode we change file permissions by using weightage.

note :- while changing the privileges in this mode we need to be calculate weightage of existing permissions. If you want to add a new privileges to anyone it should be add to new weightage to existing weightage.

⇒ was to add execute privilege to owner of file - (test.sh)

chmod

:- Consider group and others are having read privileges we need to mention those privileges along with new one.

chmod 744 test.sh

Eg 2: chmod 007 test.sh

above syntax will be reframe by unix as

chmod 007 test.sh

which means we are removing all privileges of owner and group and providing complete privileges for others. To make sure always mention 3 digits when you are changing privileges by using weitages.

⇒ was to provide read and execute privileges for owner,

read privileges - group, execute - others.

chmod 541 test.sh.

↓ ↓ ↓

4 1 1 4 1.

Scanned with CamScanner

2 symbolic mode :-

This mode we need to add the privileges / substract (revoke) privileges by using symbols.

- Owner - u : group - g : others - o

r - read,
w - write
x - execute.

+ is the symbol to add privileges

- is the symbol to revoke existing privileges. Weightage

In this mode we need not to calculate weights of existing privileges

⇒ Wap to add execute - x privileges to owner

- chmod u+x test.sh
- chmod u+x, g-x, o+x test.sh (or)
- chmod ug+x, go+w test.sh
- chmod go-rwx test.sh.

→ chmod 745 test.sh ✓ (00).
chmod u+rwx, g+r, o+rx test.sh ✓

✓ Stikybit :-

It is used to protect a directory from others - or groups.

- if any of directory contains stikybit others and groups can't able to delete directory. Below are the ways to set stikybit.

chmod +t directoryname.
chmod 1777 directoryname
↓
stikybit

/ Chown :-

It is used to change the owner of file.

- Chown username : groupname file/directory name
- Chown username file/directory name
- Chown ram : laxman test.sh
- Chown ram test.sh

Scanned with CamScanner

chgrp :-

It is used to change the group of a file.

- chgrp username : groupname file/directoryname
- chgrp ram : bsmsm test.gsh
- chgrp ram test.sh

wc -l /etc/passwd :-

It will give the no. of users which are existing system. If output is 20. which means there are 20 users and each line contains 7 fields fields are differentiated by

Username : encrypted passwd : id : gid : echos : default working directory : default shell.

Ex:-

shareef : xxx : 10 : 02 : normal user : /home/master : /bin/bash
mansoor : xxx : 11 : 20 : Noorn : /home/mastoor : /etc/bash
basha : xxx : 20 : 20 : Admin : /home/basha/ : /etc/ksh

Passwd :-

- It is used to change the password of user.

Syntax : passwd ↵

enter current password :
enter new password :
Confirm new password :

find :-

It is used to findout the files or directories from current, root or specified directory.

Syntax :- find <path> <options> "filename" -print

After find keyword we must have to specify path as shown below.

. - represents current directory

/ - represents root directory.

/<pathname> - represents full path of file.

We can able to search a file by using all properties

Scanned with CamScanner

such as owner, group, filetype, permissions. . . .

⇒ Search a file based on name and time :-

Syntax- find <path> -type <typename> -name "filename" -print

c - character file

b - blocked file

l - link files

f - normal file / regular file

d - directory file.

Write a syntax to search a file "details.txt" from current, root and /bin directory.

find . -type f -name "details.txt" -print

find / -type f -name "details.txt" -print 2> /dev/null

1> / error.txt

find /bin

Above commands performance is bit slow because they are filtering the data based on output.

Below is another way to improve the performance and search a file details.txt.

find / -name "details.txt" -print 2> /dev/null

1> / error.txt

2>&1 /dev/null.

⇒ write a syntax to search a directory "India" from current, root, and /var directory.

find . -type d -name "India" -print

" / " " " "

" /var " " " " " 2>/dev/null

find . -name "India" -print.

=> Search a file based on **links** :-

(-link)

Syntax:- find <path> <-type f> -links ±number -print

=> Was to find out the file names whose no.of links is more than 5 less than 5 or = 5 :

+ more than the specified number
- less than the specified number.

find / -type f -links +5 -print
-5 -print
5 -print

(or)

find / -links +5 -print
-5 -print
5 -print

find . -type f -links " "

=> Search a file based on **inode** number (-inum) :-

Write a syntax to find out all file names whose are connected to 1234 inode number

find / -type f -inum "1234" -print 2>/dev/null

=> Search a file based on **size** :-.

find <path> -type f -size "±in blocks kB" -print **2>/dev/null**

optional.

find command can't take input in MBs, TBs, GBs, it should be always blocks kBs.

If you want to find out files whose size is more than 3GB we need to convert 3GB into equalent kBs.

find . -type f -size +1000 -print
-1000 -print
1000 -print.

Scanned with CamScanner

Permissions (-perm) :-

find path -type <typename> -permissions (perm) 3 digit number -print

Search the files by using absolute mode :-

=> Was to search a file from root directory whose permissions are 655.

find / -type f -perm 655 -print 2> /dev/null.

find . -type d -perm 777 -print 2> /dev/null

✓ Search a file based on days :-

unix allows us to search the files by using dates / no. of days or time, usually each file contains 3 types of date and time

access
modified
created time.

* * *

=> Was to search all files from current directory who are modified

* >7 days, <7 days, and =7 days

atime - access time	amin -
mtime - modified time	cmin -
ctime - created time.	mmin -

find <path> -type <typename> atime / mtime / ctime ±<number> -print.

=> find . -type f -mtime 7 -print

=> find . type f -mtime -7 -print

=> find . type f -mtime +7 -print.

unix allows us to search the file by using minutes as well below are the options

- amin
- cmin
- mmin

Scanned with CamScanner

=> was to find out the files were created ... 1

find . -type f -mmin 180 -print 2> /dev/null.

Search a file based on owner/group :-

ow - owner
gr - group

=> was to find out the files from root directory whose owner is xxxx

find / -type f -owner "xxxx username" -print

find /bin -type f -grp "grp1" -print.

- find command allows us to use "and, or" boolean operators

Signs:
and - a
or - o

=> was to find out all files whose owner is 'hari' and 'size should be greater than 5000 blocks.

- owner - hari
- size - 5000 blocks

find . -type f \(-owner size+5000 -a -size \).

=> was to find out the files by using below criteria.

group -grp1, & permissions are 777 or size is > 3000 and permissions are 666 and links are 5 & -mtime is 7 days.

find / -type f \(-group "grp1" -a -perm 777 -a -a -link +5) -o (-size +3000 -a -perm 666)\) -print.

Code like find / -type f \((Condition 1) -o (Condition 2)\) -print

Scanned with CamScanner

- exec:

It is used to redirect the input from standard source to other sources (previous command outputs...)

it also passes input to next command concurrently which means in previous command output is 100 lines exec will be iterate 100 times

- Xargs:-

the functionality is similar to exec but instead of redirecting i/p one by one to next command it will construct an array and give input to next command at one time.

> was to find out the files which are created more than 7 days and delete them

find . -type f -mtime +7 -exec rm -rf {} \;

\; it is used to repeat the loop

{} It is used to

find . -type f -mtime +7 Xargs rm -rf {} \;

find / -type f -name "*" -exec grep -i "storef" {} \;
2> /dev/null

find / -type f -name "*" | Xargs grep -in "storef" {} \;
2> /dev/null

delete size '0' files:-

find . -type f -size 0 -exec rm -rf {} \;

Scanned with CamScanner

⇒ File Comparison Commands :-

Below are the commands to compare two files content line by line and displays the differences.

Below are the pre-requisites.

files contents should be in order ascending or descending.

✓ Cmp :- (Compare) : two files byte by byte)

It is used to compare the differences b/n two files and displays differences in bytes.

Syntax:- Cmp file1 file2

By default it displays first differences then comeout. if you want to all differences we need to use an option "l".

-l :

Cmp -l file1 file2

Cmp shows the less volume data file as difference. Suppose no of lines are for file1, file2 are 10, 20 respectively o/p is file1

✓ diff :- (differences) :

It displays differences b/n two files. ~~differences~~ Compare two files line by line

< : represents diff in file1

> : represents diff in file2.

Syntax:- diff file1 file2

✓ Comm :- [Compare two sorted files line by line].

It Compares two files and displays common records between them. By default it displays 3 fields output.

field 1 contains records which are not in file2

field 2 contains records which are not in file1.

field 3 contains records common records b/n two files.

N1	N2
1	1
2	2
3	5
4	6

Syntax:- Comm file1 file2.

Ex:- Comm N1, N2.

o/p:-

field 1	field 2	field 3
3	5	1
4	6	2.

Scanned with CamScanner

options:-

-1 : to avoid field1
-2 : to avoid field2
-3 : to avoid field3.

Comm -12 file1 file2.

✓ Split:-

It is used to split the file data into more than one file as per the requirements.

Syntax:- split filename.

split -n <filename>.

Consider abc.txt is one of the file contains 12,000 lines, with the help of split command we can create 4 files and we can put 3000 lines in each file.

Ex:- split -3000 abc.txt.

✓ csplit:-

It is used to split the files based on character.

Syntax:- csplit <word/character> <filename>

Ex:- csplit "shareef" abc.txt

✓ tr:- (-s, -d, -c).

It is an utility for translating, deleting or squeezing repeated characters.

Syntax:- tr <options> set1 set2 < filename.

The characters in set1 will be replace with characters in string2 with character by character, where as first character in string1 replaced in string2 first character.

Q> Was to convert lower case to upper case or UC to LC.

Ans:- Information.txt

tr '[A-Z]' '[a-z]' < information.txt

tr '[:upper:]' '[:lowercase]' < information.

Scanned with CamScanner

Q) was to translate { } into [ ].

A: tr '{}' '[]' < information.txt

tr ' ' ',' < information.txt

Q) was to convert squeeze repetition of characters.

Ans- echo "we are learning unix" | tr -s '[:space:]' ' '

_ Options:

-s

-

Q) was to delete either characters, numbers or unprintable characters.

Ans- echo "1234 bangalore 23456" | tr -d '[:digit:]'

-c : Complementing.

Q) was to delete characters:

Ans- echo "1234 bangalore 123456" | tr -cd '[:digit:]'

** Q) was to delete non printable characters.

tr -cd '[:print:]' < filename

it is used to delete non printable characters from a file.

Q) was to merge all lines into single line.

tr -s '\n' ' ' < filename

BAngaloRE

Q) was to convert characters from UC to LC and LC to UC in single instance.

tr '[:lower:]' '[:upper]' | tr '[:upper]' '[:lower]';

echo "Room!

It works like this also

Cat abc | tr '[A-Z]' '[A-Z]'

"[a-z]" "[A-Z]"

A-Z A-Z

[a-z] [A-Z] —

echo "hi THIS is sheret SM" |

tr '[:lower:]' '[:upper]' '[:upper:]' '[:lower]'

tr '[:lowr:][:upper:]' '[:upper:][:lower:]'

Scanned with CamScanner

tee :- (-a,

tee name is derived from T splitter in plumbing which splits water into two directions, similarly tee copies the data from standard i/p to each file and also to standard o/p.

(or)

It is used displays the previous command o/p on screen and also we can transfer same o/p to multiple files as an i/p.

Ex:- ls -lrt | wc -l

it gives the count of entries in given current directory but entries will not display on screen. with help of below syn we can able to displays entries on screen and we can count entries.

Ex:- ls -lrt | tee | wc -l

ls -lrt | tee filename | wc -l

ls -lrt | tee f1 f2 f3 .. fn

The o/p ls -lrt will can store in above specified files.

options:-

-a :- append

it allows us to append the data to existing files instead of overwriting.

Ex:- ls -lrt | tee -a out.txt

Ex:- ls -lrt | tee file.txt | grep -i "jan2016" |

tee out.txt | wc -l

✓ cal :

it is used to display current 3 months o/p.

o/p would be previous, current, next months.

cal

cal 12

Cal 12
it displays december month details of current year.

Cal 2015
it displays details of 2015 year.

Cal -10 2016 : it displays 10 months of 2016

Options :-

-j : to get julian days.

-s : to it allows to make sunday as first day of week.

date :- (Tue May 24 12:36:28 IST 2016)
it is used to display system date and time.

$ date↵
2016-05-23 09:00:00 pm IST

Options :-

+ : it allows us to extract specific one or more fields from date command.

%a : it extracts abbrivated week day name

%A : it extracts full week day name

%b : it extracts abbrivated month name.

%B : it extracts full month name.

%c : it displays date and time. (Tue 24 may 2016 12:41:20 PM IST)

%d : " day of the month.

%m : " month

%y : " year.

%M : " minitues

%H : " Hours

%S : " Seconds

%p : " Meridians Am/Pm

\n : new line operator.

Ex:- date +%A↵
monday.

'C' Century
date +%c
2016-05-23 09:01:00

date :

date "+Date : %d-%m-%y %n time : %H : %M : %S"

Scanned with CamScanner

Ex:- date - month - years

date +%d - %m - %y ↵

23 - 05 - 2016

Ex:- 23-05-2016 Time: 09:01:10

date "+ %d -%m - %y Time: %H : %m : %s"

date + "%D" 06/16/16 (L5)

%x 06/16/2016

⇒ was to display today's date (yyyy-mm-dd) to assign today-date variable and print it.

date "+ %y - %m - %d"

today_date = `date "+%y - %m - %d"`

echo "$ today-date.

was to display below format

today date is : 2016-05-23 and

time is : 09:04:10 ↵

echo "today date is : `date +%y - %m - %d` and time is : `time +%H:%m:%s` " ↵

✓ bc : (best Calculator)

; it define as basic/best calculator. it allows us to perform mathematical calculations

+, -, x, ÷

it can take i/p from echo or command prompt.

Syn:- bc ↵.

20+10

20-10

20 ÷ 10

20 x 10 ctrl+C.

30

10

2

200

Scanned with CamScanner

Editors in Unix

There are different kind of editors available in unix.

1. Ed - Ken Thomson & Dennis Richey.
2. Ex
3. Vi } Bill Joy.
4. Vim
5. Emacs

Ed :-

It is a line editor it allows us to perform changes on existing file. but normal users can't use this because it is complex editor to work

Ex :-

It is a super set of ed. editor the functionality & behaviour similar to ed.

Vi :-

{Visual Interface} (VI)

Developed by Bill Joy 1976 at university of California. in berkly. It is used to create new files or allows us to edit the existing file Content. as needed.

It is having 3 modes.

Command : (default)

Insert

Ex:- Command modes.

→ **Command :-**

This mode all keys are interrupt with their editors which means in this mode if you press a key word "l" cursor will move to one character to right from current position. similarly there are some keys which will invoke with their default behaviour.

The main usage of this mode is to navigate cursor.

'Esc' is the keyword. to bring back command mode if we are in other modes.

→ **Insert :-**

It allows us to insert new data in existing files or new files. (i is the key to convert insert mode).

Scanned with CamScanner

→ Ex- Command :-

In this mode we can able to save the file, replace existing strings with new strings and we can copy specific data from one open file to another file.

Esc : is the key to come out from current mode to Command mode.

✓ options in Command mode :-

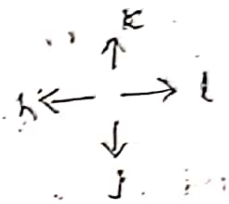
Syntax :- vi existing filename / new file name.

h : moves the cursor one character left

l : " " " " right

j : " " one line down

k : " " " " up



w : moves the cursor right first character of next word.

b : moves the cursor back to end of previous word.

e : moves the cursor to end of current word.

0 : (zero) : moves the cursor beginning of Current line.

$: moves the cursor end of current line.

H : to move the cursor first line of the screen in current page

M : middle of the screen

L : Last line of the screen.

G : to move the cursor last line of beginning of word.

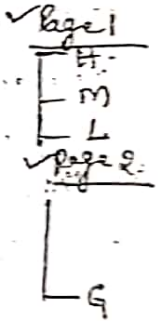
nG : It goes to specified line. (10G : 10th line)

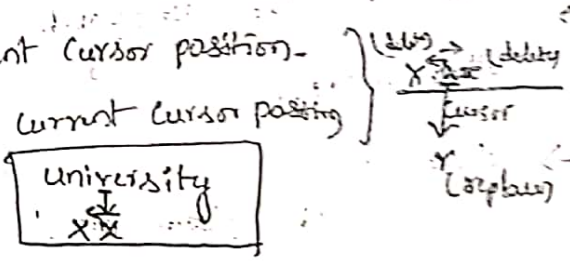
x : It deletes the character at current cursor position.

X : It deletes the character left of current cursor passing (back space)

dd : delete the current line.

gg :- first word of the file.





Scanned with CamScanner

ndd : to delete n lines from current position to down ↓

ex:- 5dd, 10dd, 20dd

dw : to delete current word.

d0 (zero) : to delete line from cursor position to starting line. ←

d$: to delete line from cursor current position to end line. →

. : it perform to repeat previous operation.

u : to undo last changes (ctrl+z).

U : restore all changes which are performed. It bring back files to starting open position

r : it allows us to replace current cursor position to new character. (char)

R : it allows us to replace the characters from cursor position to end of line.

g : it gives the line numbers

J : it g joins current line with below line (5J, 10J, 6J,).

nJ : to join n lines.

i : enter the text i/p mode, at current cursor position. [before Cursor]

I : it allows us enter the text at beginning of the line. [before line of cursor]

a : enter the text i/p mode and allows insert data after the cursor.

A : it allows to insert data after end of the current line.

o : enter the text i/p by opening with new line immediatly below.

O : enter " " " above.

~ : to convert the character from upper to lower from current cursor position

Esc : to go back to command mode from command / Ex-Command mode

ctrl+f : move the cursor next page starting line.

ctrl+b : move " " previous " " "

/pattern: to search patterns in forward direction, from current cursor position

?pattern: " " " " backword " "

*n : to repeat last search command / go back next occurrand in forward direction

N : opasite to 'n'.

Scanned with CamScanner

options in Ex-Command mode:—

Esc : :- is the keyword to convert from current mode to Ex-command mode.

Esc:w :- to save the changes

Esc:q :- to quit from vi editor

Esc:wq :- to save & quit from vi editor.

Esc:zz :- It will save the buffer data.

Esc:w q:- to Comeout from vi editor without saving changes.

Esc:w! :- over ride the existing file with the content of buffer and quit from editor.

Esc: 1,$ s/str1/str2/g

- 1,$ ↓ range (1,20) (1,5)
- s ↓ substitute
- str1 ↓ existing string
- str2 ↓ Newone (replace)
- g ↓ global

Hi this is sharief from Bangalore
Bangalore having lot of job opertunity
Banglore is beautiful city
Bangalore is a hub of IT
Bangalore
Bangalore

s/Bangalore/Bengalore/ - Current line : first occurrence

s/11/11/g - all occurrence line

%s/11/11/g - all occurances of document

184,209s/11/11/g - range

Esc : s/Bangalore/Bengalore -> it replace one first occurance of Bangalore with Bengalore

Esc : s/Bangalore/Bengalore/g : It replace Bengalore with Bengalore all occurances of current line

Esc : 1,$ s/Bangalore/Bengalore/3g -> it replaces all lines Bangalore with Bengalore only third occurance

Esc: 10,20 s/Bangalore/Bangalore/g :- it replace all occurances from Beng -- to Beng from 10 - 20 lines

Esc: 1,$ s/Bangalore/Bengalore/g : it replace Bangalore to Beng!

Scanned with CamScanner

Y/yy - Copy current line.

yw - Copy current word where cursor is located

p - to paste line below the cursor

P - to paste copied line above the cursor.

Ex-Command:

Esc: m,n co p : It copies lines from mth position to nth position after line numb

Ex:- -5,10 co 50 (Ctrl+c ; Ctrl+V)
- 5 co 50

Esc: m,n mo p/: It moves lines number from mth position to nth position
m, mo p after p. (Ctrl+x, Ctrl+V)

Esc: m,n w. <file name> :

It writes/copies lines from mth position to nth position to specified file name.

Ex:- 10, 20 w Ex_file.txt.

Esc: set nu :-

It assigns line numbers for each line

Esc: set ic :-

It allows us to search a string in ignore case.

Esc: set nu!

Esc: set no :-

It undo the roll numbers.

Esc: ! <Command> :-

It allows use to execute commands in vi editor.

Ex:- Esc: ! pwd / ls / uname / who / hostname / ...

vi +<number> <existing file name> :-

It allows us to place the cursor specified line while opening.

Ex: vi +30 abcd.txt.

Space.

Scanned with CamScanner

With help of vi editor we can able to open more than one file. as shown in below.

Ex: vi file1 file2 .....

Below are the options to navigate between the files.

Esc:n :- to goto next file.

Esc:rew :- to go back previous file.

Esc:bn :- it goes to the next file without saving content of current file

Esc:brew :- it goes to the previous file

Esc:f :- It displays file name which is currently working.

Esc:args :- [filename exist lines -:%:]

It displays numbers how many files we have opened.

Disk related Commands :-

df :- [filesystem 1k-blocks used Available use% mounton]

It is used to get disk space statistics in terms of file systems. with help of this command we can find out | analyse how much memory allocated for each file system along with utilization.

Consider it system is having 10 file systems it displays 10 lines o/p. and each line contains 6 fields as following.

Syntax:- df

file-system	allocated space	used space	available	utilization	mounted on.
/	50000	20000	30000	40%	
/bin	300	150	150	50%	/bin

Options :-

-k : to display o/p in kilobytes (KB).

-h : " " " human readable.

-a : all information.

-i : to get inode numbers [filesystem Inodes Iused IFree IUse% mount

Scanned with CamScanner

✓ du (disk usage) :—

- It is used to get the disk usage of each attribute inside the file system.
- with help of du command we can analyse how much space it's used each file or directory, inside the file system.

Syntax:- du↵

It provides 2 fields o/p:

Used space	name of file/directory.
200	abc.txt

options:-

-h : human readable
-k : kilobytes (KB)
-a : all details.

Current directory

du↵

76	./basha_dir
4	./newone
36	./pasha_extract
492	.

Q:- how to get top 10 utilized (disk) files from current file system-

Ans:- du | sort -nr | head -10

✓ dfspace :—

It provides how much space is free for each file system.

Syntax:- dfspace↵

file system name	file space.

Memory related and networking commands :— (top, vmstack, free;)

✓ free :—

It is used to get how much memory is free in system.

Syntax:- free -m ↵

	total	used	free	shared	buffer	cached
mem:	1031204	982652	48552	0	108580	659144
-/+ buffers/cache:		214928	816276			
swap:	2064376	1300	2063076			

cat /proc/cpuinfo :—

this is contain cpu information of system.

*** ✓ top :—

It is used to get cpu information dynamically. functionality is similar to task manager in windows.

Syntax:- top↵

Scanned with CamScanner

The first few lines of top command o/p contains details of memory including swap.

below are the important fields of top command.

pid/ Processid:	User Root	Pr priority	NI	VIRT	RES	SHR	SS	%cpu	%mem	time	Com
1234		20	0	1330	1230	11	S	0.3	0.1	0:0.10	

NI: Nice value

VIRT = Virtual memory

Res : Resident memory size.

SHR : Shared memory size.

SS : Process status

%cpu : cpu utilization process

%mem : memory utilization of process

time : starting time of process

Command : Command / program name

Ctrl+C -> Come out.

lsof :-

It displays list of the files which are using by running program/process

Syntax :- lsof ↵ (list of file).

(16)

✓ Vmstat :-

It define as virtual memory statistics. It provides details about process, memory, paging, i/p o/p statistics, disk details & cpu information.

It is an alternative command of 'top'

Syntax :- vmstat ↵

Options :-

-d :
It reports only disk statistics.

-a :
it reports active and inactive memory.

-m :
It reports cpu usage information.

-p : paging information.

-n - to specified intervals.

Scanned with CamScanner

netstat :- (network statistics) -t like ports which are using by processes, active ports and ports. --

** it displays network statistics.

Syntax:- netstat ↵ netstat -atp

options:-

-a : all n/w statistics including active and inactive.

-t : active ports. -n :- numeric headers ports,

-p : displays PID processID

-l : display display network servers.

netstat -plant | grep -i PID ↵
netstat -atp | grep -i process name ↵

✓ Iostat :- it displays Iostatistics along with CPU usage information.

Syntax:- Iostat ↵

✓ nslookup :-

it provides hostname and ipaddress of URL's.

Syntax:- nslookup www.google.com

✓ ping :-

It is used to validate weather particular server is up and running or not. If server is up and running it provides details of server like portnumber, ip address. --- If it is not working you will get a error msg like connection timedout.

Syntax:- ping 10.20.30.555 ↵

ping airtel.ap.com

✓ hostname :-

it provides hostname of system.

options:-

-i : provides ipaddress of system.

Syntax:- hostname -i

✓ ifconfig :- It provides ipaddress, ipv4 address, hostname, domain name :- of system

Syntax:- ifconfig ↵

lpa :- To get system along with router ports.

syntx:- lpa ↵

Telnet :- It used connect remote server would cannot specific port active or not connecting server.

Syntax:- lpaddr/hostname : port no ↵

Scanned with CamScanner

File Transfer Protocal commands:-

In unix we can able to transfer the file b/n two servers by using below commands.

1. FTP (File transfer protocal)
2. SFTP (Secure file transfer protocal)
3. SCP (Secure copy Paste)

we can able to transfer the file with or without human interaction by using above commands. If you want to transfer the files without using password/human interaction we need to generate authentication keys b/n two servers by using below steps.

1) ssh-keygen -t rsa/dsa :

-t : type of authentication
ssh: Secure Socket host

It is a command to generate authentication keys. above command will ask us enter the passpair

→ enter passphare

as a best praitice don't enter any passphare press enter

→ enter passphare.

after successs full completion of command it will generate below two files at ".ssh" directory. In default working location of user. if directory is not there it will create.

rsa: remote rsh secure access
↳ It is preferable.

- id-rsa.pub :-

it is a public key contains authentication details of system.

- id-rsa :-

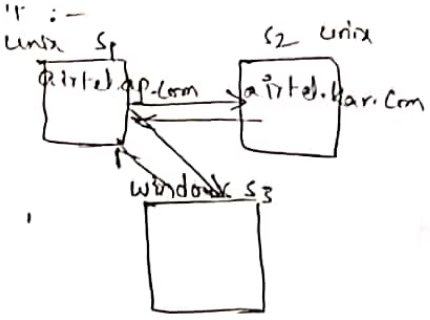
It is a private file contains details of system. normal user can't understand content of file.

we need to place the content of id_rsa.pub into another server in authorised_keys file by using

cat >> authorised_keys
==
ctrl+c
(or)
vi authorised_key (this file is available in .ssh folder.

Scanned with CamScanner

ftp :-



Intranet

It is used to transfer the file b/n unix → windows, windows → unix and unix → unix servers. If you want to transfer the data b/n or unix → windows we need to connect unix server from command prompt by using ftp.

Syntax :- $ ftp hostname | ipaddress.

enter the username: } once credintials are validated it will
enter the password : } goto ftp prompt.

ftp> _ _

bye → it is used come out from ftp prompt.

put :-
get it is used to transfer the file from local server to remote server.

get :-
put it is used to transfer the file from remote server to local server.

mget | mput :- to transfer multiple files or relieve the multiple files

prompt :-
It is used to enable or delable user interaction while transfering files

Syntax :- prompt n → without user interaction

prompt y : - with user interaction

binary :- to convert binary mode.

passive :- to convert passive mode

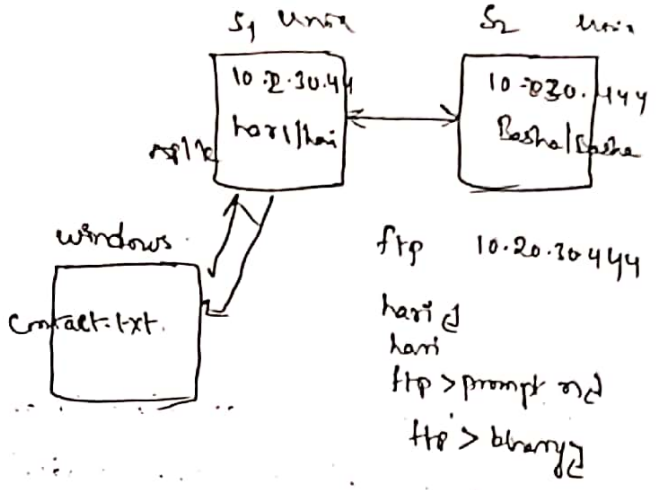
!<command> :- it allows to execute commands in local server.

lcd

! pwd

! --

Scanned with CamScanner



Command syntax:-

put < path > [filename [path]
mput < path to path > [pwd]
put Information.txt
put Information.txt d:/ram/
mput /home/ram/ece India d:/ram/country
get d:/ram/telephone.txt
mget d:/ram/A* /home/ram/India

Write a syntax to transfer file from unix to windows.

ftp > get home/hari/information.txt "/d/files"
ftp > mget /home/hari/rpl* "/d/files/"

Was to transfer files from win to unix

files : ftp > put [redacted] "/d/files/information.txt
ftp > mput "/d/files/India/* /home/hari/

To transfer the files between windows/unix and vice versa we have an external tools called "winscp, Coreftp, these are the gui tools, we can drag and drop files.

SFTP :-

It define as secure file transfer protocol functionality and options/ syntax is similar to ftp but below are the differences

FTP	SFTP
— Over the n/w it provides less security while transfering the files	— It provides more security while transfering files
— port number : 21	— port number : 22
— Manual interaction should be required to use	— If you generate authentication keys we need not to enter the password even we can automate task.

Scanned with CamScanner

Sqlplus :-

it is used to connect db from unix server.

below are prerequisites to do before connecting.

i) we need to verify wether oracle client is installed or not in system. by using below command.

echo $ORACLE_HOME↵

if you are not getting any o/p we need to be install.

ii) under ORACLE home directory there is a file called "tnsnames.ora" which contains details of db (tnsnames).

iii) if there is no tns entries which you want to connect we need to add.

Note:- to get the details of db contact db administrators / teammates.

below is the command to connect db.

Syntax: sqlplus username@/password@dbname.

options :-

=> -s : to connecting silent mode

=> -/nolog :- it convert unix prompt to sql prompt. but it won't connect to db. so it

=> -Conn : to connect db. sql prompt

Ex:- -> sqlplus scott/tiger@orcl.

SQL> ==

exit

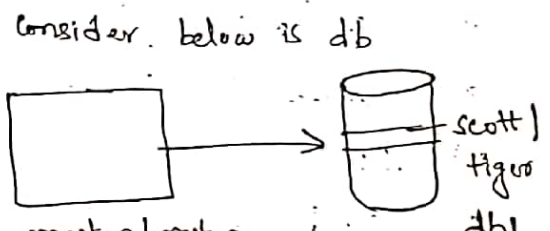
-> sqlplus -s scott/tigers@orcl

==

exit

-> sqlplus /nolog

->Sql> Conn username/pass



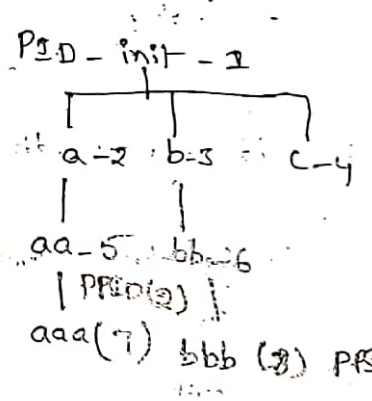
Scanned with CamScanner

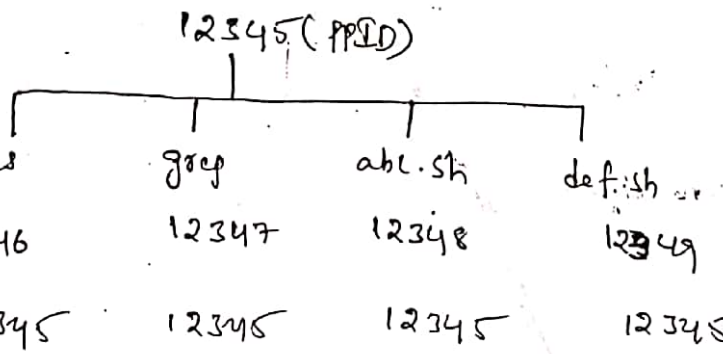
Process & Associated Commands:-

whenever we execute a program in unix or windows os will establish special settings to that program (like needed memory, priority, --)

- A process def: "running instance of program." Each process contains pid, PPID (Parent-Process_ID).
- PID is a unique identification of each process and p
- PPID is a parent process ID of program. which means which program is the help to execute your programs, for normal users shell process ID will become as PPID of all the programs which are executed by user.
- shell is the first program will execute by system whenever particular user logins to system.
- "init" is the first process in unix it will be start by os whenever we boot the system. the PID of init is '1'
- whenever a user executes any of external commands or programs that will add log into process table. Below are the different types of process statuses.

i) submit
ii) running / executing
iii) completed
iv) success
v) fail
vi) hold
vii) Terminated
viii) zombie
ix) orphan





Scanned with CamScanner

SCP :-

It define as secure copy paste.

✓ It is used to transfer the ~~files~~ files / directory b/w two unix servers.

✓ If authentication keys are established we need not to use the password.

below are the syntaxes.

- scp <options> filename username@servername : /path → sending file
- scp <options> username@servername : /path/filename path → Receive file.

: Is the differentiator b/w authentications and filelocation. It is a mandatory

options :-

-R : transfers the directories

scp information.txt Basha@Airtel.KA.com : /home/basha/files/ } send

scp -R /home/hari/AP/ " " }

scp Basha@Airtel.KA.com : /home/basha/files/details.txt . } receive.

scp " : /home/basha/karnataka /home/hari/AP/ }

Inorder to use scp command make sure we have to connect any of the server by using putty assume if you connect to Airtel.Ap.com above are the syntaxes to transfers files to airtel.KA or receive the files

SSh :-

It is used to connect another unix machines from current server remotively. one

one our activities are complete we can use exit command connect from from remote server.

Syntax :- ssh username@servername ↵

——

——

——

exit.

option :-

-p : to specify port number to connect the server.

ssh -P ~~ka~~ 1234 username@servername

Scanned with CamScanner

telnet:-

functionality similar to ssh.

Syntax:- telnet servername/ipaddress

enter username:

enter password:

exit

option:-

-p: to specify specific port number.

telnet -p1234 servername/ipaddress

Scanned with CamScanner

Earlier PID of that ...

- once a process ID is completed that id can be used for another process but there will not two process contain same PID at a time.
- **PS** is a command to list-down process details which are running on system.

Syntax:- ps ↵

PID	TTY	time	Command
12345	pts/1	00:000	sh/bash/...
12346	pts/1	00:10	ps

- above two are default processes. which will always display when we execute "ps" command.

- options :-
  - -e : each and every process related to ppp all users.
  - -a : including inactive & active processes
  - -f : full path
  - -F : more details compare to 'f'.
  - -u : to specify specific user.

Ex:- ps -e | wc -l

ps -a | wc -l

Syntax:- ps -u <username>

Ex:- ps -u shruti.

ps ↵ → It displays only current process details by default.

Syntax:- ps -f /* <fullpath> */

UID	PID	PPID	CPU	TTY	Time	stat	Command.

Note:- Suppose your linux version (flavour) is BSD type.

Instead of ps -f we enter below are usefull

→ ps -aux

→ ps -ux

To get about threads :

Ps -eLf

Ps -e axms

Pstree : It is used to get process tree details on particular process

Syntax:- ptree PID (in 5 digit number) [from reference]

Ps -ejH

ajx f

Scanned with CamScanner

⇒ was to get the associated process of abc.sh program

ps -ef | grep -i abc.sh | grep -v "grep"

↓

it will displays two pids
i) one is abc.sh
ii) another is grep cmnd id

→ Here it will shows abc.sh detail suppose pid is 12345.

ptree 12345

daemon process :-

These processes we called it as system processes which can't be killed by normal users or administrators.

- These processes will starts whenever system is start and they will terminate whenever we shutdown the system.
- Some of deamon processes are
  - init
  - v-hand
  - scheduler
  - Bd-flash.

✓ init :-

which it is a first program which is created by system by this we can change system runlevels.

init 0 ↵ :- to shutdown the system
init 1 ↵ :- " restart " in single user mode
init 2 ↵ :- " " " multi "
with resource shade.
init 3 ↵ :- to restrat the system in multi user mode without resource shade.
init 6 ↵ :- to halt and reboot

Scanned with CamScanner

pwdx :-

It is used to get the location of program.

Syntax:- pwdx PID (5 digit number)

Multi Tasking Capability

Unix allows us to execute one program in foreground more than one in background which means we can also perform multiple process activites at a time.

& : is an utility to execute a process in background

Ex:- i) • ls -lrt &

ii) grep -i "shoeuf" abc.txt &

iii) sort -k 1, 2 abc.txt -o "out.txt &

whenever we execute a program in backend immediatly it will show process id of process(background)

disadvantages :-

> The output of background process should be redirected to any of the file by using > symbol. as shown in below.

ls -lrt & > file.out.

if you won't redirect the o/p to another file the o/p of background process will be overwritten foreground o/p.

> due to some reasons your session got terminated all the processes which are running in background will be aborded intermidiately

> if too many processes are running in background it will degrade the performance of system.

above disadvantages can be overcome by using "nohup" command.

Scanned with CamScanner

✓ nohup :-

It is used to schedule or execute a process in background.

Syntax:- nohup ls -lrt &

nohup <scriptname/cmd name> &

Ex:- nohup ./abc.txt &

nohup sort -nr file1 ; ; wc -l &

nohup ps -eaf &

- We need not redirect the o/p to some other file because o/p of program will be insert/add into "nohup.out" file which would be existing default working directory. If the file is not exist it will create. else it will append the data to "nohup.out" file.
- eventhough your session got terminated due to some x issues processes would be completed, because those will be attached to init process ID which is '1'.
- nohup cmd can prioritized the processes which we executed. so it won't reduce the system performance.

✓ Jobs :-

It is used to extract the process details which are running in background.

Syntax:- jobs

o/p: PID CMD Name.

✓ fg :-

It is used to execute a process in foreground which are running in background.

Syn:- fg <PID>

Ex:- fg 12345

✓ bg :-

It is used execute a process in background which are running in foreground.

Syn: bg <PID>

Ex:- bg 12345

Note: To execute above cmd open a duplicate session by right click on top of screen then find out PID of process by using ps cmd.

Scanned with CamScanner

✓ Zombie :-

I define as undied process or (definate) defunct process.

A process has completed its execution but still entriciare available in process table due to that process (program) will keep on executing. Therefore lot of system wastage of memory. and it will also reduce the performance of system. these kind of processes will never die until user kills the process or restart of syste

below are the ways to findout zombie processes.

i) ps -eaf | grep -i "z"

ii) top | head -3 (or) grep -i "zombie"

* * *

✓ Kill :-

It is used to kill the particular processes.

Syntax:- kill <options/signal> PID.

-9 : Sure kill

-15 : gracefull kill

-0 : kill dependent process of specified process ID.

below are the reasons to kill a the programs.

i) If terminal is hang (session)

ii) A program which is running has gone indefinate loop. or it is running more than expected time.

iii) By mistakenly if we execute any of the programs.

was to kill a process abc.txt.

kill -9 PID of abc.txt.

to kill a process we need to find out PID by using below command.

ps -eaf | grep -i "abc.txt" | grep -v grep

(12345)

kill -9 12345.

Scanned with CamScanner

Scheduling commands:—

Unix allows us to schedule the programs as per our requirement.

- even it is a feature of unix. Below are the commands we have in unix to schedule the programs.

✓ at :—

It is used to schedule a process only one time either now or feature.

Syntax:- at <time> ↵ > at.out.

Command1
Command2
program1
Commandn.
ctrl+c

{ at, Batch, Crontab }

time options:—

- now+10minutes
- now+3hours
- 09:15
- 23:30
- now+ one months
- 09:15 Sunday
- 09:15 January 20 2016

options:—

-l :- to display all entities which are scheduled by using at command.

at -l

-r :- It is used to remove all entries which are scheduled by at command.

-f :- to invoke at file.

⇒ was to schedule some set of commands at 20:35 today.

at 20:35 at.out

ls -lrt > at.out

ps -eaf >> at.out

ps -eaf | wc -l >> at.out

sort -k2 -nr telephone.txt >> at.out ~~abc.txt~~

./abc.sh >> at.out

~~java -jar abc.java >> at.out~~ ⇒ invoke java program in

Batch :-

It is used to schedule a process but it will be execute wherever system resources are free.

- We can't specify specific date and time.

Syntax:- batch
Command1
:
Commandn
Ctrl+C.

Ex:- batch
same as at block
Ctrl+C.

Crontab :-

It is used to schedule a process forever which means it allows us to schedule a job/process as per your requirement. (frequently, specific time ...).

Below are prerequisites :

i) Crontab services should be up and running.
ps -ef | grep -i "crontab"

ii) It should be attached to any of the editor preferable is "vi".

iii) We must have the previleges to make an entries.

Syntax:-	minutes	hours	DM	MY	DW	script/program/cmd name
	1	1	1	1	1	
	0-59	0-23	1-31	1-12	0-6	

Options:-

-e :- (crontab -e)
it allows us to make an new entries or edit existing entries.

-l :-
it listdown all the entries which are scheduled by using crontab.

-r :- (danger)
it removes all entries which are scheduled by using crontab command.

-> Was to schedule a process dec-31-2016 at 10:30 pm. abc.sh

crontab -e
30 22 31 12 * /home/master/shareef_dir/abc.sh.
ESC : wq

o/p: one entry added in crontab.

Note:- whatever the fields we don't know while scheduling by using crontab mention it as '*'.

- program name should be contain exact path from root.

was to schedule a process abc.sh every 30 minutes.

0,30 * * * * /home/mastan/shareef_dir/abc.sh

*/30 * * * * "

[0-59]/2 * * * * "

every 2 hours :

* 0-23/2 * * * "

* 0,2,4,6,8,10,12,14,16,18,20,22, * * * /home/mastan/shareef_dir/abc.sh

* */2 * * * /home/mastan/shareef_dir/abc.sh

was everymonths first day and last day.

00 00 1 * * /home/mastan/abc.sh > /home/mastan/abc.txt

30 05 (date "+%d" -eq cal | head -2 | tail -1 | awk -F " " '{print $NF}')
* * /home/mastan/abc.sh

date -d +1 day "+%d" -eq 1

was to schedule a job on every months 2nd & 4th saturday - 10:30 PM

30 22 7-14,21-28 * 6 /home/mastan/abc.sh

date "+%d" -eq 31

30 10 2,4,6,8,10,12,14,16,18,20,22,24,26,28,30

Note: Most of scenario we don't delete an entries from Crontab we will just comment the line

✓ Sed :-

It define as stream editor. It is an powerful utility to manipulate data and transform text.

- It can take i/p either from file or pipe (|)
- It works as a file processing i/p line by line
- Sed cmd can take i/p at a time one line from file then perform operations on it.
- If file is having 'n' lines sed cmd operations iterates 'n' times.

By using sed cmd we can perform an operations like print specific lines, delete lines, substitute one string with another string, write sed operations o/p to new file etc.,

Syntax:- sed <options> 'operations' filename.

operations must be enclosed with single quotes.

Options:-

-n : to suppress default sed behaviour.

By default sed display the o/p on screen which are effecting by sed operations specifically when we are printing the lines by using p to avoid this behaviour we can use '-n'.

-i :

By default sed operations are temporary if you want changes on file on peremanently we can use /-i.

-f :

to invoke sed file

Syntax:- sed -f filename.sed.

Consider below is the file telephone.txt.

Name	PHNO	Age
rami	1234	25
shareef	1235	24
mastan	1236	21
mansoor	1237	28
shakeer	1238	27
shashe	1239	29
rafeeq	1231	30
mehtab	1232	31

→ display content of file :

sed '3p' filename (telephone.txt)

shareef 1235 24 (default behaviour)

shareef 1235 24 (operations of sed)

Scanned with CamScanner

→ sed -n '3p' telephone.txt — suppress behaviour of default

o/p:- shareef 1234 24

→ sed -n '2,3 p' telephone.txt. — specific lines

shareef 1234 24
mastan 1235 21

→ sed -n '2,5, 6,8 p' telephone.txt — range

shareef — — * * * '2,5 p' — range lines
mastan — —
manoor — —
shashi — —
rafeeq — —

→ sed -n '2~2 p' telephone.txt. — even lines. (~ range).

→ sed -n '1~2 p' telephone.txt — odd number

→ sed -n '1p;$ p' telephone.txt : — 1st and last line.

→ sed -n '1,5 p; 10,15 p ; $p' telephone.txt — 1 to 5, 10 to 15 and last lines

→ sed -n '30, $ p' telephone.txt. — 30th line to end.

→ sed -n '/shareef/p' telephone.txt — to print lines by using text

→ sed -n '/shareef/,/rafeeq/p' telephone.txt — text range.

→ sed -n '/^$/p' telephone.txt — blank lines (print count)

→ sed -n '^,$ p' telephone.txt

→ sed -n '^,$ p' telephone.txt | wc -l

Deleting :- (d):

By using 'd' keyword we can able to delete lines from file.

- whatever the options we learn for 'p' we can use for 'd'.
- replace p with 'd' and 'n' with '-i'.

write (w) :-

Syntax :- It is used to write the content of sed operations o/p to some other files.

*** → was to delete empty lines : sed -i '/^$/d' telephone.txt

sed -i '^,$ d' telephone.txt.

Insert empty lines :- sed -i 'p;$/p' telephone.txt

Count empty lines: n -n " " | wc -l.

Scanned with CamScanner

→ Sed -n '1,4 pw <target file>' telephone.txt - move the lines to another

Substitute :-

Sed <options>

Syntax:- 's/string1/string2/g' filename.

- s → Substitute
- string1 → Existing string
- string2 → Replacing string
- g → global

Sed command allows us to substitute one string with another string without opening the file.

⇒ wai to replace a string manesoor with mabubasha.

→ Sed -i 's/manesoor/mabubasha/g' filename.

⇒ replace 2nd occurance of string :-

Sed -i 's/kareem/karnataka/2g' telephone.txt

⇒ replace * Shareef with Mahaboob on 2nd and 5th occurance.

10-30 lines

Sed -i '10,30 s/Shareef/Mahaboob/2g ; 10,30 s/Shareef/Mahaboob/5g telephone.txt.

⇒ replace basha with bangalore between Shareef to rafeeq lines

Sed -i '/Shareef/rafeeq/ s/basha/bangalore/g' telephone.txt

* to add "hi" keyword begining of line and "hello" end of each line

Sed -i 's/^/hi/g ; s/$/hello/g' telephone.txt

to count no. of lines and move first half lines to one file next half lines to another file.

mid_line_no =. `(( cat filename | wc -l/2 ))`

n-line_no = $(((m_line_no+1)))

Sed '1, $m_line_no w file1.out ; $n_line_no, $ w file2.out' filename

Scanned with CamScanner

Insert new lines : (-i).

By using sed cmd we can insert new lines without opening file.

Syntax:- sed -i 'ni ==
==' filename

→ to insert below lines at 4th line in telephone.txt.

sed -i '4i rajesh 1238 26
Seeta 1239 26' telephone.txt

Next

Awk

It is a powerful utility to manipulate and process files. It is used for data processing, programing and extracting specific fields from file.

— It was developed by Aho, weinberger, keringhon at AT&T bell labaratory. The name is derived from the first letters of 3 developers as Awk.

— Awk can take input as rows or columns of file and we can process on it.

— It has inbuilt functions as similar as other programming languages even it can support boolean operators (and, or, !), conditional statements and iterative conditional stmts.

— By using Awk we can generate reports as per requirement.

Below are the syntaxes of Awk.

Command line:-

awk <options> '<conditions> {operations}' filename

↳ Command line syntax

programing syntax:-

awk

begin
{
== beginingstmt
}
{
== Body
}
end
{
== Endstmt
}

Scanned with CamScanner

options :-

- -F/FS : to specify field separator
  NF : no. of fields
  NR : No. of Records
  ORS : o/p Record separator
  OFS : o/p Field separator

$1...$n → to extract specific fields. $1 -- $nf - last field
print → to print data
printf → format printing.
%d → to align digits either forward/backward
%s → %os string alignment.
++ → auto increment
-- → auto decrement.

awk -option '{ }' filename

Examples :-

Note: awk can take i/p either from file or previous command output.

1) Extract 9th field from ls -lrt :-

ls -lrt | awk -F " " '{print $9}' — 9th field
(space delimiter)
$NF — last field.

2) Extract 1st and last field :

ls -lrt | awk -F " " '{print $1,$NF}' filename

3) Extract last 2 fields :

awk -F " " '{print $(NF-1), $NF}' filename.

4) no. of lines & fields from each line of a document

awk -F " " '{print NR, NF}' filename

5) file size > 300 bytes (ls -l | awk '{if ($5>200) print $0}'
ls -lrt | grep '^-' | awk -F " " '$6>300{print $0}' filename

6) to search and display the lines.

awk -F " " '/shruf/ {print $0}' filename

Scanned with CamScanner

7) To display o/p record separator as "test" and o/p field separator as "!"

```
awk -F " " 'ORS = test, OFS = "!" {print $0}' filename
```

8) Extract the 1st and last fields and allign the data 5 char right.

```
awk -F "!" '{printf "+5%d, +5%s" "$1, $NF}' filename
```

(+5%d, +5%s: allignment; $1, $NF: extract)

Consider below test file

Test.txt

Hi am in banglore, Hi wher r u

Hi what are you doing Hi

Hi Hello Hi Hi

Hi Hi

By using grep -ic "Hi" we can get how many lines the string "Hi" is existed but it can't give how many times exactly the string Hi is existed, because grep can't consider if string exists more than one occurance in same file.

⇒ Below is the syntax we can able to count how many times string will be existed

```
awk -F " "
    begin
        { print " to count no.of times : Hi"
        }
    for (i=0; i<=NF; i++)
        if [$i == "Hi"]
        c++
    end
        { print " no.of occurance of Hi : $c"
        } < file name
```

⇒ was to Extract file from current directory owner is: shareef and group

```
ls -lrt | grep '^-' | awk -F " "
        '[$4 = "shareef" && $5 = "master"]'
```

11) No of lines:

awk ' END END { print NR}' telephone.txt

12) Even numbers:

awk ' { NR % 2 == 0 ' telephone.txt

13) print every line Longer than 40 characters

awk ' length ($0) > 40 ' telephone.txt.

14) allignment:

awk ' { printf " %.5s %.5s\n ", $1, $2} ' telephone.txt

Base Name :-

It is used to get name of filename from given a part.

basename /home/ labcol.txt

o/p: abc.txt

dirname :- It is used to extract path from the given input.

Scanned with CamScanner

Communication :-

Unix allows us to do below type of communications:

1. Chating
2. Sending an emails
3. To send broad cast messages
4. To read news paper

Below are the pre requisites to chat with other users.

The person who wants to communicate by you must be login into the system. He should be ready to chat with.

below are the commands will help us to find out how many users are connected and how user not ready / ready for chating.

⇒ who -T :-

The above command will display who are connected to system if user ready for communication he contains + symbol before terminal type. else they are not ready for communication.

⇒ finger :-

The functionality is similar to who -T command. The users who are ready for communication they don't contain + symbol before terminal.

mesg :-

It is used to enable or disable chating with other users

mesg n (disable for chating)

mesg y (enable for chating).

write :-

It is used to communicate with other user who are connected and ready for communicate.

write

Scanned with CamScanner

✓ wall :-

It is used to send broad cast msgs who are logged in system.

it is an administrator command

wall
--
--
--
ctrl+c.

mail :-

It is used to send an email to other users

syntax:- mail -s <" subject" < to list>

-c <carbon list> -b < block carbon list>

- a <attachment> < filename (or) full path >

-f : from which user you have recieved the msg

> was to send a testmail to abc@gmail.com.

mail -s "Test" abc@gmail.com, bcc@hotmail.com <

> was to send content of body.txt as mail body to abc@gmail.com

mail -s "important information" abc@gmail.com < /home/master/body.txt

(cat /home/master/body.txt) | mail -s "important" abc@gmail.com.

/home/master

- body.txt
- telephone.txt

mail -s "contacts file" abc@gmail.com -a /home/master/telephone.txt
< /home/master/body.txt.

(cat /home/master/body.txt) | mail -s "contacts file"
abc@gmail.com -a /home/master/telephone.txt

Scanned with CamScanner

(cat /home/mastan/body.txt) | mail -s "Telephonefile" -f abc@gmail.com -t abc@gmail.com -b xyz@gmail.com -c xyzabc@gmail.com -a : /home/mastan/Telephone.txt.

✓ mailx :-

It is used to send an interactive msgs the functionality an syntax is similar to mail command.

✓ Sendmail :-

It is used to send an electronic mail.

—x—x—x

env :-

It displays all environmental / system variables.

Syntax:- env↵

Important env variables :-

- PATH
- ORACLE_HOME
- JAVA
- HOME
- TZ
- PS1
- SHELL
- PS2

all env variables are in caps

✓ PATH :

it is a collan (:) delimited directory. whenever we execute a cmd or prgram it checks all directories which are assigned to path.

echo $HOME :-

$: it is used to invoke value of variable.

above syntax provides default working directory.

cho $SHELL :-

it displays default shell which is assigned.

set :-

It is used to get system variables along with user defined variables

Syn. set

Scanned with CamScanner

- basename :-

it is used extract file name if it is associated with path.

syn ← basename /home/bsc mastan/abc.txt txt

- dirname :-

to extract dir name of file if it associated with path.

Syntax:- dirname /home/mastan/abc.txt

Important shell Variables :-

- $$:-

It gives PID of current shell. which is being used.

- $! ← It gives PID of last background process which executed successfully

- $? :

It is used to get the exit status of last executed command if previous cmd was success o/p would be '0' apart from '0' whatever o/p which mean previous wasn't executed successfully.

```
ls -lrt
pwd↵
echo $?
pwdxx↵
echo $?
```

- $0 : It is used to get shell name which is currently using.

```
echo $0  → sh
ksh↵  (it is used to change one shell to another).
echo $0
bash↵
echo $0
```

- $* : to access all positional parameters - (or) Commandline args

$# : It is used to get count of positional parameters which is passed

$@ ← same as $* - but it constructor array and access

Scanned with CamScanner

11) no-of lines:

awk 'END { print NR }' telephone.txt

12) Even numbers:

awk 'NR%2==0' telephone.txt

13) print every line Longer than 40 characters

awk 'length($0) > 40' telephone.txt.

14) allignment:

awk '{ printf "%5s %5s\n", $1, $2 }' telephone.txt

Base Name :-

It is used to get name of filename from given a part.

basename /home/ labcol.txt

o/p: abc.txt

dirname :- It is used to extract path from the given input.

vi first.sh

```
echo "Raju"
echo "Hello the world"
```

esc, shift+:, wq → Enter

echo ఎక్కువసార్లు esc, i enter చేయండి

execute : sh first.sh

~~for~~

```
#!/bin/bash
for table in {2..20..2}
do
  echo "table for 2: $table"
done
```

```
#!/bin/bash
for (( i=10; i>=1; i-- ))
do
  echo "$i"
done.
```

Scanned with CamScanner

SHELL SCRIPTING

It is a collection of commands.

The main advantage of shell scripting is to perform repetitive tasks.

Ex:-

- Customize your work environment.
- automating your daily tasks.
- Perform some important system activities such as shutdown the system format disk, backup of system.. etc.
- perform same operations on many files.

Disadvantages:-

It requires higher degree of attention while writing the script

It is too sensitive.

Small mistakes can cause sivious problems.

The extention of shell script is either .sh/.ksh/.bash by using vi editor we creates shell script as follows.

vi filename.sh

write a shell script to print your name along with system date and time.

```
vi first.sh
#!/bin/ksh
echo "Hi this is shaik, this is my first program"
echo "learn nicely".
echo "Thanks"
echo "system date and time is :'date'"
:wq.
```

#!/bin/ksh :-

It is the first line of shell script it will never executes because it is started with #.

we need to call it as c-bank statements

whenever we execute a program shell will check wather that program contain on cbank statement or not. if it exists your program will execute/invoke by special shell.

Scanned with CamScanner

Scanned with CamScanner

... is not exist your program will execute by using another shell.

- with the help of this line we can develop program according to the shell. like declaring the variables, exporting the values of variables... etc.

⇒ How to debugging of shell script :-

we can debug shell script either in command prompt or program it self.

Command prompt :-

By using set command we can debug shell script.

1st method :-

```
set -vx scriptname
set -v  scriptname
set -x  scriptname.
```

Ex:- set -vx first.sh

-v : To make sure each line will be tested before it is executed.

-x : make sure each command will be tested before it executed.

2nd method :-

By mentioning -vx after 1st line after at shell script

Ex:- ~~#!/bash~~ #!/bin/ksh -vx

In above method program will be debug whenever we mention the -vx along with 1st line.

⇒ execute shell scripting :-

After creation of shell scripting we must have to change or assign execute privileges to owner of the file or others. Because default file permissions are 644.

Below are the commands to add execute privilizes.

```
chmod 744 first.sh
chmod u+x first.sh.
```

Scanned with CamScanner

- Below are the techniques to execute shell scripting.

1) ./<scriptname> :

It shows the program is exist in current directory from root.

* : above technique can work if your in current directory.

2) By specifying exact path :- <path>/<scriptname>

By using this method we can execute program from anywhere

3) By using shellname :- <shellname> <scriptname>

ksh first.sh
bash first.sh
csh first.sh

we can execute the program eventhough if you dont have execute privileges by using this method.

* Consider there is a big program. if you want to debug from specific lines. we can debug as shown in below-

vi abc.txt
!/bin/ksh -vx
-vx → start debugging
+vx → stop debugging.
-vx
+vx
:wq

Next

✓ read :-

It is used to read the runtime variables, these are the variables which we pass values after executing script.

Syntax :- read Var_1 Var_2 - - - Var_n.

vi test-read.sh

Scanned with CamScanner

vi Intrest.sh

```
#!/bin/ksh
echo "enter the principle rate, Time and Intrest rate";
read P T R
Intrest = $(( $P * $T * $R /100 ))
total_Amount = $P + $Intrest
echo "Intrest Amount: $Intrest \n
total Amount: $total_Amount."
```

Procedure to write

Execute: ./Intrest.sh

enter the values 1000 18 2

esc, shift+:, wq, Enter

✓ Wa script to perform all mathematical calculations to runtime variable values.

✓ Wa script to displays percentage of student of given six subject values.

✓ **read only :-**

It is used to assign value of variable permenantly/ constantly throughout the program

Syntax:-

```
readonly a=10
readonly b=20
```

Executing:-

```
./test_readonly.sh
```

O/P:

```
Bangalore
Bangalore
Bangalore.
```

Script:

test_readonly.sh

```
#!/bin/ksh
readonly a="Bangalore"
echo $a
a="Bellandur"
echo $a
a=30
echo $a
:wq
```

✓ **unset :-**

It is used to unset the behaviour of unset variable as dynamic variables.

Syntax:- unset variablenames;

Scanned with CamScanner

Ex:- read only

```
readonly a=10
echo $a
a=20
echo $a
unset a
a=50
echo $a
```

Variables :—

These are the place holders it allows us to assign values.

- With help of these values we can fullfil program. so below are points to remember while declaring the variables.

* Variable name can be combination of characters & digits along with underscore (_).
* Name of variable start with character. either caps or small.
* Underscore(_) is the keyword to combine more than 2 words as a single word.
* apart from underscore(_) we can't use any of special character.

Examples :— ab, aB, Ab, AB, abc123, abc_def, abc$123 (X)

Positional Parameters (or) Command line arguments :—

The parameters which are passing while executing the script we can call it as positional parameters. The advantage of these parameters are :-

- we can execute one program with different values. and even it allows to automate script or program.
- we can access positional parameters by using their position. starting with $0 end with $9.

$0 --- $9 Parameters

- $0 it gives program name inside shell script.
- $1 gives shell name outside the program.

Scanned with CamScanner

$9 : " 9th "

By default we can access 9 positional parameters.

- We can access positional parameters either in chronological or non chronological order.
- We can assign positional parameters to a variables as shown in below.

$Var_1 = \$1$.

$Var_2 = \$2$

$Var_9 = \$9$

⇒ Script to access and display all positional parameters which we pass.

vi positional.sh

```
#! bin/sh
echo "script access positional parameters"
echo " first argument : $1 "
echo " second        : $2
       third         : $3
       fourth        : $4
       fifth         : $5
       sixth         : $6
       seventh       : $7
       eighth        : $8
       ninth         : $9
       10th          : $10
       11th          : $11
echo "
       Number of positional parameters :
       params are : $#
```

:wq

./positional.sh Hi I am in bangalore, 10 20 30 40 50 60 70

Scanned with CamScanner

✓ shift :-

It is used to access positional parameters if it is more than 9.

- It shifts position from one location to another location.

Syntax:- shift <followed by number>.

vi shift.sh.

```
#! bin/bash
echo $1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12 $13
echo $*
shift 9
echo $1 $2 $3 $4 $5
echo $#
```

:wq

./shift.sh I am sharief currently i'm in bangalore 10 20 30 40 50 60 70 80 90

O/P:-

I am sharief currently i'm in bangalore 10 20

****

- export :-

It is used to export value of a variable from one shell to another. (or) system. It is used to declare Variables as global.

```
a=10
echo $a
export a
ksh
echo $a
```

```
ksh
echo $a
bash
echo $a
```

```
read m1 --- m6
total=$m1+---+$m6
percentage= $total/6
```

student.bash

total=`expr "$m1+---$m6" | bc`

Next

✓ If :-

It is a conditional statement it allows the statements to execute when condition is true.

Syntax:-

```
if [ condition ]
then
    stmt1;
    --
    stmtn;
fi
```

Scanned with CamScanner

$9 : " 9th "

By default we can access 9 positional parameters.

- We can access positional parameters either in chronological or non chronological order.
- We can assign positional parameters to a variables as shown in below.

Var1 = $1.
Var2 = $2
:
Var9 = $9

⇒ Script to access and display all positional parameters which we pass.

vi positional.sh

```
#! bin/sh
echo "script access positional parameters"
echo " first argument : $1"
echo " second        : $2
      third          : $3
      fourth         : $4
      fifth          : $5
      sixth          : $6
      seventh        : $7
      eighth         : $8
      ninth          : $9
      10th           : $10
      11th           : $11
echo "Number of positional parameters :
      params are : $#
```

:wq

./positional.sh Hi I am in bangalore, 10 20 30 40 50 60 70

Scanned with CamScanner

✓ **Shift :-**

It is used to access positional parameters if it is more than 9.

- It shifts position from one location to another location.

Syntax :- shift <followed by number>

vi shift.sh

```
#! bin/bash
echo $1 $2 $3 $4 $5 $6 $7 $8 $9 $10 $11 $12 $13
echo $*
shift 9
echo $1 $2 $3 $4 $5
echo $#
```

:wq

./shift.sh I am sharief currently i'm in bangalore 10 20 30 40 50 60 70 80 90

O/p :-

I am sharief currently i'm in bangalore 10 20

****

- export :-

It is used to export value of a variable from one shell to another. (or) system. It is used to declare Variables ~~value~~ as global.

```
a=10
echo $a
export a
ksh
echo $a
```

ksh

```
echo $a
```

bash

```
echo $a
```

```
read m1 --- m6
total = $m1 + --- + $m6
percentage = $total/6
```

student.bash

total=`expr "$m1+---$m6"`

Next

✓ **If :-**

It is a conditional statement it allows the stmts to execute when condition is true.

Syntax :-

```
if [ condition ]
then
    stmt1
    --
    stmtn
fi
```

Scanned with CamScanner

evaluate the condition and executes some set of stmts.

⇒ script: to count positional parameter if it is more than or less than 2 term -ate the program else perform mathematical calculation.
if operations Complete successfully send a mail of results to your mailid.

vi if_test.bash

```
#!/bin/bash
echo " script checks the positional parameters ".
echo " script started at 'date'
if [ $# -ne 2 ]
  Then
echo " please pass two positional parameters"
echo "example if_test.bash 10 20 "
if Exit 1
fi
Sum = $(( $a+$b ))
Sub = $(( $a-$b ))
mult = $(( $a * $b ))
if [ $? -eq 0 ]
  Then
mail x -s " result of $0 script " shareef@gmail.com < $sum $sub $mult
                                                      (or)
echo $sum $sub $mult > out.txt
< /home/bin master/shell_dir
```

Note:-
If stmt must be followed by "then" keyword and end with "fi" keyword.

✓ else:- It is a part of if stmt it allows us to execute some set of stmts eventhough condition is false.

Syntax:-
```
if [Condition]
  Then stmts
  else stmts
fi
```

Scanned with CamScanner

⇒ Script:- to find given number is even or odd.

vi if_else_even_odd.bash

```
#!/bin/bash
echo "script started at `date` > /home/krastan/shouts_dir/$0.out
if [$# -eq 1]
Then
    if [ $(($1%2 == 0))]
    Then
        echo "given number is even" >> /home/krastan/shouts_dir/$0.out
    else
        echo "given number is odd" >> /home/krastan/shouts_dir/$0.out
    fi
else
    echo "script can take one argument"
    echo "./even_odd.sh"
fi
```

Nested loop.

⇒ given number is polyndrome or not.

rev :- It is used to print given number or character in reverse order.

⇒ to find out the sum of given number. $\left(\frac{n(n+1)}{2}\right)$

`expr "$n($n+1)/2" | bc`

else if :-

it allows us to evaluate multiple conditions at a time.

Syntax :- if [condition]
then
stmts
elif / elif [condition]
Then
stmts
else
stmts
fi

```
echo "enter a number"
read n
rem=$(($n%2))
if [$rem -eq 0]
then
    echo "even"
else
    echo "odd"
fi
```

Scanned with CamScanner

rev: It is used to print given no. of characters in reverse order.

echo 121 | rev o/p: 121

echo Bang | rev o/p: gnaB

Script:- grade:

if >75 → distinct, <75 and >60 → 1st class

<60 and >50 → 2nd class <50 and >35 → 3rd class else fail.

vi grade.bash

```
# /bin/bash
if [ $# -ne 6]
then
  exit 1
fi
grade=$(( $1+$2+$3+$4+$5+$6))
if [ $grade >=75]
then
   echo "student obtain distinction"
else if [ $grade >=60 -o <75]
        then
          echo "1st class"
          else if [ $grade >=50 -o <60]
                then
                  echo "2nd class"
                  else if [ $grade >=35 -o <50]
                        then
                          echo "3rd class"
                          else
                            echo "fail"
                fi
```

Scanned with CamScanner

unix allows us to perform below operations with if conditions

Numaric test

string test

file level test.

Numeric test:-

As part of this we can evaluate one or more number

below are operations.

-eq (=) , -ne (!=) , -gt (>) , -lt (<)

-ge (>=) , -le (<=) ,

String tests-

Below are the operations.

$str_1 = 10$

$str_2 = 20$

[$str_1 = $str_2] | [$str_1 -eq $str_2]

above conditions is true if values at str_1 & str_2 are same

below conditions are true if str_1 & str_2 diff

[$str_1 -ne $str_2] / [$str_1 != $str_2]

-> -n $str_1 : The length at the string is more than 0 (true)

-z $str_1 : True the length of string is '0'

$str_1 : True if is not a null string.

File Comparison Test:-

As part of this test we can determine type of file and it presence below are the options.

-f : true if it is a regular file.

-f <filename>

-s : true if it is a regular file and size>0

Scanned with CamScanner

-c <filename> : character special file

-b <filename> : a block special file

-k <filename> : if given dir contain sticky bit

-l <filename> : link file

-r r : true file having own permission of user

-w <filename> : file having write permission

-x <filename> : " " execute " of user

-s <filename> : file exist and it size is gt 0

Polindrome:-

```
echo -n "enter number"
read n
# store single digit
sd=0
# store num in reverse order
rev=""
# store original number
on=$n
while [ $n -gt 0 ]
do
    sd=$((($n % 10))
    n=$(($n / 10))
# store previous number and current digit in reverse
    rev=$(echo ${rev}${sd})
done
if [ $on -eq $rev ]
then
    echo "number is polindrome"
else
    echo "number is not polindrome"
fi
```

Students:-

```
echo "enter name of student:"
read name
echo "enter roll number"
read num
echo "enter the marks of student"
read m1 m2 m3 m4 m5 m6
echo "name of the student is $name"
echo "number of the student = $num"
echo "marks obtained"
echo "$m1 m2 m3 m4 m5 m6"
echo "$m1 $m2 $m3 $m4 $m5 $m6"
per='echo \($m1+---+$m6\)/6 | bc'
echo "percentage is $per"
if test $per -ge 60
    then
        echo "Grade: first"
elif test $per ge 50 -a $per -gt 59
    Then
        echo "Grade: second"
elif test $per -ge 40 -a $per -le 49
    then
        echo "Grade: third"
else
    echo "fail"
fi
```

Scanned with CamScanner

$V=\$(((r+1))*n))$

Next.

✓ expr:- It is used to perform mathematical calculations.

$V=$ ' expr " $var_1 * var_2$ " '

$v_1 =$ ' expr " ($n+1) " | bc '

$v_2 =$

✓ let :- It is used perform mathematical calculations

let var_1 (arithmetic expression) var_2

let $var_1 * var_2$

Ex:- let x=10 * 2 | 10 + 2 | 10÷2 | 10/3 . .

***
**

Script: to give last month last date followed by the year.

note:. whenever you execute a program you script has to be return last-month last date.

Ex:- If you run scriptom 201601 => 20151231

201602 -> 20160131

!

201612 -> 20161130

Scanned with CamScanner

date-bash

```
#! /bin/bash
c_month = ` date "+%m" `
c_year  = ` date "+%y" `
preced  = 0
if [ $c_month == 1]
 then
   year-1 = $(($c-year -1))
   A_date = ${year-1}1231
 else
   l_month = $(($c_month -1))$

a_day= ` cal $c_year $ l_month | tail -2 | head -1 |
        awk -F " " '{ print $NF }'
if [ $l_month -le 9]
Then
  A_day = $ c_year $preced $l_month $a_day.
else
  A_day = $year $ l_month $a_day.
 fi
fi
 echo " last_month : last date: $A_day"
```

script:-

- To execute a procedure called xyz.prc. on scott/tiger@orcl db
- xyz.prc procedure will load data into temp table.
- extract the data from table t. and place into transaction.txt files of home location of you linux system.
- If transaction.txt file size is gt 0 move to remote server which is /home/mastan@airtelap.com.

```
#! /bin/ bash -vx
log-file = $Home/ $0.out
mail-Id  = " shareefsmv89@gmail.com"
```

Script :-

To take i/p from user and display file type.

vi filetest.bash

```
#!/bin/bash
echo " enter the entity name following exact path "
read fname
  if [-s $fname]
    Then
      echo " $fname is a file and having size >0 "
    elif [-d $fname]
      Then
        echo " $fname is a directory "
    elif [-l $fname]
      Then echo " ... link file"
    elif
    fi echo " $fname might be a block file or character file
```

✓ Case :-

It is a another way of controlling the sequence of execution. The functionality of case is similar to elif statement

Syntax :-

```
case <Control variable> in
    case 1 : ---
             ---
             ;; } stmts
    case 2 : ---
             ---
             ;; } stmts
    case 3 : ---
             ---
             ;; } stmts
    *)     : default stmts
             ;;
esac.
```

Scanned with CamScanner

esac:-
it is used to end case statement.

<Control variable> :-

It is contains the value to perform the operations which mentioned in choices.

Scripts:-

To perform some specific set of operations based on login name or based on execute person.

vi case_test_bash

```
#! /bin/bash
login_name = `whoami`
login_name1 = `logname`
case $login_name/a.out
  a) ps -eaf >> $ mastan/shell_scripting/a.out
     ps -eaf >> /home/mastan/disk_usage.bash
     ls -lrt >> /home/--/a.out
# how to execute another shell script in current shell script
    <exactpath of script>
    ↓
    space
  b) . /home/mastan/transfer.sh
     --
     ;;
  c) . /home/mastan/file_test.sh
     ;;
  *) echo "you are not authenticated to use this script"
  esac.
```

Scanned with CamScanner

```
echo $0 script started at 'date' > $log-file
sqlplus -s scott/tiger@ord <<eof
whatever SQLerror exit 1
whatever OS error exit 2
exec @xyz.prc
eof
>> $log-file
if [ $? -ne 0]
then
   mailx -s "$0 script got failed"
   $mail-Id < $log-file
exit
 else
    SQLPLUS -s scott/tiger@ord <<!!
whenever OSerror exit 1
whenever SQLerror exit 2
Spool $Home/transaction.txt
Select * from emp;
Spool off;
!!
if [$? -ne 0]
Then
  mailx -s "$0 script got failed at file generated:
  $mail-Id < $log-file
else if [ -s $Home/transaction.txt]
     then
        scp $Home/transaction.txt shaeb.smv84@gmail.com:$Home
     if [ $? -ne 0]
        then
          mailx -s "$0 script got failed at transaction" $mail-Id
            < $log file.
     fi
fi else mailx -s "$0 script got failed at transaction" $mail-Id < $log file
fi
mailx -s "$0 script terminated information.txt at 'date' successfully"
      $ mail-Id < $log-file
```

Scanned with CamScanner

Next:

Script:-

To fullfill below requirement:

i) check disk usage of /home

ii) If it is more than 75 delete the files which are modified more than 7 days.

iii) again check the disk utilization still if it more than 75 inform to respective persons.

vi disk_usage.bash

```
# !bin/bash
mail_id = "shoeb@gmail.com
Threshhold = 75
usage = 'df -k /home |tail -1| awk -F" " '{print $5}'
        |awk -F '%." '{print $1}'
if [ $usage -gt ${threshhold} ]
  Then
find /home -typ -f -mtime +7 -exec rm -rf {} \ ;
usage = 'df -k /home |tail -1| awk -F " " '{print $5}'
        |awk -F "%." '{print $1}'
if [ $usage -gt ${threshhold} ]
  Then
mailx -s "The disk usage /home is : $usage' after removing more
        than 7 days modified files
    - $mail <1home/body.txt.
else
mailx -s " The disk usage /home is:: ${usage}" $mailid </home/bodey.txt
  fi
else
  mailx -s " The disk usageof /home is <75" $mailid </home/bo dy2.txt
fi
```

Scanned with CamScanner

Iterative control statements :-

Iterative Control statement :- Unix allows us to execute set of statements, more than one time below are the available operators.

1) while :-

It is a iterative control statement it allows us to execute set of statements or commands specific no. of times.

it executes set of stmts at least one time. eventhough if condition is wrong.

Syntax:-

```
Initialization
while <condition>
do
   statements
   ---
   increment operator
done.
```

- while should be followed by do and end with done, between do and done we can need to mention stmts or cmds. that needs to be execute.
- Increment operator is mandatory else loop will goes to indefinite.
- Condition will be evaluated ~~x no. of times~~ untill condition is false.

Script:- To print line numbers from 1–10. (vi while-test.bash)

```
#!bin/bash
i=1
while [i<= 10]
 do
 echo "numberis $i"
  i = `expr $i+1`
done.
```

Script:- To read i/p line by line from file and display on screen. (read-file.sh)

```
#!bin/bash
echo "enter the filename followed by path"
read    fname
        exec <$fname
while read line
 do
    echo $line
done
exec < tty.
```

Scanned with CamScanner

✓ line: is a variable to access line by line information from file.

✓ exec: is used to redirect the stand i/p from one to another.

Script:- Requirement:-

i) check the DB connectivity every 5 times if it is not running send mail to respective members

ii) every 30 min send an emails to other teams if it is not working

iii) if db is up and running terminate it.

Vi dbConnectivity.bash

```
#!/bin/bash
mail_id = "_ _ _ _"
DBA_mailid = "_ _ _ _"
y=1                          while x=1
do                           do
    sqlplus -s scott@tiger@orcl <<!!
    select * from emp;
    !!
    if [ $? -eq 0 ]
    then
        x=0
    exit
    else.
x=1
mailx -s "orcl DB is not up and running" $mail_Id < /home/mastan/body.txt
sleep 300
y='expr $y+1'
if [ $(( $y%6 == 0 )) ]
    Then then
mailx -s "orcl DB is not up and running from last 30 minutes"
    $DBA_mailId < /home/mastan/body1.txt
fi
    continue
fi
done
```

Scanned with CamScanner

It is used to resume/hold program specific no. of sec

Syntax:- sleep sec

ex:- sleep 3000| 30| 100 - -

✓ Continue :-

It is used to continue program after reading specific no. of portion

✓ for :-

It is a iterative control statement. In unix we have two styles of syntaxes.

1. Method :-

this method works as similar as while loop but below are the differences and syntax.

Syntax:-
```
for (initialization; condition; increment)
do
    stmts;
done.
```

- for loop allows us to mention condition, initialization & increment to gather but while we can't.
- for loop can't executes set of stmts even one time but while loop allows
- we use for loop, if you know how many times your loop need to be execute but we use while loop we don't know how many times loop needs to be execute.
- while loop increment operator must be associated with either addition or subtraction but for loop allows us to mention other things.
- for loop condition evaluates one time throughout the program. but while loop evaluates the condition untill it false.
- Script:- 1-10 line numbers:

note: we can use for loop at command prompt. but we can't execute while loop.

```
for ((i=1; i<=10; i++)) | for i in {0..5}
do                      | for i in
    echo $i
done.
```

Scanned with CamScanner

Script:-

given number is prime or not.

```
vi prime.bash
#! /bin/bash
echo "enter the number:"
read num
for((i=2 , i<$num ; i++))
do
  if [ (($num % $i -eq 0)) ]
  then
    echo "given number $num is not a prime"
  else
    echo "the given number $num is prime"
  fi
done
```

Diff operators in for loop:-

* for i in {0..5}
* for i in {0..10..2}
* for i in (seq 1 20 3)

Next

Script:- To print

```
*
**
* * *
* * * *
* * * * *
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
vi Rightangletriangle.bash
# !bin/bash
echo " This is the printing of * s in a right triangle format"
for(( i=1; i<=5; i++))
  do
    for(( j=1; j<=$i ; j++))
      do
        echo -n "*"
      done
  done
```

-n: it disables printing of in new line o/p will be printing same line

```
for  * * * * *
       * * * *
         * * *
           * *
             *
for(( i=5; i>=1; i--))
```

This is most frequently used syntax while writing shell scripting.
- In this method we need not to declare variable, condition and increment operator all will be taken care by for loop.

The main advantage of this loop is we can give i/p as file, command o/p or range

Syntax :-
```
for <control_variable> in source
do
  stmts;
  ==
  ==
done.
```
source → file | range | command i/p

control-variable is heart in this way, it will iterate loop n no. of times which contains by i/p.

Script:- to print line numbers from 1--100 or 100-1

```
for i in {1..100}
do
  echo "$i"
done
```

```
for i in {100..1}
do
  echo "$i"
done.
```

Script:- to delete all empty lines from ~~them~~ all files and copy same file to another server then delete it.

```
for i in `ls -lrt | grep '^-' | awk -F ' ' '{print $NF}' 2>/dev/null`
do
  sed -i '/^$/d' $i
  scp $i Basha@airtel.co.in :$Home
  rm -rf $i
done.
```

Script:- Consider there is a file called connections.txt. this file contains 100 server details, each row contains one server details differentiated with "|"

connections.txt

username	password	server
Shank	***	airtel
Anand	***	Vodafone
[illegible]	***	BSNL

Scanned with CamScanner

Script:-

To check the Connectivity of each Server whichever Server is not working Capture to one file and email file to respective members.

```
vi server_connection.bash
#!bin/bash
mail_id = "shareeb89@gmail.com"  -> touch
  for i in 'cat $Home/Connection.txt'
    do
      uname = 'awk -F "!" '{print $1}' $i
      pwd = awk -F "!" '{print $2}' $i
      sname = awk -F "!" '{print $3}' $i
      ssh $uname@sname <<!
        pwd
        !!
      if [ $? -ne 0]
        Then
          echo $i >> $Home/Connection.txt
        fi
      mailx -s "Servers not working" $mail_id < $Home/Connections.txt
  done.
```

Script:-

To check wether file is existed and size is_gt 0 then transfer all files to another server.

```
vi file_existance_transfer.bash
#!bin/bash
cd $Home/data
for in `ls -lrt | grep '^-' | awk -F " " '{print $NF}'`
  do
    if [ -s $i]
      Then
    scp $i  Shareeb@airtel.co.in : $Home
    if [ $? -ne 0]
      Then
    mailx -s "$i script not transfered" "xx..." <$Home/body.txt
    fi
  fi
  rm -rf $i
```

Scanned with CamScanner

to execute all positional parameters (procedures) in scott/tiger@orcl db

vi db_execute.bash

```
#!/bin/bash
mail_id = "shaaf.snsr@gmail.com"
for i in $*
do
sqlplus -s scott/tiger@orcl<<!!
whenever sqlerror exit1
whenever DBerror exit2
exec $i
!!
if [ $? -ne 0]
then
  mailx -s "$i Procedure failed" $mail_id < $Home/body.txt
```

note:-

@ symbol to execute .sql files in data base from unix shell script

```
SQLPlus -s scott/tiger@orcl <<!!
@: /$Home/mastan/data.sql
!!
```

data.sql:-

is a unix file. it contains set of DML and ddl stmts above. is the way to execute .sql file in db. which exist in unix.

Script:-

- delete 1st line and last line
- delete 1st field
- add Hi as first field and Hello as last field.
- insert a line 'Test' at 5th line
- Transfer same file to another server

Scanned with CamScanner

vi sed prac.bash

Ques

```
#!/bin/bash
mail-id = "shareef.smv@gmail.com"
cd $Home/maitan/shareef_dir
for i in `ls -lrt | grep '^-' | awk -F " " '{print $9}'`
do
# sed -i '1d;$d' $i  (or)
{ sed -i '1d' $i
  sed -i '$d' $i
✓ awk -F " " '{print $1 = " "; $0}' $i > out.txt
✓ mv out.txt $i
✓ sed -i 's/^/hi/g ; s/$/hello/g' $i
✓ sed -i '4i Text' $i
  scp -$i bash@gm --- $Home.
done.
```

Function

- It enable us to breare down overal funtionality of a script into smaller and logical sub sections.
- Functions are predefined set of programs it takes parameters and perform an action.
- Functions will never execute until we call.

Syntax:-

```
function / function()
{
  --
  --} statements
}
```

- The main usage of functions are we can execute same code with different parameters as needed. it reduces programming lines and makes others to understand easily.

Scanned with CamScanner

vi function_week.sh

```
#! /bin/bash
mail_id="shaheb.smv@gmail.com"
disk_space
{
usage='df -k /home | tail -1 | awk -F" " '{print $5}' |
        awk -F "%" '{print $1}' |
}
db_connectivity
{
user=$1
pass=$2
pdb=$3
proc=$4
sqlplus -s $1/$2@$3 <<!!
exec $proc
!!
chk_step
}
chk_step
{
if [ $1 -ne 0 ]
then
mailx -s $where $mail_id < $Home/ failure_body.txt
else
mailx -s $where $mail_id < $Home/ server sitstp_body.txt.
}
disk_space
if [ $usage > 35 ]
then
where="checking disk space"
find /home -type f -mtime +7 -exec rm -rf {} \;
chk_step $where
fi
```

Scanned with CamScanner

```
disk_space    $where
if [ $usage > 75 ]
  Then
    where = " checking disk space second time"
    Chk_step $where
  exit
fi
```

```
# db_Connectivity  scott  tiger
where = " connecting db".
db_Connectivity scott tiger orcl abc.proc
Chk_step $where
where = " connecting to airtel.ap.com.db"
db Connectivity Ram Ram orcl args.proc
Chk_step $where.
where = " connecting to airtel.kar.com.db"
db Connectivity System Admin123 produce.proc.
Chk_step $where
```

// **break:-**

It is used to comeout the program once we reach to certain position. It should be associate with if condition.

```
for (i = 1; i <= 10 ; i++)
  do
    echo $i
    if [ $i == 8 ]
      Then
        break
    fi
  done.
```

Scanned with CamScanner

until :-

It is a part of iterative control stmt the functionality and syntax is as while.

Syntax:- initialization

until [Condition]

do

statements;

=

=

=

increment

done.

- The only difference b/n while and until is until will only validate condition but while loop validates conditions and statements.
- So set of stmts we can execute by using until loop atleast once even though it contains errors.

How to execute other programing language programs in shell scripting.

Java -jar <programname>

python <programname>

pl <programname>

Ex: Java -jar shscript.java

python shscript.python.

pl shscript.pl.

profile :-

It is a system file available in default working location it contains system define variable and paths.

- If you want to declare any of the variable as a global we need to add in .profile even we can add important shell scripts.
- This file will be execute whenever we login system.

Scanned with CamScanner

eval :-

It is used to evaluate command line arrguments in script.

Ex:- x=y
y=3
echo $$x

* above command displays o/p as follows

1234x

Instead of displaying as .11 displays 1234x because $$ is a system variable and displays processed of current shell. if you want always to display actual value

Echo :- eval 'echo $$x'

o/p :- z.

3104359453 3

POTHAPU CHENCHU BHASKAR

Crm jobs every 5 minitues.

*/5 * * * * filepath with file
m h

*/5

*/5