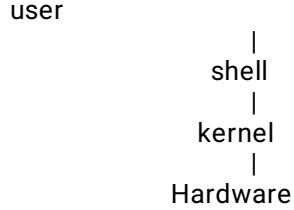
UNIX

----> Unix is an operating system. That contains setup program and act as a link between computer and user.
----> Different flavours in unix. such as: redhat linux, hp unix, Solaris, Linux, SunOS.

Features of unix :

---> Unix is a multi-tasking operating system.
---> Unix provide more security when compared to windows.
---> The security can be done by limitising the user, group, others and providing read,write,execute permissions.
---> User can usely transfer / exchange files in unix.

Unix architecture :



Basic Commands:

$logname ---- It is used to display user name.
$pwd ---- It is used to display the present working directory.
$date ---- It is used to display the system updated time.
$date +%d ----- It is used to display only date.
$date +%m ---- It is used to display month number.
$date +%h ----- It is used to diplay month name(only in three characters).
$date +%A ---- It is used to display full day name.
$date +%H ---- It is used to display only hour.
$date +%M ---- It is used to display minutes.
$date +%S ---- It is used to display seconds.
$cal ---- It is used to display the calender.
$cal month year ---- It is used to display the particular month calender of that year.
$cal year ---- It is used to display the total particular year calender.
$tty ---- It is used to display the terminal name.
$uname ---- It is used to display the present s/w name.
$uname -r ---- It is used to display the version of s/w using.
$hostname ---- It is used to display the server name.
$hostname -i ---- It is used to display the server ip address.
$who ---- It is used to display the who are currently connected to the server.
$who am i ---- It is used to display the current user details.
$whoami ---- It is used to display the current user name.
$echo $USER ----- It is also used to display the current user name.
$who -r ---- It is used to display to find run level of current server.
$clear ---- It is used to display the clear the screen/page.
$exit ---- It is used to exit.
$history ---- It is used to see the previous history what commands we are used.
$history -c ----- It is used to clear the previous history what commands we are used.
$env ---- It is used to display the environment variables.
$echo $0 ---- to see current shell.
$- ---------- to see current shell settings.
$chsh --------- It is used to change the shell.
$netstat ------ to check network statistics.
$lostat ------- to check input,output statistics.
$vmstat ------- to check virtual memory statistics.

$find . –name <file_name> or find / -name <file_name> ----- to find location of a file.

CPU/MEMORY UTILIZATION:

$Top ------- to see cpu utilization.(we want to press 'q' to stop run level). We get PID, USER, …. In the 9th column we get % CPU & in 10th column % MEM.
$Top -m ------ to see memory utilization.

Process :

$ps -ef ----- it displays current running process list in the linux server. We get UID, PID, PPID, C, STIME, TTY, TIME, CMD details in a table format.
$ps -f ----- it displays current running process list in the current user account.

---

inode number:
----- > Inode number is number, used to uniquely identify a file in a file system and these inode numbers are generated by the system.
----- > We can't delete the inode number, even root user can't get the details of the inode table and its physical location.
It's all are managed by System / kernel, for example when you will create a file / dir one inode number will be allocated
and when you are removing it any file / dir that inode number will be free.

$df –i ------ we can see here how many inodes are free and used.
$ls -il <file_name> or <dir_name> ------ to find inode number for a directory or a file.

CREATING FILES:

CAT ---- >It is used to create/insert data/open the file. It can create only one file at a time. It can open multiple files.

$cat >filename --- It will create a file and automatically asking insert the data into file.
$cat <filename --- It is used to open the file.
$cat >>filename --- It is used to reinsert/append the data into file.
$cat file1 file2>>file3 --- It is used to copy/append the data from file1,file2 to file3. If data is already exited in file3 it will not be override the new data append.
$cat filename --- It will display the file data.
$cat -n filename ---- It will diplay the file data with line numbers.
$cat file1 file2 … filen --- It will open the mutiple files.
$cat /etc/*release -- It is used to display about os details.
$cat /etc/passwd ----- to see all users information.
$cat /dev/null > filename ----- to empty file.
$cat>.filename ----- to create hidden files.
$cat /proc/cpuinfo ----- It will display the cpu information
$mv .filename filename ----- to unhide the files.
$ls -a ----- ls used to list hidden files.
$cat filename | tail -n | cut -d"," -f n ------ It is used to display the particular word in the particular line of a file.

TOUCH ---- >It is used to create the empty files. It will create multiple files at a time.

$touch filname ---- It will create the empty file.
$touch file1 file2 ... filen ---- It will create the multiple files at a time.
$touch filename{1..n}.type ---- It will create 'n' number of files at a time(type means txt,c,.....).

REMOVE(rm) ---- >It is used to remove/delete the file.

$rm filename
$rm file1 file2 ... filen ---- It will remove multiple files at a time.
$rm -i filename ---- It is used to remove the file with conformation/permission.
$rm -f filename ---- It is used to remove the file with forcebly.
$rm -rf filename ---- We don't use this cmd in real time. Bcz it will delete the permanently.
$rm -r ---- It is used to delete the directories which are non-empty.

MAKING DIRECTORIES(mkdir) ---- > It is used to create the directories.

$mkdir dirname
$mkdir dir1 dir2 ... dirn
$mkdir .dirname
$mv .dirname dirname

REMOVING DIRECTORIES(rmdir) ---- > It is used to remove the directories.

$rmdir dirname
$rmdir dir1 dir2 ... dirn
$rmdir -i dirname
$rmdir -f dirname

COPY ---- > It is used to copy the data from one file to another file. The target file is not created firstly then it will create target file and copy the data.

$cp file1 file2
$cp -p <source file> <target file> ------ to copy the file without changing the time stamp.
$cp -r <source file> <target file> ----- to copy the directories.

LINKS ---- > It is used to create the links b/w files.

---- > There are two types of links. such as :(i). soft link (on both files and dir) (ii). hard link (only on files)

$ln –s <source_file> <target_file> ------ to create soft link.
$ln <source_file> <target_file> ------ to create hard link.
$ls -l <sourcefile_name> ------It display either link is created or not.
$ls -ltr | grep ^l ------ It is used to display only link files.
$ls -ltr | grep ^- ------ It is used to list the only files.
$ls -ltr | grep ^d ------ It is used to list the only directories.

* The difference b/w the soft link and hard link is if we remove the source file in softlink and the target file is unusable.
But if we remove source file in hard link but we can use target file data.
And in hard link the source inode and target inode numbers are same.
But in soft link the source inod and target inode numbers are different.
in soft original file is deleted the destination file is not usable.
in hard link original file is deleted the destination file is still useful.

RENAME / MOVE ---- > It is used to rename the file name.

$mv oldf_n newf_n

CHANGE DIRECTORY(cd) ---- > It is used to change from one directory to another directory.

$cd dirname
$cd ..
$cd ~
$cd .

LIST(ls) ---- > It is used to list of files data.

$ls
$ls -a ----- It is used to list the hidden files
$ls -r ---- It will display the data reversly
$ls -R ---- It will display the data recursively
$ls -t ---- It will display the data based on time
$ls -i f/d_n ---- It will display the file/dir inode number
$ls -ld ---- It is used to display the long list dirctories
$ls -lrt ---- It is used to list files in reverse order based on time stamp
$ls -ltr a* ----- It is used to display the list of files/dir which starts with 'a'.
$ls -ltr a?b?c ---- It is used to display the list of files/dir which have 1st,3rd and 5th same letters.
$ls -ltr [abcd]* ---- It is used to display the list of files/dir which are starting with that letters.
$ls -ltr [a-c] [h-s] [l-s] ?? --- It is used to display the list of files/dir which are starting with that letters and remaining.
$ls -lhr -------- It is used to display the list of files/dir in kb.{h - human readable}
$ls -l ------- To list down file/folder lists alphabetically.
$ls -lt ------- It will display all details in table format.

first field -- It explains file type and permissins(r - read, w - write, x - execute). It starts with '-' means it is a normal file.
It starts with 'd' means it is a directory.
It starts with 'L' means it is a link symbol.
It starts with 'M' means it is a shared memory.

Second field -- It displays the no. of links allocated for the file and by default '1' for files and '2' for diectories.

Third field -- It specified the user name.

Fourth field -- It specified the group name.

Fifth field -- It is specified the file/dir size. By default '4096' for the directory.

6th,7th,8th fields -- It specifies the month,date and time.

9th field -- It specifies file/dir name.

WILD CARD CHARACTERS(wc) ---- > It is used to display the total lines,words,characters in the file.

```
$wc   filename
$wc    -l   filename
$wc    -w   filename
$wc    -c    filename
$wc    -lw   filename
$wc    -wc   filename
$wc    -lc    filename
```

COMPARE(cmp) ---- > It is used to compare the two files i.e byte by byte. It will display number(how many bytes and first occurance line are differ).

```
$cmp   file1    file2
```

DIFFERENCE(diff) ---- > It is used to see the difference b/w two files. It will display both tables to show the difference.

---> If it shows '<' symbol before the line. Then we want to append the line.
---> If it shows '>' symbol after the line. Then we want to delete the line.

```
$diff    file1    file2
```

FILE NAME(file) ---- > It is used to see the which type of file.

```
$file    filename
```

FILTERING FILES:

---- > We can insert the data into tables by two methods. such as:

(a). Tab Method (b). Delimiter key(flat files)--{,}

```
EX(a):$cat >filename
1    2    3    4
abcd     efgh     ijkl  mno
pqrs     tuvw     xyz abcd
efgh     ijkl  mno     pqrs
tuvw     xyza     bcde     fghi
```

ctrl+d

```
EX(b):$cat >filename
1    2    3    4
abcd,efgh,ijkl,mnop
qrst,uvw,xyz,abcd
```

efgh,ijkl,mnop,qrst
uvw,xyz,abcd,efgh

ctrl+d

CUT ---- > It is used to cut the data which we want.

```
$cut    -f    2   filename
$cut    -f    2-4     filename
$cut    -f    2-   filename
$cut    -d","     -f    2     filename
$cut    -c    1-10      filename
```

PASTE ---- > It is used to paste the data from one file to another file.

```
$paste   file1   file2
```

TRANSFER(tr) ---- > It is used to change/transfer the character in the file.

```
$tr   "aeiou"   "AEIOU"   < filename
$tr    "[a-z]"   "[A-Z]"   < filename
$tr    ","   "\t"   < filename
$tr    -d    "a" < filename
$tr    -d      "aeiou"     < filename
$tr    -d      '\n'     <filename
$tr     '\n'     '   '   <filename
$cat     filename | tr     '[ :space:]'   '\n'      ----- we want to write 'space' in cmd.
$cat     filename | tr     '[ :space:]'   '\t'
$cat     filename | tr     -d     '[A-z]'
$echo     " my name is dileep " | tr      '[a-z]'     '[A-Z]'
```

SORT ---- > It is used to sorting the file.

```
$sort   filename
$sort    -r     filename      ---------- It displays file contents in ascending order.
$sort    -u   filename       ---------- It displays unique lines.
$sort      -n     filename          ---------- It displays file contents    based on numeric
comparision.
$sort    -f      +3     -2      filename
```

UNIQ ---- > It is used to display the uniq data in the files.

```
$uniq     filename
$uniq     -u      filename     ----- It displays non duplicate lines.
$uniq     -d      filename     ----- It displays only duplicate lines.
$uniq     -c      filename     ----- It counts how many times the lines are repeated in the file.
```

String Attributes ::: 1. Grep 2. Sed

1. GREP ---- > It is used to globaly search the regular expression & print it.

$grep "word" filename ---------- It searches the word in the given file.
$grep "word" file1 file2 ... filen ------- It searches the word in the given n number of files.
$grep "word" * ---------- It searches the word in current directory all files.
$grep -i "word" filename ----- It ignores case sensitive.
$grep -n "word" filename ----- It prints lines with the line numbers containing word.
$grep -c "word" filename ----- It counts no. of lines having word.
$grep -v "word" filename ----- It displays the lines do not containing the word/string.
$grep -o "word" filename ------ It prints only pattern containing word.
$grep -l "word" filename ------ It prints only filenames containing word.
$grep --color "word" filename ------ It highlights the pattern with color.

--- > There are three types of regular patterns in grep. such as:

(1). Character Pattern
(2). Word Pattern
(3). Line Pattern

(1). Character Pattern:

$grep "aeiou' filename
$grep "b[aeiou]" filename
$grep "b....d" filename
$grep "bcd*" filename

(2). Word Pattern:

\< ---- Starting Of the word
\> ---- Ending Of the word

$grep "\<word\>" filename
$grep "\<word" filename
$grep "word\>" filename

(3). Line Pattern:

^ --- starting word
$ --- starting word

$grep "^word$" filename
$grep "^word" filename
$grep "word$" filename
$grep "^\<word\>$" filename
$grep "^$" filename
$grep "^...$" filename
$grep "^[bkt]" filename
$grep "^[^bkt]" filename
$grep -v "^\." filename ----- It displays non-empty lines.
$grep -v "^$" filename ----- It deletes the empty lines.
$grep -c "^$" filename ----- It counts the empty lines.

FASTER GREP ---- > It is used to search the morethan one string at a time in a file.

$fgrep "word
>word

>word" filename

EXTENDED GRIP(egrip) ---- > It is having both the grep & fgrep commands and also supports wildcard characters.

$egrep "ab[2]c" filename
$egrep "ab[2,5]" filename
$egrep "ab[3,]d" filename
$egrep "(word | word | word)" filename

---

2. STREAM EDITOR(SED) --- It is used to search & replace the string in file.

$sed "s/oldstring/newstring/g" filename
$sed -i "s/oldstring/newstring/g" filename
$sed "s/oldstring/newstring/" filename
$sed "s/oldstring/newstring/gi" filename
$sed -n "2p" filename
$sed -n "$p" filename
$sed -n '1,5!p' filename
$sed -n '1p;$p' filename
$sed -n '1d;$d' filename
$sed -n '1d' filename
$sed -n '$d' filename
$sed '2,4d' filename
$sed '2,4!d' filename
$sed '/^L/d' filename
$sed '/x$/d' filename
$sed '/[xX]$/d' filename
$sed -n '/^A\|^L/p' filename ---------- Print lines beginning with either 'A' or 'L'.
$sed "s/^$/sentence" filename
$sed '/^$/d' filename
$sed 's/^/./g' filename
$sed 's/^/#/g' filename
$sed 's/$/#/' filename
$sed 'n;d' filename ---------- Print every alternate line.
$sed 'n;n;N;d' filename ---------- Print every 2 lines and give gap two lines. By default one gap will occur.
$sed 'n;n;N;N;d' filename ---------- Print every 2 lines and give gap three lines.
$sed '/string/d' filename
$sed '/string/!d' filename
$sed '/string/p' filename
$sed '/string/!p' filename
$sed -n '/Unix/,${/X$/p;}' filename -------- Print lines ending with 'X' within a range of lines.
$sed -n '/Solaris/,/HPUX/{//!p;}' filename -------- Print range of lines excluding the starting and ending line of the range.
$sed -n '/[ux]/p' filename ---------- Print lines which contain the character 'u' or 'x'.
$sed -n '/[xX]$/p' filename --------- Print lines which end with 'x' or 'X'.
$sed -n '/[ux]/p' filename ---------- Delete lines which contain the character 'u' or 'x'.
$sed -n '/[xX]$/p' filename --------- Delete lines which end with 'x' or 'X'.
$sed -e "s/oldstring/newstring/g
>s/oldstring/newstring/g
>s/oldstring/newstring/g" filename
$sed 's/\$/red/g' filename
$sed '2s/oldstring/newstring/g' filename
$sed '2s/oldstring/newstring/2' filename
$sed 's/oldstring/newstring/1' filename

$sed '2,5s/oldstring/newstring/g' filename
$sed '2,5w newfilename' filename
$sed 's/[^,]*,//' filename ---------- To remove the 1st field or column and To print the remainning fields.
$sed 's/.*,//' filename ---------- To print only the last field and remove all fields except the last field.
$sed 's/,.*//' filename ---------- To print only the 1st field.
$sed 's/,[^,]*,//' filename ---------- To delete the 2nd field.
$sed 's/ /+/g' filename | bc ---------- To find sum of all columns / numbers in a line.
$sed 's/,/+/g' filename | bc ---------- In case of file being comma separated instead of space separated
$sed -i '4 i linedetails' filename ---------- It will insert the new line details into 4th row.

HEAD ---- > It is used to find the top lines of the file.

$head -n filename

TAIL ---- > It is used to find the last lines of the file. By default “tail” prints the last 10 lines of a file, then exits.

$tail filename
$tail -n filename
$tail +n filename
$tail -f filename ---- It will display file growth i.e full table like cat & to come out press ctrl+c.

ex : tail -f /var/log/cron

$tail -f -s <sleep interval in seconds> /path/to/file

ex : $tail -f -s 5 /var/log/secureMar

$tail /path/to/file1 /path/to/file2

$tail -fq /var/log/secure /var/log/cron ------ If you want to remove this header, use the -q option for quiet mode.

----> Now what if I have a very huge /var/log/messages and I am only interested in the last certain number of bytes of data, the -c option can do this easily.
observe the below example where I want to view only the last 500 bytes of data from /var/log/messages.

$tail -c <number of bytes> /path/to/file

ex: $tail -c 500 /var/log/messages

PIPING (|) : It is used to combine two or more commands(previous command of output it will give the next command of input)

$ls | grep "^d"

tee : It is used that to append the data from one file to another file as well as to display the data on screen.it will copy the running file data as well

$cat emp | tee file1

FILE PERMISSIONS:

who	permissions	
USER/OWNER - u	+ ---- give permission	read - r
GROUP - g	- ---- deny permission	write - w
OTHER - o	= ---- assign permission	execute - x

$chmod u+r filename
$chmod g-w filename
$chmod o=x filename

Numeric Codes : 0 - no permission , 1 - x , 2 - w , 4 - r , 3 - w,x , 5 - r,x , 6 - r,w , 7 - r,w,x

Octal Codes : read - 4 , write - 2 , execute - 1

$chmod 755 file/dirname

$Umask ------ It is a default file permission, default umask value is 002.

$chown ownername file/dirname ---- It is used to change owner name.

$chgrp groupname file/dirname ---- It is used to change group name.

STICKY BIT ::: We will give sticky bit in two ways. such as: 1. Symbolic (t) 2. Numerical / Octal way (1)

----> If the owner of the directory doesn't want to delete any of the files / sub-dir under the directories we can use sticky bit operation.
----> By using sticky bit, we can make sure that no one can delete files / sub-dir. But they can read and modify the data.

Syntax : chmod 0+t dirname ---- to add the sticky bit
chmod 0-t dirname ---- to remove the sticky bit

Q. How can we identify weather sticky bit is exited or not for the directory ?

Ans : when we use 'ls -ld', if we see 't' that sticky bit exit for the directory.

COMMUNICATION COMMANDS :

1. WRITE ::: It is used to write a message to another user account, but he should be logged into the server.

Syntax : $write username/terminalname

ex : $write techo2
Hello, I am in technosoft lab
ctrl+d

$write techo1
I am also in lab
ctrl+d

$mesg n --- to deny messages.
$mesg y --- to allow messages.

---

2. WALL ::: It is used to send broadcast message to all users, whoever connected to the server and mesg in 'y'.

ex : $wall
welcome
ctrl+d

---

3. MAIL ::: It is used to send the mails.

syntax : $mail username

---
ctrl+d

ex : $mail techo1
subject : hello
hi, how r u ?
where r u ?
ctrl+d

ex : $mail techo1 techo2 ..... techon
subject : hello
hi, how r u ?
where r u ?
ctrl+d

$mail ---- It is used to open the mails. (mailno, senders name, date, time, subject will be displayed)
$&2 ---- It is used to second mail.
$&3 ---- It is used to third mail.
$&q ---- It is used to quit from the mail box.
$&w filename ---- It writes the current mail contents to given files.
$&r ---- It is used to reply.
$&p ---- It is used to print out.
$&d ---- It is used to delete currnt mail.
$&d 2 ----- It is used to delete second mail.
$&d 1-10 ---- It is used to delete 1-10 mails.

NOTE ::: By default all mails will store in primary mail box(/var/spool/mail).
All opened mails in primary mail box are be tranferred to secondary mail box(mbox).

$mail -f ---- It is used to open the secondary mails.

Mailx ::: It is used to send the mails from the server. we can attach the files also in 'mailx' command. we have options like 'cc' and 'bcc'.

syntax : $mailx mailid's
subject :*******
****
****
ctrl+d

ex : $mailx -s "subject" -c "mailid's" -b "mailid" filename

----> In the above command '-c' specify to add mails in CC (corbon copy) and '-b' specify to add mails in BCC (blind carbon copy) .

ex : $echo " i am dileep" | mailx -s subject mailid's

----> The above command is used to send ' i am dileep ' as body of the mail.

ex : $cat filename | mailx -s subject mailid's

or

$mailx -s subject < filename mailid's

----> The above commands is used to send the file data as a body.

UUENCODE ::: The uuencode command converts a binary file to ASCII data. This is useful before using BNU/uucp mail to send the file to a remote system.
The uudecode command converts ASCII data created by the uuencode command back into its original binary form.

----> It is used to send the files as attachment while sending the mail.

syntax : uuencode [SourceFile] [OutputFile]

ex : $uuencode file1 file2 ... filen | mail -s "subject" "mailid's"

ex : $uuencode unix1 unix2 | mail jsmith@mysys

NETWORKING COMMANDS :

1. Telnet ::: Telnet protocol is In-Built in windows/linux. It is used to connect the remote servers. Default telnet port number is '23'.

syntax : $telnet ipaddress/servername.portnumber
login :
password:

----> 'Ctrl+]' to comeout from the telnet then press 'q'.

2. Ftp ::: FTP stands for 'file transfer protocol'. It is used for to transfer files from one server to another server account.

syntax : $ftp ipaddress/servername
login:
password:
ftp>

NOTE : In real time, we use 'winscp' tool to transfer files. In this tool, we just drag and drop the files.

ftp> ls ---- to display server side files.
ftp> !ls ---- to display client side files.
ftp> pwd ---- to display server side present working directory.
ftp> !pwd ---- to display client side present working directory.
sftp> put <server side filepath> <client side filepath> ---- to transfer files from server side to client side.
sftp> mput file1 file2 file3....filen /var/tmp/
sftp> get <client side filepath> <server side filepath>
sftp> mget /var/tmp1/file1 file2 ... filen /var/tmp/
sftp> put -r dir /var/tmp/
sftp> get -r dir /var/tmp/
sftp> mput -r dir1 dir2 ... dirn /var/tmp/
sftp> mget -r dir1 dir2 ... dirn /var/tmp/
sftp> bye ---- It is used to come out from the ftp/sftp server.
ftp> help ------ It will display the list of commands that we are using in ftp. [ex: ftp> help ls , ftp> help dir ,.....]

-----> By default ftp will get the data in ASCII mode. In ASCII mode the target file some times the contnets don't display properly.
-----> To get the data in binary mode we have to type "ftp> binary" / "ftp> bi".

Difference b/w ftp and sftp :

1. The main difference b/w ftp and sftp is 'Encryption'. In FTP cmd the encryption can't be done where as in sftp cmd the encryption can be done.
2. By using sftp we transfer files more sequrely than ftp.
3. In real time, most of the clients prefer using sftp.
4. The port numbers of ftp - 21 and sftp - 22.
5. The sftp runs over ssh(Sequair Shoket Host) so that the port no is same for sftp and ssh.

NOTE :: SCP (secure copy)/vsftp is also one of the transfer file cmd.

SSH(secure socket host) ::: It is used to login to the another mission/server from current mission/server without loging out from current mission.

---> As well as it is used to run the another server scripts and commands in current server.

syntax : to loging to the another server

$ssh username@ipaddress/hostname (in this cmd 'ipaddress' and 'hostname' are the remote server details)
password: ******

---> Once we loging to the new server to come out from new server to local server we have to type 'exit'.

syntax : to run remote/another server scripts/commands from local/current server.

syntax : $ssh username@ipaddress/hostname <script/cmd>

ex : $ssh dileep@168.126.23.2 /var/tmp/dil.sh

Q. HOW TO GENERATE KEY'S USING 'SSH' FOR THE PASSWORDLESS LOGING'S

AND TRANSFORMING FILE'S ?

Ans : In unix we have an option called as "keygen" for generating RSA (Remote Secure Authentication key's).

---> Once we generate the key and copy that into the remote server, we can perform all the actions such as: ftp, sftp, scp, ssh without using any password, it is calling as 'passwordless authentication'.

---> Generally Admin team contain access to generate the key's.

syntax : $ssh -keygen -t -rsa
generating public/private rsa key pair then enter file location where to save the key ':filelocation' then press enter
then Enter password :
then Re-enter password :

/* your identification has been saved in filename.
your public key has been saved in filename. */

---> After rsa generation key's the file which contains ssh key's as to be copied in remote server. Then only the key's will be work.

Ex : $ssh -keygen -t -rsa
Generating public/private rsa key pair
enter file location where to save the key :/home/user/.ssh/id-rsa
enter password : 123456
re-enter password : 123456

/* your identification has been saved in '.ssh/id-rsa'.
your public key has been saved in '.ssh/id-rsa-pub'. */

Note : 'id-rsa' is the file which contains key's and 'id-rsa' stores in '.ssh' directory.

JOB CONTROL :

Jobs are three types. Such as: 1. Foreground Jobs 2. Background Jobs 3. Nohup jobs

NOTE :: By default all jobs are come under foreground jobs.

-----> In foreground, user can execute only one job.
-----> In background, user can execute many jobs.

$jobs ---- It is used to display only background jobs with jobid's.

EX :: $cp file1 file2 ---- foreground job

$cp file1 file2 & ---- Background job

Q. How to kill foreground jobs ?

Ans : ctrl+c

Q. How to suspend foreground jobs ?

Ans : ctrl+z

Q. How to resume suspended foreground jobs ?

Ans : $fg jobid

Q. How to bring background job to foreground job ?

Ans : $fg jobid

Q. How to send foreground job to background job ?

Ans : First suspend foreground job by using "ctrl+z" and execute the following command "$bg".

3. nohup ::: The nohup jobs will create in server account. So nohup jobs will execute even the user disconnects from his account.

-----> The nohup command creates one nohup.out file & it appends the nohup job output into nohup.out file.

ex : $nohup cp file1 file2 &

PROCESS :::

---> A process is nothing but which is running of a program.
---> Each process is spwaned originated by another process.
---> The process which originated has a process is called as 'parent process' and the new process is known as 'child process'.

$ps -ef ---- It displays currently running process list in the linux server.

$ps -f ---- It displays currently running process list in the current user account.

$kill -9 pid ---- It is used to kill the process.

$kill -9 'ps -ef | awk '/search something/' { print $2 }' ---- to kill more than two processid's.

$pkill processname ---- It is used to kill the process with processname.

NICE VALUE (NI) ::: Nice value is nothing but the process priority, it ranges from -19 to 19.

---> If it is '-19' then it is the highest priority process, it means the cpu has to spend extra time for the particular process.
---> If it is '19' then it is very less priority process.
---> By default if we run any process the nice value will be '0'.
---> Generaly admin people will allocate the nice value for the particular process.

TYPES OF PROCESS : 1. Daemon Process / init process (parent 'p1' only)
2. Orphan Process (parent 'p1' and child 'p2')
3. Zombie Process (parent 'p1' and child 'p2' and child 'p3' and child 'p4' .....)

JOB SCHEDULING ::: By using four ways we can schedule the jobs. such as : 1. crontab 2. at 3. batch 4. autosys

1. CRONTAB : It executes the given jobs repeatedly in the server account and by default it sends the crontab jobs output to user mail account.

----> By default the crontab command accepts 5 fields : (a). Minutes (0-59)

(b). Hour (0-23)

(c). Day (1-31)

(d). Months (1-12)

(e). Weekdays (0(sun)-6)

Q. How to open crontab editor?

Ans : $crontab -e * * * * * sh script.sh
| | | | |
M H D M W

Ex : $crontab -e 50 8 1-10 2,3,6 4,6 sh script.sh

$crontab -e 50 8 5-15 4,5,7 2,5 sh script.sh>file1 --- it writes output into file1 file.

$crontab -l ----- It is used to list the crontab jobs.

$crontab -r ----- It is used to remove the cronjob jobs.

2.AT : It executes the given jobs only once.

EX : 1. $at now
at>ls
ctrl+d

2. $at now+15minutes
at>sh script.sh
ctrl+d

3. $at now+2hours
at>sh script.sh>file1
ctrl+d

4. $at now+3days
at>ls
ctrl+d

5. $at now+1month
at>sh script.sh
ctrl+d

6. $at 12:20am aug 01
at>sh script.sh
ctrl+d

7. $at 4pm tomorow
at>sh script.sh
ctrl+d

Q. How to list all at/command jobs ?

Ans : $at q

Q. How to remove at/command jobs ?

Ans : $atrm jobid

3. BATCH ::: The batch jobs will execute, when the server is free and server load is less.

ex : $batch
at>ls
ctrl+d

ZIP FILES :::

$gzip filename ------ It is used to make a zip file.

note : The zip files are saved in .gz files.

$zcat filename.gz ----- It is used to open zip files.

$gunzip filename ----- It is used to unzip a file.

$compress filename

$uncompress filename.z

DISK STATUS :::

1. $df ----- It displays disk space in bytes.

2. $df -h ------ It displays disk space in kilo bytes.

3. $du ------ It displays disk usage.

4. $du -k * ----- It displays disk usage in kilo bytes.

Q. How do you resolve/clear the disk space issue?

Once we get an alert message on issuing disk space (filesystem uttilization or partition) is full, then we go to respective server and we will check the space utilization/availability using '$df -h' / '$du -k *' command. If there is very less space available then by using below 3 ways we can clear the disk space issue.

(a). By deleting unwanted data or by removing unwanted log files.

(b). By compressing data in the partition.

(c). By moving data from current partition to another partition.

FREE ::: It is used to check the memory of the server. By default it will also show the data in 'KB'.

----> By using 'free -g' we check the memory usage in 'GB'.

SWAP MEMORY ::: It is the additional memory which is allocated to the server. By using 'free -gt' (t - total) cmd we can check tha swap memory along with normal memory.

Netstat ::: It is used to find out the network statistics. It means it will display the information such as the port number and address the mission is being connected to and it will shows the states of establishment.

syntax : $netstat

ex : $netstat | head -3

Netstat -l ::: It is used to find out network statistics as well as particular port status whether it is lessing or not.

Q. How to find out whether particular ports listing or not ?

Ans : $netstat -l | grep portnumber

or

$netstat -a | grep portnumber

IOSTAT ::: It is used to input, output statistics of the current mission.

syntax : $iostat

VMSTAT ::: It is used to find out virtual memory statistics.

syntax : $vmstat

FIND COMMAND ::: Find command is used to searching for files from different directories based on name, type, size, permissions, inode number, links, time, maxdepth, mindepth, etc..

-----> It is an usefull command when we don't known the file location or created files by particular time.

$find . -user username

SEARCHING FOR THE FILES BASED ON NAMES :

$find . -iname filename/dirname
$find / -iname filename/dirname

----> In the above cmd '.' denotes current directory and '/' denotes root directory. And 'i' denotes, it ignore the case sensitive.

$find . -iname "di*"
$find . -iname "*ep"

SEARCHING FOR ONLY THE FILES / DIRECTORIES :

$find . -type f
$find . -type f -iname filename
$find . -type f -empty
$find . -type d
$find . -type d -iname dirname
$find . -type d -empty

SEARCHING FOR THE FILES BASED ON INODE NUMBER :

$find . -inum inodenumber
$find . -inum 131288

SEARCHING FOR THE FILES BASED ON LINKS :

$find . -links numberoflinks
$find . -links +3/-3/3

SEARCHING FOR THE FILES BASED ON PERMISSIONS :

$find . -perm "permission numbers"
$find . -perm "777"

SEARCHING FOR THE FILES BASED ON SIZE(char, words, kb, mb, gb) :

$find . -size "+size"
$find . -size "+10c"
$find . -size "+10w"
$find . -size "+10k"
$find . -size "+10G"
$find . -size "+10M"

SEARCHING FOR THE FILES BASED ON Accessed, Created, Modified TIME :

$find . -ctime "(required time)"
$find . -ctime +7
$find . -atime +7
$find . -mtime +7 ----> this cmd which are modified 7 days before will give output.
$find . -mtime -7 ----> this cmd which are modified in last 7 days will give output.
$find . -mtime 7 ----> this cmd which are modified on exact before 7th day.

SEARCHING FOR THE FILES BASED ON Created, Modified, Accessed FOR 24hrs and before :

$find . -cmin "required minutes"
$find . -cmin +30
$find . -cmin +120
$find . -cmin -30
$find . -cmin 30

SEARCHING FOR THE FILES BASED ON MAXDEPTH & MINDEPTH :

EX : $find . -iname file1
o/p : ./file1

./var/file1
./var/temp/file1
./var/temp/perm/file1

$find . -maxdepth <range> -iname filename
$find . -maxdepth 1 -iname file1

o/p : ./file1

$find . -maxdepth 2 -iname file1

o/p : ./file1
./var/file1

$find . -maxdepth 3 -iname file1

o/p : ./file
./var/file1
./var/temp/file1

$find . -mindepth 1 -iname file1

o/p : ./file1
./var/file1
./var/temp/file1
./var/temp/perm/file1

$find . -mindepth 2 -iname file1

o/p : ./var/file1
./var/temp/file1
./var/temp/perm/file1

$find . -mindepth 3 -iname file1

o/p : ./var/temp/file1
./var/temp/perm/file1

-----> xargs ::: It is used as cmd/option which pass the output list or argument list from the previous command & pass next cmd. When we do XARGS the
argument list pass to the next cmd as a single line argument.

$find . -cmin -5 | xargs rm
$find . -cmin -5 | xargs gzip
$find . -cmin -5 | xargs mv

-----> exce ::: It is also an option which takes the output of previous command and process output one by one(1/1) to the next command.

$find . -cmin -5 | exec rm -rf {} ./dirname

I/O REDIRECTION :::

----> In all operating systems there is standard input and output devices. In unix also we have standard input, output and standard output errors.
----> In unix each device is associated with a number to denote from where input should be taken & output should be return,errors should be return.

STREAM	DEVICE	VALUES
standard input	keybord	0
standard output	terminal screen	1
standard output errors	terminal screen	2

ex ::: cat filename -------> standard input
hi, i am dileep -------> standard output

$find / -iname ab 2>file1

----> when we use above cmd only the output will be display on the screen & the errors will be thrown to the 'file1'.

$find / -iname ab>file2

----> when we use above cmd only the errors will be display on the screen. The output will be redirected to the 'file2'.

$find / -iname ab>&file3

----> when we use above cmd nothing will be display on the screen. Both standard output & errors will be thrown to the file.

STAT ::: It is basicaly used to find out the last accessed, modified and changed time, size, blocks, device, inode number, links, uid, gid of the file.

syntax : $stat filename

---> Whenever we see the file that is called 'Access time'.
---> When we do any modification in the date of the file that is called 'modified time'.
---> When we change the filename/permissions of user without any modification to data that is called 'change time'.

SPLIT ::: It is used to split the largest file into small file.

---> By default the file split into 1000 lines for a file.
---> We can split the file based on size also.

syntax : $split filename

---> When we use split command without any option it will split the files into 1000 lines for a file.
---> By using '-ln' option we can specify the line numbers that how many we would like to split.

syntax : $split -ln filename

---> The new files after spliting named/saved as 'xaa, xab,xac....., xn'.

syntax : $split -ln filename F

---> The new files after spliting named/saved as 'Faa, Fab,Fac....., Fn'.

syntax : $split -ln -d filename F

---> The new files after spliting named/saved as 'F01, F02,F03....., F0n'.

syntax : $split -ln -a 1 -d filename F

---> The new files after spliting named/saved as 'F1, F2,F3....., Fn'.

syntax : $split -b 10k filename

---> The above cmd split the file into 10kb files. [b - bit]

syntax : $split -b 10m filename

---> The above cmd split the file into 10mb files.

syntax : $split -n 2 -a 1 -d filename F

---> The above cmd split the file into 2 files of equal length.

PING ::: It is used to find out whether the particular server is up and running or not

syntax : $ping ipaddress/servername

ifconfig ::: It is used to find out the ipaddress and configuration of the particular server.

syntax : $ifconfig

nslookup ::: It is used to find out the server and address & name server and internet address.

syntax : $nslookup ipaddress

ex : $nslookup 169.176.40.3

AWK ::: It is one of the scripting language of the command, which is used for manipulate the data and generate the reports.

---> AWK perform the actions in following way:
1. It can scan file line by line.
2. Splits each input line into fields.
3. Compare input line with the pattern.
4.AWK is the cmd which is mainly used to generate the reports.

syntax : $awk [option] filename

ex : $awk /30/ emp ---- It display containing '30' records.

ex : cat>emp

```
        10          ll          30          40          50          60    ------ record1{ $0 }
        am          in          as          up          87          am    ------ record2 { $0 }
        ui          io                pl                      ty                      yu
kn    ------ record3 { $0 }
        iu                  yu                  po                  lk                  jk
```

kl ------ record4 { $0 }
11 50 30 in in
in ------ record5 { $0 }
am as in
| | | |
| |
field1 field2 field3 field4 field5 field6
{ $1} { $2 } { $3 } { $4 } { $5 } { $6 }

SYSTEM VARIABLES IN awk COMMAND :

FS : field separater (by default white space b/w fields)
RS : record separater (by default new line b/w records)
NF : number of fields in table each record.
NR : number of records in a file.
OFS : output field separater (by default white space b/w fields)
ORS : output record separater (by default new line b/w records)

1. $awk '{ print NF } ' filename

2. $awk '{ print NR } ' filename

2. $awk '{ print $NF }' filename ----- It is used to display the last field in the file.

3. $awk ' { print NF, $0 }' filename ----- It display both number of fields and records data.

4. $awk '{ print $2,$3 }' filename ---- It display 2nd, 3rd fields in a file.

or

$awk -F " " '{ print $2,$3 } ' filename

5. $awk -F " " '/30/ { print $2 } ' filename ---- It display before value of '30' in the second field.

6. $awk -F " " '/30/ { print $2,$4 } ' filename ------ It display before value and after value of the '30'.

7. $awk -F " " '/30/ { print $2 "," $4 } ' filename ----- It display b/w two fields ' , '.

8. $awk -F " " '/30/ { print $2 "dileep" $4 } ' filename ----- It display b/w two fields ' dileep '.

9. $awk -F " " '$2 ~/30/ { print $4 } ' filename ----- It display that second field '30' having the exact value in fouth field.

10. $awk -F " " '$2 !~/30/ { print $4 } ' filename ----- It display that second field '30' not having the exact value in fouth field.

11. $awk -F " " '/^in/ { print $4}' filename

12. $awk -F " " '/am$/ { print $1}' filename

13. $awk '{ OFS="......"; print $2,$3 } ' filename

14. $awk '{ ORS="......"; print $0 } ' filename

15. $awk '{print $1+$2+$3}' filename

16. $awk -F"," '{print $1+$2+$3}' filename

AWK with operators :::

1. Arithmetic operators : + --- add , - ---- substract , * ---- multiplication , / ---- divizon , % ---- modulus

2. Logical operators : && ---- and , || ---- or , ! ---- not

3. Relational operators : < --- lessthan , > --- greaterthan , <= --- lessthanequalto , >= --- greaterthanequalto , == ---- equalsto , != ---- notequalsto

ex : $awk -F " " ' $2>30&&$3<50 { print $2,$3 } ' emp

$awk -F " " ' $2>30||$3<50 { print $2,$3 } ' emp

$awk -F " " ' $2>30!$3<50 { print $2,$3 } ' emp

Q. How do you print last character of the last field in the file ?

Ans : $awk '{print NF}' emp | tail -1 | rev | cut -c1

Q. How to list out only the job which are scheduled from 6am to 2pm ?

Ans : $crontab -l | awk '$2>=06&&$2<=14'

AWK with Scripting :

syntax : $awk ' BEGIN { action } ----- pre-processesing
actions ----- body
END { action }' filename ----- post-processesing

ex : cat>emp

```
101     1000    20    HR
102     2000    30    MA
103     3000    40    FM
104     4000    50    AC
```

ex : $awk ' BEGIN { print " eno sal dno desg " }
{ print $0}
END { print " report is generated\n....." } ' emp

o/p : eno sal dno desg

```
101     1000    20    HR
102     2000    30    MA
103     3000    40    FM
104     4000    50    AC
report is generated
.....
```

Backup commands ::: 1. ufsdump 2. tar 3. cpio

1. ufsdump :

a. Used for complete file system backup .
b. It copies every thing from regular files in a file system to special character and block device files.
c. It can work on mounted or unmounted file systems.

syntax : $ufsdump option source-directory destination-directory

ex : $ufsdump 0cvf /dev/rmt/0 /dev/rdsk/c0t0d0s0
or
$ufsdump 0cvf /dev/rmt/0 /usr

2. tar:

a. Used for single or multiple files backup .
b. Can't backup special character & block device files (0 byte files).
c. Works only on mounted file system.

ex : $tar cvf /dev/rmt/0

----> Viewing a tar backup on a tape using below command

ex : $tar tvf /dev/rmt/0

----> Extracting tar backup from the tape using command

ex : $tar xvf /dev/rmt/0

(Restoration will go to present directory or original backup path depending onrelative or absolute path names used for backup)

3. cpio:

a. Used for single or multiple files backup .
b. Can backup special character & block device files .
c. Works only on mounted file system.
d. Need a list of files to be backed up .
e. Preserve hard links and time stamps of the files .

---> Backup restore and disk copy with tar : Back up all the files in current directory to tape .

ex : $find . -depth -print | cpio -ovcB > /dev/rmt/0

---> cpio expects a list of files and find command provides the list , cpio hasto put these file on some destination and a > sign redirect these files to tape .
This can be a file as well .

----> Viewing cpio files on a tape using below command

ex : $cpio -ivtB < /dev/rmt/0

----> Restoring a cpio backup using below command

ex: $cpio -ivcB < /dev/rmt/0

Vi editor ::: Vi is online editor used to manipulate the data of a file. Vi is a command to open Vi editor.

syntax : $vi
---> We can perform following operators using Vi editor. such as :

1. Creating new files
2. Modifying existing file
3. To view the file

---> There are 3 different modes in Vi editor. such as:

A. Command mode (Default mode)
B. Input / Insert mode
C. Ex command mode

A. Command mode : In this all the keys are pressed by the user / interpreter to the editor command's.

---> In cmd mode, the keys that are hit cannot be displayed on the screen.

---> By default, when we open a file using vi editor it will be under command mode.

CMD MODE OPERATIONS / COMMANDS :

1. w(nw) ---- next word starting
2. e(ne) ---- word ending
3. b(nb) ---- word beginning
4. b ------ moves the cursor to the back to thefirst character of previous word
5. e ------ moves the cursor to the end of the current word
6. $ ---- end if the current line(end key)
7. ^ ---- beginning of the current line(home key)
8. H ---- beginning of the current page
9. M ---- middle of the current page
10. L ---- end of the current page
11. G ---- to go to the beginning of last line of a file
12. nG ---- n is any number.....ex : 1G, 2G,...To go to the beginning of nth lines of a file
13. R ---- to replace the text position from the cursor
14. ctrl+f ---- forward on page (pagedown)
15. ctrl+b ---- backward one (pageup)
16. x(nx) ---- Delete current character (del key)
17. X ---- Delete previous character (backspace key)
18. dw ------- to delete word at the current position
19. dw ------ to delete current word at the cursor position
20. ndw ----- to delete n words at the cursor position
21. dd ----- to delete current line
22. ndd ------to delete n lines
23. d$ ----- delete current position to end of the line
24. d^ ----- delete current position to beginning of the line
25. yw(nyw) ---- to copy a word
26. yy(nyy) ---- to copy a line
27. y$ -------- it copies current position to end of the line
28. y^ -------- it copies current position to beginning of the line
29. P ------- paste above the cursor
30. p ------- paste below the cursor
31. J ------- to join a line
32. 0 (zero) ------ to go to the beginning of current line.
33. cc ------ to clear a line
34. u ------- undo last cmd change

35. U ------ to undo all changes in current line
36. zz / wq ------ save & quit
37. q ----- without saving it executes

B. INSERT MODE ::: In this mode persson inserts of new text and editing new text / old text, replacement of existing data.

----> We need to come to insert mode from cmd mode to perform only operating. The following are the key's that are used to enter into isert mode :

1. i ------ to go to insert mode from cmd mode and to insert the data at the cursor position
2. I ----- to go to insert mode and to insert the data at beginning of the current line
3. A ------ to go to insert mode and it places cursor at end of the current line
4. a ------ to go to insert mode and it places cursor at right side of the cursor line
5. O ----- it inserts new line of the cursor
6. o ----- it inserts new line below of the cursor
7. ESC ----- is the key to shift from insert mode to command mode
8. : ----- is the command to shift to ex-command mode from command mode

C. EX-CMD MODE OPERATIONS / COMMANDS :

1. $:sh ----- Temparary return to the shell for executing unix commands.
2. $:w ----- to save changes without quite.
3. $:q ------ to quit the file after saving changes.
4. $:wq ------ to save and quit.
5. $:q! ------ to quit the file without saving.
6. $:! <unix command> ------ to execute unix commands without exiting from vi editor.
7. $:nd ------ to delete nth line.
7. $:$d ------ to delete last line.
8. $:m,nd ----- to delete lines from m to n.
9. $:m mo p ----- to move line m after p.

ex : $:5 mo 2

---> to move 5th line after 2nd line.

10. $:m,n mo p ----- to move lines m to n after p.
11. $:m co p ----- to copy line m after p.
12. $:m,n co p ----- to copy lines m to n after p.
13. $:/string/ ----- to search top to bottom.
14. $:?string? ----- to search bottom to top.
15. $:Set nu ----- set line numbers.
16. $:Set nonu ----- to remove line numbers.
17. $:$ ------ places cursor at last line in the file.
18. $:n ------ places cursor at nth line.
19. $:N ------- repeats last search cmd in opp direction.
20. $: starting line no, ending line no s/old string/new string/gi ----- search and replace string.

ex : $:1,$ s/unix/linux/gi
$:1,$ s/^/unix ---------- it adds 'unix' at beginning of each line.
$:1,$ s/$/; ---------- is adds ';' at end of each line.

Examples :

$:s/str1/str2 ----- replace the first occurrance of old string with new string at the line

where cursor is placed.

$:s/str1/str2/g ----- replace all occurancess old string with new string in the line where cursor is placed.

$:ns/str1/str2 ----- replace first occurance of old string with new string at nth line or the line.

$:ns/str1/str2/g ----- replace all occurance of old string with new string in nth line of the file.

$:m,ns/str1/str2/2 ----- replace 2nd occurance of old string with new string from lines m to n.

$:m,ns/str1/str2/g ------- replace all occurance of old string with new string from lines m to n.

$:1,$s/str1/str2/g ------ replace all occurances of old string with new string from 1st line to end of file.

$:1,.s/str1/str2/g ----- replace all occurance of old string with new string from 1st line to current line.

$:., s/str1/str2/g ----- replace all occurance of old string with new string from current line to end of file.

$:% s/str1/str2 ------ replace 1st occurance of old string with new string in all lines of the file.

$:% s/str1/str2/g ------ replace all occurance of old string with new string in all lines of the file.

Vi with multiple files :

syntax : $vi file1,file2, …. , filen

---> By using above command we can open 'n' number of files in different screens in vi editor.

---> When we open that by default the screen will be open with file1, we need to use ':n' to go to next file, we need to use ':prve' to come to previous file.

---> To display the filename when we are currently in we use ' :f '.

---> To display all the filenames which we opened, we can use ' :ARGS'.

Q. How to open a file by placing cursor in desired line ?

Ans : $vi +n filename

Q. How do u goto a particular line where the desired exit ?

Ans : $vi +/pattern filename

ex : $vi +/abc file1

---> The above cmd place ur cursor at the every first line where ' abc ' exited.

FINGER COMMAND :

---> In Unix, finger is a program you can use to find information about computer users. It usually lists the login name, the full name, and possibly other details
about the user you are fingering. These details may include the office location and phone number (if known), login time, idle time, time mail was last read,
and the user's plan and project files.

Syntax : $finger username@node.domain

Ex : finger dvader@mentor.cc.purdue.edu

---> You will get output similar to the following:

```
[mentor.cc.purdue.edu]
Login name: dvader                      In real life: Darth Vader
Directory: /home/mentor/d/dvader        Shell: /bin/csh
Last login Tue Jul 17 15:21 on ttyQ7 from expert.cc.purdue.edu
Unread mail since Wed Jul 18 13:00:54 2001
```

SAR COMMAND : sar collects reports or saves system activity information.

Q. Difference b/w Windows and Unix ?

Ans : UNIX : 1) Unix is a CLUI (Command Line User Interface) OS based.
2) Unix is the multi user operating system.
3) Command Based.
4) Unix is Free-source OS, you can Modify code of OS as per your business requirement.
5) Multi Processing.
6) Windows support plug n play.
7) In unix we can restrict the permission of each user.
8) unix file system in hierachical model.

WINDOWS : 1) Windows is a GUI OS Based.
2) windows is single user operating system.
3) Menu Based
4) Windows is licenced OS, It means you have to buy it.
5) No Multi Processing.
6) While Unix doesn't plug n play.
7) windows file system is flat type.
