
UNIX
Concepts and Applications

UNIX Reference

Contents

I. UNIX History

In **1957** Bell Labs found they needed an operating system for their computer center that at the time was running various batch jobs.

In **1965** Bell Labs adopted third generation computer equipment and decided to join forces with General Electric and MIT to create Multics (Multiplexed Information and Computing Service). 'GE' was owned by 'Thomas Alva Edison' and MIT is a research university.

- ➢ Bell Labs, frustrated by the size and complexity of Multics and the project was failed. Finally the code is kept in MIT University.

When Multics was withdrawn Ken Thompson and Dennis Ritchie needed to rewrite an operating system for DEC PDP-7 [Programmed Data Processor with 4K memory for user programs]. The result was a system called UNICS (UNiplexed Information and Computing Service)--an 'improved version for Multics'.

In **1972** Ritchie rewrote the language 'B' called 'C' and UNICS is rewritten in 'C' as UNIX. This resulted into portable software, requiring only a relatively small amount of machine-dependent code to be replaced when porting UNIX to other computing platforms.
Finally UNIX become an h/w independent OS and kept in MIT.

Different organizations started customizing UNIX and run on their systems.

- ➢ HP - HP-UX
- ➢ IBM - AIX
- ➢ SUN – Solaris

But UNIX itself was far more expensive. In quest of big money, the UNIX vendors priced it high enough to ensure small PC users stayed away from it.

Andrew S. Tanenbaum, a US-born Dutch professor who wanted to teach his students the inner workings of a real operating system. It was designed to run on the Intel 8086 microprocessors.
The first version of MINIX was written for the mainstream IBM PC and compatible personal computers.

In 1991, Linus Torvalds from MIT and converted MINIX to Monolithic kernel using Python, C and Perl for faster processing. This is the birth of Linux.
Linux is a high performance and completely free, Unix-like operating system that is suitable for use on a wide range of computers and other products.

II. Introduction to UNIX

1. Multiuser System

UNIX is a multiprogramming system, it permits multiple users to run separate jobs. The CPU, memory and hard disk are shared between users.

The system breaks up a unit of time into several segments and each user is allotted a segment. At any point of time, the machine will be doing a job of a single user. The moment the allocated time expires, previous will be kept in wait state and the next job is taken.

2. Multitasking System

A single user can also run multiple tasks concurrently. In multitasking environment, a user sees one job running in the foreground, the rest can be made run in the background.

For instance, as a file is getting edited, a parallel job can be initiated to print another file. Similarly many jobs are made to run simultaneously.

3. Building-Block Approach

Every individual command in UNIX will do a simple job. UNIX has the feature of giving an output of one command to another command.

This is achieved using '|' symbol. This way multiple commands are interconnected as building blocks together in a simple way to achieve complex tasks.

4. UNIX Toolkit

Kernel by itself doesn't do much to benefit the user. UNIX is quite diverse in scope, there are general-purpose tools, text manipulation utilities, compilers & interpreters, network applications and system administration tools.

You'll also have a choice of shells. For every UNIX release, new tools are being added and old ones are either get modified or removed.

5. Programming Facility

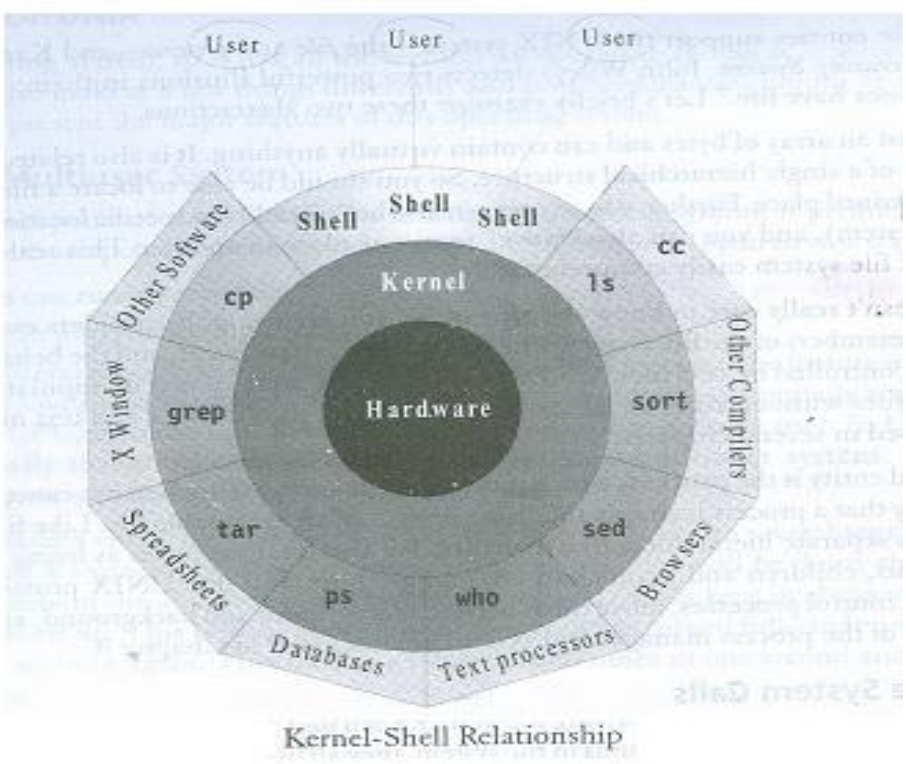
UNIX shell and awk are designed for programmers. It has all the necessary ingredients like control structures, loops and variables that establish it as a powerful programming language.

Shell scripts can invoke multiple commands together to form complex logics. Awk can be used for generating reports.

III. UNIX Architecture

There are two UNIX structure, Internal and External.

1. Internal Structure



Kernel-Shell Relationship

Kernel and Shell are the heart of UNIX. Kernel is the main program of the UNIX OS, which is directly linked with the hardware and the Shell is the user.

Kernel will be loaded into the memory when the system is booted and communicates directly with the hardware. Kernel is represented by the file '*/boot/vmlinuz*'

It manages the following functions:

- ***Driver Management***

  All device drivers are managed by the kernel. Any new hardware is connected to the system, will interact with the hardware through kernel.

- ***Process Management***

  It is responsible for forking child processes to balance multitasking and multiple users. If the application oracle is started, a parent process gets created initially. As multiple users gets connected, individual child

process gets created by the kernel for every user to process their requests.

➢ ***Memory Management***

It is mainly swapping memory pages between RAM and HDD using (LRU) Least Recently Used algorithm (i.e.) least used applications are swapped from RAM to Hard drive and brought back whenever required.

➢ ***File Management***

Kernel manages everything in UNIX as files and folders. These directories and subdirectories are organized into tree like structure called the file system.

In UNIX there are three types of files:

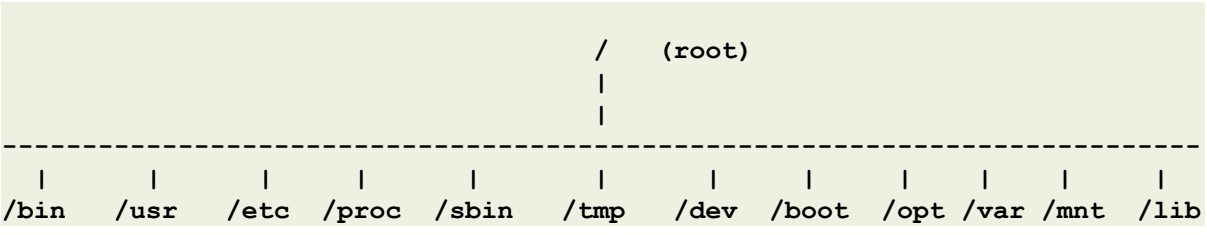
Ordinary Files: An ordinary file actually holds the user's data or a set of program instructions.
Directories: Directories store the user's files in a "folder" type of structure.
Special Files: Special files control access to certain types of hardware such as CD-ROM drives, Ethernet adapters. etc. These files are located in /dev directory.

Shell is a command interpreter, which is an interface between the user and the kernel. It mainly checks the syntax and semantics of UNIX commands. Even though there's only one kernel, there could be several shells, one for each user who is logged in.

Different types of shells available are sh (Bourne shell), csh (C shell), ksh (Korn shell) or bash (Bash shell). Execute '***echo $SHELL'*** to know the one that is running for you.

2. External Structure

File system hierarchy represents the external structure for UNIX. The files are stored as following:



Note:

➢ Accessing local machine -

a. ctrl + alt + F1 – F6 [CUI mode] (i.e.) command terminal
b. ctrl + alt +F7 [GUI mode] (i.e.) graphical terminal

- Accessing remote machine – **ssh**, **telnet**, **rlogin**, **rsync**, **FTP**, etc.,

/ - It is the parent directory for all other directories, called as ROOT directory. It is similar to "*c:*" of windows.

/root - It is the home directory for root user (super user). It provides working environment for root user.
It is similar to "*c:\Documents and Settings\Administrator*"

/home – It is the home directory for all the users and provides working environment for everyone other than root user.
It is similar to "*c:\Documents and Settings\username*"

/boot – It contains bootable files for UNIX like vmlinuz (Kernel), initrd (Initial RAM Disk), GRUB (Grand Unified Boot loader), boot.ini, etc.

/etc - It contains all configuration files like;
- /etc/passwd (user info)
- /etc/resolv.conf (DNS info)
- /etc/dhcpd.conf (DHCP server)

It is similar to "*c:\windows\system32\drivers*"

/usr - By default all softwares are installed in /usr directory (UNIX sharable resources). It is similar to "*c:\program files*"

/opt – It is optional directory for /usr. It contains all third party softwares.
It is similar to "*c:\program files*"

/bin - It contains commands used by all users (binary files)

/sbin - It contains commands used only by superuser (root user's binary files)

/dev - It contains device files information like;
- /dev/hda for hard disk
- /dev/cdrom for cdrom

/proc - It contains process files and their contents are not permanent, they keep changing. It is also called as virtual directory. The file contains useful information used by UNIX, like
- /proc/meminfo – info of RAM
- /proc/cpuinfo - info of CPU

/var - It contains variable data like mail (/var/spool/mail), log files, etc.

/mnt - It is the default mount point for any partition. It is empty by default.

/lib – It contains all library files used by OS. Kernel to h/w communication is

through drivers, whereas application to kernel communication will be through libraries.

IV. UNIX Basic Commands

1. Who

To find who are all the users logged in.

ex. *who* -> To see who is logged in,
who am i -> To know your logged information.

2. Manipulating directories

mkdir <dir_name> → To create a directory.
Mkdir <dir1> <dir2> <dir3> → To create multiple directories simultaneously.
Mkdir <dir2/file1.txt> <dir2/file2.txt> → To create multiple directories with the text file.
mkdir –p <parent_dir / child_dir> → To create a directory inside another directory.
ex. *mkdir –p one/two/three – This will create directory three inside two and directory two inside.*
rmdir <dir_name> → To remove a directory. (Directory gets removed only if it is empty)
rm –rf <dir_name> → Recursively and forcibly removes the directory.

3. Manipulating files

There are three ways to create files: touch, cat, using editors like vim, emacs, etc.

touch <file_name> → To create an empty file.
rm <file_name> → To remove a file.
rm –f <file_name> → To forcibly remove a file.
cat > <new_file_name>
Type some text …
Ctrl + d (to quit)
- File will be created with the text typed in.
cat <new_file_name> →To read the newly created file.
cat >> <new_file_name> → To append the contents of the newly created file.
cat –n file1.txt → To display the contents with line nos.

4. uname

To know the UNIX name.

ex. uname –a -> To provide all the details of UNIX

Note: ***Semantics of UNIX prompt***

[*user@server ~ | # | $]* where **~** represents the home directory,
represents admin user (root or superuser), **$** represents normal user.

5. cd – Change Directory

ex. cd /home → To go to the home directory.
cd . → To go one folder up.
cd .. / .. → To go two folders up.
cd - → To go to the previous location
cd (or) cd ~ → Takes you to the home directory

6. arch

It displays the architecture of the system.

Where, **i686** represents 32-bit
X86-64 represents 64-bit

7. clear

To clear the screen. (*ctrl + l*)

8. Coping / Moving files and folders

Copy syntax: *cp <source> <destination>*

ex. cp <file1.txt> /tmp → source current location and destination different location.
cp /tmp/<file1.txt> . → source different location and destination current location.
cp /tmp/<file1.txt> /var → Both src and dest. are in different location.
cp –r <src_dir> <dest_dir> → To copy a folder recursively.

Note: use **'-rf'** to forcibly copy when files inside directory don't have permission.

Move Syntax: *mv <source> <destination>*

ex. mv <src_dir> /tmp → source current location and destination different location.
mv /tmp/<src_dir> . → source different location and destination current location.
mv /tmp/<src_dir> /var → Both src and dest. are in different location.
mv * /opt/bckup → To move multiple files.
mv *.tar /tmp → To move group of files with tar extn. to different location.

Rename syntax: mv <old_name> <new_name>

ex. **mv backup backingup** → rename the directory in the current location
mv /opt/backingup /root/backup → Renaming in different src & dest.
Mv /opt/backingup backup → Renaming from different src to current location.

9. Redirection

The output of a command by default will go to the std. output. (i.e.) to the screen. We can also redirect the output to a file.

Ex. hostname > output.txt
Uname –a > output.txt → This will overwrite the content.
Uname –a >> output.txt → This will append the content.
Ls –ltr >> output.txt

10. ls - List Files

ex. **ls –l** <file_name> → To display elaborate info about the file.
ls –ld <dir_name> → To display elaborate info about the directory.
ls –lt → To display recently modified file on the top.
ls –lrt → To display recently modified at the bottom.
ls –la → To display all the files including hidden files. (starts with '.')
ls –l *.tar → Sorts and displays by group of tar files.
ls –l unix* → Displays by group of files with the starting matching pattern.

11. More and less

Drawback of cat is it will display only the lastpage when the output spawns more than one page.

Syntax: more <file_name> → Press 'Enter' to scroll line by line.
Press 'Space' to scroll page by page.
Press 'q' to exit.

Syntax: less <file_name> → Press 'b to scroll upward page by page.
Press 'y to scroll downward page by page.
Press 'q' to exit.

12. Head and tail

By default 'head' displays only first 10 lines of the file.

Syntax: head <file_name>

Ex: head <file_name>

Head <file_name > new_output.txt

Head –n 3 /etc/passwd > new_output.txt → Copy first 3 lines.

By default 'tail displays only first 10 lines of the file.

Syntax: tail <file_name>

Ex: tail <file_name>

tail <file_name > new_output.txt

tail –n 3 /etc/passwd > new_output.txt → Copy last 3 lines.

13. History

It will display the list of previously executed command. Using which we can refer what all the previous commands executed.

By default the no of commands executed is 100. If we customize value of the variable 'HIST_SIZE' in .profile file we can change the list of commands displayed in the output.

Syntax: history

Ex: history > hist_output.txt
History | less
History | more
History | grep –i head → To find the particular command from the history.

Ctrl + r - To search a particular command in the history.

We can also execute the command with the help of the history id.

Ex. If the o/p of the history command displays the following with history id;
100 pwd
101 ls –l
then we can execute !100 → It will execute the command pwd.

14. Command line Chaining

There are four types of command line chaining, **| , ; , && , ||**

| → First output command goes as input to the second command

Ex. tail –n 3 /etc/passwd | wc – w → This will count the words of last 3 lines.

cat /etc/passwd | wc → This will count the words of the entire file.

tail –n 3 /etc/passwd | wc –c → This will count the characters of the last three lines.

ls –ltr | more → This will the output of the command page by page.

; → Displays the output of all the command even if any of the command fails

Ex. hostname ; wc –l /etc/passwd ; uname –a ; date

&& → Displays the output of all the command until the first occurance of the failed command.

Ex. hostname && wc –l /etc/passwd && uname –a && date

|| → Displays the output of all the command until the first occurance of the correct command.

Ex. hostname || wc –l /etc/passwd || uname –a || date

15. Help command

Man → Manual (bible) information of the commands.

Synopsis: head [option] . . . [file_name]

Ex. man head

Head --help

Whatis head → It gives short information of the file.

16. Pwd

It displays the present working directory.

Ex: [sethu@EMEA root $] → It will display '/root'

[root@EMEA boot #] → It will display '/boot'

Note: $ - normal user # - root user

17. Wc

It count how many lines, words, characters.

Ex. wc /etc/passwd → Display the count of total words of the entire file.

Wc –l /etc/passwd → Displays the count of total lines of the entire file.

Wc –w /etc/passwd → Displays the count of total words of the entire file.

Wc –c /etc/passwd → Displays the count of total chars of the entire file.

18. Sort

By default it will sort the file content based on the first character in the ascending order.

Sorting precedence based on space first, number second, alphabet third.

Ex. sort test.txt → Sort the file in the ascending order.

Sort –r test.txt → Sort the file in the descending order.

Sort –n test.txt → It is a numeric sorting.

Sort –nr test.txt → It is a numeric sorting in the descending order.

Head -n || /etc/passwd || sort -r > new_test.txt → **?**

Sort -M test.txt → Sort the contents of the files monthwise.

Note: By default the delimiter is white space.

Sort –k2 testsort.txt → Sort based on the second column.

Sort -t '|' -k2 testing.txt → Delimiter is '|'

Sort -t '-' -k3 -M testing1.txt → 'Prepare a test for this command.'

testing.txt

ll|aa|ii
ee||bb|pp
ff|cc|nn
gg|hh|cc
rr|mm|oo
zz|uu|dd

19. Bc

It is a command of basic calculator.

Ex: bc
10 + 3 → o/p: 13
A=5 , B=10
$A + $B → o/p: 15

20. Echo

It will the command, which is used mainly in scripting.

Ex. echo "Welcome"

Echo –e "\t \t Welcome"

Echo –e "\n\t Welcome"

Echo 10 + 5 | bc

21. Tr

It will translate the case of the ouput of any command.

Ex. echo "welcome" | tr a-z A-Z → o/p: WELCOME

echo "WELCOME" | tr a-z A-Z → o/p: welcome

22. ls - List File

ex. ls –l <file_name> → To display elaborate info about the file.

ls –ld <dir_name> → To display elaborate info about the directory.
ls –lt → To display recently modified file on the top.
ls –lrt → To display recently modified at the bottom.
ls –la → To display all the files including hidden files. (starts with '.')
ls –l *.tar → Sorts and displays by group of tar files.
ls –l unix* → Displays by group of files with the starting matching pattern.
ls –I → To list the inode numbers

Note: The o/p of 'll' shows following information.

I column - File type either file(-) or directory (d)
II column - Permissions (rwx rwx rwx)
III column - Links
IV column - owner of the file
V column - owner's group name.
VI column - size of file in bytes.
VII column - modification date and time of the file or directory.
VII column - file or directory name.

23. File

To identify the type of the data located in the file.

Syntax: file <file_name>
Ex. file ymc.tar.bz2
Cd /lib
File libuid.so.1.2 → Displays whether 32-bit / 64-bit

24. Date

To display the current date.

Ex. date + %A → Displays on the day. For instance 'Tuesday'

date + %B → Displays on the day and month. For instance 'May Tuesday'

date + %a → Displays on the day in short form. For instance 'Tue'

date + %b → Displays on the day and month in short form. For instance 'May Tue'

date + %e → Displays on the day of the month. For instance '27 date of month'

date + %m → Displays the month of the year. For instance '05 (month)'

date + %d → Displays the day of the month. For instance '27 (date)'

date + %y → Displays the year of the century. For instance 'year 14'

date + %F → Displays complete date format. For instance '2014-05-27'

25. VI EDITOR

Vim → Visual display editor improved.

This is a command mode editor for files. Other editors are emacs, gedit, nano etc.

'vi' is most popular than other editors and it is having 3 modes:

- Command mode
- Insert mode
- Extended command mode

Insert mode:

i - Insert at current cursor position
I - Insert at start of the line
a - Append at current cursor position
A - Append at the end of the line.
o - Insert line below cursor position.
O - Insert line above the cursor position.

Note: Ins – key is same as i

Command mode:

:w - To save the file.
:wq - To save and quit.
:q - To quit without saving.
:q! - To quit without saving forcefully.
: wq! – To save and quit forcefully.
: se nu – To set the line number
: se nonu – To remove line number.
: 14 – To move the cursor to line no 14.

Extended Command mode:

Esc + dd – To delete a line.
Esc + dw – To delete the current word.
Esc + 5dd – To delete 5 lines from the current position.
Esc + yy – To yank (copy) a line.
Esc + 10yy – To yank (copy) 10 lines.
Esc + p – To paste lines above the cursor position.
Esc + P – To paste lines below the cursor position.
Esc + a – To go to next character.
Esc + A – To go to the end of the line with the insert mode.

To find and replace:

Esc + r – To replace the current character.

Esc + R – To replace all the characters from current position.

: <range> s / <find_what> / <replace_with> / <options>

Where, range - % is complete file.

10,$ - From 10th line to last line.

15,20 – From 15th line to 20th line.

Options – 'g' To replace all the occurance in the same line.

'i' To ignore the case sensitivity.

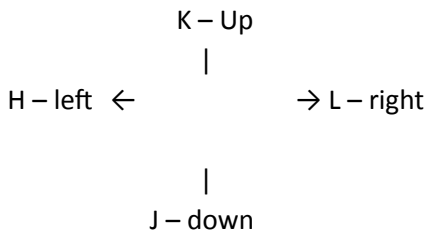
Ex. : 1,$s / cat / dog / gi

To find 'cat | CAT | Cat | cAT ' and replace with 'dog' in the entire file.

Esc + u – To undo changes step by step.

Esc + wb – To move a word backward.

Navigation:



26. Grep and egrep

To search a particular matching pattern from the entire content of the file.

Syntax: grep – options "matching pattern" <file_name>

Ex. #cat /etc/passwd

#grep "root" /etc/passwd

greptest.txt → Use the below text file to practice the grep example commands.

UNIX is an OS
OS is liux
OS is liux
Linux is OS liux

LiUX is very secure OS
Liux is an operating system
Liux is an operation system
Going to learn liux
Going to learn liux
Liux hello unix
Liux hello unix
Bye
CHENNAI is very hot
Test the file with grep command
Ibm class 2014
Unix is cli based OS
Grep testing
Ibm training at Pune
LIUX is very secure OS
CLI based OS.

grep "LIUX" greptest
grep -i "LIUX" greptest → To ignore the case.
grep -i "LIUX" greptest | grep "learn" | wc –l
grep -v "LIUX" → Inward search. (i.e.) Display lines except matched LIUX.
grep –c "liux" greptest
grep "linux" *
grep –l "liux" *
grep –l "liux" greptest /etc/passwd
grep –l "root" greptest /etc/passwd
grep "^liux" greptest /etc/passwd → Display contents starting with liux.
grep "OS$" greptest /etc/passwd → Display contents ends with OS.
ls –l | grep "^-" | wc –l > file_list_only.txt → Display only files.
ls –l | grep "^d" | wc –l > dir_list_only.txt → Display only directories.
ls –l | grep "^l" > symbolic_link_only.txt → Display only the symbolic
grep "venky" /etc/passwd → To display lines partially matching venky.
grep –w "venky" /etc/passwd → To search a particular word.
grep –A 2 -w "venky" /etc/passwd → Display above two lines from the matched word.
grep –n -A 2 -w "venky" /etc/passwd → Display above two lines from the matched word.
grep –n -B 2 -w "venky" /etc/passwd
grep –n -C 2 -w "venky" /etc/passwd -> Display above and below two lines of the matched content.
grep [0-9] preparetest → Prepare a test file for testing below commands.
grep –l [0-9] preparetest
grep * [0-9]* preparetest
grep "an*.*system" preparetest
grep –C "an*.*system" preparetest
grep –n "an*.*system" preparetest

Egrep:

To search multiple matching pattern in the file content.

Syntax: egrep“<pattern_1> | <pattern_2> | . . . | <pattern_n>” <file_name>

Ex. #egrep “root | venky” /etc/passwd
#egrep “indhu | sethu | madhavan | nanda | leela” /etc/passwd.

27. Find

To find the matching pattern across the file system.

Syntax: find <path> <options> <word_to_find>

Ex. find / -name “passwd”

find / -iname “passwd” → Searches with case insensitive.

Keywords:

-name - Search only particular file for the given name.

-iname - Search the file/folder ignoring the case.

-type - Searches either in the file or in the folder, specified in the type.

-empty – Searches only the empty files and folders.

-user – Searches only the particular user.

-perm – Searches files and directories w.r.t their permissions.

-exec - To execute any particular command after the search.

-atime – To search files and folders w.r.t the accessed time.

-mtime – To search files and folders w.r.t the modified time.

-amin – To search files and folders w.r.t the accessed minute.

-mmin – To search files and folders w.r.t the accessed minute.

-size - To search the files and folders w.r.t size wise.

Ex. # find / -name passwd

find / -iname passwd

find –type f –iname “passwd” → To search only the files.

find –type d –iname “passwd” → To search only the directories.

find –empty /opt → To search only the empty files under /opt

find –type f -empty /opt → Search only the empty files under /opt

Find –type d -empty /opt → Search only the empty directories under /opt

Find –user "venky" /home → Search only the user arun's file under /home

Find / -type f -user "venky"

Find / -type d -user "sethu"

Find /opt -perm 000 → To find files and folders with no permission.

Find /opt -perm 777 → To find files/folders with maximum permission.

Find /opt -perm 777 -exec ls –l {} \;

Find /opt -perm 777 -exec ls –ld {} \;

Find /opt -empty -type f -exec ls –l {} \;

Find /opt -empty -type d -exec ls –l {} \;

Find /opt -empty -type f -exec ls –ltr {} \;

Find /opt -type f -atime -2

Find /opt -type f -atime -2 -exec ls –ld {}\;

Find /root -atime -2

Find /opt -mtime -2

Find /root -admin -2

Find /root -mmin -2

Find /root -amin -60

Find /opt -size +1M → To display file greater than 1MB

Find /opt -type -d -size +1M → For directories

Find /opt -size -M -exec du –sh * {}\;

Find /opt -size -1M -exec ls –l * {}\;

Find /opt -size -1M -exec ls –ld * {}\;

28. sleep

This will cause a delay between commands. It will be mainly used in scripting.

Syntax: sleep n → 'n' represents the time either in minute (or) hour (or) day.

Ex. # <command_1 >: sleep n : <command_2>

ls –ltr : sleep –m : wc –l >> capture_output.txt

This will wait for a minute after the execution of first command and count the line. Finally the output will go into the desired text.

sleep -1m → one minute.

sleep -1h → one hour.

sleep -d → one day.

29. stat

To show the file status.

Syntax: #stat

Following are the semantics of the output:

File – 'commands'

Size – Size of the file / directory (176)

Blocks - Memory block (size occupied)

Ioblock - (4096)

Regurlar file – File type either regular file or directory or device file.

Inode - Numeric identification of the file (1471954)

Links - no of the links of the file or the folder.

Access – (064 / rw- r—r--)

UID - (0/root) → User

GID - (0/root) → Group

Modify –

Change -

30. du – Disk Usage

It is used to identify the size of the file.

Syntax: du -s -h → s - summary output and h - human readable format.

Ex. # du –sh ymc.tar.bz2

du –sh *

du –s ymc.tar.bz2 | sort -nr | -l

du –sh * | sort -h | head -n 2

du -sh /opt/* | sort –nr | tail –n 2

31. df - Disk Free

To display the current position of the server.

Syntax: df -h

Note: Partition convention

In Windows → c: , d: , etc.

In Linux → SATA IDE

SATA	IDE	
Sda	hda	→ Primary master.
Sdb	hdb	→ Primary slave.
Sdc	hdc	→ Secondary master.
Sdd	hdd	→ Secondary slave.

Where IDE is old hard drive and SATA / SCSI are new hard drive.
Mandatory partition for UNIX are / , /boot and /swap.

32. Diff

It will show the difference between the files.

Syntax: diff <file_1> <file_2>

O/P: < First file first difference
< First file second difference.

> Second file first difference.
> Second file second difference.

Practice: Check the difference between zip and tar files.

Ex. # cat > difftest1.txt →
1
2
3
4
ctrl + d

cat > difftest2.txt →
4
3
6
7
ctrl + d

diff difftest1.txt difftest2.txt

33. Passwd

To change the existing password of an user.

Syntax: passwd <user_name>

Ex.

passwd -l <user_name> → To lock the password of the user.
passwd -u <user_name> → To unlock the password of the user.
passwd –d <user_name → To remove the password of the user.
above command is similar to removing the

password in /etc/shadow.

passwd "venky"
Enter password: <enter new password>
Confirm password: <re-enter new password>

34. su – Switch User

It will switch between user login.

Syntax: su - <user_name>

Ex. # su sethu → It will switch into sethu's login. If you and the user are in the same group then, you have the permission for all the files / folders common to that particular group.

35. Id

It displays the uid (user id) and gid (group id) of all the users.

Syntax: id

Ex: # id <user_name> → It will display the uid and gid of that particular user.

36. Free

It displays only memory related information.

Syntax: free <options>

Ex: # free –m → Displays the free memory currently running in the server.

37. last and lastb

last command will display last login commands. By default the no of commands displayed will be 10. We can use different options to alter the display of the command.

Syntax: last <options>

Ex: # last –n where 'n' no of logins to be displayed.

last – 5 → displays last 5 logins

lastb command will display last bad login commands. By default the no of commands displayed will be 10. We can use different options to alter the display of the command.

Syntax: last <options>

Ex: # lastb –n where 'n' no of bad logins to be displayed.

last – 5 → displays last 5 bad logins.

38. vmstat

It will monitors the current memory and CPU usage of the system. It is similar to the 'top' command.

Syntax: vmstat <options>

39. netstat

It will display the port and network services currently running in the server.

Syntax: netstat <options>

Ex. # netstat -ntlp

40. locate

It is used for partial find of the search.

Syntax: locate <file_name>

Ex: # touch Indhu.txt sethu.txt Madhavan.txt

updatedb → It will update the locate database.

locate Indhu

locate sethu

locate madhavan

41. tty

It will display the terminal type.

Syntax: tty

Ex: # tty - Displays the current terminal type

echo "Welcome" > /dev/pts/2 → Displays the output only for that particular terminal.

Note : pts - pseudo terminal (represents GUI)
tty - represents command terminal.

42. Cut

It is similar to awk and sed command but some additional functionality. This command will cut the contents w.r.t the options provided.

Syntax cut <options> <file_name>

cut –c1 greptest → It will cut the first character of each line in the file.

ls –ltr | cut –c1 | grep "d" | wc –l → What is the output ?

cut -c1,2 greptest (first and second character of each line in the file)

cut -c1,2,3 greptest (first,second and third character of each line in the file)

cut –c1 – 3 greptest (first and third character of each line in the file)

where , is next to next and - is upto

cut -c6- greptest (From character 6 till end of each line in the file)

cut -f1 greptest (First field of each line in the file)

cut –d ' ' -f1 greptest (First field identified by the delimiter space)

cut –d ' ' -f3-6 greptest

cut –d ' ' -f6- greptest

43. Sequence

It generates the sequence number. It will be used to create a test files with different size.

Syntax: seq <number> > <file_name>

seq 1000 → Generates 1 to 1000 numbers

#seq 0 2 100 → Generates numbers from 0 to 100 in multiples of 2.

It will be used to create test files upto GB.

44. ln – Link / Unlink

There are two types of links Hard link and Soft link. Links are mainly to create pipe line between two physical location.

Symbolic link:

This will create a reference which is similar to shortcuts in windows.

Syntax: ln –s <source_file_path> <link_file_path>

Ex. # ln -s /home/new /opt/test

If the file is a symbolic link, the first character in the information is 'l'

(i.e.) after the above command, if we execute 'ls –l /opt/test' under /opt following output would display:

o/p: l rw- r-x r-x

To unlink: unlink <link_file_path>

Ex. # ulink /opt/test

Advantages:

If we are running a application for instance under /opt/app1 and unfortunately we don't have much space under that partition for the application to get executed everyday.
Under these circumstances, we can create a symbolic link to another location where the application pointing to that, then the data will be accumulated in different partition for the application finally to get executed.

Note: In case of symbolic link, if the source gets deleted then the link also will get deleted.

Hard link:

This will create a copy of the source file. It will be similar to backups.

Syntax: ln <source_file_path> <link_file_path>

Ex. # ln /home/new /opt/test

To unlink: ulink <link_file_path>

Ex. # ulink /opt/test

Note: In of hard link, the link file exists even if the source file gets deleted.
If a file has more than one link count that means it has a hard link.

Hard link	Soft link
Both the size of the file will be same.	The size of the link file will be equal to the no. of characters in the name of the original file.
It cannot be created across partition.	It can be created across the partition.
Inode number of the both the file is same.	Inode no of the source and link file is different.
If the original file is deleted then also the link file will contain the data.	If the original file is deleted then the link file is broken and the data is lost.
It is a '***BACKUP***' file.	It is a '***SHORTCUT***' file.

V. Archieve and Compression

The advantage of archiving and compression is that, when we transfer large files across network, we can reduce the bandwidth and time consumption.

Through compression we can reduce the bandwidth consumption when we transfer files across network.

Through archiving we can pack multiples files together and transfer between remote machines.

1. Zip

This is a compression utility.

Syntax: zip <options> <destination.zip> <source_files>

Ex.

zip –r newzipfile.zip file1.txt file2.txt file3.txt

unzip newzipfile.zip

2. Gzip

This is another utility for compression preferred more than zip.

Syntax: tar <options> <destination.tar> <source_files>

Ex. # tar –cvf mynewbkp.tar other.xml txt_file.txt data.csv

, where c-create, v-verbose, f-mentioning file name

du –sh mynewbkp.tar o/p: 38M

gzip mynewbkp.tar o/p: mynewbkp.tar.gz

du –sh mynewbkp.tar.gz o/p: 9.2 M

3. Gunzip

This is used to for decompression.

Syntax: gunzip <source.tar.gz>

Ex: # gunzip mynewbkp.tar.gz o/p: mynewbkp.tar

tar –tvf mynewbkp.tar
, where t-For viewing

tar –xvf mynewbkp.tar o/p: other.xml, txt_file.txt, data.csv
, where x-extract

tar –cvzf newbkp.tar.gz other.xml txt_file.txt data.csv

tar –xvzf newbkp.tar.gz
, where x-extract z-compress and archieve.

4. bzip2

This another compression utility.

Syntax: bzip2 <file_name>

Ex. # tar –cvf mynewbkp.tar other.xml txt_file.txt data.csv

bzip2 mynewbkp.tar .bz2 o/p: mynewbkp.tar.gz

du –sh mynewbkp.tar.bz2 o/p: 6.5 M

5. bunzip

This is used to for decompression.

Syntax: gunzip <source.tar.gz>

Ex: # bunzip mynewbkp.tar.bz2 o/p: mynewbkp.tar

bzcat mynewbkp.tar.bz2 → To view

du –sh mynewbkp.tar.bz2 o/p: 6.5M

tar –xvf mynewbkp.tar o/p: other.xml, txt_file.txt, data.csv

The times are in mm:ss and the size is in KB for compressing 2.5 GB files.

	comp time	comp. size	decomp time
gzip	14:31	349,736	0:55
bzip2	39:44	275,344	9:46

Yet, BZIP2 is an open source lossless data compression algorithm that makes it possible to retrieve the original data of a compressed file.

VI. File Permission

1. Umask

It is used to modify the default permission of a file and directory.
The default umask of the root user is 022.
The default umask of the normal user is 002.

Umask	File	Directory
0	Rw	rwx
1	Rw	rw
2	R	x
3	R	r
4	W	wx
5	W	w
6	Null	x
7	Null	null

Ex:

	File	Dir
Default permission	666	777
Subtract	022	022
Result permission	644	755

__ directory - d rwx rwx rwx
Umask 000 --|__ file - - rw- rw- rw-

__ directory - drwx r-x r-x
Umask 022 --|__ file - -rw- r-- r—

__ directory - drw- r-- r--
Umask 0033 --|__ file - -rwx r-- r—

Umask **0**022 → Sticky bit used to set gid, uid. Mainly used by sysadmin.

2. Chmod

To see the current permission of any file.

Syntax: ls –ld <file_name>

Permissions are applied on three levels;

- Owner (or) user level
- Group level
- Others level

Access modes are of three types;

- r – read only
- w – write / edit / delete / append
- x – execute or run a command

	File	Directory
r	To open a file.	To list the contents 'ls'
w	To write / edit / delete / append the file.	To add / delete / rename contents of the directory.
x	To execute a command in the shell script.	To enter into the directory 'cd'

The o/p of the command 'ls –ld'

```
-    rw- r- - r- -  2  root  root   54  15  march . . .   file.txt
1      2        3    4     5      6   7     8              9
```

where,

1 - File type '-' for file and 'd' for directory.
2 - Links
3 - Owner
4 - Group name
5 - Size in bytes
6 -
7 - Modification date
8 - File name

File types
- - normal file
d - directory
l - Link file
b - block file (hard disk, cdrom)
c - Character file (keyboard, mouse)

Permission can be set on any file or directory by any of the following two methods:

- Numeric method (absolute method)
- Character method (Symbolic method)

to set the permission on files and folders '***chmod'*** command is used.

By default the permission of the file and folders are:

	File	Dir
root (superuser)	644	755
normal user	644	775

where, read – 4
execute – 1

To set the permission on file and folder:
Syntax: chmod <permission> <file/folder_name>

Ex1. # chmod 777 /file1.txt (absolute method)
chmod u=rwx, g=rwx, o=rwx /file1.txt (symbolic method)
(or)
chmod ugo=rwx /file1.txt

Ex2. # chmod 000 /file2.txt (absolute method)
chmod u=-, g=-, o=- /file2.txt (symbolic method)
(or)
chmod ugo=- /file2.txt

Ex3. # chmod –R a+rwx /dir_1 → Assigning permission recursively into all files and folders under the directory '**dir_1'**.

Changing the group ownership

Ex4. # chgrp unix share
Changes the group share to unix.

Advanced permission:

There are 3 types of advance permission:

- SUID
- SGID
- Stickybit

To set advance permission by using numeric method we use 'a digit' before the permission.
For instance before 666 use 1666.

/var/gdm contains the sticky bit (gnome display manager)
suid = 4, sgid = 2, stickybit = 1

SUID : If suid is set on any command then any normal user can run the command with the privileges of root user.
Ex. : By default 'ping' command is having the suid. So all the users can run this

command, but if suid is removed then the permission is denied.

whereis ping o/p: /bin/ping
ls –ld ping o/p: -rwsr-xr-x → SUID is set by default.

To remove SUID:

chmod **0**755 /bin/ping
ls -ld ping o/p: -rwxr-xr-x → SUID is removed.

Sticky bit : If sticky bit is set, then only owner can delete the file and folder.

Ex. : # mkdir /mywork
chmod **1**777
ls –ld /mywork

su venky
touch /mywork/file1.txt
su root
rm /mywork/file1.txt → Even a root user cannot delete the file1.txt.

SGID : sgid is used for group inheritance, files and directories will get from their parent directory.

Ex. :
mkdir /mywork
chmod 777 /mywork
chmod **2**777 /mywork

Now login as emea group member and create some files, then check their properties.

To remove SUID:

chmod 0755 /bin/ping
ls -ld ping o/p: -rwxr-xr-x → SUID is removed.

Disadvantage:

1. Unable to give permission to a specific person in the group.
2. We cannot assign individual permission using '***chmod'*** command.

3. Acl

Using acl we can assign individual permission.

Syntax: setfacl -m u:<user_name>:r <file_name>

Ex. # setfacl -m u:leela:r acl_test.txt

ls –ld → o/p: - --- r-- ---+ → indicates acl.

To find whom the acl permission is assigned.

Syntax: getfacl <file_name>

Ex. # getfacl acl_test.txt

Note: If you have the read permission then only,

r+x - only then you can go into the folder.

r+w - only then you can read what you write.

Ex. # setfacl -x u:venky test_acl.txt → To remove permission using ACL.

getfacl → To list the permissions set using ACL.

setfacl -m u:leela:r test_acl → To modify the user permission.

setfacl -R -m u:sethu:rwx <dir_name> → Recursively assign using ACL.

4. chattr

To prevent accidently deleting a file.

Ex. # touch tmp
chattr +i timp → To add the attribute.
rm –rf tmp → This will not permit the operation.

To check the attribute:

Ex. # lsattr tmp
chattr -i tmp → To remove the attribute

lsattr tmp o/p: - --- --- --- tmp

VII. User Management

When a new is created in UNIX,

his home directory is created in '/home/<user_name>'
his mailbox directory is created in '/var/spool/mail/<user_name>'
Unique UID (User id) and GID (Group id) are given to the user.

UID for system users (0 to 499)
UID for normal users (500 to 60000)

RHEL, Fedora uses UPG scheme.
UPG → User Private Group, it means whenever a user is created, he has his own private group.

1. Useradd

To add a user;

Syntax: useradd <option> <user_name>

Options are,
- -u - user id
- -G - secondary group id
- -g - primary group id
- -d - home directory
- -c - comment
- -s - shell

Ex. # useradd sethu → sethu user will be created.
useradd -u 1001 -d /usr/madhavan -s /bin/bash Madhavan.
- User Madhavan will be created with **uid**=1001, **home** = /usr/madhavan and default **shell** = /bin/bash

All the user information will be stored in /etc/passwd file. It contains 7 fields.

Ex. **Madhavan:x:1001:5000::/usr/Madhavan:/bin/bash**
where, first field – user login name.
second field – mask management.
third field – uid.
fourth field – gid.
fifth field – comments. (by default no comments)
sixth field – user's home directory.
Seventh field – user's login shell.

After we create an user, we can modify its properties by using:
Syntax: usermod <options> <username>
The options are all the one used for useradd, and include the following:
- -l - to change the login name.
- -L - to lock the account.
- -U - to unlock the account.

To add an user as a secondary member of a group:
Syntax: usermod -G <group_name> <user_name>
Provided the user and group should exist.
Note: When a account is locked, it will show ! (exclamation mark) in /etc/shadow file.

To set / modify a password for an user:

Syntax: passwd <user_name>

Ex1: passwd sethu

Command would display the following;

Enter password: <enter the desired password>

Re-enter password: <Re-enter the desired password>

The password information will be stored inside '/etc/shadow' file. It contains encrypted password. UNIX uses MD5 (Message Digest Version 5)and DES (Data Encryption Standard) algorithms for encrypting.

2. Userdel

To delete a user;

Syntax: userdel <option> <user_name>

Ex: # userdel venky

It will delete the user "venky", but the home directory will not be deleted.

userdel –r venky

It will delete the user "venky" and as well the home directory.

3. Groupadd

To create a group.

Syntax: groupadd < option> <group_name>

where -g to set GID.

Ex: # groupadd -g 1010 indhu

groupadd emea

groupmod -n cortex emea → It will modify the existing group name.

To change the group name.

Syntax: groupmod –g <gid> <group_name>

Ex. # groupmod –n 2010 indhu

4. Groupdel

To delete a group. The group must not contain any primary user.

Syntax: groupdel <group_name>

Ex. # groupdel mktg → This will delete the group marketing.

All the information of a group will be stored in the file /etc/group. It contains the list of secondary members also.

To add/delete secondary users to a group.

Syntax: gpasswd <option> <user> <group_name>

Ex. # gpasswd -a venky emea → Adds a user "venky" in the group emea.

gpasswd -M Indhu sethu leela emea → Adds multiple user under emea.

gpasswd -d venky emea → Deletes a secondary user from the group emea.

VIII. Remote Connection Management

1. ssh – Secured Shell

ssh is mainly used to access remote server from your local machine.

Syntax: ssh <ipaddress> <portno>

Ex. ssh 192.168.1.1 22 → 22 is the default port for ssh

The above command establishes a remote connection from your local machine to the server 192.168.1.1. (i.e.) The server's shell gets opened into your local system, what ever command you execute, that will get executed in the server system.

2. telnet

It is a non-secure connection, mainly used for port listening.

Port Listening:

Use the port number to check whether a particular service is enabled.

Syntax: telnet <ip_address> <port_no> → Default port no is 23

Press ctrl +] → to come out.

3. scp – Secured Copy

To copy files/folders between remote machines in a secured manner. (i.e.) the data transferred across network are encrypted using MD5 and DES algorithm.

Syntax: scp <source_file> <user_name>@<ip_address>:<destination>

Ex. # scp /tmp/data.csv user3@192.168.1.1:/opt

The above command copy the data.csv file from the local machine into the server 192.168.1.1 in the path /opt.

scp user3@192.168.1.1:/opt/data.csv /tmp → reverse copy

Note: The speed of copying small files is relatively high compared to others.

Disadvantage:
When the copy is interrupted while it has completed copying 90% of the file, it will restart copying all over again. This will be more costly between networks

Another disadvantage is if we want copy file and update it every day, scp will overwrite everything from beginning.

4. rsync – Remote Synchronization

To overcome the above disadvantages, rsync is preferred.

Syntax: rsync <options> <file_name> <user_name>@<ip_address>:<dest_path>

Ex. # rsync -av test_file.txt user3@192.168.1.1:/tmp
,where a-all, v-verbose

cat >> test_file.txt
type some text and press ctrl +d.

rsync -av test_file.txt user3@192.168.1.1:/tmp

ssh 192.168.1.1
cd /tmp
cat test_file.txt
- modified contents will be displayed.

5. Ftp – File Transfer Protocol

It will be used to upload and download files across remote machine.

Syntax: ftp <login>@<ip_address>

Ex. # ftp root@192.168.1.1

ftp> ls –l

ftp> get ymc.tar.bz2 → to get the remote file into the local machine.

ftp> mget file1.txt file2.txt file3.txt → to get multiple files.

ftp> put local_file.csv → to put local file into the remote machine.

ftp> mput file1.txt file2.txt file3.txt → to put multiple files.

IX. Profile Management

1. .bash_history

It displays history level commands. (i.e) previously used commands. User level history command gets the data from this file to display the output.

2. .bash_logout

It contains the login and logout information of all the users. All the login information of the normal user as well root user's would get displayed on executing this command.

3. .bash_profile

There are two types of variables:

- ➢ Local variable
- ➢ System variable

Local variable:

Ex. name = Ahila
Echo $name displays "Ahila"

The disadvantage is that the life of this variable will be only for that session. (local variables are always in lowercase)

System variable:

These variables are stored in the system permanently, (i.e.) they will be alive for all the sessions. (system variables are always in uppercase)

Ex. HOST, USER, LS_COLORS, etc .,

All the system variables are defined in bash_profile file.

Ex. # echo $PATH → Displays the path of the environment variable.
echo $HIST_SIZE → Displays the size of the history command.
(i.e.) If the value is 500, then only 500 history commands will get displayed.

vi .bash_profile
HIST_SIZE = 1000
source .bash_profile → to re-read the file, then only the changes gets reflected.

To create a directory and set its path in the environment variable.

mkdir /project1/oracle/bin
PATH = / project1/oracle/bin
export PATH

The correct way is:
vi .bash_profile
PATH = / project1/oracle/bin:$PATH
export PATH
source .bash_profile → To re-read the file.

Note:

Modifying the user setting in the command mode will modify values temporarily.
To modify values permanently edit the values in .bash_profiles.

4. .bashrc

This file is mainly used to store the alias for the command name.

Syntax: alias <alias_name> = <command_name>

Ex. # alias sys = hostname

To store it permanently, edit this value in the .bash_rc file and re-read the file using the command 'source'. (i.e.) source .bash_rc

Ex. # vi .bash_rc

alias l = ls –lrt

Advantage:

Every when we have a routine job to open a set of server connectivity whose host name is very lengthy. This might error prone as well. In such case alias would come very handy.

X. Process Management

A process is an application which is currently running in the system. In unix all the process are maintained in tree like structure as parent, child process.

Unix has few utilities to monitor the process that are currently running in the system. It is similar to the task manager in windows.

Example for Parent / Child process is:

For instance an oracle application is started in the server, it will be running a parent process. Then many child processes to that parent will be started as many users starts logging in to the application. Each child process is responsible for serving all the individual users.

1. top

This command is used to monitor the currently running process.
The o/p has the following parameters:

PID – process id.
User – owner who started the process.
PR – priority.
NI – Performance related parameter.
VIRT – Virtual memory.
SHR - Information about the process killed and shared for the new process.
%CPU – CPU usage.
%MEM – memory usage.
TIME+ - Time the process is running.
CMD – The command name that has started the process.

Using top we can monitor the task that is started, out of which how many are sleeping / running etc.
The CPU and memory usage as well the load average.

Syntax: top <options> <user_name>

Ex. # top –u <user_name> → To display all the process of the particular user.

top → Display all the process currently running in the system.

Usually the priority of a process range between -19 to +19.

You can change the priority using the following command:

Syntax: renice <old_priority> to <new_priority>

Ex. # renice -19 to +19

2. ps

It is similar to top, but it will get the static data of what top is giving. This is mainly used in scripting.

Syntax: ps <options> <application>

Ex. # ps –ef firefox

pstree → Displays the process in tree like structure.

Note: Zombie is child process running in the system, even when the parent process is killed. This will exhaust the memory.

3. Kill

This command is used to stop a particular process while it is running.

Syntax: kill <options> <process_name>

Ex. kill -9 firefox

-9 → Option to wipe the ram memory space so that the process gets killed immediately.

XI. Advanced Unix Commands

1. awk

It is field processor. It is similar to grep, which will find the matched pattern line by line, where as awk will find the matched pattern column wise.

It is mainly used in report generation.

Syntax: awk /matching/ '{action}' filename

Ex. # awk '{ print $0 }' greptest → It will print all the columns.

By default the delimiter is space.

awk '{ print $1 }' greptest | grep "ibm" | wc –w → o/p: ?

awk '{ print $2 }' greptest

awk '{ print $1,$2 }' greptest

awk '{ print $1,"--",$2 }' greptest

Where $1, $2 …. $n → Field 1, Field2, … Fieldn of the text with delimiter is space.

awk –F':' '{ $1 ~/root/ print $0 }' /etc/passwd
awk –F':' '{ $1 ~/root/ print $5 }' /etc/passwd
awk –F':' '{ $1 ~/root/ print $6 }' /etc/passwd
awk –F':' '{ $1 ~/root/ print $7 }' /etc/passwd

Note: Please refer to the test file greptest.txt given in the beginning of the document.

2. Sed

A common use of sed is to modify each line of a file or stream by replacing specified parts of the line. For example if you have a file that contains the lines:

1, Justin Timberlake, Title 545, Price **$6.30**
2, Taylor Swift, Title 723, Price $7.90
3, Mick Jagger, Title 610, Price $7.90
4, Lady Gaga, Title 118, Price $6.30
5, Johnny Cash, Title 482, Price $6.50
6, Elvis Presley, Title 335, Price **$6.30**
7, John Lennon, Title 271, Price $7.90

If the file name is "songs.txt" and you wanted to change all occurrences of 6.30 to 7.30 you could use the command:

sed 's/6.30/7.30/' songs.txt > songs2.txt
which writes the modified file to "songs2.txt".

So the output file would contain:

1, Justin Timberlake, Title 545, Price **$7.30**
2, Taylor Swift, Title 723, Price $7.90
3, Mick Jagger, Title 610, Price $7.90
4, Lady Gaga, Title 118, Price $7.30
5, Johnny Cash, Title 482, Price $6.50
6, Elvis Presley, Title 335, Price **$7.30**

7, John Lennon, Title 271, Price $7.90

Syntax: sed <options> <argument> <file_name>

Ex. # sed -n '/p' /etc/passwd - where n is no of lines.
sed -n '$p' /etc/passwd
sed -n '3p' /etc/passwd
sed –n '3,6p' /etc/passwd
sed -n –e '/^root/p' /etc/passwd - where e is expression
sed –n -e '/root$/p' /etc/passwd
sed –e '/^$/d' /etc/passwd
sed –i.bac '/^'$/d' /opt/passwd - use '/' backslash only for wildcards.
sed -i.bac '/6/d' /opt/passwd
sed -i.bac '/6,8/d' /opt/passwd

Note: ls –l /etc/passwd will display line count as 100
ls –l /opt/passwd will display the line count as 97 as 3 lines were deleted becoz of the command "*# sed -i.bac '/6,8/d' /opt/passwd*"

sed –n –e '1,9!p' /opt/passwd - Displays all lines except first 9 lines.

Editing the stream:

sed s/root/admin /opt/passwd - replace the I occurrence of 'root' with 'admin'

sed s/root/admin/g /opt/passwd - replaces all the word globally.