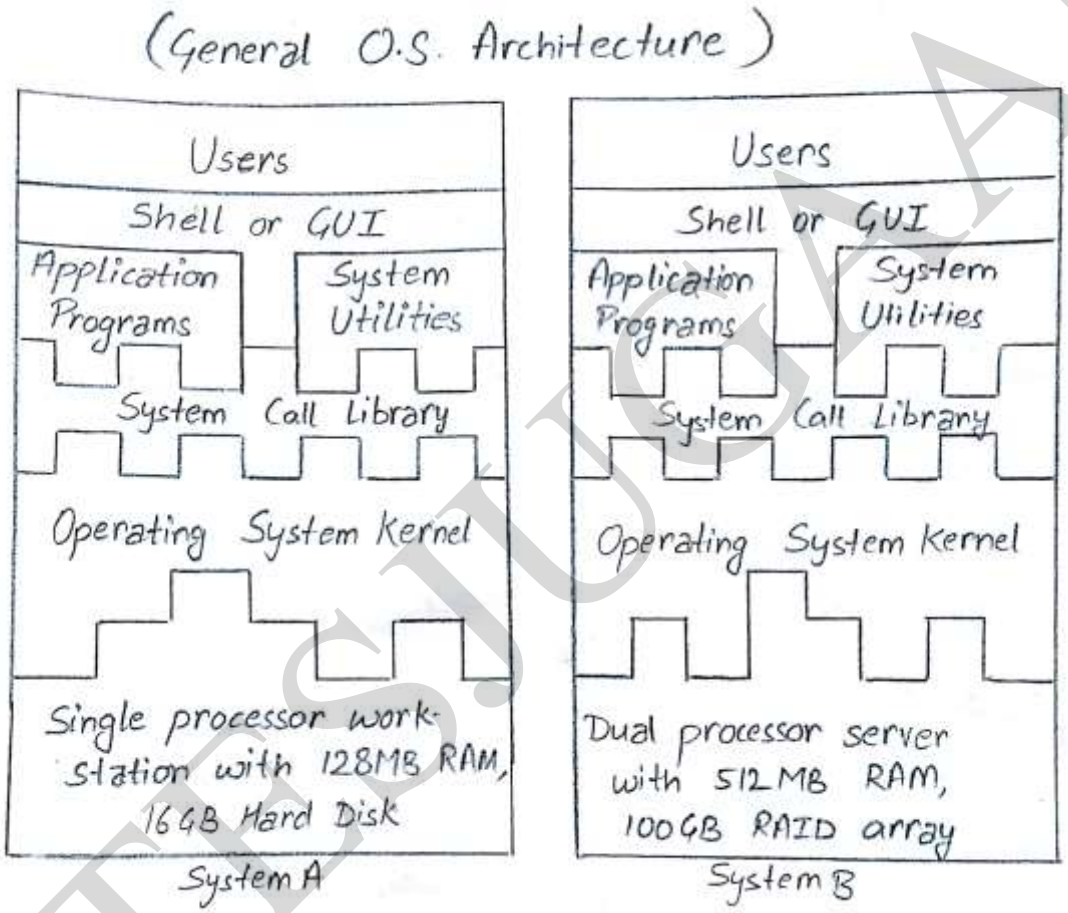
UNIT 1

An operating system (O.S.) is a resource manager. It takes the form of a set of software routines that allows users and application programs to access system resources (e.g. the CPU, memory disks, modems, printers network card, etc.) in a safe, efficient and abstract way.

For e.g. an OS ensure safe access to send data directly to the printer at any one time. An OS encourages efficient use of the CPU by suspending programs that are waiting for I/O operations to complete to make way for program that can use the CPU more productively.



General O.S. Architecture

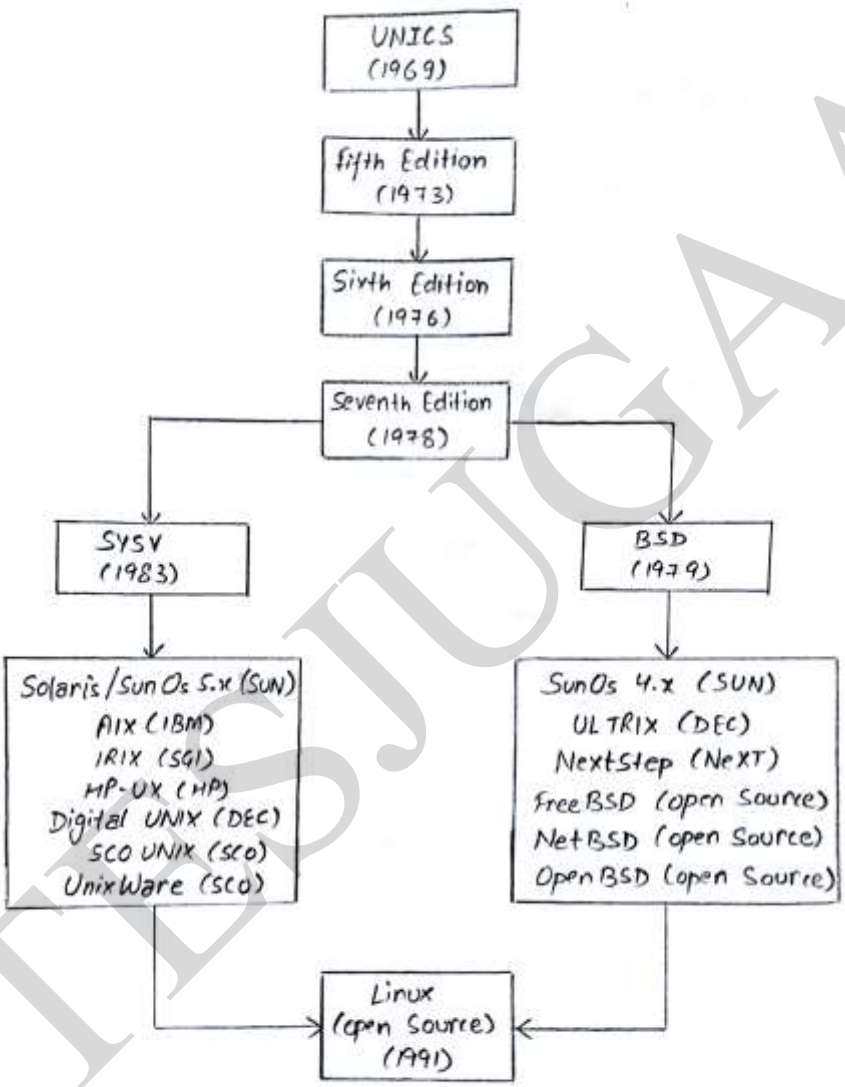
This figure represents the architecture of a typical O.S. and show how an O.S. succeeds in presenting users and application programs with a uniform interface without regard to the details of the underlying hardware. We see that:

- The O.S. Kernel is in direct control of the underlying hardware. The Kernel provides low-level device, memory and processor management functions (e.g. dealing with interrupts from hardware devices, sharing the processor among multiple programs, allocating memory for programs, etc).

- Basic hardware independent kernel services are exposed to higher-level programs through a library of system calls (e.g. services to create a file being execution of a program or open a logical network connection to another computer).

HISTORY OF UNIX

Unix has been a popular O.S. for more than two decades because of its multi-user, multi-tasking environment, stability, portability and powerful networking capabilities.



(Simplified Unix Family Tree)

In the late 1960's, researchers from General Electric M.I.T. & Bell Labs launched at a joint project to develop an ambitious multi-user, multi-tasking O.S. for mainframe computers known as MULTICS (Multiplexed Information and Computing System).

Ken Thompson then teamed up with Dennis Ritchie, the author of the first C Compiler in 1973. They rewrote the UNIX Kernel in C.

LINUX

It is a free open source Unix O.S. for pc that was originally developed in 1991 by Linus Torvalds, a finish undergraduate student. Linux is neither pure SYSV or pure BSD. To maximise code portability, it typically supports SYSV, BSD and POSIX System calls (e.g. Poll, Select, memset, memcpy).

As Linux has become more popular, several different development streams or distributed have emerged. E.g. Redhat, Slackware, Mandrake, Debian & Caldera, Mandriva, Ubuntu.

Redhat is the most popular distribution because it has been ported to a large no. of hardware platforms (including Intel, Alpha & Sparc).

ARCHITECTURE OF THE LINUX O.S.

Linux has all of the components of a typical O.S.:

1. Kernel: – The Linux Kernel includes device drivers support for a large no. of PC hardware device (graphics cards, network card, hard disks, etc), advanced processor and memory management features and support for many different types of file systems.

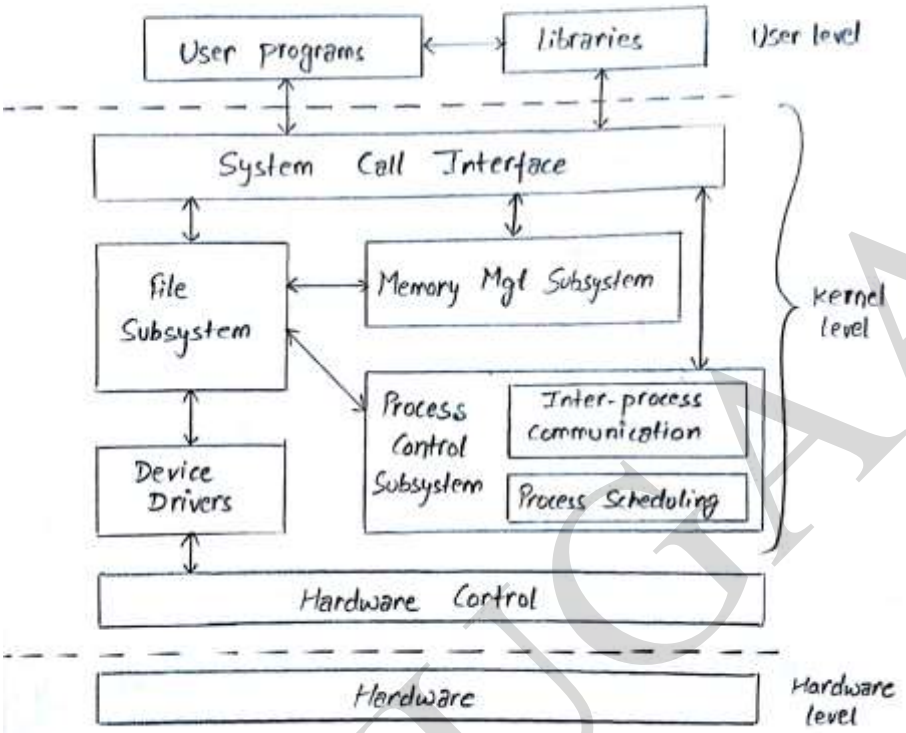
The kernel (in raw binary form that loaded directly into the memory at system start up time) is typically found in the file /boot/vmlinuz, while the source files can usually be found in usr/src/linux.

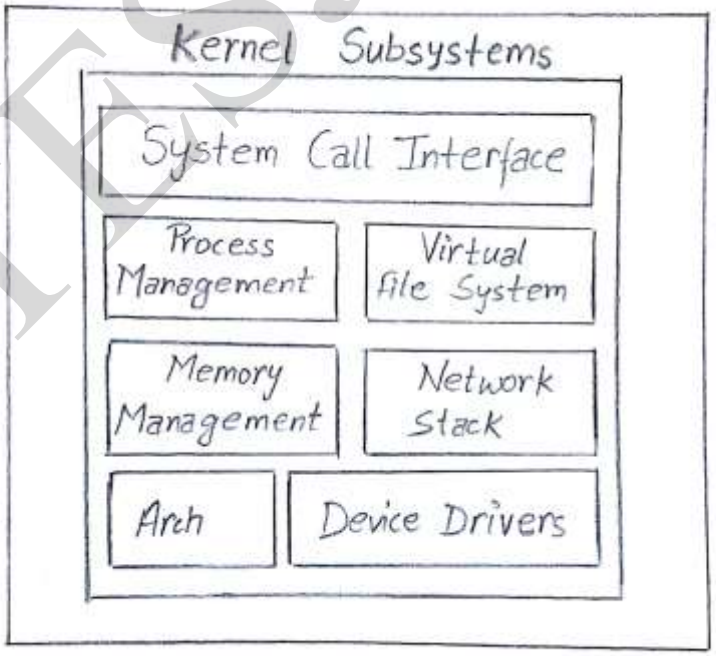
2. Shell and GUI: – Linux supports two forms of command input: through textual command line shell similar to those found on most Unix Systems. (e.g. sh – The Bourne Shell, bash – The Bourne Against Shell and csh The C Shell) and through graphical interfaces (GUIs) such as the KDE and GNOME window managers.

3. System Utilities: – Virtually every system utility that you would expect to find on standard implementations of Unix (including every system utility described in the POSIX specification) has been ported to Linux. This includes commands such as ls, cp, grep, awk, sed, bc, wc, more and so on.

Application Program: – Linux distributions typically come with several useful applications programs as standard e.g. include the emac editor, XV (an image viewer), gcc (a C Compiler), g++ (a c++ compiler), xfig (a drawing package), latex (a powerful type setting language) and Soffice (Star office, which is an MS-Office style clone that can read and write word, excel.

(Block Diagram Of System kernel)





ARCHITECTURE OF LINUX KERNEL

FEATURES OF UNIX

1. A multiuser system
2. Multitasking system
3. Repository of Applications
4. Building Block (a complex task can be broken into finite no. of simple ones)
5. Pattern matching
6. Programming facility
7. Documentation (the online help facility – man commands)

UNIX ARCHITECTURE

Unix O.S. used to manages the resources of a computer (such as CPU, memory, I/O devices, network, etc). The memory resident portion of Unix system is Kernel. File System and Process Control System are two major components of Unix Kernel. There are many features are available of Unix O.S. Kernel such as: –

1. CONCURRENCY: – As Unix is a multiprocessing O.S. many processes run concurrently to improve the performance of the system.
2. VIRTUAL MEMORY (VM): – Memory management subsystem implements the Virtual Memory concept and a user need not worry about the execute program size and the RAM size.
3. PAGING: – It is a technique to minimize the internal as well as external fragmentation in the physical memory.
4. VIRTUAL FILE SYSTEM: – A VFS is a file system used to help the different file system complexities. A user can use the same standard file system related calls to access different file systems.

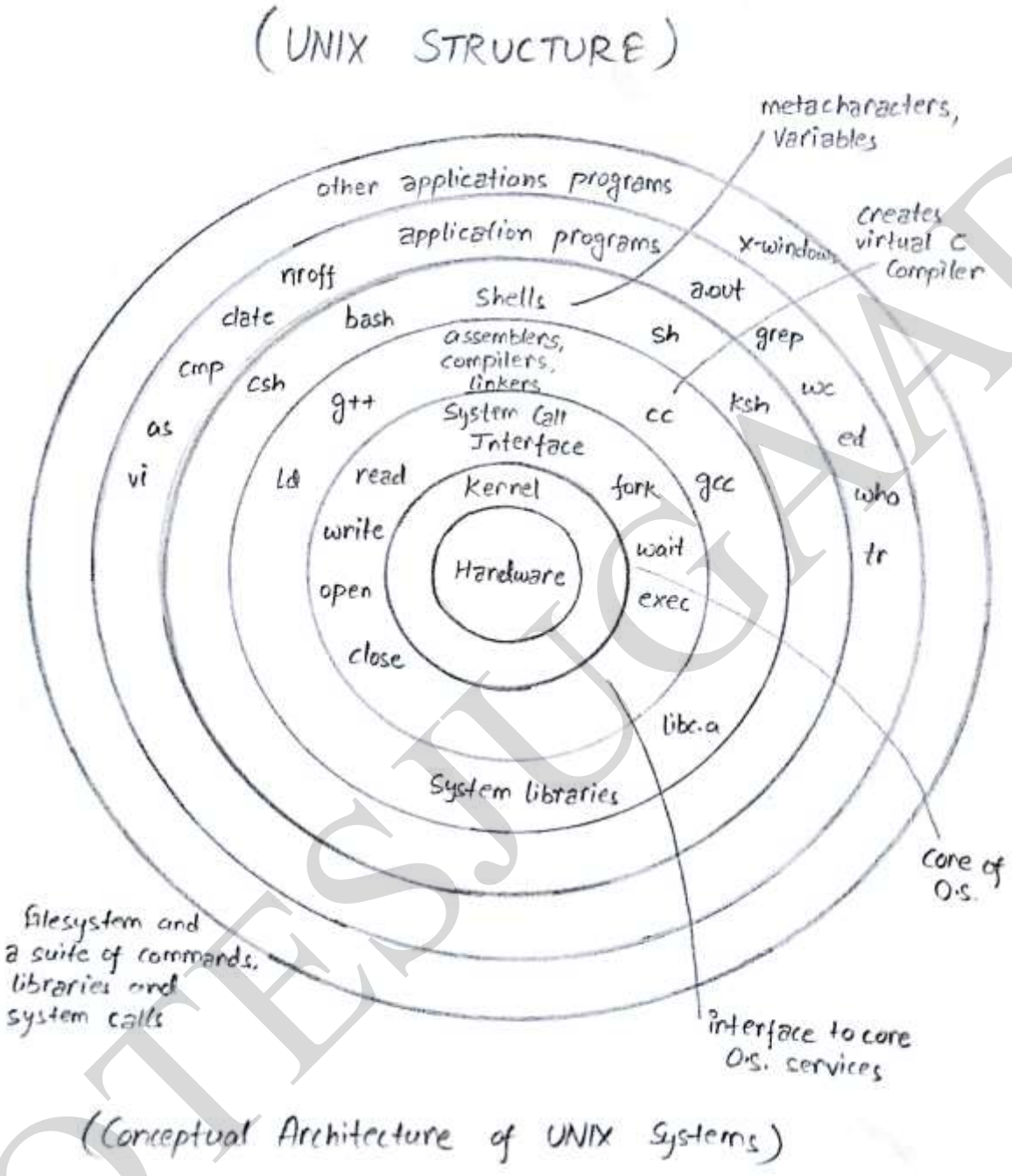
The kernel also provided some other basic services such as interrupt and trap handling system calls, scheduling & timer & clock handling, file descriptor management.

FEATURES OF UNIX ARCHITECTURE

1. Unix systems use a centralized O.S. kernel which manages system and process activities.
2. All non-kernel software is organized into separate, kernel managed processes.
3. Unix O.S. are preemptively multitasking-multiprocess can run at the same time or with in small time slices and nearly at the same time and any process can interrupted and moved out of execution by the kernel. This is known as Thread Management.
4. Files are stored on a disk in a hierarchical file system, with a single top location throughout the system (root, or "/") with both files and directories, subdirectories and sub-sub directories.
5. With few exceptions, devices and some type of communication between processes are managed and visible as files or pseudo-files with in the file system hierarchy. This is known as everything is a file.

Some other features and capabilities: –

1. Multitasking and multiuser
2. Programming interface
3. Use of files as abstraction of devices and other objects.
4. Built in Networking (TCP/IP)
5. Persistent system service processes called "daemons' and manged by init or inetd.



CONCEPTUAL ARCHITECTURE OF UNIX SYSTEMS

SOME PRINCIPLES

1. Interactive multi used and multiuser O.S. design.
2. Design objective principle of Unix, file store organisation, text processing and programming.
3. Role of C programming language with regard to potability and reliable system software.
4. Process control – signals and fork.
5. Error logging and recovery from system failures.
6. Modifiability and Application Programmer Interface (API).
7. The user's perspective on Unix.

Unix System Architecture for a monolithic kernel: –

Kernel memory	User memory	User interface
Kernel	Daemons user processes (fork, exec, wait, exit, SIGCHLD cycle)	SHELLS: sh, csh, ksh, zsh, tcsh, bash and Xwindow System Interface
Abstraction Layer	Descriptors for files, pipes, network sockets	
Hierarchical File Store	Files, pipes, process tree, sockets and other devices	/proc/etc/bin/sys/dev/mnt/usr….

GNU PROJECT/FSF

GNU is a Unix-like O.S. that is free software- it respects your freedom. You can install Linux-based versions of GNU which are entirely free software. The GNU Systems provides a collection of applications, libraries, and developer tools, plus a program to allocate resources and talk to the hardware; known as a Kernel. Richard Stallman, founder of GNU Project.

The GNU Project is a free software, mass collaboration project, announced on 27 Sep.,1983 by Richard Stallman at MIT. Its aim is to give computer users freedom and control in their use of their computers and computing devices by call abortively developing and providing software that is based on the following freedom rights: users are free to run the software, share it (copy, distributed), study it and modify it. GNU Software guarantees these freedom-rights legally (via its license) and is therefore free software.

The FSF (Free Software Foundation) is a 501(c)(3) non profit organization founded by Richard Stallman on 4 October 1985 to support the free software movement, which promotes the universal freedom to create, distributed and modify computer software with the organization preference for software being distributed under copyleft (‘share dislike”) such as with its own GNU General Public License. The FSF was incorporated in Massachusetts, USA where it is also based.

GPL (General Public License)

GPL is the most widely used free software license, which guarantees end users (individuals, organizations, companies) the freedom to use, study, share copy and modify the software. Software that ensures that these rights are retained is called free software.

The GPL grants the recipients of a computer programs the rights of the Free Software Definition and uses copyleft to ensure. The freedom is preserved whenever the work is distributed even when the work is changed or added to.

As of August 2007, the GPL accounted for nearly 65% of the 43,442 free software projects listed on free code and as of January 2006, about 68% of the project listed on sourceforge.net.

It is believed that the copyleft provided by the GPL was crucial to the success of the Linux based systems, giving the programmers who contributed to the Kernel the assurance that their work would benefit the whole world and remain free rather than being exploited by software companies that would not have to give anything back to the community.

Some versions of GPL:

1. GPL version 1 – released on 25 Feb 1989
2. GPL version 2 – released on June 1991
3. GPL version 3 – released on 29 June 2007

RUNLEVEL

In Linux, runlevel define how the system is started. After booting, the system starts as defined in /etc/init tab in the line init default. Usually this is 3 or 5 (full multiuser mode with network (3) and full multiuser mode with network and a-display).

As an alternative, the runlevel can be specified at the boot time (at the boot prompt, for instance). Any parameters that are not directly evaluated by the Kernel itself are passed to init.

To change runlevels while the system is running, enter init and the corresponding number as an argument. Only the system administrator is allowed to do this init 1 (or shutdown n now) causes the system to change to single user mode, which is used for system maintenance and administration. After finishing the work, the administrator can switch back to the normal runlevel by entering init 3, which starts all the essential programs and allowed regular users to log in and to work with the system init 0 (or shutdown h now) causes the system to halt. init 6 (or shutdown r now) causes it to shutdown with a subsequent reboot.

Available Runlevels

Runlevel	Description
0	System halt
S	Single user mode; from the boot prompt
1	Single user mode
2	Local multiuser mode without remote network (NFS)
3	Full multiuser mode with network
4	Not used
5	Full multiuser mode with network and x-display manager (KDM, GDM, XDM)
6	System reboot

If /etc/init tab is damaged, the system might not boot properly. Therefore, be extremely careful while editing /etc/init tab and always keep a backup of an intact version. To repair damage, try entering init = /bin/sh after the kernel name at the boot prompt to do boot directly into a shell. After that, replace /etc/inittab with your backup version using CP command.

The /etc/inittab file supplies the script to the init command role as a general process dispatcher. The process that constitutes the majority of the init commands process dispatching activities in the /etc/getty line process, which initiates individual terminal lines.

LINUX FILE SYSTEM AND DIRECTORY STRUCTURE

In Linux, major files can be reduced to a file. Partitions are associated with files such as /dev/hda/. Hardware components are associated with files such as /dev/modem. Detected devices are documents as files in the /proc directory.

These directories organized user files, drivers, kernels, logs, programs, utilities and more into different categories. Every File system Hierarchy Standard (FHS) starts with the root directory, also known as by its label, the single forward slash (/). All of the other directories shown in table are subdirectories of the root directories. You can also find their files on the same partition as the root directory.

/: – the root directory, the top-level directory in the FHS. All other directories are the sub directories of root, which is always mounted on same partition. All directories that are not mounted on a separate partition are included in the root directory partition.

/bin: – essential command line utilities, should not be mounted separately, otherwise it could be difficult to get to other utilities when using a rescue disk.

/boot: – include linux startup files, including the linux kernel. Can be small, 16MB usually adequate for a typical modular kernel. If you use multiple kernels, such as for testing a kernel upgrade, increase the size of this partition accordingly.

/etc: – most basic configuration files.

/dev: – hardware and software device drivers for everything from the floppy disk to terminals. Do not mount this directory on a separate partition.

/home: – home directories for almost every user.

/lib: – program libraries for the kernel and various command line utilities. Do not mount this directory on a separate partition.

/mnt: – the mount point for removable media including floppy drives, CD-ROM and ZIP disks.

/opt: – applications such as word perfect or star office.

/proc: – currently running kernel- related process including device assignments such as IRQ Parts, I/O addresses and DMA channels.

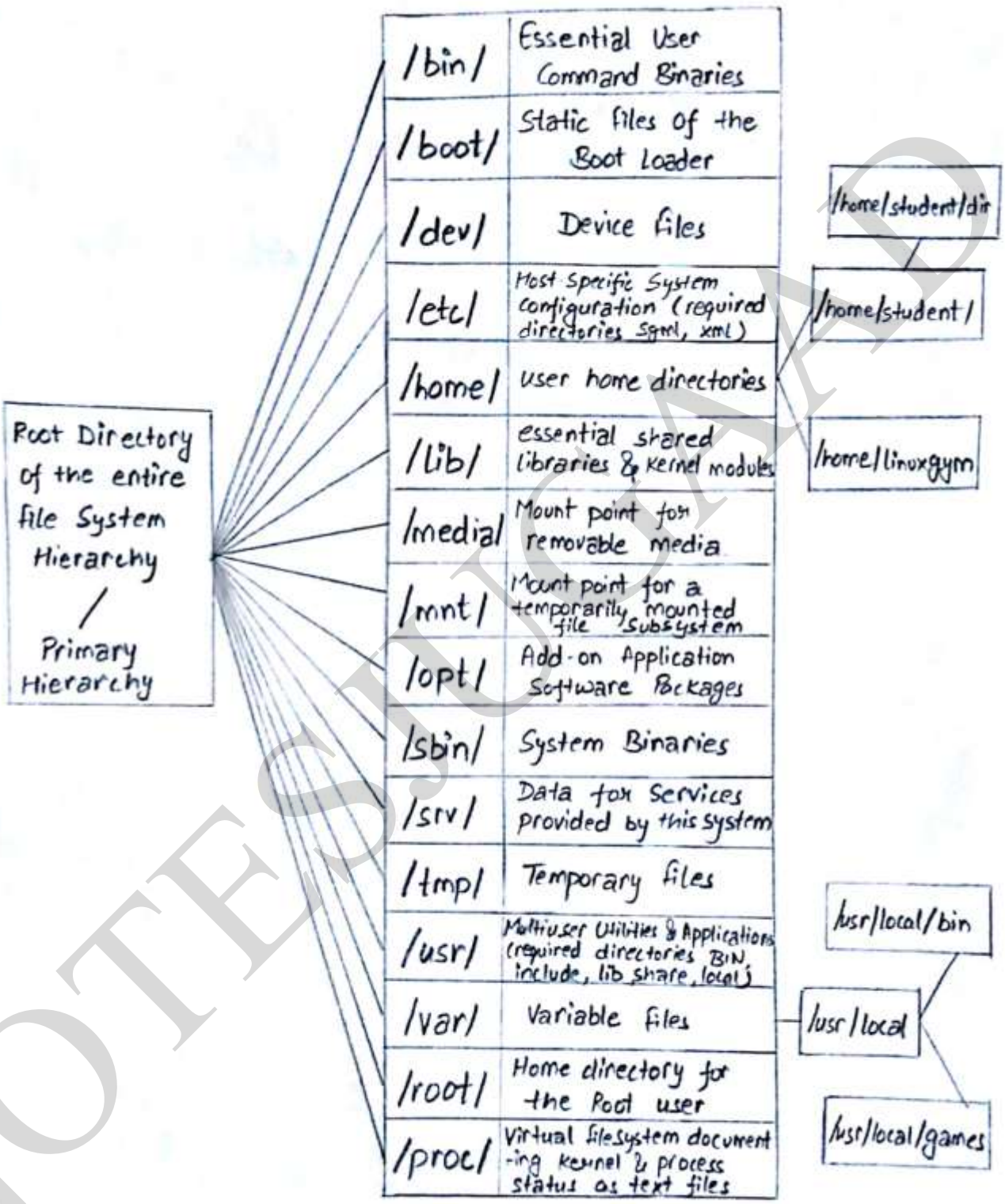
/root: – the home directory of the root user.

/sbin: – system administration commands. Don’t mount this directory separately.

/tmp: **–** temporary files, by default, Red hat Linux deletes all files in this directory periodically.

/usr: – small programs accessible to all users includes many system administration command and utilities.

/var: – variable data, including log files and printer.



(LINUX DIRECTORY STRUCTURE)

LINUX DIRECTORY STRUCTURE DIAGRAM

FUNCTIONS OF DIFFERENT DIRECTORIES

Directory	Description
/bin	All binaries needed for the boot process and to run the system in single user mode, including essential commands such as cd, ls, etc.
/boot	Holds files used during the boot process along with the Linux kernel itself.
/dev	Contains device files for all hardware devices on the system.
/etc/init.d	Contains various service startup scripts.
/etc/profile.d	Holds application setup scripts run by /etc/profile upon login.
/etc/rc.d	Contains sub directories which contain run level specific scripts.
/etc/skel	Holds example dot files used to populate a new user's home directory.
/etc/XII	Contains sub directories and configuration files for the X window system.
/home	User home directories.
/lib	Some shared library directories, files and links.
/usr/bin	Contains command available to normal users.
/usr/bin/XII	X window system binaries.
/usr/include	Holds include files used in C programs.
/usr/share	Contains shared directories for man files, info files etc.
/usr/local/bin	Common executable application files local to this system.
Lost + found	Contains files to restores files after a system crash or partition not un-mount before a system shutdown.

LINUX FILE SYSTEM

In a computer, a file system is the way in which files are named and placed logically to store, retrieve and update the data and also used to manage space on the available devices.

File system is divided in two segments called user data and metadata.

A Linux system, just like Unix makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, texts, images and so forth are all files. Input and output devices and generally all devices, are considered to be files, according to the system.

EXT-2

1. Ext-2 stands for second extended file system.
2. It was introduces in 1993, developed by Remy card.
3. This was developed to overcome the limitation of the original file system.
4. Ext-2 does not have journaling feature.
5. On flash drives, USB drives, ext-2 is recommended, as it doesn't need to do the overhead of journaling.
6. Maximum individual file size can be from 16GB to 2TB.
7. Overall ext-2 file system size can be from 2TB to 16TB.

EXT-3

1. Ext-3 stands for third extended file system.
2. It was introduces in 2001, developed by Stephen Tweedie.
3. Starting from Linux kernel 2.4.15 ext-3 was available.
4. The main benefit of ext-3 is that it allows journaling.
5. Maximum individual file size can be from 16GB to 2TB.
6. Overall ext-3 file system size can be from 4TB to 32TB.
7. There are three types of journaling available in ext-3 file system:-

- Journal: – metadata and content are saved in the journal.
- Ordered: – only metadata is saved in the journal.
- Writeback: – only metadata is saved in the journal. Metadata might be journaled either before or after the content is written to the disk.

You can convert the ext-2 file system to ext-3 file system directly (without backup/restore).

EXT-4

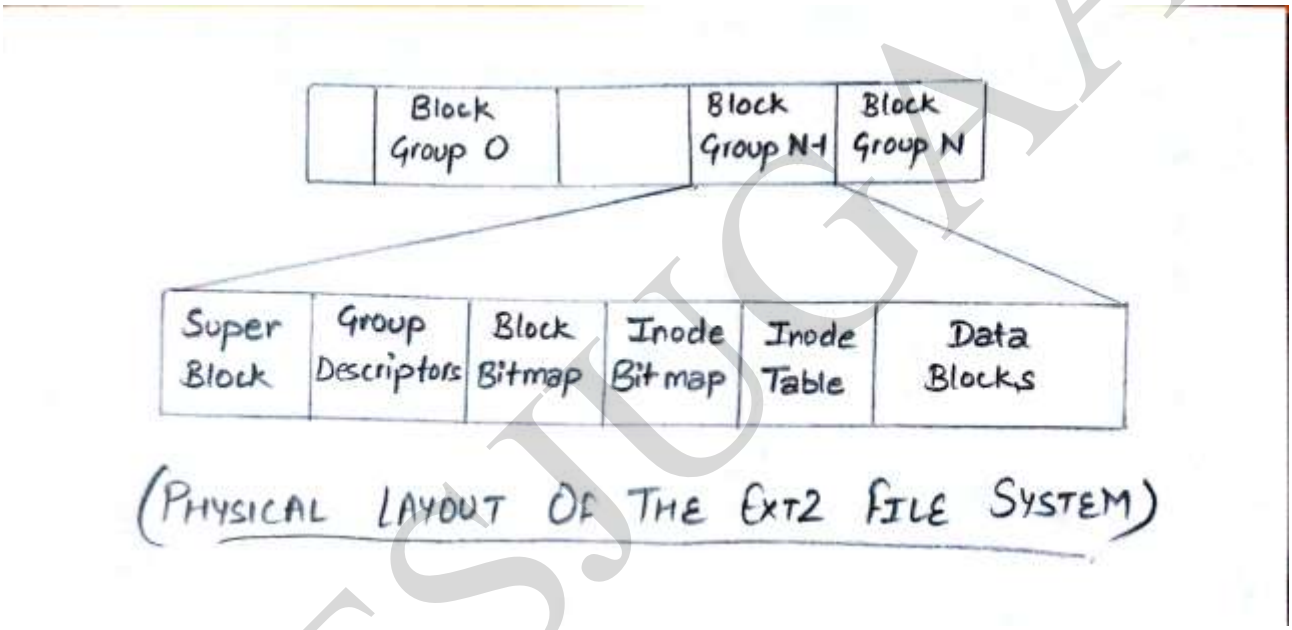
1. Ext-4 stands for fourth extended file system.
2. It was introduces in 2008.
3. Starting from Linux kernel 2.6.19 ext-4 was available.
4. Supports huge individual file size and overall file system size.
5. Maximum individual file size can be from 16GB to 16TB.
6. Overall maximum ext-4 file system size is 1EB (Exabyte).
7. Directory can contain a maximum of 64,000 subdirectory (as opposed to 32000 in ext-3)
8. You can also mount an existing ext-3 fs as ext-4 fs (without having to upgrade it).
9. Several other new feature are introduced in ext-4: multiblock allocation, delayed allocation, journal checksum, fast fsck etc. all you need to know is that these new features have improves the performance and reliability of the file system when compared to ext-3.
10. In ext-4, you also have the option of turning the journaling features "off".

(Note: 1EB=1024PB (petabyte); 1PB=1024TB (terabyte).)

In Linux, as it is for Unix[tm], the separate file systems may use are not accessed by device identifiers (such as a drive no. or drive name) but instead they are combined into single hierarchical tree structure that represents the file system as one whole single entity. Linux adds each new file system into this single file system tree as it mounted.

VFS (Virtual File System) allows Linux to support many, often very different file system, each printing a common software so that all file systems appear identical to the rest of the Linux kernel and to programs running into the system. Linux's VFS layer allows you to transparently mount the many different file systems at the same time.

Ext-2 the second extended file system was devised as an extensible and powerful file system for Linux. It is also the most successfully file system so far in the Linux community and is the basic for all of the currently shipping Linux distributions.



(PHYSICAL LAYOUT OF THE EXT2 FILE SYSTEM)

Ext-2 Inode

In ext-2 file system, the inode is the basic building blocks, every file and directory in the file system is described by one and only one inode. The ext-2 inodes for each block group are kept in the inode table together with a bit map that allows the system to keep track of allocated and unallocated inodes shows the following fields:-

Mode: – this holds two pieces of information, what this inode describes and the permissions that users have to use it. For ext-2 an inode can describe one of file directory symbolic links, block device, character device.

Owner information: – the user and group identifiers of the owners of this file or directory.

Size: – the size of the file in bytes.

Time stamps: – the time that the inode was created and the last time that it was modified.

Data block: – pointers to the blocks that contains the data that this inode is describing. The first eleven are pointers to the physical blocks containing the data described by this inode and the last three pointers contain more and more levels of information.

COMMANDS IN LINUX

1. **HELP COMMAND**: – display information about builtin commands. It provides the facility to show the help about the command and Linux & Unix.

SYNTAX: – help [-dms] [patterns…]

Display brief summaries of shell builtin commands. If pattern in specified, gives detailed help on all commands matching pattern, otherwise the list of help topics is printed.

OPTIONS: –

-d	Output short description for each topic.
-m	Display usage in pseudo-manpage format.
-s	Output only a short usage synopsis for each topic matching.

EXAMPLE: –

help echo

2.**MAN COMMAND**: – On Linux and other Unix-like operating system man is the interface used to view the system's reference manuals.

SYNTAX: – man [-c file] [-d] [-D] [–warnings [=warnings]]

man [-hv]

man is the system's manual viewer; it can be used to display manual pages, scroll up and down, search for occurrence of specific text, and other useful functions.

Each argument given to man is normally the name of a program, utility or function.

OPTIONS: –

-h, –h	Print a help message and exit
-v, –version	Display version information and exit
-c file, –config file=file	Use configuration file *file* rather than the default of ~/.manpath
-d, –debug	Print debugging information
-D, –default	This option, when used, is normally specified as the first option.
–warnings[=warnings]	Enable warnings from the groff text formatter. This may be used to perform sanity check on the source text of manual pages. The warnings is a comma-separated lists of warning name.

EXAMPLES: –

man man: – view the manual page for the man command.

man — nh –nj man : – view the manual page for man, with no hyphenated words or justified lines.

3. **WHATIS COMMAND**: – whatis displays short manual page descriptions. Whatis searches the manual page names and displays the manual page description of any name matched.

Name may contain wildcards (-w) or be a regular expression (-r). Using these options, it may be necessary to quote the name or escape (\) the special characters to stop the shell from interpreting them.

SYNTAX: – whatis [-dlhvV] [-r] [-w] [-s list] [-m system[,—-]] [-m path] [-l locale] [-c file] name…

OPTIONS: –

-d, –debug	Print debugging information
-v, –verbose	Print verbose warning messages
-r, –regex	Interpret each name as a regular expression
-w, — wildcard	Interpret each name as a pattern containing shell style wild cards
-l, –long	Do not trim output to the terminal width
-s list, –sections list	Search only the given manual sections.
-c,– config file	Use this user configuration file rather than the default of ~/.manpath
-h, –help	Print a help message and exit
-v, –version	Display version information

EXAMPLE: –

whatis whatis : – display a description of what whatis is.

4. **INFO COMMAND**: – info reads documentation in the info format. Info is similar to man, with a more robust structure for linking pages together. Info pages are made using the texinfo tools, and can link with other pages, creates menus and ease navigation in general.

SYNTAX: – info [option]… [menu-item…]

OPTIONS: –

-k, –apropos =STRING	Look up string in all indices of all manuals
-d, –directory =DIR	Add DIR to INFOPATH
-f, –file = filename	Specify info file to visit
-h, –help	Display this help and exit

EXAMPLES: –

info emacs: – start at emacs node from top-level dir.

info –show -options emacs :- start at node with emacs command lines options.

info –f ./foo.info :- show file ./foo.info, not searching dir.

5. **DATE COMMAND**: – The date command is used to print or sets the system's time and date information.

SYNTAX: – date [options]… [+ format]

OPTIONS: –

-d	Display time describes by string
-f	Processed once for each line of file data file

FORMAT: –

%a	Local abbreviated weekday name (e.g. Sun)
%A	Local full weekday name (e.g. Sunday)
%B	Local full month name (January)
%c	Local date and time
%d	Day of month

EXAMPLES: –

date: – running date with no options will output the system date and time

date –s: – "11/20/2003 12:48:00": – set the system date and time to Nov. 20,2003,12:48 PM.

6. **WHOAMI COMMAND**: – the whoami command writes the user name (i.e. login name) of the owner of the current login session to standard output.

whoami particularly useful when using shells such as ash and sh that do not show the name of the current user in the command prompt. It is also useful for confirming the current owner of a session after using the su command.

SYNTAX: – whoami [option]

EXAMPLE: –

whoami: – displays the name of the user who runs the command.

7. **WHO COMMAND**: – Display who is logged on to the system. It is used to display time of last system boot, list of users logged in, current run level, etc.

- Get the information on currently logged in users.

$who

- Get the time of last system boot.

$who –b

- Get information on system login processes.

$who -l

- Get the host name and user associated with stdin.

$who -m

- Get the current run level.

$who -r

- Get the list of user logged in.

$who -u

- Get no. of users logged in and their use names.

$who -q

- Get all the information.

$who –a

8. **W COMMAND**: – the w command shows who is logged into the system and what they are doing. A login, logging in or logging on is the entering of identifier information into a system by a user in order to access that system. It generally requires the user to enter two pieces of information, first a user name and then a pwd.

SYNTAX: – w [options] [username1, username2, …]

EXAMPLE: –

w: – running the w command with no arguments will show a list of logged on users and their processes.

9. **CAL COMMAND**: – cal originally appeared in version 6 of AT&T Unix. Cal displays the current month at the command line. It is a quick and convenient way to glance at the dates of the month and can be useful as part of a login script.

SYNTAX: – cal [option] [[day] [month] [year]]

OPTIONS: –

-1	Display a single month.
-3	Display three months (last, this and next)
-s	Display calendar using Sunday as the first day of week
-m	Display Monday as the first day of week
-j	Display date of the Julian calendar
-y	Display a cal. For the entire current year.

EXAMPLE: –

cal 12 2001: – display the calendar for December of the year 2001.

10. **BC COMMAND**: – arithmetic operations are the most common in any kind of programming language. Unix or Linux provide a bc command and expr command for doing arithmetic calculations.

BC command support following features: –

1. Arithmetic operator
2. Increment and decrement operator
3. Assignment operator
4. Comparison relational operator
5. Logical and Boolean operator

SYNTAX: – bc [options]

OPTIONS: –

-c	Compile only. The output is bc command that are send to the standard output.
-l	Define the math functions and initialize scale to 20, instead of the default zero.

EXAMPLES: –

$ echo “2+5” | bc

$ echo “10-4” | bc

$ echo “var=5; ++var” |bc

11. **HOSTNAME COMMAND**: – hostname is used to display the system’s DNS name and to display or set its hostname or NIS (Network Information Services) domain name.

When called without any arguments, hostname will display the name of the system as returned by gethostname function.

When called with one argument or with the –file option, hostname will set the system hostname using sethostname function.

SYNTAX: – hostname[-v] [-a] [-d] [-f]

OPTIONS: –

-d	Display the name of the DNS domain.
-f	Display the FQDN (Fully Qualified Domain Name).
-v	Be verbose with all output.

EXAMPLES: –

1. View your hostname

$ hostname

1. Change hostname from command line.

$hostname Priya-system

1. Change hostname (get name from a file).

$cat /home/root/hostname.txt

Priya-desktop

$hostname -F /home/root/hostname.txt

$hostname

Priya-desktop

12. **UNAME COMMAND**: – sometimes it is required to quickly determine details like kernel name, version, hostname then we use the uname command.

SYNTAX: – uname [option]

EXAMPLES: –

- Print the kernel name (by default)

$uname

- Get the kernel name using -s option.

$uname -s

-s: – print the name of O.S.

- Get the network node hostname using -n option.

$uname -n

- Get kernel release information using -r.

$uname -r

- Get maximum information through -a.

$uname -a

13. **ALIAS COMMAND**: – it can be useful if you want to create a shortcut to a command.

SYNTAX: – alias name = 'command' / alias [name [=value]] > alias home = 'cd/home/dave/public_html'

This will create an alias called home which will put you in the /home/dave/public_html directory. Whenever you type home at the command prompt.

alias instructs the shell to replace one string with another when executing commands. It is used to customize the shell session interface.

EXAMPLES: –

alias

alias ls = 'ls –color = auto'

add color coding to the file and directory listings of ls.

alias copy = 'cp'

alias copy "cp"

This alias means that when the command copy is read in the shell, it will be replaced with cp and that command will be executed instead.

14. **CD COMMAND**: – cd command is used to change the current directory (i.e. the directory in which the user is currently working). In Linux and other Unix like O.S., it is similar to the CD and CHDIR command in MS-DOS.

SYNTAX: – cd [option] [directory]

OPTIONS: –

-L	Force symbolic links to be followed
-P	Use the physical directory structure

EXAMPLES: –
cd../home/users/computerhope: – the above example would traverse up directory and then down into the home/users/computerhope sub directory.

Cd hope: – the above example would change the working directory to the hope subdirectory if exists.

15. **MV COMMAND**: – mv is a command that we are going to use to move files around or to rename them. mv sort of has a split personality because it serves these two functions at the same time. The mv command is used to move or rename files.

SYNTAX: – mv [option] source destination

OPTIONS: –

-b	Backup but does not accept an argument, the default method is used.
-n	Do not over write any existing file.

EXAMPLES: –

mv myfile.txt destination-directory: – move the file myfile.txt to the directory destination directory.

mv computer\hope.txt computer\hope2.txt: – removes the file “computer hope.txt to computer hope2.txt.

16. **COPY/CP COMMAND**: – to copy files and directory use the cp command under Linux, Unix BSD like O.S.. cp is the command entered in a Unix/Linux shell to copy a file from one place to another, possibly on a different file system. The original files remain unchanged and new file may have the same or a different name.

SYNTAX: – cp source destination

cp source directory

cp soure1 source2 source3 source n directory

cp [option] source destination

OPTIONS: –

-P	To copy a file to a new file and preserve the modification date, time and access control list associated with a source file
-R	To copy a directory, including all its files and subdirectories to another directory, enter (copy directories recursively)

EXAMPLES: –

$cp file.doc newfile.doc

$cp filename/tmp

Copy a file in your current directory into another directory, enter

$cp -p filename/path/to/new/location/myfile

$cp* /home/tom/backup

$cp -R* /home/tom/backup

17. **RM COMMAND**: – rm removes each specified file. It does not remove directories. If a file is unwritable, the standard input is a tty and -f or force option is not given. rm prompts the user for whether to remove the file. If the response does not begin with 'y' or 'Y', the file is skipped.

OPTIONS: –

-d, –directory	Unlike file, even if it is a non-empty directory (super-user only; this works only if your system supports 'unlink' for non-empty directories).
-f, –force	Ignore non-existent files, never prompt.
-I, –interactive	Prompt before any removal.

EXAMPLES: –

rm ./-foo

rm abc.txt

18. **CAT COMMAND**: – the cat command is considered one of the most frequently used command on Linux or Unix like operating system. It can be used for the following purpose under Unix or Linux: –

- Display text files on screen.
- Copy text files.
- Combine text files.
- Create new text files.

SYNTAX: – cat filename

cat option filename

cat file1 file2

cat file1 file2 > new combine file

EXAMPLE: –

Displaying the contents of files.

To read or read the contents of files enter:

$cat /etc/passwd

To above example the output from cat command is written to /tmp/text.txt file instead of being displayed on the monitor screen. You can view/tmp/text.txt using cat command itself:

$cat /tmp/text.txt.

CONCATENATE FILES: – concatenation means putting multiple file content together. The original file or files are not modified or deleted. In this example cat will concatenate copies of the contents of the three files /etc/hosts, /etc/resolv.conf. and /et/fstab.

$cat /etc/hosts /etc/resolv.conf. /et/fstab

CREATE A FILE: –

$cat > foo.txt

COPY A FILE: – the cat command can also be used to create a new file and transfer the data from an existing file, to make copy of

$cat oldfile.txt > newfile.txt

19. **LESS COMMAND**: – the less command is used to view files (instead of opening the file to view in an editor). less is similar to more command, but less allow both forward and backward movements.

SYNTAX: – less <filename>/<textfile>

EXAMPLES: –

less textfile: – to read the contents of a file named text file in the current directory.

ls -la | less: – the less utility is often used for reading the output of other commands. For ex. To read the output of the ls command on screen at a time.