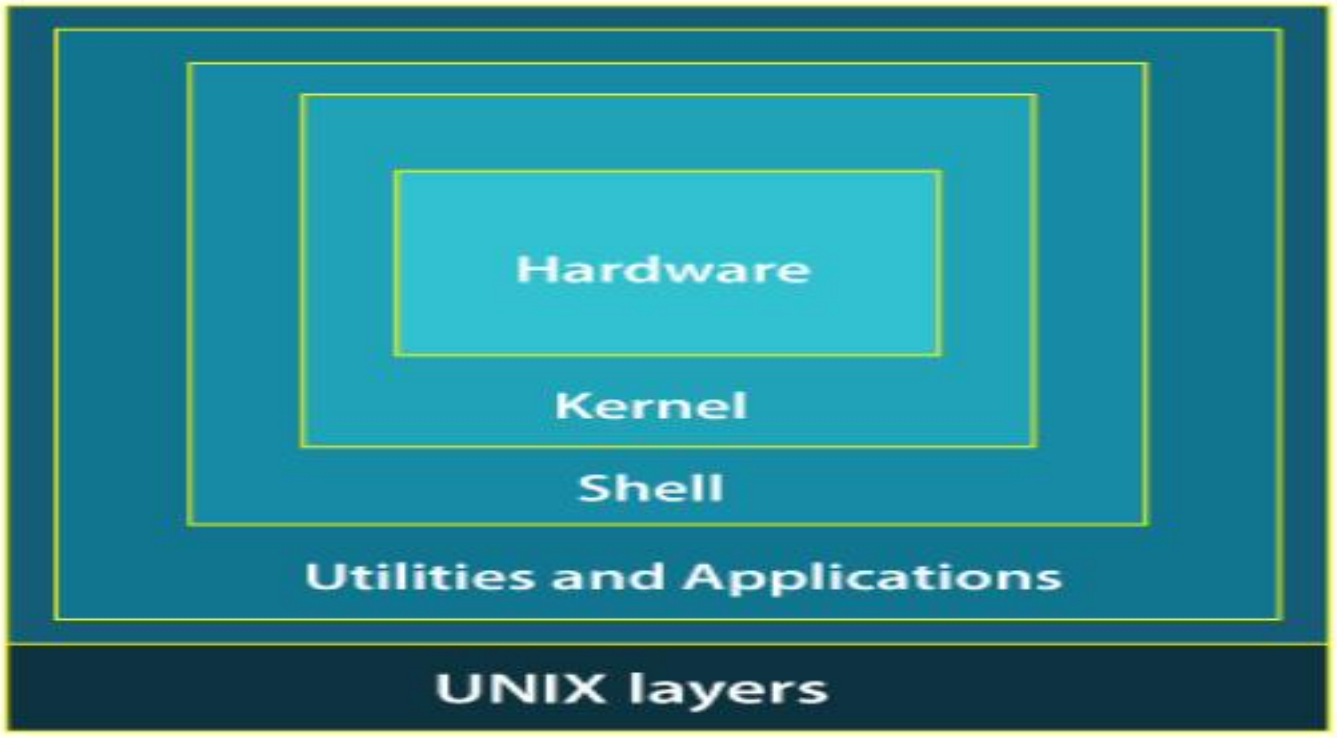
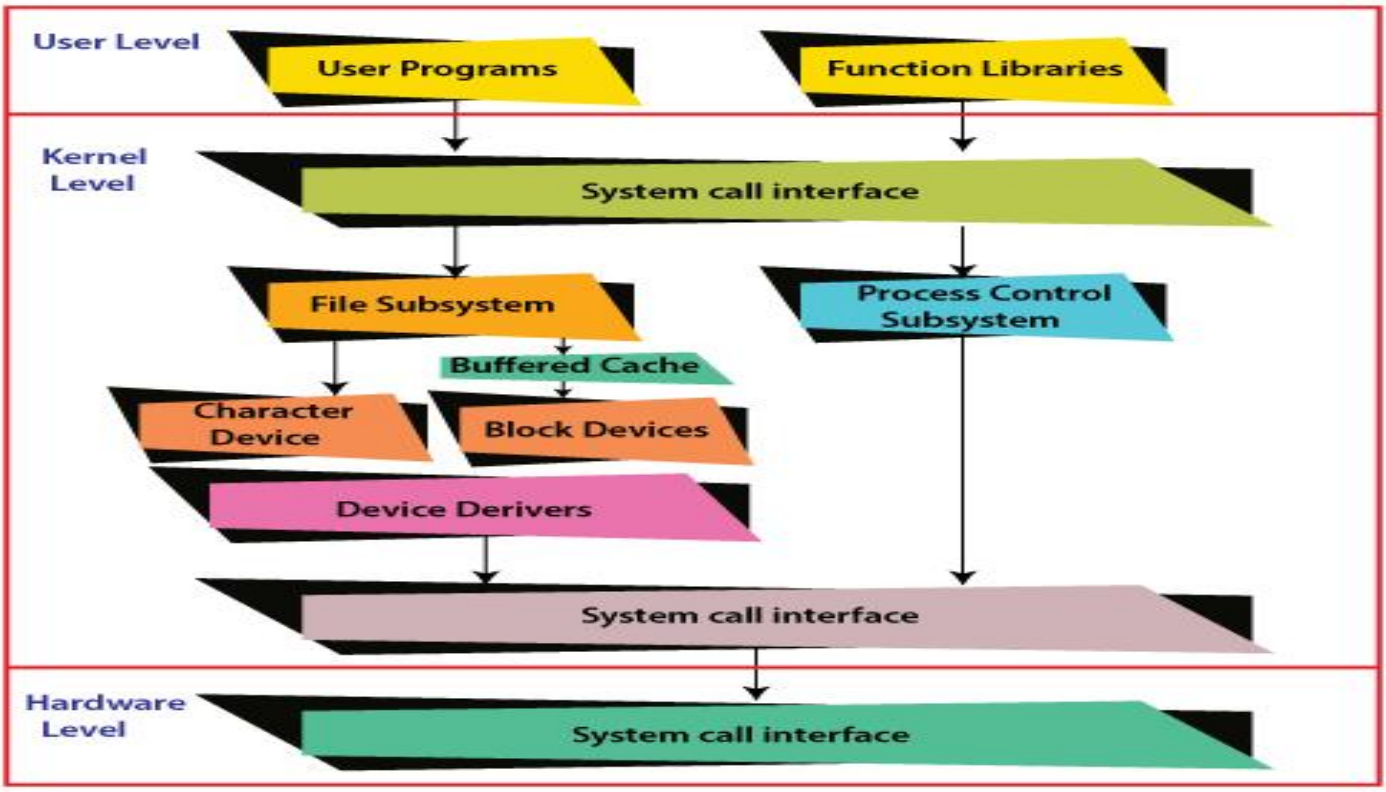
Linux Assignment – Sem VI

~ Write it in your Linux Notebooks

1. Explain Unix System Structure



While working with UNIX OS, several layers of this system provide interaction between the pc hardware and the user.
Layer-1: Hardware -
This layer of UNIX consists of all hardware-related information in the UNIX environment.
Layer-2: Kernel -
The core of the operating system that's liable for maintaining the full functionality is named the kernel.
The kernel of UNIX runs on the particular machine hardware and interacts with the hardware effectively.



Kernel Architecture

It also works as a device manager and performs valuable functions for the processes which require access to the peripheral devices connected to the computer. The kernel controls these devices through device drivers.

The kernel also manages the memory. Processes are executed programs that have owner's humans or systems who initiate their execution.

The system must provide all processes with access to an adequate amount of memory, and a few processes require a lot of it.

To make effective use of main memory and to allocate enough memory to every process.

It uses essential techniques like paging, swapping, and virtual storage.

Layer-3: The Shell -

The Shell is an interpreter that interprets the command submitted by the user at the terminal and calls the program you simply want.

It also keeps a history of the list of the commands you have typed in. If you need to repeat a command you typed it, use the cursor keys to scroll up and down the list or type history for a list of previous commands. There are various commands like cat, mv, cat, grep, id, wc, and many more.

Layer-4: Application Programs Layer -

It is the outermost layer that executes the given external applications. UNIX distributions typically come with several useful applications programs as standard.

For Example: emacs editor, StarOffice, xv image viewer, g++ compiler etc.

2. Write a short note on Linux File System

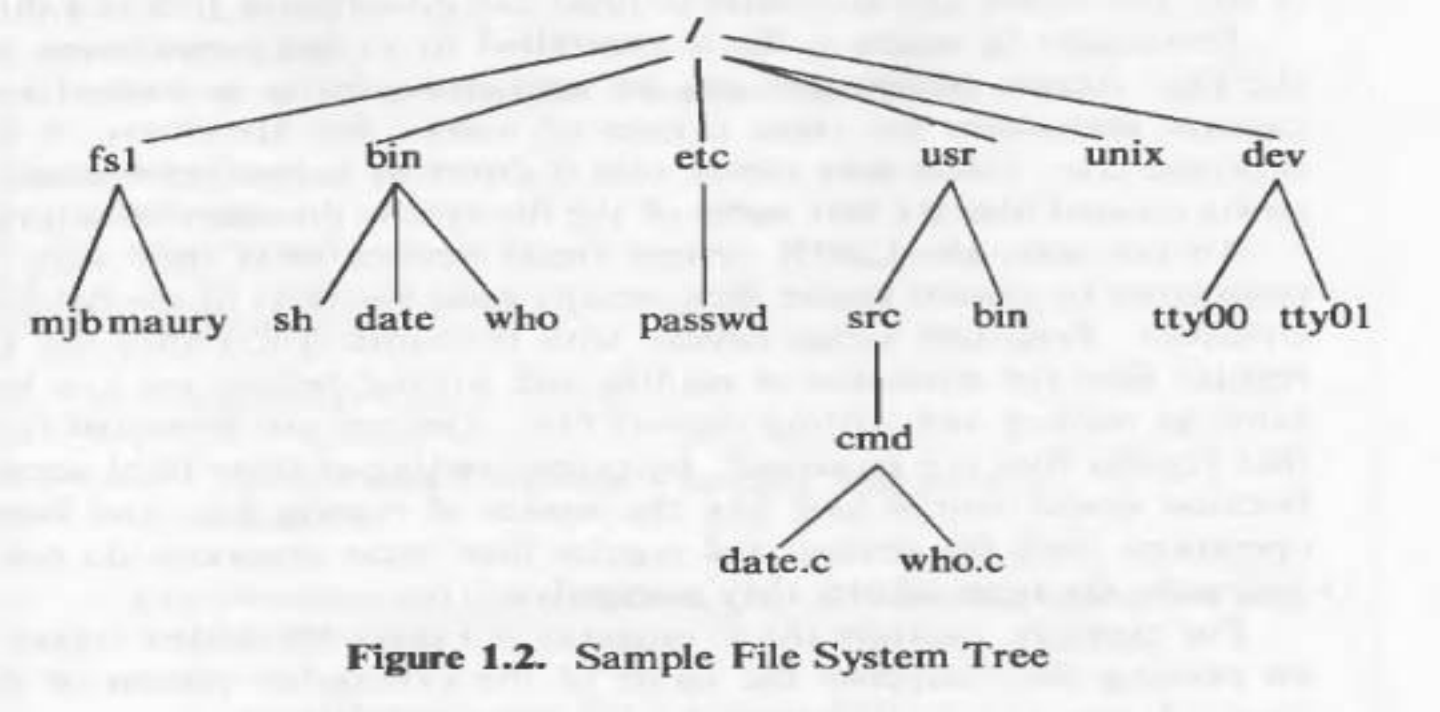


Figure 1.2. Sample File System Tree

The characteristics of unix file system are

- A hierarchal structure.
- Consistent treatment of data
- Ability to create and delete files
- Dynamic growth of files
- Peripheral devices are also treated as files
- The file system is organized as a tree. The root node is called “root” and is denoted by “/”.
- Every non leaf node in this structure is a directory and every leaf node is a file/special device file.
- The name of the file is given by the path name.

A full path name starts with the root directory i.e. a slash character and specifies the file that can be found by travestying the tree. Some examples of paths could be “/etc/passwd”, “/bin/who” and “/usr/src/programs/test.c”.

The path that starts from the root directory is called the absolute path. Alternatively we can give path of any file relative to any other directory. This path will be called relative path.

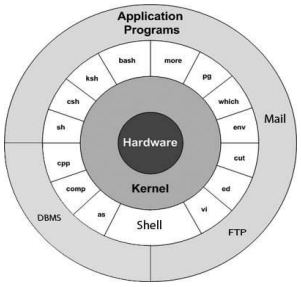
The files are just stream of bytes it is up-to the program to interpret these bytes. Directories are also files i.e. a stream of bytes but the operating system program knows how to interpret them as directories. Example program could be “ls”

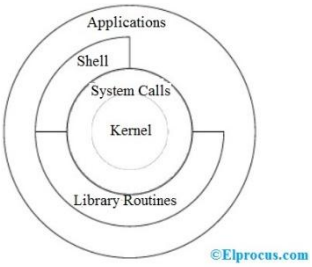
Permission to any file is governed by the file access permissions. Access permissions are set independently for read, write and execute. These permissions are set independently for the file

owner, file group and everyone else. Access permission looks like
rwx-rwx-rwx (We will see more of this in later chapters)

Unix treats devices as if they are files. Every device is treated as special files and occupy position in the file system. Programs can access devices using the same syntax as if they were accessing files. Syntax of reading and writing on devices is more or less same as reading and writing regular files. Devices are protected in the same way as files i.e. using access pemissions.

3. Explain Architecture of Unix/Linux File System





The architecture of this operating system is four layered.
It consists of Hardware, Kernel, System Call interface(shell) and application libraries/tools, utilities, etc...
The kernel controls the hardware of the computer and resides at the core of the architecture.
System calls acts as the interface between the kernel and other libraries.
These libraries include general functions and built on top of the system calls.
Shell is a special application that provides an interface to the other applications of the architecture..
The main concept that unites all the versions of Unix is the following four basics −

1. Kernel − The kernel is the heart of the operating system.
It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.
The main functions of Kernal are-
Computer hardware such as memory, disc, printers, etc.. are controlled by the kernel.
The kernel schedules the processes, control and executes various user-defined tasks.
Manages the data storage and control the computer accesses by several users.
The kernel is composed of several sub-components such as configurations including boot code, device drivers to control hardware, header files.

2. Shell −
It is the interface between the user and the kernel.
Users can interact with the shell using shell commands.
Shell has two main responsibilities which include interpreting the commands given by the users and execute them using the kernel, providing programming ability to the users to write shell commands

for a shell script to perform specific tasks.
The shell is the utility that processes your requests.
When you type in a command at your terminal, the shell interprets the command and calls the program that you want.
The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.

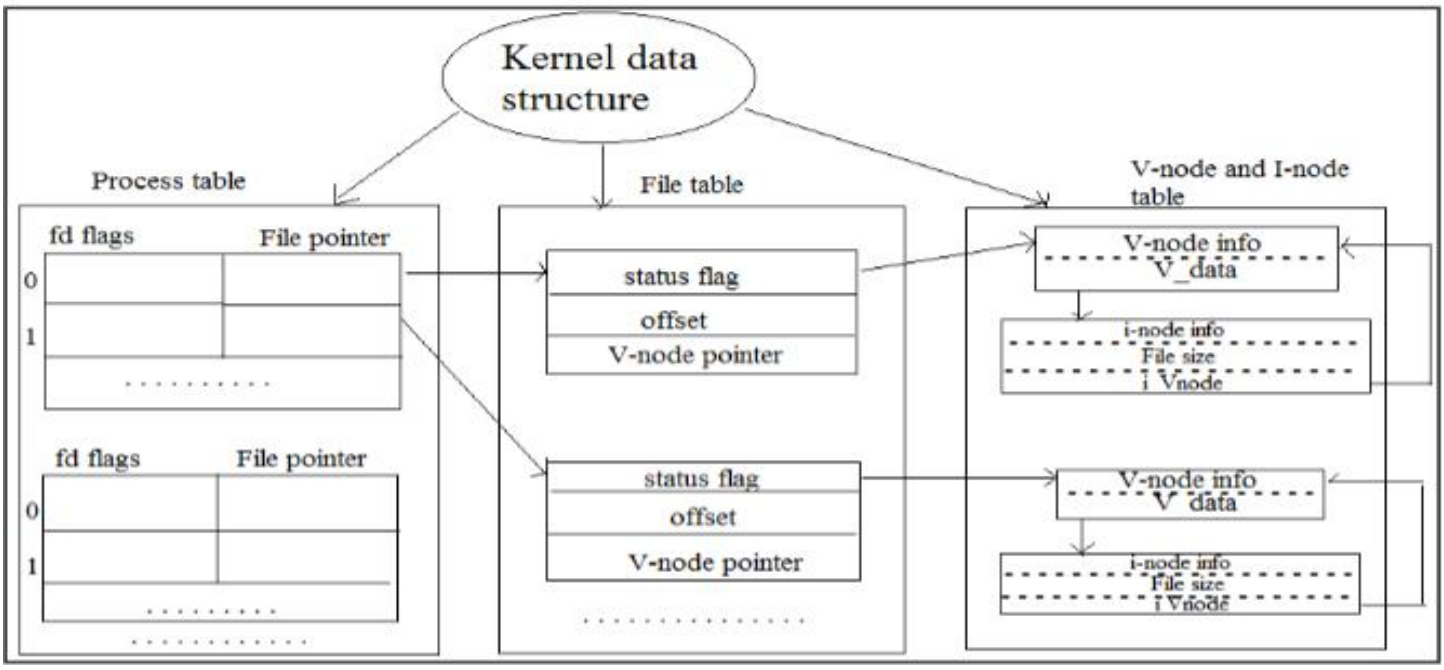
3. Commands and Utilities −
There are various commands and utilities which you can make use of in your day to day activities. cp, mv, cat and grep, etc. are few examples of commands and utilities.
There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.

4. Files and Directories −
All the data of Unix is organized into files.
All files are then organized into directories.
These directories are further organized into a tree-like structure called the file system.

4. Explain Kernel data structure with well-labelled diagrams.

The kernel data structures play a vital role because they store data about the current state of the system.
When a new process is created in the system, a kernel data structure is created as well that stores the details about that process.
Most of the kernel data structures can be access by the kernel and its subsystems. They hold the data as well as pointers to other data structures.
Consider a user logging into the system, at that time, a new process arises and kernel stores the details about that process into the physical memory.

Kernel Components
The kernel stores and creates a lot of information. So it has data about which processes are running in the system, their memory requirements, files in use, etc.
Kernel data structures are maintained using three important structures. These are process table, file table and v node/ i-node information.



Details about these are given below:

Process Table

The process table holds information about all the processes running in the system. It is required by the kernel. These include storage information, execution status, file information, etc. Process table also stores the other entries like -

Process state: (When a process forks a child, its entry in the process table is duplicated including the file information and file pointers. So the parent and the child process share a file.)

Process ID: It is created when a new process generated and identifies the process.

User ID: It determines the privileges to the users for the particular process.

Pointer: It is a pointer to a page table for managing the memory and also a pointer to the process area.

Timer: It is used for knowing which resource uses how much time.

File Table

A process table has a pointer which points to the file table.

File table holds the entries about all the files in the computer.

If multiple processes use the same file, then they contain the same file information and the file descriptor number.

Each file table entry contains information about the file such as file status (file read or file write), file offset, etc.

The file offset specifies the position for next read or write into the file.

The file table also contains v-node and i-node pointers which point to the virtual node and index node respectively. These nodes contain information on how to read a file.

V-Node and i-Node Tables

Both the v-node (virtual node) and i-node (index node) are related to the storage system of the file and the storage mechanisms.

The v-node is an abstract concept (in the form of the object) that describes the interaction with file data. All file manipulation like closing a file, opening a file is done by V-node object. V-node information stored in main memory.
The i-node data structure gives information about files or directory (i.e. actual representation of files or directory)
The information like the location of disk block where the file is stored, time at which file changes last, owner of the file, access permissions (read/write), etc. i-node information is stored in secondary storage.
A directory is names given to the i-nodes. A directory is in the form of parent and each of its children. List of file names and corresponding i-node number are stored in a directory entry.

5. Write short note on absolute and relative paths.

A path is a unique location to a file or a folder in a file system of an OS.
A path to a file is a combination of / and alpha-numeric characters.

Absolute Path-name
An absolute path is defined as the specifying the location of a file or directory from the root directory(/).
To write an absolute path-name:
Start at the root directory (/) and work down.
Write a slash (/) after every directory name (last one is optional)

For Example :
$cat abc.sql
will work only if the fie “abc.sql” exists in your current directory.
However, if this file is not present in your working directory and is present somewhere else say in /home/kt , then this command will work only if you will use it like shown below:
cat /home/kt/abc.sql
In the above example, if the first character of a pathname is /, the file’s location must be determined with respect to root. When you have more than one / in a pathname, for each such /, you have to descend one level in the file system like in the above kt is one level below home, and thus two levels below root.

An absolute path is defined as specifying the location of a file or directory from the root directory(/). In other words,we can say that an absolute path is a complete path from start of actual file system from / directory.

Relative path
Relative path is defined as the path related to the present working directly(pwd).
It starts at your current directory and never starts with a / .
To be more specific let’s take a look on the below figure in which if we are looking for photos then absolute path for it will be provided as /home/jono/photos but assuming that we are already

present in jono directory then the relative path for the same can be written as simple photos.

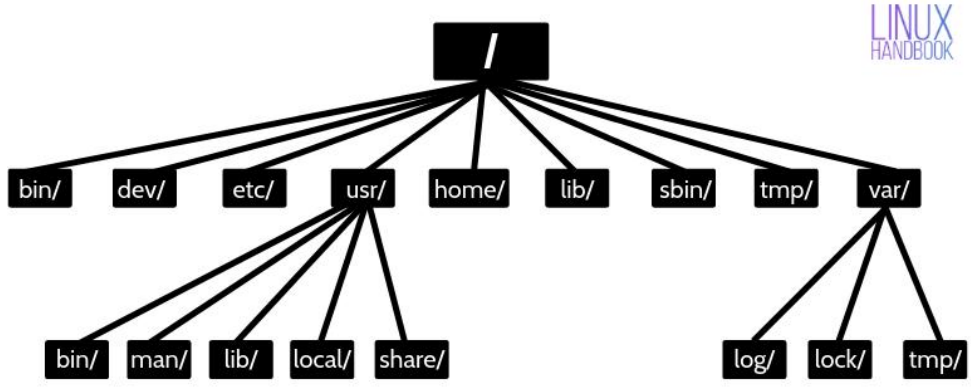
UNIX offers a shortcut in the relative pathname– that uses either the current or parent directory as reference and specifies the path relative to it. A relative path-name uses one of these cryptic symbols:
.(a single dot) - this represents the current directory. ..(two dots) - this represents the parent directory.
Now, what this actually means is that if we are currently in directory /home/kt/abc and now you can use .. as an argument to cd to move to the parent directory /home/kt as :

Example of Absolute and Relative Path
Suppose you are currently located in home/kt and you want to change your directory to home/kt/abc.

Let's see both the absolute and relative path concepts to do this:
Changing directory with relative path concept :
$pwd /home/kt
$cd abc
$pwd /home/kt/abc
Changing directory with absolute path concept:
$pwd /home/kt

$cd /home/kt/abc
$pwd /home/kt/abc

6. *Describe file system in detail with well labelled file tree structures.*



The Linux Foundation maintains Linux Filesystem Hierarchy Standard (FHS). It defines the directory structure and contents in a UNIX-like operating system.
In the FHS, all files and directories should under the root directory, even those are store on the different physical or virtual device. The root directory denoted by the symbol "/."
The UNIX/Linux operating system has the following directory structure.

/ (root)

The root is the first directory in a directory structure.
Every files and directory starts from the root directory.
Only the root user of the system has the right to write under the root (/) directory.
One another directory with name root is under the root directory (/root) is the home directory of the root user.

/bin

It is essential to command binary files location, where all command's binary files are stored, and we are using these command because "/bin" added in the environment variable "$PATH."
/bin directory stored the commands like cat, ls, cp, grep, etc.

/boot

It stores all information about boot loader.
Kernel initd, vmlinux, grub files are located into /boot directory.

/dev

The /dev directory known as device files directory. It includes terminal devices, USB, or any device attached to the system. E.g., /dev/tty1, /dev/usbmon0.

/etc

It contains all system and program configuration files.
It also contains start-up and shutdown scripts for a program used with start/stop individually.
Configuration file examples are "/etc/resolv.conf", "/etc/apache2/apache2.conf", etc.
Program script examples are "/etc/init.d/apache2", "/etc/init.d/cron", etc.

/home

The /home directory contains the user's home directories, personal files, user's personal settings, etc.
Example: /home/linuxconcept, /home/satish etc.

/lib

It stores libraries essential files for the binaries in /bin and /sbin.
Library filenames are either start with ld or lib.
Example: ld-1.32.1.so, libncron.so.7.3

/media

The /media directory used as a mount point for a removable device such as CD-ROM.
It is temporary mount point for removable device such as /media/cdrom, /media/floppy, or /media/cdrecorder etc.

/mnt

It is for a temporarily mounting filesystem, where system admin can mount a file system.

/opt

The /opt directory used for optional application software packages.
Users can install add-on application under the /opt or /opt/ sub-directory.

/sbin

It is like /bin directory containing binary files related to the system.
The /sbin directory is storing Linux operating system related binaries executable. E.g., iptables, fdisk, ifconfig, etc.

/srv

The srv stands for service.
The /srv directory contains server specific services related data, e.g., /srv/cvs stores CVS related data.

/tmp

The /tmp directory stores files which are created by system or user for a temporary purpose. So, on system reboot, all files from this directory will be deleted.

/usr

The /usr directory contains libraries, binaries, documentation, and source code for a second-level program.
The /usr/bin directory stores binary files for the user's program. If binary not found in /bin, you should look into /usr/bin.
The /usr/sbin directory stores binary files for system admin. If system binary not found in /sbin, you should look into /usr/sbin.
The /usr/lib contains libraries files for /usr/bin and /usr/sbin.
The /usr/local stores users program which you have installed from source.
The /usr/src contains the Linux kernel source, header-files, and documentation.

/proc

The /proc directory stores information about system process.
It is a pseudo filesystem stores information about the running process.
It is a virtual filesystem that contains system resources information in text format. E.g., /proc/uptime.

7. Write note on file system mounting and unmounting.

File System Mounting and Unmounting in Linux:

Mounting:

Mounting is the process of making a file system accessible by associating it with a directory (mount point) in the Linux file system tree. Mounting a file system is required to access its content, make changes to it, or copy files from it to other file systems.

The process of mounting a file system in Linux can be done using the 'mount' command. This command takes two arguments: the source file system, which could be a device or a remote file system, and the target directory, which is the mount point. The general syntax for the mount command is as follows:

```
mount [OPTIONS] SOURCE TARGET
```

For example, to mount a file system stored on /dev/sdb1 to the directory /mnt, the following command can be used:

```
mount /dev/sdb1 /mnt
```

Unmounting:

Unmounting is the process of removing an already mounted file system from the file system tree. Unmounting a file system makes it inaccessible, and all operations on the file system will fail.

The process of unmounting a file system in Linux can be done using the 'umount' command. This command takes one argument, which is the mount point of the file system that needs to be unmounted. The general syntax for the umount command is as follows:

```
umount TARGET
```

For example, to unmount the file system that was previously mounted to the directory /mnt, the following command can be used:

```
umount /mnt
```

Note: It is important to note that only the file systems that are not in use can be unmounted. This means that if any process is using the file system, the umount command will fail, and the file system cannot be unmounted.

In conclusion, mounting and unmounting file systems are essential operations in Linux, and they enable access to different file systems and the ability to make changes to them. The 'mount' and 'umount' commands are the primary tools for mounting and unmounting file systems in Linux.