

---

# Chapter Table of Contents

## Chapter 5.2

### Advanced Tasks

Aim.....	201
Instructional Objectives.....	201
Learning Outcomes.....	201
5.2.1 Running Shell Command Macros.....	202
Self-assessment Questions.....	209
5.2.2 Mounting and Unmounting File System .....	210
5.2.3 Checking and Monitoring System Performance.....	214
Self-assessment Questions.....	215
5.2.4 File Security and Permissions.....	215
Self-assessment Questions.....	222
5.2.5 Becoming Super User using su command .....	222
5.2.6 Getting System Information using uname command .....	227
5.2.7 Installing and Removing Packages using rpm command.....	229
Self-assessment Questions.....	231
Summary .....	232
Terminal Questions.....	234
Answer Keys.....	235
Activity.....	235
Bibliography.....	236
e-References .....	236
External Resources .....	237
Video Links .....	237

---





## **Aim**

To understand file systems available in Linux



## **Instructional Objectives**

After completing this chapter, you should be able to:

- Explain different types of file system available in Linux
- Describe how to create a new file system after partitioning of hard disk
- Illustrate mounting and unmounting file system
- Explain the process of checking and monitoring system performance
- Discuss file security and permission in detail
- Describe how to become a super user
- Illustrate how to extract system information using uname command
- Explain how to install and remove packages using rpm command



## **Learning Outcomes**

At the end of this chapter, you are expected to:

- Explain the procedure to run macros in shell command
- List the commands used to set widow position
- Write the commands used to auto indent and display line numbers
- Identify the basic elements of communication process
- Use various commands to communicate with other users

---

## 5.2.1 Running Shell Command Macros

A file system can be created on UNIX. Following are the steps to create the file system on UNIX:

1. First we need to prepare the device. i.e the **fdisk command** is used to partition the device.
2. The file system will be created inside the partition.
  - First the partition must be formatted and prepared for the partition with the chosen file system. This is achieved by using **mkfs** command.
  - Optionally verify the file system integrity – check for errors using **fsck**.
3. The file system needs to be mounted so that it is visible to the users.
  - The **mount** command is used to connect the file system into the existing directory tree.
  - Always use above steps in order i.e.**fdisk, mkfs and mount**.

Now let's see how to implement these commands in detail.

### Step -I Create the partition using fdisk

A partition is nothing but a section of a physical disk. It is formatted to contain a file system.

The **MBR**(Master Boot Record) of the disk i.e. the first sector contains the Partition Table which can hold exactly four partition entries.

- Here we only cover the four-entry MSDOS Partition Table type as an example for easy understanding.
- Each partition table entry specifies its disk start location, end location, and whether it is bootable

### What is a disk partition?

- Each of the four DOS partition entries on a disk can be either a Primary partition or an Extended Partition
- Up to four Primary Partitions are possible on a disk
- Only one of the four partition entries can be designated as an Extended Partition that may contain any number of additional Logical Partitions inside it (as many as you like)
- An Extended Partition takes one of the four slots and reduces the number of possible Primary Partitions to three

- You can create many Logical Partitions inside an Extended Partition, up to the size limit of the Extended Partition
- Some operating systems only recognize a limited number of Logical Partitions

### Linux Disk Names – sd?

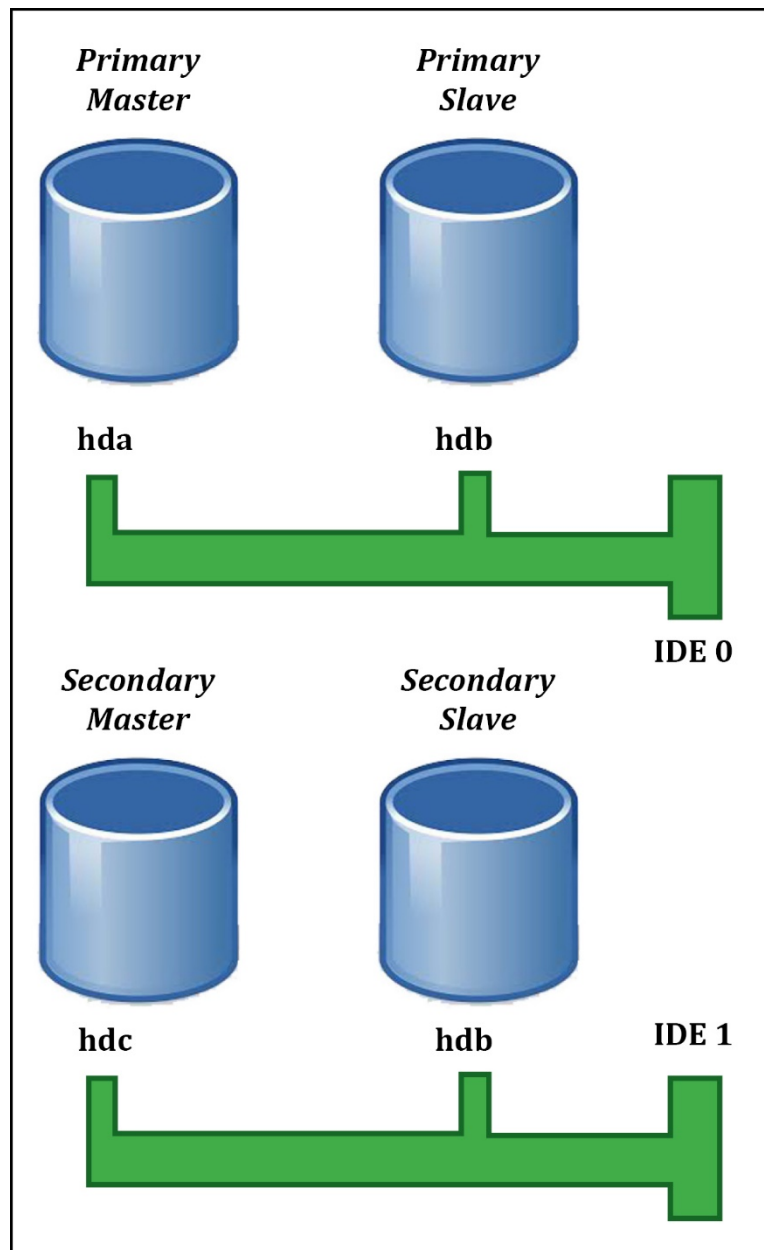


Figure 5.2.1

---

## Naming disk drives – sd?

- Primary master ---hda or sda
- Primary slave -----hdb or sdb
- Secondary master - hdc or sdc
- Secondary slave --hdd or sdd
- Other disks: sde, sdf, sdg, etc.
- “sd” used to mean only **SCSI** (Small computer system interface) disks, but modern Linux systems treat all disks, even IDE, ATA, and SATA, as SCSI disks and name them starting with “sd”

## Linux Partition Names in /dev Naming partitions

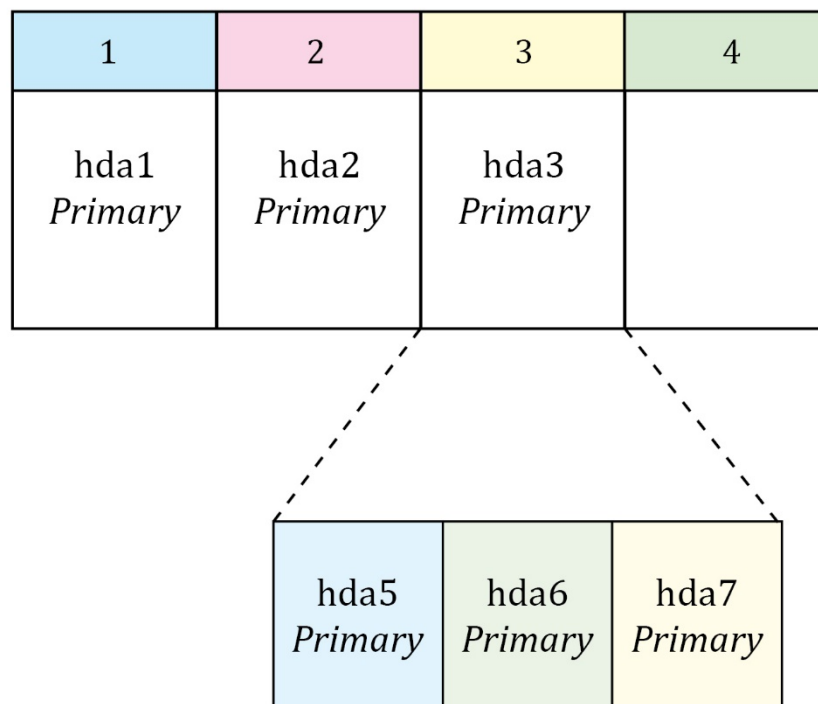


Figure 5.2.2

-The four DOS partitions always use these four names **sd?1-sd?4**

-**sd?5 – sd?63** Logical Partitions inside an Extended Partition always start at number 5

- Maximum of 4 DOS primary/extended partition entries - “Any number” of Logical Partitions inside Extended

---

**Example:** Each disk and each partition is represented in the Linux file system as a separate “device special file”, usually in the /dev directory:

- /dev/sda represents the entire first disk (sdb is the second)
- /dev/sda1 is the first of four DOS partitions (Primary or Extended) of the first disk (sda2 is the second primary/extended DOS partition)
- Logical Partitions inside an Extended Partition always start at 5
- /dev/sda5 is always the first Logical Partition (sda6 is the second)
- Partitions using the four DOS Primary Partition entries never change numbers when you create or delete them. They always number 1 to 4.
- Logical Partitions are always numbered sequentially starting at number 5; removing a Logical Partition causes all the following logical partition numbers to go down by one, e.g. 6 becomes 5, 7 becomes 6, etc.!

### Required Linux Partitions

Linux can run inside only a single partition, the **ROOT** partition, but most Linux systems use at least two partitions:

- A ROOT partition
  - e.g. it might be /dev/sda1
- A swap partition (to permit virtual memory)
  - e.g. /dev/sda2 (if swap is a primary partition)
  - or /dev/sda5 (if swap is a logical partition)
  - Or any other partition – swap can go anywhere

### Linux fdisk command

- The **fdisk** command is the universal command-line partition table manipulator for Linux: `fdisk [options] device`
  - The device is a disk name (“/dev/sda”) not a partition name (not /dev/sda1)
  - Useful “-l” option: `fdisk -l ;fdisk -l /dev/sda`

### Command line options:

Various command line options those can be used are as follows.

l - list partition types

a - toggle a bootable flag

---

d – delete a partition

n – add a new partition

p – print the partition table

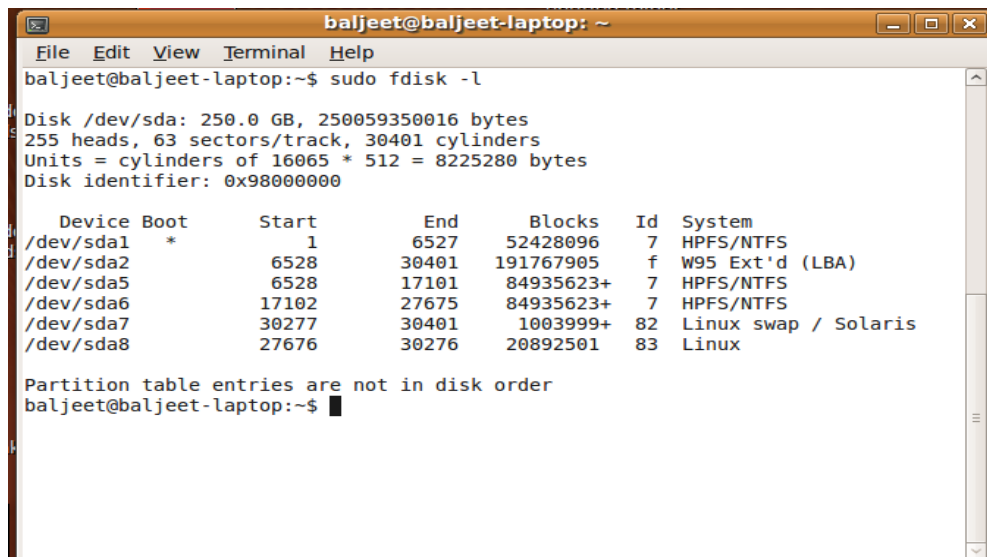
q – quit without saving changes

w - write table to disk and exit

m - display help

### Example:

`sudo fdisk -l`



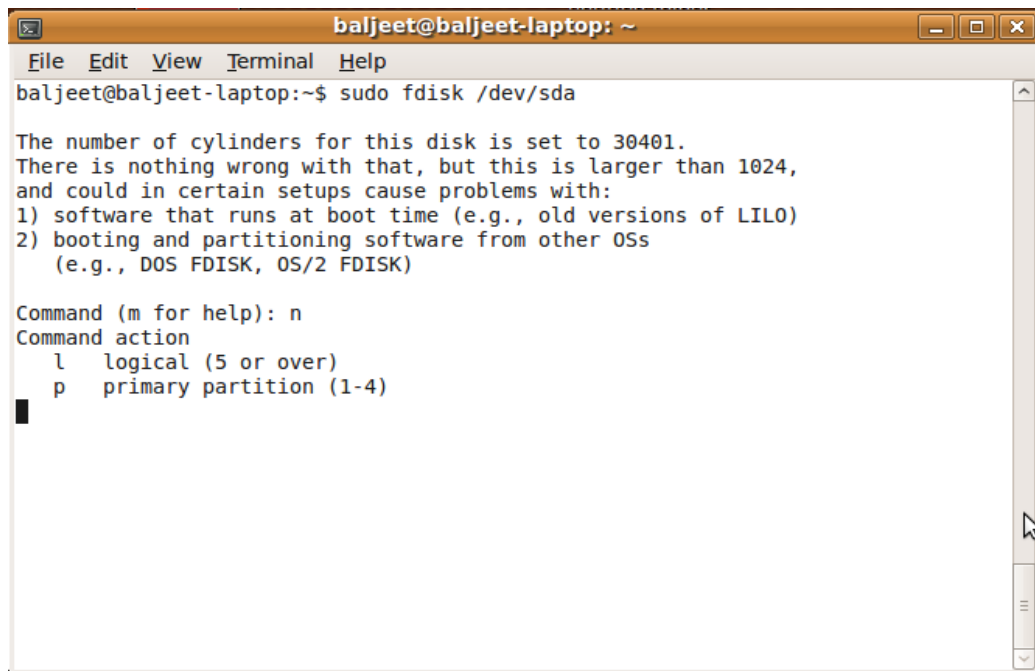
```
baljeet@baljeet-laptop: ~  
File Edit View Terminal Help  
baljeet@baljeet-laptop:~$ sudo fdisk -l  
  
Disk /dev/sda: 250.0 GB, 250059350016 bytes  
255 heads, 63 sectors/track, 30401 cylinders  
Units = cylinders of 16065 * 512 = 8225280 bytes  
Disk identifier: 0x98000000  
  
   Device Boot      Start         End      Blocks   Id  System  
/dev/sda1  *           1          6527       52428096    7   HPFS/NTFS  
/dev/sda2             6528        30401      191767905    f   W95 Ext'd (LBA)  
/dev/sda5             6528        17101       84935623+    7   HPFS/NTFS  
/dev/sda6            17102        27675       84935623+    7   HPFS/NTFS  
/dev/sda7            30277        30401        1003999+   82   Linux swap / Solaris  
/dev/sda8            27676        30276       20892501    83   Linux  
  
Partition table entries are not in disk order  
baljeet@baljeet-laptop:~$
```

*Figure 5.2.3*

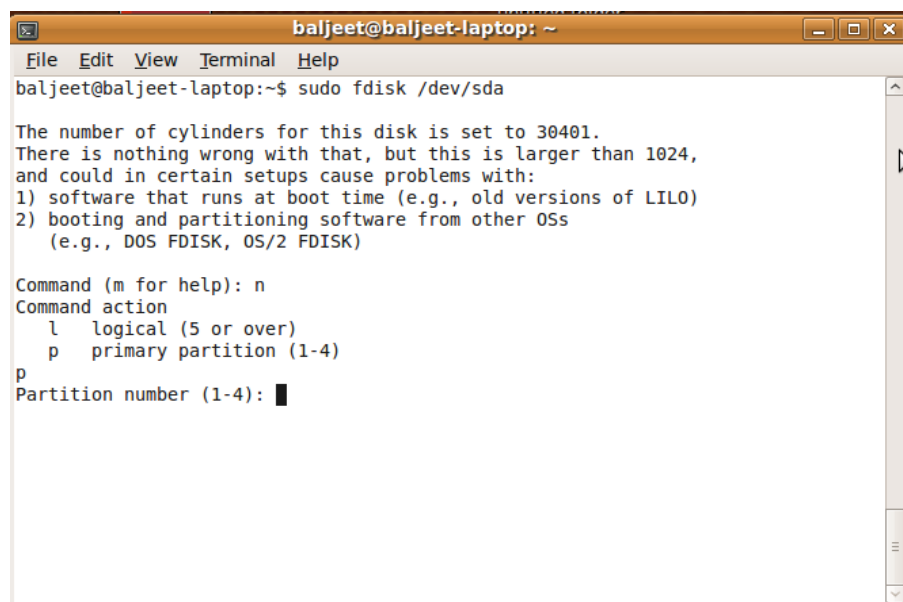
Example: Let's create a partition on the first SCSI HDD:

`fdisk /dev/sda`





```
baljeet@baljeet-laptop: ~  
File Edit View Terminal Help  
baljeet@baljeet-laptop:~$ sudo fdisk /dev/sda  
  
The number of cylinders for this disk is set to 30401.  
There is nothing wrong with that, but this is larger than 1024,  
and could in certain setups cause problems with:  
1) software that runs at boot time (e.g., old versions of LILO)  
2) booting and partitioning software from other OSs  
   (e.g., DOS FDISK, OS/2 FDISK)  
  
Command (m for help): n  
Command action  
   l   logical (5 or over)  
   p   primary partition (1-4)  
█
```



```
baljeet@baljeet-laptop: ~  
File Edit View Terminal Help  
baljeet@baljeet-laptop:~$ sudo fdisk /dev/sda  
  
The number of cylinders for this disk is set to 30401.  
There is nothing wrong with that, but this is larger than 1024,  
and could in certain setups cause problems with:  
1) software that runs at boot time (e.g., old versions of LILO)  
2) booting and partitioning software from other OSs  
   (e.g., DOS FDISK, OS/2 FDISK)  
  
Command (m for help): n  
Command action  
   l   logical (5 or over)  
   p   primary partition (1-4)  
p  
Partition number (1-4): █
```

Figure 5.2.4

## Step2- mkfs: creating file system

After creating partition, we need to create a file system on this partition to make it usable.

```
#mkfs -t ext2 /dev/sda3
```

This create ext2 file system – the standard file system that uses a block size of 1024 bytes.

---

## Now let's see in detail what is mkfs

- A file system that needs to be used must be created inside an existing drive/partition
- A partition must exist before you can create a file system.
- A file system must be created before it is mounted.
- The type defaults to the old ext2 type, but for modern Linux hard disk systems you should always specify the type explicitly as either ext3 or ext4.
- The pathname of the existing device or the partition that will be used is the `device_name`. This pathname is usually absolute. Usually the form is like `/dev/sdXN`, where the device or the disk being used is represented as X and the partition number is N on that device. It is the same how we have used while invoking `fdisk` command.
- The contents of partition are not checked by `mkfs` command. It will not take any confirmation from user before the partition is re-formatted. Whatever are the contents will be destroyed. So one needs to be very careful to specify the correct device name. Once the partition is destroyed, it cannot be recovered.
- System ID (type) of a partition given in the partition table is not the concern of `mkfs` command. Any type of file system can be created in any type of partition. For example, you can create an ext4 file system on a partition labelled in the partition table as NTFS or as Swap, however this is a bad idea.
- Creation of a partition with `fdisk` will not create any type of file system in that partition implicitly.
- The `mkfs` command creates the file system however it does not mount or make that file system available to use in Linux automatically.
- If `file -s` can't find a file system inside a disk partition, you cannot mount it – you probably forgot to make one using `mkfs`.

## Choosing a File System Type

Your system installation must have created “journaling” file systems on your virtual disk. This must be achieved by using the `-t ext4` option to `mkfs`. There is one more way to request this kind of file system i.e. use of `-j` (Journaling) option to the original `mke2fs` command.

This type of Linux journaling file system is usually called ext3 or ext4, and some distributions may have a small shell script named `mke3fs` or `mke4fs` that simply calls `mke2fs` with the appropriate `-t` or `-j` option.

The Journaling file systems are more resistant to corruption. The corruption might be caused due to sudden power loss. This is because of allowing the system to come back up more quickly

---

by avoiding a long file system check at boot time. This will not permit you to power off a running Linux system! Always a clean shut down is required. The correct command-line for an immediate, safe system shut down is:

```
# shutdown -h now      # shut down and halt (usually power off)
# shutdown -r now      # shut down and then reboot
```

Step3 – Mounting file system explained in the below section.



## Self-assessment Questions

- 1) The command partition the device into pieces using \_\_\_\_\_.
  - a) Fdisk
  - b) Mkfs
  - c) Mount
  - d) Fsck
- 2) The command format or prepare the partition with the chosen file system using \_\_\_\_\_.
  - a) Fdisk
  - b) Mkfs
  - c) Mount
  - d) Fsck
- 3) The command optionally verify the file system integrity – check for errors using \_\_\_\_\_.
  - a) Fdisk
  - b) Mkfs
  - c) Mount
  - d) Fsck
- 4) The command connect the file system into the existing directory tree using \_\_\_\_\_.
  - a) Fdisk
  - b) Mkfs
  - c) Mount
  - d) Fscke
- 5) Commands to create file system should appear in order of
  - a) mount, mkfs and fdisk
  - b) mkfs, fdisk and mount
  - c) fdisk, mkfs and mount
  - d) fdisk, mount and mkfss

---

## 5.2.2 Mounting and Unmounting File System

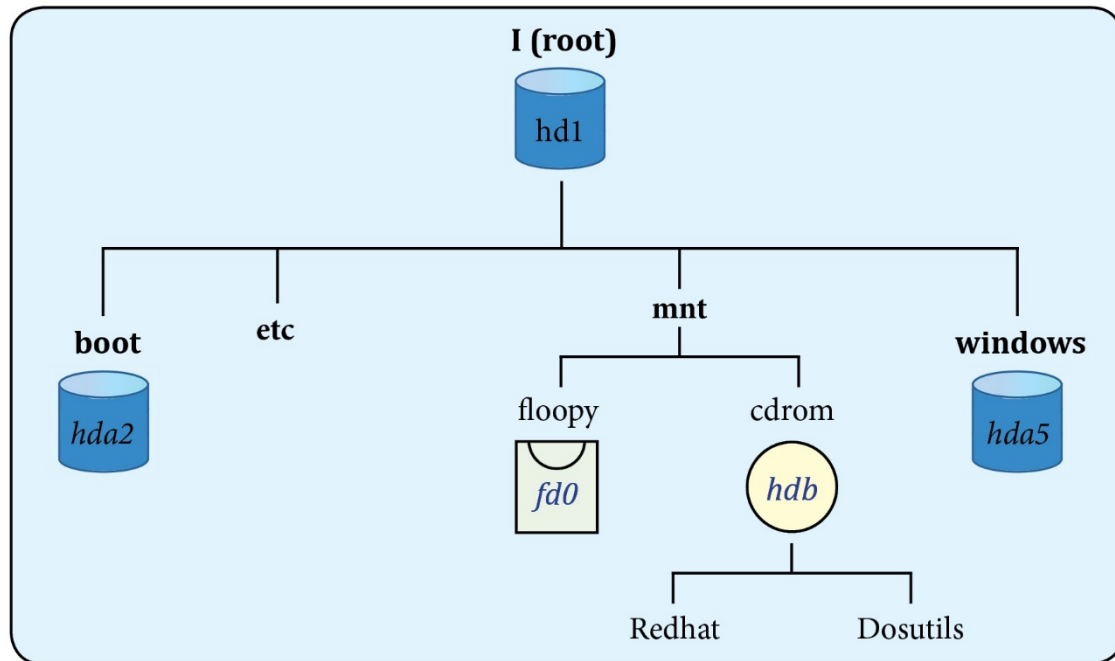


Figure 5.2.5

Linux has a single-ROOTed file system tree unlike Windows Operating System having multiple drive letters. “Mounting” attaches an existing file system found on a block device (usually a disk partition, e.g. /dev/hda2 or /dev/sda2) to the Linux directory structure, e.g. onto some directory /boot. To access the file system on a disk partition, one needs to access the directory and everything under that directory.

By attaching any number of separate file systems anywhere in the same Linux directory tree, one single ROOTed tree is created. This one single ROOTed tree is used to access every file on every disk. You can detach a file system from one place in the tree and attach it somewhere else. However in this scenario the pathnames for the files available inside the file system will change and will reflect the new mount location.

The mount command can be used to mount the file system from Unix/Linux command line. This can be done automatically at the time of system boot. At that time one needs to put their names and mount points into the **/etc/fstab** file. The file systems that are available on removable devices like USB keys, DVDs etc can also be mounted dynamically at device insertion time using rules in system configuration files (“automounting”).

---

(Most distribution use an existing `/mnt/` directory to attach file systems dynamically and temporarily to the directory structure, e.g. USB keys, CDROM and DVD, floppy disks, etc.) Desktop operating systems may also use a `/media/` directory for dynamic storage devices.

You can only mount file systems, not partitions.

**Note:** An empty partition (one created with `fdisk` but not initialized with `mkfs`) cannot be mounted. If `file -s` can't find a file system inside a disk partition, you cannot mount it – you probably forgot to make one using `mkfs`.

**Manual mount:** Mount command can be used along with various options based on the requirement. Let's now see in detail, how we can use this command.

- **mount**

When used without any options, lists all the currently mounted devices and their contained file systems. This is a very common command usage.

- **mount -a**

When used with `-a` mounts all the file systems listed for auto mounting in the `/etc/fstab` file. This is possible for root only.

- **mount [options] { device\_name | directory\_name }**

- This command mounts the file system in `/etc/fstab` that matches the given name.
- You give either the `device_name` or the `directory_name` of the entry in the `fstab` file. The system gets the other one from the `fstab`.

e.g. `mount /dev/sda1`      `# /etc/fstab` will supply the mount point

e.g. `mount /home`                      `# /etc/fstab` will supply the partition device name

- **mount [options] device\_name directory\_name**

- Ignore `/etc/fstab` and mount the file system inside the given `device_name` onto the `givedirectory_name` mount point.
- The partition must have a file system created in it (e.g. with **mkfs**).
- The `directory_name` must already be exist. (You may need to create it.)
- e.g. `mount /dev/sda1 /home`      `# /etc/fstab` is not used

---

## Details on using mount

- Syntax: **mount [options] device\_name directory\_name**
- The pathname which of the device containing the file system is the **device\_name**. This path name is usually absolute. e.g. a partition name such as /dev/sdb9. The device holds the partition name and not the disk name. To mount the file system, the partition must have a recognized file system created inside it already. This is because the file system inside a partition is mounted and not the partition.
- The **directory\_name** is the pathname (often absolute) of an existing Linux directory, usually an empty directory. While mounting a file system inside the partition, it will be mounted on this existing directory mount point. In case of failure, make sure the directory exists!
- If required, mkdir command can be used for creating a new empty directory mount point.
- You can “cover up” a non-empty directory by mounting on top of it; this temporarily hides the original contents of the directory and is not usually a good idea.
- The modern mount commands can identify the type of the file system residing inside a partition. However rarely you may need to specify the file system type in the partition. To specify this -t option is used.
- If the type can't automatically be recognized, it usually means the partition has no file system created on it at all. This is because you forgot to use mkfs first. To check this use file -s.
- •You can pass mount-time options using the -o option.
- the options are the same options you can use in the /etc/fstab file
- e.g. mount -o 'ro,noatime' /dev/fd0 /mnt/floppy
- If the user is mounting the file system (if not root), then the user must have access rights to the mount point directory and to the mount command.

**Important:** - Remember that when a file system is mounted onto a directory that already contains files, will temporarily hide those files until the file system is unmounted again.

### Examples of mount command:

- mkdir /mnt/foo ; mount /dev/sda1 /mnt/foo
- mkdir /mnt/dosfloppy ; mount -t vfat /dev/fd0 /mnt/dosfloppy
- mkdir /mnt/foobar ; mount /dev/sdb9 /mnt/foobar

Note: -t option is used to specify filesystem.

---

## Un-mounting a mounted file system: umount

- To unmount a file system, the command used is `umount`. Either the device name or mount point can be specified to this command but not both.
- `umount /dev/sda1`
- `umount /home`
- While attempting to unmount a file system, make sure that no files are open inside it and no programs have it as a current directory.
- If the message appears as “device is busy” that means a process is still accessing the device.
- A common mistake is to try (and fail) to unmount the file system of the current directory – you have to `cd` all your shells and processes out of the file system first.
- While attempting to unmount the file system sometimes you need to search and kill the processes that are using a file system as a current directory.
- The commands to list files that are open on a file system (use these as root, and give the Linux directory that is the root of the file system):
  - `fuser /boot`
  - `lsof /`



## Self-assessment Questions

- 6) Unlike Windows with its multiple drive letters, Linux has a \_\_\_\_\_ file system tree.
- |                    |                  |
|--------------------|------------------|
| a) multiple-ROOTed | b) single-ROOTed |
| c) multiple        | d) single        |
- 7) You can only mount file systems, not partitions.
- |         |          |
|---------|----------|
| a) TRUE | b) FALSE |
|---------|----------|
- 8) Which of the following statement is true:
- a) An empty partition can be mounted.
  - b) You can only mount file systems, not partitions.
  - c) You cannot mount file systems, but partitions.
  - d) You can only mount partitions.

---

## 5.2.3 Checking and Monitoring System Performance

Command	Description
nice/renice	Run a program with modified scheduling priority
Netstat	Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships
Time	Time a simple command or give resource usage
Uptime	System Load Average
Ps	Report a snapshot of the current processes.
Vmstat	Report virtual memory statistics
Gprof	Display call graph profile data
Prof	Process Profiling
Top	Display system tasks

You can use Manpage Help to check complete syntax for each command mentioned here.





## Self-assessment Questions

- 9) The command \_\_\_\_\_ run a program with modified scheduling priority.
- |                |            |
|----------------|------------|
| a) Nice/renice | b) Netstat |
| c) Time        | d) Uptime  |
- 10) The command \_\_\_\_\_ print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
- |                |            |
|----------------|------------|
| a) Nice/renice | b) Netstat |
| c) Time        | d) Uptime  |
- 11) The command \_\_\_\_\_time a simple command or give resource usage
- |                |            |
|----------------|------------|
| a) Nice/renice | b) Netstat |
| c) Time        | d) Uptime  |
- 12) The command \_\_\_\_\_ report virtual memory statistics.
- |                |            |
|----------------|------------|
| a) Nice/renice | b) Netstat |
| c) Vmstat      | d) Uptime  |

## 5.2.4 File Security and Permissions

ls -l

chmod

The important component of UNIX which provides a secure method for file storage is File Ownership. The attributes for every file in UNIX are as follows:

- **Owner permissions** – It specifies what all actions the owner of the file can perform on the file.
- **Group permissions** – It specifies what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions** – It specifies what action all other users can perform on the file.

---

## The Permission Indicators

The file permissions are indicating what all operations an user can perform related to file

```
$ls -l /home/amrood
```

```
-rwxr-xr--1amrood  users 1024Nov200:10myfile  
drwxr-xr--1amrood  users 1024Nov200:10mydir
```

The first column specifies different access mode ie. The permissions associated with a file or directory. The permissions are for read, write and execute. These are divided into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) –

- The first three characters (2-4) represent the permissions for the file's owner. For example `-rwxr-xr--` represents that owner has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example `-rwxr-xr--` represents that group has read (r) and execute (x) permission but no write permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example `-rwxr-xr--` represents that other world has read (r) only permission.

## File Access Modes

The permissions of a file is most important in Unix system. The basic Unix permissions are the **read**, **write**, and **execute** permissions, which are as follows –

### 1. Read

Allows to read ie. view the file contents.

### 2. Write

Allows to modify, or remove the content of the file.

### 3. Execute

The user with the execute permissions can run a file as a program.

---

---

## Directory Access Modes

The Directory access modes are listed and organized in the same way as any other file access modes are listed and organized. There are a few new things to be mentioned:

### 1. Read

Access to a directory means that the user can read the contents of the directory. The user can look at the filenames inside the directory.

### 2. Write

Access means that the user can add or delete files to the contents of the directory.

### 3. Execute

Executing a directory doesn't really make a lot of sense so think of this as a traverse permission.

A user must have execute access to the bin directory in order to execute ls or cd command.

## Changing Permissions

When it is required to change file or directory permissions, then the chmod (change mode) command will be used. The two ways of using chmod command are: symbolic mode and absolute mode.

Let's see how to use chmod in Symbolic Mode

When a beginner wants to modify file or directory permissions then to use the symbolic mode is the easiest way. By using symbolic permissions you can add, delete, or specify the permission set you want. While specifying this the operators should be used which are shown in the following table.

Chmod operator	Description
+	Adds the designated permission(s) to a file or directory.
-	Removes the designated permission(s) from a file or directory.

---

=	Sets the designated permission(s).
---	------------------------------------

Here's an example using testfile. Running `ls -l` on testfile shows that the file's permissions are as follows –

```
$ls -l testfile
-rwxrwxr--1amrood  users 1024Nov200:10testfile
```

Then each example `chmod` command from the preceding table is run on testfile, followed by `ls -l` so you can see the permission changes –

```
$chmodo+wxtestfile
$ls -l testfile
-rwxrwxrwx1amrood  users 1024Nov200:10testfile
$chmod u-x testfile
$ls -l testfile
-rw-rwxrwx1amrood  users 1024Nov200:10testfile
$chmod g=rxtestfile
$ls -l testfile
-rw-r-xrwx1amrood  users 1024Nov200:10testfile
```

Here's how you could combine these commands on a single line:

```
$chmodo+wx,u-x,g=rxtestfile
$ls -l testfile
-rw-r-xrwx1amrood  users 1024Nov200:10testfile
```

Using `chmod` with Absolute Permissions

---

Let's now see how to use `chmod` command with absolute permission. This is the second mode to modify permissions. It uses a number to specify each set of permissions for the file instead of operators. The table below shows the permission and the number used for that permission.

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

---

Here's an example using testfile. Running `ls -l` on testfile shows that the file's permissions are as follows –

```
$ls -l testfile  
  
-rwxrwxr--1amrood  users 1024Nov200:10testfile
```

Then each example `chmod` command from the preceding table is run on testfile, followed by `ls -l` so you can see the permission changes –

```
$ chmod755testfile  
  
$ls -l testfile  
  
-rwxr-xr-x 1amrood  users 1024Nov200:10testfile  
  
$chmod743testfile  
  
$ls -l testfile  
  
-rwxr---wx1amrood  users 1024Nov200:10testfile  
  
$chmod043testfile  
  
$ls -l testfile  
  
---r---wx1amrood  users 1024Nov200:10testfile
```

## Changing Owners and Groups

When a user creates an account on Unix, every user is assigned with an owner ID and a group ID. The permissions those are mentioned above are assigned to the users based on Owner and Groups.

To modify the owner and the group of files, there are two commands as follows. –

- **chown** – This command stands for "change owner". It is used to change the owner of a file.
- **chgrp** – This command stands for "change group". It is used to change the group of a file.

---

The **chown** command is used to change the ownership of a file. The basic syntax for this is as follows –

```
$ chown user filelist
```

The value of user can be either the name of a user on the system or the user id (uid) of a user on the system.

Following example –

```
$ chownamroodtestfile  
  
$
```

Changes the owner of the given file to the user amrood.

NOTE: The super user, root, has the unrestricted capability to change the ownership of a any file but normal users can change only the owner of files they own.

### Changing Group Ownership

The **chgrp** command is used to change the group ownership of a file. The basic syntax is as follows –

```
$ chgrpgroupfilelist
```

The value of group can be the name of a group on the system or the group ID (GID) of a group on the system.

Following example –

```
$ chgrp special testfile  
  
$
```

Changes the group of the given file to **special** group.



## Self-assessment Questions

- 13) A user must have \_\_\_\_\_ access to the bin directory in order to execute ls or cd command.
- |            |                   |
|------------|-------------------|
| a) Read    | b) Write          |
| c) Execute | d) Read and write |
- 14) The chown command changes the ownership of a file.
- |         |          |
|---------|----------|
| a) True | b) False |
|---------|----------|
- 15) The chgrp command changes the group ownership of a file.
- |         |          |
|---------|----------|
| a) True | b) False |
|---------|----------|

## 5.2.5 Becoming Super User using su command

NAME

su - change user ID or become superuser

SYNOPSIS

su [options] [username]

### DESCRIPTION

The **su** command is used to become another user during a login session. Invoked without a **username**, **su** defaults to becoming the superuser. The optional argument **-** may be used to provide an environment similar to what the user would expect had the user logged in directly. Additional arguments may be provided after the username, in which case they are supplied to the user's login shell. In particular, an argument of **-c** will cause the next argument to be treated as a command by most command interpreters. The command will be executed by the shell specified in `/etc/passwd` for the target user.

You can use the **--** argument to separate **su** options from the arguments supplied to the shell. The user will be prompted for a password, if appropriate. Invalid passwords will produce an error message. All attempts, both valid and invalid, are logged to detect abuse of the system.



---

The current environment is passed to the new shell. The value of **\$PATH** is reset to /bin:/usr/bin for normal users, or /sbin:/bin:/usr/sbin:/usr/bin for the superuser. This may be changed with the **ENV\_PATH** and **ENV\_SUPATH** definitions in /etc/login.defs.

A subsystem login is indicated by the presence of a "\*" as the first character of the login shell. The given home directory will be used as the root of a new file system which the user is actually logged into.

## OPTIONS

The options which apply to the **su** command are:

**-c, --command**COMMAND

Specify a command that will be invoked by the shell using its **-c**.

The executed command will have no controlling terminal. This option cannot be used to execute interactive programs which need a controlling TTY.

**-, -l, --login**

Provide an environment similar to what the user would expect had the user logged in directly.

When **-** is used, it must be specified as the last **su** option. The other forms (**-l** and **--login**) do not have this restriction.

**-s, --shell**SHELL

The shell that will be invoked. The invoked shell is chosen from (highest priority first):

The shell specified with **--shell**.

If **--preserve-environment** is used, the shell specified by the **\$SHELL** environment variable.

The shell indicated in the /etc/passwd entry for the target user.

/bin/sh if a shell could not be found by any above method.

If the target user has a restricted shell (i.e. the shell field of this user's entry in /etc/passwd is not listed in /etc/shells),

---

Then the **--shell** option or the **\$SHELL** environment variable won't be taken into account, unless **su** is called by root.

#### **-m, -p, --preserve-environment**

Preserve the current environment, except for:

**\$PATH** reset according to the `/etc/login.defs` options **ENV\_PATH** or

**ENV\_SUPATH** (see below);

**\$IFS** reset to “<space><tab><newline>”, if it was set. If the target user has a restricted shell, is option has no

effect (unless **su** is called by root).

Note that the default behavior for the environment is the following:

The **\$HOME**, **\$SHELL**, **\$USER**, **\$LOGNAME**, **\$PATH**, and **\$IFS** environment variables are reset.

If **--login** is not used, the environment is copied, except for the variables above.

If **--login** is used, the **\$TERM**, **\$COLORTERM**, **\$DISPLAY**, and

**\$XAUTHORITY** environment variables are copied if they were set.

Other environments might be set by PAM modules.

#### CAVEATS

This version of **su** has many compilation options, only some of which may be in use at any particular site.

#### CONFIGURATION

The following configuration variables in `/etc/login.defs` change the behavior of this tool:

##### **CONSOLE\_GROUPS** (string)

List of groups to add to the user's supplementary groups set when logging in on the console (as determined by the **CONSOLE** setting).

---

Default is none.

Use with caution - it is possible for users to gain permanent access to these groups, even when not logged in on the console.

**DEFAULT\_HOME** (boolean)

Indicate if login is allowed if we can't cd to the home directory.

Default is no.

If set to yes, the user will login in the root (/) directory if it is not possible to cd to her home directory.

**ENV\_PATH** (string)

If set, it will be used to define the PATH environment variable when a regular user login. The value is a colon separated list of paths (for example /bin:/usr/bin) and can be preceded by PATH=. The default value is PATH=/bin:/usr/bin.

**ENV\_SUPATH** (string)

If set, it will be used to define the PATH environment variable when the superuser login. The value is a colon separated list of paths (for example /sbin:/bin:/usr/sbin:/usr/bin) and can be preceded by PATH=. The default value is PATH=/sbin:/bin:/usr/sbin:/usr/bin.

**SULOG\_FILE** (string)

If defined, all su activity is logged to this file.

**SU\_NAME** (string)

If defined, the command name to display when running "su -". For example, if this is defined as "su" then a "ps" will display the

---

command is "-su". If not defined, then "ps" would display the name of the shell actually being run, e.g. something like "-sh".

#### **SYSLOG\_SU\_ENAB** (boolean)

Enable "syslog" logging of **su** activity - in addition to sulog file logging.

#### FILES

/etc/passwd

User account information.

/etc/shadow

Secure user account information.

/etc/login.defs

Shadow password suite configuration.

#### EXIT VALUES

On success, **su** returns the exit value of the command it executed.

If this command was terminated by a signal, **su** returns the number of this signal plus 128.

If **su** has to kill the command (because it was asked to terminate, and the command did not terminate in time), **su** returns 255.

Some exit values from **su** are independent from the executed command:

0

success (**--help** only)

1

---

---

System or authentication failure

126

The requested command was not found

127

The requested command could not be executed

## 5.2.6 Getting System Information using uname command

### NAME

uname - print system information

### SYNOPSIS

**uname** [*OPTION*]...

### DESCRIPTION

Print certain system information. With no *OPTION*, same as **-s**.

Tag	Description
<b>-a, --all</b>	print all information, in the following order, except omit <b>-p</b> and <b>-i</b> if unknown:
<b>-s, --kernel-name</b>	
	print the kernel name

---

<b>-n, --nodename</b>	
	print the network node hostname
<b>-r, --kernel-release</b>	
	print the kernel release
<b>-v, --kernel-version</b>	
	print the kernel version
<b>-m, --machine</b>	
	print the machine hardware name
<b>-p, --processor</b>	
	print the processor type or "unknown"
<b>-i, --hardware-platform</b>	
	print the hardware platform or "unknown"
<b>-o, --operating-system</b>	

---

---

	print the operating system
<b>--help</b>	display this help and exit
<b>--version</b>	
	output version information and exit

## 5.2.7 Installing and Removing Packages using rpm command

Installing, uninstalling, upgrading, querying, listing, and checking RPM packages on Linux system is done by using RPM command.

RPM stands for **Red Hat Package Manager**.

To manage the RPM software packages, the RPM command can be used. While using this one will have root privilege. Also the RPM command should be used with appropriate options.

Let us take an rpm of Mysql Client and run through all our examples.

### - Installing a RPM package Using rpm -ivh

RPM filename has packagename, version, release and architecture name.

For example, In the MySQL-client-3.23.57-1.i386.rpm file:

- MySQL-client – Package Name
- 3.23.57 – Version
- 1 – Release
- i386 – Architecture

While installing RPM, it checks if the system is suitable for the software those are available in PRM package. It also figures out the location to install the files available in PRM package. Then

---

it installs on the system and adds that piece of software into its database of installed RPM packages.

The following rpm command installs Mysql client package.

```
# rpm -ivh MySQL-client-3.23.57-1.i386.rpm

Preparing...      ##### [100%]
1:MySQL-client    ##### [100%]
```

rpm command and options

- -i : install a package
- -v : verbose
- -h : print hash marks as the package archive is unpacked.

dpkg on Debian, pkgadd on Solaris, depot on HP-UX can also be used to install packages.

### **- Query all the RPM Packages using rpm -qa**

You can use rpm command to query all the packages installed in your system.

```
# rpm -qa

cdrecord-2.01-10.7.el5

bluez-libs-3.7-1.1

setarch-2.0-1.1

.

.
```

- -q query operation
- -a queries all installed packages



Many times it is required before installation that whether a particular package is already installed on the system or not. To achieve this the rpm and grep command are used in combination as shown below. The following command checks whether cd record package is installed on your system.

```
# rpm -qa | grep 'cdrecord'
```

## -Query a Particular RPM Package using rpm -q

The above example shows all packages those are currently installed. After installation of a package to check the installation, you can query a particular package and verify as shown below.

```
# rpm -q MySQL-client
```

MySQL-client-3.23.57-1

```
# rpm -q MySQL
```

package MySQL is not installed

Note: To query a package, you should specify the exact package name. If the package name is incorrect, then rpm command will report that the package is not installed.



## Self-assessment Questions

16) You can find User account information in \_\_\_\_\_.

- a) /etc/passwd                      b) /etc/shadow  
c) /etc/login                        d) /etc/login.defs

17) You can find secure User account information in \_\_\_\_\_.

- a) /etc/passwd                      b) /etc/shadow  
c) /etc/login                        d) /etc/login.defs

18) You can find Shadow password suite configuration in \_\_\_\_\_.

- a) /etc/passwd                      b) /etc/shadow  
c) /etc/login                        d) /etc/login.defs



## Summary

- A file system can be created by using the commands in order i.e. fdisk, mkfs and mount.
- A partition is section of a physical disk. It usually formatted to contain a file system. Each partition table entry specifies its disk start location, end location, and whether it is bootable.
- Linux can run inside only a single partition, the ROOT partition, but most Linux systems use at least two partitions: A ROOT partition, a swap partition. A file system needs to be created inside an existing drive/partition. Before creating a file system, you must have an existing partition. You must have a file system created before you can mount it.
- Your system install likely created “journaling” file systems on your virtual disk, using the -t ext4 option to mkfs.
- Linux has a single-Rooted file system tree unlike Windows Operating System. “Mounting” attaches an existing file system found on a block device (usually a disk partition, e.g. /dev/hda2 or /dev/sda2) to the Linux directory structure, e.g. onto some directory /boot.
- You can un-mount a file system using the umount command and give either the device name or the mount point, but not both.
- There are some free tools available to monitor and manage performance on UNIX systems, and to provide a guideline on how to diagnose and fix performance problems in Unix environment.
- File ownership is an important component of UNIX that provides a secure method for storing files.
- To change file or directory permissions, you use the chmod (change mode) command. There are two ways to use chmod: symbolic mode and absolute mode.
- The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode. With symbolic permissions you can add, delete, or specify the permission set you want by using the operators.

- 
- `chown` – The `chown` command stands for "change owner" and is used to change the owner of a file.
  - `chgrp` – The `chgrp` command stands for "change group" and is used to change the group of a file.
  - The super user, root, has the unrestricted capability to change the ownership of a any file but normal users can change only the owner of files they own.
  - The `su` command is used to become another user during a login session.
  - Invoked without a username, `su` defaults to becoming the superuser. The
  - Optional argument `-` may be used to provide an environment similar to
  - What the user would expect had the user logged in directly.
  - `RPM` command is used for installing, uninstalling, upgrading, querying, listing, and checking RPM packages on your Linux system. RPM stands for Red Hat Package Manager. With root privilege, you can use the `rpm` command with appropriate options to manage the RPM software packages.



## Terminal Questions

1. Explain different types of file system available in Linux. Also describe how to create a new file system after partitioning of hard disk.
2. Illustrate mounting and unmounting file system with appropriate example.
3. Explain the process of checking and monitoring system performance.
4. How to become a super user? Discuss file security and permission in detail.
5. Illustrate how to extract system information using `uname` command with appropriate example.



## Answer Keys

Self-assessment Questions	
Question No.	Answer
1	a
2	b
3	d
4	c
5	c
6	b
7	a
8	b
9	a
10	b
11	c
12	c
13	c
14	a
15	a
16	a
17	b
18	d



## Activity

**Activity Type:** Offline/Online

**Duration:** 30 Minutes

**Description:**

Do a research on system performance monitoring tools and prepare report on that.

---

# Bibliography



## e-References

- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix  
[http://www.iddevelopment.info/data/Unix/Linux/LINUX\\_PartitioningandFormattin gSecondHardDrive\\_ext3.shtml](http://www.iddevelopment.info/data/Unix/Linux/LINUX_PartitioningandFormattin gSecondHardDrive_ext3.shtml)
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix <http://www.tecmint.com/command-line-tools-to-monitor-linux-performance/>
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix <http://www.ansoncheunghk.info/article/7-useful-command-tools-monitor-linux-performance>
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix <http://www.rpm.org/max-rpm/ch-rpm-erase.html>
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix <http://www.thegeekstuff.com/2010/07/rpm-command-examples/>
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix  
[http://www.iddevelopment.info/data/Unix/Linux/LINUX\\_PartitioningandFormattin gSecondHardDrive\\_ext3.shtml](http://www.iddevelopment.info/data/Unix/Linux/LINUX_PartitioningandFormattin gSecondHardDrive_ext3.shtml)
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix <http://www.tecmint.com/command-line-tools-to-monitor-linux-performance/>
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix <http://www.ansoncheunghk.info/article/7-useful-command-tools-monitor-linux-performance>
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix <http://www.rpm.org/max-rpm/ch-rpm-erase.html>

- 
- This website was referred on 3rd May 2016, while developing content for advance tasks in Unix <http://www.thegeekstuff.com/2010/07/rpm-command-examples/>



## External Resources

- Maurice J. Bach, The Design of Unix Operating System, (2010) Pearson Education
- S. Prata, Advance UNIX, a Programmer's Guide, (2011), BPB Publications, and New Delhi,
- B.W. Kernighan & R. Pike, The UNIX Programming Environment, (2009) Prentice Hall of India.
- Jack Dent Tony Gaddis, Guide to UNIX Using LINUX, (2010) Vikas/ Thomson Pub. House Pvt. Ltd.



## Video Links

Topic	Link
The Linux File System	<a href="https://www.youtube.com/watch?v=2qQTXp4rBEE">https://www.youtube.com/watch?v=2qQTXp4rBEE</a>
Directory structure of the UNIX file system	<a href="https://www.youtube.com/watch?v=PEmi550E7zw">https://www.youtube.com/watch?v=PEmi550E7zw</a>



**Notes:**

