# UNIT 2 NETWORK ADMINISTRATION ACTIVITIES

**Structure** **Page Nos.**



## 2.0 INTRODUCTION

System administration is a critical activity that includes installation of software , configuration, commissioning, monitoring and finally problem diagnosis & troubleshooting of systems. It is a challenging task to the system administrator to ensure the uptime of the systems all times. Creation of a suitable file system with adequate disk partitions requires the system administrator enough working knowledge. Once system commissioning is over, the administrator has to hardening of OS in terms of removing unnecessary packages that come along with OS, close unnecessary ports and other such related. System and kernel management is another activity where the administrator has to keep on doing as part o regular system resources monitoring. It is very essential to the system administrator to know and use at least some useful tools/commands that are being used as part of problem diagnosis and troubleshooting so that system health can be maintained always and in turn make the systems uptime always.

## 2.1 OBJECTIVES

After going through this unit, you will be able to:

- know how to manage software packages;
- understand what is a file system and its structure;
- know how to manage users;
- understand the management of system and kernel; and
- find commands that are useful for troubleshooting.

## 2.2 MANAGING SOFTWARE PACKAGES

Software Packages are archives of files that include all the files that make up a piece of software(such as an application itself, shared libraries, development packages containing files that needed to build software against a library, etc) and also has a set of instructions to make them work. A package is properly integrated into the

distribution it has been built for installation paths, dependencies, desktop integration, and proper startup scripts for servers, etc.

**Package manager**

Generally, in modern Linux distributions like openSUSE, software installation is done with a package manager. Package manager is a collection of software tools to automate the process of installing, upgrading, configuring, and removing software packages for a computer's operating system. It works on top of RPM (Red hat Package Manager) and gets software packages from repositories (such as online servers, CDs and DVDs), solves the dependencies and installs them on your system. The package manager also makes it easy to remove packages later or to update them. The number of packages available for installation depends on which repositories you have added. Package manager typically maintains a database of software dependencies and version information to prevent software mismatches and missing prerequisites.

**Package management systems**

Package management systems are designed to save organizations time and money through remote administration and software distribution technology that eliminate the need for manual installs and updates. This can be particularly useful for large enterprises whose operating systems are based on Linux and other Unix-like systems, typically consisting of hundreds or even thousands of distinct software packages.

**Package Metadata**

Packages are distributions of software, applications and data. Packages also contain metadata that contains

- Summary (software name, etc)
- Description
- A list of files contained in the package
- The version of the software it contains as well as the release number of the package
- When, where and by whom it has been built
- What architecture it has been built for
- Checksums of the files contained in the package
- The license of the software it contains
- Which other packages it requires to work properly

After installation, metadata is stored in a local package database.

**Package dependencies**

The important aspect of the packages archive is the relations they contain. The packages also relates files to other packages as the packaged applications need an execution environment (like other tools, libraries, etc.) to actually run the application. Package dependencies are used to express such relations.

For example, package A needs the packages of B, C and D to be installed in order to work properly.

Package dependencies are transitive, which means that when package A needs package B, and package B needs package C, package A also needs package C, which is why you sometimes end up with lots of packages to install although you just wanted that one application.

Dependencies on libraries (typically packages with a name that starts with "lib") are very common and pretty much every single application depends on a set of library packages.

Packages and package dependencies are very important aspects of Linux distributions (as well as other BSD and UNIX systems) because they provide a modular way to set up and manage an operating system and its applications. This is especially true for library packages. For example, the package `openssl` contains cryptographic libraries that are used by many applications and other libraries (e.g. for SSL encryption). When a new, improved version of `openssl` is available, all the applications that use it will benefit from it just by upgrading that single package to the newer version. It is also a very efficient way to maintain a stable and secure system. It means, when a security hole, exploit or bug affects a library used by one or many applications, upgrading the single package will fix it for all of them. Figure 1 Shows the linkages between user systems, package manager, repository, meta data, packages and package dependencies.

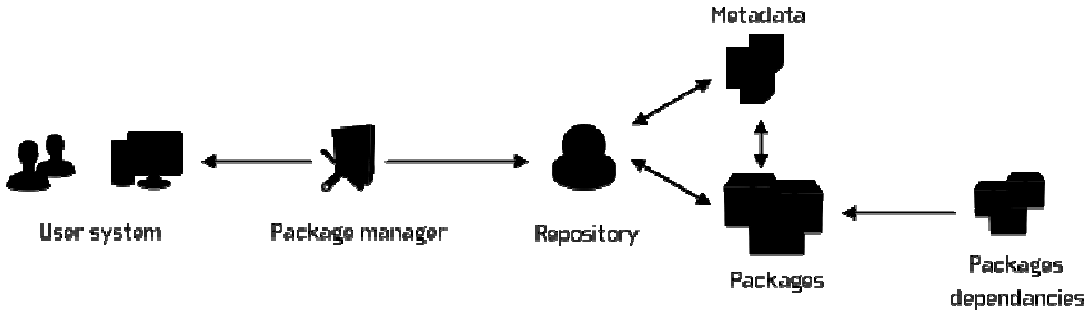



**Figure 1: Linkage between user systems, package manager and repository**

### Package formats

In Linux distribution, the native software comes in three kinds of package formats.

- tgz (tar gzip files), which are basically source code archives and can hold anything the package maintainer thinks useful. Apart from the archive format itself, it necessary to extract the files, but there is nothing standardized about the content of a tgz file. They need to be compiled in order to run the software.

- rpm (RPM Package Manager), which are pre-compiled archives that are created by Red Hat Linux and standardized by the LSB (Linux Standard Base- to lower the overall costs of supporting the Linux platform)..

- deb (Debian) which are pre-compiled archives that are used on Debian based system.

However, if the archives format noticed the system of the required dependencies, they don't provide the dependency management capability and they will just present any encountered problem to the user at first sight, and leave it for user to decide what to do.

For instance if you want to install a RPM package **A** that has dependencies to RPM package **B**, RPM will not automatically install package **B** but just tell you that it needs package **B** and stop. It's up to the user to install package **B** and then afterwards package **A**. Now imagine package **B** has dependencies on package **C** and package **D** and package **D** has dependencies to package **E** and so on and so on. You end up chasing package dependencies manually down all the branches of this very huge tree.

**RPM : rpm Package Manager**

rpm is a powerful package manager, which can be used to build, install, query, verify, update, and erase individual software packages. A package consists of an archive of files and meta-data used to install and erase the archive files. The meta-data includes helper scripts, file attributes, and descriptive information about the package. Packages come in two varieties: binary packages, used to encapsulate software to be installed, and source packages, containing the source code and recipe necessary to produce binary packages.

One of the following basic modes must be selected: Query, Verify, Signature Check, Install/Upgrade/Freshen, Uninstall, Initialize Database, Rebuild Database, Resign, Add Signature, Set Owners/Groups, Show Querytags, and Show Configuration.

**Use of rpm with some options**

The general form of an rpm install command is

**rpm** {**-i|--install**} [**install-options**] *PACKAGE_FILE ...*

The general form of an rpm upgrade command is

**rpm** {**-U|--upgrade**} [**install-options**] *PACKAGE_FILE ...*

This upgrades or installs the package currently installed to a newer version. This is the same as install, except all other version(s) of the package are removed after the new package is installed.

**rpm** {**-F|--freshen**} [**install-options**] *PACKAGE_FILE ...*

This will upgrade packages, but only if an earlier version currently exists. The *PACKAGE_FILE* may be specified as an **ftp** or **http** URL, in which case the package will be downloaded before being installed.

## 2.3 FILE SYSTEM

A file system is a way in which files are stored, accessed, overwritten, and deleted on a media format by using a computer and the operating system (OS) installed onto that computer. Different types of OS typically have different file systems, though they are often somewhat similar in design and how files are accessed through the OS. Generally, a number of different types of file systems that have been designed and implemented, with new types being created and used by newer OS versions. A file system at its most basic level be defined as the way in which data is stored in individual files on a computer hard drive or other media, and how that data is then accessed again in the future. File system have methods and data structures that an operating system uses to keep track of files on a disk or partition and the way the files are organized on the disk. A computer user will typically work on a computer to create, alter, save, and delete a variety of files for different purposes.

Some file systems are used on local data storage devices; others provide file access via a network protocol (e.g. NFS, SMB, etc). Some file systems are "virtual", in that the "files" supplied are computed on request (e.g. procfs) or are merely a mapping into a different file system used as a backing store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

## Aspects of file systems

Following are main aspects of file systems:

### Space management

File systems allocate space in a granular manner, usually multiple physical units on the device. The file system is responsible for organizing files and directories, and keep track of which areas of the media belong to which file and which are not being used.

### File system fragmentation

Fragmentation occurs when unused space or single files are not contiguous. Once file system is built, files are created, modified and deleted. When a file is created, the file system allocates space for the data. Some file systems permit or require specifying an initial space allocation and subsequent incremental allocations as the file grows. If files are deleted, the space reserved for it will be allocated and use by other files. This creates used and unused areas of various sizes. This is called as a free space fragmentation. When a file is created, and there is no contiguous space available for its initial allocation, the space must be assigned in fragments. When a file is modified such that it becomes larger it may exceed the space initially fragmented.

### Filenames

A filename is used to identify a storage location in the file system. Most file systems have restrictions on the length of filenames. In some file systems, filenames are case-insensitive (i.e. filenames such as ‘ABC’ and ‘abc’ refer to the same file); in others, filenames are case-sensitive (i.e. the names ‘ABC’ and ‘abc’ refer to two separate files).

Most modern file systems allow filenames to contain a wide range of characters from the Unicode character set. Most file system interface utilities have restrictions on the use of certain special characters, disallowing them within filenames (the file system may use these special characters to indicate a device, device type, directory prefix, or file type). However, these special characters might be allowed by enclosing the filename with double quotes ("). For simplicity, special characters are generally discouraged within filenames.

### Directories

File systems typically have directories (also called folders) which allow the user to group files into separate collections. This may be implemented by associating the file name with an index in a table of contents or an inode in a Unix-like file system. Directory structures may be flat (i.e. linear), or allow hierarchies where directories may contain subdirectories.

### Metadata

Metadata is a information that is typically associated with each file within a file system. The length of the data contained in a file may be stored as the number of blocks allocated for the file or as a byte count. The time that the file was last modified may be stored as the file's timestamp. File systems might store the file creation time, the time it was last accessed, the time the file's meta-data was changed, or the time the file was last backed up. Other information includes the file's device type (e.g. block, character, socket, subdirectory, etc.), its owner user ID and group ID, its access permissions and other file attributes (e.g. whether the file is read-only, executable, etc.).

### Design limitations

All file systems have some functional limit that defines the maximum storable data capacity within that system. These functional limits are a best-guess effort by the designer based on how large the storage systems are right now and how large storage systems are likely to become in the future. Disk storage has continued to increase at near exponential rates, so after a few years, file systems have kept reaching design limitations that require computer users to repeatedly move to a newer system with ever-greater capacity.

File system complexity typically varies proportionally with the available storage capacity. The file systems of early 1980s home computers with 50 KB to 512 KB of storage would not be a reasonable choice for modern storage systems with hundreds of gigabytes of capacity. Likewise, modern file systems would not be a reasonable choice for these early systems, since the complexity of modern file system structures would consume most or all of the very limited capacity of the early storage systems.

### Types of file systems

The following are some of the file system types and classified into disk/tape file systems, network file systems and special-purpose file systems.

### Disk file systems

Disk file systems are the file systems which manage data on permanent storage devices. Magnetic disks are the most common of such devices. A disk file system takes advantages of the ability of disk storage media to randomly address data in a short amount of time. In disk files systems, data access is fast. It supports multiple users (or processes) to access various data on the disk. Some of the example disk file systems are FAT (FAT12, FAT16, FAT32), exFAT, NTFS, ext3, ext4, etc.

In a disk file system, there is typically a master file directory, and a map of used and free data regions. Any file additions, changes, or removals require updating the directory and the used/free maps. Random access to data regions is measured in milliseconds so this system works well for disks.

### Optical file system

Optical media use different file systems than hard disks or flash media, because of the write-once nature of most optical discs. Even rewritable discs use different file systems because of the way that rewritable media is managed. ISO 9660 and Universal Disk Format (UDF) are two common formats for Compact Discs, DVDs and Blu-ray discs.

### Flash file systems

A flash file system is a file system designed for storing files on flash memory devices. These are becoming more common as the number of mobile devices is increasing, the cost per memory size decreases, and the capacity of flash memories increases.

### Tape file systems

A tape file system is a file system and tape format designed to store files on tape in a self-describing form. Magnetic tapes are sequential storage media with significantly longer random data access times than disks. Tape requires linear motion to wind and unwind potentially very long reels of media. This tape motion may take several seconds to several minutes to move the read/write head from one end of the tape to the other.

### Network file systems

A network file system is a file system that acts as a client for a remote file access protocol, providing access to files on a server. Examples of network file systems include clients for the NFS, AFS, SMB protocols, and file-system-like clients for FTP and WebDAV(Web Distributed Authoring and Versioning). WebDAV is an extension of the Hypertext Transfer Protocol (HTTP) that facilitates collaboration between users in editing and managing documents and files stored on World Wide Web servers.

### Shared disk file systems

A shared disk file system is one in which a number of machines (usually servers) have access to the same external disk subsystem (usually a SAN). The file system arbitrates access to that subsystem, preventing write collisions. Examples include GFS2 from Red Hat, GPFS from IBM, etc.

### Special file systems

A special file system presents non-file elements of an operating system as files so they can be acted on using file system APIs. This is most commonly done in Unix-like operating systems, but devices are given file names in some non-Unix-like operating systems as well.

### Flat file systems

A flat file system is a system of files in which every file in the system must have a different name. Flat file system stores all its files in the same directory. In this system, every file has a different name since there is only one list of files. Flat file systems cannot contain multiple tables.

In Windows 95 and most other operating system today, files are managed in a hierarchical file system with a hierarchy of directories and subdirectories, each containing a number of files (or subdirectories). The operating system allows more than one file to have the same name as long as it is stored in a different directory. Early versions of the Macintosh and DOS operating systems used a flat file system.

### File systems and operating systems

Many operating systems require support for more than one file system. Sometimes the OS and the file system are so tightly interlink, due to this, it is difficult to separate out the file system.

There needs to be an interface provided by the operating system software between the user and the file system. This interface can be textual (such as provided by a command line interface, such as the Unix shell) or graphical (such as provided by a graphical user interface, such as file browsers).

### Unix-like operating systems

Unix-like operating systems create a virtual file system, which makes all the files on all the devices appear to exist in a single hierarchy. This means, there is one root directory, and every file existing on the system is located under it somewhere. Unix-like systems can use a RAM disk or network shared resource as its root directory. Figure 2 shows a sample Unix directory structure.

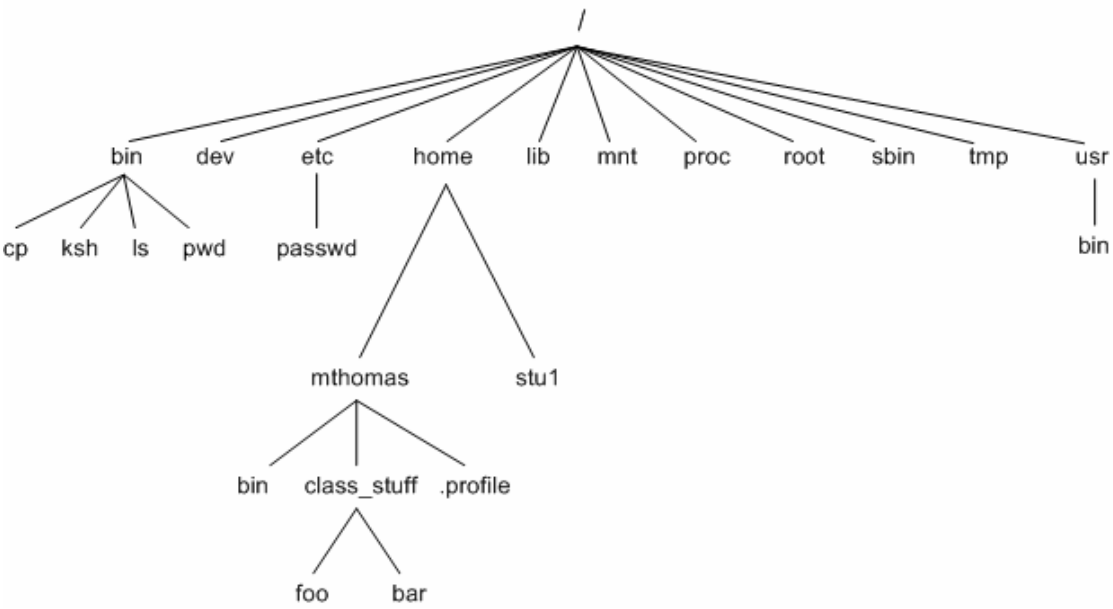



**Figure 2: Unix directory Structure**

In order to access a file system in UNIX, first it needs to mount it. Mounting a file system is simply making the particular file system accessible at a certain point in the Unix directory tree. When mounting a file system, it does not matter, if the file system is in a hard disk partition, CD-ROM, floppy, or USB storage device. You simply need to know the device name associated with the particular storage device and a directory you would like to mount it to. Unix-like operating systems often include software and tools that assist in the mounting process.

In many situations, file systems other than the root need to be available as soon as the operating system has booted. All Unix-like systems therefore provide a facility for mounting file systems at boot time. System administrators define these file systems in the configuration file fstab (*vfstab* in Solaris), which also indicates options and mount points.

In some situations, there is no need to mount certain file systems at boot time, although their use may be desired thereafter. There are some utilities for Unix-like systems that allow the mounting of predefined file systems upon demand.

Removable media have become very common with microcomputer platforms. They allow programs and data to be transferred between machines without a physical connection. Common examples include USB flash drives, CD-ROMs, and DVDs. Utilities have therefore been developed to detect the presence and availability of a medium and then mount that medium without any user intervention.

An automounter is a program or software facility which automatically mounts file systems in response to access operations by user programs. This is usually used for file systems on network servers, rather than relying on events such as the insertion of media, as would be appropriate for removable media.

**Linux Operating System**

The Linux file system is a hierarchically structured tree where every location has its distinct meaning. Linux supports many different file systems. It is similar to Unix file system. The file system is a tree-shaped structure. The root of the tree, which is called the 'file system root' but is always depicted as being above all other and is

identified by the slash character "/". It is the highest place you can go to and beneath it are almost always only directories. Figure 3 shows the sample directory of a Linux file system.

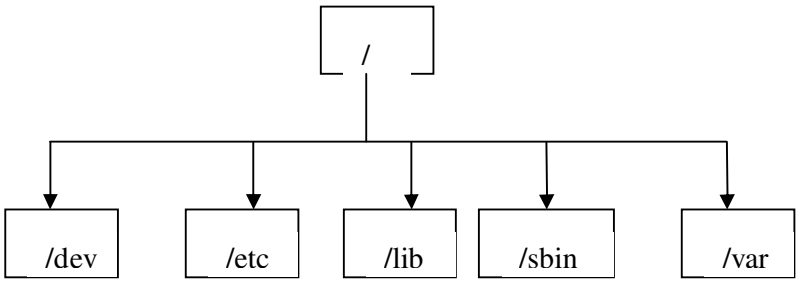



**Figure 3: Sample Linux File Structure**

The /dev directory contains file system entries which represent devices that are attached to the system. These files are essential for the system to function properly.

The /etc directory is reserved for configuration files that are local to your machine. No binaries are to be put in /etc.

The /lib directory should contain only those libraries that are needed to execute the binaries in /bin and /sbin.

The /proc directory contains special files that either extract information or send information to the kernel.

The /sbin directory is for executables used only by the root user.

The /var directory writes log files

### Microsoft Windows File System

Windows makes use of the FAT, NTFS, exFAT and ReFS file systems. Windows uses a *drive letter* abstraction at the user level to distinguish one disk or partition from another. For example, the path `C:\WINDOWS` represents a directory `WINDOWS` on the partition represented by the letter C. Drive C is most commonly used for the primary hard disk partition , on which Windows is usually installed and from which it boots. Generally, Drive A and Drive B are used for two floppy drives. This tradition has become so firmly ingrained that bugs exist in many applications which make assumptions that the drive that the operating system is installed on is C.

### File system commands

The following are some of the file system commands in Linux

- mount (to mount a file system)

syntax: #mount [-lhV]

-h prints a help message;

-V prints a version string; and just

-l lists all mounted file systems (of type type).

Note: use 'man' utility in Linux to know more on mount command

- umount (to unmount a file system)

syntax : #umount [-hV]

-V Print version and exit.

-h Print help message and exit.

Note: use ‘man’ utility in Linux to know more on umount command

- df [–hk] (to know file system disk space)

-h print sizes in human readable format (e.g., 1K 234M 2G)

-k prints in kilobytes

Note: use ‘man’ utility in Linux to know more on df command

- du [-bB] (estimate file space usage)

-b in bytes

-B in Block-size

Note: use ‘man’ utility in Linux to know more on du command

## 2.4 MANAGING USERS

The most important responsibilities of any system administrator is managing and monitoring of users. This section explains on how a new user account is created, deleted and other such related in a Linux operating system. The following explains on different types if users.

**Users**

Generally, users on a system are identified by a username and a userid. The username is something that users would normally refer to, but as far as the operating system is concerned this is referred to using the userid (or uid). The username is typically a user friendly string, such as your name, whereas the userid is a number. The userid numbers should be unique (one number per user).

**Special Users**

There are some default users on all systems when first installed. Other than the root user however these can normally be disabled from logging in. One should be more careful about deleting usernames as sometimes these are used by different tasks running on the system. Special users normally have names such as sys, bin, adm etc.

The root user has an userid of 0, which has a special meaning. The root user has full permissions to do anything on the system. It is not bound by any of the permissions on the system. There are some tasks that can only be performed by root. To use root permissions, one would normally login under a normal userid and “su” to root or use sudo as required.

**Switch User (su)**

One of the features of Linux is the ability to change userid when logged into a system. The command ‘su’ is sometimes referred to as superuser, however this is not completely correct. In the early days of UNIX it was only possible to change to the root user, which made for the superuser command however it is now possible to change to any user using the su command. It is more correct to refer to the command as the switch user command.

The switch user command "su" is used to change between different users on a system, without having to logout. The most common use is to change to the root user, but it can be used to switch to any user depending upon the user’s settings. To switch to a different user other than root, then the username is used as the last option on the command.

With regard to user accounts there are three basic types of Linux user accounts: administrative (root), regular, and service.

The Linux administrative root account is automatically created at the time of installation of Linux, and it has administrative privileges for all services on Linux Operating System. The root account is also known as super user. Regular users have the necessary privileges to perform standard tasks on a Linux computer such as running word processors, databases, and Web browsers. Regular users can store files in their own home directories. Since regular users do not normally have administrative privileges, it is not possible to delete critical operating system configuration files. Services such as Apache, Squid, mail, games, and printing have their own individual service accounts. These accounts exist to allow each of these services to interact with your computer.

Each user on a Red Hat Enterprise Linux system is assigned a unique user identification number, also known as a UID. UIDs below 500 are reserved for system users such as the root user and service users.

**User group**

A user group is a group of one or more users. A user can be a member of more than one group. In Red Hat Enterprise Linux, when a user is added, a private user group (primary group) is created—meaning that a user group of the same name is created and that the new user is the sole user in that group.

The user information is stored in the system files such as /etc/passwd and /etc/shadow files, and that additionally, group membership information is stored in the /etc/group file. The following explains on how user accounts created, modified and deleted with appropriate commands along with examples:

**Understand fields in /etc/passwd**

The password file name in Linux is ‘passwd’ and is stored in /etc directory. The path of a password file in Linux is ‘/etc/passwd’.

The /etc/passwd file stores essential information, which is required during login i.e. user account information. The structure of each entry (record) in a password file is in a seven(7) colon format. The /etc/passwd contains one entry per line for each user (or user account) of the system. All fields are separated by a colon (:) symbol. Total seven fields as follows.

Here is an example of a ***passwd*** file:

**root:x:0:0:root:/root:/bin/bash**

**bin:x:1:1:bin:/bin:/bin/bash**

**daemon:x:2:2:daemon:/sbin:/bin/bash**

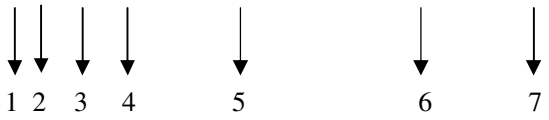
**news:x:9:13:News system:/etc/news:/bin/bash**

**uucp:x:10:14::/var/lib/uucp/taylor_config:/bin/bash**

**cquoi:x:500:100:Cool.....:/home/cquoi:/bin/bash**

Generally, passwd file entry looks like the following:

murli:x:1025:1024:A Murli M Rao:/usr1/murli:/bin/bash




1) Username: It is used when user logs in. It should be between 1 and 32 characters in length. (ex. murli)
2) Password: An x character indicates that encrypted password is stored in /etc/shadow file.
3) User ID (UID)**:** Each user must be assigned a user ID (UID). UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups. (ex. 1025)
4) Group ID (GID): The primary group ID (stored in /etc/group file) (ex. 1024)
5) User ID Info**:** The comment field. It allow you to add extra information about the users such as user's full name, phone number etc. (ex. A Murli M Rao)
6) Home directory: The absolute path to the directory the user will be in when they log in. If this directory does not exists then users directory becomes /(ex. /usr1/murli)
7) Command/shell: The absolute path of a command or shell (/bin/bash). Typically, this is a shell. (ex. /bin/bash)

**Check Your Progress 1**

1. What are the common metadata in packages?

……

……

……

……

……

……

2. Explain the different parts of a passwd file.

……

……

……

……

……

……

**User Administration**

Add New User in Linux

Command: useradd (Enables a super user or root to create a new user or updates default new user information)

Syntax

*Useradd [options] <user-name>*

The following is more in details

*useradd [-c comment] [-d home_dir] [-e expire_date] [-f inactive_time] [-g initial_group] [-G group[,...]] [-m [-k skeleton_dir]] [-p passwd] [-s shell] [-u uid [ -o]] login*

*useradd -D [-g default_group] [-b default_home] [-f default_inactive] [-e default_expire_date] [-s default_shell]*

**Example**

#useradd murli (creates a user murli with default shell, hole directory, UID, GID automatically)

(or) one can use various options as listed above to have specific values

User Account Modification

Command: usermod (Enables a super user or root user to modify a users account)

**Syntax**

*Usermod [options]< user-name>*

The following is more in details

*usermod [-c comment] [-d home_dir [ -m]] [-e expire_date] [-f inactive_time] [-g initial_group] [-G group[,...]] [-l login_name] [-p passwd] [-s shell] [-u uid [ -o]] [-L|-U] login*

#usermod -G others admin murli

Delete User account in Linux

**Command: userdel** ( Enables a super user to remove a users account)

Syntax

*userdel [-r] login*

[ –r Files in the user's home directory will be removed along with the home directory itself and the user's mail spool. Files located in other file systems will have to be searched for and deleted manually]

**Syntax**

**Example**

#userdel –r murli (the user account 'murli' can be deleted)

**Groups Administration in Linux**

The file **/etc/group** contains a list of the users who belong to the different groups. As a matter of fact, whenever a large number of users may have access to the system, they are frequently placed in different groups, each of which has it own access rights to the files and directories.

It has different fields that are separated by "**:**":

groupe_name : special_field : group_number : member1, member2

Here is an example of a ***/etc/group*** file:

**root:x:0:root**

**bin:x:1:root,bin,daemon**

**daemon:x:2:**

**tty:x:5:**

**disk:x:6:**

**lp:x:7:**

**wwwadmin:x:8:**

**kmem:x:9:**

**wheel:x:10:**

**mail:x:12:cyrus**

**news:x:13:news**

**Group Creation**

A system administrator can manage a group's account. The various tasks that a system administrator can perform include adding, modifying and deleting group account.

Command: groupadd (Creates a new group account)

Syntax

*groupadd [-g gid [-o]] group*

| | |
|---|---|
| -g gid | The numerical value of the group's ID. This value must be unique, unless the -o option (override) is used . The value must be non-negative. The default is to use the smallest ID value greater than 99 and greater than every other group. Values between 0 and 99 are typically reserved for system accounts. |

**Example**

#groupadd test1

This will create a test1 group

Group Modification

**Command: groupmod (**Enables a super user or root to modify a group)

Syntax

*groupmod [-g gid [-o]] [-n group_name ] group*

| | |
|---|---|
| -g gid | The numerical value of the group's ID. This value must be unique, unless the -o option (override) is used. The value must be non-negative. Values between 0 and 99 are typically reserved for system groups. Any files which the old group ID is the file group ID must have the file group ID changed manually. |
| -n group_name | The name of the group will be changed from group to group_name. |

**Example**

#groupmod test1 test2  (This will modify group name test1 to test2)

Group Deletion

**Command: groupdel (**The name of the group you wish to delete.)

Syntax

groupdel groupname

Example

#groupdel test2  (this will delete the test2 group)

## 2.5 SYSTEM AND KERNEL MANAGEMENT

An operating system is the first piece of software that the computer executes when you turn the machine on. The operating system loads itself into memory and begins managing the resources available on the computer. It then provides those resources to other applications that the user wants to execute.

**System Management**

Once system is booted, the operating system has to perform various tasks such as task scheduling, memory management, disk management , I/O and security management.

The following explains on typical services that an operating system provides.

**Task scheduler**

The task scheduler is able to allocate the execution of the CPU to a number of different tasks. Some of those tasks are the different applications that the user is running, and some of them are operating system tasks. The task scheduler is the part of the operating system that lets you print a document from your word processor in one window while you are downloading a file in another window and recalculating a spreadsheet in a third window.

**Memory manager**

The memory manager controls the system's RAM and normally creates a larger virtual memory space using a file on the hard disk.

**Disk manager**

The disk manager creates and maintains the directories and files on the disk. When you request a file, the disk manager brings it in from the disk.

**Network manager**

The network manager controls all data moving between the computer and the network.

**Other I/O services manager**

The OS manages the keyboard, mouse, video display, printers, etc.

**Security manager**

The OS maintains the security of the information in the computer's files and controls who can access the computer.

**System Management Commands**

The following are some of the system management commands:

- who (who is logged in)

Syntax: who [-abd]

-a, --all

-b, --boot

time of last system boot

-d, --dead

print dead processes

Example: [murli@imssit ~]$ who

murli pts/0 Jun 2 14:15 (10.10.115.3)

murli pts/1 Jun 2 15:44 (10.10.115.3)

Note: use ‘man’ utility in Linux to know more on mount command

- pwd (displays path of current working directory)

Syntax: pwd

Example: [murli@imssit ~]$ pwd

/home/murli

Note: use ‘man’ utility in Linux to know more on mount command

Other such related command ,which are to be used as part of system management are the following:

- top, vmstat, ps, kill, df, du, mount, unount, tar, cpio, head, tail, mkdir, chdir, chown, chgrp, chmod, nice, find, grep, etc.

Note: use ‘man’ utility in Linux to know more on al he above commands

**Kernel Management**

The kernel is the main component of most computer operating systems. It is a bridge between applications and the actual data processing done at the hardware level. The kernel's responsibilities include managing the system's resources (the communication between hardware and software components). Usually, as a basic component of an operating system, a kernel can provide the lowest-level abstraction layer for the resources (especially processors and I/O devices) that application software must control to perform its function. It typically makes these facilities available to application processes through inter-process communication mechanisms and system calls. Figure 4 shows the kernel and its interactive components.

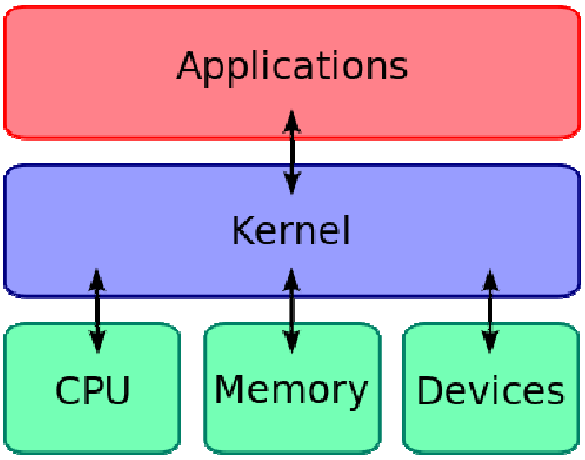



**Figure 4: Kernel and its interactive components**

Operating system tasks are done differently by different kernels( monolithic and micro kernels), depending on their design and implementation. While monolithic kernels execute all the operating system code in the same address space to increase the performance of the system, micro kernels run most of the operating system services in user space as servers, aiming to improve maintainability and modularity of the operating system. A range of possibilities exists between these two extremes.

The kernel's primary function is to manage the computer's hardware and resources and allow other programs to run and use these resources. Typically, the resources consist of:

**The Central Processing Unit**

This is the most central part of a computer system, responsible for running or executing programs. The kernel takes responsibility for deciding at any time which of the many running programs should be allocated to the processor or processors (each of which can usually run only one program at a time)

**The computer's memory**

Memory is used to store both program instructions and data. Typically, both need to be present in memory in order for a program to execute. Often multiple programs will want access to memory, frequently demanding more memory than the computer has available. The kernel is responsible for deciding which memory each process can use, and determining what to do when not enough is available.

**Input/Output (I/O)**

The I/O devices present in the computer, such as keyboard, mouse, disk drives, USB devices, printers, displays, network adapters, etc. The kernel allocates requests from applications to perform I/O to an appropriate device and provides convenient methods for using the device.

**Inter-process communication (IPC)**

Kernels usually provide methods for synchronization and communication between processes called inter-process communication (IPC).A kernel may implement these features itself, or rely on some of the processes it runs to provide the facilities to other processes, although in this case it must provide some means of IPC to allow processes to access the facilities provided by each other. Finally, a kernel must provide running programs with a method to make requests to access these facilities.

**Linux Kernel Management**

The Linux kernel allows drivers and features to be compiled as modules rather than as part of the kernel itself. This means that users can often change features in the kernel or add drivers without recompiling, and that the Linux kernel doesn't have to carry a lot of unnecessary baggage.

The following explains on how to see the already loaded in running kernel, how to add modules to the kernel and finally on how to remove modules from kernel.

**What is Loaded?**

The first utilities that you'll want to get to know are lsmod and modinfo. Open a terminal and run lsmod. Note that you won't need to use sudo or log in as root just to probe the modules in the system.

You'll see output like this when you use lsmod:

```
Module          Size  Used by
parport_pc      18855  0
ppdev            5030  0
lp               7462  0
sco              7209  2
parport         27954  3 parport_pc,ppdev,lp
bridge          39630  0
stp              1440  1 bridge
bnep             9427  2
```

This shows the modules that are loaded, their size, and whether they're being used by other modules. Take the parport module, for instance. It's being used by several other modules, but what are they? The modinfo utility will tell us. To do so run the following:

Run modinfo parport, and you'll see something like this:

```
filename:       /lib/modules/2.6.32-5-amd64/kernel/drivers/parport/parport.ko
license:        GPL
depends:
vermagic:       2.6.32-5-amd64 SMP mod_unload modversions
```

It tells us where the module is found, and its license, and that it has no dependencies.

**Removing Modules**

Modules can be removed using the rmmod utility. The usage is simple, just rmmod *modulename*. However, if we try to remove the parport module, we get this error:

```
ERROR: Module parport is in use by parport_pc,ppdev,lp
```

You can force module removal using rmmod -f, but that's not a good idea, usually. A better way to do it is to use modprobe -r which will automatically look to see what other modules depend on it, and unload those modules as well. If they're in use, then modprobe will refuse to remove them as well, unless you use the -f option with modprobe too.

### Installing Modules

What if you have a module you want to load into the kernel? You can do that with insmod or modprobe.

The preferred method is modprobe, because it will also load any modules that the requested module depends on. For instance, if I didn't have the parport module loaded and went to load the lp or parport_pc modules, modprobe would go ahead and load parport as well.

To load a module using modprobe run modprobe *modulename*.

### Blacklisting Modules

You may on occasion need to "blacklist" a module. Why would you need this feature? Sometimes a module will cause a conflict with another module, is superseded by another module, or is otherwise undesirable.

To blacklist a module, the easiest way to do it (there's usually more than one way to do things...) is to add the module to /etc/modprobe.d/blacklist.conf. For instance, on Debian systems the evbug module is automatically blacklisted because it's not something most users will need. To add a module to the blacklist, just add one line to the blacklist.conf file:

blacklist *modulename*

## 2.6 BASIC TROUBLESHOOTING

Troubleshooting is a form of problem solving, often applied to repair failed products or processes. It is a logical, systematic search for the source of a problem so that it can be solved, and so the product or process can be made operational again. Troubleshooting is needed to develop and maintain complex systems where the symptoms of a problem can have many possible causes.

The following are some of the commands being used in a Linux environment for problem diagnosis and troubleshooting:

- Use tail –f to watch log file in real time, advantage is simple you can spot error or warning message in real time.

Command : **# tail –f /var/log/maillog**

- Use telnet command to see if you get response or not. Sometime you will also see some informative message:

**telnet ip port**

Example(s):

**# telnet localhost 53**
**# telnet localhost 25**

- Make sure you can see PID of your service.

**pidof service-name**

**cat /var/run/service.pid**

Example(s):

**# pidof sshd**
**# cat /var/run/sshd.pid**

- You need to make sure that your DNS server or third party DNS server (ISP) is accessible. This is an important step, as many network services depend upon DNS; especially sendmail/postfix or Squid etc for example. Run dig or nslookup. No timeout should occur.

  **# dig your-domain.com**
  **# nslookup gw.isp.com**
  **# more /etc/resolv.conf**

- For networking troubleshooting, make sure your ip address configuration is right, gateway, routing, hostname etc all configured. Here is list of tools on RedHat Linux to verify or modify information:

Hostname verification or setup tools

- **hostname** : To get hostname of server.
- **more /etc/sysconfig/network** : To see hostname and networking details
- **more /etc/hosts** :Make sure at least localhost entry do exist.

Ethernet configuration tools

- **ifconfig** : To see running network card information.
- **ifconfig eth0 up|down** : To enable|disable network interface
- **service network reload|restart|stop|start** : To reload (after changed made in ip config file)|restart|stop|start network interface with all properties.
- **route|netstat –rn** : To print routing table
- **ping ip-address** : To see if host is alive or dead
- **more /etc/modules.conf** : To see your network card configuration alias for eth0 exists or not.
- **lsmod** : To list loaded modules (read as drivers), here you need to see that eth0 module is loaded or not, if not loaded then use insmod to insert (load) driver.
- **dhclient** : Dynamic Host Configuration Protocol Client, run this if your Ethernet card is not getting ip from DHCP box on startup; this command does by default shows useful information.

**To see if service blocked because of access control**

- **iptables –n –L** : To list all iptable rules; useful to see if firewall blocks service or not.
- **service iptables stop|start** : To start|stop iptables
- **more /etc/xinetd.conf**

OR

- **more /etc/xinetd.conf/SERVICENAME** = To list configuration of xinetd server. Again useful to see if firewall xinetd based security blocks service or not (xinetd includes host-based and time-based access control)
- **more /etc/hosts.allow** : To see list of hosts allowed to access service.
- **more /etc/hosts.deny** : To see list of hosts NOT allowed to access service. **NOTE** first TCP wrappers (hosts.allow|hosts.deny) checked and then xinetd-based access control checked.
- **more /etc/path/to/application.conf** : See your application configuration file for access control. For example smb.conf and many other applications/services got own access control list in application. You need to check that as well.

**Some more commands**

Getting ram information

#cat /proc/meminfo

or if you want to get just the amount of ram you can do:

#cat /proc/meminfo | head -n 1

**Getting cpu info**

Sometimes in troubleshooting we want to know what processor we are dealing with along with how much cpu is currently being used by our OS and programs. We can do this with these two commands.

#cat /proc/cpuinfo

#top

Example :[murli@imssit ~]$ top

Output

top - 14:25:45 up 10 days, 21:07, 1 user, load average: 0.00, 0.05, 0.02

Mem: 4147692k total, 4124540k used, 23152k free, 57604k buffers

Swap: 10241396k total, 562856k used, 9678540k free, 2250180k cached

| PID | USER | PR | NI | VIRT | RES | SHR | S | %CPU | %MEM | TIME+ | COMMAND |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5511 | oracle | 16 | 0 | 1280m | 264m | 262m | S | 0.3 | 6.5 | 4:08.74 | oracle |
| 5665 | oracle | 16 | 0 | 1280m | 314m | 311m | S | 0.3 | 7.8 | 12:13.93 | oracle |
| 17154 | murli | 16 | 0 | 2776 | 1048 | 772 | R | 0.3 | 0.0 | 0:00.16 | top |
| 1 | root | 16 | 0 | 2900 | 540 | 464 | S | 0.0 | 0.0 | 0:01.38 | init |
| 2 | root | RT | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.31 | migration/0 |
| 3 | root | 34 | 19 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | ksoftirqd/0 |
| 4 | root | RT | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.20 | migration/1 |
| 5 | root | 34 | 19 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | ksoftirqd/1 |
| 6 | root | RT | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.22 | migration/2 |
| 7 | root | 34 | 19 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | ksoftirqd/2 |
| 8 | root | RT | 0 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.12 | migration/3 |
| 9 | root | 34 | 19 | 0 | 0 | 0 | S | 0.0 | 0.0 | 0:00.00 | ksoftirqd/3 |

```
 10 root      5 -10     0    0    0 S  0.0  0.0   0:00.00 events/0
 11 root      5 -10     0    0    0 S  0.0  0.0   0:00.00 events/1
 12 root      5 -10     0    0    0 S  0.0  0.0   0:00.00 events/2
 13 root      5 -10     0    0    0 S  0.0  0.0   0:00.00 events/3
 14 root      5 -10     0    0    0 S  0.0  0.0   0:00.00 khelper
 15 root      5 -10     0    0    0 S  0.0  0.0   0:00.00 kthread
```

Check out how much hard drive space is left

#df -h

Example :[murli@imssit ~]$ df –h

Output

```
Filesystem            Size  Used Avail Use% Mounted on
/dev/cciss/c0d0p5      16G  2.3G   13G  16% /
/dev/cciss/c0d0p1     289M   28M  247M  10% /boot
none                  2.0G     0  2.0G   0% /dev/shm
/dev/cciss/c0d0p7     7.7G   51M  7.3G   1% /tmp
/dev/cciss/c0d0p2     202G  182G   11G  95% /u01
/dev/cciss/c0d0p3      29G   21G  6.6G  76% /u02
/dev/cciss/c0d0p8     4.9G  2.8G  1.9G  60% /usr
```

Check out how much disk partitions usage

#df –k

Example: [murli@imssit ~]$ df -k

Output

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/cciss/c0d0p5     16126420   2307900  12999208  16% /
/dev/cciss/c0d0p1       295561     27863    252438  10% /boot
none                   2073844         0   2073844   0% /dev/shm
/dev/cciss/c0d0p7      8064272     51480   7603140   1% /tmp
/dev/cciss/c0d0p2    211663320 190318352  10593068  95% /u01
/dev/cciss/c0d0p3     30233928  21790480   6907636  76% /u02
/dev/cciss/c0d0p8      5036284   2854352   1926100  60% /usr
```

To estimate file space usame

#du

Example: [murli@imssit ~]$ du

Output

```
16      ./.kde/Autostart
24      ./.kde
16      ./.xemacs
104     .
```

Kill a process

#ps -A | grep ProgramName

#kill 7207

**Show all network connections**

#netstat

Example: [murli@imssit ~]$ netstat

Output

Active Internet connections (w/o servers)

| Proto | Recv-Q | Send-Q | Local Address | Foreign Address | State |
|---|---|---|---|---|---|
| tcp | 0 | 0 | imssit:64734 | imssit:1521 | ESTABLISHED |
| tcp | 0 | 0 | imssit:64742 | imssit:1521 | ESTABLISHED |
| tcp | 0 | 0 | imssit:1521 | imssit:32786 | ESTABLISHED |
| tcp | 0 | 0 | imssit:1521 | imssit:40238 | ESTABLISHED |

List all files that are currently open on the system

#ls –of

Example: [murli@imssit ~]$ ls –of

Output

.. . .bash_logout .bashrc .emacs .zshrc
.bash_history .kde .gtkrc .xemacs .bash_profile

List all processes running

#ps-aef

Example: [murli@imssit ~]$ ps –aef

Output

| UID | PID | PPID | C | STIME | TTY | TIME | CMD |
|---|---|---|---|---|---|---|---|
| root | 1 | 0 | 0 | May22 | ? | 00:00:01 | init [5] |
| root | 2 | 1 | 0 | May22 | ? | 00:00:00 | [migration/0] |
| root | 3 | 1 | 0 | May22 | ? | 00:00:00 | [ksoftirqd/0] |
| root | 4 | 1 | 0 | May22 | ? | 00:00:00 | [migration/1] |
| root | 5 | 1 | 0 | May22 | ? | 00:00:00 | [ksoftirqd/1] |
| root | 6 | 1 | 0 | May22 | ? | 00:00:00 | [migration/2] |
| root | 7 | 1 | 0 | May22 | ? | 00:00:00 | [ksoftirqd/2] |
| root | 8 | 1 | 0 | May22 | ? | 00:00:00 | [migration/3] |
| root | 9 | 1 | 0 | May22 | ? | 00:00:00 | [ksoftirqd/3] |
| root | 10 | 1 | 0 | May22 | ? | 00:00:00 | [events/0] |
| root | 11 | 1 | 0 | May22 | ? | 00:00:00 | [events/1] |
| root | 12 | 1 | 0 | May22 | ? | 00:00:00 | [events/2] |
| root | 13 | 1 | 0 | May22 | ? | 00:00:00 | [events/3] |
| root | 14 | 1 | 0 | May22 | ? | 00:00:00 | [khelper] |

**Check Your Progress 2**

1. Write the syntax and uses of "useradd" command in Linux.

   ……..

   ……..

   ……..

   ……..

   ……..

2. Which command is used to display real-time running tasks in a Linux environment? Explain the significance of this command using an example.

   ……..

   ……..

   ……..

   ……..

   ……..

3. How the “Disc Uses” is checked in Linux? Explain with an example.

   ……..

   ……..

   ……..

   ……..

   ……..

## 2.7 SUMMARY

In this unit, the need of a package manager for installation and un-install a package is clearly explained. Different types of file systems and their need and usage was discussed. The unit also covered on how to manage user accounts and also on system and kernel management. Finally the unit explained with some examples on troubleshooting commands.

## 2.8 ANSWERS TO CHECK YOUR PROGRESS

**Check Your Progress 1**

1. Packages contains metadata that contains
   - Summary (software name, etc)
   - Description
   - A list of files contained in the package
   - The version of the software it contains as well as the release number of the package
   - When, where and by whom it has been built

- What architecture it has been built for
- Checksums of the files contained in the package
- The license of the software it contains
- Which other packages it requires to work properly

2. Generally, passwd file entry looks like the following:

murli:x:1025:1024:A Murli M Rao:/usr1/murli:/bin/bash


1. Username: It is used when user logs in. It should be between 1 and 32 characters in length. (ex. murli)
2. Password: An x character indicates that encrypted password is stored in /etc/shadow file.
3. User ID (UID)**:** Each user must be assigned a user ID (UID). UID 0 (zero) is reserved for root and UIDs 1-99 are reserved for other predefined accounts. Further UID 100-999 are reserved by system for administrative and system accounts/groups. (ex. 1025)
4. Group ID (GID): The primary group ID (stored in /etc/group file) (ex. 1024)
5. User ID Info**:** The comment field. It allow you to add extra information about the users such as user's full name, phone number etc. (ex. A Murli M Rao)
6. Home directory: The absolute path to the directory the user will be in when they log in. If this directory does not exists then users directory becomes /(ex. /usr1/murli)
7. Command/shell: The absolute path of a command or shell (/bin/bash). Typically, this is a shell. (ex. /bin/bash)

**Check Your Progress 2**

1. 'useradd' is a command to create a new user in Linux environment

Syntax is

Useradd [options] <user-name>

or

useradd [-c comment] [-d home_dir] [-e expire_date] [-f inactive_time] [-g initial_group] [-G group[,...]] [-m [-k skeleton_dir]] [-p passwd] [-s shell] [-u uid [ -o]] login

Example:# useradd murli

2. 'top' is a command to display real-time running tasks in a Linux environment

Example :[murli@imssit ~]$ top

**Output**

top - 14:25:45 up 10 days, 21:07, 1 user, load average: 0.00, 0.05, 0.02

Mem: 4147692k total, 4124540k used, 23152k free, 57604k buffers

Swap: 10241396k total, 562856k used, 9678540k free, 2250180k cached

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5511 oracle    16   0 1280m 264m 262m S  0.3  6.5   4:08.74 oracle
 5665 oracle    16   0 1280m 314m 311m S  0.3  7.8  12:13.93 oracle
17154 murli     16   0  2776 1048  772 R  0.3  0.0   0:00.16 top
    1 root      16   0  2900  540  464 S  0.0  0.0   0:01.38 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.31 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      RT   0     0    0    0 S  0.0  0.0   0:00.20 migration/1
    5 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/1
```

3. ‘df-k’ is a command to know the disk usage

Example: [murli@imssit ~]$ df -k

**Output**

```
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/cciss/c0d0p5     16126420   2307900  12999208  16% /
/dev/cciss/c0d0p1       295561     27863    252438  10% /boot
none                   2073844         0   2073844   0% /dev/shm
/dev/cciss/c0d0p7      8064272     51480   7603140   1% /tmp
/dev/cciss/c0d0p2    211663320 190318352  10593068  95% /u01
/dev/cciss/c0d0p3     30233928  21790480   6907636  76% /u02
/dev/cciss/c0d0p8      5036284
```

## 2.9 FURTHER READINGS

1. Computer Networks by Andrew S Tanenbaum , Fifth Edition
2. SA2, Redhat System Administration I & II, Student Workbook
3. Cisco Certified Network Associate Study Guide, Seventh Edition by Todd Lammle
4. Redhat Enterprise Linux System Administration
5. http://en.wikipedia.org/wiki/linux
6. http://en.wikipedia.org/wiki/kernel