unit 1

Introduction to Operating System

Software is a collection of programs used to perform various tasks of the System *(System and Computer interchangeably used)*. Basically Software is classified into two types. They are System Software and Application Software. Application Software's are developed for user applications and data handling.

System Software is a program that acts as interface between user, Application Software and Hardware. The Figure 1.1 shows the relationship among software and hardware.

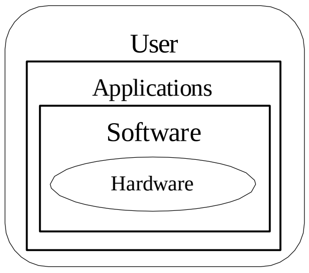


Figure 1.1 **Relationships among Hardware and Software**

Systems software is a program which is devoted to supervision the computer, such as the operating system, file management utilities, and disk management operations.

1.1 What is Operating System?

An Operating System (OS) is a one of the System Software which manages various recourses of System such as Hardware and Software. Figure 1.2 shows the functions of OS. An Operating System performs the following tasks:

- Memory Management
- Processor Management
- File Management
- Storage Management
- Security
- Scheduling

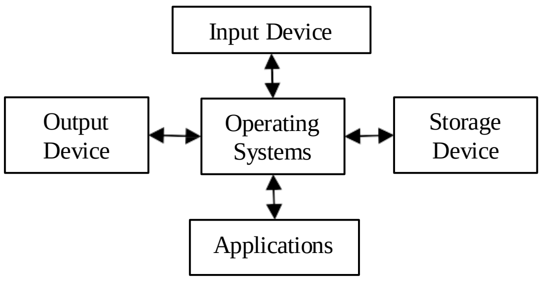


Figure 1.2 **Functions of Operating System**

Some of widely used OS are as follows:

1. *Batch Operating System*

In Batch Operating System environment non-interactive jobs are executed in a series at one time. In Batch processing users prepare jobs using off-line devices such as punched cards and submit to computer operator. This system has drawbacks such as lack of communication between user and system, CPU time may be idle, etc.

2. *Time Sharing Operating System*

In time sharing environment multitasking can be achieved because of ability of the Processor to handle multiple jobs at a time. Multiple jobs are carried out by the CPU by switching between the jobs, which make user to receive an immediate response.

3. *Distributed Operating System*

In distributed systems, it uses multiple processors for handling numerous real-time applications and many users at a time. Data processing jobs are shared among the processors accordingly.

4. *Network Operating System*

Network OS is used in the client / server environment in which request and response happening over the remote systems. Few advantages of Network OS are a Centralized server which makes storing and accessing data in convenient manner, security in server is more than normal system, upgradation to new technologies and hardware can be easily included into the system and etc.

5. *Real Time Operating System*

A Real Time Operating System is software that quickly switches between jobs, giving the feeling that multiple programs are being performed at the same time on a single processing core.

1.2 Most Popular OS

Most popularly used OS are *Windows, Linux, Mac OS* and etc. **Windows** OS is owned by Microsoft. Various types of Windows are available for various platforms. Linux is open source software with variant versions. **Linux** is mostly used in server, Internet, IoT and etc. It is a OS which is much convenient for networking environment. Difference between Linux and Windows are:

- Windows OS is licensed commercial version whereas Linux is open source operating system.
- Windows cannot be modified based on users need whereas Linux has access to source code and code can be altered as per user need.
- Windows are slow on older hardware whereas Linux will run faster than windows.
- Windows gather the user information’s which has privacy concern. In Linux distributions no user data is collected.
- Regarding killing(closing) application if they hung is much hard in Windows when compared Linux.
- Wide variety free software’s are supported by Linux when compared with Windows.
- Security is much more in Linux when compared with Windows.

1.3 Functions of an Operating System

Command Interpretation

CPU can do the tasks such as copy a file, delete a file, rename a file, etc., which are given by the user. The commands given by users have to be interpreted to binary code which is made up of zeros and ones. CPU can use the code and make the operations. Translation of user's commands into binary code is done by OS which helps CPU to interpret the command.

Process Management

A process is defined as a program in execution. Random access memory (RAM) is main memory where program resides for execution. When developer executes a program, the OS gets the program to the memory, CPU starts the execution of instruction one by one at a time. In a multiprogramming atmosphere, a lot of processes run on the computer simultaneously. To enable more than a few processes to be executed at the same time, the OS splits the CPU time. The OS assigns time slots to the different processes in the memory which makes process to be executed concurrently.

Memory Management

The OS assigns the available memory to the processes. Because main memory is limited, it is not possible to get all the processes loaded into the memory at a given time. Memory executes the current running process, where as remaining process are stored in HDD (Hard Disk Drive) that are not executed at that time. HDD is used as secondary storage device. Processes are moved to main memory when it is needed to be executed. OS performs the operation of swapping processes between HDD and Main memory

Input / Output (I/O) Operations and peripheral Management

When applications are executed that are available in main memory, may need to accept input and generate output of the operation. On its own applications cannot do the I/O operations. Here comes the role of OS which performs the operation of handling the I/O devices. Operating System has the controlling mechanism over the peripherals devices that are connected to System. Communication between the peripherals and CPU is managed by OS.

File Management

File management is another main operation of an OS. The OS allows user to do file operations, such as creating, naming, opening, reading, and closing a file.

Types of Systems

1. ***Single User Systems***

Single user system is environment in which system can be operated by single user at a time. An example of a single user system is a Personal Computer (PC). It is a general purpose computer that can run programs to carry out a sample tasks.

Single User Operating Systems

MS DOS (Disk Operating System) is an example of a single user operating system.

2. ***Multiuser Systems***

In multiuser system, at a time many users can work concurrently. Multiuser systems can execute multiple processes concurrently and share hardware resources, such as printers and disks.

Fundamental ideas of a Multiuser System

The CPU of multiuser system is more useful than that of single user system. Two operations namely, *multiprogramming* and *multitasking* are supported in multiuser systems. A multiuser system made of a system with several terminals attached to it. The terminal can be of two types, *smart* or *dumb*.

A dumb terminal is one which has a monitor and a keyboard and no hard disk or CPU will be available with the system. The dumb terminal is interface between a user and multiuser system.

A smart terminal is made up with a CPU and peripherals device. It can work independently in the multiuser system environment. A smart terminal can have any OS which is loaded into its hard disk. A smart terminal can connect to the server when required.

Multiuser Operating System

Linux and Windows NT terminal server are examples of multiuser operating systems. In such system environment, more than one can connect to the system and work alongside at any given time.

1.4 Introduction to Linux Operating Systems

UNIX operating system is base for Linux, Mac OS X. At AT&T's Bell Laboratory in late 1960s and early 1970s research team developed UNIX operating system that would be available for multiple users with more security.

In 1980s and 1990s, Corporations started licensing UNIX. During late 1980s, there was group of people interested in developing a free operating system that would be alike to UNIX. Linus Torvalds launched free, *open-source* Linux kernel software in 1991. Open source software is one which has rights to modified and redistributed with is fully visible code.

Linux is the *kernel*, it is not a complete operating system. Kernel is an interface between hardware and the input/output requests. Remaining part of the operating system includes many GNU libraries, utilities and other software. This operating system in whole called as GNU/Linux. The GNU/Linux OS is usually referred to as the Linux OS.

There are quite a few distributors of Linux. All the distributors use the Linux kernel. Some of the distributors are:

Distributors Name	Web Site
Red Hat	http://www.redhat.com
Caldera	http://www.caldera.com
Mandrake	http://www.linux-mandrake.com
Debian	http://www.debian.com
SuSE	http://www.suse.com
Slackware	http://www.slackware.com

History of Linux OS

Linux is an operating system is available for personal computers from 1991. At beginning, Linux can be operated only in Intel 8086 processor. Over the years many versions of Linux were distributed with support of running all the processors. At present, Linux is one of OS that running on a wide range of processors such as Intel, AMD, Motorola, SPARC and IBM. Linux is alike to UNIX which has many concepts from UNIX and equipped with UNIX API.

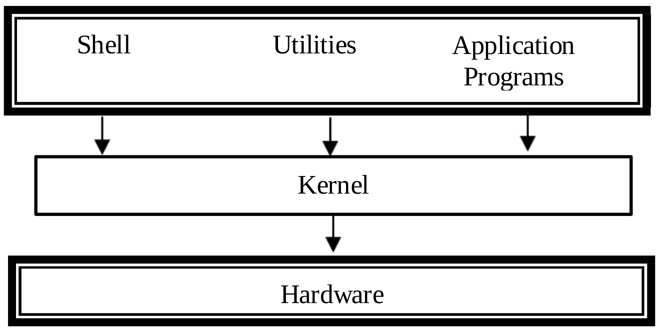
Linux is without a doubt the mostly growing operating system. It is used in various areas right form embedded systems to mainframe. One of the exciting and most significant specifications about Linux is that it is open-sourced. The Linux kernel is licensed under the GNU General Public License (GPL); the kernel source code is freely available and can be modified to suit the needs of user machine.

Users can make suggestion for alterations to the kernel code. When a new version of Linux is introduced, user can work on the new version to fix bugs, if any. To keep constancy, Linus Torvalds ensures quality control and then combines all the new code into the kernel. This is a main cause for the accomplishment of Linux. Linux has an official mascot – the Linux penguin – Tux. Linux has free software, such as text editors, browsers, and programming tools.

The Linux Penguin – Tux

Architecture of Linux

The GNU/Linux OS made-up of a kernel, a shell, utilities, and application programs. The Linux architecture is shown below:



Kernel

The OS program *kernel* is core element of Linux system. The kernel manages the resource of a system, allocating them to various users and tasks. It communicates directly with hardware device, which makes way for easy program creation and moving them across different hardware platform. Users do not interact with a kernel directly. Actually, the logon process starts a new interactive program called the *shell* for each user.

Shell

Linux has a simple interface called the shell. The shell provides services for a user. A shell can be used by user to interact with the system. The users don't want to have an idea about how interaction happens with hardware. Some of the common shells in Linux are *bash, sh, tcsh, csh,* and *ksh.*

Utilities and Application programs of Linux

Various utilities and commands collections are available in Linux which can be used service processing. Utilities and commands can be executed with shell. Software from independent venders such as Application programs, Database Management, Word Processors, etc can be installed and used in Linux.

Difference between UNIX and Linux

With UNIX as a reference model new OS Linux was developed. Fundamental structure and concepts of Linux and UNIX is identical to each other. Linux is can be treated as a new version of UNIX. The main difference between Linux and UNIX is that Linux is open source which provides Linux as free of cost. Linux has some many development tools, such as C, C++, and scripting languages such as Perl, Python and etc.

Features	Linux	UNIX
Shells	Bash, pdksh, tcsh, zsh, ash	Bourne, Korn, C
Vendors	Red Hat, Caldera, Debian, LinuxPPC, SUSE	AT & T, MULTITCS, BSD, SCO, HF, Ux, IRIX, Ultrix, XENIX, Sun Solaris
Licensing	Free of Cost	Licensed

Comparison of Linux and UNIX

Linux Features and Utilities

Multiprogramming

Simultaneously programs can be executed in Linux by different users. This concept is called multiprogramming.

Time Sharing

Time-sharing is one of the main idea behind the Multiprogramming concept. In this environment OS has to deal with various programs concurrently. Programs are queued in the memory and CPU time is shared with programs. Every program will be given a time slot by CPU for a specific period and again it is placed in the queue till it gets next time slot.

Multitasking

A module or program can be divided into tasks such as reading, writing, input etc. It is role of OS to handle the execution of multiple tasks. This is called as multitasking. Scheduling process of task are carried out by kernel.

Virtual Memory

The amount of physical memory not always be sufficient to execute large applications or enable multiple applications to be active is at a given time. In such situations, a portion that is a logical portion of the hard disk can be utilized as artificial or virtual memory. The OS places the programs and data, which are not frequently used, in this portion of the hard disk and loads them in the memory, whenever required. Therefore, the OS uses the resources of the computer optimally.

Samba

Server Message Block protocol or *SMB* is used for sharing the files and printers. MSB is the protocol used by Microsoft OS to share files and printers. Samba is a suite programs that implement the SMB protocol on Linux. Using Samba user can share a Linux file system with Windows 95, Windows 98, or Windows NT. Similarly, user can use SMB to share printers connected to Windows OS Computer with Linux.

Cron Scheduler

Cron is a scheduler program is used by Linux. It is used to execute the commands, scripts, or programs at scheduled times.

Licensing

Linux is licensed under the *GHU G*eneral *P*ublic *L*icense *(GHU GPL)*. The licensing for Red Hat Linux states that a person can make any number of copies of software and distribute it freely, or charge a price for it. User can also download Linux free of cost from the internet.

Web Server

A Web server is a process that runs on the OS and enables users to access resource that have been published in the form of Web page. A Web server can enable users to access resources on a wide area network (WAN) or the Internet. Linux OS bundles with Apache Web server.

Shells available in Linux

1. *Bourne Shell*

The Bourne shell was developed at AT & T by Stephen Bourne. This shell is circulated with all UNIX systems and is resides in the */bin* directory. The executable file name for Bourne shell is *sh*.

2. C Shell

The C shell was developed by William Joy at the University of California at Berkeley. It is similar to the C programming language. The C shell derives its name from its programming language, which resembles the C programming language in syntax. The executable file name for the C shell is *csh*.

3. Korn Shell

In the AT & T Laboratory Korn shell was developed by David Korn. The Korn shell combines the features of both the Bourne and C shells. The executable file name for the Korn shell is *ksh*.

4. Restricted Shell

Restricted shell can be used for limited access on the OS to an usr. The restricted shell typically used for guest users who only need limited rights and permissions.

5. Bash Shell

The Bash shell is an enhancement on the Bourne shell, hence the name Bash, which is an acronym for Bourne Again Shell. Bash is product of Free Software Foundation’s GNU project. The Bash shell is the default shell for most Linux systems and is stored in the */bin* directory. The Bash shell stores all of the commands that user use in a session. In addition, it stores the commands that user used in previous session. In Red Hat Linux, the *sh* command is a symbolic link to *bash*.

6. Tcsh Shell

Tcsh stands for Tom’s C Shell and is an enhancement of the C shell. The Tcsh shell is also known as TC shell. In Linux, the *csh* command is a symbolic link to the Tcsh shell. User can execute the Tcsh shell by typing either *csh* or *tcsh* at the command prompt.

7. A Shell

The A shell was developed by Kenneth Almquist of the University of Berkeley. It emulates the Bourne shell. The A shell is suitable for computers that have limited memory. The executable file name for the A shell is *ash*.

8. Z Shell

The Z shell offers the features of Tcsh and Korn shells. In addition, it provides a large number of utilities and extensive documentation. The executable file name for Z shell is *zsh*.

Beginning a Linux: Logging on

Telnet program can be used by User to connect to a system which is in Windows 95/98/NT OS's to the system running the Linux OS. User can start *telnet* program by typing the following command at prompt.

```
telnet <hostname or IP address>
```

For example, if user is working on a system running in Windows 95 OS and the TCP/IP address to the Linux server is 192.198.0.1, user would need to give the following command to connect to the Linux Server.

```
telnet 192.168.0.1
```

When a connection is established between the Linux and Windows, a message similar to the one shown below would appear on the screen:

```
Red Hat Linux release 7.1 (Seawolf)
Kernel 2.4.2.2 on an 1686
Login:
```

Each user has an identification called the user name or the logon name, which the user needs to enter when the login: prompt appears on the screen

The steps to log on to a Linux server are:

At the *login:* prompt, enter your logon name and password.

Information about users, such as user names and passwords, is stored in the shadow and passwd files in the /*etc* directory. Users enter their logon name and password at command prompt. Given information's are verified with *shadow* and *passwd* files.

If the *login*: information provided by the user does not match the information in the *shadow* and *passwd* files, an error message: "Login incorrect" is displayed. Only authorized users can make logon process which makes system more secure. When a valid user name is entered, the shell prompt is displayed on the screen. The shell prompt is given below:

```
[user name@localhost current_directory_name]$
```

At the shell prompt, user name is the logon name of the user, and current_directory_name is the current working directory of the user.

When user comes, a new logon account has to be created. Administrator creates a new login and assigns a home directory to the user. The home directory is the default directory for a user when the user logs on. In Linux, logon names are usually the names of users, and their home directory usually has the same name. For instance, if user name is tom and home directory name is also tom, then after logging on. The entire logon process appears just as the one shown below:

```
login: tom
Password:
Last login: Sat Sep 18 12:18:02 [tom 172.17.55.167 tom]$ _
```

Security for Users: Passwords

In Linux, the logon process makes sure that only approved users can access the system that runs on Linux. As an added advantage a security measure in Linux allows user to specify a password associated with user logon name. To log on, user need to enter both the user name and the password. The combination of the two is checked by Linux to verify if the user is an authorized one.

Changing the User Password

To ensure the reliability of data, users should change passwords at frequent intervals. A user can change a password by using the *passwd* command. Below example shows how users can change their password.

Example

```
$ passwd
Changing password for Steve
(Current) UNIX password:
New UNIX password:
Retype New UNIX password:
```

passwd: all authentication tokens updated successfully

The *passwd* command prompts the user to enter the new password two times. If the new passwords do not match then password can be cannot changed. The message *Sorry, passwords do not match* will be displayed.

```
[Steve@localhost Steve]$ passwd
Changing password for Steve
(current)UNIX password:
New UNIX password:
Retype new UNIX password:
Sorry, passwords do not match
New UNIX password:
```

The *root* User

A special user account called *root* which has permissions to control, modify, and configure various resources on the system. Another name for of *root* user is *super* user. The root user is the ***administrator*** of Linux and has all the rights required to control the working of the system. The root user is accountable for system maintenance. Various roles of root user are:

- Creating and assigning user accounts and passwords to new user.
- Allocating storage space to users.
- Assigns access permissions.
- Preventing access of unauthorized user.

The prompt of the root user is denoted by a # sign. The *root* user can change the password of any user in Linux. The root user can execute the *passwd* command followed by a user name to change the password for that user.

Example

```
# passwd Steve
Changing password for Steve
(Current) UNIX password:
New UNIX password:
Retype New UNIX password:
passwd: all authentication tokens updated successfully
[root@localhost root]#
```

Referring to the Linux Help Manual

Online help is an integral part of every software. In Linux, online help pages are referred to as the manual pages. Manual pages provide help to a user about the various commands and the options available for the commands. The manual pages are part of the

Lima Documentation Project (LDP). The syntax to access help using man pages is as follows:

Syntax

```
	man <command name>
```

For example, the command to obtain information about the various options of the passwd command is:

```
	$ man passwd;
```

Ending a Linux Session: Logging Off

After you log on to Linux, the work session continues until you instruct the shell to terminate the session. Typing exit or logout at the command prompt end's the current Linux session. User can also press <CTRL>+d to end the Linux session. In order to keep the files secure, user must end thier Linux session.

Exercises

Fill in the Blanks

1. Linux was developed by ______________
2. UNIX is licensed, Linux is ________________
3. The executable file name of C shell _________
4. User password can be changed with _________command
5. _________is program used to connect Linux from Windows environment.
6. __________is command which provides information related to Linux commands.
7. Shortcut key to close a Linux session is ____________

Answer in details

1. Write detail on introduction to Operating Systems.
2. Explain the evolution of Linux.
3. Discuss the functions of Operating Systems.
4. Describe the developments and features of Linux OS.
5. Explain the process of logon and logout in Linux with example.
6. What is role of root user? Explain.
7. How you will change password in Linux? Give Example.

*****

UNIT 2
Files and Directories Management

2.1 File System

The Linux File System

In a secondary storage data are stored in the structure of files. OS has to provide a method to handle the thousand of files in organized manner. A file system can defined as a method of storing and managing files and directories in the storage device.

File system of Linux is little different from other Operating Systems. It follows the hierarchical file structure which makes the users to store and access the files in the directories.

Directories are much alike a drawers of a cabinet. Drawer used to contain files, documents. As we do in drawer files and directories can be created on a disk to store files. In a cabinet, a user specifies names for the drawer labels and contents. Similarly, in a Linux file system, a user can give a directory name where collection of files to be placed in the directory.

Files are placed in the relevant directories based on its data. Linux OS admin is responsible for creating home directory for users. For example, Arun is home directFoirley for the user named Arun. In this directory he can store all his files or can create directories in his home directory. The directory Arun haveng two files, namely prog1 and prog2, two directories named as Desktop and Templates. In the Desktop directory it has two files, Default and start. In Linux, root (/) directory is main directory of entire file system. It is divided into bin, boot, home, usr, etc. Each of these directories is organized to store specific types of files.

- The */bin* directory is used to store Linux utilities. These utilities are the commands in the Linux. It is named bin because, utilities of Linux are stored in the binary format.
- Device related files stored in */dev* directory.
- The */etc* directory stores the data related to the operating system, including the OS programs and configuration files. Commands such as passed file are placed in this directory which can be used by users.
- Data 1ibraries for the compilers are placed in */lib* directory.
- The */home* directory contains the home directories of all the users.
- The */usr* directory stores the operating system files that are not required during the startup process.
- The */var* directory contains information specific to different utilities available in Linux.

In a Hierarchy file system, files are retrieved using its path which has the details such as file name, preceded by directory name which contains the file. The file name and different directory names in the path are separated by '/' symbol.

In a hierarchical structure, the OS provides faster access to files because files are placed as groups. To find a file, the OS needs to search current directory only. When files are stored in other than *root* directory, then OS needs to search the entire list of files to locate a specific file.

File Naming rules in Linux

In Linux file names should be made with following rules

- Can be up to 256 characters long.
- Can contain special characters, except for '/ '
- Can contain both uppercase and lowercase alphabets.
- File names are case-sensitive.
- Should not contain a blank or a tab.

Referring to Your Home Directory

When a user is created on Linux is assigned with directory to store files. Default directory will be home directory of the user when user logged on.

When user working in Linux, user may be access the home directory frequently. Tilde (~) symbol can be used to specify the path of home directory instead using complete path every time.

Types of Files in Linux

In Linux, information is stored in files. In Linux, a device is also a file. Therefore, all the information sent to monitor is considered as file is sent. In Linux, there are three types of files:

- Ordinary
- Directory
- Special

Ordinary Files

Users created files are classified as ordinary files. These files contain all data, program, object and executable files. A user can modify the files.

Directory Files

In Linux a new user is created, it automatically creates a directory with user name as name for storing the files created by the users. It holds the details about the files stored in it. For example, a directory called Suresh is created in the directory /home directory which as structure of /home/Suresh. It stores all the files and directories of Suresh along with details of the user Suresh. A directory structure cannot altered by user, but a user can add file or subdirectory to the home directory.

Special Files

There is file type called Special files in Linux. These files mostly associated with input/output devices where stored in standard Linux directories such as /dev and /etc., it cannot be altered by user. Various special files supported by Linux are:

- Character device files
- Block device files
- Hard links
- Symbolic links

Character Device Files

To a read or write a character at time through a device, a character device files is used. One of the examples for character device file is modem. It is also known as sequentially accessed devices.

Block Device Files

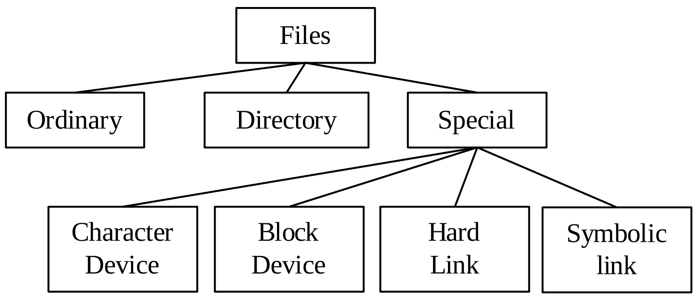
A block of data can be accessed as one block using Block device files. A block data is made up of either 512 or 1,024 bytes. Kernel reads or writes data in blocks. Minimum one block is handled at a time. Data are collected and stored in the memory for user needed. This process is carried by the kernel. It allows the random access method, which makes I/O in faster and efficient manner. HDD is device is primary example for Block device operations. I/O devices may be either character device or block device depending on commands used by the device.

Hard Links

Hard links are special files that allow a single file to have multiple names. User can specify a hard link only for a file and not for a directory. User can specify hard links for files only when they are on the same file system.

Symbolic Links

Symbolic links are also known as soft links. Symbolic links are similar to hard links, but user can specify symbolic links for files across different file systems.



Types of Files in Linux

Types of Users in Linux

System Administrator

A System Administrator (SA) is administrator who handles the various operations of OS environment such as creating user, granting and revoking, restricting rights, handling the files, directories and etc. SA is authorized person for handling the entire OS which starts with system console on which OS is installed. Admin creates users, group, group policies and etc., The SA makes backups to prevent data loss if the operating system crashes. In Linux, the SA is also known as the root user. The SA has all the rights for the Linux operating system.

File Owner

File owner of the user created are concern user. They can do any operation on that file, such as copying, deleting, or editing.

Group Owner

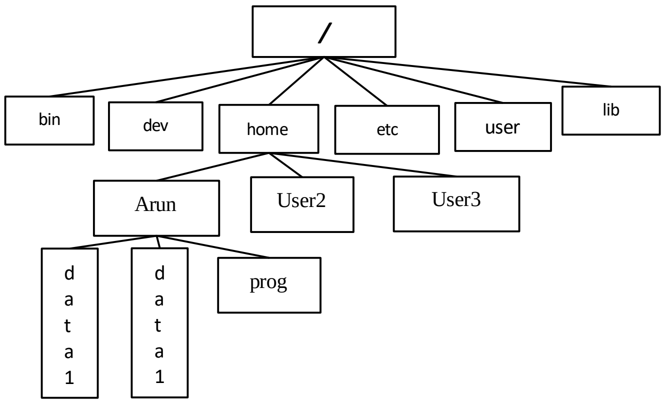
In a project development a team of users has to develop the modules which are confidential. In this case, the developed programs have to be stored in home directory of team leader. In this case, programmers are the file owners of their own program files. A programmer might need to link a program with another program to test the program. Therefore, each program also belongs to other programmers. SA needs a group to handle the programs in collectively manner. A group can be created with one of the member as group owner to handle these programs all together. In Linux, you can define the users who belong to a group. In addition, similar to a user name, you can specify a name for the group of users.

Other Users

All the users of the Linux operating system who are not members of the group are known as other users for the files of that group. Other users do not belong to the particular group.

2.2 Commands to handle Directories

User can type the commands at the Linux prompt. In Linux, user needs to type all commands in lowercase characters. For each of the commands the file hierarchy will be used.



A Sample Linux File Hierarchy

Identifying the Current Directory Path

pwd (print working directory) is the command used to display the current directory with complete path

Example

```
[Suresh@localhost Suresh]$ pwd
/home/Suresh
[Suresh@localhost bin]$ _
```

Suresh's currently working directory is */home/Suresh* in which the user files and directories are stored.

Changing the Current Directory

cd (change directory) is the command used to change the current directory to a specified directory. Assume that Suresh logs on to a Linux server and enters the following command:

Example

```
[Suresh@localhost Suresh]$ pwd
/home/Suresh
[Suresh@localhost Suresh]$ cd /usr/bin
[Suresh@localhost bin]$ pwd
/usr/bin
[Suresh@localhost bin] $ _
```

Note that the complete path is specified with the *cd* command. User can also use a relative path with the *pwd* command. Consider the following example:

```
[Suresh@localhost /usr]$ pwd
/usr
[Suresh@localhost /usr]$ cd bin
[Suresh@localhost bin]$ pwd
/usr/bin
[Suresh@localhost bin]$ _
```

In the above example the working directory */usr* is changed to */usr/bin* by the user *Suresh*. He gives only the directory name not complete path of the bin directory. Linux presumes that the directory is under the current directory and bin is a subdirectory of the current working directory. User can also use the ..(double period) option with the *cd* command to switch to the parent directory of the current directory.

Example

```
[Suresh@loealhost Suresh] $ pwd
/home/Suresh
[Suresh@loealhost Suresh] $ cd ..
[Suresh@loealhost /home]$ pwd
/home
[Suresh@loealhost /horne]$ cd ..
[Suresh@loealhost /]$ pwd
/
[Suresh@loealhost /]$ _
```

The two periods refer to the parent directory of the current directory. Note that user needs to specify a space between the words *cd* and the two periods but not between the periods. If user give *cd* command without any path then the current directory switch to the home directory of the user.

Example

```
[Suresh@localhost bin]$ pwd
```

```
/usr/bin
[Suresh@localhost bin]$ cd
[Suresh@loealhost Suresh] $ pwd
/home/Suresh
[Suresh@loealhost Suresh] $ _
```

The tilde symbol (~) is used to denote the complete path of home in Linux. Let us assume there are two directories baseball and basketball in Suresh s home directory.

Example

```
[Suresh@localhost etc] $ pwd
/etc
[Suresh@localhost etc]$ cd ~/baseball
[Suresh@localhost baseball] $ pwd
/home/Suresh/baseball
[Suresh@localhost baseball] $ cd ~
[Suresh@localhost Suresh] $ pwd
/home/Suresh
[Suresh@localhost Suresh] $ _
```

To switch from the directory datal to directory data2, Suresh needs to type the following *cd* command:

Example

```
[Suresh@localhost datal] $ pwd
/home/Suresh/datal
[Suresh@localhost datal] $ cd ../data2
[Suresh@localhost data2] $ pwd
/home/Suresh/data2
[Suresh@localhost data2] $ _
```

Creating a Directory

User can use of mkdir (make directory) command to create directories. For example

```
[Suresh@localhost Suresh] $ mkdir pro-file
[Suresh@localhost Suresh] $ _
```

When user types wants to create directory or sub-directory *mkdir* command can be used. Above example shows a subdirectory pro-file is created in the current directory. User can also specify the complete path with the *mkdir* command.

Example

```
[Suresh@localhost Suresh]$ mkdir /temp/pro-file
[Suresh@localhost Suresh]$ _
```

In the above example, the directory pro-file is created in the /*temp* directory.

Removing a Directory

User can use the *rmdir* (remove directory) command to remove a specified directory.

Example

```
[Suresh@localhost Suresh]$ rmdir pro-file
[Suresh@localhost Suresh]$ _
```

Here, the pro-file directory is deleted. User can also specify the complete path with the rmdir command.

Example

```
[Suresh@localhost Suresh]$ rmdir /home/Suresh/tennis
[Suresh@localhost Suresh]$ _
```

The above command removes the tennis directory, which is located in Suresh's home directory.

Listing the Contents of a Directory

To list the names of files and subdirectories that stored in a directory can be done with *ls* command.

Example

```
 [Suresh@localhost Suresh]$ ls /home/Suresh
DEADJOE X      baseball    comm        tennis
Desktop a.out basketball program.cc
 [Suresh@localhost Suresh] $
```

In the above example, all the files and directories in the Suresh directory are listed. To display the files and directories in the current directory, user need not specify the directory name with the *ls* command. In the above output, only the file names and not the file types are displayed. User should use the -1 option with the *ls* command to display a detailed list of files and directories. Example

```
[Suresh@localhost Suresh]$ ls -1
total 22
drwxr-xr-x  5 Suresh Suresh  1024 Sep 20 13:00 Desktop
-rw-rw-r-   1 Suresh Suresh     5 Sep 21 11:47 XYZ
-rwxrwxr-   1 Suresh Suresh  1290 Sep 20 16:13 prg1.out
drwxrwxr-x  2 Suresh Suresh  1024 Sep 21 00:11 Bball
drwxrwxr-x  2 Suresh Suresh  1024 Sep 21 00:12 Basket
drwxrwxr-x  2 Suresh Suresh  1024 Sep 21 00:12 Com.cc
-rw-rw-r-   1 Suresh Suresh    10 Sep 21 14:45 program.c
drwxrwxr-x  2 Suresh Suresh  1024 Sep 21 12:26 Cricket
      ↑     ↑    ↑      ↑     ↑    ↑   ↑   ↑      ↑
      1     2    3      4     5    6   7   8      9
```

The symbols *r*, *w,* and *x* in the first column specify the read, write, and execute permissions of a file or directory. The outcome of the 1s -1 is enlighten in the below table:

Column #	Description
1	File type and File Access Permissions (FAPs)
2	Number of links
3	File owner
4	Group owner (group name)
5	File size (in bytes)
6, 7 and 8	Day and time of the last modification to the file
9	Name of the file

A group is a collection of users or groups who share common attributes. In Linux, there are three kinds of users: file owners, group owners, and other users. The file and group owners are included in the detailed list for a file. The first character of column number 1 specifies the type of file. The following table explains the symbol and the associated file type:

Symbol	File Type
-	Ordinary
d	Directory
b	Block special file
c	Character special file
l	Symbolic link
p	Named pipe (also called FIFO)

The other common options available with the 1s command are as follows:

Option	Function
-a	Lists all the files, including the hidden files
-F	Shows the file type along with the name ('/' for a directory, '*' for an executable).
-R	Does a recursive listing, that is, displays the contents of the specified directory d all the subdirectories.
-r	Displays files and subdirectories in the reverse order.
-S	Sorts by descending order of the file size.
-A	Displays the files of all directories except the. and " directories.

User can identify a hidden file by its name. The name of a hidden file begins with a period. Hidden files are not displayed in an ordinary list. To display hidden files, user need to use the -a option with the *ls* command. When using the *1s* command, user can also combine more than one parameter. For example, a sample output of the *ls -al* command is displayed. User can also type the command as *ls -la* or *ls -a -1*. For example

```
[Suresh@localhost Suresh] $ 1s -al
total 42
drwx---          9  Suresh  TECH      1024  Sep   21 16:10 .
drwxr-xr-x      19  root    root      1024  Sep   2111:21 ..
-rw-r-r-         1  Suresh  Suresh    1422  Sep   20 13:00 .Xdefaults
-rw-r-r-         1  Suresh  Suresh      24  Sep   20 13:00 .bash  logout
-rw-r-r-         1  Suresh  Suresh     230  Sep   2013:00 .bash_profile
-rw-r-r-         1  Suresh  Suresh     124  Sep   2013:00  .bashr-:.
drwxr-xr-x       3  Suresh  Suresh    1024  Sep   2013:00  .kde
-rw-r-r-         1  Suresh  Suresh     966  Sep   2013:00  .kderc
drwxr-xr-x       5  Suresh  Suresh    1024  Sep   20 13:00 Desktop
-rw-rw-r-        1  Suresh  Suresh       5  Sep   2111 :47 X
-rwxrwxr-x       1  Suresh  Suresh   12901  Sep   2016:13  a.out
drwxrwx r-x      2  Suresh  Suresh    1024  Sep   21 15:41 Baseball
drwxrwxr-x       2  Suresh  Suresh    1024  Sep   21 00:12 basketball
drwxrwxr-x       2  Suresh  Suresh    1024  Sep   21 00:12 carom
-rw-rw-r-        1  Suresh  Suresh      10  Sep   21 15:40 program.cc
drwxrwxr-x       2  Suresh  Suresh    1024  Sep   21 12:26 tennis
```

Notice that the first two directories are present in all such directory lists. A period (.) represents the current directory, and two periods (..) represent the parent directory. The first two directories are the hidden directories. When user type the *cd . .* command, can switch to the parent directory. This is because (..) is a directory that points to the parent directory.

2.3 File operation related Commands

Displaying the Content of Files

cat (concatenate) command can be used by users to visualize the contents of a specified file. The *cat* command can vertically concatenate the contents of more than one file,

Example

```
[Suresh@localhost Suresh]$ cat datal
A sample file
[Suresh@localhost Suresh]$ _
```

If file is available in the current directory no need specify the complete path, otherwise user has to specify the complete path to display a file in another directory. User can also use the *cat* command to display the contents of more than one file, as shown in the following command:

Example

```
[Suresh@localhost Suresh]$ cat datal data2
A sample file
Another sample file
[Suresh@localhost Suresh]$ _
```

The** head **and** tail **Commands

Users use the *head* command to display the specified number of lines from the beginning of a file. User can use the *tail* command to display the specified number of lines from the end of a file. If user does not specify a parameter, 10 lines are displayed by default. The syntax for both the commands is:

Syntax

head [options] [file name]
tail [options] [file name]

User can use the head or tail command to display a certain number of lines on the screen.

Example

```
$ head -3 /etc/passwd root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbiIl:
```

This command displays the first three lines from the beginning of the file */etc/passwd.*
Some of the options available with the head command are given below:

Option	Function
-c, -bytes=SIZE	Displays the first <size> bytes.
-n, -lines=NUMBER	Displays the first <number> of lines instead of the first 10 lines.
-v, --verbose	Displays headers with file names.

Example

```
$ ls -1| tail -2
 -rw-rw-r-     1  Suresh  Suresh  38 Oct 19 22:28     Y
 -rw-rw-r-     1  Suresh  Suresh  48 Oct 12 05:09     y.bak
```

This command displays the last two files from the output of the *1s -1* command. Some of the options available with the tail command are given below:

Option	Function
-c, -bytes=N	Displays the last <n> bytes
-f, -follow	Displays appended data as the file grows.
-n, -lines=N	Displays the last <number> of lines instead of the last 10 lines
-v, -verbose	Displays headers with file names.

Copying Files

User can use the *cp* (copy) command to copy the contents of the source file into a target file.

Syntax

cp **[options] <source file/s> <destination directory/file>**

Example

```
[Suresh@localhost Suresh]$ cp datal data2
```

In the above example, the contents of the datal file are copied to a new file, data2. If the data2 file already exists, its contents will be overwritten by the contents of the datal file. To copy files across directories, users need to specify the complete path with the *cp* command. User can also copy a directory and all its files and subdirectories by using the *cp* command with the - r option.

Example

```
[Suresh@localhost Suresh]$ cp -r temp tempo
```

The above command copies the temp directory and all its files and subdirectories to the tempo directory. If the tempo directory exists, all the contents are added to that directory. If the tempo directory does not exist, it is created in the current working directory. Other common options of the *cp* command and their functions are given below:

Option	Function
-i	Prompts before overwriting
-l	Links a file instead of copying it
-s	Creates a symbolic link
-v	Verbose - explains what is being done, in detail

Removing Files

User can use the *rm* (remove) command to delete files or directories.

Syntax

rm [options] file/s

Example

```
[Suresh@localhost Suresh]$ rm datal data2
```

The above command will remove the datal and data2 files from the current directory. If the file to be deleted is not in the current directory, user needs to type the complete path.

Example

```
[Suresh@localhost Suresh]$ rm /home/Suresh/datal
```

User can use the -r option with the *rm* command to remove a directory along with its subdirectories.

Example

```
        [Suresh@localhost Suresh]$ rm –r tempo
```

The above command removes the tempo directory along with all its subdirectories. Other commonly-used options with the m command are displayed in the table below:

Option	Function
-i	Prompts before removing
-f	Removes a file by force, ignores the non-existence of a file (if the file does not exist, the command does not flag an error)
-r or –R	Deletes recursively, that is, deletes a directory along with its subdirectories.
-v	Verbose - explains what is being done, in detail

Moving and Renaming Files

User can use the *mv* (move) command to move a file or directory from one location to another or rename a file or directory. Note that moving a file to another location is different from copying the file. When user moves a file, a new file is not created.

Syntax

mv [option] source destination

Example

```
   [Suresh@localhost Suresh]$ mv comm communication
```

In the above example, the *comm* directory is renamed to *communication*. User can move a file from one directory to another directory, as shown below:

Example

```
[Suresh@localhost Suresh]$ mv data3 /home/Suresh /Programs
```

In the above example, the file data3 is moved from the current directory to the /home/Suresh/programs directory.

Example

```
   [Suresh@localhost Suresh)$ mv communication temp
```

In the above example, the directory *temp* exists in the current directory. Therefore, the *communication* directory is moved from the current directory to the *temp* directory. Some of the options available with the *mv* command are given below:

Option	Function
-i	Interactive, prompts before overwriting at the destination location
-f	If the file exits at destination, if overwrites the contents of the file without prompting
-v	Verbose - explains what is being done, in detail

Displaying the Contents Pagewise

User can use the *cat* command to display the contents of a file on the screen. However, if the displayed file is large, the entire contents of the file scroll up the screen. To view a file one screen at a time, user can use the more or the less command. User can use the more command to display the contents of a file one screen at a time. When user view the contents of a file by using the more command, user cannot scroll up after scrolling down.

Syntax

***more* [options] <file name>**

Example

```
[Suresh@lsrv0l HOWTO]$ more XWindow-User-HOWTO
```

The above command displays a pagewise list of the contents of the file, XWindow-User- HOWTO. Some common options of themore command and their functions are given below:

Option	Function
-num	Specifies an integer which is the screen size (number of lines) to be displayed
-f	Counts the lines logically instead of counting the screen lines
-s	Displays multiple blank lines as one.
-p	Disables scrolling, clears the screen to display the text

The *less* command is similar to the more command, except that user can scroll up while viewing the contents of a file. In addition, the *1ess* command is faster than the *more* command. This is because the *1ess* command accepts and caches data in the memory. Therefore, the data is available in the memory, and user can move up and down a screen faster.

Syntax

***less* [options] <file name>**

Example

```
[Suresh@lsrv0l HOWTO]$ less XWindow-User-HOWTO
```

The above command displays a pagewise list of the contents of the file *XWindow-User- HOWTO*. To move up and down the screen, user can use the arrow keys. User can also specify a number to move down the screen. To quit the display, user need to press the q key. Some common options of the 1ess command and their functions are given below:

Option	Function
-d	Scrolls forward N number of lines. Default is one half of the screen.
-u	Scrolls backward N number of lines. Default is one half of the screen
-r	Refreshes the screen
-f	Scrolls forward to read the end of a file mat is being modified

Wildcard Characters

In Linux, user can perform an operation on a set of files without specifying all the names of files on which the operation is to be performed. User can use special characters in the command instead of actual file names.

The shell interprets these special characters, also known as wildcard characters, as a specific pattern of characters. The shell then compares all the file names under the directory specified in the command to locate file names that match the pattern. The command is executed on files whose names match the pattern. The following table lists the wildcards available with a description of each:

Option	Function
*	Matches none or one character or a string of characters
?	Matches exactly one character
[ ]	Matches exactly one of a specified set of characters

*The * Wildcard*

The shell interprets the * wildcard as a string of none, one, or more characters.

Example

```
[Suresh@localhost Suresh]$ cat c*
```

Here, c * will match the files whose names start with c. User can also repeat the * wildcard in the command line.

Example

```
[Suresh@localhost Suresh]$ cat chap*.*
```

The above command displays all the files that begin with *chap* and contain any sequence of characters or no character, followed by a period, followed by any sequence of characters, or no character. Note that . is not a special character for *.

The ? Wildcard

The shell interprets the ? wildcard as a string of exactly one occurrence of any character.

Example

```
[Suresh@iocalhost Suresh]$ ls *.?
```

The above command displays all the files that contain any character(s) before a period, followed by a single character after the period.

The [ ] Wildcard

The [ ] wildcard can be used to restrict the characters to be matched.

Example

```
[Suresh@localhost Suresh]$ cat a[123]
```

The above command displays the contents of files with two character file names starting with ‘a’ and with the next character as 1, 2, or 3 such as *al, a2*, and *a3*.

Reference :

Operating System LINUX,NIIT,PHI,2006,Eastern Economy Edition.

Prepared by :

R.Nagadevi,
Assistant Professor,
Department of BCA,
Vidyasagar College of arts and science,
Udumalpet.

Reference Website :

www.scribd.com