UNIT 1

Introduction

The Operating system

An operating system is an software that manages computer hardware and provides a convenient and safe environment for running programs. It acts as an interface between programs and hardware resources that these programs access. It is loaded into memory when a computer is booted and remains active as long as machine is on.

Key features of an operating system :

- Operating system allocates memory for the programs and loads the program to the allocated memory.
- Operating system loads the CPU registers with control information related to the program.
- The instructions provided in the program are executed by the CPU. The operating system keeps track of the instruction that was last executed. This enables it to resume a program if it had to be taken out of the CPU before it completed execution.
- If program need to access the hardware, it makes a call to the operating system rather attempt to do the job itself.

  Eg:-If program needs to read a file on disk, the operating system directs the disk controller to open the file and make the data available to the program.
- After the program has completed execution the operating system cleans up the memory and registers and makes them available for the next program.

Modern operating system are MULTI-PROGRAMMING. They allow multiple programs to be in memory. On computers with single CPU only one program can run at any instant, rather than allow a single program to run to completion without interruption, an operating system allows a program to run for a small instant of time, save its current state and then load the next program in the queue. The operating system creates a process for each program and then control the switching of these processes.

Eg for operating system:- DOS,WINDOWS, UNIX etc.

1.1 The UNIX operating system:

Unix (trademarked as **UNIX**) is a family of multitasking, multiuser computer **operating systems** that derive from the original AT&T**Unix**, developed in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others.Unix is a operating system.

It has practical everything an operating system should have and several features which other operating system never had. Unix provides many features, but still many people still prefer stay away from unix, because It requires different type of commitments to understand the subject even when the user is an experienced computer professional.It introduces certain concepts not known to computing community before.

We can interact with unix system through a command interpreter called shell.It achieves unusual tasks with a few keystrokes.unix operating system doesn't tell whether user is right or wrong and doesn't warn the consequences of their actions.

1.2 History of Unix

Unix is actually a very old operating system ,until unix came on the scene,operating systems were designed with a particular machine in mind.They were written in low leval language.Programs designed for one system simply wouldn't run on another .so two persons named Ken Thompson and Dennis Ritchie, of AT & T ,in 1969,started to build a flexible operating system that would run on any hardware. they designed and built a elegant file system, command interpreter(shell) and a set of utilities. But this system was once again hardware dependent.

In 1973,Dennis Ritchie rewrote the entire system in C –a high level language and it was portable. This was the first version of unix.

The University of California Berkley(UCB),created a unix of its own. They called it BSD UNIX.(Berkley software distribution).These versions became quite popular world wide.BSD Unix had a standard editor of the unix system(vi) and a popular shell(c shell).Berkley also created a better file system, a more versatile mail feature and a better method of linking files. Later they also offered with their standard distribution a networking protocol software(TCP/IP) that made the internet possible.

Sun used the BSD unix as a foundation for developing their own brand of unix called SunOS. Today their version of UNIX is known as solaries. Others had their own brands such as IBM had AIX,HP offered HP-UX.

As each vendor modified and enhanced Unix to create their own versions ,the original unix lost its identity as a separate product. There was no Standard version of unix. Hence AT & T Released SVR$(system V Release 4), a standard unix.

Even before the advent of SVR4, big things were happening in the U.s Defense Department .They created the first communication network called ARPANET and it used TCP/IP protocol and they tried to implement TCP/IP on BSD Unix. The incorporation of TCP/IP into unix and its use as the basis of development were two key factors in the rapid growth of the internet.

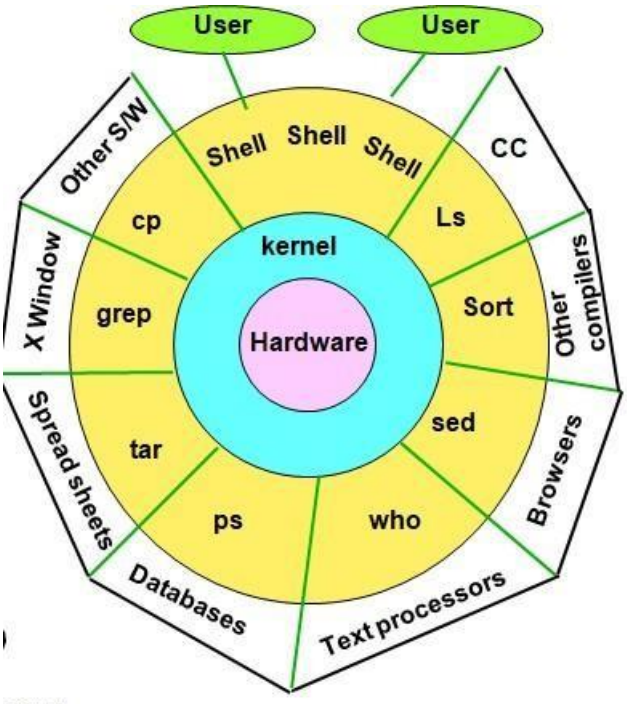
In the mean time ,Microsoft released a new operating system called Windows, which used GUI(Graphical user interface) that uses mouse rather than complex commands to execute a job. Windows was a big threat to UNIX, because it was a windowed system. So Unix badly needed a windowed type interface for its survival hence the Massachusetts Institute of Technology(MIT) introduced X Window-the first windowing system for unix.

Linus Torvalds-Father of Linux developed Linux. Linux is distributed free under general public licence, which makes it mandatory for developers and sellers to make the source code public. Linux is particularly strong in networking and internet features.

The most popular Linux flavors include Red Hat, Caldera, Mandrake, Fedora etc.

1.3 Unix Arcitecture /components

The UNIX architecture and command usage



Division of labor: Kernel and shell

The main concept in the unix architecture is the division of labor between two agencies the KERNEL and SHELL. The kernel interacts with hardware and shell interacts with user.

The kernel is the core of the operating system- a collection of routines mostly written in C. It is loaded into memory when the system is booted and communicates directly with the hardware. User applications (programs) that need to access the hardware, uses the services of kernel. These programs access the kernel through a set of function called system calls.

Apart from providing support to user programs, the kernel also performs other tasks like managing system's memory, scheduling processes, decides their priorities etc. So the kernel is often called the operating system- a program's gateway to the computer's resource.

The shell is the command interpreter, it translates command into action. It is the interface between user and kernel. System contains only one kernel, but there may be several shells.

When user enters a command through the keyboard the shell thoroughly examines keyboard input for special characters, if it finds any, it rebuilds simplified command line and finally communicates with kernel to see that the command is executed.For eg,consider a echo command which has lots of spaces between the arguments:

Eg: $echo Sun solaris

In the above example shell first rebuilds the command line i.e. it will compress all extra spaces, then it will produces output.

Sun solaris

The file and process:

Two simple entities supports the unix system is the file and process.

A file is just an array of bytes and can contain virtually anything. It is also related to another file by being part of a single hierarchical structure. So it is necessary to locate a file within reference to a pre-determined place. Unix considers even directories and devices as files.Files will be arranged in a hierarchical structure in unix system.

A process is the name given to a file when it is executed as program. So the process is an time image of executable file. Process are treated as living organism which have parents, children and grand-children and are born and die.Like files ,processes also belong to a hierarchical tree structure.

The system calls:

Unix system's kernel, shell and applications are written in C. There are several commands in the unix system, they uses system call functions to communicate with kernel.

Eg: Unix command writes into the file using write system call.Open system call can be used to open the file.

1.4 Features of UNIX:

- **Multi-user system:**

Unix is a multi-user system i.e. multiple user can use the system at a time, resources are shared between all users. In unix systems, computer breaks up a unit of time into several segments. So at any point in time, the machine will be doing the job of a single user. The moment the allocated time expires, the previous job will be preempted and next user's job is taken up. This process goes on until clock has turned full circle and the first user's job is taken up once again.Unix is a multi-programming system, it permits multiple programs to run. This can happen in two ways:

- Multiple user can run separate jobs.
- A single user can also run multiple jobs.

- **Multi-tasking system:**

A single user can run multiple tasks concurrently, in multitasking environment a user sees one job running in the foreground, the rest running in the background. It is possible to switch the job between background and foreground, suspend or even terminate them.

- **Building block approach:**

Unix contains several commands each performs one simple job. It is possible to connect different with the pipe (|) to get different jobs done. The commands that can be connected in this way are called filters, because they filter or manipulate data in different way.for ex: ls command lists all files.wc command counts the number of words .They can be combined using pipes(|) to find the total number of files in the directory.

- **Unix tool kit:**

  Unix contains set of tools like general purpose tools, text manipulation utilities (called filters), compilers, interpreters, networked application and system administration tools.

- **Pattern matching:**

  Unix contains pattern matching features, using this features it is possible to match the different strings.

  Eg: '*'

  $ls chap*

  Here '*' is the special character, which matches the filename, which starts with 'chap'.

- **Programming facility:**

  Unix shell is also a programming language. It has control structures, variables, loops that establish it as a powerful programming language. This features can be used to design a shell scripts.

- **Documentation:**

  Unix contains 'man' pages for each command, it contains references for commands and their configuration files.Apart from man page, we can even get the command information in internet.there are several newsgroups on unix where we can fire our queries to get the solution to a problem.

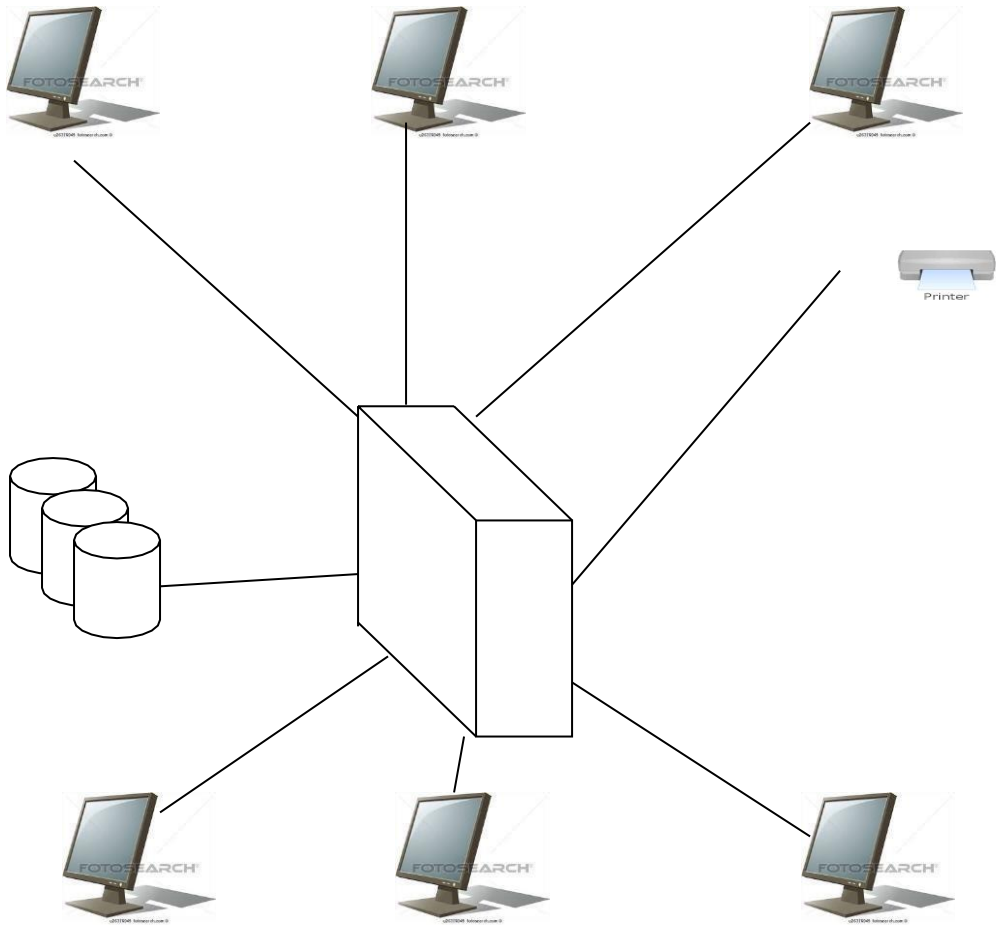
1.5 Unix Environments

Unix can be used in 3 different environments:

1) Personal Environment
2) Time sharing environment
3) Client-server environment

Personal environment : Even though unix is a multi user operating system ,it can be installed in personal computers. This is called as personal environment.

Time sharing environment :

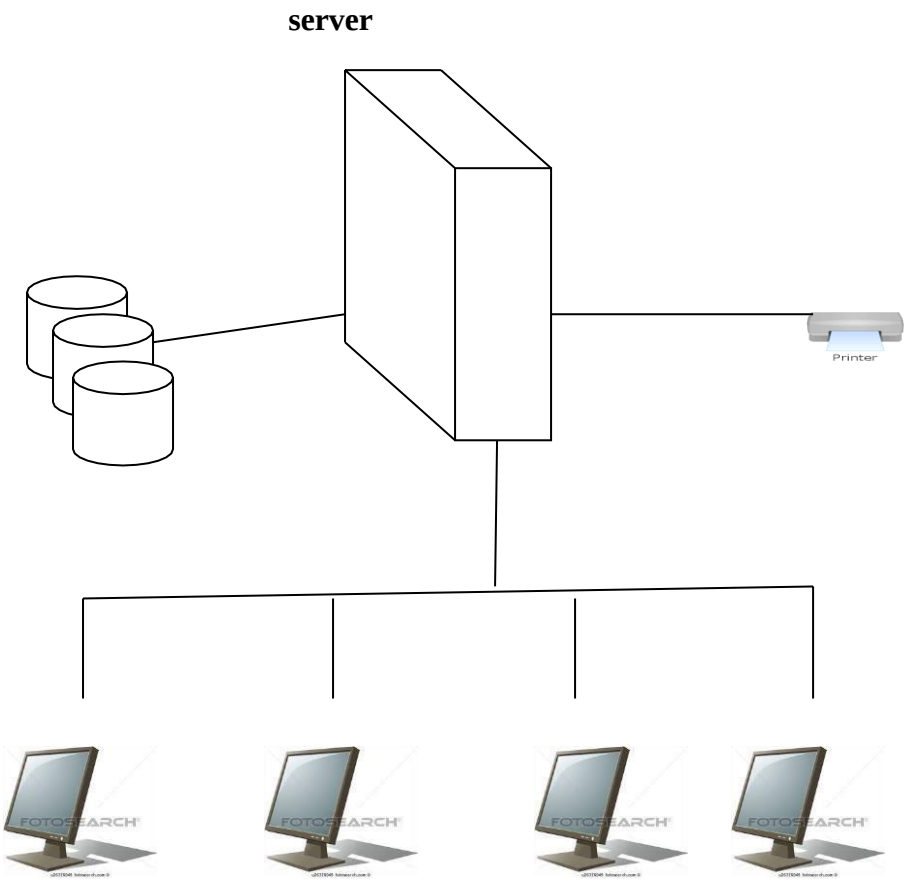


The time sharing environment

Employees in large companies often work in so called time sharing environment.In a time sharing environment many users are connected to one or more computers.Their terminals are often nonprogrammable.also in a time sharing environment the output devices and storage devices are shared by all users .A typical time sharing environment has been shown shown in above figure.

In time sharing environment,all of the computing must be done by the central computer.centeral computer has many duties.It must control the shared resources .I t must manage the shared data and printing and it must also do the printing. All the workload will be given to central computer.Hence the system becomes slow.

Client-server Environment



The client –server environment

A client server computing environment splits the computing function between a central computer called server and users computers.In client –server environment users microcomputers or workstations are called as clients.The client is a one which requests for service. The central computer which is a powerful microcomputer,minicomputer or supercomputer is called a server.Server is a one serves the request of a client.The users are given personal computers or workstations so that some of the computations can be moved off to clients.Here workload will be shared between server and clients and hence the system becomes fast.

Unix Structure:

Note: For Diagram refer the Unix Architecture.

The main components of the Unix structure are:

1. Kernel.
2. Shell.
3. Application.

1. **Kernel:**

The kernel provides a bridge between the hardware and the user. It is a software application that is central to the operating system. The kernel handles the files, memory, devices, processes and the network for the operating system. It is the responsibility of the kernel to make sure all the system and user tasks are performed correctly.

2. **Shell:**

The program between the user and the kernel is known as the shell. It translates the many commands that are typed into the terminal session. These commands are known as the shell script. There are two major types of shells in Unix. These are Bourne shell and C Shell. The Bourne shell is the default shell for version 7 Unix.

The character $ is the default prompt for the Bourne shell. The C shell is a command processor that is run in a text window. The character % is the default prompt for the C shell.

3. **Application:**

The applications and utility layer in Unix includes the word processors, graphics programs, database management programs, commands etc. The application programs provide an application to the end users.

For example, a web browser is used to find information while gaming software is used to play games. The requests for service and application communication systems used in an application by a programmer is known as an application program interface (API).

1.6 POSIX and single unix specification:

POSIX- Portable Operating System standard for Computer Environment (specially for UNIX) is the group of standards developed by IEEE for operating system.

Two standards from Posix family are POSIX.1 and POSIX.2

POSIX.1 deals with C application program interface.

POSIX.2 deals with shell and utilities.

- X/OPEN (open group) developed standard for UNIX i.e. X/OPEN portability guide (XPG).
- In 2001, X/OPEN and IEEE unified these two standards and called it as SUSV3-Single Unix
- Specification Version 3. This is based on "write once, adopt anywhere" approach, i.e. once software has been developed on any POSIX compliant UNIX system, it can be easily ported to another POSIX compliant UNIX system with minimum modification.

1.7 The General features of unix commands/command structure

Command structure:

The syntax of command is

$command [options] [arguments]

For ex:

$ ls –x chap*

In the above example ls is the command, –x is a option and chap* is a argument. The command with its arguments and options is known as command line.

The first word in the command line is called as command, all subsequent words are actually called as arguments. options are also arguments but given a special name because their list is predetermined. There should be atleast one space to separate command and arguments. multiple spaces are also allowed but shell compresses them to form a simplified command line.

Options:

There is a special type of argument, that is mostly used with a '-' sign. Such a arguments are called options.

Eg: $ls –l

-l : long listing showing seven attributes of a file.

-l is an option to ls command. Options are also arguments but their list is predetermined. Options can normally be combined with single '-' sign instead of using

$ls –l –a –t

i.e.

$ls –lat

File name arguments:

Many unix commands use a filename as argument, so the command can take input from the file. It is even possible to give multiple filename as argument.

Eg:

$ls –l chap1 chap2

The command with its arguments and options is known as command line.

Exceptions:

There is exceptions to the general syntax of commands. There are commands (pwd) that don't accept any arguments. Some commands (EX: who) may or may not take arguments. Some commands like

'ls' can run without arguments(ls), with only options (ls -l), with only filenames (ls chap1 chap2) or can be run with options and filenames (ls –l chap1 chap2).

1.8 Understanding some basic commands

echo: Displaying a message:

echo command can be used to

- To display a message (like echo "cseise")
- To evaluate shell variables (echo $PAGER)

Originally echo was an external command, but now all shells have echo as built in command. We can see escape sequence with echo, but we should use –e option.

Some escape sequences are:

\a	Bell
\b	Backspace
\c	No newline
\f	Form feed
\n	New line
\r	Carriage return
\t	Tab
\v	Vertical tab
\\	Backslash
\0n	ASCII character represented octal value n

echo interprets number octal when it is preceded by \0.

Eg: ASCII value of (ctrl-g) is 7, which results in sounding of a beep, we can use this value as an argument to echo but only after preceding it with \0

$echo '\07'

--beep heard- -

printf : An alternative to echo:

printf can be used as alternative command to echo. The difference is echo command by default inserts '\n' character at the end, but printf does not insert newline at the end.

Eg: $printf cseise

Output: cseise $ //prompt will be in the sameline.

Printf also accept all escape sequence used by echo. We can use format specifiers with printf it acts as place holder for variable.

Some format specifiers are:

%30s	String printed in a space 30 character wide
%s	String
%d	Decimal integer
%6d	Integer printed in a space 6 characters wide
%o	Octal integer
%x	Hexadecimal integer
%f	Floating point number

Eg:

$printf "Value of 255 is %o in octal and %x in hexadecimal\n" 255 255

Output: Value of 255 is 377 in octal and ff in hexadecimal.

Date : to display date and time

The Unix system maintains an internal clock meant to run perpetually. When the system is shut down a battery backup keeps the clock ticking. Date command is used to display both and time as follows:

$date

Tue Aug 28 1:58:03 IST 2018

Date displays 6 fields of output:

1. day 2.Month name 3. Date 4.time 5.time zone 6.year

To extract individual fields we can use following options with date

Options with date:

+%a – day

+%h – Month name

+%d – date

+%T –time

+%Z- Time Zone

+%Y-year

+%H-Hour

+%M- Minute

+%S - Second

Ex:

$ date +%h

Aug

$date +"%h %d "
Aug 28

ls: listing files:

It is possible to list the names of the files in the directory using ls command.

Eg: $ls

README
Chap01
Chap02
Chap03

We can use special shorthand notation * with ls command.

Eg: $ls chap*

Chap01
Chap02
Chap03

i.e. it will displays the filename starts with 'chap'.
ls command can also be used to search for an command

$ ls a.txt
The above command searches for file a.txt and if it is present it displays it otherwise displays the message "file not found".

Options used with ls

-a : Displays all files including hidden files,current directory(.) and parent directory(..)
-d dirname : Searches for directory with name dirname,if found displays it,otherwise produces a
message directory not found.
-i : displays file with inode number
-l : displays all files with seven attributes such as permissions,links,owner name ,group
name,size in bytes,date and time ,file name.
-r : displays files in reverse alphabetical order
-R : Displays recursive list of files
-t : Sorts file based on last modified time
-u : sorts files based on last access time
-x : multi columnar output

Directing output to a file:

Unix has special symbols called meta-characters for creating and storing information in files. '>' symbol can be used to save the information in the file.

Eg: $ls > list

Stores the output of ls command in the file 'list'.

wc: counting number of lines in a file :
we can use 'wc' command to count number of lines, words and characters in a file.

Eg: $wc list.txt

6 6 42 list.txt

Output shows that the file 'list' contains 6lines, 6words and 42 characters.

Feeding output of one command to another:

It is possible to give output of one command as the input to the another command '|' (pipe) it connects two commands to create pipeline.

Eg: $ls | wc

6 6 42 //no file name

who: Who are the users?:

Unix maintains an account of all users, who are logged on to the system. 'who' command displays listing
of these users.

```
$who
abc     pts/1   Jan     1       20:25   (:0)
                                        (heavens.co
xyz     pts/10  Jan     1       14:49   m)
```

First column shows the username, second shows device name of user's terminal, third, fourth, fifth shows
date and time of logging in. User can login remotely to a unix system. Last column shows the machine name from where user logged in. (:0) indicates user logged in from his own terminal.

We can use options with who command.

-H	:	To display the header.
-u	:	detailed list

Eg: $who –Hu

NAME	LINE	TIME	IDLE	PID	COMMENTS
abc	pts/1	Jan 13 07:51	0:48	11040	(:0)
xyz	pts/2	Jan 13 07:56.		11052	heavens.com

First column shows name of the user, second shows terminal name, 3rd,4th,5th shows date and time. 6th column shows system's IDLE timing, i.e. from how long system is idle. A '.' Against xyz shows that activity has occurred in the last minute before the command was invoked.

'abc' seems to idling for the last 48minutes. The PID is the process ID, a number that uniquely identifies a process. We can use the argument 'am''i' with who to know who logged in the system.

Eg: $who am i

abc pts/1 Jan 13 07:51 (:0)

passwd: changing password:

We can use passwd command to change the password.

Eg: $passwd
Passwd: changing password for xyz
Enter login password: ******
New password: **********
Re-enter new password: **********
Passwd (SYSTEM) :passwd successfully changed for xyz

When user enters a password, the string is encrypted by the system. Encryption generates string of random characters and stores in the file /etc/shadow

Password framing rules:

Some of the rules which should be followed while framing password:

- Don't choose a password similar to old one.
- Don't use commonly used names like names of friends, relatives, pets, etc.
- Use mix of alphabetic or numeric characters.
- Don't write password in a easily accessible document.

- Change password regularly.

cal: The calendar-

Using cal command it is possible to see the calendar of specific month.

Syntax:

$cal [[month] year]

Arguments are optional here. So 'cal' can be used without arguments.

```
$cal                    //displays the calendar of current
                        month

                    January 2013
    Su   Mo   Tu   We   ThFr   Sa
                  1    2    3    4     5
     6    7    8    9   10   11   12
    13   14   15   16   17   18   19
    20   21   22   23   24   25   26
    27   28   29   30   31
```

We can use cal with pager more or less.
Eg:
$ (cal 2003; cal 2006; cal 2007) | more
We can use cal command with arguments.
Eg:
$cal 03 2006
Displays calendar of march 2006.
When 'cal' is used with arguments month is optional but year is mandatory. A single argument to cal is interpreted as year.

Eg: $cal 2003
Displays the calendar of 2003

Combining commands:

Unix allows to specify more than one command in the command line. Each command has to be separated from the other by a ; (semi-colon).

Eg: $wc note ; ls –l note

Output will be line, word and character count of the file note and long listing of attributes of file note.

Eg: $ (wc note; ls –l note) > list

Output of 'wc note' and 'ls –l' will be redirected to the file list.

1.9 Meaning of Internal and External commands

Internal and external commands:

External commands will have independent existence in one of the directories specified in the PATH variable.

Eg: $ls
ls is /bin/ls

'ls' command having an independent existence in the /bin directory (or /usr/bin). So 'ls' is a external command.

Internal commands doesn't have independent existence, they are shell built-ins.

Eg: $type echo
echo is a built-in.

Shell is an external command with difference, i.e. shell possesses its own set of internal commands. If a command exists both as an internal command of the shell as well as an external one, the shell will give top priority to its internal command.

Eg: echo command, which is also found in /bin directory but rarely ever executed, because the shell makes sure that the internal echo command takes precedence over the external.

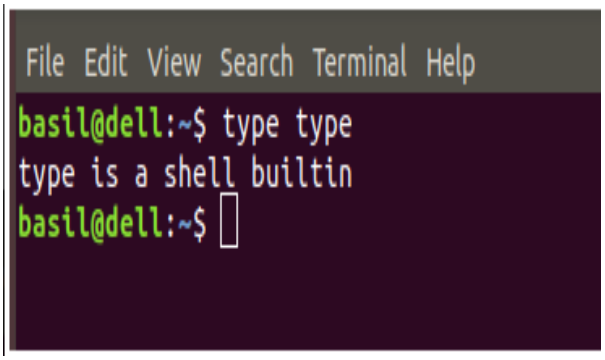
Command arguments can take the form of an expression (in grep), a set of instructions (in sed), or a program (in awk and perl).

1. 10 The type command: knowing the type of a command and locating it.

The type command is used to describe how its argument would be translated if used as commands. It is also used to find out whether it is built-in or external binary file.

Synatx:

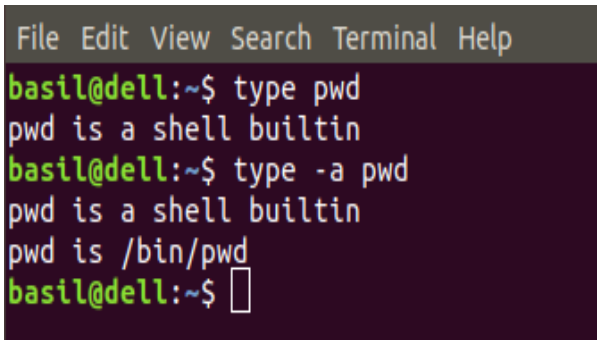
type [options] command names



Type command options:

-a: This option is used to find out whether it is an alias, keyword or function and it also displays the path ofan executable. If available.

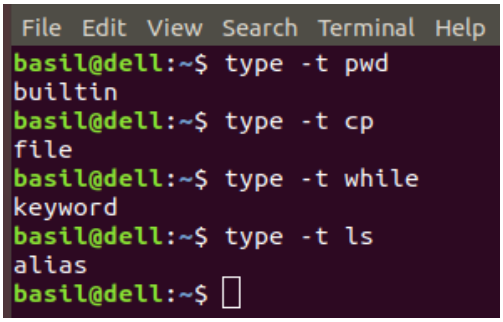
Ex: type –a pwd



-t: This option will display a single word as an output. alias – if command is a shell alias

1. keyword – if command is a shell reserved word
2. builtin – if command is a shell builtin
3. function – if command is a shell function
4. file – if command is a disk file

Ex: type -t pwd
type -t cp
type -t ls
type -t while



1.11 The root login

The unix system provides a special login name for the exclusive use of the administrator. It is called root.This account doesn't need to be separately created but comes with every system. Its password is generally set at the time of installation of the system and has to be used on logging in:

Login : root
Password : ******[Enter}
:

The prompt of root is #,unlike the $ or % used by nonprivileged users.Once you login as root, you are placed in roots home directory.Depending on the system this directory could be / or /root.roots PATH list is also different from the one used by other users:

/sbin : /bin : /usr/sbin : /usr/bin /usr/dt/bin

su: Becoming the superuser

Any user can acquire superuser status with the su command if he/she knows the root password. For example ,the user Juliet becomes a superuser in this way :

```
 $ su
Password: *********
# pwd
/home/Juliet
```

Though the current directory doesn't change ,the # prompt indicates that Juliet now has powers of a superuser.To be in root's home directory on superuser login use su –l.

Creating user's environment :
Users often rush to the administrator with the complaint that a program has stopped running.The administrator first tries running it in a simulated environment .Su ,when used with a -,recreates the users environment without taking the login-password route:

Su – henry

This sequence executes henry's profile and temporarily creates henry's environment.This mode can be terminated by hitting [ctrl-d] or using exit.

UNIX FILES

Introduction:
Files are the building blocks of any operating system. When you execute a command in UNIX, the UNIX kernel fetches the corresponding executable file from a file system, loads its instruction text to memory, and creates a process to execute the command on your behalf. In the course of execution, a process may read from or write to files. All these operations involve files. Thus, the design of an operating system always begins with an efficient file management system.

Naming Files:
A filename can consists of upto 255 characters. Files may or may not have extensions, and can cosists of practically any ASCII character except the / and the NULL character(ASCII value 0). We are permitted to use control characters or other unprintable characters in a filename. The following are the valid filenames in UNIX:

- .last_time
- list.
- ^V^B^D-++bcd
- -{}[]
- @#$%*abcd
- a.b.c.d.e

The third filename contains three control characters. These characters shoud be definitely be avoided in framing filenames. Moreover, since the UNIX system has a special treatment for characters like $, `, ?, *, & among others, it is recommended that only the following characters be used in filenames.

Alphabetic characters and numerals.
The period(.), hyphen(-) and underscore(_).
Unix imposes no rules for framing filename extensions.
DOS/Windows users must also keep these two points in mind:

- A file can have as many dots embedded in its name; a.b.c.d.e is a perfectly valid filename. A filename can also begin with a dot or end with one.
- UNIX is case sensitive to case; Chap01, chap01. And CHAP01 are three different filenames, and it's possible for them to coexist in the same directory.

Basic file types/categories:

1. Ordinary (Regular) File.
2. Directory File.
3. Device File.

1. Ordinary (Regular) File:

An ordinary file or regular file is the most common file type. All programs you write belong to is ppe. An ordinary file itself can be divided into two types:

1. Text File.
2. Binary File.

1. Text File:

A Text file contains only printable characters, and you can often view the contents and make sense out of them. All C and Java program sources, shell and perl scripts are text files. A text file contains lines of characters where every line is terminated with the newline character, also known as line feed (LF). When you press (Enter) while inserting text, the LF character is appended to every line. You won't see this character normally, but there is a command (od) which can make it visible

2. Binary File:

A binary file, on the other hand, contains both printable and unprintable characters that cover the entire ASCII range (0 to 255). Most UNIX commands are binary files, and the object code and executables that you produce by compiling C programs are also binary files. Picture, sound and video files are binary files as well. Displaying such files with a simple cat command produces unreadable output and may even disturb your terminal's settings.

2. Directory File:

A directory contains no data, but keeps some details of the files and subdirectories that it contains The UNIX file system is organized with a number of directories and subdirectories, and you can also create them as and when you need. You often need to do that

to group a set of files pertaining to a specific application. This allows two or more files in separate directories to have the same filename.

A directory file contains an entry for every file and subdirectory that it houses. If you have 20 files in a directory, there will be 20 entries in the directory Each entry has two components: .

- The filename
- A unique identification number for the file or directory (called the inode number),

If a directory bar contains an entry for a file foo, we commonly (and loosely) say that the directory bar contains the file foo Though we'll often be using the phrase "contains the file rather than "contains the filename, you must not interpret the statement literally A directory contains the filename and not the file's contents.

3. Device File:

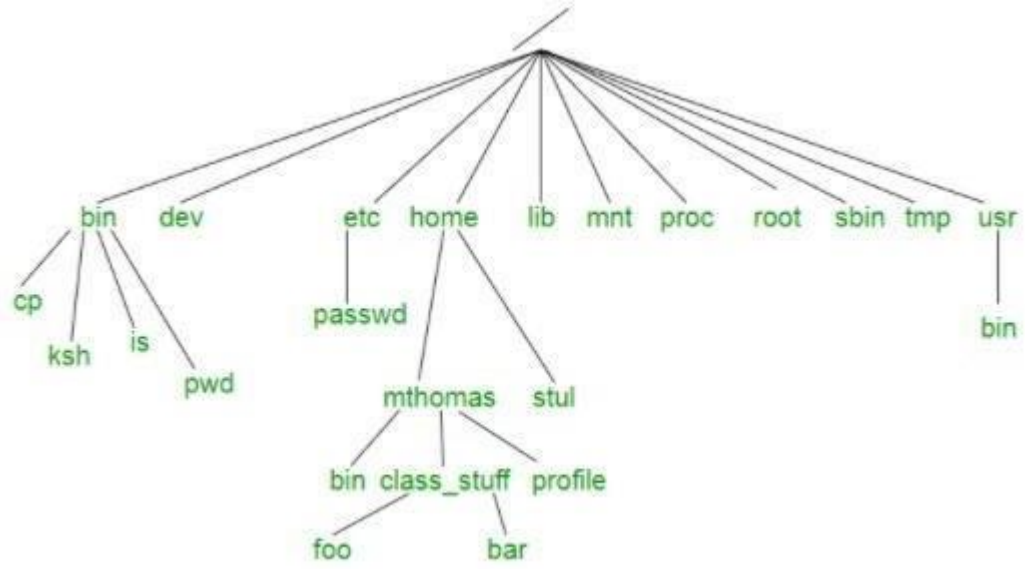
You'll also be printing files, installing software from CD-ROMs or backing up files to tape. All of these activities are performed by reading or writing the file representing the device. For instance, when you restore files from tape you read the file associated with the tape drive. It is advantageous to treat devices as files as some of the commands used to access an ordinary file also work with device files.

Device file names are generally found inside a single directory structure, /dev. A device file is indeed special, it's not really a stream of characters. In fact, it doesn't contain anything at all.

Organization of files & Standard Directories:

Unix file system is a logical method of **organizing and storing** large amounts of information in a way that makes it easy to manage. A file is a smallest unit in which the information is stored. Unix file system has several important features. All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system.

Files in Unix System are organized into multi-level hierarchy structure known as a directory tree. At the very top of the file system is a directory called “root” which is represented by a “/”. All other files are “descendants” of root.



Directories or Files and their description:

- **/ :** The slash / character alone denotes the root of the filesystem tree.
- **/bin :** Stands for "binaries" and contains certain fundamental utilities, such as ls or cp, which are generally needed by all users.
- **/boot :** Contains all the files that are required for successful booting process.
- **/dev :** Stands for "devices". Contains file representations of peripheral devices and pseudo-devices.
- **/etc :** Contains system-wide configuration files and system databases. Originally also contained "dangerous maintenance utilities" such as init,but these have typically been moved to /sbin or elsewhere.
- **/home :** Contains the home directories for the users.
- **/lib :** Contains system libraries, and some critical files such as kernel modules or device drivers.
- **/media :** Default mount point for removable devices, such as USB sticks, media players, etc.
- **/mnt :** Stands for "mount". Contains filesystem mount points. These are used, for example, if the system uses multiple hard disks or hard disk partitions. It is also often used for remote (network) filesystems, CD-ROM/DVD drives, and so on.
- **/proc :** procfs virtual filesystem showing information about processes as files.
- **/root :** The home directory for the superuser "root" – that is, the system administrator. This account's home directory is usually on the initial filesystem, and hence not in /home (which may be a mount point for another filesystem) in case specific maintenance needs to be performed, during which other filesystems are not available. Such a case could occur, for example, if a hard disk drive suffers physical failures and cannot be properly mounted.
- **/tmp :** A place for temporary files. Many systems clear this directory upon startup; it might have tmpfs mounted atop it, in which case its contents do not survive a reboot, or it might be explicitly cleared by a startup script at boot time.
- **/usr :** Originally the directory holding user home directories,its use has changed. It now holds executables, libraries, and shared resources that are not system critical, like the X Window System, KDE, Perl, etc. However, on some Unix systems, some user accounts may still have a home directory that is a direct subdirectory of /usr, such as the default as in Minix. (on modern systems, these user accounts are often related to server or system use, and not directly used by a person).
- **/usr/bin :** This directory stores all binary programs distributed with the operating system not residing in /bin, /sbin or (rarely) /etc.
- **/usr/include :** Stores the development headers used throughout the system. Header files are mostly used by the **#include** directive in C/C++ programming language.
- **/usr/lib :** Stores the required libraries and data files for programs stored within /usr or elsewhere.
- **/var :** A short for "variable." A place for files that may change often – especially in size, for example e-mail sent to users on the system, or process-ID lock files.
- **/var/log :** Contains system log files.

- **/var/mail :** The place where all the incoming mails are stored. Users (other than root) can access their own mail only. Often, this directory is a symbolic link to /var/spool/mail.
- **/var/spool :** Spool directory. Contains print jobs, mail spools and other queued tasks.
- **/var/tmp :** A place for temporary files which should be preserved between system reboots.

Hidden Files:

To show all the hidden files in the directory, use '-a option'. Hidden files in Unix starts with '.' in its file name.It will show all the files including the '.' (current directory) and '..' (parent directory).

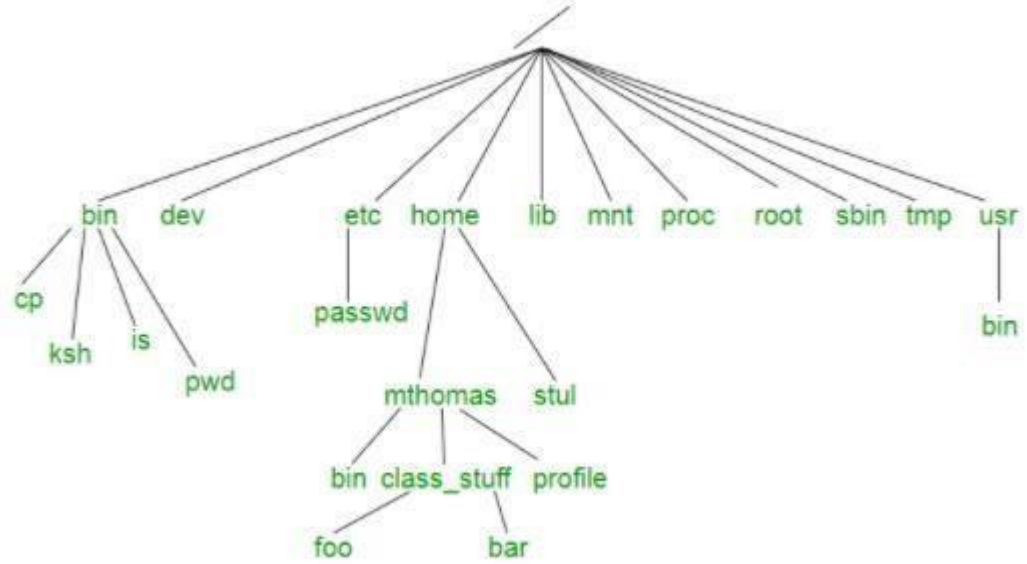
ls –a

```
[SantoshReddyP@webminal.org ~]$ls -a
.     A.txt          .bash_profile  .magic_string.txt  .Sandy.txt  VA036
..    .bash_history  .bashrc        Pgm3.txt           Unix        .viminfo
.ABC  .bash_logout   .emacs         Pgm7.txt           Unixs.txt   .zshrc
```

ls –A: To show the hidden files, but not the '.' (current directory) and '..' (parent directory).

```
[SantoshReddyP@webminal.org ~]$ls -A
.ABC           .bash_profile      Pgm3.txt    Unixs.txt
A.txt          .bashrc            Pgm7.txt    VA036
.bash_history  .emacs             .Sandy.txt  .viminfo
.bash_logout   .magic_string.txt  Unix        .zshrc
```

Parent child relationship:



- All files in unix are related to one another.
- The file system is organized in a hierarchical structure.
- The implicit feature of unix file system is that there is a top directory which is called as root which serves as the reference point for all files.

- Top is represented by a /(front slash).
- Root directory(/) has several sub directories under it.
- These subdirectories in turn have more sub directories and other files under them.
    - Ex: bin and usr are two directories directly under /, while cp and pwd are subdirectories under bin.
- Every file must have a parent and it should be possible to trace the ultimate parentage of a file to root.
- Thus, the home directory in the above figure is the parent for mthomas, while / is the parent of home and the grand parent of mthomas.
- In the parent-child relationship, the parent is always a directory.
    - Ex: login.sql is an ordinary file, it cannot have a directory under it.

The Home Variable and The Home Directory:

- When we log on to the system, UNIX automatically places you in a directory called the home directory. It is called by the system when a user account is opened.
- If you log in using the login name Kumar, you will land up in a directory that could have the pathname /home/kumar.
- Home directory can be changed by the user.
- The shell variable HOME knows your home directory.
    - E: $echo $HOME
        - /home/Kumar
- The above output is an absolute pathname.
- The slashes act as a delimiter to the file and directory names except the first /(root).

Directory Commands:

1. **pwd: Checking your current directory**
    - User can move around from one directory to another, but at any point of time, if user wants to find out in which directory he is present then user can use Print Working Directory(pwd) command.

      Ex: pwd
      /home/kumar/SantoshReddyP

      pwd also displays the absolute pathname.

2. **cd: Changing the current directory:**
    - User can move around the UNIX file system using cd (change directory) command.
    - When used with the argument, it changes the current directory to the directory specified as argument, progs:

      $ pwd
      /home/kumar
      $cd progs

```
$ pwd
/home/kumar/progs
```

- Here we are using the relative pathname of progs directory. The same can be done with the absolute pathname also.

```
$cd /home/kumar/progs
$ pwd
/home/kumar/progs
$cd /bin
$ pwd
/bin
```

- cd can also be used without arguments:

```
$ pwd
/home/kumar/progs
$cd
$ pwd
/home/kumar
```

- cd without argument changes the working directory to home directory.

```
$cd /home/sharma
$ pwd
/home/sharma
$cd
 /home/kumar
```

3. mkdir: Making Directory

- Directories are created with mkdir (make directory) command. The command is followed by names of the directories to be created. A directory patch is created under current directory like this:

```
$mkdir patch
```

- You can create a number of subdirectories with one mkdir command:

```
$mkdir patch dba doc
```

- For instance the following command creates a directory tree:

```
$mkdir progs progs/cprogs progs/javaprogs
```

- This creates three subdirectories – progs, cprogs and javaprogs under progs.
- The order of specifying arguments is important. You cannot create subdirectories before creation of parent directory.
- For instance following command doesn‘t work

```
$mkdir progs/cprogs progs/javaprogs progs
mkdir: Failed to make directory “progs/cprogs”; No such directory
mkdir: Failed to make directory “progs/javaprogs”; No such directory
```

- System refuses to create a directory due to fallowing reasons:
  - The directory is already exists.
  - There may be ordinary file by that name in the current directory.
  - User doesn‘t have permission to create directory.

4. **rmdir: Removing a Directory**
   - rmdir command removes the directory.
   - The directory must be empty before using rmdir with the directory name.
     - Ex: rmdir pis
   - Multiple directories can be removed.
     - Ex: $rmdir pis/data pis/progs pis
- Here first subdirectories are removed and at last pis is removed.
- A subdirectory cannot be removed unless the user is placed in a directory which is hierarchically above that directory.
  - Ex:$pwd
    /home/kumar/pis/progs
    $cd /home/kumar/pis
    /home/kumar/pis
    $pwd
    /home/kumar/pis
    $rmdir progs
- rmdir : Things to remember
  - You can‘t remove a directory which is not empty
  - You can‘t remove a directory which doesn‘t exist in system.
  - You can‘t remove a directory if you don‘t have permission to do so.

Using . and .. in Relative Pathname:

- User can move from working directory /home/kumar/progs/cprogs to home directory /home/kumar using cd command like
  $pwd
  /home/kumar/progs/cprogs
  $cd /home/kumar
  $pwd
  /home/kumar
- Navigation becomes easy by using common ancestor.
- . (a single dot) - This represents the current directory
- .. (two dots) - This represents the parent directory
- Assume user is currently placed in /home/kumar/progs/cprogs
  $pwd
  /home/kumar/progs/cprogs
  $cd ..
  $pwd
  /home/kumar/progs
- This method is compact and easy when ascending the directory hierarchy. The command cd .. Translates to this ―change your current directory to parent of current directory‖.
- The relative paths can also be used as:
  $pwd

/home/kumar/progs
$cd ../..
$pwd
/home

- The following command copies the file prog1.java present in javaprogs, which is present is parent of current directory to current directory.

$pwd
/home/kumar/progs/cprogs
$cp ../javaprogs/prog1.java .

- Now prog1.java is copied to cprogs under progs directory.

File Related Commands:

1. **cat: Displaying and creating files:**
   - cat command is used to display the contents of a small file on the terminal.

```
$ cat cprogram.c
 # include <stdioh>
void main ()
{
Print("hello:);
}
```

   - As like other files cat accepts more than one filename as arguments

```
$ cat ch1 ch2
        It contains the contents of chapter1
        It contains the contents of chapter2
```

   - In this the contents of the second files are shown immediately after the first file without any header information. So cat concatenates two files- hence its name.

cat options:

- Displaying Nonprinting Characters (-v)
  cat without any option it will display text files. Nonprinting ASCII characters can be displayed with –v option.
- Numbering Lines (-n)
  -n option numbers lines. This numbering option helps programmer in debugging programs.

Using cat to create a file

- cat is also useful for creating a file. Enter the command cat, followed by > character and the filename.

```
$ cat > new
This is a new file which contains some text, just to
Add some contents to the file new
[ctrl-d]
```

$_

- When the command line is terminated with [Enter], the prompt vanishes. Cat now waits to take input from the user. Enter few lines; press [ctrl-d] to signify the end of input to the system To display the file contents of new use file name with cat command.

  $ cat new
  This is a new file which contains some text, just to
  Add some contents to the file new

2. cp: Copying a file

- The cp command copies a file or a group of files. It creates an exact image of the file on the disk with a different name. The syntax takes two filename to be specified in the command line.
- When both are ordinary files, first file is copied to second.

  $ cp csa csb

- If the destination file (csb) doesn‘t exist, it will first be created before copying takes place. If not it will simply be overwritten without any warning from the system.
- Example to show two ways of copying files to the cs directory:

  $ cp ch1 cs/module1 ch1 copied to module1 under cs
  $ cp ch1 cs ch1 retains its name under cs

- cp can also be used with the shorthand notation, .(dot), to signify the current directory as the destination. To copy a file „new“ from /home/user1 to your current directory, use the following command:

  $cp /home/user1/new new *destination is a file*
  $cp /home/user1/new . *destination is the current directory*

- cp command can be used to copy more than one file with a single invocation of the command. In this case the last filename must be a directory.

  Ex: To copy the file ch1,chh2,ch3 to the module , use cp as

  $ cp ch1 ch2 ch3 module

- The files will have the same name in module. If the files are already resident in module, they will be overwritten. In the above diagram module directory should already exist and cp doesn‘t able create a directory.
- UNIX system uses * as a shorthand for multiple filenames.

  Ex:

  $ cp ch* usp Copies all the files beginning with ch

cp options

- Interactive Copying(-i) : The –i option warns the user before overwriting the destination file, If unit 1 exists, cp prompts for response

  $ cp -i ch1 unit1
  $ cp: overwrite unit1 (yes/no)? Y

- A y at this prompt overwrites the file, any other response leaves it uncopied.

Copying directory structure (-R) :

- It performs recursive behavior command can descend a directory and examine all files in its subdirectories.
- -R : behaves recursively to copy an entire directory structure

  $ cp -R usp newusp $ cp -R class newclass

- If the **newclass/newusp** doesn't exist, **cp** creates it along with the associated subdirectories.

3. rm: Deleting Files:

- The rm command deletes one or more files.

  Ex: Following command deletes three files:
  $ rm mod1 mod2 mod3
- Can remove two chapters from usp directory without having to cd

  Ex:
  $rm usp/marks ds/marks
- To remove all file in a directory use *

  $ rm *
- Removes all files from that directory

rm options

- Interactive Deletion (-i) : Ask the user confirmation before removing each file:
  $ rm -i ch1 ch2 rm: remove ch1 (yes/no)? ? y rm: remove ch1 (yes/no)? ? n [Enter]
- A _y' removes the file (ch1) any other response like n or any other key leave the file undeleted.
- Recursive deletion (-r or -R): It performs a recursive search for all directories and files within these subdirectories. At each stage it deletes everything it finds.

  $ rm -r * *#Works as rmdir*

- It deletes all files in the current directory and all its subdirectories.
- Forcing Removal (-f): rm prompts for removal if a file is write-protected. The -f option overrides this minor protection and forces removal.

  rm -rf* #Deletes everything in the current directory and below

4. mv: Renaming the files:

The mv command renames (moves) files. The main two functions are:

- It renames a file(or directory)
- It moves a group of files to different directory

- It doesn't create a copy of the file; it merely renames it. No additional space is consumed on disk during renaming.

  Ex: To rename the file csb as csa we can use the following command

  $ mv csb csa

- If the destination file doesn‘t exist in the current directory, it will be created. Or else it will just rename the specified file in mv command.
- A group of files can be moved to a directory.

  Ex: Moves three files ch1,ch2,ch3 to the directory module

  $ mv ch1 ch2 ch3 module

  - Can also used to rename directory

    $ mv rename newname
  - mv replaces the filename in the existing directory entry with the new name. It doesn't create a copy of the file; it renames it
  - Group of files can be moved to a directory
  - mv chp1 chap2 chap3 unix

5. wc: Counting Lines, words, and Characters:

- wc command performs Word counting including counting of lines and characters in a specified file. It takes one or more filename as arguments and displays a four columnar output.

  $ wc ofile 4 20 97 ofile

  - Line: Any group of characters not containing a newline
  - Word: group of characters not containing a space, tab or newline
  - Character: smallest unit of information, and includes a space, tab and newline
  - wc offers 3 options to make a specific count. –l option counts only number of lines, - w and –c options count words and characters, respectively.

    $ wc -l ofile 4 ofile $ wc -w ofile 20 ofile
  - Multiple filenames, wc produces a line for each file, as well as a total count.

    $ wc -c ofile file 97 ofile 15 file 112 total

6. od: Displaying Data in Octal

- od command displays the contents of executable files in a ASCII octal value.

  $ more ofile this file is an example for od command ^d used as an interrupt key
- -b option displays this value for each character separately.
- Each line displays 16 bytes of data in octal, preceded by the offset in the file of the first byte in the line.

$ od –b file

```
0000000 164 150 151 163 040 146 151 154 145 040 151 163 040 141 156 040 0000020 145
170 141 155 160 154 145 040 146 157 162 040 157 144 040 143 0000040 157 155 155 141
156 144 012 136 144 040 165 163 145 144 040 141 0000060 163 040 141 156 040 151 156
164 145 162 162 165 160 164 040 153 0000100 145 171
```

-c character option

- Now it shows the printable characters and its corresponding ASCII octal representation

  $ od –bc file

```
od -bc ofile
0000000 164 150 151 163 040 146 151 154 145 040 151 163 040 141 156 040
          T   h   i   s       f   i   l   e       i   s       a   n
0000020 145 170 141 155 160 154 145 040 146 157 162 040 157 144 040 143
          e   x   a   m   p   l   e       f   o   r       o   d       c
0000040 157 155 155 141 156 144 012 136 144 040 165 163 145 144 040 141
          o   m   m   a   n   d  \n   ^   d       u   s   e   d       a
0000060 163 040 141 156 040 151 156 164 145 162 162 165 160 164 040 153
          s       a   n       i   n   t   e   r   r   u   p   t       k
0000100 145 171
          e   y
```

- Some of the representation:
  - The tab character, [ctrl-i], is shown as \t and the octal vlaue 011
  - The bell character , [ctrl-g] is shown as 007, some system show it as \a
  - The form feed character,[ctrl-l], is shown as \f and 014
  - The LF character, [ctrl-j], is shown as \n and 012
  - Od makes the newline character visible too.