6/08/2020 UNIX PROGRAMMING:- ①

Smitha.N
Asst. Professor
CMRIT.

Module-1 :-

Introduction :-

* **Operating System :-**

An OS is the software that manages the computer hardware and Provides a Convenient and safe environment for running programs.

* **UNIX Operating System :-**

→ Ken Thompson & Dennis Ritchie Developed in 1960's @ AT&T lab.

→ 1973, it was rewritten in C.

→ User can interact with command interpreter called the Shell.

→ UNIX is Security-conscious, can be accessed by only those who maintain an account on the computer-System.

→ Eg :- commands are
tput clear, date, cal, who, wc, exit.

→ TCP/IP protocols used by the internet for communication.

→ Richard Stallman and Linus Torvalds developed Linux. GNU (General Public License)

1) UNIX components / Architecture :-

* UNIX is powerful OS.
* Division of labor : kernel & shell.
  → Terrific idea is division of labor between two agencies, the kernel & the shell.
  → Kernel :-
    * kernel interacts with the machine's hardware
    * Core of OS.
    * a collection of routines mostly written in C.
    * loaded into memory when system is booted and communicates with h/w.
    * User pgms that need to access the h/w uses services of the kernel. when performs on user's behalf.
    * User pgms access kernel through set of functions called "System calls".
    * manages system memory.
    * Schedules process, decides priorities.
  → Shell :-
    * Command interpreter, translates commands into action,
    * Interface between user & kernel,
    * several shells can exists inaction one for each user logged in.

* Types of shell.
  - → Bourne shell (sh)
  - → C shell (csh)
  - → Korn shell (ksh) (b) bash shell (bash)
  - → $ echo $SHELL

  Gives the type of shell running for:

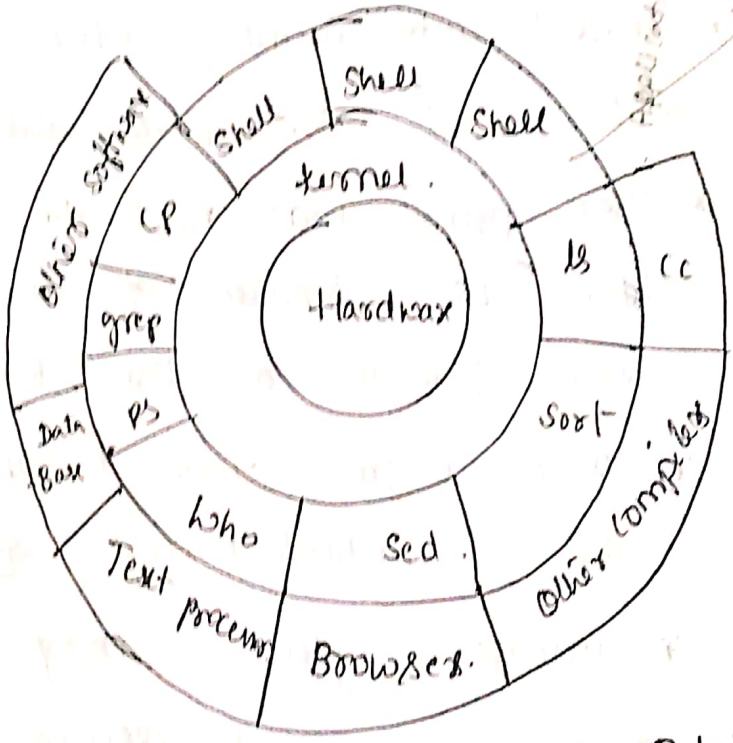

Fig:- The kernel-shell Relationship.

The File and Process :-

* Two simple entities support the UNIX system
  - → The file → have space
  - → The Process → have life.

* File is just an array of bytes, UNIX have different tools for text manipulation.

* When file is executed it's called process. When is just like living organism which can have parents, children and grandchildren.

The System calls:-

→ Used to communicate with the kernel.

→ System calls are described in POSIX specification.

→ Eg:- write(), write to a file.

---

② Features of UNIX:-

UNIX has all the features of an os plus. possesses features unique to itself. listed below.

ⓐ UNIX: A multiuser system:-

→ UNIX is multiprogramming in 2 ways.

* Multiple user can run separate jobs.

* A single user can also run multiple jobs.

→ UNIX, user share resources ∴ UNIX is a multi user System.

ⓑ UNIX: A multitasking System too:-

→ Eg:- User can edit a file and print another one on the printer.

→ kernel is designed to handle a users

multiple needs.

→ One job running in the foreground and next run in background. and also switch can happen.

(c) The building-Block Approach :-

→ small is beautiful technique used.

→ Few hundred commands each performs one simple job.

→ Eg :- ls → lists files & dir in current dir.

wc → counts, no of lines, no of words, no of characters.

ls | wc → gives count of ls o/p.

In Unix we dont a command to do the above job, instead different small commands are interconnected for specialized function.

(d) The UNIX Toolkit :-

→ Kernel alone is not powerfull

→ UNIX host applications

→ like general-purpose tools, text manipulation, compilers, interpreter, n/w application etc.

(e) Pattern Matching :-

→ UNIX has pattern matching features.

Eg:- * → means many.

↳ metacharacter not a special character.

→ UNIX also uses regular expression for pattern matching.

(f) Programming Facility :-

→ The UNIX shell is also a programming language.

→ It has all necessary constructs like control structures, loops & variables.

→ These features are used to design shell scripts. (Pgm used to invoke UNIX commands).

* S/m functions can be automated & controlled.

(g) Documentation :-

→ Online help facility available is man command. Its important reference for other commands.

→ Vast resources available on the internet where we can post queries related to UNIX.

(3) POSIX And the Single UNIX Specification :-

→ Portable Operating System Interface for Computer Environments (POSIX) were developed at the behest of the Institution of Electrical & Electronics Engineers (IEEE).

→ POSIX is OS based on UNIX. 2 Std of POSIX are POSIX.1 and POSIX.2.

→ POSIX.1 → Capplication program interface - Sys calls.

→ POSIX.2 → deals with shell and utilities.

→ X/Open & IEEE unification resulted in Single UNIX Specification, version 3 (SUSV3).

General features of UNIX commands / command Structure

→ It could be simple @ complex

→ UNIX command may @ may not have arguments. Arguments can be an option @ a filename.

→ Syntax :-

command option(s) filename(s)

→ options modify the way in which command works. Its a single letter prefixed with a dash (-) @ multiple could be combined in a single dash (-ab).

→ argument filename is the name of a file that we want to use.

Eg:- $ ls → List filenames/dir

$ ls -l → list entire information for each file

$ ls -l chap1 → list information about a particular file by name chap1.

Basic UNIX commands:-

① echo :-

* Display a message on the terminal. ② issues prompt for taking input from user.

Eg:- echo "hi" → display message.

echo $shell → evaluates shell variables.

* Different escape sequences used by echo & printf.

\a	bell
\b	backspace.
\c	No newline (cursor in same line)
\f	formfeed.
\n	newline.
\t	tab.
\v	vertical tab.
\\	backslash. etc.

* Eg:- $echo " Enter filename : \c"

Enter filename : $_ (prompt & cursor in same line)

(2) printf :

→ used instead of echo.

→ only Bash shell has printf (built-in).

→ printf also uses escape sequences like echo.

→ %s format string acts as a placeholder for strings. like Unix has many.

① %s → string.

② %.30s → prints only 30 characters.

③ %d → decimal integer

④ %6d → as above by only 6 characters

⑤ %o → octal integer.

⑥ %x → hexadecimal integer

⑦ %f → floatary point number.

→ Eg:- $ printf "My current shell is %s\n" $SHELL

① my current shell is /usr/bin/bash.

② $ printf "The value of 255 is %o in octal\n"

The value of 255 is 377 in octal.

---

(3) who : who are the users?

→ UNIX is multiple users.

→ To know multiple user. who command used.

Eg:- $ who

kumar	console	may 9	09:31	(:0)
vipul	pts/4	may 9	09:31	(:0.0)

(4) Passwd :-

→ Used to change the user account passwords.

→ Syntax :-

passwd [options] [username]

Example :-

$passwd ↵

changing password

current password :

Enter new password :

Retype " ——— :

passwd : password updated successfully.

---

(5) date :-

→ Display the system date

→ $date

Fri Aug 7 1:50:40 IST 2020

→ format specified by + symbol followed by %operator

Eg:- $date +%m

8

$date +%h

Aug

$date +"%h %m"

Aug 08

format specifiers.

d - day of the month (1 to 31)

y - the last 2 digits of the year

H, M, S → hr, min, sec

D → Date in the format mm/dd/yy

T → Time in the format hh:mm:ss

⑥. cal :-

→ invoked to see calendar of an month @ a complete year.

→ syntax :-

cal [[month] year]

optional.

→ cal

Aug 2005

Su	mo	Tu	We	Th	Fr	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			

→ cal 03 2006

March 2006

Calendar displayed.

→ cal 2003 | more.

i/p.

as one year Calendar doesn't fit in one screen.

---

⑦. printf :-

→ only available in bash shell.

→ Syntax :-

$ print "No filename entered \n"

No filename entered

$.

$ printf "My current shell is %.s\n" $SHELL

My current shell is /usr/bin/bash.

%s – string.
%.30s – prints first 30 char
%d – decimal integer
%6d → 6 decimal integer
%o → Octal integer.

(12) %x – hexadecimal
%f – floating point number

Eg:- $ printf "The value is %o" 255
The value is 377.

---

(8) ls :-

→ ls command used to list the names of the files / directories in current directory.

→ $ ls
chap01
chap02
helpdir

→ $ ls -l

-rw-r--r-- 1 kumar users 5609 Apr 23 9:30 chap01
-rw-r--r-- 1 kumar users 26129 Apr 24 11:55 chap02
-rw-r--r-- 1 kumar users 27385 Apr 15 1230 chap03

(file owner, group owner, file size, filename)
(regular file, user, group, others, no of hardlinks, modified)

→ above command used with option -l which shall give detailed discription about the files and directories stored in the current director. where read, write, execute permission, user name file size, creation month, time, date, and name of file is displayed.

Combining commands:-

→ We can combine 2 or more commands in Unix for execution. Each command has to be separated from each other by ; (semicolon)

→ Eg:- ① wc note ; ls -l note.

② (wc note ; ls -l note) > newlist

o/p of 2 commands are redirected to newlist.

→ ; known as metacharacter.

Meaning of Internal and External Commands:-

UNIX commands are classified into two types.

① Internal commands.
Eg:- cd,

② External Commands. Eg:- ls, cat.

① Internal commands:-

→ built into the shell.
→ Execution speed is really high
→ No process are created for executing these commands.
→ current directory simply gets changed on executing it

→ by typing help in the command prompt we can list all the inbuilt commands.

→ by typing 'type command' we can check whether command is inbuilt @ not.

Eg:- $type cd
cd is a shell builtin. } type command clearly specifies it as built-in(internal)

$type cat
cat is /bin/cat } for external it gives path of the command from where it is executed.

② External commands :-

→ External commands are not built into the shell.
→ They are executables present in a separate file.
→ When external commands has to be executed, new process is created
→ Eg:- cat command, which is in /usr/bin.
The executable /usr/bin/cat gets executed.

The type command : knowing the type of a command and locating it :-

The type command is used to find out whether it is built-in @ external binary file.

Syntax :- type [options] command names

Example :- $ type cd.
cd is a shell builtin.

→ options :-

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

Eg:- type -a pwd
pwd is a shell builtin
pwd is /bin/pwd

type pwd
pwd is a shell builtin.

-t :- This option will display a single word as an output.

* . alias : if command is a shell alias
* . keyword : " —— is a shell reserved word
* . builtin :- " —— is a shell builtin
* . function :- " —— is a shell function.
* . file : " —— is a disk file.

Eg:- type -t pwd.
builtin
type -t cp
file

type -t while
keyword
type -t ls
alias.

-p:- Returns name of the disk file which would be executed by the shell. else nothing.

Eg:- type -p bash
/bin/bash.

The root login :-

→ Root is the superuser account in Unix.
→ account for administrative purpose.
→ has highest access rights on the system.
→ User id is 0.

Becoming the Super User : Su command :-

→ Two ways to become super user
  * . log in as root directly.
  * . execute the command su while logged into another user account.
  * . su command may be used to change one's current account to that of a different user after entering the proper password.
  * . It takes user name corresponding to the desired account as its argument. root is default if no argument is provided.
  * . exit @ ctrl-D to exit root account.

UNIX files:

→ The file is a container for storing information.

→ UNIX treats directories, files, physical devices like hard disk, memory, CD-ROM, printers as files. Shell and kernel are also files.

Naming files:

→ file name can consist of upto 255 characters except / and NULL character.

→ Recommended
  * Alphabetic characters & numerals
  * The period (.), hyphen (-) and underscore (_)

→ UNIX imposes no file extension rule, like S.sh for shell script but application imposes
  Eg:- C compiler → pgm.c.

→ file name can have any number of dots
  Eg:- a.b.c.d.

→ UNIX is case sensitive.
  Eg:- Chapo1, Chapo1, CHapo1 are different

Basic file types / categories :-

There are three types.

1. Ordinary files
2. Directory files
3. Device files.

(1) Ordinary (Regular) files :-

→ common file type.

→ two types

* Text file
* Binary file.

→ Text file :-

* contains only printable characters.
* we can view and make sense out of them.
* Eg:- C, Java, phython programs.
* Text file consists of lines of character ended with newline character called linefeed (LF).

→ Binary file :-

* consists of both printable and unprintable characters.
* Eg:- object code, executables produced from compiling C programs, picture, sound.

② Directory File :-

→ contains no data, keeps some details of the file and subdirectories

→ Directory consists of entries with 2 components

* . The filename.

* . A unique identification number for the file @ directory (called inode no)

③ Device file :-

→ printing files, installing software from CD-ROM @ backup files to tape. All these activities performed by reading @ writing the file representing the device.

→ Device filenames are generally found in /dev directory.

The Parent - child Relationship :- (x)

→ All the files in UNIX are related to one another.

→ The file system in UNIX is a collection of files organized in a hierarchical structure.

→ UNIX file system has a top, which serves as the reference point for all files. This top called root and is represented by /. root is a directory.

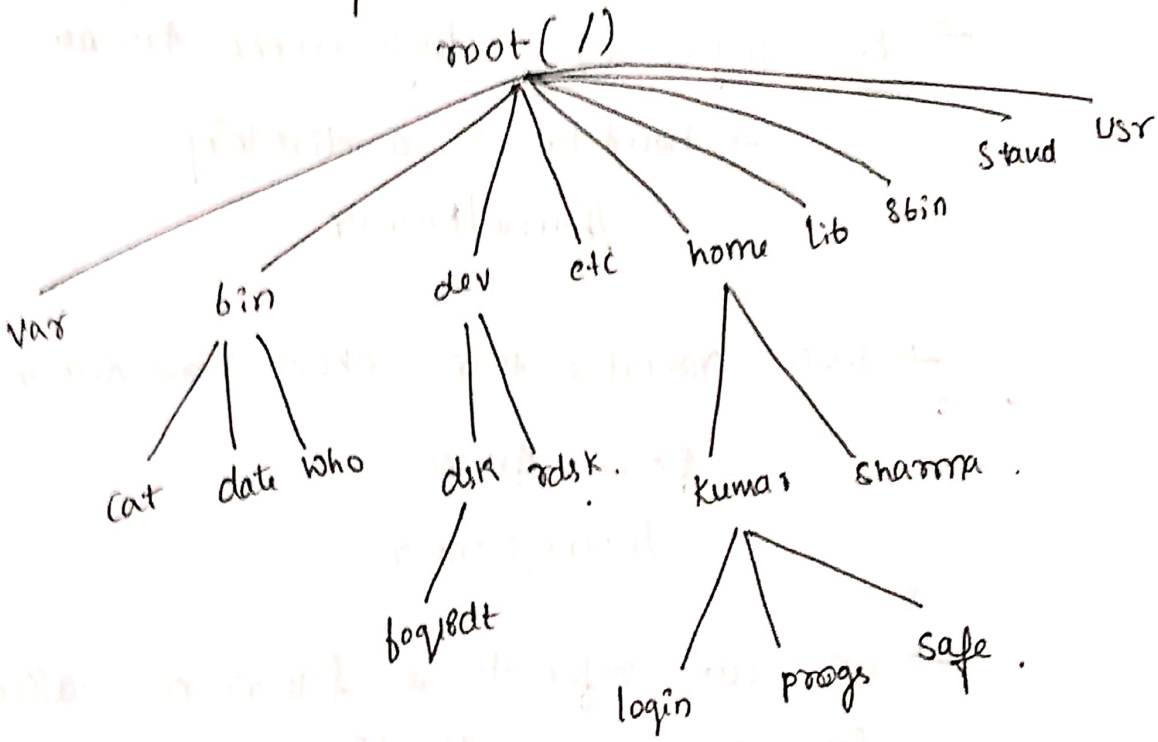

Fig:- The UNIX File System Tree.

→ root directory has a no of sub-directories.

→ sub directories can have more sub-directories and other files under them.

→ Eg:- home directory is the parent of kumar
/ is parent of home. and grandparent of

The HOME variable : The HOME DIRECTORY :-

→ Once we login to the System, UNIX automatically place us in a directory called the "home directory".

→ This directory is created by system when user account is opened.

→ Eg:- login using login name kumar
→ land up in a directory
/home/kumar.

→ Shell variable HOME shows home directory.

$echo $HOME
/home/kumar.

→ we can refer to a file name called foo in home directory as

$HOME/foo. @ ~/foo.
↓
refers to home directory

Reaching required files – The PATH variable :-

→ It is an environment variable, specifies a set of directories where executable programs are located.

→ All the commands like ls, mv, mkdir etc reside in /usr/bin directory.

→ Shell checks path, where to look which is part of an environment variables. called $PATH

→ echo $PATH → Specifies list of one or more directory name separated by colon(:)

/usr/username/bin : /usr/local/bin : /usr/bin : .

→ Set your path.

$PATH = $PATH : /home/Smitha

Now all the executables in home directory could be directory executed by shell i.e $a.out instead of ./a.out.

→ For setting permanently path.

.bashrc → bash , .cshrc → C shell

$ gedit .bashrc

type @ the end.

PATH = $PATH : /home/Smitha

Save it.

and open new terminal and check echo $PATH Shell be set Permanently.

Relative and absolute Pathnames :-

Absolute Pathname :-

* Eg:- `cat login.sql` → file presumed to exist in current directory.

If login.sql exists in /home/kumar and if ur placed in /usr bin use

```
cat /home/kumar/login.sql
```

* In the above Eg:- /home/kumar/login.sql is called an absolute pathname where first character is / (root).

* Absolute pathname of files are unique.

Relative Pathnames :-

* Relative pathname doesn't begin with / (root). like absolute pathname.

* Eg:- cd progs → move to dir progs

cat login.sql → display file login.sql.

if scripts is another dir inside progs then we need not use absolute path.

cd progs/scripts. → relative pathname.

The dot (.) and double dots (..) notations to represent present and parent directories and their usage in relative pathnames.

→ A shortcut called Relative Pathname uses either current (or) parent directory as reference and specifies the path relative to it.

→ . (a singe dot) – Represents the current directory.

.. (two dot) – Represents the parent directory.

→ Eg:- $ pwd

/home/kumar/progs/data/text

$ cd ..

$ Pwd

/home/kumar/progs/data

Moves one level up.

→ cd .. → "change your directory to the Parent of the current directory".

Eg:- $ cd ../..

$ pwd

/home/kumar

Moves 2 level up.

→ . (represent current directory)

→ Eg:- ① $cp ../sharma/.profile .

copy command filename

copies .profile to current directory(.)

② cp /home/sharma/abc .

Directory commands :-

① pwd :- checking your current directory.

→ User can move from one directory to another directory.

→ At any point of time, user shall be located in only one directory called current directory.

→ pwd command (print working directory) used to know current directory.

Eg:-
```
$pwd
/home/kumar
```

---

② cd :- Changing the current directory :-

→ we can change directory using cd command by specifying directory name as argument.

Eg:-
```
$pwd
/home/kumar
```

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

→ last example, was changing to subdirectory under current directory.

→ If we want to switch to other directory like /bin

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

→ cd can be used without argument, reverts to home directory.

```
Eg:- $ pwd
     /home/kumar/progs.
     $cd
     $pwd
     /home/kumar.

Eg:- $pwd
     /home/sharma.
     $ cd
     $pwd
     /home/kumar.
```

Sudha. N
CSE.

(3) mkdir :- Making directories

* To create directory under current directory.

```
$ mkdir patch
```
→ dir name.

* To create multiple subdirectories with one mkdir command.

```
$mkdir patch obs doc
```
3 sub-dir name.

* To create directory tree.

```
$mkdir pis pis/progs pis/data.
```

* Reason for failure to create directory
  - → dir by that name may already exist.
  - → there may be ordinary file by that name in the current directory.
  - → permission set dont allow to create new directory.

(4) rmdir : Removing Directories :-

→ used to remove directories

→ Eg:- `rmdir pis` [directory must be empty]

→ `rmdir $pis/data pis/progs pis`

to remove directory trees.

→ Rules:-

*. you can't delete a directory with rmdir unless it is empty

*. you can't remove a subdirectory unless you are placed in a directory which is hierarchically above the one you have chosen to remove.

Eg:- $cd progs
$ pwd
/home/kumar/pis/progs.
$ rmdir /home/kumar/pis/progs.
rmdir: Dir doesn't exist

$ cd /home/kumar/pis
$ pwd
/home/kumar/pis
$rmdir progs

File related Commands :-

① **cat** :- used to display the contents of a small file on the terminal.

```
$cat dept.c
Hello CM !!

$ cat chap01.c chap02.c
Hello CM1 !
Hello CM2 !!
```

*. **Cat options (-v and -n) :-**

→ **Displaying Nonprinting characters (-v) :-**

→ **Numbering lines (-n) :-**

* **Cat used to create a File :-**

```
$ cat > foo
Hello CM
how r u !!
[ctrl-D]
$-
```

after pressing enter cat now waits to take input from the user after entering lines press ctrl-D to signify the end of i/p and all the typed i/p shall goes to the filename foo.

```
$ cat foo
Hello CM
how r u !!
```

display foo content on terminal.

⑧ mv : Renaming files ::

→ Used to rename / move files

→ Two distinct functions.

* Renames a file / dir
* move a group of files to a different directory.

→ No additional space is consumed on disk during renaming.

Eg:- mv chapo1 mano1.

Chapo1 renamed as mano1.

→ Group of files can be moved to a directory.

Eg:- mv ch1 ch2 ch3 progs

files → dir progs.

moved to

→ mv could be used to rename directory

Eg:- mv P1 P2

③ rm :- deleting files :-

→ used to delete one @ more files.

→ Eg:- `rm ch1 ch2 ch3` (removes 3 file ch1 ch2, ch3)

→ without changing directory we can remove file from other directory.

Eg:- `rm progs/chapo1 progs/chap2`

@ `rm progs/chapo[12]`

→ To remove all files in a directory

```
$ rm *
```

// use very carefully because UNIX Sbm shell not ask whether we are sure to delete all the files.

→ rm options :-

*. Interactive Deletion (-i) :- ask user for confirmation before removing each file.

```
$rm -i ch1 ch2 ch3
rm: remove ch1 (yes/no) ? ? Y  →delete ch1
rm: remove ch2 (yes/no) ? ? n
rm: remove ch3 (yes/no) ? [Enter]   } if not delete
```

Y → removes the file.
Any other response leaves the file undeleted.

*. Recursive Deletion (-r or -R) :-

→ rm performs a tree walk → a thorough recursive search for all subdirectories and files within these subdirectories. With -r rm deletes directories as well.

Eg:- rm -r * . // deletes all files in the current directory and all its subdirectories

*. Forcing Removal (-f) :-

→ some files shall be write protected if we wish to forcibly remove we have to use -f

Eg:- rm -f ch1

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

Smitha.N.
CSE

5. cp :- Copying A file :-

→ used to copy a file @ a group of files.

→ creates exact image of the file on disk with different name.

Eg:- `cp ch1 ch2`

here ch1 → source file

ch2 → destination file.

(if it doesn't exists, it creats first Befor copying takes place).

If ch2 exists, then it shall be overwritten by ch1 content.

→
```
$ cp chapo1 progs/unit1
  cp chapo1 progs
```

chapo1 copied to unit1 under progs

chapo1 retains its name under progs.

→
```
$ cp /home/sharma/abc abc
$cp /home/sharma/abc .
```

Destination is a file.

Destination is the current directory.

→ cppy more than one file @ once.

```
$ cp chapo1 chapo2 chapo3 progs.
```

dir name.

@ $cp chap* progs.

→ cp options :-

* Interactive copying (-i)

warns user before overwriting the destination file.

```
$cp -i chapo1 unit1.
cp : overwrite unit1 (yes/no)? y.
```

y → overwrite, any other response leaves it uncopied.

* copying Directory Structure (-R) :-

→ command can descend a directory and examine all files in its subdirectories.

Eg:- cp -R prog newprogs

(5) wc : counting lines, words and characters :-

→ command used to count lines and characters.

→ It shall take one / more filenames as arguments and displays a four-columnar output.

→ Eg:-

```
$cat  myfile
   Hi  GM.
$wc   myfile
   1   2   5   myfile
```

1 line, 2 words, 5 character in filename myfile.

→ $wc -l myfile
1 myfile Number of lines

→ $ wc -w myfile
2 myfile Number of words.

→ $ wc -c myfile
5 myfile Number of characters

→ wc could be used with multiple filenames.

```
$wc  myfile  chap01
  1   02   5   myfile
  2   04   16  chap01
```

(6). od :- Displaying Data in octal :-

→ Many executable files contains nonprintable characters and most unix commands don't display them properly.

→ od command makes these characters visible by displaying the ASCII octal value of its input

→ -b option displays this value for each character separately.

→ Eg:- od -b odfile

```
0000000  127  150  151  164  145  040  163  160  141  143  145
                                                    040  151  156
0000020
```

Each line displays 16 bytes of data in octal, preceded by the offset (position) in the file of the first byte in the line.

Eg:- od -bc odfile

```
0000000  127  150  151  164  145  163  160  141  143  145  040  151  156  143
           w    h    i    t    e         s    p    a    c    e         i    n
```

here first line represents octal value of each byte and second line represents printable & non escape sequence

w → octal value is 127

Hidden files:

→ files in filesystem utilities do not display by default when showing a directory listing. Basically used for storing user preferences @ state of a utility.

→ In UNIX OS system, there types of files are commonly called dot file @ dotfile.

→ `$ ls -a` used to display hidden files which starts with a dot character.

Eg:- .config , .bashrc

→ `$ ls -al` // for long listing.

Standard directories in UNIX:

→ System directories are located directly below the root directory, are essential to the startup & continous operation of the System.

→ /bin :- contains progms used to managing the System.
Eg:- date, ls, cp . all the executables

→ /dev :- contains system device files. Provides an interface to a particular device.
Eg disk drives, tape drives @ CD-ROM drives

→ /etc :- System specific configuration files, files essential for system startup are located in the /etc.

→ /home :- all the users of the s/m are stored.

Eg:- /home/smitha , smitha's files and programs are stored in this directory.

→ /opt :- contains software files that not installed when the OS is installed.

Usually contains third-party software vendors.

→ /sbin :- Programs for administering a system are located in the /sbin directory.

Eg:- f disk (used to partition a disk).

Shutdown (used for stopping a system)

→ /tmp :- used for holding temporary files. important files should not be kept in this directory.

→ /usr :- contains files and programs related to the users of a system. Typically read-only, could be shared with other computer s/m on a network.

→ /var :- files with varying content are stored in the /var directory.

Eg:- System log files.

—— * ——