UNIX-:

The unix operating system is set of program that act as a link between the computer and the user.

The computer program that allocate the system resources and coordinate all details of the computer's internals is called the operating system @ the kernel

Users communicate with the kernel through a program know as a the shell. The shell is a command line interpreter, it translates commands entered by the user and convert them into a language i.e understood by kernel

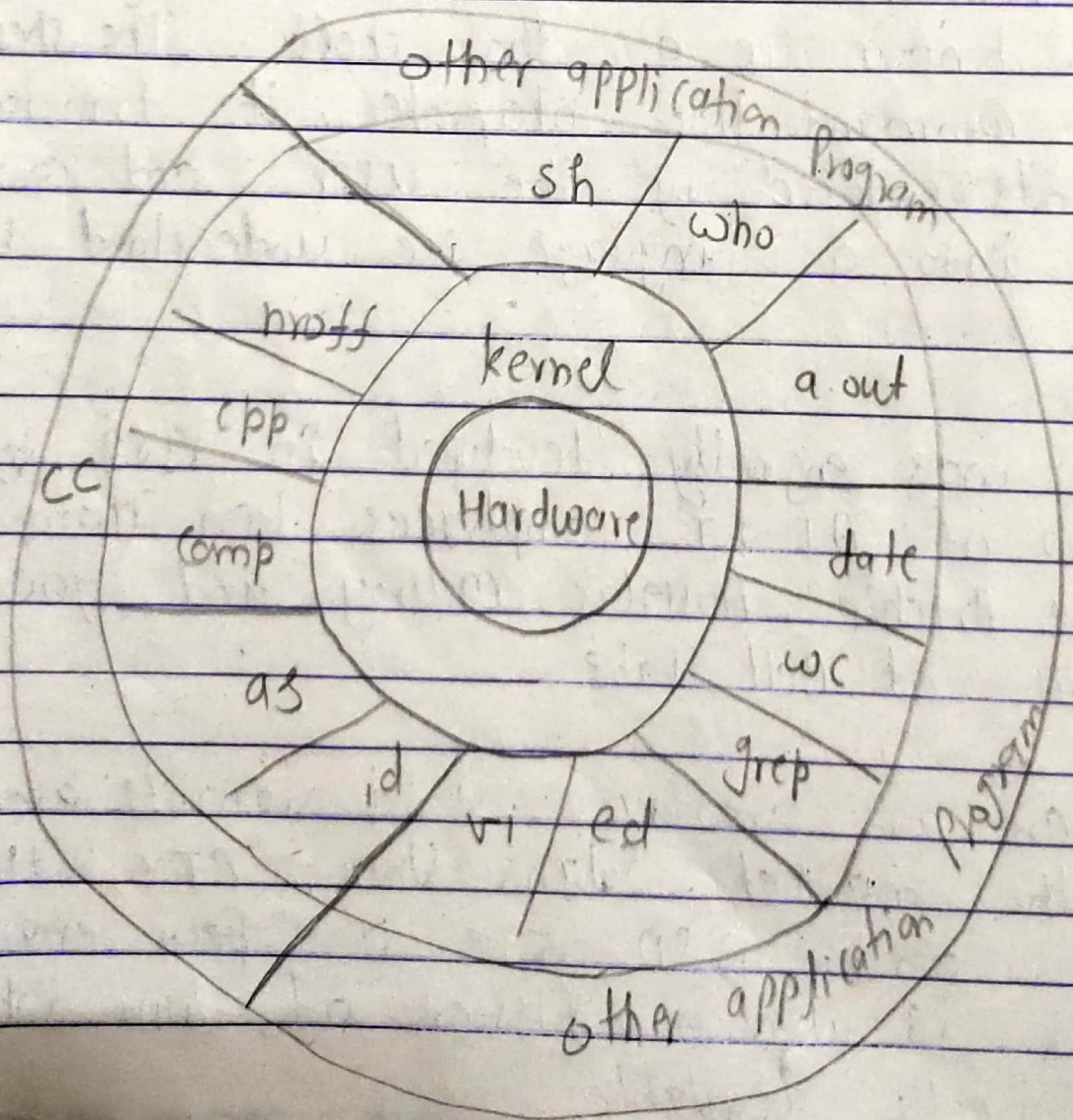
-> Unix was originally developed in 1969 by a group of AT&T employees ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossana at Bell Labs.

-> There are various unix variants available in the market. Solan's Unix, AIX, HP Unix and BSD are a few examples. Linux is also a flavor of Unix which is freely available.

→ Several people can use a unix computer at the same time; hence unix is called a multiuser system.

→ A user can also run multiple programs at the same time; hence Unix is a multitasking environment.

Unix Architecture-:



Kernel-:

The kernel is a part of the O.S. It interacts directly with the hardware of the computer through a device i.e built into ~~it~~ the kernel.

→ The kernel takes responsiblity for deciding at any time which of the many running programs should be allocated to processer.

→ The kernel is responsible for deciding which memory each process can use, and determining what to do when not enough is available

→ The kernel allocates requests from applications to perform I/O to an appropriate device and provides convenient methods for using the device

→ kernel also provide methods for synchronization and communication b/w processes (called inter-process communication IPC)

Main fn-: Memory Management
Controlling access to the computer
maintaining the file system

Handling interrupts
Handling errors
Performing input & output services
Allocate the resources of the computer among users

Shell -:

It is a software program and it acts as a mediator b/w the kernel and the user. It reads the command and then interpret them and sends a request to execute a program. So shell is also called a command interpreter.

It contains nearly 100 system calls. System calls tell the kernel to carry out various tasks for the program.

Opening a file
Writing a file
Obtaining information about a file
Executing programs
Terminating a process
Changing the priority of process
Getting the time & date

Hardware-:

The hardware includes all the parts of a computer including clocks, timers, devices, parts etc. in the Unix OS architecture

Unix Command & libraries-:

This layer of Unix architecture includes user-written applications, using shell programming language and libraries of Unix.

feature of Unix-:

→ It is a multi-user system where the same resources can be shared by different users

→ It provides multi-tasking, where in each user can execute many processes at the same time.

→ It was first OS that was written in HLL (C language). This made it easy to port to other machine with minimum adaptations

→ It provides a hierarchical file structure which allows easier access and maintenance of data

→ Unix has buit-in networking fh so that different user can easily exchange information.

→ Unix functionality can be extended through user programs built on a standard programming interface

Portability -:

System hides the machine architecture from the user, making it easier to write the application that can run.

Unix Shell -:

Unix has facilities called pipes and filters, which permit the user to create complex program from simpler programs.

Utilities

Background Processing

Software development Tools.

Maturity -:

Unix is a time-tested operating system. It offers a bug free environment and high level of reliability.

Internal and External Commands:-

The unix is command-based i.e things happen because of the commands. All unix commands are seldom more than four characters long

They are grouped into two categories:

Internal Commands:

Internal commands are something which is built into the shell. For the shell built in commands, the execution speed is realy high. It is because no process needs to be spawned for executing it.

for eg -> when using the "cd" command, no process is created. The current directory simply gets changed on executing it.

External Commands:-

External command are not built into the shell. These are executable present in a seprate file. When an external command has to be executed, a new process has to be spawned and the command gets executed.

for eg -> when you execute the "cat" command, which usually is at /usr/bin, the executable /usr/bin/cat gets executed.

Getting list of internall command

If you are using bash shell ~~yo ue~~ you can get the list of shell buit-in commands with "help" command

```
$ help
    // this will list all the shell built in Command //
```

How to find out whether a command is Internal & enternel -:

To know whether it is internal or external with help of "type" command.

Eg => $type cat

cat is /bin/cat

// specifying that cat is external type//

$type cd

cd is a shell built in

General Purpose Utilities:-

File System-:

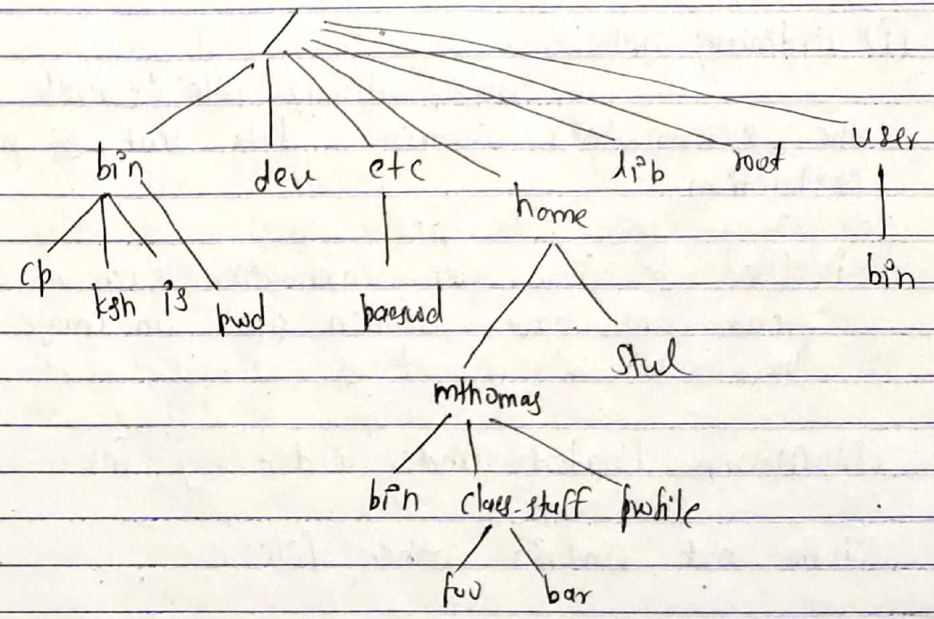
Unix file system is a logical method of organizing and storing large amounts of information in a way that makes it easy to manage.

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 file system.

File in Unix system are organized into multi-level hierarchy structure known as directory tree.

At very top of the file system is a directory called "root" which is represented by a "/". All other file are "descendants" of root.



Types of Unix file System-:

(i) Ordinary files
(ii) Directories
(iii) Special files
(iv) Pipes
(v) Socket
(vi) Symbolic links

(1) Ordinary files :-

An ordinary file is a file on the system that contains data, text, or program instruction.

(i) Used to store your information, such as some text you have written or an image you have drawn

(ii) Always located under a directory file

(iii) Do not contain other files

(iv) In long format output of ls -l, this type of file is specified by the "-" system.

(2) Directories :-

Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.

A directory file contains an entry for every file and subdirectory that it's have.

If you have 10 files in a directory, there will be 10 entries in the directories. Each Entry has two component

(i) The filename
(ii) A unique identification number for the file or directory (called the inode number)

→ Branching points in the hierarchical tree.
→ Used to aganize groups of files.
→ May contain ordinary files, special file or other directories.
→ Never contain "real" information which you would work with (such as text).
→ All files are descendants of the root directory, (named /) located at the top of tree.

This type of file is specified by the "d" symbol.

(3) Special files :-

Used to represent a real physical device such as a printer, tape device or terminal used for input/output operation.

Device file ⊗ special file are used for device input/output on Unix and linux system.

They apear in file system just like an ordinary file ⊙ a directory.

On unix system there are two flavors of specials files for each device.

(i) Character special files
(ii) Block special files

Character special files-:

When a character special file is used for device Input/output, data is transfered one character at a time. This type of accees is called access.

In long femat output of ls -l, character special files are marked by the "c" symbol.

Block Device Special file-:

When a block special file is used for device Input/output, data is transfered in large fixed-size block. This type of accees is called block device accees

In long femat output of ls -l, block special files are marked by the "b" symbol.

→ For terminal device, it's one character at a time.

Inodes-:

An inode number is a uniquely existing number for all files in linux and all Unix type system.

When a file is created on a system, a file name and Inode number is assigned to it.

Generally, to accees a file, a user uses the

Device file & special file are used for device Input/output on Unix and linux system.

They apear in file system just like an ordinary file & a directory.

On unix system there are two flavors of specials files for each device.

(i) Character Special files
(ii) Block special files

Character special files-:

When a character special file is used for device Input/output, data is transfered one character at a time. This type of access is called access.

In long format output of ls -l, character special files are marked by the "c" symbol

Block Device Special file-:

When a block special file is used for device Input/output, data is transfered in large fixed-size block. This type of access is called block device access

In long format output of ls -l, block special files are marked by the "b" symbol.

→ For terminal device, it's one character at a time.

Inodes-:

An inode number is a uniquely existing number for all files in linux and all Unix type system.

When a file is created on a system, a file name and Inode number is assigned to it.

Generally, to access a file, a user uses t

file name but internally file name is first mapped with respective Inode number stored in a table.

Note-: Inode doesn't contain the file name. Reason for this is to maintain hard links for files. When all other information is seperated from the file name then only we can have various file names pointing to same Inode.

Inode Contents-:

An inode is a data structure containing metadata about the file.

following contents are stored in the Inode from a file:

(i) User ID of file
(ii) Group ID of file
(iii) Device ID
(iv) file size
(v) Date of Creation
(vi) Permission
(vii) Owner of the file
(viii) file protecting flag
(ix) Link counter to determine number of hard links

Inode Table-:

The Inode table contains all the Inodes and is created when file system is created.

The df -i command can be used to check how many inodes are free and left unused in the system.

Inode Number-:

Each Inode has a unique number and Inode number can be seen with the help of ls -li command.

Creating file:-
User can create a file in unix using following ways.

(i) Using CAT command
(ii) Using touch command
(iii) Using Echo and print command
(iv) Using different text editors -Vi emac

(i) Cat command in unix:-
→ User can create a new file using 'cat' command in unix.

→ Using shell prompt directly user create a file.

→ Using 'cat' command user will able to open a specific file also.

syntax:-
$ cat > file_name

Eg → $ cat > test.txt
This is first test file.

Press: CTRL + D

After completion of your text you need to press control + d which used to save the file and come out of the prompt.

Note:- for creating file operator (>) is mandatory.

(a) To open created file:-

syntax:- $ cat < file_name
(OR)
$ cat file_name

Note:- for opening the file < operator is not mandatory.

multiple file open:-

$ cat file1 file2 file3 --- file N

(b) To change (OR) append data in file:-

syntax:- $ cat >> file_name
:
[ctrl+d] e (to save data & close file)

Eg => $ cat >> test.txt

This is first unix file and appending

CTRL + D

Note:- To append the data (or) to change the data in file we need to use >> operator

(2) touch command in unix:-

→ touch command is used to create the empty (or) zero byte file.

→ touch command is basically used to create multiple files in linux (or) unix.

→ Using cat command one can create (or) update one file at a time but using touch command user can only create multiple zero byte file but can't update multiple files at a time.

Syntax:-

→ To create single zero byte file

$ touch file_Name

→ To create multiple zero byte files:

$ touch file1 file2 file3 .... fileN

Deleting files :-

→ $ rm :- To delete a file

Eg:- $ rm file_name

→ $ rm -i :- To delete a file with permission / confirmation

Eg:- $ rm -i file_name

→ $rm -f :- It deletes a file forcefully.

→ deleting multiple file:-

$ rm file1 file2 file3 --- fileN

③ Create a file using Echo and printf command-:

Using Echo as well as printf command user can create file in unix.

syntax-:

echo ----- lines of fibe. [\n] --> nameoffile.txt

printf ----- lines of file ---> nameof file.txt

Eg-:

echo this is one line file. > test.txt

printf this is 2 line file \n this is second line > test1.txt

④ Using Text Editor / Creating a file using Vi Editor-:

Using Vi editor user can create a file. To create a file you can use Vi. command to create file.

syntax:

Vi file name.txt

Eg-> Vi test.txt

Copying file in Unix-:

cp stands for copy. This command is used to copy files or group of files or directory.

It creates an exact image of a file on a disk with different file name.

cp command require at least two filenames in its argument.

syntax-:

cp [option] Source Destination

cp [option] Source Destination.

cp [option] source-1 source-2 source-3.. source-n Directory

(i) Two file names-:

If the command contain two file names, then it copy the

content 1st file to 2nd file. If 2nd file not exist, then first it creates one then copy

Syntax-:

cp src_file dest_file

Eg-: $ ls
a.txt

$ cp a.txt b.txt

$ ls
a.txt b.txt

(ii) One or more more arguments-: If the command has one or more arguments (which specify file name) and an argument specifying directory name then this command copies each source file to the destination. (If it's not exist then first create)

syntax-:

cp src_file1 src_file2 src_file3 Dest_directy

Eg-: suppose there is directory name geeksforgeeks having a text file a.txt, b.txt and a directory name new in which we are going to copy all files

$ ls
a.txt b.txt new

$ ls new

$ cp a.txt b.txt new

$ ls new

a.txt b.txt

(iii) Two Directory name-: If the command containes two directory names, cp copies all files of the source directory to the destination directory, creating any files or directories needed.

Eg-: cp -R src_directory Dest_directory.

Options -:
There are many options of cp command.

Suppose a directory name geeksforgeeks contain two files having same content name as a.txt and b.txt.

(i) -i (interactive) -:
With this option system first warns the user before overwriting the destination file

Eg> $ cp -i a.txt b.txt

cp: overwrite 'b.txt' ? y

$ cat b.txt
GfG

(ii) -b (backup) -:
With this option cp command creates the backup of the destination file in the same folder with different name and in different format

Eg> $ ls
a.txt b.txt

$ cp -b a.txt b.txt

$ ls
a.txt b.txt b.txt~

(iii) -f (force) -:
If the system is unable to open destination file for writing operation because the user doesn't have writing permission for this file then by using -f option with cp command, destination file is deleted first and then copying of content is done from source to destination

Eg:- $ ls -l b.txt
-r -xr -xr -x+1 User User 3 Nov 24 05:45 b.t

User, group and others doesn't have writing permission
without -f option, command not executed

$ cp a.txt b.txt

Options -:

There are many options of cp command.

Suppose a directory name geeksforgeeks contain two files having same content name as a.txt and b.txt.

(i) -i (interactive) -:

With this option system first warns the user before overwriting the destination file

Eg> $ cp -i a.txt b.txt

cp: overwrite 'b.txt' ? y

$ cat b.txt
GfG

(ii) -b (backup) -:

With this option cp command creates the backup of the destination file in the same folder with different name and in different format

Eg> $ ls
a.txt b.txt

$ cp -b a.txt b.txt

$ ls
a.txt b.txt b.txt~

(iii) -f (force) -:

If the system is unable to open destination file for writing operation because the user doesn't have writing permission for this file then by using -f option with cp command, destination file is deleted first and then copying of content is done from source to destination

Eg -: $ ls -l b.txt
-r -xr -xr -x+1 User User 3 Nov 24 08:45 b.tx

User, group and others doesn't have writing permission.
without -f option, command not executed.

$ cp a.txt b.txt

cp: can't create regular file 'b.txt' : permission denied.

=> with -f option, command executed successfully

$ cp -f a.txt b.txt

Renaming files :-

① Renaming files with "mv" Command :-

Simple way to rename files and folders is with the mv command. Its primary purpose is moving files and folders, but it can also rename them.

Syntax :-

mv (option) file1.txt files.txt

files.txt (old name)
files.txt (new name)

② rename Command :-

Syntax :-

rename [option] expression replacement file...

(i) rename -s (subtitle) :-

This option rename the files ignoring the symbolic links

rename -s 's/root/new/' sym.png

(ii) rename -v (verbose) :-

This option is used to show which files is being renamed, if there is any

rename -v 's/jpeg/png/*.jpeg

(iii) rename -n (no Action) :-

This option comes into play when the user wants to see only the final change

rename -n 's/png/jpeg/' *.png

(iv) rename -o :-
This option will not going to overwrite the existing files

rename -o 's/jpeg/png/' *.jpeg

Creating links :-

→ A link in UNIX is a pointer to a file

→ like pointers in any programming languages, links in UNIX are pointers pointing to a file or a directory.

→ Creating a link is a kind of shortcut to access a file.

→ There are two types of links.

(i) Soft links or Symbolic links
(ii) Hard links.

These links behave differently when the source of the link is move or removed.

for eg :- If we have a file a.txt, if we create a hard link to the file and then delete the file, we can still access the file using hard link.

But if we create a soft link of the file and then delete the file, we can't access the file through the soft link

Hard link :-

(i) Each hard linked file is assigned the same inode value as the original therefore they reference the same physical location.

(ii) ls -l command shows all the links with link column shows number of links.

(iii) links have actual file contents.

(iv) Removing any link, just reduces the link count, but doesn't affect other links.

(v) We can't create a hard link for a directory to avoid recursive loops.

(vi) If original file is removed then the link will still show the content of the file

(vii) command-: $ ln [orginal filename] [link name]

Soft links-:

(i) A Soft link is similar to the file shortcut feature which is used in Windows OS. Each soft link file contain a seprate Inode value that points to the original file

(ii) ls -l command shows all links with first column value l? and the link points to original file

(iii) Soft links contains the path for original file not content.

(iv) Removing soft link doesn't affect anything but removing original file, the link becomes dagling.

(v) A soft link can link to a directory.

(vi) command-: $ ln -s [original filename] [link name]

Absolute and Relative Pathnames in Unix :-

A path is a unique location to a file or a folder in a file system of an OS.

A path to a file is a combination of / and alpha-numeric characters.

÷ Absolute Path-name :-

An absolute path is defined as the specifying the location of a file or directory from the root directory (/)

To write an absolute path name :-

→ Start at the root directory (/) and work down

→ Write a slash (/) after every directory name (last one is optional)

for eg :- $ cat abc.sql

will work only if the file "abc.sql" exist in your current directory. However, if this file is not present in your working directory and is present somewhere else say in /home/kt, then is command will work only if you will use it like shown below.

→ cat /home/kt/abc.sql

÷ Relative Path :-

Relative path is defined as the path related to the present working directory. It starts at your current directory and never starts with a /

Unix offers a shortcut in the relative pathname that uses either current or parent directory as reference and specifies the path relative to it.

A relative path-name uses one of these cryptic symbols:

→ . (a single dot) - this represents the current directory.

→ .. (two dots) -: this represents the parent directory

Eg.:
If we are currently in directory /home/kt/abc and now you can use .. as an argument to cd to move to the parent directory /home/kt as:

```
$ pwd
/home/kt/abc

$ cd ..
$ pwd
/home/kt
```

Note - /.. → moving one level up.

```
$ pwd
/home/kt/abc

$ cd ../..
$ pwd
/home
```

Comparing file :-

(i) cmp :-
This command is used to compare two files character by character.

syntax :- cmp [options] file1 file2

(ii) comm :-
→ This command is used to compare two sorted files

syntax :- comm [options] file1 file2

→ One set of options allows selection of 'column's' to suppress.

-1 : suppress lines unique to file 1 (column 1)
-2 : suppress lines unique to file2 (column 2)
-3 : suppress lines common to file1 & file2 (column 3)

Eg :- Only show column 3 that contains lines common b/w file1 & file2.
$ comm -12 file1 file2

(iii) diff :-
This command is used to compare two files line by line

syntax :- diff [options] file1 file2

Eg :- $ diff file1 file2

(iv) dircmp :-
This command is used to compare the content of directories

This command works on older versions of Unix. In order to ~~compare~~ compare the directories in the newer version of Unix, we can use diff -r.

syntax :- dircmp [options] dir1 dir2

Eg :- compare content of dir1 & dir2

$ dircmp dir1 dir2

(v) uniq :-

This command is used to filter the repeated lines in a file which are adjacent to each other.

Syntax:- uniq [options] [input [output]]

Eg :- Omit repeated lines which are adjacent to each other in file1 and print the repeated lines only once.

$ uniq file1

Compress and decompress files :-

The most common programs used to compress file in Unix-like systems are.

(i) gzip
(ii) bzip2

:- Compress and decompress files using Gzip program :-

The gzip is a utility to compress and decompress files using lempel-ziv coding (lz 777) algorithm.

(a) Compress files :-

To compress a file named Test.txt, replacing it with a gziped compression version, run:

$ gzip Test.txt

Gzip will replace the original file Test.txt ~~which is a~~ with a gzipped compressed version named Test.txt.gz.

(b) Decompress file :-

To decompress the file Test.txt.gz, replacing it with the original uncompressed version,

$ gzip -d Test.txt.gz

We can also use gunzip to decompress the files.

$ gunzip Test.txt.gz

Compress and decompress file using bzip2 program :-

The bzip2 is very similar to gzip program, but uses different compression algorithm named the Burrows-wheeler block sorting text compression algorithm & Huffman coding.

The files compressed using bzip2 will end with .bz2 extension.

(a) Compress :-

To compress a file using bzip2, replacing it with compressed version,

```
$ gz bzip2 Test.txt
```

⇒ If you don't want to replace the original file, use -c flag and write the output to a new file

```
$ bzip2 -c Test.txt > output.txt.bz2
```

(b) Decompress :-

```
$ bzip2 -d Test.txt.bz2
```

(OR)

```
$ bunzip2 Test.txt.bz2
```

Archining files :-

The most common programs to archive files and directories are

(i) tar
(ii) zip

Archive files using Tar Command :-

Tar is an Unix command which stands for Tape Archive. It is used to combine & store multiple files into a single file. There are four main operating modes in tar utility

(i) c- Create an archive from files (s)

(ii) x- Extract an archive

(iii) r- Append files ~~the~~ to the end on archive

(iv) t- list the contents of the archive

(a) Create a new Archive:-

Let Gen that contains three different types of files.

```
$ ls Gen/
Test.txt  image.png  song.mp3
```

Now create a new archive of the directory Gen.

```
$ tar cf Gen.Tar Gen/
```

c -> Refers create new archive
f -> Refers the file name

Similarly, to create an archive from a set of files in the current working directory

```
$ tar cf archive.tar file1 file2 file3
```

(b) Extract archives:-

To extract an archive in the current directory,

```
$ tar xf Gen.tar
```

: Archive files and Directories using zip Program:-

(a) Creating a new archive:-

To archive a group of files into one, simply do.

```
$ zip file.zip file1 file2 file3
```

Here, we are create an archive named file.zip from file1, file2 & file3. You don't have to use the .zip extension at the end of the file name.

Similarly, to create an archive of a directory

```
$ zip directory.zip /home/
```

(b) Extract Archives:

Extracting archives is as simple as creating archives.

To extract an archive, simply do.

```
$ unzip directory.zip
```