Module 3

Ordinary file handling

[Displaying and creating files (cat), Copying a file (cp), Deleting a file (rm), Renaming/ moving a file (mv), Paging output (more), Printing a file (lp), Knowing file type (file), Line, word and character counting (wc), Comparing files (cmp), Finding common between two files (comm), Displaying file differences (diff), Creating archive file (tar), Compress file (gzip), Uncompress file (gunzip), Archive file (zip), Extract compress file (unzip), Brief idea about effect of cp, rm and mv command on directory.]

Types of Files:

Regular files (-): It contain programs, executable files and text files.
Directory files (d): It is shown in blue colour. It contains list of files.

i) Special files
ii) Block file (b)
iii) Character device file (c)
iv) Named pipe file (p)
v) Symbolic link file (l)
vi) Socket file (s)

Linux File Commands

Command	Description
file	Determines file type.
touch	Used to create a file.
rm	To remove a file.
cp	To copy a file.
mv	To rename or to move a file.
rename	To rename file.

Linux cat command: to display file content
The 'cat' command can be used to display the content of a file.

Syntax:

cat <fileName>

cp command in Linux with examples

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 arguments.

Syntax:

cp [OPTION] Source Destination cp
[OPTION] Source Directory
cp [OPTION] Source-1 Source-2 Source-3 Source-n Directory

First and second syntax is used to copy Source file to Destination file or Directory.
Third syntax is used to copy multiple Sources(files) to Directory.
cp command works on three principal modes of operation and these operations depend upon number and type of arguments passed in cp command :

Two file names : If the command contains two file names, then it copy the contents of 1st file to the 2nd file. If the 2nd file doesn't exist, then first it creates one and content is copied to it. But if it existed then it is simply overwritten without any warning. So be careful when you choose destination file name.

cp Src_file Dest_file

Suppose there is a directory named geeksforgeeks having a text file a.txt. Example:

$ ls
a.txt

$ cp a.txt b.txt

$ ls
a.txt b.txt

One or more arguments : If the command has one or more arguments, specifying file names and following those arguments, an argument specifying directory name then this command

copies each source file to the destination directory with the same name, created if not existed but if already existed then it will be overwritten, so be careful !!.
cp Src_file1 Src_file2 Src_file3 Dest_directory

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

$ ls
a.txt b.txt new

Initially new is empty
$ ls new

$ cp a.txt b.txt new

$ ls new
a.txt b.txt

Note: For this case last argument must be a directory name. For the above command to work, Dest_directory must exist because cp command won't create it.

Two directory names : If the command contains two directory names, cp copies all files of the source directory to the destination directory, creating any files or directories needed. This mode of operation requires an additional option, typically R, to indicate the recursive copying of directories.

cp -R Src_directory Dest_directory
In the above command, cp behavior depend upon whether Dest_directory is exist or not. If the Dest_directory doesn't exist, cp creates it and copies content of Src_directory recursively as it is. But if Dest_directory exists then copy of Src_directory becomes sub-directory under Dest_directory.

Options:

There are many options of cp command, here we will discuss some of the useful options: Suppose a directory named geeksforgeeks contains two files having some content named as a.txt and b.txt. This scenario is useful in understanding the following options.

$ ls geeksforgeeks a.txt
b.txt

$ cat a.txt
GFG

$ cat b.txt

GksfrGks

1. -i(interactive): i stands for Interactive copying. With this option system first warns the user before overwriting the destination file. cp prompts for a response, if you press y then it overwrites the file and with any other option leave it uncopied.

```
$ cp -i a.txt b.txt
cp: overwrite 'b.txt'? y

$ cat b.txt
GFG
```

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

```
$ ls
a.txt  b.txt

$ cp -b a.txt b.txt

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

3. -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 file.

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

User, group and others doesn't have writing permission.

Without -f option, command not executed

```
$ cp a.txt b.txt
cp: cannot create regular file 'b.txt': Permission denied
```

With -f option, command executed successfully

```
$ cp -f a.txt b.txt
```

4. -r or -R: Copying directory structure. With this option cp command shows its recursive behavior by copying the entire directory structure recursively.

Suppose we want to copy gksfrgks directory containing many files, directories into gfg directory(not exist).

```
$ ls gksfrgks/
a.txt  b.txt  b.txt~  Folder1  Folder2
```

Without -r option, error $
cp gksfrgks gfg
cp: -r not specified; omitting directory 'gksfrgks'

With -r, execute successfully
$ cp -r gksfrgks gfg

$ ls gfg/
a.txt b.txt b.txt~ Folder1 Folder2

5. -p(preserve): With -p option cp preserves the following characteristics of each source file in the corresponding destination file: the time of the last data modification and the time of the last access, the ownership (only if it has permissions to do this), and the file permission-bits.
Note: For the preservation of characteristics you must be the root user of the system, otherwise characteristics changes.

$ ls -l a.txt
-rwxr-xr-x+ 1 User User 3 Nov 24 08:13 a.txt

$ cp -p a.txt c.txt

$ ls -l c.txt
-rwxr-xr-x+ 1 User User 3 Nov 24 08:13 c.txt
As we can see above both a.txt and c.txt(created by copying) have same characteristics.

Examples:

Copying using * wildcard: The star wildcard represents anything i.e. all files and directories. Suppose we have many text document in a directory and wants to copy it another directory, it takes lots of time if we copy files 1 by 1 or command becomes too long if specify all these file names as the argument, but by using * wildcard it becomes simple.

Initially Folder1 is empty
$ ls
a.txt b.txt c.txt d.txt e.txt Folder1

$ cp *.txt Folder1

$ ls Folder1
a.txt b.txt c.txt d.txt e.txt

rm command in Linux with examples

rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes

references to objects from the filesystem, where those objects might have had multiple references (for example, a file with two different names). By default, it does not remove directories. This command normally works silently and you should be very careful while running rm command because once you delete the files then you are not able to recover the contents of files and directories. Syntax:

rm [OPTION]... FILE...
Let us consider 5 files having name a.txt, b.txt and so on till e.txt.

$ ls
a.txt b.txt c.txt d.txt e.txt
Removing one file at a time
$ rm a.txt

$ ls
b.txt c.txt d.txt e.txt

Removing more than one file at a time
$ rm b.txt c.txt

$ ls
d.txt e.txt
Note: No output is produced by rm, since it typically only generates messages in the case of an error. Options: 1. -i (Interactive Deletion): Like in cp, the -i option makes the command ask the user for confirmation before removing each file, you have to press y for confirm deletion, any other key leaves the file un-deleted.

$ rm -i d.txt
rm: remove regular empty file 'd.txt'? y

$ ls
e.txt
2. -f (Force Deletion): rm prompts for confirmation removal if a file is write protected. The -f option overrides this minor protection and removes the file forcefully.

$ ls -l total
0
-r--r--r--+ 1 User User 0 Jan 2 22:56 e.txt

$ rm e.txt
rm: remove write-protected regular empty file 'e.txt'? n

$ ls
e.txt

$ rm -f e.txt

$ ls
Note: -f option of rm command will not work for write-protect directories. 3. -r (Recursive Deletion): With -r(or -R) option rm command performs a tree-walk and will delete all the files and sub-directories recursively of the parent directory. At each stage it deletes everything it finds. Normally, rm wouldn't delete the directories but when used with this option, it will delete. Below is the tree of directories and files:

$ ls
A

$ cd A

$ ls
B C

$ ls B
a.txt b.txt

$ ls C
c.txt d.txt
Now, deletion from A directory(as parent directory) will be done as:

$ rm * rm: cannot remove 'B': Is a
directory rm: cannot remove 'C': Is
a directory

$ rm -r *

$ ls
Every directory and file inside A directory is deleted. 4. –version: This option is used to display the version of rm which is currently running on your system.

$ rm --version rm (GNU
coreutils) 8.26
Packaged by Cygwin (8.26-2)
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later .
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Paul Rubin, David MacKenzie, Richard M. Stallman, and
Jim Meyering.

Applications of wc Command Delete file whose name starting with a hyphen symbol (-): To remove a file whose name begins with a dash ("-"), you can specify a double dash ("–") separately before the file name. This extra dash is necessary so that rm does not misinterpret the file name as an option. Let say there is a file name -file.txt, to delete this file write command as:

```
$ ls
-file.txt

$ rm -file.txt rm:
unknown option -- l
Try 'rm ./-file.txt' to remove the file '-file.txt'.
Try 'rm --help' for more information.

$ rm -- -file.txt

$ ls
```

mv command in Linux with examples

mv stands for move. mv is used to move one or more files or directories from one place to another in a file system like UNIX. It has two distinct functions:
(i) It renames a file or folder.
(ii) It moves a group of files to a different directory.
No additional space is consumed on a disk during renaming. This command normally works silently means no prompt for confirmation.

Syntax:

```
mv [Option] source destination
```

Let us consider 4 files having names a.txt, b.txt, and so on till d.txt. To rename the file a.txt to geek.txt(not exist):

```
$ ls
a.txt  b.txt  c.txt  d.txt

$ mv a.txt geek.txt

$ ls
b.txt  c.txt  d.txt  geek.txt
```

If the destination file doesn't exist, it will be created. In the above command mv simply replaces the source filename in the directory with the destination filename(new name). If the destination

file exist, then it will be overwrite and the source file will be deleted. By default, mv doesn't prompt for overwriting the existing file, So be careful !!

Let's try to understand with an example, moving geeks.txt to b.txt(exist):

```
$ ls
b.txt  c.txt  d.txt  geek.txt

$ cat geek.txt
India

$ cat b.txt geeksforgeeks

$ mv geek.txt b.txt

$ ls
b.txt  c.txt  d.txt

$ cat b.txt
India
```

Options:

1. -i (Interactive): Like in cp, the -i option makes the command ask the user for confirmation before moving a file that would overwrite an existing file, you have to press y for confirm moving, any other key leaves the file as it is. This option doesn't work if the file doesn't exist, it simply renames it or move it to new location.

```
$ ls
b.txt  c.txt  d.txt  gk.txt

$ cat gk.txt
India

$ cat b.txt
gk

$ mv -i gk.txt b.txt
mv: overwrite 'b.txt'? y

$ ls
b.txt  c.txt  d.txt

$ cat b.txt
India
```

2. -f (Force): mv prompts for confirmation overwriting the destination file if a file is writeprotected. The -f option overrides this minor protection and overwrites the destination file forcefully and deletes the source file.

```
$ ls
b.txt  c.txt  d.txt  geek.txt

$ cat b.txt
gksfrgks

$ ls -l b.txt
-r--r--r--+ 1 User User 13 Jan  9 13:37 b.txt

$ mv gk.txt b.txt mv: replace 'b.txt', overriding mode
0444 (r--r--r--)? n

$ ls
b.txt  c.txt  d.txt  gk.txt

$ mv -f gk.txt b.txt

$ ls
b.txt  c.txt  d.txt

$ cat b.txt
India
```

3. -n (no-clobber): With -n option, mv prevent an existing file from being overwritten.
In the following example the effect is for nothing to happen as a file would be overwritten.

```
$ ls
b.txt  c.txt  d.txt  gk.txt

$ cat b.txt
gksfrgks

$ mv -n gk.txt b.txt

$ ls
b.txt  c.txt  d.txt  gk.txt

$ cat b.txt
Gksfrgks
```

4. -b(backup): With this option, it is easier to take a backup of an existing file that will be overwritten as a result of the mv command. This will create a backup file with the tilde character(~) appended to it.

```
$ ls
b.txt  c.txt  d.txt  gk.txt

$ mv -b gk.txt b.txt

$ ls
b.txt  b.txt~  c.txt  d.txt
```

5. –version: This option is used to display the version of mv which is currently running on your system.

```
$ mv --version mv (GNU
coreutils) 8.26
Packaged by Cygwin (8.26-2)
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later .
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Mike Parker, David MacKenzie, and Jim Meyering.
```

more command in Linux with Examples

more command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large (For example log files). The more command also allows the user do scroll up and down through the page. The syntax along with options and command is as follows. Another application of more is to use it with some other command after a pipe. When the output is large, we can use more command to see output one by one.

Syntax:

more [-options] [-num] [+/pattern] [+linenum] [file_name]

[-options]: any option that you want to use in order to change the way the file is displayed. Choose any one from the followings: (-d, -l, -f, -p, -c, -s, -u)
[-num]: type the number of lines that you want to display per screen.
[+/pattern]: replace the pattern with any string that you want to find in the text file.
[+linenum]: use the line number from where you want to start displaying the text content.
[file_name]: name of the file containing the text that you want to display on the screen.

While viewing the text file use these controls:

Enter key: to scroll down line by line.
Space bar: To go to the next page. b
key: To go to back one page.

Options:

-d : Use this command in order to help the user to navigate. It displays "[Press space to continue, 'q' to quit.]" and displays "[Press 'h' for instructions.]" when wrong key is pressed. Example:

more -d sample.txt

lp - Unix, Linux Command

NAME
lp: submits files for printing or alters a pending job..

EXAMPLES
Example-1:

To print the /etc/motd file on printer lp0 attached to device dlp0, enter:

lp /etc/motd

Example-2:

To queue the MyFile file and return the job number, enter:

lp myfile

file command in Linux with examples

file command is used to determine the type of a file. .file type may be of human-readable(e.g. 'ASCII text') or MIME type(e.g. 'text/plain; charset=us-ascii'). This command tests each argument in an attempt to categorize it.

It has three sets of tests as follows:

filesystem test: This test is based on the result which returns from a stat system call. The program verifies that if the file is empty, or if it's some sort of special file. This test causes the file type to be printed. magic test: These tests are used to check for files with data in particular

fixed formats. language test: This test search for particular strings which can appear anywhere in the first few blocks of a file.
Syntax:

file [option] [filename]
Example: Command displays the file type

file email.py file
name.jpeg file
Invoice.pdf file
exam.ods
file videosong.mp4

wc command in Linux with examples

wc stands for word count. As the name implies, it is mainly used for counting purpose.

It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.
By default it displays four-columnar output.
First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument. Syntax:

wc [OPTION]... [FILE]...
Let us consider two files having name state.txt and capital.txt containing 5 names of the Indian states and capitals respectively.

$ cat state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh

$ cat capital.txt
Hyderabad
Itanagar
Dispur
Patna
Raipur
Passing only one file name in the argument.

$ wc state.txt

5 7 58 state.txt

OR

$ wc capital.txt

5 5 39 capital.txt

Passing more than one file name in the argument.

$ wc state.txt capital.txt

5 7 58 state.txt

5 5 39 capital.txt

10 12 97 total

Note : When more than file name is specified in argument then command will display fourcolumnar output for all individual files plus one extra row displaying total number of lines, words and characters of all the files specified in argument, followed by keyword total. Options:
1. -l: This option prints the number of lines present in a file. With this option wc command displays two-columnar output, 1st column shows number of lines present in a file and 2nd itself represent the file name.

With one file name

$ wc -l state.txt

5 state.txt

With more than one file name

$ wc -l state.txt capital.txt

5 state.txt

5 capital.txt

10 total

cmp Command in Linux with examples

cmp command in Linux/UNIX is used to compare the two files byte by byte and helps you to find out whether the two files are identical or not.

When cmp is used for comparison between two files, it reports the location of the first mismatch to the screen if difference is found and if no difference is found i.e the files compared are identical.
cmp displays no message and simply returns the prompt if the files compared are identical.
Syntax:
cmp [OPTION]... FILE1 [FILE2 [SKIP1 [SKIP2]]]

SKIP1 ,SKIP2 & OPTION are optional and
FILE1 & FILE2 refer to the filenames .
The syntax of cmp command is quite simple to understand. If we are comparing two files then obviously we will need their names as arguments (i.e as FILE1 & FILE2 in syntax). In addition to this, the optional SKIP1 and SKIP2 specify the number of bytes to skip at the beginning of each file which is zero by default and OPTION refers to the options compatible with this

command about which we will discuss later on. cmp Example : As explained that the cmp command reports the byte and line number if a difference is found. Now let's find out the same with the help of an example. Suppose there are two files which you want to compare one is file1.txt and other is file2.txt :

$cmp file1.txt file2.txt
If the files are not identical : the output of the above command will be :
$cmp file1.txt file2.txt
file1.txt file2.txt differ: byte 9, line 2

/*indicating that the first mismatch found in two
files at byte 20 in second line*/
If the files are identical : you will see something like this on your screen: $cmp
file1.txt file2.txt
$ _
/*indicating that the files are identical*/

comm command in Linux with examples

comm compare two sorted files line by line and write to standard output; the lines that are common and the lines that are unique.

Suppose you have two lists of people and you are asked to find out the names available in one and not in the other, or even those common to both. comm is the command that will help you to achieve this. It requires two sorted files which it compares line by line.
Before discussing anything further first let's check out the syntax of comm command: Syntax :

$comm [OPTION]... FILE1 FILE2
As using comm, we are trying to compare two files therefore the syntax of comm command needs two filenames as arguments.
With no OPTION used, comm produces three-column output where first column contains lines unique to FILE1 ,second column contains lines unique to FILE2 and third and last column contains lines common to both the files. comm command only works right if you are comparing two files which are already sorted. Example: Let us suppose there are two sorted files file1.txt and file2.txt and now we will use comm command to compare these two.

// displaying contents of file1 //
$cat file1.txt
Apaar
Ayush Rajput
Deepak
Hemant

// displaying contents of file2 //
$cat file2.txt
Apaar

Hemant
Lucky
Pranjal Thakral
Now, run comm command as:

// using comm command for
comparing two files //
$comm file1.txt file2.txt
Apaar
Ayush Rajput
Deepak
Hemant
Lucky
Pranjal Thakral

diff command in Linux with examples

diff stands for difference. This command is used to display the differences in the files by comparing the files line by line. Unlike its fellow members, cmp and comm, it tells us which lines in one file have is to be changed to make the two files identical.

The important thing to remember is that diff uses certain special symbols and instructions that are required to make two files identical. It tells you the instructions on how to change the first file to make it match the second file.

Special symbols are:

a : add c :
change d
: delete
Syntax :

diff [options] File1 File2
Lets say we have two files with names a.txt and b.txt containing 5 Indian states.

$ ls
a.txt b.txt

$ cat a.txt
Gujarat
Uttar Pradesh
Kolkata
Bihar
Jammu and Kashmir

$ cat b.txt
Tamil Nadu
Gujarat
Andhra Pradesh
Bihar
Uttar pradesh
Now, applying diff command without any option we get the following output:

$ diff a.txt b.txt
0a1
> Tamil Nadu
2,3c3
< Uttar Pradesh
 Andhra Pradesh
5c5
 Uttar pradesh
Let's take a look at what this output means. The first line of the diff output will contain:

Line numbers corresponding to the first file,
A special symbol and
Line numbers corresponding to the second file.
Like in our case, 0a1 which means after lines 0(at the very beginning of file) you have to add Tamil Nadu to match the second file line number 1. It then tells us what those lines are in each file preceded by the symbol:

Lines preceded by a < are lines from the first file.
Lines preceded by > are lines from the second file.
Next line contains 2,3c3 which means from line 2 to line 3 in the first file needs to be changed to match line number 3 in the second file. It then tells us those lines with the above symbols.
The three dashes ("—") merely separate the lines of file 1 and file 2.
As a summary to make both the files identical, first add Tamil Nadu in the first file at very beginning to match line 1 of second file after that change line 2 and 3 of first file i.e. Uttar Pradesh and Kolkata with line 3 of second file i.e. Andhra Pradesh. After that change line 5 of first file i.e. Jammu and Kashmir with line 5 of second file i.e. Uttar Pradesh.

Tar command in Linux/Unix with Examples

The tar command is short for tape archive in Linux. This command is used for creating Archive and extracting the archive files. In Linux, it is one of the essential commands which facilitate archiving functionality. We can use this command for creating uncompressed and compressed archive files and modify and maintain them as well.

Syntax of tar command:
tar [options] [archive-file] [directory or file to be archived]

Options in the tar command
Various options in the tar command are listed below:

-c: This option is used for creating the archive.
-f: This option is used for creating an archive along with the provided name of the file.
-x: This option is used for extracting archives.
-u: It can be used for adding an archive to the existing archive file.
-t: It is used for displaying or listing files inside the archived file.
-A: This option is used for concatenating the archive files.
-v: It can be used to show verbose information.
-j: It is used for filtering archive tar files with the help of tbzip.
-z: It is a zip file and informs the tar command that makes a tar file with the help of gzip.
-r: This option is used for updating and adding a directory or file in an existing .tar file. -W: This option is used for verifying the archive file.

ZIP command in Linux with examples

ZIP is a compression and file packaging utility for Unix. Each file is stored in single .zip {.zip-filename} file with the extension .zip.

zip is used to compress the files to reduce file size and also used as file package utility. zip is available in many operating systems like unix, linux, windows etc.
If you have a limited bandwidth between two servers and want to transfer the files faster, then zip the files and transfer.
The zip program puts one or more compressed files into a single zip archive, along with information about the files (name, path, date, time of last modification, protection, and check information to verify file integrity). An entire directory structure can be packed into a zip archive with a single command.
Compression ratios of 2:1 to 3:1 are common for text files. zip has one compression method (deflation) and can also store files without compression. zip automatically chooses the better of the two for each file to be compressed.
The program is useful for packaging a set of files for distribution; for archiving files; and for saving disk space by temporarily compressing unused files or directories. Syntax :

zip [options] zipfile files_list Syntax
for Creating a zip file:

$zip myfile.zip filename.txt

unzip lists, tests, or extracts files from archives of the zip format, which are most commonly found on MS-DOS and Windows systems. The default behavior (with no options) is to extract into the current directory (and possibly the subdirectories below it) all files from the specified zip archive. A companion program, zip, creates zip archives. Both zip and unzip are compatible with archives created by PKWARE's PKZIP and PKUNZIP programs for MSDOS.

Gzip Command in Linux

gzip command compresses files. Each single file is compressed into a single file. The compressed file consists of a GNU zip header and deflated data.
If given a file as an argument, gzip compresses the file, adds a ".gz" suffix, and deletes the original file. With no arguments, gzip compresses the standard input and writes the compressed file to standard output.
Difference between Gzip and zip command in Unix and when to use which command

ZIP and GZIP are two very popular methods of compressing files, in order to save space, or to reduce the amount of time needed to transmit the files across the network, or internet. In general, GZIP is much better compared to ZIP, in terms of compression, especially when compressing a huge number of files.
The common practice with GZIP, is to archive all the files into a single tarball before compression. In ZIP files, the individual files are compressed and then added to the archive. When you want to pull a single file from a ZIP, it is simply extracted, then decompressed. With GZIP, the whole file needs to be decompressed before you can extract the file you want from the archive.
When pulling a 1MB file from a 10GB archive, it is quite clear that it would take a lot longer in GZIP, than in ZIP.
GZIP's disadvantage in how it operates, is also responsible for GZIP's advantage. Since the compression algorithm in GZIP compresses one large file instead of multiple smaller ones, it can take advantage of the redundancy in the files to reduce the file size even further. If you archive and compress 10 identical files with ZIP and GZIP, the ZIP file would be over 10 times bigger than the resulting GZIP file.

Syntax :
gzip [Options] [filenames]

Example:
$ gzip mydoc.txt
This command will create a compressed file of mydoc.txt named as mydoc.txt.gz and

gunzip command in Linux with examples
gunzip command is used to compress or expand a file or a list of files in Linux. It accepts all the files having extension as .gz, .z, _z, -gz, -z , .Z, .taz or.tgz and replace the compressed file with the original file by default. The files after uncompression retain its actual extension.
Syntax:
gunzip [Option] [archive name/file name]
[File attributes, File and directory attributes listing and very brief idea about the attributes, File ownership, File permissions, Changing file permissions – relative permission & absolute permission, Changing file ownership, Changing group ownership, File system and inodes, Hard link, Soft link, Significance of file attribute for directory, Default permissions of file and directory and using umask, Listing of modification and access time, Time stamp changing (touch), File locating (find)]