Display Content of a File:

You can use **cat** command to see the content of a file. Following is the simple example to see the content of above created file:

```
$ cat filename
This is unix file....I created it for the first time.....
I'm going to save this content in this file.

hello
$
```

Display Line numbers

To display the contents of a file with the line number in front of each line, use option -n.

```
$ cat filename -n
1 This is unix file....I created it for the first time.....
2 I'm going to save this content in this file.
3
4 hello
$
```

You can display line numbers by using **-b** option along with **cat** command as follows:

```
$ cat filename -b
1 This is unix file....I created it for the first time.....
2 I'm going to save this content in this file.
3 hello
$
```

Create a file

Cat is also useful to create a file.

```
$ cat> f1
Hi
How are you
[Ctrl-d]
$
```

Append in a existing file

```
$ cat>> f1
bye
[Ctrl-d]
$
```

Copy file content

you can copy the contents of one file into another as shown below.

```
$ cat program.pl >backup_prgm.pl
```

Concatenate Contents of Multiple Files

Through cat command, you will be able to concatenate contents of more than one file into a new file.

```
$ cat program.pl program2.pl >all_pgrm.pl
```

-v option is used to display nonprinting characters.

Copying a file (cp)

cp command is used to copies a file or group of files.

```
$ cp chap1 back_chap1
```

It will copy content of chap1 file into back_chap1 file. If back_chap1 is not exists, it will create it then copy the content of chap1. If back_chap1 exist, it overwrites it.

```
$ cp chap1 progs/backup_1
```

Here progs is directory. It will create copy of chap1 from current directory to progs directory with new name backup_1.

```
$ cp chap1 progs
```

It will create copy of chap1file from current directory to progs directory with same name.

```
$ cp chap1 chap2 chap3 progs
```

It will copy chap1, chap2, chap3 files in to progs directory.

You can compress above sequence using the * as a suffix to chap.

```
$ cp chap* progs
```

Interactive Copying (-i)

The –i option warns the user before overwriting the destination file.

```
$ cp -i chap1 progs/backup_1
cp: overwrite backup_1 (yes/no) ? y
```

If backup_1 exists, cp prompts for a response. A **y** response overwrites the file, any other response leaves it uncopied.

Copying Directory Structures (-R)

The cp –R command behaves recursively to copy an entire directory structure.

$ cp –R progs newprogs	newprogs must not exist

If newprogs doesn't exist, cp creates it and copy all the contents of progs into newprogs. But if newprogs exists, progs becomes a subdirectory under newprogs. Sometime it is not possible to copy file because if the source file is read-protected or destination file or directory is write-protected.

rm(Deleting Files)

The rm(remove) command deletes one or more files. This command do not give any warning.

Example:

```
$rm chap1
```

It delete the file chap1.

```
$rm chap1 chap2 chap3
```

It delete the files chap1, chap2 and chap3.

Without cd you can remove the files from particular directory.

```
$ rm progs/chap1 progs/chap2
```

The above command delete chap1 and chap2 file from progs directory.

```
$ rm *
```

The above command delete all files in the current directory without warning.

Interactive Deletion (-i)

–ioption give ask confirmation message to user before removing each file.

```
$ rm –i chap1 chap2
rm: remove chap1 (yes/no)? ?y
```

```
rm: remove chap2 (yes/no)? ?n
```

y remove file chap1, any other response leaves the file undeleted.

Recursive Deletion (-r or –R)

-r or R option is recursive deletion which deletes everything whether it is files, subdirectories or files within subdirectories. Normally, rm would not remove directories, but when you used with recursive option it removes.

```
$ rm –r *
or
$ rm –R*
```

The above delete all files, subdirectories and files within subdirectories.

Forcing Removal (-f)

rm prompts for removal if a file is write-protected. The –f option overrides this minor protection and forces removal. When you combine it with the –r option, it could be most risky.

```
$ rm –rf *
```

Deletes everything in current directory.

mv (Renaming files)

The mv command renames or moves files.

It has two functions:

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

```
$ mv chap1 c1
```

It rename file chap1 to c1. No additional space is consumed on disk during renaming. If the c1 file already exist it overwrites it without warning.

mv can also used for rename a directory.

```
$ mv progs newprogs
```

It is also possible to move group of files to particular directory.

```
$ mv chap1 chap2 chap3 backupdir
```

The above command moves chap1, chap2,chap3 in backupdir directory.

Paging output (more)

With more command, the content of the file display on the screen, one page at a time. At the bottom of the screen, you will see filename and percentage of the file that has been viewed.

Options

Option	Function
F or Spacebar	Scroll forward a page at a time
B	Move back one page

Repeat features

If you want to forward or backward multiple page at a time, it is possible with more command. **10f** scroll forward 10 pages. **5b** scroll back 5 pages.

Searching for pattern

You can search for a pattern with the / command followed by the string.

/hello

It will search first hello in your file, for repeat this search press **n**.

Using more in pipeline

To see the output of another command pagewise, more can be piped with that command.

For example ls output doesn't fit on the screen, if there are two many files in the directory. so do following:

```
$ ls | more
```

Now the output of ls will comes one page at a time.

Printing a file (lp)

The command **lp** prints a file on printer.

The following lp command prints a single copy of food file.

```
$lp food
request id is laserp-525(1 file)
$
```

The prompt is returned immediately after the job is submitted. The file is not actually printed at the time the command is invoked, it depends on the number of jobs already lined up in the queue. Several users can print their files in this way without conflict.

Options

If there are multiple printers, you have to use –d option with the printer name.

-d option

-d*printername*

```
$lp –dlaser chap1                    can also provide space after –d
```

-t option

-t*title*

-t option followed by title string prints the title on the first page.

```
$ lp –t "First chapter" chap1
```

-n option

-n*num*

You can also print multiple copies with –n option.

```
$ lp –n3 –m chap1
```

It will print 3 copies of chap1 file. –m option mails user a message after the file has been printed.

cancel

You can cancel any printing job submitted by you using cancel command. cancel uses the request-id or printer name as argument.

$ cancel laser	cancel current job on printer laser
$ cancel pr1-320	cancel job with request-id pr1-320

You can cancel only those jobs that your own. But the system administrator can cancel any job. Cancel is effective only when a job remains in the print queue. If it is already being printed , cancel can't do anything.

Home Directory:

The directory in which you find yourself when you first login is called your home directory.

```
$ echo $HOME
/home/amrood
```

You will be doing much of your work in your home directory and subdirectories that you'll be creating to organize your files.

You can go in your home directory anytime using the following command:

```
$cd ~
$
```

Here ~ indicates home directory. If you want to go in any other user's home directory then use the following command:

```
$cd ~username
$
```

To go in your last directory you can use following command:

```
$cd -
$
```

Print Working Directory (pwd)

To determine where you are within the filesystem hierarchy at any time, enter the command **pwd** to print the current working directory:

```
$pwd
/user0/home/amrood
$
```

Absolute/Relative Pathnames:

Directories are arranged in a hierarchy with root (/) at the top. The position of any file within the hierarchy is described by its pathname. Elements of a pathname are separated by a /.

Absolute Pathname :

A pathname is absolute if it is described in relation to root, so absolute pathnames always begin with a /.

These are some example of absolute filenames.

```
/etc/passwd
/users/sjones/chem/notes
/dev/rdsk/Os3
```

Example

```
$cat  /progs/progs1/abc.txt
```

Relative Pathname:

A pathname can also be relative to your current working directory. Relative pathnames never begin with /. Relative to user amrood' home directory, some pathnames might look like this:

```
chem/notes
personal/res
progs
```

Example:

```
$ cat abc.txt
```

Will work if abc.txt in current directory only.

Listing Directories:

To list the files in a directory you can use the following syntax:

```
$ls dirname
```

Following is the example to list all the files contained in /usr/local directory:

```
$ls /usr/local
```

```
X11 bin gimp jikessbin
ace doc include lib share
atalketc info man ami
```

Listing Directories and Files(ls):

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 filesystem.

Listing Files and Direcotries:

To list the files and directories stored in the current directory. Use the following command:

```
$ls
bin         hosts       lib         res.03
ch07        hw1         pub         test_results
ch07.bak    hw2         res.01      users
docs        hw3         res.02      work
```

Here is the sample output of the above command: Filenames and directory arranged in ASCII collating sequence. (number first, uppercase and then lowercase).

Display all information about file and directories (-l option)

The command **ls**supports the **–l** option which would help you to get more information about the listed files, it gives long list:

```
$ls -l
total 1962188
drwxrwxr-x 2 amroodamrood 4096 Dec 25 09:59 uml
-rw-rw-r-- 1 amroodamrood 5341 Dec 25 08:38 uml.jpg
```

```
drwxr-xr-x 2 amroodamrood 4096 Feb 15 2006 univ
drwxr-xr-x 2 root root 4096 Dec 9 2007 urlspedia
-rw-r--r-- 1 root root 276480 Dec 9 2007 urlspedia.tar
drwxr-xr-x 8 root root 4096 Nov 25 2007 usr
drwxr-xr-x 2 200 300 4096 Nov 25 2007 webthumb-1.01
-rwxr-xr-x 1 root root 3192 Nov 25 2007 webthumb.php
-rw-rw-r-- 1 amroodamrood 20480 Nov 25 2007 webthumb.tar
-rw-rw-r-- 1 amroodamrood 5654 Aug 9 2007 yourfile.mid
-rw-rw-r-- 1 amroodamrood 166255 Aug 9 2007 yourfile.swf
drwxr-xr-x 11 amroodamrood 4096 May 29 2007 zlib-1.2.3
$
```

Here is the information about all the listed columns:

1. First Column: represents file type and permission given on the file. Below is the description of all type of files.

2. Second Column: represents the number of memory blocks taken by the file or directory.

3. Third Column: represents owner of the file. This is the Unix user who created this file.

4. Fourth Column: represents group of the owner. Every Unix user would have an associated group.

5. Fifth Column: represents file size in bytes.

6. Sixth Column: represents date and time when this file was created or modified last time.

7. Seventh Column: represents file or directory name.

In the ls -l listing example, every file line began with a d, -, or l. These characters indicate the type of file that's listed.

Prefix	Description
-	Regular file, such as an ASCII text file, binary executable, or hard link.
b	Block special file. Block input/output device file such as a physical hard drive.
c	Character special file. Raw input/output device file such as a physical hard drive
d	Directory file that contains a listing of other files and directories.
l	Symbolic link file. Links on any regular file.
p	Named pipe. A mechanism for interprocess communications
s	Socket used for interprocess communication.

Meta Characters:

Meta characters have special meaning in Unix. For example ***** and **?**aremetacharacters. We use ***** to match 0 or more characters, a question mark **?**matches with single character.

For Example:

```
$ls ch*.doc
```

Displays all the files whose name start with ch and ends with .doc:

ch01-1.doc ch010.doc ch02.doc ch03-2.doc

ch04-1.doc ch040.doc ch05.doc ch06-2.doc

ch01-2.doc ch02-1.doc c

Here * works as meta character which matches with any character. If you want to display all the files ending with just .doc then you can use following command:

```
$ls *.doc
```

Hidden Files (-a option):

An invisible file is one whose first character is the dot or period character (.). UNIX programs (including the shell) use most of these files to store configuration information.

Some common examples of hidden files include the files:

.profile :the Bourne shell (sh) initialization script

.kshrc :the Korn shell (ksh) initialization script

.cshrc :the C shell (csh) initialization script

.rhosts :the remote shell configuration file

To list invisible files, specify the -a option to ls:

```
$ ls -a
.       .emacs      .kshrc      .rhosts     ch07.bak

..      .exrc       .profile    ch07        bin

$
```

Single dot **.** : This represents current directory.

Double dot **..** : This represents parent directory.

Hidden file except current and parent directory (-A option)

-A option also show hidden files and directory but it exclude current directory and parent directory from list

```
$ ls -A
.emacs      .kshrc      .rhosts     ch07.bak
.exrc       .profile    ch07        bin

$
```

Classify the file (-F option)

- You can use -F which classifies the file with different special character for differen
- Symbolic links with trailing at the rate sign (@)
- Regular files with nothing

```
$ ls -F
Desktop/  Documents/  Ubuntu-App@  firstfile  Music/  Public/  Templates/
```

Display One File Per Line(-1 option)

To show single entry per line, use -1 option as shown t kind of files.

- Directories with trailing slash (/)
- Executable files with trailing asterisk (*)
- FIFOs with trailing vertical bar (|)

below.

```
$ ls -1

bin
```

```
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
```

Display File Size in Human Readable Form (-h option)

Use **ls -lh** (h stands for human readable form), to display file size in easy to read format. i.e M for MB, K for KB, G for GB.

```
$ ls -l
-rw-r----- 1 ramesh team-dev 9275204 Jun 12 15:27 arch-linux.txt.gz*

$ ls -lh
-rw-r----- 1 ramesh team-dev8.9M Jun 12 15:27 arch-linux.txt.gz
```

Display Directory Information Using ls -ld

When you use "ls -l" you will get the details of directories content. But if you want the details of directory then you can use -d option as., For example, if you use ls -l /etc will display all the files under etc directory.

```
$ ls -l /etc

total 3344
```

```
-rw-r--r--   1 root root   15276 Oct  5  2004 a2ps.cfg

-rw-r--r--   1 root root    2562 Oct  5  2004 a2ps-site.cfg

drwxr-xr-x   4 root root    4096 Feb  2  2007 acpi

-rw-r--r--   1 root root      48 Feb  8  2008 adjtime

drwxr-xr-x   4 root root    4096 Feb  2  2007 alchemist
```

But, if you want to display the information about the /etc/ directory, use -ld option as shown below.

```
$ ls -ld /etc
drwxr-xr-x 21 root root 4096 Jun 15 07:02 /etc
```

Order Files Based on Last Modified Time Using ls -lt

To sort the file names displayed in the order of last modification time use the -t option. You will be finding it handy to use it in combination with -l option.

```
$ ls -lt
total 76
drwxrwxrwt  14 root root  4096 Jun 22 07:36 tmp
drwxr-xr-x 121 root root  4096 Jun 22 07:05 etc
drwxr-xr-x  13 root root 13780 Jun 22 07:04 dev
drwxr-xr-x  13 root root  4096 Jun 20 23:12 root
drwxr-xr-x  12 root root  4096 Jun 18 08:31 home
drwxr-xr-x   2 root root  4096 May 17 21:21 sbin
lrwxrwxrwx   1 root root    11 May 17 20:29 cdrom -> media/cdrom
drwx------   2 root root 16384 May 17 20:29 lost+found
```

```
drwxr-xr-x  15 root root  4096 Jul  2  2008 var
```

Order Files Based on Last Modified Time (In Reverse Order) Usingls -ltr

To sort the file names in the last modification time in reverse order. This will be showing the last edited file in the last line which will be handy when the listing goes beyond a page.

```
$ ls -ltr

total 76
drwxr-xr-x  15 root root  4096 Jul  2  2008 var
drwx------   2 root root 16384 May 17 20:29 lost+found
lrwxrwxrwx   1 root root    11 May 17 20:29 cdrom -> media/cdrom
drwxr-xr-x   2 root root  4096 May 17 21:21 sbin
drwxr-xr-x  12 root root  4096 Jun 18 08:31 home
drwxr-xr-x  13 root root  4096 Jun 20 23:12 root
drwxr-xr-x  13 root root 13780 Jun 22 07:04 dev
drwxr-xr-x 121 root root  4096 Jun 22 07:05 etc
drwxrwxrwt  14 root root  4096 Jun 22 07:36 tmp
```

Order File by last access time (-u option)

To sort the file name in the last access time use –u option.

Display Files Recursively (-R option)

To show all the files recursively, use -R option.

```
$ ls -R

.:

documents  sum.pl



./documents:

file.txt
```

Display File Inode Number (-i option)

Use -i option to display inode number of file.

```
$ ls -i

10584066 documents     3482450 sum.pl
```

Display column wise (-x option)

Use –x option to display list columnwise.

```
$ ls–x

82_abc.html      TA.ch              calc               cp.sh

Dept.lst emp.lst           helpdir            progs

$
```

Listing Directory Content

Give the directory names with ls command to display the content of the directories.

```
$ ls –x mydir progs
Mydir:
abc.txt        ef.sh        gh.txt

progs:
arr.pl         n1.sh        n2.sh
```

Creating/Making Directories (mkdir):

Directories are created by the following command:

```
$mkdir  dirname
```

Here, directory is the absolute or relative pathname of the directory you want to create. For example, the command:

```
$mkdir mydir1
$
```

Creates the directory mydir1 in the current directory.

Here is another example:

```
$mkdir /tmp/testdir1
$
```

This command creates the directory testdir1 in the /tmp directory. The **mkdir**command produces no output if it successfully creates the requested directory.

If you give more than one directory on the command line, mkdir creates each of the directories.

```
$mkdir docs pub
$
```

The above Creates the directories docs and pub under the current directory.

Creating Parent Directories:

Sometimes when you want to create a directory, its parent directory or directories might not exist. In this case, mkdir issues an error message as follows:

```
$mkdir /tmp/testdir2/test1
mkdir: Failed to make directory "/tmp/testdir/test1";
No such file or directory
$
```

In such cases, you can specify the **-p** option to the **mkdir**command. It creates all the necessary directories for you. For example:

```
$mkdir -p /tmp/testdir2/test1
$
```

Above command creates all the required parent directories.

It is also possinle to create a directory tree in UNIX.

```
$ mkdir up  up/up1  up/up2
```

This creates three directories, up directory and two subdirectory up1 and up2 under up directory.

Removing Directories:

Directories can be deleted using the **rmdir**command as follows:

```
$rmdir dir1
$
```

Note: It will remove dir1 directory if it is empty. To remove a directory make sure it is empty which means there should not be any file or sub-directory inside this directory.

You can delete multiple directories at a time as follows:

```
$rmdir up/up1 up/up2 up
$
```

Above command removes the directory tree just created before.First you have to delete subdirectory then parent directory.

The rmdir command produces no output if it is successful.

Changing Directories:

You can use the **cd** command to change the current directory to the directory specified as argument. You can use it to change to any directory by specifying a valid absolute or relative path.

The syntax is as follows:

```
$cd dirname
$
```

Here, dirname is the name of the directory that you want to change to. For example, the command:

```
$ pwd
/home/karan
$cd  progs
$ pwd
/home/karan/progs
```

```
$cd /usr/local/bin
$
```

Changes to the directory /usr/local/bin. From this directory you can cd to the directory /usr/home/amrood using the following relative path:

```
$cd ../../home/amrood
$
```

Renaming Directories (mv):

The mv (move) command can also be used to rename a directory. The syntax is as follows:

```
$mv olddir newdir
$
```

You can rename a directory **mydir** to **yourdir** as follows:

```
$mv mydir yourdir
$
```

The directories . (dot) and .. (dot dot)

The filename . (dot) represents the current working directory; and the filename .. (dot dot) represent the directory one level above the current working directory, often referred to as the parent directory.

Knowing the file types (file)

UNIX provides the file command to determine the type of file, especially of an ordinary file. You can use it with one or more filenames as arguments.

```
$ file arch.zip
Arch.zip      ZIP archive
```

Counting Words in a File:

You can use the **wc**command to get a count of the total number of lines, words, and characters contained in a file. Following is the simple example to see the information about above created file:

Example 1:

```
$ wc filename
2 19 103 filename
$
```

Here is the detail of all the four columns:

1. First Column: represents total number of lines in the file.

2. Second Column: represents total number of words in the file.

3. Third Column: represents total number of bytes in the file. This is actual size of the file.

4. Fourth Column: represents file name.

Example 2:

```
$ wc –l filename
2 filename
$
```

Example 3:

```
$ wc –w filename
19 filename
$
```

Example 4:

```
$ wc –c filename
103 filename
$
```

You can give multiple files at a time to get the information about those file. Here is simple syntax:

```
$ wc filename1 filename2 filename3

3       29      100     filename1
2       10      60      filename2
1       5       25      filename3
6       44      185     total
$
```

Displaying data in octal (od)

Some files (especially Executable files) contain nonprinting characters, and most UNIX commands don't display them properly.

Following is an file1 contains some of these characters that don't show up normally.

```
$ more file1
```

File comparision

There are three commands in unix to compare two files. cmp,comm.,diff.

Comparing two files (cmp)

The cmp command is used to check whether two files are identical or not.

If two files are identical, cmp displays no message, but simply returns the prompt. We can check it with two copies of the same file.

```
$ cmp chap01 chap01
$ _
```

It display no message means the content of above two files are identical.

```
$ cmp chap01 chap02
Chap01 chap02 differ : char7, line 1
```

The two files are compared byte by byte, and the location of the first mismatch (in the seventh character of first line) is displayed to the screen.

Cmp doesn't bother about possible subsequent mismatches but displays a detailed list when used with –l(list) opton.

What is common (comm)

Comm command compares two sorted files line-by-line.

With no options, **comm** produces three-column output. Column one contains lines unique to **FILE1**, column two contains lines unique to **FILE2**, and column three contains lines common to both files. Each of these columns can be suppressed individually with options.

Options

-1	suppress column 1 (lines unique to **FILE1**)
-2	suppress column 2 (lines unique to **FILE2**)
-3	suppress column 3 (lines that appear in both files)

```
$ cat file1
c. k. shukla
chanchalsinghvi
s.n. dasgupta
sumitchakrobarty
$ cat file2
anilaggarwal
barunsengupta
c.k. shukla
lalitchowdury
s.n.dasgupta
```

Both files are sorted and have some differences.

```
$ comm file1 file2
        anilaggarwal
        barunsengupta
                c.k. shukla
chanchalsinghvi
        lalitchowdury
                s.n. dasgupta
sumitchakrobarty
```

The firt column contains two lines unique to the first line. And the second column shows three lines unique to the second file. The third column displays two lines common to both files.

$ commn -3 file1 file2	select lines not common to both files
$ comm -13 file1 file2	select lines present only in second file

Converting one file to other (diff)

diff command is also used to display file differences. It also tells you which lines in one file have to be changed to make the two files identical.

```
$ diff file1 file2
OR
$ diff file[12]
0a1,2                   Append after line 0 of first file
>anilaggarwal           this line
>barunsengupta          and this line
2c4                     change line 2 of first file
<chanchalsinghvi        Replacing this line
--                      with
>lalitchaudhari         this one
4d5                     Delete line 4 of first file
<sumitchakrobarty containing this one
```

Maintaining several version of a File (-e)

Diff –e produces a set of instructions only, but these instructions can be used with the ed editor to convert one file to other. This facility saves disk space by letting us stores the oldest file in entirety, and only the changes between consecutive versions.

Compressing and Archiving Files

To conserve disk space you will need to compress large and infrequently used files. Moreover, before sending a large file as an email attachment, it's good to compress file first.

UNIX offers following compression and decompression utilities:

gzip and gunzip (.gz)

zip and unzip (.zip)

gzip

gzip is used to compress the file. gzip works with one or more filenames, it provides the extension .gz to the compressed filename and removes the original file.

Example 1

```
$wc –c lib.html
3875302 lib.html
$ gzip lib.html
$wc –c lib.html
788096 lib.html
```

Example 2

```
$ gzip lib1.html lib2.html
```

gzip options

Uncomressing a "gzipped" file (-d)

To restore the original and uncompressed file, you have two options: uses either **gzip –d**

Or

Gunzip with one or more filenames as arguments. The .gz extension is optional to give.

$ gzip –d lib.html	Retrieves lib.html
$ gunzip lib.html.gz	Same as Above
$ gunzip lib1.html.gz lib2.html.gz	Retrieve two files

Recursive Compression (-r)

We can also compress all files within the directory using the –r option. The argument must directory.

$ gzip –r progs	Compress all files in progs directory

To decompress all files in this directory you need to use

gunzip –r progs	Retrieves file in compressed progs
gzip –dr progs	Same as above

tar

For creating disk archive that contains a group of files or an entire directory structure, we use tar command.

Options

Option	Description
-c	Create an archive
-x	Extract files from archive
-t	Display files in archive
-f arch	Specify the archive arch

Create an archive (-c)

To create an archive we need to specify the name of the archive (with –f), the copy or write operation (-c) and the filenames as arguments. –v (verbose) option to display the progress while tar works.

```
$ tar –cvf archive.tar lib1.html  lib2.html
a lib1.html 3785K
a lib2.html 364K
```

here a indicates append. Here the archive.tar is an archive file which archived two files lib1.html and lib2.html.

tar command is also used to backup two or more directories.

```
$ tar –cvf progs.tar prog1 prog2 prog3
```

If the archive file is very big, you can compress it with gzip.

```
$ gzip archive.tar                Archived and compressed
```

This creates a “tar-gzipped” file, archive.tar.gz.

Extracting files from Archive (-x)

tar command uses –x option to extract files from an archive.

$ tar –xvf progs.tar	Extract the three directories

But to extract files from .tar.gz file (example archive.tar.gz), you must first use gunzip to decompress the archive and then run tar command

$ gunzip archive.tar.gz			**Retrieves archive.tar**
$ tar –xvf archive.tar			**Extract files**
x	lib1.html,	3875302 bytes,	7569 tape blocks
x	lib2.html,	372267 bytes,	728 tape block
tar –xvf archive.tar lib1.html			Extract only lib1.html

Viewing the archive (-t)

To view the contents of the archive, use the –t (table of contents) option. It doesn't extract files, but simply shows their attributes.

$ tar –tvf archive.tar					
-rw-r--r--	102/10	3875302	Aug 24 19:49	2002	lib1.html
-rw-r--r--	102/10	372267	Aug 24 19:48	2002	lib2.html

zip

Zip done two works. It comibnes the compressing function of gzip and archival function of tar. Zip required the first argument to be the compressed file name, the remaining arguments are fils and directories to be compressed.

```
$ zip archive.zip lib1.html lib2.html
adding : lib1.html (defaulted 80%)
adding: lib2.html (defaulted 66%)
```

The good feature of this command is that it doesn't overwrite an existing compressed file, if archive.zip exists, files will either be updated or appended to the archive depending on wheather they already exist in the archive.

Recursive compression (-r)

```
$ zip –r student_home.zip
```

it recursively compresses the all the directory structure. If from student's home directory if we write above command it will compress whole directory structure of students.

unzip

Files are restored with unzip command. Unzip does a noninteractive restoration if it doesn't overwrite an existing files:

```
$ unzip archive.zip
Archive: archive.zip
        Inflating: lib1.html
        Inflating: lib2.html
```

But if the uncompressed file exist on disk, unzip makes sure before overwrites.

Replace lib1.html ? [y]es, [n]o, [A]ll, [N]one, [r]ename : y

You can responde it with y or n. you can also rename the file (r) to prevent overwriting or direct unzip to perform the decompression on the remaining files noninteractively (A).

Viewing the Archive(-v)

You can view the compressed archive with the –v option.

```
$ unzip –v archive.zip
```