Linux

Linux Commands

ls ➔ ls command is used to list files or directories in Linux and other Unix-based operating systems.

flags change how the ls command works:

```
ls [flags] [directory]
```

the ls command to list the contents of the current working directory:

```
ls
```

List files in another directory

Type the ls [directory path here] command to list the contents of another directory:

```
ls Desktop
```

ls /   list the contents of the root directory:

- **List files in the user's home directory (/home/user)**
- Type **the ls ~** command to list the contents in the users's home directory:
- **List only directories**
- Type the **ls -d */** command to list only directories:
- **List files with subdirectories**
- Type the ls * command to list the contents of the directory with it's subdirectories:
- **List files in long format**
- Type **the ls -l** command to list the contents of the directory in a table format with columns including:
- **List files including hidden files**
- Type the **ls -a** command to list files or directories including hidden files or directories. In Linux, anything that begins with a . is considered a hidden file:

cp command

- **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.
- 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.

- **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
- **cp 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**

```
$ ls
a.txt  b.txt  new
Initially new is empty
$ ls new
$ cp a.txt b.txt new
$ ls new
a.txt  b.txt
```

- Option 'r' with the copy command can be used to copy a directory including all its content from a source directory to the destination directory.
- cp –r
- cp -r <sourceDirectory> <destinationDirectory>
- cp -r library /home/sssit/Documents
- -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
```

mv command

- **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.
- mv [Option] source destination
- 4 files having names a.txt, b.txt, and so on till d.txt. To rename the file a.txt to gk.txt(not exist):

- $ ls
- a.txt b.txt c.txt d.txt
- $ mv a.txt gk.txt
- $ ls
- b.txt c.txt d.txt gk.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).

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

- $ cat gk.txt
- India

- $ cat b.txt
- CDAC

- $ mv gk.txt b.txt

- $ ls
- b.txt c.txt d.txt

- $ cat b.txt
- India

-i (Interactive):

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

Linux cp --backup

If the file you want to copy already exists in the destination directory, you can backup your existing file with the use of this command.

Syntax:

```
cp --backup <filename> <destinationDirectory>
```

Example:

```
cp --backup file2.txt /home/sssit/Downloads
```

‘sort’ Command

- **Sort** is a Linux program used for printing lines of input text files and concatenation of all files in sorted order. Sort command takes blank space as field separator and entire Input file as sort key. It is important to notice that sort command don’t actually sort the files but only print the sorted output, until your redirect the output.
- 1. First we will be creating a text file (tec.txt) to execute ‘sort‘ command
- The option ‘-e‘ in the below command enables interpretion of backslash and /n tells echo to write each string to a new line.

-
  ```
  echo -e "computer\nmouse\nLAPTOP\ndata\nRedHat\nlaptop\ndebian\nlaptop" > tec.txt
  ```
- 2. Before we start with ‘sort‘ lets have a look at the contents of the file and the way it look.

-
  ```
  $ cat tec.txt
  ```

3. Now sort the content of the file using following command.

```
$ sort tec.txt
```

Note: The above command don’t actually sort the contents of text file but only show the sorted output on terminal.

4. Sort the contents of the file ‘**tec.txt**‘ and write it to a file called (**sorted.txt**) and verify the content by using cat command.

```
$ sort tec.txt > sorted.txt
$ cat sorted.txt
```

5. Now sort the contents of text file ‘tec.txt‘ in reverse order by using ‘-r‘ switch and redirect output to a file ‘reversesorted.txt‘. Also check the content listing of the newly created file.

```
$ sort -r tec.txt > reversesorted.txt
$ cat reversesorted.txt
```

Grep Command

- grep stands for Globally Search For Regular Expression and Print out. It is a command line tool used in UNIX and Linux systems to search a specified pattern in a file or group of files.
- Without passing any option, grep can be used to search for a pattern in a file or group of files. The syntax is:

- grep '<text-to-be-searched>' <file/files>
- Note that single or double quotes are required around the text if it is more than one word.
- For example, say we have the following files (called grep.txt):

```
$cat > grep.txt
```

Hello, how are you

I am grep

Nice to meet you

```
grep you grep.txt
```

- 1. -n (--line-number) - list line numbers

```
grep you grep.txt –n
```

2. -c (--count) - prints the number of lines of matches

```
grep you grep.txt –c
```

3. -v (--invert-match) - prints the lines that do not match the specified pattern

```
grep you grep.txt -v –n
```

4. -i (--ignore-case) - used for case insensitivity

```
# command 1
grep You grep.txt
# command 2
grep YoU grep.txt -i
```

- **5. -l (--files-with-matches) - print file names that match a pattern**

```
# command 1
grep you grep.txt -l
# command 2
grep You grep.txt -i –l
```

```
# result 1
grep.txt
# result 2
# all files in the current directory that matches
# the text 'You' case insensitively
```

6. `-w` (--word-regexp) - print matches of the whole word

By default, grep matches strings which contain the specified pattern. This means that grep yo grep.txt will print the same results as grep yo grep.txt because 'yo' can be found in you. Similarly, 'ou’.

With the option -w, grep ensures that the matches are exactly the same pattern as specified. Example:

```
grep yo grep.txt -w
```

- **7. -o (--only-matching) - print only the matched pattern**
- By default, grep prints the line where the matched pattern is found. With option -o, only the matched pattern is printed line by line. Example:
- **grep yo grep.txt –o**

- **8. -A (--after-context) and -B (--before-context) - print the lines after and before (respectively) the matched pattern.**
- **grep grep grep.txt -A 1 -B 1**

```
Hello, how are you
I am grep
Nice to meet you
```

- **This matched pattern is on line 2. -A 1 means one line after the matched line and -B 1 means one line before the matched line.**

Regular expressions for patterns

- **1. ^pattern - start of a line**
- This pattern means that the grep will match the strings whose lines begin with the string specified after ^. Example:
- **grep ^I grep.txt –n**

```
2: I
```

- **2. pattern$ - end of a line**
- In contrast with ^, $ specifies patterns that will be matched if the line ends with the string before $. Example:
- **grep you$ grep.txt**

```
1: Hello, how are you
3: Nice to meet you
```

Cat Command

The cat (short for “concatenate“) command is one of the most frequently used commands in Linux/Unix-like operating systems. cat command allows us to create single or multiple files, view content of a file, concatenate files and redirect output in terminal or files.

```
$ cat [OPTION] [FILE]...
```

1. Display Contents of File

The below example will show the contents of /etc/passwd file.

```
# cat /etc/passwd
```

2. View Contents of Multiple Files in terminal

In below example, it will display the contents of the test and test1 file in the terminal.

```
# cat test test1
```

```
Hello everybody

Hi world,
```

3. Create a File with Cat Command

We will create a file called test2 file with the below command.

```
# cat >test2
```

Awaits input from the user, type desired text, and press CTRL+D (hold down Ctrl key and type ‘d‘) to exit. The text will be written in the test2 file. You can see the content of the file with the following cat command.

```
# cat test2
hello everyone, how do you do?
```

4. Use Cat Command with More & Less Options

If a file having a large number of content that won’t fit in the output terminal and the screen scrolls up very fast, we can use parameters more and less with the cat command as shown below.

```
# cat song.txt | more
# cat song.txt | less
```

5. Display Line Numbers in File

With the -n option you could see the line numbers of a file song.txt in the output terminal.

```
# cat -n song.txt
```

6. Display $ at the End of File

In the below, you can see with the -e option that ‘$‘ is shows at the end of the line and also in space showing ‘$‘ if there is any gap between paragraphs. This option is useful to squeeze multiple lines into a single line.

```
# cat -e test
```

```
hello everyone, how do you do?$
$
Hey, am fine.$
How's your training going on?$
$
```

8. Display Multiple Files at Once

In the below example we have three files test, test1, and test2, and able to view the contents of those files as shown above. We need to separate each file with ; (semicolon).

```
# cat test; cat test1; cat test2
```

```
This is a test file
This is the test1 file.
This is test2 file.
```

9. Use Standard Output with Redirection Operator

We can redirect the standard output of a file into a new file else existing file with a '>' (greater than) symbol. Careful, existing contents of the test1 will be overwritten by the contents of the test file.

```
# cat test > test1
```

10. Appending Standard Output with Redirection Operator

Appends in existing file with ‘>>‘ (double greater than) symbol. Here, the contents of the test file will be appended at the end of the test1 file.

```
# cat test >> test1
```

12. Redirecting Multiple Files Contain in a Single File

This will create a file called test3 and all output will be redirected in a newly created file.

```
# cat test test1 test2 > test3
```

13. Sorting Contents of Multiple Files in a Single File

This will create a file test4 and the output of the cat command is piped to sort and the result will be redirected to a newly created file.

```
# cat test test1 test2 test3 | sort > test4
```

head, tail

1. head Command

The head command reads the first ten lines of a any given file name. The basic syntax of head command is:

head [options] [file(s)]

For example, the following command will display the first ten lines of the file named ‘/etc/passwd‘.

the first ten lines of each file separately.

```
# head /etc/passwd
```

If more than one file is given, head will show the first ten lines of each file separately. For example, the following command will show ten lines of each file.

```
# head /etc/passwd /etc/shadow
```

If it is desired to retrieve more number of lines than the default ten, then ‘-n‘ option is used along with an integer telling the number of lines to be retrieved. For example, the following command will display first 5 lines from the file ‘/var/log/yum.log‘ file.

```
# head -n5 /var/log/yum.log
```

there is no need to use ‘**-n**‘ option.

```
# head  -5 /var/log/yum.log
```

The tail command allows you to display last ten lines of any text file. Similar to the head command above, tail command also support options ‘n‘ number of lines and ‘n‘ number of characters.

The basic syntax of tail command is:

```
# tail [options] [filenames]
```

For example, the following command will print the last ten lines of a file called ‘access.log‘.

```
# tail access.log
```

If more than one file is provided, tail will print the last ten lines of each file as shown below.

```
# tail access.log error.log
```

Similarly, you can also print the last few lines using the ‘-n‘ option as shown below.

```
# tail -5 access.log
```

```
# cat /etc/passwd
```

```
# echo 'Hi Tec-Team' > 1
# echo 'Keep connected' > 2
# echo 'Share your thought' > 3
# echo 'connect us tec.com@gmail.com' > 4
```

```
# cat 1 2 3 4 > 5
# cat 5
# cat > tec.txt
```

We can have custom end maker for ‘cat’ command. Here it is implemented.

```
# cat > test.txt << end

I am Ankit
Here i am writing this post
Hope your are enjoying
end
```

```
# cat month
# tac month
```

man command

- man command in Linux is used to display the user manual of any command that we can run on the terminal.

- **$ man [COMMAND NAME]**

- **$ man printf**

- **$ man 2 printf**

locate Command

The locate command and find command is used to search a file by name. But, the difference between both commands is that locate command is a background process and searches the file in the database whereas, find command searches in the filesystem. The locate command is much faster than find command.

Using locate Command

Firing locate command to look for a file is pretty easy and straightforward. All you need to do is type:

```
$ locate LAMP-Setup.odt
```

```
$ locate sample.txt
```

2. Limit Search Queries to a Specific Number

You can limit your search returns to a required number to avoid redundancy with your search results using the -n command.

For example, if you want just 20 results from your queries, you can type the following command:

```
$ locate "*.html" -n 20
```

3. Display The Number of Matching Entries

If you want to display the count of all matching entries of file “tec“, use the locate -c command.

```
$ locate -c [tec]*
1550
```

```
$ locate -c [.txt]*
```

It will count files ending with .txt.

Ignore Case Sensitive Locate Outputs. This command is configured to process queries in a case sensitive manner. It means SAMPLE.TXT will show a different result than sample.txt.

```
$ locate -i *SAMPLE.txt*
```

5. Refresh mlocate Database Since locate command relies on a database called mlocate. The said database needs to be updated regularly for the command utility to workefficiently.

To update the mlocate database, you use a utility called updatedb. It should be noted that you will need superuser privileges for this to work properly, is it needs to be executed as root or sudo privileges.

```
$ sudo updatedb
```

rm command

- 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.
- **rm [OPTION]... FILE...**
- **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**

Let us consider 5 files having name a.txt, b.txt and so on till e.txt.

1. -i (Interactive Deletion):

$ rm -i d.txt

Linux rm -r

With rm '-r' option, you can delete a directory having sub directories inside it. So you don't need to delete sub-directories manually.

Syntax:

rm -r <dirname>

Example:

rm -r sandy

m *extension

rm *.txt

1. **rm -f <file name>**

1. **rm -rf <directory name>**

rmdir command

The **rmdir** command removes each directory specified on the command line, if they are empty. That is, each directory removed must contain no files or directories, or it cannot be removed by **rmdir**.

If any specified directory is not empty, **rmdir** will not remove it, and will proceed to try and remove any other directories you specified.

rmdir [**-p**] [**-v** | **--verbose**] [**--ignore-fail-on-non-empty**] *directory* ...

Remove the directory **mydir**, if it's empty.

- **rmdir mydir**
- **rmdir dir1 dir2 dir3**

- Remove the directories **dir1**, **dir2**, and **dir3**, if they are empty. If any are not empty, an error message will be printed for that directory, and the others will be removed.
- **rmdir dir/subdir dir**

- Remove the directory **dir/subdir** if it's empty. Then, remove directory **dir**, if it's empty after **dir/subdir** was removed.

- **rmdir -p dir/subdir**

- Same as the above command. **rmdir** attempts to remove **dir/subdir**, then attempts to remove **dir**.

How to Ignore Non-Empty Directories?

By default, if you are trying to remove a non-empty directory, the rmdir command will produce an error message. To ignore the error message that arises on deleting a non-empty directory, use the –ignore-fail-on-non-empty option.

Syntax:

```
$ rmdir--ignore-fail-on-non-empty test-dir
```

How to Remove Parent Directories Using rmdir command?

rmdir command can also perform its operation over the parent directories. It has the capacity to delete both the directory and its parent directories in one shot. You can make use of this feature with the help of-p option.

Consider the following example.

```
$ rmdir -p test/testdir/
```

mkdir command

mkdir command in Linux allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once as well as set the permissions for the directories. It is important to note that the user executing this command must have enough permissions to create a directory in the parent directory, or he/she may receive a ‘permission denied’ error.

```
mkdir [options...] [directories ...]
```

-v or –verbose: It displays a message for every directory created.
Syntax:

```
mkdir -v [directories]
```

```
Mkdir –v one two three
```

- **-p**: A flag which enables the command to create parent directories as necessary. If the directories exist, no error is specified.

- **mkdir -p first/second/third**

- If the first and second directories do not exist, due to the -p option, mkdir will create these directories for us. If we do not specify the -p option, and request the creation of directories, where parent directory doesn't exist

- **-m**: This option is used to set the file modes, i.e. permissions, etc. for the created directories. The syntax of the mode is the same as the **chmod** command. **Syntax:**

- **mkdir -m a=rwx [directories]**

- **Mkdir –m a=rwx first**

-

cd’ Command

cd‘ (**Change Directory**) command is one of the most important and most widely used command for newbies as well as system administrators. For admins on a headless server, ‘**cd**‘ is the only way to navigate to a directory to check log, execute a program/application/script and for every other task. For newbie it is among those initial commands they make their hands dirty with.

1. Change from current directory to /usr/local.

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

2. Change from current directory to /usr/local/lib using absolute path.

```
avi@tec:/usr/local$ cd /usr/local/lib
```

3. Change from current working directory to /usr/local/lib using relative path.

```
avi@tec:/usr/local$ cd lib
```

4. (a) Switch back to previous directory where you working earlier.

```
avi@tec:/usr/local/lib$ cd –
```

4. (b) Change Current directory to parent directory.

```
avi@tec:/usr/local/lib$ cd ..
```

5. Show last working directory from where we moved (use '–' switch) as shown.

```
avi@tec:/usr/local$ cd --
```

6. Move two directory up from where you are now.

```
avi@tec:/usr/local$ cd ../ ../
```

7. Move to users home directory from anywhere.

```
avi@tect:/usr/local$ cd ~
avi@tec:~$
```

or

```
Cd
```

9. Your present working Directory is "/usr/local/lib/python3.4/dist-packages/ ", change it to "/home/avi/Desktop/ ", in one line command, by moving up in the directory till '/' then using absolute path.

```
avi@tec:/usr/local/lib/python3.4/dist-packages$ cd ../../../../../home/avi/Desktop/
avi@tec~/Desktop$
```

1. Navigate from your current working directory to /etc/v__ _, Oops! You forgot the name of directory and not supposed to use TAB.

```
avi@tec:~$ cd /etc/v*
```

Cut Command Linux command cut is used for text processing. You can use this command to extract portion of text from a file by selecting columns.

```
$ cat test.txt
cat command for file oriented operations.
cp command for copy files or directories.
ls command to list out files and directories with its attributes.
```

Select Column of Characters To extract only a desired column from a file use -c option. The following example displays 2nd character from each line of a file test.txt

```
$ cut -c2 test.txt
```

```
a
p
S
```

2. Select Column of Characters using Range

Range of characters can also be extracted from a file by specifying start and end position delimited with -. The following example extracts first 3 characters of each line from a file called test.txt

```
$ cut -c1-3 test.txt
```

```
cat
cp
ls
```

3. Select Column of Characters using either Start or End Position Either start position or end position can be passed to cut command with -c option. The following specifies only the start position before the ‘-‘. This example extracts from 3rd character to end of each line from test.txt file.

```
$ cut -c3- test.txt
```

```
t command for file oriented operations.
 command for copy files or directories.
 command to list out files and directories with its attributes.
```

The following specifies only the end position after the ‘-‘. This example extracts 8 characters from the beginning of each line from test.txt file.

```
$ cut -c-8 test.txt
```

```
cat comm
cp comma
ls comma
```

The entire line would get printed when you don’t specify a number before or after the ‘-‘ as shown below.

```
$ cut -c- test.txt
cat command for file oriented operations.
cp command for copy files or directories.
ls command to list out files and directories with its attributes.
```

4. Select a Specific Field from a File

Instead of selecting x number of characters, if you like to extract a whole field, you can combine option -f and -d. The option -f specifies which field you want to extract, and the option -d specifies what is the field delimiter that is used in the input file.

The following example displays only first field of each lines from /etc/passwd file using the field delimiter : (colon). In this case, the 1st field is the username. The file

```
$ cut -d':' -f1 /etc/passwd
root
daemon
Bin
```

5. Select Multiple Fields from a File

You can also extract more than one fields from a file or stdout. Below example displays username and home directory of users who has the login shell as “/bin/bash”.

```
$ grep "/bin/bash" /etc/passwd | cut -d':' -f1,6
root:/root
bala:/home/bala
```

To display the range of fields specify start field and end field as shown below. In this example, we are selecting field 1 through 4, 6 and 7

```
$ grep "/bin/bash" /etc/passwd | cut -d':' -f1-4,6,7
```

Ln Command

A symbolic link, also known as a symlink or soft link, is a special type of file that points to another file or directory.

Links Types

There are two types of links in Linux/UNIX systems:

Hard links. You can think a hard link as an additional name for an existing file. Hard links are associating two or more file names with the same inode . You can create one or more hard links for a single file. Hard links cannot be created for directories and files on a different filesystem or partition.

Soft links. A soft link is something like a shortcut in Windows. It is an indirect pointer to a file or directory. Unlike a hard link, a symbolic link can point to a file or a directory on a different filesystem or partition.

How to Use the ln Command

ln is a command-line utility for creating links between files. By default, the ln command creates hard links. To create a symbolic link, use the -s (--symbolic) option.

The ln command syntax for creating symbolic links is as follows:

```
ln -s [OPTIONS] FILE LINK
```

If both the FILE and LINK are given, ln will create a link from the file specified as the first argument (FILE) to the file specified as the second argument (LINK). if only one file is given as an argument or the second argument is a dot (.)

The symbolic_link parameter is optional. If you do not specify the symbolic link, the ln command will create a new link in your current directory:

```
ln -s source_file symbolic_link
ln -s my_file.txt my_link.txt
```

To verify that the symlink was successfully created, use the ls command:

```
ls -l my_link.txt
```

Creating Symlinks To a Directory # The command for creating a symbolic link to a directory is the same as when creating a symbolic link to a file. Specify the directory name as the first parameter and the symlink as the second parameter.

For example, if you want to create a symbolic link from the /mnt/my_drive/movies directory to the ~/my_movies directory you would run:

```
ln -s /mnt/my_drive/movies  ~/my_movies
```

Overwriting Symlinks

```
ln -s my_file.txt my_link.txt
```

To overwrite the destination path of the symlink, use the -f (--force) option.

```
ln -sf my_file.txt my_link.txt
```

Removing Symlinks

```
unlink symlink_to_remove
```

Removing a symbolic link using the rm command is the same as when removing a file:

```
rm symlink_to_remove
```

Gzip Command

- Gzip is one of the most popular compression algorithms that allow you to reduce the size of a file and keep the original file mode, ownership, and timestamp.
- Gzip also refers to the .gz file format and the gzip utility which is used to compress and decompress files.
- Gzip compresses only single files and creates a compressed file for each given file. By convention, the name of a file compressed with Gzip should end with either .gz or .z.
- If you want to compress multiple files or directory into one file, first you need to create a Tar archive and then compress the .tar file with Gzip. A file that ends in .tar.gz or .tgz is a Tar archive compressed with Gzip.
- Gzip is most often used to compress text files, Tar archives, and web pages

```
gzip filename
```

gzip will create a file filename.gz and delete the original file.

By default, gzip keeps the original file timestamp, mode, ownership, and name in the compressed file.

Keep the original file

If you want to keep the input (original) file, use the -k option:

```
gzip -k filename
```

Another option to keep the original file is to use the -c option which tells gzip to write on standard output and redirect the output to a file:

```
gzip -c filename > filename.gz
```

Verbose output

Use the -v option if you want to see the percentage reduction and the names of the files that are being processed:

```
gzip -v filename
```

Compress multiple files

You can also pass multiple files as arguments to the command. For example, to compress the files named file1, file2, file3, you would run the following command:

```
gzip file1 file2 file3
```

The command above will create three compressed files, file1.gz, file2.gz, file3.gz.

gunzip command

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.

gunzip [Option] [archive name/file name] Gunzip is a command-line tool for decompressing Gzip files.

gunzip filename.gz

The command will restore the compressed file to its original name, owner, mode and timestamp. By default, once decompressed, gunzip will remove the compressed file. Use the -k option to keep the file:

gunzip -k filename.gz **gunzip -lv filename**

The gunzip command also accept multiple files as arguments:

```
gunzip file1.gz file2.gz file3.gz
```

To recursively decompresses all files in a given directory, use the -r option:

```
gunzip -r directory
```

List the Compressed File Contents

When used with the -l option, gunzip shows information about the given compressed files:

```
gunzip -l filename.gz
```

Zip and Unzip Commands

Zip is a popular cross-platform command used for compressing and archiving data. Compression saves space by shrinking the size of data while archiving makes the transfer of data easier by combining multiple files or directories into a single file.

Installing Zip and Unzip

```
 sudo apt install zip
sudo apt install unzip
```

ZIP Command in Linux

To create a zip file, you will need to provide the name for the zipped archive and the files that need to be included in the zip.

```
$ zip options zip_file file1 file2...
```

Zip single file

To zip a single file named testfile.txt; to a zip file named test.zip, the command would be:

```
$ zip test.zip testfile.txt
```

To confirm if the zip file has been created, issue the command in the Terminal below:

```
$ ls -l
```

Zip multiple files

You can also zip multiple files with the zip command. Let’s create some files named testfile1.txt, testfile2.txt, testfile3.txt, and testfile4.txt using the following command in the Terminal:

```
$ touch testfile1.txt testfile2.txt testfile3.txt testfile4.txt
```

```
$ zip files.zip testfile1.txt testfile2.txt testfile3.txt testfile4.txt
```

You can also use the wildcard to specify multiple files having the same extension.

```
$ zip files1.zip *.txt
```

Zip a file to a different directory

If you need to zip a file to some directory other than the current directory, use the following syntax:

```
$ zip /path/to/directory.zip_file filename
```

For instance, to zip testfile.txt file to Documents/docs/ directory named as test.zip, the command would be:

```
$ zip Documents/docs/test.zip testfile.txt
```

Add file to an existing zip

You can also add a file to a pre-existing zip file. Use the following syntax to do so:

```
zip -u zip_file filename
```

```
$ zip -u files1.zip testfile4.txt
```

Remove a file from a zip file

Similarly, you can also remove a file from an already existing zip file. Use the following syntax to do so:

```
$ zip -d zip_file filename
```

```
$ zip -d files1.zip testfile1.txt
```

Create a password-protected zip file

You can also create password-protected zip files using the -e option with the zip command as follows:

```
$ zip -e zip_file filename
```

```
$ zip -e files3.zip *.txt
```

Unzip Command in Linux

With the unzip command, you can easily extract content from a zip file. To unzip a file, use the following syntax:

```
$ unzip options zip_file
```

Unzip single zip files

To unzip a single zip file, let’s say test.zip, the command would be:

```
$ unzip test.zip
```

Unzip a file to a different directory

You can also unzip a file to some other directory rather than the current directory using the -d option as follows:

```
$ unzip zip_file -d /path/to/directory
```

Unzip multiple files

Consider there are multiple zip files in a current directory that you want to unzip. You can unzip all of them using a single command as follows:

```
$ unzip ‘*.zip’
```

Tar Command The Linux “tar” stands for tape archive, which is used by a large number of Linux/Unix system administrators to deal with tape drives backup

The tar command is used to rip a collection of files and directories into a highly compressed archive file commonly called tarball or tar, gzip and bzip in Linux.

The tar is the most widely used command to create compressed archive files and that can be moved easily from one disk to another disk or machine to machine.

1. Create tar Archive File in Linux

The below example command will create a tar archive file tecmint-14-09-12.tar for a directory /home/tecmint in the current working directory. See the example command in action.

```
# tar -cvf tec-14-09-12.tar /home/tec/
```

Let’s discuss each option used in the above command to create a tar archive file.

c – Creates a new .tar archive file.

v – Verbosely show the .tar file progress.

f – File name type of the archive file.