Experiment 8

Aim: File system: Comparing Files using diff, cmp, comm

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 preceeded 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*.

Now let's see what it looks like when **diff** tells us that we need to delete a line.

```
$ cat a.txt
Gujarat
Andhra Pradesh
Telangana
Bihar
Uttar pradesh

$ cat b.txt
Gujarat
Andhra Pradesh
Bihar
Uttar pradesh

$ diff a.txt b.txt
3d2
< Telangana
```

Here above output **3d2** means delete line 3rd of first file i.e. *Telangana* so that both the files **sync up** at line 2.

Example

For example, we have two files as file.txt and file1.txt. The data has been inserted into file.txt as shown below –

```
I need to buy apples.
I need to run the laundry.
I need to wash the dog.
I need to get the car detailed.
```

file1.txt contains the data as shown below

```
I need to buy apples.
I need to do the laundry.
I need to wash the car.
I need to get the dog detailed.
```

Use the diff command to compare both the files as shown below –

```
linux@linux:~$ diff /home/linux/Desktop/file.txt /home/linux/Desktop/file1.txt
```

The above command should give the result as shown below –

```
linux@linux:~$ diff /home/linux/Desktop/file.txt /home/linux/Desktop/file1.txt
2,4c2,4
< I need to run the laundry.
< I need to wash the dog.
< I need to get the car detailed. --- > I need to do the laundry.
> I need to wash the car.
```

```
> I need to get the dog detailed.
```

From the output, **2,4c2,4** means "Lines 2 through 4 in the first file needs to be changed in order to match lines 2 through 4 in the second file".

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 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
```

1. **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*/
```

2. **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*/
```

Options for cmp command

1. **-b(print-bytes) :** If you want cmp displays the differing bytes in the output when used with **-b** option.

```
//...cmp command used with -b option...//

$cmp -b file1.txt file2.txt
file1.txt file2.txt differ: 12 byte, line 2 is 154 l 151 i

/* indicating that the difference is in 12 byte ,which is 'l' in file1.txt and 'i' in file2.txt.*/
```

The values 154 and 151 in the above output are the values for these bytes, respectively.

2. -i [bytes-to-be-skipped] : Now, this option when used with cmp command helps to **skip a particular number of initial bytes from both the files** and then after skipping it compares the files. This can be done by specifying the number of bytes as argument to the -i command line option.

```
//...cmp command used with -i option...//

$cmp -i 10 file1.txt file2.txt
$_

/*indicating that both files are identical after 10 bytes skipped from both the files*/
```

Note that in cases like these (where you use -i to skip bytes), the byte at which the comparison begins is treated as byte number zero.

3. -i [bytes to be skipped from first file]: [bytes to be skipped from second file] : This option is very much similar to the above -i [bytes to be skipped] option but with the difference that now it **allows us to input the number of bytes we want to skip** from both the files separately.

```
//...cmp command used with -i option...//

$cmp -i 10:12 file1.txt file2.txt
$_

/*indicating that both files are identical after 10 bytes skipped from first file and 12 bytes skipped from second file*/
```

4. -l option : This option makes the cmp command print byte position and byte value for all differing bytes.

```
//...cmp command used with -l option...//

$cmp -l file1.txt file2.txt
20   12   56
21  124   12
```

```
22  150  124
23  151  150
24  163  151
25   40  163
26  146   40
27  150  151
28   12   24
29  124  145
30  157  163

/*indicating that files are different displaying the position of differing bytes along with the differing bytes in both file*/
```

The first column in the output represents the position (byte number) of differing bytes. The second column represents the byte value of the differing byte in the first file, while the third column represents the byte value of the differing byte in the second file.

5. -s option : This allows you to suppress the output normally produced by cmp command *i.e* it compares two files without writing any messages. This gives an exit value of 0 if the files are identical, a value of 1 if different, or a value of 2 if an error message occurs.

```
//...cmp command used with -s option...//

$cmp -s file1.txt file.txt
1

/*indicating files are different without displaying the differing byte and line*/
```

6. -n [number of bytes to be compared] option : This option allows you to limit the number of bytes you want to compare ,like if there is only need to compare at most 25 or 50 bytes.

```
//...cmp command used with -n option...//

$cmp -n 50 file1.txt file2.txt
$_

/*indicating files are identical for starting50 bytes*/
```

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

The above output contains of three columns where **first column** is separated by zero tabs and contains names only present in file1.txt, second **column** contains names only present in file2.txt and separated by one tab and the **third column** contains names common to both the files and is separated by two tabs from the beginning of the line.
This is the default pattern of the output produced by comm command when no option is used.

Options for comm command:

1. **-1: suppress** first column (lines unique to first file).
2. **-2: suppress** second column (lines unique to second file).
3. **-3: suppress** third column (lines common to both files).

Note: The options 4 to 8 are rarely used but options 1 to 3 are very useful in terms of the desired output user wants.

Using comm with options

1. Using -1,-2 and -3 options: The use of these three options can be easily explained with the help of example:

```
//suppress first column using -1//

$comm -1 file1.txt file2.txt
        Apaar
        Hemant
 Lucky
 Pranjal Thakral

//suppress second column using -2//
$comm -2 file1.txt file2.txt
       Apaar
Ayush Rajput
Deepak
       Hemant

//suppress third column using -3//
$comm -3 file1.txt file2.txt
Ayush Rajput
Deepak
       Lucky
       Pranjal Thakral
```

Note that you can also suppress multiple columns using these options together as:

```
//...suppressing multiple columns...//
```

```
$comm -12 file1.txt file2.txt
Apaar
Hemant

/* using -12 together suppressed both firstand second columns */
```

How to ignore case sensitivity

To ignore case sensitivity when using `comm` pass the `-i` option. This will ignore case sensitivity in the comparison. Consider the two following files that have been sorted.

```
cat words1.txt
Apple
Banana
Orange

cat words2.txt
apple
banana
orange
```

If these files are compared using `comm` there will be nothing common to these two files.

```
comm words1.txt words2.txt
Apple
Banana
Orange
        apple
        banana
        orange
```

If case sensitivity is ignore with `-i` option the words will be shown as common to both files. Note that the uppercase words is shown as it is from the first file.

```
comm -i words1.txt words2.txt
            Apple
            Banana
```

```
Orange
```

Assignment:

1. **What is byte value for a byte?**
2. **What is the difference between diff and cmp command?**

Experiment 9

Aim: Compressing files: tar, gzip, bzip2, compress, uncompress, file.

Compressing Files at the Shell Prompt

Linux provides the bzip2, gzip, and zip tools for compression from a shell prompt. The bzip2 compression tool is recommended because it provides the most compression and is found on most UNIX-like operating systems. The gzip compression tool can also be found on most UNIX-like operating systems. To transfer files between Linux and other operating system such as MS Windows, use zip because it is more compatible with the compression utilities available for Windows.

Compression Tool	File Extension	Decompression Tool
bzip2	.bz2	bunzip2
gzip	.gz	gunzip
zip	.zip	unzip

Compression Tools

By convention, files compressed with bzip2 are given the extension .bz2, files compressed with gzip are given the extension .gz, and files compressed with zip are given the extension .zip.

Files compressed with bzip2 are uncompressed with bunzip2, files compressed with gzip are uncompressed with gunzip, and files compressed with zip are uncompressed with unzip.

Bzip2 and Bunzip2

To use bzip2 to compress a file, enter the following command at a shell prompt:

```
bzip2 filename
```

The file is compressed and saved as filename.bz2.

To expand the compressed file, enter the following command:

```
bunzip2 filename.bz2
```

The filename.bz2 compressed file is deleted and replaced with filename.

You can use bzip2 to compress multiple files and directories at the same time by listing them with a space between each one:

```
bzip2 filename.bz2 file1 file2 file3 /usr/work/school
```

The above command compresses file1, file2, file3, and the contents of the /usr/work/school/ directory (assuming this directory exists) and places them in a file named filename.bz2.

For more information, enter man bzip2 and man bunzip2 at a shell prompt to read the man pages for bzip2 and bunzip2.

Gzip and Gunzip

To use gzip to compress a file, enter the following command at a shell prompt:

```
gzip filename
```

The file is compressed and saved as filename.gz.

To expand the compressed file, enter the following command:

```
gunzip filename.gz
```

The filename.gz compressed file is deleted and replaced with filename.

Compress Every File in a Folder and Subfolders

You can compress every file in a folder and its subfolders by using the following command:

```
gzip -r foldername
```

This doesn't create one file called *foldername.gz*. Instead, it traverses the directory structure and compresses each file in that folder structure.

You can use gzip to compress multiple files and directories at the same time by listing them with a space between each one:

```
gzip -r filename.gz file1 file2 file3 /usr/work/school
```

The above command compresses file1, file2, file3, and the contents of the /usr/work/school/ directory (assuming this directory exists) and places them in a file named filename.gz.

Compress file with gzip by specifying compression level

Another notable advantage of gzip is it supports compression level. It supports 3 compression levels as given below.

- 1 – Fastest (Worst)
- 9 – Slowest (Best)
- 6 – Default level

To compress a file named ostechnix.txt, replacing it with a gzipped compressed version with best compression level, we use:

```
$ gzip -9 ostechnix.txt
```

To get minimum compression at the fastest speed run the following command:

```
gzip -1 filename
```

To get maximum compression at the slowest speed run the following command:

```
gzip -9 filename
```

You can vary the speed and compression level by picking different numbers between 1 and 9.

For more information, enter man gzip and man gunzip at a shell prompt to read the man

pages for gzip and gunzip.

Zip and Unzip

To compress a file with zip, enter the following command:

```
zip -r filename.zip filesdir
```

In this example, filename.zip represents the file you are creating and filesdir represents the directory you want to put in the new zip file. The -r option specifies that you want to include all files contained in the filesdir directory *recursively*.

To extract the contents of a zip file, enter the following command:

```
unzip filename.zip
```

You can use zip to compress multiple files and directories at the same time by listing them with a space between each one:

```
zip -r filename.zip file1 file2 file3 /usr/work/school
```

The above command compresses file1, file2, file3, and the contents of the /usr/work/school/ directory (assuming this directory exists) and places them in a file named filename.zip.

For more information, enter man zip and man unzip at a shell prompt to read the man pages for zip and unzip.

compress command in Linux with examples

compress command is used to reduce the file size. After compression, the file will be available with an added *.Z* extension. File permissions will still remain the same as before using *compress* command. This command uses the adaptive Lempel-Ziv coding and it ignores the symbolic links. And also this command is part of *ncompress* package, which contains utilities for fast compression and decompression.
Note: If no files are specified then the standard input is compressed to the standard output.
Syntax:

```
compress [OPTION]... [FILE]...
```

Force Compression

When the file already exist, compress command will prompt user before overwriting as shown below.

```
# compress users.txt

users.txt.Z already exists.

Do you wish to overwrite users.txt.Z (y or n)? n

users.txt.Z not overwritten
```

However, the above is not very helpful, when you are using compress inside a shell script, and when you know, it is Ok to overwrite an existing file. In that case, you can use -f option to force the compress command to overwrite the output file without prompting user as shown below.

```
# compress -f users.txt
```

Uncompressing a File: uncompress Command

The uncompress command restores a compressed file to an uncompressed state.

```
$ uncompress options filename
```

To uncompress the test.tar.Z file and restore it to the test.tar file, enter the following command:

```
# uncompress -v test.tar.Z
test.tar.Z:  -- replaced with test.tar
```

The **-v** option displays additional messages about the action being performed.

Archiving Files at the Shell Prompt

A tar file is a collection of several files and/or directories in one file. This is a good way to create backups and archives.

Some of tar's options include:

- -c — create a new archive

- -f — when used with the -c option, use the filename specified for the creation of the tar file; when used with the -x option, unarchive the specified file
- -t — show the list of files in the tar file
- -v — show the progress of the files being archived
- -x — extract files from an archive
- -z — compress the tar file with gzip
- -j — compress the tar file with bzip2

To create a tar file, enter:

```
tar -cvf filename.tar directory/file
```

In this example, filename.tar represents the file you are creating and directory/file represents the directory and file you want to put in the archived file.

You can tar multiple files and directories at the same time by listing them with a space between each one:

```
tar -cvf filename.tar /home/mine/work  /home/mine/school
```

The above command places all the files in the work and the school subdirectories of /home/mine in a new file called filename.tar in the current directory.

To list the contents of a tar file, enter:

```
tar -tvf filename.tar
```

To extract the contents of a tar file, enter:

```
tar -xvf filename.tar
```

This command does not remove the tar file, but it places copies of its unarchived contents in the current working directory, preserving any directory structure that the archive file used. For example, if the tarfile contains a file called bar.txt within a directory called foo/, then extracting the archive file results in the creation of the directory foo/ in your current working directory with the file bar.txt inside of it.

Remember, the tar command does not compress the files by default. To create a tarred and bzipped compressed file, use the -j option:

```
tar -cjvf filename.tbz file
```

tar files compressed with bzip2 are conventionally given the extension .tbz; however, sometimes users archive their files using the tar.bz2 extension.

The above command creates an archive file and then compresses it as the file filename.tbz. If you uncompress the filename.tbz file with the bunzip2 command, the filename.tbz file is removed and replaced with filename.tar.

You can also expand and unarchive a bzip tar file in one command:

```
tar -xjvf filename.tbz
```

To create a tarred and gzipped compressed file, use the -z option:

```
tar -czvf filename.tgz file
```

tar files compressed with gzip are conventionally given the extension .tgz.

This command creates the archive file filename.tar and compresses it as the file filename.tgz. (The file filename.tar is not saved.) If you uncompress the filename.tgz file with thegunzip command, the filename.tgz file is removed and replaced with filename.tar.

You can expand a gzip tar file in one command:

```
tar -xzvf filename.tgz
```

What is an Archive file?
An Archive file is a file that is composed of one or more files along with metadata. Archive files are used to collect multiple data files together into a single file for easier portability and storage, or simply to compress files to use less storage space.

Examples:

1. Creating an uncompressed tar Archive using option -cvf : This command creates a tar file called file.tar which is the Archive of all .c files in current directory.

```
$ tar -cvf file.tar *.c
```

Output :

```
os2.c

os3.c

os4.c
```

2. Extracting files from Archive using option -xvf : This command extracts files from Archives.

```
$ tar -xvf file.tar
```

Output :

```
os2.c

os3.c

os4.c
```

3. gzip compression on the tar Archive, using option -z : This command creates a tar file called file.tar.gz which is the Archive of .c files.

```
$ tar -cvzf file.tar.gz *.c
```

4. Extracting a gzip tar Archive *.tar.gz using option -xvzf : This command extracts files from tar archived file.tar.gz files.

```
$ tar xvzf file.tar.gz
```

5. Creating compressed tar archive file in Linux using option -j : This command compresses and creates archive file less than the size of the gzip. Both compress and decompress takes more time then gzip.

```
$ tar cvfj file.tar.tbz example.cpp
```

Output :

```
$tar cvfj file.tar.tbz example.cpp

example.cpp
```

6. We can pass a file name as an argument to search a tarfile : This command views the archived files along with their details.

```
$ tar tvf file.tar filename
```

7. Viewing the Archive using option -tvf

```
$ tar tvf file.tar
```

Output :

```
-rwxrwxrwx root/root      191 2017-09-17 02:20 os2.c

-rwxrwxrwx root/root      218 2017-09-17 02:20 os3.c

-rwxrwxrwx root/root      493 2017-09-17 02:20 os4.c
```

Both Linux and UNIX include various commands for Compressing and decompresses (read as expand compressed file). To compress files you can use gzip, bzip2 and zip commands. To expand compressed file (decompresses) you can use and gzip -d, bunzip2 (bzip2 -d), unzip commands.

Compressing files

Syntax	Description	Example(s)
gzip {filename}	Gzip compress the size of the given files using Lempel-Ziv coding (LZ77). Whenever possible, each file is replaced by one with the extension .gz.	gzip mydata.doc gzip *.jpg ls -l
bzip2 {filename}	bzip2 compresses files using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding. Compression is generally considerably better than that achieved by bzip command (LZ77/LZ78-based compressors). Whenever possible, each file is replaced by one with the extension .bz2.	bzip2 mydata.doc bzip2 *.jpg ls -l
zip {.zip-filename} {filename-to-compress}	zip is a compression and file packaging utility for Unix/Linux. Each file is stored in single .zip {.zip-filename} file with the extension .zip.	zip mydata.zip mydata.doc zip data.zip *.doc ls -l
tar -zcvf {.tgz-file} {files} tar -jcvf {.tbz2-file} {files}	The GNU tar is archiving utility but it can be use to compressing large file(s). GNU tar supports both archive compressing through gzip and bzip2. If you have more than 2 files then it is recommended to use tar instead of gzip or bzip2. **-z**: use gzip compress **-j**: use bzip2 compress	tar -zcvf data.tgz *.doc tar -zcvf pics.tar.gz *.jpg *.png tar -jcvf data.tbz2 *.doc ls -l

Decompressing files

Syntax	Description	Example(s)
gzip -d {.gz file} gunzip {.gz file}	Decompressed a file that is created using gzipcommand. File is restored to their original form using this command.	gzip -d mydata.doc.gz gunzip mydata.doc.gz
bzip2 -d {.bz2-file} bunzip2 {.bz2-file}	Decompressed a file that is created using bzip2command. File is restored to their original form using this command.	bzip2 -d mydata.doc.bz2 gunzip mydata.doc.bz2
unzip {.zip file}	Extract compressed files in a ZIP archive.	unzip file.zip unzip data.zip resume.doc
tar -zxvf {.tgz-file} tar -jxvf {.tbz2-file}	Untar or decompressed a file(s) that is created using tar compressing through gzip and bzip2filter	tar -zxvf data.tgz tar -zxvf pics.tar.gz *.jpg tar -jxvf data.tbz2

List the contents of an archive/compressed file

Some time you just wanted to look at files inside an archive or compressed file. Then all of the above command supports file list option.

Syntax	Description	Example(s)
gzip -l {.gz file}	List files from a GZIP archive	gzip -l mydata.doc.gz
unzip -l {.zip file}	List files from a ZIP archive	unzip -l mydata.zip
tar -ztvf {.tar.gz} tar -jtvf {.tbz2}	List files from a TAR archive	tar -ztvf pics.tar.gz tar -jtvf data.tbz2

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.

Linux file command

file command is used to determine the file type. It does not care about the extension used for file. It simply uses file command and tell us the file type. It has several options.

Syntax:

1. file **<filename>**

Linux File Command Options

Option	Function
file -s	Used for special files.
file *	Used to list types of all the files.
file /directory name/*	Used to list types of all the files from mentioned directory.
file [range]*	It will list out all the files starting from the alphabet present within the given range.

directoryname/* option : This is used to display all files filetypes in particular directory.

Syntax:

```
file directoryname/*
```

Example:

```
file work/*
```

[range]* option: To display the file type of files in specific range.

Syntax:

```
file [range]*
```

Example:

```
file [a-z]*

file [a-e]*
```

Assignment:

1. **Find out difference between gzip bzip compress and zip.**
2. **Find more commands used for compression of data in files.**
3. **Identify if a file is compressed using gzip command can be uncompressed using bunzip2 or uncmpress or any other command except gunzip.**