Useful Commands

Commands for directory files:

$mkdir a b c	//for creation of 3 subdirectories in the current directory
$cd a	// to move to subdirectory a
$cd /	// to move to root directory
$cd ..	// to move to parent of current directory or to move one level up in directory tree structure
$rmdir a b	// to delete the directories a & b in a signle attempt provided both directories are empty
$rm -r a	// to remove entire subdirectory tree structure having directory “a” as root
$pwd	//display current path

Directory file has only two information: file names along with their i-node numbers which exist as subdirectories or regular, device or any other type of files.

The i-node number of any file entry e.g. “abc” in directory points to a i-node table that contains information about file “abc”. This information includes: owner name, group name, date of creation/modification or last access, the rights “r w & x” which pwner, group or others enjoy over that file.

Commands for regular files:

$cat abc	//lists or displays the contents of file “abc”
$cat file1 file2 file3	//displays the contents of files- file1, file2 and file3 one-by-one (scrolled vertically)
$cat file1 file2 file3>file4	//copies the contents of files- file1, file2 and file3 one-by-one in file4. so file4 shall have concatenated contents of all three files. . output of command cat is redirected to file4 as we are using output “>” redirection meta character.
$cat > file5	// to create file “file5” as after this command cursor goes to next line so that we can type contents from console (input file or keyboard). At lat use “^d” to mark end of input file. This will now transfer all typed contents to file “file5”. Actually, absence of file name between cat and “>” assumes std input file which is keyboard.
$rm file5	// to delete the regular file “file5”. Explore other additional options which can be used with rm. e.g. rm -i
$cp file1 file2	//copy “file1” contents to new file “file2”
$mv file1 file2	//move contents from “file1” to “file2” or in other language, “file1” is renamed as “file2”
$sort file1	//sort contents of “file1”. Explore other available options of sort. e.g. sort -n
$tr ‘[a-z]’ ‘[A-Z]’ < file1	//convert contents of “file1” to capital. Input for tr command is taken from file1 as we are using input “<” redirection meta character.

examples:

```
user@ubuntu:~$ cat emp
12|amit singh|manager|50000
11|alok singh|programmer|64000
10|zafar alam|operator|75000
19|sooraj dubey|system manager|55000
```

Use of cut command to cut the file vertically field wise

```
user@ubuntu:~$ cut -d"|" -f2,4 emp
amit singh|50000
alok singh|64000
zafar alam|75000
sooraj dubey|55000
```

Use of cut command to cut the file vertically character wise

```
user@ubuntu:~$ cut -c2,4 emp          // cut character 2 & 4 across “emp” vertically
2a
1a
0z
9s
user@ubuntu:~$ cut -c2-7 emp          // cut character 2 to 7 across “emp” vertically
2|amit
1|alok
0|zafa
9|soor
```

Use of cut command to cut the string

```
user@ubuntu:~$ echo "madan mohan"|cut -c1-7 // cut character 1 to 7 from previous string
madan m
```

Use of Paste command to join two files vertically

File1

```
user@ubuntu:~$ cat emp
12|amit singh|manager|50000
11|alok singh|programmer|64000
10|zafar alam|operator|75000
19|sooraj dubey|system manager|55000
```

File2

```
user@ubuntu:~$ cat dept
1|CE department|Dr. SR Chaurasia
2|CSE Department|Dr. Udai Shanker
3|ITCA Department|Dr. S. Prakash
4|EE Department| Dr. V.K. Giri
5|ECE Department|Dr. Chauhan
user@ubuntu:~$ paste -d"|" emp dept
12|amit singh|manager|50000|1|CE department|Dr. SR Chaurasia
11|alok singh|programmer|64000|2|CSE Department|Dr. Udai Shanker
10|zafar alam|operator|75000|3|ITCA Department|Dr. S. Prakash
19|sooraj dubey|system manager|55000|4|EE Department| Dr. V.K. Giri
|5|ECE Department|Dr. Chauhan
```

Use of head command

```
user@ubuntu:~$ head -3 emp            //display first 3 lines from top
12|amit singh|manager|50000
11|alok singh|programmer|64000
10|zafar alam|operator|75000
```

Use of tail command

```
user@ubuntu:~$ tail -3 emp            //display last 3 lines from bottom
11|alok singh|programmer|64000
10|zafar alam|operator|75000
19|sooraj dubey|system manager|55000
user@ubuntu:~$ tail -n +3 emp         //display all lines starting from line number 3
10|zafar alam|operator|75000
19|sooraj dubey|system manager|55000
```

Use of grep command

```
user@ubuntu:~$ grep "manager" emp
12|amit singh|manager|50000
19|sooraj dubey|system manager|55000
```

File Permissions Commands: The *chmod* and *chown* commands are used to control access to files in UNIX and Linux systems.

- **chown :** Used to change the owner of the file. **e.g. chown master file1.txt** //makes master as a owner of file1.txt
- **chgrp :** Used to change the group owner of the file.**e.g. chgrp sunil abc.txt** //changes group of abc.txt from existing group to newgroup “sunil”
- **chmod :** Used to modify the access/permission of a user. It stands for “change mode”, because the nine security characters are collectively called the security “mode” of the file.

1. The first argument you give to the “chmod” command is ‘u’, ‘g’, ‘o’. We use:
   u for user
   g for group
   o for others,
   you can also use a combination of them (u,g,o).
   This specifies which of the three groups you want to modify.

2. After this use
   a '+' for adding
   a '-' for removing
   and a "=" for assigning a permission.
3. Then specify the permission r,w or x you want to change.
   Here also you can use a combination of r,w,x.
   This specifies which of the three permissions "rwx" you want to modify
4. use can use commas to modify more permissions
5. Finally, the name of the file whose permission you are changing

An example will make this clearer.
For example, if you want to give "execute" permission to the world ("other") for file "xyz.txt", you would start by typing

```
chmod o
```

Now you would type a '+' to say that you are "adding" a permission.

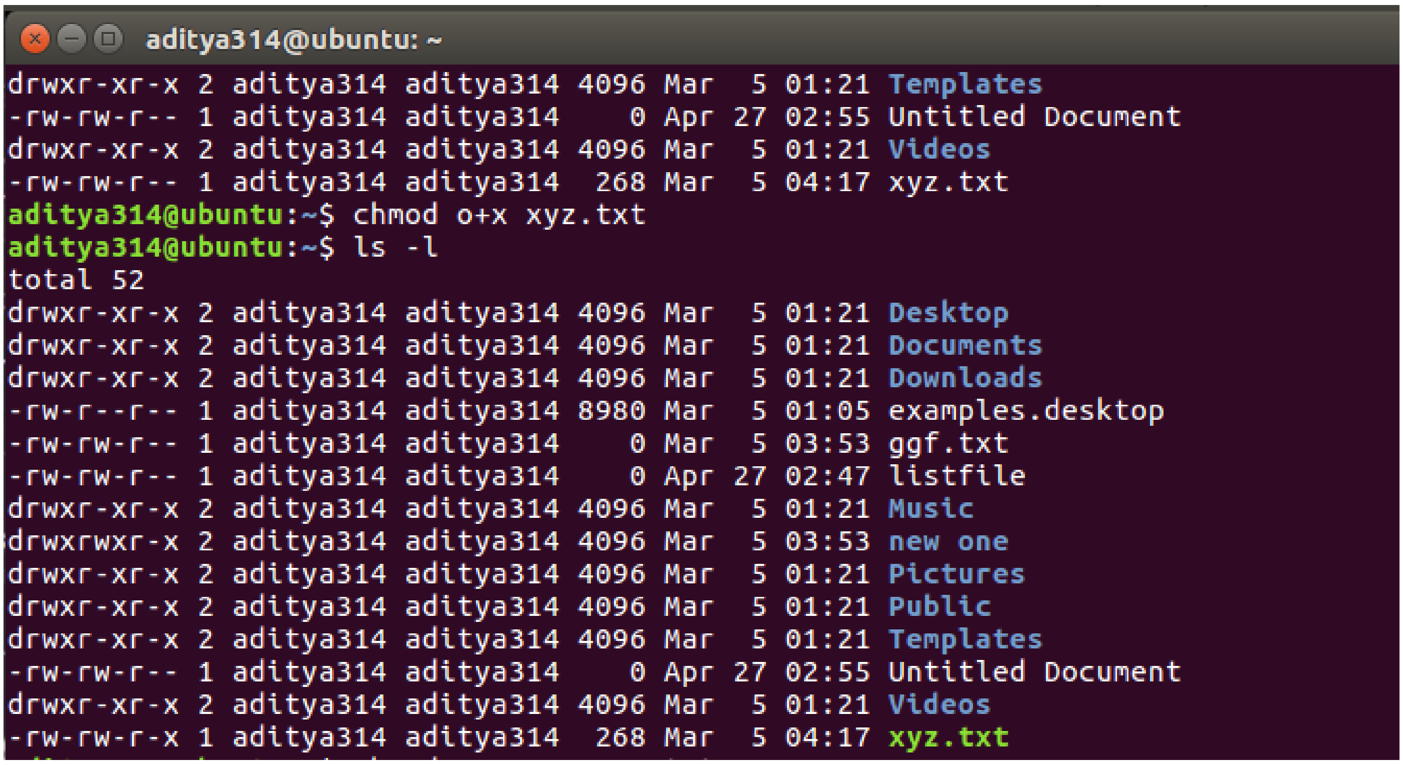
```
chmod o+
```

Then you would type an 'x' to say that you are adding "execute" permission.

```
chmod o+x
```

Finally, specify which file you are changing.

```
chmod o+x xyz.txt
```

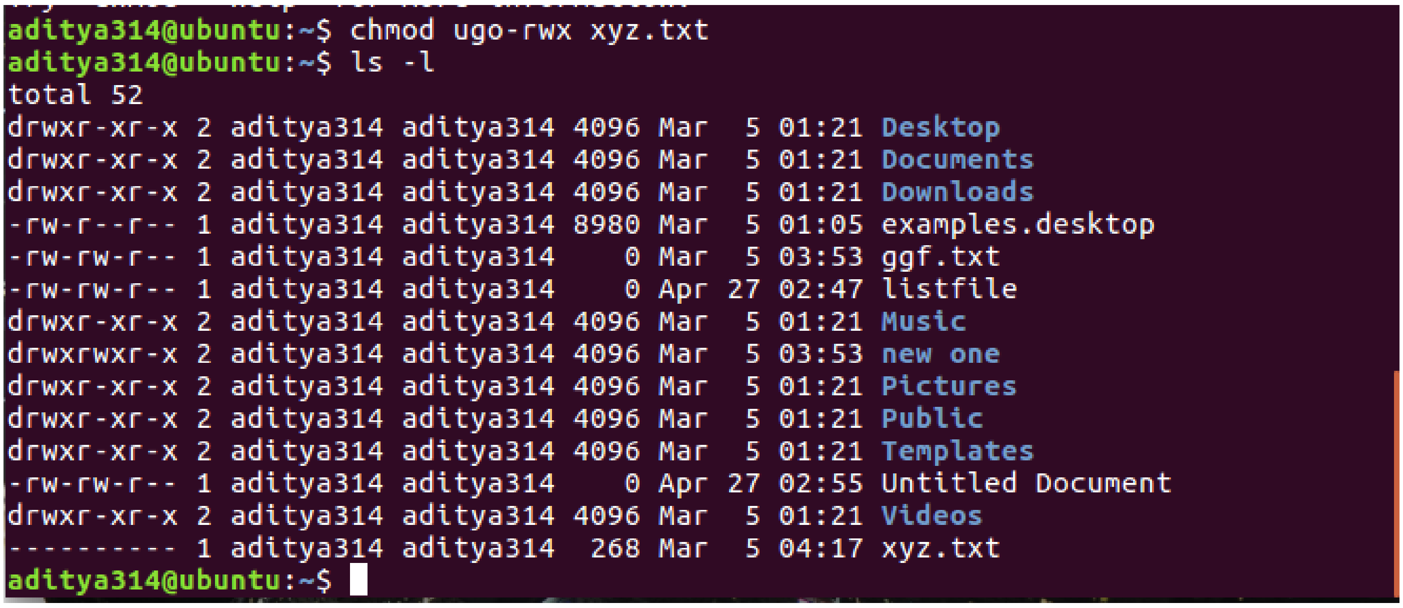


```
aditya314@ubuntu: ~
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Templates
-rw-rw-r-- 1 aditya314 aditya314    0 Apr 27 02:55 Untitled Document
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Videos
-rw-rw-r-- 1 aditya314 aditya314  268 Mar  5 04:17 xyz.txt
aditya314@ubuntu:~$ chmod o+x xyz.txt
aditya314@ubuntu:~$ ls -l
total 52
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Desktop
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Documents
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Downloads
-rw-r--r-- 1 aditya314 aditya314 8980 Mar  5 01:05 examples.desktop
-rw-rw-r-- 1 aditya314 aditya314    0 Mar  5 03:53 ggf.txt
-rw-rw-r-- 1 aditya314 aditya314    0 Apr 27 02:47 listfile
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Music
drwxrwxr-x 2 aditya314 aditya314 4096 Mar  5 03:53 new one
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Pictures
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Public
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Templates
-rw-rw-r-- 1 aditya314 aditya314    0 Apr 27 02:55 Untitled Document
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Videos
-rw-rw-r-x 1 aditya314 aditya314  268 Mar  5 04:17 xyz.txt
```

You can also change multiple permissions at once. For example, if you want to take all permissions away from everyone, you would type

```
chmod ugo-rwx xyz.txt
```

The code above revokes all the read(r), write(w) and execute(x) permission from all user(u), group(g) and others(o) for the file xyz.txt which results to this.



```
aditya314@ubuntu:~$ chmod ugo-rwx xyz.txt
aditya314@ubuntu:~$ ls -l
total 52
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Desktop
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Documents
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Downloads
-rw-r--r-- 1 aditya314 aditya314 8980 Mar  5 01:05 examples.desktop
-rw-rw-r-- 1 aditya314 aditya314    0 Mar  5 03:53 ggf.txt
-rw-rw-r-- 1 aditya314 aditya314    0 Apr 27 02:47 listfile
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Music
drwxrwxr-x 2 aditya314 aditya314 4096 Mar  5 03:53 new one
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Pictures
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Public
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Templates
-rw-rw-r-- 1 aditya314 aditya314    0 Apr 27 02:55 Untitled Document
drwxr-xr-x 2 aditya314 aditya314 4096 Mar  5 01:21 Videos
---------- 1 aditya314 aditya314  268 Mar  5 04:17 xyz.txt
aditya314@ubuntu:~$
```

Another example can be this:

```
chmod ug+rw,o-x abc.mp4
```

The code above adds read(r) and write(w) permission to both user(u) and group(g) and revoke execute(x) permission from others(o) for the file abc.mp4.

Something like this:

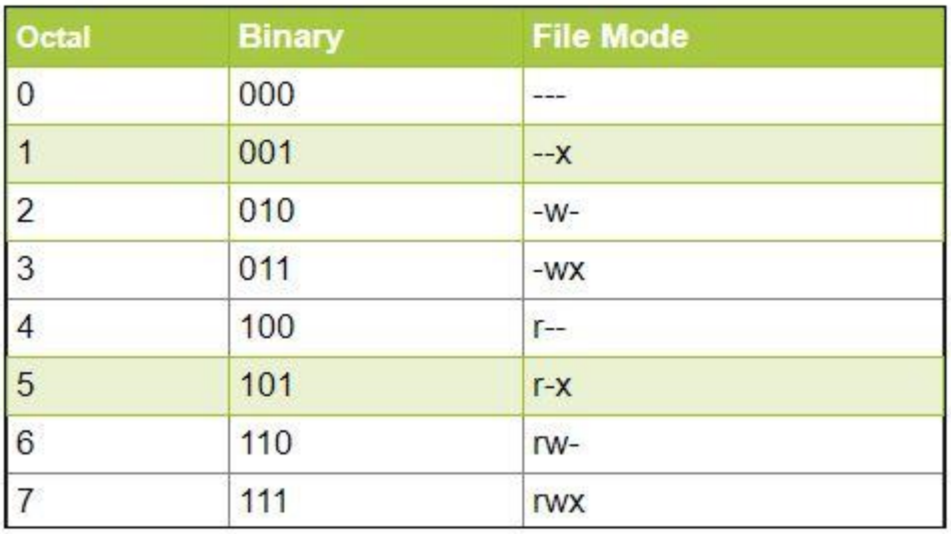
```
chmod ug=rx,o+r abc.c
```

assigns read(r) and execute(x) permission to both user(u) and group(g) and add read permission to others for the file abc.c.

There can be numerous combinations of file permissions you can invoke, revoke and assign. You can try some in your linux system.

The octal notations

You can also use octal notations like this.



Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Using the octal notations table instead of ‘r’, ‘w’ and ‘x’. Each digit octal notiation can be used of either of the group ‘u’,’g’,’o’.

So, the following work the same.

```
chmod ugo+rwx [file_name]
```

```
chmod 777 [file_name]
```

Both of them provides full read write and execute permission (code=7) to all the group.

Same is the case with this..

```
chmod u=r,g=wx,o=rx [file_name]
```

```
chmod 435 [file_name]
```

Both the codes give read (code=4) permission to user, write and execute (code=3) for group and read and execute (code=5) for others.

And even this…

```
chmod 775 [file_name]
```

```
chmod ug+rwx,o=rx [file_name]
```

Both the commands give all permissions (code=7) to user and group, read and execute (code=5) for others.

SetUID, SetGID, and Sticky Bits in Linux File Permissions

As explained in the article **Permissions in Linux**, Linux uses a combination of bits to store the permissions of a file. We can change the permissions using the `chmod` command, which essentially changes the 'r', 'w' and 'x' characters associated with the file.

Further, the ownership of files also depends on the `uid` (user ID) and the `gid` (group ID) of the creator. Similarly, when we launch a process, it runs with the `uid` and `gid` of the user who launched it.

1. The `setuid` bit

This bit is present for files which have executable permissions. The `setuid` bit simply indicates that when running the executable, it will set its permissions to that of the user who created it (owner), instead of setting it to the user who launched it. Similarly, there is a `setgid` bit which does the same for the `gid`.

- To locate the `setuid`, look for an 's' instead of an 'x' in the executable bit of the file permissions.
- An example of an executable with `setuid` permission is `passwd`, as can be seen in the following output.

  ```
  ls -l /etc/passwd
  ```

  This returns the following output:

  ```
  -rwsr-xr-x root root 2447 Aug 29  2018 /etc/passwd
  ```

As we can observe, the 'x' is replaced by an 's' in the user section of the file permissions.

- To set the `setuid` bit, use the following command.

  ```
  chmod u+s
  ```

- To remove the `setuid` bit, use the following command.

  ```
  chmod u-s
  ```

2. The `setgid` bit

The `setgid` affects both files as well as directories. When used on a file, it executes with the privileges of the group of the user who owns it instead of executing with those of the group of the user who executed it.

When the bit is set for a directory, the set of files in that directory will have the same group as the group of the parent directory, and not that of the user who created those files. This is used for file sharing since they can be now modified by all the users who are part of the group of the parent directory.

- To locate the `setgid` bit, look for an ‘s’ in the group section of the file permissions, as shown in the example below.
  ```
  -rwxrwsr-x root root 1427 Aug 2 2019 sample_file
  ```
- To set the `setgid` bit, use the following command.
  ```
  chmod g+s
  ```
- To remove the `setgid` bit, use the following command.
  ```
  chmod g-s
  ```

Security Risks

The `setuid` bit is indeed quite useful in various applications, however, the executable programs supporting this feature should be carefully designed so as to not compromise on any security risks that follow, such as buffer overruns and path injection. If a vulnerable program runs with root privileges, the attacker could gain root access to the system through it. To dodge such possibilities, some operating systems ignore the `setuid` bit for executable shell scripts.

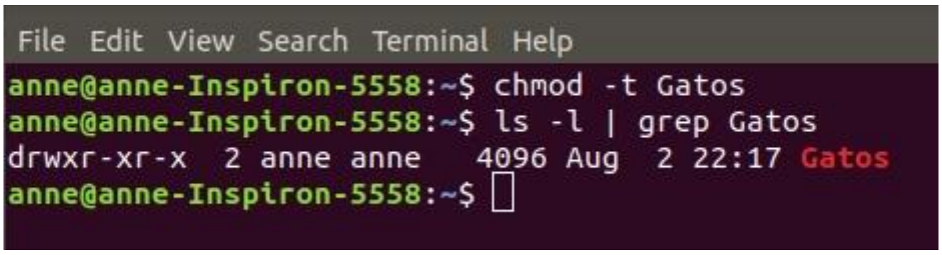
3. The sticky bit

The sticky bit was initially introduced to ‘stick’ an executable program’s text segment in the swap space even after the program has completed execution, to speed up the subsequent runs of the same program. However, these days the sticky bit means something entirely different.

When a directory has the sticky bit set, its files can be deleted or renamed only by the file owner, directory owner and the root user. The command below shows how the sticky bit can be set.

```
chmod +t
```

Simply look for a ‘t’ character in the file permissions to locate the sticky bit. The snippet below shows how we can set the sticky bit for some directory “Gatos”, and how it prevents the new user from deleting a file in the directory.



```
anne@anne-Inspiron-5558:~$ chmod -t Gatos
anne@anne-Inspiron-5558:~$ ls -l | grep Gatos
drwxr-xr-x  2 anne anne   4096 Aug  2 22:17 Gatos
anne@anne-Inspiron-5558:~$ 
```