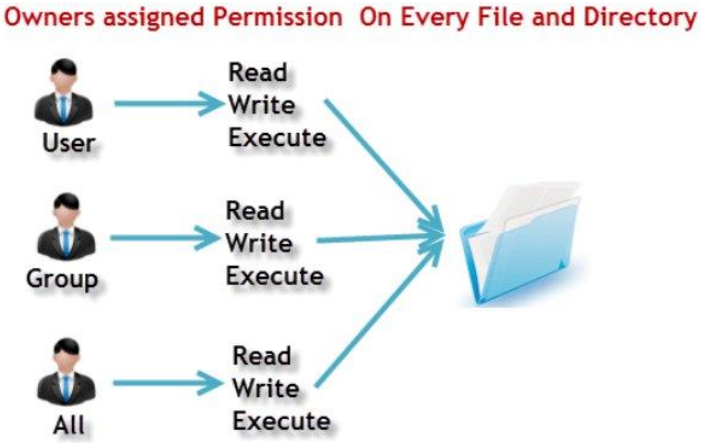
Linux File Permissions

Every file and directory in your UNIX/Linux system has following 3 permissions defined for all the 3 owners discussed above.

- **Read:** This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists its content.
- **Write:** The write permission gives you the authority to modify the contents of a file. The write permission on a directory gives you the authority to add, remove and rename files stored in the directory. Consider a scenario where you have to write permission on file but do not have write permission on the directory where the file is stored. You will be able to modify the file contents. But you will not be able to rename, move or remove the file from the directory.
- **Execute:** In Windows, an executable program usually has an extension ".exe" and which you can easily run. In Unix/Linux, you cannot run a program unless the execute permission is set. If the execute permission is not set, you might still be able to see/modify the program code(provided read & write permissions are set), but not run it.



.guru99.com/images/PermissionsConcept.png File Permissions in Linux/Unix

The speaker was discussing (write the main keyword from the lecture).

He/she mentioned that (write a combination of two/three words from the lecture).

He/She then discussed (write a combination of two/three words from the lecture).

Furthermore, he/she talked about (write a combination of two/three words from the lecture).

Finally, the speaker concluded that (write a combination of two/three words from the lecture).

The characters are pretty easy to remember.

r = read permission
w = write permission
x = execute permission
– = no permission

Let us look at it this way.

The first part of the code is **'rw-'**. This suggests that the owner 'Home' can:

- Read the file
- Write or edit the file
- He cannot execute the file since the execute bit is set to '-'.

By design, many Linux distributions like Fedora, CentOS, Ubuntu, etc. will add users to a group of the same group name as the user name. Thus, a user 'tom' is added to a group named 'tom'.

The second part is **'rw-'.** It for the user group 'Home' and group-members can:

- Read the file
- Write or edit the file

The third part is for the world which means any user. It says **'r–'.** This means the user can only:

- Read the file

Group

Changing file/directory permissions in Linux Using 'chmod' command

Say you do not want your colleague to see your personal images. This can be achieved by changing file permissions.

We can use the '**chmod'** command which stands for 'change mode'. Using the command, we can set permissions (read, write, execute) on a file/directory for the owner, group and the world.

Syntax:

```
chmod permissions filename
```

There are 2 ways to use the command –

1. **Absolute mode**
2. **Symbolic mode**

Absolute(Numeric) Mode in Linux

In this mode, file **permissions are not represented as characters but a three-digit octal number**.

The table below gives numbers for all for permissions types.

Number	Permission Type	Symbol
0	No Permission	—
1	Execute	–x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r–
5	Read + Execute	r-x
6	Read +Write	rw-
7	Read + Write +Execute	rwx

Changing Ownership and Group in Linux

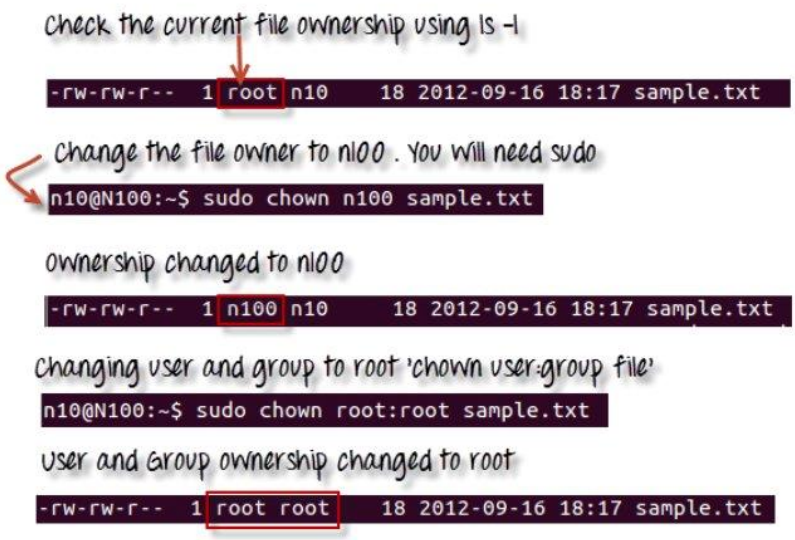
For changing the ownership of a file/directory, you can use the following command:

```
chown user filename
```

In case you want to change the user as well as group for a file or directory use the command

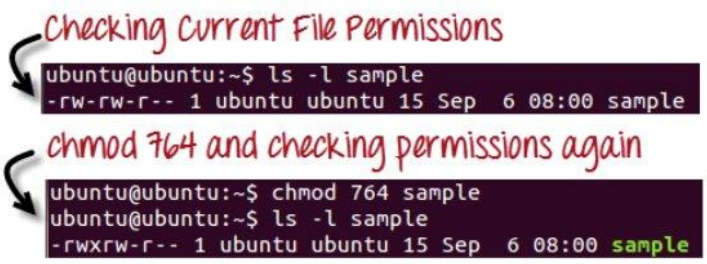
```
chown user:group filename
```

Let's see this in action

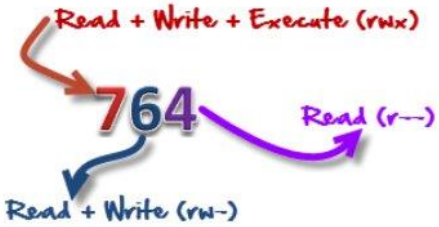


:om/images/chown_comm(1).png

Let's see the chmod permissions command in action.



In the above-given terminal window, we have changed the permissions of the file 'sample to '764'.



'764' absolute code says the following:

- Owner can read, write and execute
- Usergroup can read and write
- World can only read

Let's see file permissions in Linux with examples:

ls – l on terminal gives

```
ls - l
```

Here, we have highlighted **'-rw-rw-r–'**and this weird looking code is the one that tells us about the Unix permissions given to the owner, user group and the world.

Here, the first '–' implies that we have selected a file.p>

Else, if it were a directory, **d** would have been shown.

The characters are pretty easy to remember.

r = read permission

.guru99.com/images/Directory.png