Linux File Permissions and Ownership Explained with Examples - Linux Handbook

About Abhishek

Learn everything you need to know about Linux file permissions. Also learn how to change the file permissions and ownership.

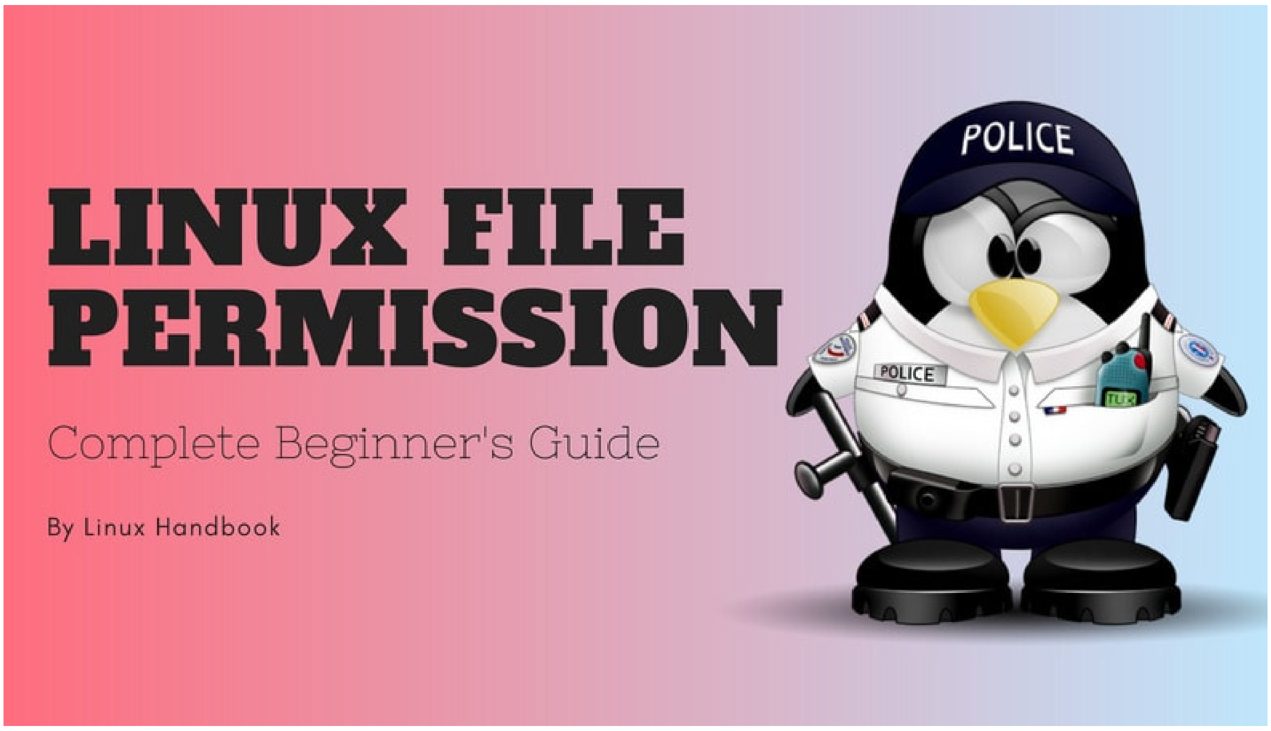
By design, Linux is a multi-user operating system. In an enterprise system, there would be multiple users accessing the same system. But if any user could access and modify all files belonging to other users or system files, this would certainly be a security risk.

This is why UNIX and thus Linux (Linux is a Unix-like system) has built-in security measure in place. This ensures that a file or directory can be accessed, modified or executed by only desired users.

Which file would be accessed by which user is decided by two factors in Linux:

- File ownership
- File permission

Understanding file ownership and permission is crucial for a Linux user. I'll explain these terms in detail here.



File ownership in Linux

Note: I might use the term file here but it is applicable to directories as well. I guess you know that directories are files anyway.

Every file and directory in Linux has three kind of owners:

User

User is the owner of the file. When you create a file, you become the owner of the file. The ownership can be changed as well but we'll see that later.

Group

Every user is part of a certain group(s). A group consists of several users and this is one way to manage users in a multi-user environment.

For example, if you have dev team, QA team and sysadmin team accessing the same system, you should create separate groups for them. This way, you can manage files and security of the system effectively. It saves time because instead of manually adding permission for each user, you can simply add them to a group and change the permission for the group. You'll see how to do it later in this article.

Even if you are the only user of the system, you'll still be part of many groups. Distributions like Ubuntu also create a group with name same as the user's name.

Tip: Run the command **groups** to see what usergroups you belong to.

Other

Other can be considered as a super group with all the users on the system. Basically, anyone with access to the system belongs to this group.

In other words, 'User' is a single user, Group is a collection of users and Other consists of all the users on the system.

File permissions in Linux

Every file and directory in Linux has the following three permissions for all the three kinds of owners:

Permissions for files

- Read – Can view or copy file contents
- Write – Can modify file content
- Execute – Can run the file (if its executable)

Permissions for directories

- Read – Can list all files and copy the files from directory
- Write – Can add or delete files into directory (needs execute permission as well)
- Execute – Can enter the directory

Understanding file permissions and ownership in Linux

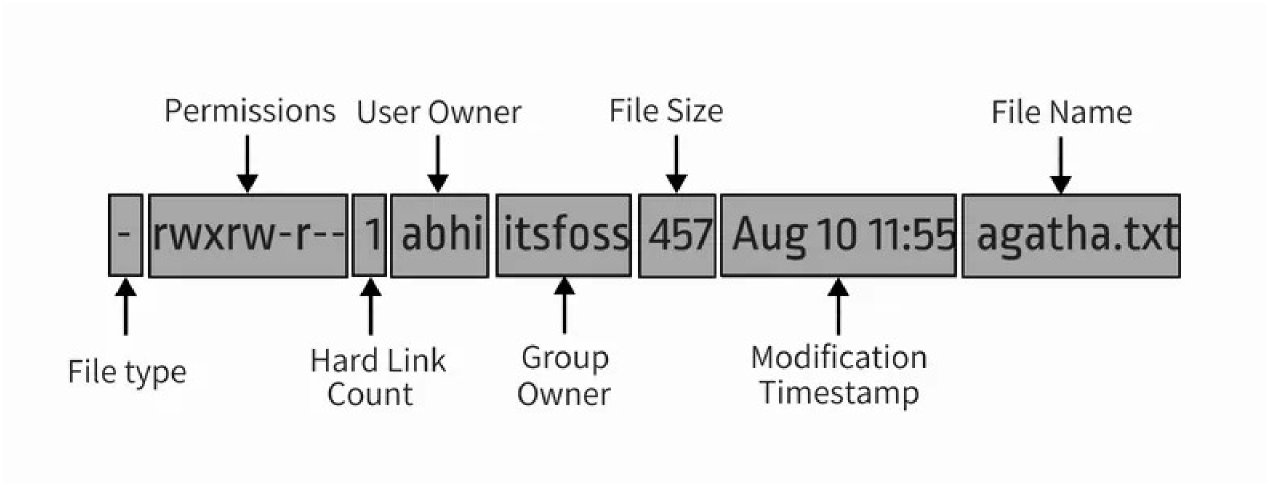
Now that you are aware of the basic terminology of file permissions and ownership, it's time to see it in action.

You can use the 'stat command' or the 'ls command' to check the file permissions.

If you use the ls command with option -l on a file, you'll see an output like this:

```
-rwxrw-r-- 1 abhi itsfoss 457 Aug 10 11:55 agatha.txt
```

Let me explain this output with a picture:



Let me further explain the entire output in detail:

- **File type**: Denotes the type of file. d means directory, – means regular file, l means a symbolic link.
- **Permissions**: This field shows the permission set on a file. I'll explain it in detail in the next section.
- **Hard link count**: Shows if the file has hard links. Default count is one.
- **User**: The user who owns the files.
- **Group**: The group that has access to this file. Only one group can be the owner of a file at a time.
- **File size**: Size of the file in bytes.
- **Modification time**: The date and time the file was last modified.

- **Filename**: Obviously, the name of the file or directory.

Now that you have understood the ls -l command output, let's focus on the file permission part.

In the above command, you see the file permission like this in the nine digit **format**:

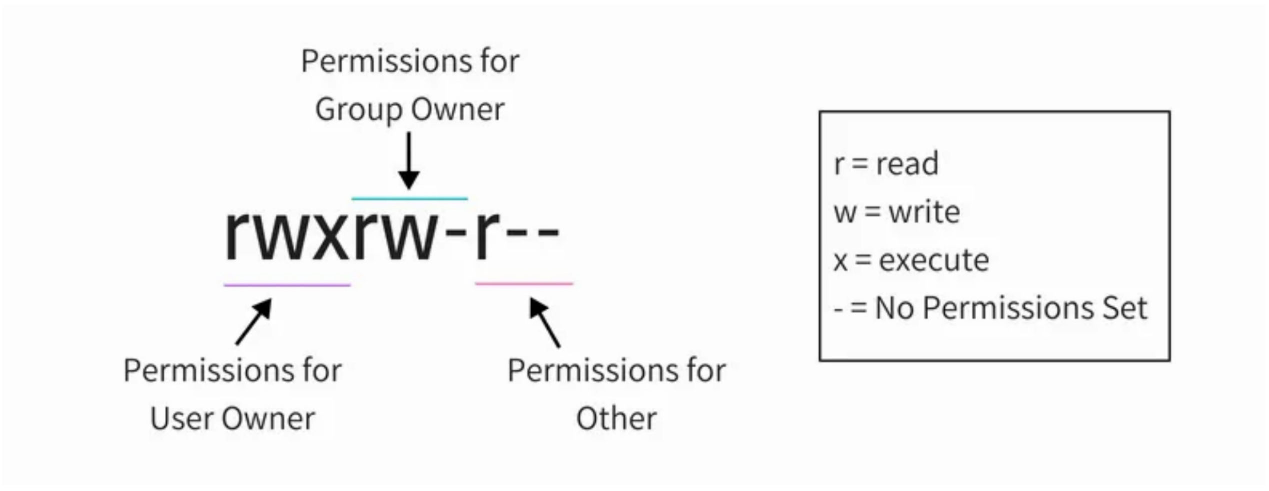
```
rwxrw-r--
```

Each letter denotes a particular permission:

- r : Read permission
- w : Write permission
- x : Execute permission
- – : No permission set

Permissions are always in the order of read, write and execute i.e., rwx. And then these permissions are set for all three kind of owners (see the ownership section) in the order of User, Group and Other.

This picture will explain things better:



So, if you look at the above picture now, you can say the following things about the file permissions:

- The file has read, write and execute permissions for the User owner. But who is this use owner of the file? You have this info in the output of ls -l (i.e. user abhi).
- The file has read and write permissions for the Group but not execute. Which group is it? You have the group info in the output of the command ls -l (i.e. group itsfoss).
- The file has only read permission for Other i.e. everyone that has access to the system. You don't need to know which other is it because 'other' means all the users.

Now if you see the entire ls -l command once again, you can read the file permissions and ownership together.

```
-rwxrw-r-- 1 abhi itsfoss 457 Aug 10 11:55 agatha.txt
```

The file agatha.txt is owned by user abhi and abhi has read, write and execute permission. All the members of group istfoss have read and write access to this file while everyone else has only read access to this file.

Note: Root user has super powers and normally, it has read, write and execute permissions to all the files, even if you don't see it in file permissions.

A single user may be the member of several groups but only the primary group of the user is the group owner of a file created by the user. The primary group of a user can be found using the command *id -gn <username>*. Leave the username blank if you are trying to find your own primary group.

Now that you know the how to find out permissions on a file, let's see how can you change the permission and ownership of a file.

Change file permissions in Linux

You can use chmod command for changing the permissions on a file in Linux.

Trivia: Permissions used to be called *mode of access* and hence chmod was the short form of *change the mode of access*.

There are two ways to use the chmod command:

- Absolute mode
- Symbolic mode

Using chmod in absolute mode

In the absolute mode, permissions are represented in numeric form (octal system to be precise). In this system, each file permission is represented by a number.

- r (read) = 4
- w (write) = 2
- x (execute) = 1
- – (no permission) = 0

With these numeric values, you can combine them and thus one number can be used to represent the entire permission set.

Number	Permission
0	—
1	–x
2	-w-
3 (i.e. 2+1)	-wx
4	r–
5 (i.e. 4+1)	r-x
6 (i.e. 4+2)	rw-
7 (i.e. 4+2+1)	rwx

Can you guess the file permission in numbers on agatha.txt file in our example so far? That's right, it's 764.

Now that you know what number represents which permission, let's see how to change file permission using this knowledge.

Suppose you want to change the file permission on agatha.txt so that everyone can read and write but no one can execute it? In that case, you can use the chmod command like this:

```
chmod 666 agatha.txt
```

If you list agatha.txt now, you'll see that the permission has been changed.

```
-rw-rw-rw- 1 abhishek abhishek 457 Aug 10 11:55 agatha.txt
```

Using chmod in symbolic mode

The problem with the absolute mode is that you should always provide three numbers for all the three owners even if you want to change the permission set for just one owner.

This is where you can use the symbolic mode with chmod command.

In symbolic mode, owners are denoted with the following symbols:

- u = user owner
- g = group owner
- o = other
- a = all (user + group + other)

Symbolic mode uses mathematical operators to perform the permission changes:

- + for adding permissions
- – for removing permissions
- = for overriding existing permissions with new value

Now that you know let's see how to use chmod command in symbolic mode.

In our previous example, if you want to add execute permission for group owner, you can use chmod command like this:

```
chmod g+x agatha.txt
```

If you look at the permissions on this file now, you'll see that execute permission has now been added:

```
-rw-rwxrw- 1 abhi itsfoss 457 Aug 10 11:55 agatha.txt
```

You can also combine multiple permission changes in one command. Suppose you want to remove the read and write permission and add execute permissions for Other. You also want to add execute permission for the User owner. You can do all of it one single command:

```
chmod o-rw+x,u+x agatha.txt
```

The resulting permissions would be like this:

```
-rwxrwx--x 1 abhi itsfoss 457 Aug 10 11:55 agatha.txt
```

If you want to change the permissions for all three kind of users at the same time, you can use it in the following manner:

```
chmod a-x agatha.txt
```

This will remove the execute permission for everyone.

```
-rw-rw---- 1 abhi itsfoss 457 Aug 10 11:55 agatha.txt
```

Some people find converting the file permissions from one mode to another a tiresome job. This why I created this little tool that allows you to calculate Linux file permissions in various modes online.

Change file ownership in Linux

To change the ownership of a file, you can use the command chown. You may easily guess that chown stands for change owner.

You can change the user owner of a file in the following manner:

```
chown <new_user_name> <filename>
```

If you want to change the user as well as group, you can use cown command like this:

```
chown <new_user_name>:<new_user_group> <filename>
```

If you just want to change the group, you can either use chown command in this manner:

```
chown :<new_user_group> <filename>
```

or use chgrp command specifically used for changing group owner of a file or directory. You can guess that chgrp stands for change group.

```
chgrp <new_user_group> <filename>
```

In our example so far, if you want to change the user owner and group to root, you can use the chown command like this:

```
sudo chown root:root agatha.txt
```

This will change the ownership of the file to root for both user and the group.

```
-rw-rw---- 1 root root 457 Aug 10 11:55 agatha.txt
```

Notice that I had to use sudo with chown? It's because the root is involved here and to deal with root, you need superuser rights.

Tip: Two groups cannot own the same file.

Bonus Tip: Is there a precedence in file permissions?

Think of a situation, where the user owner doesn't have any permissions, group has read permission while others have read and write permissions.

```
----r--rw- 1 abhi itsfoss 457 Aug 10 11:55 agatha.txt
```

Now, if the user abhi tries to read the file using cat or less command, will he be able to? The answer is no because it doesn't have the read permission.

But user abhi is part of group itsfoss and the group has read access. Heck! other has read and write permission. This should mean that everyone (including user abhi) can read and write the file, right? Wrong!

In Linux, the precedence takes from user and then group and then to other. Linux system checks who initiated the process (cat or less in our example). If the user who initiated the process is also the user owner of the file, the user permission bits are set.

If owner of the file didn't initiate the process, then the Linux system checks the group. If the user who initiated the process is in the same group as the owner group of the file, group permissions bit are set.

If this process owner is not even in the group as the file's group owner, then the other permission bits are set.

What next?

I hope you liked the article and now you have a better understanding of how file permissions work in Linux.

There are some advanced file permissions like SUID, GUID and sticky bit that you may learn next, if you want to.

If you have any questions or suggestions or if you just want to say thanks, please leave a comment below. If you liked the article, please share it on social media or various forums. This will help us and other Linux users as well.