# **File Permissions in Linux/Unix**

|  |
| --- |
| Linux is a clone of UNIX, the **multi-user operating system** which can be accessed by many users simultaneously. Linux can also be used in mainframes and servers without any modifications. But this raises security concerns as an unsolicited or **malign user** can **corrupt, change or remove crucial data**. For effective security , Linux divides authorization into 2 levels   * Ownership * Permission   The concept of **permissions** and **ownership** is crucial in Linux . Here, we will discuss both of them. Let us start with the **Ownership**. |

## **Ownership in Linux files**

Every file and directory in your Unix/Linux system is assigned 3 types of owner , given below

### **User**

A user is the owner of the file. By default, the person who created a file becomes its owner. Hence , a user is also sometimes called an owner.

### **Group**

A user- group can contain multiple users. All users belonging to a group will have the same access permissions to the file. Suppose you have a project where a number of people require access to a file. Instead of manually assigning permissions to each user, you could add all users to a group, and assign group permission to file such that only this group members and no one else can read or modify the files.

### **Other**

Any other user who has access to a file. This person has neither created the file nor he belongs to a usergroup who could own the file. Practically, it means everybody else. Hence, when you set the permission for others, it is also referred as set permissions for world.

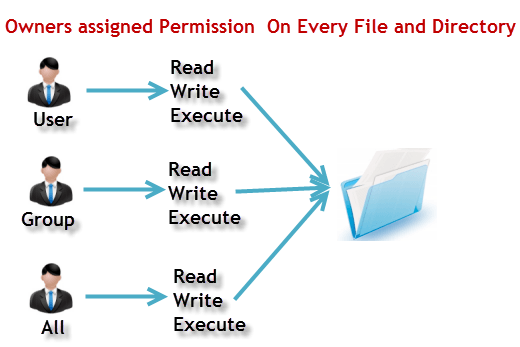
Now, the big question arises how does **Linux distinguishes** between these three user types so that a user 'A' cannot affect a file which contains some other user 'B's' vital information/data. It is like you do not want your colleague, who works on your Linux computer, to view your personal images. This is where **Permissions** set in and they define **user behavior**.

Let us understand the **Permission system** on Linux.

## **Permissions**

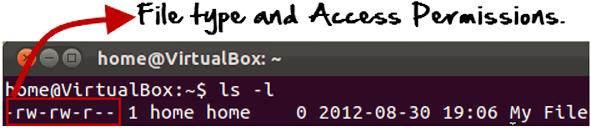
Every file and directory in your UNIX/Linux system has following 3 permissions defined for all the 3 owners discussed above

* **Read** : This permission give you the authority to open and read a file. Read permission on a directory gives you the ability to lists it's content.
* **Write:** The right 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 write permission on the 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 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 actually run it.



Lets see this in action

**ls - l** on terminal gives



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



Here, the first **'-'** implies that we have selected a file.

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

File Permissions in Linux/Unix

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:

|  |  |
| --- | --- |
| File Permissions in Linux/Unix | * Read the file * Write or edit the file * He cannot execute the file since , the execute bit is set to '-'. |

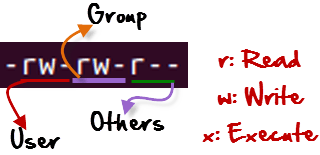
By design many Linux distributions like Fedora , CentOS , Ubuntu etc , will add users to a group of the same group name as the user name. Thus, a user 'tom' is added to a group named 'tom'.

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



## **Changing file/directory permissions with '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**

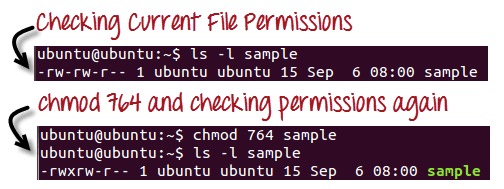
### ***Absolute(Numeric) Mode***

In 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 |

Let's see the chmod command in action.



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

|  |  |
| --- | --- |
| File Permissions in Linux/Unix | '764' absolute code says the following:   * Owner can read, write and execute * Usergroup can read and write * World can only read   **This is shown as '-rwxrw-r-** |

This is how you can change the permissions on a file by assigning an absolute number.

### **Symbolic Mode**

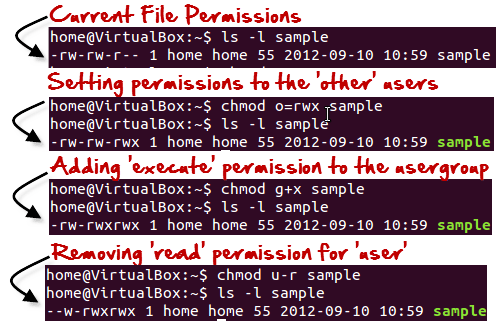
In the Absolute mode you change permissions for all 3 owners. In the symbolic mode you can modify permissions of a specific owner. It makes use of mathematical symbols to modify the file permissions

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **+** | Adds a permission to a file or directory |
| **-** | Removes the permission |
| **=** | Sets the permission and overrides the permissions set earlier. |

The various owners are represented as -

|  |  |
| --- | --- |
| **User Denotations** | |
| u | user/owner |
| g | group |
| o | other |
| a | all |

We will not be using permissions in numbers like 755 but characters like rwx. Let's look into an example



**Changing Ownership and Group**

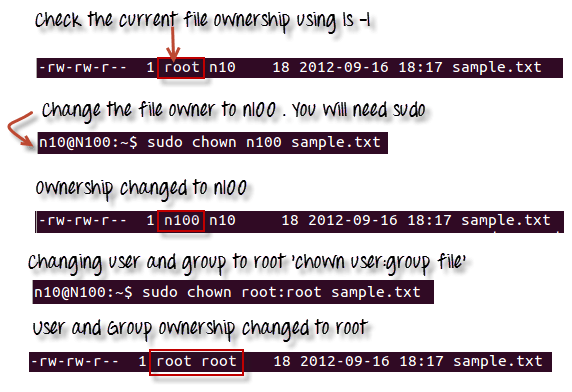
For changing the ownership of a file/directory , you can use the following command:

***chown user***

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

**chown user:group**

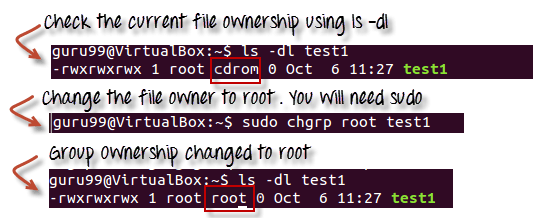
Let's see this in action



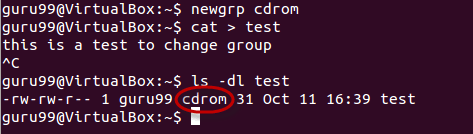
In case , you want to change group-owner only , use the command

**chgrp**

'**chgrp'** stands for change group.



**Tip**

* The file /etc/group contains all the groups defined in the system
* You can use the command "groups" to find all the groups you are a member of
* File Permissions in Linux/Unix
* You can use the command newgrp to work as a member a group other than your default group
* .
* You cannot have 2 groups owning the same file.
* You do not have nested groups in Linux. One group can not be sub-group of other
* x- eXecuting a directory means Being allowed to "enter" a dir and gain possible access to sub-dirs
* There are other permissions that you can set on Files and Directories which will be covered in a later advanced tutorial

Summary:

* Linux being a multi-user system uses permissions and ownerships for security\
* There are three user types on a Linux system viz. User, Group and Other
* Linux divides the file permissions into read, write and execute denoted by r,w and x
* The permissions on a file can be changed by 'chmod' command which can be further divided into Absolute and Symbolic mode
* The 'chown' command can change the ownership of a file/directory. Use the following commands: chown user file or chown user:group file
* The 'chgrp' command can change the group ownership **chrgrp group filename**
* What does x - eXecuting a directory mean? A: Being allowed to "enter" a dir and gain possible access to sub-dirs.