FILE PERMISSIONS

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.

1. Ownership
2. Permission

# OWNERSHIP OF LINUX FILES

Every file and directory on 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 the world.

# 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.

## 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.

# Example:

**akriti@CG-DTE-Student:/sys$ ls -la**

*total 0*

*dr-xr-xr-x 12 root root 0 Jun 7 07:52 .*

*drwxr-xr-x 1 root root 4096 Jun 3 10:30 ..*

*drwxr-xr-x 2 root root 0 Jun 7 07:52 block*

*drwxr-xr-x 2 root root 0 Jun 7 07:52 bus*

*drwxr-xr-x 7 root root 0 Jun 7 07:52 class*

*drwxr-xr-x 2 root root 0 Jun 7 07:52 dev*

*drwxr-xr-x 4 root root 0 Jun 7 07:52 devices*

*drwxr-xr-x 2 root root 0 Jun 7 07:52 firmware*

*drwxr-xr-x 3 root root 0 Jun 7 07:52 fs*

*drwxr-xr-x 5 root root 0 Jun 7 07:52 kernel*

*drwxr-xr-x 3 root root 0 Jun 7 07:52 module*

*drwxr-xr-x 2 root root 0 Jun 7 07:52 power*

**akriti@CG-DTE-Student:~$ ls -l cat.txt**

*-rw-rw-rw- 1 akriti akriti 31 Jun 7 08:18 cat.txt*

The row of ls -l command displays the permissions. It has 10 characters, each with a significant meaning:

Char 1 : (d/-): Whether it is a file(-) or a directory (d)

Char 2-4 : Permissions of the user

Char 5-7 : Permissions of the group

Char 8-10 : Permissions of the others

where r = read permission

w = write permission

x = execute permission

- = no permission

# Changing file/directory permissions with 'chmod' command

The command you use to change the security permissions on files is called “chmod”, which stands for “change mode”, because the nine security characters are collectively called the security “mode” of the file.

## Syntax:

*chmod permissions filename*

## Symbolic Mode

1. The first argument you give to the “chmod” command is ‘u’, ‘g’, ‘o’. We use:
   * u for user
   * g for group
   * for others,

You can also use a combination of them (u,g,o).

This specifies which of the three groups you want to modify.

1. After this use
   * ‘+’ for adding
   * ‘-‘ for removing
   * and “=” for assigning a permission.
2. 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

1. Use can use commas to modify more permissions
2. Finally, the name of the file whose permission you are changing

## Absolute/Numeric Mode

|  |  |  |
| --- | --- | --- |
| **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 |

Example:

**akriti@CG-DTE-Student:~$ ls -l cat.txt**

*-rw-rw-rw- 1 akriti akriti 31 Jun 7 08:18 cat.txt*

**akriti@CG-DTE-Student:~$ chmod 766 cat.txt**

**akriti@CG-DTE-Student:~$ ls -l cat.txt**

*-rwxrw-rw- 1 akriti akriti 31 Jun 7 08:18 cat.txt*