“Chmod” and “Chown” commands in Linux

The Linux operating system is a multi-user operating system. It has a security system set up that controls which clients and groups approach the files and indexes in the system. chmod and chown commands are tools for empowering clients to access files.

In Linux, clients can have a place with at least one group. Likewise, two clients and groups can be the proprietors of files and registries. Just as the subtleties of proprietorship, each file has metadata about its access permissions. Chmod and Chown are the tools we use to control possession and access permissions of files and directories.

The chown and chmod are popular command-line commands that are utilized to tackle access management of different files in Linux-based OS computers. The chmod command, likewise known as change mode, is utilized to change/alter permissions of a given file as indicated by a specific mode. The chown means "change owner" and is utilized to change/alter the owner of a given file or directories.

Exploring File Permissions with Chmod and Chown Command

You can utilize file permissions to control access to their files. Sysadmins can implement a security strategy dependent on file authorizations. All files have three sorts:

Owner – Process or person who made the file.

Group – All clients have a primary group and they own the file, which is helpful for sharing files or giving access.

Others – Users who are not the owner nor an individual from the group.

Understanding the Chown Command

The chown command’s syntax is as follows:

**chown [options] [user] [:group] file (s)**

Here the USER is the client, and GROUP is the group name, of course, and FILES is the file/files in the directory.

To find more info on chown command, you can use this command mentioned below:

**chown - - help**

In the first place, let’s make a directory (test) with files named as “test1.txt” and “test2.txt”:

**mkdir test**

**touch test/test1.txt**

**touch test/test2.txt**

Then, using the command mentioned below, list all the permissions for these files:

**ls -l test/**

Here, you will see that root is the group owner and user of the files:

**total0**

**-rw-r--r-- 1 root root 0 Aug 26 16:41 test1.txt**

**-rw-r--r-- 1 root root 0 Aug 26 16:41 test2.txt**

Now, you will change the files’ ownership of user to vyom user with:

**chown vyom test/test1.txt**

**chown vyom test/test2.txt**

Next, see the permissions listed for the two files:

**ls -l test/**

The output should be:

**total0**

**-rw-r--r-- 1 vyom root 0 Aug 26 16:41 test1.txt**

**-rw-r--r-- 1 vyom root 0 Aug 26 16:41 test2.txt**

Now, change the files’ ownership of group to vyom group with:

**chown :vyom test/test1.txt**

**chown :vyom test/test2.txt**

Then, again check the listed permissions with:

**ls -l test/**

You will see this result:

**total0**

**-rw-r--r-- 1 vyom vyom root 0 Aug 26 16:41 test1.txt**

**-rw-r--r-- 1 vyom vyom root 0 Aug 26 16:41 test2.txt**

With chown command, you can utilize - R option to change ownership.

For instance, to change the group and user ownership to vyom user and group for test directory and all sub-registries, you need to run this:

**chown -R vyom:vyom test**

Understanding the chmod command

The chmod command’s syntax is as follows:

**chmod [options] [permission] file(s)**

Or,

**chmod USER:GROUP:OTHERS FILE(s)**

Here, permissions characterize the group, users and others’ permissions. You can speak to these permissions with octal numbers or with symbols.

The following list portrays a rundown of a few permissions that can be utilized to assign permissions for group, users and others.

0 : — , none

1 : – x, execute as it were

2 : - w-, write as it were

3 : - wx, write and execute

4 : r–, read-as it were

5 : r-x, read and execute

6 : rw-, read and write

7 : rwx, read, write, execute

For more data about chmod, run this:

**chmod -- help**

How about we see some examples of chmod:

In the first place, make a directory named as permissions and make three different files namely file1, file2, and file3 in the directory:

**mkdir test**

**touch permissions/file1**

**touch permissions/file2**

**touch permissions/file3**

Now, change file1’s permission so everyone has complete access granted.

**chmod 777 permissions/file1**

Here, the initial 7 set the permission for the owner, the subsequent 7 sets the permission for the group.

To check the permission of file1, you can use the command mentioned below:

**ls -l permissions/file1**

**-rwxrwxrwx 1 root root 0 Aug 27 11:08 permissions/file1**

To assign the aforementioned permissions, you can use the following command:

**chmod u=rwx, g=rwx, o=rwx permissions/file1**

Now, assign file2’s permissions with the goal that solitary file2’s owner is granted complete access:

**chmod 700 permissions/file2**

Then, see the permission with this command:

**ls -l permissions/file2**

You should see this result:

**-rwx------ 1 root root 0 Aug 27 11:08 permissions/file2**

Now, assign file3’s permissions so owner is granted complete access to the file, individuals from others and group have rx permission:

**chmod 755 permissions/file3**

Then, you can check file’s permission with this command:

**ls -l permissions/file3**

You should see this result:

**-rwxr-xr-x 1 root root 0 Aug 27 11:08 permissions/file3**

You can likewise utilize symbols to permit and keep explicit permissions from getting the file.

For instance, deny the read permission for file1 to everybody using this command:

**chmod a-r permissions/file1**

To execute permission to everybody, you can use the command mentioned below:

**chmod a+x permissions/file1**

You can set the permission of file1 to write permission to the owner with this command:

**chmod u+w permissions/file1**

With chown command, you can utilize - R option to change the ownership recursively.

For instance, grant complete access to the directory permission with all sub-indexes and files recursively:

**chmod -R 777 Permissions**

You can likewise assign a file sticky bit permission with the goal that just the file owner or the root client can erase the file.

To assign file1’s sticky bit permission, you can do it with this command:

**chmod 1755 permissions/file1**

You would now be able to check file1’s permission demonstrated as follows:

**ls -l permissions/file1**

Here, you will see that t flag is now included in the file1:

**-rwxr-xr-t 1 root root 0 Aug 27 11:08 permissions/file1**