

How set file permissions using chmod on Linux

By Coding-Bootcamps.com

first run `ls -l` or `ls -l -a` (to see hidden files)

On each line, the first character identifies the type of entry that is being listed. If it is a dash (-) it is a file. If it is the letter d it is a directory.

The next nine characters represent the settings for the three sets of permissions.

- The first three characters show the permissions for the user who owns the file (*user permissions*).
- The middle three characters show the permissions for members of the file's group (*group permissions*).
- The last three characters show the permissions for anyone not in the first two categories (*other permissions*).

There are three characters in each set of permissions. The characters are indicators for the presence or absence of one of the permissions. They are either a dash (-) or a letter. If the character is a dash, it means that permission is not granted. If the character is an r, w, or an x, that permission has been granted.

The letters represent:

- *r*: Read permissions. The file can be opened, and its content viewed.
- *w*: Write permissions. The file can be edited, modified, and deleted.
- *x*: Execute permissions. If the file is a script or a program, it can be run (executed).
- For example:
- --- means no permissions have been granted at all.
- rwx means full permissions have been granted. The read, write, and execute indicators are all present.

Understanding The Permission Syntax

To use chmod to set permissions, we need to tell it:

- *Who*: Who we are setting permissions for.
- *What*: What change are we making? Are we adding or removing the permission?
- *Which*: Which of the permissions are we setting?

We use indicators to represent these values, and form short "permissions statements" such as u+x, where "u" means "user" (who), "+" means add (what), and "x" means the execute permission (which).

The “who” values we can use are:

- *u*: User, meaning the owner of the file.
- *g*: Group, meaning members of the group the file belongs to.
- *o*: Others, meaning people not governed by the *u* and *g* permissions.
- *a*: All, meaning all of the above.

If none of these are used, `chmod` behaves as if “*a*” had been used.

The “what” values we can use are:

- *-*: Minus sign. Removes the permission.
- *+*: Plus sign. Grants the permission. The permission is added to the existing permissions. If you want to have this permission and only this permission set, use the *=* option, described below.
- *=*: Equals sign. Set a permission and remove others.

The “which ” values we can use are:

- *r*: The read permission.
- *w*: The write permission.
- *x*: The execute permission.

Example 1

We want the user dave to have read and write permissions and the group and other users to have read permissions only. We can do using the following command:

```
chmod u=rw,og=r new_file.txt
```

//important: Using the “*=*” operator means we wipe out any existing permissions and then set the ones specified.

Example 2

How about adding a permission *without* removing the existing permissions settings? We can do that easily too.

We can add the execute permission for everyone with the following command:

```
chmod a+x new_script.sh
```

Example 3

setting permission on multiple files at the same time

```
chmod o-r *.page
```

Numerical Shorthand

Another way to use chmod is to provide the permissions you wish to give to the owner, group, and others as a three-digit number. The leftmost digit represents the permissions for the owner. The middle digit represents the permissions for the group members. The rightmost digit represents the permissions for the others.

The digits you can use and what they represent are listed here: [0-7 means binary value]

- 0: (000) No permission.
- 1: (001) Execute permission.
- 2: (010) Write permission.
- 3: (011) Write and execute permissions.
- 4: (100) Read permission.
- 5: (101) Read and execute permissions.
- 6: (110) Read and write permissions.
- 7: (111) Read, write, and execute permissions.

Using this method, you set the permissions that you wish to have; you do not add these permissions to the existing permissions. So if read and write permissions were already in place you would have to use 7 (111) to add execute permissions. Using 1 (001) would remove the read and write permissions and add the execute permission.

Example 4

Let's add the read permission back on the ".page" files for the others category of users. We must set the user and group permissions as well, so we need to set them to what they are already. These users already have read and write permissions, which is 6 (110). We want the "others" to have read and permissions, so they need to be set to 4 (100).

[it is very important to know users and groups prior to running below command]

chmod 664 *.page