Difference Between chmod and chown

The **chmod** and **chown** commands are used to control access to files in UNIX and Linux systems.

The chmod command stands for “change mode”, and allows changing permissions of files and folders, also known as “modes” in UNIX. The chown command stands for “change owner”, and allows changing the owner of a given file or folder, which can be a user and a group. That’s the difference between them in a nutshell.

chmod and chown are working together: chmod can control which permissions are available to owner and owner’s group for a given file or directory, while chown quickly changes who on the system gets access to a file.

Let’s take a quick look at the basic usage of these commands.

chmod examples

The chmod command can be used in a couple of different ways,

with permissions (or modes) set by numbers or by letters. Permissions can be given to a user who owns the file (u = user), group of said user (g = group), everyone else (o = others) or all users (a). And the basic permissions that can be given include read (r), write (w), and execute (x). There are also X, s, and t, but they're less commonly used.

When using numbers you can use a numeric value such as 644 to set permissions. The position of the value represents to whom is the permission given, and the actual value represents which (or how much) permissions are given as a sum total of each permission's unique value.

First position (in the above example 6) refers to the user. Second refers to the group of the user, and the third refers to all others.

Numeric values for permissions are:

4 = read 2 = write 1 = execute

So a value of 4 will only give read rights, but a value of 6 will give read and write rights because it is a sum of 4 and 2. 5 will give only read ane execute rights, and 7 will give all rights. Do this calculation for each numerical position and you'll end up with the desired value. So in the example of 644 we're giving the user who owns the file the permission to read and write (but not execute), the group of that user the permission to read only, and others the right to read only as well.

To set this mode with chmod on a file called important.txt we would simply run this command:

$ **chmod 644 important.txt**

Note that making a file executable, if it were a script or a program, amounts to simply giving someone or everyone a permission to execute. If this was an important.sh bash script

we could allow the owner to execute, and others to read with the 744 mode, or everyone to execute with 755.

$ **chmod 755 important.sh**

Now, we can also use letters to accomplish the same thing, and we've already mentioned the relevant letters above. This is probably easier to remember than using numbers. For example, to accomplish the 644 permissions above we would run this:

$**chmod u+rw,go+r important.txt**

So we're saying file owner user gets read and write permissions, group and others get to read.

The second example, with the important.sh file being made executable we could just run this:

$**chmod u+rwx,go+rx important.sh**

If important.sh already had permissions set to 644 we can add everyone execute rights by simply running:

$ **chmod +x important.sh**

Not specifying the letter for anyone is treated as if we said "a", for all.

Finally, if we're setting permissions to a folder we need to specify the -R option (standing for "recursive"):

$**chmod -R 644 important-files/**

chown command

Basic usage of chown is pretty straightforward. You just need to remember that first comes the user (owner), and then the group, delimited by a colon.

This command will set the user "daniel", from the group of "admins" as owners of the directory "important-files":

$**chown -R greys:admins important-files**

Just like with chmod, the -R is there when it’s a directory.

It’s possible to change just the owner of a file, without specifying a group (which will stay intact):

$**chown -R greys important-files**

See Also

- lrwxrwxrwx permissions
- Find Out File Type and Permissions in Perl
- Change Ownership of Files/Directories in Unix
- Basic unix commands
- Finding an Owner of a File in Unix