File Permissions - Chmod Command in Linux

In Linux, access to the files is managed through the file permissions, attributes, and ownership. This ensures that only authorized users and processes can access files and directories.

This tutorial covers how to use the `chmod` command to change the access permissions of files and directories.

Linux File Permissions

Before going further, let's explain the basic Linux permissions model.

In Linux, each file is associated with an owner and a group and assigned with permission access rights for three different classes of users:

- The file owner.
- The group members.
- Others (everybody else).

File ownership can be changed using the `chown` and `chgrp` commands. There are three file permissions types that apply to each class:

- The read permission.
- The write permission.
- The execute permission.

This concept allows you to specify which users are allowed to read the file, write to the file, or execute the file. File permissions can be viewed using the `ls` command:

```
ls -l filename.txt
```

```
-rw-r--r-- 12 linuxize users 12.0K Apr  8 20:51 filename.txt
|[-][-][-]-   [------] [---]
| |  |  | |      |       |
| |  |  | |      |       +-----------> 7. Group
| |  |  | |      +-------------------> 6. Owner
| |  |  | +--------------------------> 5. Alternate Access Method
| |  |  +----------------------------> 4. Others Permissions
| |  +-------------------------------> 3. Group Permissions
| +----------------------------------> 2. Owner Permissions
+------------------------------------> 1. File Type
```

The first character shows the file type. It can be a regular file (`-`), directory (`d`), a symbolic link (`l`), or any other special type of file.

The next nine characters represent the file permissions, three triplets of three characters each. The

first triplet shows the owner permissions, the second one group permissions, and the last triplet shows everybody else permissions. The permissions can have a different meaning depending on the file type.

In the example above (`rw-r—r—`) means that the file owner has read and write permissions (`rw-`), the group and others have only read permissions (`r—`).

Each of the three permission triplets can be constructed of the following characters and have a different effects, depending on whether they are set to a file or to a directory:

Effect of Permissions on Files

Permission	Character	Meaning on File	
Read	-	The file is not readable. You cannot view the file contents.	
	r	The file is readable.	
Write	-	The file cannot be changed or modified.	
	w	The file can be changed or modified.	
Execute	-	The file cannot be executed.	
	x	The file can be executed.	
	s	If found in the user triplet it sets the setuid bit. If found in the group triplet, it sets the setgid bit. It also means that x flag is set. When the setuid or setgid flags are set on an executable file, the file is executed with the file's owner and/or group privileges.	If found in the user triplet it sets the setuid bit. If found in the group triplet, it sets the setgid bit. It also means that x flag is set.
	S	Same as s but the x flag is not set. This flag is rarely used on files.	When the setuid or setgid flags are set on an executable file, the file is executed with the file's owner and/or group privileges.
	t	If found in the others triplet it sets the sticky bit. It also means that x flag is set. This flag is useless on files.	
	T	Same as t but the x flag is not set. This flag is useless on files.	

Effect of Permissions on Directories (Folders)

In Linux, Directories are special types of files that contain other files and directories.

Permission	Character	Meaning on Directory
Read	-	The directory's contents cannot be shown.
	r	The directory's contents can be shown. (e.g. You can list files inside the directory with ls.)
Write	-	The directory's contents cannot be altered.
	w	The directory's contents can be altered. (e.g. You cannot create new files, delete files ..etc.)

Permission	Character	Meaning on Directory
Execute	-	The directory cannot be changed to.
	x	The directory can be navigated using cd.
	s	If found in the user triplet, it sets the setuid bit. If found in the group triplet it sets the setgid bit. It also means that x flag is set. When the setgid flag is set on a directory the new files created within it inherits the directory group ID (GID), instead of the primary group ID of the user who created the file. setuid has no effect on directories.
	S	Same as s but the x flag is not set. This flag is useless on directories.
	t	If found in the others triplet it sets the sticky bit. It also means that x flag is set. When the sticky bit is set on a directory, only the file's owner, the directory's owner, or administrative user can delete or rename the files within the directory.
	T	Same as t but the x flag is not set. This flag is useless on directories.

Using chmod

The chmod command takes the following general form:

```
chmod [OPTIONS] MODE FILE...
```

The chmod command allows you to change the permissions on a file using either a symbolic or numeric mode or a reference file. We will explain the modes in more detail later in this article. The command can accept one or more files and/or directories separated by space as arguments.

Only root, the file owner or user with sudo privileges can change the permissions of a file. Be extra careful when using chmod, especially when recursively changing the permissions.

Symbolic (Text) Method

The syntax of the chmod command when using the symbolic mode has the following format:

```
chmod [OPTIONS] [ugoa…][-+=]perms…[,…] FILE...
```

The first set of flags ([ugoa…]), users flags, defines which users classes the permissions to the file are changed.

- u - The file owner.
- g - The users who are members of the group.
- o - All other users.
- a - All users, identical to ugo.

If the users flag is omitted, the default one is a and the permissions that are set by umask are not affected. The second set of flags ([-+=]), the operation flags, defines whether the permissions are to be removed, added, or set:

- - Removes the specified permissions.
- + Adds specified permissions.
- = Changes the current permissions to the specified permissions. If no permissions are specified

after the = symbol, all permissions from the specified user class are removed.

The permissions (`perms…`) can be explicitly set using either zero or one or more of the following letters: `r`, `w`, `x`, `X`, `s`, and `t`. Use a single letter from the set `u`, `g`, and `o` when copying permissions from one to another users class.

When setting permissions for more than one user classes (`[,…]`), use commas (without spaces) to separate the symbolic modes.

Below are some examples of how to use the `chmod` command in symbolic mode:

- Give the members of the group permission to read the file, but not to write and execute it:

```
chmod g=r filename
```

- Remove the execute permission for all users:

```
chmod a-x filename
```

- Repulsively remove the write permission for other users:

```
chmod -R o-w dirname
```

- Remove the read, write, and execute permission for all users except the file's owner:

```
chmod og-rwx filename
```

- The same thing can be also accomplished by using the following form:

```
chmod og= filename
```

- Give read, write and execute permission to the file's owner, read permissions to the file's group and no permissions to all other users:

```
chmod u=rwx,g=r,o= filename
```

- Add the file's owner permissions to the permissions that the members of the file's group have:

```
chmod g+u filename
```

- Add a sticky bit to a given directory:

```
chmod o+t dirname
```

Numeric Method

The syntax of the chmod command when using numeric method has the following format:

```
chmod [OPTIONS] NUMBER FILE...
```

When using the numeric mode, you can set the permissions for all three user classes (owner, group, and all others) at the same time.

The NUMBER can be a 3 or 4-digits number.

When 3 digits number is used the first digit represents the permissions of the file's owner, the second one of the file's group and the last one all other users.

Each write, read, and execute permissions have the following number value:

- r (read) = 4
- w (write) = 2
- x (execute) = 1
- no permissions = 0

The permissions number of a specific user class is represented by the sum of the values of the permissions for that group. To find out the file's permissions in numeric mode simply calculate the totals for all users classes. For example, to give read, write and execute permission to the file's owner, read and execute permissions to the file's group and only read permissions to all other users you would do the following:

- Owner: rwx=4+2+1=7
- Group: r-x=4+0+1=5
- Others: r-x=4+0+0=4

Using the method above we come up to the number 754, which represents the desired permissions.

To set up the `setuid`, `setgid`, and `sticky` bit flags use four digits number.

When the 4 digits number is used, the first digit has the following meaning:

- setuid=4
- setgid=2
- sticky=1
- no changes = 0

The next three digits have the same meaning as when using 3 digits number.

If the first digit is 0 it can be omitted, and the mode can be represented with 3 digits. The numeric mode 0755 is the same as 755.

To calculate the numeric mode you can also use another method (binary method), but it is a little more complicated. Knowing how to calculate the numeric mode using 4, 2, and 1 is sufficient for most users.

You can check the file's permissions in the numeric notation using the stat command:

```
stat -c "%a" filename
```

```
644
```

Here are some examples of how to use the chmod command in numeric mode:

- Give the file's owner read and write permissions and only read permissions to group members and all other users:

```
chmod 644 dirname
```

- Give the file's owner read, write and execute permissions, read and execute permissions to group members and no permissions to all other users:

```
chmod 750 dirname
```

- Give read, write, and execute permissions, and a sticky bit to a given directory:

```
chmod 1777 dirname
```

- Recursively set read, write, and execute permissions to the file owner and no permissions for all other users on a given directory:

```
chmod -R 700 dirname
```

Using a Reference File

The –reference=ref_file option allows you to set the file's permissions to be same as those of the specified reference file (ref_file).

```
chmod --reference=REF_FILE FILE
```

For example, the following command will assign the permissions of the file1 to file2:

```
chmod --reference=file1 file2
```

Recursively Change the File's Permissions

To recursively operate on all files and directories under the given directory, use the -R (–recursive) option:

```
chmod -R MODE DIRECTORY
```

For example, to change the permissions of all files and subdirectories under the /var/www directory to 755 you would use:

```
chmod -R 755 /var/www
```

Operating on Symbolic Links

Symbolic links always have 777 permissions.

By default, when changing symlink's permissions, chmod will change the permissions on the file the link is pointing to.

```
chmod 755 symlink
```

Chances are that instead of changing the target ownership, you will get a “cannot access ‘symlink’: Permission denied” error.

The error occurs because by default on most Linux distributions symlinks are protected, and you cannot operate on target files. This option is specified in `/proc/sys/fs/protected_symlinks`. `1` means enabled and `0` disabled. It is recommended not to disable the symlink protection.

Changing File Permissions in Bulk

Sometimes there are situations where you would need to bulk change files and directories permissions.

The most common scenario is to recursively change the website file’s permissions to 644 and directory’s permissions to 755.

Using the numeric method:

```
find /var/www/my_website -type d -exec chmod 755 {} \;
find /var/www/my_website -type f -exec chmod 644 {} \;
```

Using the symbolic method:

```
find /var/www/my_website -type d -exec chmod u=rwx,go=rx {} \;
find /var/www/my_website -type f -exec chmod u=rw,go=r {} \;
```

The find command will search for files and directories under /var/www/my_website and pass each found file and directory to the chmod command to set the permissions.

Last update: **2020/03/05 15:57**