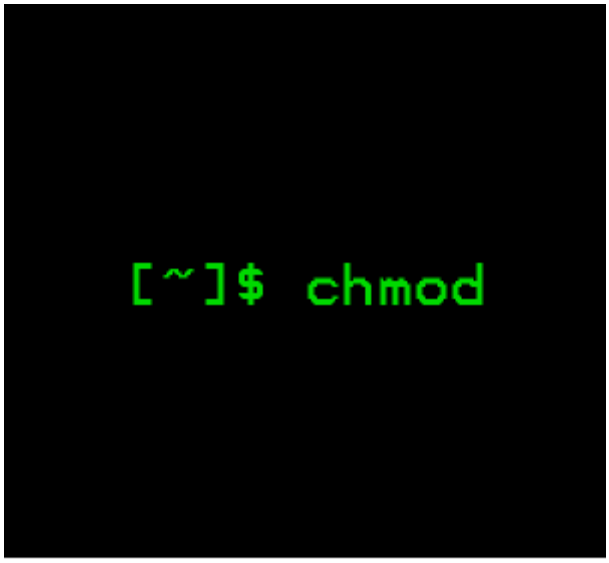
Linux and Unix chmod command



http://www.computerhope.com

About chmod

chmod is used to change the permissions of files or directories.

Overview

On Linux and other Unix-like operating systems, there is a set of rules for each file which defines who can access that file, and how they can access it. These rules are called file permissions or file *modes*. The command name **chmod** stands for "change mode", and it is used to define the way a file can be accessed.

Before continuing, you should read the section What Are File Permissions, And How Do They Work? in our documentation of the **umask** command. It contains a comprehensive description of how to define and express file permissions.

In general, **chmod** commands take the form:

```
chmod options permissions filename
```

If no *options* are specified, **chmod** modifies the permissions of the file specified by *filename* to the permissions specified by *permissions*.

permissions defines the permissions for the owner of the file (the "user"), members of the group who owns the file (the "group"), and anyone else ("others"). There are two ways to represent these permissions: with symbols (alphanumericcharacters), or with octal numbers (the digits **0** through **7**).

Let's say you are the owner of a file named **myfile**, and you want to set its permissions so that:

1. the **u**ser can **r**ead, **w**rite, and e**x**ecute it;
2. members of your **g**roup can **r**ead and e**x**ecute it; and
3. **o**thers may only **r**ead it.

This command will do the trick:

```
chmod u=rwx,g=rx,o=r myfile
```

This is an example of using symbolic permissions notation. The letters **u**, **g**, and **o** stand for "**user**", "**group**", and "**other**". The equals sign ("**=**") means "set the permissions exactly like this," and the letters "**r**", "**w**", and "**x**" stand for "read", "write", and "execute", respectively. The commas separate the different classes of permissions, and there are no spaces in between them.

Here is the equivalent command using octal permissions notation:

```
chmod 754 myfile
```

Here the digits **7**, **5**, and **4** each individually represent the permissions for the user, group, and others, in that order. Each digit is a combination of the numbers **4**, **2**, **1**,

and **0**:

4 stands for "read",

2 stands for "write",

1 stands for "execute", and

0 stands for "no permission."

So **7** is the combination of permissions **4**+**2**+**1** (read, write, and execute), **5** is **4**+**0**+**1** (read, no write, and execute), and **4** is **4**+**0**+**0** (read, no write, and no execute).

chmod syntax

```
chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE FILE...
chmod [OPTION]... --reference=RFILE FILE...
```

Options

-c, **--changes**	Like **--verbose**, but gives verbose output only when a change is actually made.
-f, **--silent**, **--quiet**	Quiet mode; suppress most error messages.
-v, **--verbose**	Verbose mode; output a diagnostic mesage for every file processed.

--no-preserve-root	do not treat '**/**' (the root directory) in any special way. This is the default.
--preserve-root	Do not operate recursively on '**/**'.
--reference=*RFILE*	Set permissions to match those of file *RFILE*, ignoring any specified *MODE*.
-R, **--recursive**	change files and directories recursively.
--help	Display a help message and exit.
--version	Output version information and exit.

Technical Description

chmod changes the file mode of each specified *FILE* according to *MODE*, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.

The format of a symbolic mode is:

```
[ugoa...][[+-=][perms...]...]
```

where *perms* is either zero or more letters from the set **r**, **w**, **x**, **X**, **s** and **t**, or a single letter from the set **u**, **g**, and **o**. Multiple symbolic modes can be given, separated by commas.

A combination of the letters **u**, **g**, **o**, and **a** controls which users' access to the file will be changed: the user who owns it (**u**), other users in the file's group (**g**), other users not in the file's group (**o**), or all users (**a**). If none of these are given, the effect is as if **a** were given, but bits that are set in the umask are not affected.

The operator **+** causes the selected file mode bits to be added to the existing file mode bits of each file; **-** causes them to be removed; and **=** causes them to be added and causes unmentioned bits to be removed except that a directory's unmentioned set user and group ID bits are not affected.

The letters **r**, **w**, **x**, **X**, **s** and **t** select file mode bits for the affected users: read (**r**), write (**w**), execute (**x**), execute only if the file is a directory or already has execute permission for some user (**X**), set user or group ID on execution (**s**), restricted deletion flag or sticky bit (**t**). For directories, the execute options **X** and **X** define permission to view the directory's contents.

Instead of one or more of these letters, you can specify exactly one of the letters **u**, **g**, or **o**: the permissions granted to the user who owns the file (**u**), the permissions granted to other users who are members of the file's group (**g**), and the permissions granted to users that are in neither of the two preceding categories (**o**).

A numeric mode is from one to four octal digits (**0**-**7**), derived by adding up the bits with values **4**, **2**, and **1**. Omitted digits are assumed to be leading zeros. The first digit selects the set user ID (**4**) and set group ID (**2**) and restricted deletion or sticky (**1**) attributes. The second digit selects permissions for the user who owns the read (**4**), write (**2**), and execute (**1**); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.

chmod never changes the permissions of symbolic links; the **chmod** system call cannot change their permissions. This is not a problem since the permissions of

symbolic links are never used. However, for each symbolic link listed on the command line, **chmod** changes the permissions of the pointed-to file. In contrast, **chmod** ignores symbolic links encountered during recursive directory traversals.

Setuid And Setgid Bits

chmod clears the set-group-ID bit of a regular file if the file's group ID does not match the user's effective group ID or one of the user's supplementary group IDs, unless the user has appropriate privileges. Additional restrictions may cause the set-user-ID and set-group-ID bits of *MODE* or *RFILE* to be ignored. This behavior depends on the policy and functionality of the underlying **chmod** system call. When in doubt, check the underlying system behavior.

chmod preserves a directory's set-user-ID and set-group-ID bits unless you explicitly specify otherwise. You can set or clear the bits with symbolic modes like **u+s** and **g-s**, and you can set (but not clear) the bits with a numeric mode.

Restricted Deletion Flag (or "Sticky Bit")

The restricted deletion flag or sticky bit is a single bit, whose interpretation depends on the file type. For directories, it prevents unprivileged users from removing or renaming a file in the directory unless they own the file or the directory; this is called the restricted deletion flag for the directory, and is commonly found on world-writable directories like **/tmp**. For regular files on some older systems, the bit saves the program's text image on the swap device so it will load more quickly when run; this is called the sticky bit.

chmod examples

```
chmod 644 file.htm
```

Set the permissions of **file.htm** to "owner can read and write; group can read only; others can read only".

```
chmod u=rw example.jpg
```

Change the permissions for the owner of **example.jpg** so that the owner may read and write the file. Do not change the permissions for the group, or for others.

```
chmod u+s comphope.txt
```

Set the "Set-User-ID" bit of **comphope.txt**, so that anyone who attempts to access that file does so as if they are the owner of the file.

```
chmod u-s comphope.txt
```

The opposite of the above command; un-sets the SUID bit.

```
chmod 755 file.cgi
```

Set the permissions of **file.cgi** to "read, write, and execute by owner" and "read and execute by the group and everyone else".

```
chmod 666 file.txt
```

Set the permission of **file.txt** to "read and write by everyone.".

```
chmod a=rw file.txt
```

Accomplishes the same thing as the above command, using symbolic notation.

Viewing Permissions in The File Listing

A quick and easy way to list a file's permissions are with the long listing (**-l**) option of the **ls** command. For example, to view the permissions of **file.txt**, you could use the command:

```
ls -l file.txt
```

...which will display output that looks like the following:

```
-rwxrw-r-- 1   hope   hopestaff  123   Feb 03 15:36   file.txt
```

Here's what each part of this information means:

-	The first character represents the file type: "**-**" for a regular file, "**d**" for a directory, "**l**" for a symbolic link.
rwx	The next three characters represent the permissions for the file's owner: in this case, the owner may **r**ead from, **w**rite to, and/or e**x**ecute the file.
rw-	The next three characters represent the permissions for members of the group that the file belongs to. In this case, any member of the file's owning group may **r**ead from or **w**rite to the file. The final dash is a placeholder; group members do

	not have permission to execute this file.
r--	The permissions for "others" (everyone else). Others may only **r**ead this file.
1	The number of hard links to this file.
hope	The file's owner.
hopestaff	The group to whom the file belongs.
123	The size of the file in blocks.
Feb 03 15:36	The file's mtime (date and time when the file was last modified).
file.txt	The name of the file.

Related commands

chown — Change the ownership of files or directories.
getfacl — Display file access control lists.
ls — List the contents of a directory or directories.