UNIX Permissions and chmod Examples

In some cases, guides and manuals may refer to "permissions" as "privileges". In the case of this document, we will use the word "permissions".

Because the Unix is a multi-user machine, you need to protect files and directories from deletion and editing by other people. Before you can understand file permissions (how UNIX protects files), you should have a basic understanding of how files are accessed and how the Unix operating system categorizes users.

Access

You and any other users can access files in three ways:

- **read** - You do this when you look or display a file on your screen.
- **write** - You do this when you edit or change a file in some way or even delete it.
- **execute** - Some files are actually sets of instructions (programs) for the computer. These need to have execute privileges.

Users

There are three types of users that can access files:

- **owner** - the owner of the file.
- **group** - a member of a group to which the owner belongs. This is an *electronic* group, and on an academic server (for example) may be your department or a special group set up for working on the web site.
- **other** - everyone else with an account on the server who isn't a member of the group.

Nine Ways

Three types of users accessing a file in three different ways equals a total of nine possible ways to access a plain file!!! OR NINE POSSIBLE PERMISSIONS ON ACCESSING A FILE!

The owner of a file can:	read from the file write to the file execute the file
A member of the owner's group can:	read from the file write to the file execute the file
Anyone else can:	read from the file write to the file execute the file

What is the permission for a file/directory?

Now that you understand the many ways a file can be accessed, how do you know what the permissions are? Simply put, permissions are who and how the file can be accessed. To gain a better understanding of permissions you need to list the current files in a directory.

From a Unix prompt, enter ls -l

The resulting directory displayed shows several columns of information, such as this:

```
-rwxr-xr--  1 caldwell  ecat  21678 Sep 13 09:27 internetguide.html
-rwxrwxr-x  1 ffolinda  ecat   3128 Dec 31 09:19 people.html
drwxr-xr-x  4 rcasler   ecat    512 Dec 16 17:07 pubs/
drwxr-xr-x  2 ffolinda  ecat    512 Jan  2 08:56 web/
```

The permission explanation is the left most column. The owner name is in the middle of the display (the account logon name) and the group name follows the owner's name.

For the above file named internetguide.html, the permissions are: **-rwxr-xr--**
Permissions are set by the characters in the following way:

r	the capability to read a file
w	the capability to write/edit/delete a file.
x	the capability to execute a file or search a directory.
-	No capability to access file.

A **d** at the left most position indicates a directory area; a **-** at the left most position indicates a file.

the next three letters indicate the permissions for the owner

```
-rwxr-xr--     read/write/execute for owner
```

the middle three letters indicate the permissions for the group

```
-rwxr-xr--     read/execute for group
```

the last three letters indicate the permissions for the world

```
-rwxrwxr--     read for world
```

Following are few examples on how to use chmod Symbolic Notation.

These are the symbolic representation of three different roles:

- u is for user,
- g is for group,
- and o is for others.

These are the symbolic representation of three different permissions:

- r is for read permission,
- w is for write permission,
- x is for execute permission.

1. Add single permission to a file/directory

Changing permission to a single set. + symbol means adding permission. For example, do the following to give execute permission for the user irrespective of anything else:

```
$ chmod u+x filename
```

2. Add multiple permission to a file/directory

Use comma to separate the multiple permission sets as shown below.

```
$ chmod u+r,g+x filename
```

3. Remove permission from a file/directory

Following example removes read and write permission for the user.

```
$ chmod u-rx filename
```

4. Change permission for all roles on a file/directory

Following example assigns execute privilege to user, group and others (basically anybody can execute this file).

```
chmod a+x filename
```

5. Make permission for a file same as another file (using reference)

If you want to change a file permission same as another file, use the reference option as shown below. In this example, file2′s permission will be set exactly same as file1′s permission.

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