Introduction

Managing user permissions is an important administrative task. There are many commmand line tools available such as `chown`, `chgrp`, and `chmod`. Use command line tools in either : symbolic and numeric mode to alter user permissions. The symbolic method uses symbols such as `u`, `+`, `x`, `r`, and so on, to representing owners, groups, and permissions to alter permissions. The numeric method uses a numbering scheme.

Each file and directory has permissions for the owner (UID), group (GID), and everyone else (WORLD). The permissions for each group consist of three (binary) bits. There are 10 bits in total: `--- --- ---` (the 10th bit is the setuid or sticky bit - this will be described later). To alter permissions for owner use first three bits. Use the next three bits to alter permissions of group, and use the last three bits to alter permissions for everyone else.

Use the `ls -l` command to view file and directory permissions:

```
ls -l file1

-rw-rw-r-- 1 user1 group1 0 Oct 22 17:51 file1
```

The above example shows the following permissions for the `file1` file:

- user1: read and write
- group1: read and write
- everyone else: read

The three main symbols used to represent permissions are `r`, `w`, and `x`. These represent read, write, and execute permissions respectively. File permissions are as follows:

- read: files can be opened and viewed using commands such as `cat` and `less`
- write: edit, save, and delete files
- execute: allows you to execute the file (files will not be executable unless you also have read permissions)

Permissions for directories are as follows:

- read: list the contents using the `ls` command
- write: edit, save, and delete files within said directory
- execute: allows you to change into said directory using the `cd` command. Execute permissions are required to perform a long listing using the `ls -l` command. Without execute permissions the `ls -l` command will return output similar to the following:

```
ls -l test1/
```

```
ls: cannot access test1/file1: Permission denied

ls: cannot access test1/file2: Permission denied

total 0

-????????? ? ? ? ?                ? file1

-????????? ? ? ? ?                ? file2
```

Directory write Permissions
If a user has write permissions on a directory, that user can remove any file within that directory regardless of file permissions.

The Concept Of umask

The *User File Creation Mask* or `umask` value is the `bash` built-in that defines the default mode for newly created files or directories. The `umask` value is determined when the effective permissions value of the new files is subtracted from a value of the full file permissions. On Linux systems, full file permissions are:

- 777 - for directories
- 666 - for files

The default `umask` values on Fedora are set in `/etc/bashrc`. The default `umask` for user *root* is 0022 and for regular users (users whose UID is identical to their primary group's GID), `umask` value is set to 0002. For practical purposes, the first *0* is ignored - it denotes that this value is an *octal* number. To calculate the default creation mode (effective permissions) of all files and directories created by *root* user, subtract 022 from the full access mode value of 777 for directories or 666 for files. This means that:

- new files created by *root* have permissions set to 644 (666 - 022), or `rw-r--r--` in a symbolic denotation
- new directories created by *root* have permissions set to 755 (777 - 022), or `rwxr-xr-x` in a symbolic denotation

Using the same technique, default permissions for files and directories created by normal users are:

- 664 (666 - 002) or `rw-rw-r--` for files
- 775 (777 - 002) or `rwxrwxr-x` for directories

User can change this behavior by running `umask` command with a desired mode as a command argument:

```
umask 0022
```

This will cause all new files created by user to have permissions set to 644 and all new directories' permissions set to 755. The change will be in effect until the shell environment is re-initialized. To make permanent changes add `umask` command to user's `~/.bashrc` file. For example, user may wish to have all new files and directories accessible only by himself. In other words, effective permissions on new files should be 600 or `rw-------` and on new directories 700 or `rwx------`. To achieve this, add the line:

```
umask 0077
```

to the end of `~/.bashrc` file, which is the file located under the user's home directory.

Managing Permissions Using The Command Line Interface

Symbolic Method

The following table describes the symbols used to change permissions using the symbolic method. Familiarize yourself with this table before proceeding to the next section:

add a permission

u	the owner of the file or directory
g	the group the file or directory belongs to
o	everyone else
a	everyone (u, g, and o)
=	assign a permission
r	read permissions
w	write permissions
x	execute permissions
t	directory sticky bit
s	setuid or setgid

To add a permission to a user, group, or everyone else, use the `+` symbol. The following example adds execute permissions for the owner (`u`):

```
chmod u+x file1
```

To add execute permissions to the owner, and the group, use the following command:

```
chmod u+x,g+x file1
```

Please note there is no space between the `u+x` and `g+x`. Permissions do not have to be specified separately. The following has the same result as running the `chmod u+x,g+x file1` command:

```
chmod ug+x file1
```

You must list all permissions needed when you assign permissions using the `=` symbol. For example, if the owner of the `file1` file has read, write, and execute permissions, the follow command removes all but the owners read permissions:

```
chmod u=r file1
```

Note, if the group and everyone else had permissions, the previous command would not remove those permissions. You must only list all the permissions if you specify the owner, group, or everyone else when using the `chmod` command.

Use the `-` symbol to remove permissions. For example, if the owner of the `file1` file had execute permissions, the following command would remove those permissions:

```
chmod u-x file1
```

Numeric Method

The following table describes the numbering scheme used when changing permissions using the numeric method:

Number	Permissions	`ls -l` Output
0	no permissions	---
1	execute	--x
2	write	-w-
3	write and execute	-wx
4	read	r--
5	read and execute	r-x
6	read and write	rw-
7	read, write, and execute	rwx

Use the `chmod` command to change permissions regardless of whether you are using the symbolic or numeric method.

To set permissions using the numeric method, use the `chmod xxx` command, where `xxx` are values between `0` and `7`. The table above describes the permissions each value (0-7) applies. The first value is the permission for the owner. The second value is for the group, and the third value is for everyone else.

Use the following command to assign the owner read, write, and execute permissions, and remove all permissions for the group and everyone else:

```
chmod 700 file1
```

View the permissions using the `ls -l` command:

```
ls -l

-rwx------ 1 user1 user1 0 Oct 27 16:02 file1
```

Use the following command to add read and write permissions for the `file1` file for the owner, group, and everyone else:

```
chmod 666 file1
```

To change permissions on a folder, and all files and sub-directories within that folder, use the `-R` option:

```
chmod -R 700 folder1
```

This applies mode `700` permissions to the `folder1` folder, and recursively changes the permissions of all files and sub-directories within the `folder1` folder.

Permissions on Directories

Execute permission on a directory does not allow files within that directory to be executed. Rather, it allows users to change into that directory using the `cd` command. It also allows user to perform a long listing using `ls -l` command. However, files within a directory can be executed if said files have execute permissions.

Managing Permissions Using The Graphical User Interface

Follow these steps to access a graphical user interface (GUI) for managing permissions on files and folders:

- Right click on the file or folder.
- On the menu that appears, click the *Properties* menu item.
- Click the *Permissions* tab.

Folder Permissions

The following table describes the *Folder Access* permissions. Changes to *Folder Access* permissions take immediate effect:

Create and delete files	read, write, and execute
Access files	read and execute
List files only	read
None	no permissions, all actions are denied

The following table describes *File Access* permissions. This allows finer-grained control of files within directories. Changes to *File Access* permissions take effect only after clicking the *Apply permissions to enclosed files* button.

Read and write	read and write
Read-only	read

When the *File Access* is set to ---, clicking *Apply permissions to enclosed files* keeps the current file permissions without changing them.

If the *Execute: Allow executing file as program* box is ticked, execute permissions are applied for everyone to files within that directory. If the *Execute: Allow executing file as program* box is not ticked, and you click the *Apply permissions to enclosed files* button, execute permissions are not removed from files within that directory.

If you are a member of a *Secondary Group*, change the group owner using the *Group*drop-down menu.

File Access Permissions
After closing and re-opening the properties window for a file or directory, the *File Access* permissions appear as ---. Your permissions are still set. Use the `ls -l [foldername]` command to view the file access permissions. If you are already in the `[foldername]` directory, use the `ls -l` command.

File Permissions

The following table describes the *Access* permissions for files. Changes to *Access* permissions take immediate effect:

Read-only	read
Read and write	read and write
None	no permissions, all actions are denied

Ticking the *Execute: Allow executing file as program* box applies execute permissions for everyone to the file.

If you are a member of a *Secondary Group* you can change the group owner using the *Group* drop-down menu.

Special Permissions

There are two special permissions that can be set on executable files: Set User ID (setuid) and Set Group ID (sgid). These permissions allow the file being executed to be executed with the privileges of the owner or the group. For example, if a file was owned by the root user and has the setuid bit set, no matter who executed the file it would always run with root user privileges.

Set User ID (setuid)

You must be the owner of the file or the root user to set the setuid bit. Run the following command to set the setuid bit:

```
chmod u+s file1
```

View the permissions using the `ls -l` command:

```
ls -l file1

-rwSrw-r-- 1 user1 user1 0 2007-10-29 21:41 file1
```

Note the capital `S`. This means there are no execute permissions. Run the following command to add execute permissions to the `file1` file, noting the lower case `s`:

```
chmod u+x file1

ls -l file1

-rwsrw-r-- 1 user1 user1 0 2007-10-29 21:41 file1
```

Note the lower case `s`. This means there are execute permissions.

Alternatively, you can set the setuid bit using the numeric method by prepending a `4` to the mode. For example, to set the setuid bit, read, write, and execute permissions for owner of the `file1` file, run the following command:

```
chmod 4700 file1
```

Set Group ID (setgid)

When the Set Group ID bit is set, the executable is run with the authority of the group. For example, if a file was owned by the `users` group, no matter who executed that file it would always run with the authority of the `users` group. For example, run the following command as to set the setgid bit on the `file1` file:

```
chmod g+s
```

Note that both the setuid and setgid bits are set using the `s` symbol. Alternatively, prepend a 2 to the mode. For example, run the following command as root to set the setgid bit, and read, write, and execute permissions for the owner of the `file1` file:

```
chmod 2700 file1
```

The setgid is represented the same as the setuid bit, except in the group section of the permissions:

```
ls -l file1

-rwx--S--- 1 user1 user1 0 2007-10-30 21:40 file1
```

Use the `chmod u+s` command to set the setuid bit. Use the `chmod g+s` command to set the setgid bit.

Special Permissions for Directories

There are two special permissions for directories: the sticky bit and the setgid bit. When the sticky bit is set on a directory, only the root user, the owner of the directory, and the owner of a file can remove files within said directory.

Sticky Bit

An example of the sticky bit is the `/tmp` directory. Use the `ls -ld /tmp` command to view the permissions:

```
ls -ld /tmp

drwxrwxrwt  24 root root  4096 2007-10-30 22:00 tmp
```

The `t` at the end symbolizes that the sticky bit is set. A file created in the `/tmp` directory can only be removed by its owner, or the root user. For example, run the following command to set the sticky bit on the `folder1` folder:

```
chmod a+t folder1
```

Alternatively, prepend a `1` to the mode of a directory to set the sticky bit:

```
chmod 1777 folder1
```

The permissions should be read, write, and execute for the owner, group, and everyone else, on directories that have the sticky bit set. This allows anyone to `cd` into the directory and create files.

Set Group ID

When the setgid bit is set on a directory, all files created within said directory inherit the group ownership of that directory. For example, the `folder1` folder is owned by the user `user1`, and the group `group1`:

```
ls -ld folder1

drwxrwxr-x 2 user1 group1 4096 2007-10-30 22:25 folder1
```

Files created in the `folder1` folder will inherit the `group1` group membership:

```
touch folder1/file1

ls -l folder1/file1

-rw-rw-r-- 1 user1 group1 0 2007-10-30 22:29 folder1/file1
```

To set the setgid bit on a directory, use the `chmod g+s` command:

```
chmod g+s folder1
```

View the permissions using the `ls -ld` command, noting the `s` in the group permissions:

```
ls -ld folder1

drwxrwsr-x 2 user1 group1 4096 2007-10-30 22:32 folder1
```

Alternatively, prepend a `2` to the directories mode:

```
chmod 2770 folder1
```