Understanding file permissions on Unix:

A brief tutorial

[Source: http://www.dartmouth.edu/~rc/help/faq/permissions.html]

Every user on a Unix system has a unique username, and is a member of at least one group (the primary group for that user). This group information is held in the password file (/etc/passwd). A user can also be a member of one or more other groups. The auxiliary group information is held in the file /etc/group. Only the administrator can create new groups or add/delete group members (one of the shortcomings of the system).

Every directory and file on the system has an owner, and also an associated group. It also has a set of permission flags which specify separate read, write and execute permissions for the 'user' (owner), 'group', and 'other' (everyone else with an account on the computer) The 'ls' command shows the permissions and group associated with files when used with the -l option. On some systems (e.g. Coos), the '-g' option is also needed to see the group information.

An example of the output produced by 'ls -l' is shown below.

```
drwx------ 2 richard staff  2048 Jan  2 1997  private
drwxrws--- 2 richard staff  2048 Jan  2 1997  admin
-rw-rw---- 2 richard staff 12040 Aug 20 1996  admin/userinfo
drwxr-xr-x 3 richard user   2048 May 13 09:27 public
```

Understanding how to read this output is useful to all unix users, but especially people using group access permissions.

Field 1: a set of ten permission flags.
Field 2: link count (don't worry about this)
Field 3: owner of the file
Field 4: associated group for the file
Field 5: size in bytes
Field 6-8: date of last modification (format varies, but always 3 fields)
Field 9: name of file (possibly with path, depending on how ls was called)

The permission flags are read as follows (left to right)

position	**Meaning**
1	directory flag, 'd' if a directory, '-' if a normal file, something else occasionally may appear here for special devices.
2,3,4	read, write, execute permission for User (Owner) of file
5,6,7	read, write, execute permission for Group
8,9,10	read, write, execute permission for Other

value	Meaning
-	in any position means that flag is not set
r	file is readable by owner, group or other
w	file is writeable. On a directory, write access means you can add or delete files
x	file is executable (only for programs and shell scripts - not useful for data files). Execute permission on a directory means you can list the files in that directory
s	in the place where 'x' would normally go is called the set-UID or set-groupID flag.

On an executable program with set-UID or set-groupID, that program runs with the effective permissions of its owner or group.

For a directory, the set-groupID flag means that all files created inside that directory will inherit the group of the directory. Without this flag, a file takes on the primary group of the user creating the file. This property is important to people trying to maintain a directory as group accessible. The subdirectories also inherit the set-groupID property.

The default file permissions (umask):

Each user has a default set of permissions which apply to all files created by that user, unless the software explicitly sets something else. This is often called the 'umask', after the command used to change it. It is either inherited from the login process, or set in the .cshrc or .login file which configures an individual account, or it can be run manually.

Typically the default configuration is equivalent to typing 'umask 22' which produces permissions of:

```
-rw-r--r-- for regular files, or
drwxr-xr-x for directories.
```

In other words, user has full access, everyone else (group and other) has read access to files, lookup access to directories.

When working with group-access files and directories, it is common to use 'umask 2' which produces permissions of:

```
-rw-rw-r-- for regular files, or
drwxrwxr-x for directories.
```

For private work, use 'umask 77' which produces permissions:

```
-rw------- for regular files, or
drwx------ for directories.
```

The logic behind the number given to umask is not intuitive.

The command to change the permission flags is "chmod". Only the owner of a file can change its permissions.

The command to change the group of a file is "chgrp". Only the owner of a file can change its group, and can only change it to a group of which he is a member.

See the online manual pages for details of these commands on any particular system (e.g. "man chmod").

Examples of typical useage are given below:

```
chmod g+w myfile
```

give group write permission to "myfile", leaving all other permission flags alone

```
chmod g-rw myfile
```

remove read and write access to "myfile", leaving all other permission flags alone

```
chmod g+rwxs mydir
```

give full group read/write access to directory "mydir", also setting the set-groupID flag so that directories created inside it inherit the group

```
chmod u=rw,go= privatefile
```

explicitly give user read/write access, and revoke all group and other access, to file 'privatefile'

```
chmod -R g+rw .
```

give group read write access to this directory, and *everything* inside of it (-R = recursive)

```
chgrp -R medi .
```

change the ownership of this directory to group 'medi' and *everything* inside of it (-R = recursive). The person issuing this command must own all the files or it will fail.