This laboratory explores:

- categories of users
- how UNIX systems set access permissions to file system entities like files and directories

Categories of Users

Users and groups are used in UNIX systems for assigning permissions to file system entities.

Users

Everyone in class has a user account. Typically each person gets assigned their own user account.

Special Users

There is a user account that is for the administrative user or superuser. This user can do anything to any entity in the file system at any time. The name for this user is **root**. This user owns many of the system files we use, such as the system commands we have run on the command line in previous labs.

Groups

Users can belong to one or more groups.

Typically, when creating a user account, a group is created having the same name as the user account being created, and the user account being created will be added as a member of that group. For example, user *btv01* belongs to group *btv01*. This is the user account's personal group. Generally, only the user account having a particular name is a member of the group having the same name. For example, only user *btv01* belongs to group *btv01*,

In addition, for convenience, multiple users can belong to a group. For example, all users in this class belong to the *btv* group.

World/Other

The *world* or *other* category refers to all users.

File System Access Permissions

The traditional permission scheme associated with each file system entity involves three categories:

1. user (one user)
2. group (one group)
3. world/other (actually, users other than the above user who do not belong to the above group)

The operating system maintains three types of permissions for the user, group, and world categories:

1. read permission

2. write permission
3. execute/search permission

For a file:

- a user having read permission can view the contents of a file
- a user having write permission can modify the contents of a file
- a user having execute permission can have the operating system interpret the contents of a file as instructions to the computer

For a directory:

- a user having read permission can view the contents of a directory (that is to say, the list of entities associated with the directory)
- a user having write permission can modify the contents of a directory
- a user having execute permission can search within the directory, and can execute the directory's files that are marked as executable

Exercises

Setup

Ensure your home directory is your working directory, then execute the following commands:

```
% mkdir lab05
% cd lab05
% cp ~bmcdonald/lab05/runnable .
% cp ~bmcdonald/lab05/executeOnly .
% touch readOnly readWrite writeOnly
% mkdir searchDir
% touch searchDir/found searchDir/notFound
```

Don't miss the dots at the end of the *cp* commands above; they are required, and refer to the current directory as the destination of the copy operation. The *cp* command has the format:

```
% cp sourceFile(s) destination
```

You must specify arguments representing the source and the destination for the *cp* command to work.

The *cp* command indicates failure by generating a usage message; it may not be obvious to you that an error has occurred.

Groups

Run the **groups** command to find out to what groups you belong:

```
% groups
```

Viewing Permissions

Execute the following command:

```
% ls -l
```

That's ell ess space dash ell, to spell out the characters phonetically. You will see something like:

```
total 2
-rw-r--r--  1 bmcdonald  bmcdonald    0 Feb 22 00:28 executeOnly
-rw-r--r--  1 bmcdonald  bmcdonald    0 Feb 22 00:28 readOnly
-rw-r--r--  1 bmcdonald  bmcdonald    0 Feb 22 00:28 readWrite
-rw-r--r--  1 bmcdonald  bmcdonald    0 Feb 22 00:28 runnable
drwxr-xr-x  2 bmcdonald  bmcdonald  512 Feb 22 00:29 searchDir/
-rw-r--r--  1 bmcdonald  bmcdonald    0 Feb 22 00:28 writeOnly
```

Also try:

```
% ls -l *
```

The first column of characters indicates the type of entity and the permissions available on the entities. The first character in the first column will be a dash if the entity is a file, and a **d** if the entity is a directory. The remaining nine characters in the first column indicate the read/write/execute permissions for user, group, and world (in that order, three consecutive characters each, from left to right). The third column indicates the user associated with the entity and to which the user permissions refer; this is the *owner* of the entity. The fourth column indicates the group associated with the entity and to which the group permissions refer. The fifth column indicates the size of the entity; if it is a file, it is the length of the file in bytes. The sixth, seventh, and eighth columns represent the date and time on which the file was last changed. The ninth column is the name of the entity; the slash at the end of the entry for searchDir indicates that the entry is for a directory.

The *touch* command will change the date last changed shown in the *ls -l* listing to the current date.

Changing Permissions

Use the chmod command to change permissions of a file. Run the following:

```
% chmod 444 readOnly
% chmod 200 writeOnly
% chmod 111 executeOnly
% chmod 755 runnable
```

Run **ls -l** to look at the permissions of your files.

Run the following:

```
% cat readOnly
% cat writeOnly
% cat readWrite
% cat executeOnly
% cat runnable
```

Run the following:

```
% echo junk >> readOnly
% echo junk >> readWrite
% echo junk >> writeOnly
% cat readWrite
% cat writeOnly
% chmod 644 writeOnly
% cat writeOnly
% chmod 200 writeOnly
```

Run the following:

```
% ./runnable
% ./executeOnly
```

Run the following:

```
% ls searchDir
% cat searchDir/found
% chmod 111 searchDir
% ls searchDir
% ls searchDir/found
% cat searchDir/found
% chmod 755 searchDir
```

Initializing Permissions

I got too sleepy to finish this one. We will use the *umask* command to initialize file permissions, and then use the *touch* command to examine the effect of the *umask* command.