Access Permissions

- UNIX is a multi-user system. Every file and directory in your account can be protected from or made accessible to other users by changing its access permissions. Every user has responsibility for controlling access to their files.

- Permissions for a file or directory may be any or all of:

```
r   -   read
w   -   write
x   -   execute = running a program
```

- Each permission (rwx) can be controlled at three levels:

```
u   -    user = yourself
g   -    group = can be people in the same project
o   -    other = everyone on the system
```

- File access permissions are displayed using the ls -l command. The output from the ls -l command shows all permissions for all levels as three groups of three according to the scheme:

```
owner read (r)
owner write (w)
owner execute (x)
   group read (r)
   group write (w)
   group execute (x)
      public read (r)
      public write (w)
      public execute (x)

which are displayed as:        -rwxrwxrwx
```

  Example outputs from the ls -l command:

```
-rw-------  2 smith  staff 3287 Apr  8 12:10 file1
   - User has read and write permission. Group and
     others have no permissions.

-rw-r--r--  2 smith  staff 13297 Apr  8 12:11 file2
   - User has read and write permission.  Group and
     others can only read the file.

-rwxr-xr-x  2 smith  staff 4133 Apr  8 12:10 myprog
   - User has read, write and execute permission.
     Group and others can read and execute the file.

drwxr-x---  2 smith  staff 1024 Jun 17 10:00 SCCS
   - This is a directory. The user has read, write and
     execute permission. Group has read and execute
     permission on the directory. Nobody else can
     access it.
```

- Note: a directory must have both r and x permissions if the files it contains are to be accessed.

- The chmod command is used to change access permissions for files which you own. The syntax is:

```
     chmod      permission_triads         filename
             [who][action][permissions]
```

where:

```
      who            action          permissions

     u = user        + = add         r = read
     g = group       - = remove      w = write
     o = other                       x = execute
     a = all
```

Examples:

```
     chmod   a+r   sample.f
        - Adds read permission for all users to the file
          sample.f.

     chmod   o-r   sample.f
        - Removes read permission for others to the file
          sample.f.

     chmod   og+rx   prog*
        - Adds read and execute permissions for group and
          others to all files which contain "prog" as the
          first four characters of their name.

     chmod   +w   *
        - Adds write permission for user to all files in
          current directory.
```

- File access permissions can also be changed by a numerical (octal) chmod specification. Read permission is given the value 4, write permission the value 2 and execute permission 1.

```
     r  w  x
     4  2  1
```

These values are added together for any one user category:

```
     0   =   no permissions
     1   =   execute only
     2   =   write only
     3   =   write and execute (1+2)
     4   =   read only
     5   =   read and execute (4+1)
     6   =   read and write (4+2)
     7   =   read and write and execute (4+2+1)
```

So access permissions can be expressed as three digits. For example:

```
                           user    group   others

     chmod 640 file1       rw-     r--     ---
     chmod 754 file1       rwx     r-x     r--
     chmod 664 file1       rw-     rw-     r--
```

- Never set write permission for all other users on a file or directory which is in your home directory. If you do other users will be able to change its content. This can represent a serious security risk.

- The umask command is used to set your default file permissions. Typically, the umask command is included as part of your .profile, .cshrc or .login file.

  The umask command accepts only octal specifications. Note that these are different than those used by the chmod command, and in fact, represent which permissions to "mask out", or remove.

Octal number	Access permissions given	
0	rwx	read, write and execute
1	rw-	read and write
2	r-x	read and execute
3	r--	read only
4	-wx	write and execute
5	-w-	write only
6	--x	execute only
7	---	no permissions

  Example umask commands:

  ```
  umask 077
     - Subtracts 077 from the system defaults for files
       (666) and directories (777). Results in default
       access permissions for your files of 600
       (rw-------) and for directories of 700
       (rwx------).

  umask 002
     - Subtracts 002 from the sytem defaults to give a
       default access permission for your files of 664
       (rw-rw-r--) and for your directories of 775
       (rwxrwxr-x).

  umask 022
     - Subtracts 022 from the system defaults to give a
       default access permission for your files of 644
       (rw-r--r--) and for your directories of 755
       (rwxr-xr-x).
  ```