8. Access Permissions

This lab will deal with the different types of permissions you can give to files and directories in UNIX and how to change those permissions.

Lab 8.1 - Basic Types of Permissions

UNIX supports access control and allows you to set the permissions on files & directories that you own. Every file and directory has associated with it ownership, and access permissions. This means other people cannot access or change your files or directories unless you have granted them permission. There are two methods of setting file permissions which we will discuss later in this tutorial.

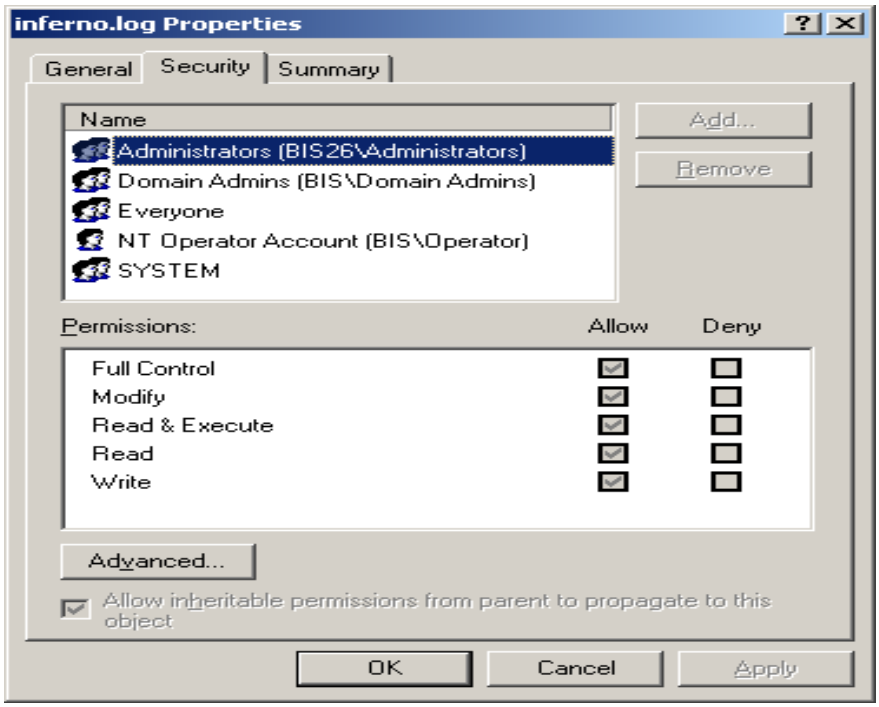

Figure 8.1: File Permissions in Windows

In Windows, as you can see from figure 8.1 above, there are different kinds of permissions you can set for your files and folders such as write, read, modify etc. UNIX also has permissions associated with its files and folders and they are defined as read, write, and execute. The read, write, and execute permissions are referred to as **r**, **w**, and **x**, respectively.

Read permission (r)

For a file, having read permission allows you to view the contents of the file.

For a directory, having read permission allows you to list the directory's contents.

Write permission (w)

For a file, write permission allows you to modify the contents of the file.

For a directory, write permission allows you to alter the contents of the directory, i.e., to add or delete files.

Execute permission (x)

For a file, execute permission allows you to run the file, if it is an executable program, or script. Note that file execute permission is irrelevant for nonexecutable files. For a directory, execute permission allows you to cd to the directory, and make it your current working directory.

Each of these permissions can be set for any one of three types of user that exist in UNIX:

- **u** the user who owns the file (usually you)
- **g** members of the same group of which the owner belongs
- **o** all other users (those who do not own it or are not part of the same group)

UNIX allows users to be placed in groups, so that the control of access is made simpler for administrators. When created, all files have an owner and group associated with them. The owner is the same as the username of the person who created the files and the group is the name of the creator's default login group, such as third_yrs. Only the root account (administrator) can change the ownership of a file.

Lab 8.2 - Viewing permissions

To see the current permissions, owner, and group for a file or directory, type the following commands:

cd - *to get back to your home directory*

ls -l

This will display the contents of your home directory using the ls command, with the -l option which will list the long directory list entry. Figure 8.2 is an example of what you might have seen with the ls –l command:

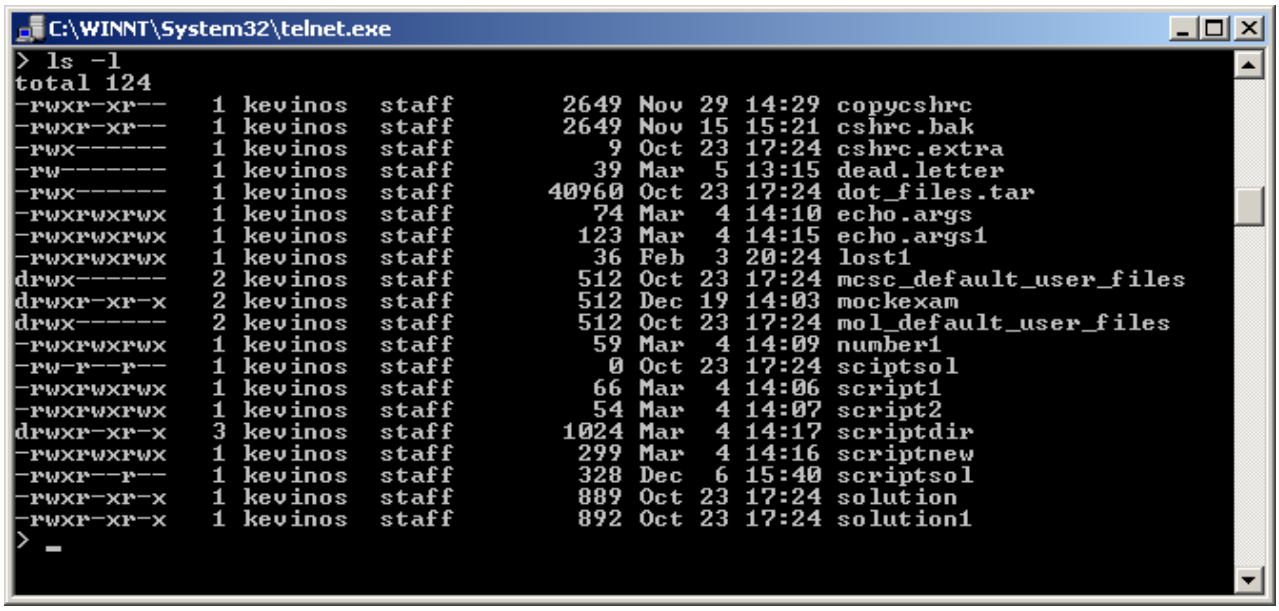

Figure 8.2: Output of the ls –l command

The third column (you) tells the owner of the file or directory, and the fourth column is the name of the group for the file or directory.

The permissions are listed in the first column on the left hand side. It is a 10 symbol 'string' consisting of the symbols **d**, **r**, **w**, **x**, **-**, (and, occasionally, **s** or **S but very rarely)**.

The first letter of this string is whether the item is a directory or a file. If the first letter is a "d", then the item is a directory. When you typed the ls –l command you will see that Training has a d as its first character. Notice, for the file banana, the first letter is "-". These are the two you will be dealing with. However note there is also:

Special files

Special files represent input/output (i/o) devices, like a tty (terminal), a disk drive, or a printer. Because UNIX treats such devices as files, a degree of compatibility can be achieved between device i/o, and ordinary file i/o, allowing for the more efficient use of software. Special files can be either *character special files,* that deal with streams of characters, or *block special files*, that operate on larger blocks of data.

Links

A link is a pointer to another file. Remember that a directory is nothing more than a list of the names and i-numbers of files. A directory entry can be a *hard link*, in which the i-number points directly to another file. A hard link to a file is indistinguishable from the file itself. When a hard link is made, then the i-numbers of two different directory file entries point to the same inode. For that reason, hard links cannot span across file systems. A *soft link* (or *symbolic link*) provides an indirect pointer to a file. A soft link is implemented as a directory file entry containing a pathname. Soft links are distinguishable from files, and can span across file systems.

The 9 remaining characters indicate the permissions, or access rights, and are taken as three groups of 3.

File type user group others

\- r w x r w x r w x

- The left group of 3 gives the file permissions for the **user** that owns the file (or directory);
- the middle group of 3 gives the permissions for the **group** of people to whom the file (or directory) belongs;
- the rightmost group of 3 gives the permissions for all **others**.

A dash '-' instead of r, w or x means that it does not have that permission.

The only one who can modify or delete any of your files or directories is the owner you (or the administrator- "root").

Some examples

-rwxrwxrwx is a **file** that everyone (all 3 types) can **read**, **write** and **execute** (and delete).

-rw------- is a file that only the owner can read and write - no-one else can read or write and no-one has execution rights.

Lab 8.3 - Setting file/directory permissions

UNIX allows you to set the permissions on files & directories that you own. The command to change the file permission mode is **chmod**.

Lab 8.3.1 - First method: chmod through 'ugoa' values

u user that owns the file.

g group that owns the file.

o other (everyone else).

a all (everybody , u,g and o).

To define the kind of change you want to make to the permissions, use the plus sign (+) to add a permission, the minus sign (-) to remove a permission and the equal sign (=) to set a permission directly.

The permissions as you have seen already are:

r read the file.

w write or edit the file.

x execute the file.

The following examples show how to use the **chmod** utility to change permissions on a file named **temp**.

```
touch temp      // create a temporary file called temp
ls –l temp      // to see its permissions
```

The following command removes all permissions for the file temp from all users, including the owner. Note that you give the command chmod, then give the permission before finally giving the name of the file whose permissions you want to change :

```
chmod a–rwx temp
ls –l temp
```

```
----------   1 user3998       0 Jan  8 15:55 temp
```

The following command gives the owner read, write & execute permissions on the file temp:

```
chmod u+rwx temp
ls -l temp
-rwx------   1 user3998       0 Jan  8 15:55 temp
```

The following command below adds read, write & execute permissions on the file to your group:

```
chmod g+rwx temp
ls –l temp
-rwxrwx---   1 user3998       0 Jan  8 15:55 temp
```

The following command adds the read permission for other users:

```
chmod o+r temp
ls –l temp
-rwxrwxr--   1 user3998       0 Jan  8 15:55 temp
```

Lab 8.3.2 - second method: chmod numerically

There is a shorthand way of setting permissions by using numbers. 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:

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—

The first number is for the user, second number is for the group and the third number is for others.

Examples:

To remove all permissions on the temp file type:

```
chmod 000 temp
ls –l temp
---------- 1 user3998       0 Jan  8 15:55 temp
```

To give yourself, the owner, read, write & execute permissions on the file temp type:

```
chmod 700 temp
ls –l temp
-rwx------ 1 user3998       0 Jan  8 15:55 temp
```

To give your group read, write & execute permissions as well as the owner(you) type:

```
chmod 770 temp
ls –l temp
-rwxrwx--- 1 user3998       0 Jan  8 15:55 temp
```

To give all users read permission as well type:

```
chmod 774 temp
ls –l temp
-rwxrwxr-- 1 user3998       0 Jan  8 15:55 temp
```

Lab 8.4 - Changing group ownership of files and directories

Every user is a member of one or more groups. To find out which groups you belong to use the command:

groups

To find out which groups another user belongs to use the command:

groups *username*

Your files and directories are owned by the group (or one of the groups) that you belong to. This is known as their "group ownership".

To list the group ownership of your files:

ls –gl

Exercise 8.1 (Using the letters method)

Create a blank file called **Delta**

Check the permissions on the file

Change the permissions using the letters method (o, g, u, a) so that:

- the user has read write and execute permissions
- the group has read and write permissions
- everyone else has just read permission

Exercise 8.2 (Using the numerical method)

Create a file called **Delta2**

Check the permissions on the file

Change the permissions using the numerical method (1,2,4) so that:

- the user has read write and execute permissions
- the group has read permissions only
- everyone else have no permissions