# Using Basic File Permissions

## Objectives

This module introduces file permissions and describes how to view and change permissions.

Upon completion of this module, you should be able to:

- View file and directory permissions
- Determine file or directory access
- Change the permissions
- Modify the default permissions

# Viewing File and Directory Permissions

All files and directories in the Solaris OS have a standard set of access permissions. These access permissions control which files can be accessed by whom, and provides a fundamental level of security for the system. You can use the `ls -l` command and the `ls -n` command to view the permissions for a given file or directory. You can also change the permissions for certain files.

## Security Fundamentals

One of the important functions of a secure system is to limit access to authorized users and prevent unauthorized users from accessing the files. Although the system administrator maintains the primary security of a system, users also play a role in keeping the system secure.

The Solaris OS uses two basic measures to prevent unauthorized access to a system and to protect data:

- The first measure is to authenticate a user's login by verifying that the user name and password exist in the `/etc/passwd` and `/etc/shadow` files.
- The second measure is to protect file and directory access automatically. The Solaris OS assigns a standard set of access permissions at the time of creation of files and directories.

**Note –** The Solaris OS also provides a special user account on every system, called the `root` user. The `root` user, often referred to as the superuser, has complete access to every user account and all files and directories. The `root` user can override the permissions placed on all files and directories.

## Viewing Permission Categories

To view the permissions for files and directories, use the `ls -l` command. Figure 7-2 shows the information displayed for the `dante` file.

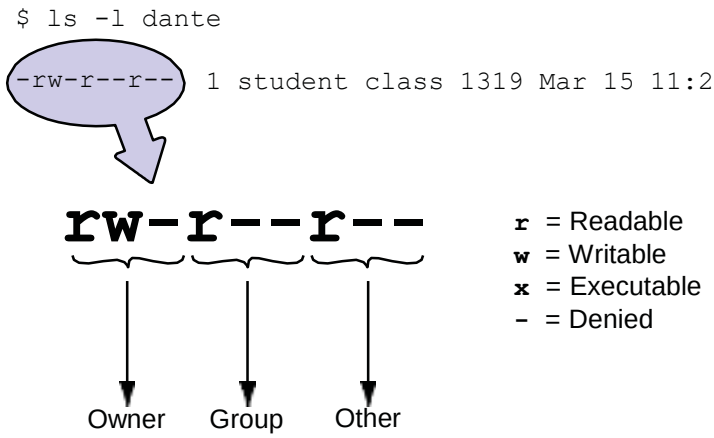


**Figure 7-2** Permissions Example

The first field of information displayed by the `ls -l` command is the file type. The file type typically specifies whether it is a file or a directory. A file is represented by a hyphen (-). A directory is represented by the letter `d`.

The remaining fields represent three types of users: owner, group, and other.

Table 7-1 describes each type of user with a brief description of each.

**Table 7-1** Types of Users

| Field | Description |
|---|---|
| Owner | Permissions used by the assigned owner of the file or directory. |
| Group | Permissions used by members of the group that owns the file or directory. |
| Other | Permissions used by all users other than the file owner, and members of the group that owns the file or the directory. |

Each type of user has three permissions, called a permission set. Each permission set consists of read, write, and execute permissions. Each file or directory has three permission sets for the three types of users. The first permission set represents the owner permissions. The second permission set represents the group permissions. The last permission set represents the other users' permissions.

## Permission Types

The read, write, and execute permissions in the owner, group, and other permission sets are represented by the characters `r`, `w`, and `x` respectively. The presence of any of these characters, such as `r`, indicates that the particular permission is granted. The dash (`-`) symbol in place of a character in a permission set indicates that a particular permission is denied.

### Owner Permissions

The owner permission set determines the type of access the owner has for the file or directory.

The three characters in this set of permissions represent read, write, and execute permissions for the owner.

### Group Permissions

The group permission set determines the type of shared file access for each user belonging to the file owner's group. A group is a collection of users who can access files owned by the group, based on the permission set.

The three characters in this set of permissions represent read, write, and execute permissions.

**Note –** The system administrator creates and maintains groups in the `/etc/group` file. The system administrator assigns users to groups according to the need for shared file access.

### Other Permissions

The other permission set determines the type of access for all other users who have access to the system, but who do not own the file or directory, and are not a member of the file's or directory's group.

## Permission Characters and Sets

The read, write, and execute permissions are interpreted differently when assigned to a file than when assigned to a directory.

Table 7-2 shows the permission definitions.

**Table 7-2** Permission Characters

| Permission | Character | Access for a File | Access for a Directory |
|---|---|---|---|
| Read | `r` | You can display file contents and copy the file. | You can list the directory contents with the `ls` command. |
| Write | `w` | You can modify the file contents. | You can modify the contents of a directory, such as deleting a file. You must also have the execute permission for this to happen. |
| Execute | `x` | You can execute the file if it is an executable. You can execute a shell script if you also have read and execute permissions. | You can use the `cd` command to access the directory. If you also have read access, you can run the `ls -l` command on the directory to list contents. If you do not have read access you can run the `ls` command as long as you know the file name. |

**Note –** For a directory to be of general use, it must at least have read and execute permissions.

Table 7-3 shows examples of different permission sets for files and directories.

**Table 7-3** Permission Sets

| Permissions | Description |
|---|---|
| `-rwx------` | This file has read, write, and execute permissions set for the file owner only. Permissions for group and other are denied. |
| `dr-xr-x---` | This directory has read and execute permissions set for the directory owner and the group only. |
| `-rwxr-xr-x` | This file has read, write, and execute permissions set for the file owner. Read and execute permissions are set for the group and other. |

When you create a new file or directory, the Solaris OS assigns initial permissions automatically. The initial default permissions for a file and directory are then modified based on the default `umask` value set on the host.

**Note –** You can assign execute permissions on files with the `chmod` command. The `chmod` command is described later in this module. Execute permissions are not assigned by default when you create a file.

# Determining File or Directory Access

The following sections describe how to use the `ls -n` command in the Solaris OS to determine ownership of files and directories.

## Using the `ls -n` Command

All files and directories have an associated user identification number (UID) and a group identification number (GID). The UID identifies the user who owns the file or directory. The GID identifies the group of users who own the file or directory. A file or directory can belong to only one group at a time. The Solaris OS uses these numbers to track ownership and group membership of files and directories.

To view the UIDs and GIDs, run the `ls -n` command on the `/var/adm` directory.

```
$ ls -n /var/adm
total 244
drwxrwxr-x  5 4       4           512 Nov 15 14:55 acct
-rw-------  1 5       2             0 Jun  7 12:28 aculog
drwxr-xr-x  2 4       4           512 Jun  7 12:28 exacct
-r--r--r--  1 0       0        308056 Nov 19 14:35 lastlog
drwxr-xr-x  2 4       4           512 Jun  7 12:28 log
-rw-r--r--  1 0       0          6516 Nov 18 07:48 messages

... (output truncated)
```

Figure 7-3 describes the fields in the output of the `ls -n` command.

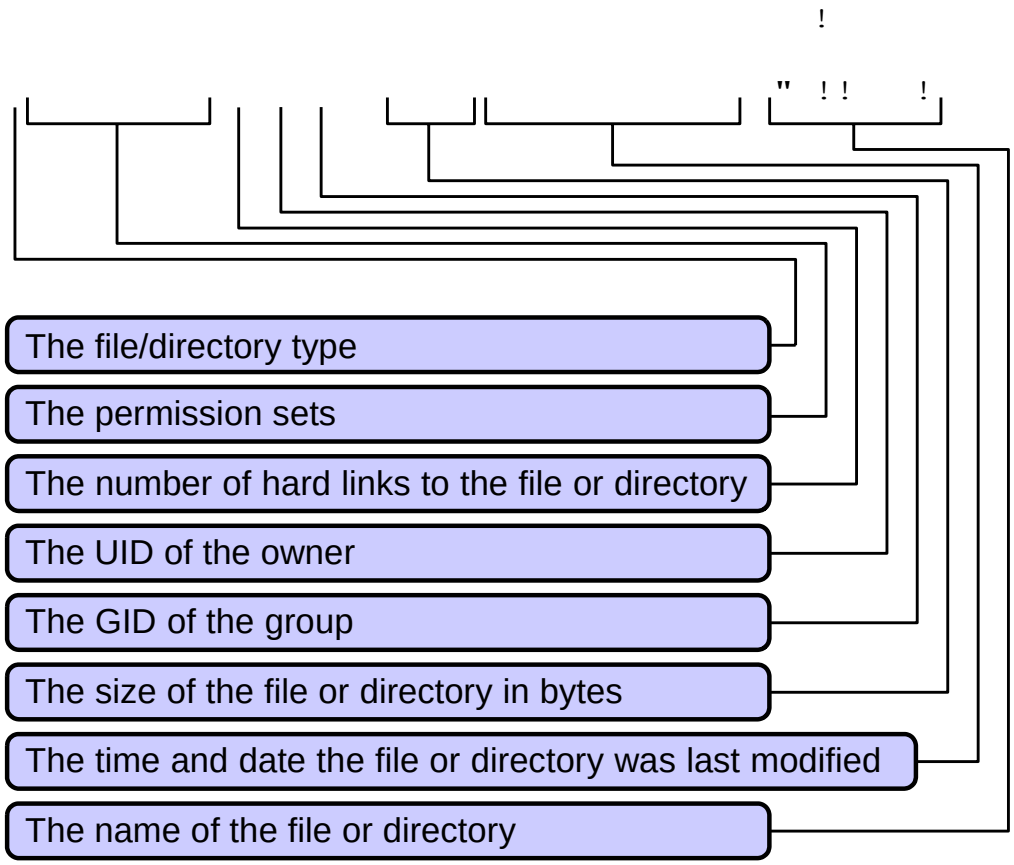


**Figure 7-3** Description of the Output of the `ls -n` Command

**Note –** A hard link is a pointer that shows the number of files or directories a particular file is linked to within the same file system.

## Determining Permissions

When a user attempts to access a file or directory, the Solaris OS compares the UID of the user to the UID of the file or directory. If the UIDs match, the permission set for the owner determines whether the owner has access to the file or directory.

If the UIDs do not match, the Solaris OS compares the user's GID and the GID of the file or directory. If these numbers match, the group permissions apply.

If the GIDs do not match, the Solaris OS uses the permission set for other to determine file and directory access.

Figure 7-4 shows the decision tree for determining file and directory permissions.

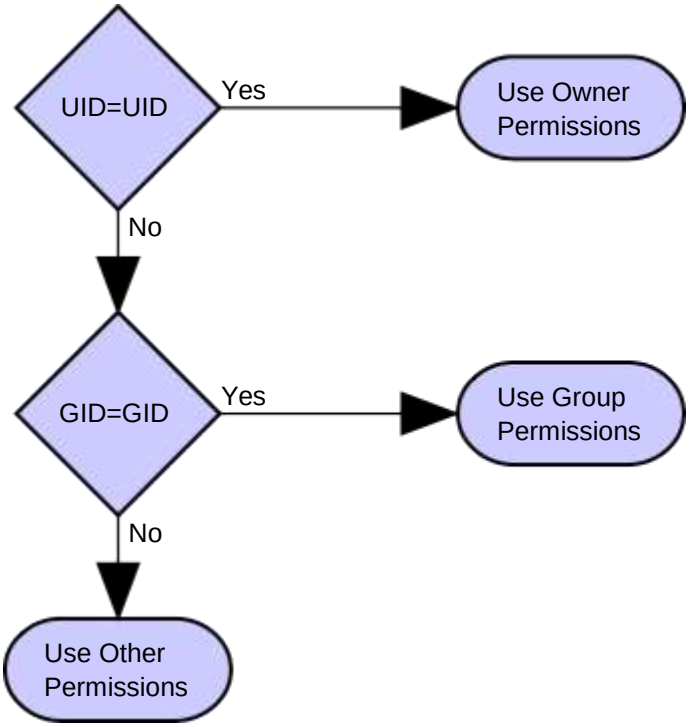


**Figure 7-4** Process for Determining Permissions

# Changing the Permissions

You can change the permissions set on files or directories using the `chmod` command. Either the owner of the file or directory or the `root` user can use the `chmod` command to change permissions.

## Permission Modes

The `chmod` command can change permissions specified in either symbolic mode or octal mode.

- Symbolic mode — uses combinations of letters and symbols to add or remove permissions for each type of user.
- Octal mode — uses octal numbers to represent each permission. Octal mode is also referred to as absolute mode.

## Changing Permissions in Symbolic Mode

The syntax for the `chmod` command in the symbolic mode is:

```
chmod symbolic_mode filename
```

The *`symbolic_mode`* option consists of three parts: the user category (owner, group, or other) affected, the function performed, and the permissions affected. For example, if the option is `g+x`, the executable permission is added for the group.

Figure 7-5 shows the *symbolic_mode* options.

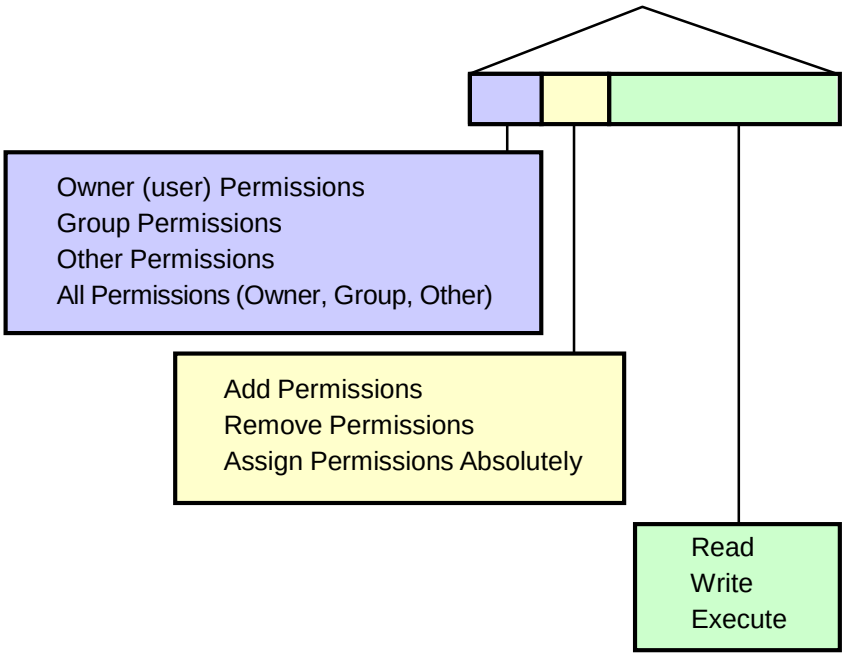


**Figure 7-5** Symbolic Mode Command Syntax

The following examples show you how to modify permissions on files and directories using symbolic mode.

To remove the read permission for other users, run the following commands:

```
$ ls -l dante

-rw-r--r--  1 student    class      1319 Jan 22 14:51 dante
$ chmod o-r dante
$ ls -l dante
-rw-r-----  1 student    class      1319 Jan 22 14:51 dante
$
```

To remove the read permission for the group, run the following commands:

```
$ chmod g-r dante
$ ls -l dante
-rw-------  1 student    class      1319 Jan 22 14:51 dante
$
```

To add an execute permission for the owner and a read permission for the group and other, run the following commands:

```
$ chmod u+x,go+r dante
$ ls -l dante
-rwxr--r--  1 student    class       1319 Jan 22 14:51 dante
$
```

To assign read and write permissions for owner, group, and other, run the following commands:

```
$ chmod a=rw dante
$ ls -l dante
-rw-rw-rw-  1 student    class       1319 Jan 22 14:51 dante
$
```

## Changing Permissions in Octal Mode

The `chmod` command syntax in octal mode is:

chmod *octal_mode filename*

The *octal_mode* option consists of three octal numbers, from 0–7, which represent a combination of permissions for the file or directory.

Table 7-4 shows the octal numbers for each individual permission.

**Table 7-4** Assigned Octal Values for Permissions

| Octal Value | Permission |
|---|---|
| 4 | Read |
| 2 | Write |
| 1 | Execute |

These numbers are combined into one number for each permission set.

Table 7-5 shows the octal numbers that represent a combined set of permissions.

**Table 7-5** Octal Digits for Permission Sets

| Octal Value | Permission Sets | Binary |
|---|---|---|
| 7 | `rwx` | 111 (4+2+1) |
| 6 | `rw-` | 110 (4+2+0) |
| 5 | `r-x` | 101 (4+0+1) |
| 4 | `r--` | 100 (4+0+0) |
| 3 | `-wx` | 011 (0+2+1) |
| 2 | `-w-` | 010 (0+2+0) |
| 1 | `--x` | 001 (0+0+1) |
| 0 | `---` | 000 (0+0+0) |

You can modify the permissions for each category of users by combining octal numbers. The first octal number defines owner permissions, the second octal number defines group permissions, and the third octal number defines other permissions.

Table 7-6 shows the permission sets for the three-digit octal numbers.

**Table 7-6** Combined Values and Permissions

| Octal Mode | Permissions |
|---|---|
| 644 | `rw-r--r--` |
| 751 | `rwxr-x--x` |
| 775 | `rwxrwxr-x` |
| 777 | `rwxrwxrwx` |

The `chmod` command fills in any missing octal digits to the left with zeros. For example:

```
$ chmod 44 dante
$ ls -l dante
----r--r-- 1 student    class       1319 Jan 22 14:51 dante
$
```

The previous example shows that the `chmod 44 dante` command is interpreted as `chmod 044 dante`

**Caution –** Not using the correct octal values or leaving off one or more of the values can lead to unwanted access to files or directories.

The following examples show how to modify permissions on files and directories using the octal mode.

**Note –** Each example builds on the resulting permissions from the previous example.

To set permissions so that the owner, group, and other have read and execute access only, enter the following commands:

```
$ chmod 555 dante
$ ls -l dante
-r-xr-xr-x  1 student    class       1319 Jan 22 14:51 dante
$
```

To change owner and group permissions to include write access, enter the following commands:

```
$ chmod 775 dante
$ ls -l dante
-rwxrwxr-x  1 student    class       1319 Jan 22 14:51 dante
$
```

To change the group permissions to read and execute only, enter the following commands:

```
$ chmod 755 dante
$ ls -l dante
-rwxr-xr-x  1 student    class       1319 Jan 22 14:51 dante
$
```

# Modifying Default Permissions

Every new file or directory has a set of default permissions assigned to it at the time of creation. The user mask affects the default file permissions assigned to the file or directory. You can set the user mask using the `umask` command in a user initialization file. You can modify the default permissions set at the time of creation, using the `umask` utility.

## The `umask` Utility

The `umask` utility affects the initial permissions for files and directories when the files and directories are created. The `umask` utility is a three-digit octal value that is associated with the read, write, and execute permissions. The first digit determines the default permissions for the owner, the second digit determines the default permissions for the group, and the third digit determines the default permissions for other.

In the Solaris OS, the default `umask` value is `022`.

To view the `umask` value, run the `umask` command:

```
$ umask
022
$
```

The Solaris OS assigns initial permission values automatically when files and directories are created.

The initial permission value specified by the system at the time of file creation is `666` (`rw-rw-rw-`).

The initial permission value specified by the system for a directory at the time of its creation is `777` (`rwxrwxrwx`).

To determine the `umask` value you want to set, remove the value of the permissions you want from `666` (for a file) or `777` (for a directory). The remainder is the value to use with the `umask` command. For example, suppose you want to change the default mode for files to `644` (`rw-r--r--`). The difference between `666` and `644` is `022`, which is the value you would use as an argument to the `umask` command.

Table 7-7 shows the file and directory permissions that are created for each of the `umask` octal values. You can also determine the `umask` value you want to set using the values listed in the table below.

**Table 7-7** File and Directory Permissions for `umask` Values

| **`umask` Octal Value** | **File Permissions** | **Directory Permissions** |
|---|---|---|
| 0 | `rw-` | `rwx` |
| 1 | `rw-` | `rw-` |
| 2 | `r--` | `r-x` |
| 3 | `r--` | `r--` |
| 4 | `-w-` | `-wx` |
| 5 | `-w-` | `-w-` |
| 6 | `---` | `--x` |
| 7 | --- | `---` (none) |

To set the default file permissions in a user initialization file to `rw-rw-rw-`, run the following command:

```
umask 000
```

## Applying the `umask` Utility

You can calculate the default permissions for new files and directories by applying the `umask` value to the initial value specified by the system in the octal mode.

For example, the initial permissions for a new file in the symbolic mode are as follows:

```
rw-rw-rw-
```

This set of permissions corresponds to read-write access for the owner, group, and other. This value is represented in the octal mode:

```
420420420 or 666
```

Use the default `umask` value of `022` to mask out the write permission for the group and other.

The result in the octal mode is:

```
420400400 or 644
```

The result in the symbolic mode is derived, as shown in Table 7-8.

**Table 7-8** Symbolic Mode Permission Fields

| Permission Field | Description |
|---|---|
| `rw-rw-rw-` | Initial value specified by the system for a new file |
| `----w--w-` | Default `umask` utility value to be removed |
| `rw-r--r--` | Default permissions assigned to newly created files |

When you mask out certain permissions from the initial value, the default permissions assigned to the new files remain.

All newly created files are assigned read and write access for the owner, and read access for the group and other.

You can apply this same process to determine the default permissions when you create new directories.

For directories, the initial value specified by the system is:

```
rwxrwxrwx
```

This corresponds to read, write, and execute access for the owner, group, and other. This value is represented in octal mode as:

```
421421421 or 777
```

Use the default `umask` value of `022` to mask out the write permission for the group and other.

The result in the octal mode is:

```
421401401 or 755
```

The result in the symbolic mode is derived as shown in Table 7-9.

**Table 7-9** Symbolic Mode Permission Fields

| Permission Field | Description |
|---|---|
| `rwxrwxrwx` | Initial value specified by the system for a new directory |
| `----w--w-` | Default `umask` utility value to be removed |
| `rwxr-xr-x` | Default permissions set for newly created directories |

When you mask out certain permissions from the initial value, the default permissions assigned to the new directories remain.

All newly created directories are assigned read, write, and execute access for the owner, and read and execute access for the group and other.

## Changing the `umask` Value

Some users require a more secure `umask` value of `027`, which assigns the following access permissions to newly-created files and directories.

- Files have read and write permissions for the owner, read permission for the group, and no permissions for other.

  `rw-r-----`

- Directories have read, write, and execute permissions for the owner, read and execute permissions for the group, and no permissions for other.

  `rwxr-x---`

You can change the `umask` value to a new value on the command-line. For example, to change the `umask` value to `027` and verify the new value, run the command:

```
$ umask 027
$ umask
027
$
```

The new `umask` value affects only those files and directories that are created from this point forward. However, if the user logs out of the system, the new value (`027`) is replaced by the old value (`022`) on subsequent logins because the `umask` value was changed using the command-line.

# Exercise: Changing File Permissions

In this exercise, you practice reading permissions on files and changing permissions using the symbolic mode and the octal mode.

## Preparation

Ensure that the umask value is set to 022 on your system. If not, set the umask value to 022 by running the following command:

```
$ umask 022
```

### RLDC

In addition to being able to use local classroom equipment, this lab was designed to also use equipment located in a remote lab data center. Directions for accessing and using this resource can be found at:

`http://remotelabs.sun.com/`

Ask your instructor for the particular SSH configuration file that you should use to access the appropriate remote equipment for this exercise.

## Tasks

To view permissions on files and change permissions, complete the following steps:

1. Perform the following commands in your home directory:

```
$ mkdir perm
$ cd /etc

$ ls -l group motd shadow vfstab
$ cp group motd shadow vfstab ~/perm
```

When trying to copy the `shadow` file, the error message `cp: cannot open shadow: Permission denied` appears. Why?

_____

_____

```
$ ls -l ~/perm
$ cd
$ cp -r /etc/skel perm
```

2. Change to the `perm` directory, and list the contents of the directory.

```
$ cd perm
$ ls -l
```

Fill in the permission sets for each file in Table 7-10, and write the three-digit octal number that represents the combined set of permissions.

**Table 7-10** Permission Sets

| File or Directory | Permissions | | | Octal Value |
|---|---|---|---|---|
| | Owner | Group | Other | |
| `group` | | | | |
| `motd` | | | | |
| `skel` | | | | |
| `vfstab` | | | | |

3. Create a new file and a new directory.
   a. What are the default permissions given to the new file?

      _____

   b. What are the default permissions given to the new directory?

      _____

4. Distinguish between the symbolic mode and the octal mode.

   _____

   _____

5. Using the symbolic mode, add write permission for the group to the `motd` file.

   _____

6. Using the octal mode, change the permissions on the `motd` file to `-rwxrw----`.

   _____

7. Using the octal mode, add write permission for other on the file named `group`.

   _____

8. Identify the GID and UID for the `motd` file. Which command did you use?

_____

_____

9. Create a new file called `memo` in your `dir4` directory.

_____

10. Remove the read permission for the owner from the `memo` file in the `dir4` directory. Use either the symbolic mode or the octal mode.

_____

a. What happens when you try to use the `cat` command to view the `memo` file?

_____

b. What happens when you try to copy the `memo` file?

_____

_____

11. What is the function of the `umask` utility? What is the default `umask` that exists on your system?

_____

12. Change the `umask` to `027`. Which command did you run?

_____

13. Create a new file and a new directory. Record the access permissions. Which command did you run?

_____

_____

14. Change the `umask` back to `022`.

15. Create a new file and a new directory.

```
$ touch test2file
$ mkdir test2dir
```

16. Record the access permissions.

```
$ ls -l test2file

$ ls -ld test2dir
```

# Exercise Summary


**Discussion –** Take a few minutes to discuss what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

# Exercise Solutions: Changing File Permissions

To view permissions on files and change permissions, complete the following steps:

1. Perform the following commands in your home directory:

```
$ mkdir perm
$ cd /etc

$ ls -l group motd shadow vfstab
-rw-r--r--  1 root     sys          290 Jun  5 09:40 group
-rw-r--r--  1 root     sys           49 Apr  6  2002 motd
-r--------  1 root     sys          795 Jun 28 15:25 shadow
-rw-r--r--  1 root     sys          257 Jun  5 10:01 vfstab
$ cp group motd shadow vfstab ~/perm
```

When trying to copy the `shadow` file, the error message `cp: cannot open shadow: Permission denied` appears. Why?

*Only the owner of this file, who is the `root` user, has read permission.*

```
$ ls -l ~/perm
-rw-r--r--  1 root     sys          290 Jun  5 09:40 group
-rw-r--r--  1 root     sys           49 Apr  6  2002 motd
-rw-r--r--  1 root     sys          257 Jun  5 10:01 vfstab

$ cd
$ cp -r /etc/skel perm
```

2. Change to the `perm` directory, and list the contents of the directory.

```
$ cd perm
$ ls -l
-rw-r--r--  1 student    class        290 Jul 29 14:34 group
-rw-r--r--  1 student    class         49 Jul 29 14:34 motd
drwxr-xr-x  2 student    class        512 Jul 29 14:34 skel
-rw-r--r--  1 student    class        420 Jul 29 14:34 vfstab
```

Fill in the permission sets for each file in Table 7-11 on page 7-25, and write the three-digit octal number that represents the combined set of permissions.

**Table 7-11** Permission Sets

| File or Directory | Permissions | | | Octal Value |
|---|---|---|---|---|
| | Owner | Group | Other | |
| group | rw- | r-- | r-- | 644 |
| motd | rw- | r-- | r-- | 644 |
| skel | rwx | r-x | r-x | 755 |
| vfstab | rw- | r-- | r-- | 644 |

3. Create a new file and a new directory.

   a. What are the default permissions given to the new file?

rw-r--r--

   b. What are the default permissions given to the new directory?

rwxr-xr-x

4. Distinguish between the symbolic mode and the octal mode.

   *The symbolic mode uses combinations of letters and symbols to add or remove permissions for each type of user.*

   *The octal mode uses octal numbers to represent each permission. The octal mode is also referred to as the absolute mode.*

5. Using the symbolic mode, add write permission for the group to the `motd` file.

```
$ chmod g+w motd
$ ls -l
```

6. Using the octal mode, change the permissions on the `motd` file to `-rwxrw----`.

```
$ chmod 760 motd
$ ls -l
```

7. Using the octal mode, add write permission for other on the file named `group`.

```
$ chmod 646 group
$ ls -l
```

8. Identify the GID and UID for the `motd` file. Which command did you use?

```
$ ls -n motd

-rwxrw----  1  11001    10         49 Nov 19 15:16 motd
```

9. Create a new file called `memo` in your `dir4` directory.

```
$ touch ~/dir4/memo
```

10. Remove the read permission for the owner from the `memo` file in the `dir4` directory. Use either the symbolic mode or the octal mode.

```
$ chmod u-r ~/dir4/memo
```

*or*

```
$ chmod 244 ~/dir4/memo
```

a. What happens when you try to use the `cat` command to view the `memo` file?

*You cannot use the* `cat` *command, because read permission has been removed for the user. Even though you are part of the group, the permissions are viewed in the order in which they appear. The following message appears:*

```
cat: cannot open /export/home/student/dir4/memo
```

b. What happens when you try to copy the `memo` file?

*You cannot copy the file, because the user has no read permission. The following message appears:*

```
cp: cannot open /export/home/student/dir4/memo: Permission denied
```

11. What is the function of the `umask` utility? What is the default `umask` that exists on your system?

*The* `umask` *utility modifies the default permissions set for files and directories at the time of creation. To view the default* `umask` *value on your system, run the* `umask` *command.*

```
$ umask
022
$
```

12. Change the `umask` to `027`. Which command did you run?

```
$ umask 027
```

13. Create a new file and a new directory. Record the access permissions. Which command did you run?

```
$ touch testfile
$ mkdir testdir
$ ls -l testfile
-rw-r-----
$ ls -ld testdir
drwxr-x---
```

14. Change the `umask` back to `022`.

```
$ umask 022
```

15. Create a new file and a new directory.

```
$ touch test2file
$ mkdir test2dir
```

16. Record the access permissions.

```
$ ls -l test2file
-rw-r--r--
$ ls -ld test2dir
drwxr-xr-x
```

# Notes: