6

Using Basic File Permissions

Objectives

After completing this lesson, you should be able to:

- View file and directory permissions
- Change ownership
- Change permissions
- Modify default permissions

Lesson Agenda

- Viewing File and Directory Permissions
- Changing Ownership
- Changing Permissions
- Modifying Default Permissions

Securing Files and Directories

- One of the important functions of a secure system is to limit access to authorized users and prevent unauthorized users from accessing the files or directories.
- UNIX and Linux use two basic means to prevent unauthorized access to a system:
  - To authenticate both a privileged user account and an unprivileged user account by verifying that the username and password exist and have been correctly entered
  - To protect file and directory access, the UNIX and Linux OSes assign a standard set of access permissions at the time of file and directory creation. These permissions are called an Access Control List (ACL).

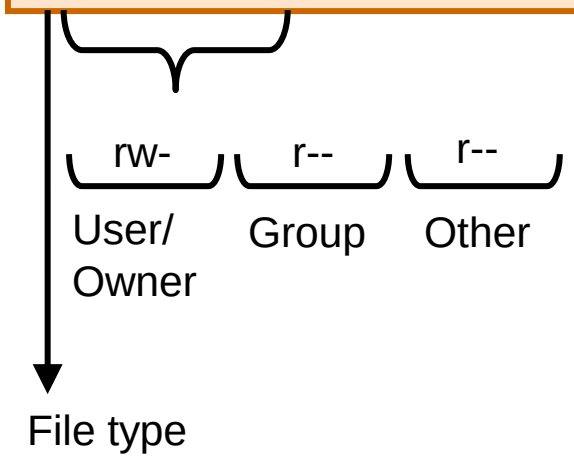
File and Directory Permissions (ACL)

- All files and directories in UNIX and Linux have a default set of standard access permissions.
- These access permissions control who can access what files, and provides a fundamental level of security to the files and directories in a system.
- The standard set of access permissions are established by a user's `umask` settings. The `umask` command is described in more detail later in this lesson.

Viewing Permission Categories

To view the permissions for files and directories, use the `ls -l` or `ls -n` commands.

```
$ ls -l dante
-rw-r--r-- 1 oracle        1319 Mar 15 11:23 dante
```



`r` = Readable
`w` = Writeable
`x` = Executable
`-` = No permission

Permission Groups

- There are three permission groups:
  - User (Owner) who owns the file or directory
  - Group
  - Other
- The table describes the permission groups and their scope:

Permission Groups	Description
User/Owner (u)	Permissions used by the assigned user/owner of the file or directory
Group (g)	Permissions used by members of the group that owns the file or directory
Other (o)	Permissions used by **all** users other than the file owner, and members of the group that owns the file or the directory

Permission Sets

- Each permission group has three permissions, called a permission set.
- Each set consists of **r**ead, **w**rite, and e**x**ecute permissions that are represented by the characters `r`, `w`, and `x`, respectively.
- Each file or directory has three permission sets for the three types of permission groups.
- The first permission set represents the user/owner permissions, the second set represents the group permissions, and the last set represents the other (world) permissions.
- The presence of any of these characters, such as `r`, indicates that the particular permission is granted.
- A dash (–) symbol in place of a character in a permission set indicates that a particular permission is denied.
- UNIX and Linux assign initial permissions automatically based on a user's `umask` when a new file or directory is created.

Interpreting File and Directory Permissions

Permissions	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, but only if you also have **read** permissions.	You can modify the contents of a directory, by deleting (`rm`) 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 the contents. If you do not have **read** access, you can run the `ls` command as long as you know the file name.

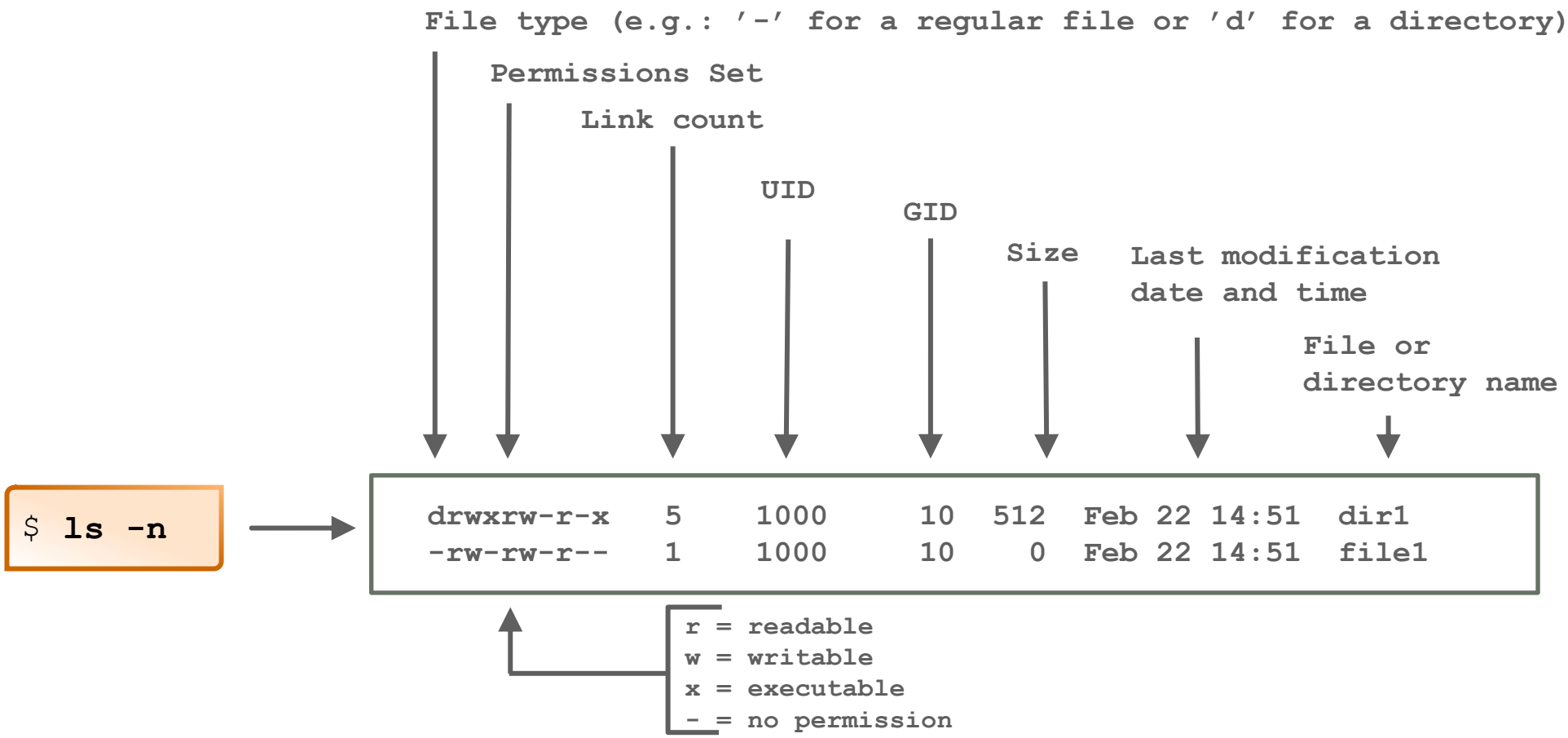
Determining File or Directory Access Permissions

- The `ls -l` and `ls -n` commands display the ownership of files and directories and their corresponding permissions.
- All files and directories have an associated `username` and a user identification number (UID) and a `group` name and a group identification number (GID).
- To view the UIDs and GIDs, run the `ls -n` command on the `/var/adm` directory.

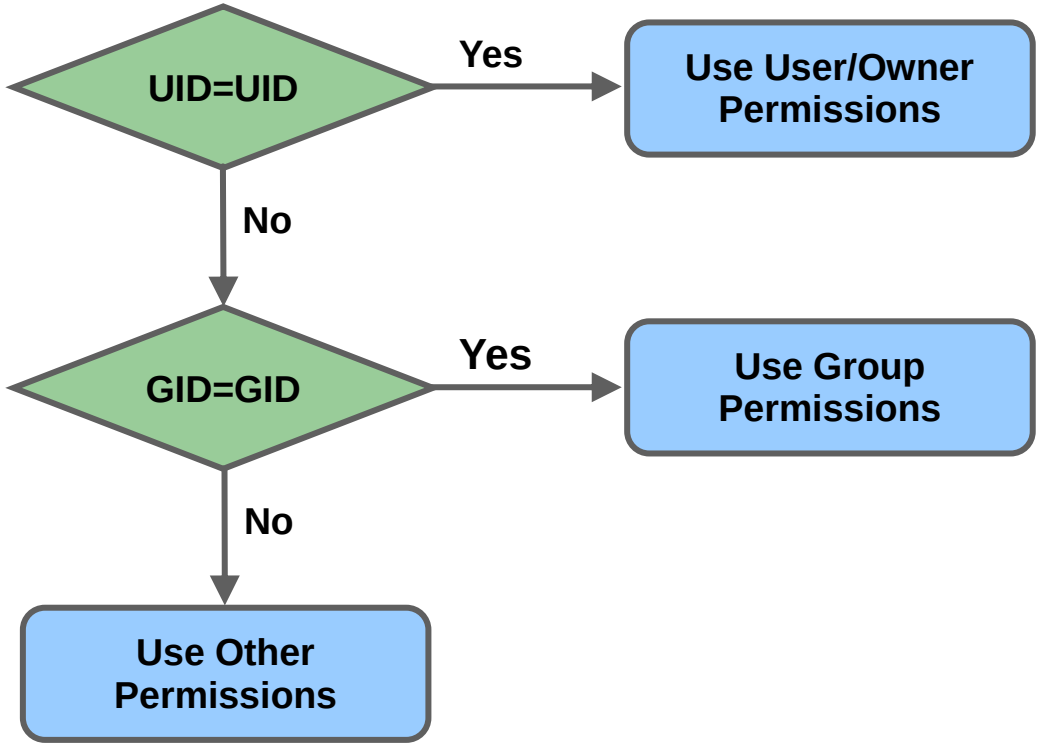
```
$ ls -n /var/adm
total 244
drwxrwxr-x     5 4     4         512 Nov 15 14:55 acct
-rw-------     1 5     2           0 Jun  7 12:28 aculog
drwxr-xr-x     2 4     4         512 Jun  7 12:28 exacct
-r--r--r--     1 0     0      308056 Nov 19 14:35 lastlog
drwxr-xr-x     2 4     4         512 Jun  7 12:28 log
...(output truncated)
```

Interpreting the `ls -n` Command

- The `ls -n` command displays the UID and GID listing of file information.



Determining Permissions



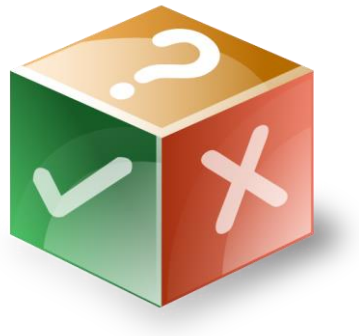
Quiz

Which of the following directories have read and execute permissions set for the owner and group only?

a. `dr-xr-x---`

b. `dr-x---r-x`

c. `d---r-xr-x`



Lesson Agenda

- Viewing File and Directory Permissions
- Changing Ownership
- Changing Permissions
- Modifying Default Permissions

Changing Ownership on Files or Directories

- Every file and directory in UNIX and Linux is owned by somebody.
- The `ls -l` command shows the `username` and the `group` that owns the object.
- The `ls -n` command shows the `UID` and `GID` numbers corresponding to who owns the object.
- There are two commands:
  - The `chown` command can be used to change both `username` and `group` ownership.
  - The `chgrp` command changes only `group` ownership.

Changing Both `username` and `group` Ownership

- The syntax for the `chown` command is:

```
$ chown [options] [newusername][:newgroup] filename
```

```
$ ls -l dante
-rw-r--r-- 1 student  class     1319 Mar 15 11:23 dante
$ chown oracle:oracle dante
$ ls -l dante
-rw-r--r-- 1 oracle   oracle    1319 Mar 15 11:23 dante
```

- You can change the ownership only for files and directories that you own. However, the system administrator can change the ownership of any object.
- For more information about the `chown` command options, see the `chown` man pages.

Caution: If you change the `username` ownership of a file or directory, you have just given that object away, and you cannot get it back without help from the system administrator.

Changing `group` Ownership

- The syntax for the `chgrp` command is:

```
$ chgrp [options] newgroup filename
```

- For more information about the `chgrp` command options, see the `chgrp` man pages.

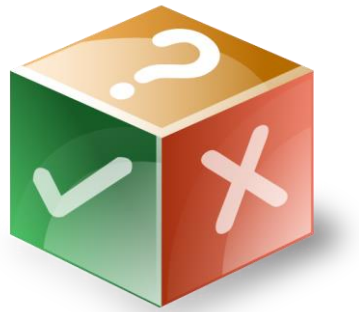
Note: If you still own a file or directory, you can always change the `group` ownership.

Quiz

You can give away a file or directory ownership that you cannot get back.

a. True

b. False



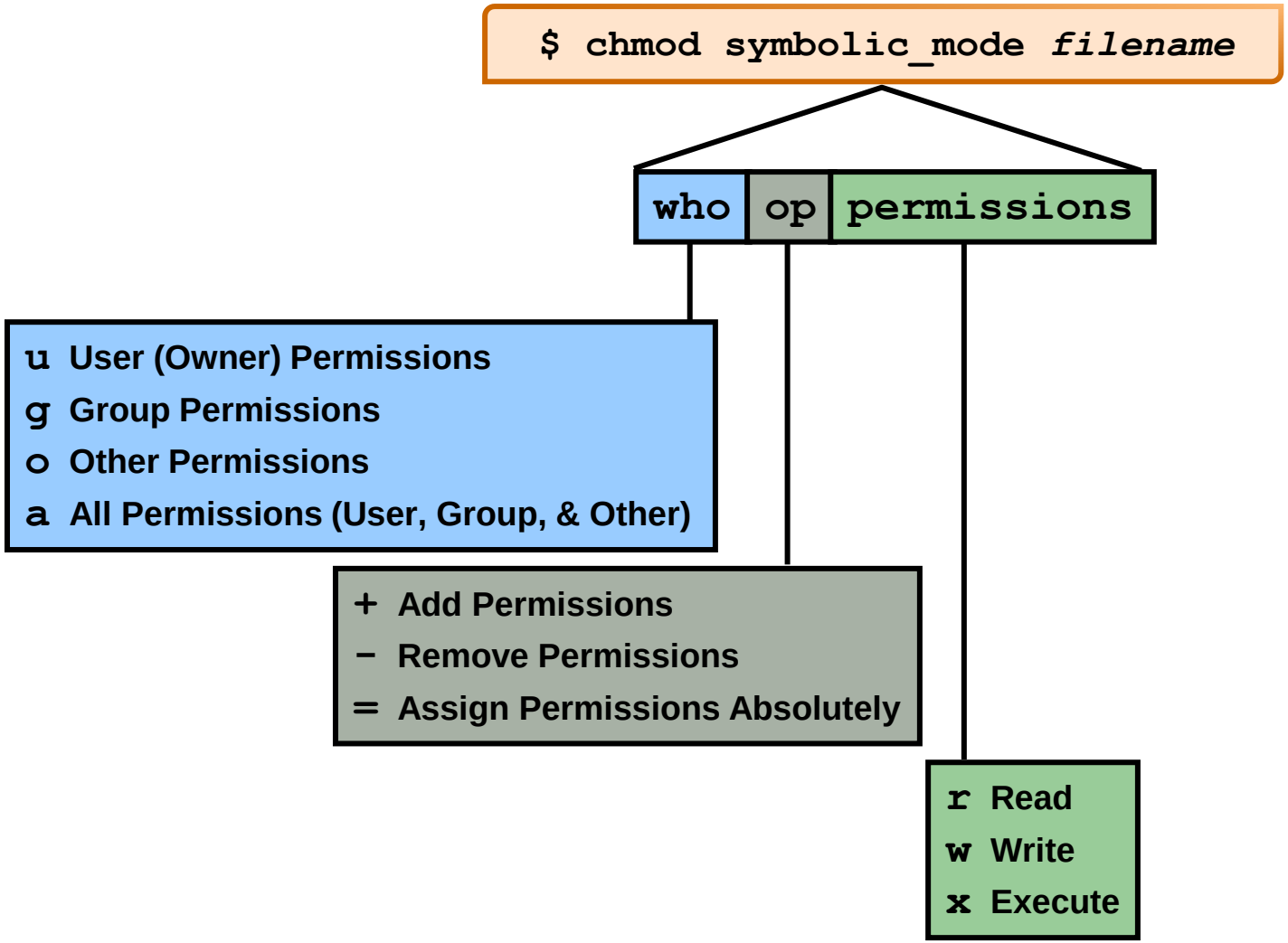
Lesson Agenda

- Viewing File and Directory Permissions
- Changing Ownership
- Changing Permissions
- Modifying Default Permissions

Changing Permissions

- You can change the permissions on files and directories by using the `chmod` command.
- Either the user/owner of the file or directory, or the `root` user can use the `chmod` command to change permissions.
- The `chmod` command can be used in either symbolic or octal mode.
  - **Symbolic mode** uses a combination of letters and symbols to add or remove permissions for each permission group.
  - **Octal mode**, also called absolute mode, uses octal numbers to represent each permission group.

Changing Permissions: Symbolic Mode



Changing Permissions: Symbolic Mode

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

```
$ chmod [options] [symbolic_mode] filename
```

- The format of the `symbolic_mode` consists of three parts: `[ugoa] [+-=] [rwx]`
  - The user category `[ugoa]`: User/owner, group, other, or all
  - The function to be performed `[+-=]`: Add, remove or set equal
  - The permissions affected `[rwx]`: Read, write, and execute
    Plus special file permissions and *sticky bit* `[st]` (described in the next slide)
- If the option is `g+x`, the executable permission is added to the *group* permissions.
- For more information about the `chmod` command options, see the `chmod` man pages.

Special File Permissions: Setuid, Setgid, and Stick Bit

- When special file permissions are set on executable files, the `user` who runs that executable file executes it with the permissions of the UID or GID, who has the "`s`" in place of the "`x`" for executable privileges.

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x   1  root     root  27856 May    4  2014 /usr/bin/passwd
```

- The restricted deletion flag "`t`" (sticky bit) is a permission bit that protects the files within a directory from being deleted by anyone, except the user who owns the file, the owner of the directory, or the `root` user.

```
$ ls -l /tmp
drwxrwxrwt  68  root     root   4096 Apr   15 07:02 /tmp
```

- Typically, only the `/tmp` directory has the "`t`" in place of the "`x`", which makes it "world readable and world writable".

Changing Permissions: Octal Mode

- The syntax for the `chmod` command in octal mode is:

```
$ chmod [octal_mode] filename
```

- The `octal_mode`, sometime called the `absolute_mode`, option consists of three octal numbers, 4, 2, and 1, that represent a combination (sum) of the permissions, from 0–7, for the file or directory.

Octal Value	Permission
4	Read
2	Write
1	Execute

Changing Permissions: Octal Mode

Octal Value	Permission	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 (see the notes page)	011 (0+2+1)
2	-w- (see the notes page)	010 (0+2+0)
1	--x (see the notes page)	001 (0+0+1)
0	---	000 (0+0+0)

Changing Permissions: Octal Mode

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

Octal Mode	Permissions
640	`rw-r-----`
644	`rw-r--r--`
750	`rwxr-x---`
751	`rwxr-x--x (see the notes page)`
755	`rwxr-xr-x`
777	`rwxrwxrwx`

Changing Permissions: Octal Mode

- Set permissions so that the owner, group, and other have read and execute access only.

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

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

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

Note: `chmod 44 dante` becomes `chmod 044 dante`.

Quiz

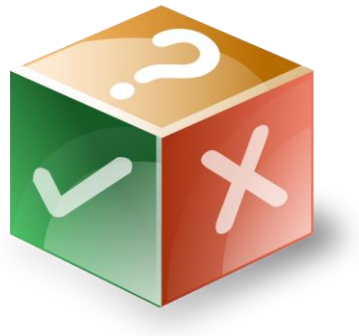
What is the correct octal value for the “write and execute” file permission?

a. 3

b. 5

c. 6

d. 7



Quiz

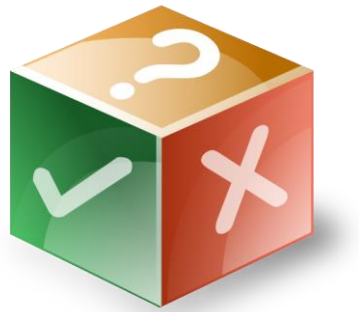
What is the correct permission set for the `rwxr-xr-x` octal mode?

a. 775

b. 644

c. 755

d. 674



Lesson Agenda

- Viewing File and Directory Permissions
- Changing Ownership
- Changing Permissions
- Modifying Default Permissions

Umask: A bash Shell Built-in Command

- When files and directories are created, initial permission values are automatically assigned.
- The initial maximum permission value for a file is 666 (rw-rw-rw-) and 777 (rwxrwxrwx) for a directory.
- The user mask affects and modifies the default file permissions assigned to the file or directory.
- You can set the user's mask by using the umask command in a user initialization file.
- To view the umask value, run the umask command.

```
$ umask
0022
```

Note: The default umask value for users in Oracle Solaris is 0022 (022), whereas the default value for Oracle Linux is 0002 (002).

Determining the `umask` Octal Value

umask Octal Value	File Permissions	Directory Permissions
0	rw-	rwx
1	rw-	rw-
2	r--	r-x
3	r--	r--
4	-w- (see the notes page)	-wx (see the notes page)
5	-w- (see the notes page)	-w- (see the notes page)
6	---	--x (see the notes page)
7	---	--- (none)

Applying the `umask` Value

- When you mask out certain permissions from the initial value, the default permissions assigned to the new files and directories remain.
- The table displays the results in symbolic mode.

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 (0022)
`-rw-r--r--`	Default permissions assigned to newly created files
`drwxrwxrwx`	Initial value specified by the system for a new directory
`w w`	Default umask utility value to be removed (0022)
`drwxr-xr-x`	Default permissions set for newly created directories

Changing the `umask` Value

- You can change the `umask` value to a new value on the command line.
- For example, you might require a more secure `umask` value of say `027`, which assigns the following access permissions to newly created files and directories:
  - Files with read and write permissions for the user/owner, read permission for the group, and no permissions for other (`rw-r-----`), octal value 640
  - Directories with read, write, and execute permissions for the user/owner, read and execute permissions for the group, and no permissions for other (`rwxr-x---`), octal value 750

```
$ umask 027
$ umask
0027
```

Note: The default `umask` is set in `/etc/profile` file.

Summary

In this lesson, you should have learned how to:

- View file and directory permissions
- Change ownership
- Change permissions
- Modify default permissions

Practice 6: Overview

This practice covers the following topics:

- 6-1: Changing File Ownership
- 6-2: Changing File Permissions
- 6-3: Modifying Default Permissions