Chapter 13: File Permissions and Ownership

Introduction to permissions

- Permissions allow users to protect files and directories
- Three permissions types: read, write and execute
- Permissions have different meaning on files vs directories

Files vs directories

Permission (Symbol)	Effect of File	Effect on Directory
read (r)	Allows for file contents to be read or copied.	Without execute permission on the directory, allows for a non-detailed listing of files. With execute permission, `ls -l` can provide a detailed listing.
write (w)	Allows for contents to be modified or overwritten.	Allows for files to be added or removed from directory. For this permission to work, the directory must also have execute permission.
execute (x)	Allows for file to be run as a process, although text script files require read permission, as well.	Allows user to change to the directory, if parent directories have execute permission, as well

Displaying permissions

- Permissions are displayed with `ls -l` command:

  $ ls -l /bin/ls
  -rwxr-xr-x 1 root root 118644 Apr 4 2014 /bin/ls

  First character: file type (- for file, d for directory)
  Next 9 characters: permissions
  Next 1 character: # of filenames referencing this file
  1st "root": file owner
  2nd "root": the primary group name the owner belongs to
  118644: file size in bytes
  Apr 4 2014: date created or last modified
  /bin/ls: file or directory name

Permission sets

- Permissions are broken into three sets:
  - The first set for the user who owns the file displayed (**rwx**r-xr-x)
  - The second set for the group that owns the file displayed (rwx**r-x**r-x)
  - The last set for everyone else (rwxr-x**r-x**)
- The term "everyone else" means anyone who is not the user that owns the file or a member of the group that owns the file

Changing File Ownership

- When a file or directory is created, the owner is automatically assigned using the effective user ID at the time of creation
- The `chown` command is used to change the ownership of a file or directory
- File owner can only be changed by a user with root privileges:

```
# ls -l testfile
-rw-rw-r-- 1 user1 user1 212 Apr 4 2014 testfile
# chown root testfile
# ls -l testfile
-rw-rw-r-- 1 root user1 212  Apr 4 2014 testfile
```

Changing the Primary Group

- When a file or directory is created, the primary group of the effective user ID creating it is automatically assigned
- The `chgrp` command is used to change the primary group of a file or directory
- The primary group can only be changed by the file owner and a user with root privileges:

```
# ls -l testfile
-rw-rw-r-- 1 user1 user1 212 Apr 4 2014 testfile
# chgrp admins testfile
# ls -l testfile
-rw-rw-r-- 1 user1 admins 212  Apr 4 2014 testfile
```

Understanding Permissions

- Based on the following permissions:
  -rwxr-xr-x 1 root root 118644 Apr 4 2014 /bin/ls

  - If you are the root user, your permissions would be “rwx”, or read, write, and execute
  - If you are not the root user, but are a member of the root group, your permissions would be "r-x", or read and execute
  - If you are not the root user or a member of the root group, then your permissions would be "r-x", or read and execute

Changing Permissions

- The `chmod` command is used to change the permissions of a file or directory
- To change the permissions of a file, you must either be the user who owns the file or the root user
- Two methods used to change:
  - Symbolic (relative) method – uses a combination of letters and symbols to add or remove
  - Octal (numeric) method – uses three numbers to represent file permissions for owner, group, everyone else

The symbolic Method

- First, specify who (u, g, o, a):

Symbol	Meaning
u	The user who owns the file
g	The group who owns the file
o	People other than the user owner or member of the group owner (others)
a	To refer to the user, group and others (all)

- Next, specify an operator (+, =, -):

Symbol	Meaning
+	Add the permission, if necessary
=	Specify the exact permission
-	Remove the permission, if necessary

- Lastly, specify permission (r, w, x):

The symbolic method

- Examples:

Example	Meaning
chmod u+x myscript	Adds execute permission for the user owner
chmod g-w file1	Removes write permission from the group owner
chmod o=r,g-w,u+x myscript	Assigns others read, removes write from the group owner and adds execute permission from the user owner
chmod a=- file1	Assign everyone no permission

The octal method

- Uses numeric values for permissions

Permission	Octal Value
read (r)	4
write (w)	2
execute (x)	1

- Must always specify 3 values (owner, group, everyone else)

Example	Meaning
chmod 764 myscript	Results in rwxrw-r--
chmod 644 myfile	Results in rw-r--r--
chmod 744 myscript	Results in rwxr--r--
chmod 000 myfile	Results in ---------

Permission Scenarios

Advanced Permissions

Permission	Symbol	Octal Value	Purpose
setuid on a file	An "s" where normally you see the "x" for the user owner permissions, set with u+s	4000	Causes an executable file to execute under user owner identity instead of user running command.
setgid on a file	An "s" where normally you see the "x" for the group owner permissions, set with g+s	2000	Causes an executable file to execute under group owner identity instead of user running command.

Advanced Permissions

Permission	Symbol	Octal Value	Purpose
setgid on a directory	An "s" where normally you see the "x" for the group owner permissions, set with g+s	2000	Causes new files and directories that are created inside to be owned by the group that owns the directory.
sticky on a directory	A "t" where you normally see the "x" for the others permissions, set with o+t	1000	Causes files inside directory to be able to be removed only by the user owner, or the root user.

Default Permissions

- The `umask` command is used to set default permissions
- Only affects the permissions placed on new files and directories at the time they are created
- Does not affect the special advanced permissions of setuid, setgid or sticky bit

Default Permissions

- The `umask` command uses same numeric values as `chmod`: read=4, write=2, execute=1
- Value is subtracted from maximum permissions:

  rw-rw-rw- for files

  rwxrwxrwx for directories

Default Permissions Examples

umask	File permissions	Directory permissions	Description
002	664 or rw-rw-r--	775 or rwxrwxr-x	Default for ordinary users
022	644 or rw-r--r--	755 or rwxr-xr-x	Default for root user
007	660 or rw-rw----	770 or rwxrwx---	No access for others
077	600 or rw-------	700 or rwx------	Private to user