Lesson 4. Directory Ownerships and Permissions

Key Concepts

- Because directories are also files, they have a user owner, a group owner, and a set of permissions.
- Read permissions allow a user to list the contents of a directory.
- Write permissions allow a user to add or remove files.
- Execute permissions allow a user to access a file within the directory.
- Directory permissions are modified with the **chmod** command.

Discussion

When someone is using a file within Linux, they are generally either reading its information, modifying its information, or trying to execute the file as a script or application. Therefore the permission types already discussed, namely (r)ead, (w)rite, and e(x)ecute have very natural interpretations.

To Linux, a directory is just a special type of file, therefore it also has the same types of permissions ((r)ead, (w)rite, and e(x)ecute), a user owner, a group owner, and the same classes of access ((u)ser, (g)roup, and (o)ther.) However, directories are obviously used differently. Would it be meaningful to open a directory in an editor, such as **pico /home/elvis**? Because people use directories differently, directory permissions have different interpretations.

What do people do with directories? They list their contents with the **ls** command. They remove files from them, create new files within them, and move files from one directory to another. Directory permissions should allow a directory owner to control who can perform which of these operations.

Linux considers listing a directory's contents (as with the **ls** command) analogous to "read"ing a directory, and therefore someone must have (r)ead permissions to list its contents. Adding or removing a file from a directory is considered "write"ing to the directory, and therefore someone must have (w)rite permissions in order to shuffle files within the directory.

There is no reasonable analogy to "execute"ing a directory, so Linux doesn't try to define a similar behavior. Instead, the e(x)ecute permission controls a behavior for directories which has nothing to do with command execution. In order to access any file within a directory, a user must have e(x)ecute permission. This permission could more appropriately be called "access" permission, but because the third permission was already called "execute" for regular files, the same word (and letter) is used for directories as well. In order to refer to any file within a directory (including subdirectories!), a user must have e(x)ecute permissions.

The first row of the following table should look familiar. It restates how to interpret permissions for regular files, as presented earlier. A row for directories has been added in order to compare and contrast permission interpretation for both file types.

Table 4-1. Permissions for Regular Files and Directories

	(r)ead	(w)rite	e(x)ecute
regular file	view the file	modify the file	use the file as a command

	(r)ead	(w)rite	e(x)ecute
directory	list directory contents	add or remove files	access file within the directory

Examples

Example 1. New Directory Defaults

Newly created regular files are readable by everybody, but can only be modified by the user and group owner of the file. How are newly created directories handled? Consider nero, who is collecting census data from his various provinces. He decides to create a directory called `/tmp/census` to hold all of his data.

```
[nero@station nero]$ mkdir /tmp/census
[nero@station nero]$ ls -ld /tmp/census/
drwxrwxr-x    2 nero     nero         4096 Jan 16 15:33 /tmp/census/
```

Why did nero need to add the **-d** command line switch to the **ls** command?

Note that the default permissions for newly created directories are `rwxrwxr-x`. These permissions have the following implications:

1. Anyone can access files within the directory.
2. Anyone can list the files within the directory.
3. Only the directory owner (or members of the group owner) can add or remove files from within the directory.

For example, decides he would like to browse nero's census information. Notice that julius can browse the directories, and the files within the directories, but because of the default directory permissions, he cannot add or remove new files. Because of the default file permissions, he can view, but not modify, the contents of the files.

```
[julius@station julius]$ ls -al /tmp/census/
total 20
drwxrwxr-x    2 nero     nero         4096 Jan 16 15:48 .
drwxrwxrwt   23 root     root         4096 Jan 16 15:45 ..
-rw-rw-r--    1 nero     nero           42 Jan 16 15:48 egypt.dat
-rw-rw-r--    1 nero     nero           42 Jan 16 15:48 gaul.dat
-rw-rw-r--    1 nero     nero           42 Jan 16 15:47 iberia.dat
[julius@station julius]$ rm /tmp/census/iberia.dat
rm: remove write-protected regular file `/tmp/census/iberia.dat'? y
rm: cannot remove `/tmp/census/iberia.dat': Permission denied
[julius@station julius]$ echo "110 CE     42" > /tmp/census/thrace.dat
-bash: /tmp/census/thrace.dat: No such file or directory

[julius@station julius]$ cat /tmp/census/gaul.dat
110 CE          45430
120 CE          53200
130 CE          55820
[julius@station julius]$ echo "140 CE     583420" >> /tmp/census/gaul.dat
-bash: /tmp/census/gaul.dat: Permission denied
```

Example 2. Home Directories

Notice that a user's home directory does not follow the default permissions.

```
[nero@station nero]$ ls -ld ~
drwx------    3 nero     nero         4096 Jan 16 16:04 /home/nero
[nero@station nero]$ ls -l /home/
total 120
drwx------    3 alice    alice        4096 Jan 15 08:04 alice
drwx------    3 augustus augustus     4096 Jan 14 15:22 augustus
drwx------    3 austin   austin       4096 Jan 14 15:22 austin
drwx------    3 blondie  blondie      4096 Jan 14 13:46 blondie
...
[nero@station nero]$ ls -l ~augustus
ls: /home/augustus: Permission denied
```

In Red Hat Linux, home directories are "protected". By default, only the user that owns a home directory has access. Have you noticed that most of our exercises involving multiple users accessing a file have used the `/tmp` directory rather than a user's home directory? Why have we not used a user's home directory instead?

Example 3. Creating a `~/pub` Directory

Nero would now like to make his census data available to the world at large. In Red Hat Linux, there are generally only two places where users can create files, the `/tmp` directory and `~` (the user's home directory.) In the first example, nero chose to create a `census` directory within `/tmp`. In Red Hat Linux, however, the `/tmp` directory is "swept." If a file within the `/tmp` is not accessed for 10 days, it is removed from the system.

In order to create a permanent, publicly accessible location for his census data, nero chooses to create a public subdirectory within his home directory. Conventionally, such a subdirectory in Linux is often called `pub`. As the following sequences will reveal, sharing files from a user's home directory is not as easy as just creating a world (r)eadable and e(x)ecutable directory.

First nero creates the `~/pub` directory, and copies the census information from `/tmp/census` over to it.

```
[nero@station nero]$ mkdir pub
[nero@station nero]$ cp /tmp/census/* pub
[nero@station nero]$ ls -al /home/nero/pub/
total 20
drwxrwxr-x    2 nero     nero         4096 Jan 16 16:13 .          ❶
drwx------    4 nero     nero         4096 Jan 16 16:12 ..         ❷
-rw-rw-r--    1 nero     nero           42 Jan 16 16:13 egypt.dat
-rw-rw-r--    1 nero     nero           42 Jan 16 16:13 gaul.dat
-rw-rw-r--    1 nero     nero           42 Jan 16 16:13 iberia.dat
```

❶ Recall that "." always refers to the current directory, in this case `/home/nero/pub`.

❷ Recall that ".." always refers to the current directory's parent, in this case `/home/nero`.

Being the conscientious sort, nero double checks the permissions on the newly created directory and files. On `/home/nero/pub`, he finds permissions of `rwxrwxr-x`, implying that others can access and list

files from within the directory. The data files themselves have permissions of `rw-rw-r--`, implying that others have read access to the files, so all seems in order. He tells julius where to find the information.

Interested in the data, julius tries to access one of the files. Unfortunately, all does not go well:

```
[julius@station julius]$ cat /home/nero/pub/egypt.dat
cat: /home/nero/pub/egypt.dat: Permission denied
```

What has nero neglected? Recall that in order to access a file within a directory, including subdirectories, a user must have access permissions for the directory. The permissions on `/home/nero/pub` are fine, but consider the permissions on `/home/nero` (".." in the above listing, or listed again below):

```
[nero@station nero]$ ls -ld /home/nero/
drwx------    4 nero     nero         4096 Jan 20 14:05 /home/nero/
```

If julius were able to access `/home/nero/pub`, all would be fine. But because julius does not have access permissions for `/home/nero`, he cannot access `/home/nero/pub`! In order to create a publicly accessible directory within a home directory, a user must allow people to access their home directory. Nero fixes the problem in the following sequence of commands.

```
[nero@station nero]$ chmod o+x /home/nero/
[nero@station nero]$ ls -al /home/nero/pub/
total 20
drwxrwxr-x    2 nero     nero         4096 Jan 16 16:13 .
drwx-----x❶   4 nero     nero         4096 Jan 16 16:14 ..
-rw-rw-r--    1 nero     nero           42 Jan 16 16:13 egypt.dat
-rw-rw-r--    1 nero     nero           42 Jan 16 16:13 gaul.dat
-rw-rw-r--    1 nero     nero           42 Jan 16 16:13 iberia.dat
```

❶ Note that now others have e(x)ecute permissions to `/home/nero`.

julius tries again to examine the file. Note that because the permissions on `/home/nero/pub` allow others both (r)ead and e(x)ecute access, julius can both access files and get a directory listing.

```
[julius@station julius]$ cat /home/nero/pub/egypt.dat
110 CE          45430
120 CE          53200
130 CE          55820
[julius@station julius]$ ls /home/nero/pub
egypt.dat  gaul.dat  iberia.dat
```

In contrast, nero's home directory, `/home/nero`, now has permissions of `rwx-----x`, allowing julius only e(x)ecute permissions. Therefore, if julius already knows a file is in the directory (because, for example, nero told him it was there), julius can get to it. julius cannot, however, browse the contents of the directory with the **ls** command.

```
[julius@station julius]$ ls /home/nero
ls: /home/nero: Permission denied
```

Example 4. Protecting Home's Subdirectories

Often, a user elects to allow other users to have access to their home directories (consider, for example, the previous online exercise). By allowing others e(x)ecute, but not (r)ead, permissions to their home directory, other users must know that a directory exists within the home directory in order to access it. Because other users may not use the **ls** command to discover the contents of their home directory, users home directories remain private, and only the parts they choose to expose are available to other users.

While not adding (r)ead permissions to a home directory provides some protection against the browsing of other users, it is not foolproof. Other users can still "guess" at the contents of a directory to which they have e(x)ecute but not (r)ead permissions. For example, users commonly create a directory called `~/mail`, for storing mail messages. Suppose nero has given others e(x)ecute access to his home directory (as in the previous exercise), and later creates such a `~/mail` directory. If elvis were to guess that such a directory existed, the default permissions on `~nero/mail` would allow him to browse its contents. This is played out in the following transcript:

```
[nero@station nero]$ ls -ld ~
drwx-----x    3 nero     nero         4096 Jan 20 16:41 /home/nero
[nero@station nero]$ mkdir mail
[nero@station nero]$ cal 2002 > mail/sent
[nero@station nero]$ ls -al mail
total 12
drwxrwxr-x    2 nero     nero         4096 Jan 20 16:41 .
drwx-----x    4 nero     nero         4096 Jan 20 16:41 ..
-rw-rw-r--    1 nero     nero         2027 Jan 20 16:41 sent

[elvis@station elvis]$ cat ~nero/mail/sent
                              2002

       January               February                March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
       1  2  3  4  5                     1  2                     1  2
 6  7  8  9 10 11 12    3  4  5  6  7  8  9    3  4  5  6  7  8  9
13 14 15 16 17 18 19   10 11 12 13 14 15 16   10 11 12 13 14 15 16
20 21 22 23 24 25 26   17 18 19 20 21 22 23   17 18 19 20 21 22 23
27 28 29 30 31         24 25 26 27 28         24 25 26 27 28 29 30
                                              31

...
```

Online Exercises

Online Exercise 1. Creating Public Directories for Distinct Groups

Objective: Create group accessible directories within a user's home directory.

Estimated Time: 30 mins.

Your account should be a member of two secondary groups, music and wrestle, in addition to your private group:

```
[student@station student]$ id
uid=2293(student) gid=2296(student) groups=2296(student),301(music),302(wrestle)
```

From within your home directory, you would like to share information with other musicians and other wrestlers, but you would not like musicians to see wrestlers' information, and *vice versa*. You would prefer if one group didn't realize that the other group's shared directory even existed. Consider adopting the following plan:

1. Allow others access to your home directory.
2. Create a publicly accessible, but not listable, subdirectory to your home directory, `~/pub`.
3. Create two subdirectories of `~/pub`, both accessible and listable to the members of music and wrestlers, respectively: `~/pub/music` and `~/pub/wrestlers`

After you have created the subdirectories with the appropriate permissions, create some files within the directories to share among group members with the appropriate group ownerships and permissions. For the music group, the file should be called `~/pub/music/lyrics`, while for the wrestle group, the file should be called `~/pub/wrestle/plan`. When you are finished, you should be able to confirm the appropriate behavior using your student_b and student_c accounts (also a members of the groups wrestle and music, respectively).

```
[student_a@station student_a]$ ls /home/student/pub/wrestle
plan
[student_a@station student_a]$ cat /home/student/pub/wrestle/plan
pin the other guy
[student_a@station student_a]$ ls /home/student/pub
ls: /home/student/pub: Permission denied

[student_c@station student_c]$ ls ~student/pub/music
lyrics
[student_c@station student_c]$ cat ~student/pub/music/lyrics
row, row, row your goat
[student_c@station student_c]$ ls ~student/pub
ls: /home/student/pub: Permission denied
```

Deliverables:

1. A Home directory accessible to all, browsable only by you.
2. A `~/pub` directory accessible to all, browsable only by you.
3. A `~/pub/music` directory, accessible and readable only by members of the group music.
4. A file `~/pub/music/lyrics`, readable and writable by members of the group music.
5. A `~/pub/wrestle` directory, accessible and readable only by members of the group wrestle.
6. A file `~/pub/wrestle/plan`, readable and writable by members of the group wrestle.

Solution: The following series of commands demonstrates one possible solution to the specifications above.

```
[student@station student]$ ls -ld ~
drwx------    3 student  student      4096 Jan 20 15:17 /home/student
[student@station student]$ chmod o+x ~
[student@station student]$ mkdir ~/pub
[student@station student]$ chmod o-r ~/pub
[student@station student]$ ls -al ~/pub
total 8
drwxrwx--x    2 student  student      4096 Jan 20 15:17 .
drwx-----x    4 student  student      4096 Jan 20 15:17 ..
```

```
[student@station student]$ mkdir ~/pub/music ~/pub/wrestle
[student@station student]$ chmod o-rx ~/pub/music/ ~/pub/wrestle/
[student@station student]$ chgrp music ~/pub/music
[student@station student]$ chgrp wrestle ~/pub/wrestle
[student@station student]$ ls -al ~/pub
total 16
drwxrwx--x    4 student  student      4096 Jan 20 15:18 .
drwx-----x    4 student  student      4096 Jan 20 15:17 ..
drwxrwx---    2 student  music        4096 Jan 20 15:18 music
drwxrwx---    2 student  wrestle      4096 Jan 20 15:18 wrestle

[student@station student]$ echo "row, row, row your goat" > ~/pub/music/lyrics
[student@station student]$ chgrp music ~/pub/music/lyrics
[student@station student]$ chmod o-r ~/pub/music/lyrics
[student@station student]$ echo "pin the other guy" > ~/pub/wrestle/plan
[student@station student]$ chgrp wrestle ~/pub/wrestle/plan
[student@station student]$ chmod o-r ~/pub/wrestle/plan
[student@station student]$ ls -l ~/pub/*
/home/student/pub/music:
total 4
-rw-rw----    1 student  music          24 Jan 20 15:20 lyrics

/home/student/pub/wrestle:
total 4
-rw-rw----    1 student  wrestle        18 Jan 20 15:22 plan
```

Online Exercise 2. Protecting Subdirectories within Home

Objective: Protect a newly created subdirectory within your home directory from unintended browsing.

Estimated Time: 10 mins.

This lab assumes that others already have e(x)ecute permissions on your home directory. Create a `memos` subdirectory in your home directory, and modify its permissions so that other users on the system have no access to the directory. Create a file within the directory, and use one of your alternate accounts to confirm that other users cannot access the file.

Solution: The following series of commands proved one possible solution to the above specifications. (The output assumes you have just completed the previous exercise.)

```
[student@station student]$ ls -ld ~
drwx-----x    4 student  student      4096 Jan 20 16:50 /home/student
[student@station student]$ mkdir memos
[student@station student]$ chmod o-rx memos/
[student@station student]$ ls -l
total 8
drwxrwx---    2 student  student      4096 Jan 20 16:50 memos
drwxrwx--x    4 student  student      4096 Jan 20 15:18 pub
[student@station student]$ echo "feed cat" > memos/todo

[student_a@station student_a]$ ls ~student/memos
ls: /home/student/memos: Permission denied
[student_a@station student_a]$ cat ~student/memos/todo
cat: /home/student/memos/todo: Permission denied
```

Deliverables:

1. A directory called `~/memos`, which is browsable by only the directory owner.

Questions

Question Set 1. Analyzing Directory Permissions

Use the following table of users (with group memberships) and files (with user owner, group owner, and permissions) to answer the following questions. Note that the permissions includes an initial letter that indicates if the file is a regular file ("-") or a directory ("d").

user	group memberships
elvis	elvis, wrestle, physics, emperors, music
ventura	ventura,wrestle,governor
pataki	pataki,governor
blondie	blondie, music
prince	prince, music
einstein	einstein, physics
maxwell	maxwell, physics

filename	user	group	permissions
`governor/`	ventura	governor	`drwxrwx---`
`governor/budget2001`	ventura	governor	`-rw-r--r--`
`governor/budget2002`	ventura	governor	`-rw-r-----`
`music/`	root	music	`drwxr-xr-x`
`music/contrib/`	root	music	`drwxrwxrwx`
`music/contrib/stand_by_your_spam.txt`	blondie	blondie	`-rw-rw-r--`
`music/drafts/`	prince	music	`drwxrwx---`
`music/drafts/1998.txt`	prince	prince	`-rw-rw-r--`
`music/drafts/little_orange_corvette.txt`	prince	music	`-rw-rw-r--`
`physics/`	einstein	physics	`drwxrwx--x`

filename	user	group	permissions
`physics/published/`	maxwell	physics	`drwxrwxr-x`
`physics/published/equations`	maxwell	maxwell	`-rw-r--r--`
`physics/published/relativity`	einstein	einstein	`-rw-rw-r--`
`physics/working/`	einstein	physics	`drwxr-x---`
`physics/working/time_travel`	einstein	physics	`-rwxr--r--`

1.

Which user can read the file `governor/budget2001`?

A. ( ) elvis
B. ( ) ventura
C. ( ) prince
D. ( ) maxwell

2.

Which user can modify the file `physics/published/relativity`?

A. ( ) einstein
B. ( ) maxwell
C. ( ) elvis
D. ( ) blondie

3.

Which users can modify the file `music/drafts/little_orange_corvette`?

A. ( ) elvis and einstein
B. ( ) elvis, prince, and blondie
C. ( ) prince and einstein
D. ( ) only prince

4.

Which users can create a new file in the directory `physics/published/`?

A. ( ) only maxwell
B. ( ) maxwell and ventura
C. ( ) all users can
D. ( ) elvis, maxwell, and einstein

5.

Which users can list files in the directory `physics/`?

A. ( ) elvis, einstein, and maxwell
B. ( ) only einstein
C. ( ) all users can
D. ( ) elvis, prince, and blondie

6.

Which users can list files in the directory `music/`?

A. ( ) elvis, einstein, and maxwell
B. ( ) only root
C. ( ) all users can
D. ( ) elvis, prince, and blondie

7.

Who can read the file `governor/budget2001`?

A. ( ) All users
B. ( ) Only ventura
C. ( ) ventura and pataki
D. ( ) ventura and maxwell

8.

Who can read the file `governor/budget2002`?

A. ( ) All users
B. ( ) Only ventura
C. ( ) ventura and pataki
D. ( ) ventura and maxwell

9.

Which users can read the file `physics/working_timetravel`?

A. ( ) elvis, maxwell, and einstein
B. ( ) only einstein
C. ( ) all users
D. ( ) einstein and maxwell

10.

Which users can remove the file `music/contrib/stand_by_your_spam.txt`?

A. ( ) only blondie
B. ( ) all users
C. ( ) elvis, maxwell, and einstein
D. ( ) prince, blondie, and elvis