Security: Managing File Permisions and Ownership

Learning Objectives

After completing this reading, you will be able to:

- Explain file ownership and permissions
- View file and directory permissions
- Make a file private

Why do we need file permissions and ownership?

Linux is a multi-user operating system. This means that by default, other users will be able to view any files you store on the system. However, you may have some files - such as your personal tax documents or your employer's intellectual property documents - that are private or confidential. How can you protect these sensitive documents from being viewed or modified by others?

File ownership and permissions

There are three possible levels of file ownership in Linux: user, group, and other.
Whoever creates a file, namely the **user** at the time of creation, becomes the owner of that file by default. A **group** of users can also share ownership of a file. The **other** category essentially refers anyone in the universe with access to your Linux machine - careful when assigning ownership permission to this level!

Only an official owner of a file is allowed to change its permissions. This means that only owners can decide who can read the file, write to it, or execute it.

Viewing file permissions

Let's say you've entered the following lines of code:

```
1. 1
2. 2
3. 3
4. 4
5. 5
```

```
1. $ echo "Who can read this file?" > my_new_file
2. $ more my_new_file
3. Who can read this file?
4. $ ls -l my_new_file
5. -rw-r--r-- 1 theia users 25 Dec 22 17:47 x
```

Copied!

Here we've echoed the string `"Who can read this file?"` into a new file called `my_new_file`. The next line uses the `more` command to print the contents of the new file. Finally, the `ls` command with the `-l` option displays the file's (default) permissions: `rw-r--r--`

The first three characters (`rw-`) define the **user** permissions, the next three (`r--`) the **group** pemissions, and the final three (`r--`) the **other** permissions.

So you, being the user, have the permission `rw-`, which means you have read and write permissions by default, but do not have execution permissions. Otherwise there would be an `x` in place of the last `-`.

Thus by looking at the entire line, `rw-r--r--`, you can see that anyone can read the file, nobody can execute it, and you are the only user that can write to it.

> **Note:** The `-` at the very beginning of the line in the terminal means that the permissions are referring to a file. If you were getting the permissions to a directory, you would see a `d` in the front for "directory".

Directory permissions

The permissions for directories are similar but distinct for files. Though directories use the same `rwx` format, the symbols have slightly different meanings.

The following table illustrates the meanings of each permission for directories:

Directory Permission	Permissible action(s)
`r`	List directory contents using `ls` command
`w`	Add or remove files or directories
`x`	Enter directory using `cd` command

Setting appropriate permissions on directories is a best practice for both security and stability reasons. Though this reading focuses on security, you will learn more about other reasons for setting file permissions and ownership later in this course.

Making a file private

You can revoke read permissions from your group and all other users by using the `chmod` command. Ensure successful modification by using the `ls -l` command again:

```
1. 1
2. 2
3. 3

1. chmod go-r my_new_file
2. ls -l my_new_file
3. -rw------- 1 theia users 24 Dec 22 18:49 my_new_file
```

Copied!

In the `chmod` command, `go-r` is the permission change to be applied, which in this case means removing for the group (`g`) and others (`o`) the read (`r`) permission. The `chmod` command can be used with both files and directories.

Executable files - looking ahead

You've learned what it means to read or write to a file, but what does it mean to have permissions to **execute** a file in Linux?

A Linux file is executable if it contains instructions that can be directly interpreted by the operating system. Basically, an exectuable file is a ready-to-run program. They're also referred to as **binaries** or **executables**.

In this course, you will become very familiar with a particular kind of executable called a **script**, which is a program written in a scripting language. You'll learn all about **shell scripting**, or more specifically **Bash**

scripting, which is writing scripts in Bash (*born-again shell*), a very popular shell scripting language. A shell script is a plain text file that can be interpreted by a shell.

Formally speaking, for a text file to be considered an executable shell script for a given user, it needs to have two things:

1. Execute permissions set for that user
2. A directive, called a "shebang", in its first line to declare itself to the operating system as a binary

All of this will become more clear to you soon when we get to the topic of shell scripting.

Summary

In this reading, you learned that:

- There are three possible levels of file ownership in Linux - user, group, and other - which determine who can read, write to, and execute a file
- You can use the `ls -l` command to view file and directory permissions
- You can change permissions on a file by using the `chmod` command

Authors

Jeff Grossman

Other Contributors

Rav Ahuja

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-05-04	1.4	Benny Li	QA Pass
2023-04-28	1.3	Nick Yi	QA Pass
2023-04-10	1.2	Nick Yi	ID Review
2023-01-11	1.1	Jeff Grossman	Added directory permissions content
2022-12-22	1.0	Jeff Grossman	Created initial version of the reading