Google IT Support Certificate: Course 3: Operating Systems and You

Week 2: Users and Permissions

Quiz: Creating, Modifying, and Removing File and Folder Permissions in Linux

Introduction

In this lab, you'll learn the foundations of how managing user permissions work on a Linux machine. Using the new commands you learned in Bash, you'll fix up the permissions of some files and folders.

What you'll do

- Familiarize yourself with the process of changing permissions within a file and folder in Linux
- Change the ownership of a specific file and folder

Checking permissions

Before you can even begin changing the permission of a file or folder, you need to first check the permissions of the specific file/folder. To display ownership and permissions for a file, you can use **ls** with the **-l** flag and the name of the file you're interested in with the command `ls -l [FILENAME]`

There's a file named "**important_document**" on your machine in the "**/home/qwiklab/documents**" directory. You can change to this directory from your current one using this command:

```
cd ../qwiklab/documents
```

Copied!

content_copy

Check out its current permissions with this command, and take a look at the output below:

```
ls -l important_document
```

Copied!

content_copy

```
-rw------- 1 student student 16 Aug  4 11:38 important_document
```

You can see that it's owned by the "root" user, and that the owner has read and write (but not execute) permissions while everyone else has none at all.

Changing file permissions

Now, change the permissions of "important_document" (from the previous step) so that the owner has execute permissions on top of their current permissions. To do this, you'll use the **chmod** command, with the argument **700**. The two zeros keep everyone, but the owner, from having any permissions at all, and the seven grants all available permissions to the owner (including execute). Keep in mind that because the file is owned by "root" you'll need to use **sudo**:

```
sudo chmod 700 important_document
```

Copied!

content_copy

You can check the permissions from the below command. You'll now see that the execution permission has been added:

```
ls -l important_document
```

Copied!

content_copy

```
-rwx------ 1 student student 16 Aug  4 11:38 important_document
```

Click *Check my progress* to verify the objective.

Modify permissions on important_document

Check my progress

Changing folder permissions

Now you'll do a similar process, this time on a directory. First, move up one directory:

```
cd ..
```

Copied!

content_copy

In this directory there's a folder called "secret_folder". View its current permissions using **ls**, this time with the **-ld** flag rather than **-l** because you're viewing a directory instead of a normal file:

```
ls -ld secret_folder/
```

Copied!

content_copy

```
drw-r--r-- 2 root root 4096 Aug  4 11:38 secret_folder/
```

As you can see, the owner of the file (the root user) has read and write permissions, and everyone else can read only.

The goals for the lab, related to this file, are below:

1. The owner should have all permissions.
2. The group should have only write permission.

3. People other than the owner and the group should have no permissions.

Head's up: When using **chmod** on a directory, files within that directory are not affected. While this isn't relevant to this specific lab, it's important to remember.

Previously, we used a numerical argument to set the permissions for a file. If you want to avoid figuring out the number that matches the permission levels, you can use an alternate syntax. To satisfy the first condition, you only need to add the execute permission to the owner, since they already have read and write permissions. To add execute to the owner's permission, you can use the command below. (Note that "u" stands for "user" and "x" stands for "execute".)

```
sudo chmod u+x secret_folder/
```

Copied!

content_copy

You can check the permissions again to see that the owner can now read, write, and execute:

```
ls -ld secret_folder/
```

Copied!

content_copy

```
drwxr--r-- 2 root root 4096 Aug  4 11:38 secret_folder/
```

Now you can fix the group's permission. They currently have read permission and don't have write permission, which you can fix with two similar commands. These can be done in either order; "g" stands for "group" (like "u" from before), and "w" and "r" stand for "write" and "read" respectively:

```
sudo chmod g+w secret_folder/
```

Copied!

content_copy

```
sudo chmod g-r secret_folder/
```

Copied!

content_copy

You can check the permissions again to see that the group now has only write permissions, and the owner has the same permissions as before:

```
ls -ld secret_folder/
```

Copied!

content_copy

```
drwx-w-r-- 2 root root 4096 Aug  4 11:38 secret_folder/
```

Finally, you can remove read permissions from everyone else using the command below ("o" stands for "other"):

```
sudo chmod o-r secret_folder/
```

Copied!

content_copy

You can see that all the criteria for this file are now met using **ls** again:

```
ls -ld secret_folder/
```

Copied!

content_copy

```
drwx-w---- 2 root root 4096 Aug  4 11:38 secret_folder/
```

Using **chmod** this way is easier to remember, but takes lots more commands. All the previous steps could also have been done using the numerical notation, like this:

```
sudo chmod 720 secret_folder/
```

Copied!

content_copy

Click *Check my progress* to verify the objective.

Modify permissions on secret_folder

Check my progress

Changing owners

Now you'll change the owner of a file. In your current directory, there's a folder called "taco". Use **ls** to examine its permissions and see who the owner of the file is:

```
ls -ld taco/
```

Copied!

content_copy

```
drwxr-xr-x 2 root root 4096 Aug  4 11:38 taco/
```

You can see from this that the root user currently owns this file. There's another user account on the machine called "cook". Go ahead and make "cook" the owner of the file, using the **chown** command like this:

```
sudo chown cook /home/qwiklab/taco
```

Copied!

content_copy

Now you can use **ls** again to see that the owner of the file has been successfully changed:

```
ls -ld taco/
```

Copied!

content_copy

```
drwxr-xr-x 2 cook root 4096 Aug  4 11:38 taco/
```

Click *Check my progress* to verify the objective.

Change owner of Taco

Check my progress

More practices

There are a few more files present on your machine that you can practice on. First, move into the "documents" folder:

```
cd documents/
```

Copied!

content_copy

There's a file in this folder called "not_so_important_document". View its permissions to see its current state:

```
ls -l not_so_important_document
```

Copied!

content_copy

```
-rw-r----- 1 student student 20 Aug  4 11:38 not_so_important_document
```

The owner can read and write, the group can read, and everybody else has no permissions at all. Now, use **chmod** to change the permissions so that these criteria are met:

1. The owner has all permissions.
2. The group has read and write permissions.
3. Everyone has read permissions.

To give the owner execution permissions, you can use the same command from earlier:

```
sudo chmod u+x not_so_important_document
```

Copied!

content_copy

Remember to use **ls** to double-check that everything you do behaves how you expect:

```
ls -l not_so_important_document
```

Copied!

content_copy

```
-rwxr----- 1 student student 20 Aug  4 11:38 not_so_important_document
```

The group already has read permissions, so all you need to do is add write permissions:

```
sudo chmod g+w not_so_important_document
```

Copied!

content_copy

```
ls -l not_so_important_document
```

Copied!

content_copy

```
-rwxrw---- 1 student student 20 Aug  4 11:38 not_so_important_document
```

Finally, you need to give everyone else read permissions. You can use the "o+r" argument to add read permissions to people other than the owner or group, but you can also use "a+r". This adds read permission to everyone (owner, group, and other). Because the owner and the group already have read permissions, this will only change the permissions for everyone else, but the end result is the same:

```
sudo chmod a+r not_so_important_document
```

Copied!

content_copy

You should be able to view the permissions again and see that all criteria for this file have been met:

```
ls -l not_so_important_document
```

Copied!

content_copy

```
-rwxrw-r-- 1 student student 20 Aug  4 11:38 not_so_important_document
```

Again, you can accomplish the same result using a numerical argument to set the permissions, rather than incrementally changing them. Here's the command that meets all three criteria at once:

```
sudo chmod 764 not_so_important_document
```

Copied!

content_copy

Click *Check my progress* to verify the objective.

Change permissions of not_so_important_document

Check my progress

Adding multiple permissions at once

Finally, you'll learn how to use the non-numeric argument to add multiple permissions at once. There's one more file in the current

directory, named "public_document". First, view its current permissions:

```
ls -l public_document
```

Copied!

content_copy

```
-rw-r--r-- 1 student student 14 Aug  4 11:38 public_document
```

For this file, you want everyone (owner, group, and anyone else) to have all permissions. You can add read, write, and execute permissions to everyone at once using this command:

```
sudo chmod a+rwx public_document
```

Copied!

content_copy

This should make the file as open as possible, where every user has every permission:

```
ls -l public_document
```

Copied!

content_copy

```
-rwxrwxrwx 1 student student 14 Aug  4 11:38 public_document
```

Using the numeric argument form of **chmod**, this same result could be accomplished with this command:

```
sudo chmod 777 public_document
```

Copied!

content_copy

Click *Check my progress* to verify the objective.

Change owner of public_document

Check my progress

Conclusion

Congrats! You've successfully used **chmod** on both directories and normal files, in multiple formats. You can directly set a file's permissions numerically, or add and remove specific permissions one at a time. You've also successfully used **chown** to change the owner of a file. Really great work!