File permissions in Linux

Project description

The research team at my organization needs to update the file permissions for certain files and directories within the `projects` directory. The permissions do not currently reflect the level of authorization that should be given. Checking and updating these permissions will help keep their system secure. To complete this task, I performed the following tasks:

Check file and directory details

The following code demonstrates how I utilized appropriate Linux commands to determine the existing permissions set for a specific directory in the file system.

```
researcher2@8f6620810cca:~$ pwd
/home/researcher2
researcher2@8f6620810cca:~$ ls
projects
researcher2@8f6620810cca:~$ cd projects
researcher2@8f6620810cca:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Aug 28 18:56 .
drwxr-xr-x 3 researcher2 research_team 4096 Aug 28 19:35 ..
-rw--w---- 1 researcher2 research_team   46 Aug 28 18:56 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Aug 28 18:56 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Aug 28 18:56 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 28 18:56 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 18:56 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 18:56 project_t.txt
researcher2@8f6620810cca:~/projects$
```

The first four lines of the screenshot displays the commands I entered to make sure what directory I was in so I can then `cd` my way into the appropriate `projects` directory. I then used the `ls` command with the `-la` option to display a detailed listing of the file contents that also returned hidden files. The output of this command indicates that there is one hidden directory named `drafts`, one hidden file named `.projects_x.txt`, and five other project files. The 10-character string in the first column represents the permissions set on each file or directory.

Describe the permissions string

The 10-character permissions string can be deconstructed to determine who is authorized to access the file and their specific permissions. The characters and what they represent are as follows:

- **1st character:** This character is either a `d` or hyphen `(-)`. Depending on which character is present, it will indicate whether the file is a directory `(d)` or a regular file `(-)`.
- **2nd-4th characters:** These characters will indicate whether the `user` will have the read `(r)` write `(w)` and/or execute `(x)` permissions. When the `user` is not granted a specific permission, a hyphen `(-)` will instead be in place of the character.
- **5th-7th:** These characters will indicate whether the `group` will have the read `(r)` write `(w)` and/or execute `(x)` permissions. When the `group` is not granted a specific permission, a hyphen `(-)` will instead be in place of the character.
- **8th-10th characters:** These characters will indicate whether `other` will have the read `(r)` write `(w)` and/or execute `(x)` permissions. When `other` is not granted a specific permission, a hyphen `(-)` will instead be in place of the character.

Change file permissions

The organization determined that `other` should not have write permissions to any of their files. To comply with this, I referred to the file permissions I previously returned. I determined that `project_k.txt` must have the write access removed for `other`.

The following code demonstrates how I used Linux commands to do this:

```
researcher2@8f6620810cca:~/projects$ chmod o-w project_k.txt
researcher2@8f6620810cca:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Aug 28 18:56 .
drwxr-xr-x 3 researcher2 research_team 4096 Aug 28 19:35 ..
-rw--w---- 1 researcher2 research_team   46 Aug 28 18:56 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Aug 28 18:56 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 18:56 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 28 18:56 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 18:56 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 18:56 project_t.txt
researcher2@8f6620810cca:~/projects$
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. The `chmod` command changes the permissions on files and directories. The first argument indicates what permissions should be changed, and the second argument specifies the file or directory. In this example, I removed the write

permissions from `other` for the `project_k.txt` file. After this, I used `ls -la` to review the updates I made.

Change file permissions on a hidden file

The research team recently archived `.project_x.txt`. They do not want anyone to have write access to this project, but the user and group should have read access.

The following code demonstrates how I used Linux commands to change the permissions:

```
researcher2@8f6620810cca:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@8f6620810cca:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Aug 28 18:56 .
drwxr-xr-x 3 researcher2 research_team 4096 Aug 28 19:35 ..
-r--r----- 1 researcher2 research_team   46 Aug 28 18:56 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Aug 28 18:56 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 18:56 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 28 18:56 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 18:56 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 18:56 project_t.txt
researcher2@8f6620810cca:~/projects$
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. I know `.project_x.txt` is a hidden file because it starts with a period (`.`). In this example, I removed write permissions from the user and group, and added read permissions to the group. I removed write permissions from the user with `u-w`. Then, I removed write permissions from the group with `g-w`, and added read permissions to the group with `g+r`.

Change directory permissions

My organization only wants the `researcher2` user to have access to the `drafts` directory and its contents. This means that no one other than `researcher2` should have execute permissions.

The following code demonstrates how I used Linux commands to change the permissions:

```
researcher2@803f52a26038:~/projects$ chmod g-x drafts
researcher2@803f52a26038:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Aug 28 20:42 .
drwxr-xr-x 3 researcher2 research_team 4096 Aug 28 21:19 ..
-r--r----- 1 researcher2 research_team   46 Aug 28 20:42 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Aug 28 20:42 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 20:42 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Aug 28 20:42 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 20:42 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Aug 28 20:42 project_t.txt
researcher2@803f52a26038:~/projects$
```

The first two lines of the screenshot display the commands I entered, and the other lines display the output of the second command. I previously determined that the group had execute permissions, so I used the `chmod` command to remove them. The `researcher2` user already had execute permissions, so they did not need to be added.

Summary

I changed multiple permissions to match the level of authorization my organization wanted for files and directories in the `projects` directory. The first step in this was using `ls -la` to check the permissions for the directory. This informed my decisions in the following steps. I then used the `chmod` command multiple times to change the permissions on files and directories.