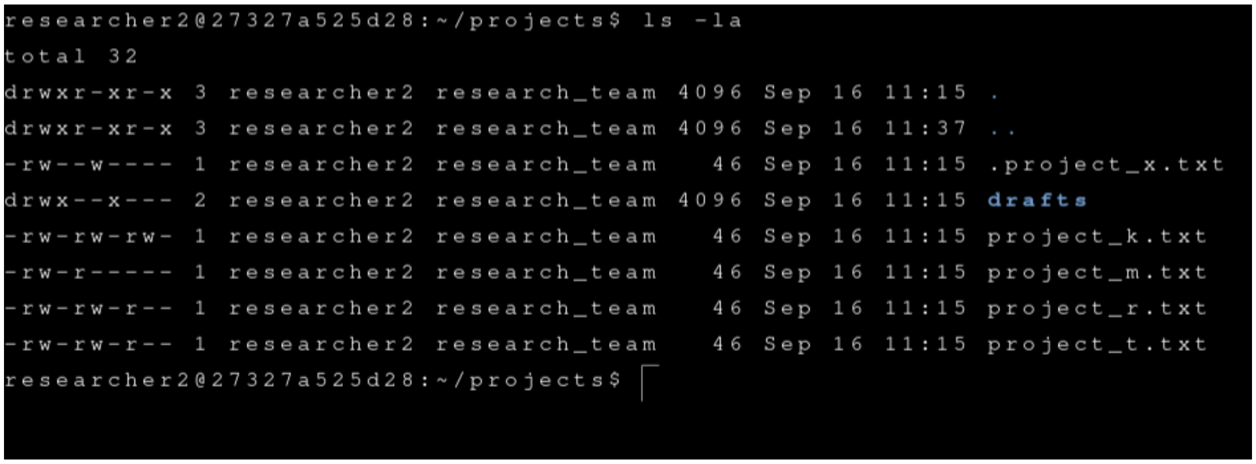
File permissions in Linux

Project description

The research team within our organization requires a revision of the file permissions assigned to specific files and directories within the `projects` directory. The current permissions do not align with the intended level of authorization. This action is vital for enhancing system security. To accomplish this, I undertook the following steps.

Check file and directory details

I utilized the `ls -la` command to list all files, including hidden ones, along with their permissions. This allowed me to examine and determine the current permissions set for a particular directory within the file system.



```
researcher2@27327a525d28:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 16 11:15 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 16 11:37 ..
-rw--w---- 1 researcher2 research_team   46 Sep 16 11:15 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 16 11:15 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Sep 16 11:15 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 16 11:15 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 11:15 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 11:15 project_t.txt
researcher2@27327a525d28:~/projects$
```

The first line of the screenshot displays the command I entered, and the other lines display the output. The code lists all contents of the `projects` directory. The output of my command indicates that there is one directory named `drafts`, one hidden file named `.project_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

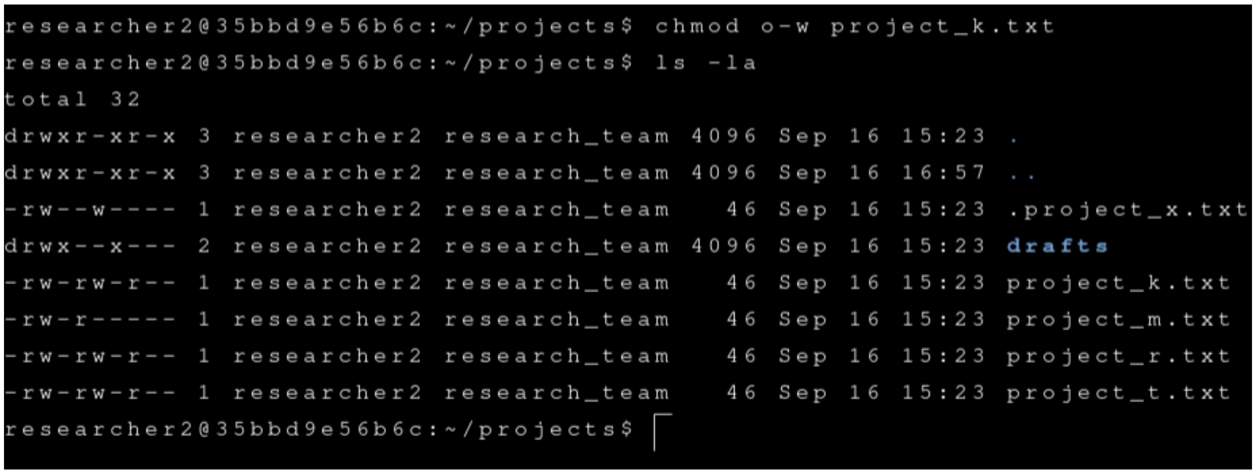
The string of ten characters can be deconstructed to determine who is authorized to access the file and their specific permissions. The characters and what they represent are as follows:

- **1st character**: This character is either a `d` or hyphen (`-`) and indicates the file type. If it's a `d`, it's a directory. If it's a hyphen (`-`), it's a regular file.
- **2nd-4th characters**: These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for the *user*. When one of these characters is a hyphen (`-`) instead, it indicates that this permission is not granted to the *user.*
- **5th-7th characters:** These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for the *group*. When one of these characters is a hyphen (`-`) instead, it indicates that this permission is not granted for the *group.*
- **8th-10th characters:** These characters indicate the read (`r`), write (`w`), and execute (`x`) permissions for *other.* This owner type consists of all other users on the system apart from the *user* and the *group*. When one of these characters is a hyphen (`-`) instead, that indicates that this permission is not granted for *other.*

Change file permissions

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

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



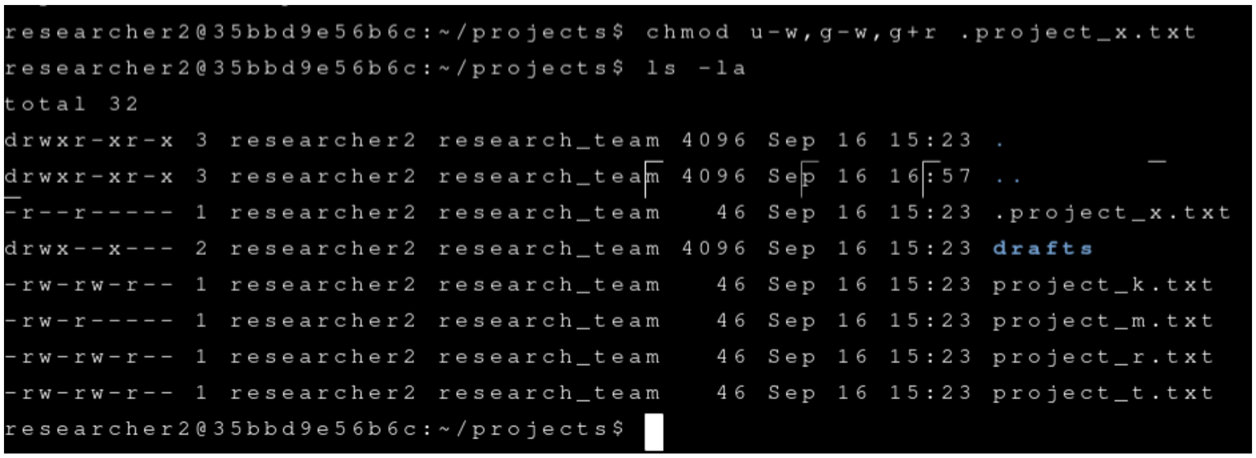
```
researcher2@35bbd9e56b6c:~/projects$ chmod o-w project_k.txt
researcher2@35bbd9e56b6c:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 16 15:23 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 16 16:57 ..
-rw--w---- 1 researcher2 research_team   46 Sep 16 15:23 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 16 15:23 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 16 15:23 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_t.txt
researcher2@35bbd9e56b6c:~/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 the `o-w` argument,' o' represents *other*, the '-' symbol indicates removing and 'w' represents write. In this example, I removed 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 at my organization 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:



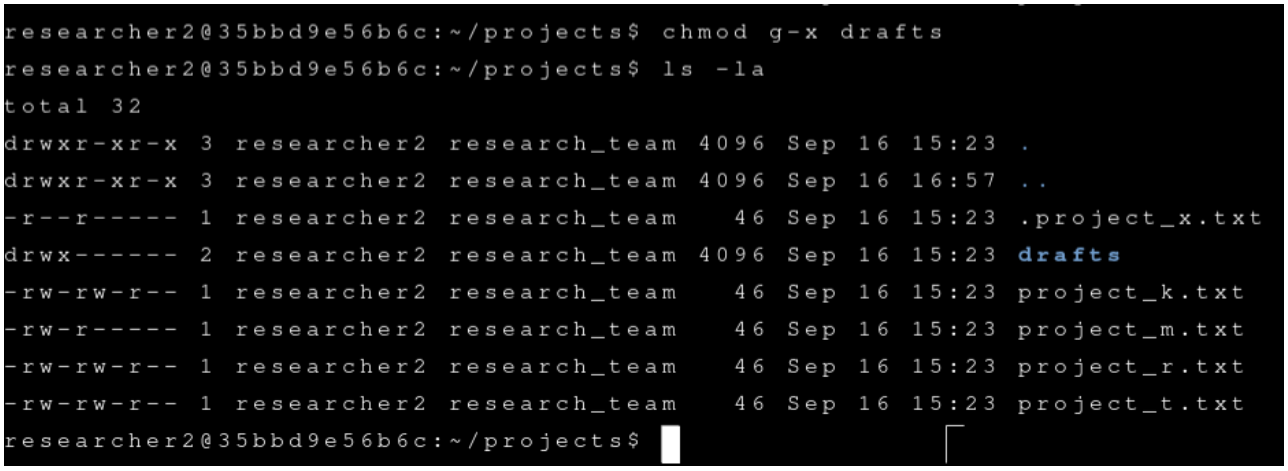
```
researcher2@35bbd9e56b6c:~/projects$ chmod u-w,g-w,g+r .project_x.txt
researcher2@35bbd9e56b6c:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 16 15:23 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 16 16:57 ..
-r--r----- 1 researcher2 research_team   46 Sep 16 15:23 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Sep 16 15:23 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 16 15:23 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_t.txt
researcher2@35bbd9e56b6c:~/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@35bbd9e56b6c:~/projects$ chmod g-x drafts
researcher2@35bbd9e56b6c:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Sep 16 15:23 .
drwxr-xr-x 3 researcher2 research_team 4096 Sep 16 16:57 ..
-r--r----- 1 researcher2 research_team   46 Sep 16 15:23 .project_x.txt
drwx------ 2 researcher2 research_team 4096 Sep 16 15:23 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Sep 16 15:23 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Sep 16 15:23 project_t.txt
researcher2@35bbd9e56b6c:~/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 made changes to various permissions to align with my organization's desired authorization levels for files and directories within the `projects` directory. Initially, I utilized the `ls -la` command to inspect the permissions of the directory, which guided my subsequent actions. Using the `chmod` command, I proceeded to modify permissions on both files and directories accordingly.