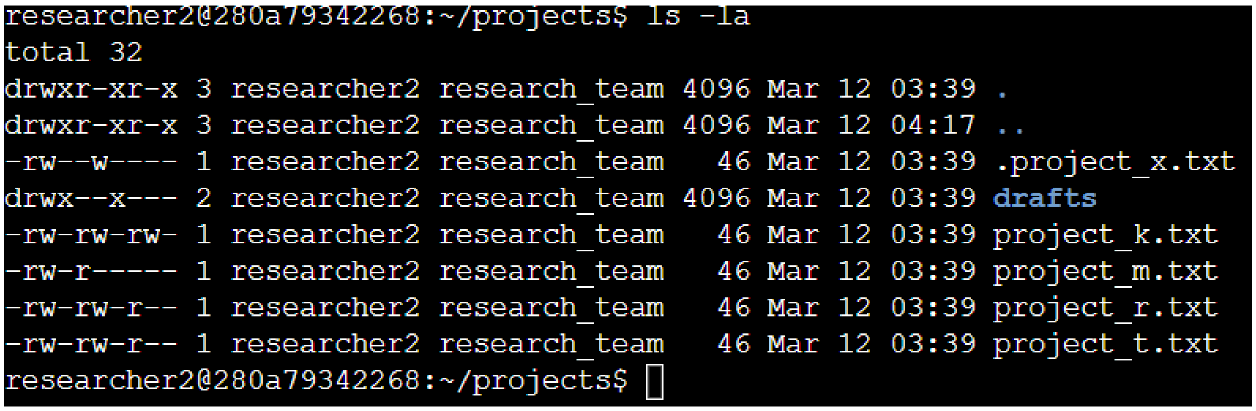
File permissions in Linux

Project description

The research team at my organization needs to review and update file permissions for specific files and directories within the projects directory. The current permissions do not align with the required authorization levels. Ensuring proper permissions will enhance system security. To address this, I carried out the following tasks:

Check file and directory details

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



```
researcher2@280a79342268:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Mar 12 03:39 .
drwxr-xr-x 3 researcher2 research_team 4096 Mar 12 04:17 ..
-rw--w---- 1 researcher2 research_team   46 Mar 12 03:39 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Mar 12 03:39 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Mar 12 03:39 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Mar 12 03:39 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_t.txt
researcher2@280a79342268:~/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. I 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 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 **10-character permissions string** represents the access rights assigned to a file or directory in a Unix-like system. It follows this format:

[Type] [Owner] [Group] [Others]

Breakdown:

1. **The first character**: Indicates the file type
   - `-` ⟶ Regular file
   - `d` ⟶ Directory
   - `l` ⟶ Symbolic link
   - `c` ⟶ Character device
   - `b` ⟶ Block device
2. **The next nine characters** (divided into three groups of three):
   - **Owner permissions** (User who owns the file)
   - **Group permissions** (Users in the same group)
   - **Other permissions** (Everyone else)

Each group follows this pattern:

- `r` ⟶ Read
- `w` ⟶ Write
- `x` ⟶ Execute
- `-` ⟶ No permission

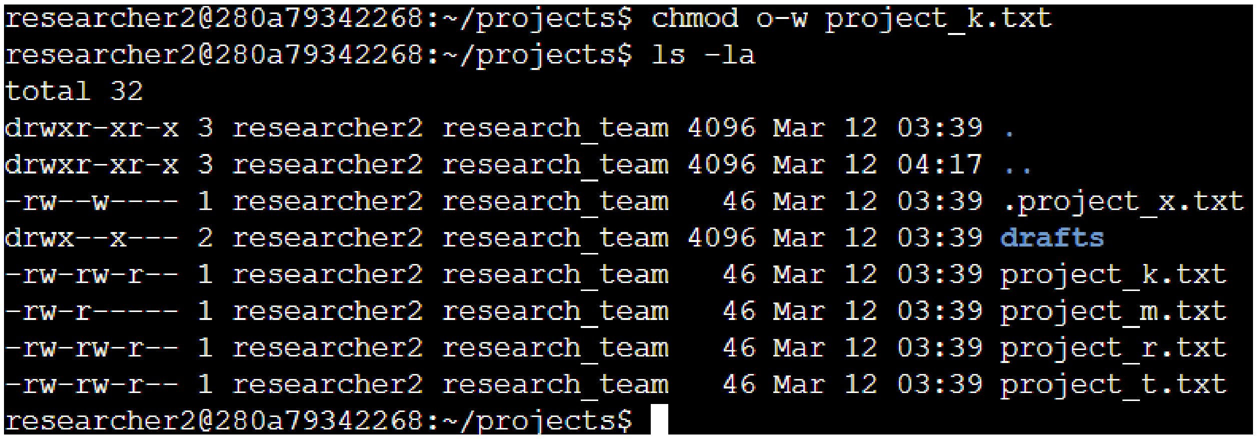
Example:

drwxr-xr--

- `d` ⟶ It is a **directory**
- `rwx` ⟶ **Owner** has **read, write, and execute** permissions
- `r-x` ⟶ **Group** has **read and execute** permissions, but **no write**
- `r--` ⟶ **Others** have **only read** permissions

Change file permissions

Before modifying the file permissions, I needed to ensure that the authorization levels for files in the `projects` directory were correctly set to maintain security and prevent unauthorized changes. I decided to review the current permissions using the `ls -la` command, which displayed a detailed list of all files, including hidden ones, along with their respective permission settings. Upon reviewing the output, I noticed that `project_k.txt` allowed write access to others, which posed a potential security risk. To address this, I proceeded to modify the file permissions to restrict unnecessary write access while maintaining appropriate readability.

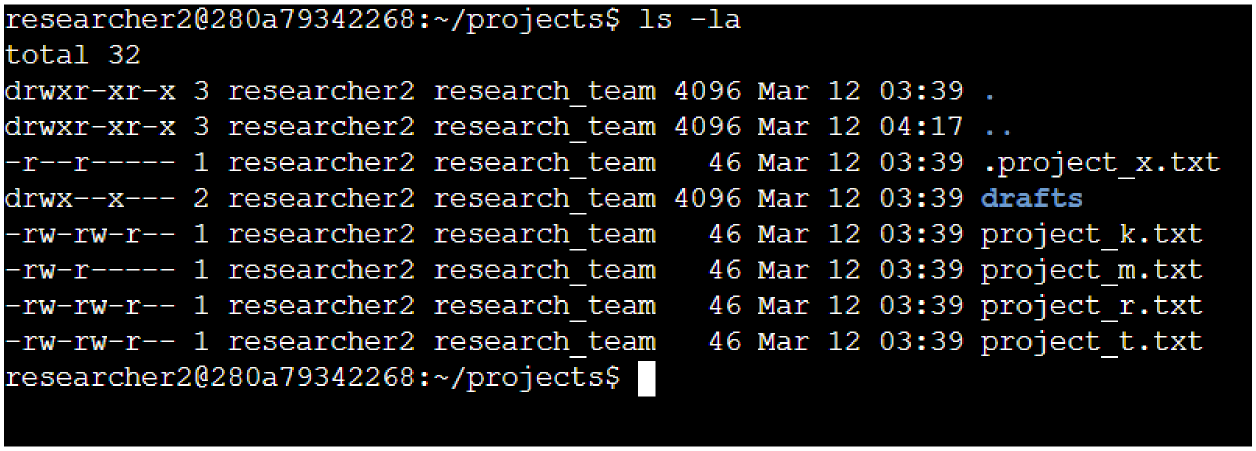


```
researcher2@280a79342268:~/projects$ chmod o-w project_k.txt
researcher2@280a79342268:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Mar 12 03:39 .
drwxr-xr-x 3 researcher2 research_team 4096 Mar 12 04:17 ..
-rw--w---- 1 researcher2 research_team   46 Mar 12 03:39 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Mar 12 03:39 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Mar 12 03:39 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_t.txt
researcher2@280a79342268:~/projects$
```

In the screenshot, I modified the file permissions for `project_k.txt` using the `chmod` command. I executed `chmod o-w project_k.txt` to remove the write (`w`) permission for others (`o`), ensuring that only I and the group members could retain their original permissions while preventing others from modifying the file. After making this change, I ran the `ls -la` command to verify the updated permissions. The output confirmed that `project_k.txt` now has the permission string `-rw-r--r--`, meaning I, as the owner, have read and write permissions, the group (`research_team`) has only read access, and others can no longer write to the file but can still read it. This change enhances security by preventing unauthorized modifications while maintaining readability for all users.

Change file permissions on a hidden file

Before modifying the file permissions, I needed to assess the current settings for all files, including hidden ones, to ensure proper access control. To do this, I used the `ls -la` command, which provided a detailed list of files in the `projects` directory along with their permission settings. This allowed me to identify any files that required adjustments to enhance security or enable necessary modifications. During this review, I noticed that the hidden file `.project_x.txt` had restrictive permissions, which prompted me to take action to update them appropriately.



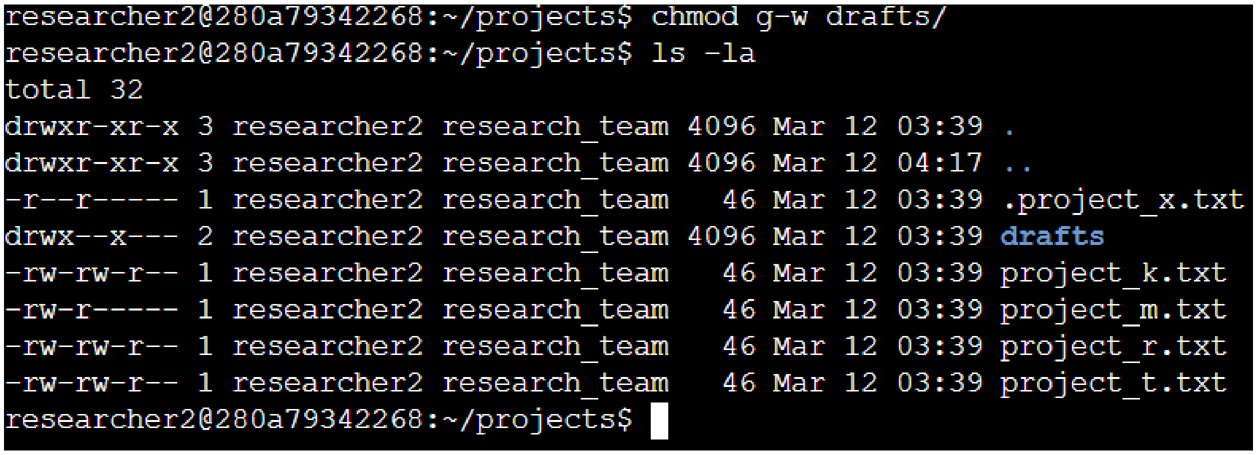
```
researcher2@280a79342268:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Mar 12 03:39 .
drwxr-xr-x 3 researcher2 research_team 4096 Mar 12 04:17 ..
-r--r----- 1 researcher2 research_team   46 Mar 12 03:39 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Mar 12 03:39 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Mar 12 03:39 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_t.txt
researcher2@280a79342268:~/projects$ 
```

Upon reviewing the file permissions in the `projects` directory using the `ls -la` command, I noticed that the hidden file `.project_x.txt` had restrictive permissions set to `-r--r-----`. This means that I, as the owner, only had read access, while the group had read access as well, and others had no permissions at all. To modify the permissions of this hidden file, I needed to use the `chmod` command. For example, if I wanted to grant myself write access while keeping the current restrictions for the group and others, I could execute `chmod u+w .project_x.txt`. This would update the permissions to `-rw-r-----`, allowing me to edit the file while maintaining security. Making such adjustments ensures that the right level of authorization is maintained for sensitive or hidden files.

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@280a79342268:~/projects$ chmod g-w drafts/
researcher2@280a79342268:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Mar 12 03:39 .
drwxr-xr-x 3 researcher2 research_team 4096 Mar 12 04:17 ..
-r--r----- 1 researcher2 research_team   46 Mar 12 03:39 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Mar 12 03:39 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Mar 12 03:39 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Mar 12 03:39 project_t.txt
researcher2@280a79342268:~/projects$
```

To ensure proper access control for the `drafts` directory, I reviewed its current permissions using the `ls -la` command. The initial permissions showed that the group (`research_team`) had write access to the directory, which meant any group member could modify its contents. To enhance security and prevent unintended modifications, I executed the command `chmod g-w drafts/`, removing the write (`w`) permission for the group. After running this command, I verified the changes using `ls -la`, which confirmed that the updated permissions for `drafts` were now `drwxr-x--x`, meaning that the group could still read and execute files within the directory but could no longer make changes. This adjustment helps maintain control over directory modifications while allowing necessary access.

Summary

I changed multiple permissions to align with the level of authorization my organization required for files and directories in the `projects` directory. The first step was using the `ls -la` command to review the existing permissions, which helped me identify files and directories that needed modifications. Based on this information, I used the `chmod` command multiple times to adjust permissions accordingly. I restricted write access for others on specific files, updated hidden file permissions to allow necessary modifications, and removed group write access from a directory to prevent unintended changes. These adjustments ensured a secure and well-regulated file system while maintaining appropriate access for authorized users.