File permissions in Linux

Project description

The research team in my company has requested an update of the permissions to access specific files within the `projects` directory. After examination, it appears that the existing permissions do not match the authorisation that should be given. Updating these permissions will allow the research team to carry out their job and keep the system secure. To achieve this, I applied the following:

Check file and directory details

To determine the existing permissions in the `/home/researcher2/projects` directory, the following command was used in Linux:

```
researcher2@2fa5bc0fe905:~/projects$ ls -la
```

The standard output shows the current permissions for each file and directory in the `/home/researcher2/projects` directory:

```
drwxr-xr-x 3 researcher2 research_team 4096 Oct 19 11:56 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 19 12:12 ..
-rw--w---- 1 researcher2 research_team   46 Oct 19 11:56 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 19 11:56 drafts
-rw-rw-rw- 1 researcher2 research_team   46 Oct 19 11:56 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 19 11:56 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 19 11:56 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 19 11:56 project_t.txt
```

In order to check the existing permissions for each file I used the `ls` command together with the `-la` option to display a detailed listing of the file contents, including hidden files. The output of my command indicates that there are eight elements within the folder: one directory named `drafts`; one hidden file named `.project_x.txt`; and five other project files. The 10-character string at the beginning of each line indicates the permissions set on each file or directory.

Describe the permissions string

```
-rw-r----- 1 researcher2 research_team   46 Oct 19 11:56 project_m.txt
```

In the example above, the line describing permissions for the `project_m.txt` file begins with a 10-digits string. Each character conveys different information about the permissions.

The **first character** indicates the type of file we're looking at:

- `d` for directory
- `-` for a regular file

The **second to fourth characters** indicate the type of permission for the User owner:

- `r` for read
- `w` for write
- `x` for execute
- in case of lack of permission, `-` will appear.

The **fifth to seventh characters** indicate the type of permission for the Group owner:

- `r` for read
- `w` for write
- `x` for execute
- in case of lack of permission, `-` will appear.

The **eighth to the tenth characters** indicate the type of permission for the Other owner:

- `r` for read
- `w` for write
- `x` for execute
- in case of lack of permission, `-` will appear.

In the example above, we are looking at a regular file type named `project_m.txt` where the User owner has permission to both read and write the file, the Group owner has read only permission and the Other owner lacks any permission. In this case, no owner has execute permission on the file.

Change file permissions

```
-rw-rw-rw- 1 researcher2 research_team   46 Oct 19 11:56 project_k.txt
```

I received a request to revoke write permission for the Other owner from the file named `project_k.txt`.

The following command allowed me to remove write permission for Other from the specified file:

```
chmod o-w project_k.txt
```

The `chmod` command allowed me to change permissions on the specified file with two arguments:

- the first argument after `chmod` describes how to change permissions. In this case `o-w` indicates that we want to remove (`-` sign) write permission to the Other owner;
- The second argument is the file name that needs the permission modified.

Change file permissions on a hidden file

I was able to check permissions for active and archived files by using `ls -la`. Archived files are hidden files that are recognisable by a `.` immediately in front of the file name.

```
researcher2@2fa5bc0fe905:~/projects$ ls -la
total 32
drwxr-xr-x 3 researcher2 research_team 4096 Oct 19 11:56 .
drwxr-xr-x 3 researcher2 research_team 4096 Oct 19 12:12 ..
-rw--w---- 1 researcher2 research_team   46 Oct 19 11:56 .project_x.txt
drwx--x--- 2 researcher2 research_team 4096 Oct 19 11:56 drafts
-rw-rw-r-- 1 researcher2 research_team   46 Oct 19 11:56 project_k.txt
-rw-r----- 1 researcher2 research_team   46 Oct 19 11:56 project_m.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 19 11:56 project_r.txt
-rw-rw-r-- 1 researcher2 research_team   46 Oct 19 11:56 project_t.txt
```

The `.project_x.txt` file was recently archived by the research team, therefore no owner was meant to have write permissions on it. It appeared that both User and Group owners still had write permissions on the specific file.

The following `chmod` command:

```
chmod u-w,g-w,g+r .project_x.txt
```

allowed me to revoke write permissions to both User (`u-w`) and Group (`g-w`). At the same time, I was able to add read permission to Group by adding `g+r`.

Change directory permissions

Currently, both User and Group owners have execute permission enabled for the directory named `drafts`. However, the company decided that only User owner (in this case `researcher2`) should have execute permission on that specific directory.

```
drwx--x--- 2 researcher2 research_team 4096 Oct 19 11:56 drafts
```

In order to modifiy the permissions accordingly I used the following command on Linux:

```
chmod g-x drafts
```

In the above example, I only needed to remove execute permission from the Group owner by specifying `g-x` in the first argument of the `chmod` command, since the User owner already had full permission for the `drafts` directory.

Summary

In the above example, I have shown my ability to update the existing permissions in the `projects` directory using Linux command-line in order to match the level of authorization my organization wanted. The main command that allowed me to achieve this was `chmod`.