Control access to files

Linux #redhat #filesystem #files

File Permissions

- Protect your environment, your files, and your directories from being viewed by other users or being deleted by other users
- UNIX is a multi-user system. Every file and directory in your account can be protected from or made accessible to other users by changing its access permissions. Every user has responsibility for controlling access to their files.
- Permissions for a file or directory may be restricted to by types
- There are 3 types of permissions
  - `r` - read
  - `w` - write
  - `x` - execute = running a program
    - If a file itself is a script or a program it has an `x` permission, which determines if it is executable.
    - If a user is allowed to use a program they would have an executable permissions to it.
- Each permission (rwx) can be controlled at thee levels:
  - `u` - user = yourself
  - `g` - group = can be people in the same project
  - `o` - other = everyone on the system
- File or Directory permission can be displayed by running `ls -l` command
  - E.g. `-rwxrwxrwx`
    - The first bit shows it is a file
    - The next three bit shows it has read, write, and executable permissions by the user.
    - The second three bits are for the group.
    - The third three bits are for the others.
- Command to change permission.
  - `chmod`

Example using `ls -l` command:

```
[user@localhost ~]$ ls -l jerry
```

- We are looking for the file named "jerry"

Output:

```
-rw-rw-r--. 1 mmarin mmarin 0 Jun  4 15:31 jerry
```

- The second three bits shows that the user "mmarin" has only read and write permissions
  - The user does not have executable permission, it means this file it is not a script. If it was a script. it would have an `x` to it.
- The next three bits show that the group itself also has `r` to read, and `w` to write to the file
- The last three bits shows that the "others" can only read this file.

Remove permissions `-`

Example using `chmod` command:

```
[user@localhost ~]$ chmod g-w jerry
```

- `chmod` is changing the permissions of the file named "jerry"
- The `g` option specifies that changes will be done at the group level (the second three bites)
- The `-w` term specifies that we are taking the write permission out of the group level of the "jerry" file.

Output from `ls -l jerry` :

```
-rw-r--r--. 1 mmarin mmarin 0 Jun  4 15:31 jerry
```

- Now it has no `w` in the second three bites

Example using `chmod` command:

```
[user@localhost ~]$ chmod u-w jerry
```

- `chmod` removing the write permission from the user level of the "jerry" file

A file with no permissions would look like this:

```
----------. 1 mmarin mmarin 0 Jun  4 15:31 jerry
```

Remove or add permissions to ALL levels

Example using `chmod` command:

```
[user@localhost ~]$ chmod a-r jerry
```

- `chmod` is changing the permissions of the file named "jerry"
- The `a` option stands for "all" and specifies that changes will be done for ALL levels (user, group, others).
- The `-r` terms specifies that we are removing the read permission for all levels

Output from `ls - l jerry:

```
--w--w----. 1 mmarin mmarin 0 Jun  4 15:31 jerry
```

- Note there is no permission `r` in any of the levels.

What happens if you want to read, write, or delete a restricted file?

Example using `rm` command:

```
[user@localhost ~]$ rm jerry
```

- Attempting to remove the file called "jerry" which has no permissions

Output:

```
rm: remove write-protected regular empty file 'jerry'?
```

- if you type "yes" next to this warning the file will actually be deleted because you are the actual creator of the file. But it does give you a warning sign.

Example using `cat` command:

```
[user@localhost ~]$ cat jerry
```

- We know the "jerry" file has no permissions

Output:

```
cat: jerry: Permission denied
```

- There is no read permission to read this file with the `cat` command

Add permissions `+`

Example using `chmod` command:

```
[user@localhost ~]$ chmod g+w jerry
```

- `chmod` is changing the permissions of the file named "jerry"
- The `g` option specifies that changes will be done at the group level (the second three bites)
- The `+w` term specifies that we are adding the write permission to the group level of the "jerry" file.
- This line revert the effects done by the previous command.

Output from `ls -l jerry` :

```
-rw-rw-r--. 1 mmarin mmarin 0 Jun  4 15:31 jerry
```

Example using `chmod` command:

```
[user@localhost ~]$ chmod u+rw jerry
```

- Add read AND write permissions in the user level to the "jerry" file

Output from `ls -l jerry` :

```
-rw-------. 1 mmarin mmarin 0 Jun  4 15:31 jerry
```

Example using `chmod` command:

```
[user@localhost ~]$ chmod g+rw jerry
```

- Add read AND write permissions in the group level to the "jerry" file

Output from `ls -l jerry` :

```
-rw-rw----. 1 mmarin mmarin 0 Jun  4 15:31 jerry
```

Example using `chmod` command:

```
[user@localhost ~]$ chmod o+r jerry
```

- Add read permission in the "others" level to the "jerry" file

Output from `ls -l jerry` :

```
-rw-rw-r--. 1 mmarin mmarin 0 Jun  4 15:31 jerry
```

Directories and the `x` permission

- If a directory contains the `x` permission for any user, group or others is beacause you could `cd` into that directory.

Example of how a directory with this characteristic would look like using the `ls -l` command:

```
drwxr-xr-x. 2 mmarin mmarin   6 Jun  5 12:34 seinfeld
```

- You are allowed to use:
  - `cd seinfield/`
    - Remember: When attempting to access a directory, it always has to end with the `/` sign as it is not a file.
    - To return to the previous directory location you can use `cd ..`

Example using `chmod` command:

```
[user@localhost ~]$ chmod a-x seinfeld/
```

- Removing the `x` (executable) permission at ALL (`a`) levels for the 'seinfeld/' directory.

Output from `ls -l` :

```
drw-r--r--. 2 mmarin mmarin   6 Jun  5 12:34 seinfeld
```

Example using `cd` command:

```
[user@localhost ~]$ cd seinfeld/
```

- Since we removed the `x` permission from the 'seinfeld/' directory this command won't be allowed to run

Output:

```
-bash: cd: seinfeld/: Permission denied
```

This is how you can protect your files and directories from being manipulated or deleted by accident.

File Ownership

There are 2 owners of a file or directory

- User and group
  - User is the one who creates the directory
  - The group is the group that the user belongs to
    - If you belong to two different groups and a permission applies to only one of the groups. As long as you are in that group, then you would have that permission.

Command to change file ownership

- `chown` and `chgrp`
  - `chown` changes the ownership of a file
  - `chgrp` changes the group ownership of a file

Recursive ownership change option (Cascade)

- `-R` - It will not only change the indicated directory but also all the files and directories within that parent directory.

You have to become root or use `sudo` in order to change the ownership or group ownership of a certain file.

Example using `chown` command:

```
[root@localhost mmarin]# chown root lisa
```

- Note we are in the home directory of the "mmarin" user (`/home/mmarin/`).
- `chown` is changing the ownership of the file called "lisa" to the root user.

Output from `ls -l lisa` :

```
-rw-r--r--. 1 root mmarin 0 Jun  4 16:00 lisa
```

Example using `chgrp` command:

```
[root@localhost mmarin]# chgrp root lisa
```

- `chgrp` is changing the group ownership of the lisa file to the root group.
- Note that the root group is not the same as the root user.

Output from `ls -l lisa` :

```
-rw-r--r--. 1 root root 0 Jun  4 16:00 lisa
```

Attempting to remove a file owned by other user inside your home directory

Example using `rm` command:

```
[user@localhost ~]$ rm lisa
```

Output:

```
rm: remove write-protected regular empty file 'lisa'? y
```

- Note that when the file you want to delete is not owned by yourself it will throw out a warning before proceeding to delete with the `rm` command.
- You can enter "y" to ignore the warning and actually delete the "lisa" file.
- The reason the file is gone now is because when you go one step back to the home directory `/home/` (you can do this by doing `cd ..` to go a step back) You will see that the "mmarin" user home directory, has the permission of yourself to read, write, and execute. Meaning anything inside of the directory you could read, you could write, and then you could execute. So regardless if whoever puts the file in whoever owns the file or whichever groups owns it, you could go ahead and still delete it.

Example using `ls -ltr` command:

```
[user@localhost home]$ ls -ltr
```

- Note we are currently located in the `/home/` directory.

Output:

```
total 4
drwx------.  3 spiderman superheros   78 Jun  9 18:14 spiderman
drwx------.  3 babubutt  babubutt     78 Jun 10 18:10 babubutt
drwx------.  4 ironman   superheros  113 Jun 11 11:47 ironman
drwx------. 20 mmarin    mmarin     4096 Jun 12 01:16 mmarin
```

- We can clearly see the `rwx` permissions for our user to manipulate files inside this directories.

Example using `touch` command:

```
[user@localhost etc]$ touch testfile
```

- Note the user is currently located at the `/etc` directory.
- `touch` is attempting to create a file inside this `/etc` directory named "testfile"

Output:

```
touch: cannot touch 'testfile': Permission denied
```

- It won't allow you because you do not have write permission for this file.
- How do you check this?, you do `ls -l /` in any location.

Example using `ls -l /` command:

```
[user@localhost ~]$ ls -l /
```

Output:

```
total 28
dr-xr-xr-x.   2 root root    6 Aug  9  2021 afs
lrwxrwxrwx.   1 root root    7 Aug  9  2021 bin -> usr/bin
dr-xr-xr-x.   5 root root 4096 Jun  3 13:38 boot
drwxr-xr-x.  20 root root 3300 Jun 12 00:14 dev
drwxr-xr-x. 132 root root 8192 Jun 12 01:05 etc
drwxr-xr-x.   6 root root   68 Jun 10 18:10 home
lrwxrwxrwx.   1 root root    7 Aug  9  2021 lib -> usr/lib
lrwxrwxrwx.   1 root root    9 Aug  9  2021 lib64 -> usr/lib64
drwxr-xr-x.   2 root root    6 Aug  9  2021 media
drwxr-xr-x.   2 root root    6 Aug  9  2021 mnt
drwxr-xr-x.   2 root root    6 Aug  9  2021 opt
dr-xr-xr-x. 304 root root    0 Jun 12 00:14 proc
dr-xr-x---.   4 root root 4096 Jun 11 12:22 root
drwxr-xr-x.  45 root root 1160 Jun 12 00:15 run
lrwxrwxrwx.   1 root root    8 Aug  9  2021 sbin -> usr/sbin
drwxr-xr-x.   2 root root    6 Aug  9  2021 srv
dr-xr-xr-x.  13 root root    0 Jun 12 00:14 sys
drwxrwxrwt.  17 root root 4096 Jun 12 01:09 tmp
drwxr-xr-x.  12 root root  144 Jun  3 13:10 usr
drwxr-xr-x.  20 root root 4096 Jun  3 13:37 var
```

- This is the most general view of all the directories of the machine.
- If we notice, the 'etc' directory is owned by root and the group is owned by group as well. Now that only the root user has the permission to write to it.
  - `drwxr-xr-x. 132 root root 8192 Jun 12 01:05 etc`
- What about the people who are in root's group?
  - They do not have a permission to write to it (no `w` permission at the group level)
- What about everybody else?
  - They do not have a permission to write to it (no `w` permission at the others level)

Example using `rm` command:

```
[user@localhost etc]$ rm test123
```

- Note that a file named "test123" was previously created by the root user inside the `/etc` directory.
- In this line a normal user is attempting to use the `rm` command to delete the "test123" file from the `/etc` directory

Output:

```
rm: remove write-protected regular empty file 'test123'? y
rm: cannot remove 'test123': Permission denied
```

- Note that even after ignoring the warning by inputting "y" (yes) we still cannot delete the file. The user has permission denied.
  - Why?
    - Go to `cd /` and run `ls -l` and check for the `/etc` directory permissions

Example using ls -l command:

```
[user@localhost /]$ ls -l
```

- Note we are currently in the `/` directory.

Output:

```
..
drwxr-xr-x. 132 root root 8192 Jun 12 01:05 etc
...
```

- Note there is no `w` (write) permission for neither group or others levels.

Create and configure set-GID directories for collaboration

Set-GID directories are directories that have the set-GID bit set. When a file is created in a set-GID directory, the file inherits the group ownership of the directory instead of the user's default group ownership. This is useful for collaboration because it allows multiple users to work on the same files with the same group ownership.

To create a set-GID directory, you can use the following commands:

```
$ sudo mkdir /path/to/directory
```

```
$ sudo chmod g+s /path/to/directory
```

This will create a directory with the set-GID bit set. When a user creates a file in this directory, the file will have the same group ownership as the directory

Umask & base permissions

More on Chapter 11. Managing file system permissions

Change default umask value with default umask value in RHEL 9.0 is 0022 for both root and normal user!