Abdullah ÇAY

Bash and Linux - Users and Permissions (EN)

45 minutes   Normal

Machine status

Ubuntu Server 18.04 LTS

SSD Volume Type

64-bit x86

Stopped

Connect

Reset   Start

DataScientest • com

Introduction to Linux

Users and permissions

As Windows or MacOS, Ubuntu can manage different users on the same machine. To prevent some malicious users to damage a working server, Linux provides a system of users and permissions. This can be divided into two parts: **ownership** and **permissions**.

User groups

Users can be grouped together in a user group that shares the same set of rights on some objects. For example, we can decide that every user of a group is allowed to use one program or not. This makes it easier to deal with numerous users and different levels of permissions. You can check in which groups the current user is by using the `groups` command. A user can be in several groups.

> Use this command to see in which groups the user is

Ownership and groups

When a file is created, the user that created it is defined as the owner of the file. The owner of the file has different rights on this file from all the other users. The file is also owned by a group the user is in. The users that are in this group have a certain set of permissions on the file that may not be the same as the owner user. Finally, every other user have another set of permissions.

Superuser

In the commands that you have run in the previous parts, some were not allowed. For example, if you were to run `rm /usr/bin/python3` as `ubuntu` user, you will get an error `rm: cannot remove '/usr/bin/python3': Permission denied`. As user `ubuntu` you do not have the right to remove this file. But there is a special user called `root` that can do anything. To connect as the root user, you can use `sudo su`.

> Connect as the root user

Note that the message is changed. You are no longer connected at `~` because this is not the home of the current user. Moreover, the `$` has changed to `#`.

> Exit the root user by typing in `exit`

To perform a single command as the root user, you can use `sudo` (super user do) followed by the command you want to run. Be aware that because you can do anything with this root user, you can also damage the files that are essential to the good working of the machine. Normally `sudo` is password protected but we decided, for simplicity that we could do without a password: it is bad practice though and this is solely for the purpose of this lesson.

Permissions

- reading permissions: you can open and read the file but cannot write in it
- writing permissions: you can write data into the file, rename it or delete it
- executing permissions: you can execute the content of a file

How to show permissions and ownership

To show permissions and ownership on a file, we can use the `ls -l` command.

> Create a file called `my_file` in your current directory

```
touch my_file
```

> Run the following command

```
ls -l my_file
```

The ouptput should be something similar to:

```
-rw-rw-r-- 1 ubuntu ubuntu 0 Sep 28 20:08
my_file
```

The information available can be read according to the following parts

- `-rw-rw-r--` : nature and permissions on the object
- `1` : number of links to that object
- `ubuntu` : name of the user that owns the object
- `ubuntu` : name of the group that owns the object
- `0` : size of the file
- `Sep 28 20:08` : last modification date
- `my_file` : name of the object

In the first part, we can see the nature and the permissions on this object

The first part tell the rights on the file. We find again the first character and then three strings of three characters.

- `-` : this gives us the nature of the object, we have `-` for a file, `d` for a directory and `l` for a link
- `rw-` : permissions of the owner user
- `rw-` : permissions of the user in the owner group
- `r--` : permissions of every other users

`r` means that the user or group of user has reading permission while `w` stands for writing permission and `x` for execution permission. For example, a group that has `rwx` permissions has every right on this file. On the other hand `rw-` means only reading and writing priviledges.

We are now going to create a user to see this.

> Write something into `my_file` as `ubuntu` user

Show / Hide solution

```
echo hello world > my_file
```

> Run the following command to add a user named `my_other_user`

```
# -m creates the /home/my_other_user directory
sudo useradd -m my_other_user
```

To connect as another user, you can use `sudo su` followed by the name of the user.

> Connect as `my_other_user`

```
sudo su my_other_user
```

Create a file my_other_file as my_other_user in ~ and leave with exit

```
echo hello world from other > ~/my_other_file
exit
```

As ubuntu user, we can read the content of the file. We can also show its information: `-rw-rw-r-- 1 my_other_user my_other_user 0 Sep 28 20:27 /home/my_other_user/my_other_file`. But as we do not belong to the my_other_group user group, we do not have writing permissions:

Try to modify the file with touch

```
touch /home/my_other_user/my_other_file
```

We can add our current user to the my_other_user group:

Run the following command to add ubuntu user to my_other_user group

```
sudo usermod -a -G my_other_user ubuntu
```

You can check that the change has been applied by using groups. Now we can modify the file. We can also chose to change the owner of the file with chown:

```
sudo chown ubuntu /home/my_other_user/my_other_file
```

And if we want to change the owner group, we can also do it:

Run the following command

```
sudo chown ubuntu:docker /home/my_other_user/my_other_file
ls -l /home/my_other_user/my_other_file
```

We can also create groups by using `groupadd some_groupname`. Rather than changing ownership, we can change the permissions. To do so, we can use chmod. There are two ways to use chmod.

The first way is to specify the destination permissions:

```
chmod 777 my_file
ls -l my_file
```

The number 777 is a ten base representation of the permissions. In binary, we can read:

- 0 : ---
- 1 : --x
- 2 : -w-
- 3 : -wx
- 4 : r--
- 5 : r-x
- 6 : rw-
- 7 : rwx

So 777 means rwxrwxrwx.

The other way is to add or delete some permissions to a group of users:

```
chmod a-rwx my_file
ls -l my_file
```

Here we specify a group of user:

- o is for the other users

Then we have a + or a - to add or delete permissions. And finally we specify the permissions to add or delete rwx, rx, ... For example, `chmod o+x my_file` gives execution permission to all the users in the other "group". Note that to change a file or a folder permissions you must be the owner user or the super user.

You can delete users or groups by using `userdel` ou `groupdel`.

Links

We have talked about three types of objects: files, directories and links. A link is simply a reference to another file or folder. To create one we can use `ln -s`.

> Run the following command

```
echo hello from a real file > my_real_file

ln -s my_real_file my_link

ls -l

cat my_link
```

We can see that printing the content of the symbolic link is the same as printing the content of the actual file. This is interesting to simplify the interaction with a file.

> Remove reading permissions from all users to `my_link` and try to print its content and the content of the actual file

```
# removing reading permissions
chmod a-r my_link

# listing permissions
ls -l

# printing its content
cat my_link

# printing the actual file
cat my_real_file

```

In this lesson, we have seen how to use permissions and ownership to grant or deny access to certain files, folders and symbolic links. We have talked about reading and writing priviledges but execution rights may be still a bit unclear. In the next part, we will see how to create executable files and how to run them.

Validate