LINUX · FILE ACCESS CONTROL

Permissions and Ownership

Who can do what with which file or folder?



BASIC IDEAS

Linux is a multi-user system

More than one user can operate the computer at the same time.

Users must not be allowed to interfere with:

Other users' files

Files that belong to another user are not theirs to read or change.

System files and folders

Configuration files, boot programs, the home directory and so on.

So Linux needs a way to decide who can access a file or folder, and what they can do with it.

BASIC IDEAS

The control mechanism has two parts

Permissions

What a user or a group of users can do to a file or a folder.

Ownership

Who is working with the file: the owner, a member of the group or anyone else. That decides which set of permissions applies.

Linux has two methods: the traditional access permissions (this lesson) and ACLs, which give finer-grained control.

THE MODEL

Three classes of users

u

Owner

The creator of the file by default. Only the owner can assign and change its permissions.

g

Group

Members of the group share the same permissions on the file.

o

Others

Everyone who is neither the owner nor in the group.

root is special: permissions and ownership have no practical effect on root, who can change any file or folder.

THE MODEL

Three types of permission

	FILE	FOLDER
r read	View the contents (cat, nano, geany…)	List the contents (ls, tree…)
w write	Add, remove or change the contents	Add, remove or rename entries (cp, rm, mv…)
x execute	Run it, if it is a program	Enter it (cd)

In practice, creating or deleting entries in a folder needs w and x together.

THE MODEL

Nine parameters for every file

	r · read	w · write	x · execute
Owner	r	w	x
Group	r	-	x
Others	r	-	-

Example: rwxr-xr--. Each of the nine values is either allowed or denied.

THE MODEL

Which set of permissions applies?

1

Owner?

If you are the owner, you get the owner's permissions.

2

In the group?

Not the owner but a group member: you get the group's permissions. The others' set is ignored.

3

Anyone else

Not the owner and not in the group: you get the others' permissions.

The first match wins. An owner with r-- cannot write, even if the others have rwx.

COMMANDS

Six commands to know

`ls -ls`

Displays permissions and ownership, plus size and date

`tree -pug`

Shows permissions, owner and group as a tree

`chmod`

Sets or unsets access permissions

`chown`

Changes the owner and the group

`chgrp`

Changes the group

`id`

Shows the groups a user belongs to

COMMANDS · INSPECT

Reading the output of ls –ls

```
376 -rwxr-xr-- 1 dacomo teachers 382911 Nov 23 zpack.tar.gz
```

376	Size in blocks of 1024 bytes
-rwxr-xr--	Type (1 character) plus the nine permissions
1	Links: 1 for a file, 1 or more for a folder
dacomo	Owner
teachers	Group
382911 Nov 23 zpack.tar.gz	Size in bytes, date, name

COMMANDS · INSPECT

The nine characters, in three groups

rwx

Owner

Permissions 1 to 3: read, write and execute.

r-x

Group

Permissions 4 to 6: read and execute.

r--

Others

Permissions 7 to 9: read only.

Before the nine comes the type: - file, d folder, l link. Inside the nine, a dash means “not granted”.

COMMANDS · INSPECT

tree -pug and id

tree -pug

Shows the permissions, owner and group of a folder and everything inside it, as a tree.

```
tree -pug /home/dacomo
```

p = permissions · u = owner · g = group

id

Lists the groups a user belongs to.

```
uid=1000(dacomo) gid=1000(dacomo)
groups=1000(dacomo),24(cdrom),
27(sudo),29(audio),…
```

Here dacomo is a member of dacomo, cdrom, sudo, audio and more.

COMMANDS · CHMOD

chmod numeric mode

Three digits, one per class. Each digit adds up r = 4, w = 2, x = 1.

0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Synopsis: chmod <permissions> file_or_folder_name

COMMANDS · CHMOD

Numeric mode: two examples

```
chmod 754 prova.sh
```

Owner	`7 rwx`	read, write, execute
Group	`5 r-x`	read and execute
Others	`4 r--`	read

```
chmod 310 prova.sh
```

Owner	`3 -wx`	write and execute
Group	`1 --x`	execute
Others	`0 ---`	nothing

COMMANDS · CHMOD

Symbolic mode

```
chmod [ugoa][+-][rwx] file_or_folder_name
```

Who

u	user (owner)
g	group
o	others
a	all

Action

+	add
-	remove

Permission

r	read
w	write
x	execute

COMMANDS · CHMOD

Symbolic mode: examples

`chmod u+r prova.sh`	Adds read permission to the owner
`chmod g-x prova.sh`	Removes execute permission from the group
`chmod a+x prova.sh`	Adds execute permission for everyone
`chmod ug+rw prova.sh`	Adds read and write to owner and group
`chmod ugo-wx prova.sh`	Removes write and execute from owner, group and others
`chmod a-wr prova.sh`	Removes read and write from everyone

COMMANDS · CHOWN AND CHGRP

Changing owner and group

```
chown <new_owner>:<new_group> file_or_folder_name
chown <new_owner> file_or_folder_name
chgrp <new_group> file_or_folder_name
```

`chown etpclot:users prova.sh`	Sets the owner to etpclot and the group to users
`chown etpclot prova.sh`	Sets only the owner to etpclot
`chgrp users prova.sh`	Sets only the group to users

RECURSIVE OPTION

-R applies the change to everything inside

`chmod -R 755 /home/student00`	Every file and folder inside gets rwxr-xr-x
`chown -R etpclot:users /home/student00`	Every file and folder inside gets owner etpclot and group users
`chgrp -R users /home`	Every file and folder inside gets the group users
`ls -ls -R /boot`	Lists /boot and everything inside it

Careful: -R changes everything below the folder. Check the path before you press Enter.

INTERESTING FACTS

Seeing the contents is not entering the folder

In the graphical interface you can list a folder that you cannot enter.

1

Change into the folder

Needs x on the folder.

DENIED

2

List the contents

Needs r on the folder.

ALLOWED

Example: a folder that gives you r-- only. You see the names, but you are not inside it.

INTERESTING FACTS

Running a program: with or without x?

Run it directly

```
./backup.sh
```

The system checks the x permission. Without x you get “Permission denied”, for compiled programs and scripts alike.

Run it through an interpreter

```
bash backup.sh
```

The interpreter (bash, python3…) is the program that runs. It only needs to read the script, so r is enough.

The interpreter is an external program, so it is subject to permissions and ownership too.

PRACTICE

Check your understanding

1. What does chmod 640 notes.txt give to owner, group and others?

2. Which command removes write permission from the group on data.csv?

3. A file is -r--rw-rwx. You are in its group, not the owner. Can you write to it?

4. A folder gives you r-- only. Can you cd into it? Can you list its names?

HOMEWORK

Read “About Permissions”

Section 5 of the lesson: writing proper sentences about permissions. This reading is mandatory.

Optional readings

https://linuxize.com/post/understanding-linux-file-permissions/

https://linuxcommand.org/lc3_lts0090.php