Sistemas Informáticos (Computer Systems)

Unit 05. Linux - Part 2

Authors: Sergi García, Alfredo Oltra

Updated October 2022

Licencia

Nomenclatura

A lo largo de este tema se utilizarán diferentes símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

📖 **Important**

❕ **Attention**

💬 **Interesting**

INDEX

UNIT 05. LINUX - PART 2

1. USERS IN LINUX

Linux is a multi-user operating system.

Users on Linux have a name associated to them, but internally they are identified by a number. This identifier is called UID. If two users have different name but the same UID, they are internally the same user. More information in https://en.wikipedia.org/wiki/User_identifier

Basically, there are two kinds of users: normal users and root.

- A normal user is a user with UID greater than 0 and can do limited operations and only access/modify the resources that he has permission to access.
- Root user is a user with UID=0. It is the main administrator of the system and virtually can do almost everything (change configuration, install programs, install drivers, run servers, read/delete any file, etc.).

❗ **Attention:** to do operations being root user is very dangerous (you can do a mistake and broke your system). If you enter in a system being root, you have to know very well what you are doing.

1.1 Files "/etc/passwd" and "/etc/shadow"

The list of users is stored in a file called "*/etc/passwd*". It stores several attributes like UID, home directory, if user is enabled or not, etc.

If we execute "*cat /etc/passwd*" we can view its content.

More information about "*/etc/passwd*" file can be found in https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/

Encrypted password can also be stored in "*/etc/passwd*", but it is not recommended for security reasons ("*/etc/passwd*" could be read by everybody).

For this reason, there is another file to store passwords called "*/etc/shadow*" that only the root user can read and modify. More information about "*/etc/shadow*" file can be found in

https://www.cyberciti.biz/faq/understanding-etcshadow-file/

💬 **Interesting:** summarizing, "*/etc/passwd*" stores general info of users and "*/etc/shadow*" stores encrypted passwords.

1.2 Command "sudo" and sudoers list

A few lines ago, we have said that there are 2 kinds of users: root and normal users. It is an inefficient and insecure way to manage admin accounts.

For this reason, modern Linux distributions like Ubuntu or Mint:

- By default, the root account is deactivated (you can't log in as root).
- There is a list called "sudoers". In this list, you can give several privileges to normal users.
- The most common (and useful) privilege is to "became root" temporally using a command called "sudo" before the instruction to perform.

With this tool and this configuration, the system can have more than one admin (each user that is in "sudoers list" can perform root operations).

Also, it is mandatory to use the command "sudo" before the command run as root. It increases security because it is supposed that if you use "sudo" you know what are you doing.

Example:

If user pepe (UID=1001) is in "sudoer list" and executes "*sudo cat fichero.txt*"

It executes the command "*cat fichero.txt*" being root (UID=0).

> 💬 **Interesting:** when you run first time in your session a sudo command (or your last sudo command was a lot of time ago), the system ask you your own login for security reason.

More information in https://en.wikipedia.org/wiki/Sudo

1.3 Command "su"

Command "*su*" is an abbreviation of "*Switch User*".

This command can be called:

- Without parameters: in this case, it tries to log as root (UID=0). It works even if root account is disabled.
- With parameters: it has a parameter that is the username that you want to log in.
- If you run the command being root, it automatically logs as the user. If you are a normal user, it asks you the user password.

Example:

"*su pepe*"

The system will try to log in as the user "pepe".

"*sudo su*"

The system will try you to log as root (UID=0).

More information in https://en.wikipedia.org/wiki/Su_(Unix)

1.4 Creating users on Linux

In this page, you can read information of how to create users (by command line) and if you wish, give them "sudo" privileges:

https://www.digitalocean.com/community/tutorials/how-to-add-and-delete-users-on-ubuntu-16-04

Also, you can watch an example with graphical interface in this video

https://www.youtube.com/watch?v=DQHS1tQ2Xt8

When you create a user on Linux, the default content of its new home directory is obtained from directory "*/etc/skel*". It works like a "template". More information in http://linuxg.net/the-unix-and-linux-skeleton-directory-etcskel/

2. GROUPS IN LINUX

Linux let you create groups of users. It is useful to give permissions or privileges (like "sudoers list") to a complete group (For example, you can give "sudo" privilege to a group and each member of this group could run sudo command to became root).

> 💬 **Interesting:** normally, Ubuntu and similar distributions (Lubuntu, Mint, etc.) are configured in a way that being member of group "sudo" let you do "sudo operations".

A user can be a member of several groups at a time.

Like users, groups have a name, but internally they are identified by an integer GID. If two groups

share the same GID, internally they are the same group.

2.1 File "/etc/group"

There is a file "*/etc/group*" where groups are listed. Each line is a group, and it stores some information like "name", "GID" and the most important value: the complete list of users that are members of that group.

More information about "*/etc/group*" in

https://www.cyberciti.biz/faq/understanding-etcgroup-file/

2.2 Creating groups on Linux

In this link, you can watch how to create a group and add an existing username to that group using console :

http://www.omnisecu.com/gnu-linux/redhat-certified-engineer-rhce/how-to-create-a-new-group-in-linux-using-groupadd-command.php

Also, you can view how to do it graphically in this video https://www.youtube.com/watch?v=ZNeWntArcOg

3. FILES AND DIRECTORIES IN LINUX

3.1 Types of files

On Linux, there are those types of files:

- **Regular files**: contains information. They are regular files, like we use every day.
- **Directories**: they are special files with references to other directories and files.
- **Links**:
  - **Symbolic links:** it is a file that contains the route to other file. It is similar to Windows shortcuts. If you delete the original file, the symbolic link remains, but it points to a non-existent file.
  - **Hard links**: it is not a type of file, it is a second name to a file. If you create a hard link of a file, for the file system they are the same file and there is no way to know which is the original. If a file have more than one reference, it is only deleted when all references are deleted.
  - **Special files**: they are files that usually represent physical devices, like storage units, printers, etc.

3.2 Hidden files

On Linux, hidden files are files that start with "**.**" like "*.bash*". When you list a directory, they don't appear, unless you use "*-a*" parameter. You can see them using "ls -a".

4. PERMISSIONS ON LINUX

In Linux using command line command "*ls -l*" you can view detailed information about files and directories. This information contains permissions of each file or directory.

The main types of permissions on Linux are:

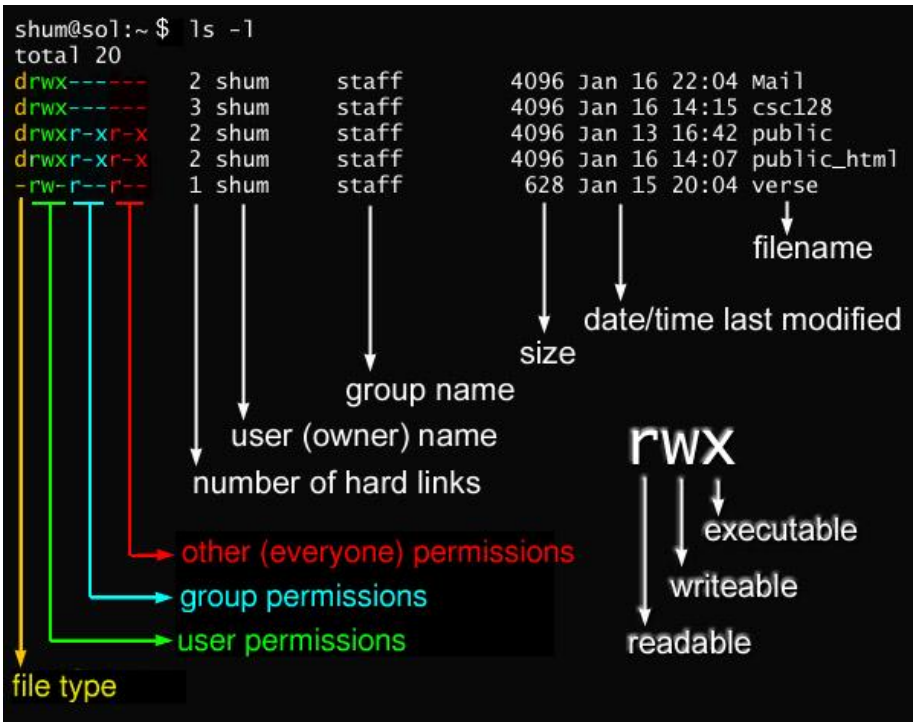
- **Read permission:**
  - **In a file**: lets to read its content.
  - **In a directory**: lets to list its files, directories names and attributes (command ls).
- **Write permission**:

- **In a file**: you can modify content of the file.
- **In a directory**: you can delete or create files and directories in that directory.

- **Execute permission**:
  - **In a file**: you can run the file (like Windows "*.exe*").
  - **In a directory**: you can enter the directory ("*cd*" command).

These main permissions should be defined in 3 groups: owner (affects to owner of the file), group (affects to member of the group) and others (affect to other users).

An example of "*ls -l*" command applied to permissions:



4.1 Permission grant algorithm

To determine if a permission is granted or not, it follows the next algorithm:

1. First, check if user is root (UID=0). If it is true, permission is granted.
2. Secondly, check if the user is the owner. If it is the owner, "owner permissions" are applied.
3. Thirdly, if the user is not root or the owner, but it is a member of the group associated to the file, "group permissions" are applied.
4. Lastly, if the user is not root, not the owner and not a member of the group, "other permissions" are applied.

It is possible to find contradictions like "others" have more permissions than "owner". If "others" can write and owner can't, although it is strange, it is a valid configuration.

4.2 Using "chmod" command to set permissions

Command "*chmod*" is used to set permissions. Only root and owner of the resource can change permissions.

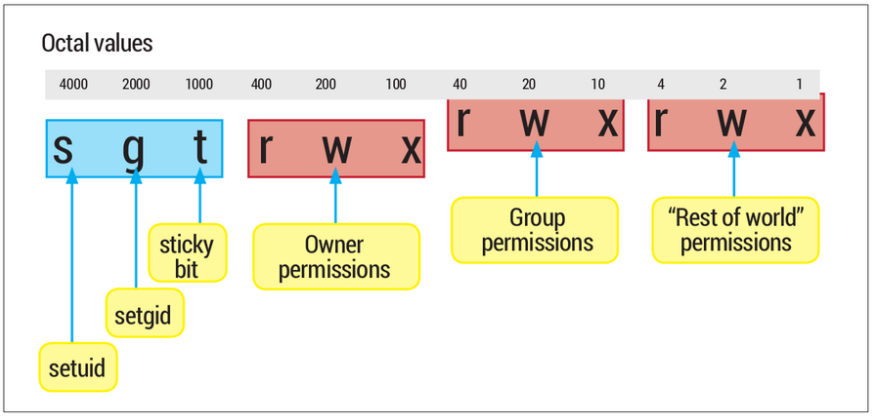
"*chmod*" mainly has two notations:

- **Alpha notation:**
  - **Example**: "*chmod u=rwx, g=rx, o=- myFile.txt*" It puts all permissions to owner, read and execution to group and nothing to others.

- **Octal notation:**
  - Uses "Binary value" of an octal value to set permissions. For example, 5 is 101 in binary, and it is equivalent in "*rwx*" to read and execute permissions.
  - Example: "*chmod 750 myFile.txt*" It puts the same permissions that last example.
  - More information about it in http://www.perlfect.com/articles/chmod.shtml

4.3 Special permissions

We have talked about 9 bits of permissions ("*rwx*" for owner, "*rwx*" for groups and "*rwx*" for other). But there are 3 bits more: "setUID", "setGID" and "Sticky bit":



- **setUID**: https://en.wikipedia.org/wiki/Setuid
  - In files: if **setUID** permission is activated, when you execute that file, you don't execute it with your own **UID**, you execute it with owner **UID**.
  - In directories: if **setUID** permission is activated, if you create a file or a directory, the owner isn't you, is the owner of the parent directory where you are.
- **setGID**: https://en.wikipedia.org/wiki/Setuid
  - The same as **setUID**, but with group **ID** instead user **ID.**
- **Sticky bit:** https://en.wikipedia.org/wiki/Sticky_bit
  - Nowadays is mainly used in directories. If somebody has write permission in a directory, he can create files and directories, but he also can delete any file or directory. If sticky bit is activated in a directory, any person with write permissions can create files and directories, but only can delete files and directories that are owned by him.
    - The only exception are root and owner of the parent directory.

More information about those permissions in
http://www.unixrock.com/2013/09/how-to-use-setuid-setgid-and-stickybit.html

5. MAIN LINUX COMMANDS

In this section, we are going to describe the main console commands on Linux systems. If you want to obtain detailed information about each of them, you can use "*man command*".

💬 **Interesting:** "man" is a command that show manual/help of other commands. It is very useful, and it is available in several languages (English, Spanish, etc.).

💬 **Interesting:** it is useful to have a Cheat Sheet. There are a lot of them. For example, this is very interesting https://linuxopsys.com/wp-content/uploads/2022/06/linux-cheat-sheet.pdf

Command	What it does	Example
Commands to manage the interface		
man	Shows help of a command.	*man ls*
clear	Clear screen.	*Clear*
echo	Show a literal text in screen.	*echo "Hello World"*
exit	Closes the session in console.	*exit*

Command	What it does	Example
Commands to configure the system		
date	Set date of the system.	*date #Shows date* *date -s #Sets date*
cal	Shows the calendar.	*cal*
shutdown	Shutdown the system.	*shutdown*
reboot	Reboot the system.	*reboot*

Command	What it does	Example
Commands to obtain information about disks		
du	Shows disk usage for each file.	*du -h* *#Human readable format*
df	Shows information about filesystems.	*df -h* *#Human readable format*

Command	What it does	Example
Commands to manage files and directories		
touch	Creates an empty file.	*touch myfile.txt*
vi / nano	Creates/edits a text file.	*nano myfile.txt* *vi myfile.txt*
mkdir	Creates a directory.	*make mydir*

`cat` `more`	Shows the content of a text file.	*cat myfile.txt* *more myfile.txt*
`grep`	Searches a text patron in a text file.	*grep root /etc/password*
`ls`	Shows contents of a directory	*ls* *ls -la*
`cd`	Changes directory	*cd /home #Absolute route* *cd ../myDir #Relative route*
`pwd`	Shows current route	*pwd*
`rm`	"rm" deletes files. "rm -r" deletes a directory recursively.	*rm myfile* *rm -r myDirectory*
`cp`	"cp" copy a file. "cp -r" copies a directory recursively.	*cp myFile /home/admin* *cp -r myDir /home/admin*
`mv`	Moves/renames a file or a directory.	*mv myFileOldName /home/myNewName*
`ln`	"ln" creates a hard link. "ln -s" creates a symbolic link (like Windows shortcuts).	*ln myFile hardLinkMyFile* *ln -s myFile shortcutMyFile*
`mount`	Mount a device in a folder.	*mount /dev/sda1 /media/myDisk*

Command	What it does	Example
Commands related to permissions		
chmod	Changes permissions of a file or a directory.	*chmod 750 myFile*
chown	Changes proprietary/group of a file or a directory.	*chown newuser:newgroupt my file*

6. BIBLIOGRAPHY

[1] "The Linux command line" Creative Commons book http://linuxcommand.org/tlcl.php

[2] "Linux commands Handbook"
https://bjpcjp.github.io/pdfs/devops/linux-commands-handbook.pdf