# Introduction to Linux Sybille Voisin (CC-IN2P3)

March  $17^{th}$ , 2025

# 1 Operating system

Linux is an operating system, i.e. the software that enables a computer to run and execute programs. It acts as an intermediary between hardware (processor, memory, hard disk, etc.) and software (web browser, text editor, etc.).



**Warning:** Do not confuse Unix and Linux: Unix is a proprietary operating system, while Linux is open source.



**Note:** There are various Linux distributions (Ubuntu, Debian, Fedora, etc.), adapted to different uses: personal computers, servers, embedded systems, supercomputers...

Linux is mainly used in web servers, supercomputers and by developers, because it's stable, secure and efficient.

The three layers of Linux:

- The kernel: close to the metal. Launches the machine, manages the graphics card, network, etc.
- The shell: a program for executing utilities and interacting (via windows or terminal).
- Utilities (ls, firefox...): programs run through the shell.

Two modes of use:

- Graphical user interface (GUI)
- Command line interface (CLI)

You're familiar with the first, as it corresponds to clickable windows, and the second to a terminal.

# 2 Shell

## 2.1 Terminal

Whatever your Linux version, you will find a "terminal" application that you can launch, and the default command interpreter is **Bash** (Bourne Again Shell), the most common command interpreter under Linux.

The terminal executes a **REPL** (read, eval, print, loop):

- 1. **Read**: The user types a command which is read by the interpreter,
- 2. Eval: This command is executed and returns a character string,
- 3. Print: The string is displayed on the screen,
- 4. Loop: Repeat.



**Tip:** In a code extract, a shell command is symbolized by a \$ (often called prompt).



**Notes:** A terminal can be accessed physically (in front of the machine) by running a program, or remotely via the ssh (secured shell) network service.

## 2.2 Basic commands

All these commands accept various options, the documentation for which can be consulted by typing: man followed by the command you want. Example:

#### \$ man ls

Example of file and directory management basic commands:

- cd: Change Directory Move around the directory tree
- 1s: List Displays contents of current directory
- cp: Copy files or directories
- mv: Move or rename files or directories
- rm: Delete files or directories
- cat: View file contents
- echo: Display a message or the contents of a variable
- touch: Create an empty file or reset a file's timestamp
- mkdir: Creates a new directory
- grep: Searches for a string in a file
- $\bullet\,$  wc: Counts the number of lines, words and characters in a file
- find: Searches for files or directories according to criteria
- awk: A text and file processing language



**Notes:** For a larger albeit non-exhaustive list of commands, see the following documentation:  $https://doc.ubuntu-fr.org/tutoriel/console_commandes_de_base.$ 

### 2.2.1 Different ways to create, read and modify a file

Select the terminal on your laucher. It is time to start with some simple file manipulations!

\$ touch myfile

\$ echo "Adding some content to the file" > echo\_file

```
$ echo "Appending a 2nd line" >> echo_file
```

\$ cat echo\_file
Adding some content to the file
Appending a 2nd line

```
$ cat echo_file > cat_file
```

A way to use cat command to add content to a file:

```
$ cat << EOF >> filename
> typing a first line
> typing a 2nd line
> EOF
$ cat filename
typing a first line
typing a 2nd line
```

It can be used as a copy-paste method to show the content of a file you are providing your audience. Copy-paste, as an example, the following lines on your terminal and execute:

```
cat << EOF >> filename
1st line of the script
2nd line of the script
3rd line...
EOF
```

With file editors such as Vi (*Vim*) and **Emacs**, creation and editing are all in the same command line:

\$ vi my\_vi-file

- To edit type i (allows you to enter the INSERT mode)
- To quit the file:
  - [ESC] (quits the INSERT mode) then type:
    - \* :wq if you want to save your modifications,
    - \* :q! if you want to discard your modifications.
- Vim cheatsheet: https://cheatsheets.zip/vim

```
$ emacs my_emacs-file
```

- You may edit immediately the file
- To quit: CTRL-x CTRL-c
  - You will be asked if you want to save or discard your modifications
- Emacs cheatsheet: https://cheatsheets.zip/emacs

### 2.3 Access rights and permissions

There are three categories of rights: user, group, rest of the world.

#### 2.3.1 Representing permissions with letters

Example:

```
$ ls -l fichier.txt
-rwxr-xr-- 1 utilisateur groupe 1234 fév 18 12:34 fichier.txt
```

Let's decompose the first part -rwxr-xr--.

Generally the first character represents:

- - : a file
- d : a folder
- 1 : a symbolic link

In our example, it is a -, so it is a file.

The following nine characters are in groups of three. In our example we have:

- -**rwxr**-**xr**-- (owner) : The user who owns the file
- -rwxr-xr-- (group) : Users in the same group
- -rwxr-xr-- (others) : All other users

Each letter means :

- r (read) : Can read the file
- w (write) : Can modify the file
- x (execute) : Can execute the file (if it's a program or script)

In our example:

- The owner rwx can do everything
- The group r-x can read and execute, but not write
- Others r-- can only read

#### 2.3.2 Representation with numbers (octal notation)

Each permission is associated with a number:

r = 4w = 2x = 1

These values are added together for each group:

rwx = 7 (4+2+1)r-x = 5 (4+0+1)r-- = 4 (4+0+0)

Thus, our -rwxr-xr-- file has 754 permissions.

### 2.3.3 Modify permissions

Allow execution also to the rest of the world (with numbers):

```
$ chmod 755 fichier.txt
$ ls -l fichier.txt
-rwxr-xr-x 1 utilisateur groupe 1234 fév 18 13:24 fichier.txt
```

Examples with letters

- chmod u+w fichier.txt  $\hookrightarrow$  Add (+) write (w) to owner (u)
- chmod g-x fichier.txt  $\hookrightarrow$  Remove (-) execution (x) from group (g)
- chmod o-r fichier.txt  $\hookrightarrow$  Remove (-) read (r) from other (o)

#### 2.4 SSH key configuration

SSH (secure shell) is an encrypted protocol used to administer and communicate with servers. SSH key pairs are two encrypted keys that can be used to authenticate a client to an SSH server. Each key pair consists of a public and a private key. The public key will be used to encrypt messages that only the private key can decrypt.

To connect to an interactive server in the CC-IN2P3 (cca machine), you need to use the SSH command.

```
$ ssh user@cca.in2p3.fr
```

then enter the password you have been given.

If you need to generate an SSH key, run:

```
$ ssh-keygen -t rsa -b 4096 -C "user@example.com"
```

- -t rsa: Uses the RSA algorithm (most common).
- -b 4096: Sets key length to 4096 bits (more secure).
- -C "user@example.com": Adds a comment to the key (useful for identifying it).

When asked where to save the key, press Enter to accept the default location (~/.ssh/id\_ed25519). If a key already exists, choose another name or save the old one before continuing.

You can also set a password to secure the key.

Then copy the public key to the cca.

```
$ ssh-copy-id user@adresse_cca
```

Connect to cca with SSH key:



Note: @cca.in2p3.fr is the address of the cca machine. You can connect to a specific machine, for example: @cca013.in2p3.fr. If you wish to connect to another center, the address will obviously change.

#### 2.5Public key for Gitlab

Open a shell and execute the following command:

\$ ssh-keygen -t ed25519 -C "user@example.com"



Note: Why ed25519? It's more secure and faster than RSA. But if your system does not support it, use : \$ ssh-keygen -t rsa -b 4096 -C "user@example.com"

When asked where to save the key, press Enter to accept the default location ( $^{\prime}$ .ssh/id\_ed25519). If a key already exists, choose another name or save the old one before continuing.

You can also set a password to secure the key.

Now we need to send the public key to GitLab. Retrieve the public key with this command:

\$ cat ~/.ssh/id\_ed25519.pub

#### 2.5.1Connect to your GitLab account and edit your profile

On https://gitlab.in2p3.fr/

#### Gitlab IN2P3

You already have an account on the IN2P3 Gitlab platform (read more ... )

▶ You are a new user from IN2P3 or from another academic organization (read more...)

You do not fit any of the situations above (read more...)

Please take a moment to read the documentation. Terms of use | Privacy



Remember me

To **login**, use the password you were given.

₩		Your work / Proje	acts		
Sybille Voisin @testaccount01					
Set status			Welco	me to G	itLab. Svbille!
Edit profile			Ready to get started wi	th Gitl ab? Follo	w these steps to get familiar with us:
Preferences			nousy to get started in	an onedo. i one	warese steps to get familier with do.
Sign out			① Explore public projects		Q Learn more about GitLab
Merge requests	>		Public projects are an easy way to allow everyone to have read-only access.		Take a look at the documentation to discover all of GitLab's capabilities.
🖸 To-Do List					
Milestones					
X Snippets					
S Activity					
lmport history					

## 2.5.2 Add SSH key to your profile

Copy the contents of the key and connect to your gitlab account. Go to User settings -> SSH Keys and Add new key.

₩			+ 🗇	User S	ettings / SSH Key	ys						
D	ະນ		⊠ 3	q1 Gitlab has been updated. More info here,								
Q	Search or	go to										
User sett	ings					Search settings						
Ø Profi	le					SSH Keys						
8º Acco	Account			SSH keys allow you to establish a secure connection between your computer and GitLab. SSH fingerprints verify that the client is connecting to the correct host. Check the current instance								
88 Appl	ications					connguration.						
🟳 Chat				Your SSH keys 🖉 2					Add new key			
🖾 Acce	ess tokens					Title	Kay	Lisage time	Created	Lastucad	Evpires	Actions
🖾 Ema	ils					Title	Key	Usage type	Created	Last used	Expires	Actions
						sybille.voisin@cc.in2p3.fr	P	Authentication & Signing	1 month ago	3 weeks ago	Never	Revoke 🖞
🔂 Pass	word											
A Pass A Notif	word					svbille.voisin	P	Authentication & Signing	2 months ago	2 months ago	Never	Revoke 17
<ul> <li>Pass</li> <li>⊥ Notif</li> <li>2 SSH</li> </ul>	word fications Keys					sybille.voisin	ρ	Authentication & Signing	2 months ago	2 months ago	Never	Revoke 🖞
<ul> <li>Pass</li> <li>D Notif</li> <li></li></ul>	word fications Keys Keys					sybille.voisin	P	Authentication & Signing	2 months ago	2 months ago	Never	Revoke D
<ul> <li>Pass</li> <li>A Notif</li> <li>A SSH</li> <li>A GPG</li> <li>C Prefe</li> </ul>	word fications Keys Keys erences					sybille.voisin	ρ	Authentication & Signing	2 months ago	2 months ago	Never	Revoke
<ul> <li>Pass</li> <li>Notif</li> <li>SSH</li> <li>GPG</li> <li>Prefi</li> <li>Com</li> </ul>	word fications Keys Keys erences ment Temp	plates				sybille.voisin	ρ	Authentication & Signing	2 months ago	2 months ago	Never	Revoke D
<ul> <li>Pass</li> <li>A Notif</li> <li>A SSH</li> <li>A GPG</li> <li>Com</li> <li>Com</li> <li>Com</li> </ul>	word fications Keys Keys erences ment Temp re Sessions	olates s				sybille.voisin	P	Authentication & Signing	2 months ago	2 months ago	Never	Revoke D
<ul> <li>Pass</li> <li>Notif</li> <li>SSH</li> <li>GPG</li> <li>Prefi</li> <li>Com</li> <li>Activ</li> <li>Auth</li> </ul>	word fications Keys Keys erences ment Temp re Sessions entication	olates s Log				sybille.voisin	β	Authentication & Signing	2 months ago	2 months ago	Never	Revoke D

Paste the key into the field, enter a title, a date (you can leave the date blank) and click on  ${\bf Add}$   ${\bf Key}.$ 

₩ □ +	User Settings / SSH Kays
O Search or go to	SSH Keys SSH keys allow you to establish a secure connection between your computer and GitLab. SSH fingerprints verify that the client is connecting to the correct host. Check the current instance
User settings	configuration. Your SSH keys $p^2$
Profile     Account     Applications	Add an SSH key Add an SSH key for secure access to GitLab. Learn more. Key
<ul> <li>Chat</li> <li>Access tokens</li> <li>Emails</li> </ul>	
Password  Notifications	Banjas with tech-sent tandes.ebx2.nistx2581 tandes.ebx2.nistx20241 tandes.ebx2.nistx5211 tech.ad255101 tech.ad255101 tech.ad255100
SSH Keys     GPG Keys	edgina mini sani sa, ocusa sinzi nisyzuo, ecusa sinzi nisybuo4, ecusa sinzi nisybuzi, sani euzuota, akecusa sinzi nisybuo@opensan.com, or akisani edu25519@opensh.com. Title
Preterences     Comment Templates     Active Sessions	Example: MacBook key Key titles are publicly visible,
<ul> <li>Authentication Log</li> <li>Usage Quotas</li> </ul>	Usage type Authentication & Signing
	Expiration date         2026-02-28<
	Add key Cancel

To test that the connection is working properly, run the command :

```
$ ssh -T git@gitlab.in2p3.fr
```

If everything is set up correctly, you'll see a message like:

"Welcome to GitLab, @user!"

# 3 BASH scripts

A sequence of commands allows the administrator to automate certain tasks, known as **scripts**, stored in a file with the extension **.sh**.

Script arguments can be invoked with 1,2 etc., their number with \$# and their list with \$\*.

Here, for example, is a **delete.sh** script which, instead of deleting the file passed as an argument, moves it to a "trash" folder on the user's HOME.

The first line indicates the interpreter used.

```
#! /bin/bash
# This script has one argument: a filename
# - if necessary, creates a trash directory in the user's HOME directory
# - moves the file given as argument into this directory trash
# first check whether this folder exists in ~/ :
if [ -d ~/trash ]
then
    echo "The trash directory already exists in your Home."
else # if it doesn't, we create it
    mkdir ~/trash
    echo "Trash directory does not exist. It has been created."
fi
# Move the file given as an argument to trash.
mv $1 ~/trash
```