Linux
https://www.linuxfoundation.org/

An Introduction to Linux and the command line

Sébastien LE ROUX sebastien.leroux@ipcms.unistra.fr

INSTITUT DE PHYSIQUE ET DE CHIMIE DES MATÉRIAUX DE STRASBOURG,
DÉPARTEMENT DES MATÉRIAUX ORGANIQUES,
23 RUE DU LOESS, BP43,
F-67034 STRASBOURG CEDEX 2, FRANCE

FEBRUARY 11, 2026

Contents

License

Operating System

An Operating System (OS), Système d'Exploitation (SE) in French, is system software that manages computer hardware, software resources, and provides common services for computer programs.

Roughly speaking the OS is the first program that will start when you turn on your computer, without the OS it is not possible to use other programs.
There are two main OS families, the MS-DOS family, and the much larger UNIX family. MS-DOS is the ancestor of Windows and UNIX is the ancestor of MacOSX, BSD systems, and more importantly Linux:

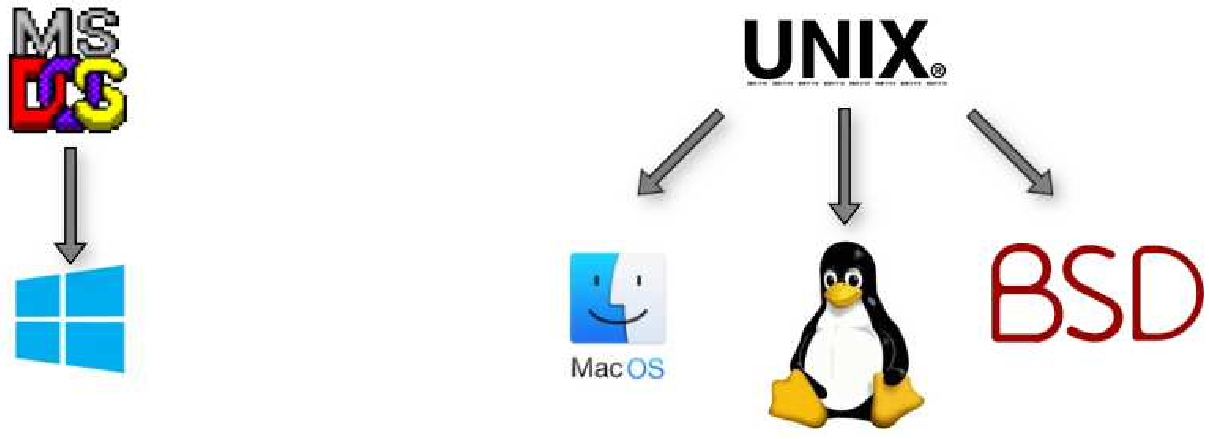


Figure 2.1: The main Operating Systems family tree.

As illustrated in figure [Fig. 2.1] MacOS (OSX, iOS), Linux and BSD (Berkley Software Distribution, open source UNIX) systems are somehow cousins.

2.1 Few numbers about the OS

2.1.1 Personal computers

	MacOS		
77 %	13 %	6 %	2 %

Table 2.1: Overall distribution on personal computers.

2.1.2 Smartphones and mobile devices

android	iOS	Others
75 %	20 %	5 %

Table 2.2: Overall distribution on smartphones and mobile devices.

2.1.3 Linux and the Internet

As I am writing this document the world wide web is made of an assembly of more than **270 000 000** web servers on 18 000 000 physical servers located all around the globe. Those 18 000 000 computers works 24h a day, 365 days a year to ensure that internet is internet as we know it today.
Here are few numbers to illustrate the importance of Linux for the internet:

71 %	29 %

Table 2.3: Overall distribution on internet web servers.

Few more numbers to emphasize even more Linux major importance over the internet:

- For the top `1 000 000`, meaning the `1 000 000` of the most used, and important web servers across the internet, Linux represents `96.5 %` of the OS.
- For the entire cloud computing systems, Linux represents `90 %` of the OS.
- The super computing centers exclusively use Linux.

2.1.4 Overall for every type of devices

android		MacOS + iOS	
40.5 %	34.2 %	22.3 %	1 %

Table 2.4: Overall distribution on all devices.

As you can see in table [Tab. 2.4], and that might not be a surprise for you Android, is the most popular operating system out there. That seems obvious considering the number of cell phones and mobile devices.
What might be a surprise for you however is that Android actually is a Linux distribution (see section [Sec. 4.1] for more details) !

If you are using a mobile running Android, just so you know you already are a Linux user !!!

Therefore the appropriate way to look at the distribution of OS, all devices considered is:

		MacOS + iOS
41.5 %	34.2 %	22.3 %

Table 2.5: The true overall distribution of OS, all devices considered.

Yes the truth is that Linux already is the most popular Operating System out there, welcome to the open source world !

Open Source Software

Adapted from the Wikipedia web page

Open-source software (OSS) is computer software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose. Open-source software may be developed in a collaborative public manner, making a prominent example of open collaboration, meaning any capable user is able to participate online in development, making the number of possible contributors indefinite. The ability to examine the code facilitates public trust in the software.

3.1 A little bit of history

- 1983 : Creation of GNU and the GPL license by Richard M. Stallman (**rms**)

- 1985 : Creation of the Free Software Foundation by RMS

- 1991 : Linux is developed by Linus Torvalds

- 2004 : Open source software officially enter UNESCO world heritage

3.2 The Four Essential Freedoms of Free Software

Thereafter the word "free" does not refer to price; it refers to freedom.
The first definition published by the FSF in February 1986 had two points:

- The freedom to copy a program and redistribute it to your neighbors, so that they can use it as well as you.

- The freedom to change a program, so that you can control it instead of it controlling you; for this, the source code must be made available to you.

In 1996, when the https://www.gnu.org website was launched, "free software" was defined referring to "three levels of freedom" by adding an explicit mention of the freedom to study the software (which could be read in the two-point definition as being part of the freedom to change the program).

Finally, another freedom was added, to explicitly say that users should be able to run the program. The existing freedoms were already numbered one to three, but this freedom should come before the others, so it was added as "freedom zero".
The modern definition defines free software by whether or not the recipient has the following four freedoms:

0. The freedom to run the program as you wish, for any purpose (freedom 0).

1. The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1). Access to the source code is a precondition for this.

2. The freedom to redistribute copies so you can help your neighbor (freedom 2).

3. The freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Freedoms **1** and **2** require source code to be available because studying and modifying software without its source code is highly impractical.
Proprietary is unlikely to give you access to any of these freedoms, or maybe freedom 0 if you pay and under certain conditions.
Open Source / Free software will give you access to freedom 0 and/or 1, and/or 2, and/or 3, depending on the software license.
Indeed like any other product open source software should be protected by a license, and there are many:

- the GPL licenses
- The BSD licenses
- The Apache license
- The Creative Commons license
- etc...

3.3 Famous open source software

Here are some famous open source software:

- Office:

  Libre Office The Document Foundation

- Multimedia:

- Operating Systems:
  - Linux distributions:

    android

  - BSD (open source UNIX):

    OpenBSD NetBSD FreeBSD

- Desktop managers:

  GNOME XFCE

- Internet:

  php APACHE HTTP SERVER PROJECT MySQL

Linux

Figure 4.1: Tux the official mascot of the Linux kernel.

Adapted from the Wikipedia web page

Linux is a family of open-source Unix-like operating systems based on the Linux kernel, an operating system kernel first released on September 17, 1991, by Linus Torvalds.

Linux was originally developed for personal computers based on the Intel x86 architecture, but has since been ported to more platforms than any other operating system.
Because of the dominance of the Linux-based Android on smartphones, Linux also has the largest installed base of all general-purpose operating systems.
Linux also runs on embedded systems, i.e. devices whose operating system is typically built into the firmware and is highly tailored to the system. This includes routers, automation controls, smart home technology, televisions, automobiles (for example, Tesla, Audi, Mercedes-Benz, Hyundai, and Toyota all rely on Linux), digital video recorders, video game consoles, and smartwatches.

Linux is one of the most prominent examples of free and open-source software collaboration. The source code may be used, modified and distributed commercially or non-commercially by anyone under the terms of its respective licenses, such as the GNU General Public License.

Greg Kroah-Hartman is the lead maintainer for the Linux kernel and guides its development. William John Sullivan is the executive director of the Free Software Foundation, which in turn supports the GNU components. Finally, individuals and corporations develop third-party non-GNU components. These third-party components comprise a vast body of work and may include both kernel modules and user applications and libraries.
Linux vendors and communities combine and distribute the kernel, GNU components, and non-GNU components, with additional package management software in the form of Linux distributions.

4.1 Linux distributions

Adapted from the Wikipedia web page

A Linux, or GNU/Linux, distribution (often abbreviated as distro) is an operating system made from a software collection that is based upon the Linux kernel and, often, a package management system. Linux users usually obtain their operating system by downloading one of the Linux distributions, which are available for a wide variety of systems ranging from embedded devices (for example, OpenWrt) and personal computers (for example, Linux Mint) to powerful supercomputers (for example, Rocks Cluster Distribution).
A typical Linux distribution comprises a Linux kernel, GNU tools and libraries, additional software, documentation, a window system (the most common being the X Window System, or, more recently, Wayland), a window manager, and a desktop environment.
Most of the included software is free and open-source software made available both as compiled binaries and in source code form, allowing modifications to the original software. Usually, Linux distributions optionally include some proprietary software that may not be available in source code form, such as binary blobs required for some device drivers.

A Linux distribution may also be described as a particular assortment of application and utility software (various GNU tools and libraries, for example), packaged together with the Linux kernel in such a way that its capabilities meet the needs of many users. The software is usually adapted to the distribution and then packaged into software packages by the distribution's maintainers. The software packages are available online in so-called repositories, which are storage locations usually distributed around the world. Beside glue components, such as the distribution installers (for example, Debian-Installer and Anaconda) or the package management systems, there are only very few packages that are originally written from the ground up by the maintainers of a Linux distribution.

As illustrated in figures [Fig. 4.2 and Fig. 4.3] the Linux ecosystem in extremely wide, almost one thousand Linux distributions exist. Because of the huge availability of software, distributions have taken a wide variety of forms, including those suitable for use on desktops, servers, laptops, netbooks, mobile phones and tablets, as well as minimal environments typically for use in embedded systems. There are commercially-backed distributions, such as Red Hat Enterprise Linux, Fedora, CentOS (backed by Red Hat), openSUSE (backed by SUSE) and

Ubuntu (backed by Canonical Ltd.), and entirely community-driven distributions, such as Debian, Slackware, Gentoo or Arch Linux.
Most distributions come ready to use and pre-compiled for a specific instruction set, while some distributions (such as Gentoo) are distributed mostly in source code form and compiled locally during installation.

In the following snapshot examples will be taken from:

Ubuntu 20.04.3 LTS and 22.04.2 LTS, both running the GNOME desktop.

For more details see chapter [Chap. 5].

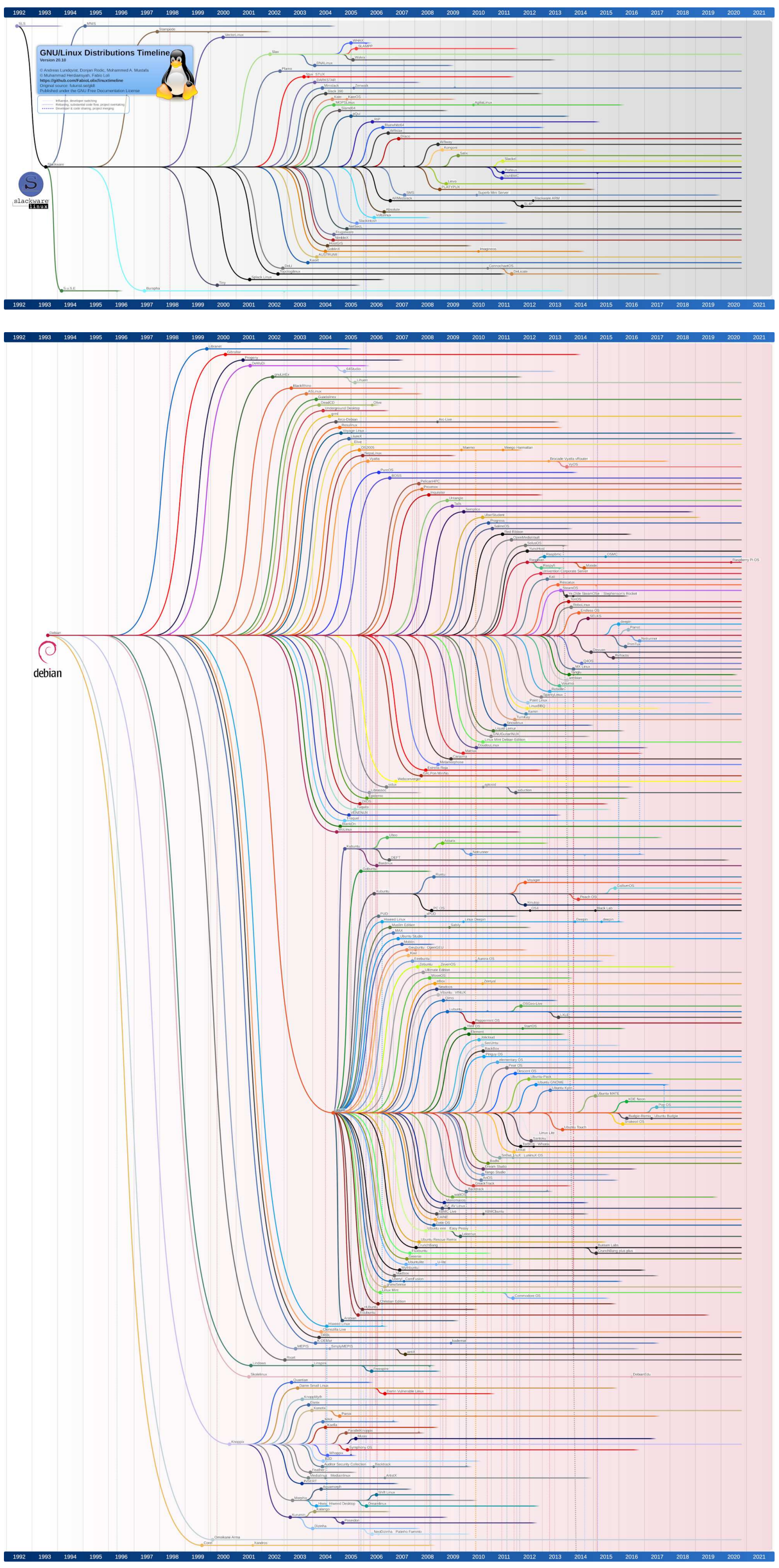


Figure 4.2: The Linux Distribution Time Line, top: Slackware tree, bottom: the Debian tree.

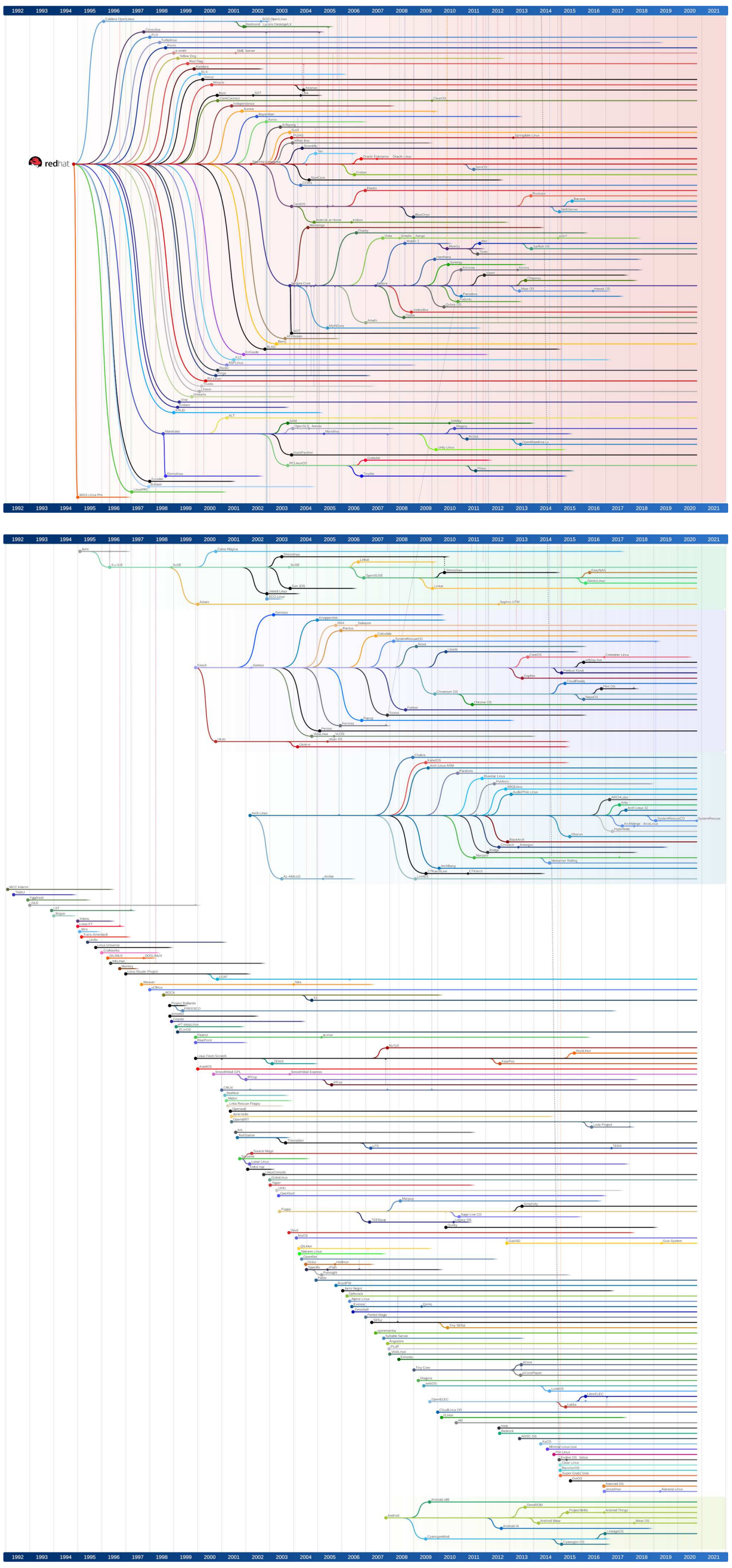


Figure 4.3: The Linux Distribution Time Line, top: the Red Hat tree, bottom: the others tree.

4.2 Linux fundamentals

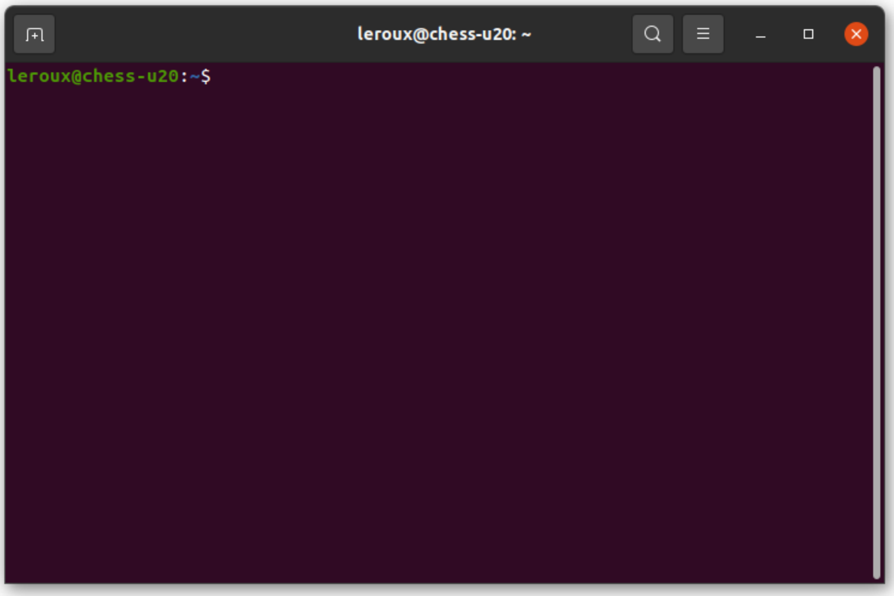
4.2.1 The terminal

That could seems surprising to put it first, yet considering that most of the examples and commands thereafter come from using the terminal it seems somehow appropriate to introduce it at the beginning. Also the purpose of this tutorial is to introduce the basics required to understand how to use this powerful tool.

The Linux command line is a text interface to your computer. Often referred to as the shell, terminal, console, prompt or various other names, it can give the appearance of being complex and confusing to use.

Yet someday, maybe sooner than expected, you will search the net looking for help to solve your Linux issue(s), then you will likely be faced with that simple truth: the answers are, and will, always, be provided in terms of commands that you need to copy and paste from the website to the terminal (you might find some more traditional help, in non-Linux terms that is ... using mouse and/or menus). Combined with the power and flexibility the command line offers, using it may be essential when trying to follow instructions online, and even more if you want to get the best of your Linux system.

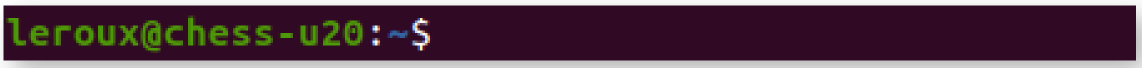
After opening the terminal (see section [Sec. 5.2.2.3]) you should end up with a rather dull looking window with an odd bit of text at the top, much like figure [Fig. 4.4].



```
leroux@chess-u20:~$
```

Figure 4.4: The GNOME terminal in Ubuntu 20.04.3 LTS

Depending on your Linux system the colors may not be the same, and the text will likely say something different, but the general layout of a window with a large (mostly empty) text area should be similar.
When you type a command it appears on the same line as the odd text:



```
leroux@chess-u20:~$
```

The prompt, can look like (Ubuntu Linux):

```
user@localhost:~$
```

or (Fedora Linux):

```
user@localhost ~]$
```

`user`	=	the name of the user who opened the terminal
`localhost`	=	the name of the computer
~	=	the active location in the directory structure (see section [Sec. 4.2.2.1]).

Figure 4.5: The prompt, or, command line interpreter.

That text (see figure [Fig. 4.5]) is there to tell you the computer is ready to accept a command, it's the computer's way of prompting you. In fact it's usually referred to as the prompt, and you might sometimes see instructions that say "bring up a prompt", "open a command prompt", "at the bash prompt" or similar. They're all just different ways of asking you to open a terminal.
When you run a command any output it produces will usually be printed directly in the terminal, then you'll be shown another prompt once it's finished. Some commands can output a lot of text, others will operate silently and won't output anything at all. Don't be alarmed if you run a command and another prompt immediately appears, as that usually means the command succeeded.

Case sensitivity: be extra careful with case when typing in the command line.
Typing **`LS`** instead of **`ls`** will produce an error, but sometimes the wrong case can result in a command appearing to run, but not doing what you expected.

Chapter [Chap. 6] introduces the basics of using the command line, and you can read more in The Linux Command Line by William Shotts.

A basic command glossary for the command line can be found in chapter [Chap. 8].
And my "Basic tutorial to BASH programming" illustrates much more about what can be done with the terminal using programming.

4.2.2 Filesystem Hierarchy Standard

Adapted from the Wikipedia web page

The Filesystem Hierarchy Standard (FHS) defines the directory structure and directory contents in Linux distributions. It is maintained by the Linux Foundation. The latest version is 3.0, released on 3 June 2015.

4.2.2.1 Directory Structure

In the FHS, all files and directories appear under the root directory `/`, even if they are stored on different physical or virtual devices. Table [Tab. 4.1] presents a brief description of the Linux Directory Structure, most of these directories exist in all Unix-like operating systems and are generally used in much the same way; however, the descriptions here are those used specifically for the FHS and are not considered authoritative for platforms other than Linux.

Directory	Description
/	Primary hierarchy root and root directory of the entire file system hierarchy.
/bin	Essential command binaries that need to be available in single-user mode, including to bring up the system or repair it, for all users (e.g., cat, ls, cp).
/boot	Boot loader files (e.g., kernels, initrd).
/dev	Device files (e.g., /dev/null, /dev/disk0, /dev/sda1, /dev/tty, /dev/random).
/etc	Host-specific system-wide configuration files "Editable Text Configuration" or "Extended Tool Chest".
/home	Users' home directories, containing saved files, personal settings, etc.
/lib	Libraries essential for the binaries in **/bin** and **/sbin**.
/lib32	Libraries essential for the binaries in **/bin** and **/sbin**.
/lib64	Libraries essential for the binaries in **/bin** and **/sbin**.
/media	Mount points for removable media such as USB-keys or CD-ROMs.
/mnt	Temporarily mounted file systems.
/opt	Add-on application software packages.
/proc	Virtual file system providing process and kernel information as files. Generally, automatically generated and populated by the system, on the fly.
/root	Home directory for the root user.
/run	Run-time variable data: Information about the running system since last boot, e.g., currently logged-in users and running daemons.
/sbin	Essential system binaries (e.g., fsck, init, route).
/srv	Site-specific data served by this system, such as data and scripts for web servers, data offered by FTP servers, and repositories for version control systems.
/sys	Contains information about devices, drivers, and some kernel features.
/tmp	Directory for temporary files (see also /var/tmp). Often not preserved between system reboots and may be severely size-restricted.
/usr	Secondary hierarchy for read-only user data; contains the majority of (multi-)user utilities and applications. Should be shareable and read-only.
/usr/bin	Non-essential command binaries (your every day applications); for all users.
/usr/include	Standard include files (for programming purposes).
/usr/lib	Libraries for the binaries in **/usr/bin** and **/usr/sbin**.
/usr/lib32	Alternative-format libraries
/usr/lib64	Alternative-format libraries
/usr/local	Tertiary hierarchy for local data, specific to this host. Typically has further subdirectories (e.g., bin, lib, share).
/usr/sbin	Non-essential system binaries (e.g., daemons for various network services).
/usr/share	Architecture-independent (shared) data.
/usr/srce	Source code (e.g., the kernel source code with its header files).
/usr/X11R6	X Window System, Version 11, Release 6 (up to FHS-2.3, optional).
/var	Variable files: files whose content is expected to continually change during normal operation of the system, such as logs, spool files, and temporary e-mail files.
/var/cache	Application cache data.
/var/lib	State information. Persistent data modified by programs as they run (e.g., databases, packaging system metadata, etc.).
/var/lock	Lock files. Files keeping track of resources currently in use.
/var/log	Log files. Various logs.
/var/mail	Mailbox files. In some distributions, these files may be located in the deprecated **/var/spool/mail**.
/var/opt	Variable data from add-on packages that are stored in **/opt**.
/var/run	Run-time variable data. This directory contains system information data describing the system since it was booted.
/var/spool	Spool for tasks waiting to be processed (e.g., print queues and outgoing mail queue).
/var/spool/mail	Deprecated location for users' mailboxes.
/var/tmp	Temporary files to be preserved between reboots.

/mandatory, /optional

Table 4.1: The Linux Directory Structure

4.2.2.2 Writing file paths

On a Linux system the path of a file, or a directory, is written starting from the root directory `/`, directory are separated using the `/` symbol. Using this notation the path to reach the home directory of a user can be written:

- Starting from the root directory `/`:

```
/home/user
```

- The particular path to the home directory has a command shortcut frequently used: `~`

```
/home/user    =    ~
```

Then the path to reach the directory named `Desktop` and located in this home directory, `user` could use the following notations:

- Starting from the root directory `/`:

```
/home/user/Desktop
```

- Starting from the home directory `~`:

```
~/Desktop
```

Similarly the path to the file `MyFile` located on the `Desktop` of `user` will be written:

- Starting from the root directory `/`:

```
/home/user/Desktop/MyFile
```

- Starting from the home directory `~`:

```
~/Desktop/MyFile
```

4.2.3 User management

The Linux system is designed as a multi-user system, it means that many users can work with the system at one time. That is, on a Linux system it is possible to distinguish three types of users:

- The "super" user or administrator (often called **root**) who can do everything in particular administrate and thus change the configuration of the system.
- The "`sudo`ers" for "**super user do**"-users who can use the "`sudo`" command to request admin privileges. When using the "`sudo`" command, **suoders** are required to confirm their identity by entering their user password.
- The "standard" user who plays, works, in short who uses the computer but with a restricted access.

Depending on the Linux distribution the super user can:

- Have an account on the computer and therefore access his/her personal home directory (`/root`). In that case to administrate the computer the super user has to log-in using the "`su -`" command.

- Not have an account, then only users that have been granted permission can administrate the computer using the "`sudo`" command.

Notice that on a Linux system it is usual to find group of users, ie. users that share the same privileges and that have been gathered into a group.

4.2.4 File permissions

The purpose of this section is to introduce the basic ideas on the file permission system on a Linux/Unix system. On a Linux system file permissions does not necessarily means administrating the system by installing the driver for the newly installed piece of hardware or updating the program library of the computer. Understanding the file permission system of the Linux system is the most basic prerequisite to becoming a 'Power' user, ie. a user having the power to understand what he/she is doing with the computer.

The golden rule of a Linux/Unix system is that everything is a file. The computer is seen by the operating system as a file tree and hence each component (screen, keyboard, mouse, graphic card ...) is seen as a file. When a hardware component is added to the computer, a file is added in the tree of the operating system `/`. Therefore permissions to manipulate the hardware from the software point of view are handled like the one of a file on the hard drive.
In a file tree one can distinguish two objects the first are the 'simple files' and the second the 'directories'. For both permissions are handled on the same way:

- The different permissions that can be granted for a simple file are:
  - read: to visualize its content
  - write: to modify its content (ex: editing)
  - execute: to execute its content (ex: program)
- The different permissions that can be granted for a directory are:
  - read: to visualize its content
  - write: to modify its content (ex: adding new files)
  - execute: to go inside this directory (ex: changing directory)

To obtain the information regarding the permissions of a file in the Linux tree one case use the `ls` command (see section [Sec. 8.1]):

```
user@localhost:~/Desktop$ ls -lh MyFile
-rwxrw-r--. 1 user ipcms 1.0K  15 févr.  2011 MyFile
```

The syntax for this line is the following:

- "`-`" means "file", alternatively "`d`" means directory and "`l`" means symbolic link.
- "`rwxrw-r-`" are the permissions on the file.
- "`1`" is the number of physical links of the file with the hard drive.
- "`user`" is the name of the owner of the file.
- "`ipcms`" is the name of the group the owner of the file belongs to.
- "`1.0K`" is the size of the file on the hard drive.
- "`15 févr.  2011`" is the last modification date of the file.
- "`MyFile`" is the name of the file

The permissions on the file can be decomposed in 3 series of 3 letters: r (for read), w (for write) and x (for execute), also the symbol `-` in place of a letter means that the permission is denied. The first 3 letters refer to the owner of the file, the second to the group the owner belong to, and the third to all other users of the computer. In the case of the file `MyFile` the owner of the file, `user`, has all permissions (read, write execute). The members of the group the owner of the file belongs to (so the members of the group `ipcms`) can read and modify the file. Finally other users can only read the file `MyFile`.

4.2.4.1 Adjusting file permissions: using the terminal

It is possible to define the access, utilization or modification permissions of a file with the `chmod` command (see section [Sec. 8.1]) and using 3 numbers:

(0	=	no permission at all	=	`---`)
1	=	execute	=	`--x`
2	=	write	=	`-w-`
4	=	read	=	`r-`

As well as their combinations:

3	=	1 + 2	=	execute + write	=	`-wx`
5	=	1 + 4	=	execute + read	=	`r-x`
6	=	2 + 4	=	write + read	=	`rw-`
7	=	1 + 2 + 4	=	execute + write + read	=	`rwx`

Linux distinguishes 3 classes of users for whom it is possible to grant permissions on a file:

- The owner of the file.
- The group the owner belongs to.
- All other users recognized by the system.

The permissions on the file being granted for each user category by a single combination of the number 1, 2 and 4.

For example it is possible to grant permissions 644 to the file `MyFile`:

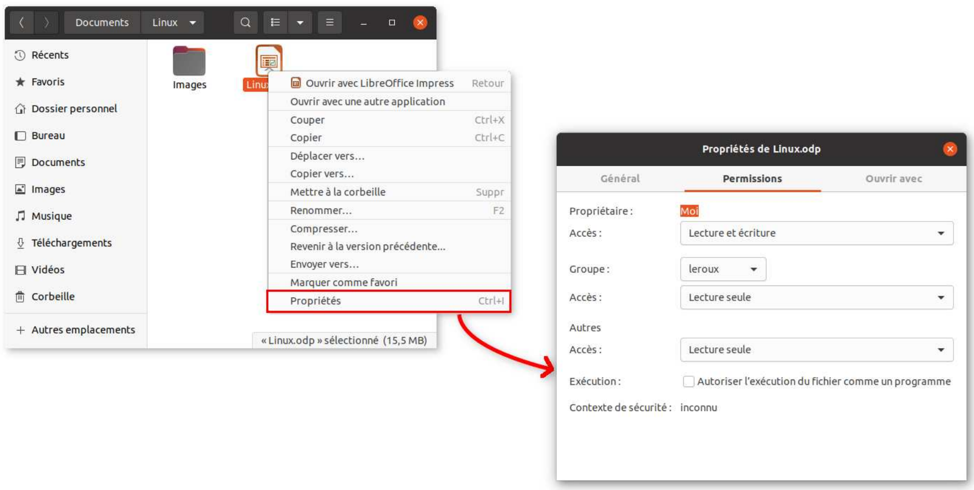
```
user@localhost:~/Desktop$ chmod 644 MyFile
```

The command will grant permission to read and modify the file to the owner, thus removing the permission to execute it. It will grant permission to read the file to the members of the group the owner belongs too, thus removing the permission to modify the file. And it will not modify the permissions granted to all the other users of the computer.

4.2.4.2 Adjusting file permissions: in graphic mode

The procedure is illustrated in figure [Fig. 4.6]:

- Using the keyboard: press Ctrl + i
- Using the mouse:
  1. Use the right click contextual menu on the file or directory to check.
  2. Click on "Properties".
  3. Select the "Permissions" tab to check and adjust file or fold permissions.



Propriétaire : Moi
Accès : Lecture et écriture
Groupe : leroux
Accès : Lecture seule
Autres
Accès : Lecture seule
Exécution : ☐ Autoriser l'exécution du fichier comme un programme
Contexte de sécurité : inconnu

Figure 4.6: Adjusting file permissions in graphic mode.

4.2.5 Environment variables

Adapted from the Wikipedia web page

4.2.5.1 Introduction

Environment variables are a set of dynamic named values that can affect the way running processes will behave on a computer. They can be said in some sense to create the operating environment in which a process runs.
In a Linux system each process has its own private set of environment variables. By default, when a process is created it inherits a duplicate environment of its parent process, except for explicit changes made by the parent when it creates the child. From shells such as bash, you can change environment variables for a particular command invocation by indirectly invoking it via `env` or using the `ENVIRONMENT_VARIABLE=VALUE` command notation (see section [Sec. 8.3].
Examples of environment variables include:

- **PATH**: lists directories the shell searches, for the commands the user may type without having to provide the full path.
- **HOME**: indicates where a user's home directory is located in the file system.
- **PWD**: indicates the working directory.
- **TERM**: specifies the type of computer terminal or terminal emulator being used.
- **SHELL**: specifies the type of command interpreter being used (BASH, TCSH ...).
- **PS1**: specifies how the prompt is displayed in the Bourne shell and variants.

Shell scripts and batch files use environment variables to communicate data and preferences to child processes. They can also be used to store temporary values for reference later in the script.
In Unix/Linux, an environment variable that is changed in a script or compiled program will only affect that process and possibly child processes. The parent process and any unrelated processes will not be affected.
In Unix, the environment variables are normally initialized during system start-up by the system init scripts, and hence inherited by all other processes in the system. Users can modify them in the profile script for the shell they are using.
The variables can be used both in scripts and on the command line. They are usually referenced by putting special symbols in front of or around the variable name. For instance, to display the program search path, in most scripting environments, the user has to type:

```
user@localhost:~$ echo $PATH
```

The **env** command (see section [Sec. 8.3]), displays all environment variables and their values.

4.2.5.2 Working principles of environment variables

A few simple principles govern how environment variables achieve their effect.

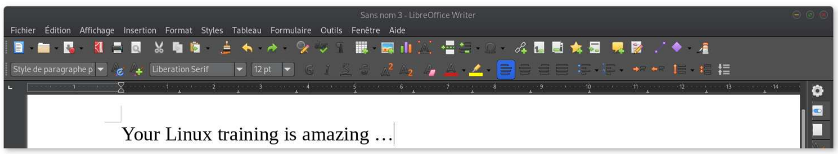
- **Local to process**
  Environment variables are local to the process in which they were set. That means if we open two terminal windows (Two different processes running shell) and change value of environment variable in one window, that change will not be seen by other window.
- **Inheritance**
  When Parent process creates a child process, the child process inherits all the environment variable and their values which parent process had.
- **Case sensitive**
  The names of environment variables are case sensitive.
- **Persistence**
  Environment variables persistence can be session-wide or system-wide.

4.3 Linux tips and tricks

4.3.1 The mouse copy/paste

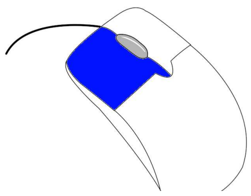
There is a very nice copy/paste tool available when using the mouse in Linux, and that is a standard feature for all Linux:

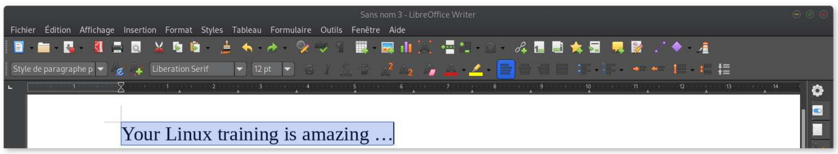
1. If you are reading/editing some text, anywhere, and in any software:


Sans nom 3 - LibreOffice Writer
Fichier Édition Affichage Insertion Format Styles Tableau Formulaire Outils Fenêtre Aide
Style de paragraphe p
Liberation Serif
12 pt

Your Linux training is amazing …

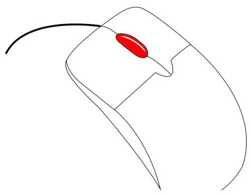
2. Then if you select this text using the mouse left button:

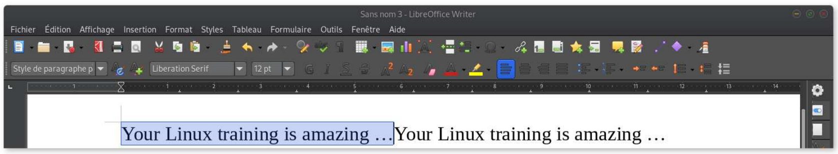



Sans nom 3 - LibreOffice Writer
Fichier Édition Affichage Insertion Format Styles Tableau Formulaire Outils Fenêtre Aide
Style de paragraphe p
Liberation Serif
12 pt

Your Linux training is amazing …

3. This text is immediately copied in a memory buffer (in the memory of your computer) and you can very easily past it back pressing the mouse middle button (scroll or else):



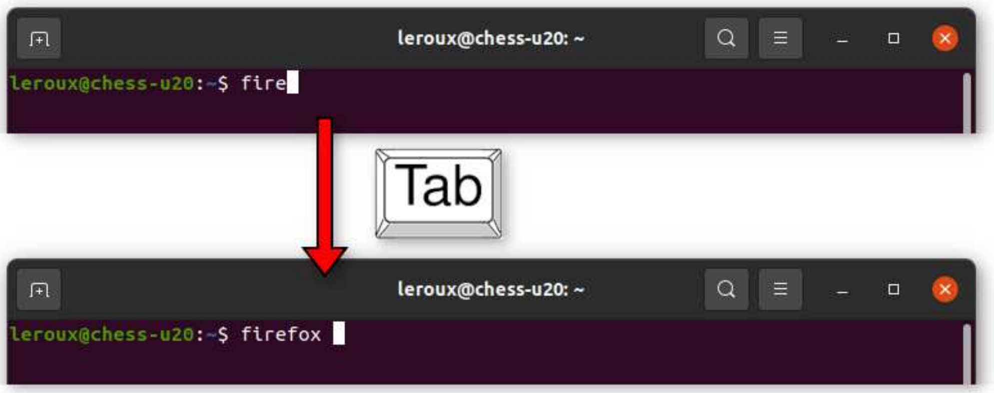

Sans nom 3 - LibreOffice Writer
Fichier Édition Affichage Insertion Format Styles Tableau Formulaire Outils Fenêtre Aide
Style de paragraphe p
Liberation Serif
12 pt

Your Linux training is amazing …Your Linux training is amazing …

4.3.2 Auto-completion

The auto-completion, or completion, is a terminal tips. When you enter commands, if the name of the command is partially entered (written in the terminal), then if your press the $\boxed{\leftrightarrows}$ / $\boxed{\text{tab}}$ on the keyboard then Linux will try to complete the instruction/line for you:

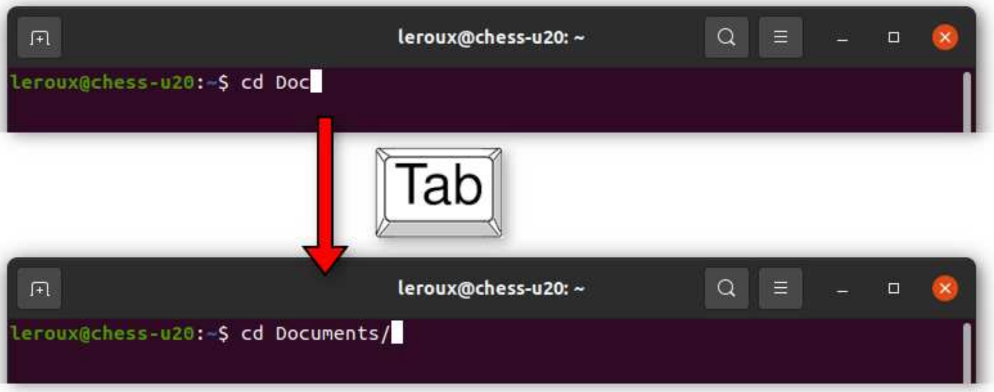
- If there is only one possibility the system will complete the command / line:
  - For a command:



```
leroux@chess-u20:~$ fire
```

```
leroux@chess-u20:~$ firefox 
```

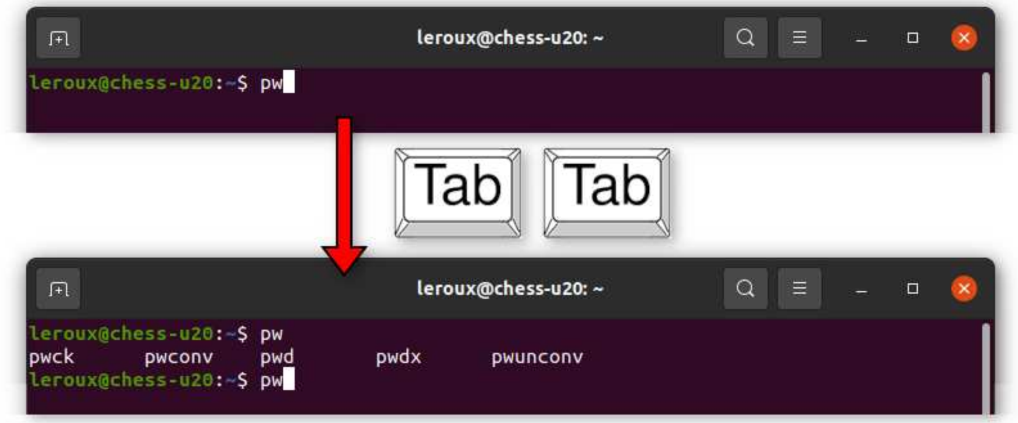
  - For a file or a path:



```
leroux@chess-u20:~$ cd Doc
```

```
leroux@chess-u20:~$ cd Documents/
```

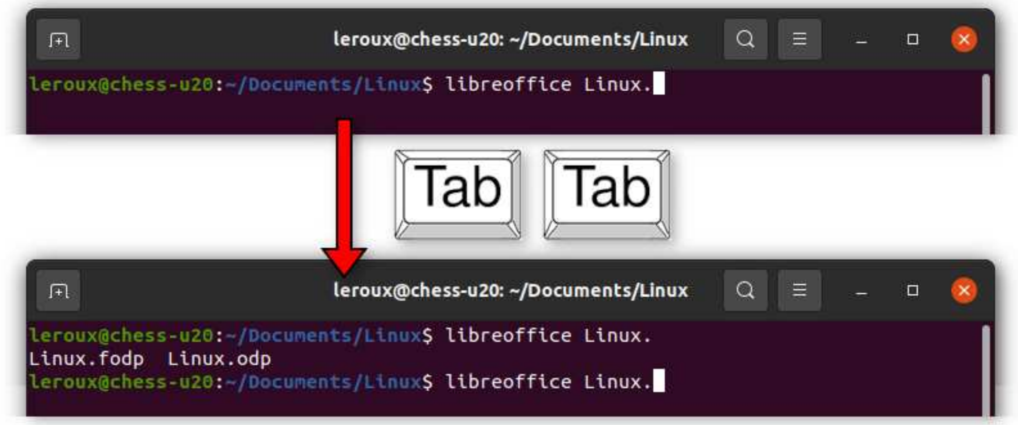
- Otherwise, if there are many options, press ↹ / tab a second time the system will present the options available to complete the command / line:
  - For a command:



```
leroux@chess-u20:~$ pw
```

```
leroux@chess-u20:~$ pw
pwck        pwconv      pwd         pwdx        pwunconv
leroux@chess-u20:~$ pw
```

  - For a file or a path:


leroux@chess-u20: ~/Documents/Linux
Tab
Tab
leroux@chess-u20: ~/Documents/Linux

```
leroux@chess-u20:~/Documents/Linux$ libreoffice Linux.
```

```
leroux@chess-u20:~/Documents/Linux$ libreoffice Linux.
Linux.fodp  Linux.odp
leroux@chess-u20:~/Documents/Linux$ libreoffice Linux.
```

4.3.3 Command line tips

You find bellow few tips that come handy when using the command line:

Change directory to the user home directory:	`cd`
Change to the previous directory where you are coming from:	`cd -`
Change to the upper directory in the system tree:	`cd ..`
Move at the beginning of the line:	Ctrl + a
Move at the end of the line:	Ctrl + e

Table 4.2: Tips and tricks for the command line.

Ubuntu

Figure 5.1: The official logos of the Ubuntu GNU/Linux distribution.

Ubuntu is a Linux distribution based on Debian, composed mostly of free and open-source software, and developed by Canonical.
Every 6 months 4 editions of Ubuntu are released:

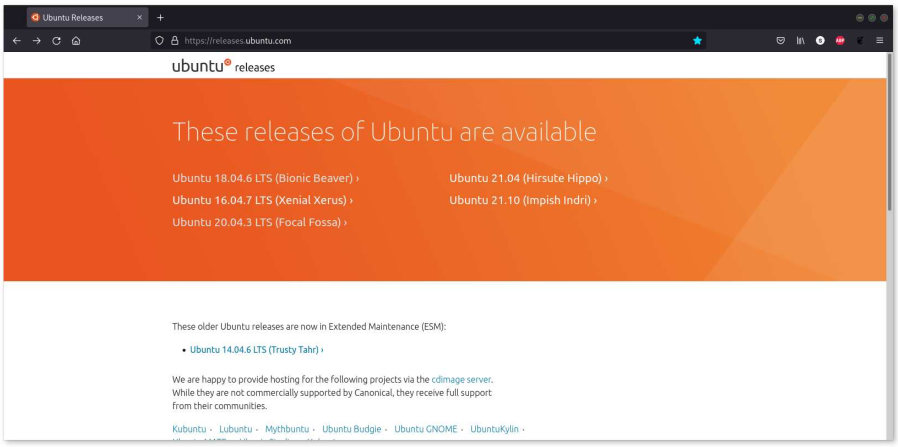
- Desktop
- Server
- Internet Of Things (IOT) devices and robots
- Cloud

With Long-Term Support (LTS, 5 years support) releases every 2 years.
Each Ubuntu release has a version number that consists of the year and month number of the release. For example, the first release, Ubuntu 4.10, was released on 20 October 2004.
The most recent long-term support release is Ubuntu "Noble Numbat" 24.04 LTS, which is supported until 2025 under public support and until 2030 as a paid option for companies and free for individuals who register. The latest standard release is Ubuntu 23.04 "Lunar Lobster", which is supported for nine months.

Ubuntu is named after the Nguni Bantu, philosophy of ubuntu, which Canonical indicates means "humanity to others" with a connotation of "I am what I am because of who we all are".

5.1 Downloading and installing Ubuntu

All active releases of Ubuntu can be found on the website https://releases.ubuntu.com/:


Ubuntu Releases
https://releases.ubuntu.com

ubuntu releases

These releases of Ubuntu are available

Ubuntu 18.04.6 LTS (Bionic Beaver) ›
Ubuntu 16.04.7 LTS (Xenial Xerus) ›
Ubuntu 20.04.3 LTS (Focal Fossa) ›
Ubuntu 21.04 (Hirsute Hippo) ›
Ubuntu 21.10 (Impish Indri) ›

These older Ubuntu releases are now in Extended Maintenance (ESM):

- Ubuntu 14.04.6 LTS (Trusty Tahr) ›

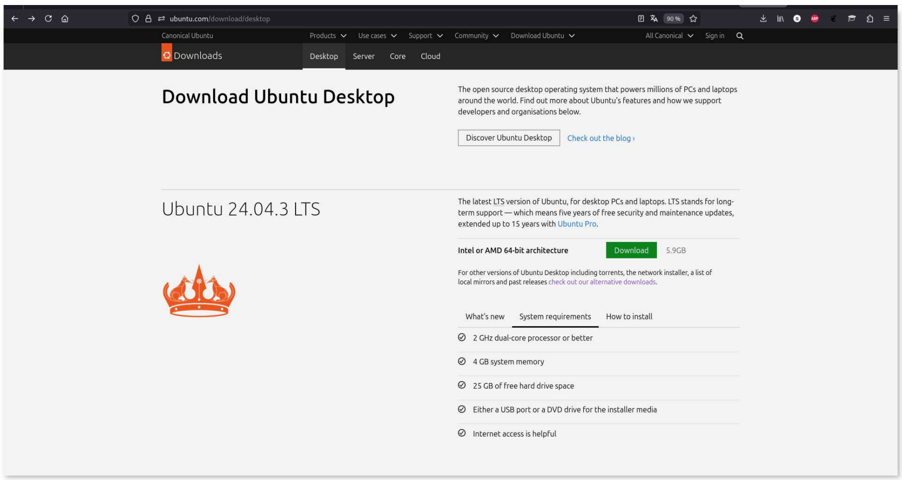
We are happy to provide hosting for the following projects via the cdimage server.
While they are not commercially supported by Canonical, they receive full support
from their communities.

Kubuntu · Lubuntu · Mythbuntu · Ubuntu Budgie · Ubuntu GNOME · UbuntuKylin ·

Figure 5.2: Ubuntu releases website.

To begin with Linux I strongly recommend that you try, and/or install, the most recent LTS release of Ubuntu, as I am writing this tutorial this would be Ubuntu "Noble Numbat" 24.04 LTS.
Start with downloading the ISO image, either following the link from the release website (see figure [Fig. 5.2], or go directly to the website dedicated to the latest LTS release, website:

https://ubuntu.com/download/desktop


ubuntu.com/download/desktop

Canonical Ubuntu · Products · Use cases · Support · Community · Download Ubuntu · All Canonical · Sign in

Downloads · Desktop · Server · Core · Cloud

Download Ubuntu Desktop

The open source desktop operating system that powers millions of PCs and laptops around the world. Find out more about Ubuntu's features and how we support developers and organisations below.

Discover Ubuntu Desktop · Check out the blog ›

Ubuntu 24.04.3 LTS

The latest LTS version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support — which means five years of free security and maintenance updates, extended up to 15 years with Ubuntu Pro.

Intel or AMD 64-bit architecture · Download · 5.9GB

For other versions of Ubuntu Desktop including torrents, the network installer, a list of local mirrors and past releases check out our alternative downloads.

What's new · System requirements · How to install

- 2 GHz dual-core processor or better
- 4 GB system memory
- 25 GB of free hard drive space
- Either a USB port or a DVD drive for the installer media
- Internet access is helpful

Figure 5.3: Ubuntu most recent LTS desktop release website.

The direct link to download Ubuntu "Noble Numbat" 24.04 LTS ISO image:

https://releases.ubuntu.com/noble/ubuntu-24.04.3-desktop-amd64.iso

5.1.1 After the download

After downloading Ubuntu "Noble Numbat" 24.04 LTS, 2 choices are available:

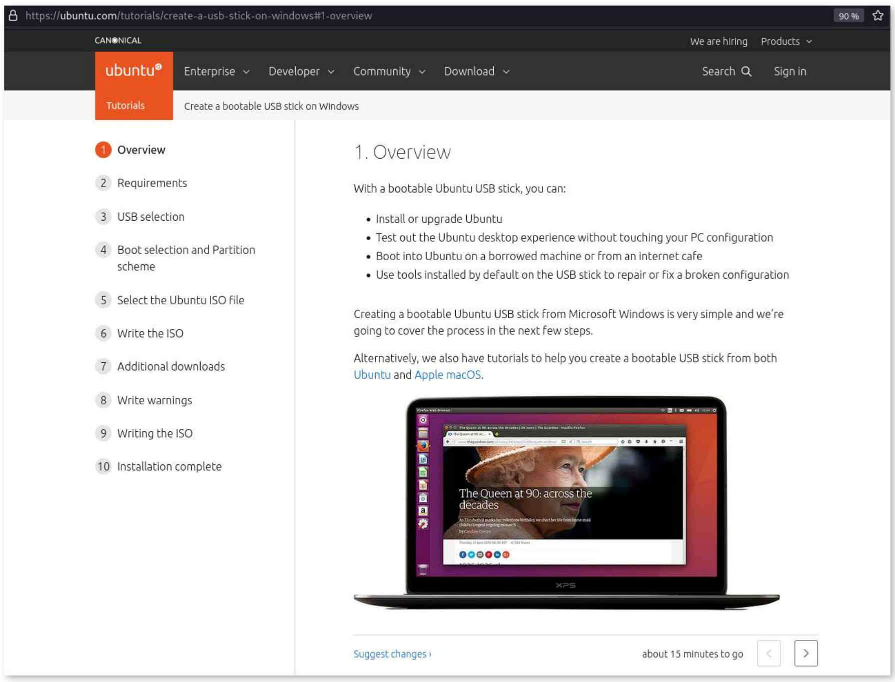
1. Simply trying it without installing it: this is the easiest way to discover Ubuntu and Linux.
2. Installing it on your hard drive.

In both cases you will need to prepare a USB key, and you will have to boot from that USB key at start-up, ie. when starting your computer ask it to start using the key.

To prepare the USB key simply follow the tutorial(s) at:

- For MS Windows: https://ubuntu.com/tutorials/create-a-usb-stick-on-windows
- For MacOSX: https://ubuntu.com/tutorials/create-a-usb-stick-on-macos

There a complete assistant that will guide you trough the few steps required to do it:



1 Overview
2 Requirements
3 USB selection
4 Boot selection and Partition scheme
5 Select the Ubuntu ISO file
6 Write the ISO
7 Additional downloads
8 Write warnings
9 Writing the ISO
10 Installation complete

1. Overview

With a bootable Ubuntu USB stick, you can:

- Install or upgrade Ubuntu
- Test out the Ubuntu desktop experience without touching your PC configuration
- Boot into Ubuntu on a borrowed machine or from an internet cafe
- Use tools installed by default on the USB stick to repair or fix a broken configuration

Creating a bootable Ubuntu USB stick from Microsoft Windows is very simple and we're going to cover the process in the next few steps.

Alternatively, we also have tutorials to help you create a bootable USB stick from both Ubuntu and Apple macOS.

Figure 5.4: Preparing a USB key to test or install Ubuntu.

In the following I will consider that you prepared the USB key successfully and that your are ready to use it.

5.1.1.1 Testing Ubuntu

To do that simply start your computer with the USB key plugged in, and boot on the USB key. If your computer doesn't automatically boot from USB, try holding F12 when your computer first starts. With most machines, this will allow you to select the USB device from a system-specific boot menu.
However you might need to enter the BIOS menu to modify the boot sequence, then you need to ask your computer to start by booting on the USB key.
When this is done simply jump to section [Sec. 5.2] of this manual.

5.1.1.2 Installing Ubuntu

In this manual I will not cover the installation process, simply because nowadays it is really simple to install Linux, and, because many step by step tutorials already exist and are certainly much better than what I could think of.
To install Ubuntu on your hard drive you can follow the tutorial at:

https://ubuntu.com/tutorials/install-ubuntu-desktop

Few personal recommendations:

1. Make a copy of all your data before installing Linux
2. Multiple boot, ie. having multiple OS on the same computer, is naturally possible:

   https://www.tecmint.com/install-ubuntu-alongside-with-windows-dual-boot/

   Remember this after the installation:

   - Linux will be able to see, mount, use the MS Windows partition(s).
   - MS Windows will not be able to see, mount, use the Linux partition(s).

Overall if you need help, ask for it !

5.2 Using Ubuntu

In the next section I will use examples from Ubuntu "Noble Numbat" 24.04 LTS and from Ubuntu "Jammy Jellyfish" 22.04 LTS. The first version of this manual was based on Ubuntu "Focal Fossa" 20.04 LTS and only few visuals details changed compared to Ubuntu "Noble Numbat" 24.04 LTS, the examples are perfectly valid. I will notify the reader of the differences if required.

5.2.1 The first steps

Right after the installation is over, or after the boot on the USB key if your are simply trying Ubuntu, you will face the following screen:

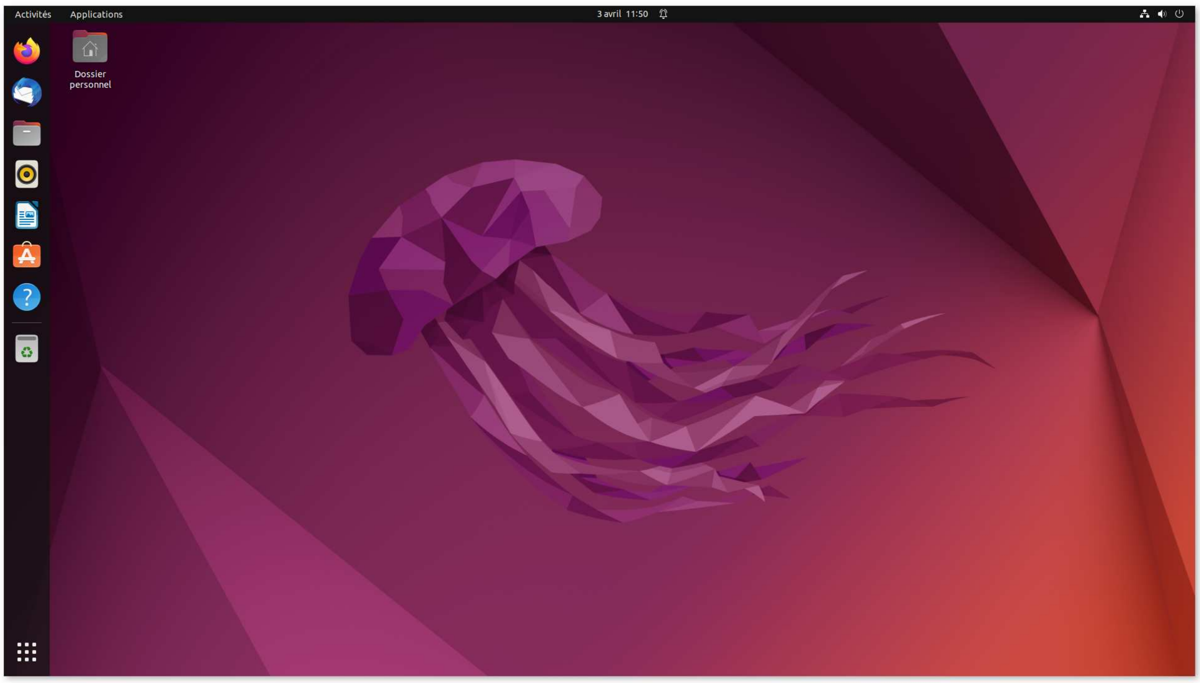


Figure 5.5: Ubuntu "Jammy Jellyfish" 22.04 LTS, with the GNOME desktop.

This is the default GNOME desktop configured for Ubuntu.
If this is your first step with Linux, just so you this is the Ubuntu look, but there is as many looks, not only because GNOME can be customized, but also because there are many different desktop managers in the Linux / Open source world.
For example Linux can also looks like the examples in figure [Fig. 5.6].

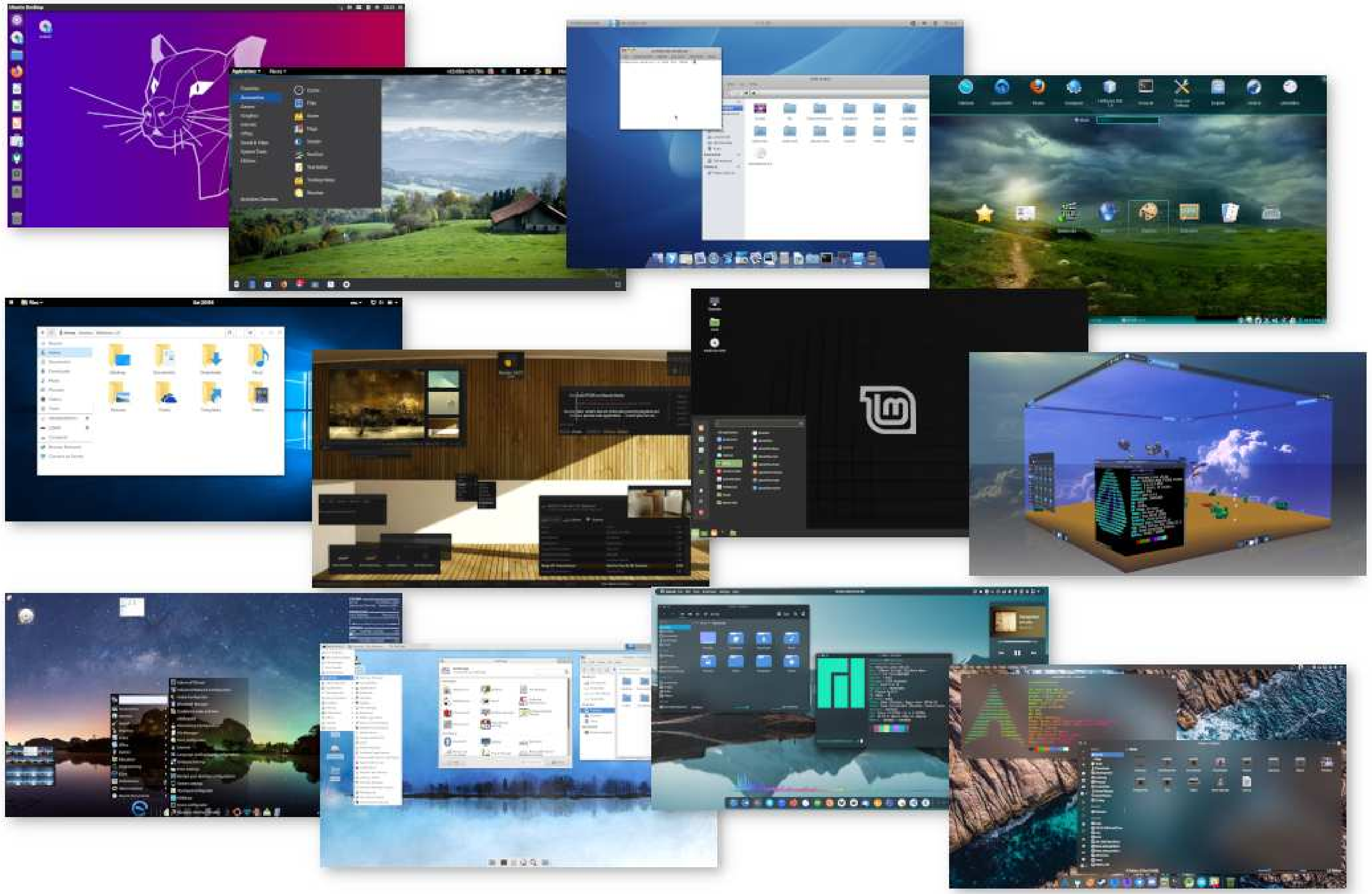


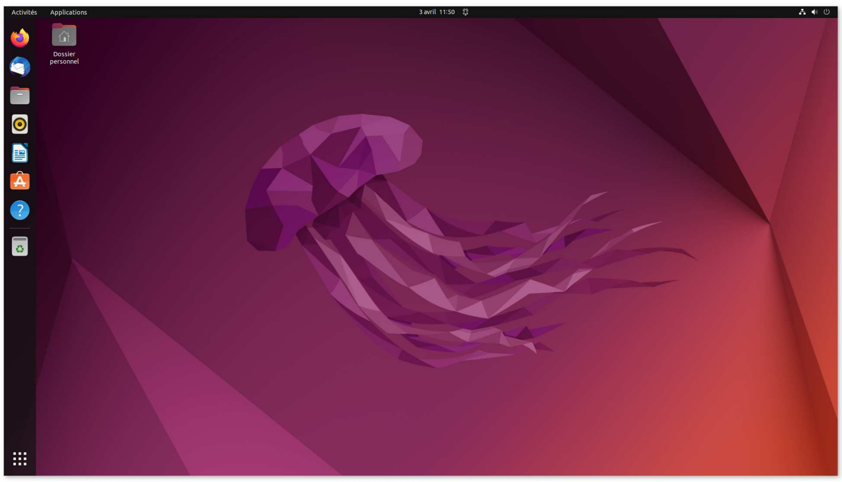
Figure 5.6: Few examples of Linux desktop managers.

Table [Tab. 5.1] presents a list of some of the most popular desktop managers in the Linux world, but that list is not exhaustive, remember that you are now in the open source world and that you might stumble on something different.

GNOME	https://www.gnome.org/
KDE	https://kde.org/
Cinnamon	https://projects.linuxmint.com/cinnamon/
Xfce	https://www.xfce.org/
MATE	https://mate-desktop.org/
LXQt	ttps://lxqt-project.org/
Enlightenment	https://www.enlightenment.org/
Deepin	https://www.deeping.org/
Pantheon	https://elementary.io/

Table 5.1: Some of the most popular Linux desktop managers.

In Ubuntu the default desktop manager is called GNOME, not only it is really intuitive and easy to use, but it has also been customized by the Ubuntu team so that everything is easier for beginners. Please keep in mind that most of the examples hereafter are suitable for the combination Ubuntu "Noble Numbat" 24.04 LTS + GNOME desktop" and that few aspects might change for another Linux distro, and even more for another desktop manager.



5.2.2 The GNOME desktop in Ubuntu

Immediately after installing Ubuntu, or after starting the computer on the USB key to try it, you will be face with the screen in figure [Fig. 5.5].
On that screen you will notice a shortcut to your home directory (`/home/user`) on the desktop, as well a shortcut to the trash bin. You will also notice a docker bar on the left side of the screen that presents shortcuts to your favorite programs.
On the Ubuntu/GNOME desktop you will find 2 areas worth of interest for you:

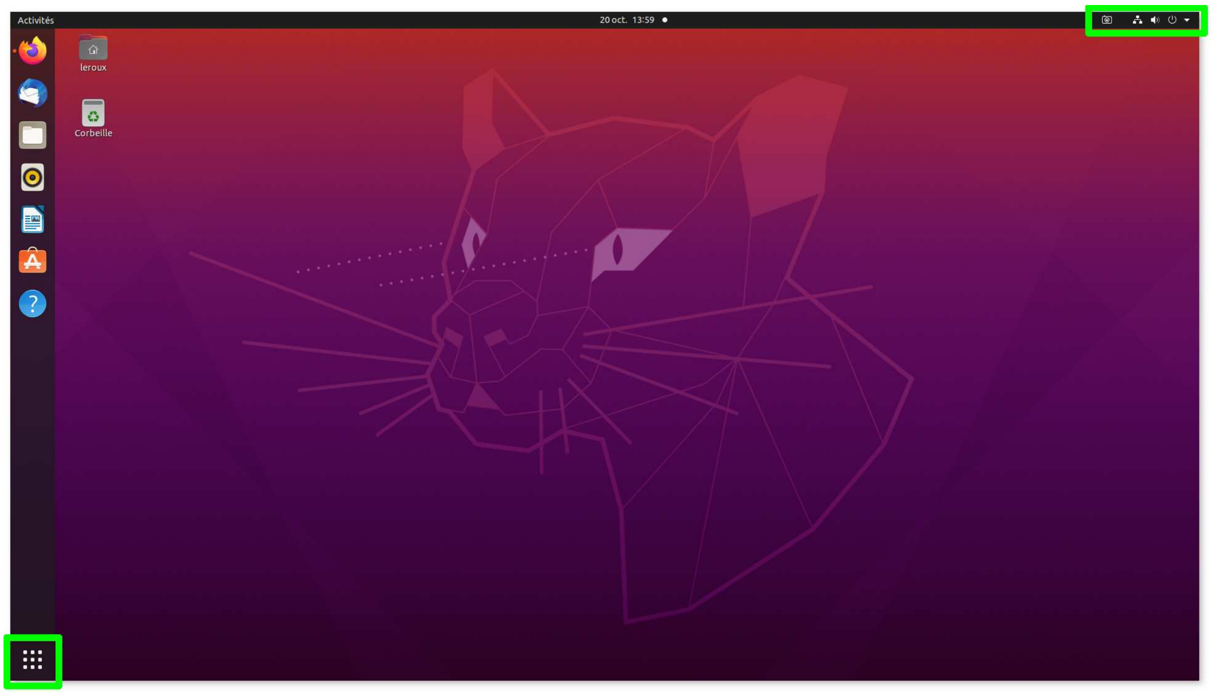


Figure 5.7: First steps with the GNOME desktop in Ubuntu "Focal Fossa" 20.04 LTS, only the background image is different in Ubuntu "Noble Numbat" 24.04 LTS.

- The first in the top right corner of the screen, allows to open the parameters/shutdown menu. From there you can access the Ubuntu control panel and turn off your computer.
- The second in the bottom left corner of the screen, allows to open the application menu. From there you can browse the programs available on your computer.

5.2.2.1 The control panel

As illustrated in figure [Fig. 5.8] if you select "Parameters" on the menu available on the top right corner of the screen, or alternatively the menu available with a right click on the desktop, you can access the Ubuntu control panel:

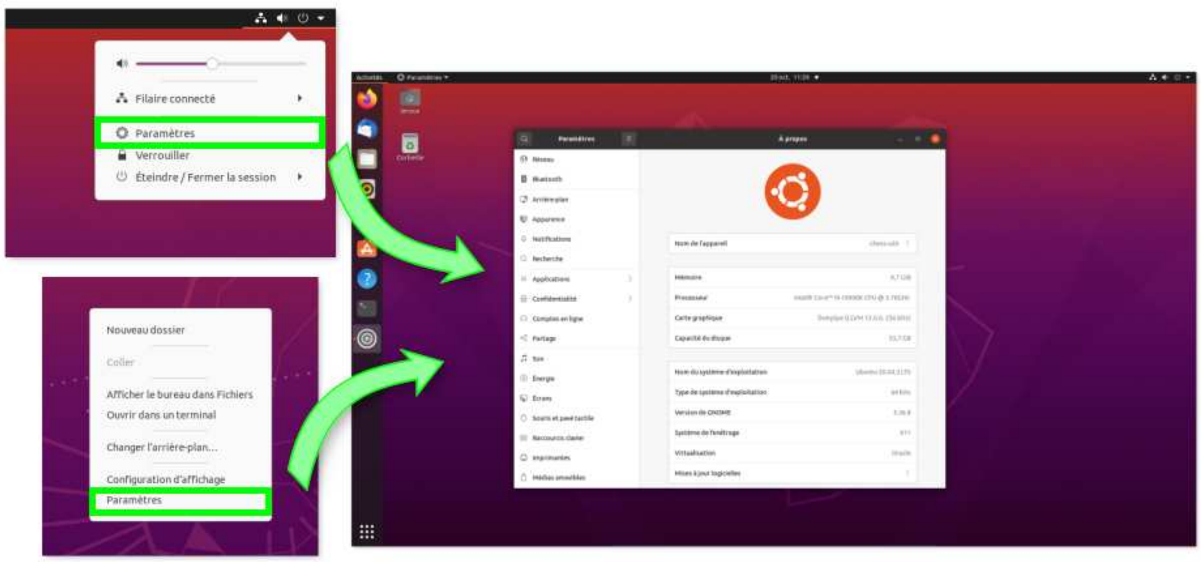


Figure 5.8: Opening the control panel in Ubuntu.

5.2.2.2 The application panel

As illustrated in figure [Fig. 5.9] if you click in the bottom left corner of the GNOME desktop then you will open the applications panel:

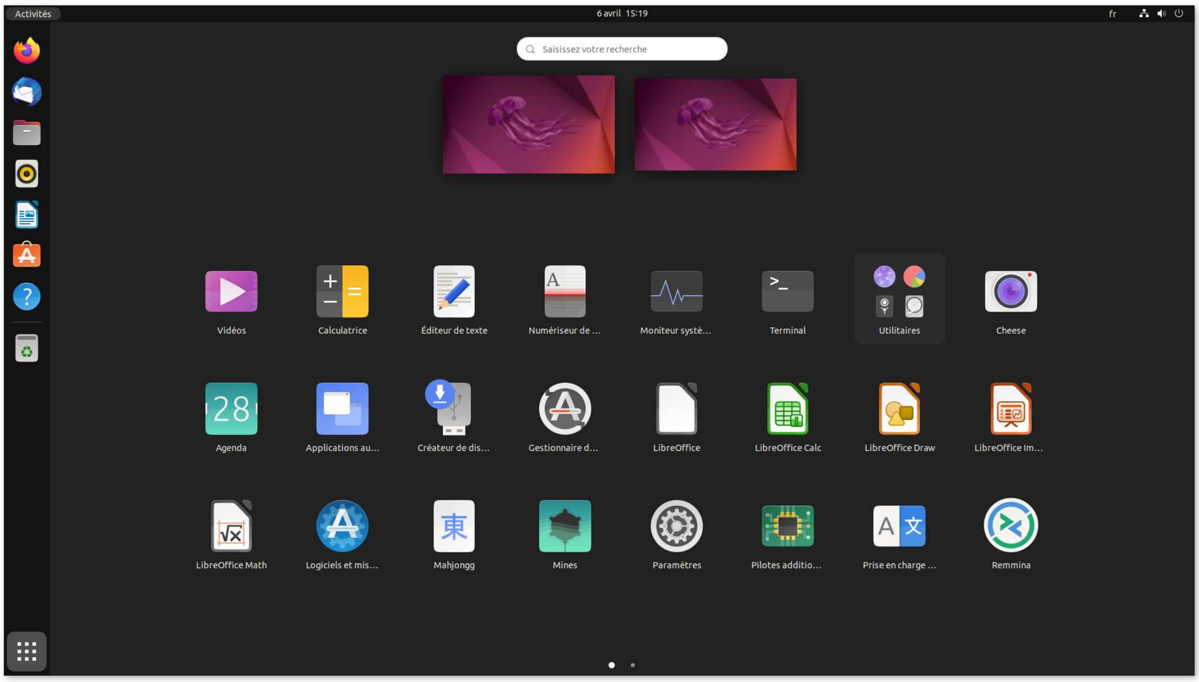


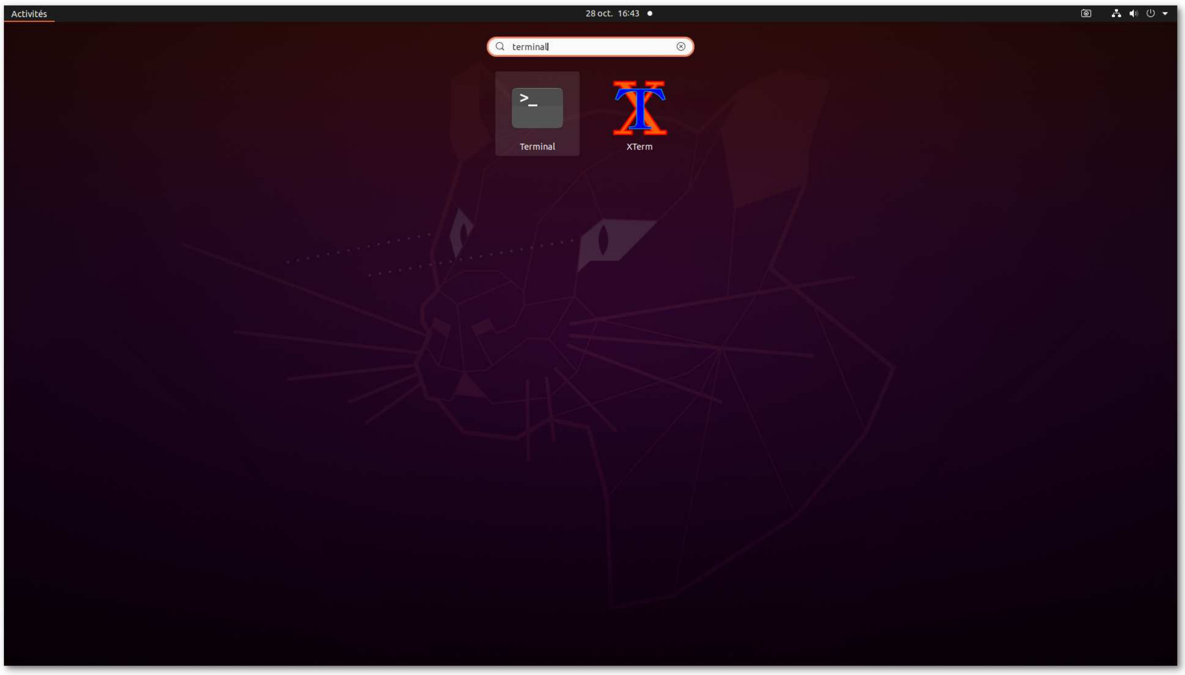
Figure 5.9: Opening the applications panel in Ubuntu "Jammy Jellyfish" 22.04 LTS.

From there you can easily browse immediately available programs, if what you are looking for does not appear immediately you can look it up using the research area.

5.2.2.3 Terminal

To open a terminal you can:

- Use the mouse: open the GNOME application panel and search for "Terminal":



- Use the keyboard short-cut (works for most Linux), press: Ctrl + Alt + t

Up to this point I pretty much covered all basic points to get your ready to try and test Linux and Ubuntu, therefore if you are simply testing Ubuntu, then your are ready to try, test, in short discover the Linux world.

Otherwise if you are in a post-installation setup of your all new Ubuntu "Noble Numbat" 24.04 LTS system, then please read the next pages. I will give you few tips and tricks to help you start the easiest and safest way with Ubuntu.

5.2.3 After Installation

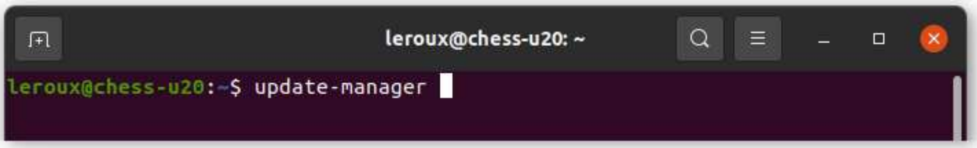
Here are some things I recommend to do immediately after installing Ubuntu:

- Check for updates: see section [Sec. 5.2.3.1]
- Enable partners repositories: see section [Sec. 5.2.3.2]
- Install possibly missing hardware drivers
- Install extra multimedia codecs

5.2.3.1 Check for updates

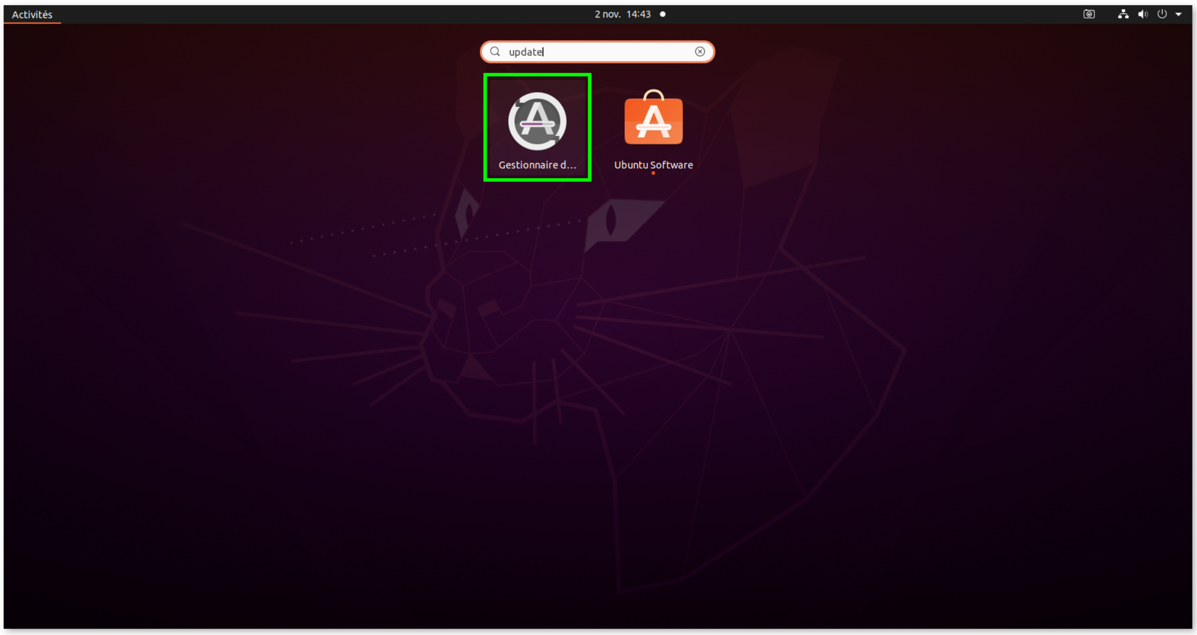
To check for updates, providing that you did not during the installation:

1. Open the update-manager utility:
   - Using the terminal, simply enter the command: `update-manager`

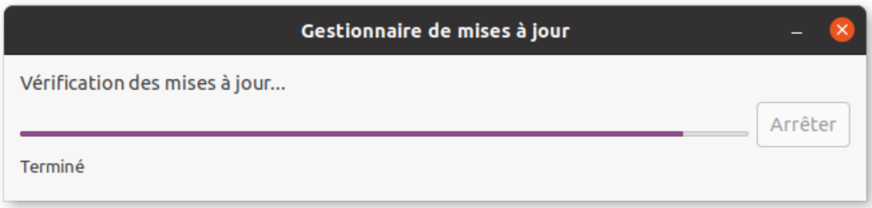


```
leroux@chess-u20:~$ update-manager
```

   - Using the application panel, search for "update(s)":



2. After pressing Enter , or clicking on the appropriate icon, the system will start looking for updates:



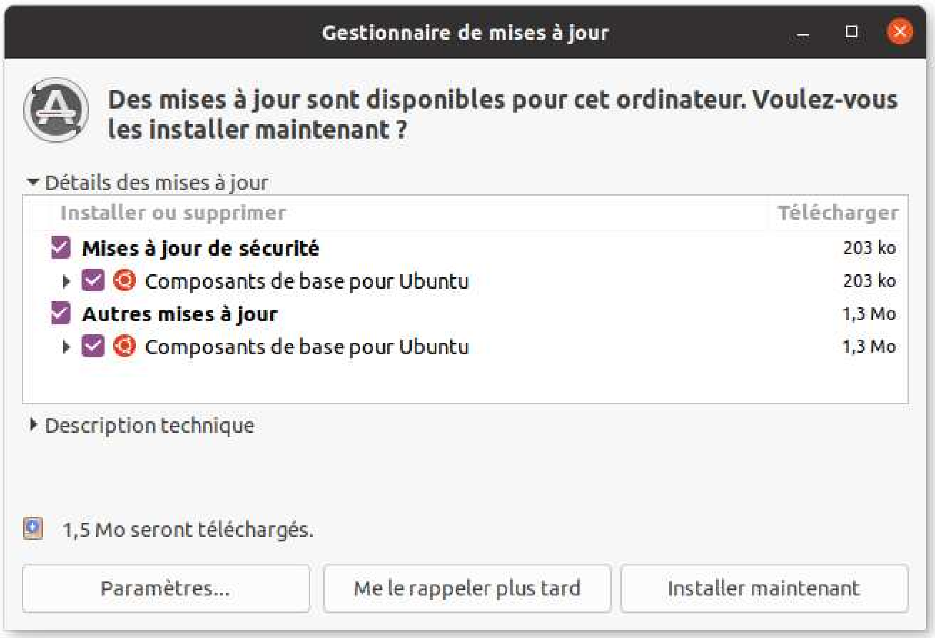
Gestionnaire de mises à jour

Vérification des mises à jour...

Arrêter

Terminé

3. If updates are available then the "**`update manager`**" dialog will appear:



Gestionnaire de mises à jour

Des mises à jour sont disponibles pour cet ordinateur. Voulez-vous les installer maintenant ?

Détails des mises à jour

Installer ou supprimer	Télécharger
✓ Mises à jour de sécurité	203 ko
✓ Composants de base pour Ubuntu	203 ko
✓ Autres mises à jour	1,3 Mo
✓ Composants de base pour Ubuntu	1,3 Mo

Description technique

1,5 Mo seront téléchargés.

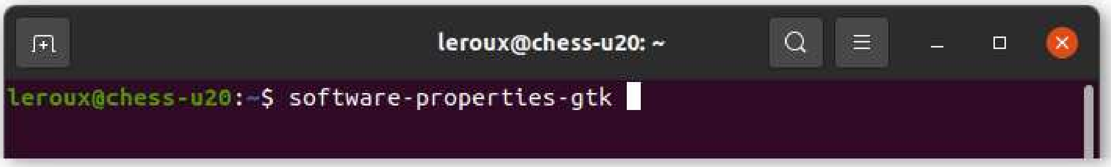
Paramètres... | Me le rappeler plus tard | Installer maintenant

To update the computer simply click on "**`Install now`**" and follow the procedure.

5.2.3.2 Enable partners repositories

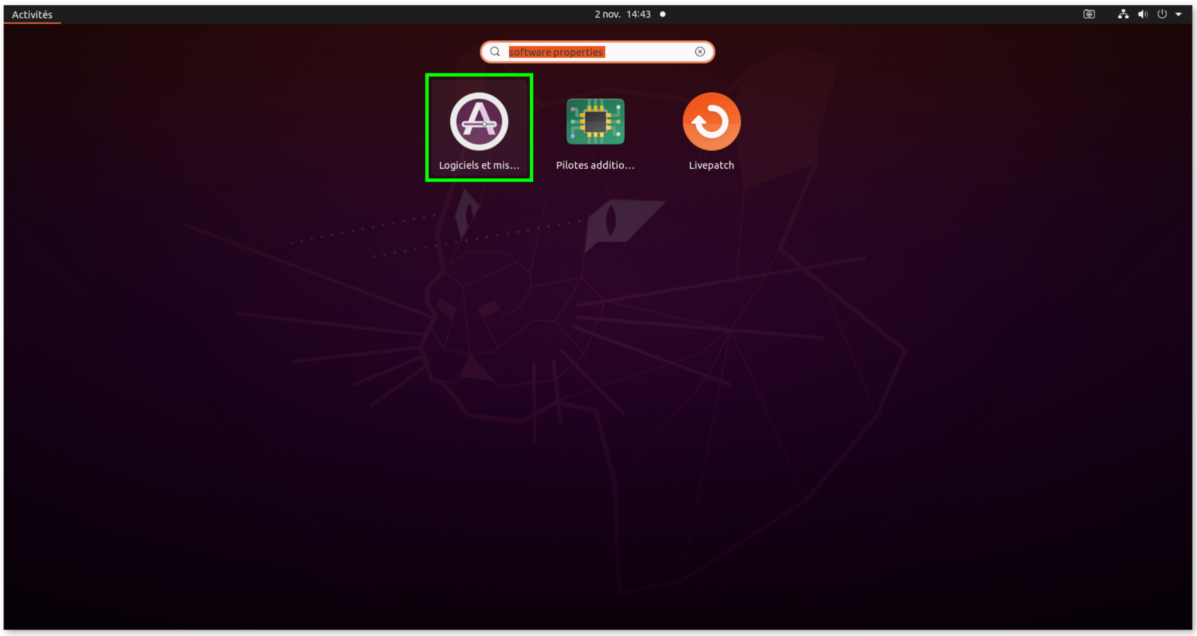
Some third-party software that does not limit distribution is included in Ubuntu's multiverse component. The package "ubuntu-restricted-extras" additionally contains software that may be legally restricted, including support for MP3 and DVD playback, Microsoft TrueType core fonts, Sun's Java runtime environment, Adobe's Flash Player plugin, many common audio/video codecs, and unrar, an unarchiver for files compressed in the RAR file format.

1. Open the update configuration utility:
   - Using the terminal, simply enter the command: `software-properties-gtk`

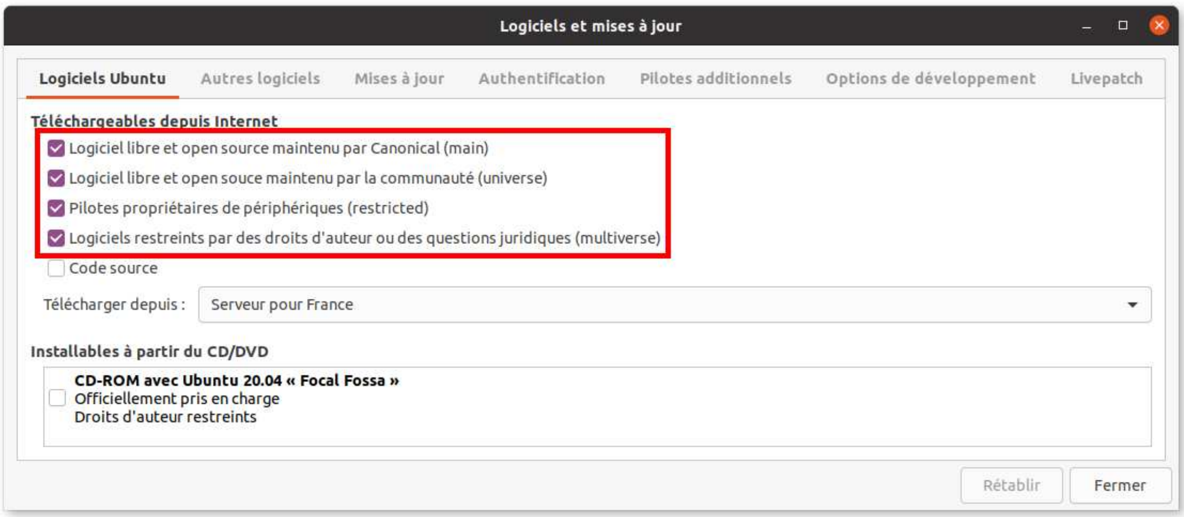


```
leroux@chess-u20:~$ software-properties-gtk
```

   - Using the application panel, search for "software properties":



2. Then the "`software properties`" dialog will appear:



Téléchargeables depuis Internet

- [✓] Logiciel libre et open source maintenu par Canonical (main)
- [✓] Logiciel libre et open souce maintenu par la communauté (universe)
- [✓] Pilotes propriétaires de périphériques (restricted)
- [✓] Logiciels restreints par des droits d'auteur ou des questions juridiques (multiverse)
- [ ] Code source

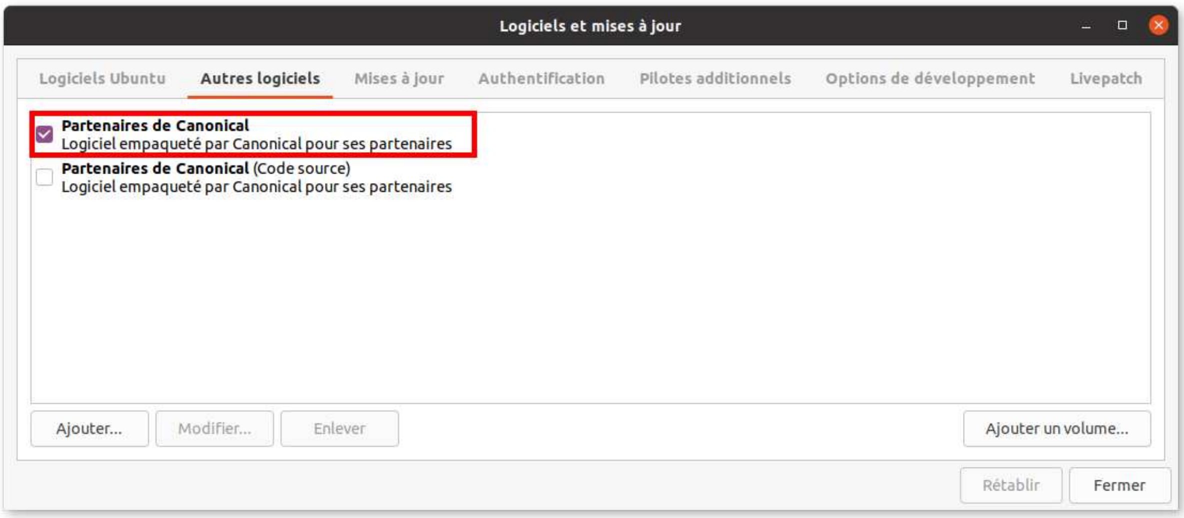
Télécharger depuis : Serveur pour France

Installables à partir du CD/DVD

- [ ] **CD-ROM avec Ubuntu 20.04 « Focal Fossa »**
  Officiellement pris en charge
  Droits d'auteur restreints

On the "`Ubuntu software`" tab, you need to make sure that all options are checked as illustrated.

3. After that switch to the "`Other software`" tab:



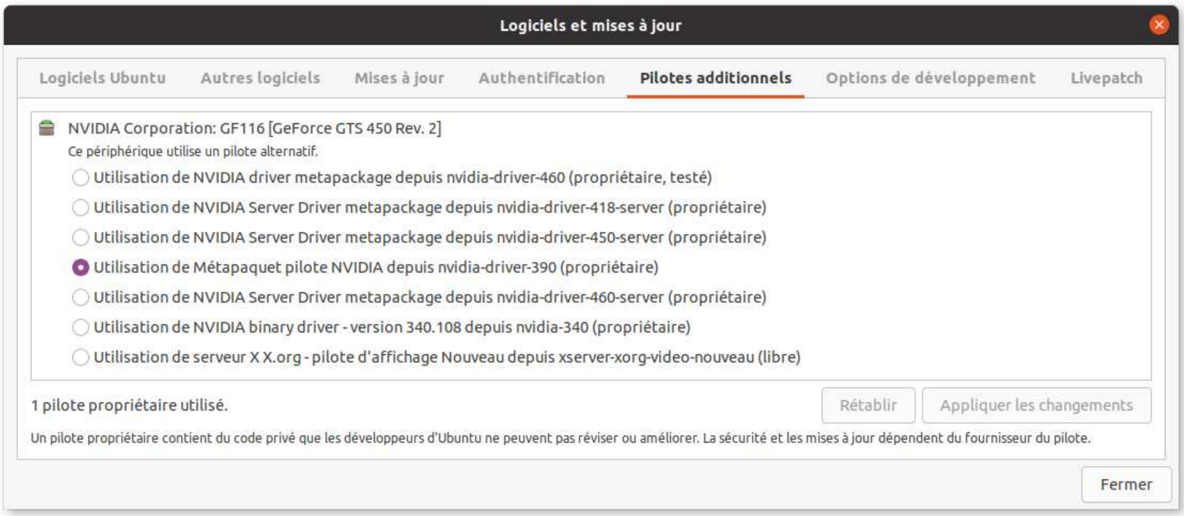
- [✓] **Partenaires de Canonical**
  Logiciel empaqueté par Canonical pour ses partenaires
- [ ] **Partenaires de Canonical** (Code source)
  Logiciel empaqueté par Canonical pour ses partenaires

Then make sure that the "`Canonical partners`" option is checked.

After that your Ubuntu will have way more locations to search packages in, more on-line repositories, when looking to install programs.

5.2.3.3 Install possibly missing hardware drivers

On Linux open source drivers are installed during the installation process, and not proprietary drivers made by the manufacturer of your video card (or other devices for that matter). However manufacturer's drivers are usually faster to render 3D content than open source drivers, to check if more suitable drivers are available for your hardware use the "**`software properties`**" dialog, and open the "**`Additional drivers`**" tab:



Logiciels et mises à jour

NVIDIA Corporation: GF116 [GeForce GTS 450 Rev. 2]
Ce périphérique utilise un pilote alternatif.

- Utilisation de NVIDIA driver metapackage depuis nvidia-driver-460 (propriétaire, testé)
- Utilisation de NVIDIA Server Driver metapackage depuis nvidia-driver-418-server (propriétaire)
- Utilisation de NVIDIA Server Driver metapackage depuis nvidia-driver-450-server (propriétaire)
- [✓ Utilisation de Métapaquet pilote NVIDIA depuis nvidia-driver-390 (propriétaire)]
- Utilisation de NVIDIA Server Driver metapackage depuis nvidia-driver-460-server (propriétaire)
- Utilisation de NVIDIA binary driver - version 340.108 depuis nvidia-340 (propriétaire)
- Utilisation de serveur X X.org - pilote d'affichage Nouveau depuis xserver-xorg-video-nouveau (libre)

1 pilote propriétaire utilisé.

Un pilote propriétaire contient du code privé que les développeurs d'Ubuntu ne peuvent pas réviser ou améliorer. La sécurité et les mises à jour dépendent du fournisseur du pilote.

Figure 5.10: The "**`Additional drivers`**" tab of "**`software properties`**".

If some drivers are available for your video card, or other peripherals, they will likely all be listed here, and you will have to pick to one(s) that you think is/are appropriate.
For graphics card proprietary drivers are packaged by generation, roughly depending on how old is your card. As illustrated in figure [Fig. 5.10] the graphic card on my computer is a nvidia GPU, and Ubuntu offers several different drivers. To know which one to install check the webpage of the manufacturer of your GPU:

- NVIDIA: https://www.nvidia.com/en-us/drivers/unix/
- AMD/(former ATI): the company provides drivers built specifically for Ubuntu, as well as for other Linux, so you might only have a single option to choose from, if not you can simply search the appropriate package on the following webpage:

  https://www.amd.com/en/support

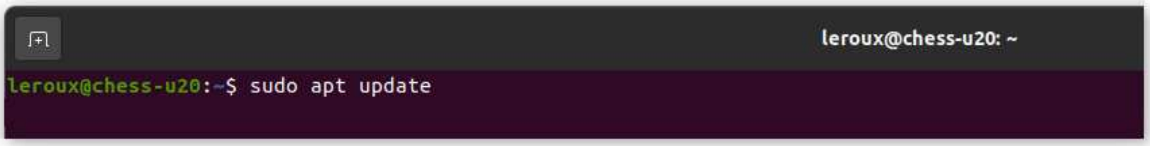
  Also the website provides help for the installation process here:

  https://amdgpu-install.readthedocs.io/en/latest/

5.2.3.4 Install extra multimedia codecs

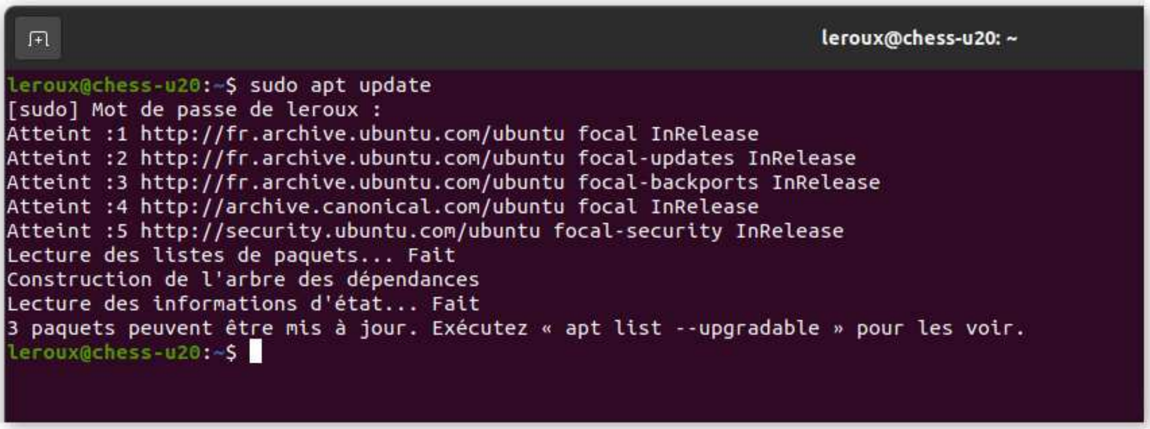
Multimedia codecs are essential for playing audio and video files. By default, only open source multimedia codecs are installed on Ubuntu 22.04 LTS. To install all other codecs and tools you will need to do the following (I will use here some command that will be discussed in section [Sec. 5.2.4]:

1. Open the terminal (see section [Sec. 5.2.2.3]
2. Update the repository cache (just to ensure what was done in section [Sec. [5.2.3.2] is active):
   - Enter the command: `sudo apt update`



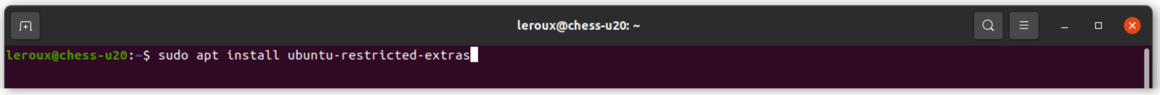
```
leroux@chess-u20:~$ sudo apt update
```

   - Press Enter , enter your password and follow the procedure, you should end-up with something like:



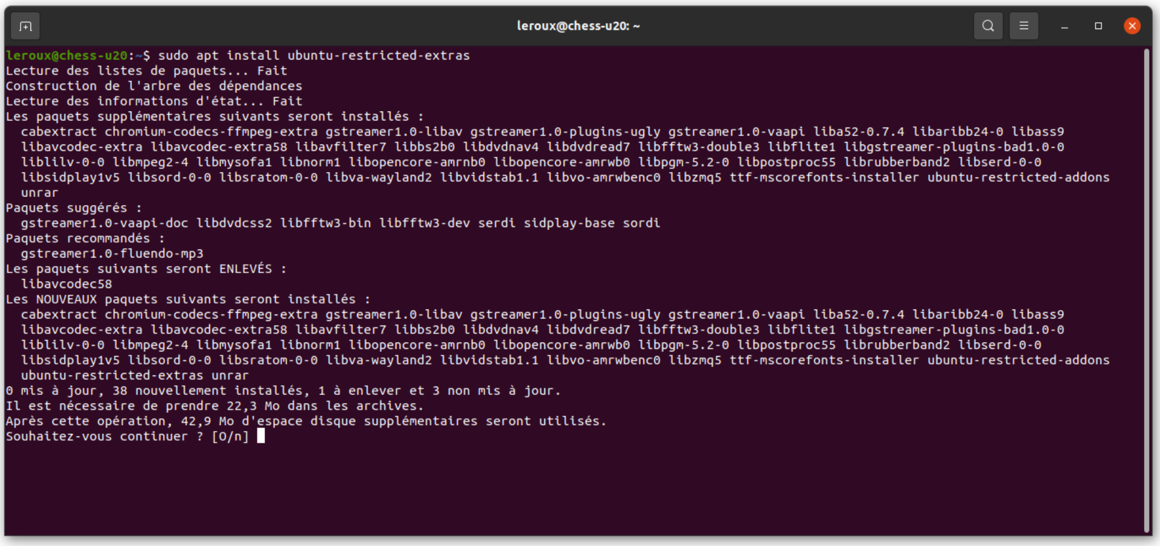
```
leroux@chess-u20:~$ sudo apt update
[sudo] Mot de passe de leroux :
Atteint :1 http://fr.archive.ubuntu.com/ubuntu focal InRelease
Atteint :2 http://fr.archive.ubuntu.com/ubuntu focal-updates InRelease
Atteint :3 http://fr.archive.ubuntu.com/ubuntu focal-backports InRelease
Atteint :4 http://archive.canonical.com/ubuntu focal InRelease
Atteint :5 http://security.ubuntu.com/ubuntu focal-security InRelease
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
3 paquets peuvent être mis à jour. Exécutez « apt list --upgradable » pour les voir.
leroux@chess-u20:~$
```

3. Now to install the codecs:
   - Enter the command: `sudo apt install ubuntu-restricted-extras`



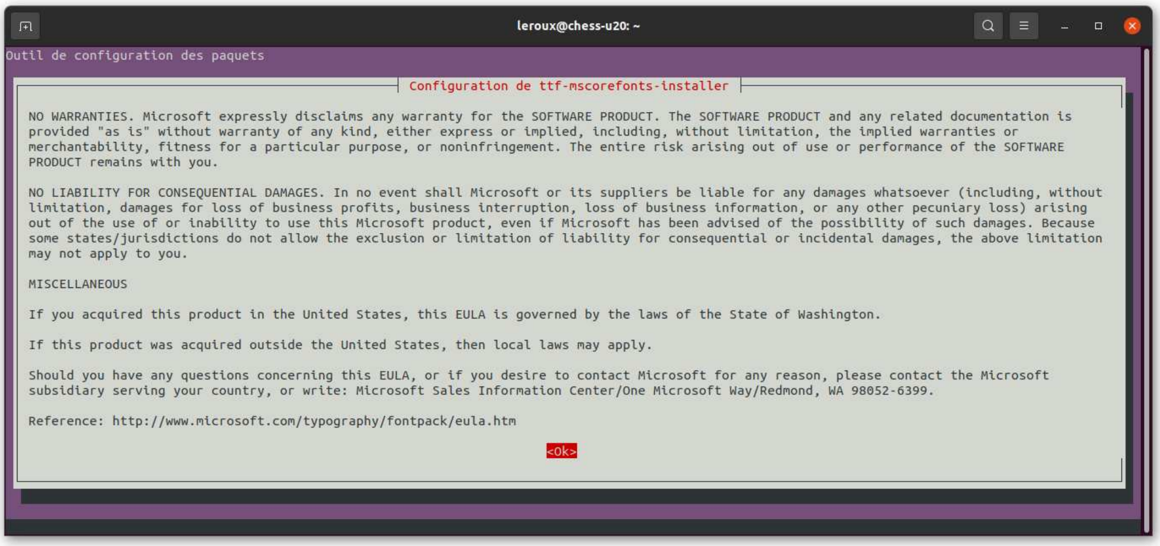
```
leroux@chess-u20:~$ sudo apt install ubuntu-restricted-extras
```

- After pressing Enter you should en-up with with something like:



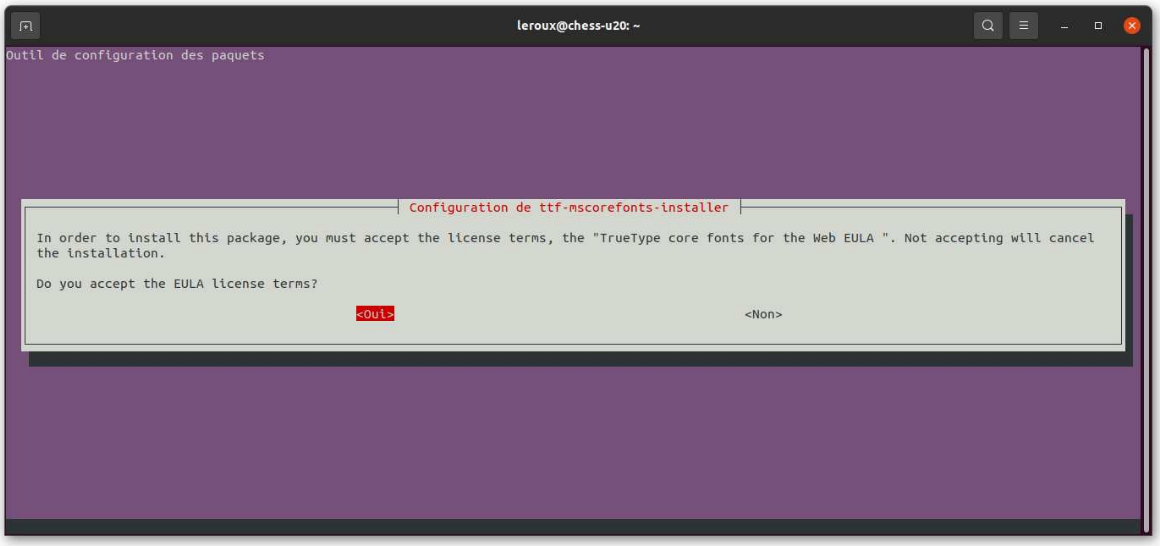
```
leroux@chess-u20:~$ sudo apt install ubuntu-restricted-extras
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets supplémentaires suivants seront installés :
  cabextract chromium-codecs-ffmpeg-extra gstreamer1.0-libav gstreamer1.0-plugins-ugly gstreamer1.0-vaapi liba52-0.7.4 libaribb24-0 libass9
  libavcodec-extra libavcodec-extra58 libavfilter7 libbs2b0 libdvdnav4 libdvdread7 libfftw3-double3 libflite1 libgstreamer-plugins-bad1.0-0
  liblilv-0-0 libmpeg2-4 libmysofa1 libnorm1 libopencore-amrnb0 libopencore-amrwb0 libpgm-5.2-0 libpostproc55 librubberband2 libserd-0-0
  libsidplay1v5 libsord-0-0 libsratom-0-0 libva-wayland2 libvidstab1.1 libvo-amrwbenc0 libzmq5 ttf-mscorefonts-installer ubuntu-restricted-addons
  unrar
Paquets suggérés :
  gstreamer1.0-vaapi-doc libdvdcss2 libfftw3-bin libfftw3-dev serdi sidplay-base sordi
Paquets recommandés :
  gstreamer1.0-fluendo-mp3
Les paquets suivants seront ENLEVÉS :
  libavcodec58
Les NOUVEAUX paquets suivants seront installés :
  cabextract chromium-codecs-ffmpeg-extra gstreamer1.0-libav gstreamer1.0-plugins-ugly gstreamer1.0-vaapi liba52-0.7.4 libaribb24-0 libass9
  libavcodec-extra libavcodec-extra58 libavfilter7 libbs2b0 libdvdnav4 libdvdread7 libfftw3-double3 libflite1 libgstreamer-plugins-bad1.0-0
  liblilv-0-0 libmpeg2-4 libmysofa1 libnorm1 libopencore-amrnb0 libopencore-amrwb0 libpgm-5.2-0 libpostproc55 librubberband2 libserd-0-0
  libsidplay1v5 libsord-0-0 libsratom-0-0 libva-wayland2 libvidstab1.1 libvo-amrwbenc0 libzmq5 ttf-mscorefonts-installer ubuntu-restricted-addons
  ubuntu-restricted-extras unrar
0 mis à jour, 38 nouvellement installés, 1 à enlever et 3 non mis à jour.
Il est nécessaire de prendre 22,3 Mo dans les archives.
Après cette opération, 42,9 Mo d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [O/n]
```

- Follow the procedure to install the extra codecs and packages, at some point the installer offers to install MS fonts for Linux (that might be useful for compatibility purposes with your old documents during your transition to the open world and Linux) and you will end-up with the following screen:



```
Outil de configuration des paquets

Configuration de ttf-mscorefonts-installer

NO WARRANTIES. Microsoft expressly disclaims any warranty for the SOFTWARE PRODUCT. The SOFTWARE PRODUCT and any related documentation is
provided "as is" without warranty of any kind, either express or implied, including, without limitation, the implied warranties or
merchantability, fitness for a particular purpose, or noninfringement. The entire risk arising out of use or performance of the SOFTWARE
PRODUCT remains with you.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. In no event shall Microsoft or its suppliers be liable for any damages whatsoever (including, without
limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising
out of the use of or inability to use this Microsoft product, even if Microsoft has been advised of the possibility of such damages. Because
some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation
may not apply to you.

MISCELLANEOUS

If you acquired this product in the United States, this EULA is governed by the laws of the State of Washington.

If this product was acquired outside the United States, then local laws may apply.

Should you have any questions concerning this EULA, or if you desire to contact Microsoft for any reason, please contact the Microsoft
subsidiary serving your country, or write: Microsoft Sales Information Center/One Microsoft Way/Redmond, WA 98052-6399.

Reference: http://www.microsoft.com/typography/fontpack/eula.htm

                                              <Ok>
```

- Press ⟺ / tab to activate the `<Ok>` button, then press Enter , and validate a choice to install or not the fonts:



```
Outil de configuration des paquets

Configuration de ttf-mscorefonts-installer

In order to install this package, you must accept the license terms, the "TrueType core fonts for the Web EULA ". Not accepting will cancel
the installation.

Do you accept the EULA license terms?

<Oui>                                            <Non>
```

4. After that a lot of text is likely to pop down the terminal, do not be afraid, when this is done the codecs and extra utilities will be installed.

5.2.4 Installing new software

5.2.4.1 Using the terminal and the `apt` command

To install software open the terminal and use the `apt` command:

```
user@localhost:~$ sudo apt install package-name
```

Example:

```
user@localhost:~$ sudo apt install synaptic
```

You can see that installing software requires to request admin privileges using the "`sudo`" command therefore you will need to be in the "`sudoers`" group to perform this action.
All the other examples provided thereafter to install software are tools that combine a search engine, to look for programs, and graphical interface to the `sudo`+`apt` commands.

5.2.4.2 Using `Ubuntu Software`

For beginners the easiest way to install software on Ubuntu is to used the "**`Ubuntu Software`**" dialog, to open it either:

- Use the mouse an click on the appropriate icon in the deck of the left side of the desktop.
- Open the terminal and use the **`snap-store`** command.

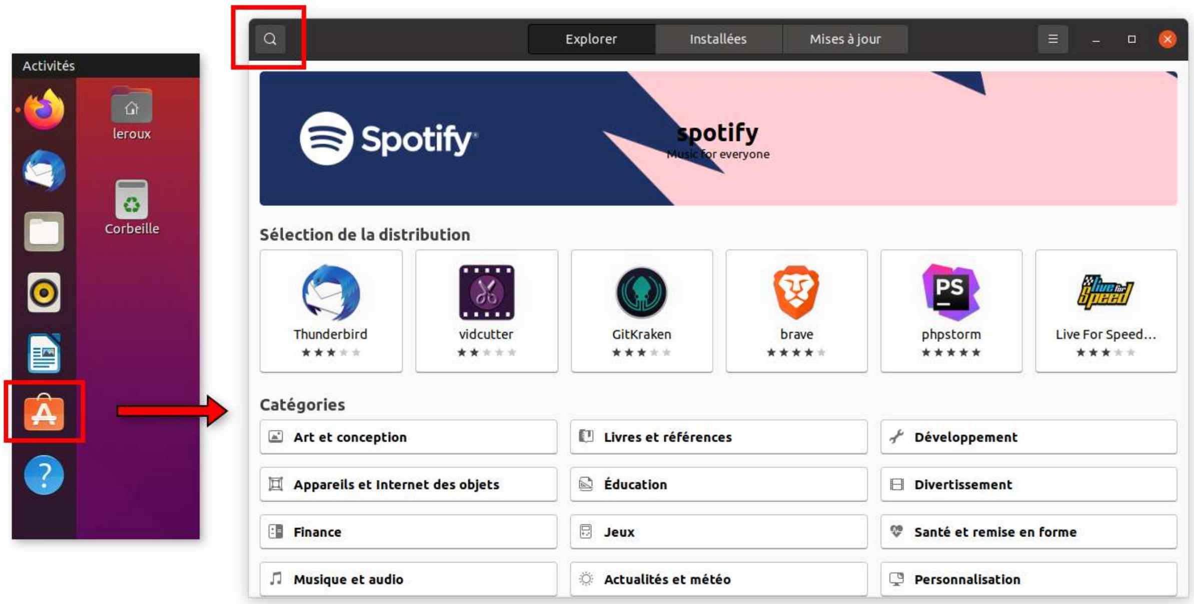


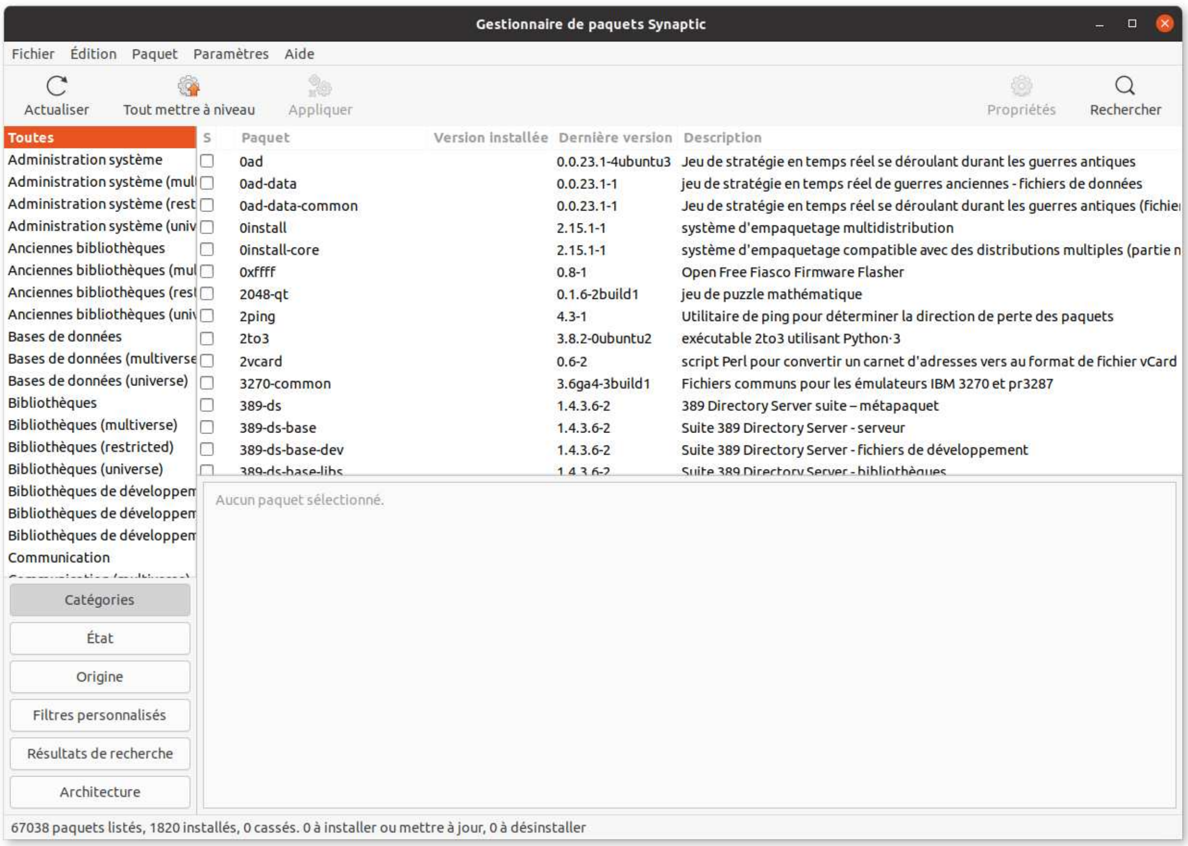
Figure 5.11: The "**`Ubuntu Software`**" dialog in Ubuntu.

From there it is very easy to browse the software library and pick something to install. The bottom menus offer to browse the software library by categories, if you do not find what you need use the search engine by click on the top left search icon.

5.2.4.3 Using `Synaptic`

"**`Synaptic`**" is a very advanced and popular software installer in the Linux world, it offers a much more comprehensive and detailed search engine than "**`Ubuntu Software`**". Remember that Ubuntu "Noble Numbat" 24.04 LTS can offer up to 20 000 programs to install and that it can be tricky to find your way is such a vast library.
The pre-requisite to be able to use **`Synaptic`**, using **`Ubuntu Software`**, or **`apt`**:

```
user@localhost:~$ sudo apt install synaptic
```


Gestionnaire de paquets Synaptic
Fichier
Édition
Paquet
Paramètres
Aide
Actualiser
Tout mettre à niveau
Appliquer
Propriétés
Rechercher
Toutes
Administration système
Administration système (mul
Administration système (rest
Administration système (univ
Anciennes bibliothèques
Anciennes bibliothèques (mul
Anciennes bibliothèques (rest
Anciennes bibliothèques (univ
Bases de données
Bases de données (multiverse
Bases de données (universe)
Bibliothèques
Bibliothèques (multiverse)
Bibliothèques (restricted)
Bibliothèques (universe)
Bibliothèques de développem
Bibliothèques de développem
Bibliothèques de développem
Communication
Catégories
État
Origine
Filtres personnalisés
Résultats de recherche
Architecture
Aucun paquet sélectionné.

S	Paquet	Version installée	Dernière version	Description
☐	0ad		0.0.23.1-4ubuntu3	Jeu de stratégie en temps réel se déroulant durant les guerres antiques
☐	0ad-data		0.0.23.1-1	jeu de stratégie en temps réel de guerres anciennes - fichiers de données
☐	0ad-data-common		0.0.23.1-1	Jeu de stratégie en temps réel se déroulant durant les guerres antiques (fichie
☐	0install		2.15.1-1	système d'empaquetage multidistribution
☐	0install-core		2.15.1-1	système d'empaquetage compatible avec des distributions multiples (partie n
☐	0xffff		0.8-1	Open Free Fiasco Firmware Flasher
☐	2048-qt		0.1.6-2build1	jeu de puzzle mathématique
☐	2ping		4.3-1	Utilitaire de ping pour déterminer la direction de perte des paquets
☐	2to3		3.8.2-0ubuntu2	exécutable 2to3 utilisant Python·3
☐	2vcard		0.6-2	script Perl pour convertir un carnet d'adresses vers au format de fichier vCard
☐	3270-common		3.6ga4-3build1	Fichiers communs pour les émulateurs IBM 3270 et pr3287
☐	389-ds		1.4.3.6-2	389 Directory Server suite – métapaquet
☐	389-ds-base		1.4.3.6-2	Suite 389 Directory Server - serveur
☐	389-ds-base-dev		1.4.3.6-2	Suite 389 Directory Server - fichiers de développement
☐	389-ds-base-libs		1.4.3.6-2	Suite 389 Directory Server - bibliothèques

67038 paquets listés, 1820 installés, 0 cassés. 0 à installer ou mettre à jour, 0 à désinstaller

Figure 5.12: The "**`Synaptic`**" dialog in Ubuntu.

5.2.4.4 What if you can't find what you are looking for ?

If you want more on-line repositories to choose and install software from you can browse the dedicated Ubuntu web page:

https://doc.ubuntu-fr.org/depots_focal

Note that some software companies do not provide packages via the standards Ubuntu distribution system, therefore you might need to check their website and in particular the section dedicated to the Linux download and/or installation. They usually guide you the procedure from the start to finish.

5.2.5 Ubuntu tips and tricks

Now that you are trying Ubuntu you might want to tweak its look and enjoy the full potential of the GNOME desktop.
You will see that the GNOME desktop is often referred to as GNOME shell, that is GNOME and all the extensions, plugins, tools, greffons, that can be installed to improve the user experience. GNOME shell was pre-configured by the Ubuntu team in a way they though was appropriate

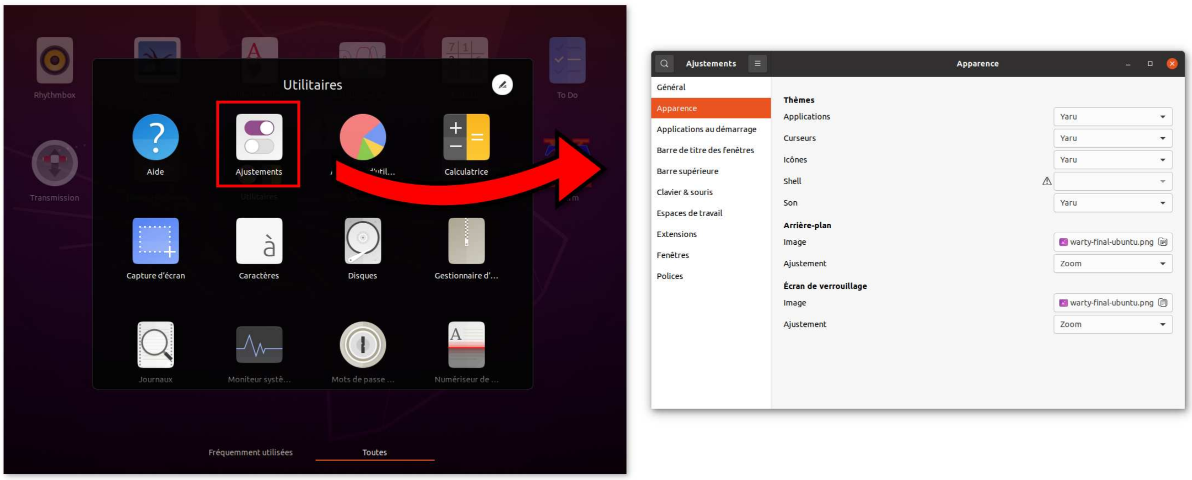
for most users, in particular beginners in the Linux world. The interface is very easy to handle, but unfortunately is letting little room for a proper configuration. Because Ubuntu decided to simplify the user experience they also kept hidden some the tools required to administer / customize GNOME shell.
In the following I will show you how to install the tools that will improve your experience with GNOME shell.

5.2.5.1 Tweaks - Ajustements

The Ubuntu control panel presented in section [Sec. 5.2.2.1] is the default tool provide in Ubuntu to configure GNOME shell. If works perfectly but provided a limited user experience of the possibility of GNOME shell, to get more you will need first to install the "**`Tweaks`**" program ("**`Ajustements`**" in French). To do that you can use "**`Ubuntu Software`**", "**`Synaptic`**" (search for "**`gnome-tweaks`**") or the following command line:

```
user@localhost:~$ sudo apt install gnome-tweaks
```

`Tweaks` provide a complete interface to customize the GNOME desktop (user themes, looks, window system ...), you can launch **`Tweaks`** via the application menus, you will find the appropriate launcher in the "utilities" submenu (see figure [Fig. 5.13]) Alternatively you can enter the **`gnome-tweaks`** command in the terminal:



Thèmes
Applications Yaru
Curseurs Yaru
Icônes Yaru
Shell
Son Yaru
Arrière-plan
Image warty-final-ubuntu.png
Ajustement Zoom
Écran de verrouillage
Image warty-final-ubuntu.png
Ajustement Zoom

Figure 5.13: The "**`Tweaks`**" dialog in Ubuntu "Jammy Jellyfish" 22.04 LTS.

`Tweaks` allows to install user themes that can be found at:

https://www.gnome-look.org/

5.2.5.2 Extensions

In Ubuntu GNOME shell extensions can not be managed using the Ubuntu control panel nor using **`gnome-tweaks`**, to do that it is required to install:

- Ubuntu "Jammy Jellyfish" 22.04 LTS: using **`gnome-shell-extension-prefs`**

  ```
  user@localhost:~$ sudo apt install gnome-shell-extension-prefs
  ```

  The **`Extensions`** provides an interface to turn on/off all the installed GNOME shell extensions, to launch **`Extensions`** open the application menus and search for "**`extensions`**", then click on the appropriate launcher. Alternatively you can enter the `gnome-shell-extension-prefs` command in the terminal:

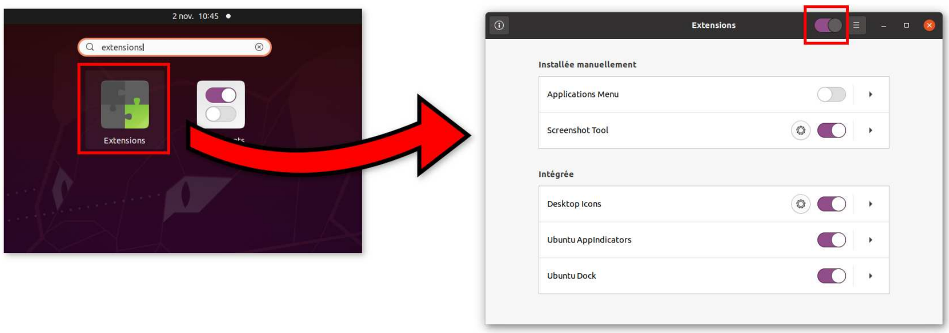
  

  Figure 5.14: The "**`Extensions`**" dialog in Ubuntu "Jammy Jellyfish" 22.04 LTS.

  `Extensions` allows to turn on/off each extensions of GNOME shell, but also all extensions at once using the button located in the top bar of the application window (see figure [Fig. 5.14]). When possible **`Extensions`** also offers a configuration interface to the extension.

- Ubuntu "Noble Numbat" 24.04 LTS: using **`gnome-shell-extension-manager`**

  ```
  user@localhost:~$ sudo apt install gnome-shell-extension-manager
  ```

  `Extension manager` provides an interface to turn on/off all the installed GNOME shell extensions, To launch **`Extension manager`** open the application menus and search for "**`extension`**", then click on the appropriate launcher. Alternatively you can enter the `extension-manager` command in the terminal (see figure [Fig. 5.15]):
  `Extension manager` allows to turn on/off each extensions of GNOME shell, but also all extensions at once using the button located in the top bar of the application window (see figure [Fig. 5.15]). When possible the "Extension manager" also offers a configuration interface to the extension.

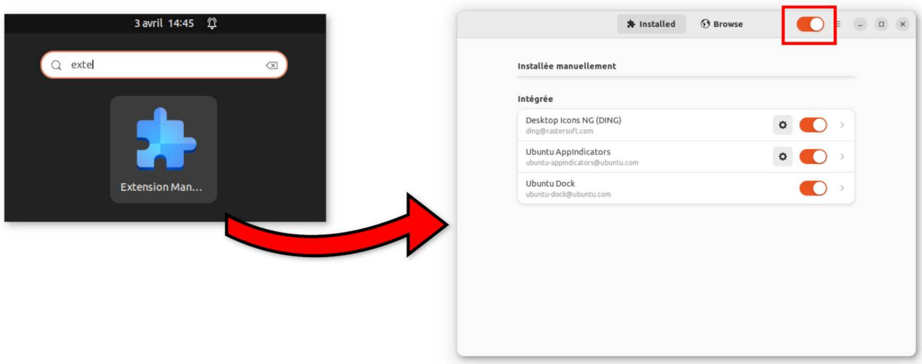


Figure 5.15: The "**`Extension manager`**" dialog in Ubuntu "Noble Numbat" 24.04 LTS.

`Extension manager` also provides a tab "Browse" to search for, and install, new extensions:

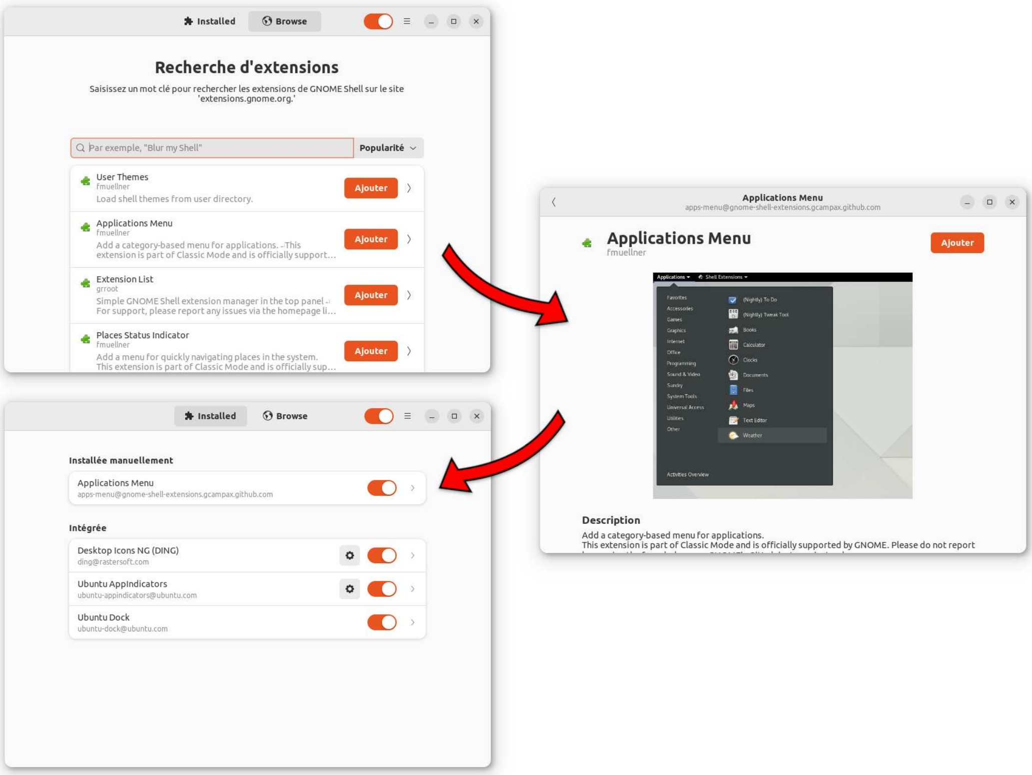

Installed
Browse
Recherche d'extensions
Saisissez un mot clé pour rechercher les extensions de GNOME Shell sur le site 'extensions.gnome.org.'
Par exemple, "Blur my Shell"
Popularité
User Themes
fmuellner
Load shell themes from user directory.
Ajouter
Applications Menu
fmuellner
Add a category-based menu for applications. -This extension is part of Classic Mode and is officially support...
Ajouter
Extension List
grroot
Simple GNOME Shell extension manager in the top panel - For support, please report any issues via the homepage li...
Ajouter
Places Status Indicator
fmuellner
Add a menu for quickly navigating places in the system. This extension is part of Classic Mode and is officially sup...
Ajouter
Applications Menu
apps-menu@gnome-shell-extensions.gcampax.github.com
Applications Menu
fmuellner
Ajouter
Description
Add a category-based menu for applications.
This extension is part of Classic Mode and is officially supported by GNOME. Please do not report
Installed
Browse
Installée manuellement
Applications Menu
apps-menu@gnome-shell-extensions.gcampax.github.com
Intégrée
Desktop Icons NG (DING)
ding@rastersoft.com
Ubuntu AppIndicators
ubuntu-appindicators@ubuntu.com
Ubuntu Dock
ubuntu-dock@ubuntu.com

Figure 5.16: Adding an extension using the "**`Extension manager`**".

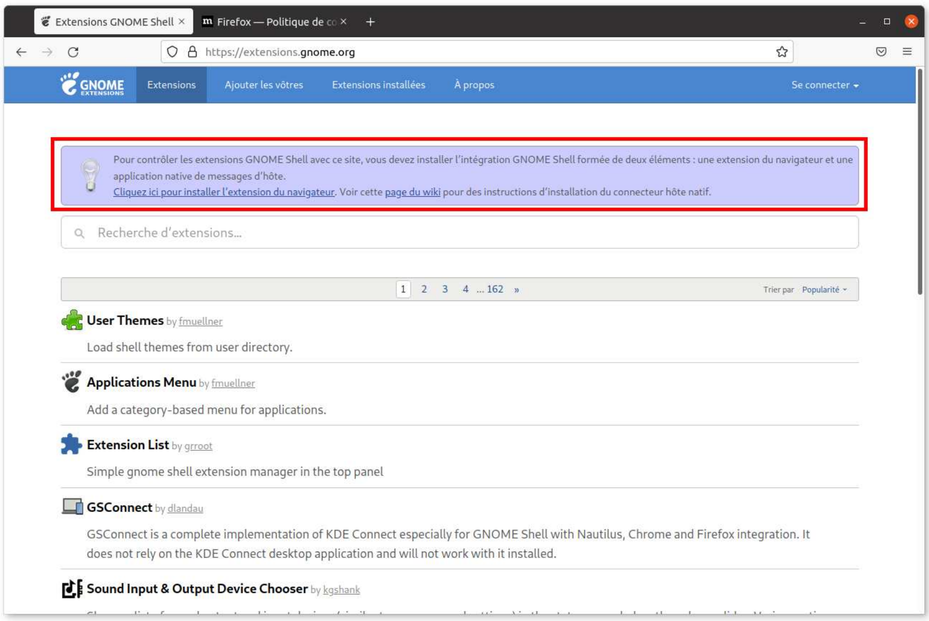
5.2.5.3 Ubuntu "Focal Fossa" 20.04 LTS only: "https://extensions.gnome.org/"

Even if you can still do this in Ubuntu "Noble Numbat" 24.04 LTS, the feature to easily install on-line extension has been natively added to the "Extension manager" starting in Ubuntu "Jammy Jellyfish" 22.04 LTS, therefore the content of this section only applies to Ubuntu "Focal Fossa" 20.04 LTS.

This webpage is the easiest way to browse and install extensions for the GNOME desktop:

https://extensions.gnome.org/

1. At your first visit the website will present a message inserted in a blue ribbon on top:



Pour contrôler les extensions GNOME Shell avec ce site, vous devez installer l'intégration GNOME Shell formée de deux éléments : une extension du navigateur et une application native de messages d'hôte.
Cliquez ici pour installer l'extension du navigateur. Voir cette page du wiki pour des instructions d'installation du connecteur hôte natif.

User Themes by fmuellner
Load shell themes from user directory.

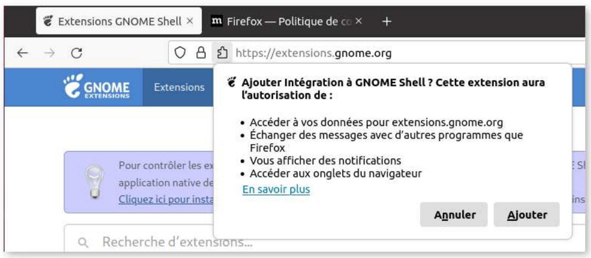
Applications Menu by fmuellner
Add a category-based menu for applications.

Extension List by grroot
Simple gnome shell extension manager in the top panel

GSConnect by dlandau
GSConnect is a complete implementation of KDE Connect especially for GNOME Shell with Nautilus, Chrome and Firefox integration. It does not rely on the KDE Connect desktop application and will not work with it installed.

Sound Input & Output Device Chooser by kgshank

The message invites you to install a web browser extension dedicated to web site:



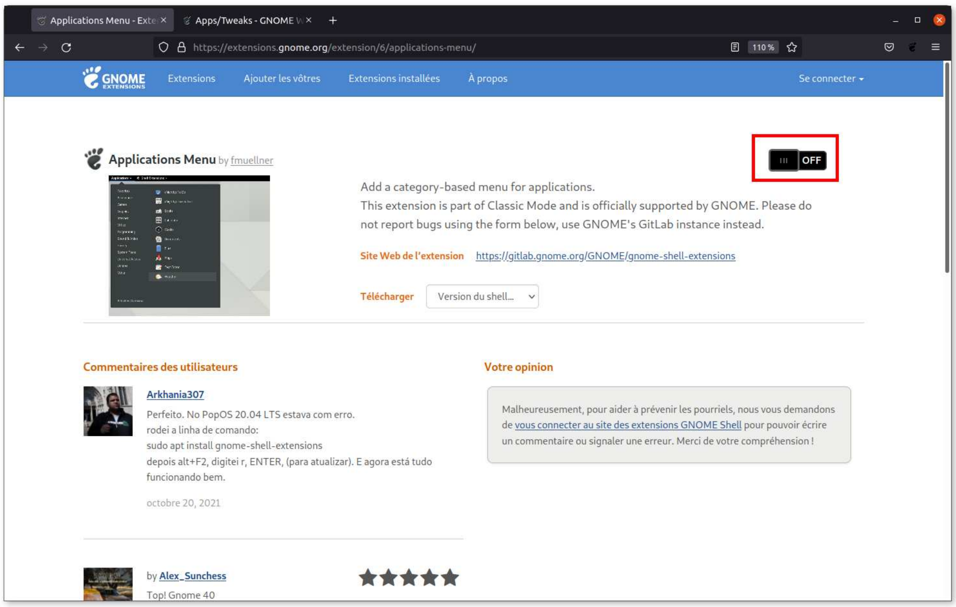
Extensions GNOME Shell
Firefox — Politique de co
https://extensions.gnome.org
GNOME EXTENSIONS
Extensions
Annuler
Ajouter
Recherche d'extensions...

Ajouter Intégration à GNOME Shell ? Cette extension aura l'autorisation de :

- Accéder à vos données pour extensions.gnome.org
- Échanger des messages avec d'autres programmes que Firefox
- Vous afficher des notifications
- Accéder aux onglets du navigateur

En savoir plus

2. Follow the procedure to install the extension, then the layout of web pages of the site will look like:



Applications Menu - Exte × | Apps/Tweaks - GNOME W ×

https://extensions.gnome.org/extension/6/applications-menu/

GNOME EXTENSIONS | Extensions | Ajouter les vôtres | Extensions installées | À propos | Se connecter

Applications Menu by fmuellner — OFF

Add a category-based menu for applications.
This extension is part of Classic Mode and is officially supported by GNOME. Please do not report bugs using the form below, use GNOME's GitLab instance instead.

Site Web de l'extension https://gitlab.gnome.org/GNOME/gnome-shell-extensions

Télécharger Version du shell...

Commentaires des utilisateurs

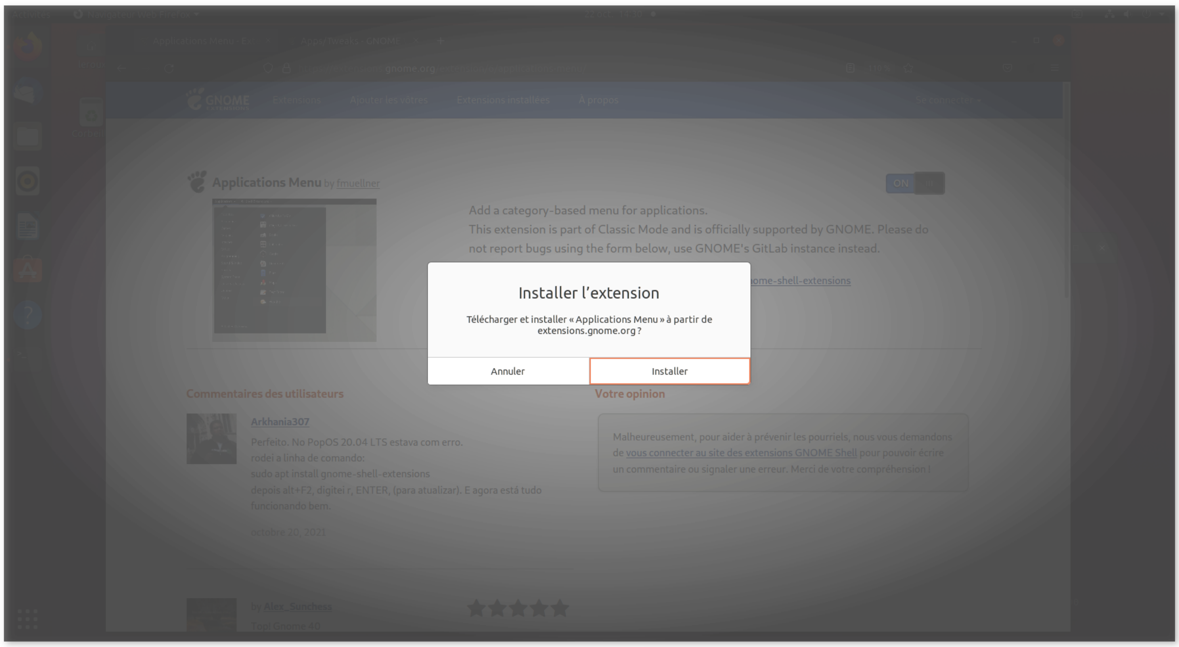
Arkhania307
Perfeito. No PopOS 20.04 LTS estava com erro.
rodei a linha de comando:
sudo apt install gnome-shell-extensions
depois alt+F2, digitei r, ENTER, (para atualizar). E agora está tudo funcionando bem.
octobre 20, 2021

by Alex_Sunchess ★★★★★
Top! Gnome 40

Votre opinion

Malheureusement, pour aider à prévenir les pourriels, nous vous demandons de vous connecter au site des extensions GNOME Shell pour pouvoir écrire un commentaire ou signaler une erreur. Merci de votre compréhension !

The switch offer a shortcut to install/activate/deactivate directly the GNOME extensions presented on the web page:

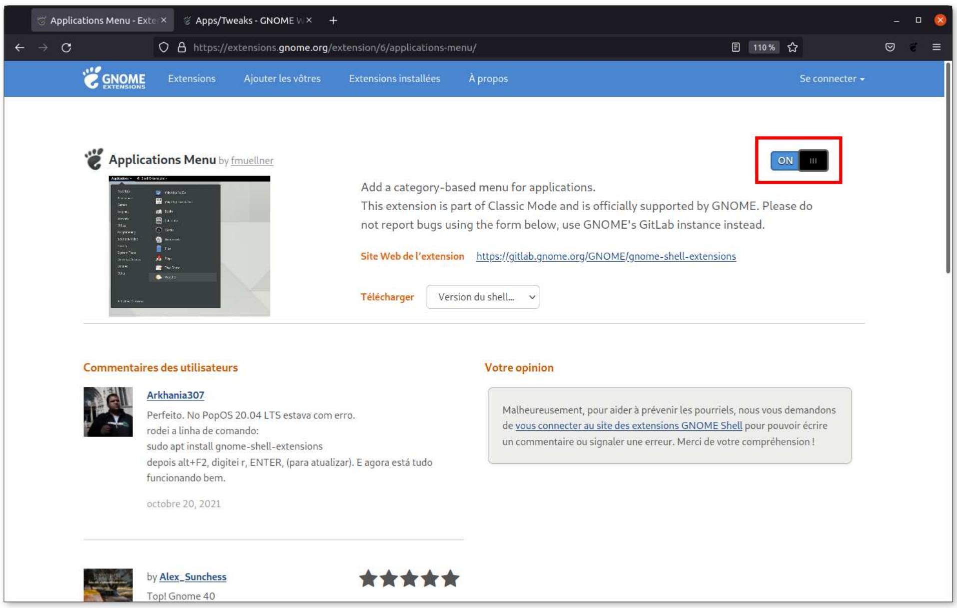


Installer l'extension

Télécharger et installer « Applications Menu » à partir de extensions.gnome.org ?

Annuler | Installer

3. If your are interested in an extensions then turn on the switch and follow the procedure. When the installation procedure is over, the switch remains active and you can turn on and off the extension for GNOME shell



Applications Menu - Exte × | Apps/Tweaks - GNOME W ×

https://extensions.gnome.org/extension/6/applications-menu/

GNOME EXTENSIONS | Extensions | Ajouter les vôtres | Extensions installées | À propos | Se connecter

Applications Menu by fmuellner

ON

Add a category-based menu for applications.
This extension is part of Classic Mode and is officially supported by GNOME. Please do not report bugs using the form below, use GNOME's GitLab instance instead.

Site Web de l'extension https://gitlab.gnome.org/GNOME/gnome-shell-extensions

Télécharger Version du shell...

Commentaires des utilisateurs

Arkhania307
Perfeito. No PopOS 20.04 LTS estava com erro.
rodei a linha de comando:
sudo apt install gnome-shell-extensions
depois alt+F2, digitei r, ENTER, (para atualizar). E agora está tudo funcionando bem.

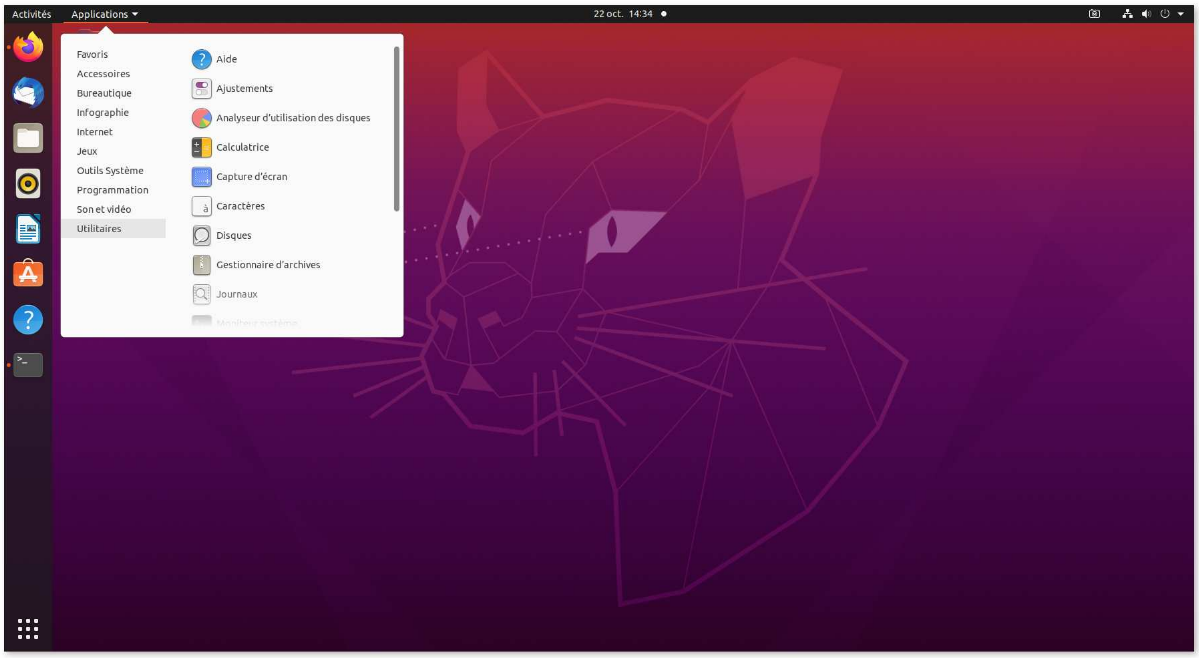
octobre 20, 2021

by Alex_Sunchess ★★★★★
Top! Gnome 40

Votre opinion

Malheureusement, pour aider à prévenir les pourriels, nous vous demandons de vous connecter au site des extensions GNOME Shell pour pouvoir écrire un commentaire ou signaler une erreur. Merci de votre compréhension !

4. The switch turn on, after the installation of the extension "Applications Menu":



Using the command line

This chapter introduces the basics about using the command line.
The next sections will focus on the presentation of the elements required to get a beginner level understanding of using this text interface to your computer.
The steps of this process can be resumed as answering the following questions:

- What is a command interpreter ?
- What is a command ?
- Where to find command(s) ?
- How to execute a command ?
- How to use a command ?
- How to get help using command(s) ?
- What are the basic commands ?
- What are the filters ?
- What is a redirection ?
- What is scripting ?

While answering these questions, some examples will be provided to illustrate some possible usages of the command line.

6.1 What is a command interpreter ?

A command interpreter, also known as a shell, is a program that allows users to interact with the system by entering commands. The command interpreter reads commands typed by the user, interprets them, and executes them. It provides a command-line interface for accessing the operating system and its utilities.
There are several command interpreters available on Linux, including BASH, TCSH, ZSH, KSH ... (the name usually ends by SH for SHell), each with its own set of features and capabilities.
Understanding how to use a command interpreter is essential for anyone who wants to work with Linux.

6.1.1 The BASH "Bourne-Again" SHell

BASH, or GNU BASH, is the shell, or command language interpreter, that will appear by default in most GNU operating systems.
BASH includes:

- Command line editing
- Unlimited size command history
- Job Control
- Shell Functions and Aliases
- Indexed arrays of unlimited size
- Integer arithmetic in any base from two to sixty-four

The manual is available online at http://www.gnu.org/software/bash/manual/.
You can also check my Basic tutorial to BASH programming.

BASH Start-up scripts

When BASH starts it executes commands in a variety of different scripts.

- When BASH is invoked as an interactive login shell it:
  1. First reads and executes commands from the file `/etc/profile`, if that file exists.
  2. Then looks for `~/.bash_profile` (in Unix/Linux language `~= $HOME`), `~/.bash_login` and `~/.profile`, in that order, and reads and executes commands from the first one that exists and is readable.
- When an interactive shell that is not a login shell is started, BASH reads and executes commands from `~/.bashrc`, if that file exists.
- When a login shell exits, BASH reads and executes commands from the file `~/.bash_logout`, if it exists.

BASH native commands

The following are the most important commands to know about to start using BASH:

- To display something, the `echo` command:
  - To display some text:

    ```
    user@localhost:~$ echo "This is the text to echo !"
    This is the text to echo !
    user@localhost:~$
    ```

  - To display the value of an environment variable, using the "`$`" symbol:

    ```
    user@localhost:~$ echo $HOME
    /home/user
    user@localhost:~$
    ```

    Without the "`$`" symbol:

    ```
    user@localhost:~$ echo HOME
    HOME
    user@localhost:~$
    ```

- To print the active working directory, the `pwd` command, see section [Sec. 6.7.1.2].
- To print the commands history, the `history` command:

  ```
  user@localhost:~$ history
     48  cd ..
     49  ls
     50  cp Ethanol.xyz ~/Documents
     52  ls -l ~/Documents
     53  cd Documents
  user@localhost:~$
  ```

- To create alias(es), the `alias` command, see section [Sec. 6.10].
- To make environment variables hereditary, the `export` command, see section [Sec. 6.10].

Special characters and character protection

Here are some of the special characters in BASH, that is characters that already have meaning, command shortcut or else, that will take priority over printing the character itself:

- The " ", space character: to avoid in file and directory names.
  Spaces are the default field (or name) separators on the command line, thus including a space in any name it-self is to be avoided.
- The `$` dollar character: what follow is a variable.
- The `*` star character: stands for any number of any character(s) = everything.
- The `?` question mark character: any single character.
- The `{}` `()` `[]` brackets characters: to encompass expressions.
- The `/` slash character: to define file and directory path(s).
- The `` ` ` `` inverse quote characters: to substitute enclosed command(s).
- The `''` single quote characters: to enclose command(s).
- The `" "` double quote characters: to enclose command(s) with variable(s) expansion.
- The `#` sharp character: to start a comment in BASH.
- The `&` `<` `>` characters: used for redirection, see [Sec. 6.9].
- The `.` and `..` characters: used for navigation in the directory tree.
- The `\` backslash character: to protect the other special characters.

The list in not exhaustive, and other characters might require to be protected, meaning preceded by a `\` on the command line. To protect a character means to tell BASH to display it, as is, instead of using its default behavior detailed above.

For example, if the directory "`Documents`" was renamed "`Doc uments`" by mistake:

```
user@localhost:~$ ls Doc*
'Doc uments'
```

To change directory to "`Doc uments`":

```
user@localhost:~$ cd Doc\ uments
user@localhost:~/Doc uments$
```

6.1.2 Linux files and command line terminology

You can distinguish 2 types of files in any computer system:

- Text files.
- The other types of file = Binary files = files that are not text files.

It is easy to work on text files using the command line, it is more challenging for binary files.

On a Linux system you can also distinguish between 2 categories of files and/or repositories:

- Standard files and repositories
- Hidden files and repositories which names, by convention, start by a dot symbol `.`, for example: `~/.bashrc` for more information about this file see section [Sec. 6.10].

Finally some terms are frequently used to describe the interaction with computers, in particular when using the command line:

- Standard input (`STDIN`): the device from which input to the system is taken, typically your keyboard or you mouse.
- Standard output (`STDOUT`): the terminal window that display information from the command interpreter.
- Standard error (`STDERR`): the potential error messages of the commands.

These might come up when looking for help, or while trying go beyond this manual.

6.2 Command ? Is executable !

Since using the command line is about giving command(s) to the command interpreter the next legitimate question to ask yourself is what is a command in Linux ?
The answer to that question is simple:

A command is a file that has the execute permission = **a file that you can execute !**

6.3 Where to find command(s) ?

The next question that comes naturally to mind then, where do you find file(s) that have the execute permission ? Where do you find the commands ?
And there are two answers to that question:

- In the `PATH`
- Anywhere you need, because you can create command(s) yourself.

6.3.1 `PATH`

`PATH` is the environment variable [EV] (see [Sec. 4.2.5] for more information) that lists all the directories that potentially contains commands:

- To show all [EV] and their values use the `env` command:

```
user@localhost:~$ env
```

- To show the content of any [EV] variable `VAR` use the `echo` command + `$` symbol:

```
user@localhost:~$ echo $VAR
```

  For example to show the content of `PATH`:

```
user@localhost:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/snap/bin
user@localhost:~$
```

  This commands displays a list of directories separated by `:` symbols.
  When the command interpreter will look for command(s) entered through the command line it will look successively in all the directory(ies) listed in the `PATH` for that particular command. Based on the previous example it will look:

- First in: `/usr/local/sbin`
- Then in: `/usr/local/bin`
- Then in: `/usr/sbin`
- And so on ...

The search will stop a the first occurrence of the command in any directory.

You can list the content of any directory listed in the **PATH** so see what commands are available inside.

6.3.2 Creating a command

To do that you basically only need to give the execution permission to the file you want to execute, that means to use as command, to do that use the **chmod** command (see section [Sec. 4.2.4.1] for more information):

```
user@localhost:~$ chmod 700 MyCommand
```

Now that **MyCommand** has the proper file permissions it is possible to try to execute, and use it, as a command.

6.3.3 Locating a command

Also to know where the command you are using is coming from, use the **which** command:

```
user@localhost:~$ which ls
/usr/bin/ls
user@localhost:~$
```

This will locate the command you want to use providing that this command is in the **PATH**.

6.3.4 Adding a command to the PATH

See section [Sec. 6.10].

6.4 How to execute a command ?

The answer to this question depends on the way to locate the command for the interpreter:

- The command is in the **PATH** then simply use its name:

```
user@localhost:~$ MyCommand
```

- The command is not in the **PATH**:
  - Use the complete directory path followed by the name of the executable:

    ```
    user@localhost:~$ ~/Documents/MyCommand
    ```

    or

    ```
    user@localhost:~$ /home/user/Documents/MyCommand
    ```

    or

    ```
    user@localhost:~$ Documents/MyCommand
    ```

  - Change directory to the directory of the executable and use `./` symbols:

    ```
    user@localhost:~$ cd Documents
    user@localhost:~/Documents$ ./MyCommand
    ```

    `./` simply means: look in the current active directory.

Note that the command is executed, and will therefore have an effect, in the active working directory.

If you intend to execute a command in another directory then:

- Change directory to the target directory of the command, see section [Sec. 6.7.1.3].
- If possible supply the target directory of the command as an argument.

6.5 How to use a command ?

Commands can receive both:

- Argument(s), data for the command to process:

  ```
  user@localhost:~$ cd Documents
  ```

  The `cd` command will change working directory to the specified directory.

- Option(s), that will specify how the command operates:

  ```
  user@localhost:~/Documents$ ls -la
  ```

  The `ls` command will list the content of the active directory, the results will be provided following the instructions of the `-la` options.

However very often commands can be used without argument(s) or option(s):

```
user@localhost:~/Documents$ ls
```

Many commands accept options.
By convention options follow the command name, and that start by:

- `-letter(s)` where `letter(s)` is one or more single letter(s), each of these letters being used to specify a particular option, ex:

  ```
  user@localhost:~/Documents$ ls -lh
  ```

  `-letter(s)` options can be provided together, as previously, or separately:

  ```
  user@localhost:~/Documents$ ls -l -h
  ```

- `--word` where word is an entire word, used to specify a particular option, ex:

  ```
  user@localhost:~/Documents$ ls --human-readable
  ```

- Either `-letter(s)` and/or `--word(s)`:

  ```
  user@localhost:~/Documents$ ls -lt --human-readable --all
  ```

  Very often `-letter` option has a `--word` equivalent option.

  ```
  user@localhost:~/Documents$ ls -ltah
  ```

  is equivalent to:

  ```
  user@localhost:~/Documents$ ls -l --all -t --human-readable
  ```

Each command has is own set of options and there is no automatic writing to help you figure out what to use and/or how to use it.

6.6 How to get help ?

As illustrated in the previous section getting help on the command you want to use can be important. To do that you can check either:

- The short help of the command: using the `-h` or `--help` options
- The detailed user manual of the command: using the `man` command.

6.6.1 The `-h` or `--help` options

By convention most commands accept the option `-h` or `--help`, sometimes both.
Then the command will simply display a short help message:

```
user@localhost:~$ mkdir --help
Utilisation : mkdir [OPTION]... RÉPERTOIRE...
Créer le ou les RÉPERTOIREs s'ils n'existent pas.

Les arguments obligatoires pour les options longues le sont aussi pour les
options courtes.
  -m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
  -p, --parents     no error if existing, make parent directories as needed,
                    with their file modes unaffected by any -m option.
  -v, --verbose     print a message for each created directory
  -Z                   définir le contexte de sécurité SELinux de tous les
                         répertoires créés au type par défaut
      --context[=CTX]  comme -Z ou, si CTX est indiqué, définir le contexte de
                         sécurité SELinux ou SMACK à CTX
      --help     afficher l'aide et quitter
      --version  afficher des informations de version et quitter

Aide en ligne de GNU coreutils : <https://www.gnu.org/software/coreutils/>
Signalez les problèmes de traduction à : <traduc@traduc.org>
Documentation complète <https://www.gnu.org/software/coreutils/mkdir>
ou disponible localement via: info '(coreutils) mkdir invocation'
```

6.6.2 The `man` command

The `man` command allows to acces the user manual of the command, with usually, much more detailed information.
Manual, or man, pages can be long and it can somehow be complicated to get the information you are interested in. No worry here is a trick to help you, using as example the man pages of the `grep` command (see [Sec. 6.8]):

- Access the man pages of the **grep** command:

```
user@localhost:~$ man grep
```

- The first page should look quite similar to:

```
GREP(1)                          Commandes de l'utilisateur                          GREP(1)

NOM
       grep, egrep, fgrep - Afficher les lignes correspondant à un motif donné

SYNOPSIS
       grep [OPTION...] MOTIF [FICHIER...]
       grep [OPTION...] -e MOTIF ... [FICHIER...]
       grep [OPTION...] -f MOTIF_FICHIER ... [FICHIER...]

DESCRIPTION
       grep  cherche  un  MOTIF  dans  chaque  FICHIER.  MOTIF  est  un  ou  plusieurs  motifs  séparés
       par un retour à la ligne et grep affiche chaque ligne correspondant à un motif.
       Généralement, MOTIF devrait être entre guillemets lorsque grep est utilisé dans un interpréteur
       de commandes.

       Un fichier « - » signifie l'entrée standard. Si aucun FICHIER n'est donné, les recherches
       récursives explorent le répertoire de travail et celles non récursives lisent l'entrée
       standard.

       En  outre,  les  variantes  egrep  et  fgrep  sont respectivement les mêmes que les programmes
       grep -E et grep -F. Ces variantes sont obsolètes mais sont fournies pour la rétro
       compatibilité.

OPTIONS
   Informations générales sur le programme
       --help Afficher un message d'utilisation et quitter.

       -V, --version
              Afficher le numéro de version de grep et quitter.

   Syntaxe du motif
       -E, --extended-regexp
              Interpréter le MOTIF comme une expression rationnelle étendue (ERE, voir ci-dessous).

       -F, --fixed-strings
              Interpréter le MOTIF comme étant une chaîne figée, pas une expression rationnelle.
```

- With a prompt on the last line to input instructions:

```
 Manual page grep(1) line 1 (press h for help or q to quit)
```

- You can use this prompt to search the manual pages using the **/** command follow by a motif or keyword you are looking for:

```
/mot
```

- Pressing Enter will initiate a search for this motif in the document, here **mot**, and the screen will be refreshed to display the text including the first occurrence found.
- Immediately after that press n to jump to the next motif, and so on.

6.7 Basic commands

This section introduces basic commands and the related options.
I decided to regroup the commands as follow:

- File system management: **ls**, **pwd**, **cd**, **touch**, **mkdir**, **mv**, **cp**, **rmdir**, **rm**
- File printing: **wc**, **cat**, **tac**, **more**, **tail**, **cut**
- File management: **chown**, **chmod**, **diff**, **ln**

6.7.1 File system management options

6.7.1.1 The `ls` command

To list content(s):

- **ls** *"list content"*
  - usage: `ls [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls`
  - options: `-l` (detailed list), `-t` (time sort), `-a` (show hidden files), `-h` (human readable)

Using the `ls` command:

```
user@localhost:~$ ls
Bureau Documents Images Modèles Musique Public snap Téléchargements Vidéos
user@localhost:~$
```

Options can be used to modify the output of the `ls` command:

- The detailed list, using: `-l`

```
user@localhost:~$ ls -l
total 36
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Bureau
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Documents
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Images
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Modèles
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Musique
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Public
drwx------.  2 user ipcms 4096 avril 12 11:04 snap
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Téléchargements
drwxr-xr-x.  2 user ipcms 4096 avril 12 11:04 Vidéos
user@localhost:~$
```

The syntax for the line output by the "**ls -l**" command is the following:

– "-" means "file", alternatively "d" means directory and "l" means symbolic link (see section [Sec. 6.7.3.4]).
– "rwxr-xr-x" are the permissions on the file.
– "2" is the number of physical links of the file with the hard drive.
– "user" is the name of the owner of the file.
– "ipcms" is the name of the group the owner of the file belongs to.
– "4096" is the size of the file on the hard drive.
– "avril 12 11:04" is the last modification date, time stamp, of the file.

- List files including the hidden files, "all file", using: **-a** or **--all**

```
user@localhost:~$ ls -a
.               .bash_logout  .cache     Images   Musique  snap
..              .bashrc       .config    .local   .profile Téléchargements
.bash_history   Bureau        Documents  Modèles  Public   Vidéos
user@localhost:~$
```

- List the content of the entire directory tree, using: **-R**
  Recursive display of all file(s) and directory(ies) in the file system tree, see section [Sec. 6.7.1.4].

By default the **ls** command lists the content(s) of the active working directory.
However arguments can be used to specify the element(s), file(s) and/or directory(ies) to be listed:

```
user@localhost:~$ ls /home
user   lost+found
user@localhost:~$
```

or similarly:

```
user@localhost:~$ ls -l /home
total 20
drwx------. 53 user   ipcms   4096 14 avril 16:25 user
drwx------.  2 root   root   16384 14 oct.   2020 lost+found
user@localhost:~$
```

The previous command lists the content of the **/home** directory, that contains:

- The user home directory **user** or **/home/user** or **~**
- A **lost+found** directory that is an automatic construct of the file system.
  You will find a **lost+found** at the root of each Linux partition.

6.7.1.2 The `pwd` command

To print the complete path of the active working directory in the file system:

- **pwd** *"print working directory"*
  - usage: `pwd`
  - ex: `user@localhost:~$ pwd`

Using the `pwd` command:

```
user@localhost:~$ pwd
/home/user
user@localhost:~$
```

The `pwd` command command simply tells you where you are.

6.7.1.3 The `cd` command

To change the active directory:

- **cd** *"change directory"*
  - usage: `cd [ARG]`
  - ex: `user@localhost:~$ cd MyDirectory`

Using the `cd` command:

```
user@localhost:~$ cd Documents
user@localhost:~/Documents$
user@localhost:~/Documents$ pwd
/home/user/Documents
user@localhost:~/Documents$
```

Writing conventions exist to help you out in navigating the directory tree:

- Change to the user home directory **$HOME** or **~**:

  ```
  user@localhost:~$ cd
  ```

- Change back to the previous directory you are coming from, using: `-`

  ```
  user@localhost:~$ cd -
  ```

- Change to the upper n^{th} directory(ies) in the system tree: `../` $\times$ (n-1) + `..`
  - One directory up, using: `..`

```
user@localhost:~$ cd ..
```

  - Two directories up, using: `../..`

```
user@localhost:~$ cd ../..
```

  - And so on, with `../` as many times as required.

6.7.1.4 The `mkdir` command

To create new directory(ies):

- **`mkdir`** *"create directory"*
  - usage: `mkdir [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ mkdir MyNewDirectory`
  - options: `-p` (with parents)

Using the `mkdir` command:

```
user@localhost:~/Documents$ mkdir NewDirectory
user@localhost:~/Documents$ ls
NewDirectory
user@localhost:~/Documents$
```

You can also directly create the entire directory tree, using: `-p`

```
user@localhost:~/Documents$ mkdir -p NewDirectory/Level-1/Level-2/Level-3
user@localhost:~/Documents$ ls -R NewDirectory
NewDirectory:
Level-1

NewDirectory/Level-1:
Level-2

NewDirectory/Level-1/Level-2:
Level-3

NewDirectory/Level-1/Level-2/Level-3:
user@localhost:~$ cd NewDirectory/Level-1/Level-2/Level-3
user@localhost:~/Documents/NewDirectory/Level-1/Level-2/Level-3$ pwd
/home/user/Documents/NewDirectory/Level-1/Level-2/Level-3
user@localhost:~/Documents/NewDirectory/Level-1/Level-2/Level-3$ cd
user@localhost:~$
```

6.7.1.5 The `touch` command

The `touch` command updates the time stamp of file(s) and directory(ies), if the specified file name does not exist then the `touch` command will create an empty file with this name:

- `touch` *"change time stamp"*
  - usage: `touch [ARG]`
  - ex: `user@localhost:~$ touch MyFile`

Using the `touch` command:

```
user@localhost:~$ touch NewFile
user@localhost:~$ ls -l NewFile
drwxr-xr-x.  1 user ipcms    0 avril 12 14:25 NewFile
user@localhost:~$
```

In this case the `touch` command simply created the empty file `NewFile`.

6.7.1.6 The `mv` command

To move file(s) or directory(ies), the `mv` command can be applied to both file(s) and directory(ies), however moving a directory will also move all the files the directory contains along with it:

- **`mv`** *"move"*
  - usage: `mv [ARG1] [ARG2]`
  - ex: `user@localhost:~$ mv MyFile MyNewFile`

Using the `mv` command:

- Move file(s) from one location to another:

```
user@localhost:~$ mv NewFile Documents/
user@localhost:~$ ls Documents
NewDirectory  NewFile
user@localhost:~$
```

- Move directory(ies) from one location to another:

```
user@localhost:~$ mv Documents/NewDirectory/Level-1 Documents/
user@localhost:~$ ls Documents
Level-1  NewDirectory  NewFile
user@localhost:~$ ls Documents/Level-1
Level-2
user@localhost:~$
```

- Move object to the current active directory using: `.`

```
user@localhost:~$ cd Documents
user@localhost:~/Documents$ mv Level-1/Level-2 .
user@localhost:~/Documents$ ls
Level-1  Level-2  NewDirectory  NewFile
user@localhost:~/Documents$
```

- Move object to the upper directory using: `..`

```
user@localhost:~/Documents$ cd Level-2
user@localhost:~/Documents/Level-2$ ls
Level-3
user@localhost:~/Documents/Level-2$ mv Level-3 ..
user@localhost:~/Documents/Level-2$ ls ..
Level-1  Level-2  Level-3  NewDirectory  NewFile
user@localhost:~/Documents/Level-2$
```

6.7.1.7 The `cp` command

To copy file(s) and directory(ies):

- **`cp`** *"copy"*
  - usage: `cp [ARG1] [ARG2]`
  - ex: `user@localhost:~$ cp MyFile MyNewFile`

Using the `cp` command:

- To copy file(s):

```
user@localhost:~/Documents/Level-2$ cp ../NewFile .
user@localhost:~/Documents/Level-2$ ls
NewFile
user@localhost:~/Documents/Level-2$
```

- And directory(ies) recursively, using: `-r`

```
user@localhost:~/Documents/Level-2$ cd ..
user@localhost:~/Documents$ cp -r Level-2 Level-4
user@localhost:~/Documents$ ls
Level-1  Level-2  Level-3  Level-4  NewDirectory  NewFile
user@localhost:~/Documents$ cp -r Level-2 Level-4
user@localhost:~/Documents$ cp -r Level-1 Level-4/level-2
user@localhost:~/Documents$
```

If the destination directory already exists, then the contents of the initial directory will be added to the contents of the existing destination directory.

6.7.1.8 The `rmdir` command

To remove empty directory(ies):

- **`rmdir`** *"remove empty directory"*
  - usage: `rmdir [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ rmdir MyOldDirectory`
  - options: `-p` (with parents), `-r` (recursive), `-f` (force)

Using the `rmdir` command:

- To remove an empty directory:

```
user@localhost:~/Documents$ rmdir Level-3
user@localhost:~/Documents$ ls
Level-1  Level-2  Level-4  NewDirectory  NewFile
user@localhost:~/Documents$
```

- To remove parents empty directories:

```
user@localhost:~/Documents$ rmdir -p Level-4/Level-2/Level-1
user@localhost:~/Documents$ ls
Level-1  Level-2  NewDirectory  NewFile
user@localhost:~/Documents$
```

6.7.1.9 The `rm` command

To remove file(s) and directory(ies):

- **`rm`** *"remove"*
  - usage: `rm [OPTION] ... [ARG]...`
  - ex: `user@localhost:~$ rm MyFile`

Using the `rm` command:

On Ubuntu "Noble Numbat" 24.04 LTS the `rm` command is dangerous as is, in particular because the `-i` option is not a default option. I strongly advise to modify your `~/.bashrc` file to modify the default behavior of the `rm` command, see section [Sec. 6.10].

- To remove files:

```
user@localhost:~/Documents$ rm NewFile
user@localhost:~/Documents$ ls
Level-1  Level-2  Level-4  NewDirectory
user@localhost:~/Documents$
```

  Note that the `rm` command **DEFINITELY** remove the file.

- To ask, prompt a question, before every removal, using: `-i`

```
user@localhost:~/Documents$ rm -i Level-2/NewFile
rm: remove 'Level-2/NewFile' ? n
user@localhost:~/Documents$
```

  To proceed then requires to enter `y` (Yes) or `n` (No).
  To make this automatic edit your `~/.bashrc` file, see section [Sec. 6.10].

- To remove directory(ies), recursively a file tree, using: `-r`

```
user@localhost:~/Documents$ rm -r Level-2
user@localhost:~/Documents$
```

6.7.2 File printing options

For the purpose of this section we will consider that we have 2 files named `C60.xyz` and `Ethanol.xyz` in the **$HOME**/`Documents` directory:

```
user@localhost:~$ ls Documents
C60.xyz  Ethanol.xyz
user@localhost:~$
```

`C60.xyz` and `Ethanol.xyz` are a simple text files, that contains respectively the atomic coordinates the C_{60} Buckminster Fullerene and the ethanol molecule, both in XYZ file format [1].

6.7.2.1 The **wc** command

To count characters, words, lines in a file.

- **wc** *"words and lines count"*
  - usage: `wc [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ wc -l MyFile`
  - options: `-l` (number of lines)

Using the **wc** command:

- To count characters, using: `-m`

```
user@localhost:~/Documents$ wc -m Ethanol.xyz
525 Ethanol.xyz
user@localhost:~/Documents$
```

- To count words, using: `-w`

```
user@localhost:~/Documents$ wc -w Ethanol.xyz
37 Ethanol.xyz
user@localhost:~/Documents$
```

- To count lines, using: `-l`

```
user@localhost:~/Documents$ wc -l Ethanol.xyz
11 Ethanol.xyz
user@localhost:~/Documents$
```

6.7.2.2 The `cat` command

To display the content of the file:

- **`cat`** *"display in standard output"*
  - usage: `cat [ARG]`
  - ex: `user@localhost:~$ cat MyFile`

Using the `cat` command:

```
user@localhost:~/Documents$ cat Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

6.7.2.3 The `tac` command

To display the content of a file in reverse order:

- **`tac`** *"opposite of `cat`"*
  - usage: `tac [ARG]`
  - ex: `user@localhost:~$ tac MyFile`

Using the `tac` command:

```
user@localhost:~/Documents$ tac Ethanol.xyz
O      -1.1981291111      0.0180941111     -0.9072448889
H      -2.1126961111      0.0649821111     -0.6649928889
H      -0.6822251111      0.9536901111      0.8665561111
H      -0.7580021111     -0.8263228889      0.9315601111
H       1.2501238889     -0.9611748889     -0.6188868889
H       1.3252608889      0.8030881111     -0.6846978889
H       1.6265438889     -0.0376928889      0.8456121111
C      -0.4620761111      0.0306281111      0.2946991111
C       1.0111998889     -0.0452918889     -0.0626048889

         9
user@localhost:~/Documents$
```

6.7.2.4 The `more` command

To display the content of a file screen by screen:

- **`more`** *"display in standard output screen by screen"*
  - usage: `more [ARG]`
  - ex: `user@localhost:~$ more MyFile`

Using the `more` command:

```
user@localhost:~/Documents$ more C60.xyz
60

 C        -1.168000       -3.284000       -0.100000
 C        -0.588000       -3.148000       -1.380000
 C         0.808000       -3.164000       -1.228000
 C         1.096000       -3.308000        0.140000
 C        -0.124000       -3.384000        0.836000
 C        -2.340000       -2.576000        0.216000
 C        -2.472000       -1.968000        1.480000
 C        -1.428000       -2.064000        2.420000
 C        -0.256000       -2.776000        2.096000
 C         0.836000       -2.088000        2.664000
 C         2.056000       -2.012000        1.972000
 C         2.184000       -2.624000        0.708000
 C         2.992000       -1.792000       -0.096000
 C         2.704000       -1.648000       -1.464000
 C         1.612000       -2.332000       -2.032000
 C         1.016000       -1.488000       -2.988000
 C        -0.380000       -1.472000       -3.140000
 C        -1.184000       -2.304000       -2.336000
 C        -2.360000       -1.596000       -2.012000
 C        -2.936000       -1.732000       -0.736000
 C         1.740000       -0.276000       -3.012000
 C         2.784000       -0.376000       -2.068000
 C         3.360000       -0.668000        0.668000
--Plus--(38%)
```

The `C60.xyz` is much longer than the `Ethanol.xyz` file, and it is not possible to display the entire content of the file on a single terminal window. The `more` command is like a `cat` command that pause for each screen, to keep going and to continue displaying the content of the file simply press [ ].

6.7.2.5 The `tail` command

To display the last lines of a file:

- **`tail`** *"show me the tail"*

  - usage: `tail [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ tail -f MyFile`
  - options: `-f` (last ten lines with update), `--lines=n` (last n lines)

Using the `tail` command:

- By default display the last 10 lines of the file:

```
user@localhost:~/Documents$ tail C60.xyz
 C       -1.612000        2.332000        2.032000
 C        1.184000        2.304000        2.336000
 C        2.340000        2.576000       -0.216000
 C        0.256000        2.776000       -2.096000
 C       -2.188000        2.624000       -0.708000
 C       -0.808000        3.164000        1.228000
 C        0.588000        3.148000        1.380000
 C        1.168000        3.284000        0.100000
 C        0.124000        3.384000       -0.836000
 C       -1.096000        3.308000       -0.140000
user@localhost:~/Documents$
```

- Display the last 10 lines of the file and keep updating the output, using: `-f`

```
user@localhost:~/Documents$ tail -f C60.xyz
 C       -1.612000        2.332000        2.032000
 C        1.184000        2.304000        2.336000
 C        2.340000        2.576000       -0.216000
 C        0.256000        2.776000       -2.096000
 C       -2.188000        2.624000       -0.708000
 C       -0.808000        3.164000        1.228000
 C        0.588000        3.148000        1.380000
 C        1.168000        3.284000        0.100000
 C        0.124000        3.384000       -0.836000
 C       -1.096000        3.308000       -0.140000
```

  This is useful to follow work in progress: a program performing some calculation is outputting data in a file, the "`tail -f`" command allows to follow the results without opening the file in an editor and having to refresh its content.
  Note that with the `-f` option the prompt remains blocked to update the standard output with the data, to end the task and get the prompt back press: Ctrl + c .

- Display the last *n* lines of a file, using: `--lines=`*n*

```
user@localhost:~/Documents$ tail --lines=2 C60.xyz
 C        0.124000        3.384000       -0.836000
 C       -1.096000        3.308000       -0.140000
user@localhost:~/Documents$
```

6.7.2.6 The `cut` command

To cut, extract and display columns of a file:

- **`cut`** *"cut, extract columns"*
  - usage: `cut [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ cut -c5-10 MyFile`
  - options: `-c` (character numbers: `-cA-B,C-D,E-F...`)

The `cut` command extracts columns based on the character number in the line, think about the file as a grid of L lines $\times$ C characters, then `cut` command allows to extract columns by specifying selections of characters between 1 and C:

- Cut between character 1 to 3:

```
user@localhost:~/Documents$ cut -c 1-3 Ethanol.xyz


C
C
H
H
H
H
H
H
O
user@localhost:~/Documents$
```

- Use coma separated list to specify more than one column:

```
user@localhost:~/Documents$ cut -c 1-3,6-13,24-31,42-49 Ethanol.xyz
       9

C     1.011  -0.045  -0.062
C    -0.462   0.030   0.294
H     1.626  -0.037   0.845
H     1.325   0.803  -0.684
H     1.250  -0.961  -0.618
H    -0.758  -0.826   0.931
H    -0.682   0.953   0.866
H    -2.112   0.064  -0.664
O    -1.198   0.018  -0.907
user@localhost:~/Documents$
```

6.7.3 File management options

For the purpose of this section we will consider that there is one more file named `Ethanol-mod.xyz` in the **$HOME**`/Documents` directory:

```
user@localhost:~$ ls Documents
C60.xyz  Ethanol.xyz Ethanol-mod.xyz
user@localhost:~$
```

`Ethanol-mod.xyz` contains the atomic coordinates of modified ethanol molecule, with two H atoms substitued by N atoms.

6.7.3.1 The `chown` command

To change the owner of a file / directory:

- **`chown`** *"change owner"*
  - usage: `chown [OPTION] ... [ARG1] [ARG2]`
  - ex: `user@localhost:~$ chown newuser.newgroup MyFile`
  - options: `-R` (recursive)

The `chown` command allows to change the owner and / or the group of a file / directory:

```
user@localhost:~$ chown -R leroux.ipcms Documents
user@localhost:~$
```

Will change recursively the ownership of all directory(ies) and files located in `~/Document` including this directory.

6.7.3.2 The chmod command

To change the permission(s) of a file / directory:

- **chmod** *"change permissions"*
  - usage: `chmod [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ chmod 755 MyFile`
  - options: **xyz** with **x**, **y** and **z** between 0 and 7, **-R** (recursive)

The **chmod** command allows to define the access, utilization or modification permissions of a file using 3 numbers:

(0	=	no permission at all	=	`---`)
1	=	execute	=	`--x`
2	=	write	=	`-w-`
4	=	read	=	`r-`

As well as their combinations:

3	=	1 + 2	=	execute + write	=	`-wx`
5	=	1 + 4	=	execute + read	=	`r-x`
6	=	2 + 4	=	write + read	=	`rw-`
7	=	1 + 2 + 4	=	execute + write + read	=	`rwx`

Linux distinguishes 3 classes of users for whom it is possible to grant permissions on a file:

- The owner of the file.
- The group the owner belongs to.
- All other users recognized by the system.

The permissions on the file being granted for each user category by a single combination of the number 1, 2 and 4.

For example it is possible to grant permissions 640 to the `~/Documents/Ethanol.xyz` file:

```
user@localhost:~$ chmod 640 Documents/Ethanol.xyz
user@localhost:~$
```

This will grant permissions to read and modify the file to the owner, and remove the possibility to execute it. It will grant permission to read the file for the members of the group the owner belongs too, and remove the possibility to modify and execute the file. Finally it will not grant any permission to the other users of the system.

6.7.3.3 The `diff` command

Difference(s) between two files:

- **`diff`** *"difference(s) between two files"*
  - usage: `diff [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ diff File-1.dat File-2.dat`
  - options: `--color=always` (to color the output)

The `diff` command compares files line by line to print out the difference:

```
user@localhost:~/Documents$ cat Ethanol-mod.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
N       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
N      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$ diff --color=always Ethanol.xyz Ethanol-mod.xyz
6c6
< H       1.3252608889      0.8030881111     -0.6846978889
---
> N       1.3252608889      0.8030881111     -0.6846978889
8c8
< H      -0.7580021111     -0.8263228889      0.9315601111
---
> N      -0.7580021111     -0.8263228889      0.9315601111
user@localhost:~/Documents$
```

When a difference is found, the `diff` command first tells where the difference is found and what the difference is using a combination of "`line-1`+`letter`+`line-2`":

- `line-1` line number where the difference is found in the first file.
- `letter` is a letter keyword that describes the change in the line:
  - `a` = add the line.
  - `c` = change the line.
  - `d` = delete the line.
- `line-2` line number where the difference is found in the second file.

Then the differences are printed out, for the first file (here `Ethanol.xyz`) the output lines begin by the symbol `<`, while from the second file (here `Ethanol-mod.xyz`) the output lines begin by the symbol `>`.

6.7.3.4 The `ln` command

To create symbolic link(s) / shortcut(s) in the Linux world:

- **`ln`** *"create a link"*
  - usage: `ln [OPTION] ... [ARG1] [ARG2]`
  - ex: `user@localhost:~$ ln -s MyFile MyLink`
  - options: `-s` (create symbolic link)

Using the `ls` command to create a symbolic link:

To create a symbolic it is required to provide the complete path, location in the file system, of the object you want to create a link with.

- With a file:

```
user@localhost:~$ ln -s /home/user/Documents/Ethanol.xyz ../Link-Ethanol.xyz
user@localhost:~$ cd ..
user@localhost:~$ ls -l Link*
lrwxrwxrwx. 1 user ipcms   11 14 avril 12:48 Link-Ethanol.xyz -> /home/user/Documents/Ethanol.xyz
user@localhost:~$
```

The symbolic link `Link-Ethanol.xyz` acts as a shortcut:

```
user@localhost:~$ cat Link-etanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~$
```

- To create a symbolic link with a directory:

```
user@localhost:~$ ln -s /home/user/Documents Dcs
user@localhost:~$ ls -l Dc*
lrwxrwxrwx. 1 user ipcms   11 14 avril 12:49 Dcs -> /home/user/Documents
user@localhost:~$
```

The symbolic link `Dcs` acts as a shortcut to the `~/Documents` directory.

```
user@localhost:~$ cp Link-Ethanol.xyz Dcs/Ethanol-copy.xyz
user@localhost:~$ ls Documents
C60.xyz  Ethanol.xyz  Ethanol-copy.xyz  Ethanol-mod.xyz
user@localhost:~$
```

6.8 Filters

Filters are a type of command-line utility designed to manipulate and process text data.
The most common filters are `awk`, `sed` and `grep` command.

- `grep` is used to search and filter text data. It searches through a specified file for lines that match a given pattern or regular expression, and returns the matching lines.
- `sed` is used to perform operations on text data, such as find and replace, deleting or substituting lines, and appending or inserting lines.
- `awk` is a versatile text processing tool that can perform a variety of tasks, such as selecting and transforming columns of data, filtering out unwanted data, and performing calculations.

Filters are commonly used in combination with other Linux utilities and commands to perform more complex data processing tasks.
Like other commands filters use options, but importantly they require to use **regular expressions** or **regexp**.
Regular expressions, regexp, are a set of rules and patterns used to match and manipulate text data, they are very often specific to a particular filter.
The general form however remains the same:

```
user@localhost:~$ filter option(s) 'regular expression' file
```

or:

```
user@localhost:~$ filter option(s) "regular expression" file
```

The regexp appears between quotes, simple "`'`" or double "`"`", either way.
The next sections will introduce a few examples of regular expressions, in the hope that these could be useful.

6.8.1 The `grep` command

Find the matching lines:

- **`grep`** *"print lines matching a pattern"*
  - usage: `grep [OPTION] ... [REGULAR EXPRESSION] [ARG]`
  - ex: `user@localhost:~$ grep "TOTAL ENERGY =" cpmd.out`
  - options: `-n` (print line number), `-A NUM` (print `NUM` lines after pattern), `-B NUM` (print `NUM` lines before pattern), `-v` (invert correspondence = non-matching lines)

The `grep` command is used to find line(s) that contain pattern(s) in text files, those patterns being described by the regexp. Thus as for the other filters the regexp happens to be the most important part of the instructions given to the command line.
Here are some examples of regular expressions for the `grep` filter:

- To search for a pattern:

  ```
  grep 'linux' File
  ```

  To simply look for every line(s) that contain the pattern `linux`

- To search every line that start by a pattern, using: `^`

  ```
  grep '^linux' File
  ```

  To look for every line(s) that start by the pattern `linux`

Here are some examples of options for the `grep` filter:

- Always color the output, using: `--color=always`

  ```
  user@localhost:~/Documents$ grep --color=always 'C' Ethanol.xyz
  C       1.0111998889     -0.0452918889     -0.0626048889
  C      -0.4620761111      0.0306281111      0.2946991111
  user@localhost:~/Documents$
  ```

  To look for every line(s) that start by the pattern `C`

- Print the line number(s) where the patterns is/are found, using: `-n`

  ```
  user@localhost:~/Documents$ grep --color=always -n '^O' Ethanol.xyz
  11:O      -1.1981291111      0.0180941111     -0.9072448889
  user@localhost:~/Documents$
  ```

  To look for every line(s) that start by the pattern `O`, and print the line number(s).

- Print *n* lines after each pattern is found, using: **-A**

```
user@localhost:~/Documents$ grep --color=always -A 5 '^C' Ethanol.xyz
C        1.0111998889     -0.0452918889     -0.0626048889
C       -0.4620761111      0.0306281111      0.2946991111
H        1.6265438889     -0.0376928889      0.8456121111
H        1.3252608889      0.8030881111     -0.6846978889
H        1.2501238889     -0.9611748889     -0.6188868889
H       -0.7580021111     -0.8263228889      0.9315601111
H       -0.6822251111      0.9536901111      0.8665561111
user@localhost:~/Documents$
```

  To look for every line(s) that start by the pattern `C`, and print 5 lines after each occurrence of the pattern.

- Print *n* lines before each pattern is found, using: **-B**

```
user@localhost:~/Documents$ grep --color=always -B 5 '^O' Ethanol.xyz
H        1.3252608889      0.8030881111     -0.6846978889
H        1.2501238889     -0.9611748889     -0.6188868889
H       -0.7580021111     -0.8263228889      0.9315601111
H       -0.6822251111      0.9536901111      0.8665561111
H       -2.1126961111      0.0649821111     -0.6649928889
O       -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  To look for every line(s) that start by the pattern `O`, and print 5 lines before each occurrence of the pattern.

- Inverse results, print all but the line(s) that contain(s) the pattern, using: **-v**

```
user@localhost:~/Documents$ grep --color=always -v 'H' Ethanol.xyz
         9

C        1.0111998889     -0.0452918889     -0.0626048889
C       -0.4620761111      0.0306281111      0.2946991111
O       -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  To look for every line(s) that does not contain the pattern `H`

6.8.2 The `sed` command

Filter and transform text:

- **`sed`** *"stream editor for filtering and transforming text"*
  - usage: `sed [OPTION] ... [REGULAR EXPRESSION] [ARG]`
  - ex: `user@localhost:~$ sed 's/MYPATH/$HOME/g'`

The `sed` command is used to search, substitute, delete pattern(s) in text lines.
Here are some examples of regular expressions for the `sed` filter:

- Substitute pattern(s), using: `'s/OLD/NEW/VAL'`
  - Substitute all pattern(s) on each line(s), using: `'s/OLD/NEW/g'`

```
user@localhost:~/Documents$ sed 's/1/?/g' Ethanol.xyz
         9

C       ?.0???998889     -0.04529?8889     -0.0626048889
C      -0.462076????      0.030628????      0.294699????
H       ?.6265438889     -0.0376928889      0.8456?2????
H       ?.3252608889      0.803088????     -0.6846978889
H       ?.250?238889     -0.96??748889     -0.6?88868889
H      -0.758002????     -0.8263228889      0.93?560????
H      -0.682225????      0.953690????      0.866556????
H      -2.??2696????      0.064982????     -0.6649928889
O      -?.?98?29????      0.0?8094????     -0.9072448889
user@localhost:~/Documents$
```

  - Substitute the nth pattern on each line(s), using: `'s/OLD/NEW/n'`

```
user@localhost:~/Documents$ sed 's/C /Fe/1' Ethanol.xyz
         9

Fe      1.0111998889     -0.0452918889     -0.0626048889
Fe     -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  - Substitute all starting from the nth pattern on each line(s), using: `'s/OLD/NEW/gn'`

```
user@localhost:~/Documents$ sed 's/8/ /g3' Ethanol.xyz
         9

C       1.01119988 9     -0.045291   9     -0.062604   9
C      -0.4620761111      0.0306281111      0.2946991111
H       1.62654388 9     -0.037692   9      0. 456121111
H       1.32526088 9      0. 030  1111     -0.6 4697   9
H       1.25012388 9     -0.961174   9     -0.61   6   9
H      -0.7580021111     -0.826322   9      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928  9
O      -1.1981291111      0.0180941111     -0.907244   9
user@localhost:~/Documents$
```

- Adding pattern(s), using: `'s/\(OLD\)/MOD\1THIS/VAL'`
  - Around all pattern(s) on each line(s), using: `'s/\(OLD\)/MOD\1THIS/g'`

```
user@localhost:~/Documents$ sed 's/\(\.\)/ \1 /g' Ethanol.xyz
         9

C       1 . 0111998889     -0 . 0452918889     -0 . 0626048889
C      -0 . 4620761111      0 . 0306281111      0 . 2946991111
H       1 . 6265438889     -0 . 0376928889      0 . 8456121111
H       1 . 3252608889      0 . 8030881111     -0 . 6846978889
H       1 . 2501238889     -0 . 9611748889     -0 . 6188868889
H      -0 . 7580021111     -0 . 8263228889      0 . 9315601111
H      -0 . 6822251111      0 . 9536901111      0 . 8665561111
H      -2 . 1126961111      0 . 0649821111     -0 . 6649928889
O      -1 . 1981291111      0 . 0180941111     -0 . 9072448889
user@localhost:~/Documents$
```

Starting from the first `.`, insert spaces around each `.`, note that the dot symbol `.` is a special symbol and must therefore be protected using `\`, the proper writing being then `\.`, see section [Sec. 6.1.1]

  - Around the n^{th} pattern on each line(s), using: `'s/\(OLD\)/MOD\1THIS/n'`

```
user@localhost:~/Documents$ sed 's/\(0\)/\*\1\*/2' Ethanol.xyz
         9

C       1.0111998889     -*0*.0452918889     -0.0626048889
C      -0.462*0*761111      0.0306281111      0.2946991111
H       1.6265438889     -0.*0*376928889      0.8456121111
H       1.3252608889      *0*.8030881111     -0.6846978889
H       1.2501238889     -*0*.9611748889     -0.6188868889
H      -0.758*0*021111     -0.8263228889      0.9315601111
H      -0.6822251111      *0*.9536901111      0.8665561111
H      -2.1126961111      0.*0*649821111     -0.6649928889
O      -1.1981291111      0.*0*180941111     -0.9072448889
user@localhost:~/Documents$
```

Around the second `0` character on each line, insert `*` characters, note that the star symbol `*` is a special symbol and must therefore be protected using `\`, the proper writing being then `\*`, see section [Sec. 6.1.1]

  - Around all, from the n^{th} pattern on each line(s), using: `'s/\(OLD\)/MOD\1THIS/gn'`

```
user@localhost:~/Documents$ sed 's/\(0\)/ \1 /g2' Ethanol.xyz
         9

C       1.0111998889     - 0 . 0 452918889     - 0 . 0 626 0 48889
C      -0.462 0 761111       0 . 0 3 0 6281111       0 .2946991111
H       1.6265438889     -0. 0 376928889       0 .8456121111
H       1.3252608889       0 .8 0 3 0 881111     - 0 .6846978889
H       1.2501238889     - 0 .9611748889     - 0 .6188868889
H      -0.758 0  0 21111     - 0 .8263228889       0 .93156 0 1111
H      -0.6822251111       0 .95369 0 1111       0 .8665561111
H      -2.1126961111      0. 0 649821111     - 0 .6649928889
O      -1.1981291111      0. 0 18 0 941111     - 0 .9 0 72448889
user@localhost:~/Documents$
```

- Performing action(s) on line(s):
  - On a single target line, using: `'VAL'`

```
user@localhost:~/Documents$ sed '3 s/C/F/1' Ethanol.xyz
         9

F       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  - On a the last line, using: `'$'`

```
user@localhost:~/Documents$ sed '$ s/O /Cl/1' Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
Cl     -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  - On a range of lines, using: `'VAL,VBL'`, `VBL` ∈ [`VAL+1`, `$`]

```
user@localhost:~/Documents$ sed '6,9 s/0/ /g3' Ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8 3 881111     - .6846978889
H       1.2501238889     -0.9611748889     - .6188868889
H      -0.7580 21111     - .8263228889       .93156 1111
H      -0.6822251111      0.95369 1111       .8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

- Deleting line(s):
  - Delete n^{nth} line, using: `'VALd'`

```
user@localhost:~/Documents$ sed '3d' Ethanol.xyz
        9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8 3 881111     - .6846978889
H       1.2501238889     -0.9611748889     - .6188868889
H      -0.7580 21111     - .8263228889       .93156 1111
H      -0.6822251111      0.95369 1111       .8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

    To delete the third line.

  - To delete the last line, using: `'$d'`

```
user@localhost:~/Documents$ sed '4,$d' Ethanol.xyz
        9

C       1.0111998889     -0.0452918889     -0.0626048889
user@localhost:~/Documents$
```

  - Delete a range of lines, using: `'VAL,VBLd'`, `VBL` ∈ [`VAL+1`, `$`]

```
user@localhost:~/Documents$ sed '4,8d' Ethanol.xyz
        9

C       1.0111998889     -0.0452918889     -0.0626048889
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

    To delete from line 4 up to line 8.

- Deleting pattern(s), matching line(s) using: `'/PATTERN/d'`

```
user@localhost:~/Documents$ sed '/C/d' Ethanol.xyz
        9

H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

6.8.3 The `awk` command

Find pattern(s) and process the matching line(s):

- **`awk`** *"pattern scanning and processing language"*
  - usage: `awk [OPTION] ... [REGULAR EXPRESSION] [ARG]`
  - ex: `user@localhost:~$ awk '{printf $NF\n}' MyFile`

The `awk` command is used to find patterns and process the lines in which patterns are found, how to output the result(s) being described by the regexp.
Here are some examples of regular expressions for the `awk` filter:

- Print the whole line, nth word on the line, the last word on the line:
  - Print the whole line that was processed, using: `$0`

```
user@localhost:~/Documents$ awk '{print $0}' Ethanol.xyz
        9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  - Print the last word on the line, using: `$NF`

```
user@localhost:~/Documents$ awk '{print $NF}' Ethanol.xyz
9

-0.0626048889
0.2946991111
0.8456121111
-0.6846978889
-0.6188868889
0.9315601111
0.8665561111
-0.6649928889
-0.9072448889
user@localhost:~/Documents$
```

  - Print the nth word on the line, $n \in [1\text{-NF}]$, using: `$`n

```
user@localhost:~/Documents$ awk '{print $1}' Ethanol.xyz
9

C
C
H
H
H
H
H
H
O
user@localhost:~/Documents$
```

- Print and append new line, or, print:
  - Print and append new line, using: `print`

```
user@localhost:~/Documents$ awk '{print $1}' Ethanol.xyz
9

C
C
H
H
H
H
H
H
O
user@localhost:~/Documents$
```

  - Only print, using: `printf`

```
user@localhost:~/Documents$ awk '{printf $1}' Ethanol.xyz
9CCHHHHHHOuser@localhost:~/Documents$
```

- Print more: text, tab, new line, or else, using: `"  "`
  - Print text (between double quotes):

```
user@localhost:~/Documents$ awk '{printf $1" "}' Ethanol.xyz
9  C C H H H H H H Ouser@localhost:~/Documents$
```

    To print first words of each line, each word being separated by spaces.

  - Print tab, using: `"\t"`

```
user@localhost:~/Documents$ awk '{printf $1"\t"}' Ethanol.xyz
9               C       C       H       H       H       H       H       H       O       user@localhost:~/Documents$
```

    To print first words of each line, each word being separated by tabs.

  - Print new line, using: `"\n"`

```
user@localhost:~/Documents$ awk '{printf $1"\n"}' Ethanol.xyz
9

C
C
H
H
H
H
H
H
O
user@localhost:~/Documents$
```

    To print the first words of each line separated by new lines, as `'{print $1}'`.

- To search for pattern(s), using: `'/pattern/'`

```
user@localhost:~/Documents$ awk '/C/' Ethanol.xyz
C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
user@localhost:~/Documents$
```

  This can be combined with print commands:

```
user@localhost:~/Documents$ awk '/C/ {print $2}' Ethanol.xyz
1.0111998889
-0.4620761111
user@localhost:~/Documents$
```

- Counting matching lines with pattern(s), using: `'/pattern/{++var} END'`

```
user@localhost:~/Documents$ awk '/C/{++count} END {print "Count = ", count}' Ethanol.xyz
Count = 2
user@localhost:~/Documents$
```

  To print the number of lines that contains the pattern `C`.

- Conditional search, based of the number of character(s) or word(s) using:

  1) `'length($N)'`, $N \in [0 - \mathrm{NF}]$ or `'NF'`
  2) Test operators: `<`, `>`, `<=`, `>=`, `==`

  – To search for lines that contains *n* characters, using: `'length($0)'`

```
user@localhost:~/Documents$ awk 'length($0) < 5' Ethanol.xyz
9

user@localhost:~/Documents$
```

  To search for lines that contain less than 5 characters.

  – To search for words that contains *n* characters, using: `'length($n)'`

```
user@localhost:~/Documents$ awk 'length($2) == 12' Ethanol.xyz
C       1.0111998889     -0.0452918889     -0.0626048889
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
user@localhost:~/Documents$
```

  To search for lines where the second word contains 12 characters.

  – To search for lines based on the number of words, using: `'NF'`

```
user@localhost:~/Documents$ awk 'NF == 4' Ethanol.xyz
C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~/Documents$
```

  To search for lines that contain 4 words.

- Substitutions, using: `'{$N="New"; print $0}'`, $\mathbf{N} \in [1-\mathrm{NF}]$

```
user@localhost:~/Documents$ awk 'NF == 4 {$1="X"; print $0}' Ethanol.xyz
X 1.0111998889 -0.0452918889 -0.0626048889
X -0.4620761111 0.0306281111 0.2946991111
X 1.6265438889 -0.0376928889 0.8456121111
X 1.3252608889 0.8030881111 -0.6846978889
X 1.2501238889 -0.9611748889 -0.6188868889
X -0.7580021111 -0.8263228889 0.9315601111
X -0.6822251111 0.9536901111 0.8665561111
X -2.1126961111 0.0649821111 -0.6649928889
X -1.1981291111 0.0180941111 -0.9072448889
user@localhost:~/Documents$
```

  If the line has 4 words, substitute the first word by `X`, then print the line.

Here is an example of option for the **awk** filter:

- Specify the field separators, using: **-F**

```
user@localhost:~/Documents$ awk -F "." '{print $1}' Ethanol.xyz
         9

C       1
C      -0
H       1
H       1
H       1
H      -0
H      -0
H      -2
O      -1
user@localhost:~/Documents$
```

  Spaces are the default field separators, using the **-F** allows to adjust this parameter.
  In the previous example the **-F** option is used to define the dots as new field separators.

6.9 Redirections

6.9.1 Sending a job to the background

Foreground job to the background

To retrieve access to the prompt blocked by the last command:

- Ctrl + z , **then** bg *"send foreground job to the background"*
  - usage: Ctrl + z in terminal, followed by: `user@localhost:~$ bg`
  - ex: `user@localhost:~$ bg`

Using the Ctrl + z +bg command:

```
user@localhost:~$ MyScript
```

The MyScript command is blocking the prompt, it should be required for it to stop before inputting new commands.

If press Ctrl + z , then the MyScript command will freeze, and you will get the access back to the prompt, then enter the bg command:

```
user@localhost:~$
user@localhost:~$ bg
```

Job directly to the background

To send directly a job to the background, so that the prompt remains accessible:

- & *"send job to the background"*
  - usage: `[CMD] [OPTION] ... [ARG] &`
  - ex: `user@localhost:~$ gedit &`

Using the & redirection:

```
user@localhost:~$ MyScript &
user@localhost:~$
```

6.9.2 Multiple commands on the same line

To enter multiple commands on the same line, use the `&&` symbols:

- `&&` *"multiple commands on the same line (one after another)"*
  - usage: `[CMD] [OPTION] ... [ARG] && [CMD] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls -l && cd`

Using `&&` commands will be executed in the order their appear on the line:

```
user@localhost:~$ cd Documents && mkdir test && ls && cd
test
user@localhost:~$
```

6.9.3 Redirect standard output in file

Redirect in new file

To redirect the output of a command in a new file (erase existing):

- `>` *"redirect standard output in file (erase existing)"*
  - usage: `[CMD] [OPTION] ... [ARG] > MyFile`
  - ex: `user@localhost:~$ ls -l > MyFile`

Using the `>` redirection:

```
user@localhost:~$ cat ~/Documents/Ethanol.xyz > cat-ethanol.xyz
user@localhost:~$ ls cat-*
cat-ethanol.xyz
user@localhost:~$ cat cat-ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~$
```

Redirect in existing file

To append the output of a command at the end of a file:

- `>>` *"redirect standard output and append at the end of file"*
  - usage: `[CMD] [OPTION] ... [ARG] >> MyFile`
  - ex: `user@localhost:~$ ls -l >> MyFile`

Using the >> redirection:

```
user@localhost:~$ cat ~/Documents/Ethanol.xyz >> cat-ethanol.xyz
user@localhost:~$ ls cat-*
cat-ethanol.xyz
user@localhost:~$ cat cat-ethanol.xyz
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
         9

C       1.0111998889     -0.0452918889     -0.0626048889
C      -0.4620761111      0.0306281111      0.2946991111
H       1.6265438889     -0.0376928889      0.8456121111
H       1.3252608889      0.8030881111     -0.6846978889
H       1.2501238889     -0.9611748889     -0.6188868889
H      -0.7580021111     -0.8263228889      0.9315601111
H      -0.6822251111      0.9536901111      0.8665561111
H      -2.1126961111      0.0649821111     -0.6649928889
O      -1.1981291111      0.0180941111     -0.9072448889
user@localhost:~$
```

Redirect in new file, include error messages

To redirect the output and the potential error(s) of a command in a new file (erase existing):

- **`>& (or &>)`** *"redirect standard output and standard error in file (erase existing)"*
  - usage: `[CMD] [OPTION] ... [ARG] >& MyFile`
  - ex: `user@localhost:~$ ls -l >& MyFile`

Redirect in existing file, include error messages

To append the output and the potential error(s) of a command at the end of a file:

- **`&>>`** *"redirect standard output and standard error and append at the end of file"*
  - usage: `[CMD] [OPTION] ... [ARG] &>> MyFile`
  - ex: `user@localhost:~$ ls -l &>> MyFile`

Redirecting in file and sending to the background

It is obviously possible to both redirect in a file, and send the process to the background:

```
user@localhost:~$ MyScript >& result &
user@localhost:~$
```

In this example both the output and the error of the `MyScript` command are redirected in the file `result` and the job is sent to the background so that the prompt remains available.

6.9.4 Redirecting in another command: the pipe

To redirect the result of a command in another command:

- `|` (called **pipe** or **pipeline**) *"redirect in an other command"*
  - usage: `[CMD] [OPTION] ... [ARG] | [CMD] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls -l | more`
  - ex: `user@localhost:~$ ls -l | grep MyMotif`
  - ex: `user@localhost:~$ ls -l | awk '{printf $NF" "}'`

The pipe is extremely important in using the command line, it allows to combine efficiently commands on a single line.

Using the pipe `|` redirection:

- Single redirection:
  - Example 1:

```
user@localhost:~$ cat ~/Documents/Ethanol.xyz | wc -l
11
user@localhost:~$
```

    The output of `cat` is redirected in the "`wc -l`" command to count lines.

  - Example 2:

```
user@localhost:~$ ls -l | grep 'Doc*'
drwxr-xr-x.  2 user ipcms 4096 avril 13 21:23 Documents
user@localhost:~$
```

    The output of "`ls -l`" is redirected in the `grep` filter that selects line(s) that contain(s) the word described by the regular expression, in this example: `Doc*`.
    When using pipe(s) filters work on the output stream and not on files.

  - Example 3:

```
user@localhost:~$ ls -l | grep '^d'
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Bureau
drwxr-xr-x.  3 user ipcms 4096 avril 13 21:23 Documents
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Images
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Modèles
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Musique
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Public
drwx------.  3 user ipcms 4096 avril 12 18:44 snap
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Téléchargements
drwxr-xr-x.  2 user ipcms 4096 avril 12 18:44 Vidéos
user@localhost:~$
```

    The output of "`ls -l`" is redirected in the `grep` filter that selects the lines that `^d` = that start by `d`

- Multiple redirections:
  - Example 1:

```
user@localhost:~$ ls -l | grep '^d' | awk '{printf $NF" "}'
Bureau Documents Images Modèles Musique Public snap Téléchargements Vidéos
user@localhost:~$
```

    The output of "`ls -l`" is redirected in the **`grep`** filter that selects line(s) that start by **`d`**, and then in the **`awk`** filter that prints the last word on each lines separated by spaces, hence creating the list of all directories in the active directory.

  - Example 2:

```
user@localhost:~$ ls -l | awk '/^d/ {print $NF}' | sed '/snap/d'
Bureau
Documents
Images
Modèles
Musique
Public
Téléchargements
Vidéos
user@localhost:~$
```

    The output of the "`ls -l`" command is redirected in the **`awk`** filter that selects the line(s) that start by **`d`**, and then in the **`sed`** filter that removes the `snap` pattern.

  - Example 3:

```
user@localhost:~$ cat ~/Documents/Ethanol.xyz | sed '1,2d' |
awk '{printf "At=\""$1"\", x=\""$2"\", y=\""$3"\", z=\""$4"\"\n"}'
At="C", x="1.0111998889", y="-0.0452918889", z="-0.0626048889"
At="C", x="-0.4620761111", y="0.0306281111", z="0.2946991111"
At="H", x="1.6265438889", y="-0.0376928889", z="0.8456121111"
At="H", x="1.3252608889", y="0.8030881111", z="-0.6846978889"
At="H", x="1.2501238889", y="-0.9611748889", z="-0.6188868889"
At="H", x="-0.7580021111", y="-0.8263228889", z="0.9315601111"
At="H", x="-0.6822251111", y="0.9536901111", z="0.8665561111"
At="H", x="-2.1126961111", y="0.0649821111", z="-0.6649928889"
At="O", x="-1.1981291111", y="0.0180941111", z="-0.9072448889"
user@localhost:~$
```

    The output of the **`cat`** command is redirected in **`sed`** that removes the first two lines, and then in **`awk`** that prints the content modified. Note that in the last sequence double quotes must be protected to be printed out, using `\"`.

There is no limit to the number of pipes you can redirect your commands in.

Pipe to standard output and file

To redirect the output of a command to both the standard output and a file:

- **`tee`** *"read from standard output and redirect to standard output and file (erase existing)"*
  `Special invocation from a pipe |`
  - usage: `[CMD] [OPTION] ... [ARG] | tee [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls -l | tee MyFile`
  - ex: `user@localhost:~$ cpmd file.inp | tee file.out`

The `tee` command after a pipe output the result of the command that precedes the pipe to both the standard output and a file which name is specified as argument of the `tee` command.

Using the pipe `tee` command:

```
user@localhost:~$ ls -l | tee ls.out
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Bureau
drwxr-xr-x. 3 user ipcms 4096 avril 13 21:23 Documents
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Images
-rw-rw-r--  2 user ipcms 4049 avril 13 21:40 ls.out
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Modèles
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Musique
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Public
drwx------. 3 user ipcms 4096 avril 12 18:44 snap
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Téléchargements
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Vidéos
user@localhost:~$ cat ls.out
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Bureau
drwxr-xr-x. 3 user ipcms 4096 avril 13 21:23 Documents
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Images
-rw-rw-r--  2 user ipcms 4049 avril 13 21:40 ls.out
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Modèles
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Musique
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Public
drwx------. 3 user ipcms 4096 avril 12 18:44 snap
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Téléchargements
drwxr-xr-x. 2 user ipcms 4096 avril 12 18:44 Vidéos
user@localhost:~$
```

6.10 Scripting

Scripting, to write and use scripts, means to program a list of command(s) and action(s) to be performed by the command interpreter in a file. The command interpreter will be able to process this file and the commands that it contains later on.

A BASH script is a basic text file composed of at least 2 lines, and looks like:

```
#!/bin/bash

# This little example illustrates how to say 'Hello' in BASH programming
echo "Hello"
```

The first `#!/bin/bash` tells the system which program to use to run the script, here the BASH command interpreter; the other lines describe the commands to be performed.
Comments can be inserted at any place using the `#` followed by a space.

In BASH like in any other programming language you will find many, somewhat different and somewhat alike, possibilities to achieve the same goal. The examples provided in the previous sections illustrate that easily.

Once the script has been written you have 2 options to use this script as a command:

- Use the **`bash`** command and use the file name as argument, ex:

  ```
  user@localhost:~$ bash MyScript
  ```

- Change the file permissions on the script to make it executable, then use it as a command:

  ```
  user@localhost:~$ chmod 755 MyScript
  user@localhost:~$ ./MyScript
  ```

6.10.1 Numbering on the command line

On the prompt **`command(s)`**, **`option(s)`**, **`argument(s)`** **`redirection(s)`** and **`regular expression(s)`** are separated by empty spaces, and are numbered according to their order of appearance on the line, starting from 0 for the first element:

```
user@localhost:~$ cat MyFile | grep -v 'MyKeyword'
```

On the previous line:

Object	Number
The `cat` command	0
The "`MyFile`" argument	1
The "`|`" redirection, see section [Sec. 6.9.4]	2
The `grep` command	3
The "`-v`" option	4
The `'MyKeyword'` regular expression, see section [Sec. 6.8]	5

6.10.2 Option(s) and argument(s)

It is possible to pass option(s) and/or argument(s) to a script by referring to the number they will appear on the command line (see section [Sec. 6.10.1]), ex:

```
user@localhost:~$ MyScript MyOption1 MyOption2 MyArg
```

Corresponding possible structures of the file `MyScript` that contains command(s):

```
#!/bin/bash

ls -$1 -$2 $3
```

or

```
#!/bin/bash

# Variable declarations using VAR=VAL

# Declaring a variable OPT1, to store the 1st option
OPT1=$1
# Declaring a variable OPT2, to store the 2nd option
OPT2=$2
# Declaring a variable ARG, to store the argument
ARG=$3

ls -$OPT1$OPT2 $ARG
```

If `MyOPtion1`, `MyOption2` and `MyArg` are respectively `l`, `t` and `MyFile` then the script `MyScript` will simply be equivalent to:

```
user@localhost:~$ ls -lt MyFile
```

6.10.3 Examples of scripts

Example 1:

```
#!/bin/bash

# Creating a variable DIRS that contains the space separated list
# of directory(ies) in the active directory:
DIRS=`ls -l | grep '^d' | awk '{printf $NF" "}'`
# In order to redirect the result(s) of command(s) in a variable
# it is required to put the expression in between use ` ` symbols

# Then display the list:
echo $DIRS
```

Example 2:

```
#!/bin/bash

# This script receives an argument, a given directory to go to:
cd $1

# Creating a variable DIRS that contains the space separated list
# of directory(ies) in this given directory:
DIRS=`ls -l | awk '/^d/ {printf $NF" "}'`

# Then display each element of the list:
for DIR in $DIRS
do
  echo $DIR
done
```

For more examples check my Basic tutorial to BASH programming.

6.10.4 The configuration file "~/.bashrc"

As already mentioned in section [sec. 6.1.1] this text file is read by BASH every time that a shell starts. Then BASH executes the command(s) in ∼/.bashrc to prepare, set up, the next command line session.
Consequently ∼/.bashrc is like a script file that can be used to fine tune the behavior of the command line.
Among the interesting things to tune:

Modifying the PATH variable

Let us consider that you want to include a new directory in the **PATH**, so that commands inside this directory could be located easily by the command interpreter.

```
user@localhost:~$ ls bin
MyScript   MyCommand   MyApp
user@localhost:~$
```

The directory ~/bin contains 3 files, and these files have the execution permission, meaning that they can be used as commands.
However for the timing being to use the **MyScript** command it is required to do either:

```
user@localhost:~$ ./bin/MyScript
```

or

```
user@localhost:~$ /home/user/bin/MyScript
```

or

```
user@localhost:~$ cd bin
user@localhost:~$ ./MyScript
```

You can simply this by editing the ~/.bashrc as follow:

```
# Creating a new PATH variable, that contains:
# - 1) the preserved content of the existing PATH variable "$PATH"
# - 2) adding the new ~/bin directory, in the PATH format ":DIR"
PATH=$PATH:~/bin
```

Creating aliases

Aliases are user made commands inserted in the file `~/.bashrc`.
The aliases behave like commands in the **`PATH`** and are therefore available directly once the shell has started, also they have priority over already existing command(s) and will therefore be executed instead. This last feature is particularly interesting to modify the behavior of already existing commands.

To create an alias use the following syntax:

```
alias NAME='WHAT TO EXECUTE'
```

Convenient aliases to insert in your file `~/.bashrc`:

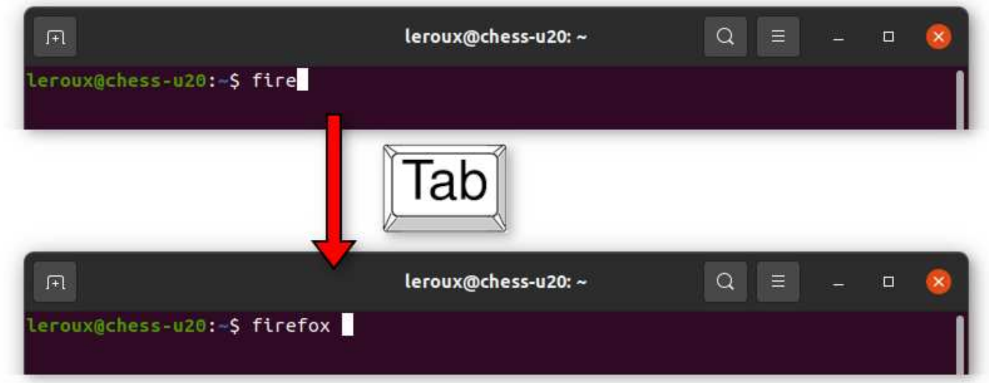
```
# Shortcuts to the "ls" options
alias ll='ls -lh'
alias la='ls -ah'
alias lt='ls -th'
alias lla='ls -lah'
alias llt='ls -lth'
alias llta='ls -ltah'

# Always request confirmation before deleting something "rm -i"
alias rm='rm -i'

# Always use color(s) with "diff"
alias diff='diff --color=always'

# Always use color(s) with "grep"
alias grep='grep --color=always'
```

6.11 Command line tips and tricks

6.11.1 Auto-completion

The auto-completion, or completion, is a command line tips. When you enter commands, if the name of the command is partially entered (written in the terminal), then if your press the ↹ / tab on the keyboard then Linux will try to complete the instruction/line for you:

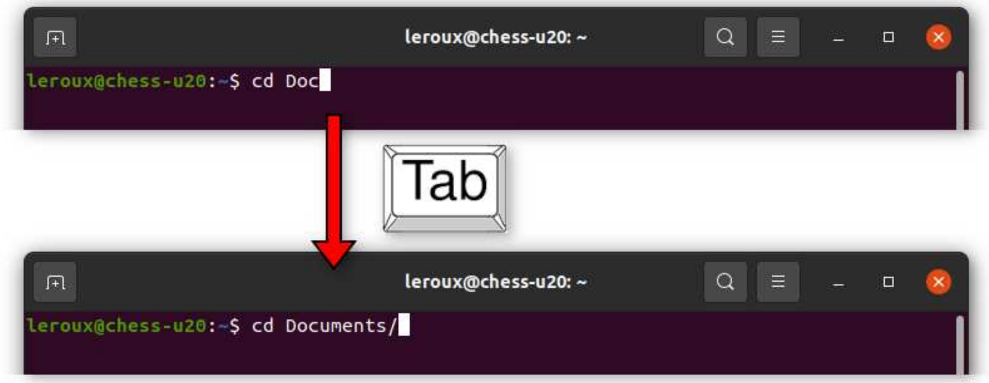
- If there is only one possibility the system will complete the command / line:
  - For a command:



```
leroux@chess-u20:~$ fire
```

```
leroux@chess-u20:~$ firefox
```

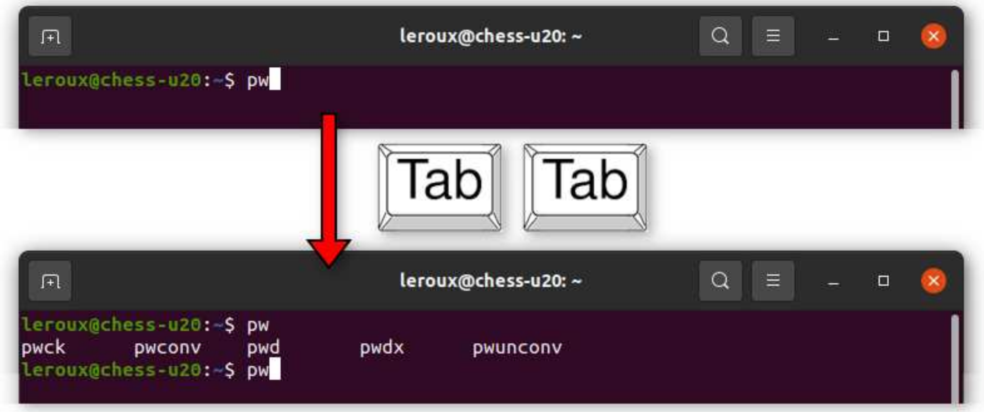
  - For a file or a path:



```
leroux@chess-u20:~$ cd Doc
```

```
leroux@chess-u20:~$ cd Documents/
```

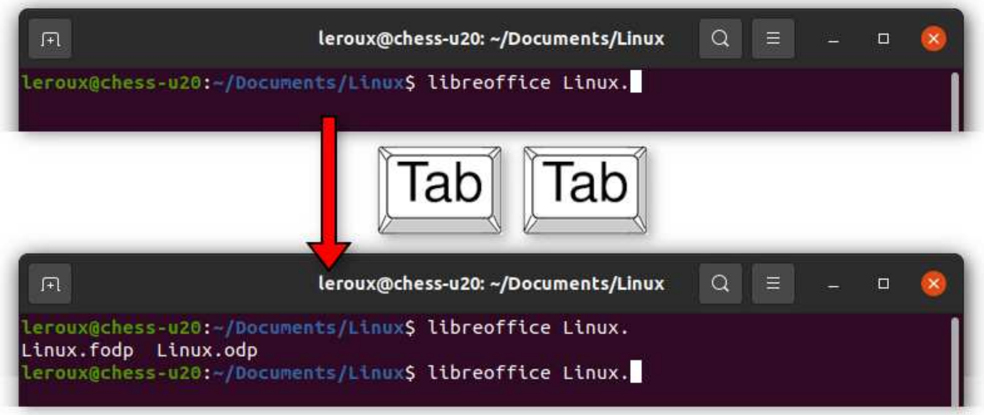
- Otherwise, if there are many options, press ⟷ / tab a second time the system will present the options available to complete the command / line:
  - For a command:



```
leroux@chess-u20:~$ pw
```

```
leroux@chess-u20:~$ pw
pwck        pwconv      pwd         pwdx        pwunconv
leroux@chess-u20:~$ pw
```

  - For a file or a path:


leroux@chess-u20: ~/Documents/Linux
Tab
Tab
leroux@chess-u20: ~/Documents/Linux

```
leroux@chess-u20:~/Documents/Linux$ libreoffice Linux.
```

```
leroux@chess-u20:~/Documents/Linux$ libreoffice Linux.
Linux.fodp  Linux.odp
leroux@chess-u20:~/Documents/Linux$ libreoffice Linux.
```

6.11.2 Tips and tricks

Change directory to the user home directory:	`cd`
Change to the previous directory where you are coming from:	`cd -`
Change to the upper directory in the system tree:	`cd ..`
Move at the beginning of the line:	Ctrl + a
Move at the end of the line:	Ctrl + e
Kill a task blocking the command line:	Ctrl + c
To display the last command:	Ctrl + p
To use the last command:	**!!**, followed by Enter
To use the n^{th} last command:	**!**$-n$, followed by Enter
Previous commands:	
- See :	Page ↑
- Navigate :	Page ↓
- Execute :	Enter

Table 6.1: Tips and tricks for the command line.

Linux drawbacks and advantages

7.1 Drawbacks

None, the only issue that you can possibly face with Linux concerns video games.
You can play with Linux, but the most recent games will likely not be available.

7.2 Advantages

There are so many advantages in using Linux:

- **It's free !**

  Yes that sounds easy but computer manufacturer actually sold Linux equipped laptop or workstation. The difference compared to the same equipment with MS Windows: it's a 100 € cheaper !

- **No need to change your computer to use the newest Linux !**

  The usual method that GAFAM (Google, Apple, Facebook, Amazon, Microsoft) use to justify for you to change your equipment: a new version of their OS.
  Materials are more and more expensive, financially, but even more ecologically, so why change your equipment, why consume more ?
  Why buy a new computer, if the actual one runs fine, just to get the new MS windows ?
  Just to make you think about it, how long should you use your computer so that the CO_2 footprint of its usage (from energy production, between 1.8 kg CO_2/years in France up to 22 kg CO_2/year in Africa or Asia [2]) becomes as large of the CO_2 footprint of its production, between 250 and 500 kg CO_2) ?

  `In France at best:` $\frac{250}{1,8} = 139$ `years !!!`

- **High security: no viruses ... no viruses at all !**

- **High stability:** the world best servers use Linux, you use Linux everyday on your smartphone to check your bank account, why don't you use it on your computer ?
- **Ease of use:** that is particularly true with the Ubuntu distro prepared for beginners ... not to mention that most of you are already using Linux on their smartphone every day ;-)
- **It's Free = Open Source: the code source is available !!!**

  This is probably the most important reason why you should use Linux, the code source is available. The Linux ecosystem mostly relies on free software, therefore almost all code source are available. That means that the code can be checked, read, by any developer, ensuring the program does what it should.
  In the digital age do you prefer to entrust your privacy to a proprietary (often GAFAM own) software, or to an open source program ?

Command Glossary

8.1 Standard commands

- **`man`** *"call for help - manual pages"*
  - usage: `man [ARG]`
  - ex: `user@localhost:~$ man ls`

- **`ls`** *"list content"*
  - usage: `ls [OPTION]` ... `[ARG]`
  - ex: `user@localhost:~$ ls`
  - options: `-l` (detailed list), `-t` (time sort), `-a` (show hidden files), `-h` (human readable)

- **`pwd`** *"print working directory"*
  - usage: `pwd`
  - ex: `user@localhost:~$ pwd`

- **`cd`** *"change directory"*
  - usage: `cd [ARG]`
  - ex: `user@localhost:~$ cd MyDirectory`

- **`touch`** *"change time stamp"*
  - usage: `touch [ARG]`
  - ex: `user@localhost:~$ touch MyFile`

- **`mv`** *"move"*
  - usage: `mv [ARG1] [ARG2]`
  - ex: `user@localhost:~$ mv MyFile MyNewFile`

- **`cp`** *"copy"*
  - usage: `cp [ARG1] [ARG2]`
  - ex: `user@localhost:~$ cp MyFile MyNewFile`

- `mkdir` *"create directory"*
  - usage: `mkdir [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ mkdir MyNewDirectory`
  - options: `-p` (with parents)

- `rmdir` *"remove empty directory"*
  - usage: `rmdir [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ rmdir MyOldDirectory`
  - options: `-p` (with parents), `-r` (recursive), `-f` (force)

- `rm` *"remove"*
  - usage: `rm [OPTION] ... [ARG]...`
  - ex: `user@localhost:~$ rm MyFile`

- `df` *"display information on file system"*
  - usage: `df [OPTION] ...`
  - ex: `user@localhost:~$ df -h`
  - options: `-h` (human readable)

- `du` *"disk usage"*
  - usage: `du [OPTION] ... [ARG] ...`
  - ex: `user@localhost:~$ du -sh MyFile`
  - options: `-s` (total size), `-h` (human readable)

- `cat` *"display in standard output"*
  - usage: `cat [ARG]`
  - ex: `user@localhost:~$ cat MyFile`

- `tac` *"opposite of `cat`"*
  - usage: `tac [ARG]`
  - ex: `user@localhost:~$ tac MyFile`

- `more` *"display in standard output screen by screen"*
  - usage: `more [ARG]`
  - ex: `user@localhost:~$ more MyFile`

- `less` *"advanced `more`, allows to navigate and search"*
  - usage: `less [ARG]`
  - ex: `user@localhost:~$ less MyFile`

- `clear` *"clean the terminal"*
  - usage: `clear`
  - ex: `user@localhost:~$ clear`

- `echo` *"display something"*
  - usage: `echo [ARG]`
  - ex: `user@localhost:~$ echo`
- `cut` *"cut, extract columns"*
  - usage: `cut [OPTION]` ... `[ARG]`
  - ex: `user@localhost:~$ cut -c5-10 MyFile`
  - options: `-c` (character numbers: `-cA-B,C-D,E-F...`)
- `tail` *"show me the tail"*
  - usage: `tail [OPTION]` ... `[ARG]`
  - ex: `user@localhost:~$ tail -f MyFile`
  - options: `-f` (last ten lines with update), `-lines=n` (last n lines)
- `wc` *"words and lines count"*
  - usage: `wc [OPTION]` ... `[ARG]`
  - ex: `user@localhost:~$ wc -l MyFile`
  - options: `-l` (number of lines)
- `ln` *"create a link"*
  - usage: `ln [OPTION]` ... `[ARG1] [ARG2]`
  - ex: `user@localhost:~$ ln -s MyFile MyLink`
  - options: `-s` (create symbolic link)
- `chmod` *"change permissions"*
  - usage: `chmod [OPTION]` ... `[ARG]`
  - ex: `user@localhost:~$ chmod 755 MyFile`
  - options: `xyz` with `x`, `y` and `z` between 0 and 7, `-R` (recursive)
- `chown` *"change owner"*
  - usage: `chown [OPTION]` ... `[ARG1] [ARG2]`
  - ex: `user@localhost:~$ chown user.group MyFile`
  - options: `-R` (recursive)
- `ps` *"process information"*
  - usage: `ps [OPTION]` ...
  - ex: `user@localhost:~$ ps -auwx`
  - options: `-ax` (all processes), `-u` (user oriented format), `-w` (wide output)
- `top` *"system usage"*
  - usage: `top [OPTION]` ...
  - ex: `user@localhost:~$ top`
- `kill` *"terminate process"*
  - usage: `kill [OPTION]` ... `[ARG]` ...
  - ex: `user@localhost:~$ kill -9 PID`
  - options: `-9` (send kill signal)

- `su` *"switch user"*
  - usage: `su [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ su - user`
  - options: `-` (use shell login)

- `sudo` *"run command as root"*
  - usage: `sudo [OPTION] ... [CMD] [OTION] ... [ARG]`
  - ex: `user@localhost:~$ sudo ls -l MyDirectory`

- `env` *"know you environment"*
  - usage: `env`
  - ex: `user@localhost:~$ env`

- `diff` *"difference(s) between two files"*
  - usage: `diff [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ diff File-1.dat File-2.dat`
  - options: `-color=always` (to color the output)

- `sleep` *"wait some time please"*
  - usage: `sleep [TIME]`
  - ex: `user@localhost:~$ sleep 3600`

- `which` *"check $PATH to know which binary is used"*
  - usage: `which [ARG]`
  - ex: `user@localhost:~$ which cpmd`

- `whereis` *"find out the location of binary file(s), library file(s) and manual page(s)"*
  - usage: `whereis [ARG]`
  - ex: `user@localhost:~$ whereis gimp`

- `tar` *"create archive"*
  - usage: `tar [OPTION] ... [ARG1] [ARG2] ...`
  - ex: `user@localhost:~$ tar -jcf MyFile.tar.bz2 MyFile`
  - options: `-c` (create archive), `-x` (extract archive), `-f` (filename), `-j` (bzip2), `-z` (gzip), `-t` (list file in archive)

- `find` *"look for a file"*
  - usage: `find [PLACE] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ find /home -name "MyFile"`
  - options: `-name` (name is)

- `passwd` *"change the password"*
  - usage: `passwd [OPTION] ... [USERNAME]`
  - ex: `user@localhost:~$ passwd`

8.2 Redirection commands

- `&` *"send job to the background"*
  - usage: `[CMD] [OPTION] ... [ARG] &`
  - ex: `user@localhost:~$ gedit &`

- `Ctrl` + `z` `, then bg` *"send foreground job to the background"*
  - usage: `Ctrl` + `z` in terminal, followed by: `user@localhost:~$ bg`
  - ex: `user@localhost:~$ bg`

- `&&` *"multiple commands on the same line (one after another)"*
  - usage: `[CMD] [OPTION] ... [ARG] && [CMD] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls -l && cd`

- `>>` *"redirect standard output and append at the end of file"*
  - usage: `[CMD] [OPTION] ... [ARG] >> MyFile`
  - ex: `user@localhost:~$ ls -l >> MyFile`

- `>& (or &>)` *"redirect standard output and standard error in file (erase existing)"*
  - usage: `[CMD] [OPTION] ... [ARG] >& MyFile`
  - ex: `user@localhost:~$ ls -l >& MyFile`

- `&>>` *"redirect standard output and standard error and append at the end of file"*
  - usage: `[CMD] [OPTION] ... [ARG] &>> MyFile`
  - ex: `user@localhost:~$ ls -l &>> MyFile`

- `| (called pipe or pipeline)` *"redirect in an other command"*
  - usage: `[CMD] [OPTION] ... [ARG] | [CMD] [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls -l | more`
  - ex: `user@localhost:~$ ls -l | grep MyMotif`
  - ex: `user@localhost:~$ ls -l | awk '{printf $NF" "}'`

- `tee` *"read from standard output and redirect to standard output and file (erase existing)"*
  `Special invocation from a pipe |`
  - usage: `[CMD] [OPTION] ... [ARG] | tee [OPTION] ... [ARG]`
  - ex: `user@localhost:~$ ls -l | tee MyFile`
  - ex: `user@localhost:~$ cpmd file.inp | tee file.out`

8.3 Bash commands

- `alias` *"create command alias"*
  - usage: `alias NEW_CMD='[CMD] [OPTION]... [ARG]'`
  - ex: `user@localhost:~$ alias ll='ls -lth'`

- `ENVIRONMENT_VARIABLE=` *"create environment variable"*
  - usage: `ENVIRONMENT_VARIABLE=[VALUE]`
  - ex: `user@localhost:~$ MYVAR=100`

- `export ENVIRONMENT_VARIABLE=` *"create inheritable environment variable"*
  - usage: `export ENVIRONMENT_VARIABLE=[VALUE]`
  - ex: `user@localhost:~$ export MYVAR=100`

8.4 Filter commands

- `grep` *"print lines matching a pattern"*
  - usage: `grep [OPTION] ... [REGULAR EXPRESSION] [ARG]`
  - ex: `user@localhost:~$ grep "TOTAL ENERGY =" cpmd.out`
  - options: `-n` (print line number), `-A NUM` (print `NUM` lines after pattern), `-B NUM` (print `NUM` lines before pattern), `-v` (invert correspondence = non-matching lines)

- `sed` *"stream editor for filtering and transforming text"*
  - usage: `sed [OPTION] ... [REGULAR EXPRESSION] [ARG]`
  - ex: `user@localhost:~$ sed 's/MYPATH/$HOME/g'`

- `awk` *"pattern scanning and processing language"*
  - usage: `awk [OPTION] ... [REGULAR EXPRESSION] [ARG]`
  - ex: `user@localhost:~$ awk '{printf $NF\n}' MyFile`

Bibliography

[1] Minnesota Supercomputer Center (1993). (Cité page 77.)

[2] The Shift Project. Déployer la sobriété numérique (2020). (Cité page 113.)

This document has been prepared using the Linux operating system and free software:

The text editor	"gVim"
The vector graphics editor	Inkscape
The GNU image manipulation program	"GIMP"
And the document preparation system	"$\LaTeX\ 2_{\varepsilon}$"