

Practice 1: Introduction to Linux (1 session)

Week of **September 9th, 2024**

Objectives:

- Acquire basic concepts about the use of the Linux operating system.
- Acquire elementary knowledge about editing, compiling and running programs in C.

1. Introduction to Linux

Currently there are many operating systems and with different characteristics. Three of the most widely used operating systems today are WINDOWS, UNIX and Mac OS X. We will work with LINUX Ubuntu which is freely distributable.



LINUX is an Operating System based on UNIX and therefore quite different internally from WINDOWS, but very similar in terms of the philosophy of management as an end user. In this practice we will work with LINUX and make comparisons and references to WINDOWS.

Ubuntu is what is called a **LINUX distribution**. A distribution is nothing more than a selection of certain programs and a certain graphical environment, which are packaged in a DVD or in a file downloadable over the Internet. Not everyone has the same tastes or uses their computer for the same tasks, so there are distributions oriented to multimedia, programming, games, and others. Ubuntu is geared towards ease of use for the non-computer user and is one of the most popular in the LINUX world.

At present there are a large number of distributions (Ubuntu, Debian, Red Hat, Fedora, Mint, etc.), each of them created to meet specific needs and with a specific objective, such as ease of use, security, use by a specific group, etc ...



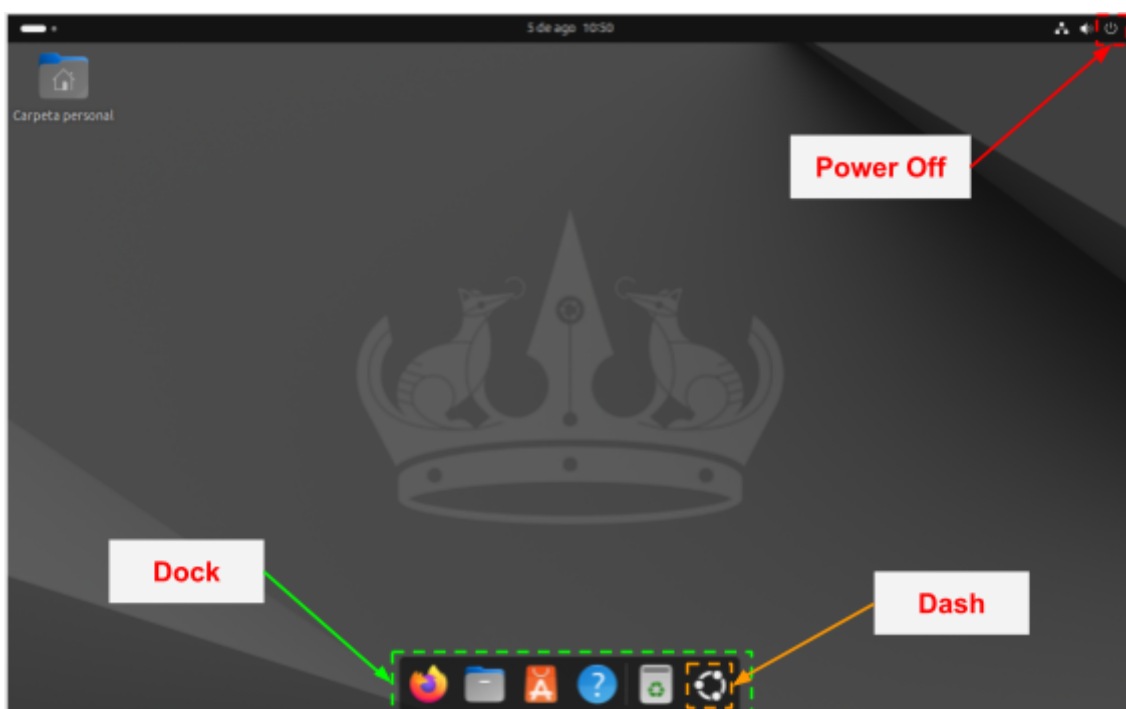
1.1. How to start a work session

When you turn on the PC, and once the BIOS checks have been completed, we will select the operating system from which we want to boot. On the computers of the laboratories it is possible to work in WINDOWS or LINUX.

When the computer is turned on, it requests the OS with which we want to work. In our case, we will choose LINUX.

When LINUX starts, we will find ourselves in front of its graphical environment. In order to be able to use the computer we may be asked to enter user and password. In this case, we will use the same user and password we use to access UACloud.

This academic year **Ubuntu 22.04 LTS** is installed on the computers of the laboratories of the EPS. The user environment in this release looks something like this:



1.2. The graphical environment of Ubuntu LINUX

Like WINDOWS, Ubuntu has a graphical environment with which all operations can be performed "at the click of a mouse". The desktop is what appears on the screen and consists of several elements:

- **Launcher or Dock:** In the version of Ubuntu installed in the labs, the *Dock* has been enabled as the application launcher, whereas in the default installation the application launcher appears on the left side of the screen. The *Dock* is the button bar at the bottom of the screen. It has icons for accessing frequently used programs. If you have a Mac you will see that it is very similar to the *dock*. With the rightmost button we can activate the **Dash**, which is used to search for other programs and files on our machine.
- **Top panel:**
 - On the left hand side is the button to switch desktops. Ubuntu provides two virtual desktops by default, so we can have some applications opened in one virtual desktop and other applications opened in the other one, this feature allows 'cleaner' working environments.

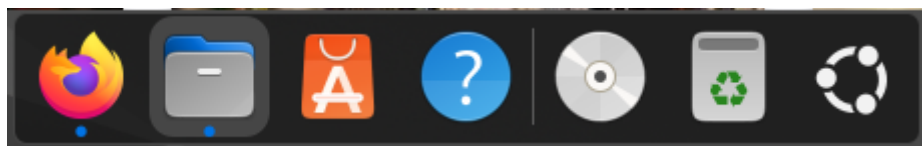
- In the center of the panel is the time and date.
- On the right side is the area of system indicators (network, volume, etc.). By clicking on this area we can access all the indicator controls, including the button to turn off the machine.

As it happens in Windows, the windows of the open applications are shown on the desktop.

1.3. Run programs

You can **launch a program by clicking on its icon in the launcher**. For example, **try launching Firefox, the internet browser**.

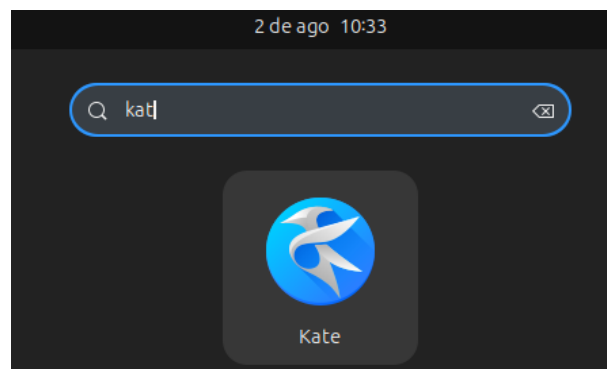
You will see that in the docker a dot has appeared under the Firefox icon. This dot indicates that the Firefox application is open. If you now click on the folder icon, you will start the file browser. You will then see that a dot also appears under the folder icon. To bring a program to the foreground, just click on its icon, but you can also use the keyboard shortcut Alt+TAB, just like in Windows.



You probably have too many programs installed on your computer to fit all in the launcher, so there is a button that serves to search among all the programs (and in general among all the files) that you have. It is the last button on the right of the Dock, the so-called **Dash**, which has the Ubuntu icon.



If you click on this button, a text box will appear where you can type the name of what you are looking for. For example, you can try searching for the Kate text editor. As you type the letters you will see the icons of the programs and files that match what you have written. To start the program or open the file, just click on it.

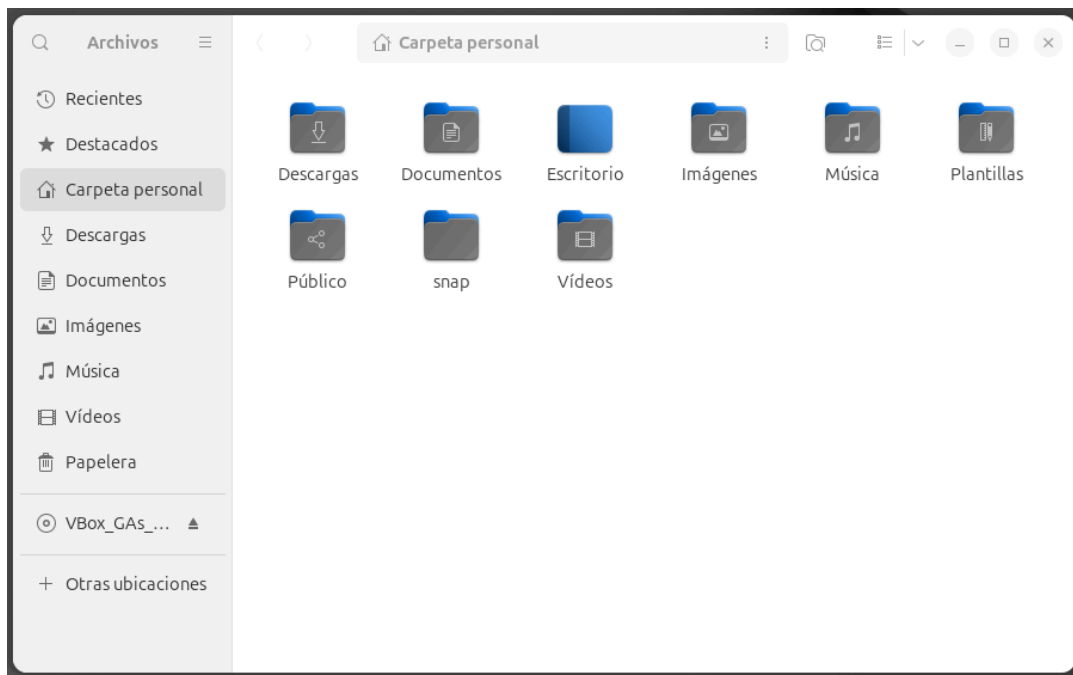


1.4. Storing data on disk (folders and files)

As in WINDOWS, and in the vast majority of current operating systems, in LINUX our data is stored in files, and classified in folders or directories. All this information is stored on the hard drives of our computer and on external media, USB type.

In WINDOWS you can access your disks and files through the "My Computer" icon, using the file explorer. **In Ubuntu, as you have already seen, you can do it with the folder icon that appears in the Dock.** By default you will be in your personal folder (in Windows there is also a user's personal folder). As you can

see, you have subfolders for documents, Internet downloads, images, etc., however, in the EPS computers most of the folders will be empty or with only sample data.



In the Linux of the EPS computers you cannot store information permanently. **When you turn off the computer and turn it back on, all the data you created during the session will have been erased.** That is why it is important that you always bring a USB flash drive or similar to save the programs you are doing in class, or you can also save them in some other way (Dropbox, Drive, etc.).

Linux's security restrictions are by default much more rigid than in Windows, so **you can usually only create, move, and delete files and folders within your personal folder**, but not outside of it. Note that the desktop is inside your personal folder, as you can see if you open it in the file browser.

Of course you can also copy and move files and create directories with the mouse just like you would in Windows.

A comparison between Linux and Windows file systems

You probably know that in WINDOWS disk drives have letters assigned to them. Thus, the first hard disk would be the C: drive and if you have more hard drives installed (or you have several partitions or removable USB disks) they would be assigned the D:, E:, etc.

In WINDOWS, certain directories within the hard disk have a special meaning, for example "My Documents", or "Program Files". In LINUX/UNIX the same thing happens, except that the names are a little more cryptic, for example the approximate equivalent to "Program Files" would be "/usr", and "My Documents" would be "/home/your_username/Documents".

1.4.1. Storage on external media (USB stick)

To copy files from the computer's hard drive to an external drive, the file browser mentioned in the previous section is also used. It allows you to copy files in both directions. When you connect a USB disk, its icon will automatically appear on the desktop and also within the file browser, along with the rest of the disk drives.

As in Windows it is convenient that if we copy information to a USB memory, before disconnecting it from the computer we will make sure that the data has already been copied. To do this, on the desktop we access the context menu of the USB device (by clicking with the right mouse button) and choose the option **Safely remove drive**. Notice that the USB stick disappears from the desktop (in UNIX/LINUX slang it is said that the drive has been "unmounted").

1.5. Users and permissions

Linux is a multi-user system so it is necessary the (secure) administration of the different users who are going to make use of the system resources. In Linux we can find 2 types of users:

- **Superuser (or root)**: This will be the system administrator. This user can do everything, in principle there are no restrictions for him.
- **Final users**: These users will have more or less privileges and will make use of system resources. They can use some programs and applications and have a working directory.

If we want to perform any system administration task, we must do it as root.

Linux has a more restrictive security policy than other operating systems. So not all users will be able to see all the files, or modify them or run a certain application.

The permission system in Linux is based on a scheme of users and groups. Thus, each of the users (or groups) is assigned rights (or permissions) on the files and directories.

This is one of the features that helps Linux to be known as an operating system more immune to the viruses that we can find on computers. Since viruses must be able to write a file to infect it, with the Linux permission system, viruses cannot be copied to any file.

All files and directories in Linux have permissions that check who can or cannot do some action with it.

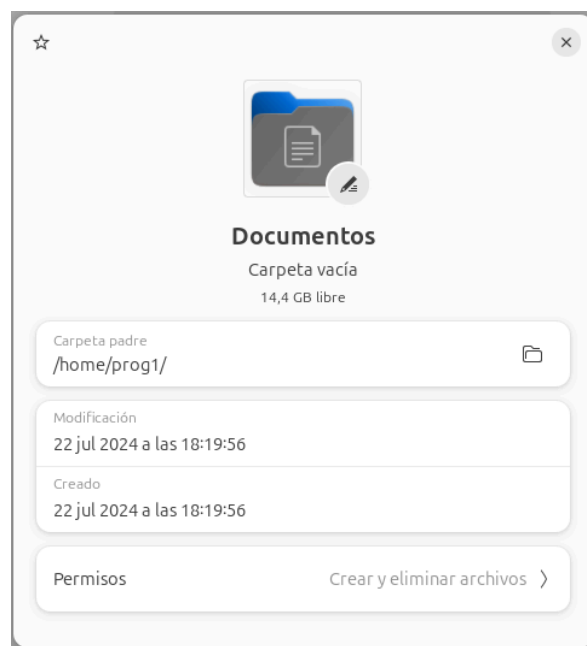
On Linux, each file and directory has a number of permissions. Permissions determine what type of access a user can have to a file or folder. There are three types of permissions:

- **Read permission**: Allows you to read the contents of a file or list the contents of a folder.
- **Write permission**: Allows you to modify the contents of a file and create or delete the files in a folder.
- **Execution permission**: Allows you to run binary files or use the folder to create a valid execution path.

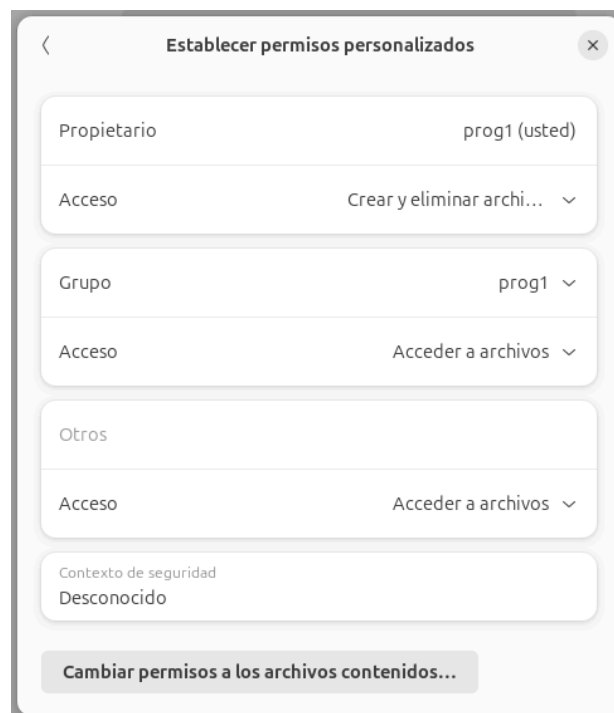
In addition, these 3 permissions (read, write, execute) can be set on a file or folder to:

- **The owner**: The owner of the file. The user who created the file or folder.
- **The group** to which the owner belongs.
- **Others**: Other users who do not belong to the same group to which the user belongs.

To see the permissions that a certain file or folder has, we must right-click on it and select the Properties option.



In the window that appears we can see the complete path of the folder in which the selected file or folder is saved, the date of last modification and the date of creation. To see the permissions assigned to it, click on the 'Permisos' option that appears at the bottom of the window:

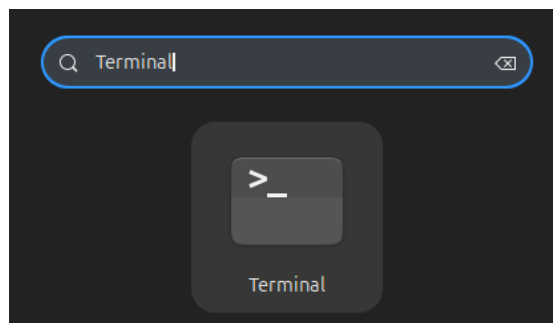


In the image above we see an example of the permissions of a folder, and how it separates the permissions according to whether they are for the owner of the file, for the group or for others. If our user has write permission on this file, we can modify the different permissions.

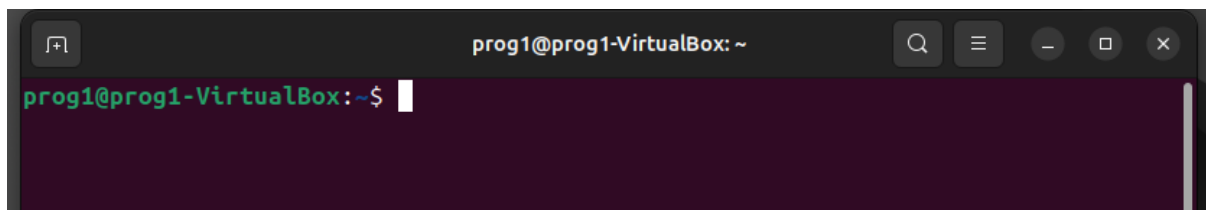
1.6. Opening a terminal

Although the graphical environment is very comfortable and intuitive, there are many tasks that, once accustomed, are performed more quickly by typing directly commands to the system. Orders are typed in a special window called **Terminal**.

To open a terminal, click on the dash icon, type 'Terminal' in the text box to search for the application and then click on the terminal icon.



A window appears with the \$ symbol (called the command **prompt**). This symbol indicates that the system is ready to receive your orders.



Typically, the shell will appear as follows: user_name@computer_name, followed by \$ or #. The \$ symbol means that the user is a regular user, while the # symbol indicates that the user is 'root' (the administrator user and therefore with privileges to make changes).

Most UNIX/LINUX commands are abbreviations of two and three rather cryptic letters, which are very quick to type but not very intuitive for the novice user. However, you can try some:

\$ date

(This command will tell you the date and time of the system)

\$ pwd

(This command will tell you which directory you are currently working on. By default it is your home directory, /home/user_name)

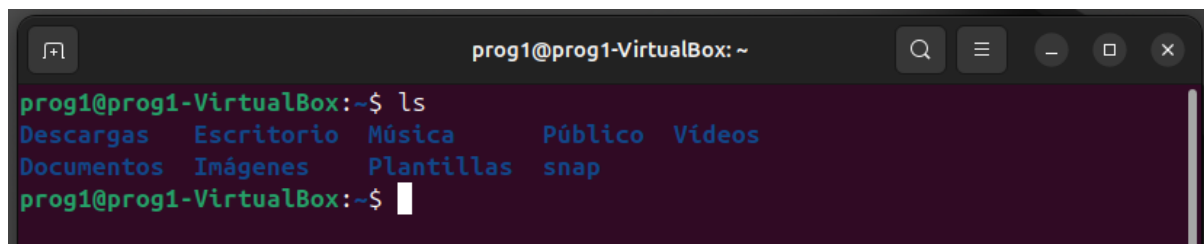
\$ ls

(This command will list the contents of the directory you are currently working in.)

In LINUX we have available the commands that we have been typing by pressing the cursor up, to go back, and pressing the cursor down, to go forward. In addition, you can edit the current line by simply pressing the cursor left and right and taking the cursor to the place of the line that you want to modify, deleting and/or adding new characters.

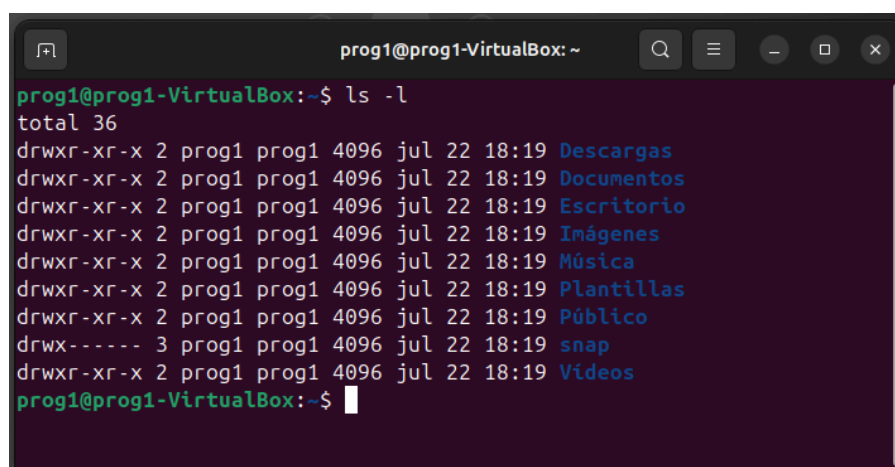
ls command

List the contents of the directory in which we are.



```
prog1@prog1-VirtualBox: ~  
prog1@prog1-VirtualBox:~$ ls  
Descargas  Escritorio  Música      Público  Videos  
Documentos Imágenes   Plantillas  snap  
prog1@prog1-VirtualBox:~$
```

Commands can have a number of options that vary the behavior of the command somewhat. For example, one of the most commonly used options of the ls command is the -l option.



```
prog1@prog1-VirtualBox:~$ ls -l  
total 36  
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Descargas  
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Documentos  
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Escritorio  
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Imágenes  
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Música  
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Plantillas  
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Público  
drwx----- 3 prog1 prog1 4096 jul 22 18:19 snap  
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Videos  
prog1@prog1-VirtualBox:~$
```

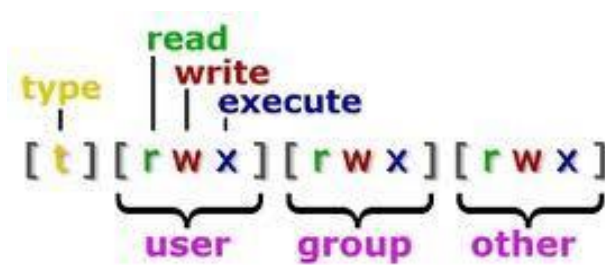
This command lists the contents of the directory with the properties of the different files or folders it contains.

Starting from right to left it shows:

- Name of the file or directory
- Time
- Date
- Size
- The group it belongs to
- The owner of the document or directory
- Access permissions

The first part of each line is a bit more complicated. Let's take a closer look.

- The first letter indicates the file type. Directories have a **d** as their first letter.
- In the Users and Permissions section we have seen how files can have 3 types of permissions (read, write and execute) for the owner, for the group and for the rest of users. The first 3 letters indicate the read, write, and execute permission for the owner, the next 3 indicate the permissions for the group, and the last 3 indicate the permissions for all other users.



If we look at the "Plantillas" permissions in the first line we have obtained we see that:

```
drwxr-xr-x 2 prog1 prog1 4096 jul 22 18:19 Plantillas
```

- Is a directory
- The owner has read, write, and execute (rwx) permission.
- The group has read/execute permission only (r-x).
- All other users have only read and execute permissions (r-x).

man command

This command is the system help. It will help us to know the correct syntax and the options of the other commands. For example, if we want to know the syntax and the rest of the options of the ls command, described above, we would type:

```
prog1@prog1-VirtualBox: ~$ man ls
```

cd command

We will use this command to move from one directory to another. Its syntax is:

`cd path/to/the/directory`

Within a directory, there are 2 special directories "." and "..".

The "." refers to the current directory. While ".." refers to the parent directory. For example, if we type:

`$ cd .`

We would stay in the current directory. And if we are located in the directory /home/prog1 and type:

`$ cd ..`

It would take us to the parent directory of the current directory, i.e. to /home.

rm command

The rm command allows us to delete files or directories. Let's look at some examples. To delete a file we would write:

`$ rm helloWorld.c`

This command would delete the file called helloWorld.c.

This command has a number of options. The most commonly used option is -r. This option recursively deletes a directory. That is, it would delete the directory including all its files and subdirectories. For example:

```
$ rm -r practice1
```

You would delete the practice1 directory with all the files it contained and subdirectories.

Typically, the command prompts the user before deleting any protected files that contain the directory.

Another option used for this command is -f. This option causes all files in a directory to be deleted without prompting. For example:

```
$ rm -rf practice1
```

I would delete the practice1 directory with all its contents without asking anything, even if there were protected files.

If we wanted to delete all the files and directories of the folder in which we are. Empty the folder completely. We would type:

```
$ rm -rf *
```

The * symbol refers to the entire contents of the current directory.

mkdir command

The * symbol refers to the entire contents of the current directory.

```
$ mkdir practice1
```

mv command

To rename a file or directory we will use the mv command.

```
$ mv practice1 prac1
```

The above command would rename the file practice1 to prac1.

sudo command

If we want to use a command as root because it requires more permissions than our user has, we will use the sudo command in front of the command.

For example, if we want to see the partitions that the computer's hard drive has, we will have to use the fdisk -l command.

Enter in the Terminal:

```
$ fdisk -l
```

What happened? Nothing has happened. The system hasn't done anything because we don't have permissions to run that command.

Instead, enter in the Terminal:

```
$ sudo fdisk -l
```


And when prompted for your password, enter your password.

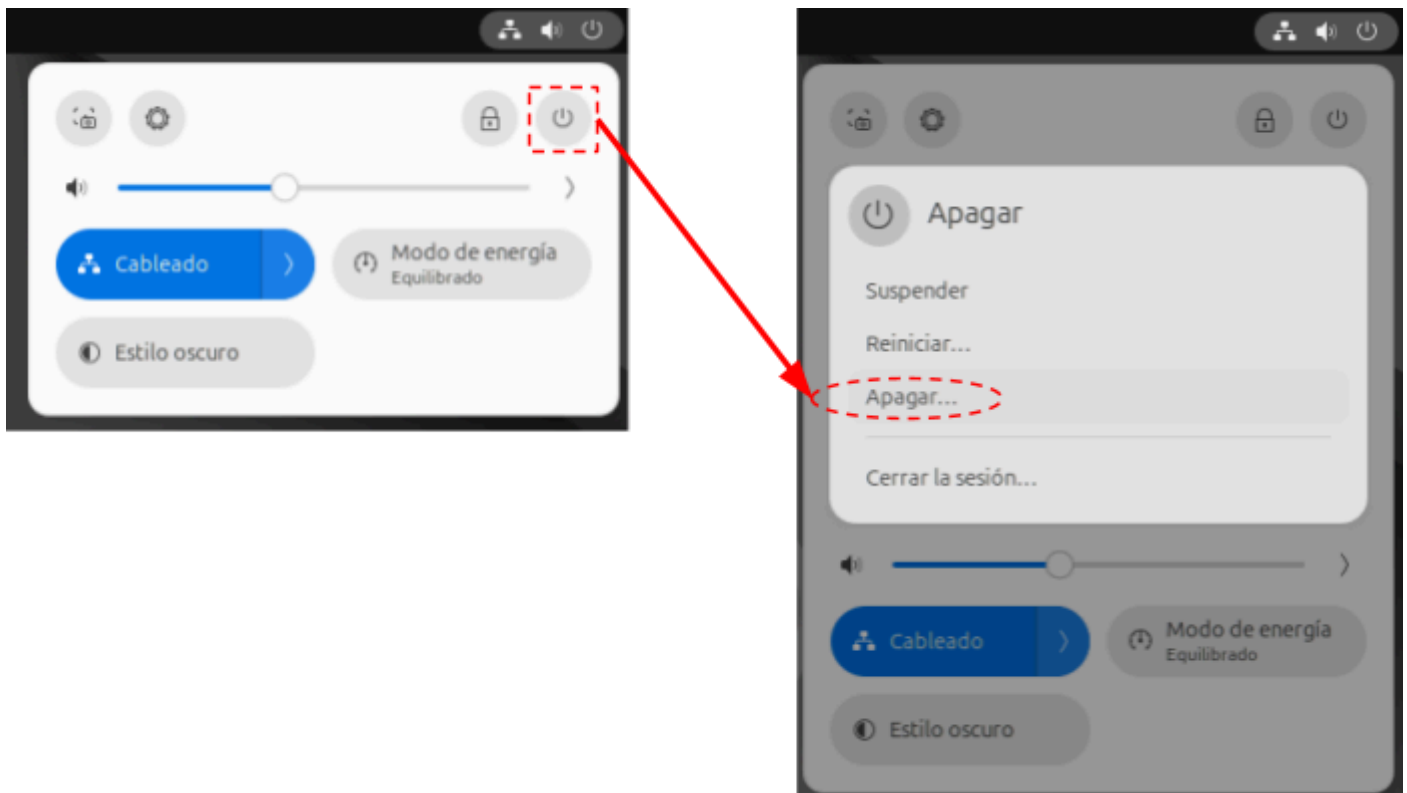
What happens now? We see in the Terminal the list of partitions that our hard disk has. By executing the command with `sudo` in front, we are doing it as root and therefore we have permission to do so.

Reviewing Linux commands

Command	Description
<code>ls</code>	It can be used with modifiers, for example ' <code>ls -l</code> '. It does not have the same options as <code>dir</code> in MS-DOS.
<code>cd</code>	To change directories. Important: The separation bar between directories in LINUX is the '/' and not the '\' as in MS-DOS.
<code>mkdir</code>	To create new directories.
<code>rm</code>	To delete files. For example: <code>rm test.c</code>
<code>rmdir</code>	To delete directories that are empty. For example: <code>rmdir directory</code>
<code>gcc</code>	To compile programs in C.
<code>gdb</code>	To debug programs written in C.
<code>man</code>	To request help with an operating system command.
<code>cat</code>	Displays the contents of a file in ASCII format.
<code>more</code>	Displays the contents of a file in ASCII format, pausing between pages.
<code>cp</code>	Copy files between different directories on the hard drive. For example: <code>cp pr1.c pr1b.c</code>
<code>mv</code>	To rename files. For example: <code>mv pr1.c pr1b.c</code>
<code>clear</code>	To clear the screen.

1.7. Ending a session with LINUX

At the end of your session you must exit the LINUX Operating System correctly. To do this we will select the rightmost icon at the top of the screen, to the right of the time, and in the menu that will appear, we click on the icon  and select the option "Apagar ...".

**Exercise 1**

Create the following directory structure on Linux using the command line.

```
Programming1
  Theory
  Practice
    Practice1
    Practice2
Maths1
  Theory
  Practice
```

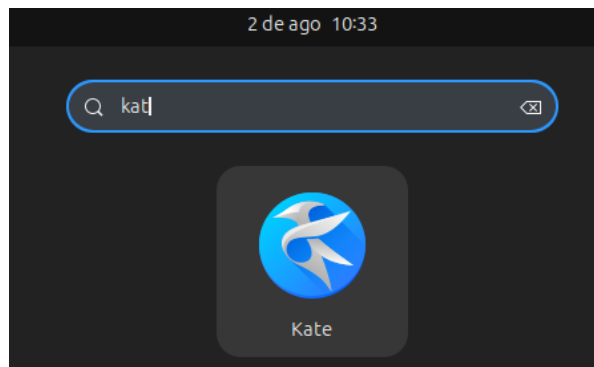
2. Your first program in C

To carry out the practices you will need to create files that contain the **source code** of your program to later interpret or compile them.

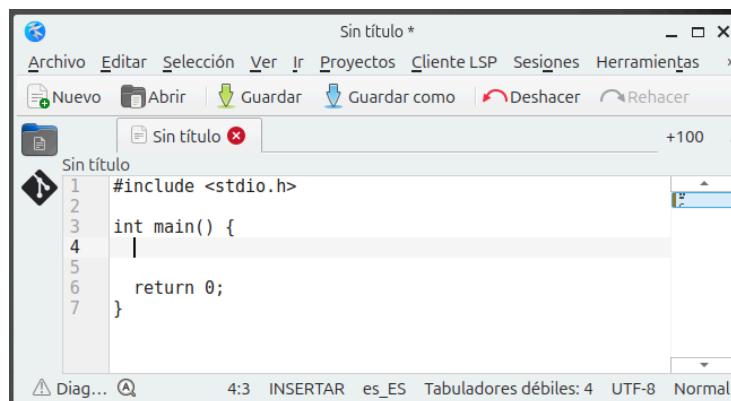
To create files you will need to use a text editor. The compilation is done from a terminal (at least we will do it that way in the first practices).

2.1. Kate (text editor)

In Ubuntu there are several text editors, but in the subject we are going to use one called Kate. To access it we can search for it with the **Dash** or if it has an icon in the **Dock** we will select it directly.



Kate editor with part of a program already written:



2.2. “Hello World!!” program

Let's create a file that contains a program written in C language. The C programming language will be the tool that will be used to perform the implementations of the algorithms.

The sample file could contain the following code:

```
#include<stdio.h>

int main() {
    printf("Hello World!!\n");
    return 0;
}
```

Once the program is written, we must save it (in Kate, Save option from the File menu). Let's save it with the name of **pr0.c**.

Now we can compile it. To be able to compile the file and generate an executable we have to do it from a terminal.

Now we type the following commands

```
$ gcc -Wall pr0.c -o hello (To compile the program and name it "hello")
```

All the information regarding the compilation of the file and the creation of the executable is shown on the screen

```
$ ./hello (To run the "hello" program)
```

```
Hello World!!
```

In this case the *compiler* we are using is "gcc" and the name of the generated executable is "hello"

To run a program in principle just type its name (followed by INTRO key). In the previous example we have put at the beginning of the name the sequence "./" because it could be necessary depending on the configuration of the system. On many Linux systems the shell searches for executable programs in a number of special system directories, but not in the current directory. The sequence "./" indicates precisely that the executable program is in the current directory. However, it is possible that your system is configured to also search the current directory and you do not need the "./" at the beginning.

If any compilation errors occur, they are displayed on the screen. If there are many and we want to consult it calmly, we must redirect them to a file as follows:

```
$ gcc -Wall pr0.c -o hello 2>error.txt
```

error.txt is the name that has been given in this particular case to the file in which the errors will be saved.

If there are no compilation errors, the result will be the execution of the object program, as seen in the previous example. The name of the executable in this case is "hello" and is written after the "-o" option.

2.2. Program to calculate your age in seconds

Now that you know how to write and compile a program, you can try typing and compiling this one. Save it with the name **pr1.c**, compile it and run it. The program should print your age in seconds.

```
#include <stdio.h>

int main() {
    int age;

    printf("Type your age: ");
    scanf("%d", &age);
    printf("You have lived %d seconds, approximately\n", age*365*24*60*60);

    return 0;
}
```

ANNEX: UBUNTU INSTALLATION

There are several alternatives to use Ubuntu without having to be installed on the hard drive of our computer, in an independent partition.

A simple way to work with LINUX is to install it in a virtual machine. In this way we do not have to repartition or format anything on the hard disk. Ubuntu will run inside a Windows window as if it were just another program on our computer. To do this, you can install the virtual machine that has been left in the Moodle of the subject, or you can use the one prepared by the Technical Service of the EPS downloadable from <https://eps.ua.es/es/eservices/ubuntuepsvirtual.html>. It contains the version of **Ubuntu** from the EPS labs along with most of the programs installed on them. On this page you will also find the installation instructions.