Basic tutorial on using the command line.

Bash. Differences between the terminal and the graphic. File system.

The command console is a way for us to communicate directly with the computer and give it orders. We give orders/commands that the system interprets and returns a response to us.

This tutorial is called GUI to CLI. GUI is an acronym in English that means Graphical User Interface. As their name indicates, they are all the graphic applications that we see on our desktop. In fact, our own desktop is a graphical interface. Examples of graphical programs in Ubuntu, Debian or almost all Linux are Firefox, OpenOffice, etc. On the other hand we have CLI (Command Line Interface) or Command Line Interface. In this type of communication with our computer, everything is more fluid and faster, although as we will see, at first it costs much more. This tutorial attempts to explain the basic concepts that revolve around it, looking for a way to become independent of our Graphic interface.

If we are not going to program and we are only interested in making sure the computer runs well, is the command line of any use to us if we can do the same thing graphically?

Yes, it does! Every time we ask someone for help, ask in a forum or simply search the Internet, we will see that everything appears in the form of instructions. The reasons are very simple

- Command line instructions have existed long before graphical programs and many have already gotten used to using them.
- Graphics programs change very often. Command line instructions hardly
- It is simpler to say or write on an Internet page cd /home/user/folder(We will see later what this means) Go to file explorer, look for the /home folder on your PC and open it, do the same with /user and then with /folder.

But if the command line is pure text, isn't it enough that when we have a problem, copy and paste a solution that a friend gives us or that we find on the Internet?
No. Many times when we have a problem and look for help we will see that we find things for someone who had something "similar" to our problem happen to them, but not "exactly" the same thing. Then we are going to have to apply that solution that we found, but with modifications that adapt to our case. For that we have to know the logic and how to work with command lines.

The command line instructions are interpreted by a program called console, or in English **shell** which literally means shell. In GNU/Linux the console that has been assigned to the

session is used, by default this is usually **Bash**, although there are many types of shells, for example ash, bsh, ksh, zsh, csh, fish. For each of these consoles the set of instructions varies a little. To see which shell we have assigned, we can see it by writing in the console "echo $SHELL". This is useful when we write things and see that the console does not understand us.

curso@h4ckdr4g:~$ echo $SHELL
/bin/bash

Depending on the Linux distribution, the graphical desktop that we use or what we liked to install, we will have different terminal or console applications such as gnome-terminal for gnome, konsole for kde, xterm among others. Although there are different types of terminal applications, they all work for us, since basically they all use the same thing: Bash. Bash was created based on Shell, the Unix terminal. Bash means Born Again SHell, Bourne's console again; Stephen Bourne was the one who wrote sh and that is what bash is based on, they are very similar although they are not fully compatible, sh is the console that was used for UNIX.

Basically, a shell is a **order interpreter**. It has many different names: Command Interpreter, Command Interpreter, Command Line Interpreter, Command Interpreter, Terminal, Console, Shell or CLI (Command line interface), the latter as opposed to GUI (Graphical User Interface). The main difference between the terminal and the graphic is that the second looks better, but consumes more resources and has fewer options.
More information: Command Lne, Bash, Stephen Bourne and Bourne Shell on Wikipedia.

Why are we interested in learning how to use the console with graphical applications?

- Because we want to know how to react when the graphic server does not work.
- Because we want to know how to communicate better with the computer.
- Because while in the graphics we have some options given by the designer, in the console we have all the options that the programmer set.
- Because the commands are faster than graphical applications, since you do not have to load graphical libraries.

One way to access the console is by selecting an application that we have on our system that serves this purpose. These are **gnome-terminal**, the **console**, the **xterm**. The latter comes in almost all systems so it is highly recommended. It consumes few resources and is quite configurable. They can be found in Applications > Accessories > Terminal or by doing ALT + F2 which will open the "Run an application" dialog, from where we can launch "xterm", "lxterm" or "gnome-terminal".

Another way to access the command line is with the combination **CTRL + ALT + F1**
With this we will access **tty1**, a screen where we can log in. An important thing with respect to Linux is that many times when it asks us for passwords it does not show us the

characters we entered, for security reasons (so that someone watching the screen does not know the number of characters). From F1 to F6 we can find more consoles (tty1-tty6). F7 is reserved for the graphics server. This is very useful because in case the graphics server crashes we can restart it, as we will see later.

Let's talk about **File System**. In Unix-based systems, such as GNU/Linux, there is a main directory on which all the others depend. It is the beginning of everything, the root in English. It is located in / and is called **Root Directory**

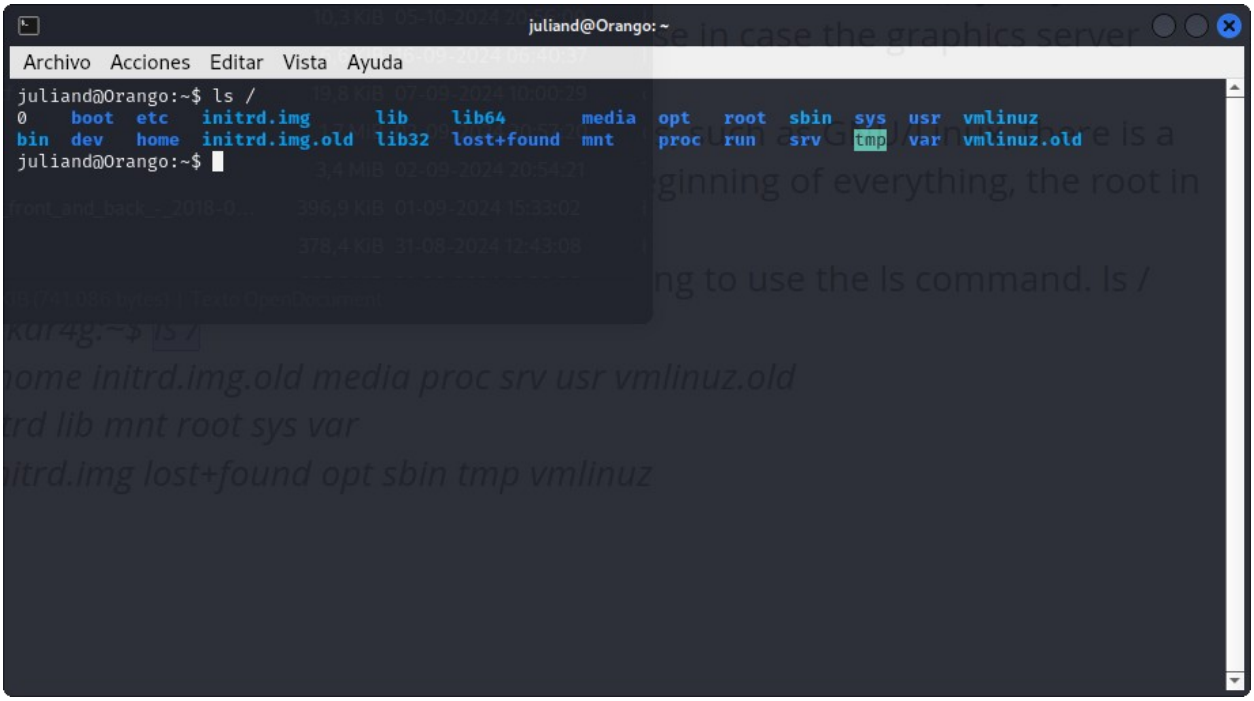
For example, to see a list of the files in / we are going to use the ls command. ls /

course@h4ckdr4g:~$ ls /

bin debian home initrd.img.old media proc srv usr vmlinuz.old

boot dev initrd lib mnt root sys var

cdrom etc initrd.img lost+found opt sbin tmp vmlinuz

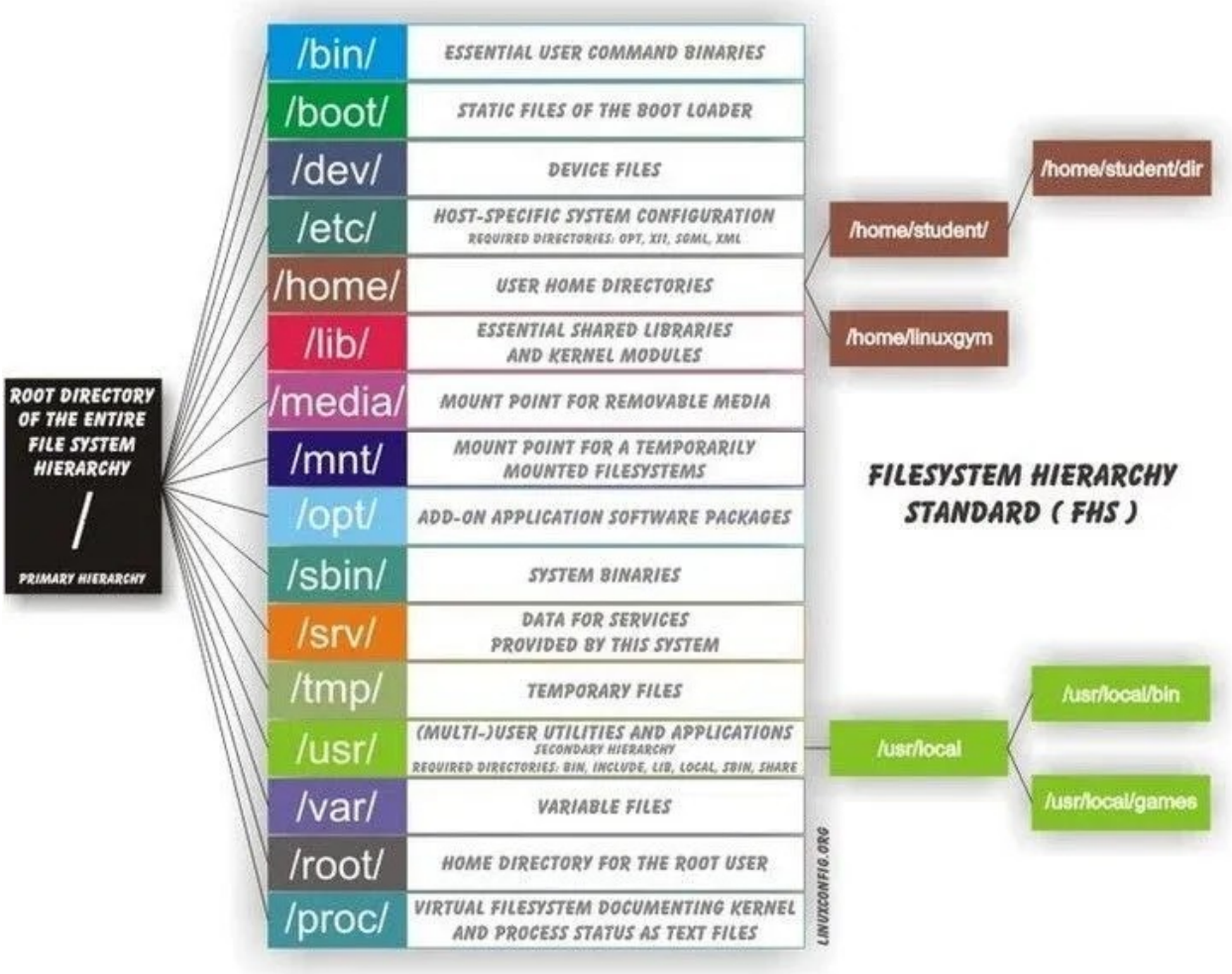


We can see what each directory is for in more detail. here. The ones that mainly interest us are the following:

/bin/: Basic commands for all system users.

/boot/: files needed to boot the system.

/dev/ It is important to understand the following concept: in Linux everything is



files: the files and folders, but also the different devices connected to the computer, from the mouse to the keyboard, including hard drives (with their partitions) and cdrom. /dev/ is where all these devices are. For example /dev/sda1 is the first partition (1) of the first disk (a).

/etc/ Most of the configuration files are located in this folder. Very important.

/home/ The users' home. Here each user has a folder with their name where they can modify everything they want without risk of damaging the system. In the other folders outside of our home we will need administrator permissions to modify. home/User is the folder where we appear when we open a new terminal

/lib/: essential libraries for the core of the system and its modules.

/media/ This is where all external memory devices will be mounted, for example pendrives, USB disks, cdrom, SD cards. By default it will try to assemble them automagically. /mnt was previously used but is now deprecated. It is common to find the

cdrom folder in the /media/ folder, in distributions like Ubuntu this folder can also be found in /cdrom

/mnt/: temporary mount point for devices.

/proc/: processes and variables of the core of the system.

/root/: home directory for the system root.

/sbin/: special commands for the system root.

/tmp/: temporary files. Depending on the distribution used (or the configuration we use), they are deleted when the system starts or every certain period of time.

/usr/: second hierarchical structure, used to store all software installed on the system.

/was/: directory for programs that coordinate printing, system message files, etc.

How can you write the names of folders and files in Linux?

In Linux **All files have a name, which must meet certain rules. It must be between 1 and 255 letters or characters in length.** Se You can use any character except the slash / and it is not recommended to use characters with special meaning in Linux, which are the following: = ^ ~ ' " ` * ; - ? [ ] ( ) ! & ~ < >. To use files With these characters or spaces you must enter the name of the file between quotes like this'file name'.

You can use numbers exclusively if you want. Uppercase and lowercase letters are considered different, and therefore carta.txt is not the same as Carta.txt or carta.Txt. The different types of files can be marked using a series of characters at the end of the name that indicate the type of file in question. Thus, text files, HTML, PNG or JPEG images have extensions .txt, .htm (or .html), .png and .jpg (or .jpeg) respectively.

Como moverse. Login. Clear. Reset. History. Reverse-i-search. Man, help e info. Pipe. Autocompletar.

In a Unix system we are always as a user. We can see this in the console itself. For example:

curso@h4ckdr4g:~$

It tells us that we are the current user on the hamlab computer. If we notice, it is the same syntax that is used for emails (user@server) this is because many email servers are Unix.

Important: we are always in a folder. By default we are in /home/user. We see this with **pwd**.

curso@h4ckdr4g:~$ pud
/home/course

It shows us which folder we are in at that moment. We can also see that it tells us after: the ~ is an abbreviation for /home/user. For example, if in any folder we do

ls ~

It shows us the contents of our personal folder.

Keyboard shortcuts:

CTRL + A: Go to the beginning of the line
CTRL + E: Go to the end of the line
CTRL + L: Makes a clear (clears the screen)
CTRL + D: Exit.
CTRL + R: History search (see below)
CTRL + U: Delete the entire line up to the beginning
CTRL + C: Cancel the running command

With the command **reset** we restart the terminal. This is very useful in case of errors or crashes.

Something very interesting is the **record**. It is accessed with the history command. With this we can see the last 500 commands entered. We can also see the last commands entered with the scroll arrows (up/down). With the CTRL + R key combination we can access a search within the history. This is Reverse-i-search mode.

The main rule is **RTFM**(What do we mean when we talk about RTFM?). The manuals are installed along with the applications, they can normally be found in /usr/share/docs, but to access any you just need to put
man command

For example, to see the clear man we put **man** clear. Other commands that are also useful for searching information are **help** and **info**. Its syntax is the same:
help command
info command
Many times we can also find help (mostly the different options for each command) with the option **--help**

--help command
To enter several commands at the same time, use **pipe** (pipeline). This is | On Spanish-speaking keyboards (with enie) it is located at ALT GR + 1. It serves to combine several commands into one. For example
lspci | grabbed the VGA
This means that the output of the lspci command will in turn be the input of the VGA grep. Normally the first times we see someone write on a console we are surprised at how fast they write. There is a trick behind this, the most important key on the keyboard, which is the **TAB** or tab (located above CAPS LOCK). This allows files and applications to complete

automatically. The magic behind this is found in an application called bash-completion.

Commands: order, options, arguments.

The basic syntax of a command is the following:

order options arguments

The command is the command itself, the name of the application being called. For example, to open Firefox you just have to put Firefox in the console.

firefox

When an application monopolizes the use of the console, you can cancel the command with **CTRL+C**. So that it does not monopolize it, you have to put the **ampersand (&)** at the end of the order. For example

firefox &

Then the options can be for example *--new-tab* and an argument a web address (*https://docs.telavivmakers.space/hamlab*)

firefox –new-tab https://docs.telavivmakers.space/ &

If we have a text file that we want to open with the text editor (gedit), it can be done with gedit file. This works with all applications, for example to open images with gimp it is gimp file, to open a video with vlc it is vlc file, to listen to music with amarok it is amarok file.

Let's now see certain basic commands:

ls. This is used to list files. With ls we only list those that are in the folder where we are, but we can also put and the name of a folder. To see all files (even hidden ones) you can use the -a option and to see file information (permissions, owner, modification date) with the -l option.

folder number

ls – her

cd: changedirectory. It is used to change directories. With CD we only go to our home. With cd.. we go to the top folder. To go to root you can with cd / . With cd - we return to the folder where we were before.

cat: prints the text of a file on the screen. Example to see information about our processor

cat /proc/cpuinfo

Other similar commands with **more** and **less**. They have the same syntax and serve the same purpose, only less is a little more complete, allowing us to go line by line and go back. The more only goes through screens.

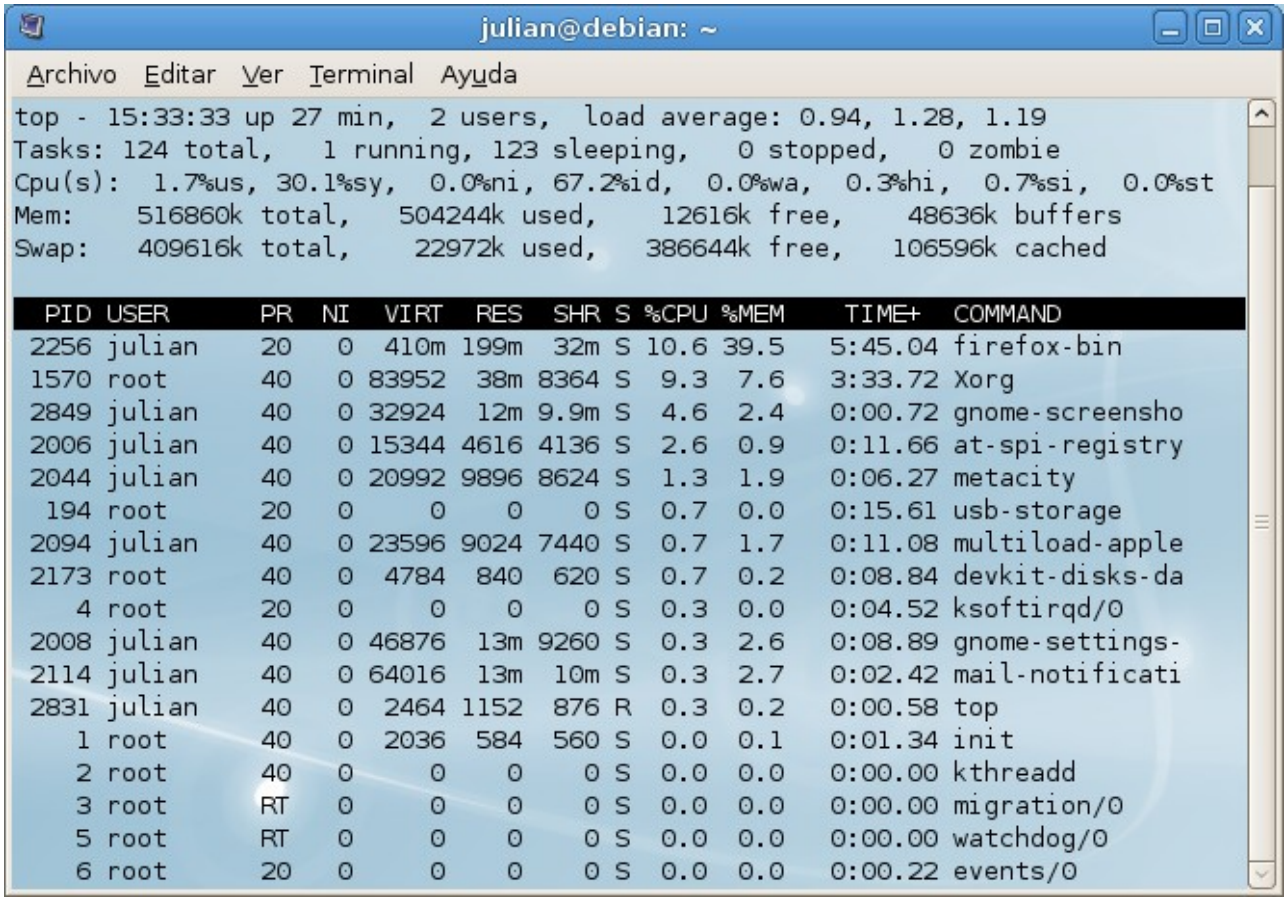
touch: simply create a file. Very useful to check that we have write permissions.

touch file

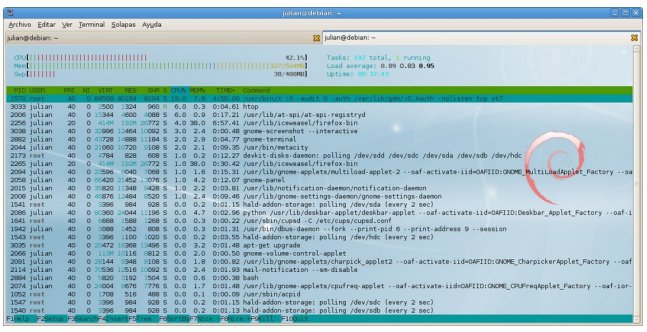
free: shows our occupied ram and swap memory and also the available one. With the -m option we see it in megabytes. Example:

free
total used free shared buffers cached
Mem: 1019200 1003256 15944 0 79716 681208
-/+ buffers/cache: 242332 776868
Swap: 2441840 96936 2344904

top: shows information about what is running on the system.



Very useful. **htop** It is a modification of the top that shows us the information in colors and allows us to kill processes from its same interface.



ps: gives a list of all running processes. To see all of them you need to set the aux options. Syntax:

ps auxf

lspci: gives a list of the different PCI devices. From VGA and Audio to Ethernet and SD. It is very useful to know the name of the components we use.

head: shows the first 10 lines. With the -n option we choose the number of lines we want to see. Example:

head /proc/cpuinfo

tail: shows the last 10 lines. Example:

tail /proc/cpuinfo

dmesg: shows a kernel log with information that our system does

cp: to copy files. Syntax:

cp origin destination

For example to make a backup copy, or backup in English, of our sources.list

sudo cp /etc/apt/sources.list /etc/apt/sources.list.backup

mv: to move or rename files. Syntax:

mv origin destination

rm: to delete files. You have to be very careful because it does not take it to the trash, but rather deletes it from the system. Syntax:

rm file

With the -rf options force recursively (delete a directory without asking). To ask us, we can do it with the -i option. An old joke with rm is rm -rf / . It is NOT AT ALL advisable to execute that command.

cal:

If we put it on, it only shows us a calendar of the month we are in. We can ask for whatever we want.

cal 1 1

January 1st of year 1 was Saturday :D

date:

It shows us the time and day we are on. With the --set option change the time.

uptime:

It tells us how long the computer has been on. Very useful for servers.
wget: to download from the Internet. Syntax:
wget address
For example:
wget http://https://docs.telavivmakers.space/home/guias-talleres-y-cursos
With the -r option download all the contents of a website.

shutdown: Turns off the system. Requires administrator permissions. It is necessary to tell it the time at which we want to turn it off. To restart, the -r (reboot) option is used, but if we only want to turn it off, it is the -h (halt) option. Example:
sudo shutdown -h now

reboot: Restart the system. Requires administrator permissions. Syntax:
sudo reboot

ps: serves to list all the processes and applications that are running at that moment. Usually used with the aux options. Syntax:
ps auxf

grip: shows us only the line that interests us. Usually used with | (pipe). For example, to know which video card (VGA) we use:
lspci | grabbed the VGA

Decompressing

takes- Multiple files combined into one, uncompressed. To extract, you need the xfv option, which would be "extract from a file showing information", x is extract, f is file, v is verbose. This option (v) is in many commands, serving to show us more information about something in particular. To compress a directory or file it can be done by changing the x (extract) to c (create)
tar xvf file.tar
tar cvf file.tar folder

gz: gzipped file. To decompress you can use gunzip.
gunzip file.gz

bz2: file compressed with bzip2. To decompress you can use bunzip2
bunzip2 file.bz2

tar.gz, tar.bz2: we can find a combined file (tar) and compressed in gz or bz2. Instead of doing it in two separate steps, which is possible, it is best to use tar to add the option z for gz and j for bz2.
tar xvzf archive.tar.gz
tar xvjf file.tar.bz2

zip: zip compressed file. To decompress, use unzip:
unzip archivo.zip

rar: compressed file in rar. Being a proprietary format, it cannot come with the Operating System. To install it, do it with sudo aptitude install unrar . Then to unzip
unrar file.rar

Text editors: nano and vim.

One very important thing is the **text editors**. There are several applications for this: emacs, joe, nano, vim. We are interested in this because, as we will see, in GNU/Linux most of the configuration files come in text.

The easiest to use is **nano**. To open a new or existing file you can do it with:
nano filename

It is quite graphic, having the options to save, exit or search among other things. To access you have to do CTRL + Key. For example to save, CTRL + o. It will ask us for confirmation. To exit, CTRL + x. By default in Ubuntu it is not installed. To install it:
sudo aptitude install nano

One of the editors that comes by default in almost all distributions is vi or **because** (the only difference between these two is that the second one has more options, it is the Vi iMproved). The way to open it is the same for all applications:
vim filename

By default it opens it in view mode. To modify something we have to press ESC, then insert and there we will be in editing mode. Once we have finished modifying what we want we will have to press ESC to return to view mode. From there to save you have to put w (for write) and to exit q (quit). To learn more about the potential of vim, we can access vimtutor, a tutorial that explains how to work with vim, how to search, copy and paste, etc.
vimtutorial

Permissions and users and groups

are: change user. Syntax:
your user

If we do not give it any option, it allows us to access as root (administrator) and be able to make all the changes we want in the system. For security reasons, Ubuntu does not recommend it, preferring to use sudo, which gives administrator permissions only to the command where we put it. For example, if we want to see what the system log says,
cat /var/log/syslog

It tells us that we do not have the necessary permissions (cat: /var/log/syslog: Permission denied)

But instead it allows us with
sudo cat /var/log/syslog

sudo: means that the next line will have administrator permissions.
The first time we start sudo we get the following message:
We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:
#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.
In Spanish:
We hope you received the usual lecture from the System Administrator. It usually comes down to these three things:
#1) Respect the privacy of others.
#2) Think before you write.
#3) With great power comes great responsibility.
More users can be added to sudo by modifying /etc/sudoers . There is a command that comes with Ubuntu that is used to modify it with the nano entering directly, the visudo. User ALL=(ALL) ALL must be added to the end of the file.
south view
Users can also be added to sudo by adding them to the admin group

User Administration

We can create users with the adduser command:
adduser user

It will ask us for the password (which we will enter twice) as well as other information (full name, phone number, etc.)
course@h4ckdr4g:~$ sudo adduser test
Password:
Adding user `test' ...
Adding new group `prueba' (1002) ...
Adding new user `prueba' (1139) with group `prueba' ...
Creating home directory `/home/prueba' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for prueba
Enter the new value, or press ENTER for the default
Full Name []:
Room Number []:
Work Phone []:

Home Phone []:
Other []:
Is the information correct? [y/N] y

To delete a user it can be done with userdel
user's user

He will leave his home untouched in case we want to keep the data he had. We can eliminate it by doing *sudo rm -rf /home/user*

Groups are also used for system administration. This comes into play with the issue of file permissions, which we will explain later. To create a group, use the addgroup command and to delete it with delgroup. And to add a user to a group, we It also works with the addgroup:

addgroup user group

A line is automatically created for all users in the /etc/passwd, where it indicates various data: user name, user ID user, ID of the group to which it belongs, where is its home and what type command line interpreter uses:

test:x:1002:1002:,,,:/home/test:/bin/bash

As we can see, it does not tell us its password, although in Actually, it does: it is the second parameter, the x. This is because It actually saves them in another file, /etc/shadow, where it is your encrypted password:

test:6QaJ6VQdc$/eTOBEPYgB/6z/0HluxqWiIuR2spMMT2B52WXA1XBJhVwcrSsrP/sqtA/MuMveE1mmfc8.zsrJV86OwuTGNdp/:14204:0:99999:7:::

To change a user's password we can do it with the passwd command.
sudo passwd user
More information about User Administration in Linux:
https://tldp.org/LDP/sag/html/managing-users.html

HOW to kill rogue processes: kill, killall, xkill.

Something very useful in Linux is that in case a process hangs or we want to stop something that is running, it is very easy to do so. For that we will use the command **kill**. The syntax is

kill -9 process-id
To know the ID of the process we will have to see it in
ps to

We will be able to see only the process we are looking for if we do
ps to | grep process

For example

curso@h4ckdr4g:~$ ps aux | grep firefox

course 5668 5.9 8.4 227208 86568 ? Sl 00:39 0:45 /usr/lib/firefox-3.0/firefox

curso 6507 0.0 0.0 3836 784 pts/0 R+ 00:52 0:00 grep firefox

The process ID is found in the second column (5668) which tells us that it corresponds to Firefox. To kill him then we would have to put

kill -9 5668

Another solution to know the ID faster is with the command **grip**. It automatically gives it to us. Example:

pgrep firefox

5668

A much faster way to do this is with the command **killall**. The syntax is

killall proceso

For example, to kill the firefox process we can put

killall firefox

Another simple way to stop a process is **xkill**. As its name indicates, this serves to kill only the graphics. It shows us a cross and we can select the window we want to kill. We can cancel it with ESC.

Most used configuration files: /etc/apt/sources.list, /etc/X11/xorg.conf, /etc/network/interfaces

/etc/apt/sources.list- Here are all the system package repositories. By default in Ubuntu there are certain repositories with restricted software (restricted, universe, multiverse) that appear commented (with # at the beginning of the line). Example of a line:

that http://us.archive.ubuntu.com/ubuntu/ focal-updates main restricted

/etc/X11/xorg.conf: the graphics server configuration file (X11). For example, we can modify certain aspects of the video (card driver, monitor frequency). In the latest version of Ubuntu, if we make a mistake with the configuration of our graphics card, a tutorial will appear that will automatically configure them for us. It can also be modified with *sudo dpkg-reconfigure xserver-xorg -phigh*

/etc/network/interfaces: This file contains the network configuration. Here is whether you are going to take a static or dynamic IP.

/etc/fstab: This file contains the configuration of the mount points that we have configured.

/etc/resolv.conf: here we configure our DNS (Domain Name Server).

/etc/passwd: shows us the users in the system, their IDs, their group ID, where

their home is, and the shell they use.

/etc/shadow: here are the encrypted passwords.

Restart graphical server (gdm, kdm)

As we already said, in **/etc/init.d/** There are many services that may be interesting to us. Among these is the graphical server (gdm in the case of gnome and kdm in the case of kde). First you would have to go to tty1 for example (CTRL+ALT+F1), log in and stop the service

sudo /etc/init.d/gdm stop

To start it again:

sudo /etc/init.d/gdm start

To restart it directly:

sudo /etc/init.d/gdm restart

We already know how to stop, start and restart services in Linux!! This is the same in almost all of them, for example, to restart Apache (web server) or Samba (file server, like CEB!)

sudo /etc/init.d/apache2 restart

sudo /etc/init.d/samba restart

In case the graphics server has hung, there is a very useful key combination that restarts it: **CTRL + ALT + BACKSPACE** (delete key above ENTER)

In case the system does not respond to us with this, there is a key combination that tells the system that it must reboot completely. It is not highly recommended to use it but it is a way to close it cleanly without hitting the shutdown button in case of a hang. The combination is quite complicated: **ALT + PRT SCR and RSEIUB**

Each of these letters has a meaning. There is a mnemonic to remember this: *Raising Skinny Elephants Is Utterly Boring* .

Install, uninstall and update applications and the system (apt-get and aptitude). Install .deb. Compile.

One of the reasons for Ubuntu's success is its way of installing packages, the same as Debian, its mother distribution. The **packages .deb** They are very easy to install:

sudo dpkg -i packagename.deb

To eliminate them it is

sudo dpkg -r packagename

We can see what packages we have installed with

sudo dpkg --list

To install them graphically you have to double click on them, accept and enter the password. We can download them on the GetDeb website: www.getdeb.net

The problem with this system is that for many applications we will first need to install other packages first, so we will have to do it by hand, one by one.

[NOTE: In other Linux systems (RedHat, Opensuse, Mandriva) another type of packaging is used, rpm.]

This is not the only way to install them, but also through the **repositories**. Repositories are directories full of software. Ubuntu has more than 30,000 packages in the repositories, but we can add more repositories. The list of all repositories can be seen in /etc/apt/sources.list, for example with nano:

nano /etc/apt/sources.list

The good thing about this system is that if you need other packages (applications or libraries) the system itself searches for them and installs them automatically. This is the fastest, simplest and most direct way to install software.

There are two applications that are used for this: **apt-get** and **aptitude**. (The difference is that the second has more options). The syntax:

sudo aptitude install packagename

For example to install Firefox

sudo aptitude install firefox

Another option that we will use a lot is remove (to delete), show (see more package information) and search (to search).

sudo aptitude remove firefox

sudo aptitude show firefox

sudo aptitude search firefox

Ubuntu, in its attempt to make our lives easier, if it sees that we put in a command that we don't have, it will tell us. For example:

curso@h4ckdr4g:~$ sl

The "sl" program is not currently installed. You can install it by typing:

sudo apt-get install sl

-su:sl:order not found

For this to appear, the command-not-found package must first be installed (*sudo aptitude install command-not-found*). By default we will already have it installed on the base Ubuntu system. If we do not have it installed then the classic one will appear:

bash:sl:order not found

In case we do not find the software in the repositories or in deb packages, there is always the option of **compile**. What is compile? In short, it is converting the source code into a language that the computer can understand it. It is one of the bases of free and open source software, since it allows the user to see what the software they are installing has, and allows the developer to modify it to their liking. Many of the free software projects are hosted on sourceforge: http://sourceforge.net

To compile, we must first unzip the file where the application is, then configure it, then tell it to prepare it and finally install it. This is done with the commands

./configure

make

sudo make install

Of course, it is always advisable to read the README and INSTALL files that appear along with the sources first.

To compile on Ubuntu and Debian you must first install the build-essential package with

sudo aptitude install build-essential

Let's see an example of a compilation with the hello program. This can be downloaded from http://www.gnu.org/software/hello/ To begin, we will download it from the terminal to the /usr/local/src directory. This directory is where we will put all the programs that we are going to compile. So

cd /usr/local/src

To download it we are going to use the wget tool.

wget ftp://ftp.gnu.org/gnu/hello/hello-2.3.tar.bz2

As we see, it is compressed in .tar.bz2, so we unzip

tar xvjf hello-2.3.tar.bz2

It automatically creates a folder called hello-2.3. We enter the folder

cd hello-2.3/

We see what files are inside with ls

ABOUT-NLS ChangeLog configure.ac gnulib man src
aclocal.m4 ChangeLog.O contrib INSTALL NEWS tests
AUTHORS config.in COPYING Makefile.am po THANKS
build-aux configure doc Makefile.in README TODO

We see that it tells us INSTALL with

less INSTALL

It tells us how to compile, so we follow the steps:

./configure

It checks that we have all the necessary dependencies to run the application. If we were missing something we would have to install it (with apt-get, dpkg or compiling) and launch configure again. The ./ is used because we want to execute it. Once the configure is finished, we click

make

This already prepares the file for subsequent installation, telling it where, what and in what order it has to compile. Finally we would need to install it, for which we need superuser permissions:

sudo make install

This finishes installing the application. We can verify that it works with

hello
Hello world!

For more information on how to compile:
http://en.wikipedia.org/wiki/Compiler
https://rc.byu.edu/documentation/unix-tutorial/unix8.php

HOW to mount devices

Although the Linux automount function has been working better and better lately, it can cause some problems. For this, we are going to see some basic notions of assembly to

get out of trouble. First, what is riding? According to
the Wikipedia, is "the action of integrating a file system hosted on a certain device within the directory tree of a operating system"So, to see what we can put together, we can see it by doing

dmesg

And it appears to us which device is the one that was recently added, giving us its identification in the system (sdx).

So, as we saw earlier, all devices in linux
They are in the folder */dev*, we will have to look for it in */dev/sdx*, or we can go looking for it in */dev/disk/by-label* or one of the other folders */dev/disk/*

Then we have two ways to mount it:

- Doing it only for this session, we must first create a folder
  sudo mkdir /media/disco
  Then we must use the command **mount**. The syntax for this is the following:
  sudo mount -t type /dev/source /media/disk
  For example, if the USB disk we put is labeled "movies" and has the standard Linux file system, we can do it with
  sudo mount -t ext3 /dev/disk/by-label/peliculas /media/disco
  Other file system types are vfat, for fat32, and ntfs-3g, for ntfs, the standard file system in Windows since NT. The latter
  driver we may not have it installed on the system, so we can do it as always with *sudo aptitude install ntfs-3g*
  To disassemble devices we can do it with
  sudo umount /media/disco
- Whenever we start the computer we have it, we are going to do it with the file **/etc/fstab**, which we already saw previously. To do it following the example, the following line would be enough:
  /dev/disk/by-label/peliculas /media/disco ext3 default 0 0
  As we see, the structure is quite simple, being
  /dev/origin /media/disk type options 0 0
  For ntfs-3g we must also add the force option, an example:
  /dev/disk/by-label/peliculas /media/disco ntfs-3g default,force 0 0
  Then to enable the changes we can do so by telling the mount command to mount everything it finds in the fstab; Of course, it will not dismantle the file system where we have the operating system because otherwise it would not be able to work. The syntax is:
  sudo mount -a
  Then, to disassemble we can simply do it with
  sudo umount /media/disco

Redes: ifconfig, iwconfig, iwlist, dhclient. nm-tool, ping.

The command **ifconfig** It gives us a list of all the network devices that we have in the system. Normally eth0 appears, which is the Ethernet (network cable) and eth1, ath0 or wlan0, which is the Wireless (wifi), although this may change depending on the system. The

-a option can be used to see all of them.

sudo ifconfig -a

Another way to see our connection status along with other information is nm-tool. Syntax:

nm-tool

It also gives us other important information such as **IP** that we have, and among other things this command is used to change this. Syntax:

sudo ifconfig IP device

Example:

sudo ifconfig eth1 192.168.1.9

Another interesting command is **iwconfig**. This is very similar to the previous one but specializes in wireless devices. Among other things, it tells us if we are connected to a Wi-Fi, what name it has if we are, the mode, frequency, speed, etc. It also serves to connect us to these. If we just put iwconfig it will show us what wireless network cards we have available. Example:

sudo iwconfig eth1 essid h4ckdr4g

To enter one that asks us **contrasenia**:

sudo iwconfig eth1 essid h4ckdr4g key s:XXXXXX

A very useful option is essid **any**, which connects us to the one that is decrypted and with the best signal.

sudo iwconfig eth1 essid any

If we want to know what networks we have available we can see it with **iwlist** and the scanning option. Example:

sudo iwlist eth1 scanning

Another useful command for networks is **dhclient**. It automatically finds us a dynamic IP (requests DHCP), so we won't have to worry about setting one ourselves. You need superuser permissions

sudo dhclient

As we said before, a file that may interest us is the **/etc/network/interfaces**. This is where we put whether the IP for any device is static or dynamic. We can then configure a dynamic IP by putting this in the file:

auto eth0
iface eth0 inet dhcp

A static IP is configured as follows:

auto eth0
iface eth0 inet static
address 192.168.1.2
netmask 255.255.255.0
gateway 192.168.1.1

The command **nm-tool** It gives us a lot of information about the network.

He **ping** Constantly send packages to the destination we want. It can be from the

local network (for example 192.168.1.1) or from the Internet (docs.telavivmakers.space). Its syntax is

ping destination

For example:

ping docs.telavivmakers.space

Another interesting command is **mtr** (my trace route). Similar commands are tracepath or traceroute. They show us the route that a request takes to some Internet address.

One of the reasons for the success of Linux is undoubtedly the **ssh**. It means secure shell and arises as an opposition to rlogin, telnet and other forms of remote connection that have fallen into disuse and are insecure. As its name suggests, ssh is secure. Security is based on GPG encryption. SSH allows us to connect to a remote computer, being the best ally of a system administrator. We have to tell it where we want to connect (an IP or address) and a username, or it will try to log in with the same one we are with on the local machine.

ssh user@IP

To connect via ssh to any computer we will first have to have installed the package **openssh-server** (*sudo aptitude install openssh-server*)

Another very interesting command is the **scp**. Its syntax is very similar to ssh, and it is used to send or take files from a remote computer with the ssh server installed. The two points at the end are very important. There we indicate the path where we want the file we send to be placed. By default it will be placed in the homepage of the user who sent it.

scp file user@IP:

Bonus: towards infinity and beyond. MC, mp3, browser, torrents, messaging, videos.

There are file browsers for the terminal. One of the best known is Midnight Commander (**mc**). In Ubuntu by default it is not installed, but we can get it with

sudo aptitude install mc

To listen to mp3 you can with **mpg123**. Syntax:

mpg123 file

It is also possible to listen to them with a much more complete player: the **mocp**.

To navigate the terminal you can use **lynx** the **links2**. Example:

lynx https://docs.telavivmakers.space

links2 has an option to view images, with the g option

links -g https://docs.telavivmakers.space

To download files **torrents**:

btdownloadcurses fichero.torrent

To see **movies in ASCII**:
mplayer -vo aa fichero

To see **images** (con ImageMagick):
image display

And of course, a good curiosity **geek**:
telnet towel.blinkenlights.nl

Bash programming (scripting) and more links.

- http://tldp.org/LDP/Bash-Beginners-Guide/html/index.html
- http://www.pixelbeat.org/cmdline.html