CENTRE ESPECÍFIC D´EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

UD 05. INTRODUCTION TO LINUX

Computer Systems
CFGS DAW

Álvaro Maceda
a.macedaarranz@edu.gva.es

2022/2023
Version:221025.1701

License

Nomenclature

Throughout this unit different symbols will be used to distinguish important elements within the content. These symbols are:

Important

Attention

Interesting

TABLE OF CONTENTS

UT 05. Introduction to Linux

1. What is Linux?

The simplest way to define Linux is that it is a Unix-style operating system. At this point of the school year we already know that it is an operating system, but what is Unix?

Although not the first operating system, Unix is, undoubtedly, the first great operating system. Its most important features were that it is portable (can run on different computer systems) multitasking and multi-user. In addition, architecturally speaking, it was created based on concepts such as simplicity and modularity so that the code was easily maintained and extended by other programmers.

Over the years, the creator company (Bell Labs[1]) was licensing the product to other companies that, in order to adapt it to more specific environments, were making modifications and creating different versions. Hence, products like Xenix (Microsoft), HP-UX (HP), IRIX (Silicon Graphics), SCO (Novell), AIX (IBM) ... were born.

All systems in the UNIX family are often called *IX

The problem with all of them is that they are proprietary versions so the code is not available for study (apart from the large amount of money that a license costs). That is why, with a purely educational objective, in the late 80's, a professor at the University of Amsterdam decided to create Minix, an OS based on the UNIX philosophy but rewritten from scratch and open source. Due to its educational nature, the author decided not to allow modifications which would very likely complicate the code much more.

It is at this time that a Finnish computer science student decides, based on Minix, to create a free clone that works on PC systems. This student was Linus Torvalds[2] and he called his operating system Linux.

1.1 GNU/Linux

In the previous section, we commented that Linux was an operating system, but that is not exactly true. Every operating system consists of a kernel or core and a set of applications that help make possible the function of the operating system. A possible classification of the different programs that accompany the kernel in an operating system could be: the shell or the terminal (which allows interaction with the user text mode), services or daemons (which are programs that run in the background), a graphical server (which allows you to draw elements on screen) or a desktop (which takes advantage of the functions of the graphical server to provide graphical access to the user).

Linux is simply the kernel of the system. To form the OS, it's accompanied by many GNU[3] utilities. That operating system is not called Linux, it's called GNU/Linux

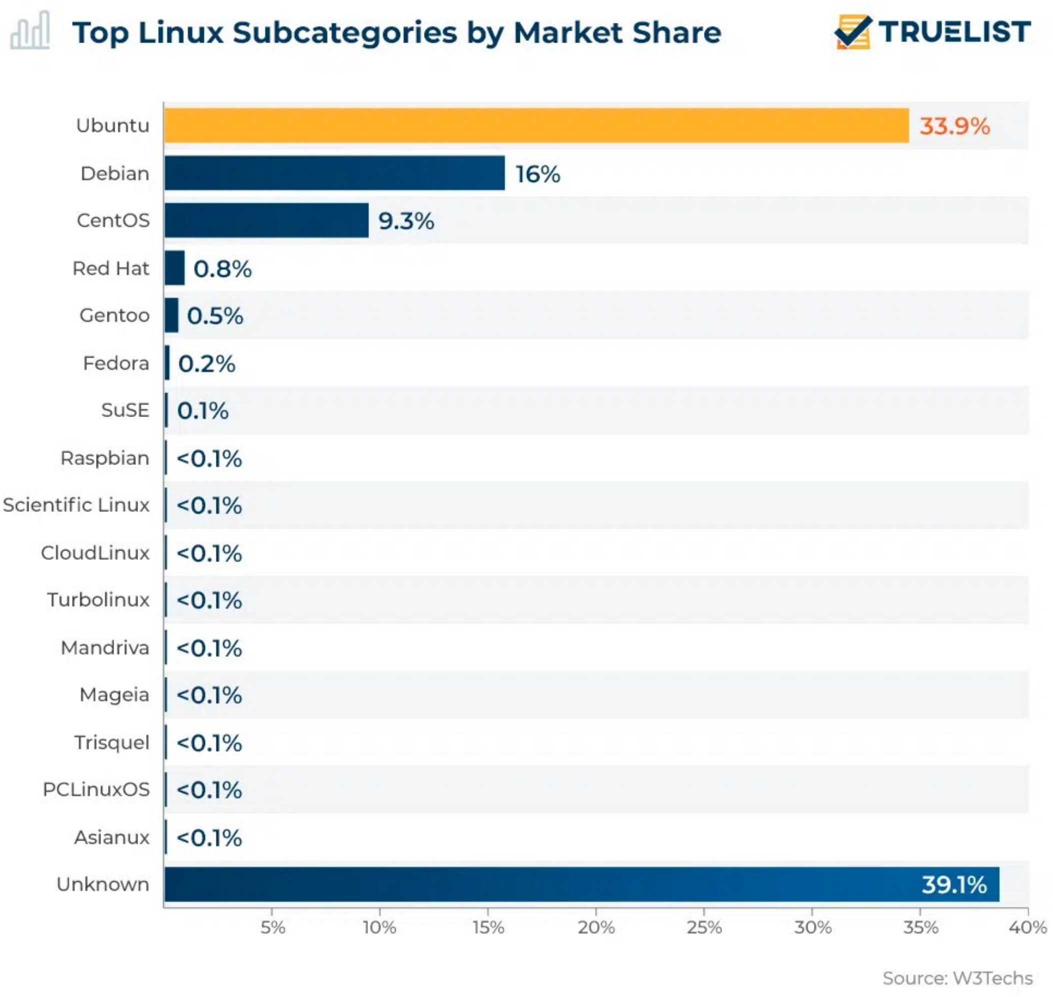
1 Founded by the inventor of the phone, Graham Bell
2 http://www.comunidadhosting.com/t/primer-mensaje-de-linus-torvals-y-comienzo-de-linux.8904/
3 GNU is a pre-Linux project whose goal is to create a free operating system. The project has developed a lot of the

Every day, everyone simplifies and calls Linux the operating system (wrongly).

1.2 What is a distribution?

The kernel and basic utilities make up the core operating system, but we know that today an operating system is released with a lot of software not necessary for the computer itself (drawing programs, text editors, etc.). The fact that the kernel and the basic utilities are freely distributed allows anyone to take these elements and accompany them with other utilities (whether basic or not) according to the way they like it or their needs. This is how distributions of Linux or, more commonly, distros arise.

There are hundreds of them in the market, but to mention some of the best known: Ubuntu (and its different "flavours" like LUbuntu, KUbuntu, Ubuntu Server ...), Open Suse, Arch, Fedora, Debian, Red Hat, Mint, Lliurex.



Statistics about the most used Linux distributions. Source: truelist.co

utilities that go with the kernel, but to this day, they have not achieved to have a stable enough kernel.

1.3 Where do we find Linux?

One of the most widespread ideas is that Unix is an operating system that is only used in academic environments or of a high technical level, and nothing is further from reality. The systems based on free versions of Unix are implanted in many computer systems. Although possibly in desktop environments Microsoft systems are still the ones that dominate most of the market, in mobile devices as in servers the reality is quite different.

For example, the two mobile operating systems par excellence (Android and iOS), are systems derived from Linux or freeBSD4 (as well as systems such as MacOS).

2. Console and graphic modes

We need a way to interact with a computer. In desktop systems, it's usual to interact with the system using a graphical interface. However, sometimes we don't need it or we can't use a graphical interface: for example, for server computers. Linux allows using both types of interfaces: we can configure which one will start when the system boots up, and also install or remove the graphical interface from the computer.

To switch between graphics and console mode, we can use *systemd*. Many Linux distributions use *systemd* to manage system settings and services. It decides which tasks the system will run when booting (units) and organizes them into targets. A target is a set of units with dependencies between them. The most important targets of *systemd* are:

- `default.target`: The target that is booted by default. This is a symbolic link to another target, like graphical.target.
- `emergency.target`: Starts an emergency shell on the console.
- `graphical.target`: Starts a system with network, multiuser support, and a display manager.
- `halt.target`: Shuts down the system.
- `multi-user.target`: Starts a multiuser system with networking, and no graphical environment.
- `reboot.target`: Reboots the system.
- `rescue.target`: Starts a single-user system without networking.

We will see this in more detail in a later unit. The important point here is that if we run the `graphical.target` we will end up with a graphics environment, and if we run `multi-user.target` we will have a text environment with a console to interact with the system.

You can view which is the current target with:

```
systemctl get-default
```

3. The terminal

People need an interface to send information to the computer and read the output from it. The input devices we use today include keyboards, mouses, microphones, and webcams. The output devices are usually one or more screens and a speaker. Those are called peripherical devices, but in Unix and Linux world they are called terminals.

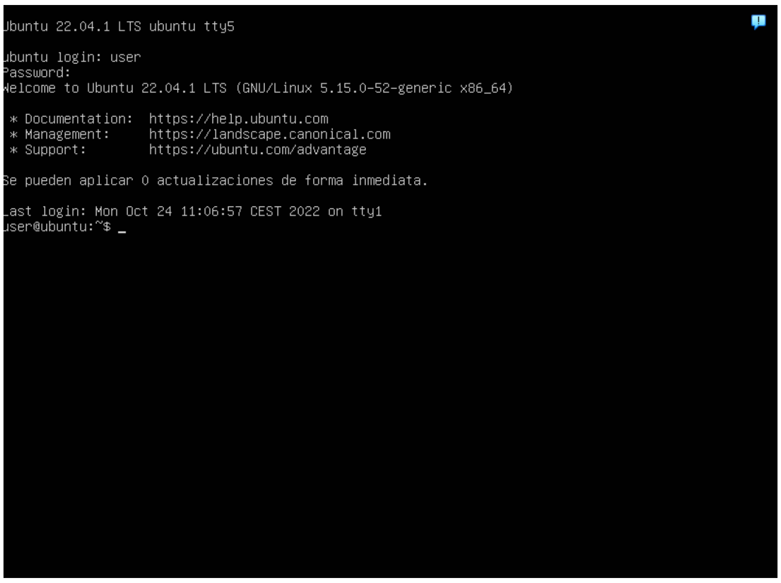
A *terminal* refers to an electronic device with a monitor and keyboard used to interact with a computer. This device is a type of physical terminal in which the programs are active in text mode, a place from which a system is monitored and controlled. Nowadays, there are so-called virtual consoles, which means that the console can be completely software. In practical terms, no difference is made between console and terminal and we will use both terms indistinctly.

The console, then, emulates a keyboard and monitor attached directly to the system. This would take the place of a serial terminal on minicomputer and mainframe Unix systems.

3.1 Virtual consoles

A virtual console is like having a full-screen Linux terminal emulator. On desktop systems, you'll frequently see boot messages before the display manager starts. When you run the system in multi-user mode (text mode) you boot to a virtual console. When the system is booting up, boot messages are displayed on a virtual console.

There are several virtual consoles available. To switch among the virtual consoles, you can use the Alt + Function Key system. There are typically seven virtual consoles you can use. If you're running a desktop system with X11, it usually starts in the seventh virtual console. To switch to the first virtual console, you'll have to press Alt + F1. If it doesn't work for you, try Ctrl + Alt + F1 instead. To return to the desktop system, you can press Alt + F7. If you're running the OS inside a Virtual Box Machine, you can switch between virtual consoles by pressing `host key+Fn`.



```
Ubuntu 22.04.1 LTS ubuntu tty5

ubuntu login: user
Password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Se pueden aplicar 0 actualizaciones de forma inmediata.

Last login: Mon Oct 24 11:06:57 CEST 2022 on tty1
user@ubuntu:~$ _
```

Example of a virtual console inside a virtual machine, with a user just logged in. You can see that this is the fifth virtual console because there is a "ubuntu tty5" message on the top of the screen

While virtual consoles are used less frequently now that graphical desktops with display managers are more common, they still have their uses in Linux today. If your desktop session locks up or something else goes wrong, you can switch to a virtual console and log in to try to fix it, or you could gracefully shut down or restart your machine.

If the desktop session fails to start, you'll be dumped into a virtual console session anyway. This is yet another reason why it pays to be familiar with the Linux command line.

Virtual consoles are still used in some distribution installation programs, particularly ones that run in text mode like server distributions. You can switch to another virtual console and see how the installation is going or why an operation seems to be taking a long time. You can also execute other commands in a shell that can be useful if you run into any errors.

3.2 The shell

The *shell* is the interface or program that interprets the command lines sent to the terminal. It works by processing the commands and interpreting the results, i.e. it takes handwritten commands and tells the operating system that they should be executed. When you are running a terminal, or you are in a virtual console, what you really see is the output of the shell. The commands you enter are sent to the shell, which interprets them. It also takes sequences of commands in order to interpret the logic between them. Its main function is to start other programs.

For example: if you open a virtual console and execute the command `whoami`, you will see your user name as the answer. What actually happens when you do that is:

1. The shell receives the command (`whoami`) and interprets it as a command that should be executed
2. The shell runs the program `whoami`, redirecting the user's input to that program (in this case, the program doesn't need an input) and the output to the screen.
3. Once the program has finished, it waits for the next command.

On Linux systems, the shell is configurable. The system administrator can install new shells and configure the default shell for each user.

There are hundreds of shells available for Linux. Nevertheless, the most used ones are:

- **bash** *(Bourne-again shell)*: It's the default login shell in most systems. It has features like a command history, command aliases, expansion, job control, etc.
- **ksh** *(Korn Shell)*: It was developed many years before bash. It runs at a greater speed and has better support for some programming structures like loops or associative arrays.
- **zsh** *(Z-shell)*: It has many similarities to bash and ksh and incorporates many of their main features. But it also incorporates many improvements like changing the directory without entering `cd`, autocompletion and spelling correction, and plugins and themes support.
- **dash** *(Debian Almquist Shell)*: It's ashell designed to be a minimalist shell with a great speed of execution. It's tiny and stable, and i'ts used as the default shell in some systems.

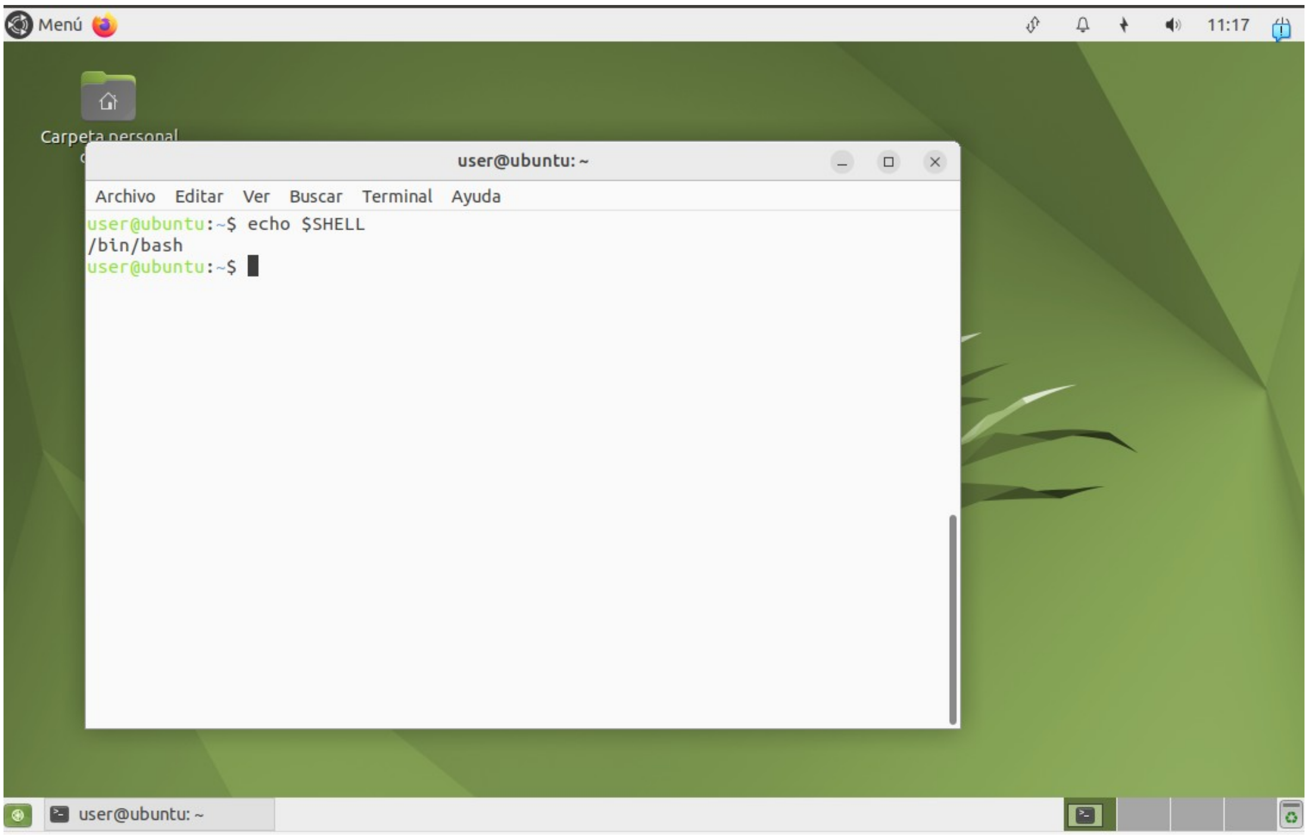
In most systems, `sh` is the default system shell. But this command points to another shell, usually a lightweight one like dash. To know which shell you are using, you can run the command:

```
echo $SHELL
```

The system administrator can change the default shell for an user with the command:

```
usermod --shell /bin/ksh myuser
```

This would change the default shell of myuser to ksh. Some users don't have a shell, because they shouldn't be accessing the system to enter commands. They are usually users running system services.

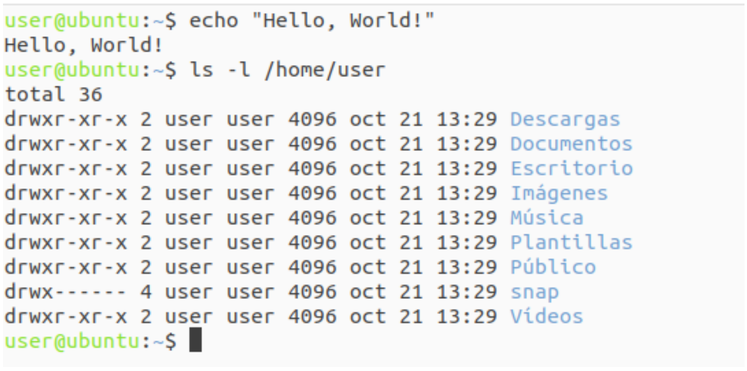


```
user@ubuntu:~$ echo $SHELL
/bin/bash
user@ubuntu:~$
```

A shell running in a graphical environment inside a virtual terminal,.an application that runs like a virtual console or a real terminal attached to the system. The aspect is similar to the virtual terminal because both are running the same program: the shell (bash, in this case)

3.3 Introduction to the console

In the console, you enter commands with arguments. The commands are executed, and the result is printed on the screen. For example:



```
user@ubuntu:~$ echo "Hello, World!"
Hello, World!
user@ubuntu:~$ ls -l /home/user
total 36
drwxr-xr-x 2 user user 4096 oct 21 13:29 Descargas
drwxr-xr-x 2 user user 4096 oct 21 13:29 Documentos
drwxr-xr-x 2 user user 4096 oct 21 13:29 Escritorio
drwxr-xr-x 2 user user 4096 oct 21 13:29 Imágenes
drwxr-xr-x 2 user user 4096 oct 21 13:29 Música
drwxr-xr-x 2 user user 4096 oct 21 13:29 Plantillas
drwxr-xr-x 2 user user 4096 oct 21 13:29 Público
drwx------ 4 user user 4096 oct 21 13:29 snap
drwxr-xr-x 2 user user 4096 oct 21 13:29 Vídeos
user@ubuntu:~$
```

The user has executed two commands:

- `echo "Hello, World!"`: The command is `echo` and the parameter is `"Hello, World!"`
- `ls -l /home/user`: The command is `ls`. The parameters are `-l` and `/home/user`

The parameters modify the behavior of the commands by indicating additional information (e.g. the "Hello, World!" of the first command indicates what should be displayed on the screen) or by selecting different behavior modes (e.g. the -l of the second command indicates that files should be displayed in a certain format).

There are two types of shell commands:

- **non-builtin commands**: They tell the shell to run an external program and display the result. For example, `ls` is a non-builtin command: the shell executes the program `/usr/bin/ls` and displays the results
- **built-in commands**: A built-in command is simply a command that the shell carries out itself, instead of interpreting it as a request to load and run some other program. They are usually faster (because loading and running a program takes time) and can change the internal state of the shell. For example, the built-in command `cd` changes the directory where the shell is running. `echo` is also a built-in command due to performance issues.

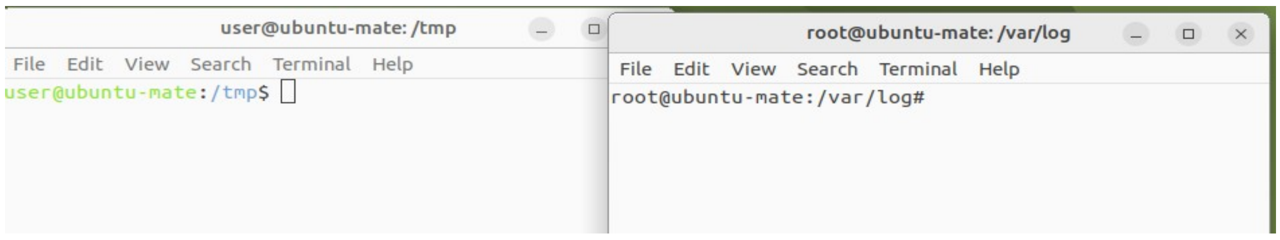
You can differentiate between them using the `which` command

```
which cat

/usr/bin/cat

which pwd

pwd: shell built-in command
```

The second one, `pwd`, is a built-in command.

3.3.1 The prompt

The shell prompt is the text that appears before we enter a command. It usually contains the current user, the computer's name and the current working directory.


user@ubuntu-mate: /tmp
File Edit View Search Terminal Help
root@ubuntu-mate: /var/log
File Edit View Search Terminal Help

```
user@ubuntu-mate:/tmp$
```

```
root@ubuntu-mate:/var/log#
```

Two terminals, one of them with the root user logged in

Most shells allow configuring the prompt. In bash, for example, you can configure the prompt using the variables PS1, PS2 and PS3. If you, for example, execute this command:

```
PS1='\[\e[0m\]\d\[\e[0m\]@\[\e[0m\]\h\[\e[0m\]:\[\e[0m\]'
```

The prompt will display the current date and the computer name.

The syntax of the shell prompt it’s not trivial, but you can use generators like https://scriptim.github.io/bash-prompt-generator/ to create a new prompt.

3.4 Bash basic commands

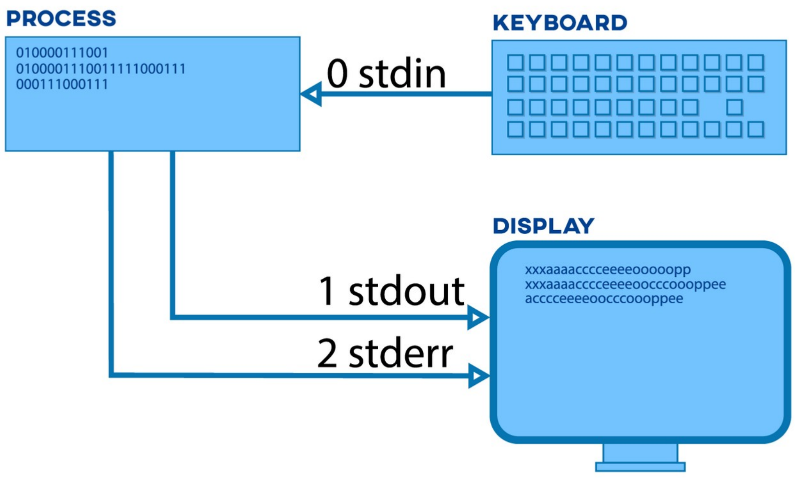
Some of the most basic and useful shell commands and the ones used in the examples are listed here. We will see the commands for managing the file system more in-depth in another unit:

- `pwd`: shows the current directory
- `ls`: list files. For example, `ls -l` shows the files in the current directory
- `cd`: changes the shell's current directory. For example, `cd /` will go to the root of the system. `cd /home/user` will go to the user's directory
- `man`: Shows info about a command: how a command works and its parameters.
- `cat`: shows the content of a file. For example, `cat /etc/hosts`
- `less`: shows the content of a file but one page at a time. It's useful for big files. For example, `less /etc/group`
- `grep`: search contents into a file or the program's input. For example `grep user /etc/paswd` will search for the text "user" inside the file `/etc/passwd`.
- `sort`: sort the contents of a file. For example, `sort /etc/passwd`.
- `wc`: counts lines and words from files. For example `wc -l /proc/devices` will count the lines on the file `/proc/devices`

You can get more information about the commands executing `man <command>`.

3.4.1 Pipes and redirection

In Unix, programs get their input from the standard input. That input is called `stdin` and it's usually the keyboard. The output is sent to the standard output, called `stdout`. If there is an error, it's sent to the standar error, `stderr`. Both `stdout` and stderr are the screen, by default.



- `stdin`: Takes input from the user via keyboard. It has the identifier 0 inside the process.
- `stdout`: Displays output on the screen. It has the identifier 1 inside the process.

- `stderr`: Shows error information on the screen. It has the identifier 2 inside the process.

Redirection

We can redirect the inputs and outputs of a process to a file or to another process. For example, if we execute this:

```
ls 1>listing.txt
```

The output of ls -l will be redirected to the file listing.txt. So if we execute cat listing.txt, we will see the output of the command.

If we want to redirect the error output to a file, we can use:

```
ls "this will raise error" 2> errors.txt
```

We can also redirect both outputs (> is equivalent to 1>):

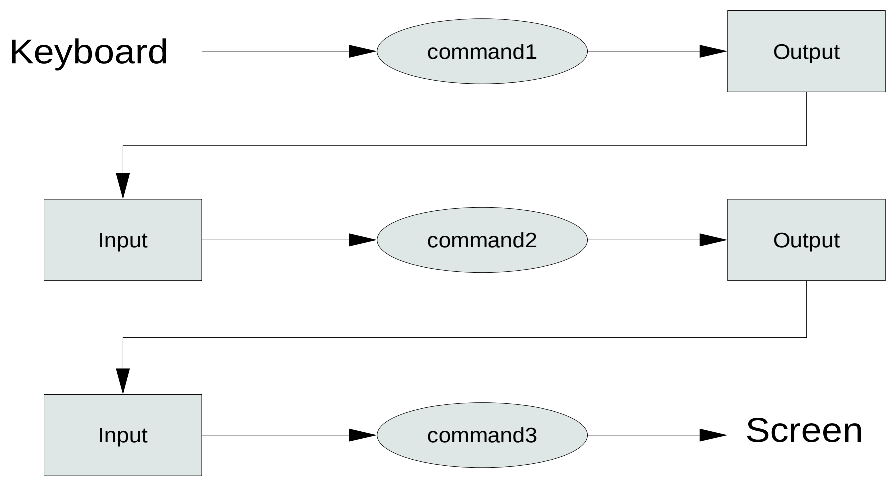
```
ls / /this_is_an_error > listing.txt 2> errors.txt
```

Pipes

Pipes allow to redirect the output of a command to the input of another command. It's like redirection, but we send the output to another program instead of a file. We do that using the sign `|` between command. For example, if we run:

```
command1 | command2 | command3
```

This is what happens:



The Linux philosophy is to use small commands that can be combined together to perform more complex tasks. Using pipes, we can combine commands so that the output of one is the input of another. For example, if we run:

```
ls -l /etc | grep 'in' | wc --lines
```

1. The command `ls -l /etc` is executed. It produces a listing

2. The listing is passed as an input to `grep`. `grep` outputs only the lines in the listing containing the string `in`
3. The lines produced by `grep` are passed as an input to `wc`. `wc` will count the number of lines and output the result to the screen

We could have also redirected the output of the above command to a file adding `> <file>` at the end of the above command.

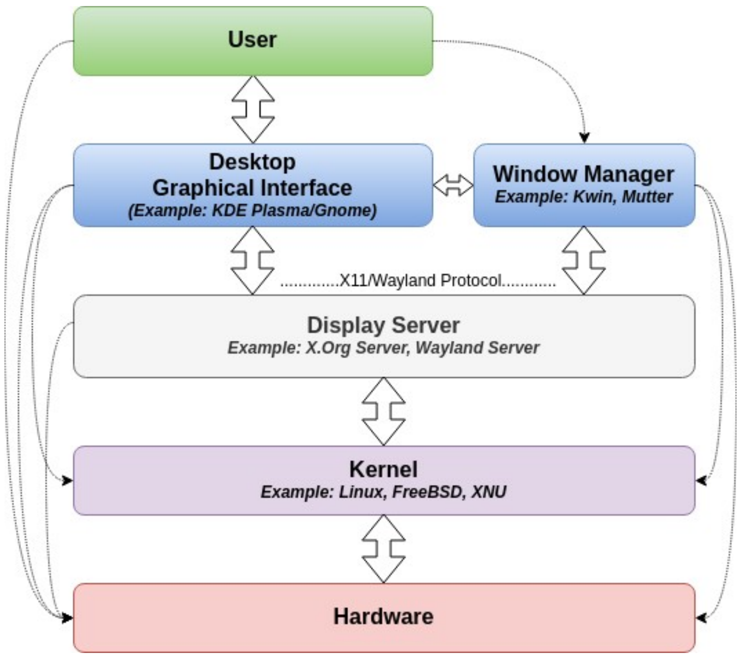
4. The graphic mode

Linux can also run in graphics mode. When we use Linux as a desktop machine, it's the mode that almost every user uses: It allows us to interact with the system in an easy and comfortable way.

The purpose of this topic is not to teach how to use a graphical system, as this is rather intuitive and dependent on the Linux desktop version used, but to learn about the architecture of the graphic mode.

4.1 The Linux display system

This is an overview of the Linux graphic architecture:


User
Desktop Graphical Interface
(Example: KDE Plasma/Gnome)
Window Manager
Example: Kwin, Mutter
............X11/Wayland Protocol...........
Display Server
Example: X.Org Server, Wayland Server
Kernel
Example: Linux, FreeBSD, XNU
Hardware

We will review this diagram from the bottom (the lowest-level components) to the top (the high-level components).

The lower layers are the hardware and the kernel. The hardware is the devices that we have in our computer: graphics cards, screens, mouse, keyboard, etc. Those devices can only be accessed through the kernel. The kernel receives the commands from the display server, which is the layer we will discuss next.

4.2 Display servers

A display server is a program whose primary task is to coordinate the input and output of its clients to and from the rest of the operating system, the hardware, and each other. The display server communicates with its clients over the display server protocol.

For instance, display servers manage the mouse and help match the mouse movements with the cursor and GUI events caused by the cursor. The display server also provides various protocols and communicates with the kernel directly. There are different sets of display server protocols and different display servers that implement a specific protocol. In this document we will talk about X windows system and Wayland.

The display server is something internal to the applications. The user probably does not know which display server is being used.

4.2.1 X windows System

The X windows system has two part: the X protocol and the X implementation. All applications and servers that implement the X protocol should be compatible.

The X window system is called that way because it is the successor of the W window system.

The X11 protocol

X is a network protocol. It has been in version 11 since 1987, so it's usually called X11. It describes how messages are exchanged between a client (application) and the display (server). These messages typically carry primitive drawing commands like "draw a box," "write these characters at this position," "the left mouse button has been clicked," etc.

In X11, the clients are the applications and the server is where the applications are drawn.

When X11 was developed, users launched an X server on their desktops (where the apps were visualized) and run the X clients (the apps) on a more powerful computer. Today, the X clients and the X server usually run on the same machine.

The X implementation

Implementing the X11 protocol is a very complex task. The X.Org project provides a reference implementation for the protocol that can be integrated in the operating systems.

4.2.2 Wayland

Wayland is a replacement for the X11 Windows system. It's aimed to be easier to develop, extend and maintain. The main difference between the X windows and Wayland system architectures is in the role of the compositor. The compositor is the software that combines all the windows to create the final image that will be shown to the user.

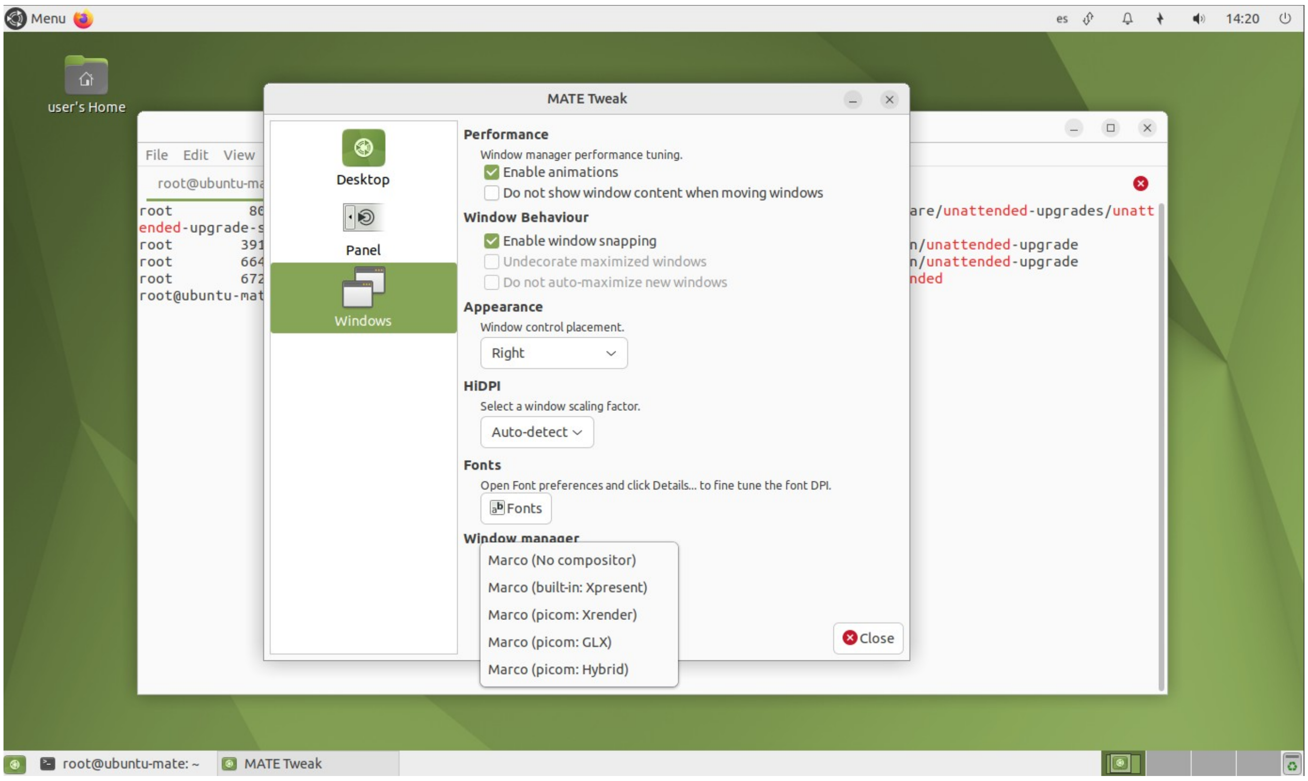
Most of the modern Linux distributions has adopted Wayland as their default display server, but provides a layer for compatibility with X windows system.

If you want to know more about how X differs from Wayland, you can check this link: https://wayland.freedesktop.org/architecture.html

4.3 The window manager

The display server provides the basic primitives for drawing on the screen. But modern graphical interfaces use windows to display applications. The window manager controls the placement and decoration of windows. It is the window manager that adds the title bar to the window, the control buttons —usually associated with the minimize, maximize and close actions— and manages the switching between open windows.

The window manager is used by the desktop environment, but you can usually install and choose another one.



```
root          86
ended-upgrade-s
root         391
root         664
root         672
root@ubuntu-mat
```

```
are/unattended-upgrades/unatt

n/unattended-upgrade
n/unattended-upgrade
nded
```

Performance
Window manager performance tuning.
- [✓] Enable animations
- [ ] Do not show window content when moving windows

Window Behaviour
- [✓] Enable window snapping
- [ ] Undecorate maximized windows
- [ ] Do not auto-maximize new windows

Appearance
Window control placement.
Right

HiDPI
Select a window scaling factor.
Auto-detect

Fonts
Open Font preferences and click Details... to fine tune the font DPI.
Fonts

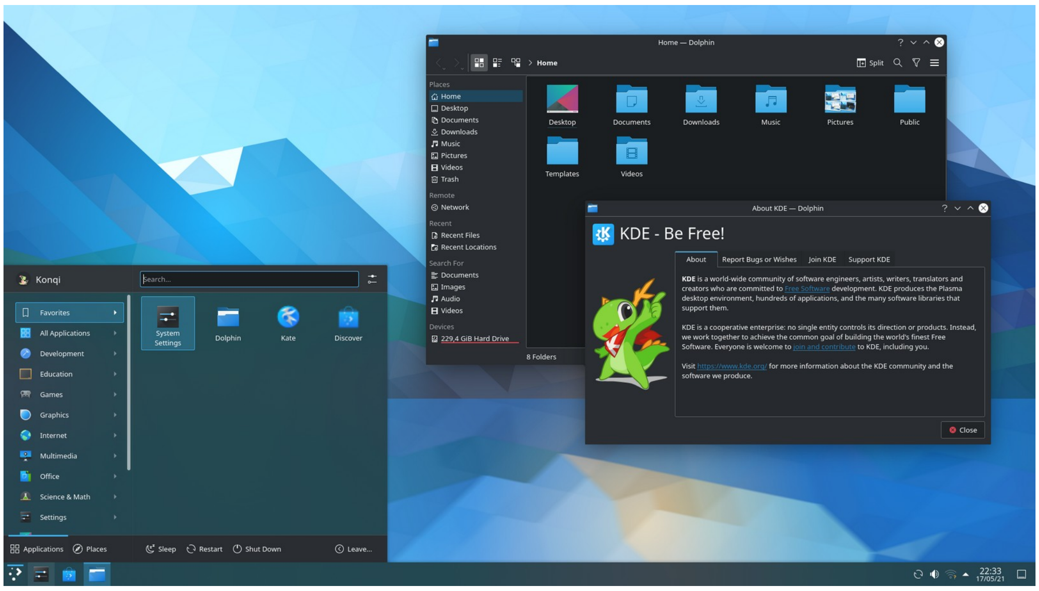
Window manager
- Marco (No compositor)
- Marco (built-in: Xpresent)
- Marco (picom: Xrender)
- Marco (picom: GLX)
- Marco (picom: Hybrid)

Selecting a window manager for a Mate desktop environment

4.4 Desktop environments

A desktop environment is what you see when you enter a graphical interface in Linux systems. We can run individual programs on a display server but what we usually want is an integrated environment that allows us to launch multiple applications, has a toolbar and a menu, etc. All this is provided by the desktop environment. They provide common graphical user interface (GUI) elements such as icons, toolbars, wallpapers, and desktop widgets. Thanks to the desktop environment, you can use Linux graphically using your mouse and keyboard like you do in other operating systems like Windows and macOS.

Most of the desktop environments have their own set of integrated applications and utilities so that users get a uniform feel while using the OS. So, you get a file explorer, desktop search, menu of applications, text editors and more.


About KDE — Dolphin

KDE - Be Free!

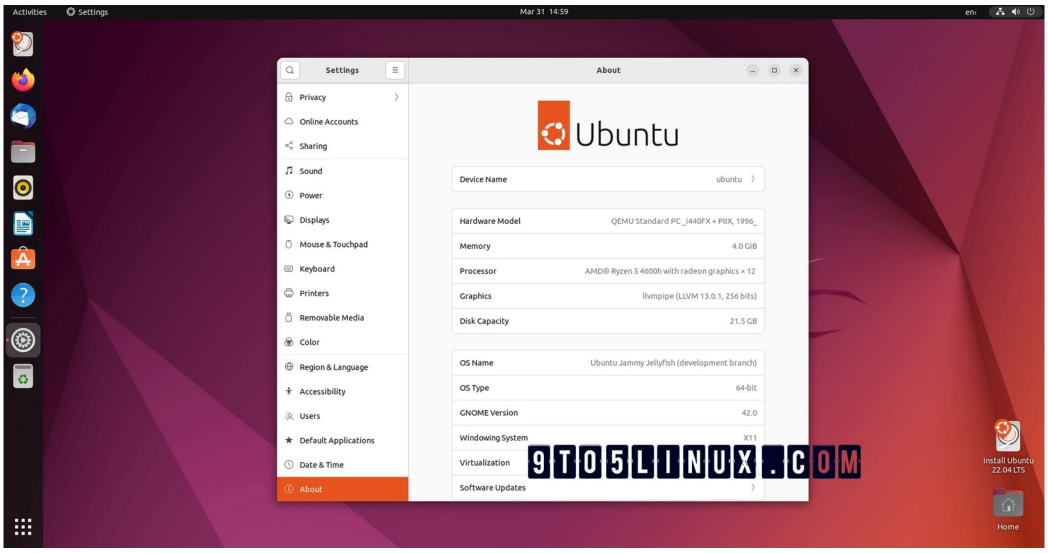
About | Report Bugs or Wishes | Join KDE | Support KDE

KDE is a world-wide community of software engineers, artists, writers, translators and creators who are committed to Free Software development. KDE produces the Plasma desktop environment, hundreds of applications, and the many software libraries that support them.

KDE is a cooperative enterprise: no single entity controls its direction or products. Instead, we work together to achieve the common goal of building the world's finest Free Software. Everyone is welcome to join and contribute to KDE, including you.

Visit https://www.kde.org/ for more information about the KDE community and the software we produce.

Plasma 5.22 desktop environment


About

Ubuntu

Device Name	ubuntu
Hardware Model	QEMU Standard PC _i440FX + PIIX, 1996_
Memory	4.0 GiB
Processor	AMD® Ryzen 5 4600h with radeon graphics × 12
Graphics	llvmpipe (LLVM 13.0.1, 256 bits)
Disk Capacity	21.5 GB
OS Name	Ubuntu Jammy Jellyfish (development branch)
OS Type	64-bit
GNOME Version	42.0
Windowing System	X11
Virtualization	
Software Updates	

Gnome 42 desktop environment

There are several desktop environments available for Linux, and these desktop environments determines what your Linux system looks like and how you interact with it. Some of the most used desktop environments are:

- **GNOME**: The GNOME environment is the most used in Linux, to the point where many people believe it is part of it. Distributions such as Debian and Fedora use GNOME as their standard default environment
- **KDE plasma**: The most widely used Linux desktop environment after GNOME. It's used in distributions like Kubuntu or openSuse.
- **Mate**: Provides the most traditional desktop experience. It is the continuation of GNOME 2.
- **LXDE**: stands for Lightweight X11 Desktop Environment. It uses minimal RAM compared to other desktop environments since it uses so few resources. It's better suited to old computers with outdated hardware.

Linux distributions ship with a default desktop environment, but you can install new ones and change them at boot time.

4.5 The display manager

We can boot the computer in text mode and then launch the graphical environment. But often what we want is to boot directly into graphical mode. This is where the display manager comes in.

A session manager is a program that allows us to log on to the computer. For example: when the computer starts in text mode, it starts a program (usually getty or agetty) that asks the user for a username and a password. It will validate if the user is allowed to log on to the system, and launch the shell after that.

If the session manager runs in graphic mode and allows us to open a graphic session into a computer, it's called a display manager. Linux distributions ship with a display manager, but we can change it if we want. Some of the most used display managers are:

- **GNOME Display Manager** (**GDM**): It's the default display manager for GNOME
- **Simple Desktop Display Manager** (**SDDM**): It's a lightweight display manager that was written from scratch in C++ and supports theme creation. This manager is gaining popularity over time. It is the successor of KDE and is used in collaboration with KDE plasma.
- **KDE Display Manager** (**KDM**): This was the old display manager for KDE4, which was based on XDM (an older display manager)
- **Light Display Manager** (**LightDM**): It's a cross-desktop display manager. Supports different desktop technologies, different display technologies (X, Wayland ...) and it's lightweight, having low memory usage and high performance.

5. Logs

Logs or system logs are text files that chronologically record all important activities and events that occur on the operating system or network. Linux records absolutely everything that happens in the operating system through logs.

Some examples of the information contained in Linux logs are as follows:

- Packages that are installed and uninstalled in the operating system.
- Information about remote accesses to your computer.
- Failed attempts to authenticate users to the computer.
- Logging of errors that occur in the programs or services we use.
- Access or exits blocked by our firewall.
- Etc.

Almost all logs are stored in the /var/log directory and its subdirectories. Some of the most relevant logs are:

- /var/log/syslog or /var/log/messages: general messages, as well as system-related information. Essentially, this log stores all activity data across the global system.
- /var/log/auth.log or /var/log/secure: store authentication logs, including both successful and failed logins and authentication methods.
- /var/log/boot.log: registers information related to booting and any messages logged during startup.
- /var/log/kern: stores Kernel logs and warning data. This log is valuable for troubleshooting custom kernels as well.
- /var/log/dmesg: messages relating to device drivers. The command dmesg can be used to view messages in this file.
- /var/log/dpkg.log: Package management log for Debian and Ubuntu, which records which packages were installed and/or uninstalled on the system. For Red Had and Fedora environments the file is the file: /var/log/rpmpkgs

To display a log, you can use the less command. For example, to display the contents of the syslog you can use:

```
less /var/log/syslog
```

You can also use the `tail` command. What tail does is output the last part of files. So, if you issue the command:

```
tail /var/log/syslog
```

it will print out only the last few lines of the `syslog` file. When you issue the command with the -f argument:

```
tail -f /var/log/syslog
```

`tail` will continue watching the log file and print out the next line written to the file

Another important command for querying logs is the `dmesg` command. It will show the kernel messages. The `-T` argument is for displaying the time of the message:

```
dmesg -T
```

Example output:

```
[Tue Oct 25 15:27:17 2022] systemd[1]: Starting Load Kernel Module pstore_blk...
[Tue Oct 25 15:27:17 2022] systemd[1]: Starting Load Kernel Module pstore_zone...
[Tue Oct 25 15:27:17 2022] systemd[1]: Starting Load Kernel Module ramoops...
[Tue Oct 25 15:27:17 2022] systemd[1]: Condition check resulted in File System Check on Root
Device being skipped.
[Tue Oct 25 15:27:17 2022] systemd[1]: Starting Load Kernel Modules...
[Tue Oct 25 15:27:17 2022] systemd[1]: Starting Remount Root and Kernel File Systems...
[Tue Oct 25 15:27:17 2022] systemd[1]: Starting Coldplug All udev Devices...
[Tue Oct 25 15:27:17 2022] systemd[1]: Mounted Huge Pages File System.
[Tue Oct 25 15:27:17 2022] systemd[1]: Mounted POSIX Message Queue File System.
[Tue Oct 25 15:27:17 2022] systemd[1]: Mounted Kernel Debug File System.
[Tue Oct 25 15:27:17 2022] systemd[1]: Mounted Kernel Trace File System.
[Tue Oct 25 15:27:17 2022] systemd[1]: Finished Create List of Static Device Nodes.
```

You can combine `dmesg` with `grep` and `less` using pipes and search for some information:

```
dmesg | grep <something> | less
```

This command is invaluable when troubleshooting hardware-related errors, warnings, and for diagnosing device failure.

6. Supplementary material

The Linux terminal:

- https://www.makeuseof.com/what-are-linux-virtual-consoles/
- https://www.linuxbabe.com/command-line/linux-terminal

User interfaces and desktops:

- https://unix.stackexchange.com/questions/596894/how-does-linuxs-display-work
- https://learning.lpi.org/en/learning-materials/102-500/106/106.1/106.1_01/
- https://learning.lpi.org/en/learning-materials/102-500/106/106.2/106.2_01/