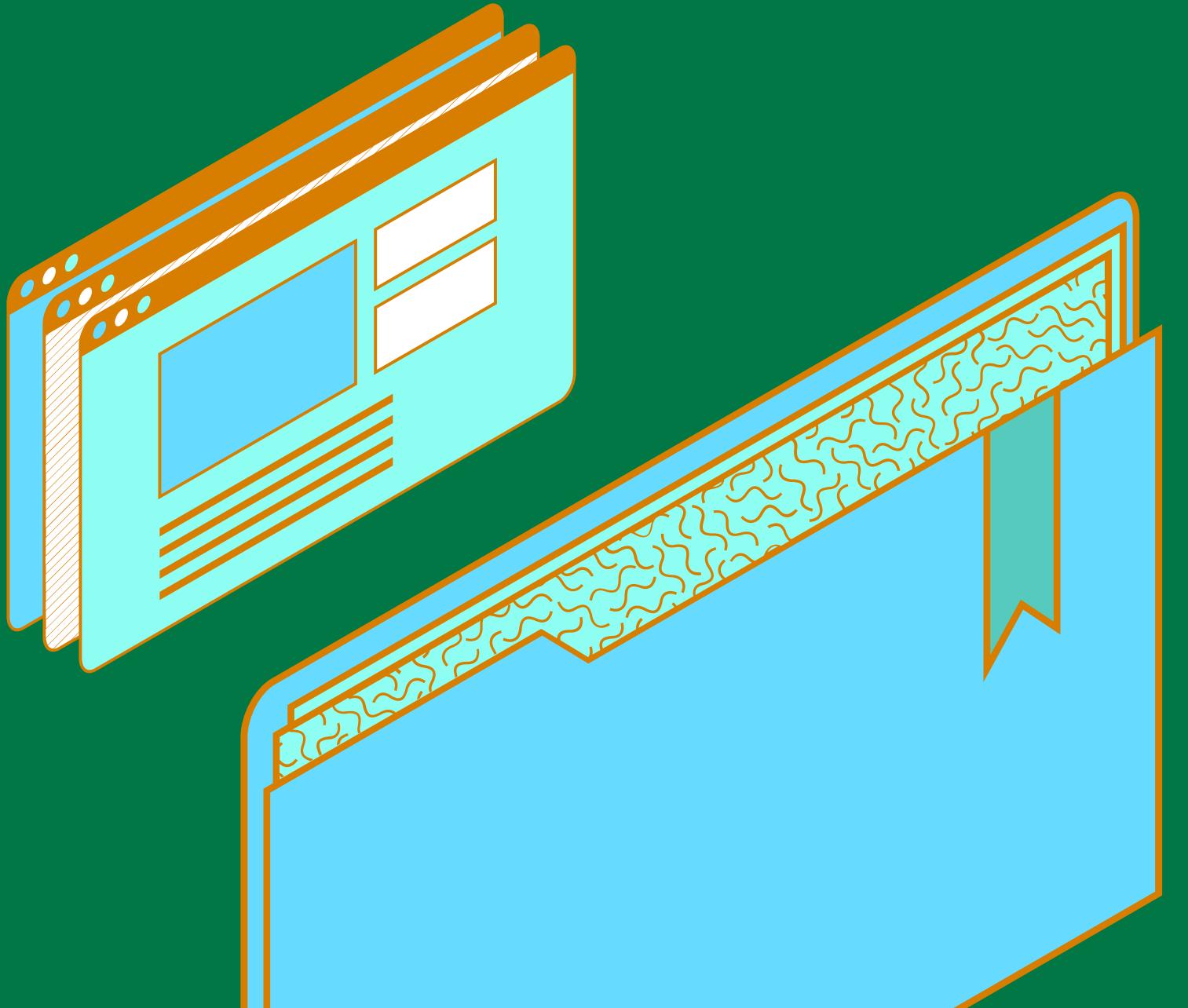


Sistemas operativos

# UNIX Y SU PAPEL EN LA HISTORIA DE LOS SISTEMAS OPERATIVOS

Por Acosta Alan y Rubio Angel

# ÍNDICE



---

01. Introducción

---

02. Historia y motivación

---

03. EL FUTURO DE UNIX

---

04. Equipos que revolucionaron UNIX

---

05. Arquitectura de UNIX

---

06. Sistema de archivos

---

07. Comandos Basicos

---

08. Comandos que revolucionaron

---

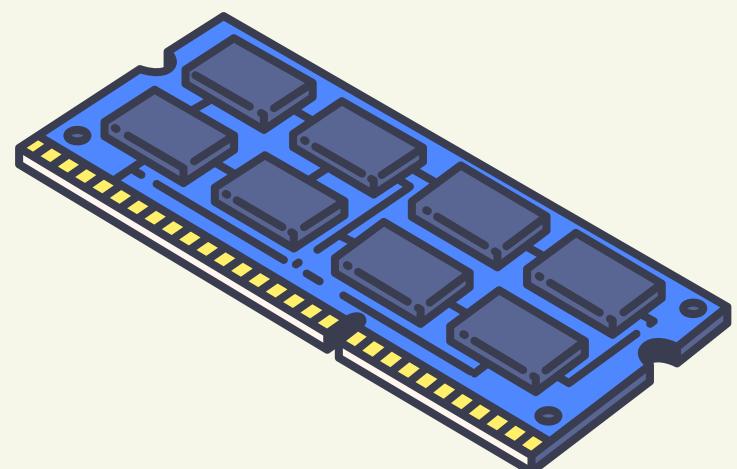
# Introducción

**Unix es un sistema operativo multitarea y multiusuario que surgió en 1969 en los laboratorios Bell (Bell Labs), creado por Ken Thompson y Dennis Ritchie.**



A pesar de su modesta creación, Unix cambió la historia de la informática.

Linux, macOS, y otros sistemas modernos son derivados de Unix.



Unix se convirtió en una base para muchas innovaciones en programación y en el desarrollo de sistemas operativos modernos.

"Una Política, Un Sistema, Servicio Universal" Declaración de misión de  
AT&T, 1907

"A primera vista, cuando uno se encuentra con él en su entorno sorprendentemente rural, el sitio principal de Bell Telephone Laboratories en Nueva Jersey parece una fábrica grande y moderna, que en cierto sentido lo es. Pero es una fábrica de ideas, por lo que sus líneas de producción son invisibles".

Arthur Clarke, Voz al otro lado del mar, 1974 citado en The Idea Factory de Jon Gertner, 2012

# Historia y motivación



# El inicio en los laboratorios Bell

Como parte de este acuerdo, AT&T dirigió una pequeña fracción de sus ingresos a Bell Labs, con el propósito expreso de mejorar las comunicaciones

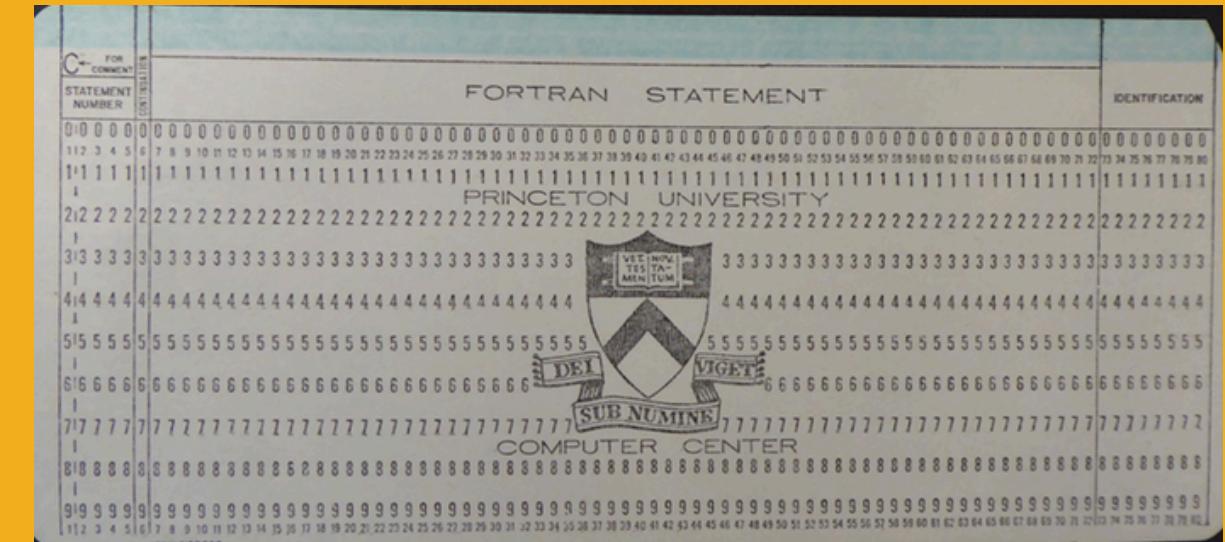
# "procesamiento por lotes"

El sistema operativo más innovador de la época fue el CTSS, el Compatible Time-Sharing System, creado en el MIT en 1964

En 1969, el panorama de los sistemas operativos era muy diferente. Sistemas como Multics eran complejos y costosos



# UNIX



Instituto de Tecnología de Massachusetts  
reclutó a otras dos organizaciones:

**GENERAL  
ELECTRIC**



**General  
Electric**

**Bell Labs**



**AT&T  
Bell Laboratories**

# Multics

Ken Thompson estaba trabajando en Multics. Debido a la complejidad del proyecto y los problemas de integración, se unió con Dennis Ritchie

“crear un sistema más simple y eficiente”

**La frase "sobre-ingeniería" aparece en varias descripciones, y Sam Morgan la describió como "un intento de trepar a demasiados árboles a la vez"**

**Ken encontró un  
“DEC PDP-7”**



**El PDP-7 se envió por primera vez en 1964.**

**Un jugador podría vagar por el sistema solar y aterrizar en diferentes planetas.**

No era muy potente, solo 8K palabras de 18 bits de memoria (16K bytes), pero tenía una **buenas pantalla gráfica**

## PDP-11



El sistema se trasladó a una nueva computadora, que ya tenía más capacidad y mejor soporte para multitarea. la PDP-11 fue la máquina más importante utilizada en los laboratorios Bell para el desarrollo de Unix

Con 24 KB de memoria y discos más grandes, el PDP-11 ofreció un mejor rendimiento

El dinero para un PDP-11 provino de Max Matthews



Director del Centro de Investigación del Habla y la Acústica.

Max apoyó porque uno de sus jefes de departamento, Lee McMahon, estaba muy interesado en el procesamiento de textos y, junto con Ossanna, fue un promotor del plan.

# ARQUITECTURA DE UNIX

KERNEL, SHELL Y SISTEMA DE ARCHIVOS

# KERNEL Y SHELL

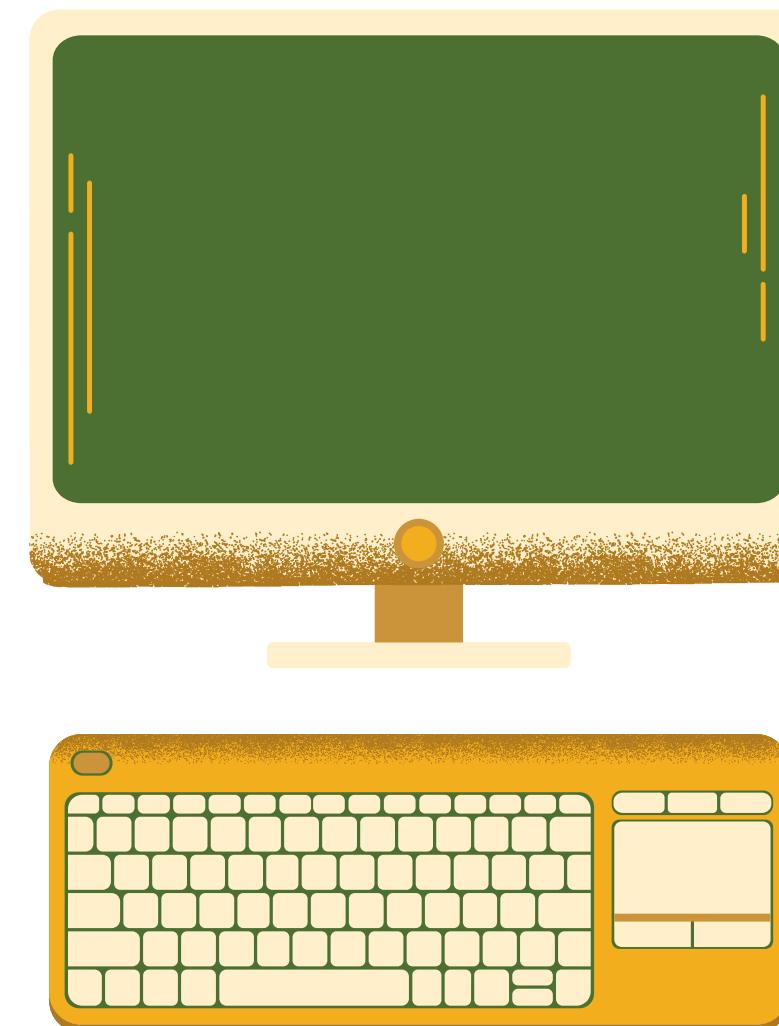
Shell y el Kernel trabajan en conjunto

## KERNEL

El núcleo de UNIX es el centro del sistema operativo

asigna tiempo y memoria a los programas

maneja el almacén de archivos y las comunicaciones en respuesta a las llamadas del sistema.



## SHELL

- El shell es el programa que permite al usuario interactuar con el sistema operativo, generalmente a través de una línea de comandos

El shell interpreta los comandos que el usuario escribe y luego interactúa con el núcleo para ejecutar esos comandos.

Para ilustrar la interacción entre el shell y el kernel, supongamos que un usuario escribe **rm myfile** (lo que elimina el archivo **myfile** ).

rm myfile

## PASO 1: EL SHELL INTERPRETA EL COMANDO

Cuando el usuario escribe el comando rm myfile

El shell primero debe **buscar** el programa rm en el sistema de archivos.

El shell busca en el sistema de archivos para encontrar la ruta de ese programa

## PASO 2: LLAMADA AL SISTEMA (SYSTEM CALL)

una llamada al sistema es una interfaz que permite a los programas de usuario (como el shell) solicitar servicios del núcleo



## PASO 3: EL NÚCLEO EJECUTA EL COMANDO

Asigna tiempo de CPU para que el programa rm se ejecute.

Accede al sistema de archivos para buscar el archivo myfile y luego lo elimina.

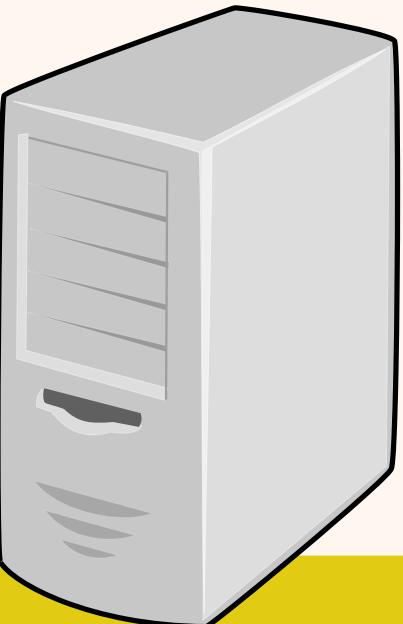
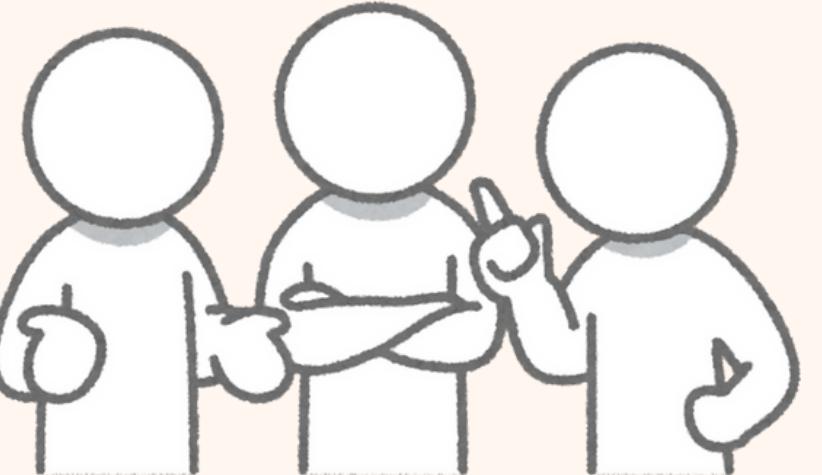
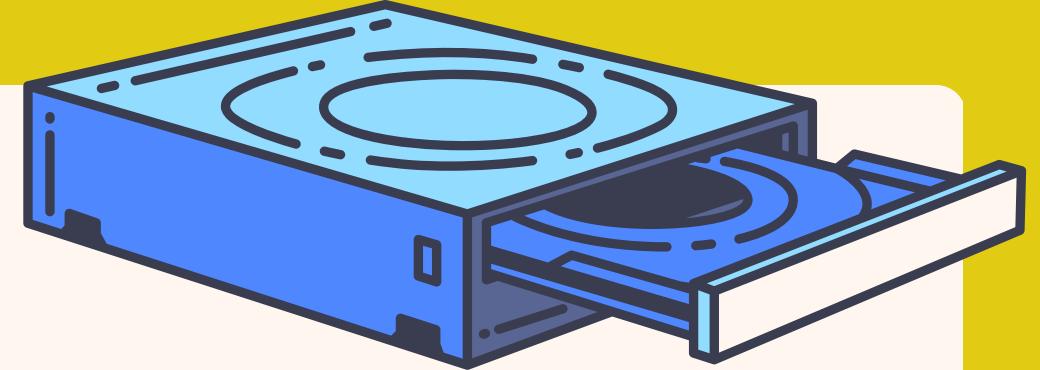
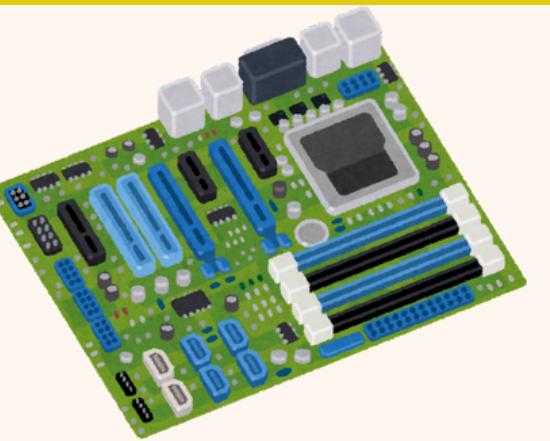
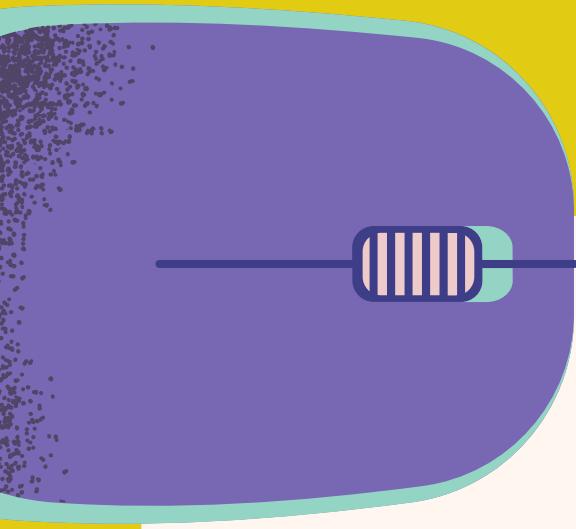
El núcleo también gestiona la memoria utilizada

## PASO 4: FINALIZACIÓN DEL COMANDO

Una vez que el programa rm ha terminado de eliminar el archivo myfile, el proceso de eliminación se completa y el control regresa al shell.

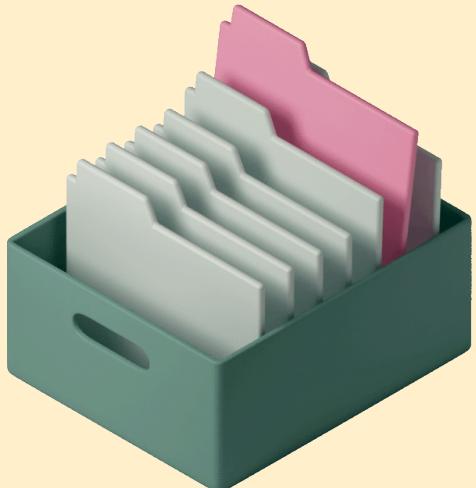
# SHELL Y KERNELL

```
rubio@MacBook-Air-de-Jose: ~ % echo "Soy el shell"
Soy el shell
rubio@MacBook-Air-de-Jose: ~ % echo $$SHELL
/bin/zsh
rubio@MacBook-Air-de-Jose: ~ % uname -a
Darwin MacBook-Air-de-Jose.local 24.2.0 Darwin Kernel Version 24.2.0: Fri Dec 6 18:40:14 PST 2024; root:xnu-11215.61.5~2/RELEASE_ARM64_T8103 arm64
rubio@MacBook-Air-de-Jose: ~ %
```

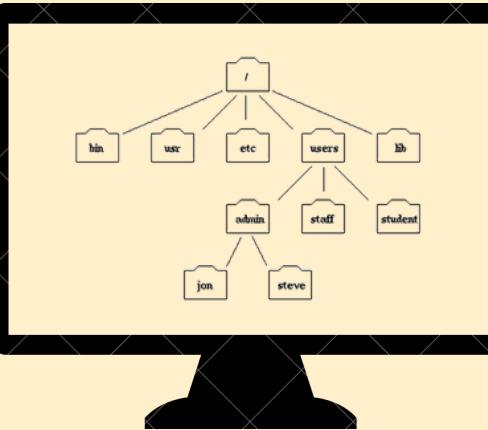


# SISTEMA DÉ ARCHIVOS

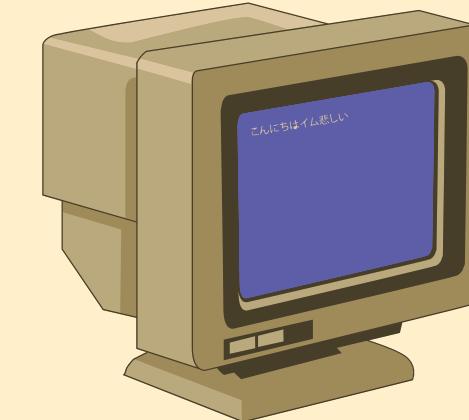
**“En Unix, todo es un archivo”**



**ARCHIVOS  
REGULARES:**



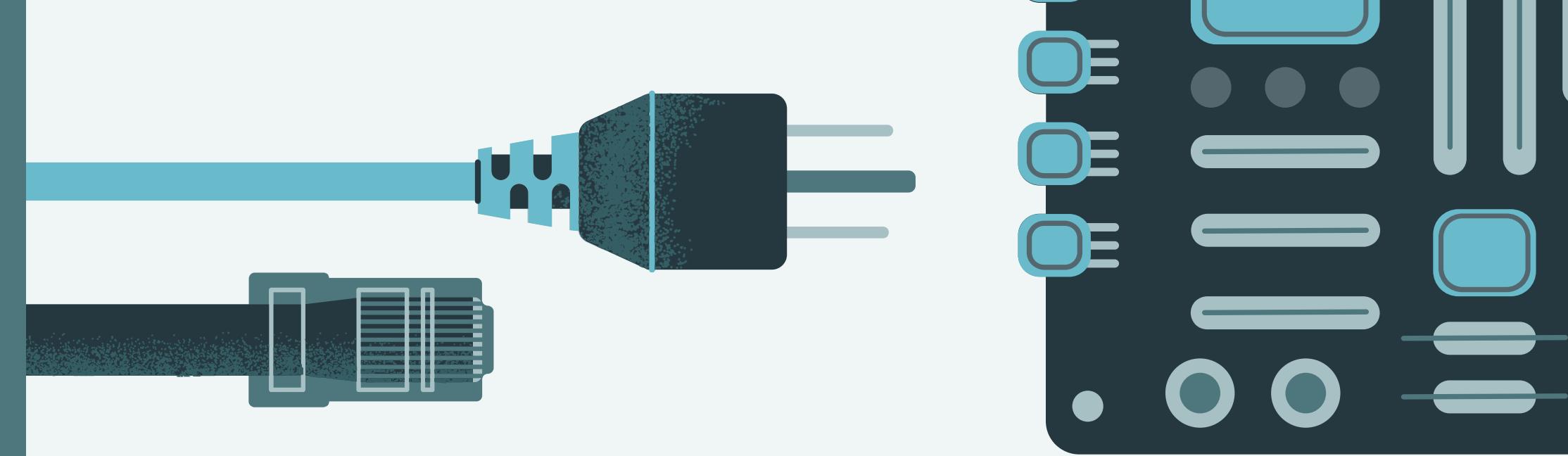
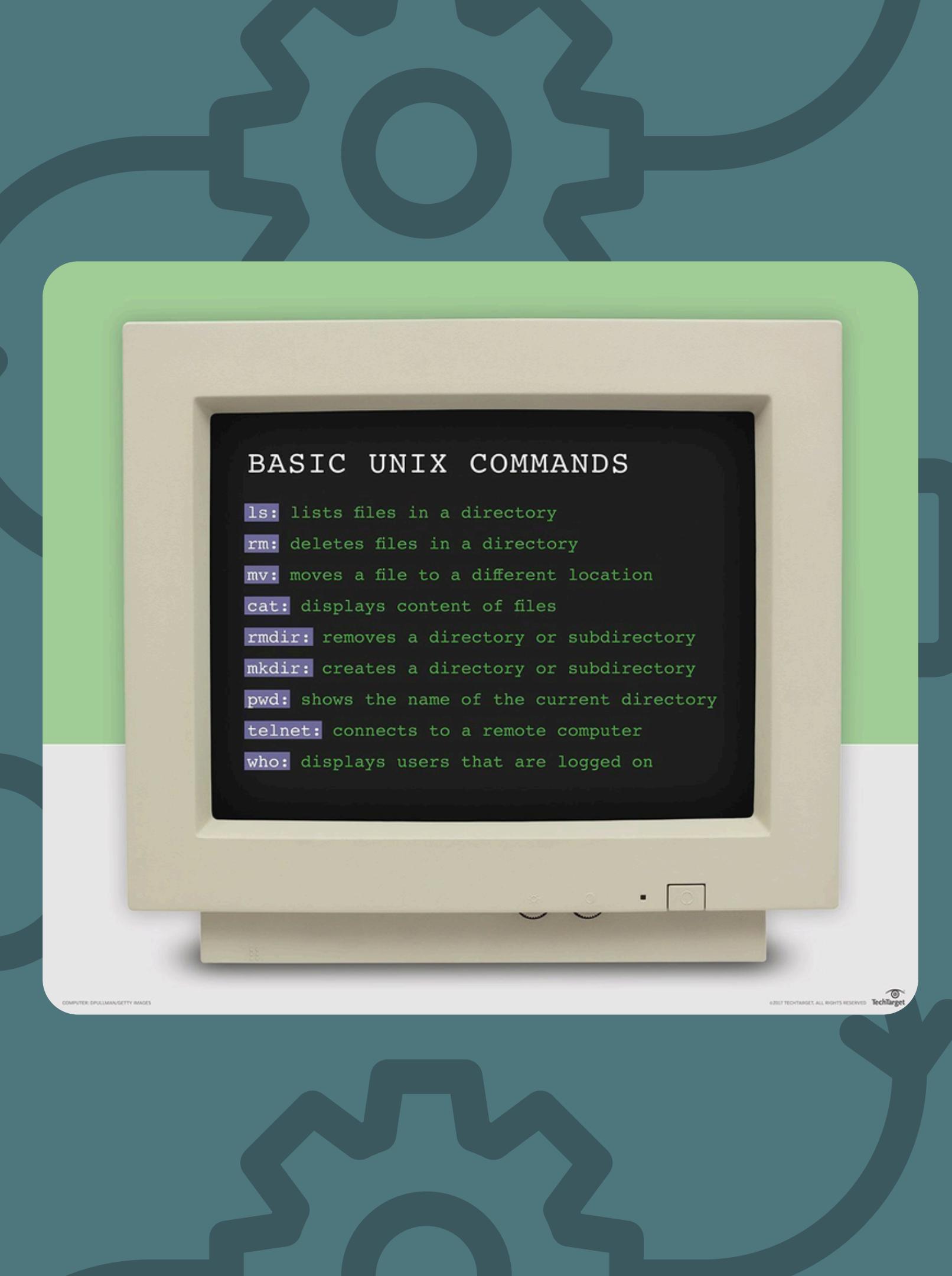
**DIRECTORIOS**



**ARCHIVOS DE  
DISPOSITIVO:**



**ENLACES  
SIMBÓLICOS**



# COMANDOS BASICOS DE UNIX

Unix se caracteriza por ofrecer comandos concisos, fáciles de recordar y orientados al usuario, los cuales permiten acceder rápidamente a información relevante del sistema.

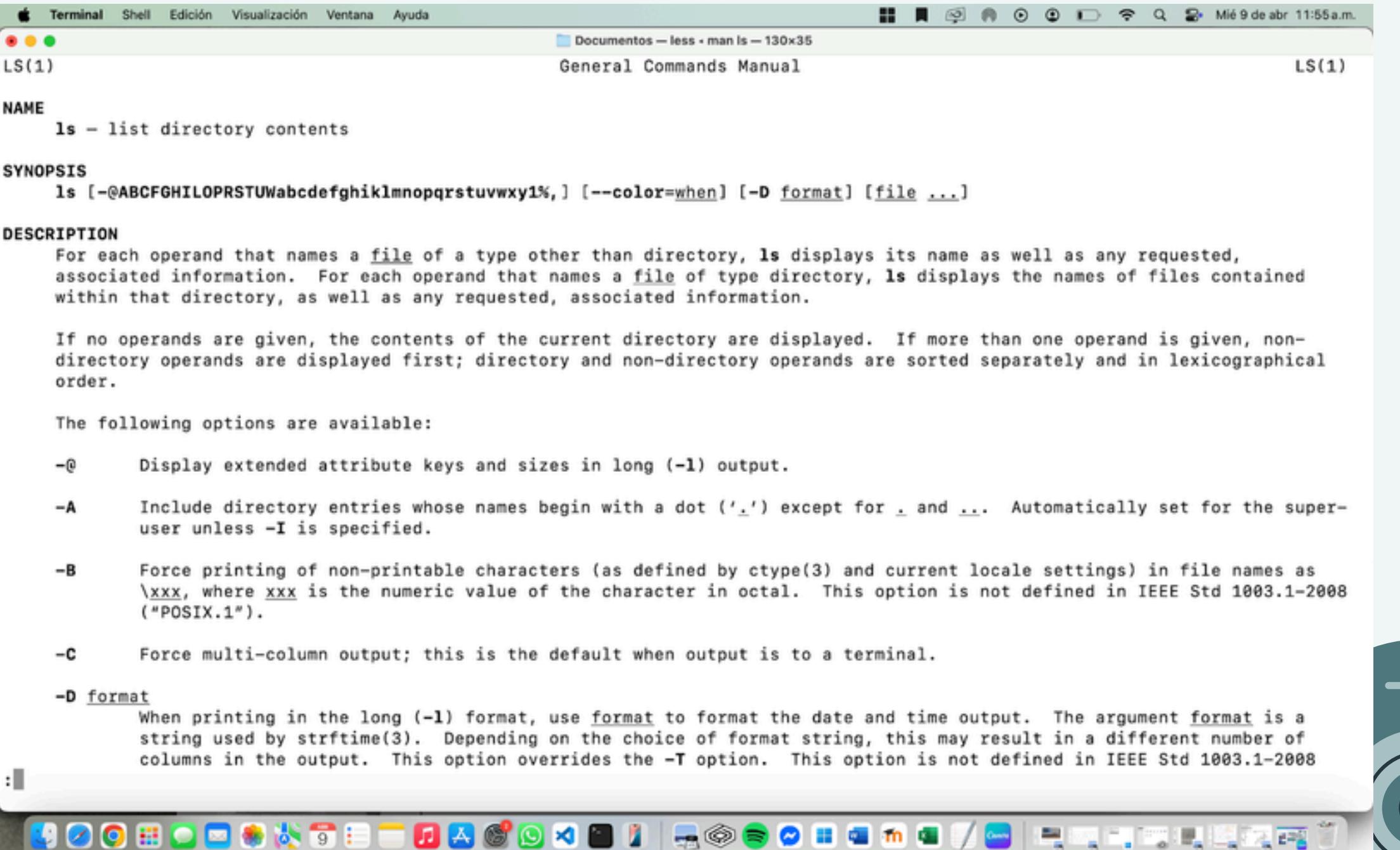
## BASIC UNIX COMMANDS

```
ls: lists files in a directory
rm: deletes files in a directory
mv: moves a file to a different location
cat: displays content of files
rmdir: removes a directory or subdirectory
mkdir: creates a directory or subdirectory
pwd: shows the name of the current directory
telnet: connects to a remote computer
who: displays users that are logged on
```

# COMANDO “MAN”

El comando man sustituyó la consulta tradicional al “Programmer’s Manual.”

Se utiliza escribiendo “man” seguido del comando que se desea consultar, por ejemplo: man ls.



The image shows a Mac OS X desktop with a terminal window open. The title bar reads "Documentos - less - man ls - 130x35" and "General Commands Manual". The terminal content is the man page for the 'ls' command, starting with the NAME section:

```
NAME
  ls - list directory contents

SYNOPSIS
  ls [-@ABCFGHIOPRSTUWabcdefghijklmnopqrstuvwxyz1%,] [--color=when] [-D format] [file ...]

DESCRIPTION
  For each operand that names a file of a type other than directory, ls displays its name as well as any requested,
  associated information. For each operand that names a file of type directory, ls displays the names of files contained
  within that directory, as well as any requested, associated information.

  If no operands are given, the contents of the current directory are displayed. If more than one operand is given, non-
  directory operands are displayed first; directory and non-directory operands are sorted separately and in lexicographical
  order.

  The following options are available:

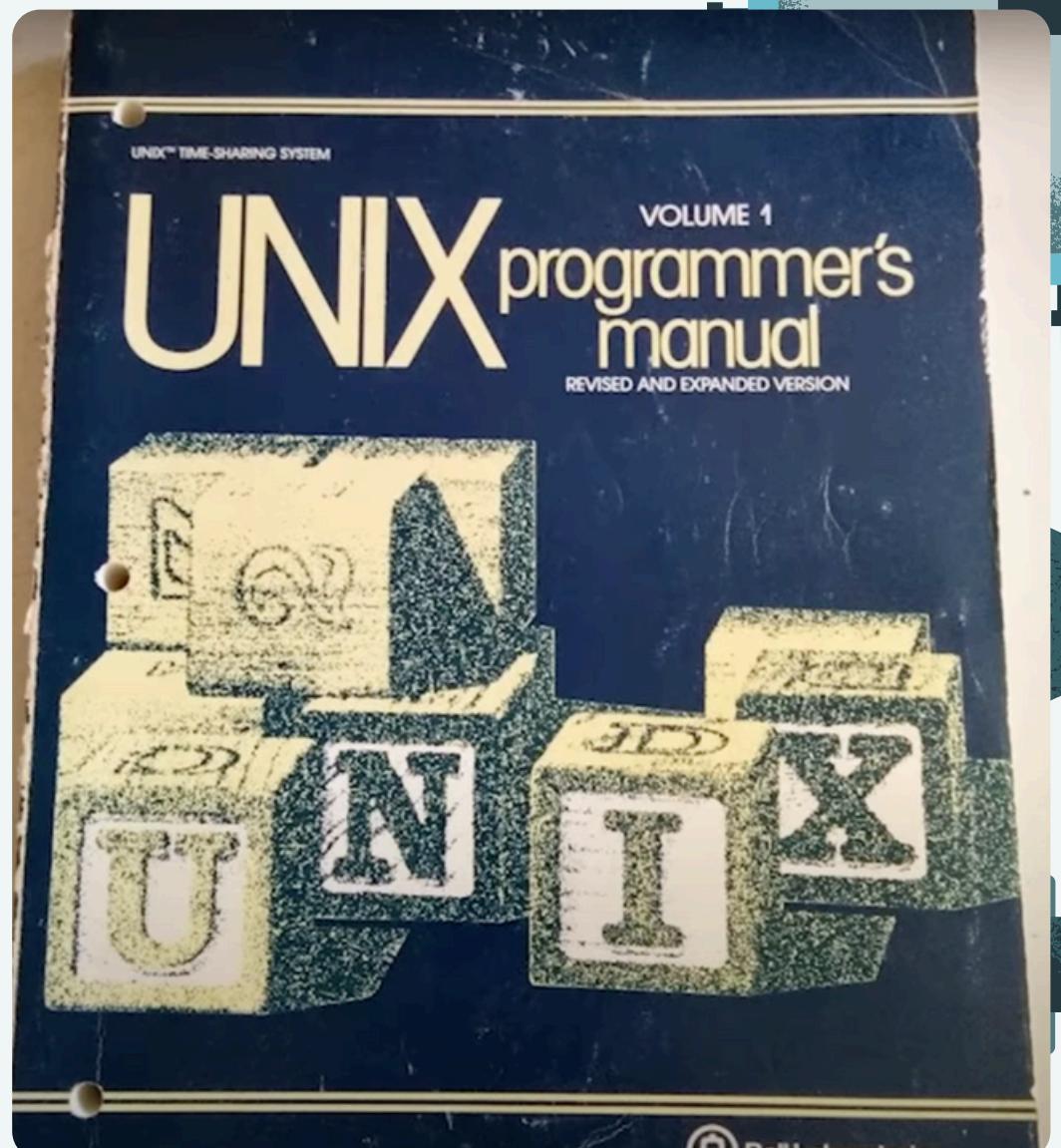
  -@      Display extended attribute keys and sizes in long (-l) output.

  -A      Include directory entries whose names begin with a dot ('.') except for . and ... Automatically set for the super-
         user unless -I is specified.

  -B      Force printing of non-printable characters (as defined by ctype(3) and current locale settings) in file names as
         '\xxx, where xxx is the numeric value of the character in octal. This option is not defined in IEEE Std 1003.1-2008
         ("POSIX.1").

  -C      Force multi-column output; this is the default when output is to a terminal.

  -D format
    When printing in the long (-l) format, use format to format the date and time output. The argument format is a
    string used by strftime(3). Depending on the choice of format string, this may result in a different number of
    columns in the output. This option overrides the -T option. This option is not defined in IEEE Std 1003.1-2008
```

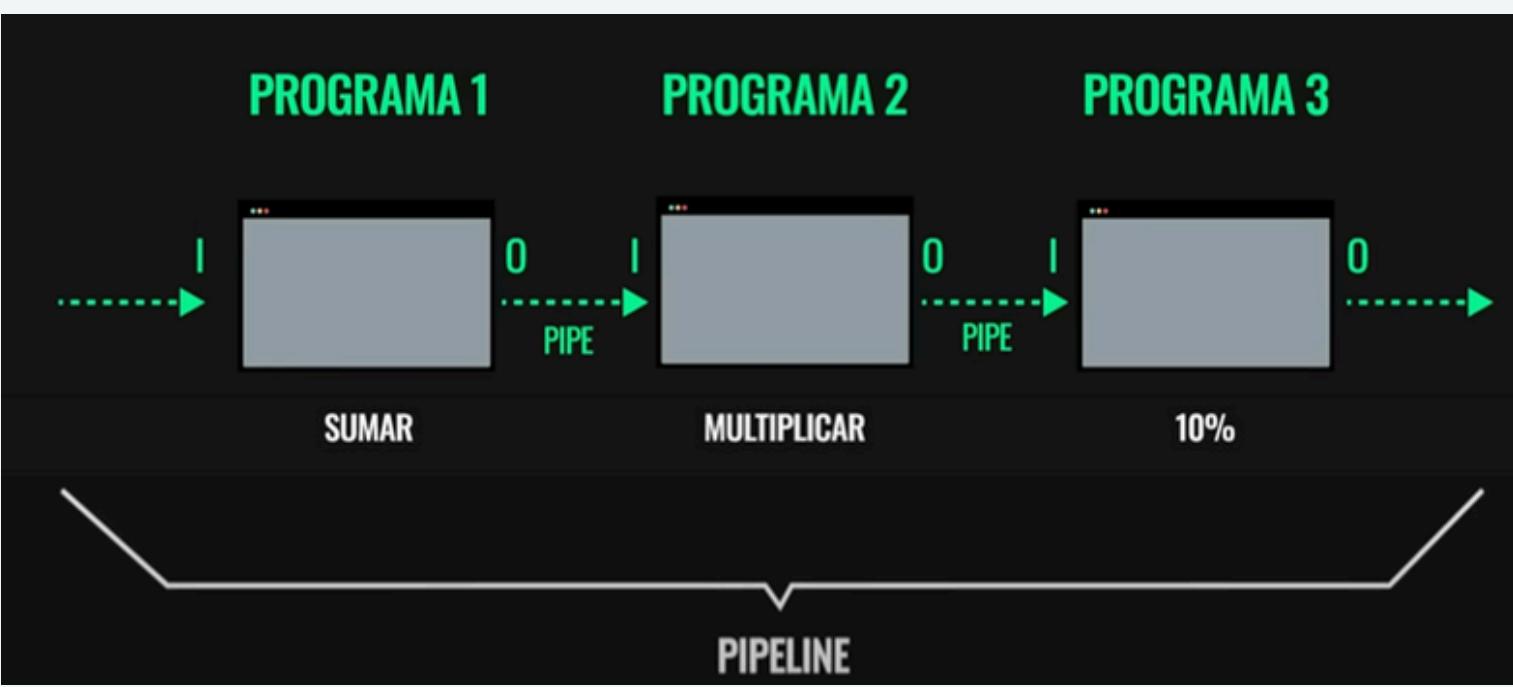


MANUAL GUIA PARA USAR UNIX

# PIPES

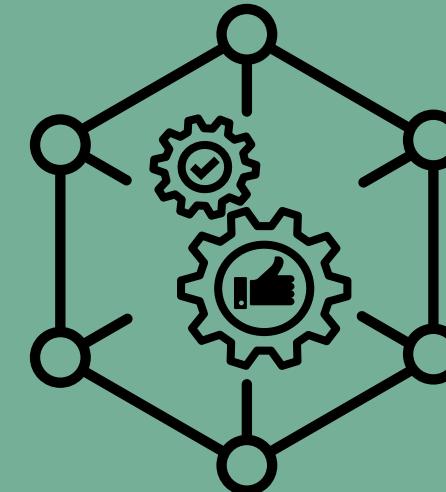
Forma de conectar la salida de un comando con la entrada de otro.

# PIPELINE



```
rubio@MacBook-Air-de-Jose Expo % cat > servidor.log <<EOF  
2025-04-14 08:00 INFO conexión establecida con 192.168.1.1  
2025-04-14 08:05 ERROR conexión fallida con 192.168.1.2  
2025-04-14 09:10 INFO conexión establecida con 192.168.1.3  
2025-04-14 10:45 ERROR conexión fallida con 192.168.1.4  
2025-04-14 11:00 ERROR conexión fallida con 192.168.1.5  
2025-04-13 17:30 ERROR conexión fallida con 192.168.1.6  
2025-04-14 13:00 INFO conexión establecida con 192.168.1.7  
2025-04-14 13:15 ERROR conexión fallida con 192.168.1.8  
2024-04-14 15:30 ERROR conexión fallida con 192.168.1.9  
EOF  
rubio@MacBook-Air-de-Jose Expo % cat servidor.log | grep "ERR  
        4  
rubio@MacBook-Air-de-Jose Expo %
```

```
rubio@MacBook-Air-de-Jose Expo % cat servidor.log | grep "ERROR conexión" | grep "2025-04-14" | wc -l  
4  
rubio@MacBook-Air-de-Jose Expo %
```



# COMANDO GREP

Global / Regular expression / Print

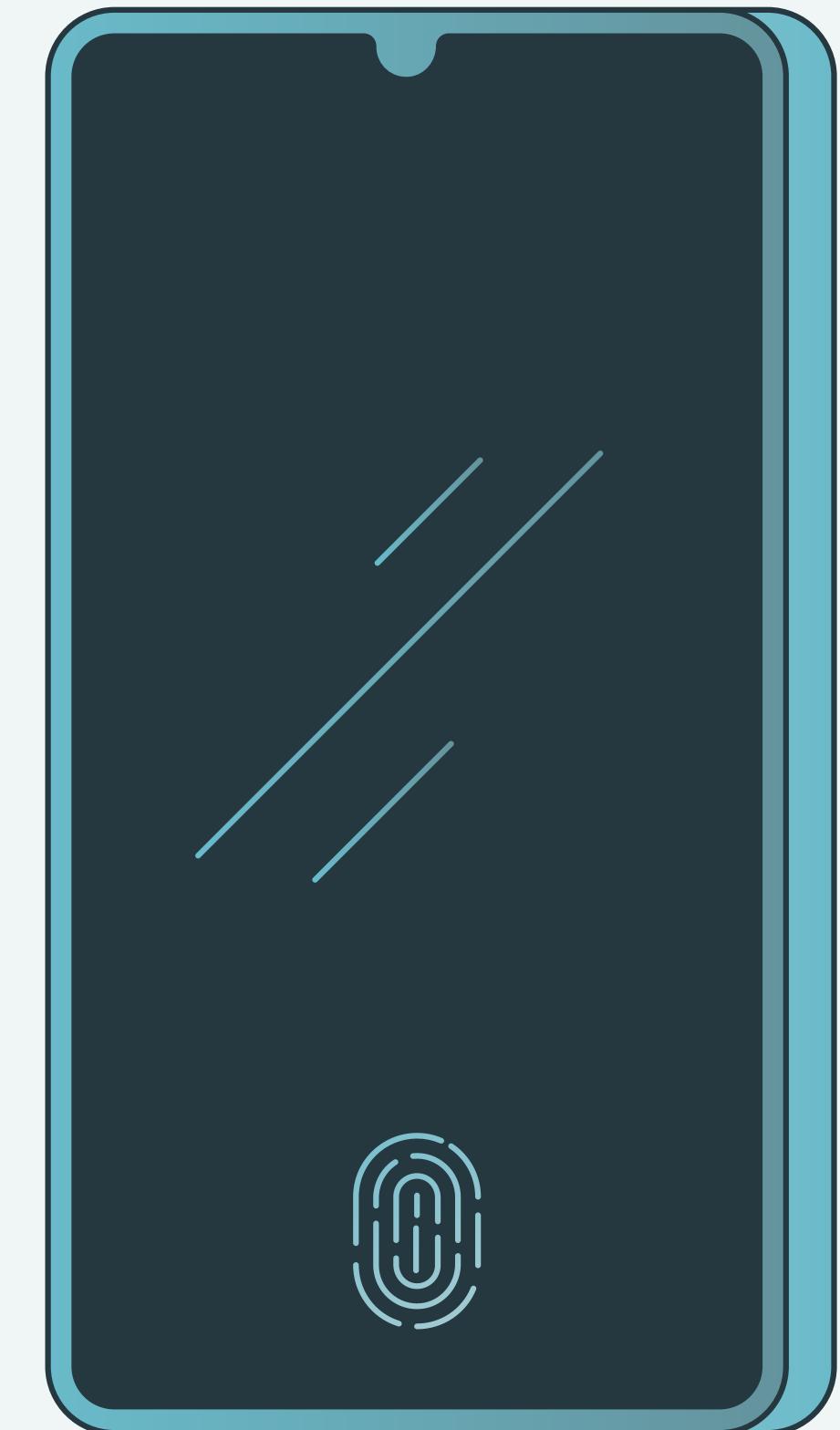
```
rubio@MacBook-Air-de-Jose Texto.txt % touch Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % echo "Aqui se encuentra dos veces la palabra error (error)" > Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % cat Error.txt
Aqui se encuentra dos veces la palabra error (error)
rubio@MacBook-Air-de-Jose Texto.txt % grep 'error' Error.txt
Aqui se encuentra dos veces la palabra error (error)
rubio@MacBook-Air-de-Jose Texto.txt % echo "Aqui no se encuentra la palabra" >> Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % grep 'error' Error.txt
Aqui se encuentra dos veces la palabra error (error)
rubio@MacBook-Air-de-Jose Texto.txt % grep 'palabra' Error.txt
Aqui se encuentra dos veces la palabra error (error)
Aqui no se encuentra la palabra
rubio@MacBook-Air-de-Jose Texto.txt % echo "Hola" >> Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % grep 'Hola' Error.txt
Hola
rubio@MacBook-Air-de-Jose Texto.txt % grep 'o' Error.txt
Aqui se encuentra dos veces la palabra error (error)
Aqui no se encuentra la palabra
Hola
rubio@MacBook-Air-de-Jose Texto.txt % grep 'H' Error.txt
Hola
rubio@MacBook-Air-de-Jose Texto.txt % grep 'h' Error.txt
rubio@MacBook-Air-de-Jose Texto.txt % grep -i 'h' Error.txt
Hola
rubio@MacBook-Air-de-Jose Texto.txt %
```



# COMANDO SED

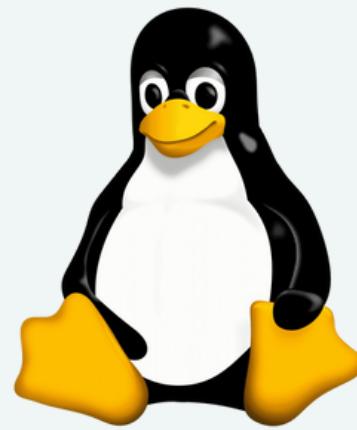
## stream editor

```
rubio@MacBook-Air-de-Jose Texto.txt % echo probando el comando sed  
probando el comando sed  
[rubio@MacBook-Air-de-Jose Texto.txt % sed 's/#/aqui/' Error.txt  
  
Solo remplaza la primer ocurrencia (aqui, # ,#)  
  
Remplaza todas las ocurrencias(%, %, %)  
  
[Las remplaza y lo guarda (cambio, cambio )%  
rubio@MacBook-Air-de-Jose Texto.txt % sed 's/%/aqui/g' Error.txt  
  
Solo remplaza la primer ocurrencia (#, # ,#)  
  
Remplaza todas las ocurrencias(aqui, aqui, aqui)  
  
[Las remplaza y lo guarda (cambio, cambio )%  
rubio@MacBook-Air-de-Jose Texto.txt % sed -i '' 's/cambio/Hola/g' Error.txt  
[rubio@MacBook-Air-de-Jose Texto.txt % cat Error.txt  
  
Solo remplaza la primer ocurrencia (#, # ,#)  
  
Remplaza todas las ocurrencias(%, %, %)  
  
Las remplaza y lo guarda (Hola, Hola )%  
rubio@MacBook-Air-de-Jose Texto.txt %
```



# UNIX Y SU LEGADO

UNIX WAS NOT DESIGNED TO STOP ITS USERS FROM DOING STUPID THINGS, AS THAT WOULD ALSO STOP THEM FROM DOING CLEVER THINGS."



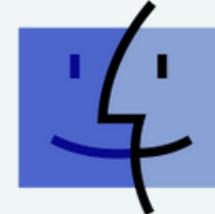
1991

Linux



1993

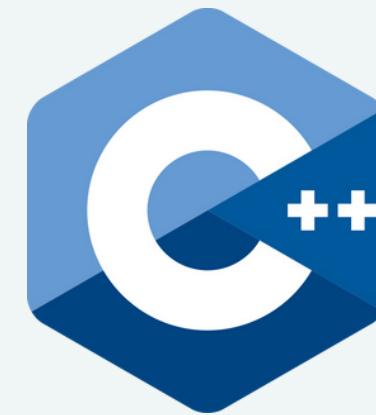
Windows NT



Mac OS

1997

MacOs



1997+

Lenguaje C y C++

# CONCLUSIONES



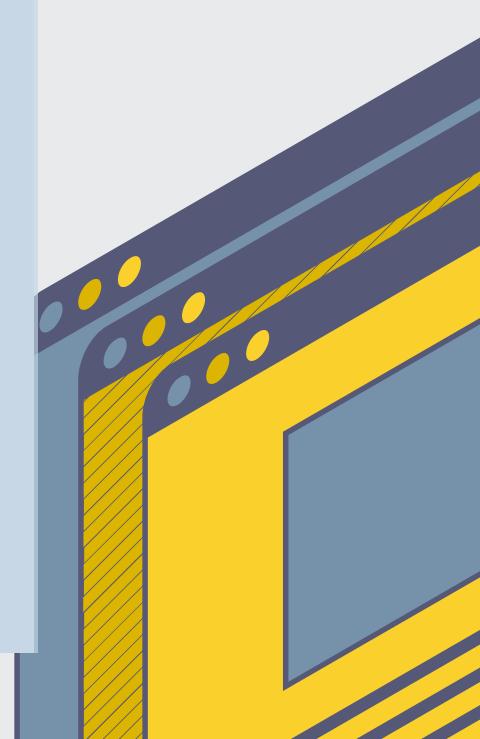
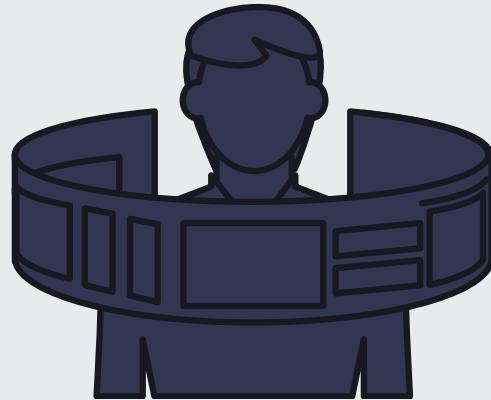
01.

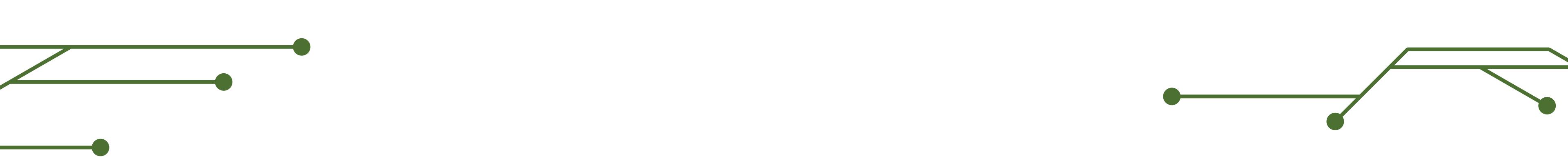
UNIX marcó un antes y un después en la historia de la computación. No solo introdujo avances técnicos fundamentales —como la multitarea, el sistema de archivos jerárquico y el uso de lenguajes portables como C—, sino que definió una filosofía de diseño que ha influido en generaciones enteras de ingenieros, desarrolladores y científicos.



02.

UNIX marcó un antes y un después en la historia de los sistemas operativos gracias a su diseño simple, modular y portátil, que influyó directamente o indirectamente en la creación de sistemas como Linux, macOS y Windows NT. Su filosofía de herramientas pequeñas y combinables, junto con su desarrollo en el lenguaje C, su desarrollo en el lenguaje C permitió que fuera fácilmente adaptado y compartido, lo que lo convirtió en la base de muchos otros sistemas. Más allá de su código, UNIX dejó una huella profunda en la filosofía del software: colaboración, reutilización y control total sobre la máquina. Hoy, su legado sigue vivo en casi todo lo que usamos.





Sistemas Operativos

# MUCHAS GRACIAS

