Bash

101

WHOAMI

@OpenBokeron

OpenBokeron
https://openbokeron.uma.es

HOJA DE RUTA

01

GUI vs CLI

¿Para qué querrías usar la caja negra esa siquiera?

02

Estructura de ficheros

Friendship ended with “C:\” now “/” is my best friend

03

Comandos básicos

Manéjate en la terminal

04

Permisos

Papers please

HOJA DE RUTA

05
Piping
El fontanero ha llegado

06
Scripting en Bash
Que comience el espectáculo

07
Comandos
Comandos to’ wapos

01

GUI vs CLI

¿Quién es CLI y por qué aparece en todos mis apuntes?

01

CLI

Comodidad

Flexible y preciso. Para hacer justo lo que quieres que haga

Integración

Es fácil integrar varios programas entre ellos

Extendido

La CLI se utiliza en escritorios, servidores.... Si tiene un teclado tiene una CLI

Liviano y rápido

Una CLI consume menos recursos que una interfaz gráfica

Automatizable

La CLI es simple de automatizar con el uso de scripts

Funcional

La versión CLI de un programa suele tener más funciones que su versión GUI

02

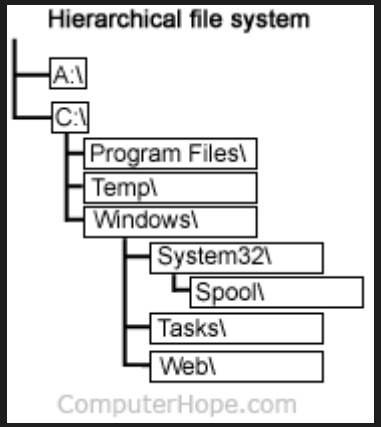
Gestión de archivos

Reject “C:\”, embrace “/”

Antes de empezar

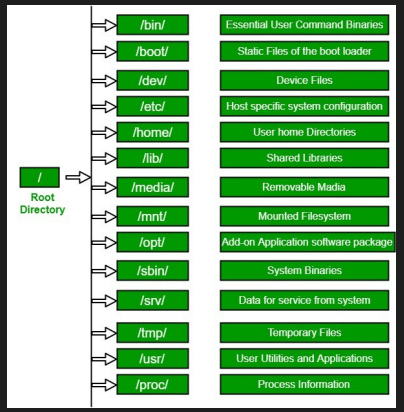
Windows

Sistema jerárquico en el que cada unidad (disco duro, pendrive...) tiene su letra (A–Z) asignada automáticamente.

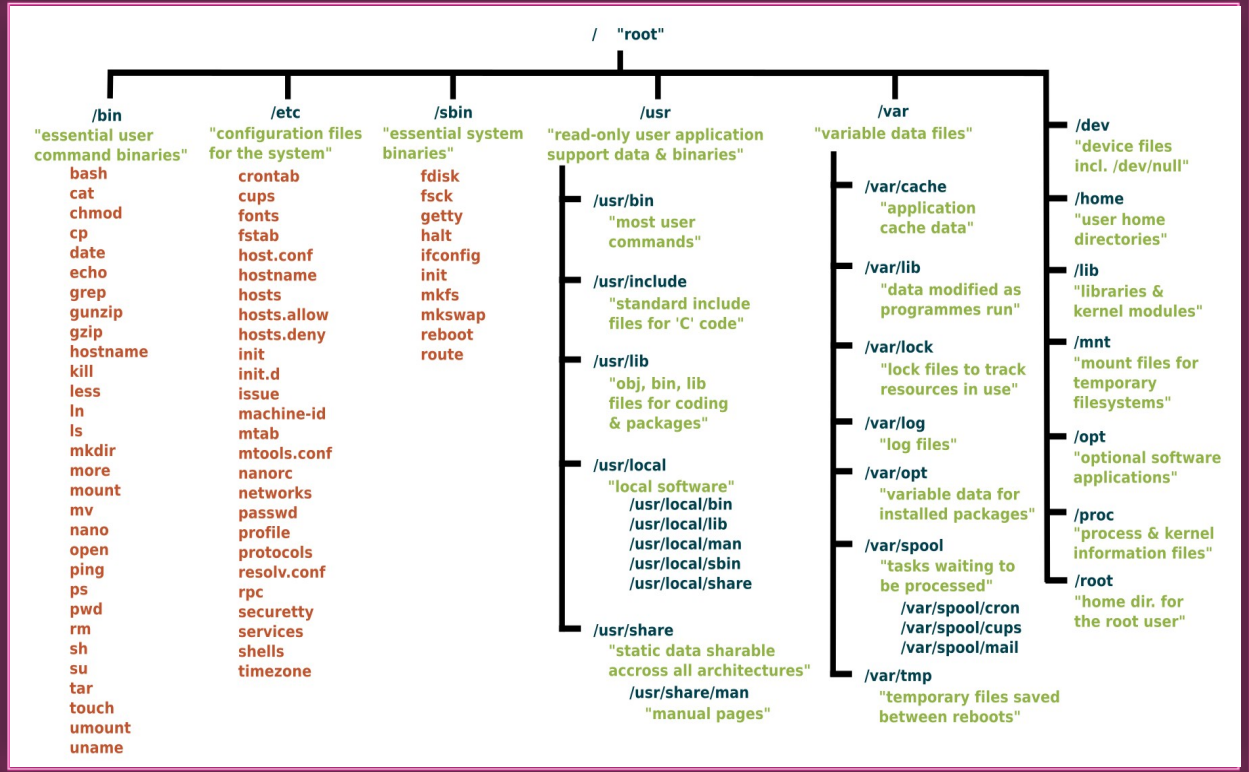


Linux

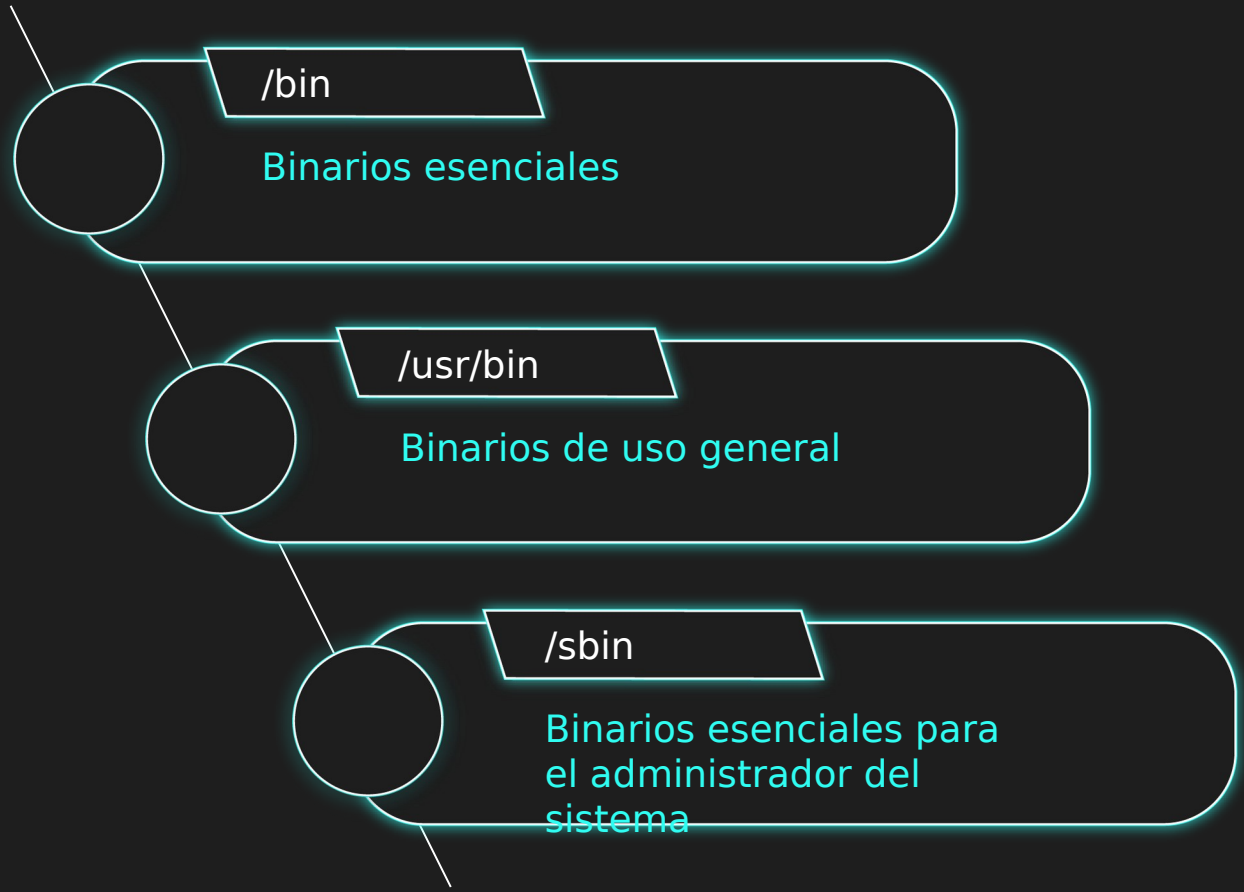
Sistema jerárquico en el que *TODAS* las carpetas y archivos descienden de “/” (raíz del sistema)



Jerarquía Linux



Momento xd de Linux
Parte ∞



¿Son lo mismo entonces?

¯_(ツ)_/¯

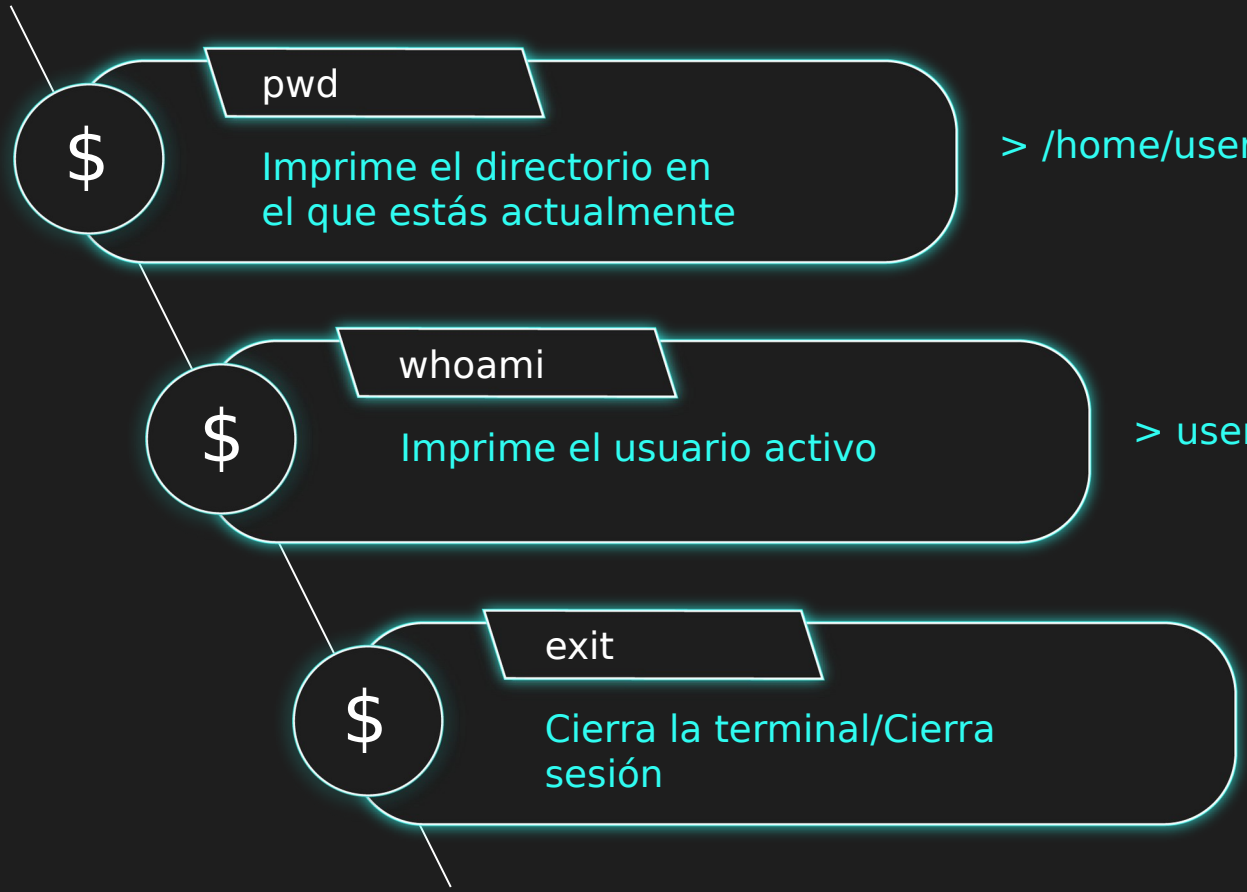
Casi

03

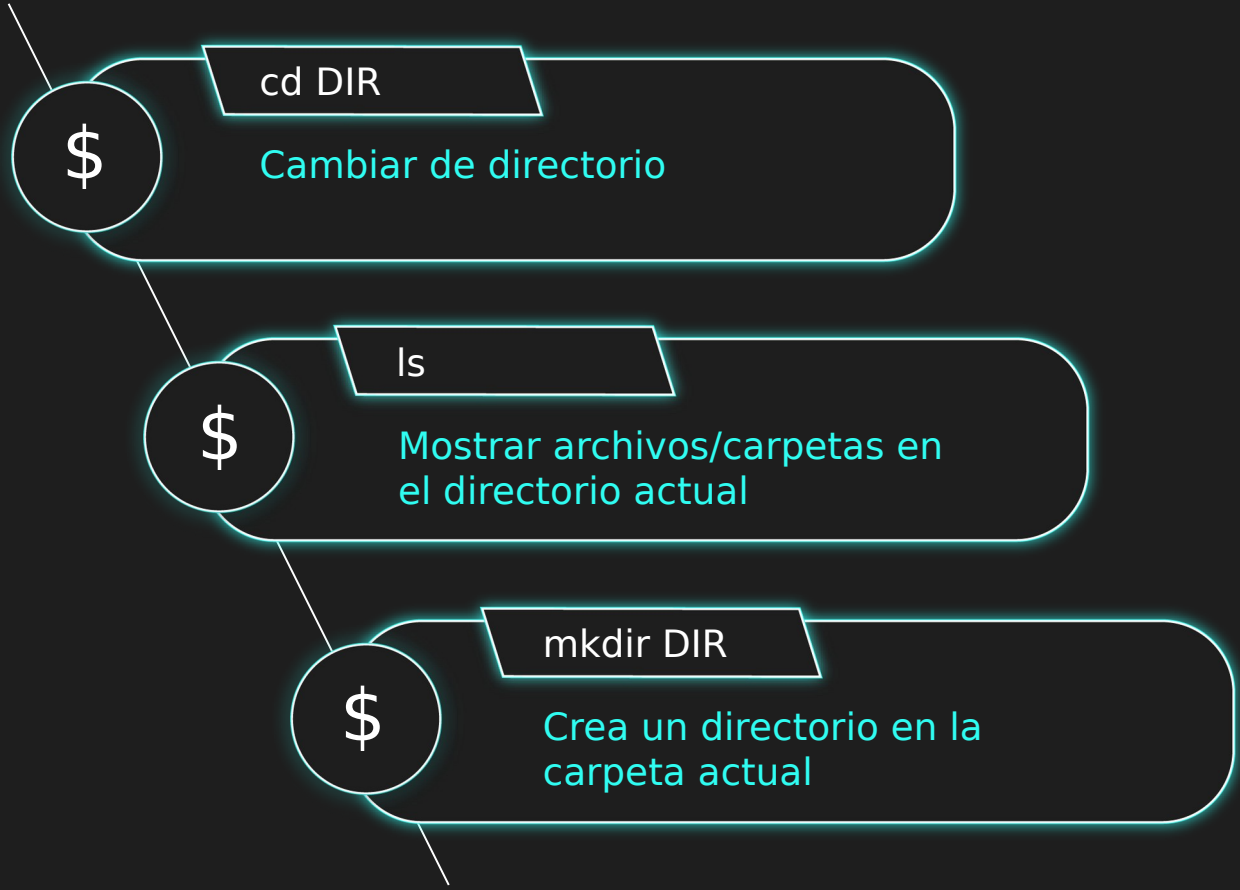
03

Comandos del día a día

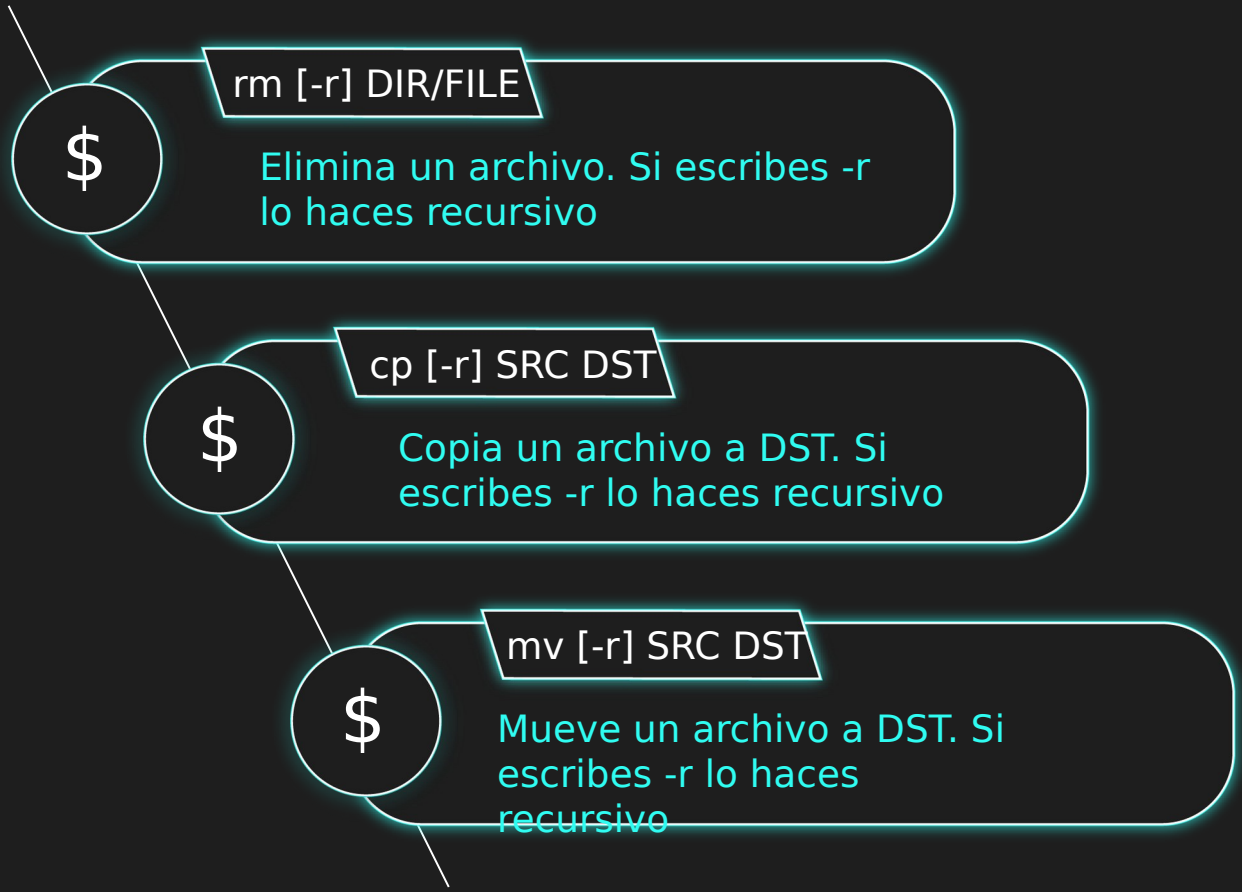
Comandos de Linux
Ubicarte un poco



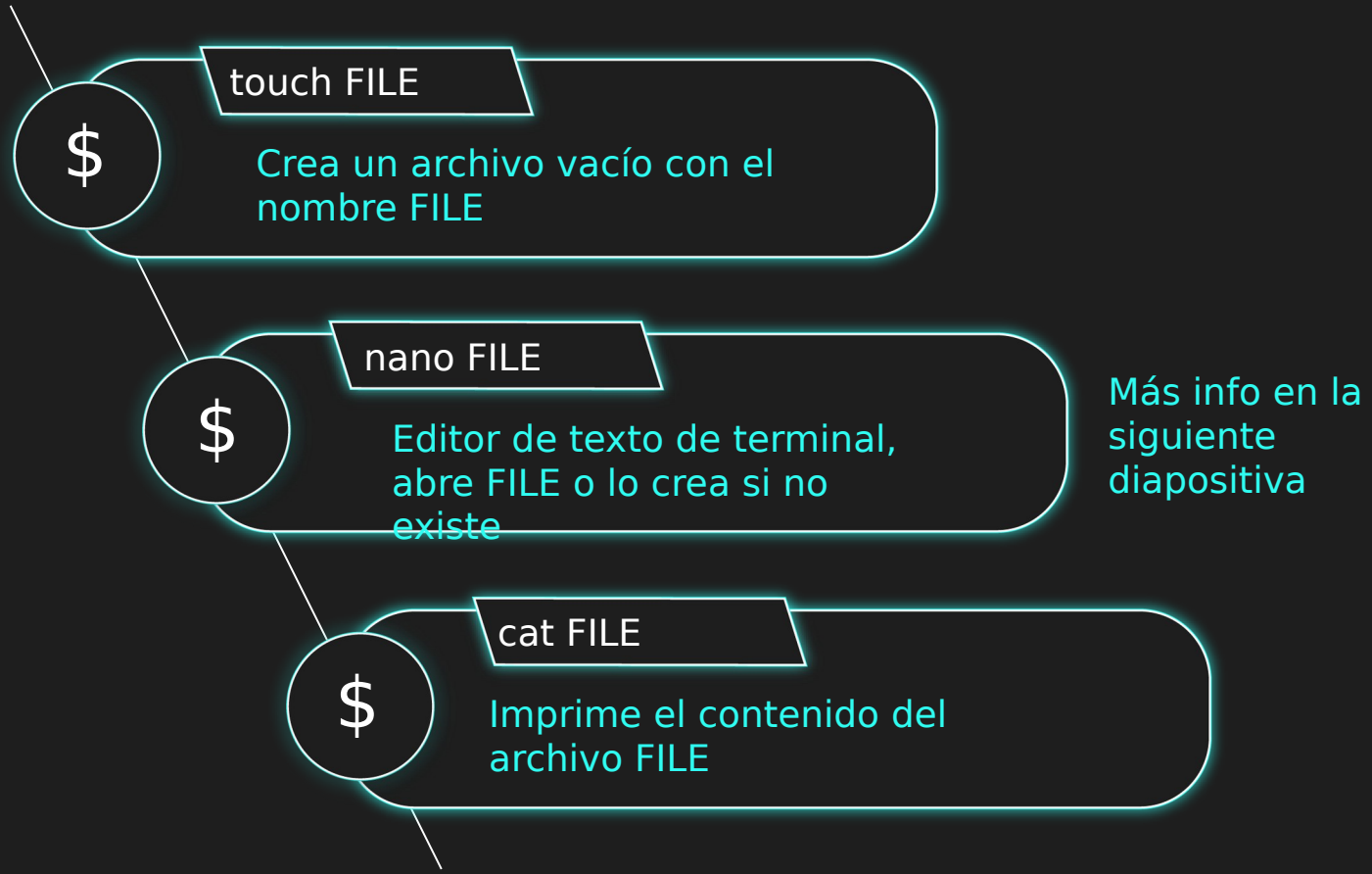
Comandos de Linux
Gestión de ficheros/carpetas



Comandos de Linux
Gestión de ficheros/carpetas



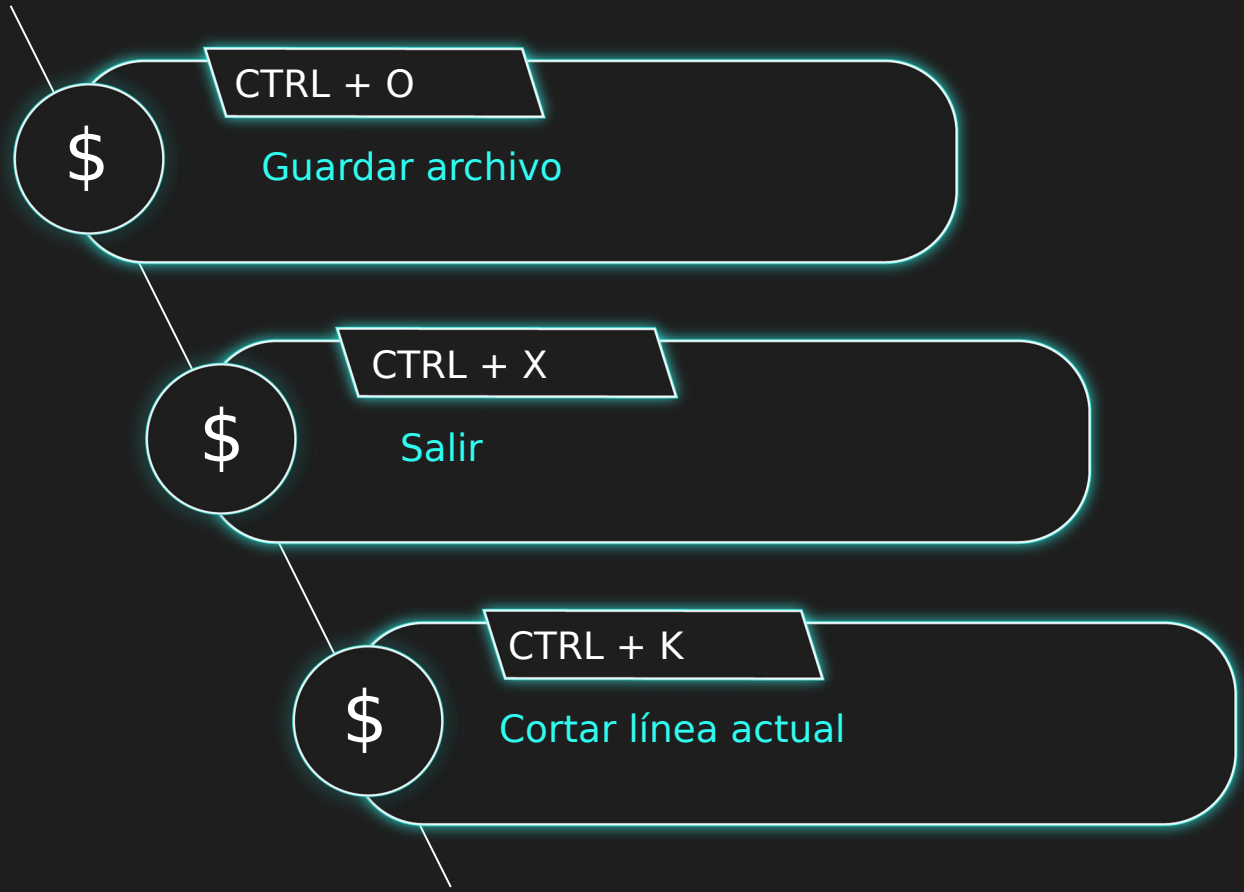
Comandos de Linux
Lectura/Escritura de archivos



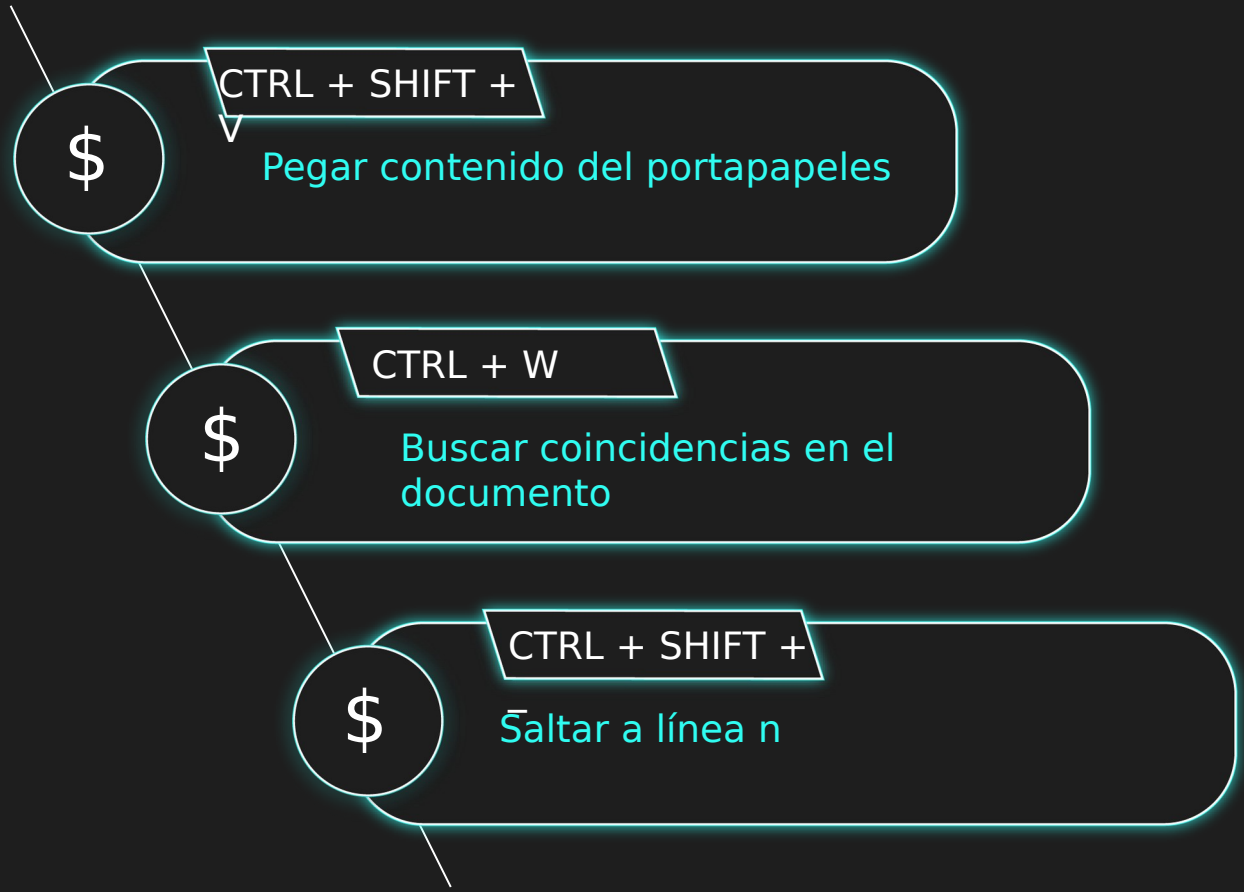
er’ nano


GNU nano 7.2
tonto el que lo lea.txt
[Nuevo fichero]
^G Ayuda
^O Guardar
^W Buscar
^K Cortar
^T Ejecutar
^C Ubicación
M-U Deshacer
M-A Poner marca
M-] A llave
M-Q Anterior
^B Atrás
^← Palabr ant
^X Salir
^R Leer fich.
^\ Reemplazar
^U Pegar
^J Justificar
^/ Ir a línea
M-E Rehacer
M-6 Copiar
^Q Buscar atrás
M-W Siguiente
^F Adelante
^→ Palabr sig

Comandos de Linux
Lectura/Escritura de archivos



Comandos de Linux
Lectura/Escritura de archivos



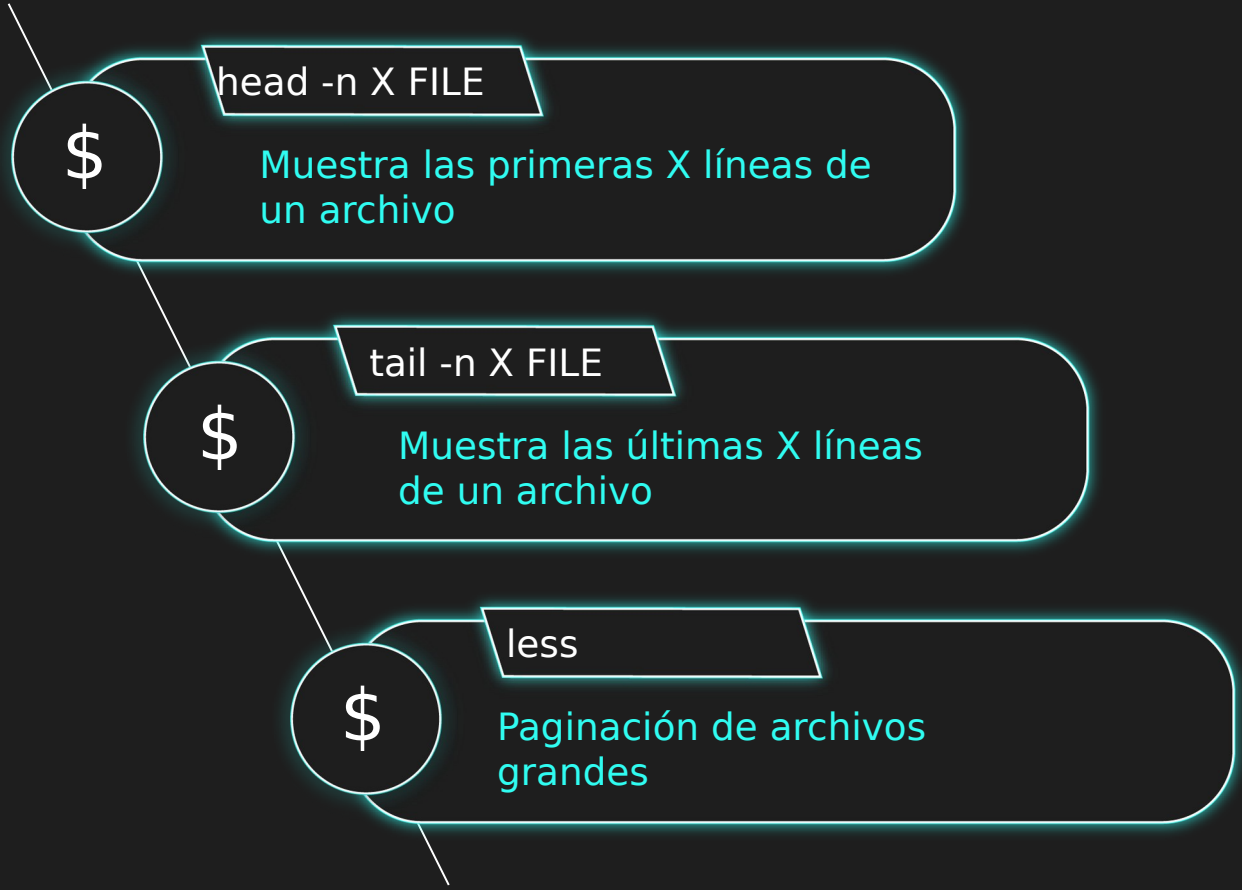
\ Espacios \

‘(mucho más problemático de lo que piensas)’

```
┌─ /mnt/Datos/Sources/h4cking ──────────────────────── pablo PCPablo:pts/0 ─┐
└─(14:37:14)──> ls                                          ──(jue,abr20)─┘
'Archivo con espacios.txt'
┌─ /mnt/Datos/Sources/h4cking ──────────────────────── pablo PCPablo:pts/0 ─┐
└─(14:37:15)──> cat Archivo con espacios.txt                ──(jue,abr20)─┘
cat: Archivo: No existe el fichero o el directorio
cat: con: No existe el fichero o el directorio
cat: espacios.txt: No existe el fichero o el directorio
┌─ /mnt/Datos/Sources/h4cking ──────────────────────── pablo PCPablo:pts/0 ─┐
└─(14:37:16)──> ▉                                         1 ↵ ──(jue,abr20)─┘
```


```
┌─ /mnt/Datos/Sources/h4cking ──────────────────────── pablo PCPablo:pts/0 ─┐
└─(14:41:44)──> cat Archivo\ con\ espacios.txt               ──(jue,abr20)─┘
Por favor evitad crear archivos/carpetas con espacios uwu
Por vuestra propia comodidad
┌─ /mnt/Datos/Sources/h4cking ──────────────────────── pablo PCPablo:pts/0 ─┐
└─(14:41:46)──> cat 'Archivo con espacios.txt'              ──(jue,abr20)─┘
Por favor evitad crear archivos/carpetas con espacios uwu
Por vuestra propia comodidad
┌─ /mnt/Datos/Sources/h4cking ──────────────────────── pablo PCPablo:pts/0 ─┐
└─(14:41:47)──> ▉                                           ──(jue,abr20)─┘
```

Comandos
Lectura/Escritura de archivos



man ls


LS(1) User Commands LS(1)
NAME
ls - list directory contents
SYNOPSIS
ls [OPTION]... [FILE]...
DESCRIPTION
List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
Mandatory arguments to long options are mandatory for short options too.
-a, --all
do not ignore entries starting with .
-A, --almost-all
do not list implied . and ..
--author
with -l, print the author of each file
-b, --escape
print C-style escapes for nongraphic characters
--block-size=SIZE
with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below
-B, --ignore-backups
do not list implied entries ending with ~
-c with -lt: sort by, and show, ctime (time of last modification of file status information); with -l: show ctime and sort by name; otherwise: sort by ctime, newest first
-C list entries by columns
--color[=WHEN]
colorize the output; WHEN can be 'always' (default if omitted), 'auto', or 'never'; more info below
-d, --directory
list directories themselves, not their contents
-D, --dired
generate output designed for Emacs' dired mode
-f do not sort, enable -aU, disable -ls --color
-F, --classify
append indicator (one of */=>@|) to entries
--file-type
likewise, except do not append '*'
--format=WORD
across -x, commas -m, horizontal -x, long -l, single-column -1, verbose -l, vertical -C
Manual page ls(1) line 1 (press h for help or q to quit)

04

Permisos

Papers please

04

Permisos

DUEÑO

Afecta al usuario propietario del archivo

GRUPO

Afecta al grupo asignado al archivo (no modifica al dueño)

TODO

Afecta a todos los otros usuarios que no son ni dueños del fichero ni pertenecen al grupo asignado

Permisos

LECTURA (4 o r)

Te permite leer el archivo

ESCRITURA (2 o w)

Te permite modificar el archivo

EJECUCIÓN (1 o x)

Te permite ejecutar el archivo (solo para binarios y scripts)

```
[root@debian ~]# chown www-data:redis test.txt
```

chown: Comando para cambiar autoría de archivo
www-data: Nuevo usuario dueño
redis: Nuevo grupo dueño
test.txt: Archivo al que cambiarle la autoría

[user@debian ~]$ chmod XYZ test.txt

chmod: Comando para cambiar permisos de archivo
XYZ: Números del 0 al 7 para representar permisos

- X: Usuario Dueño
- Y: Grupo
- Z: Todos los demás

4: Lectura; 2: Escritura; 1: Ejecución

```
[user@debian ~]$ chmod 755 test.txt
```

Este archivo se quedaría:

- Usuario: Lectura, escritura, ejecución (4 + 2 + 1)
- Grupo: Lectura y ejecución (4 + 1)
- Todos: Lectura y ejecución (4 + 1)

```
[user@debian ~]$ chmod 644 test.txt
```

Este archivo se quedaría:

- Usuario: Lectura y escritura (4 + 2)
- Grupo: Lectura (4)
- Todos: Lectura (4)

```
[user@debian ~]$ chmod 600 test.txt
```

Este archivo se quedaría:

- Usuario: Lectura y escritura (4 + 2)
- Grupo: Nada (0)
- Todos: Nada (0)

```
[user@debian ~]$ chmod +x test.txt
```

Este archivo se quedaría como estuviera anteriormente + permiso de ejecución para el dueño

[user@debian ~]$ chmod 777 test.txt

Este archivo se quedaría:

- Usuario: Lectura, escritura y ejecución (4 + 2 + 1)
- Grupo: Lectura, escritura y ejecución (4 + 2 + 1)
- Todos: Lectura, escritura y ejecución (4 + 2 + 1)

Por favor no uses esto. Y no, no hay excusa válida :3

Dios simulator

CONTROL ABSOLUTO

Puedes modificar cualquier archivo en cualquier carpeta

PELIGROSO

A Linux no le importa tus sentimientos, un comando mal escrito puede acabar en una catástrofe

EVITA CUANDO PUEDAS

Ejecuta un comando como root ***SÓLO SI ES NECESARIO***

sudo

Permisos root temporales

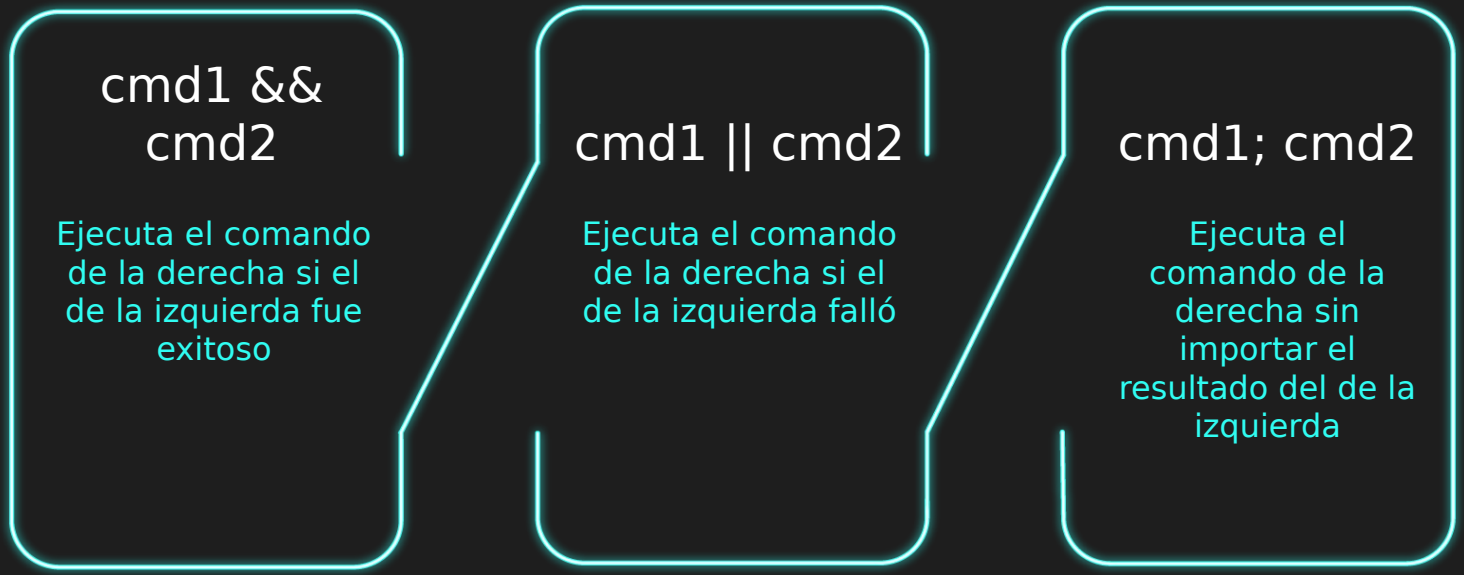
[user@debian ~]$ nano /etc/shadow
[user@debian ~]$ sudo nano /etc/shadow

Contraseña: *******

05
Piping / Redirecciones

Vete a tomar por null

Encadenar comandos



Return codes

cat NO_EXISTO.txt

cat: NO_EXISTO.txt: No existe el fichero o el directorio

(Return code 1)

cat VALIDO.txt

Output aquí

(Return code 0)

Redir/Pipes

ls / > root.txt

Redirige todo el output del comando de la izquierda al archivo de la derecha

- Crea si no existe
- Sobreescribe si ya existe

ls / >> root.txt

Redirige todo el output del comando de la izquierda al archivo de la derecha

- Crea si no existe
- Añade al final del archivo si existe

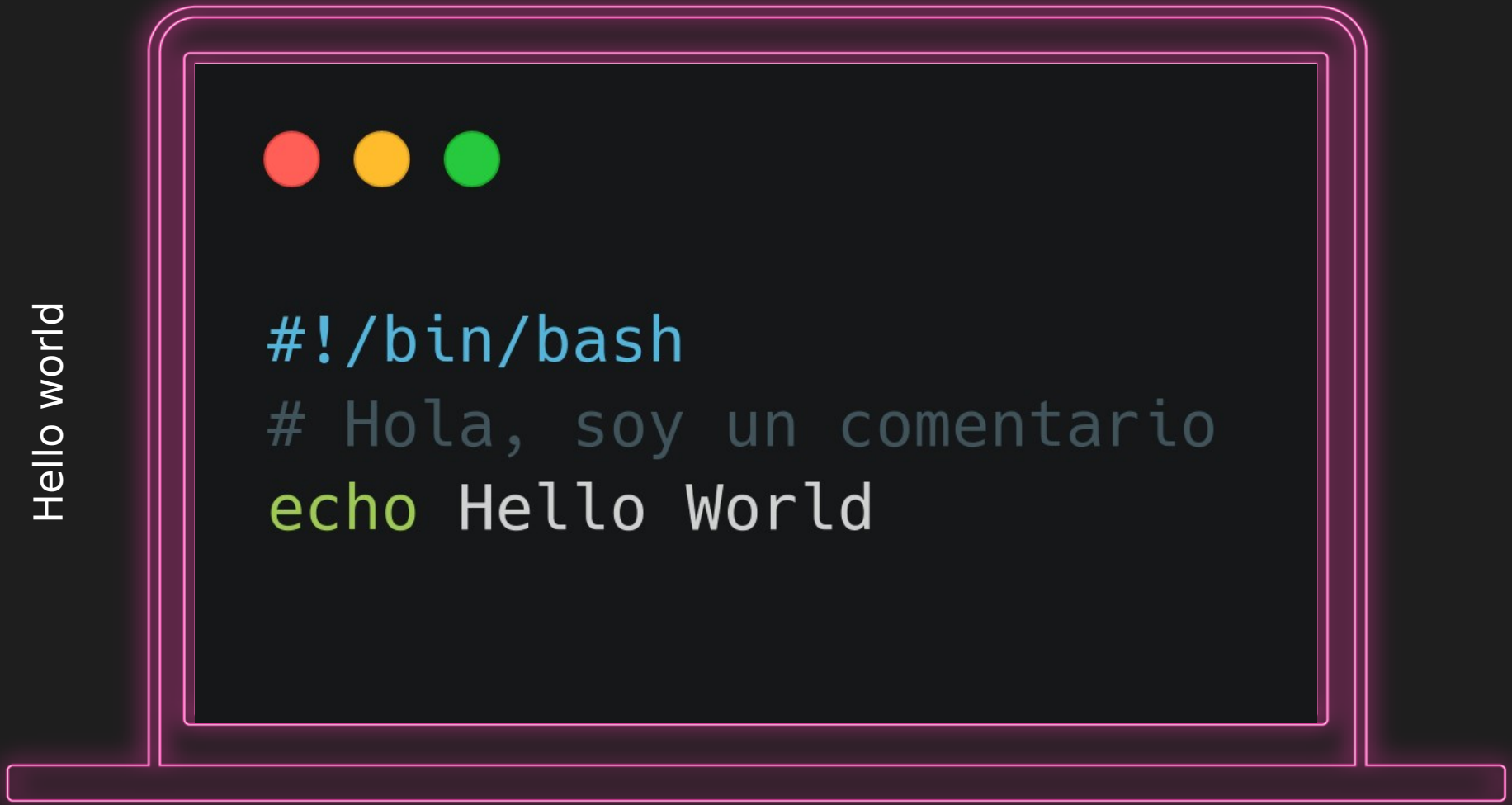
cat test.txt | sort

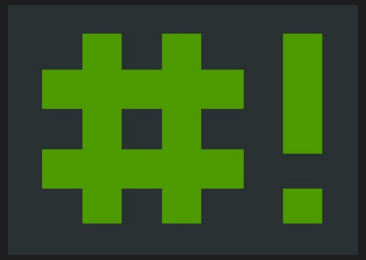
Usa el output del comando de la izquierda como input para el comando de la derecha

06
Scripting en Bash

Que comience el espectáculo

Hello world





Shebang

Primera línea de todo script que se respete

Intérprete

Especifica qué intérprete usar

Estructura

#!
PATH_INTERPRETE
SIEMPRE EN PRIMERA LÍNEA

“““Opcional”””

bash test.sh
~= ./test.sh

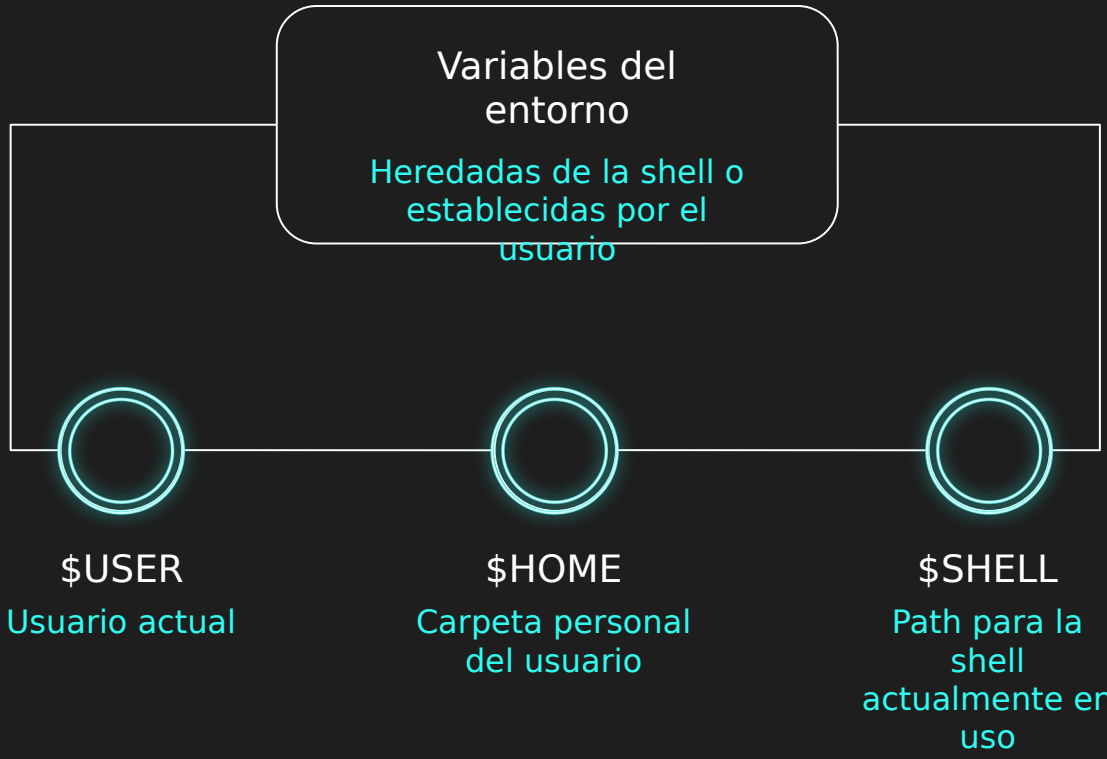
$ chmod +x test.sh

$ ./test.sh

Hello world!

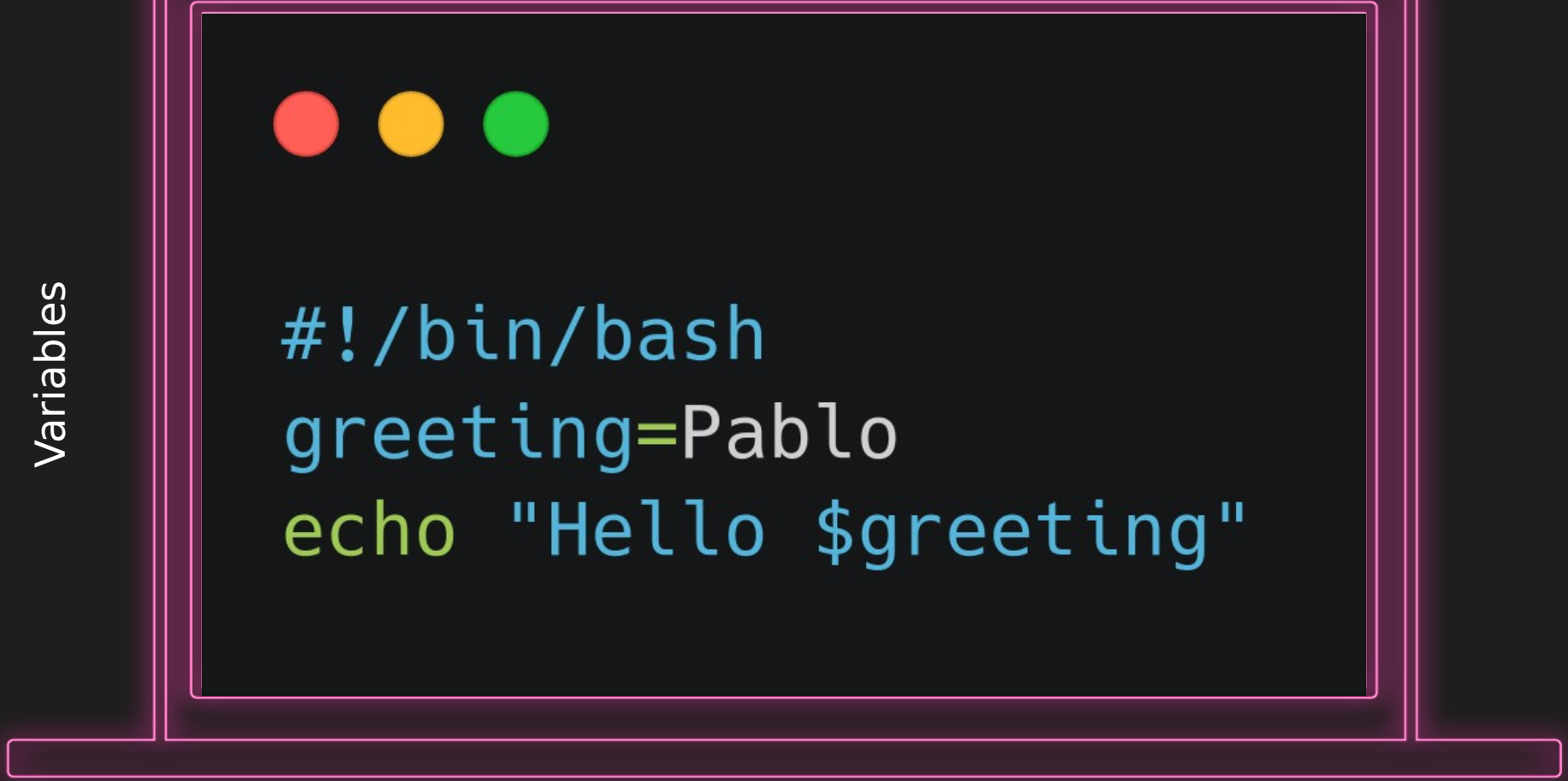
Profit

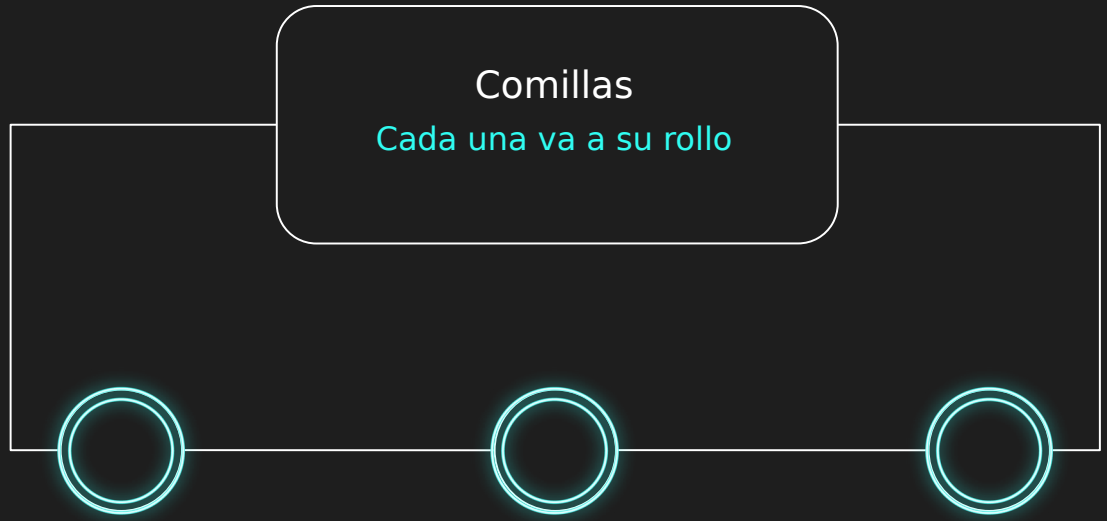

Variables
Como todo en Bash, son especialitas
Definición
VAR=VALUE
Scope
Siempre globales excepto opcionalmente en funciones con la keyword “local”
CUIDADO
VAR=Hola buenas
No es válido, asignaría VAR=Hola y después tendrías intentarías ejecutar un comando llamado “buenas”


Variables del entorno
Heredadas de la shell o establecidas por el usuario
$USER
Usuario actual
$HOME
Carpeta personal del usuario
$SHELL
Path para la shell actualmente en uso


Variables reservadas
Porfa no modifiquéis estos valores UwU
$n
Del $1 al $n
Contiene todos los argumentos enviados
$0: Nombre script
$#
Número de argumentos recibidos
$?
Return code del comando anterior

Variables

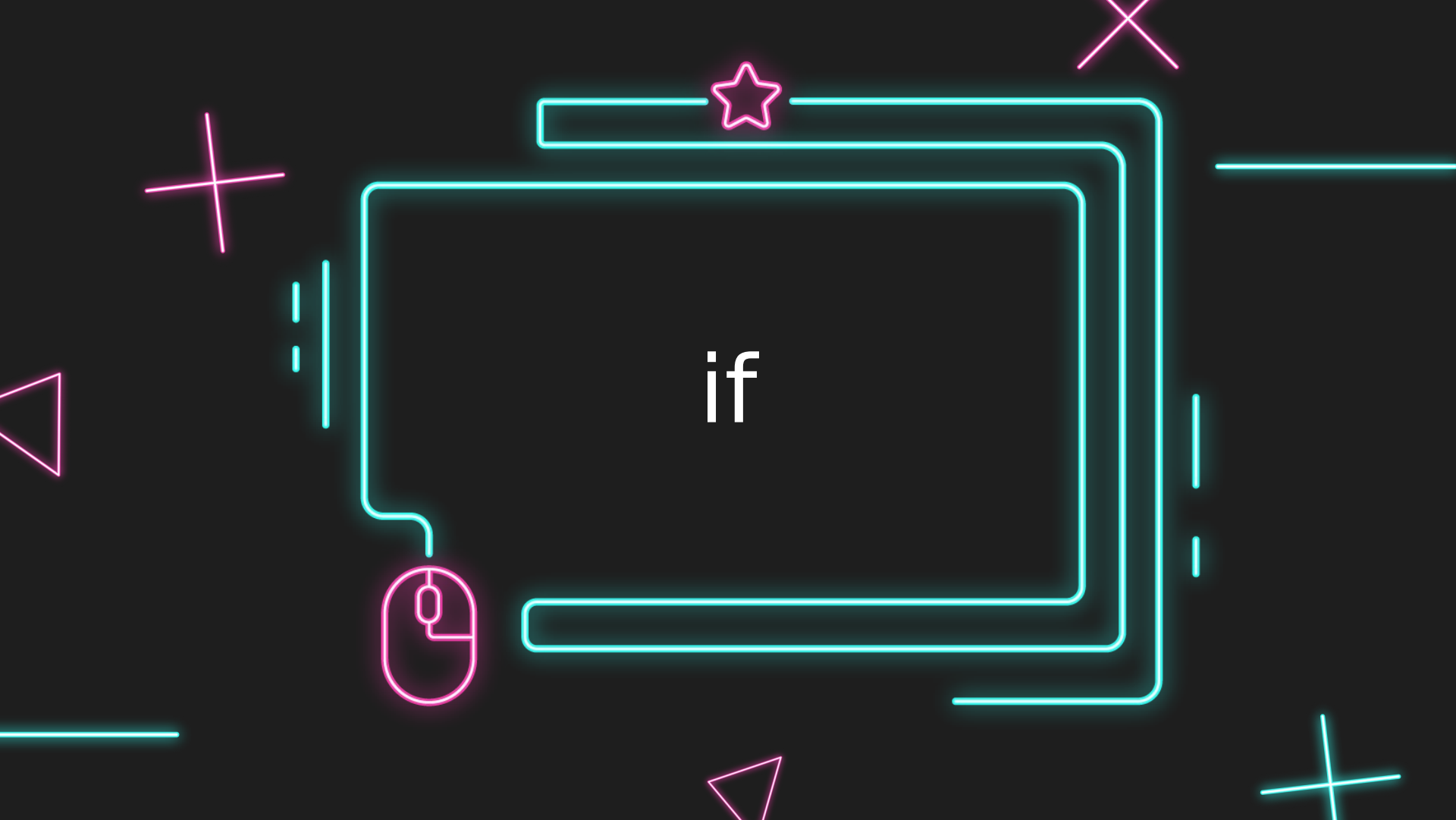



Comillas
Cada una va a su rollo
‘
“
$() o `
Interpreta todo literal, no reemplaza ninguna variable
Interpreta las variables ($NOMBRE) y concatena
Combinado con “”. Usa el output de un comando y concatena:
echo "Estoy en $(pwd)"
echo "Estoy `pwd`"

Variables


#!/bin/bash
greeting=Pablo
echo 'Hello $greeting'
echo "Hello $greeting"
echo "Hello $(whoami)"

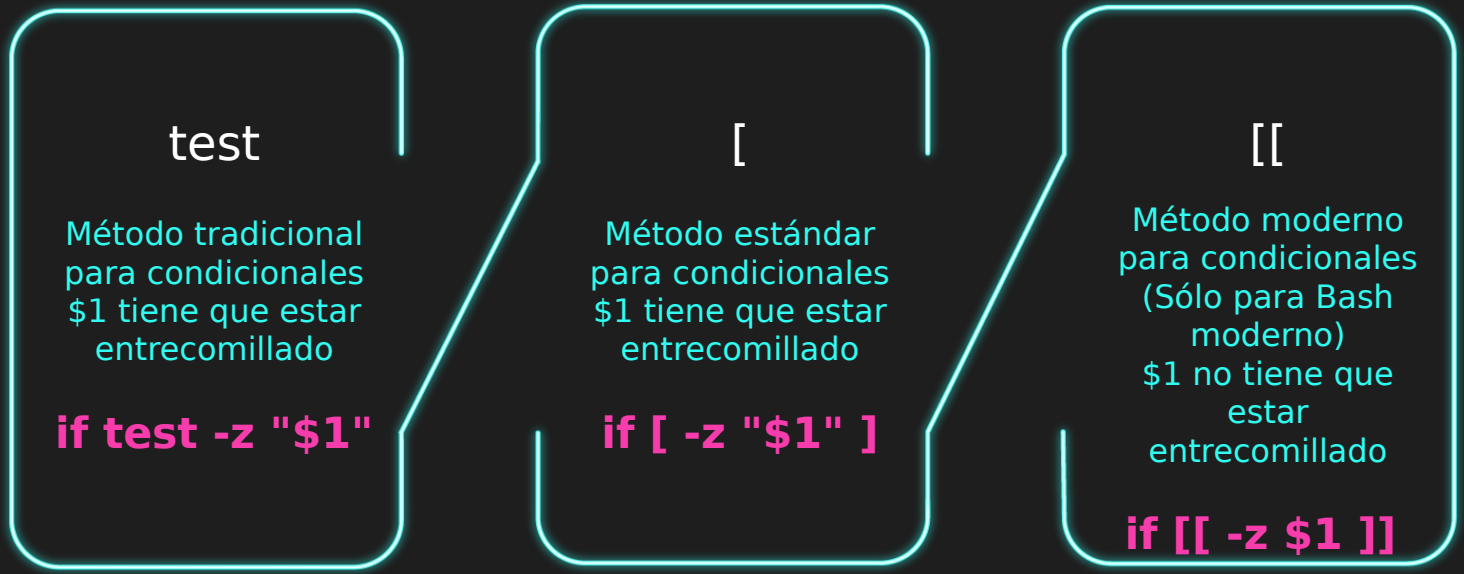

Control de
flujo / bucles


if

if

```
#!/bin/bash
# ¡Muy importantes seguir los espacios en el if!
if [[ -z $1 ]]; then
    echo "No hay parámetro"
    exit 1 # salir de script, 1 = error
fi
greeting=$1
echo "Hello $greeting"
```

if


test
Método tradicional para condicionales $1 tiene que estar entrecomillado
if test -z "$1"
[
Método estándar para condicionales $1 tiene que estar entrecomillado
if [-z "$1"]
[[
Método moderno para condicionales (Sólo para Bash moderno) $1 no tiene que estar entrecomillado
if [[-z $1]]

If Flags

-z y -n

Comprobar si un string está vacío y si no está vació

A == B y A != B

Comprobar si A es igual o no a B (Strings)

-lt y -le

Menor que y menor o igual que

-gt y -ge

Mayor que y mayor o igual que

&& y ||

Comprobar si x e y/o son igual a 0

A -eq B y A –ne B

Comprobar si A es igual o no a B (Números)

elif y else

```
#!/bin/bash
if [[ -z $1 ]]; then
    echo "No hay parámetro"
    exit 1
elif [[ $1 == Pepe ]]; then
    echo "Me caes mal"
    exit 1
else
    echo "Parámetro válido"
fi
greeting=$1
echo "Hello $greeting"
```


case
~~switch~~

HERE YOU CAN USE THREE COLUMNS

Parecidos

Equivalente al “switch” de la mayoría de lenguajes

Menos Python jajajaja ya me jodería

“if” on steroids

Permite agrupar muchos casos en un sólo bloque cómodo de leer

break

A diferencia de la mayoría de lenguajes, **no se usa “break” para salir**, sino “;;”

case

```bash
#!/bin/bash
case $1 in
    Pablo)
        echo "Mejor nombre de todos"
        ;;
    Pepe)
        echo "Fuck Pepe"
        exit 1
        ;;
    # Fíjate que ahora para los strings vacíos
    # no usamos -z, sino ""
    "")
        echo "No hay parámetro"
        exit 1
        ;;
    *)
        echo "Parámetro válido"
        ;;
esac

greeting=$1
echo "Hello $greeting"
```


while

while

```bash
#!/bin/bash
i=0
while [[ $i -lt 15 ]]
do
    echo "Soy el ciclo número $i"
    ((i++))
done

echo "He terminado el ciclo"
```


for

for

```bash
#!/bin/bash
for i in {0..14}
do
    echo "Soy el ciclo número $i"
done
echo "He terminado el primer for"

# o también
for ((i=0;i<15;i++))
do
    echo "Soy el ciclo número $i"
done
echo "He terminado el segundo for"
```

For, word boundary

```
#!/bin/bash
letters='a b c d e f g h'
# Funciona con:
# " "
# \n
# \t
for letter in $letters; do
  echo $letter
done
```

CONTROL DE BUCLES

07
Comandos to’ wapos

De esos que vas a usar en todos tus scripts

grep

grep Python ejemplo.txt

Encuentra “Python” en el archivo ejemplo.txt

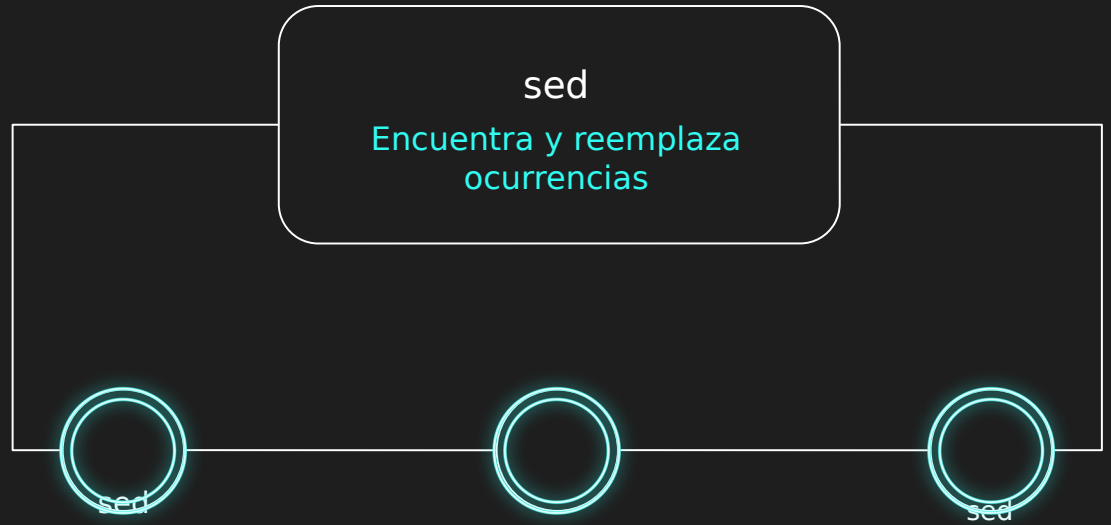
grep -r PATTERN DIR

Busca recursivamente en DIR archivos que contengan PATTERN

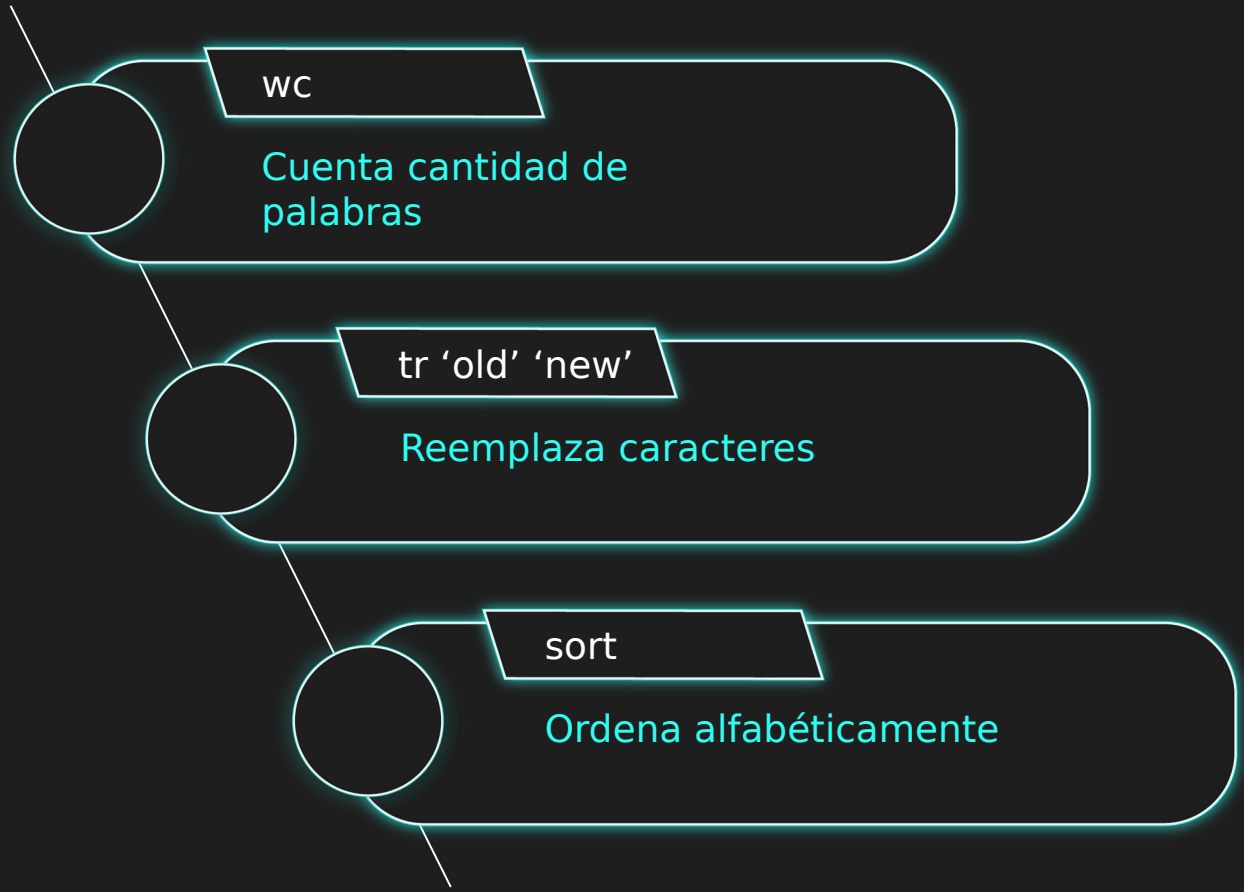
grep -in python ejemplo.txt

Encuentra “python” case-insensitive en el archivo ejemplo.txt

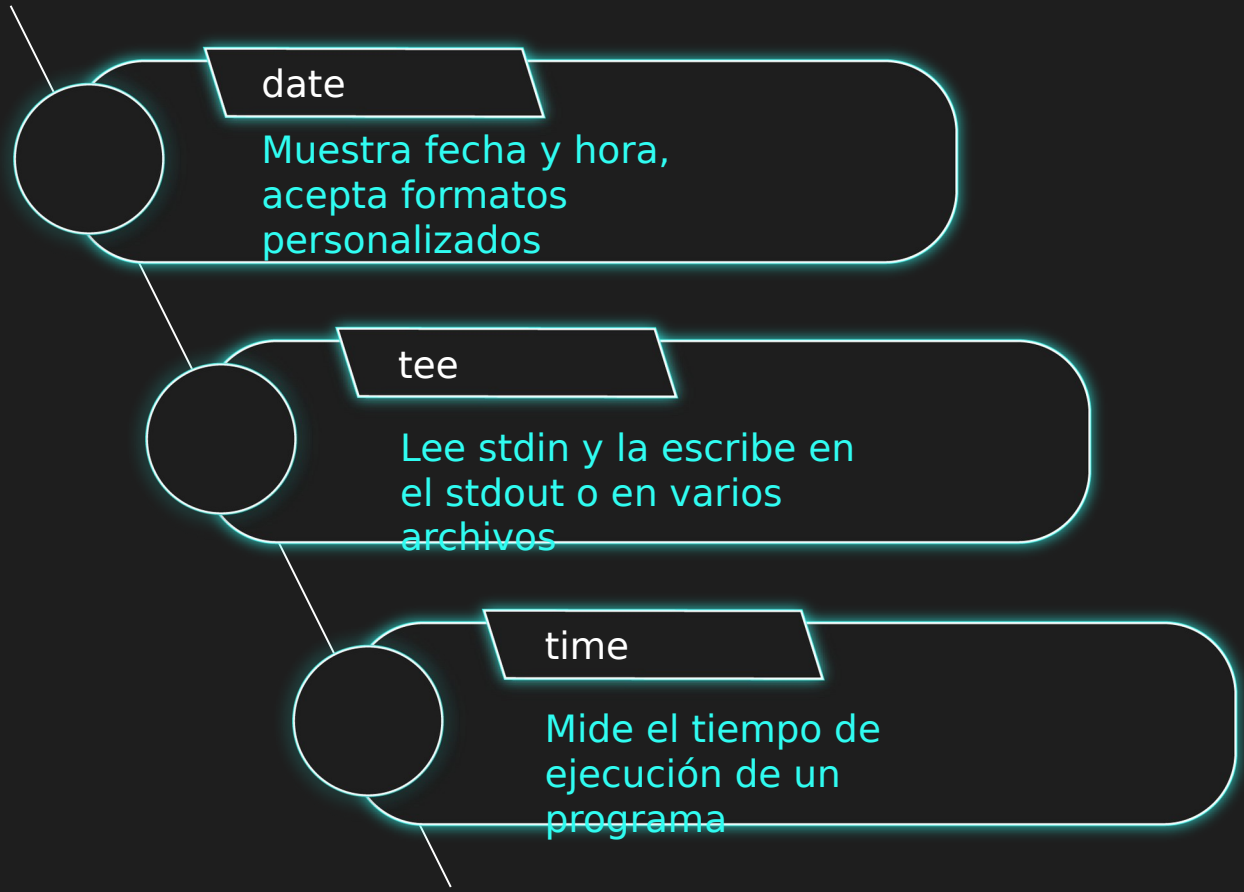
Comandos

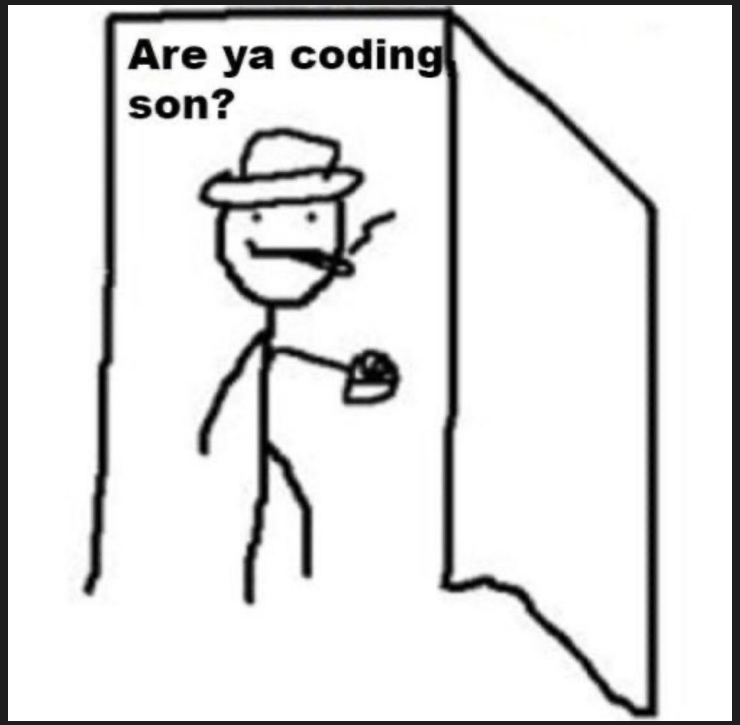

sed
Encuentra y reemplaza ocurrencias
sed 's/Debian/ArchLinux/' ejemplo.txt
Reemplaza la primera ocurrencia de “Debian” por “ArchLinux
sed 's/Debian/ArchLinux/g' ejemplo.txt
Reemplaza todas las ocurrencias de “Debian” por “ArchLinux
sed 's/Debian/ArchLinux/3g' ejemplo.txt
Reemplaza desde la tercera ocurrencia de “Debian” por “ArchLinux

Comandos


wc
Cuenta cantidad de palabras
tr ‘old’ ‘new’
Reemplaza caracteres
sort
Ordena alfabéticamente

Comandos


date
Muestra fecha y hora, acepta formatos personalizados
tee
Lee stdin y la escribe en el stdout o en varios archivos
time
Mide el tiempo de ejecución de un programa



Ahora lo ves todo en scripts

¡Gracias por venir! <3
Ahora pensarás en automatizar cada tarea en tu vida que tarde más de 90 segundos.
De nada

Recursos disponibles en nuestro repo de Gitea

Encuesta de satisfacción