

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

Gestor de paquetes

¿¿¿Pero como se instala Steam en esto???

05

Permisos
Papers, please

07

Scripting en Bash
Que comience el espectáculo

06

Piping
El fontanero ha llegado

08

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

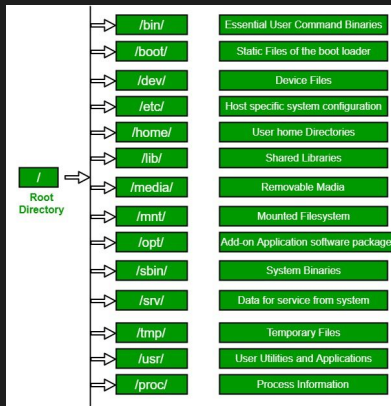
02

Gestión de archivos

Reject "C:\", embrace "/"

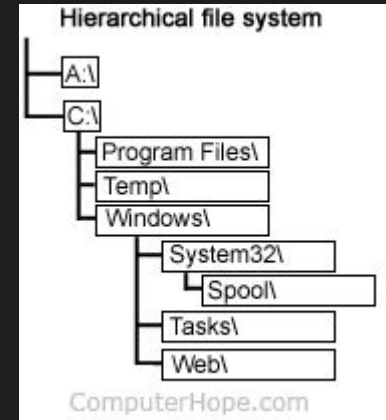


Antes de empezar



Windows

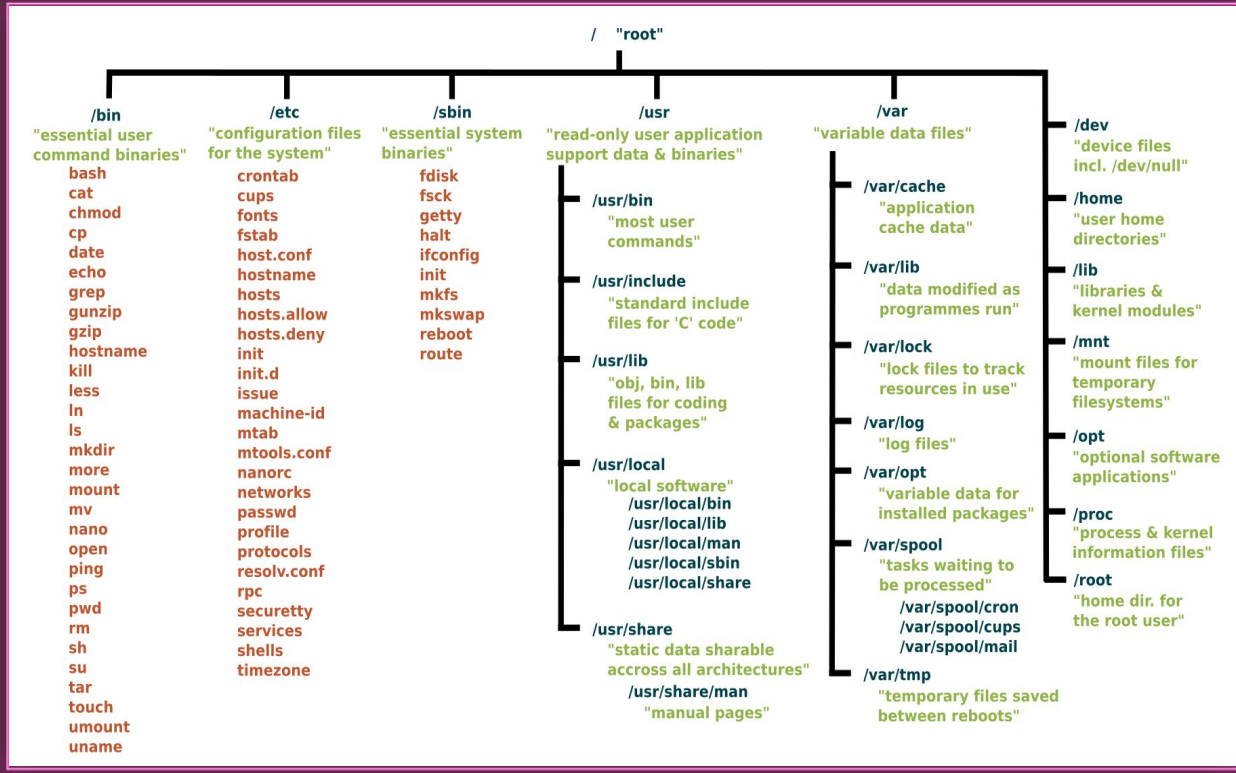
Sistema jerárquico en el que cada unidad (disco duro, pendrive...) tiene su letra (A-Z) asignada automáticamente.



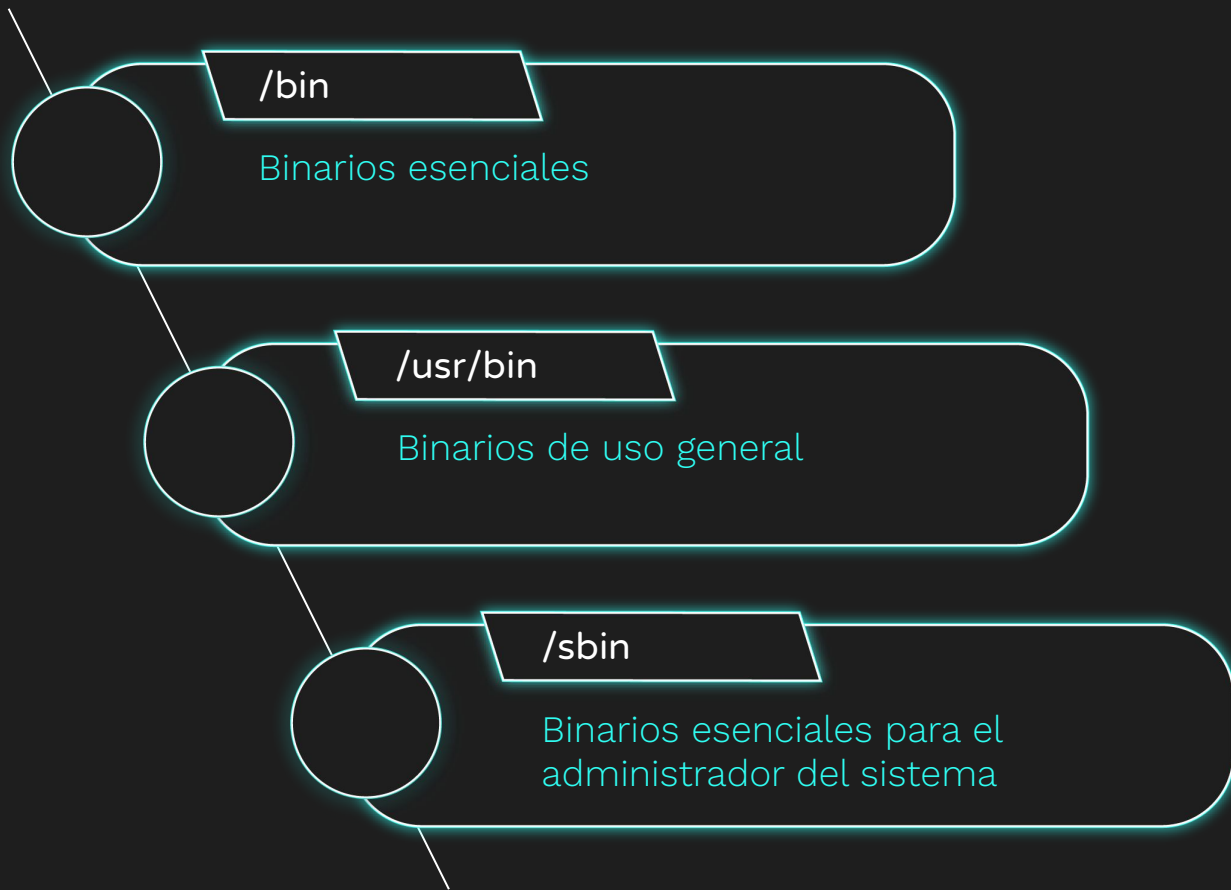
Linux

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

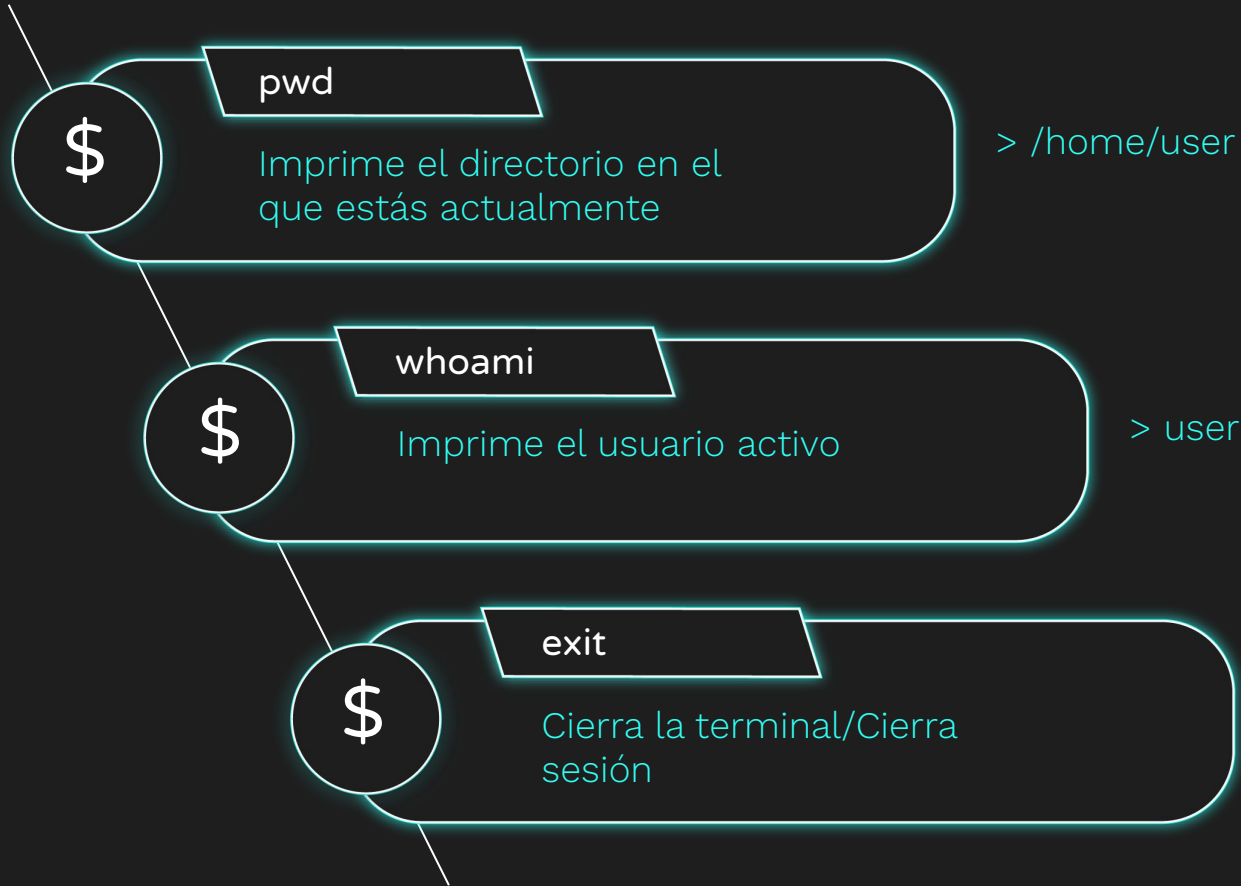
03



03

Comandos del día a día

Comandos de Linux Ubicarte un poco



Comandos de Linux Gestión de ficheros/carpetas



`cd DIR`

Cambiar de directorio



`ls`

Mostrar archivos/carpetas en el directorio actual



`mkdir DIR`

Crea un directorio en la carpeta actual

Comandos de Linux Gestión de ficheros/carpetas

\$

`rm [-r] DIR/FILE`

Elimina un archivo. Si escribes `-r` lo haces recursivo

\$

`cp [-r] SRC DST`

Copia un archivo a DST. Si escribes `-r` lo haces recursivo

\$

`mv [-r] SRC DST`

Mueve un archivo a DST. Si escribes `-r` lo haces recursivo

Comandos de Linux
Lectura/Escritura de archivos

\$

`touch FILE`

Crea un archivo vacío con el nombre FILE

\$

`nano FILE`

Editor de texto de terminal, abre FILE o lo crea si no existe

Más info en la siguiente diapositiva

\$

`cat FILE`

Imprime el contenido del archivo FILE

er' nano

```
GNU nano 7.2 tonto el que lo lea.txt
```

[Nuevo fichero]

Ctrl-A Ayuda	Ctrl-G Guardar	Ctrl-F Buscar	Ctrl-X Cortar	Ctrl-E Ejecutar	Ctrl-O Ubicación	Ctrl-U Deshacer	Ctrl-A Poner marca	Ctrl-L A llave	Ctrl-P Anterior	Ctrl-B Atrás	Ctrl-W Palabr ant
Ctrl-S Salir	Ctrl-H Leer fich.	Ctrl-R Reemplazar	Ctrl-V Pegar	Ctrl-J Justificar	Ctrl-I Ir a línea	Ctrl-E Rehacer	Ctrl-C Copiar	Ctrl-N Buscar atrás	Ctrl-W Siguiente	Ctrl-F Adelante	Ctrl-W Palabr sig

Comandos de Linux Lectura/Escritura de archivos

\$

CTRL + O

Guardar archivo

\$

CTRL + X

Salir

\$

CTRL + K

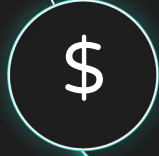
Cortar línea actual

Comandos de Linux Lectura/Escritura de archivos



CTRL + SHIFT + V

Pegar contenido del portapapeles



CTRL + W

Buscar coincidencias en el documento



CTRL + SHIFT + _

Saltar a línea n

\ 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

\$

`head -n X FILE`

Muestra las primeras X líneas de un archivo

\$

`tail -n X FILE`

Muestra las últimas X líneas de un archivo

\$

`less`

Paginación de archivos grandes

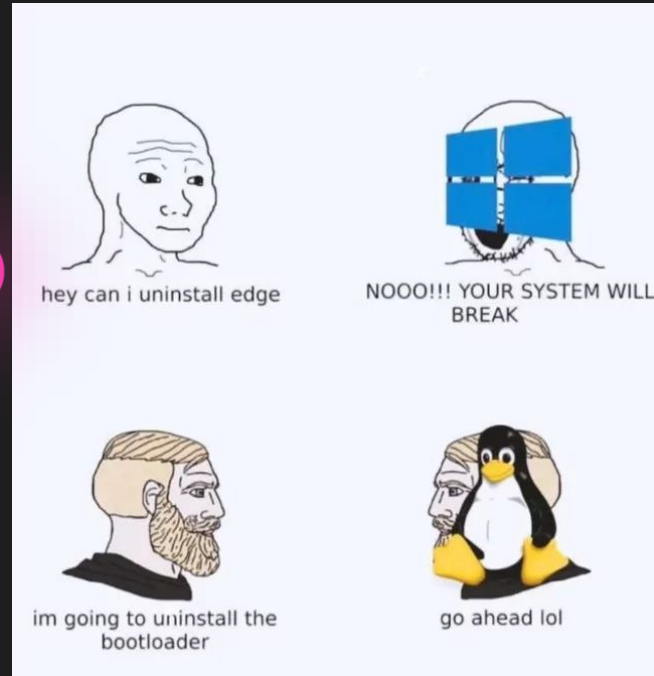
man ls

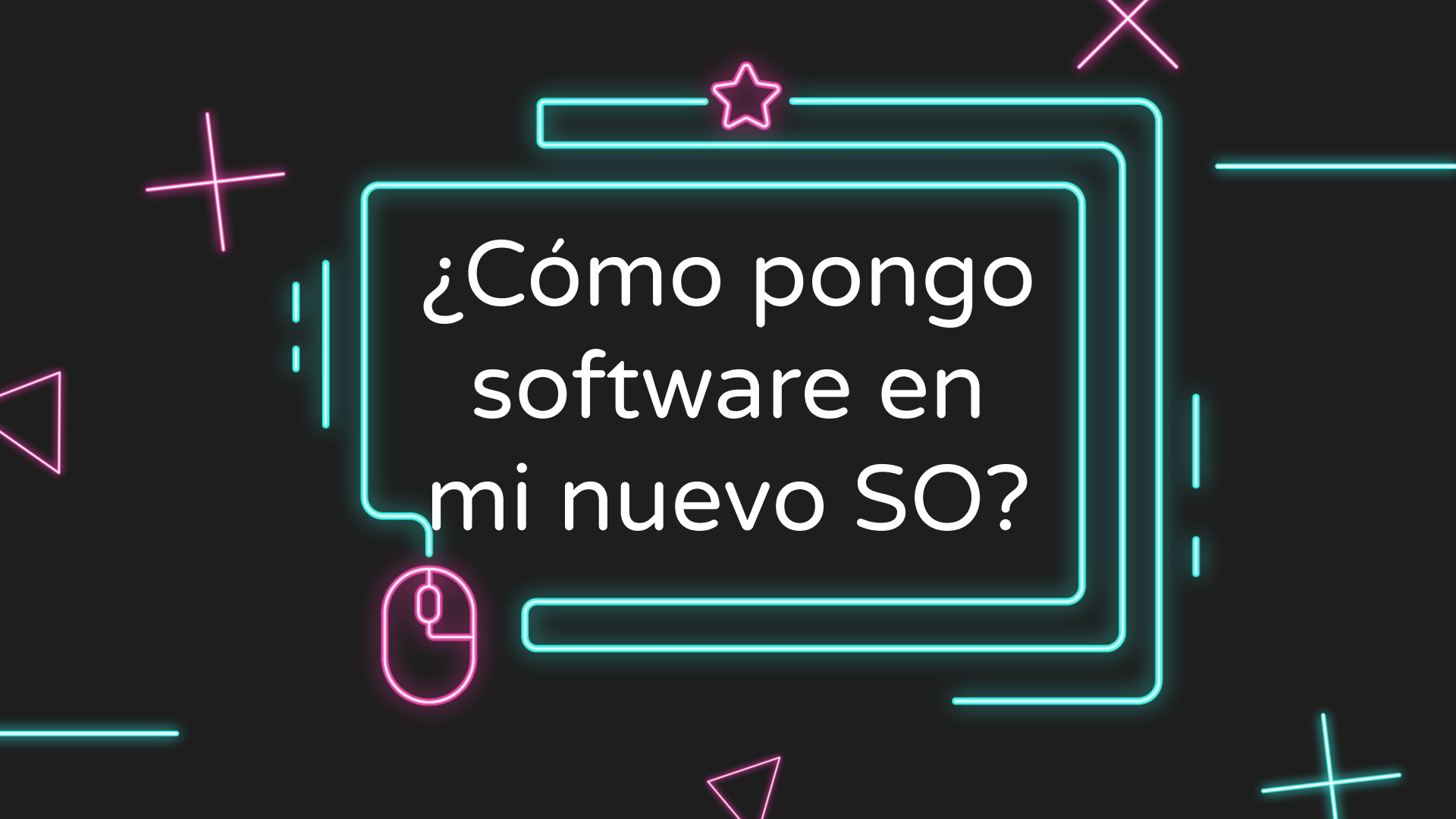
```
LS(1) User Commands LS(1)
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
DESCRIPTION
  List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
  Mandatory arguments to long options are 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 -l, verbose -l, vertical -C
Manual page ls(1) line 1 (press h for help or q to quit)
```

04

04

Gestión de software





¿Cómo pongo
software en
mi nuevo SO?

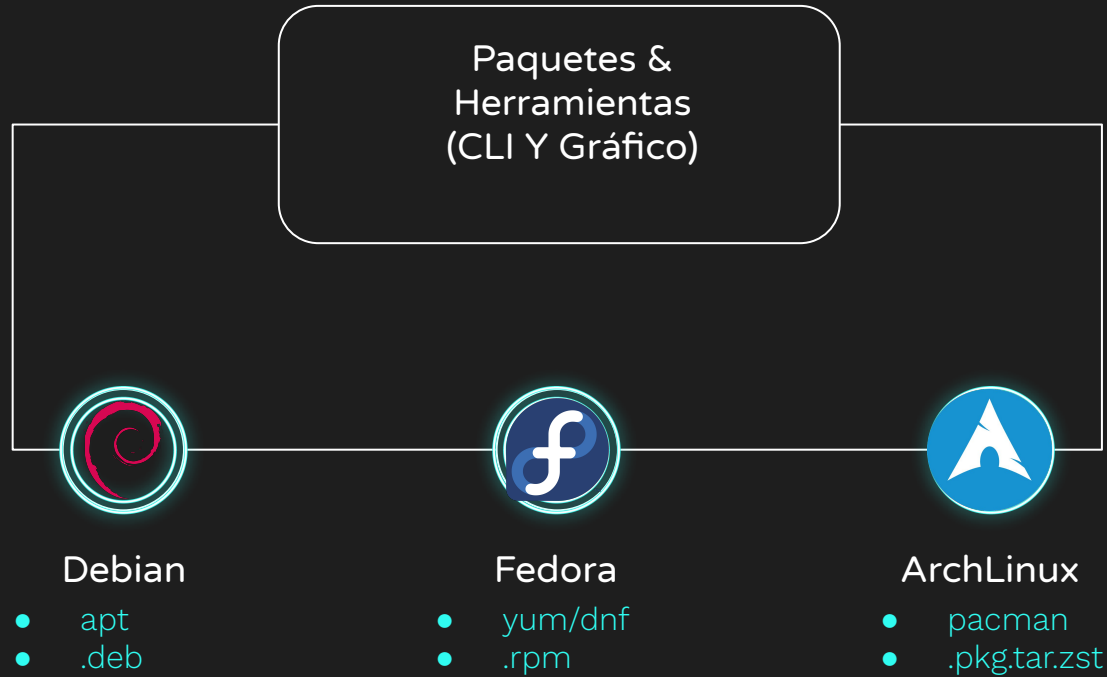


Windows !=
Linux

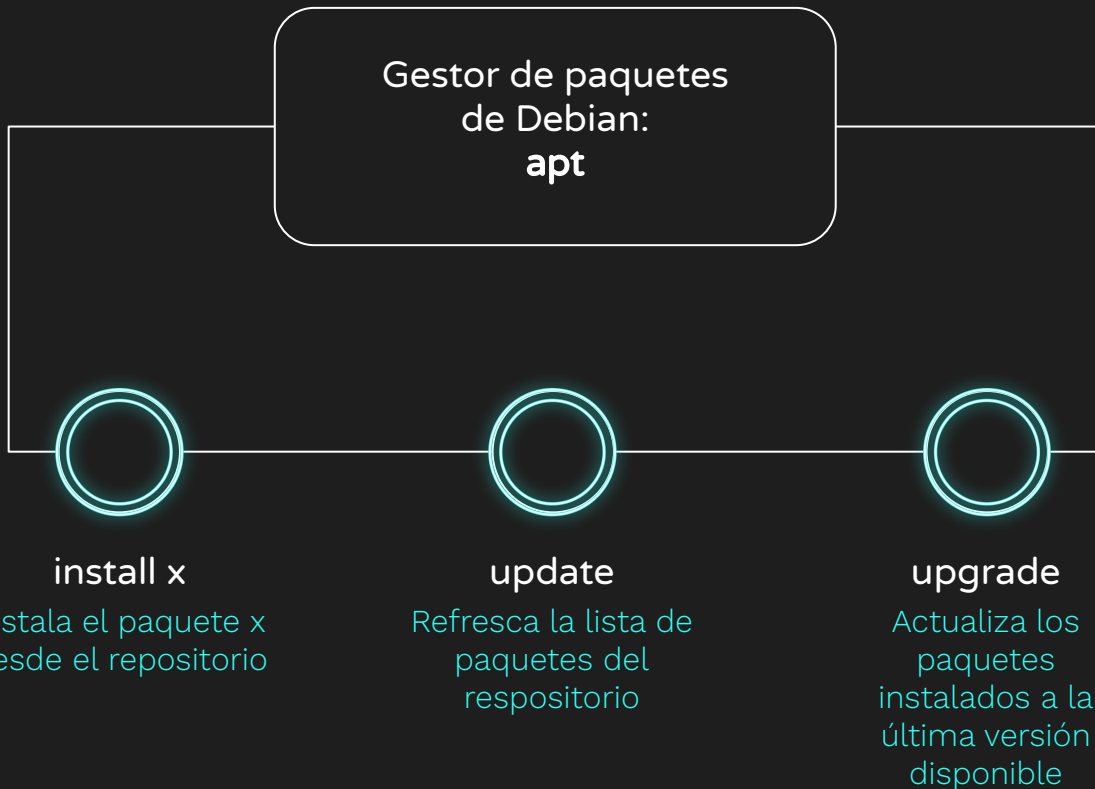
Instalación de software



Instalación de software



APT



APT



remove x

Elimina el paquete
x de tu PC

purge x

Elimina el paquete
y toda la
configuración
vinculada a x de tu
PC

autoremove

Elimina paquetes
huérfanos de tu PC
(paquetes que se
instalaron como
dependencias de otro
paquete pero que ya no
está instalado)

Instalación de software Métodos alternativos



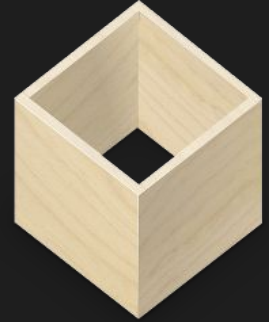
AppImage

- Usable sin instalar más dependencias
- Enfocado en portabilidad
- No aísla el programa del resto del sistema



Snap

- Respaldado por Canonical
- Instalado por defecto en Ubuntu
- Aísla la ejecución en pseudo-containers (AppArmor)



Flatpak

- Respaldado por freedesktop.org
- Aísla la ejecución en pseudo-containers (bubblewrap)

05

Permisos

Papers please

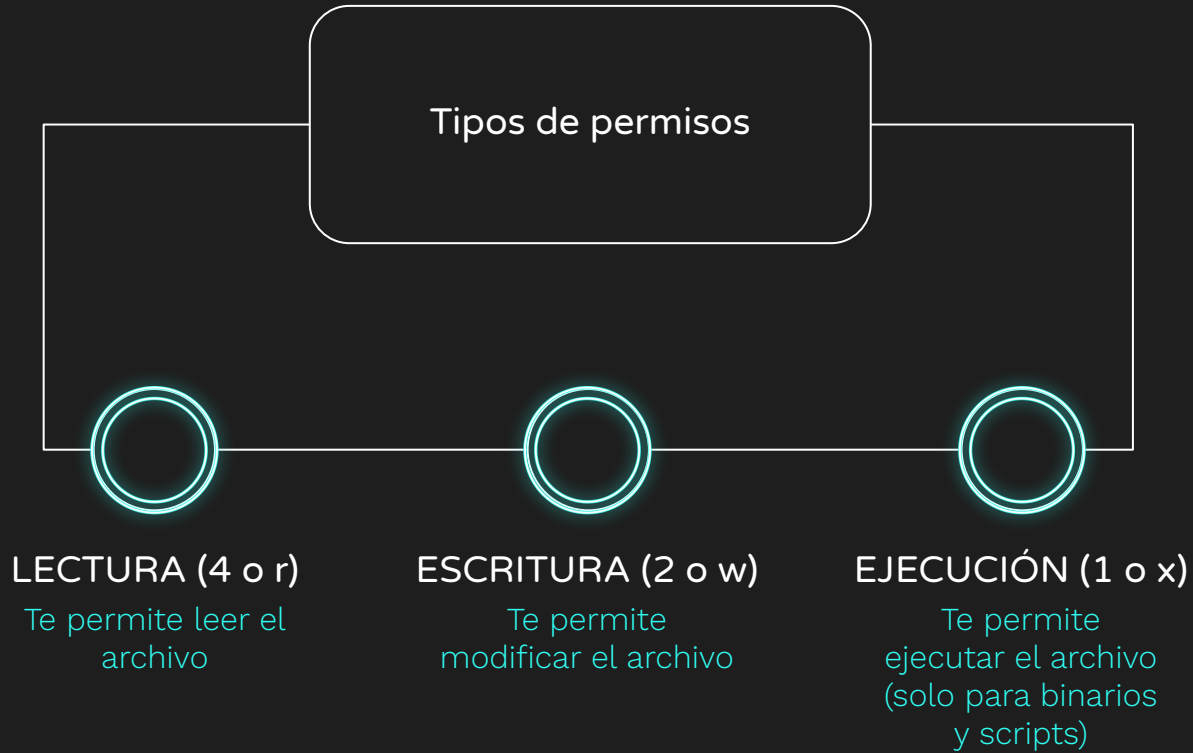


05

Permisos



Permisos



```
[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 600 test.txt
```

Este archivo se quedaría:

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

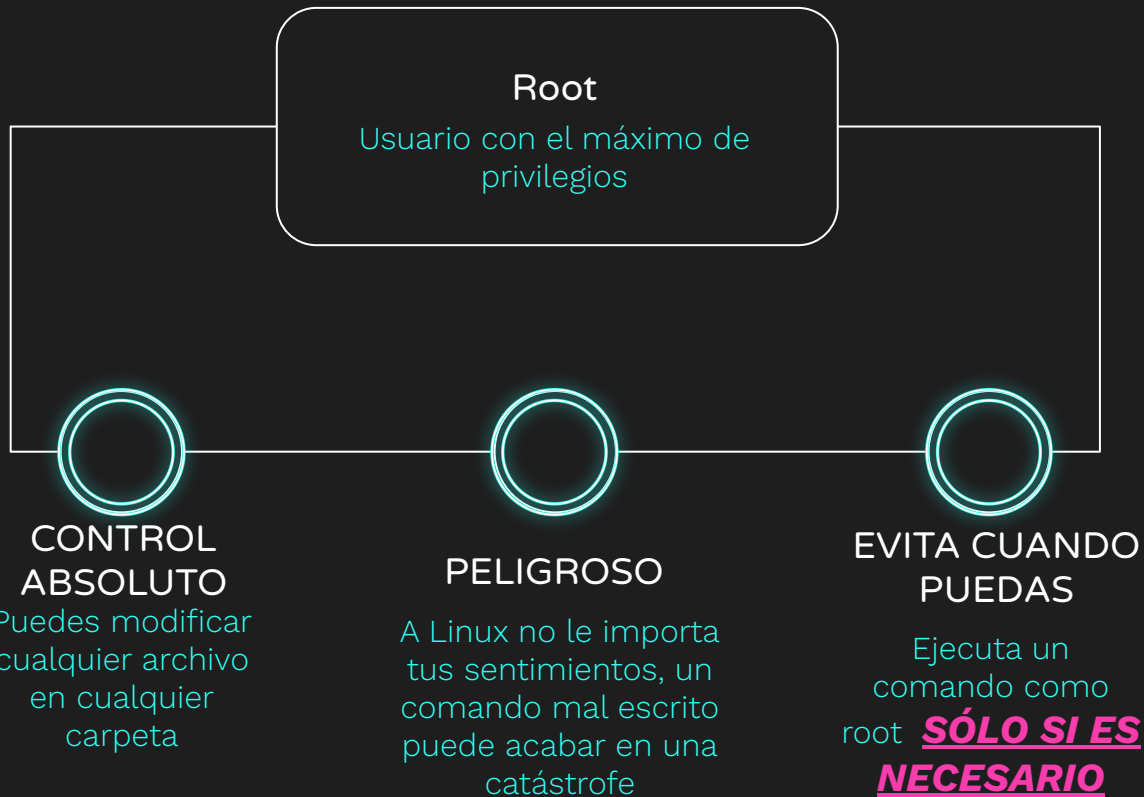

```
[user@debian ~]$ chmod 777 test.txt
```



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





sudo

Permisos root temporales

```
[user@debian ~]$ nano /etc/shadow  
[user@debian ~]$ sudo nano /etc/shadow  
Contraseña: *****
```

06

06

Piping / Redirecciones

Vete a tomar por null

Encadenar comandos

`cmd1 &&
cmd2`

Ejecuta el comando de la derecha si el de la izquierda fue exitoso

`cmd1 || cmd2`

Ejecuta el comando de la derecha si el de la izquierda falló

`cmd1; cmd2`

Ejecuta el comando de la derecha sin importar el resultado del de la izquierda

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

07

07

Scripting en Bash

Que comience el
espectáculo

Hello world



```
#!/bin/bash
```

```
# Hola, soy un comentario
```

```
echo 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`



What if You

./TEST.SH

Wanted to go
to Heaven

**ZSH:
PERMISO DENEGADO:
./TEST.SH**



But god said

imgflip.com

```
$ 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



```
#!/bin/bash  
greeting=Pablo  
echo "Hello $greeting"
```

Comillas

Cada una va a su rollo



'

Interpreta todo literal,
no reemplaza ninguna
variable



"

Interpreta las
variables
(\$NOMBRE) y
concatena



`$() o ``

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

A central graphic featuring the word "if" in white lowercase letters. The text is enclosed within a complex, multi-layered neon cyan border that forms a square with rounded corners and a thick, multi-line effect. Surrounding this central element are various neon pink icons: a star at the top, a mouse cursor at the bottom left, and several plus signs scattered around the edges. The background is solid black, and there are also some horizontal and vertical neon cyan lines extending from the central graphic.

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

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 vacío

`-lt` y `-le`

Menor que y menor o igual que

`&&` y `||`

Comprobar si x e y/o son igual a 0

`A == B` y `A != B`

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

`-gt` y `-ge`

Mayor que y mayor o igual que

`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

```
#!/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

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

echo "He terminado el ciclo"
```

The image features a central white text 'for' enclosed within a glowing cyan rectangular frame. This frame is composed of multiple parallel lines, with a pink star icon positioned at the top center. To the left of the frame, a pink mouse cursor icon is connected to the frame's border by a thin line. The background is black and decorated with various neon elements: pink plus signs, triangles, and a star, as well as cyan horizontal and vertical lines. The overall aesthetic is modern and digital.

for

for



```
#!/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

break

Sale del bucle actual

continue

Salta la iteración actual

Evitar uso

Puedes usar estas keywords, pero no abusos de ellas.

08

08

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

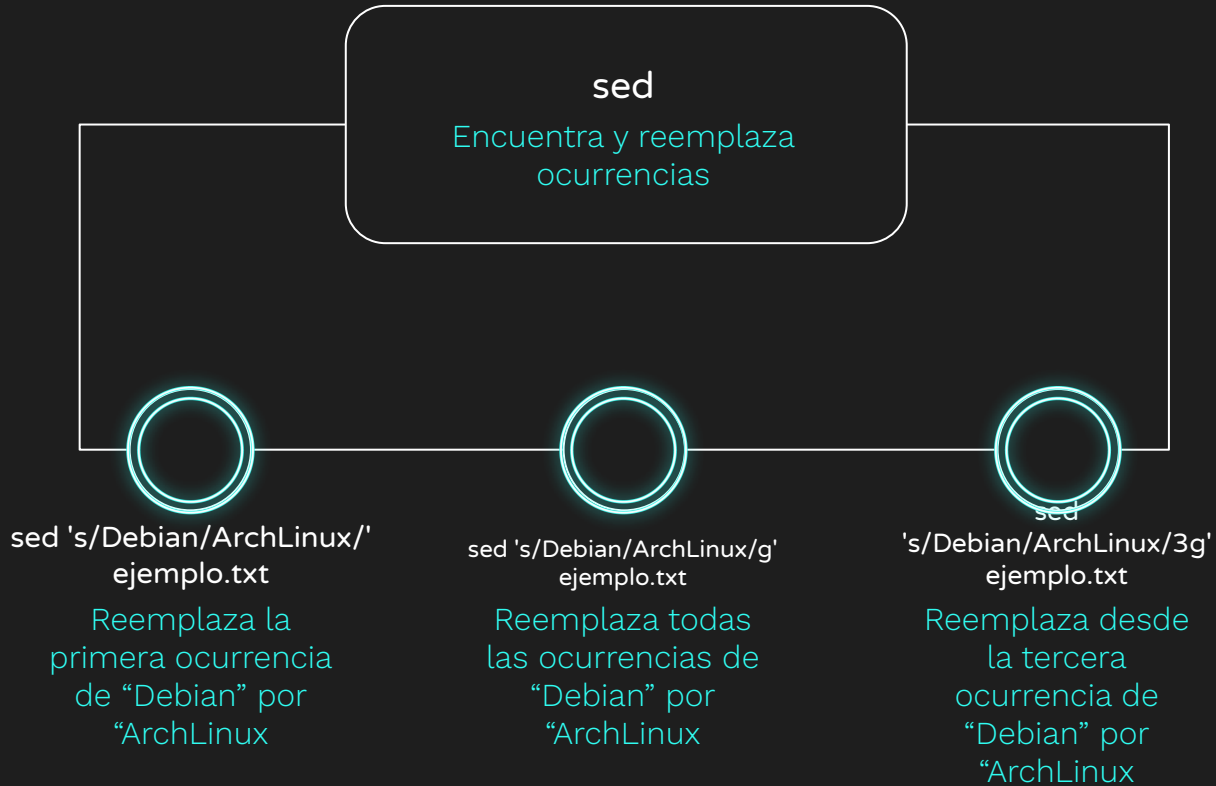
```
grep -r  
PATTERN DIR
```

Busca recursivamente
en DIR archivos que
contengan PATTERN

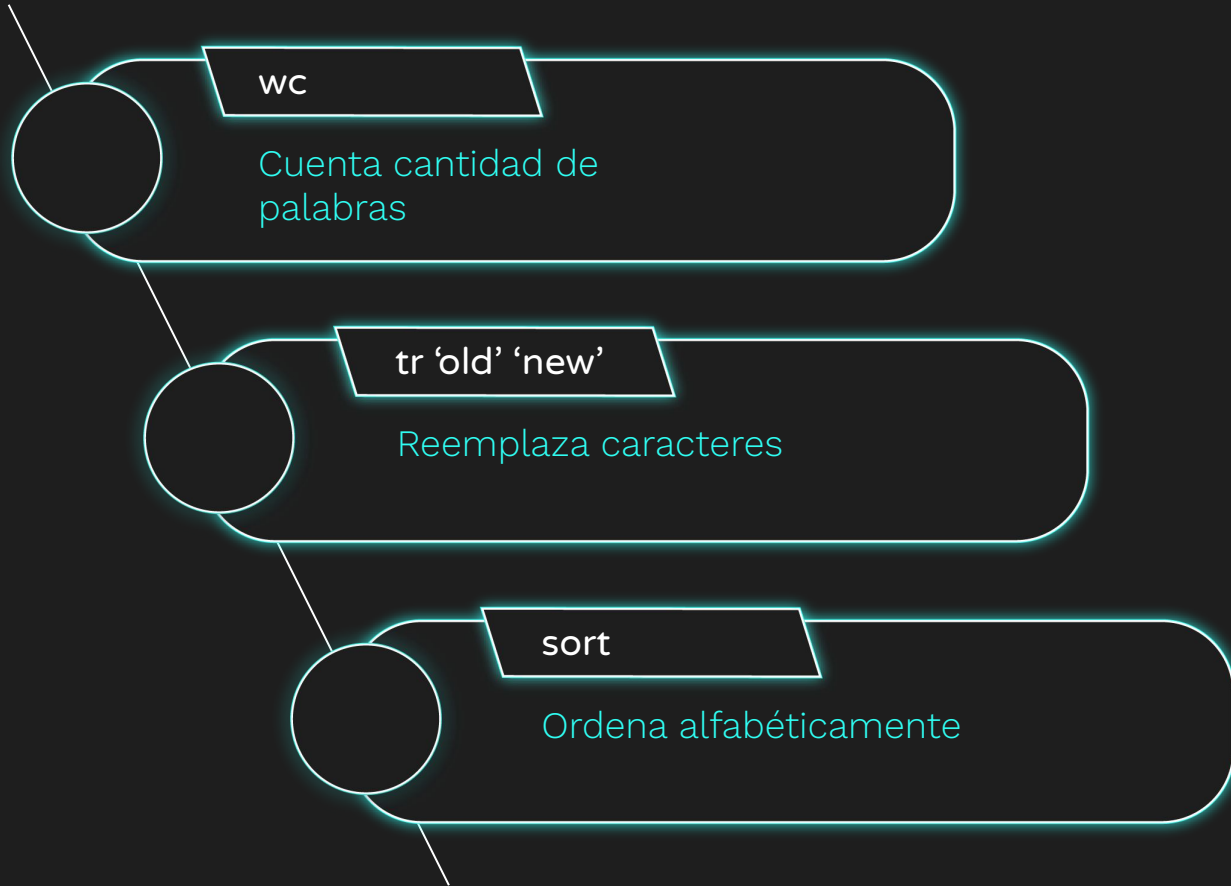
```
grep -in  
python  
ejemplo.txt
```

Encuentra "python"
case-insensitive en
el archivo
ejemplo.txt

Comandos



Comandos



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

Are ya coding
son?



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

