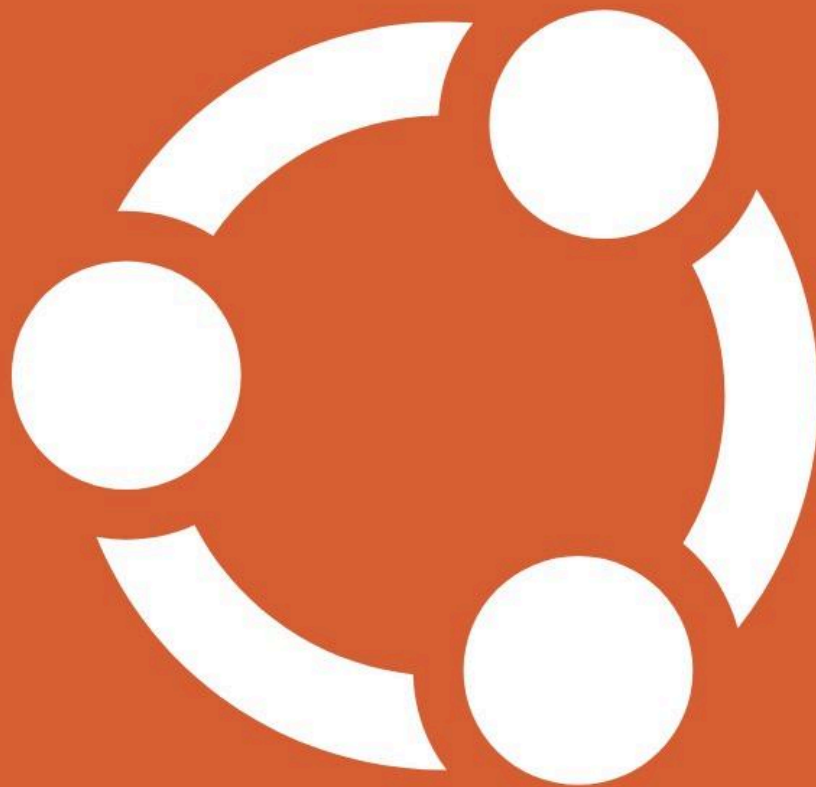


Unidad 7

# Ubuntu





# ÍNDICE

Objetivo	3
Introducción a Ubuntu	3
Entorno de escritorio	4
Sistema de archivos	7
Unidades de disco	7
swap	8
inodo	8
Directorios principales de Linux	9
Ficheros contenidos en la raíz	10
Ficheros importantes del sistema	10
Ficheros del Kernel	11
Bash	12
Estructura del prompt	12
Sintaxis de un comando	13
Comandos básicos fundamentales	14
Linux Command library	15
🌐Links comandos	16
Standard output (stdout) y standard error (stderr)	16
Operadores	17
Regular expressions RegEx	18
Comodines/wildcards	19
Otros caracteres especiales (Ver Caracteres de expresiones regulares)	20
grep	21
sed	22
tr	22
awk	23
Caracteres de expresiones regulares comunes	23
Tubería   (pipe)	24
Utilidades Built-in	25
Rutas absolutas vs. rutas relativas	26
Enlace duros y enlace simbólicos	27
Sudo	29
Usuarios	30
Permisos	32
Grupos	34
Instalación de paquetes/aplicaciones	36
Código de colores consola	37
TTY y terminales virtuales	38
Scripting	39
Networking	<b>40</b>
WSL	43
Otros	45
Glosario de términos	46



## Objetivo

- Aprender el funcionamiento de Ubuntu a nivel usuario
- Adquirir habilidades para la administración y configuración avanzada del sistema.
- Aprender a automatizar tareas mediante la línea de comandos y scripts.
- Conocer la configuración de servicios en un servidor Ubuntu.
- Aprender el funcionamiento del sistema en red
- Aplicar principios de seguridad para proteger el sistema y la red.

## Introducción a Ubuntu

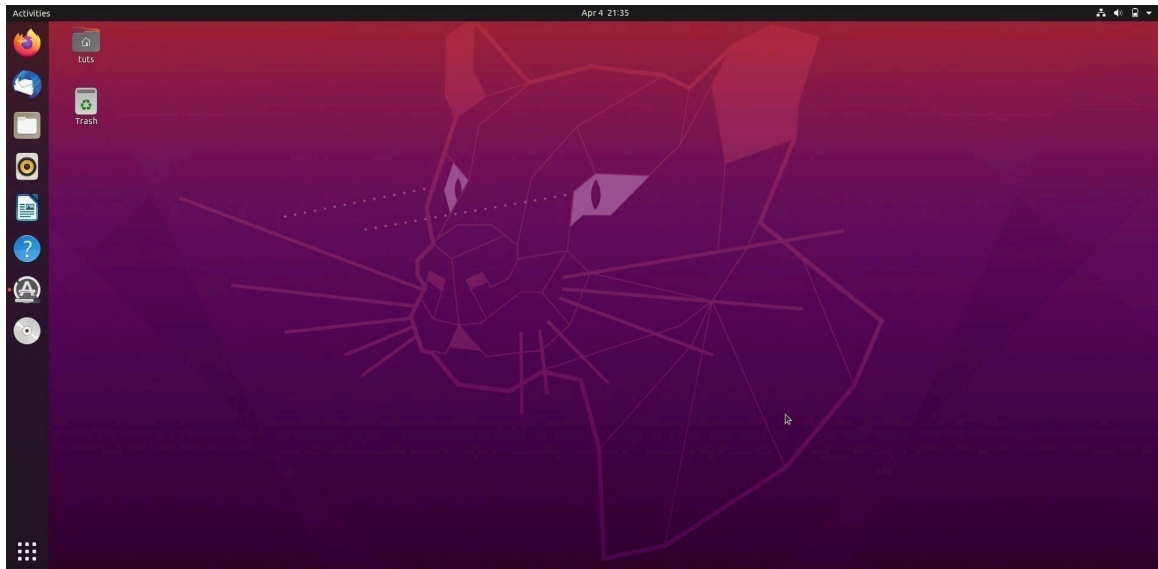
Ubuntu es una distribución GNU/Linux basada en Debian.

- Dispone de versiones de
  - Escritorio
  - Servidor
  - IoT
  - Cloud
    - AWS
    - GCP (Google Cloud)
    - Azure
- La versión estable LTS (Long Term Support) tiene soporte de 5 a 12 años según la versión. Son más estables y serán actualizadas durante ese tiempo.
- La versión PRO incluye mayor soporte de paquetes (gratis hasta 5 máquinas)
- La versión que es de pago para uso profesional se paga por el soporte.
- Soporta múltiples escritorios GUI como GNOME, KDE Plasma, Xfce y LXDE, siendo GNOME la predeterminada (versión customizada por Canonical)
- Opciones de instalación: se puede instalar Ubuntu desde un CD/DVD, una memoria USB o una imagen ISO. Además puede probarse sin instalación.

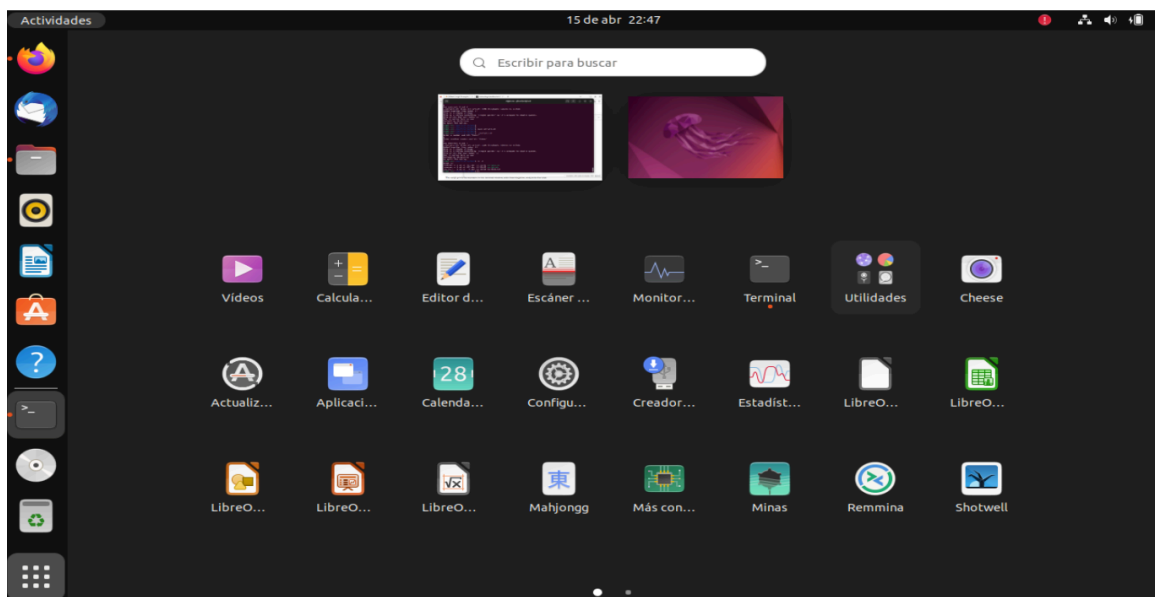


# Entorno de escritorio

- Escritorio GNOME
  - Barra de tareas (izquierda)

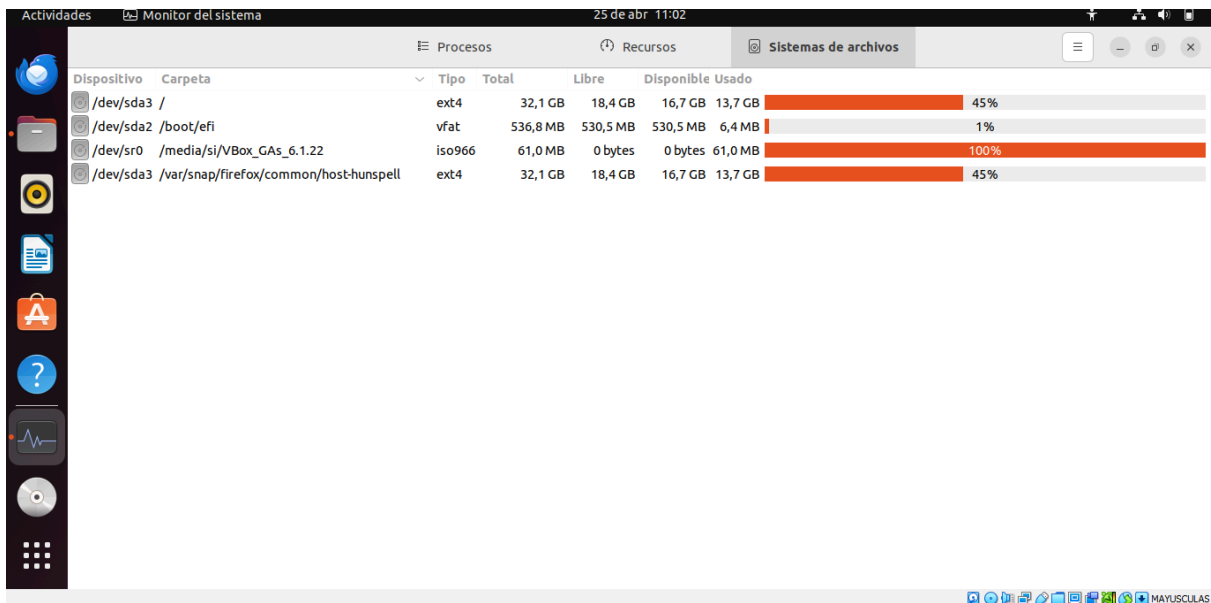


- Menú de aplicaciones (abajo izquierda)
- Panel de control





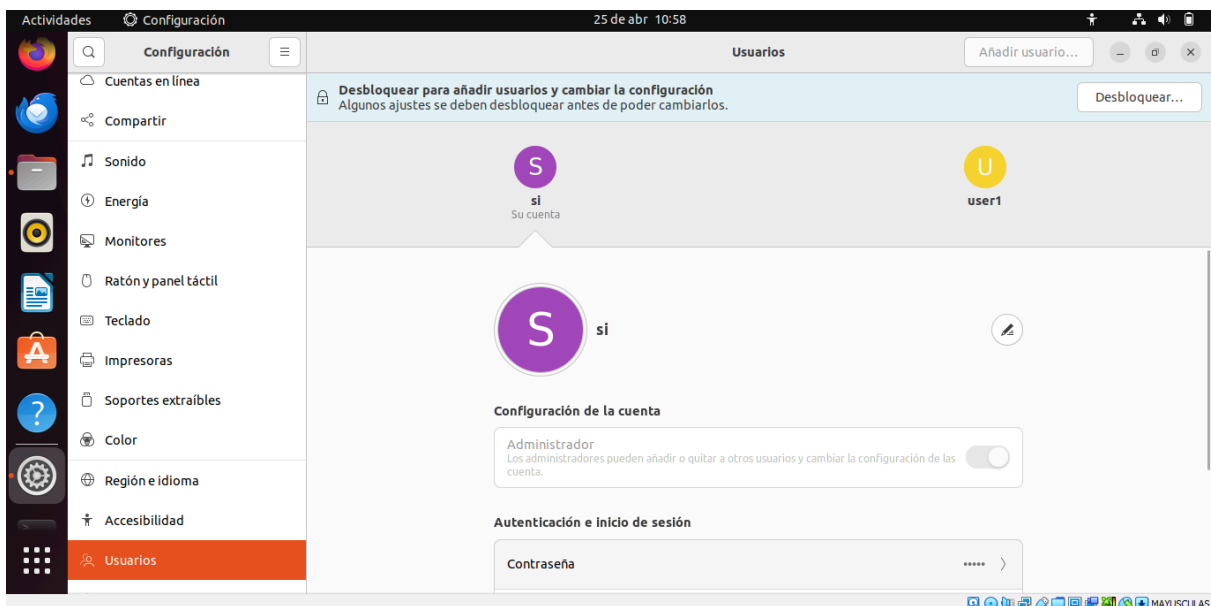
- Monitor del sistema



- Configuración

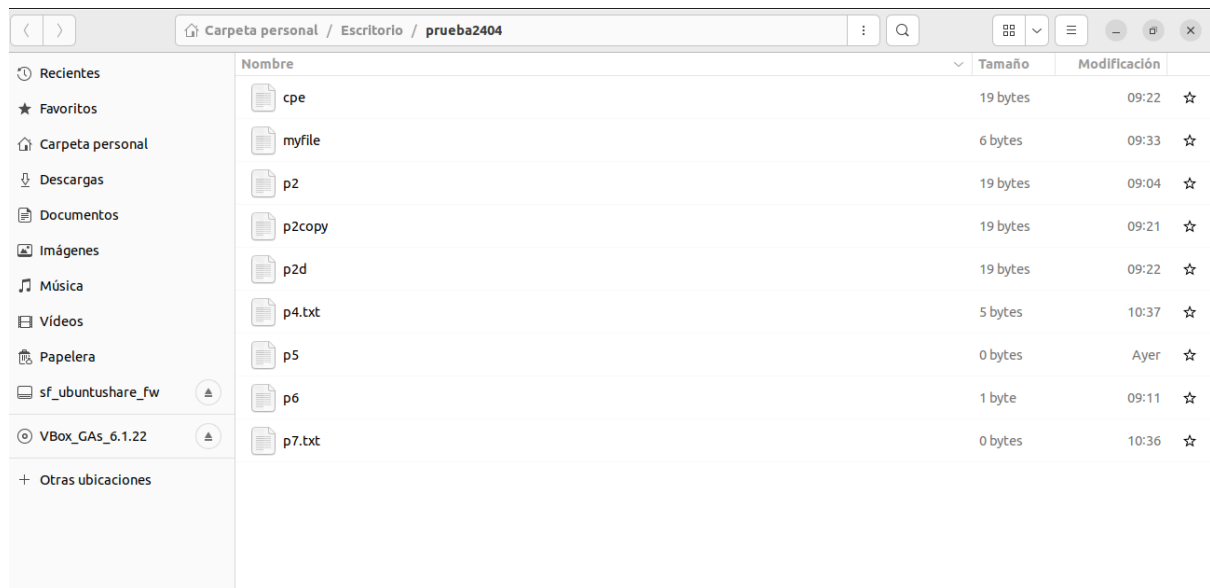
En Configuración podemos cambiar Cuentas de usuario, Red, Apariencia, etc.

Para entrar en cada de una las opciones navegar usando la barra de la izquierda o buscar el elemento (Con la lupa de arriba izquierda)





- Notificaciones
- [Nautilus](#) es el administrador de archivos oficial por defecto



# Sistema de archivos

Los sistemas tipo Unix no dividen las unidades como en Windows en unidades C:\. En su lugar, tienen un **único sistema de archivos unificado**, y las unidades individuales se pueden conectar ("**montar**") en cualquier ubicación del sistema de archivos que tenga más sentido. Para ver los dispositivos montados y su detalle: `lsblk`

El directorio `/`, a menudo denominado directorio raíz (*root directory* en inglés. ⚠ No confundir con usuario root), es la base de este sistema de archivos unificado. A partir de ahí, todo lo demás se ramifica para formar un árbol de directorios y subdirectorios.

[Fuente](#)

## Unidades de disco

Un único directorio raíz representado con `/`

Nomenclatura de los discos duros

- Discos IDE: `hd`
- Discos SATA: `sd`

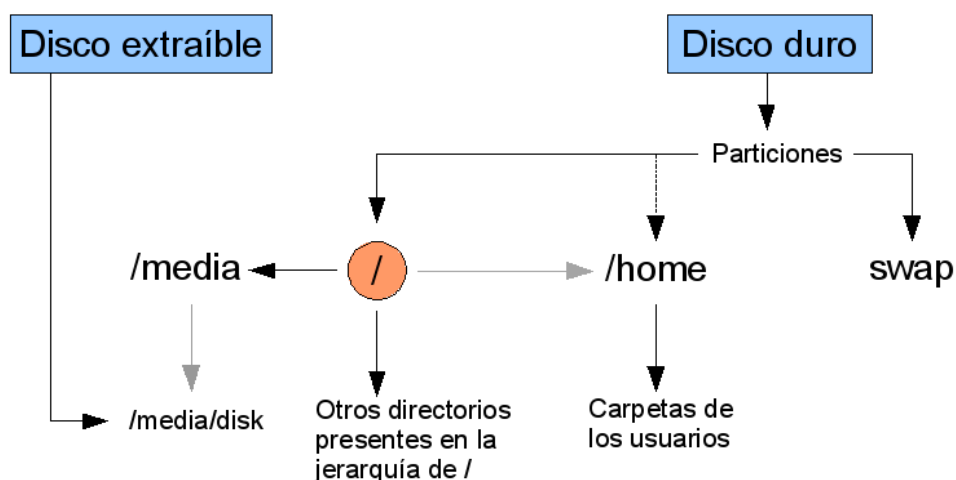
*Dependiendo del lugar donde estén conectados (a, b, c...) --> Ej. `sda`*

Particiones: se indican con un número

- 1...4: reservado para particiones primarias
- 5 en adelante: particiones lógicas

📁 **File Hierarchy Standard (FHS)** [wikipedia](#)

### Ejemplo de una jerarquía del núcleo Linux



⚠ En la actualidad **swap** no tiene porque ser una partición dedicada. Ver abajo.

[Linux File System](#)



## swap

Memoria virtual (espacio de intercambio) en el disco duro que actúa como una extensión de la memoria RAM. Funciona como una especie de desbordamiento para la memoria RAM, entrando en juego cuando la RAM se llena por completo. Ocupa espacio en el disco duro.

### ¿Swap vs archivo swap?

Tradicionalmente, Linux utilizaba una partición dedicada como swap. **Hoy en día, también es posible configurar un archivo swap dentro de una partición existente.** Elegir entre una partición o un archivo depende de tu preferencia y configuración del sistema. Las particiones swap suelen tener un rendimiento ligeramente mejor, pero los archivos swap ofrecen más flexibilidad en el caso de que necesites modificar su tamaño más adelante.

### Tamaño ideal de la partición swap

El tamaño ideal de la partición swap depende de la cantidad de memoria RAM y la distribución que tengas instalada en tu equipo:

- Equipos con poca RAM (hasta 1 GB): Se recomienda un swap del mismo tamaño que la RAM.
- Equipos con RAM moderada (entre 2 y 4 GB): Un swap con la mitad del tamaño de la RAM suele ser suficiente.
- Equipos con mucha RAM (más de 4 GB): Generalmente, no se necesita un swap de 2 GB más que la RAM (ej. Si RAM 16GB → Swap = 18GB)

Como ver en la consola el tamaño de memoria swap

```
swapon -s
```

*Su tamaño se puede cambiar por consola o con herramientas de terceros (ej. gparted)*

## inodo

Es una estructura de datos que contiene la información sobre un archivo o directorio, como permisos, propietario, grupo, tamaño, fechas de acceso/modificación, y punteros a los bloques de datos que contienen el contenido del archivo.

Es como la tarjeta de identificación de un archivo. En lugar de almacenar el nombre del archivo en sí, el inodo guarda información crítica sobre ese archivo. Almacenan archivos regulares, directorios, y enlaces simbólicos. Es identificado por un número entero.

El sistema de archivos Linux mantiene una tabla de inodos que registra la información de todos los archivos y directorios. Cuando se accede a un archivo, el sistema operativo utiliza el número de inodo para localizar la información correspondiente en la tabla de inodo.

```
ls -li #lista inodos del directorio
ls -oi nombreArchivo #lista inodo del archivo
ls -li li #lista inodos del archivo con detalle
```





## Directorios principales de Linux

**/** : Carpeta raíz del sistema

**/bin** : comandos básicos del sistema

**/sbin** : comandos de administración del sistema

**/boot** : ficheros del Kernel y arranque (GRUB)

**/dev** : ficheros de dispositivos (descriptores)

**/etc** : ficheros de configuración del sistema

**/home** : almacena carpetas de usuarios

**/root** : directorio del usuario root

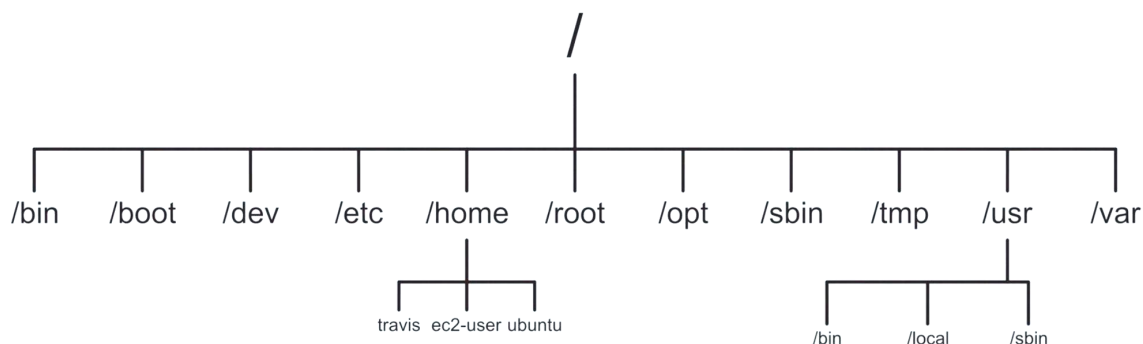
**/opt** : almacena paquetes de aplicaciones que no vienen por defecto

**/tmp** : info. Temporal de la sesión de usuario

**/usr** : (**U**nix **S**ystem **R**esources) código fuente de aplicaciones (binarios) de aplicaciones-paquetes instalados

**/var** : ficheros de tamaño variable (logs)

⚠ Ojo diferencia entre **/bin** y **usr/bin**



🌟 *travis; ec2-user; ubuntu son nombres de usuario ejemplo*

**/lib** : librerías usadas por **/bin** y **/sbin**

**/mnt** : montaje de dispositivos externos

**/proc** : muestra información del Kernel (FS virt.)

**/lost+found** : para la utilidad **fsck** (Recovery)

**/media** : montar sistemas de ficheros temporal

**/srv** : ficheros con información del sistema

**/sys** : información de dispositivos (Kernel). Evolución de **/proc**



## Ficheros contenidos en la raíz

**initrd.img:** para montar el sistema de ficheros

**initrd.img.old:** copia de backup de initrd.img

**swapfile:** archivo SWAP (memoria virtual)

**vmlinuz:** enlace simbólico al Kernel

## Directorios especiales en Linux

. → Directorio actual

.. → Directorio padre



Directorios y ficheros ocultos en Linux empiezan por un punto .



Los nombres de archivo y directorio son **CASE SENSITIVE**  
(ie., diferencian entre mayúsculas y minúsculas)

## Ficheros importantes del sistema

/etc/hostname

/etc/hosts

/etc/passwd

/etc/resolvconf

/etc/crontab

/etc/fstab

/etc/adduser.conf

/etc/apt/sources.list

/etc/shadow

/etc/network/interfaces

.bashrc

.profile

.bash\_logout

.bash\_history



## Ficheros del Kernel

Se ubican en el directorio **/boot**

<b>config</b>	Fichero de configuración del Kernel
<b>initrd.img</b>	RamDisk inicial
<b>System.map</b>	Tabla de símbolos del Kernel
<b>vmlinuz</b>	Fichero del Kernel (está comprimido)
<b>GRUB</b>	Cargador de arranque
<a href="https://www.guia-ubuntu.com/index.php/GRUB">https://www.guia-ubuntu.com/index.php/GRUB</a>	
<b>Ficheros memtest</b>	Usados para test de RAM



## Bash

Intérprete de línea de comandos que se ejecuta en una ventana de texto donde el usuario escribe órdenes en modo texto actuando así como interfaz con el ordenador.

Se le denomina también shell, terminal (nombre de la aplicación en Ubuntu), consola, prompt, command line, CLI, entre otros nombres.

 <https://ubuntu.com/tutorials/command-line-for-beginners#1-overview>

 <https://terminaldelinux.com/terminal/>

 Se puede leer y ejecutar órdenes desde un archivo, llamado guión o 'script'

---



**Atajo de teclado para abrirla en Ubuntu**

Y mayoría de distros



## Estructura del prompt



**si** → nombre usuario

**si-vm** → nombre equipo (hostname)

**~** → Indicador que estás en el directorio /home

**/Documentos** → directorio

**\$** → Indicador de tipo de usuario

**\$**: normal

**#**: root



## Sintaxis de un comando

---

`nombre_comando` [`opciones`] [`valor`]

---

**Opciones** también se denominan **flags** o **argumentos**

Las opciones puede ser:

- Un sólo carácter precedido por un guión. Ejemplo: -u
- Varios caracteres/palabra precedida por doble guión. Ejemplo: --all

*Algunos comandos tienen versión abreviada. Ejemplo: -all == a*



Los comandos son **CASE SENSITIVE** (diferencia entre mayúsculas y minúsculas).  
Pudiendo no funcionar si se cambia, o ser un comando diferente

**i** Existen muchos comandos **silent**, que no muestran salida de texto tras ejecutarse (ej. mkdir, cd)

**i** El carácter **#** precede a los comentarios.  
ie. lo que esté a su derecha NO se ejecuta.



## Comandos básicos fundamentales

pwd	print working directoy
ls	ls -l (ver permisos directorio) ls -l nombreArchivo ls -a
cd	cd nombreDirectorio cd /home cd .. cd /
whoami	usuario
mkdir nombreDirectorio mkdir /ruta/nombreDirectorio mkdir ruta/nombreDirectorio	make directory
help bashcommand comando -help	ayuda
man <i>comando</i>	Muestra la ayuda en una pantalla temporal
whatis <i>comando</i>	Definición corta
more	muestra contenido por paginas
less	muestra contenido por paginas. permite usar flechas
head	mostrar el principio de un archivo
tail	mostrar el final de un archivo
touch	crear archivo  si se precede por punto . será oculto
cat	muestra el contenido de archivos. Tb concatena ( si se le pasa + de un archivo)
nano	abre el archivo de texto
echo	escribir en pantalla
mv	mueve archivos y/o cambia de nombre
rm rmdir	Borra. 🗑️ No hay papelera!
diff	comparar datos entre dos archivos de texto y mostrar la diferencia.
>	salida del comando es redirigida a archivo.



	🌟 Sobreescribe
>>	salida del comando es redirigida a archivo. Añade, no sobreescribe (append)
<	entrada del comando es tomada de archivo
grep	permite buscar palabras o cadenas de texto en archivos o en la entrada estándar de la terminal. Ver <a href="#">grep</a>
file	<a href="#">Busqueda</a>
cp	copia -R (Copia recursiva)
df -h	disco libre
du -h	
!!	retype comando
find	busca (tb en subdirectorios)
top	procesos
ps	procesos (ps aux)
kill	acaba con proceso

## Linux Command library

<https://linuxcommandlibrary.com/>



## Links comandos

<https://www.linuxtrainingacademy.com/linux-commands-cheat-sheet/>

<https://www.geeksforgeeks.org/linux-commands-cheat-sheet/>

<https://itsfoss.com/linux-terminal-basics/>

<https://www.puttygen.com/linux-commands>

### Histórico de comandos (realizados por ese usuario en el Shell)

```
history
```

size: <https://superuser.com/questions/137438/how-to-unlimited-bash-shell-history>

---

Para saber donde se encuentra un comando, ejecutar:

#### **which comando**

 Hay elementos que pueden estar en dos localizaciones aunque varíen ligeramente (ej. en /bin y usr/bin)

#### **whereis comando**

Muestra la localización de los archivos que tengan relacionados con ese comando

---



Si separas dos comandos por punto y coma ";"  
ambos comandos serán ejecutados (por ese orden).

Se pueden añadir más de 2.

Si falla uno se ejecuta el siguiente

## Standard output (stdout) y standard error (stderr)

**Salida estándar (stdout):** Es el flujo de salida por defecto donde va la salida normal de un comando. Se muestra en el terminal por defecto. Puede redirigirla a un archivo utilizando >.

**Standard error (stderr):** Es el flujo de salida para los mensajes de error y las advertencias. También se muestra por defecto en el terminal. Puede redirigirlo a un archivo utilizando >>

Los flujos **stdout** y **stderr** pueden mezclarse a veces en el terminal, pero son canales independientes.





## Operadores

### Operadores de redirección:

**>, >>:** Redirigir la salida estándar a un archivo.

**<, <<:** Redirigir la entrada estándar desde un archivo o una cadena.

**2>, 2>>:** Redirigir la salida de error a un archivo.

**|:** Tubería para encadenar comandos, enviando la salida del primero como entrada al siguiente.

### Operadores de asignación:

**= :** Asignar un valor a una variable.

**+=, -=, \*=, /= :** Asignar el resultado de una operación a una variable (suma, resta, multiplicación, división).

### Operadores lógicos:

**&&:** AND lógico. Evalúa si ambas expresiones son verdaderas.

```
test -f archivo.txt && test -r archivo.txt && echo "El archivo archivo.txt existe y es legible"
```

**||:** OR lógico. Evalúa si al menos una expresión es verdadera.

⚠ Con grep el **or** es **|** (una sola barra)

**!:** Negación lógica. Invierte el valor de la expresión.

### Operadores de comparación:

**==, != :** Igualdad y desigualdad.

**<, >, <=, >= :** Menor que, mayor que, menor o igual que, mayor o igual que.

### Operadores de prueba:

**-e, -f, -d:** Comprobar si existe un archivo, un archivo regular o un directorio, respectivamente.

**-s:** Comprobar si un archivo tiene un tamaño mayor que 0 bytes.

**-z:** Comprobar si una variable está vacía.



## Regular expressions RegEx

Son un tipo de patrón que se utiliza cuando se trabaja con texto. Se utilizan para buscar y manipular texto de manera precisa y eficiente. Son parte fundamental de varios comandos de shell, como **grep**, **sed** y **awk**, y permiten realizar tareas complejas con archivos de texto.

- regex es *expensive* - regex es a menudo la parte de un programa que consume más CPU. Y una expresión regular no coincidente puede ser incluso más cara de comprobar que una coincidente.
- regex es *codicioso* - Es extremadamente fácil hacer coincidir mucho más de lo previsto, lo que lleva a errores. En muchas ocasiones hemos tenido problemas con regex demasiado codiciosas, causando problemas en nuestros sitios.
- regex es *opaco* - Incluso las personas que conocen bien regex tardarán un tiempo en descifrar una nueva cadena regex, y aún así es probable que cometan errores. Esto tiene un enorme coste para el mantenimiento del proyecto a largo plazo. (Eche un vistazo a esta asombrosa regex para direcciones de correo electrónico RFC822)

Si bien existen y se usan en diferentes lenguajes de programación, existen diferentes estándares o dialectos

Fuente:  <https://ubuntu.com/blog/regex-basics>

Más info: <https://webapps.lehigh.edu/hpc/training/Shell-Scripting-2.pdf>  
<https://cheatography.com/davechild/cheat-sheets/regular-expressions/>



## Comodines/wildcards

Facilitan el trabajo de administración, sustituyendo uno o varios datos alfanuméricos

- **\*** : cero, uno o varios caracteres

```
ls *txt #mostrará archivos terminados en txt
```

```
ls *x* #mostrará aquellos que tengan una x entre los caracteres del asterisco
```

```
ls t* #mostrará aquellos que empiecen por t
```

```
ls *t #mostrará aquellas que terminen en t
```

- **?** : un carácter

```
ls test?.txt #mostrará archivos como test1.txt, test2.txt...
```

Asumiendo que están disponibles. Si existe una llamado test.txt **NO** lo mostrará

- **[]** : un elemento (de los indicados). Ejemplos:  
[abc]: una a, una b o una c  
[a-c]: similar al anterior (intervalo)

⚠ Funciona con un solo número del 0-9 (menor de 10) y también con caracteres a-z. Es case sensitive.

- **{}** : Permite especificar varios archivos o directorios separados por comas dentro de un comando.
- **[!]** : Se excluyen los caracteres entre corchetes

```
ls test[!2].txt #se mostrará test1.txt, test3.txt...asumiendo que existen
```



## Otros caracteres especiales (Ver [Caracteres de expresiones regulares](#))

- **+**: matches 1 o más instancias
- **\$**: Se utiliza para saber el último carácter de un string
- **^**: Niega un conjunto de caracteres. Se usa para conocer el primer carácter de un string. O hacer un match que coincidirá con cualquier cadena que comience con la palabra dada. Equivalente a `[!]` en standard wildcards. This performs a logical "not".
- **\ (backslash)**: carácter *escape*, para proteger un carácter con significado *especial* posterior.
  - Por ejemplo, `\.` coincidencias "."; `\+` coincidencias de expresiones regulares "+"; y `\"` coincidencias de expresiones regulares "(".
  - También necesita usar expresiones regulares `\\` para hacer coincidir `"\"` (barra invertida).
  - Regex reconoce secuencias de escape comunes, como `\n` nueva línea, `\t` tabulación, `\r` retorno de carro
- **.** (*punto*): El patrón "hola." coincidirá con cualquier cadena que contenga la letra "hola" seguida de cualquier otro carácter. (similar al asterisco)
- **|** Representa el OR lógico para separar valores alternativos



Más info en: <https://tldp.org/LDP/GNU-Linux-Tools-Summary/html/x11655.htm>

## IMPORTANTE

Los ejemplos asumen que estás en el mismo directorio. Pero se le podría facilitar una ruta como se muestra en el ejemplo de abajo:

```
ls directorioA/listar/*.txt
```

Los comodines se pueden combinar. Ejemplo:

```
ls [prs]* #buscará elementos que empiecen por p,r o s y terminen en cualquier otro carácter/es
```



## grep

*Globally Search For Regular Expression and Print out*

Herramienta del shell que permite la búsqueda de patrones en **archivos**.

 <https://www.freecodecamp.org/espanol/news/grep-command-tutorial-how-to-search-for-a-file-in-linux-and-unix-with-recursive-find/>

 <https://www.ionos.es/digitalguide/servidores/configuracion/linux-comando-grep/>

### Ejemplos

```
grep dev texto.txt #buscará 'dev' en el archivo de texto dado
```

```
grep -i dev texto.txt #sin distinción de mayúsculas/minúsculas
```

```
grep -n dev texto.txt #con número de línea
```

```
grep -n dev * #en todos los archivos de ese directorio  
#con la opción 's' no muestra los mensajes de error
```

```
grep -r dev ./*.txt #buscará la palabra 'dev' en todos los txt dentro del directorio  
o subdirectorio. La opción -r es la que permite la búsqueda recursiva
```


Las comillas 'simples' o "dobles" son requeridas alrededor del texto  
si es más de una palabra la que se busca.

Con expresiones regulares extendidas **egrep** el 'or' es | (una sola barra  
acompañada del \ que es el caracter de "escape")

 En resumen se ejecutan con: **egrep o grep -E comando**

```
grep -E 'hell|mad' p1 #buscará palabras que contengan esos caracteres en  
p1
```

 Más información sobre [Expresiones regulares Y extendidas](#)

 **Grep y asterisco \* puede tener comportamiento inesperado. Ver más en:**  
<https://askubuntu.com/questions/681637/grep-the-asterisk-doesnt-always-work>



## sed

**editor de texto no interactivo.** Esto significa que **no se hace ningún cambio directamente en el archivo que estás editando**. En su lugar, primero se crea un archivo temporal cuyo contenido es posteriormente transferido al archivo de origen. Opera línea por línea.

<https://www.ionos.es/digitalguide/servidores/configuracion/comando-sed-de-linux/>

### Sintaxis comando

```
$ sed [parámetro(s)] 'orden(es)' [archivo(s)]
```

### Ejemplos

```
sed 's/palabra1/palabra2/g' archivo.txt #sustituye palabra1 por palabra2. 🌟 El archivo NO CAMBIA. Solo se muestra en consola.
```

Para cambiar el archivo:

```
sed -i 's/palabra1/palabra2/g' archivo.txt
```

```
sed -i '/^$/d' archivo.txt #eliminar líneas en blanco del archivo
```

*No quita espacio al principio o final*

## tr

Traduce, elimina y comprime caracteres de la entrada y escribe el resultado en la salida. Puede realizar operaciones como eliminar caracteres repetidos, convertir mayúsculas a minúsculas y reemplazar y eliminar caracteres básicos. Acepta dos conjuntos de caracteres, normalmente con la misma longitud (aunque no necesariamente, y sustituye los caracteres de los primeros conjuntos por los caracteres correspondientes del segundo conjunto.

```
tr [opción] ... SET1 [SET2]
```

<https://linuxize.com/post/linux-tr-command/>

<https://geekland.eu/uso-del-comando-tr-en-linux-y-unix-con-ejemplos/>

Ejemplo de cómo sustituir uno o varios **caracteres** en un archivo

```
tr 'caracteres_originales' 'caracteres_nuevos' < archivo_origen > archivo_destino
```

```
tr 'oa' 'sb' <archivoentrada >archivosalida #cambia o por s y a por b
```

```
tr 'oa' 's' < archivoentrada > archivosalida #cambia o y a por s
```



## awk

### Sintaxis comando

```
$ awk [opción] "condición {sentencia}" [archivo obtenido]
```

<https://www.ionos.es/digitalguide/servidores/configuracion/comando-awk-de-linux/>

Ejemplos: <https://linuxhandbook.com/awk-command-tutorial/>

### Opciones:

- F [Separador]: define el elemento separador de un archivo. El valor por defecto es el espacio.
- f [nombre de fichero]: especifica el archivo en el que se debe ejecutar el comando awk.
- v: se utiliza para añadir una variable.

### Caracteres de expresiones regulares comunes

.	Coincide con cualquier carácter (excepto las nuevas líneas, normalmente)
\	Escapar de un carácter especial (por ejemplo, \. coincide con un punto literal)
?	El carácter anterior puede estar presente o no (por ejemplo, /hell?o/ coincidiría con <b>hello</b> <b>helo</b> )
*	Se permite cualquier número del carácter anterior (por ejemplo, .* coincidirá con cualquier cadena de una sola línea, incluida una cadena vacía, y se usa mucho)
+	Uno o más de los caracteres anteriores ( .+ es igual .* excepto que no coincidirá con una cadena vacía)
	"o", coincide con la sección anterior o la siguiente (por ejemplo, <b>hello</b>   <b>mad</b> coincidirá con "hello" o "mad")
()	agrupar una sección. Esto puede ser útil para condicionales ( (a b) ), multiplicadores ( (hello)+ ) o para crear grupos para sustituciones (ver más abajo).
{ }	Especifique cuántos caracteres anteriores (por ejemplo, a{12} coinciden con 12 "a" seguidas)
[ ]	Combina cualquier carácter de este conjunto. -define rangos (por ejemplo, [a-z] es cualquier letra minúscula), ^significa "no" (por ejemplo, [^,]+ coincide con cualquier número de no comas en una fila)
^	comienzo de línea
\$	Fin de la línea

[Fuente](#)



## Tubería | (pipe)

En Linux, una tubería o **pipe** en inglés (representada por el símbolo de barra vertical **|**) es un mecanismo que permite **conectar la salida de un comando con la entrada de otro comando**. Crea un flujo de datos temporal unidireccional entre los comandos. Permite encadenar eficientemente varios comandos para realizar tareas complejas sin necesidad de guardar resultados intermedios en archivos.

### Sintaxis

```
comando1 | comando2 | ... | comandoN
```

comando1: El primer comando que produce una salida.

| El símbolo de tubería que redirige la salida.

comando2: El segundo comando que recibe la salida de comando1 como entrada.

...: Puedes encadenar varios comandos usando tuberías adicionales.

comandoN: El último comando en la tubería que procesa la salida final.

### Ejemplos

Listar archivos y filtrar por extensión:

```
ls | grep '.txt' # Lista archivos y muestra solo aquellos con la extensión .txt  
#equivalente a ls *.txt
```

ls lista todos los archivos en el directorio actual.

La tubería (|) envía la salida (lista de archivos) a grep.

grep '.txt' filtra la lista, mostrando solo las líneas que contienen la cadena .txt.

```
cat archivo.txt | wc -l # Cuenta el número de líneas en archivo.txt  
#devuelve el mismo resultado que wc -l archivo.txt pero sin incluir nombearchivo
```

Ordenar una lista de archivos:

```
ls | sort # Lista archivos y los ordena alfabéticamente
```

Puedes encadenar varias tuberías para crear flujos de trabajo de procesamiento más complejos.

```
cat archivo.txt | grep palabra | wc -l #muestra contenido, cuenta palabras y filtra 'palabra'
```

```
cat nombreArchivo | grep ".txt" | more #muestra contenido, filtrado por  
extension y pagina
```

### Consejos adicionales

- Usa la redirección (>, >>) para guardar la salida final de la tubería en un archivo.
- Explora las opciones de varios comandos (flags) para refinar aún más el comportamiento de cada comando en la tubería.





## Utilidades Built-in

Aquellas que vienen por defecto con el shell.

Se muestran al ejecutar **help** en la terminal

Para saber si un comando es built-in o que tipo es:

```
type -t comando #Ejemplo : comando == cd, ls, ...
```

<https://linuxhandbook.com/shell-builtin-commands/>



## Rutas absolutas vs. rutas relativas


### 1. Ruta absoluta (desde la raíz /)

/home/luis/Escritorio



Cualquier ruta que comience con una barra inclinada / es una ruta absoluta.

### 2. Ruta relativa (desde el directorio actual)

- **../** Sube un nivel en el directorio
- Ruta relativa: `cd Escritorio/carpeta`
- **./** es tu directorio
-  Las rutas se pueden 'entrecomillar o no'. Notar que cuando se copian del UI aparecen entrecomilladas.

## Ejemplos de uso de rutas/paths

### Rutas absolutas

```
$ cp /home/pedro/origen.txt /home/dev/destino.txt #destino.txt en este caso se puede omitir aunque no es aconsejable
```

### Rutas relativas

```
$ cp origen.txt destino.txt
```

### Rutas relativas

```
$ cp origen.txt ../directorioDestino/destino.txt #destino.txt en este caso se puede omitir.
```

⚠ Si directorioDestino NO existe, y no se le da nombre de archivo o cierra con / crea un archivo en el nivel superior.

```
si@si-vm:~/Escritorio/rutas/ruta1$ mv r1f ./ruta1in/destino.txt  
#copia el archivo r1f al subdirectorio ruta1in
```

⚠ Si el subdirectorio ./ruta1in NO existe....mismo caso que arriba

### Ruta absoluta y ruta relativa

```
si@si-vm:~/Escritorio/rutas/ruta1$ cp  
'/home/si/Escritorio/rutas/ruta2/filer1' ../ruta3  
# copia el archivo de ruta2 usando absoluta a ruta3 subiendo un nivel desde la ruta2
```



## Enlace duros y enlace simbólicos

### Enlaces duros (**hardlink**):

Es una referencia directa a un **archivo** existente (i.e, **no funciona con directorios**). Cuando se crea uno o varios enlaces duros, no se crea una nueva copia de los datos, sino que se incrementa el contador de enlaces del archivo original y **comparten el mismo inodo**. Esto significa que el enlace duro y el archivo original apuntan al mismo contenido en el disco duro. No se pueden crear en diferentes sistemas de archivos. Es creado precediendo por **ln**

```
ln original enlace_duro
```

### Características:

- Permiten acceder al mismo archivo con diferentes nombres.
- No ocupan espacio adicional en el disco duro, ya que solo incrementan el contador de enlaces.
- No se pueden crear en diferentes sistemas de archivos.
- Si se elimina el archivo original, el archivo nuevo no se borra

Ejemplos de uso:

- Crear alias (acceso directo) para archivos de uso frecuente.
- Compartir archivos entre diferentes usuarios sin necesidad de duplicar el contenido.
- Organizar archivos de forma más eficiente.

### Enlaces simbólicos (**symlink**):

También conocido como enlace suave o "soft link", es una referencia a un archivo o directorio existente. A diferencia de los enlaces duros, los enlaces simbólicos no apuntan directamente al contenido del archivo, sino que almacenan la ruta del archivo o directorio al que se hace referencia. Puede apuntar a archivos/directorio en otro sistema de archivos diferentes. Se marca con **l** en el tipo de archivo (ver permisos) y una flecha que apunta al original. Es creado precediendo el archivo/directorio origen y destino por **ln -s**

```
ln -s original simbolico
```

**⚠ Si al crearlo, *original* no existe, crea el link igual apuntando al *original* (" vacío").**

### Características:

- Permiten crear referencias a archivos o directorios ubicados en cualquier parte del sistema de archivos, incluso en diferentes sistemas de archivos.
- Ocupan un pequeño espacio en el disco duro para almacenar la ruta del archivo o directorio al que se hace referencia.
- Se pueden crear en cualquier lugar del sistema de archivos.
- Si se elimina el archivo o directorio original, el enlace simbólico sigue existiendo, pero apunta a un archivo o directorio inexistente.
- Cada enlace simbólico tiene **su propio número de inodo** lo que permite hacer enlaces simbólicos entre distintos sistemas de ficheros.



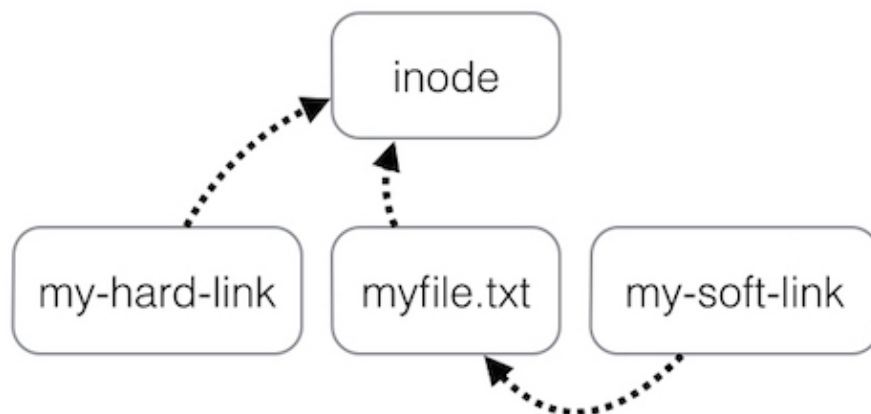
Ejemplos de uso:

- Crear punteros a archivos o directorios que se mueven o cambian de nombre con frecuencia. Ojo porque el enlace puede perder su referencia!
- Vincular bibliotecas compartidas en diferentes aplicaciones.
- Montar sistemas de archivos en diferentes ubicaciones.



Los enlaces duros son útiles cuando el archivo original se mueve. Por ejemplo, mover un archivo de /bin a /usr/bin o a /usr/local/bin. Cualquier enlace simbólico al archivo en /bin se rompería por esto, pero un enlace duro, siendo un enlace directamente al inodo para el archivo, no le importaría.

Característica	Enlaces duros	Enlaces simbólicos
Definición	Referencia directa a un archivo o directorio	Referencia a la ruta de un archivo o directorio
Ocupación de espacio	No ocupan espacio adicional	Ocupan un pequeño espacio para la ruta
Creación	Solo en el mismo sistema de archivos	En cualquier lugar del sistema de archivos
Eliminación del original	Se eliminan todos los enlaces	El enlace sigue existiendo, pero apunta a un archivo inexistente
Ejemplos de uso	Alias, compartir archivos	Punteros a archivos, bibliotecas, montaje de sistemas de archivos



Para ver los permisos y contador hardlinks  
**ls -l**


Para ver los inodos  
**ls -li**

Para ver hardlink  
**find . -inum num\_inodo**

⚠ Sobre compartir enlaces [simbólicos](#) con Vbox con máquinas Windows de Host

## Sudo

### SuperUser do

 Al ejecutado pide la contraseña la primera vez, luego la recuerda por ~15min por defecto

Localización: **/etc/sudoers**

---

#### **sudo** sólo puede ejecutar un comando a la vez

No obstante, ese comando podría a su vez ejecutar muchos otros lo que puede generar **problemas de seguridad** si el comando/script es malicioso

---

- Ejecuta un solo comando con privilegios de root: Te permite ejecutar un comando específico con permisos de root, sin necesidad de cambiar de usuario (ver **su**).
- Mayor seguridad: Registra el comando ejecutado y el usuario que lo realizó, mejorando la trazabilidad y auditoría.
- Configuración granular: Permite configurar qué usuarios y qué comandos pueden ejecutarse con sudo, otorgando un control más preciso sobre los accesos.
- Uso recomendado para tareas puntuales: Ideal para ejecutar comandos administrativos específicos sin necesidad de iniciar una sesión completa como root.

El *superuser* en sistemas antiguos era un usuario real, con un nombre de usuario real (casi siempre "root") con el que podías entrar si tenías la contraseña. Por defecto viene desactivado.

## SU

### Switch User: también se puede especificar un usuario no root

- Cambia al usuario **root** (ie., no solo ejecuta un comando): Te permite iniciar una nueva sesión con todos los privilegios de la cuenta root. **Por defecto deshabilitada.**

```
su nombreUsuario #solicitará la contraseña de dicho usuario.
```

- Acceso completo al sistema: Obtienes control total del sistema, pudiendo realizar cualquier acción administrativa.
- Menos seguro: No registra los comandos ejecutados ni el usuario que los realizó, lo que dificulta la auditoría.
- Menos granular: No permite un control tan preciso sobre qué usuarios y qué comandos pueden acceder a privilegios de root.
- Uso recomendado para tareas administrativas extensas: Útil cuando se necesitan realizar múltiples acciones administrativas o cuando se desea permanecer en un entorno root durante un período prolongado.

 **exit** (para cerrar la sesión root) o volver a otro usuario

## sudo su

Un *truco* con sudo es usarlo para ejecutar el comando **su** (sudo su *comando*). Esto le dará un shell root incluso si la cuenta root está deshabilitada. Puede ser útil cuando necesita ejecutar una serie de comandos como superusuario, para evitar tener que anteponerles sudo a todos, pero abre exactamente el mismo tipo de problemas. Ten en cuenta que **todos los comandos posteriores se ejecutarán como usuario** root. [Fuente](#)

 Más info: <https://help.ubuntu.com/community/RootSudo>

 **sudo Vs su:** [\[1\]](#) [\[2\]](#) [\[3\]](#) **AÑADIR VISUDO**



## Usuarios

Linux y por lo tanto Ubuntu es multiusuario, por lo que más de una persona puede interactuar simultáneamente con el mismo sistema. Cada usuario es identificado por su número **uid** (user identifier), que es **único**. **Al crear un usuario se crea un grupo con el mismo nombre e id que el usuario**. El root tiene UID: 0

Los usuarios se pueden crear desde el entorno gráfico o consola. Para crear un usuario en la terminal se recomienda crearlos con:

```
sudo adduser nombreUser #solicita password y crea el home. Ojo, el escritorio y demás directorios dentro del home NO se crean hasta hacer login.
```

Opción 2 (Más bajo nivel. **No recomendable**)

**sudo useradd [options] nombre\_usuario** #no solicita password al crearlo, debes poner opción password -p o añadirlo luego. Puedes añadir el grupo al crearlo con opciones. Tampoco genera el home para ello debes pasarle la opción -m. Otros problemas como el idioma

Este proceso añade una entrada a los archivos:

- **/etc/passwd**
- **/etc/shadow** (localización de contraseñas. Encriptadas/hashed)
- **/etc/group**
- **/etc/gshadow**

### Cambiar de usuario

```
su nombre_usuario
```

*Solicita la clave del usuario*

Diferencia con: **su - nombre\_usuario** (**sudo su - sin nombre de usuario va al root**)

- Cambia al directorio /home del nombre\_usuario
- ejecuta todos los archivos de inicio de ese usuario cambia el valor de algunas variables del sistema adaptándolas al nuevo usuario (HOME, SHELL, TERM, USER, LOGNAME, etc),

---

**su nombreUsuario**

Permite usar el intérprete de comandos de otro usuario sin necesidad de cerrar la sesión actual.

**exit**

*vuelve al usuario original*

---



## Detalles del usuario

```
id nombre_usuario #mostrará el uid así como grupos y otros
```

⚠ Solo usuario *root* o usuarios con permisos *sudo* pueden crear usuarios ⚠

## Borrar usuarios con su home

```
sudo deluser --remove-home username #distro tipo Debian  
sudo userdel -r username #distros RedHat. si muestra un warning se puede ignorar  
porque es que no ha se habia creado un archivo
```



## Permisos

El **primer** carácter (de la izquierda) es el tipo de archivo/directorio. Sus valores posibles son los siguientes:

- **-**: Representa un archivo regular.
- **d**: Representa un directorio.
- **l**: Representa un enlace simbólico.
- **c**: Representa un dispositivo de carácter.
- **b**: Representa un dispositivo de bloque.
- **p**: Representa un socket Unix.

Los permisos se agrupan se agrupan de 3 caracteres, correspondiendo a los siguientes *tipos de usuario*:

- 1. Propietario:** El usuario que creó el archivo o directorio.
- 2. Grupo:** Los usuarios que pertenecen al mismo grupo que el propietario.
- 3. Otros:** Todos los demás usuarios del sistema.

Para cada uno de ellos, se pueden otorgar tres permisos básicos:

- **Lectura (r)**: Permite ver el contenido de un archivo o listar el contenido de un directorio.
- **Escritura (w)**: Permite modificar el contenido de un archivo o agregar/eliminar archivos en un directorio.
- **Ejecución (x)**: Permite ejecutar un archivo como un programa o acceder a un directorio.
- Cuando no tiene permiso se denota con **-** (*ojo no en el primer dígito de la izquierda que representa un archivo regular*)

Los permisos para cada grupo se concatenan y se muestran como una cadena de tres caracteres. Por ejemplo:

- **rwX**: El propietario tiene permisos de lectura, escritura y ejecución.
- **rw-**: El grupo tiene permisos de lectura y escritura, pero no de ejecución.
- **r--**: Otros usuarios solo tienen permiso de lectura.



Si se intenta acceder a un directorio con un usuario sudoer ("administrador") que no tiene permisos de acceso, **no** se podrá (esto es por diseño y seguridad). Añadir sudo del comando al principio no cambia esto. Para sortearlo, o se realiza desde el usuario root, o se debe añadir el usuario sudoer al grupo que pertenece el directorio (⚠ Requiere reiniciar terminal o hacer un cambio de usuario para que surta efecto). Se podría también cambiar el owner pero no es lo ideal.

---





```
$ ls -l samplefile
```

1	2	3	4	5	6	7
-rw-r--r--	1	pi	pi	5	Nov 29 06:09	samplefile

File Type: -rw-r--r--  
Number of Hard Links: 1  
User Owner: pi  
Group Owner: pi  
File Size in Bytes: 5  
Modification Date: Nov 29 06:09  
File Name: samplefile

## Comandos para administrar permisos

- **chmod**: Cambia los permisos de archivos y directorios. Más [información](#)

 [Calculadora online](#) 

- **chown**: Cambia el propietario de un archivo o directorio. Más [información](#)
- **chgrp**: Cambia el grupo propietario de un archivo o directorio.

### Linux Permissions Made Easy

	user	group	everyone
-	rwx	rwx	rwx
	4 2 1	4 2 1	4 2 1
	1 1 1	1 1 1	1 1 1
	7	7	7

decimal notification: add each number to obtain the value (4 + 2 + 1 = 7)  
binary notification: convert it to decimal then you should have the value (r-x = 101 base 2 = 5 base 10)

## Ejemplos de **Permisos Absolutos** en Octal

- 754: El propietario tiene permisos completos (rwx), el grupo tiene permisos de lectura y ejecución (r-x), y otros usuarios solo tienen permiso de lectura (r--).
- 644: El propietario tiene permisos completos (rw-), el grupo tiene permiso de lectura (r--), y otros usuarios solo tienen permiso de lectura (r--).



## Permisos simbólicos

Access level	Symbol
Read	r
Write	w
Execute	x

Identity	Symbol
User	u
Group	g
Others	o

Task	Operator
Grant a level of access	+
Remove a level of access	-
Set a level of access	=

⚠ El '=' igual es asignación (ie., de derecha a izquierda)

### Ejemplos

```
chmod +x archivo #da permisos de ejecución a todos
chmod -x archivo #quita permisos de ejecución a todos

chmod u+x archivo#da permisos de ejecución al usuario (si ya los tiene
no ocurre nada)
chmod u-x archivo#quita permisos de ejecución al usuario (si no los
tiene no ocurre nada)

chmod ugo+w archivo #da permisos de escritura al user, group y others
chmod o=g archivo #igual a los permisos del user a los que tenga el grupo
```

📖 Más información y ejemplos: <https://www.redhat.com/sysadmin/manage-permissions>

ℹ Nota: Esto aplica tanto a archivos como directorios



## Grupos

Permiten organizar a los usuarios y asignarles permisos de acceso y control sobre archivos, directorios y otros recursos del sistema. Un usuario puede pertenecer a uno o varios grupos, y los permisos de grupo se aplican a todos los miembros del grupo. 🌟 Cuando se crea un usuario, se crea un grupo con el mismo nombre id, pero será el gid. Localización:

- `/etc/group`
- `/etc/gshadow`

### Se pueden crear grupo sin usuario

#### Funcionamiento de los grupos:

- **Identificador de grupo (GID):** Cada grupo tiene un identificador único (GID) que lo representa en el sistema.
- **Permisos de grupo:** Los permisos de grupo se establecen para archivos y directorios. Controlan las acciones que los miembros del grupo pueden realizar sobre ellos. Como se observa en la imagen de la anterior página, son los 3 del "medio" (sin contar el primer carácter de la izquierda de *tipo de archivo*)
- **Miembros del grupo:** Los usuarios se añaden o eliminan de un grupo utilizando comandos específicos. Los miembros del grupo **heredan** los permisos de grupo para los archivos y directorios a los que tienen acceso.

#### Creación de grupos:

Para crear un nuevo grupo, se utiliza el comando **groupadd**. La sintaxis básica es la siguiente:

```
sudo groupadd nombre_grupo
```

Donde *nombre\_grupo* es el nombre que se desea asignar al nuevo grupo. Por ejemplo, para crear un grupo llamado desarrolladores:

```
sudo groupadd desarrolladores
```

#### Ver grupos existentes:

Para ver la lista de grupos existentes en el sistema, se utiliza el comando `groups`. Este comando muestra los grupos a los que pertenece **el usuario actual**.

```
groups REVISE!! Falta el último?
```

```
groups nombreusuario#ver grupos pertenece un usuario en particular
```



### Añadir usuarios a un grupo:

Para añadir un usuario a un grupo existente, se utiliza el comando `usermod`. La sintaxis básica es la siguiente:

```
sudo usermod -aG nombre_grupo nombre_usuario
```

Donde `nombre_grupo` es el nombre del grupo al que se desea añadir el usuario y `nombre_usuario` es el nombre del usuario que se desea añadir. Por ejemplo, para añadir el usuario `pepe` al grupo `desarrolladores`:

```
sudo usermod -aG desarrolladores pepe
```

### Eliminar usuarios de un grupo:

Para eliminar un usuario de un grupo, se utiliza el comando `usermod`. La sintaxis básica es la siguiente:

```
gpasswd --delete nombre_usuario grupo
```

```
usermod -G "" username #para borrar de todos los grupos (menos el suyo propio)
```


Donde `nombre_grupo` es el nombre del grupo del que se desea eliminar el usuario y `nombre_usuario` es el nombre del usuario que se desea eliminar. Por ejemplo, para eliminar el usuario `pepe` del grupo `desarrolladores`:

```
sudo usermod -G desarrolladores pepe
```

### Eliminar un grupo:

Para eliminar un grupo existente, se utiliza el comando `groupdel`. La sintaxis básica es la siguiente:

```
sudo groupdel nombre_grupo
```

 **Importante:** No se debe eliminar un grupo si hay usuarios que todavía pertenecen a él.



## Instalación de paquetes/aplicaciones

Para saber paquetes instalados

```
dpkg --get-selections
```

O también:

```
dpkg --get-installed
```

### Instalación

- apt
- apt-get
- snap

### Desinstalación de paquetes:

```
sudo apt remove package_name
```

```
sudo apt purge nombre_paquete
```

```
sudo apt-get purge nombre_paquete
```

```
sudo apt -purge remove package_name
```



## Código de colores consola

Por defecto la consola muestra los diferentes archivos y directorios con un colores diferentes o subrayados según sea su tipo. Abajo el script para imprimirlo

 <https://askubuntu.com/questions/17299/what-do-the-different-colors-mean-in-ls>

```
#!/bin/bash
# For each entry in LS_COLORS, print the type, and description if available,
# in the relevant color.
# If two adjacent colors are the same, keep them on one line.

declare -A descriptions=(
  [bd]="block device"
  [ca]="file with capability"
  [cd]="character device"
  [di]="directory"
  [do]="door"
  [ex]="executable file"
  [fi]="regular file"
  [ln]="symbolic link"
  [mh]="multi-hardlink"
  [mi]="missing file"
  [no]="normal non-filename text"
  [or]="orphan symlink"
  [ow]="other-writable directory"
  [pi]="named pipe, AKA FIFO"
  [rs]="reset to no color"
  [sg]="set-group-ID"
  [so]="socket"
  [st]="sticky directory"
  [su]="set-user-ID"
  [tw]="sticky and other-writable directory"
)

IFS=:
for ls_color in $LS_COLORS; do
  color="${ls_color#*=}"
  type="${ls_color%*=}"

  # Add description for named types.
  desc="${descriptions[$type]}"

  # Separate each color with a newline.
  if [[ $color_prev ]] && [[ $color != "$color_prev" ]]; then
    echo
  fi

  printf "\e[%sm%s%s\e[m " "$color" "$type" "${desc:+ ($desc)}"

  # For next loop
  color_prev="$color"
done
echo
```


*Script para ver el código de colores extraído del link de arriba.*



## TTY y terminales virtuales

TTY en Linux significa "TeleTYpewriter" (Teletipo) y se refiere a un dispositivo que permite la comunicación texto a texto. En el contexto de Linux, un TTY normalmente representa una terminal virtual, que es una sesión de línea de comandos (CLI).

**Terminales virtuales:** Linux puede tener múltiples terminales virtuales activas simultáneamente. Cada terminal virtual se asigna a un TTY específico, como tty1, tty2, etc. Puedes cambiar entre estas terminales virtuales usando la combinación de teclas Ctrl + Alt + F seguido del número de la terminal (por ejemplo, Ctrl + Alt + F2).

 Número de TTY cambian según número de usuarios con terminales abiertas en diferentes sesiones

 **who**

No se limita a la sesión abierta (al contrario que whoami)

 **w**

Facilita más información

Compruébalo abriendo varias terminales. ¿Cambia el TTY?

Si lo logueas con diferentes usuarios en shell e interfaz gráfico. ¿Cambia el TTY?

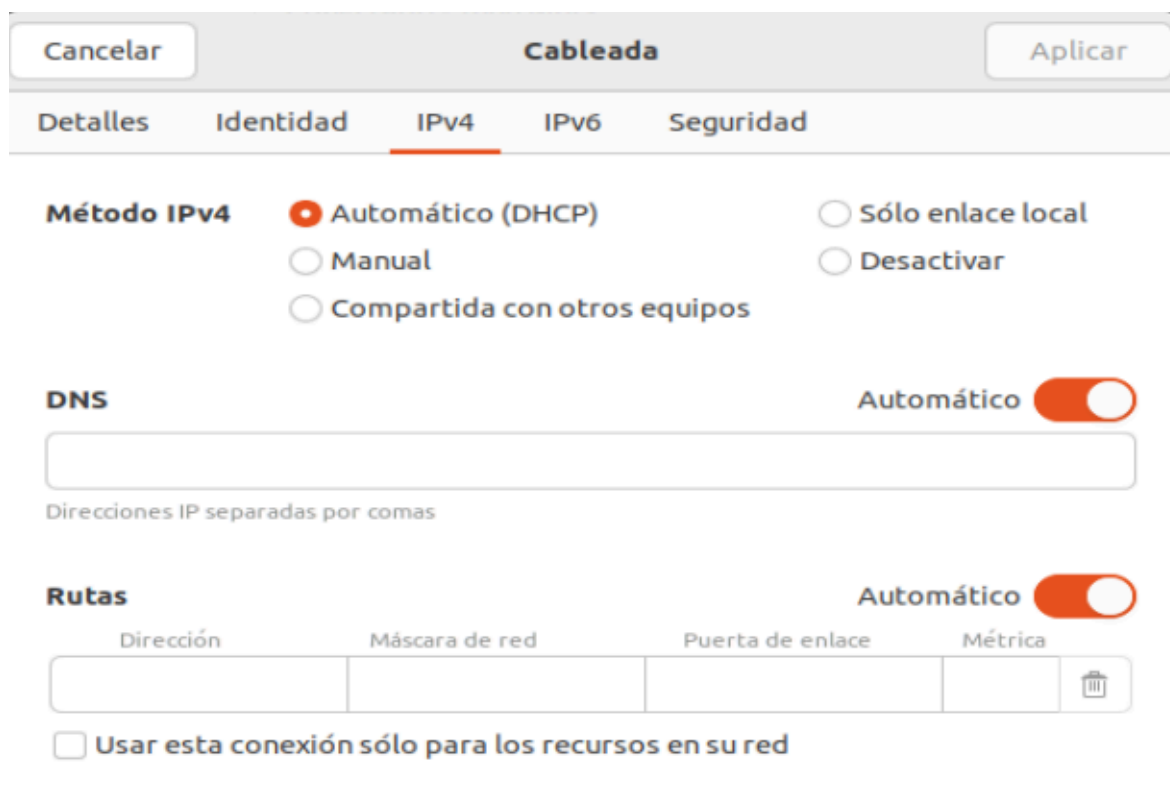
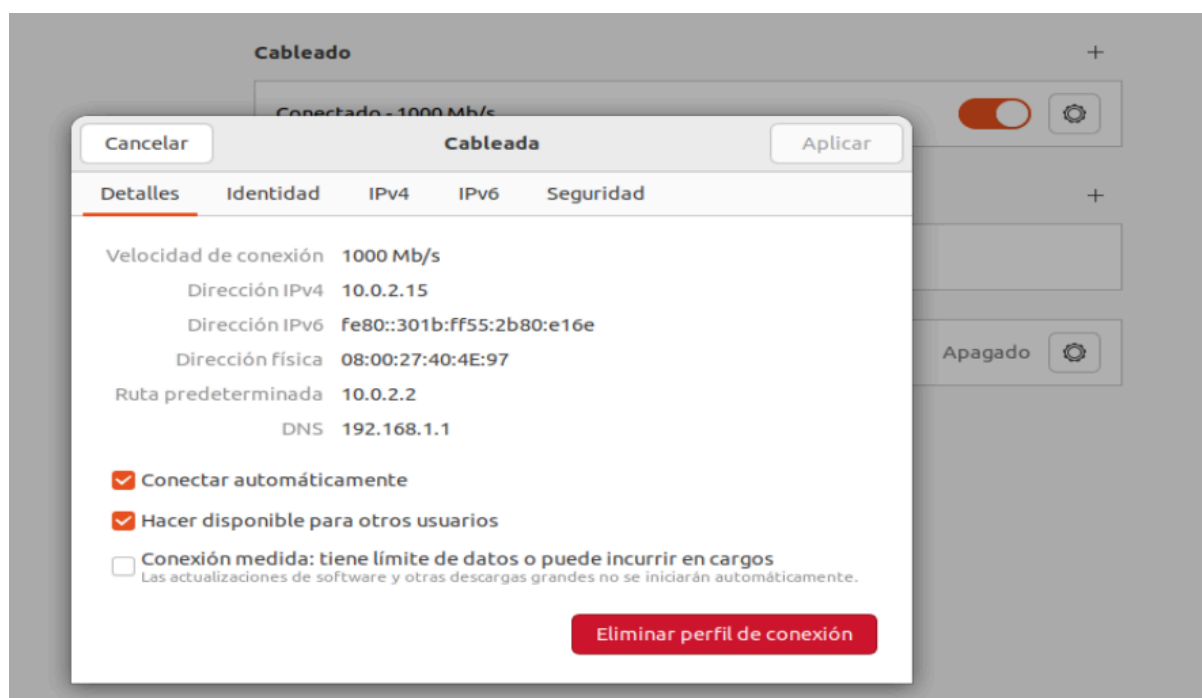
 Comprueba abriendo diferentes cmd/powershell. ¿Ocurre lo mismo?



# Networking

## Configuración networking en interface gráfico

- Red







## Networking en terminal

- net-tools → **Deprecated**


Aún se pueden utilizar pero requieren instalación y **no es recomendable** por no estar mantenidos.

Comando shell

Deprecated	Reemplazo
arp	ip n (ip neighbor)
ifconfig	ip a (ip addr), ip link ip -s (ip -stats)
iptunnel	ip tunnel
iwconfig	iw
netstat	ss ip route ( netstat -r) ip -s link (netstat -i) ip maddr (netstat -g)
route	ip r (ip route)

<https://www.cyberciti.biz/faq/linux-ip-command-examples-usage-syntax/>

<https://itsubuntu.com/50-useful-linux-networking-commands-and-tools/> (Ojo: algunos han sido *deprecated*)

- **Host table** /etc/hosts
- Cómo renovar (flush) la IP
- Remote desktop cliente: **Remmina** → VNC y SSH (RDP Windows PRO)
- Compartir en red
  - SSH
    - [Instalación y comprobación \(caso 1 y 3\)](#)
    - [Transferencia](#)
      - scp (ssh)
      - SFTP
      - SSFS
      - rsync (ssh)
  - HTTP Server:  **python3 -m http.server** (por defecto solo permite GET).
  - [NFS](#)
  - [Samba \[2\]](#)
  - iPerf3 (Herramienta para medir ancho de banda max. Requiere instalar). Uso:
    - Server: iperf3 -s
    - Cliente: iperf3 -c si-vm



# Scripting

## Control de flujo

### Operadores condicionales

- if
- else
- elif
- case

### Operadores de bucle:

- **for**: Bucle iterativo para ejecutar un bloque de código un número determinado de veces o para recorrer elementos de una lista.
- **while**: Bucle condicional que ejecuta un bloque de código mientras una condición sea verdadera.
- **until**: Bucle condicional que ejecuta un bloque de código hasta que una condición sea verdadera.

Para que un script se **ejecute** debe tener los **permisos** necesarios

```
chmod +x nombreScript.sh
```

Esto se debe realizar para todos los scripts previo a su ejecución

```
#!/bin/bash # Que interprete tiene que correr el script aka shebang  
  
nombreVariable="Hola"  
echo '$nombreVariable' # Salida: $nombreVariable  
echo "$nombreVariable" # Salida: Hola
```

### Para ejecutarlo

```
./nombreScript
```

*asumiendo que está en su misma ruta, sino podríamos usar rutas relativas o absolutas*



Un script bash puede terminarse pulsando CTRL + C  
sin esperar a que termine su operación.

---



## Diferencia entre comillas simple y dobles

- Comillas simples: Todo lo que va entre comillas simples se trata como una cadena literal. Los nombres de variables y la mayoría de los caracteres especiales no se expanden.
- Comillas dobles: Las variables y ciertos caracteres especiales entre comillas dobles se expanden. El contenido está sujeto a sustitución de variables y sustitución de comandos.

<https://www.freecodecamp.org/news/shell-scripting-crash-course-how-to-write-bash-scripts-in-linux/>

<https://www.geeksforgeeks.org/shell-script-examples/>

<https://www.howtogeek.com/808593/bash-script-examples/>

<https://www.hostinger.com/tutorials/bash-script-example>



[Cheetsheet](#)



## WSL

WSL (Windows Subsystem for Linux) es una capa de virtualización de Linux que permite ejecutar algunas distribuciones de Linux sin la necesidad de instalar una máquina virtual o dual boot. En el proceso de instalación se debe elegir la distribución y versión como se verá después (aunque viene una versión por defecto). Por ello requiere de muchos menos recursos.

### Instalación

En **Powershell** o cmd (correr como admin)

```
wsl --install
```

<https://terminaldelinux.com/terminal/wsl/instalacion-wsl/>

<https://es.linkedin.com/pulse/c%C3%B3mo-instalar-wsl2-en-windows-10-subsystem-linux-versi%C3%B3n-2-employit>

<https://learn.microsoft.com/es-es/windows/wsl/install>

🔥 Cuando termine la instalación tras reiniciar te pedirá que crees el usuario y password.

---

💻 Para ejecutarlo (ambas funcionan)

- Buscar "ubuntu" (o la distribución instalada) en el buscador de windows
  - Ejecutar **wsl** en cmd/powershell
- 

### Ruta

- Para interacción con archivos Linux files directamente desde Windows:  
\\wsl\$\nombreDistribucion
- Para acceso network-related para la distribución de WSL desde Windows.  
\\wsl.localhost\nombreDistribucion

### Instalar GUI software para WSL2

<https://learn.microsoft.com/en-us/windows/wsl/tutorials/gui-apps>

[https://techviewleo.com/run-linux-gui-applications-on-windows/?utm\\_content=cmp-true](https://techviewleo.com/run-linux-gui-applications-on-windows/?utm_content=cmp-true)


### WSL1 vs. WSL2

La diferencia principal entre WSL 1 y WSL 2 es que este último opera dentro de una máquina virtual administrada (VM).



Por lo general, Microsoft recomienda utilizar WSL 2 debido a su rendimiento superior en comparación con WSL 1 y a su compatibilidad al 100% con las llamadas al sistema. No obstante, se puede utilizar WSL 1 si un proyecto debe almacenarse en el sistema de archivos de Windows o si se requiere la compilación cruzada (cross-compilation) con herramientas de Windows y Linux para un proyecto.

 <https://learn.microsoft.com/en-us/windows/wsl/compare-versions>

 <https://tuxcare.com/blog/what-is-windows-subsystem-for-linux/#:~:text=WSL%20has%20some%20limitations%2C%20however,file%20systems%2C%20such%20as%20ext4.>

Se puede ejecutar desde

- cmd
- powershell

*En estos dos casos observa la ruta desde donde arranca que es en la que te encuentres en ese momento.*

- buscando ubuntu
- buscando wsl

Notar la diferencia de icono y en el administrador de tareas, así como la ruta desde la que apunta la consola

## WSL Vs Cygwin

[https://www.reddit.com/r/linux/comments/10x8mcw/is\\_there\\_any\\_advantage\\_to\\_wsl\\_over\\_cygwin](https://www.reddit.com/r/linux/comments/10x8mcw/is_there_any_advantage_to_wsl_over_cygwin)

## Cygwin y MiYGwin

- Compilar aplicaciones nativas de Windows.



## Otros

### Instalar guest additions en VirtualBox

```
sudo apt-get install virtualbox-guest-utils
```

O ejecutar el .sh desde el CD ROM Virtual y luego lanzar el script de arriba

Si no funcionase ejecutar el siguiente comando:

```
VBoxClient --version #en caso que diga que no está instalado ejecutar el comando que muestra
```

Shared folder windows [Gedit issue](#)

### Libro Linux

<https://archive.org/details/la-linea-de-comandos-de-linux/page/n1/mode/1up>

[https://help.ubuntu.com/community/UsingTheTerminal#File\\_.26\\_Directory\\_Commands](https://help.ubuntu.com/community/UsingTheTerminal#File_.26_Directory_Commands)

### COMANDO LIBRARY

<https://ubunlog.com/linux-command-library/amp/>

## Tutoriales y juegos aprendizaje Bash

### Tutoriales

- <https://linuxjourney.com/>
- <https://linuxsurvival.com/>

### Juegos

- [command line heroes bash](#)
- <https://overthewire.org/wargames/>
- <https://trybash.github.io/game/>

*problema resolución?*

## Localización Papelera

- trash:/// en el Interface gráfico, cuya ruta es:

```
~/.local/share/Trash$
```





# Glosario de términos

apt

apt-get

Boot loader: GRUB o ISOLINUX

Cron

Daemon (Demonio)

GRUB: cargador de arranque

File system (sistema de archivos): método de guardado y organización de archivos

Inodo

Make → Makefile

[Número Mágico](#) (no confundir con el mismo término usado en programación)

Packet Manager

root /directorio

root ("administrador")

Servicio: Programa que se ejecuta como proceso en el background

Shell: bash, tcsh, zsh