

## COMANDOS BÁSICOS LINUX

### 1. Comando `ls`

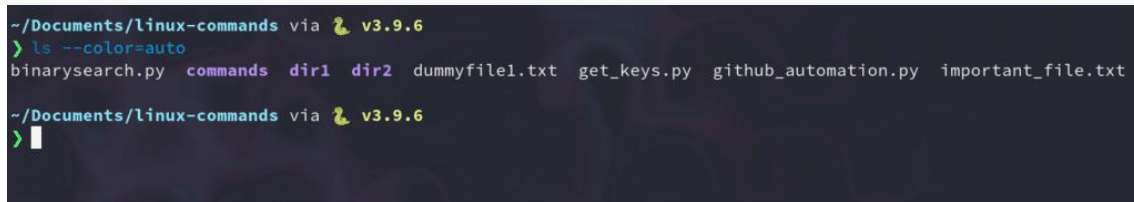
`ls` es probablemente el primer comando que todo usuario de Linux teclea en su terminal. Te permite listar el contenido del directorio que quieras (el directorio actual por defecto), incluyendo archivos y otros directorios anidados.

```
ls
```

Tiene muchas opciones, así que puede ser bueno obtener algo de ayuda usando el flag `--help`. Este flag devuelve todas los flag que puedes utilizar con `ls`.

Por ejemplo, para colorear la salida del comando `ls`, puede utilizar lo siguiente:

```
ls --color=auto
```



```
~/Documents/linux-commands via v3.9.6
> ls --color=auto
binarysearch.py  commands  dir1  dir2  dummyfile1.txt  get_keys.py  github_automation.py  important_file.txt
~/Documents/linux-commands via v3.9.6
> |
```

El comando `ls` coloreado.

Ahora la salida del comando `ls` está coloreada, y se puede apreciar la diferencia entre un directorio y un archivo.

Pero escribir `ls` con el flag de color sería ineficiente; por eso usamos el comando `alias`.

### 2. Comando `alias`

El comando `alias` te permite definir alias temporales en tu sesión de shell. Al crear un alias, se indica al shell que sustituya una palabra por una serie de comandos.

Por ejemplo, para que `ls` tenga color sin tener que teclear el flag `--color` cada vez, se usaría:

```
alias ls="ls --color=auto"
```

Como puedes ver, el comando `alias` toma un par de parámetros clave-valor: `alias NAME="VALUE"`. Ten en cuenta que el valor debe estar entre comillas.

Si quieres listar todos los alias que tienes en tu sesión de shell, puedes ejecutar el comando `alias` sin argumento.

```
alias
```

```
~/Documents/linux-commands via v3.9.6
> alias
alias .. 'cd ..'
alias ... 'cd ../../'
alias 3.. 'cd ../../..'
alias awesome_server 'Xephyr -br -ac -noreset -screen 1300x730 :1 & DISPLAY=:1 awesome ~/.config/awesome/rc.lua'
alias cdC 'cd ~/.config/'
alias cdM 'cd ~/MEGAsync/'
alias cdMG 'cd ~/MEGAsync/github'
alias cl clear
alias config '/usr/bin/git --git-dir=/home/daniel/dotfiles/ --work-tree=/home/daniel'
alias config-a 'config add'
alias config-m 'config commit -m'
alias config-p 'config push origin'
alias config-s 'config status'
alias cp 'cp -r'
alias em '/usr/bin/emacs -nw'
alias emacs emacsclient\\ -c\\ -a\\ \\'emacs\\'
alias fish_key_reader /usr/bin/fish_key_reader
alias g git
alias gc 'git clone'
```

El comando `alias`.

### 3. Comand `unalias`

Como su nombre indica, el comando `unalias` tiene como objetivo eliminar un `alias` de los ya definidos. Para eliminar el alias `ls` anterior, puedes utilizar:

```
unalias ls
```

#### 4. Comando `pwd`

El comando `pwd` significa «imprimir el directorio de trabajo» y muestra la ruta absoluta del directorio en el que se encuentra. Por ejemplo, si tu nombre de usuario es «john» y está en tu directorio Documentos, tu ruta absoluta sería `/home/john/Documents`.

Para utilizarlo, basta con escribir `pwd` en el terminal:

```
pwd
```

```
# My result: /home/kinsta/Documents/linux-commands
```

#### 5. Comando `cd`

El comando `cd` es muy popular, junto con `ls`. Se refiere a «cambiar de directorio» y, como su nombre indica, te cambia al directorio al que intentas acceder.

Por ejemplo, si estás dentro del directorio Documentos y tratas de acceder a una de sus subcarpetas llamada **Videos**, puedes entrar en ella escribiendo:

```
cd Videos
```

También puedes proporcionar la ruta absoluta de la carpeta:

```
cd /home/kinsta/Documents/Videos
```

Hay algunos trucos con el comando `cd` que pueden ahorrarte mucho tiempo al jugar con él:

### 1. Ir a la carpeta de inicio

```
cd
```

### 2. Sube un nivel

```
cd ..
```

### 3. Volver al directorio anterior

```
cd -
```

### 6. Comando `cp`

Es tan fácil copiar archivos y carpetas directamente en el terminal de Linux que a veces puede sustituir a los gestores de archivos convencionales.

Para utilizar el comando `cp`, basta con escribirlo junto con los archivos de origen y destino:

```
cp file_to_copy.txt new_file.txt
```

También puede copiar directorios enteros utilizando el flag recursiva:

```
cp -r dir_to_copy/ new_copy_dir/
```

Recuerda que en Linux, las carpetas terminan con una barra diagonal (/).

## 7. Comando `rm`

Ahora que sabes cómo copiar archivos, te será útil saber cómo eliminarlos.

Puedes utilizar el comando `rm` para eliminar archivos y directorios. Sin embargo, ten cuidado al usarlo, porque es muy difícil (aunque no imposible) recuperar los archivos eliminados de esta manera.

Para borrar un archivo normal, escribirías:

```
rm file_to_copy.txt
```

Si deseas eliminar un directorio vacío, puedes utilizar el flag recursiva (`-r`):

```
rm -r dir_to_remove/
```

Por otro lado, para eliminar un directorio con contenido en tu interior, es necesario utilizar el flag force (`-f`) y recursive:

```
rm -rf dir_with_content_to_remove/
```

## Info

Ten cuidado con esto: ¡puedes borrar todo un día de trabajo si usas mal estas dos flags!

## 8. Comando `mv`

El comando `mv` se utiliza para mover (o renombrar) archivos y directorios en el sistema de archivos.

Para utilizar este comando, debes escribir tu nombre con los archivos de origen y destino:

```
mv source_file destination_folder/
```

```
mv command_list.txt commands/
```

Para utilizar las rutas absolutas, utilizarías:

```
mv /home/kinsta/BestMoviesOfAllTime ./
```

...donde `./` es el directorio en el que te encuentras.

También puedes usar `mv` para renombrar archivos mientras los mantienes en el mismo directorio:

```
mv old_file.txt new_named_file.txt
```

## 9. Comando `mkdir`

Para crear carpetas en el shell, se utiliza el comando `mkdir`. Solo tienes que especificar el nombre de la nueva carpeta, asegurarte de que no existe y ya está.

Por ejemplo, para crear un directorio para guardar todas tus imágenes, solo tienes que escribir:

```
mkdir images/
```

Para crear subdirectorios con un simple comando, utiliza el flag padre (`-p`):

```
mkdir -p movies/2004/
```

## 10. Comando `man`

Otro comando esencial de Linux es `man`. Muestra la página del manual de cualquier otro comando (siempre que tenga uno).

Para ver la página del manual del comando `mkdir`, escribe:

```
man mkdir
```

Incluso puedes consultar la página del manual `man`:

```
man man
```

```
MAN(1)                                Manual pager utils                                MAN(1)

NAME
    man - an interface to the system reference manuals

SYNOPSIS
    man [man options] [[section] page ...] ...
    man -k [apropos options] regexp ...
    man -K [man options] [section] term ...
    man -f [whatis options] page ...
    man -l [man options] file ...
    man -w|-W [man options] page ...

DESCRIPTION
    man is the system's manual pager. Each page argument
    given to man is normally the name of a program, utility
    or function. The manual page associated with each of
    these arguments is then found and displayed. A section,
    if provided, will direct man to look only in that sec-
tion of the manual. The default action is to search in
    all of the available sections following a pre-defined
    order (see DEFAULTS), and to show only the first page
    found, even if page exists in several sections.

    The table below shows the section numbers of the manual
    followed by the types of pages they contain.

    1 Executable programs or shell commands
    2 System calls (functions provided by the kernel)
    3 Library calls (functions within program libraries)
    4 Special files (usually found in /dev)
    5 File formats and conventions, e.g. /etc/passwd
    6 Games

Manual page man(1) line 1 (press h for help or q to quit)
```


La página de manual de «man».

## 11. Comando `touch`

El comando `touch` permite actualizar los tiempos de acceso y modificación de los archivos especificados.

Por ejemplo, tengo un archivo antiguo que fue modificado por última vez el 12 de abril:



```
~/Documents/linux-commands via  v3.9.6
> ls -lah
Permissions Size User Date Modified Name
drwxr-xr-x - daniel 8 ago 15:11 .
drwxr-xr-x - daniel 8 ago 00:27 ..
drwxr-xr-x - daniel 8 ago 00:34 commands
drwxr-xr-x - daniel 7 ago 00:45 dir1
drwxr-xr-x - daniel 7 ago 00:45 dir2
drwxr-xr-x - daniel 8 ago 00:10 dir_to_copy
drwxr-xr-x - daniel 8 ago 00:12 new_dir
-rw-r--r-- 0 daniel 8 ago 00:38 BestMoviesOfAllTime
-rw-r--r-- 0 daniel 7 ago 00:44 binarysearch.py
-rw-r--r-- 0 daniel 7 ago 00:43 dummyfile1.txt
-rw-r--r-- 0 daniel 8 ago 00:18 file_to_delete.txt
-rw-r--r-- 0 daniel 7 ago 00:44 get_keys.py
-rw-r--r-- 0 daniel 7 ago 00:44 github_automation.py
-rw-r--r-- 0 daniel 7 ago 00:44 important_file.txt
-rw-r--r-- 0 daniel 8 ago 00:04 new_file.txt
-rw-r--r-- 0 daniel 12 abr 20:45 old_file
```

Fecha antigua.

Para cambiar su fecha de modificación a la hora actual, necesitamos utilizar el flag `-m`:

```
touch -m old_file
```

Ahora la fecha coincide con la de hoy (que en el momento de escribir este artículo era el 8 de agosto).

```
~/Documents/linux-commands via v3.9.6
> ls -lah
Permissions Size User Date Modified Name
drwxr-xr-x - daniel 8 ago 15:11 .
drwxr-xr-x - daniel 8 ago 00:27 ..
drwxr-xr-x - daniel 8 ago 00:34 commands
drwxr-xr-x - daniel 7 ago 00:45 dir1
drwxr-xr-x - daniel 7 ago 00:45 dir2
drwxr-xr-x - daniel 8 ago 00:10 dir_to_copy
drwxr-xr-x - daniel 8 ago 00:12 new_dir
-rw-r--r-- 0 daniel 8 ago 00:38 BestMoviesOfAllTime
-rw-r--r-- 0 daniel 7 ago 00:44 binarysearch.py
-rw-r--r-- 0 daniel 7 ago 00:43 dummyfile1.txt
-rw-r--r-- 0 daniel 8 ago 00:18 file_to_delete.txt
-rw-r--r-- 0 daniel 7 ago 00:44 get_keys.py
-rw-r--r-- 0 daniel 7 ago 00:44 github_automation.py
-rw-r--r-- 0 daniel 7 ago 00:44 important_file.txt
-rw-r--r-- 0 daniel 8 ago 00:04 new_file.txt
-rw-r--r-- 0 daniel 8 ago 16:30 old_file
```

Nueva fecha

No obstante, la mayoría de las veces no se utilizará `touch` para modificar las fechas de los archivos, sino para crear nuevos archivos vacíos:

```
touch new_file_name
```

## 12. Comando `chmod`

El comando `chmod` te permite cambiar el modo de un archivo (permisos) rápidamente. Tienes un montón de opciones disponibles con él.

Los permisos básicos que puede tener un archivo son:

- r (leer)
- w (escribir)
- x (ejecutar)

Uno de los casos más comunes de uso de `chmod` es hacer que un archivo sea ejecutable por el usuario. Para ello, escriba `chmod` y el flag `+x`, seguido del archivo en el que desea modificar los permisos:

```
chmod +x script
```

Se utiliza para hacer ejecutables los scripts, lo que permite ejecutarlos directamente utilizando la notación `./`.

### 13. Comando `./`

Tal vez la notación `./` no sea un comando en sí mismo, pero vale la pena mencionarlo en esta lista. Permite a tu shell ejecutar un archivo ejecutable con cualquier intérprete instalado en tu sistema directamente desde el terminal. Se acabó el hacer doble clic en un archivo en un gestor de archivos gráfico!

Por ejemplo, con este comando puedes ejecutar un script de Python o un programa solo disponible en formato `.run`, como XAMPP. Cuando ejecutes un ejecutable, asegúrate de que tiene permisos de ejecución (x), que puedes modificar con el comando `chmod`.

Aquí tenemos un sencillo script de Python y cómo lo ejecutaríamos con la notación `./`:

```
#!/usr/bin/python3

# filename: script

for i in range(20):

    print(f"This is a cool script {i}")
```

Así es como convertiríamos el script en un ejecutable y lo ejecutaríamos:

```
chmod +x script
```

```
./script
```

## 14. Comando `exit`

El comando `exit` hace exactamente lo que tu nombre sugiere: Con él, puedes terminar una sesión de shell y, en la mayoría de los casos, cerrar automáticamente el terminal que estás utilizando:

```
exit
```

## 15. Comando `sudo`

Este comando significa «superuser do», y te permite actuar como superusuario o usuario root mientras ejecutas un comando específico. Es la forma en que Linux se protege a sí mismo y evita que los usuarios modifiquen accidentalmente el sistema de archivos de la máquina o instalen paquetes inapropiados.

Sudo se utiliza comúnmente para instalar software o para editar archivos fuera del directorio personal del usuario:

```
sudo apt install gimp
```

```
sudo cd /root/
```

Te pedirá la contraseña de administrador antes de ejecutar el comando que hayas escrito después.

## 16. Comando `shutdown`

Como puedes adivinar, el comando `shutdown` te permite apagar tu máquina. Sin embargo, también puede utilizarse para detenerla y reiniciarla.

Para apagar el ordenador inmediatamente (el valor predeterminado es un minuto), escriba:

```
shutdown now
```

También puedes programar el apagado de tu sistema en un formato de 24 horas:

```
shutdown 20:40
```

Para cancelar una llamada de `shutdown` anterior, puedes utilizar el flag `-c`:

```
shutdown -c
```

## 17. Comando `htop`

`htop` es un visor de procesos interactivo que te permite gestionar los recursos de tu máquina directamente desde el terminal. En la mayoría de los casos, no está instalado de forma predeterminada, así que asegúrate de leer más sobre él en su página de descargas.

```
htop
```

```

0[| 1.3% 3[| 1.3% 6[| 2.4% 9[| 4.4%
1[| 2.0% 4[| 2.0% 7[| 2.4% 10[| 3.3%
2[| 5.7% 8[| 4.7% 8[| 6.1% 11[| 7.4%
Mem[|||||] 5.77G/15.0G Tasks: 114, 760 thr: 2 running
Swap[|||||] 8K/7.81G Load average: 1.73 1.33 1.19
Uptime: 14:33:50

PID USER PRI NI VIRT RES SHR %CPU%MEM TIME+ Command
12648 daniel 20 0 41.8G 1275M 108M S 10.5 8.0 1h34:18 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
1123 daniel 20 0 398M 78540 21976 S 2.0 0.5 17:04:77 /usr/lib/python /usr/bin/qtile start
726 root 20 0 6283M 75536 36920 S 1.3 0.5 1h37:14 /usr/lib/Xorg :0 -seat seat0 -auth /run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
1239 daniel 9 -11 1001M 14408 9940 S 1.3 0.1 19:49:15 /usr/bin/pulseaudio --daemonize=no --log-target=journal
12657 daniel 20 0 41.8G 1275M 108M S 1.3 8.0 6:56:41 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
388163 daniel 20 0 529M 86128 40580 S 1.3 0.5 0:52:86 alacritty
1199 daniel 20 0 398M 78540 21976 S 0.7 0.5 0:27:02 /usr/bin/python /usr/bin/qtile start
4051 daniel 20 0 1856M 556M 172M S 0.7 3.5 16:47:11 /usr/lib/brave/brave
4096 daniel 20 0 1339M 437M 131M S 0.7 2.7 1h05:42 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
4101 daniel 20 0 398M 110M 65656 S 0.7 0.7 11:26:87 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
4134 daniel 20 0 1339M 437M 131M S 0.7 2.7 17:41:35 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
4383 daniel 20 0 37.3G 150M 90520 S 0.7 0.9 0:39:22 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
12651 daniel 20 0 41.8G 1275M 108M S 0.7 8.0 1:42:30 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
386997 daniel 20 0 37.3G 192M 94428 S 0.7 1.2 5:14:14 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
387087 daniel 20 0 37.3G 192M 94428 S 0.7 1.2 3:09:35 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
387210 daniel 20 0 37.4G 162M 98056 S 0.7 1.0 1:55:13 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
908280 daniel 20 0 37.6G 424M 307M S 0.7 2.7 1:06:45 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
962613 daniel 20 0 41.5G 211M 103M S 0.7 1.3 0:17:04 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
963864 daniel 20 0 343M 58608 32556 S 0.7 0.4 0:14:15 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
1021985 daniel 20 0 37.3G 127M 97020 S 0.7 0.8 0:00:73 /usr/lib/brave/brave --type=renderer --field-trial-handle=1383049695608654963,16601138126615002718,131072
1023803 daniel 20 0 13864 7092 3500 R 0.7 0.0 0:00:41 htop
1 root 20 0 167M 11084 8228 S 0.0 0.1 0:00:00 /sbin/init
390 root 20 0 8164 4784 1648 S 0.0 0.0 0:03:77 /usr/bin/haveged -w 1024 -v 1 --foreground
400 root 20 0 59944 25860 24660 S 0.0 0.2 0:00:30 /usr/lib/systemd/systemd-journald
401 root 20 0 31828 9720 6776 S 0.0 0.1 0:00:36 /usr/lib/systemd/systemd-udev
590 avahi 20 0 12720 5308 4620 S 0.0 0.0 0:00:08 avahi-daemon: running [danielmanjaro.local]
591 root 20 0 9200 2692 2344 S 0.0 0.0 0:00:02 /usr/bin/cron -n
592 dbus 20 0 13840 6720 5016 S 0.0 0.0 0:00:46 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
593 root 20 0 476M 19968 16980 S 0.0 0.1 0:00:06 /usr/bin/NetworkManager --no-daemon
595 polkitd 20 0 2914M 23612 18368 S 0.0 0.1 0:00:12 /usr/lib/polkit-1/polkitd --no-debug
601 root 20 0 175M 8184 6980 S 0.0 0.0 0:00:07 /usr/lib/systemd/systemd-logind
602 root 20 0 14716 6828 6032 S 0.0 0.0 0:00:07 /usr/lib/systemd/systemd-machined
613 avahi 20 0 12448 688 0 S 0.0 0.0 0:00:00 avahi-daemon: chroot helper
701 polkitd 20 0 2914M 23612 18368 S 0.0 0.1 0:00:00 /usr/lib/polkit-1/polkitd --no-debug

```

La interfaz «htop».

## 18. Comando `unzip`

El comando `unzip` permite extraer el contenido de un archivo `.zip` desde el terminal. Una vez más, este paquete puede no estar instalado por defecto, así que asegúrate de instalarlo con tu gestor de paquetes.

Aquí, estamos descomprimiendo un archivo `.zip` lleno de imágenes:

```
unzip images.zip
```

## 19. Comandos `apt`, `yum`, `pacman`

Independientemente de la distribución de Linux que utilices, es probable que uses gestores de paquetes para instalar, actualizar y eliminar el software que utilizas a diario.

Puedes acceder a estos gestores de paquetes a través de la línea de comandos, y utilizarás uno u otro dependiendo de la distro que ejecute tu máquina.

Los siguientes ejemplos instalarán GIMP, un software libre y de código abierto que suele estar disponible en la mayoría de los gestores de paquetes:

## 1. Basado en Debian (Ubuntu, Linux Mint)

```
sudo apt install gimp
```

## 2. Basado en Red Hat (Fedora, CentOS)

```
sudo yum install gimp
```

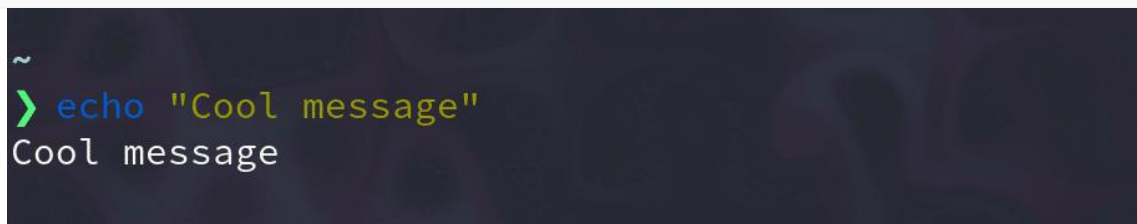
## 3. Basado en Arch (Manjaro, Arco Linux)

```
sudo pacman -S gimp
```

## 20. Comando `echo`

El comando `echo` muestra el texto definido en el terminal – es así de simple:

```
echo "Cool message"
```

A terminal window with a dark background. The prompt is a tilde (~). The command entered is 'echo "Cool message"' with 'echo' in blue and the string in green. The output 'Cool message' is shown below the command.

```
~  
> echo "Cool message"  
Cool message
```

El comando eco

Su uso principal es imprimir las variables de entorno dentro de esos mensajes:

```
echo "Hey $USER"
```



```
# Hey kinsta
```

## 21. Comando `cat`

`Cat`, abreviatura de «concatenate», permite crear, visualizar y concatenar archivos directamente desde el terminal. Se utiliza principalmente para previsualizar un archivo sin abrir un editor de texto gráfico:

```
cat long_text_file.txt
```

```
~/Documents/linux-commands via 🐘 v3.9.6  
> cat long_text_file.txt  
Not that large at all! :)
```

El comando `cat`.

## 22. Comando `ps`

Con `ps`, puedes echar un vistazo a los procesos que tu sesión de shell actual está ejecutando. Imprime información útil sobre los programas que está ejecutando, como el ID del proceso, el TTY (Teletipo), la hora y el nombre del comando.

```
ps
```

```
~  
> ps  
  PID TTY          TIME CMD  
 533494 pts/2    00:00:00 fish  
 539315 pts/2    00:00:00 ps
```

El comando `ps`.

En caso de que quieras algo más interactivo, puedes utilizar `htop`.



## 23. Comando `kill`

Es molesto cuando un programa no responde y no puedes cerrarlo por ningún medio. Afortunadamente, el comando `kill` resuelve este tipo de problemas.

En pocas palabras, kill envía una señal TERM o `kill` a un proceso que lo termina.

Puedes matar procesos introduciendo el PID (ID de los procesos) o el nombre binario del programa:

```
kill 533494
```

```
kill firefox
```

Ten cuidado con este comando – con `kill`, corres el riesgo de borrar accidentalmente el trabajo que has estado haciendo.

## 24. Comando `ping`

`ping` es la utilidad de terminal de red más popular que se utiliza para probar la conectividad de la red. `ping` tiene un montón de opciones, pero en la mayoría de los casos, lo utilizarás para solicitar un dominio o una dirección IP:

```
ping google.com
```

```
ping 8.8.8.8
```

## 25. Comando `vim`

`vim` es un editor de texto de terminal libre y de código abierto que se utiliza desde los años 90. Te permite editar archivos de texto plano utilizando combinaciones de teclas eficientes.

Algunas personas consideran que es difícil de usar — salir de Vim es una de las preguntas más vistas en StackOverflow- pero una vez que te acostumbras a él, se convierte en tu mejor aliado en la línea de comandos.

Para activar Vim, solo tienes que escribir:

vim

```

1 |
~
~
~
~
~
~
VIM - Vi IMproved
~
~
~
version 8.2.2891
~
by Bram Moolenaar et al.
~
Vim is open source and freely distributable
~
~
Become a registered Vim user!
~
type :help register<Enter>    for information
~
~
type :q<Enter>                to exit
~
type :help<Enter> or <F1>     for on-line help
~
type :help version8<Enter>   for version info
~
~
~
~
[No Name]                               (unix/) (line 0/1, col 0)\

```

El editor de texto vim.

## 26. Comando `history`

Si te cuesta recordar un comando, el `history` es muy útil. Este comando muestra una lista enumerada con los comandos que has utilizado en el pasado:

## history

```
256 man type
257 kill firefox
258 cat old_file
259 ping google.com
260 ping 8.8.8.8
261 ping -c 8 google.com
262 ps
263 cd
264 ls
265 history
[daniel@danielmanjaro ~]$
```

El comando history.

## 27. Comando `passwd`

`passwd` te permite cambiar las contraseñas de las cuentas de usuario. En primer lugar, te pide que introduzcas tu contraseña actual y, a continuación, te pide una nueva contraseña y una confirmación.

Es similar a cualquier otro cambio de contraseña que hayas visto en otros lugares, pero en este caso, es directamente en tu terminal:

`passwd`

```
~
> passwd
Changing password for daniel.
Current password:
```

El comando `passwd`

Ten cuidado al usarlo: ¡no querrás estropear tu contraseña de usuario!

## 28. Comando `which`

El comando `which` muestra la ruta completa de los comandos del shell. Si no puede reconocer el comando dado, arrojará un error.

Por ejemplo, podemos usar esto para comprobar la ruta binaria para Python y el navegador web Brave:

```
which python  
  
# /usr/bin/python  
  
which brave  
  
# /usr/bin/brave
```

## 29. Comando `shred`

Si alguna vez has querido que un archivo sea casi imposible de [recuperar](#), `shred` puede ayudarte con esta tarea. Este comando anula el contenido de un archivo repetidamente, y como resultado, el archivo dado se vuelve extremadamente difícil de recuperar.

Aquí hay un archivo con poco contenido:

A terminal window with a dark background. The prompt is ~/Documents/linux-commands via 🐙 v3.9.6. The user has entered the command cat file\_to\_shred.txt, and the output is A testing file, :))

```
~/Documents/linux-commands via 🐙 v3.9.6  
> cat file_to_shred.txt  
A testing file, :))
```

Archivo para triturar.

Ahora, hagamos que `shred` haga lo suyo escribiendo el siguiente comando:

```
shred file_to_shred.txt
```

[illegible]

Contenido sobrescrito.

Si deseas eliminar el archivo de inmediato, puedes utilizar el flag `-u`:

```
shred -u file_to_shred.txt
```

### 30. Comando `less`

`less` (opuesto a `more`) es un programa que permite inspeccionar archivos hacia atrás y hacia adelante:

```
less large_text_file.txt
```



```
~/Documents/linux-commands via 🐞 v3.9.6 took 3m38s
> tail long.txt

Hey, we're almost there.

These are the last lines of this text file.

We're trying to test out some linux commands, and this is a good sample text to do it.

To conclude, linux commands let you save a lot of time while being on a terminal or command line.

This is the end of this file bye!!!!
```

El comando tail.

Para ver solo las cuatro últimas líneas:

```
tail -n 4 long.txt
```

```
~/Documents/linux-commands via 🐞 v3.9.6
> tail -n 4 long.txt

To conclude, linux commands let you save a lot of time while being on a terminal or command line.

This is the end of this file bye!!!!
```

tail cuatro líneas.

## 32. Comando `head`

Éste es complementario al comando `tail`. `head` muestra las primeras 10 líneas de un archivo de texto, pero puede establecer cualquier número de líneas que desee mostrar con la flag `-n`:

```
head long.txt
```

```
head -n 5 long.txt
```

```
~/Documents/linux-commands via 🐙 v3.9.6
```

```
> head long.txt
```

```
Beggining of this large file!
```

```
Here goes a ton of content.  
Here goes a ton of content.  
Here goes a ton of content.  
Here goes a ton of content.  
Here goes a ton of content.  
Here goes a ton of content.  
Here goes a ton of content.  
Here goes a ton of content.
```

```
~/Documents/linux-commands via 🐙 v3.9.6
```

```
> head -n 5 long.txt
```

```
Beggining of this large file!
```

```
Here goes a ton of content.  
Here goes a ton of content.  
Here goes a ton of content.
```

El comando head.

### 33. Comando `grep`

Grep es una de las utilidades más potentes para trabajar con archivos de texto. Busca líneas que coincidan con una expresión regular y las imprime:

```
grep "linux" long.txt
```

```
~/Documents/linux-commands via 🐙 v3.9.6
```

```
> grep "linux" long.txt
```

```
We're trying to test out some linux commands, and this is a good sample text to do it.  
To conclude, linux commands let you save a lot of time while being on a terminal or command line.
```

El comando grep.

Puede contar el número de veces que se repite el patrón utilizando el flag `-c`:

```
grep -c "linux" long.txt
```



# 2

### 34. Comando `whoami`

El comando `whoami` (abreviatura de «who am i») muestra el nombre de usuario actualmente en uso:

```
whoami
```

```
# kinsta
```

Se obtendría el mismo resultado utilizando `echo` y la variable ambiental `$USER`:

```
echo $USER
```

```
# kinsta
```

### 35. Comando `whatis`

`whatis` imprime una descripción de una sola línea de cualquier otro comando, lo que lo convierte en una referencia útil:

```
whatis python
```

```
# python (1) - an interpreted, interactive, object-oriented programming language
```

```
whatis whatis
```

```
# whatis (1) - display one-line manual page descriptions
```

### 36. Comando `wc`

Wc significa «word count» (recuento de palabras) y, como su nombre indica, devuelve el número de palabras de un archivo de texto:

```
WC long.txt

# 37 207 1000 long.txt
```

Vamos a desglosar la salida de este comando:

- 37 líneas
- 207 palabras
- 1000 bytes de tamaño
- El nombre del archivo (long.txt)

Si solo necesitas el número de palabras, utiliza el indicador `-w`:

```
WC -w long.txt

207 long.txt
```

### 37. Comando `uname`

`uname` (abreviatura de «Unix name») imprime la información del sistema operativo, lo que resulta útil cuando se conoce la versión actual de Linux.

La mayoría de las veces, usted utilizará el flag `-a` (`-all`), ya que la salida por defecto no es tan útil:

```
uname

# Linux

uname -a
```

```
# Linux kinstamanjaro 5.4.138-1-MANJARO #1 SMP PREEMPT Thu Aug 5 12:15:21
UTC 2021 x86_64 GNU/Linux
```

### 38. Comando `neofetch`

Neofetch es una herramienta CLI (command-line interface) que muestra información sobre tu sistema -como la versión del kernel, el shell y el hardware- junto a un logotipo ASCII de tu distribución de Linux:

neofetch

```
~/Documents/linux-commands via v3.9.6
> neofetch
```



El comando neofetch.

En la mayoría de las máquinas, este comando no está disponible por defecto, así que asegúrate de instalarlo primero con tu gestor de paquetes.

### 39. Comando `find`

El comando `find` busca archivos en una jerarquía de directorios basándose en una expresión regex. Para utilizarlo, sigue la siguiente sintaxis:

```
find [flags] [path] -name [expression]
```

Para buscar un archivo llamado **long.txt** en el directorio actual, introduce lo siguiente

```
find ./ -name "long.txt" # ./long.txt
```

Para buscar archivos que terminen con una extensión **.py** (Python), puedes utilizar el siguiente comando:

```
find ./ -type f -name "*.py" ./get_keys.py ./github_automation.py  
./binarysearch.py
```

## Comodines

- `?`. Representa un único carácter. Así por si ejecutas `ls /dev/sda?` te listará todos los archivos que sean igual a `/dev/sda` más un único carácter, que puede ser cualquier letra o número. Así en mi caso, ha listado mis unidades, de la `/dev/sda1` a la `/dev/sda7`.
- `*`. Este comodín representa desde nada, hasta cualquier cantidad de caracteres y dígitos. Así, en el ejemplo que he indicado anteriormente, si ejecutados `ls /dev/sd*` te listará `/dev/sda`, `/dev/sda1` a `/dev/sda7` y si tienes otra unidad, `/dev/sdb`, `/dev/sdb1`,
- `[]`. En este caso, este comodín representa un rango, ya sea de caracteres o de números. Siguiendo con el ejemplo anterior, podemos listar `ls /dev/sda[1-5]`. Ya te puedes hacer una idea de lo que va a listar.
- `{}`. Esto en realidad es más un conjunto de comodines separados por comas. Igual que en casos anteriores, podrías ejecutar la

siguiente orden `ls {sda*,sdb*}`. Ojo con dejar espacios alrededor de la coma, porque te dará un error.

- `[!]`. El funcionamiento de este comodín es similar a `[]`, salvo que representa justo lo contrario. Es decir, se trata de listar todo aquello que no esté en ese rango. Por ejemplo, `ls /dev/sda[!1-5]` te mostrará `/dev/sda6` y `/dev/sda7`. Como te imaginas, esto es en mi caso, en tu equipo seguramente el resultado será otro.
- `\`. Esta es la secuencia de escape y que tienes que tener siempre muy presente. Con este carácter puedes mostrar otros caracteres. Así, si quieres crear el directorio `esta casa` y ejecutas `mkdir esta casa`, te creará dos directorios, `esta` y `casa`. Para hacer lo que quieres, tienes diferentes alternativas, o bien `mkdir "esta casa"` o bien `mkdir esta\ casa`. En esta segunda orden utilizamos `\` para escapar el carácter espacio.

#### 40. Comando `wget`

## Hoja de trucos de los comandos de Linux

Siempre que quiera una referencia rápida, solo tiene que revisar la siguiente tabla:

Comando	Uso
<code>ls</code>	Lista el contenido de un directorio
<code>alias</code>	Definir o mostrar alias
<code>unalias</code>	Eliminar las definiciones de <code>alias</code>
<code>pwd</code>	Imprime el directorio de trabajo
<code>cd</code>	Cambios en el directorio
<code>cp</code>	Copia archivos y directorios
<code>rm</code>	Eliminar archivos y directorios
<code>mv</code>	Mueve (renombra) archivos y directorios

Comando	Uso
<code>mkdir</code>	Crea directorios
<code>man</code>	Muestra la página del manual de otros comandos
<code>touch</code>	Crea archivos vacíos
<code>chmod</code>	Cambia los permisos de los archivos
<code>./</code>	Ejecuta un ejecutable
<code>exit</code>	Sale de la sesión actual del shell
<code>sudo</code>	Ejecuta los comandos como superusuario
<code>shutdown</code>	Apaga tu máquina
<code>htop</code>	Muestra información sobre procesos y recursos
<code>unzip</code>	Extrae archivos ZIP comprimidos
<code>apt</code> , <code>yum</code> , <code>pacman</code>	Gestores de paquetes
<code>echo</code>	Muestra las líneas de texto
<code>cat</code>	Imprime el contenido del archivo
<code>ps</code>	Informa del estado de los procesos del shell
<code>kill</code>	Termina los programas
<code>ping</code>	Prueba la conectividad de la red
<code>vim</code>	Edición eficiente de textos
<code>history</code>	Muestra una lista de comandos anteriores

Comando	Uso
<code>passwd</code>	Cambia la contraseña del usuario
<code>which</code>	Devuelve la ruta binaria completa de un programa
<code>shred</code>	Sobrescribe un archivo para ocultar su contenido
<code>less</code>	Inspecciona los archivos de forma interactiva
<code>tail</code>	Muestra las últimas líneas de un archivo
<code>head</code>	Muestra las primeras líneas de un archivo
<code>grep</code>	Imprime las líneas que coinciden con los patrones
<code>whoami</code>	Salidas nombre de usuario
<code>whatis</code>	Muestra descripciones de una sola línea
<code>wc</code>	Archivos de recuento de palabras
<code>uname</code>	Muestra la información del sistema operativo
<code>neofetch</code>	Muestra información sobre el sistema operativo y el hardware
<code>find</code>	Busca archivos que siguen un patrón
<code>wget</code>	Recupera archivos de Internet