# COMANDOS LINUX

## 1. Ayuda

- (**man**) para cada comando. Por ejemplo, para obtener más información sobre el comando **ls**, puedes escribir **man ls** en la terminal.

- En Linux, el comando **help** se utiliza para mostrar la ayuda interna de los comandos incorporados del shell. Este comando no funciona para todos los comandos del sistema operativo, sino solo para aquellos integrados en el shell. Los comandos externos a menudo tienen su propia opción de ayuda, que se muestra al ejecutar el comando con la opción **-h** o **--help**.
- Para usar el comando **help**, simplemente escribe **help** seguido del nombre del comando para el que deseas ver la ayuda. Por ejemplo, si deseas ver la ayuda para el comando **cd**, escribe lo siguiente en la terminal:


- Esto mostrará la ayuda interna del shell para el comando **cd**, que generalmente incluye una breve descripción de lo que hace el comando, sus opciones y argumentos, y ejemplos de su uso.
- Ten en cuenta que **help** solo funciona con los comandos integrados del shell, no con los comandos externos. Si deseas ver la ayuda para un comando externo, deberás consultar la documentación de ese comando o usar la opción de ayuda del comando (**-h** o **--help**) si está disponible.

## 2. Gestión de archivos y directorios.

1. **ls**: muestra el contenido de un directorio
2. **cd**: cambia de directorio
3. **mkdir**: crea un nuevo directorio
4. **rmdir**: elimina un directorio vacío
5. **rm**: elimina un archivo o directorio
6. **cp**: copia un archivo o directorio
7. **mv**: mueve o renombra un archivo o directorio
8. **touch**: crea un nuevo archivo vacío
9. **chmod**: cambia los permisos de un archivo o directorio
10. **chown**: cambia el propietario de un archivo o directorio

## 3. Montar discos

Se crea un **punto de montaje** dentro de la ruta **/media**.
Por ejemplo, la partición de datos de un segundo disco duro y se le asigna el

nombre 'Data', aunque puedes ponerle el que se quiera:

```
sudo mkdir /media/Data
```

```
$ sudo mkdir /media/Data
```

Al utilizar 'sudo' **nos pedirá la contraseña** ya que estamos solicitando permisos de administrador (root ). La escribimos (no aparecerá nada en pantalla) y pulsamos ENTER para confirmar.

Ahora necesitamos saber tanto la **partición** como el **disco** que queremos montar. Ejecutamos el comando:

```
lsblk
```

Cuyo resultado en mi PC es:

```
pirobtumen@ProbtPC:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 232,9G  0 disk
├─sda1   8:1    0   350M  0 part
├─sda2   8:2    0 139,4G  0 part
├─sda3   8:3    0     1K  0 part
└─sda5   8:5    0  93,1G  0 part /
sdb      8:16   0 465,8G  0 disk
├─sdb1   8:17   0 458,3G  0 part
├─sdb2   8:18   0     1K  0 part
└─sdb5   8:21   0   7,5G  0 part [SWAP]
```

Vemos que tenemos **dos discos**: **sda** y **sdb**. Para cada uno de ellos se indican las **particiones** (sda1, sda2, ..., sdb1, sdb2 ... ). Para saber cuál nos interesa podemos **ayudarnos del tamaño** ( **SIZE** ).

Para cada **sdXY**:
- **X son los disco**s y lo representan las letras a,b,c...
- **Y son las particiones** y lo representan los números.

Ahora **montamos la partición deseada** en el punto de montaje creado anteriormente:

```
sudo mount /dev/sdb1 /media/Data
```

Podemos **comprobar** si está montado con:

```
df -h
```

```
plrobtumen@ProbtPC:~$ df -h
S.ficheros      Tamaño Usados  Disp Uso% Montado en
udev              3,9G      0  3,9G   0% /dev
tmpfs             794M   9,2M  785M   2% /run
/dev/sda5          92G   5,2G   82G   6% /
tmpfs             3,9G    79M  3,8G   2% /dev/shm
tmpfs             5,0M   4,0K  5,0M   1% /run/lock
tmpfs             3,9G      0  3,9G   0% /sys/fs/cgroup
cgmfs             100K      0  100K   0% /run/cgmanager/fs
tmpfs             794M    52K  794M   1% /run/user/1000
/dev/sdb1         459G   6,1G  453G   2% /media/Data
```

1. Si nos fijamos la **última línea es el nombre de la partición que queríamos montar** y a la derecha aparece en el **punto de montaje /media/Data**.

Para desmontar el disco con **umount**

```
sudo umount /media/Data
```

## 4. Rutas absolutas vs rutas relativas

La ruta es uno de los conceptos más esenciales en Linux y esto es algo que todo usuario de Linux debe saber.

Una ruta es la forma de referirse a los archivos y directorios. Da la ubicación de un archivo o directorio en la estructura de directorios de Linux. Se compone de un nombre y de la sintaxis de la barra diagonal.

/home/itsfoss/scripts

Como usuario, tendrás que utilizar la ruta cuando quieras acceder a un determinado archivo o directorio o cuando tengas que dar la ubicación de un archivo o directorio a un comando o script.

cat /home/itsfoss/scripts/mi_script.sh

Recuerda que si la ruta comienza con la barra «/», la primera barra denota la raíz. El resto de las barras en la ruta son sólo separadores. Los principiantes a menudo se confunden entre la barra de raíz y las barras de separación.

En el diagrama anterior, tienes la primera ruta que comienza con la barra de la raíz. Hay otra ruta que no comienza con / (es decir, la raíz).

Ambas son correctas. La primera es una ruta absoluta y la segunda es una ruta relativa. Vamos a echar un vistazo detallado a ellos.

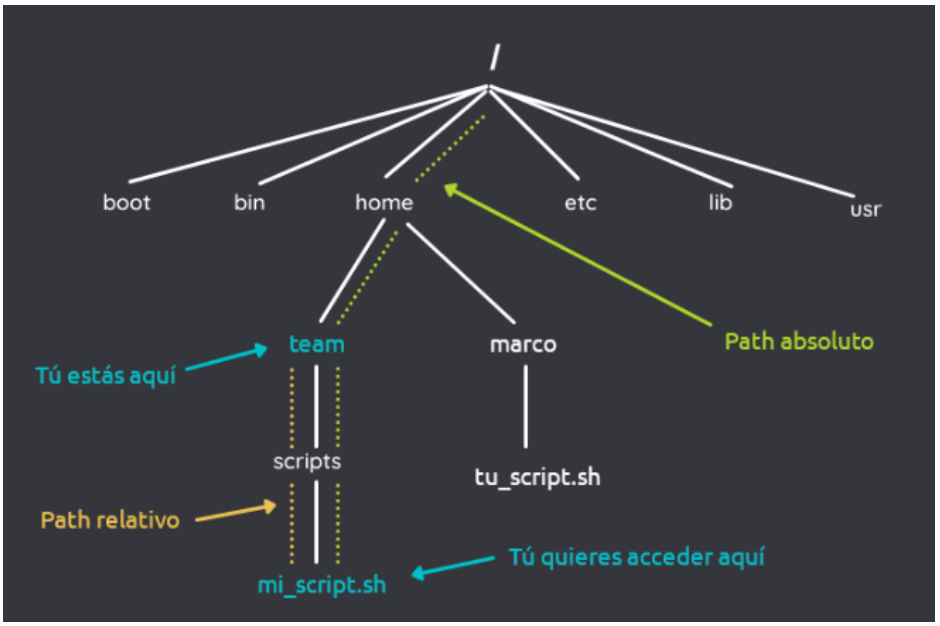
La ruta absoluta siempre comienza en el directorio raíz (/). Por ejemplo, **/home/itsfoss/scripts/mi_scripts.sh.**

Una ruta relativa comienza desde el directorio actual. Por ejemplo, si se encuentra en el directorio /home y desea acceder al archivo mi_script.sh, puede utilizar **itsfoss/scripts/mi_script.sh.**

Ya sabes que la estructura de directorios en Linux se parece a la raíz de un árbol. Todo comienza en la raíz y se ramifica desde allí.

Ahora imagina que estás en el directorio itsfoss y quieres acceder al archivo mi_script.sh.

La ruta absoluta está representada en la línea punteada verde y la ruta relativa está representada en las líneas punteadas amarillas.




Supongamos que quieres ver las propiedades del archivo mi_script.sh utilizando el comando ls.

Puedes utilizar la ruta absoluta que comienza con el directorio raíz (/):

ls -l /home/itsfoss/scripts/mi_script.sh

O bien, puedes utilizar la ruta relativa (que comienza en el directorio actual, no en /):

ls -l scripts/mi_script.sh

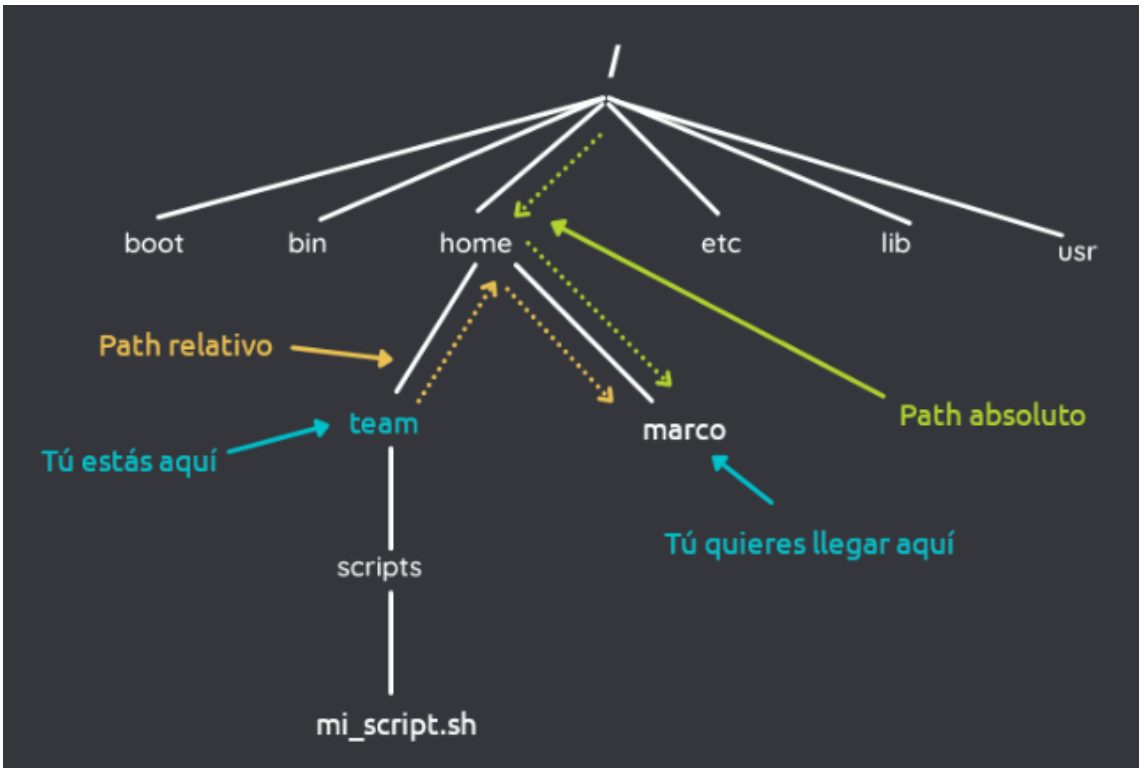
Ambos comandos darán el mismo resultado (excepto la ruta del archivo).

**Uso de la ruta relativa con los directorios . y ..**

Permíteme mostrarte otros ejemplos para explicar la diferencia entre la ruta absoluta y la ruta relativa. Pero antes de eso, debes saber acerca de dos rutas relativas especiales:

- . ( punto simple) denota el directorio actual en la ruta.
- .. (dos puntos) indica el directorio principal, es decir, un nivel superior.

Las cosas se aclararán en un momento. Echa un vistazo al escenario. En este, quieres ir al directorio marco desde el directorio itsfoss.




Puede utilizar el comando cd para cambiar de directorio. La ruta absoluta es bastante evidente aquí:

cd /home/marco

Para utilizar la ruta relativa, tendrá que utilizar la ruta relativa especial:

cd ../marco

## 5. Permisos archivos y carpetas. *chmod*

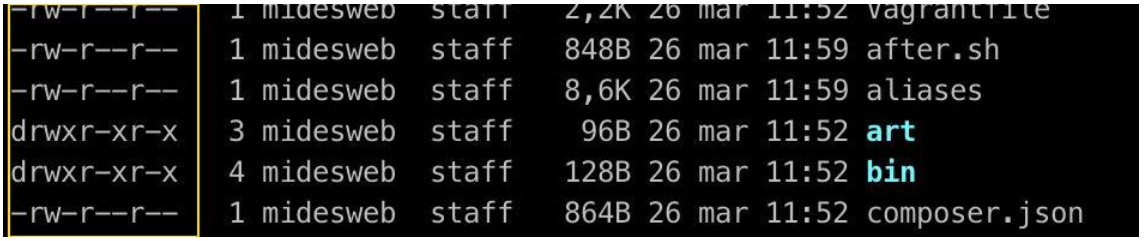
Como nos podemos imaginar, los permisos de archivo son las restricciones que tiene ese archivo para realizar operaciones con él. Estas operaciones pueden ser:

- Lectura
- Escritura (modificación)
- Ejecución

Cada archivo o directorio en sistemas Linux tiene una serie de permisos que se expresan a su vez en tres tipos de usuarios distintos.

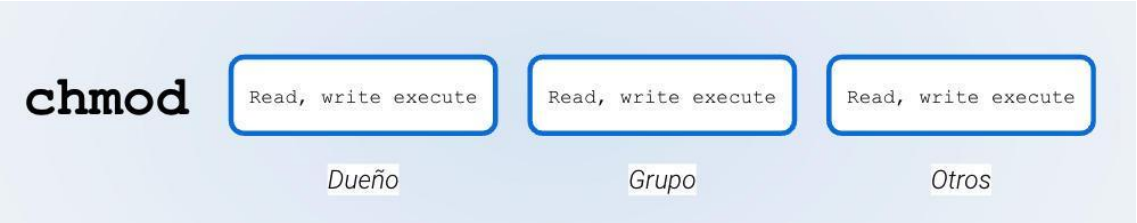
- Dueño: El usuario al que le pertenece el archivo
- Grupo: El Grupo al que pertenece el usuario dueño
- Otros: Todos los demás usuarios

Cuando hacemos un listado de directorio, usando el comando ls -l en un sistema Linux, o MacOS por ejemplo, se muestran los permisos para cada archivo o directorio, tal como aparece en la siguiente imagen.




**Cómo se expresan los permisos de archivo**
En la anterior imagen la parte donde se expresan los permisos está en un recuadro amarillo. Como puedes ver, está compuesta por varios caracteres que en principio podrían resultar difíciles de interpretar, por lo que los vamos a explicar.

Cada uno de los tipos de usuario y cada uno de los permisos que tiene cada tipo de usuario se expresan en grupos. Los grupos se estructuran como se puede ver en la siguiente imagen.




Como puedes ver, son tres grupos de tipos de usuario y para cada uno de ellos hay un caracter que indica si se puede leer, escribir, o ejecutar.

En un listado de directorio aparece una "r" si se puede leer. Para las modificaciones se indica una "w" y para las ejecuciones una "x". Si no tiene permisos para alguna de estas operaciones simplemente aparece un guión "-".

**Modo simbólico o carácter**

En el modo simbólico se asignan letras a las clases de usuarios y a los distintos permisos de acceso posibles. Combinándolos, puede definirse muy fácilmente qué derechos se han de otorgar o retirar y a qué usuarios. La tabla que verás a continuación recoge los caracteres utilizados en la notación simbólica para usuarios y permisos:

| Carácter para permisos | Significado |
|---|---|

| | |
|---|---|
| r | Permiso de lectura (*read*); también llamado bit R |
| w | Permiso de escritura (*write*); también llamado bit W |
| x | Permiso de ejecución (*execute*); también llamado bit X |
| **Carácter para clases de usuarios** | **Significado** |
| u | *user*, propietario |
| g | *group*, grupo |
| o | *other*, otros |
| a | *all*, todas las clases |

Si la definición de los derechos se hace por el modo simbólico, para vincular las clases de usuario con sus respectivos permisos se utilizan los siguientes **operadores:**

| | |
|---|---|
| **+** | Con el operador "+" se asignan más derechos de archivo a una clase de usuario. Solo se sobrescriben los derechos afectados. |
| **-** | El operador "-"retira derechos de acceso a una clase de usuario. |
| **=** | Si los permisos de archivo de una clase de usuario se han de renovar, sin importar qué derechos tuvo antes, se usa el operador "=". |

Si, por ejemplo, tenemos que modificar el archivo *ejemplo.txt* para que no solo su propietario (*user*), sino también todos los demás usuarios (*group*, *other*), tengan derechos de lectura y escritura, se tendrá que escribir el siguiente comando *chmod*:

$ chmod ugo+rw ejemplo.txt

Este cambio también podría aplicarse a todas las clases de usuario:

$ chmod a+rw ejemplo.txt

En el terminal, el archivo *ejemplo.txt* tendría ahora los siguientes derechos en lugar de los anteriores:

| **Permisos de acceso antes del cambio** | **Comando *chmod* ejecutado** | **Permisos de acceso después del cambio** |
|---|---|---|
| *-rw-------* | *a+rw* | *-rw-rw-rw-* |
| user: leer, escribir | a = all | user: leer, escribir |

| | | |
|---|---|---|
| group: sin permisos<br>other: sin permisos | + = añadir permiso<br>r = read<br>w = write | group: leer, escribir<br>other: leer, escribir |

Modo octal

Aunque la notación simbólica es una de las más utilizadas, su uso frecuente puede hacerla inmanejable. Por esta razón, muchos administradores recurren a la notación octal a la hora de atribuir derechos. Se trata de un número de tres cifras en el que cada posición representa una clase de usuario del servidor. La notación octal sigue siempre el mismo orden:

| **Posición de la cifra de la clase de usuario** | **Significado** |
|---|---|
| **1** | Corresponde a la clase de usuario “propietario” (*user*). |
| **2** | Corresponde a la clase de usuario “grupo” (*group*). |
| **3** | Corresponde a la clase de usuario “otros” (*other*). |

Para saber qué derechos se asignan a cada clase de usuario, basta con verificar el valor de cada cifra. Este resulta de la suma de los valores asociados con los respectivos derechos:

| **Valor para derechos de acceso** | **Significado** |
|---|---|
| 4 | Leer |
| 2 | Escribir |
| 1 | Ejecutar |
| 0 | Sin permisos |

De aquí podrían surgir las siguientes combinaciones de derechos de acceso:

| **Valor** | **Derecho(s) de acceso** |
|---|---|
| 0 | Ninguno |
| 1 | Solo ejecutar |
| 2 | Solo escribir |
| 3 | Escribir / ejecutar |
| 4 | Solo leer |

| | |
|---|---|
| 5 | Leer / ejecutar |
| 6 | Leer / escribir |
| 7 | Todos los permisos |

Para entender el modo octal, también recurriremos a un ejemplo. Si el propietario del archivo de nuestro ejemplo, *ejemplo.txt*, quisiera asignar derechos de lectura al *grupoXYZ*, tendrá que usar el siguiente código octal:

$ chmod 640 ejemplo.txt

Posicionando al valor *6* en primer lugar, el propietario se atribuye derechos de lectura (4) y de escritura (2). El segundo valor especifica los derechos de acceso del grupo: *4* (leer). Para otros usuarios, que ocuparían la tercera posición, no se especificó ningún derecho, indicándolo con un *0*.

**Opciones del comando chmod**

Al margen de si se usa la notación simbólica o la octal, el usuario siempre tendrá a su disposición una amplia gama de opciones en la asignación de derechos de acceso a archivos y directorios. Estos siempre se insertarán en la línea de comandos entre el comando y los permisos (o modos).

| **Código** | **Opción** | **Descripción** |
|---|---|---|
| *-R* | *recursive* | El cambio de los derechos de acceso se aplica a todos los archivos y subdirectorios dentro de una carpeta. |
| *-v* | *verbose* | Después del comando se emite un diagnóstico de todos los archivos procesados. |
| *-c* | *changes* | Después del comando se muestra un diagnóstico para todos los archivos que se han modificado. |
| *-f* | *silent* | Se silencian los mensajes de error. |

El siguiente ejemplo muestra un comando en el que el cambio de los derechos de acceso se refiere recursivamente a subdirectorios y archivos dentro de una carpeta:

$ chmod -R 744 carpetaXYZ

Para todos los archivos y subcarpetas en la *carpetaXYZ*, se aplican los siguientes permisos: el propietario recibe derechos de acceso completos (*7*), los miembros del grupo y los demás usuarios solo tienen permisos de lectura (*4*).

```
-rwx------  1 det det    0 2020-02-02 10:41 archivo1.txt
-rw-rw-r--  1 det det    0 2020-02-02 10:41 archivo2.txt
drwxr-xr-x  2 det det 4096 2020-02-02 10:44 directorio
```

***archivo1.txt***

| | Propietario | Grupo | Otros usuarios |
|---|---|---|---|
| **Leer** | ✓ | x | x |
| **Escribir** | ✓ | x | x |
| **Ejecutar** | ✓ | x | x |

***archivo2.txt***

| | Propietario | Grupo | Otros usuarios |
|---|---|---|---|
| **Leer** | ✓ | ✓ | ✓ |
| **Escribir** | ✓ | ✓ | x |
| **Ejecutar** | x | x | x |

***directorio***

| | Propietario | Grupo | Otros usuarios |
|---|---|---|---|
| **Leer** | ✓ | ✓ | ✓ |
| **Escribir** | ✓ | x | x |
| **Ejecutar** | ✓ | ✓ | ✓ |

**Agregar permisos con chmod**

En un primer caso de uso, asignaremos derechos de lectura y ejecución para todas las clases de usuarios del *archivo1.txt*, hasta ahora reservados al propietario. Utilizaremos para ello el comando *chmod* en modo simbólico:

chmod a+rx archivo1.txt

Todas las clases (*a*) obtienen así nuevos derechos (*+*) para el documento *archivo1.txt*, en concreto, permiso para leerlo (*r*) y ejecutarlo (*x*).

Si solicitamos una lista de los derechos que hemos otorgado, esta sería la entrada para *archivo1.txt*:

-rwxr-xr-x  1 det det    0 2020-02-02 10:41 archivo1.txt

**Retirar permisos con chmod**

En nuestra segunda demostración práctica, queremos retirar el derecho de escritura a todos los usuarios (también al propietario y a la clase grupo) para *archivo2.txt*. El comando que tendríamos que escribir en la consola sería (en el modo carácter):

chmod a-w archivo2.txt

En este ejemplo, el comando se refiere también a todas las clases (*a*). Con (-) se retira el derecho de escritura (*w*) en el *archivo2.txt* a todos los usuarios. La entrada del archivo en la consola quedaría entonces así:

-r--r--r--  1 det det    0 2020-02-02 10:41 archivo2.txt