

# COMANDOS BÁSICOS LINUX

Hay una extensa **lista de comandos** en **Linux** que nos permiten trabajar desde la **consola de texto**, y eso sin contar los **modificadores** que podemos añadir a esos **comandos** (de los que más tarde hablaremos), sin embargo vamos a empezar por algunos de los más básicos para ir soltándonos con nuestra amiga la **consola**:

- **ls**: Sirve para "listar" (mostrar) el contenido de una carpeta. Viene de la palabra inglesa "List".
- **pwd**: Nos dice la ruta del directorio en el que estamos.
- **mkdir**: Sirve para crear carpetas/directorios. Viene de "make directory" (crear directorio).
- **rmdir**: Se utiliza para borrar carpetas. Del inglés "remove directory" (eliminar directorio).
- **cp**: Sirve para copiar archivos o carpetas. Del término inglés Copy (copiar).
- **mv**: Se utiliza para cambiar de sitio un archivo o carpeta. Es lo equivalente a Cortar/Pegar en modo gráfico. Viene de "move" (mover).
- **rm**: Comando que empleamos para borrar archivos y carpetas. Viene de "remove" (borrar).
- **cd**: Sirve para cambiarnos de directorio. De la lengua natal de *Shakespeare* "change directory" (cambiar directorio).
- **sudo**: Comando que empleamos para adquirir privilegios de administrador. Viene de "super-user do" (hacer como superusuario).
- **su**: Mediante su podremos loguearnos como superusuario. Del inglés "super-user" (superusuario).
- **passwd**: Se usa para cambiar la contraseña de una cuenta. Viene de "password" (contraseña).
- **apt**: Nos permite comprobar actualizaciones y actualizar todo el sistema. Del inglés "advanced packets tool" (herramienta avanzada de paquetes).
- **aptitude**: Es una versión mejorada de **apt**.
- **man**: Muestra la documentación disponible para un determinado comando (manual).
- **-help**: Nos muestra ayuda de forma más resumida que con **man** de un determinado comando (ayuda).
- **clear**: Se utiliza para limpiar la pantalla (limpiar).

Estos serían los comandos más básicos. Como habréis visto, por lo general los comandos son abreviaturas de palabras en inglés.

## 1) MOSTRAR, CREAR Y BORRAR DIRECTORIOS

Vamos a practicar con ellos. Empezamos con el comando **ls**:

```
luis@jabba:~$ ls
amsn_received  Escritorio  Imágenes  Photos  Público  Videos
Documentos     Examples   Música    Plantillas  Software
```

Aquí está el contenido de mi `/home`. Como ya hemos dicho, el comando **ls** lo muestra en pantalla.

Si queremos que nos muestre el contenido de un directorio en particular, es tan sencillo como acompañar al comando **ls** del directorio que queremos. Por ejemplo para mostrar el contenido de mi carpeta **Documentos** sería así:

```
$ ls Documentos/
```

Ahora crearemos una carpeta, a modo de ejemplo, con el comando **mkdir** a la que llamaremos "carpeta de prueba":

```
$ mkdir "carpeta de prueba"
```

Una vez ejecutamos el comando, no nos sale nada que indique que se haya creado con éxito la carpeta. Simplemente se vuelve a quedar esperando instrucciones. Entonces **¿cómo sabemos si la hemos creado correctamente?** Basta con volver a utilizar el comando **ls** para que nos muestre el contenido del directorio:

```

luis@jabba:~$ mkdir "carpeta de prueba"
luis@jabba:~$ ls
amsn received  Documentos  Examples  Música  Plantillas  Software
carpeta de prueba  Escritorio  Imágenes  Photos  Público  Videos
luis@jabba:~$

```

Como vemos en la imagen de arriba el directorio llamado "carpeta de prueba" se ha creado satisfactoriamente.

**¡¡ IMPORTANTE !! Linux solo se pronuncia cuando hacemos algo mal, si todo va bien no avisa, es decir, no nos va a avisar que la carpeta se ha creado correctamente**

Supongo que os habréis percatado del **uso de las comillas**. En **Linux** es muy importante esto, ya que si queremos una carpeta llamada "**carpeta de prueba**" deberemos escribirlo como lo acabo de hacer, entre **comillas simples o dobles**.

Si lo hacemos sin comillas:

```
$ mkdir carpeta de prueba
```

Ocurre lo siguiente:

```

luis@jabba:~$ mkdir carpeta de prueba
luis@jabba:~$ ls
amsn received  de  Examples  Photos  Público
carpeta        Documentos  Imágenes  Plantillas  Software
carpeta de prueba  Escritorio  Música  prueba  Videos
luis@jabba:~$

```

Se nos crea una carpeta llamada "**carpeta**", una carpeta llamada "**de**" y una carpeta llamada "**prueba**" (las he marcado en **rojo** para diferenciarlas de la que he creado antes utilizando las comillas). Tres directorios nuevos cuando solo queríamos uno. He ahí la importancia de las comillas. Además, sin pretenderlo, esto nos ha servido para aprender a crear múltiples directorios con un solo comando (que también nos puede ser útil).

Ahora vamos a borrar las tres carpetas que acabamos de crear por error ("carpeta", "de" y "prueba"). Para ello usamos el comando **rmdir**:

```
$ rmdir carpeta de prueba
```

Como vemos en la imagen siguiente, ya se han borrado las tres carpetas.

```

luis@jabba:~$ rmdir carpeta de prueba
luis@jabba:~$ ls
amsn received  Documentos  Examples  Música  Plantillas  Software
carpeta de prueba  Escritorio  Imágenes  Photos  Público  Videos
luis@jabba:~$

```

Del mismo modo, si quisiéramos borrar la carpeta llamada "carpeta de prueba" deberíamos usar el mismo comando pero empleando las comillas.

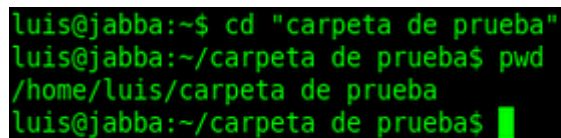
---

## 2) MOVERNOS ENTRE DIRECTORIOS

---

Ahora vamos a cambiarnos de directorio. Vamos a irnos a la carpeta que hemos creado ("carpeta de prueba") mediante el comando **cd**:

```
$ cd "carpeta de prueba"
```

A terminal window with a black background and green text. The prompt is 'luis@jabba:~\$'. The user enters 'cd "carpeta de prueba"'. The prompt changes to 'luis@jabba:~/carpeta de prueba\$'. The user enters 'pwd'. The output is '/home/luis/carpeta de prueba'. The prompt changes back to 'luis@jabba:~/carpeta de prueba\$' with a green cursor.

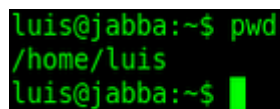
```
luis@jabba:~$ cd "carpeta de prueba"
luis@jabba:~/carpeta de prueba$ pwd
/home/luis/carpeta de prueba
luis@jabba:~/carpeta de prueba$
```

Como vemos en la imagen superior, para cerciorarnos de que estamos dentro, podemos usar el comando **pwd** que nos dice el directorio en el que estamos.

Para volver al directorio principal (nuestro /home) utilizamos el comando **cd ~** o **cd**:

```
$ cd ~
$ cd
```

Para asegurarnos que lo hemos hecho bien, volvemos a usar el comando **pwd** que nos dice nuestra ubicación actual:

A terminal window with a black background and green text. The prompt is 'luis@jabba:~\$'. The user enters 'pwd'. The output is '/home/luis'. The prompt changes to 'luis@jabba:~\$' with a green cursor.

```
luis@jabba:~$ pwd
/home/luis
luis@jabba:~$
```

Si queremos navegar hacia atrás entre los directorios, tan solo hemos de escribir:

```
$ cd ..
```

Si escribimos ese mismo comando repetidamente, lo que haremos será movernos hacia atrás entre los directorios hasta llegar al directorio raíz (/) del que salen todos los directorios en **Linux**.

Con **cd -** volvemos al directorio anterior:

```
$ cd -
```

Es sencillo: **cd ..** para ir hacia atrás y **cd -** para ir hacia delante.

Si lo que queremos es **movernos directamente** a un directorio determinado tan solo hemos de escribir **cd** seguido de la ruta a esa carpeta:

```
$ cd /home/luis/"carpeta de prueba"
```

---

### 3) BORRAR, COPIAR Y MOVER ARCHIVOS Y CARPETAS

---

Para borrar un archivo basta con usar el comando **rm**:

```
$ rm prueba.txt
```

En cambio para borrar una carpeta que tenga archivos dentro tenemos que usar el comando con un **modificador**. De no usar el modificador, el sistema no nos permitirá borrar una carpeta con contenido (para evitar eliminar cosas que no queramos). El modificador en cuestión es **-r** :

```
$ rm -r "carpeta de prueba"
```

Con este comando y su modificador borramos una carpeta y todo su contenido, así que cuidado con eliminar por error cosas que no queremos.

Si lo que queremos es **copiar** un archivo de una carpeta a otra usaremos el comando **cp**. Vamos a suponer que quiero copiar un supuesto archivo llamado **prueba.txt** que tengo en la carpeta **Documentos** a la carpeta **Plantillas**. Haremos lo siguiente:

```
$ cp /home/luis/Documentos/prueba.txt /home/luis/Plantillas
```

Si lo que queremos es mover el archivo de un directorio a otro lo que hacemos es utilizar el comando **mv** en lugar del **cp**. Por lo demás es igual:

```
$ mv /home/luis/Documentos/prueba.txt /home/luis/Plantillas
```

---

### 4) MODIFICADORES

---

Antes hemos hablado de los **modificadores**. Los modificadores (también conocidos como **argumentos**) son **caracteres** que se añaden a los comandos para ejecutar información complementaria o realizar tareas adicionales.

Por ejemplo para el caso del comando **ls**, podemos obtener mucha más información si acompañamos al comando de unos modificadores:

- **-l** nos muestra información sobre los archivos y carpetas.
- **-a** nos muestra archivos ocultos.

En el siguiente pantallazo tenemos un ejemplo de uso del modificador **-a** (las carpetas y archivos con un punto delante del nombre son los ocultos):

```

jesus@ubuntu:/var$ ls -la
total 48
drwxr-xr-x 14 root root 4096 2010-10-07 09:12 .
drwxr-xr-x 22 root root 4096 2013-10-12 04:21 ..
drwxr-xr-x  2 root root 4096 2013-10-12 04:50 backups
drwxr-xr-x 16 root root 4096 2010-10-07 09:09 cache
drwxr-xr-x  2 root root 4096 2010-10-07 09:12 games
drwxr-xr-x 61 root root 4096 2013-10-12 14:08 lib
drwxrwsr-x  2 root staff 4096 2010-10-07 02:15 local
drwxrwxrwt  3 root root   60 2013-10-21 01:12 lock
drwxr-xr-x 14 root root 4096 2013-10-20 10:12 log
drwxrwsr-x  2 root mail 4096 2010-10-07 08:56 mail
drwxr-xr-x  2 root root 4096 2010-10-07 08:56 opt
drwxr-xr-x 15 root root  640 2013-10-21 00:39 run
drwxr-xr-x  7 root root 4096 2010-10-07 09:04 spool
drwxrwxrwt  2 root root 4096 2010-10-07 02:15 tmp
jesus@ubuntu:/var$ ls -a
.  ..  backups  cache  games  lib  local  lock  log  mail  opt  run  spool  tmp
jesus@ubuntu:/var$

```

Y si queremos que nos muestre alguna información adicional podemos usar el modificador **-l**. Como vemos en la siguiente imagen podemos utilizar al mismo tiempo más de un modificador acompañando al comando. En este caso utilizo ambos:

```

jesus@ubuntu:/var$ ls -l -a
total 48
drwxr-xr-x 14 root root 4096 2010-10-07 09:12 .
drwxr-xr-x 22 root root 4096 2013-10-12 04:21 ..
drwxr-xr-x  2 root root 4096 2013-10-12 04:50 backups
drwxr-xr-x 16 root root 4096 2010-10-07 09:09 cache
drwxr-xr-x  2 root root 4096 2010-10-07 09:12 games
drwxr-xr-x 61 root root 4096 2013-10-12 14:08 lib
drwxrwsr-x  2 root staff 4096 2010-10-07 02:15 local
drwxrwxrwt  3 root root   60 2013-10-21 01:24 lock
drwxr-xr-x 14 root root 4096 2013-10-20 10:12 log
drwxrwsr-x  2 root mail 4096 2010-10-07 08:56 mail
drwxr-xr-x  2 root root 4096 2010-10-07 08:56 opt
drwxr-xr-x 15 root root  640 2013-10-21 00:39 run
drwxr-xr-x  7 root root 4096 2010-10-07 09:04 spool
drwxrwxrwt  2 root root 4096 2010-10-07 02:15 tmp

```

## 5) OTORGAR PRIVILEGIOS

Empezamos con el comando **sudo** con el que podemos trabajar desde la terminal como **superusuario**.

Cuando metemos nuestro nombre y contraseña en la pantalla de bienvenida de **Ubuntu**, nos logeamos como **usuarios**, no como **superusuarios**. La cuenta de usuario en **Ubuntu** es relativamente normal. Tiene **limitados** derechos de administración. Esto quiere decir, que cada vez que se haga algo que pueda suponer un riesgo para el sistema, nos va a solicitar hacerlo como superusuario.

Un ejemplo de su uso sería la modificación del fichero de configuración del **GRUB**. El fichero en cuestión está localizado en la carpeta **/boot/grub/** y ahí ningún usuario puede hacer modificaciones ni borrar nada si no es el **administrador** o tiene privilegios de administración. Lo segundo lo conseguimos gracias a **sudo**:

```
$ sudo gedit /boot/grub/menu.lst
```

En el ejemplo de arriba, vemos como además del **sudo** utilizamos el comando **gedit** junto a la ruta del fichero de configuración, así que aprovecho también para explicarlo.

El **gedit** es el editor de texto predeterminado de **Ubuntu**. En **Kubuntu** tenemos el **Kate**, en **Xubuntu** el **Mousepad**,... Antepone el comando que ejecuta el editor en cuestión a la ruta del fichero, porque el fichero está escrito en **texto plano** y necesitamos un editor de texto que lo ejecute y nos lo muestre en pantalla en modo gráfico, aunque de preferirlo podemos hacer que el contenido del archivo se muestre en la propia terminal mediante el comando **cat** que ya veremos más adelante.

Otro comando de gran interés y que podemos (y debemos) combinar con el anterior para poder hacer uso de él es **passwd**. Con este comando podremos cambiar la contraseña de nuestra cuenta. Introducimos en primer lugar la contraseña actual y después dos veces seguidas la nueva contraseña:

```
$ sudo passwd
```

Por otra parte, tenemos el comando **su** que aunque tiene una función similar a **sudo** también nos permite hacer **login** con otra cuenta distinta. Por ejemplo, imaginemos que tenemos otra cuenta llamada "invitado". Para logearnos como tal bastaría con escribir:

```
$ su invitado
```

---

## 6) ACTUALIZAR EL SISTEMA

---

**apt** es uno de los comandos más útiles que tenemos en las distribuciones basadas en **Debian**. Nos permite buscar, descargar e instalar todo tipo de paquetes además de comprobar actualizaciones y actualizar el sistema.

Existen varios **modificadores** (si no sabes qué son los modificadores, te recomiendo que le eches un vistazo a [este tutorial](#)) que se combinan con **apt** según lo que queramos hacer. Los más importantes son:

- `-get install`
- `-get update`
- `-get upgrade`
- `-apt-get remove`

```
$ apt-get install nombre_paquete
```

Si sabemos el nombre del paquete que queremos descargar e instalar, basta con anteponer **apt-get install** al nombre. Él solito busca, descarga e instala el paquete.

Por ejemplo:

```
$ apt-get install unrar
```

Nos descarga e instala el paquete para descomprimir archivos comprimidos en **.rar**. Para Debian es otro nombre de paquete, para ello buscar el paquete con **apt-cache search unrar**

```
$ apt-get update
```

Con este modificador, se actualizan los repositorios con lo que conseguimos actualizar el listado de todos nuestros paquetes, ganando así tiempo a la hora de hacer la búsqueda y consiguiendo que su posterior descarga sea más rápida.

Es decir, se actualiza el archivo con las versiones de los programas (paquetes) que tiene cada repositorio.

```
$ apt-get upgrade
```

Con esta sencilla orden, se actualiza nuestro sistema y todas las aplicaciones contenidas en los repositorios, con todas las posibles actualizaciones que pudiera haber.

```
$ apt-get remove nombre_paquete
```

Elimina el paquete especificado del sistema.

**NOTA:** Todos estos comandos necesitan tener privilegios de administración, por lo que tal y como hemos explicado arriba, necesitaremos combinarlos con el comando **sudo** para que tales privilegios nos sean otorgados.

El comando **aptitude** no es más que una versión mejorada de **apt** que instala junto al paquete solicitado, todos los paquetes secundarios que pudieran ser necesarios. Su uso es similar.

---

## 7) VARIOS

---

Con el comando **find** podremos buscar y encontrar el archivo o carpeta que le indiquemos:

```
$ find / -name ejemplo
```

El comando anterior buscaría en todos los sitios las carpetas y archivos que se llamen **ejemplo**.

También podemos indicarle que busque sólo en un directorio en particular. En este caso en el directorio **/home** por ejemplo:

```
$ find /home -name ejemplo
```

El comando **find** también admite el uso de **comodines**.

El comando **clear** sirve para limpiar la pantalla, aunque realmente no elimina todo lo que hay, sino que lo manda para arriba y nos despeja la consola. Tan sencillo como escribir:

```
$ clear
```

Del comando **man** y el argumento **-help** basta con decir que son imprescindibles para sacar todo el rendimiento de cada comando, ya que lo normal es que cada **comando** venga con un completo documento de ayuda sobre su **uso** y los **modificadores/argumentos** que admite:

```
$ man nombre_del_comando
```

Un ejemplo sería:

```
$ man sudo
```

Y obtendríamos toda la información relativa al comando **sudo**.

**-help** no es un comando propiamente dicho, sino que se engloba dentro del grupo de los modificadores/argumentos y es prácticamente admitido por la mayoría de **comandos** que existen. Es similar a **man** pero nos muestra la ayuda de forma bastante más resumida. Un ejemplo de su uso con el comando **find**:

```
$ find -help
```