

# Requisitos

## La terminal de GNU/Linux

Este material ha sido desarrollado en su totalidad por Román Ginés Martínez Ferrández ([rgmf@riseup.net](mailto:rgmf@riseup.net)) salvo referencias al pie de página.

Todas las imágenes utilizadas son de Dominio Público a menos que se diga lo contrario.



Creative Commons Reconocimiento – NoComercial - CompartirIgual  
CC by-nc-sa

## Sumario

1. ¿Qué es la terminal?.....	1
2. Sintaxis general.....	3
3. Comandos básicos.....	5
3.1. cd.....	5
3.2. pwd.....	5
3.3. ls.....	5
3.4. cat, more.....	6
3.5. mkdir.....	6
3.6. rm.....	7
3.7. cp.....	7
3.8. mv.....	9
3.9. wget.....	9
3.10. zip y unzip.....	9
3.11. Instalar, desinstalar y buscar programas en distribuciones basadas en Debian.....	10
3.12. sudo.....	11
4. Comodines.....	13

1

**¿Qué es la  
terminal?**

## 1. ¿Qué es la terminal?

---

El emulador de terminal o simplemente **terminal**, **consola** o **shell** es una interfaz en línea de comandos o CLI (*Command Line Interface*) que nos permite usar un ordenador y comunicarnos con él por medio de **órdenes o comandos**.

Estas fueron las primeras interfaces, luego aparecieron las interface gráficas o GUI (*Graphical User Interface*). No obstante, hoy en día las siguen usando administradores de sistema, programadores y usuarios avanzados por la gran potencia que ofrecen.

La terminal se encuentra en todos los sistemas operativos: Windows, MacOS, GNU/Linux, Android, iOS, etc.

### Nota 1

Los comandos u órdenes son en realidad programas que llevan a cabo una serie de instrucciones.

### Nota 2

Aunque no es exactamente lo mismo un emulador de terminal, un terminal, una consola o un *shell*, se suelen utilizar como sinónimos.

Lo que nosotros vamos a utilizar es en realidad un emulador de terminal o *shell*, aunque lo vamos a llamar indistintamente como terminal, consola o *shell*.

Para terminar, debes tener en presente, cuando uses una terminal, que siempre estás dentro de un directorio de trabajo de la jerarquía de directorios del Sistema Operativo.

2

# Sintaxis general

## 2. Sintaxis general

---

Los comandos, en general, siguen la siguiente sintaxis:

**comando [opciones] [argumentos]**

donde:

- **comando** es el nombre de la orden,
- **opciones** son determinadas opciones que se pueden aplicar sobre el comando y que cambian su comportamiento, y
- **argumentos** son datos que se le pasan al comando.

No todos los comandos tienen opciones y/o argumentos.

3

# Comandos básicos

## 3. Comandos básicos

### 3.1. cd

El comando `cd` (*change directory*) se utiliza, como su nombre indica, para cambiar a un directorio.

#### Directorio

Un directorio es una carpeta. Aunque son sinónimos en el contexto del *shell* se suele utilizar el término directorio.

Se usa de la siguiente manera:

```
cd <ruta absoluta o relativa>
```

Por ejemplo:

- `cd /home/roman` cambia al directorio `/home/roman`

### 3.2. pwd

El comando `pwd` (*Path WorD*) imprime por pantalla la ruta absoluta donde nos encontramos en estos momentos. Aunque tiene algunas opciones (muy pocas) nosotros lo vamos a utilizar tal cual, sin opciones ni argumentos.

### 3.3. ls

El comando `ls` (*LiSt*) imprime por pantalla los archivos y los directorios que hay donde estamos o los que hay en el directorio que le indicamos como argumento.

Se utiliza de la siguiente manera:

```
ls [opciones] [argumentos]
```

Se puede usar la orden sola o con opciones. Algunos ejemplos:

- `ls` imprime por pantalla con un formato estándar
- `ls -l` con la opción `-l` se imprime por pantalla los archivos y los directorios con información adicional como: usuario, fecha última actualización, permisos, etc.
- `ls -t` ordena los archivos y los directorios por fecha de última actualización.
- `ls -l -t` se aplican las opciones `-l` y `-t`.
- `ls -lt` se aplican las opciones `-l` y `-t`.

#### Aclaraciones

Como ves en los ejemplos de arriba las opciones constan de un guión seguido de una letra. Además se pueden concatenar opciones como ves en los dos últimos ejemplos.

Además, como se comenta arriba, se puede indicar un la ruta (absoluta o relativa) de un directorio para listar su contenido. Por defecto, se lista el contenido del directorio de trabajo (en el que estamos). Algunos ejemplos:

- `ls /etc` imprime por pantalla el contenido del directorio `/etc`
- `ls -l /etc` imprime por pantalla el contenido del directorio `/etc` mostrando información adicional (opción `-l`).
- `ls /usr/local/bin` imprime por pantalla el contenido del directorio `/usr/local/bin`

### 3.4. cat, more

Los comandos `cat` (*conCAT*) y `more` imprimen por pantalla el contenido de un fichero de texto.

Estos comandos requieren como argumento la ruta absoluta o relativa del fichero de texto:

```
cat <ruta al fichero de texto>
more <ruta al fichero de texto>
```

Algunos ejemplos:

- `cat /home/roman/informatica.txt` Muestra el contenido del fichero `informatica.txt` que está dentro de `/home/roman`.
- `more ../informatica.txt` Muestra el contenido del fichero `informatica.txt` que está en el directorio de arriba.
- `more informatica.txt` Muestra el contenido del fichero `informatica.txt` que está en el directorio de trabajo actual.

### 3.5. mkdir

El comando `mkdir` (*MaKe DiRectory*) crea un directorio en el lugar especificado. Este comando necesita, al menos, un argumento: el nombre del nuevo directorio.

Este comando lo usamos de la siguiente manera:

```
mkdir [opciones] <dir1> <dir2> <dir3> ... <dirN>
```

Algunos ejemplos:

- `mkdir /home/roman/nuevo` Crea el directorio nuevo dentro de `/home/roman`.
- `mkdir uno dos tres` Crea los directorios `uno`, `dos` y `tres` en el directorio de trabajo actual.



- `mkdir Documentos/nuevo1 nuevo2` Crea los directorios `nuevo1` dentro de `Documentos` y `nuevo2` dentro del directorio de trabajo.

Este comando tiene un opción muy útil y utilizada: la opción `-p` que permite crear los directorios padres de una ruta si no existen. Por ejemplo: si ejecutamos la orden:

```
mkdir /home/roman/trabajo/nuevo
```

pero el directorio `trabajo` no existe dentro de `roman`, entonces `mkdir` acaba con un error dado que no puede crear el directorio `nuevo` si no existe el directorio `trabajo`. Sin embargo, si ejecutamos la misma orden con la opción `-p`:

```
mkdir -p /home/roman/trabajo/nuevo
```

crearía la carpeta `home` si no existe, la carpeta `roman` si no existe, la carpeta `trabajo` si no existe y la carpeta `nuevo` si no existe. Por tanto no daría error.

### 3.6. rm

El comando `rm` (*ReMove*) se utiliza para eliminar ficheros y directorios. Se utiliza de la siguiente manera para eliminar ficheros:

```
rm <ruta al/los fichero/s a eliminar>
```

Y de la siguiente manera (con la opción `-r`) para eliminar directorios (aunque también borraría ficheros):

```
rm -r <ruta al/los fichero/s o directorio/s a eliminar>
```

Ejemplos:

- `rm Documentos/informatica.txt` Borra el fichero `informatica.txt` que hay en `Documentos`.
- `rm -r /home/roman/Documentos` Borra el directorio `Documentos` y todo lo que haya dentro (es peligroso y hay que utilizarlo con cuidado).
- `rm uno.txt dos.txt tres.txt` Borra los tres ficheros.
- `rm -r uno.txt dir1 dir2` Borra el fichero y los tres directorios.

### 3.7. cp

El comando `cp` (*CoPy*) se utiliza para copiar ficheros y directorios. Como en el caso de `rm` se utiliza la opción `-r` para copiar directorios y sin opción para copiar ficheros. Además, se necesitan dos argumentos:

- origen: ruta al fichero o directorio que se quiere copiar.
- destino: ruta al fichero o directorio donde se quiere pegar.

```
cp <origen> <destino>
cp -r <origen> <destino>
```

Algunos ejemplos:

- `cp /home/roman/infor.txt /home/roman/Documentos/infor.txt`  
Copia el fichero `infor.txt` que está en `/home/roman` dentro de `/home/roman/Documentos` con el mismo nombre.
- `cp Documentos/avatar.jpg Escritorio/avatar.jpg`  
Copia la imagen `avatar.jpg` que está dentro de `Documentos` en la carpeta `Escritorio` con el mismo nombre.
- `cp snoopdog.mp3 Musica/snoopdog.mp3`  
Copia `snoopdog.mp3` que está en el directorio de trabajo dentro de la carpeta `Musica` con el mismo nombre.
- `cp inform.txt Documentos/informatica.txt`  
Copia el fichero `inform.txt` que hay en el directorio de trabajo dentro de la carpeta `Documentos` con el nombre `informatica.txt` (se le ha cambiado el nombre al fichero).
- `cp snoopdog.mp3 Musica/.`  
Forma especial de copiar dentro de un directorio un fichero con el mismo nombre: mediante el `.` al final como ves.

## Comodines

El `“.”` es un comodín que se utiliza para ahorrarnos el escribir el mismo nombre cuando utilizamos el comando `cp`. Otro comodín que se emplea muy a menudo en órdenes como `cp` y `rm` es el `“*”`.

El `“*”` significa “todo”. Lo entenderás mejor con algunos ejemplos:

**`rm *`**

elimina todos los ficheros en la carpeta de trabajo

**`rm -r *`**

elimina todos los ficheros y directorios que hay en la carpeta de trabajo

**`rm /home/roman/Documentos/*`**

elimina todos los ficheros que hay dentro de la carpeta `/home/roman/Documentos`

**`cp * /home/roman/Documentos/.`**

Copia todos los ficheros de la carpeta de trabajo a la carpeta `/home/roman/Documentos` con el mismo nombre

**`cp -r * /home/roman/Documentos/.`**

Copia todos los ficheros y directorios de la carpeta de trabajo a la carpeta `/home/roman/Documentos` con el mismo nombre

### 3.8. mv

El comando mv (*MoVe*) se utiliza para mover archivos y directorios de un sitio a otro. Su sintaxis es muy sencilla:

```
mv <origen> <destino>
```

donde tanto origen como destino son rutas: la ruta del fichero que quiero mover y la ruta del lugar donde lo quiero mover, respectivamente.

Algunos ejemplos:

- `mv foto.png Imagenes/foto.png`  
Mueve la imagen `foto.png` a la carpeta `Imagenes` con el mismo nombre.
- `mv foto.png Imagenes/.`  
Igual que el comando anterior pero usamos el comodín `.` para indicar que la queremos mover con el mismo nombre.
- `mv foto.png Imagenes/foto_coche.png`  
Movemos la imagen `foto.png` dentro de `Imagenes` cambiando el nombre a `foto_coche.png`.
- `mv /home/roman/Documentos/* /home/roman/backup/.`  
Movemos todo lo que hay dentro de la carpeta `Documentos` del usuario `roman` al directorio `backup` que hay dentro de la carpeta personal del usuario `roman`.

### 3.9. wget

Se utiliza para descargar ficheros desde Internet (o cualquier red en general). Lo único que debemos conocer es la URL del recurso que queremos descargar. Su uso es muy simple:

```
wget <URL del recurso>
```

### 3.10. zip y unzip

Para comprimir ficheros y directorios se utiliza el comando zip de la siguiente manera:

```
zip <fichero> <lista>
```

donde:

- `fichero` es el nombre del fichero comprimido que se obtiene como resultado de comprimir la lista, y

- lista es la lista de ficheros y directorios que se van a introducir en el fichero comprimido.

Por ejemplo:

- `zip imagenes.zip imagen1.jpg imagen2.png imagen3.jpg imagen4.jpg`  
comprime las imágenes `imagen1.jpg`, `imagen2.png`, `imagen3.jpg` e `imagen4.jpg` en un fichero llamado `imagenes.zip`
- `zip backup.zip /home/roman/Documentos`  
comprime la carpeta `Documentos` del usuario `roman` en un fichero llamado `backup.zip`

Para descomprimir se utiliza la orden `unzip` tal que así:

```
unzip <fichero comprimido>
```

donde:

- fichero comprimido es el fichero comprimido en formato zip.

El resultado de descomprimir se deja en el directorio en el cual estás ejecutando la orden.

### 3.11. Instalar, desinstalar y buscar programas en distribuciones basadas en Debian

En las distribuciones de GNU/Linux basadas en Debian como son Lliurex o Ubuntu, entre otras, se utiliza un gestor de paquetes denominado APT (*Advanced Packaging Tool*) que se emplea para instalar y desinstalar programas.

Todas las distribuciones traen consigo herramientas GUI para instalar y desinstalar programas. No obstante, aprenderás a instalar y desinstalar programas mediante la herramienta para la terminal denominada `apt-get`.

Para instalar un programa mediante `apt-get` se emplea la orden siguiente:

```
apt-get install <nombre del paquete>
```

donde:

- nombre del paquete es el nombre del programa que queremos instalar.

En el mundo GNU/Linux a los programas se les denomina paquetes. En realidad, se puede decir que un programa en GNU/Linux es un conjunto de paquetes. De hecho, cuando instales algún programa verás que se instalarán varios paquetes en muchos casos.

Para desinstalar un programa mediante `apt-get` se emplea la orden siguiente:

```
apt-get remove <nombre del paquete>
```

donde:

- nombre del paquete es el nombre del programa que queremos desinstalar.

También podemos buscar programas en el repositorio de programas (una especie de “tienda de aplicaciones”) mediante la orden `apt-cache` tal que así:

```
apt-cache search <criterio de búsqueda>
```

donde:

- criterio de búsqueda es el patrón de búsqueda que deseamos utilizar para encontrar el programa que buscamos.

Solo los usuarios administradores pueden instalar y desinstalar programas.

### 3.12. sudo

Hay tareas y comandos que solo pueden ejecutar los usuarios con permisos de administrador. Para llevar a cabo dichas tareas debemos escalar en credenciales usando el comando:

```
sudo su
```

el cual nos pedirá la contraseña. Tras ellos comprueba que nuestro usuario tiene permisos de administración, si es así acabamos de entrar en un entorno peligroso porque hemos escalado en permisos y podemos hacer lo que deseemos, nada nos podrá detener.

Cuando estamos “logueados” como administradores el final del prompt tiene un `#` en vez de `$`.

Tras ellos podemos llevar a cabo tareas como:

- Instalar y desinstalar programas.
- Crear y eliminar usuarios.
- Cambiar permisos.
- Programar tareas.
- Y, en definitiva, cualquier cosa que se pueda hacer con un ordenador sin límites.

En realidad sí hay límites que root puede establecer como explicaré en clase.

Seguro que has oído hablar de “rootear un teléfono con Android”. Esto significa que has activado el usuario root que es el “Dios de los usuarios” ya puede hacer cualquier cosa y, por eso, por motivos de seguridad, viene desactivado en los dispositivos con Android y en algunas distribuciones de GNU/Linux.

En GNU/Linux hay tres tipos de usuario:

- El usuario root.
- Usuarios administradores.
- Usuarios regulares.

4

# Comodines

## 4. Comodines

Los comodines son patrones de búsqueda que permite seleccionar palabras, frases o secuencias de caracteres. Se pueden usar con prácticamente cualquier comando como `cp`, `rm` o `ls`, por ejemplo.

Estos patrones son, a su vez, secuencias de caracteres con un significado especial.

Los comodines más utilizados son los siguientes:

- `?` representa un único carácter

### **ls fic?.txt**

Ejemplo que muestra todos los ficheros cuyo nombre sea "fic" seguido de un carácter y a continuación ".txt". Por ejemplo, ficheros con los siguientes nombre:

fic1.txt

fic2.txt

fico.txt

Pero no listaría ficheros cuyo nombre fuera, por ejemplo:

fic11.txt

fico1.txt

- `*` representa cualquier conjunto de caracteres: desde 0 caracteres hasta cualquier cantidad de caracteres (letras o dígitos).

### **ls fic\*.txt**

Este ejemplo mostraría todos los ficheros que empiezan por "fic" y acaban por ".txt". Por ejemplo, ficheros con los siguientes nombres:

fichero.txt

fic1.txt

fic12.txt

fic123.txt

fica.txt

fico.txt

- `[]` representa un rango, ya sea de letras o dígitos.

```
ls fic[1-5].txt
```

Este ejemplo mostraría los ficheros cuyo nombre fueran:

```
fic1.txt
```

```
fic2.txt
```

```
fic3.txt
```

```
fic4.txt
```

```
fic5.txt
```

```
ls fic[aeiou].txt
```

Este otro ejemplo mostraría los ficheros cuyos nombres fueran:

```
fica.txt
```

```
fice.txt
```

```
fici.txt
```

```
fico.txt
```

```
ficu.txt
```

- `[]` representa justo lo contrario que el comodín `[]`.

```
ls fic[!aeiou].txt
```

Este ejemplo mostraría todos los ficheros que empiecen por "fic"; sigan por cualquier carácter excepto "a", "e", "i", "o", "u"; y termine por ".txt".

Este ejemplo mostraría ficheros con estos nombres:

```
fic1.txt
```

```
ficb.txt
```

Pero no mostraría ficheros con estos otros nombres:

```
fica.txt
```

```
fice.txt
```

```
fici.txt
```

```
fico.txt
```

```
ficu.txt
```

- `{}` no es un comodín como tal pero permite especificar varios comodines separados por puntos.



```
ls {fic*.txt,fac*.txt}
```

Este ejemplo mostraría los ficheros que empiecen por “fic” y acaben por “.txt” y también los que empiecen por “fac” y acaben por “.txt”.

- `\` no es un comodín como tal pero es útil en determinados casos. Se trata del carácter de escape y sirve para “escapar” caracteres que puedan tener un significado especial. Lo veremos con ejemplos en clase.