Comandos para la gestión de archivos / directorios en GNU/Linux

1. Caracteres comodines

Los caracteres comodines se utilizan para sustituir a un carácter o a un conjunto de caracteres. Los caracteres comodines que podemos usar son:

Caracter	Función
*****	Hace referencia a un conjunto o cadena de caracteres de cualquier tamaño, incluso de tamaño 0.
?	Hace referencia a un carácter.
[ ]	Hace referencia a un carácter. Dentro de los corchetes podemos incluir un conjunto de caracteres o un rango de caracteres, pero el corchete se sustituirá solo por un carácter.
{ }	Hace referencia a varias cadenas de caracteres, que se escribirán dentro de las llaves separadas por comas.

2. Carácter de escape

Hay ciertos caracteres que no son imprimibles y otros, como el carácter espacio, que a la hora de utilizarlo en la línea de comandos, podríamos tener problemas, porque el sistema puede creer que estamos **introduciendo dos o más argumentos** en vez de un argumento que contenga espacios en blanco.

En este caso, el nombre de archivos y directorios se protege poniéndolo entre comillas, o bien utilizando el carácter de escape \.

3. Comandos

ls (list)
Muestra **información sobre ficheros y directorios**. Si no se especifica nada muestra información sobre el directorio actual. Cuando elecutamos **ls -l**, lo que vemos es la información que contiene la tabla de i-nodos de los ficheros, directorios y ficheros especiales que están en el directorio donde ejecutamos el comando.

Sintaxis:
ls [opciones] [argumentos]

Opciones:
-d | --directory → muestra la información sobre el directorio en vez de sobre el contenido del directorio.
-a | --all → permite ver los nombres de ficheros y directorios que comienzan por punto.
-A | --almost-all → permite ver los nombres de ficheros y directorios que empiezan por punto excepto los directorios "." y "..".
-l → muestra la información en formato largo, con información adicional, como tipo de archivo, tamaño, fecha de modificación, propietario y permisos.
-h | --human-readable → junto con las opciones "l" o "s" muestra el tamaño en la unidad de medida mayor, para que se pueda entender mejor.
-i | --inode → muestra el número de i-nodo del fichero.
-n | --numeric-uid-gid → igual que "l", pero mostrando el número GID y UID en lugar de los nombres de usuario y grupo.
-c → muestra la información ordenada por día y hora de creación.
-t → el orden es por día y hora de modificación.
-r | --reverse → muestra el resultado ordenado en orden inverso.
--color → muestra el contenido coloreado.
-F |--classify → muestra información sobre el tipo de fichero. Los símbolos que aparecen junto al nombre del fichero indican:
* ejecutable
/ directorio
@ enlace simbólico (0 → si lo usamos con la opción -l)
| tubería
ningún símbolo indica fichero regular.
-R | --recursive → muestra los directories por debajo del actual de forma recursiva.
-s | --size → muestra el tamaño en bloques de cada fichero.
-S → muestra los ficheros ordenados por tamaño.

Argumento/s:
Para los argumentos de este comando se pueden utilizar caracteres comodines.
directorio
Muestra el contenido del directorio especificado.
ficheros
Muestra información sobre el fichero o ficheros que se le indiquen como argumentos.

Desde nuestro directorio, mostramos en formato largo **los ficheros tty0 a tty9 del directorio /dev**, utilizando caracteres comodines. Existen varias opciones que aparecen a continuación:

```
paco@ubuntupaco:~ $ ls -l /dev/tty[0-9]
paco@ubuntupaco:~ $ ls -l /dev/tty?
paco@ubuntupaco:~ $ ls -l /dev/tty{1,2,3,4,5,6,7,8,9}
```

Mostramos los **ficheros** del directorio **/dev** que **empiecen por tty** sin importar que caracteres haya después.

```
paco@ubuntupaco:~ $ ls -l /dev/tty*
```

Mostramos las entradas de tu directorio personal, de manera que te muestre el i-nodo de cada una e información sobre el tipo de fichero.

```
paco@ubuntupaco:~ $ ls -Fl
```

Mostramos las entradas de tu directorio personal, en formato largo.

```
paco@ubuntupaco:~ $ ls -l
```

Mostramos ahora con el tamaño de los ficheros en la unidad de medida mayor que se pueda.

```
paco@ubuntupaco:~ $ ls -lh
```

La **información que obtenemos al escribir el comando ls -l**, es decir, en formato largo es la siguiente:

1. El primer carácter hace referencia al tipo de archivo:
   - **-** → fichero regular
   - **l** → enlace simbólico
   - **d** → directorio
   - **b, c, p** → archivos especiales de bloques, de caracteres y tuberías (pipes o fifo), respectivamente.

2. Los siguientes **9 caracteres indican los permisos que el archivo tiene o no activados**. Los permisos podrán ser: **r** lectura, **w** escritura y **x** de ejecución. Un **guión** (-) significa que el permiso no está activado.

   Hay tres grupos de permisos porque el primero corresponde al propietario del fichero, el segundo a los de su grupo y el tercero al resto de usuarios del sistema.

3. El número que sigue es el **número de enlaces que tiene ese fichero**, en este caso es el número de **enlaces duros**.

4. Los siguientes nombres corresponden al **usuario propietario del archivo y del grupo al que pertenece el archivo**. El usuario deberá pertenecer al grupo.

5. Lo siguiente que vemos es el **tamaño del fichero en bloques**.

6. A continuación vemos la **fecha y hora de creación**.

7. Por último, el **nombre del fichero**. Si es un enlace simbólico, veremos también a qué archivo o directorio apunta el enlace.

Además de esta información, podemos obtener información adicional o de forma diferente si combinamos la opción -l con otras opciones.

pwd (**print working directory**)
Muestra la ruta absoluta del directorio donde nos encontramos en ese momento, es decir, el directorio de trabajo o actual.

Sintaxis:
pwd

mkdir (**make directories**)
Crea directorios.

Sintaxis:
mkdir [opciones] [directorio/s]

Opciones:
-v |--verbose → muestra un mensaje por cada directorio creado.

Argumento/s:
Crea el directorio con el nombre que se le especifique. Se puede poner uno o varios nombres de directorios.

cd (**change directory**)
Cambia de directorio.

Sintaxis:
cd [opciones] [directorio]

Opciones:
- → cambia al directorio último donde estuvimos antes del actual.
.. → cambia al directorio por encima del actual (directorio padre).
~ → cambia al directorio personal del usuario. Se obtiene el mismo resultado que si no ponemos ninguna opción ni argumento, es decir, el nombre de comando solo.

Argumento:
Cambia al directorio que se le especifique como argumento. Se le puede indicar mediante una ruta absoluta o relativa.

Mostramos la ruta absoluta del directorio donde estemos y después, nos movemos al directorio raíz. Vamos al directorio **/etc/init.d**. Utilizamos el comando que nos lleve al directorio de dónde venimos.

```
paco@ubuntupaco:~ $ pwd
paco@ubuntupaco:~ $ cd /
paco@ubuntupaco:~ $ cd /etc/init.d
paco@ubuntupaco:~ $ cd -
paco@ubuntupaco:~ $ pwd
```

Vamos a nuestro directorio personal y comprobamos que los directorios “.” y “..” son enlaces duros al mismo subdirectorio y al directorio padre.

```
paco@ubuntupaco:~ $ cd
paco@ubuntupaco:~ $ ls –ai
paco@ubuntupaco:~ $ cd.
paco@ubuntupaco:~ $ ls -di /home
paco@ubuntupaco:~ $ ls –i
paco@ubuntupaco:~ $ cd
```

Son enlaces duros porque **el número de i-nodo del directorio “.” coincide con el directorio personal y el número de inodo de “..” coincide con el del directorio /home**.

rmdir (**remove directory**)
Borra directorios si están vacíos.

Sintaxis:
rmdir [opciones] directorio/s

Argumento:
Borra los directorios que se le pasen como argumentos.

rm (**remove**)
Borra ficheros y directorios.

Sintaxis:
rm [opciones] [argumento/s]

Opciones:
-d | --directory → borra el directorio, aunque no esté vacío.
-i | --interactive → pregunta antes de borrar cada fichero o directorio.
-r | -R | --recursive → borra los directorios, los ficheros que contengan y los directorios que estén por debajo de él, de forma recursiva.
-V |--verbose → muestra un mensaje por cada directorio o fichero borrado.

Argumento/s:
Borra los ficheros o directorios cuyo nombre se le pase como argumento. Se pueden pasar más de un argumento y se pueden usar caracteres comodines.

cp (**copy**)
Copia uno o varios ficheros en otro fichero o en un directorio.

Sintaxis:
cp [opciones] fichero/s destino

Opciones:
-d | --directory → muestra la información sobre el directorio en vez de sobre el contenido del directorio.
-a | --all → permite ver los nombres de ficheros que comienzan por un punto.
-f | --force → fuerza la copia. Si el destino existe y no se puede abrir, lo borra e intenta copiar de nuevo.
-I | --interactive → pregunta antes de sobrescribir.
-r | -R | --recursive → copia directorios y los que están por debajo de él, de forma recursiva.

mv (**move**)
Mueve un fichero o ficheros a otro fichero o directorio. Es equivalente a una copia seguida del borrado del original. Puede ser usado para renombrar ficheros.

Sintaxis 1:
mv [opciones] fichero/s destino

Opciones:
-u | --update → mueve solo si el destino no existe o es anterior al fichero fuente.
-i | --interactive → pregunta antes de sobreescribir.
-f | --force → fuerza la sobreescritura.
-v | --verbose → muestra un mensaje por cada fichero movido.

Sintaxis 2:
mv [opciones] -t directorio fuente

Opciones:
-t | --target-directory=directorio → mueve todo el contenido de fuente en directorio.

file
Muestra el tipo de fichero.

Sintaxis:
file nombre

Argumento:
El argumento obligatorio deberá ser el nombre de un fichero o directorio.

du (**disk usage**)
Muestra el espacio que ocupa el fichero o directorio.

Sintaxis:
du [opciones] [argumento/s]

Opciones:
-b | --bytes → muestra el tamaño en bytes.
-h | --human-readable → muestra el tamaño en la unidad de medida mayor, para que se entienda mejor.

Argumento/s:
Como argumento se le puede pasar uno o varios ficheros o directorios.

df (**display free**)
Muestra el espacio libre en los dispositivos de almacenamiento y en las particiones montadas.

Sintaxis:
df [opciones]

Opciones:
-h | --human-readable → muestra el tamaño en la unidad de medida mayor, para que se entienda mejor.
-K | --block-size=1K.
-a | --all → muestra todos, incluso los que tengan tamaño 0.

cat (**catenate**)
Muestra el contenido de los ficheros que se le pasen como argumentos. El comando **cat** se utiliza, además, para mostrar el contenido de un fichero, para crearlo y para concatenar varios ficheros mediante los redireccionamientos.

Sintaxis:
cat [opciones] fichero/s

Opciones:
-n | --number → numera todas las líneas.

Argumento/s:
Muestra el contenido del fichero o de los ficheros que se le pasen como argumentos.

head
Muestra las **10 primeras filas** de los ficheros que se le indiquen.

Sintaxis:

head [opciones] fichero/s

Opciones:
-n | --lines=N → muestra las **n primeras líneas**, en vez de las 10 primeras.

Argumento/s:
Muestra las primeras líneas de los ficheros que se le pasen como argumentos.

tail
Muestra las **10 últimas filas** de los ficheros que se le indiquen.

Sintaxis:
tail [opciones] fichero/s

Opciones:
-n | --lines=N → muestra las **n últimas líneas**, en vez de las 10 primeras.

Argumento/s:
Muestra las últimas líneas de los ficheros que se le pasen como argumentos.

wc
Muestra el número de líneas, palabras, caracteres y bytes de los ficheros que se le indiquen, o el tamaño de la línea más larga.

Sintaxis:
wc [opciones] fichero/s

Opciones:
-c | --bytes → muestra el número de bytes.
-m | --chars → muestra el número de caracteres.
-l | --lines → muestra el número de líneas.
-w | --words → muestra el número de palabras.
-L | --max-line-length → muestra el tamaño de la línea más larga del fichero.

Argumento/s:
Muestra las últimas líneas de los ficheros que se le pasen como argumentos.

more
Muestra el contenido de los ficheros pero de forma paginada, es decir, pantalla a pantalla.

A diferencia del comando cat, cuando muestre el contenido del fichero, si este ocupa más de una pantalla, se quedará esperando que se pulse una tecla. Si es la **barra espaciadora**, mostrará la página siguiente, la tecla **Enter** muestra la línea siguiente y la **letra “q”** finaliza la ejecución.

Sintaxis:
more fichero/s

Argumento:
Muestra el contenido del fichero o ficheros que se le pasen como argumentos.

less

Muestra el contenido de los ficheros de la misma forma que **more**, con la diferencia de que podremos movernos por ellos **utilizando también las flechas de cursor**.

Sintaxis:
less fichero/s

Argumento:
Muestra el contenido del fichero o ficheros que se le pasen como argumentos.

sort
Muestra en orden ascendente el contenido de los ficheros que se les pasa como argumentos. Además, lo podremos usar para concatenar ficheros de texto utilizando los redireccionamientos y tuberías.

Sintaxis:
sort [opciones] [fichero/s]

Opciones:
-c | --check → comprueba si el fichero esté ordenado, pero **no lo ordena**. Si no está ordenado te muestra un mensaje indicando la primera línea que está fuera del orden.
-r | --reverse → ordena en sentido inverso.
-m | --merge → mezcla ficheros ya ordenados, no ordena.
-u | --unique → elimina las líneas repetidas.

Argumento/s:
Los argumentos serán ficheros. Dependiendo de la opción que elijamos, ordenará cada fichero, comprobará si están ordenados o los mezclará.

ln
Crea un enlace al fichero o al directorio que se le especifique. Si es a un directorio el enlace deberá ser simbólico.

Sintaxis 1:
ln [-s] argumento [enlace]

Argumento:
Será el fichero o directorio al que le vamos a crear un enlace.

Sintaxis 2:
ln [-s] fichero/s directorio

Sintaxis 3:
ln -t directorio fichero/s

Opciones:
-s | --symbolic → crea un enlace simbólico en vez de duro.
-t | --target-directory=directorio → especifica el directorio donde se van a crear los enlaces.

cut
Muestra solo ciertas líneas verticales de los ficheros que se le pasen como argumento.

Sintaxis:
cut [opciones] [fichero/s]

Opciones:
-b | --bytes=lista → muestra solo los bytes que se le especifiquen.
-c | --characters=lista → muestra solo los caracteres que se le especifiquen.
-d | --delimiter=delim → usa el carácter que se le especifique como delimitador en vez del tabulador.
-f | --fields=lista → muestra solo los campos que se le indiquen en lista. Puede ser un campo, una serie de campos separados por comas, y un rango.
--s | --only-delimited → no muestra las líneas que no contengan el delimitador.
--output-delimiter=cadena → usa la cadena como delimitador de salida en vez de usar el de entrada.

grep
Muestra las líneas de un fichero dado que coinciden con un cier to patrón

Sintaxis:
grep [opciones] patrón [fichero/s]
Patrón es una expresión regular.

Opciones:
-r | -R | --recursive → para buscar de forma recursiva dentro de los ficheros de un directorio.
-n | --line-number → muestra el número del lugar que ocupa en el fichero la línea encontrada.
-i | --ignore-case → no distingue entre mayúsculas y minúsculas.
-v | --invert-match → muestra las líneas que no se corresponden con el patrón.
-w | --word-regexp → el patrón debe aparecer como palabra completa y no como parte de otra palabra.
-c | --count → escribe el número de líneas que satisfacen la condición.
-l | --files-with-matches → se escriben los nombres de los ficheros que contienen líneas buscadas.

Patrón:
texto → líneas que contengan la cadena “texto”.
^texto → líneas que empiezan por “texto”.
^[^texto] → líneas que no empiezan por “texto”.
texto$ → líneas que terminen en “texto”.

Argumento/s:
Son los ficheros donde vamos a buscar las líneas que coincidan con el patrón.

Las expresiones regulares se utilizan para enumerar un conjunto de cadenas de caracteres. Utilizan ciertos caracteres que tienen un significado especial, como:

. → cualquier carácter menos salto de línea.
^ → inicio cadena o línea, con los corchetes, indican línea que no contenga la cadena.
$ → inal de cadena o línea.
() y **[]** → agrupan caracteres 7 el carácter que le procede es opcional.
| → elección entre varias opciones.
***** → cadena que se repite 0 o más veces.
+ → cadena que se repite 1 o más veces.
{n} → el carácter que le precede se repite n veces.

whereis
Localiza los ficheros ejecutables o binarios, las fuentes y las páginas del manual correspondiente a los comandos o programas instalados que se le pasen como argumento.

Sintaxis:
whereis argumento/s

Argumento/s:
Puede ser uno o varios nombres de comandos o programas.

which
Muestra la ruta absoluta del archivo del comando o de los co mandos que se le pasen como argumento.

Sintaxis;
which argumento/s

Argumento/s:
Puede ser uno o varios nombres de comandos o programas.

locate
Busca archivos dentro del sistema de archivos. Solo puede hacer búsquedas por nombre de archivo. Es muy rápido porque busca en una base de datos propia que se va actualizando periódicamente.

El comando locate tiene la desventaja de que si hacemos el cambio en el sistema de archivos y realizamos la búsqueda antes de que la base de datos se actualice, no lo encuentra. Para ello hay que ejecutar el comando **updatedb**, que sólo lo puede hacer quien tenga permisos de superusuario o administrador del sistema. Muchas distribuciones lo ejecutan periódicamente usando el comando **cron**.

Sintaxis:
locate fichero

find
Busca ficheros en un árbol de directorios. Muestra el nombre de los archivos encontrados que se correspondan con cierto conjunto de criterios. Utiliza la tabla de i-nodos para realizar sus búsquedas.

Sintaxis:
find [opciones] [directorios] [criterios] [acción] \;

Opciones:
-follow | -L → sigue los enlaces simbólicos si apuntan a directorios.

Criterios:
-type tipo → busca archivos de un tipo dado (**f** regular, **d** directorio, **l** enlace simbólico)
-name nombre → encuentra los archivos cuyo nombre coincida con el nombre que se da (**-iname** para que no distinga entre mayúsculas y minúsculas). Los criterios se pueden combinar con **-a**, **-o**, **-not**.
-maxdepth n → nivel máximo de subdirectorios a los que desciende buscando la información.
-inum n → busca los ficheros que tengan el i-nodo n.

Acciones:
-exec comando ; → ejecuta un comando sobre cada archivo encontrado. La posición del archivo se indica con **{ }** y el comando finaliza con **;** protegido por el **carácter de escape** \ para que el shell no lo interprete.

En Linux los **comandos devuelven un número al sistema para indicar si ha habido error en su ejecución**. El comando **sort**, por ejemplo, **devuelve 0 si el fichero está ordenado y 1 si no lo está**. Para comprobarlo, usamos **echo $?** que muestra el resultado del último comando ejecutado.

Buscamos en el directorio actual todos los ficheros con **extensión .doc o con extensión .txt** y los movemos al directorio **/A**.

```
paco@ubuntupaco:~ $ find -name '*.doc' -o -name '*.txt' exec mv {} /A \;
```

4. Filtros o tuberías

Las tuberías o filtros se utilizan en una línea de comandos para conectar la salida estándar de un comando con la entrada estándar de otro. Para ello se utiliza el carácter "|".

Mostramos por pantalla el i-nodo de los ficheros de nuestro directorio personal, con el tipo de fichero que es, los permisos, y el nombre del propietario. No tiene que aparecer ninguna información más, ni el nombre.

```
paco@ubuntupaco:~ $ ls -li | cut -d " " -f 1,2,4
```

5. Redireccionamientos

Cualquier proceso tiene una **entrada estándar**, **stdin**, y **dos salidas**, la **salida estándar**, **stdout**, y la **salida de errores**, **stderr**. Normalmente, la entrada estándar, **stdin**, es el **teclado** y la salida estándar, **stdout**, es la **pantalla**, pero si se produjo un **error** en la ejecución del proceso la salida por **pantalla** corresponderá a la salida de errores, **stderr**.

Sin embargo, puede que queramos cambiar la entrada estándar, la salida estándar o la salida de errores por un **fichero**, por lo que habrá que usar los redireccionamientos, con los siguientes caracteres:

- **<** Redirecciona la entrada estándar sustituyéndola por el archivo que se le indique.
- **>** Redirecciona la salida de un proceso al fichero que se le indique, borrando la información que el fichero contenía.
- **>>** Redirecciona la salida de un proceso al fichero que se le indique, pero añadiendo al final de la información del fichero la salida, con lo que no se borra el contenido del fichero.
- **2>** Redirecciona la salida de errores de un proceso al fichero que se le indique, borrando la información que el fichero contenía.
- **2>>** Redirecciona la salida de errores de un proceso al fichero que se le indique, pero añadiendo al final de la información del fichero la salida, con lo que no se borra el contenido del fichero.

En Linux, los caracteres **&>** redireccionan a un fichero tanto la salida estándar como la de errores. Los caracteres **2>&1** redirecciona la salida de errores a donde vaya la salida estándar.

En todos los casos, si el fichero indicado no existe, se crea. El número 0 indica entrada estándar, el número 1 indica salida estándar y el número 2 indica salida de error estándar, que es el único que es obligatorio escribir. Con los **redireccionamientos** y los comandos **cat** y **sort** podemos **crear** y **concatenar ficheros**.

Comandos relacionados con el redireccionamiento

Además de los caracteres vistos hasta ahora, existe un comando que también podemos utilizar para redireccionar la salida estándar hacia un fichero. La diferencia con el comando es que este además de **redireccionar al fichero también muestra la información en la salida estándar**, que normalmente es la pantalla.

tee
Lee de la entrada estándar y escribe en la salida estándar y en un fichero que se le especifique.

Sintaxis:
tee [opciones] [ficheros]

Opciones:
-a | --append → escribe al final del contenido fichero, no lo sobreescribe.
-i | --ignore-interrupts → ignora señales de interrupción.

Creamos un archivo llamado **nuevo.txt** con el comando **cat**. Escribimos en él varias palabras, una debajo de otra: **zapato**, **cuchillo**, **perro**.

```
paco@ubuntupaco:~ $ cat > nuevo.txt
zapato
cuchillo
perro
CTRL+D
```

Creamos otro archivo, **nuevord.txt** con el comando **sort**. Escribimos en él las mismas palabras del ejemplo anterior, una debajo de otra.

```
paco@ubuntupaco:~ $ sort > nuevord.txt
zapato
cuchillo
perro
CTRL+D
```

Comprobamos si **nuevo.txt** y **nuevord.txt** están ordenados.

```
paco@ubuntupaco:~ $ sort -c nuevo txt
paco@ubuntupaco:~ $ sort -c nuevord.txt
```

Mostramos el contenido del directorio raíz en formato largo y redireccionamos la salida del comando a un fichero de nuestro directorio personal llamado **inicio.txt**.

```
paco@ubuntupaco:~ $ ls -l / > inicio.txt
```

Intentamos mostrar información de un **fichero llamado tttt** en el directorio raíz. Como no existe, mostrará un error. Redireccionamos la salida de errores a un fichero llamado **error log**.

```
paco@ubuntupaco:~ $ ls /tttt
paco@ubuntupaco:~ $ ls /tttt 2> error.log
```

Buscamos todos los enlaces duros a un fichero. Redireccionamos la salida estándar a un fichero llamado **soluc** y la salida de errores a un fichero llamado **solerror**. Hay que tener cuidado por si tiene varias soluciones, para que no borre el contenido de los archivos.

```
paco@ubuntupaco:~ $ find / -inum número >> soluc 2>> solerror
```

6. Archivos especiales

Los archivos especiales tienen relación con las entradas y salidas (E/S). En Linux, las **entradas y salidas** sobre un dispositivo se hacen mediante los **archivos situados en el directorio /dev**. Cada uno de ellos se

identifica por un nombre que indica de qué tipo de dispositivo se trata. Existen diversos tipos de archivos especiales. Si queremos ver de qué tipo de archivo se trata, podríamos mirar la salida del comando ls -l, donde el primer carácter de cada línea nos indicaría el tipo de archivo:

Carácter	Tipo de dispositivo	Función
c	de caracteres	Se utilizan para los dispositivos de E/S de caracteres, como terminales o impresoras.
b	de bloques	Se utilizan para los dispositivos de bloques, como los discos.
s	sockets	Se utilizan para la comunicación de procesos a través de la red.
t	tuberías (pipes o fifo)	Se utilizan para comunicación entre procesos. Almacenan la información que se mandan entre sí.

En el **directorio /dev** encontramos la mayoría de los archivos especiales. Veremos los más utilizados, pero además de estos, hay muchos más y otros preparados para cuando se instalen nuevos dispositivos.

Dentro del **directorio /dev** tenemos **archivos especiales de caracteres**, **tty**, que hacen referencia a las **terminales virtuales** con las que podemos trabajar en el equipo. Tenemos seis terminales virtuales en modo texto, **/dev/tty1 a /dev/tty6** y una terminal en **modo gráfico**, **/dev/tty7**. Para cambiar de unos a otros, tendremos que pulsar las combinaciones de teclas CTRL+ALT+F1 a CTRL+ALT+F7, respectivamente, aunque existen más terminales en el directorio por si la necesita.

En el directorio **/etc/event.d están los ficheros tty1 a tty7**, que indican en qué niveles de arranque se iniciarán los terminales Si marcamos como comentario todas las líneas de un fichero, con el carácter # como inicio de línea deshabilitaremos la terminal virtual de ese fichero. Para que surta efecto, habrá que reiniciar el sistema. Después podemos probar que el terminal virtual no está activada pulsando la combinación de teclas CTRL+ALT+F1 a CTRL+ALT+F6, dependiendo de qué fichero sea el que hayamos modificado.

Los archivos especiales de caracteres que hacen referencia a las pseudos terminales se encuentran en el **directorio /dev/pts**, con los **nombres de 0, 1, 2**,..., dependiendo de las que tengamos abiertas. Las pseudos terminales, son procesos que emulan terminales en modo texto. En GNOME se pueden abrir mediante la aplicación **Terminal** en **Aplicaciones**.

Dentro de los dispositivos de bloques, tenemos los que hacen referencia a los **dispositivos de almacenamiento como los discos duros o las memorias de pendrives**, **/dev/sda1**, hace referencia a la **primera partición del primer disco duro**, /dev/sda2 a la segunda partición. Los dispositivos /dev/sdb1, hacen referencia al primer pendrive o a la primera partición del segundo disco duro, dependiendo de lo que tengamos instalado. Si tenemos dos discos duros, las memorias pendrives conectadas se referenciarán con el nombre /dev/sdc1, /dev/sdc2 y sucesivamente.

Las unidades de CD, CD-RW, DVD se referencian mediante los archivos especiales de bloques **/dev/sro**, **/dev/sr1**, dependiendo de los que tengamos instalados.

Un dispositivo especial de caracteres es el **dispositivo nulo /dev/null**, que se utiliza para enviarle cualquier información que no queremos utilizar, por lo que al enviarla al dispositivo nulo se perderá.

Intentamos ver el significado del **primer carácter de la salida de ls -l** con ejemplos. Utilizamos el comando **file** para ver la misma información, aunque de forma diferente.
El **primer carácter de la salida de ls -l nos indica el tipo de fichero que es**. Veamos los tipos posibles:

b Se trata de un dispositivo que se accede por **bloques**. Por ejemplo, un **disco**, el cual su acceso es por sectores.

```
paco@ubuntupaco:~ $ ls -l /dev/sda1
```

```
paco@ubuntupaco:~ $ file /dev/sda1
```

c Dispositivo que se accede por **caracteres**. Por ejemplo, el dispositivo especial **/dev/null** o una terminal.

```
paco@ubuntupaco:~ $ ls -l /dev/null
paco@ubuntupaco:~ $ file /dev/null
```

d Se trata de un **directorio**. Por ejemplo, /home/usuario, /etc, **/home**...

```
paco@ubuntupaco:~ $ ls -ld /etc
paco@ubuntupaco:~ $ file /etc
```

l Indica que se trata de un **enlace simbólico**.

```
paco@ubuntupaco:~ $ ls -l /cdrom
paco@ubuntupaco:~ $ file/cdrom
```

s Un socket o un **sistema de comunicación**.

```
paco@ubuntupaco:~ $ ls -l /dev/log
paco@ubuntupaco:~ $ file /dev/log
```

p Indica que se trata de una **tubería** (**pipe**). Permite la **comunicación entre procesos**.

```
paco@ubuntupaco:~ $ ls -l /dev/xconsole
paco@ubuntupaco:~ $ file /dev/xconsole
```

- Indica un **fichero regular**, es decir, un **fichero normal**.

```
paco@ubuntupaco:~ $ ls -l texto.txt
paco@ubuntupaco:~ $ file texto.txt
```

No sabemos dónde hay un fichero de tipo **socket**, así que buscamos por todo al árbol de directorios, buscando desde el directorio raíz. Realizamos lo mismo para los ficheros de tipo **tubería o pipe**, pero evitando que los errores salgan por pantalla, redireccionándolos al dispositivo nulo.

```
paco@ubuntupaco:~ $ ls -lR / | grep "^g"
paco@ubuntupaco:~ $ ls -lR / 2> /dev/null | grep "^p"
```