GESTIÓN DE LA INFORMACIÓN

LA CONSOLA DE COMANDOS

1. DIRECTORIOS Y FICHEROS

Un **directorio** es un fichero que contiene información sobre otros directorios y ficheros ordinarios. Se trata de una herramienta de organización de la información en el disco. Un directorio es similar a una carpeta que permite guardar en su interior documentos (ficheros) y nuevas carpetas (subdirectorios).

El directorio principal se denomina **raíz** (de la estructura en árbol). Al formatear cualquier volumen automáticamente se crea el directorio raíz, que se designa con el símbolo de la barra oblicua invertida \. De ese directorio cuelgan, a distintos niveles, los restantes directorios y subdirectorios[1].

Cada directorio dispone de una serie de entradas, cada una de las cuales hace referencia a un fichero incluido en ese directorio. Cada una de estas recoge, entre otros, los siguientes datos:

- Nombre del archivo
- Extensión (en Windows)
- Atributos del archivo
- Fecha y hora de creación
- Fecha y hora de la última modificación
- Fecha y hora del último acceso
- Cluster inicial del fichero
- Tamaño del fichero (en bytes)

El número de entradas que puede contener el directorio raíz de un disco depende de la estructura del sistema de archivos[2] . Sin embargo, el número de entradas incluidas dentro de un subdirectorio depende sólo de la capacidad de almacenamiento del disco.

Todos los directorios tienen dos entradas especiales reservadas y asignadas por defecto. Se designan con los símbolos . (punto) y .. (punto-punto). La primera de ellas, representada por un único punto, hace referencia al propio directorio, y la segunda al directorio superior o directorio padre del actual[3].

Se llama **directorio activo o actual** a aquel en el que se encuentra situado el usuario y sobre el que se ejecutan todas las operaciones y mandatos por defecto. Es la carpeta que se encuentra abierta, lo que nos permite acceder directamente a los archivos que contiene esa carpeta.

El **directorio padre** de un directorio es aquel que se encuentra en el nivel inmediatamente superior al directorio referenciado. El directorio raíz no tiene directorio padre.

Un **directorio hijo** es cualquiera que esté contenido en otro directorio.

Un **fichero** es la unidad de información que el ordenador almacena en conjunto. En definitiva, se trata de un conjunto de datos organizados y relacionados bajo un mismo nombre.

Así, por ejemplo, podemos crear un fichero de clientes que contenga la información referente a cada uno de los clientes de nuestra empresa (nombre, dirección, etc.); de la misma forma, podemos crear documentos de texto o de cálculo (cartas, informes, hoja de cálculo, etc.) y almacenarlos en ficheros sobre los que posteriormente realizaremos diferentes operaciones (recuperar, modificar, imprimir...).

La información que forma parte de un fichero está siempre codificada, de modo que lo que se almacena son bytes cuyo significado depende del modo en el que se trate esa información. Es el

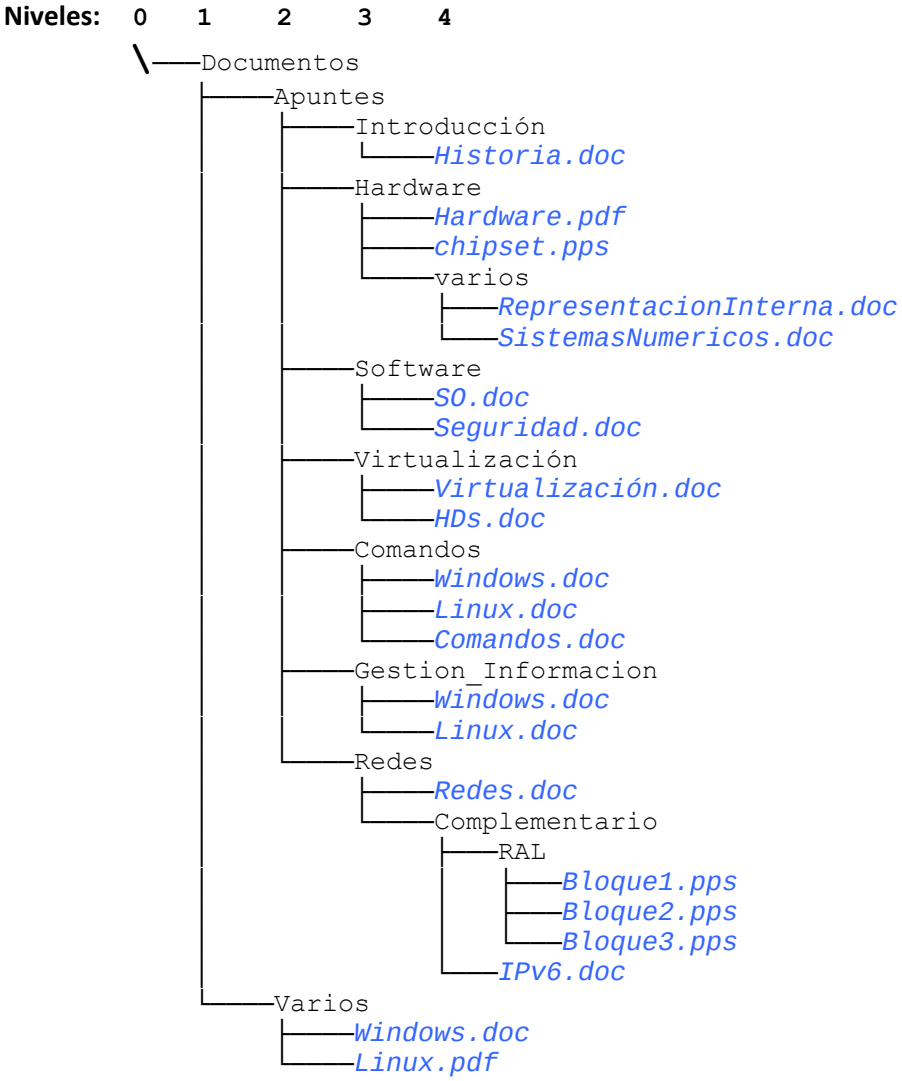
[1] Cada unidad tiene su propia estructura arbórea de directorios. Si tenemos un sistema con tres unidades, tendremos 3 directorios raíz.

[2] Si el FS es FAT32 está limitado a lo que ocupa un bloque de datos (normalmente 4 KB). Si es NTFS o ext de linux no hay límite.

[3] Excepto en el raíz que hace referencia a si mismo

programa o la aplicación que genera los datos la que permite interpretar los códigos almacenados en los archivos y traducirla a información inteligible para el usuario.

Las operaciones básicas sobre un fichero consisten: crear, modificar, eliminar, cambiarle el nombre, mover y copiar.



Estructura en árbol de la organización de la información en un disco
(en negro directorios y en azul archivos)

1.1.1 RUTA DE ACCESO

Definir una **ruta de acceso** o trayectoria de directorios no es más que seguir el camino de directorio y subdirectorios que se deben recorrer hasta llegar al elemento (fichero o directorio) deseado. Para que una ruta esté completa, debe especificarse los diferentes directorios, en orden secuencial, por los que se debe "caminar" hasta acceder al fichero destino. Para separar y diferenciar entre directorios, subdirectorios y ficheros, utilizamos el carácter especial `\`.

```
[UNIDAD:]\DIR-1\DIR-2\...\DIR-N\FICHERO.XXX
```

Donde, en Windows, `[UNIDAD:]` indica la unidad de disco en la que se encuentra el archivo `FICHERO.XXX`, siendo opcional si nos referimos a la unidad actual.

`DIR-1\DIR-2\...\DIR-N` representa a los directorios que hay que abrir sucesivamente hasta llegar al directorio que contiene el fichero citado.

En resumen, la ruta es el identificador único (nombre de pila, apellidos) de un archivo. Podemos tener varios archivos `WINDOWS.DOC`, pero solo un

```
\DOCUMENTOS\APUNTES\COMANDOS\WINDOWS.DOC
```

aunque en Windows podemos tener otro `\DOCUMENTOS\APUNTES\COMANDOS\WINDOWS.DOC` en otra unidad. En este caso y para distinguirlos, debemos recurrir a su "DNI" que es la unidad

```
D:\DOCUMENTOS\APUNTES\COMANDOS\WINDOWS.DOC
```

La ruta de acceso a un archivo forma parte del nombre del propio archivo cuando se hace referencia a él desde un directorio diferente al que lo contiene. Sin embrago, no todas las rutas siempre son completas. Cuando el usuario trabaja con la estructura de árbol de un disco, tiene una unidad y directorios activos que son precisamente en los que se encuentra trabajando.

Desde este nuevo punto de vista, muchas veces no es necesario definir una ruta o trayectoria desde su inicio, sino desde la unidad y directorio activos en los que estemos situados. Así distinguimos:

- **Rutas completas o absolutas** son aquellas que parten desde el directorio raíz independientemente de la unidad y directorio activo en los que el usuario se encuentre.

  `D:\DOCUMENTOS\APUNTES\COMANDOS\WINDOWS.DOC` Windows

  `\DOCUMENTOS\APUNTES\COMANDOS\WINDOWS.DOC` Windows y Linux

- **Rutas incompletas o relativas**. Aquellas que en su definición tienen en cuenta la posición del usuario, es decir, la unidad y directorio activo. Si el directorio actual es `\DOCUMENTOS\APUNTES` usaremos:

  `COMANDOS\WINDOWS.DOC`

Ejercicios:

Realiza los ejercicios del apartado "Manejo de rutas" de la página 27

2. WINDOWS

2.1. TIPOS DE FICHEROS

A la hora de ser nombrados, los ficheros están sometidos a una serie de reglas de designación. En general, todo nombre de fichero tiene los principales elementos:

- El nombre del fichero propiamente dicho, con el que designamos la información que contiene. Es recomendable que éste haga alusión a su contenido, de manera que al observar dicho nombre podamos rápidamente orientarnos sobre su información[4].
- El separador de nombre, que viene dado por un punto (`.`).
- La extensión va a continuación del separador del nombre y sirve para identificar los tipos de ficheros (apellido del fichero). Este elemento suele tener una longitud máxima de tres caracteres y puede ser incluso omitido.

De acuerdo con el tipo de información almacenada en los ficheros, estos pueden clasificarse en las siguientes extensiones:

- **EXE o COM**: Ficheros de programas, en los cuales los bytes que contienen son códigos de instrucción y direcciones de memoria que, una vez cargados en memoria, pueden ser interpretados por la CPU del ordenador, ya que se corresponde con códigos del conjunto de instrucciones ejecutables por el procesador.
- **TXT**: Ficheros de texto sin formato (texto plano o ficheros ASCII), son los ficheros en los que se guarda sólo texto, es decir, los códigos equivalen a un carácter imprimible.

[4] Si se trata de ficheros de datos que van a ser manipulados a través de programas (bases de datos de SQL, imágenes, etc.) conviene no usar espacios en blanco.

- **SYS**: Ficheros del sistema, que son los ficheros que contienen información acerca del hardware de equipo. Suelen ser programas en un formato especial establecido por el propio sistema operativo. Se utilizan para gestionar los elementos hardware del ordenador. Se les llama controladores de dispositivos.
- **BAT O CMD**: Ficheros de procesamiento por lotes, son ficheros de texto plano cuyo contenido son listas o relaciones de comandos `CMD` (uno por línea) y pueden ser **ejecutados** por el sistema realizando las operaciones indicadas en cada una de sus líneas. Tienen, por tanto, un tratamiento especial por parte del DOS.
- **Ficheros de datos especiales**, son los ficheros que producen los programas, en los que la información guardada sólo puede ser interpretada con ayuda del programa que creó el fichero. Es decir, son los datos del usuario pero en un formato irreconocible fuera del entorno del programa en el que fueron realizados y sus extensiones pueden ser, entre otras muchas:
  - **BKF**: identifica ficheros que son copias de seguridad.
  - **DOC**: ficheros que contienen un documento (se utiliza por omisión en WORD).
  - **MDB**: ficheros de base de datos en ACCESS.
  - **XLS**: ficheros de hoja de cálculo de EXCEL.

2.1.1 REGLAS PARA LA FORMACIÓN DE NOMBRES DE FICHEROS

El nombre del fichero puede ser introducido tanto en mayúsculas como en minúsculas.

Dentro del nombre del archivo no pueden incluirse ninguno de los símbolos siguientes (símbolos especiales)**:**

```
| \/ : * " ? < >
```

2.1.2 COMODINES

Existen dos caracteres muy especiales entre los símbolos que antes especificábamos. Estos caracteres se denominan comodines. Son `?` y `*`:

- Interrogación cerrada (`?`) .Este símbolo se utiliza para sustituir a un carácter.
- Asterisco (`*`). Este símbolo se emplea para sustituir una cadena de caracteres.

Los comodines nos permiten reunir varios ficheros que cumplen una determinada condición (cumplen un **patrón**). Supongamos que en un directorio existen diferentes ficheros y deseamos que aparezca una lista de todos los ficheros que se encuentran en él y que tengan extensión `.TXT`, escribiríamos:

```
>DIR directorio\*.TXT
```

Si usáramos el comando:

```
>DIR directorio\EX???.*
```

Visualizaríamos una lista de todos los ficheros que se encuentren en el directorio cuyo nombre tenga una longitud total de 5 caracteres, los dos primeros sean `EX` y sea cual sea su extensión longitud o tamaño.

Los caracteres comodín se pueden utilizar para copiar, renombrar o borrar ficheros.

2.2. INTRODUCCIÓN AL SHELL DE COMANDOS

El shell de comandos es un programa de software independiente que proporciona comunicación directa entre el usuario y el sistema operativo. La interfaz de usuario del shell de comandos no es gráfica y proporciona el entorno en que se ejecutan aplicaciones y utilidades basadas en caracteres. El shell de comandos ejecuta programas y muestra su resultado en pantalla mediante caracteres individuales similares al intérprete de comandos de MS-DOS `COMMAND.COM`. El shell de comandos de Windows XP utiliza el intérprete de comandos `CMD.EXE`, que carga aplicaciones y dirige el

flujo de información entre aplicaciones, para traducir los datos de entrada del usuario a un formato que el sistema operativo entiende.

Se puede usar el shell de comandos para crear y modificar archivos por lotes (también llamados secuencias de comandos) para automatizar tareas rutinarias. Por ejemplo, se pueden usar secuencias de comandos para automatizar la administración de cuentas de usuario o las copias de seguridad nocturnas. Se pueden realizar operaciones con más eficacia utilizando archivos por lotes que con la interfaz de usuario. Los archivos por lotes aceptan todos los comandos disponibles en la línea de comandos.

Se puede personalizar la ventana del símbolo del sistema para visualizarla con más facilidad y aumentar el control sobre la forma en que ejecuta los programas.

Para trabajar con pantalla completa, una vez abierta la ventana de la shell (inicio→ejecutar→CMD) hay que hacer clic con el botón derecho en la barra de nombre (azul, arriba) y seleccionar Propiedades. En la ficha *Opciones*, marcar la casilla Ventana completa y en la ficha *Fuente*, en *Tamaño*, marcar 16x20. Tras pulsar Aplicar, nos preguntará si deseamos esa configuración para la ventana actual o cada vez que la abramos.

Para ver la interfaz gráfica, pulsar la tecla de Windows y para terminar la sesión, teclear EXIT.

2.2.1 UTILIZAR VARIABLES DE ENTORNO CON CMD

El entorno del shell de comandos CMD se define mediante variables que determinan el comportamiento del shell de comandos y del sistema operativo. Puede definir el comportamiento del entorno del shell de comandos o de todo el entorno del sistema operativo si utiliza dos tipos de variables de entorno, de sistema y locales. Las variables de entorno de sistema definen el comportamiento del entorno global del sistema operativo. Las variables de entorno locales definen el comportamiento del entorno de la instancia actual de CMD.

Las variables de entorno del sistema están preestablecidas en el sistema operativo y están disponibles para todos los procesos de Windows XP. Solamente los usuarios con privilegios de administrador pueden cambiar variables del sistema. Estas variables se utilizan normalmente en secuencias de comandos de inicio de sesión.

Las variables de entorno locales sólo están disponibles cuando el usuario para el que se crearon ha iniciado sesión en el equipo. Las variables locales establecidas en la sección **HKEY_CURRENT_USER** sólo son válidas para el usuario actual, pero definen el comportamiento del entorno global del sistema operativo.

En el shell de comandos, cada instancia de CMD hereda el entorno de su aplicación principal. Por lo tanto, puede cambiar las variables en el nuevo entorno de CMD sin que afecte al entorno de la aplicación principal.

La tabla siguiente enumera algunas de las variables de entorno del sistema y locales de Windows XP.

Variable	Tipo	Contenido
%CD%	Local	cadena del directorio actual.
%COMPUTERNAME%	Sistema	nombre del equipo.
%DATE%	Sistema	fecha actual. Utiliza el mismo formato que el comando date /t generado por cdm
%ERRORLEVEL%	Sistema	código de error del último comando utilizado. Usualmente, los valores distintos de cero indican que se ha producido un error.
%HOMEDRIVE%	Sistema	letra de unidad de la estación de trabajo local del usuario conectada al directorio particular del usuario. Se establece según el valor del directorio particular del usuario que se especifica en *Usuarios y grupos locales.*
%HOMEPATH%	Sistema	ruta de acceso completa del directorio particular del usuario. Se establece según el valor del directorio particular del usuario que se especifica en *Usuarios y grupos locales*.
%PATH%	Sistema	especifica la ruta de acceso de búsqueda para los archivos ejecutables.

Variable	Tipo	Contenido
%PROMPT%	Local	configuración del símbolo del sistema del intérprete actual. Generado por CMD.
%RANDOM%	Sistema	número decimal aleatorio entre 0 y 32767. Generado por CMD.
%SYSTEMROOT%	Sistema	ubicación del directorio raíz de Windows XP.
%TEMP% y %TMP%	Sistema y usuario	directorios temporales predeterminados que utilizan las aplicaciones disponibles para los usuarios conectados actualmente. Algunas aplicaciones requieren TEMP y otras requieren TMP.
%TIME%	Sistema	hora actual. Utiliza el mismo formato que el comando time /t. Generado por CMD.
%USERNAME%	Local	nombre del usuario que ha iniciado la sesión actual.
%USERPROFILE%	Local	ubicación del perfil del usuario actual.
%WINDIR%	Sistema	ubicación del directorio del sistema operativo.

Ejercicios:

Examina los contenidos de las variable anteriores en tu equipo.

¿Hay alguna variable más que te parezca interesante?

2.2.2 CONFIGURAR LAS VARIABLES DE ENTORNO

Utilice el comando SET para crear, cambiar, eliminar o mostrar variables de entorno. El comando SET altera variables solamente en el entorno de shell actual.

Para ver una variable, escriba en el símbolo del sistema:

SET nombreVariable

Para agregar una variable, escriba en el símbolo del sistema:

SET nombreVariable=*valor*

Para eliminar una variable, escriba en el símbolo del sistema:

SET nombreVariable=

Puede usar la mayor parte de los caracteres como valores de variables, incluido el espacio en blanco. Si utiliza los caracteres especiales <, >, |, &, o ^, deberán ir precedidos del carácter de escape (^) o de comillas. Si utiliza comillas, se incluyen como parte del valor debido a que todo lo que sigue al signo igual se toma como un valor. Tenga en cuenta los siguientes ejemplos:

- Para crear el valor de variable nuevo&nombre, escriba:

```
SET nombreVariable="nuevo&nombre"
```

- Si escribe SET nombreVariable=nuevo&nombre en el símbolo del sistema, aparecerá un mensaje de error similar al siguiente:

```
"'nombre' no se reconoce como comando interno o externo, un
programa o un archivo por lotes ejecutable."
```

Los nombres de variables no distinguen entre mayúsculas y minúsculas. Sin embargo, SET muestra la variable exactamente como se escribieron al crearlas. Puede combinar letras en mayúsculas y minúsculas en los nombres de variables para que el código sea más legible (por ejemplo, NombreUsuario).

Notas

- El tamaño máximo de una variable de entorno individual es de 8192 bytes.
- El tamaño máximo total para todas las variables de entorno, incluidos los nombres de variable y el signo igual, es de 65.536 KB.

2.2.3 USAR EL VALOR DE VARIABLES DE ENTORNO

Para usar los valores de las variables (sustitución de valores) en la línea de comandos o en secuencias de comandos, ponga signos de porcentaje alrededor del nombre de la variable (por ejemplo, `%nombreDeVariable%`). Al utilizar signos de porcentaje, se asegura de que `CMD` hace referencia a los valores de la variable en lugar de realizar una comparación literal relativa al nombre de la misma. Tras asignar valor a una variable, escriba el nombre de la misma entre signos de porcentaje. `CMD` reemplaza del nombre de la variable por su valor. Por ejemplo, si deseamos incorporar una ruta a la variable PATH deberemos poner:

```
PATH=%PATH%;d:\misprogramas
```

Ya que si ponemos `PATH=PATH;D:\misprogramas` el valor de `PATH` pasará a ser "PATH;D:\misprogramas" literalmente.

2.3. CMD

Inicia una nueva instancia del intérprete de comandos, `CMD.EXE`. Si se utiliza sin parámetros, `CMD` muestra información de versión y copyright de Windows XP.

Sintáxis

```
CMD [[{/c|/k}] [/s] [/q] [/d] [{/a|/u}] [/t:pf] [/e:{on|off}]
                [/f:{on|off}] [/v:{on|off}] cadena]
```

Parámetros

`/c`

Ejecuta el comando especificado mediante el parámetro cadena y, después, se detiene.

`/k`

Ejecuta el comando especificado mediante el parámetro cadena y, después, continúa.

```
CMD /k PROMPT $p
```

`/t:`*`pf`*

Establece los colores de primer plano, *p*, y de fondo, *f*. En las tablas siguientes se enumeran los dígitos hexadecimales válidos que se pueden utilizar como valores de p y f.

Valor	Color	Valor	Color	Valor	Color	Valor	Color
0	Negro	4	Rojo	8	Gris	C	Rojo claro
1	Azul	5	Púrpura	9	Azul claro	D	Púrpura claro
2	Verde	6	Amarillo	A	Verde claro	E	Amarillo claro
3	Aguamarina	7	Blanco	B	Aguamarina claro	F	Blanco brillante

`/e:on`

Habilita las extensiones de comandos.

`/e:off`

Deshabilita las extensiones de comandos.

cadena

Especifica el comando que desea ejecutar.

`/?`

Muestra Ayuda en el símbolo del sistema.

2.4. COMANDOS CMD

Se entiende por comando la orden o mandato que se teclea en la línea de comandos.

Lo que habitualmente se denomina símbolo del sistema o *prompt* es el mensaje con el que el ordenador indica al usuario que está en disposición de aceptar órdenes o comandos propios, o de ejecutar programas de utilidades y aplicaciones diseñadas para funcionar bajo el entorno de la consola.

El símbolo del sistema está formado, habitualmente, por la unidad de disco activa (letra mayúscula seguida de dos puntos) y el directorio actual, es decir, el directorio sobre el que se realizarán por defecto todas las operaciones de gestión de archivos, si la unidad es la correspondiente al disco duro `C:` y tenemos abierto como directorio actual el directorio `DOS`, que cuelga del directorio raíz, el símbolo del sistema será `C:\DOS>`. Por el contrario, si la unidad activa es la `A:` y el directorio actual en esa unidad es el directorio raíz, el símbolo del sistema será `A:\>`.

A la hora de ejecutar cualquier comando es muy importante saber cuál es la unidad de disco sobre la que el sistema realizará las operaciones por defecto (unidad actual), que sólo puede ser una de las disponibles en el ordenador. También hay que conocer cuál es el directorio sobre el que se realizan las operaciones de archivos (directorio actual), que es uno por cada unidad de disco disponible en el equipo.

Al comando pertenecen todos los caracteres teclados a continuación del símbolo hasta pulsar la tecla `Enter`, momento en el que el comando es efectivo. En un comando se pueden distinguir tres partes diferentes separadas por un espacio, que es el separador natural del sistema operativo:

- **El nombre del comando**: Es la cadena de caracteres que se escribe en primer término e indica la acción que se va a realizar. Suelen ser abreviaturas mnemotécnicas según los criterios de los diseñadores y programadores del sistema operativo (`MKDIR`, de *MaKe DIRectorty*) .
- **Los parámetros del comando**: Determinan los elementos (archivos, directorios, unidades de disco, etc.) sobre los que el comando va a actuar. Hay comandos que no precisan parámetros, otros en los que los parámetros son opcionales y otros que pueden utilizar varios, según las acciones que se manden realizar al sistema operativo (`DIR A:`).
- **Los modificadores del comando**: Se utilizan para modificar la forma en la que un comando realiza una determinada tarea. Cada comando puede tener sus propios modificadores, que suelen expresarse mediante una barra oblicua y una letra (`/X`) (`DIR A: /W`).

Hay ciertos tipos de comandos, como los comandos de lotes y los comandos de configuración que, pese a tratarse de comandos, no están disponibles en la línea de órdenes. Los comandos de lotes son aquellos que están diseñados para funcionar exclusivamente dentro de los programas de procesamientos por lotes (ficheros `.BAT` o `CMD`).

Es posible interrumpir la ejecución de un comando pulsando `Ctrl+C`.

De acuerdo con la forma de proceder del ordenador para ejecutar las operaciones indicadas por un determinado comando, los comandos pueden clasificarse en comandos internos y comandos externos:

- Los **comandos internos** residen en la memoria principal ya que forman parte del programa `CMD`, es decir, no precisan de ninguna información adicional en el disco para ejecutarse.
- Los **comandos externos** son programas que se encuentran almacenados en disco como archivos o ficheros con extensión `.EXE`, `.COM` o `.BAT`. Cuando la cadena tecleada se corresponde con el nombre (sin extensión) de uno de los programas que constituyen los comandos externos, este programa es localizado en el disco, cargado en memoria y ejecutado. La prioridad de ejecución es: `.EXE`, `.COM`, `.BAT`

2.4.1 BUSCAR AYUDA SOBRE LOS COMANDOS CMD.

Podemos obtener ayuda sobre el uso de los comandos a través de la opción *Ayuda y soporte técnico* de Windows XP o a través del comando `HELP` desde la propia consola de comandos.

El comando `HELP` proporciona información en pantalla acerca de los comandos de sistema (comandos que no sean de red). Si se utiliza sin parámetros, el comando `HELP` presenta una lista y describe brevemente cada uno de los comandos de sistema.

Existen dos maneras de obtener Ayuda en pantalla acerca de un comando. Puede escribir `HELP comando` o bien `comando /?` que es un método ligeramente más rápido.

Ejemplo: para obtener información acerca del comando `XCOPY`, escriba una de las dos posibilidades siguientes:

```
HELP XCOPY
XCOPY /?
```

Para obtener una lista de los comandos básicos de `CMD` teclee:

```
HELP
```

2.4.2 UTILIZAR LA SINTAXIS DE COMANDOS

La sintaxis aparece en el orden en que debe escribir un comando y los parámetros que lo siguen. El ejemplo siguiente del comando `XCOPY` muestra diversos formatos de la sintaxis:

Copia archivos y directorios, incluidos subdirectorios.

XCOPY *origen* [*destino*] [**/w**] [**/p**] [**/c**] [**/v**] [**/q**] [**/f**] [**/l**] [**/g**] [**/d**[:*mm-dd-aaaa*]] [**/u**] [**/i**] [**/s** [**/e**]] [**/t**] [**/k**] [**/r**] [**/h**] [{**/a**|**/m**}] [**/n**] [**/o**] [**/x**] [**/exclude:***archivo1*[**+**[*archivo2*]][**+**[*archivo3*]] [{**/y**|**/-y**}] [**/z**]

La tabla siguiente explica cómo interpretar los diferentes formatos de texto.

Formato	Significado
Cursiva	Información que debe suministrar el usuario
Negrita	Elementos que el usuario debe escribir exactamente como se muestran
Puntos suspensivos (...)	Parámetro que se puede repetir varias veces en una línea de comandos
Entre corchetes ([])	Elementos opcionales
Entre llaves ({}); las opciones se separan mediante la barra vertical (|). Ejemplo: {par|impar}	Conjunto de opciones de las que el usuario debe elegir sólo una
Fuente `Courier`	Código o salida de un programa

COMANDOS BÁSICOS

- `CLS` Borra la pantalla.
- `DATE` Muestra o establece la fecha
- `PROMPT` Cambia el símbolo de comandos de Windows.
- `TIME` Muestra o establece la hora del sistema
- `VER` Muestra la versión de Windows

COMANDOS DE FICHEROS

- `ATTRIB` muestra/modifica los atributos de los archivos
- `COPY` Copia uno o más archivos a otro lugar
- `DEL` Elimina uno o más archivos
- `EDIT` Editor de texto
- `ERASE` Elimina uno o más archivos
- `FC` Compara dos archivos o mas archivos y muestra las diferencias entre ellos

- `FIND` Busca una cadena de texto en uno o más archivos
- `MOVE` Mueve uno o más archivos de un directorio a otro en la misma unidad
- `RENAME` **o** `REN` Cambia el nombre de uno o más archivos
- `REPLACE` Reemplaza archivos
- `TYPE` Muestra el contenido de un archivo de texto

COMANDOS DE DIRECTORIOS

- `CHDIR` **o** `CD` Muestra el nombre del directorio actual o cambia a otro directorio
- `DIR` Muestra una lista de archivos y subdirectorios en un directorio
- `MKDIR` **o** `MD` Crea un directorio
- `RMDIR` **o** `RD` Elimina un directorio
- `TREE` Muestra gráficamente la estructura de directorios de una unidad
- `XCOPY` Copia archivos y árboles de directorios

COMANDOS DE MANEJO DE DISCOS

- `CHKDSK` Comprueba un disco y muestra un informe de su estado
- `DISKCOMP` Compara el contenido de dos disquetes
- `DISKCOPY` Copia el contenido de un disquete en otro
- `FORMAT` Da formato a un disco para usarse con Windows
- `LABEL` Crea, cambia o elimina la etiqueta del volumen de un disco
- `VOL` Muestra la etiqueta del volumen y el número de serie del disco

OTROS COMANDOS

- `AT` planifica comandos para ejecutarse automáticamente
- `COLOR` colores de primer y segundo plano de la pantalla
- `COMPACT` comprime/descomprime archivos y directorios
- `EXIT` cierra la consola de comandos
- `SUBST` asocia una ruta de directorio a una letra de unidad

2.4.3 REDIRECCIÓN (*STREAM*) DE COMANDOS Y TUBERIAS (*PIPELINES*)

Todos los programas que procesan datos de entrada, tienen un dispositivo de entrada asociado (**stdin**). Igualmente todos los programas que trabajan con datos de salida tienen un dispositivo de salida asociado (**stdout**). Por defecto el de entrada es el teclado y el de salida el monitor (`TYPE`, `DATE`).

En ocasiones es interesante cambiar la entrada y salida por defecto redireccionándola a otro dispositivo o fichero. Para ello el DOS soporta los siguientes símbolos:

`<` Redirección de entrada (`comando < fichero/dispositivo`)

`>` Redirección de salida (`comando > fichero/dispositivo`)

`>>` Adición a un fichero (`comando >> fichero`)

`|` pasa la salida de un comando como entrada del siguiente (`comando | comando …`)

REDIRECCIÓN DE ENTRADA <

Consiste en cambiar a otro dispositivo o fichero el dispositivo de entrada estándar de un programa (comando).

```
DATE < Fecha.txt
```

REDIRECCIÓN DE SALIDA >

Es mucho más utilizado que la redirección de entrada. Sobre todo para enviar salidas de programas a la impresora o a un fichero.

```
TYPE Carta.txt > NUL
DIR > Estruc.txt
CHKDSK C: > Estado.txt
```

La redirección de salida siempre crea un fichero nuevo si no existe y lo sobrescribirá si existe con lo que se perderá la información anterior.

ADICIÓN DE FICHEROS >>

Consiste en redirecionar la salida hacia un fichero, pero si existe no lo destruye, sino que añade la nueva información de salida al final.

```
TYPE Carta.txt >> Cartas.txt
```

TUBERÍAS O *PIPELINES*

Una tubería, representada por el carácter de barra vertical (`|`) consiste en dirigir la salida de un comando en lugar de al *stdout* al *stdin* del siguiente comando:

```
TYPE Carta.txt | MORE
```

`TYPE` mostraría por pantalla (*stdout*) el contenido de `Carta.txt` pero `|` hace que dicha salida la tome como entrada (*stdin*) el comando `MORE`. Así, esta tubería muestra por pantalla de forma paginada el contenido de `Carta.txt`.

2.4.4 USAR FILTROS

Un filtro es un comando que no modifica un archivo sino que muestra el contenido del mismo de manera distinta, por ejemplo `FIND` "extrae" las líneas de un fichero que contienen un determinado texto, pero el contenido del fichero no se ve afectado

Los comandos filtro dividen, reorganizan o extraen partes de la información que pasa a través de los mismos. En la tabla siguiente se enumeran los comandos de filtro disponibles en Windows.

Comando	Descripción
`FIND`	Busca los caracteres especificados en archivos y en la información de salida de los comandos.
`MORE`	Muestra el contenido de un archivo o la información de salida de un comando en una ventana del símbolo del sistema cada vez.
`SORT`	Ordena alfabéticamente los archivos y la información de salida de los comandos.

Para enviar la información de entrada de un archivo a un comando de filtro, utilice el signo menor que (<). Si desea que el comando de filtro obtenga la información de entrada a partir de otro comando, utilice el carácter de barra vertical (`|`).

MORE

El comando `MORE` muestra el contenido de un archivo o la información de salida de un comando en una ventana del símbolo del sistema cada vez. Por ejemplo, para mostrar el contenido de un archivo denominado Lista.txt en una ventana del símbolo del sistema cada vez, escriba:

```
MORE < lista.txt
```

Aparecerá una ventana del símbolo del sistema con información y `-- Más --` en la parte inferior de la ventana. Para pasar a la siguiente ventana del símbolo del sistema, presione cualquier tecla del teclado, excepto PAUSA. Para detener el comando sin ver más información, presione `CTRL+C`.

El comando `MORE` resulta útil si está trabajando con un comando que produce información de salida en más de una ventana del símbolo del sistema. Por ejemplo, suponga que desea ver un árbol

de directorios del disco duro. Si tiene más directorios de los que se pueden mostrar en la ventana del símbolo del sistema, puede usar el comando TREE con una barra vertical (|) y el comando MORE, como en el ejemplo siguiente:

```
TREE c:\ | MORE
```

Aparecerá la primera ventana del símbolo del sistema con la información de salida del comando TREE, seguida del símbolo -- Más --. La información de salida se detiene hasta que presione cualquier tecla del teclado, excepto PAUSA.

FIND

El comando FIND busca en los archivos la cadena o el texto que especifique. CMD muestra todas las líneas que coinciden con la cadena o el texto especificado en la ventana del símbolo del sistema. Puede usar el comando FIND como comando de filtro o como un comando estándar de Windows XP.

Para utilizar FIND como comando de filtro, debe incluir un signo menor que (<) y la cadena o el texto que desea buscar. De forma predeterminada, las búsquedas que realiza el comando FIND distinguen entre mayúsculas y minúsculas. Por ejemplo, el siguiente comando busca la cadena "Pacific Rim" en el archivo Trade.txt:

```
FIND "Pacific Rim" < Trade.txt
```

La salida no incluye las apariciones de "pacific rim". Sólo incluye las apariciones de "Pacific Rim", con las mayúsculas.

Para guardar la información de salida del comando FIND, en lugar de mostrarla en la ventana del símbolo del sistema, escriba el signo mayor que (>) y el nombre del archivo en el que se va a guardar dicha información. Por ejemplo, el siguiente comando busca la cadena "Pacific Rim" en el archivo Trade.txt y guarda los resultados en el archivo Nwtrade.txt:

```
FIND "Pacific Rim" < Trade.txt > Nwtrade.txt
```

SORT

El comando SORT ordena alfabéticamente un archivo de texto o la información de salida de un comando. Por ejemplo, el siguiente comando ordena el contenido de un archivo llamado Lista.txt y muestra los resultados en la ventana del símbolo del sistema:

```
SORT < Lista.txt
```

En este ejemplo, el comando SORT ordena las líneas del archivo Lista.txt en una lista alfabética y muestra el resultado sin cambiar el archivo. Para guardar la información de salida del comando SORT, en lugar de mostrarla en pantalla, escriba el signo mayor que (>) y un nombre de archivo. Por ejemplo, el siguiente comando ordena alfabéticamente las líneas del archivo Lista.txt y guarda el resultado en el archivo ListaAlf.txt:

```
SORT < Lista.txt > ListaAlf.txt
```

Para ordenar la información de salida de un comando, escriba el comando seguido de un carácter de barra vertical (|) y el comando SORT (es decir, comando | SORT). Por ejemplo, el siguiente comando ordena alfabéticamente las líneas que incluyen la cadena "Luisa" (es decir, la salida del comando FIND):

```
FIND "Luisa" < listacorreo.txt | SORT
```

2.4.5 SÍMBOLOS DE PROCESAMIENTO CONDICIONAL (UTILIZAR VARIOS COMANDOS)

Puede ejecutar varios comandos desde una línea de comandos o secuencia de comandos si utiliza símbolos de procesamiento condicional.

Al ejecutar varios comandos con símbolos de procesamiento condicional, los comandos que hay a la derecha del símbolo de procesamiento condicional actúan basándose en el resultado de la ejecución del comando que hay a la izquierda del símbolo de procesamiento condicional.

Por ejemplo, puede ejecutar un comando solamente si el anterior causa un error. También puede ejecutar un comando solamente si el anterior es correcto. Puede usar los caracteres especiales enumerados en la tabla siguiente para pasar varios comandos.

Carácter	Sintaxis	Definición
`& [...]`	comando1 **&** comando2	Separa distintos comandos en una misma línea de comandos. `CMD` ejecuta el primer comando y después el segundo comando.
`&& [...]`	comando1 **&&** comando2	Ejecuta comando1 y, a continuación, ejecuta comando2 solamente si el primero se completa correctamente (runlevel=0).
`|| [...]`	comando1 **||** comando2	Ejecuta comando1 y, a continuación, ejecuta comando2 solamente si comando1 no se completó correctamente (recibe un código de error mayor que cero).
`( ) [...]`	(comando1 **&** comando2)	Agrupa o anidar varios comandos.
`; o ,`	comando parámetro1**;**parámetro2	Para separar parámetros de comandos.

Notas

- Los caracteres "y" comercial, (`&`), barra vertical (`|`) y paréntesis ( ) son caracteres especiales que deben ir precedidos del carácter de escape (`^`) o de comillas cuando se pasan como argumentos.
- Si un comando completa una operación correctamente, devuelve un código de salida cero (`0`) o no devuelve ningún código de salida.

Ejemplos.

comando1 **&** comando2.

`DATE&DIR` Hace `DATE` y luego `DIR`.

comando1**&&**comando2.

`COPY \texto.txt && DIR` hace `DIR` solo si la copia es correcta. Puede ser incorrecta si `\texto.txt` no existe

comando1||comando2.

`COPY \texto.txt || DIR` hace `DIR` cuando la copia es incorrecta, si `\texto.txt` no existe

Ejercicios:

Estudia los comandos de la consola CMD. Para ello utiliza el comando `HELP` conforme vas realizando ejercicios del apartado “Ejercicios de comandos” (página 28). Describe la acción de cada uno de ellos, prueba los modificadores y recoge los más importantes en un chuletario con el siguiente formato:

Sintaxis básica del comando | para que sirve | principales modificadores

Ejemplo:

XCOPY *origen* [*destino*] | copia directorios | /S /E /I /H

3. LINUX

3.1. TIPOS DE FICHEROS

Linux no utiliza el concepto de extensión como Windows. Independientemente de si un archivo tiene o no extensión, la asociación del programa que lo abre está determinada por la información que se incluye en el i-nodo del archivo. No obstante, se puede decir que en Linux existen básicamente 4 tipos de archivos (se distinguen por el primer carácter de la cadena de permisos, que podemos ver con `ls -l` o `ls -li`[5]):

- **Archivos ordinarios** (carácter `-`). Contienen la información con la que trabaja cada usuario.
- **Enlaces simbólicos** (carácter `l`). Se utilizan para asignar un segundo nombre a un archivo. Los enlaces simbólicos se muestran en pantalla añadiendo tras el nombre a quien apuntan:
  ```
  lrw-r--r--  1  alumno16  users  4  Mar 13 19:35  enlace -> fichero.txt.
  ```
- **Directorios** (carácter `d`). Son archivos especiales que contienen referencias a otros archivos o directorios
- **Archivos especiales.** Suelen representar dispositivos físicos, como unidades de almacenamiento, impresoras, terminales, etc. En Linux, todo dispositivo físico que se conecte al ordenador está asociado a un archivo. Linux trata los archivos especiales como archivos ordinarios (carácter `b`, `c`, etc.)
  - Carácter `b`: indica que se trata de un dispositivo por bloques (disco duro, CD, etc.)
  - Carácter `c`: indica que se trata de un dispositivo de carácter (consola, impresora, etc.)
  - Carácter `s`: indica que se trata de un dispositivo orientado a *socket* (comunicación entre procesos)

  Un caso especial son los archivos virtuales que no son un dispositivo real: `/dev/null`, `/dev/random`, etc. y que no tienen un carácter especial.

3.1.1 ENLACES

ENLACES FÍSICOS O DUROS (HARD LINKS)

No es específicamente una clase de archivo sino un segundo nombre que se le da a un archivo.

Supón que dos usuarios necesitan compartir información de un mismo archivo. Si cada uno tuviera una copia del archivo se soluciona el problema, pero las modificaciones que realice un usuario no las vería el otro.

Sin embargo, si creamos un enlace duro al archivo para cada usuario cada vez que uno de ellos modifique cualquier cosa en el archivo, el otro lo podrá ver puesto que realmente están viendo y modificando el mismo archivo.

Básicamente es una nueva entrada en otro directorio, pero tanto la entrada original como la del enlace apuntan al mismo i-nodo, que a su vez apunta a los bloques de datos del archivo. Así, habrá un único i-nodo y un único conjunto de bloques de datos pero múltiples entradas de directorio. Un enlace duro no ocupa espacio físico en el disco duro ya que únicamente está la entrada de directorio. No obstante, al ejecutar un `ls` nos informa del tamaño ya que `ls` nos muestra la información contenida en el i-nodo correspondiente.

Se sabe que un archivo tiene enlaces duros por el contador de enlaces (que muestra el comando `ls -l`). Este dato nos muestra el número de entradas de directorio que tiene un archivo. Si es

[5]
```
31592    -rw-r--r--   1        alumno16   users   505    Mar 13 19:05   fichero.txt
```
i-nodo | permisos | nºenlaces | propietario | grupo | tamaño | fecha | nombre

superior a 1 el archivo está enlazado pero para poder saber cuales son sus enlaces habrá que examinar los números de i-nodo de los archivos (`ls -li`):

```
ls -liRa / | grep -w ^inodo
```

nos mostraría la ubicación de los distintos enlaces del i-nodo indicado.

Sólo se pueden hacer enlaces duros dentro del mismo volumen y sólo cuando el sistema de archivos es de tipo `ext`.

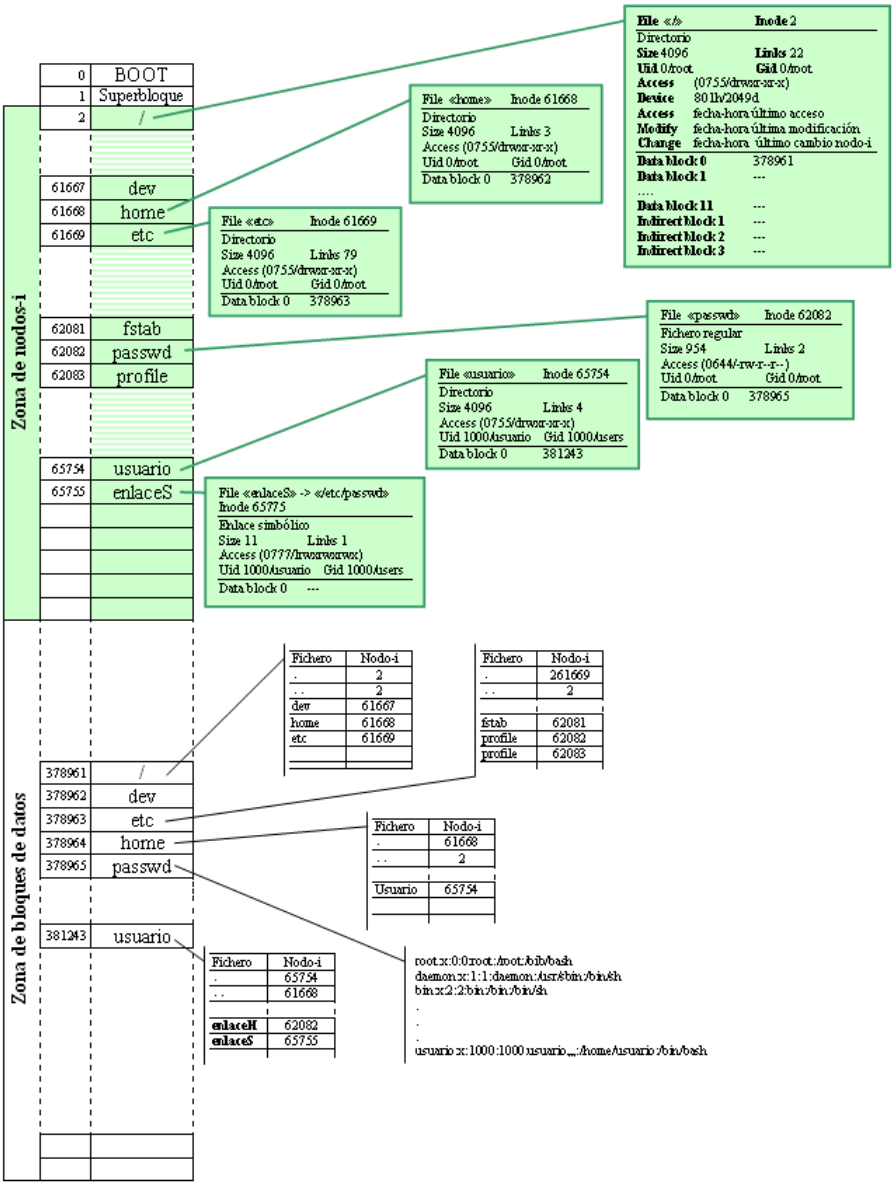
Los datos no se borrarán hasta que se hayan borrado todos y cada uno de los enlaces duros que existen. Solo se borran definitivamente los ficheros que tengan un 1 en el contador de enlaces duros.

ENLACES SIMBÓLICOS

Son el equivalente a los accesos directos de Windows. Son un fichero cuyo contenido es la ruta absoluta de otro fichero.

La diferencia con los enlaces duros es que los simbólicos hacen referencia al nombre del archivo original, mientras que los duros hacen referencia al i-nodo en el que están situados los datos del archivo original.

De esta manera, si tenemos un enlace simbólico y borramos el archivo original perderemos los datos, pero el enlace permanece aunque ya no apunta a nada.



3.1.2 REGLAS PARA LA FORMACIÓN DE NOMBRES DE FICHEROS

Linux es *case sensitive*, es decir, distingue entre mayúsculas y minúsculas por lo que podemos tener en el mismo directorio los archivos `apuntes`, `Apuntes` y `APUNTES`. Se recomienda utilizar todos en minúscula.

En cuanto a los caracteres para los nombres de archivo, se puede utilizar cualquiera, a excepción del carácter /, ya que tiene un significado especial. Es conveniente no utilizar caracteres especiales, acentos, etc. No se deben incluir espacios en blanco ni emplear los siguientes caracteres en los nombres de archivo:

```
!  #  &  (  )  `  "  ;  |  >  <  @  $  {  }  *  ?  \  Tab  +  -
```

Al contrario que en Windows, no existe un atributo de oculto. Para indicar que un fichero es oculto, se antepone al nombre un punto (`.`). Se pueden ver los archivos ocultos usando `ls -a`.

3.1.3 COMODINES

Además de los dos comodines de Windows (`*` y `?`), Linux tiene un tercer comodín que sustituye un carácter por cualquiera de los elementos indicados

- [lista]

Si usáramos el comando:

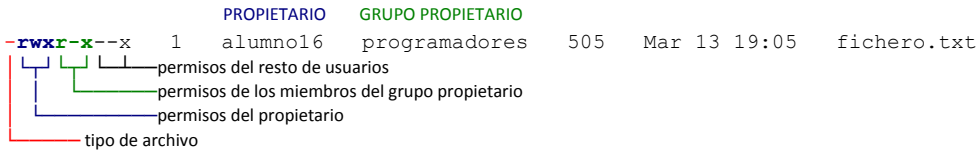
```
$ls directorio\examen[1-5]
```

Visualizaríamos una lista de todos los ficheros que se encuentren en el directorio cuyo nombre sea examen seguido de 1,2,3,4,5 y nada más. Para ver los archivos que tengan como primer carácter una letra mayúscula usaríamos el patrón `[A-Z]*`.

3.1.4 LOS PERMISOS DE LOS FICHEROS

Este es un concepto importante en Linux ya que es a través de estos permisos y la relación que el usuario tenga con ellos como se modera el acceso a los recursos en Linux[6].

Los permisos de un fichero están determinados por la cadena de 10 caracteres de los permisos, el UID del propietario y el GID del grupo propietario. Estos tres datos esta incluidos en la información del fichero que contiene el i-nodo y se pueden ver usando el comando `ls -l`:



El primer carácter de la cadena indica el tipo de archivo (ver apartado 3.1). Los tres siguientes corresponden a los que puede hacer el propietario sobre el archivo (en este ejemplo `alumno16` puede hacer todo ya que tiene `rwx`). Los tres siguientes indican lo que podrán hacer los miembros del grupo propietario (es decir, los usuarios miembros de `programadores` podrán leer y ejecutar pero no modificar ya que se ha indicado para el grupo `r-x`). Por último, el resto de usuarios del sistema (los que no pertenecen a `programadores`) solo podrán ejecutar el archivo como indica la cadena `--x`.

Existen tres tipos de permiso normal y otros tres especiales:

- **`r`** el archivo se puede leer y en los directorios indica que podemos ver su contenido por ejemplo con `ls`.
- **`w`** el archivo puede ser modificado y en los directorios indica que se pueden crear y eliminar archivos y otras carpetas dentro de él.

[6] No hay que olvidar que los dispositivos (discos duros, impresoras, etc.) se gestionan en Linux mediante los archivos especiales de dispositivo que se almacenan en `/dev`.

- **x** el archivo puede ser ejecutado. Si se trata de un directorio indica que se puede entrar el él y acceder al contenido.
- **s** en lugar de x en el propietario indica el permiso ***set-uid*** (SUID). Este hace que al ejecutarse un archivo ejecutable binario, el proceso que se genera en memoria lo hará con el UID del usuario propietario en lugar del UID del usuario que lo lanzó. Como ejemplo de este uso tenemos el comando `passwd` que puede ser ejecutado por cualquier usuario aunque solo el *root* puede hacer gestión de usuarios ya que los archivos modificados por este comando (`/etc/passwd` y `/etc/shadow`) son propiedad del *root* (aparecerá como **S** si no se ha aplicado el permiso de ejecución).

  Hay que tener precaución al usar este permiso sobre los ejecutables propiedad del *root* (comandos básicos de administración del sistema) pues puede suponer una grieta en la seguridad del sistema. Es preferible usar el comando `sudo` si necesitamos usar alguno de estos comandos.
- **s** en lugar de x en el grupo indica el permiso ***set-gid*** (SGID). Cuando se aplica a un directorio se fuerza que todos los ficheros y subdirectorios creados en él hereden su grupo(aparecerá como **S** si no se ha aplicado el permiso de ejecución).
- **t** en lugar de x en el resto de usuarios indica que se ha aplicado el ***sticky-bit*** (aparecerá como **T** si no se ha aplicado el permiso de ejecución).

  Si se aplica a un fichero ejecutable, estamos indicando que es un programa con capacidad para que varios procesos compartan su código y que este código debe mantenerse en memoria aunque alguno de los procesos que lo utilicen deje de ejecutarse. Esto permite ahorrar memoria cuando se trabaja con programas muy utilizados, como editores de texto o compiladores.

  En directorios compartidos por varios usuarios indica que sólo el propietario del archivo pueda eliminarlo del directorio. Es decir cualquier otro usuario va a poder leer el contenido de un archivo o ejecutarlo si fuera un binario, pero sólo el propietario original podrá eliminarlo o modificarlo. Si no se tuviera el *sticky-bit* activado, entonces en estas carpetas públicas, cualquiera podría eliminar o modificar los archivos de cualquier otro usuario.

Para cambiar los permisos de los ficheros se usan los siguientes comandos:

- `chown` Cambia el propietario (y el grupo).
- `chgrp` Cambia el grupo.
- `chmod` Modifica la cadena de permisos.
- `umask` Define los permisos por defecto al crear un fichero o directorio.

Además de los permisos, hay que tener en cuenta la lista de atributos. Esta se puede ver usando el comando `lsattr`:

```
su--i-dAc---- prueba.txt
```

- `-u` indica que cuando el fichero se borre, sus datos permanezcan guardados y permitan al usuario su recuperación.
- `-i` el fichero no pueda ser modificado, borrado o renombrado.
- `-A` al acceder al fichero no quedará registrada la fecha del último acceso.
- `-c` el fichero se comprime automáticamente en el disco.
- `-s` al borrar el fichero los bloques utilizados en el disco duro son escritos con ceros, de modo que los datos no se puedan recuperar por medio alguno. Es la forma más segura de eliminar datos.
- `-d` el fichero no será incluido en las copias de seguridad al utilizar el comando `dump`.

Los atributos se cambian con el comando `chattr`.

Investiga:

Para facilitar la identificación de los archivos Linux utiliza un código de colores en pantalla. Haz una relación de los distintos colores con el tipo de ficheros que representan:

Ejercicios:

Estudia los comandos de modificación de permisos y atributos. Describe la acción de cada uno de ellos, prueba los modificadores y recoge los más importantes en un chuletario con el siguiente formato:

Sintaxis básica del comando | para que sirve | principales modificadores

Ejemplo:

chown *propietario*[*:grupo*] fichero | cambia el propietario de un archivo | -Rh

3.2. INTRODUCCIÓN AL SHELL DE COMANDOS

Lo primero que hay que tener en cuenta es que en Linux se puede trabajar con múltiples shells (interpretes de comandos) cada una de las cuales admite una serie de comandos. Normalmente, la shell por defecto suele ser *bash*[7] cuyo *prompt* se presenta como `usuario@equipo:directorio$`, siendo el *prompt* propiamente dicho `$`. Para el *root*, el *prompt* se cambia por `#`:

```
alumno16@equipo:~$ [8]
root@equipo:/etc#
```

Tradicionalmente, Linux ofrece un *prompt* distinto para el usuario `root` para que se sepa visualmente que estamos trabajando como superusuario, mostrando `#` en lugar de `$`. En el caso de SuSe, además, lo muestra en color rojo.

El *prompt* genérico se configura en el fichero de configuración inicial del sistema (`/etc/bashrc`) pero se puede personalizar para cada usuario (`$HOME/.bashrc`), mediante las variables PS1 o PROMPT:

```
PS1= \u@\h:\W$    -->    usuario@equipo:directorio$
PS1=\u:\t>        -->    usuario:hora>
```

Hay que tener en cuenta que solo el *root* puede ejecutar tareas de administración del sistema ya que esta se hace mediante la manipulación de los archivos de configuración que están, normalmente, ubicados en `/etc` y solo el *root* tiene la propiedad de estos archivos.

3.2.1 UTILIZAR VARIABLES DE ENTORNO EN LA SHELL

Variable	Contenido
`$GROUPS`	Una matriz que contiene la lista de los grupos a que pertenece el usuario actual
`$EDITOR`	El editor por defecto que usarán comandos como `crontab`
`$HOME`	Directorio personal del usuario

[7] Bourne Again SHell

[8] `~` es un símbolo especial que se utiliza para referenciar al directorio *home* del usuario activo, es decir, el 6º dato del fichero `/etc/passwd` (también contenido en la variable `$HOME`). Si se teclea `cd ~` (`AltGr+4`), nos situaremos en nuestro directorio personal.

Variable	Contenido
$HOSTNAME	Nombre de la máquina.
$PAGER	Paginador por defecto.
$PATH	Lista de directorios donde buscar los programas.
$PS1	*Prompt* principal.
$PS2	El *prompt* secundario. El valor por defecto es > (solo se usa si continuamos un comando en mas de una línea mediante el uso de \)
$PWD	Directorio actual definido por el comando cd
$RANDOM	Cuando se llama esta variable un numero entero entre 0 32767 es generado
$SHELL	Intérprete de comandos por defecto.
$UID	El valor numérico real del usuario actual
$USER	Nombre del usuario.

3.2.2 CONFIGURAR LAS VARIABLES DE ENTORNO

En varias interfaces texto de Unix y Linux, como por ejemplo en bash, se muestra el valor de una variable mediante:

```
echo $PATH
```

Los comandos `env` (variables globales, accesibles por todos los usuarios conectados en ese momento) y `set` (todas la variables, tanto locales como globales) muestran las variables junto con sus respectivos valores. `env` y `set` se usan también para asignar valores a variables.

La forma de asignar un valor a una variable local es:

```
[set] variable=valor
```

Para convertir en global (y, si se desea, asignar valor) una variable se usa el comando:

```
export VARIABLE[=valor]
```

Para usar el contenido de una variable, hay que precederla del signo $ para que la shell sepa que queremos usar el contenido y no el nombre de la variable:

```
export PATH=\mnt\disco2:$PATH
```

incorpora al valor de `PATH` ya existente el directorio `\mnt\disco2`; en cambio

```
export PATH=\mnt\disco2:PATH
```

sustituye el valor de PATH por la cadena "\mnt\disco2:PATH".

Estos valores son temporales, hasta que se cierre la sesión (si es local) o se reinicie el equipo.

Para que el cambio sea permanente, debemos incluir la orden en el archivo de configuración de arranque (`/etc/profile`) o, si es para un determinado usuario, en su archivo de configuración personal (`$HOME/.bash_profile`).

3.3. COMANDOS DE LA SHELL

En Linux el símbolo del sistema o `prompt` varía según seamos un usuario normal o el *root*.

Al comando pertenecen todos los caracteres teclados a continuación del símbolo hasta pulsar la tecla `Enter`, momento en el que el comando es efectivo. En un comando se pueden distinguir tres partes diferentes separadas por un espacio, que es el separador natural del sistema operativo:

- **El nombre del comando**: La cadena de caracteres que se escribe en primer término e indica la acción que se va a realizar. Suelen ser abreviaturas mnemotécnicas según los criterios de los diseñadores y programadores del sistema operativo (`mkdir`, de *MaKe DIRectorty*).
- **Los modificadores del comando**: Se utilizan para modificar la forma en la que un comando realiza una determinada tarea. Cada comando puede tener sus propios modificadores, que suelen expresarse mediante un guión y una o mas letras (`-yX`) (`ls -laR /home`).
- **Los parámetros del comando**: Determinan los elementos (archivos, directorios, unidades de disco, etc.) sobre los que el comando va a actuar.

Es posible interrumpir la ejecución de un comando pulsando `Ctrl+C`.

También se puede pausar la ejecución pulsando `Ctrl+S`.

3.3.1 BUSCAR AYUDA SOBRE LOS COMANDOS DE LA SHELL.

Podemos obtener ayuda sobre el uso de los comandos a través del comando `man` (*man*ual page) desde la propia consola de comandos:

```
man ls
```

Dependiendo de que paginador usemos (variable de entorno PAGER) podremos movernos arriba y abajo a través del manual usando las flechas del teclado. Para buscar un texto dentro del manual teclearemos `/texto`. Para salir se pulsa la letra `Q` (*quit*).

El comando `apropos` nos muestra una lista de los comandos que contienen la palabra indicada en su página man:

```
apropos concatenar
```

Hay que tener en cuenta que muchas veces las páginas de manual están en inglés, por lo que es adecuado realizar la búsqueda tanto en español como en inglés.

3.3.2 UTILIZAR LA SINTAXIS DE COMANDOS

COMANDOS BÁSICOS

- `clear` Borra la pantalla.
- `date` Muestra o establece la fecha
- `echo` Muestra el texto por pantalla
- `export` Asigna valor a variables del sistema
- `logout` Sale de la shell
- `time` Muestra o establece la hora del sistema

COMANDOS DE FICHEROS

- `cat` Muestra el contenido de un archivo de texto
- `chattr` Muestra/modifica los atributos de los archivos
- `chmod` Muestra/modifica los permisos de los archivos
- `cp` Copia uno o más archivos a otro lugar
- `file` Determina el tipo de archivo a través del análisis parcial de su contenido
- `find` Busca el fichero especificado

- `grep` Busca una cadena de texto en uno o más archivos
- `gzip` Comprime o descomprime un archivo en formato GZip
- `ln` Crea enlaces duros o simbólicos
- `mv` Mueve o cambia el nombre de uno o más archivos
- `nano` **o** `joe` Editor de texto ASCII
- `rm` Elimina uno o más archivos
- `stat` Muestra datos del fichero indicado: nombre, tamaño, bloques, dispositivo en el que se encuentra, i-nodo, links, permisos, uid, gid, fecha de último acceso, fecha de última modificación, fecha de cambio
- `tar` Empaqueta o desempaqueta un archivo en formato .tar
- `touch` Cambia la fecha del último acceso. Si el fichero indicado no existe, crea un fichero vacío
- `type` o `which` Muestra la ubicación de un archivo señalando su "path"
- `vi` Editor de texto ASCII estándar en todas las versiones de Linux
- `whereis` Busca los archivos ejecutables, las fuentes y el manual de un comando
- `which` Busca el comando especificado

COMANDOS DE DIRECTORIOS

- `cd` Muestra el nombre del directorio actual o cambia a otro directorio
- `cp -R` Copia archivos y árboles de directorios
- `ls` Muestra una lista de archivos y subdirectorios en un directorio
- `mkdir` Crea un directorio
- `pwd` Devuelve la cadena correspondiente al directorio actual
- `rmdir` Elimina un directorio

COMANDOS DE MANEJO DE DISCOS

- `df` muestra el espacio libre en disco
- `du` muestra el espacio utilizado en disco
- `du` muestra el espacio utilizado en disco
- `mkfs` Da formato a un disco
- `mount` Monta una unidad o partición en el sistema de archivos
- `umount` desmonta una unidad del sistema de archivos

OTROS COMANDOS

- `fc` Edita el fichero del historial de comandos usados
- `free` memoria libre
- `halt` Apaga el sistema
- `history` Muestra el historial de comandos utilizados
- `lp` Imprime un archivo de texto
- `reboot` Reinicia el sistema
- `shutdown` Apaga o reinicia el sistema
- `sudo` Ejecuta un comando como superusuario
- `who` o `users` Muestra los usuarios conectados al sistema
- `whoami` Muestra el *login* del usuario actual

3.3.3 REDIRECCIÓN DE COMANDOS (*STREAM* Y *PIPELINES*)

Igual que en Windows pero, además, se puede redirigir la salida de errores (*stderr*) con `>&` o `2>`. Si ponemos `>& fichero` o `>& /dev/null` al final de un comando la salida estándar (*stdout*) y los mensajes de error que se produzcan (*stderr*) no se visualizaran por pantalla sino que se redi-

reccionan al fichero indicado o al llamado "agujero negro". Para redireccionar solamente la *stderr* hay que poner `2> fichero` o `2> /dev/null`. Suele usarse solamente que en scripts para evitar mensajes de error previstos.

3.3.4 USAR FILTROS

Linux dispone de muchos más filtros que Windows

Comando	Descripción
`cut`	Extrae caracteres de las líneas de un fichero
`expand` `unexpand`	Convierte tabuladores en espacios y viceversa
`grep`	Busca los caracteres especificados en archivos y en la información de salida de los comandos.
`head` `tail`	Extraen líneas, las primeras o las ultimas, de un fichero
`join` `paste`	Une el contenido de 2 o mas ficheros (linealmente)
`less`	Como `more` pero permite un mejor manejo del desplazamiento por el texto
`more`	Muestra el contenido de un archivo o la información de salida de un comando en una ventana del símbolo del sistema cada vez.
`nl`	Numera las líneas del fichero
`sort`	Ordena alfabéticamente los archivos y la información de salida de los comandos.
`split`	Divide un fichero en trozos del tamaño o líneas indicado
`tr`	Cambia caracteres por otros
`uniq`	Elimina líneas repetidas
`wc`	Cuenta palabras, líneas, caracteres...

Para enviar la información de entrada de un archivo a un comando de filtro, utilice el signo menor que (<) aunque en la mayoría de los casos puede obviarse. Si desea que el comando de filtro obtenga la información de entrada a partir de otro comando, utilice el carácter de barra vertical (|).

more o less

El comando `more` muestra el contenido de un archivo o la información de salida de un comando en una ventana del símbolo del sistema cada vez. Por ejemplo, para mostrar el contenido de un archivo denominado Lista.txt en una ventana del símbolo del sistema cada vez, escriba:

```
more < lista.txt     o      more lista.txt
```

Se hará una pausa al completar una ventana. Podemos avanzar o retroceder usando las teclas de cursor. Para detener el comando sin ver más información, presione `q` o `CTRL+C`.

El comando `more` resulta útil si está trabajando con un comando que produce información de salida en más de una ventana del símbolo del sistema. Por ejemplo, suponga que desea ver el contenido detallado del directorio `/etc`:

```
ls –la /etc | less
```

sort

El comando `sort` ordena alfabéticamente un archivo de texto o la información de salida de un comando. Por ejemplo, el siguiente comando ordena el contenido de un archivo llamado `Lista.txt` y muestra los resultados en la ventana del símbolo del sistema:

```
sort < Lista.txt     o     sort Lista.txt
```

En este ejemplo, el comando `sort` ordena las líneas del archivo Lista.txt en una lista alfabética y muestra el resultado sin cambiar el archivo. Para guardar la información de salida del comando `sort`, en lugar de mostrarla en pantalla, escriba el signo mayor que (>) y un nombre de archivo. Por

ejemplo, el siguiente comando ordena alfabéticamente las líneas del archivo `Lista.txt` y guarda el resultado en el archivo `ListaAlf.txt`:

```
sort Lista.txt > ListaAlf.txt
```

Para ordenar la información de salida de un comando, escriba el comando seguido de un carácter de barra vertical (`|`) y el comando `sort` (es decir, `comando | sort`). Por ejemplo, el siguiente comando ordena alfabéticamente las líneas que incluyen la cadena "Luisa" (es decir, la salida del comando `grep`):

```
grep Luisa listacorreo.txt | sort
```

Algunos de los modificadores de `sort` son:

- `-n` ordena numéricamente
- `-r` revierte el resultado (de mayor a menor)
- `-f` ignora mayúsculas/minúsculas
- `+`*nº* ordena por el campo indicado (pueden indicarse varios sort +2 +0 /etc/passwd)

cut

Se usa para mostrar partes seleccionadas de un fichero en la salida estándar, como puede se columnas o campos. Es posible seleccionar un trozo de una línea específica, varios trozos, o un rango.

- `-c` Escribe en la salida solo los caracteres especificados
- `-d` Especifica el separador (delimitador) de campos del fichero
- `-f` Escribe en la salida solo los campos especificados, delimitados por –t (por defecto, tabuladores)

En el primer ejemplo se mostraran en la salida solo los primeros diez caracteres de cada línea. En el segundo se muestra el 3º campo usando como separador de campos los dos puntos:

```
cut -c1-10 /etc/passwd
cut –d: -f3 /etc/passwd
```

paste

Une, línea a línea, dos o mas ficheros usando tabuladores como separador.

- `-d` Específica el separador (o separadores consecutivos) de campos en lugar de tabulador.

join

Igual que `paste` pero especificando un campo de coincidencia común en ambos ficheros.

- `-1`*nº* posición del campo común en el primer fichero
- `-2`*nº* posición del campo común en el segundo fichero
- `-t` carácter delimitador de campos

tr

Sustituye caracteres especificados por otros.

- `-d` elimina el carácter especificado
- `-c` elimina los caracteres no especificados
- `-s` sustituye conjuntos repetidos de un carácter por uno solo

Como ejemplo, para convertir todas las mayúsculas de un fichero en minúsculas, y almacenar el resultado en el mismo fichero:

```
tr [A-Z] [a-z] <> fichero
```

expand

Convierte tabuladores en espacios (por defecto 8).

- `-I` solo procesa los tabuladores al principio de la línea
- `-t`*nº* especifica el número de espacios a usar

unexpand

Convierte espacios en tabuladores (por defecto 8).

- `-a` convierte todos los espacios en tabuladores (si no se especifica solo convierte los del principio de línea)
- `-t`*nº* especifica el número de espacios a convertir

split

Divide un fichero en trozos (líneas o bytes). Por defecto, en trozos de 1000 líneas. El resultado se almacena en ficheros con nombre `esquemall` donde ll son letra: aa, ab, ac,… El esquema por defecto es `x`.

`split [modificador]` ***fichero*** `[esquema]`

- `-l` número de líneas de cada fichero resultante

head y tail

Muestra las líneas que indiquemos del principio o del final del archivo (por defecto 10)

- `-n` número de líneas a mostrar

Para ver los datos del último usuario incorporado al sistema:

```
tail -n1 /etc/passwd
```

grep

El comando `grep` busca en los archivos la cadena o el texto que especifique. La shell muestra todas las líneas que coinciden con la cadena o el texto especificado Puede usar el comando `grep` como comando de filtro o como un comando estándar.

- `-c` muestra el número de líneas pero no las líneas en sí
- `-i` ignora mayúsculas/minúsculas
- `-n` muestra las líneas coincidentes precedidas de su número de línea dentro del fichero
- `-v` muestra las líneas no coincidentes
- `-w` busca palabras completas
- **^** para indicar que busque al principio o fin de línea (`grep -w ^alumno16 /etc/passwd`)

wc

Muestra estadísticas de un fichero (número de palabras, líneas, caracteres, etc.)

- `-l` número de líneas
- `-c` número de caracteres
- `-L` longitud de la línea más larga

uniq

Elimina líneas repetidas del fichero. Este debe estar ordenado.

- `-d` solo muestra las líneas repetidas
- `-u` solo muestra las líneas únicas (no repetidas)

nl

Numera las líneas de un fichero con incremento +1. Comienza desde 1 en cada página lógica del fichero.

- `-a` numera todas las líneas
- `-t` solo numera las líneas no vacías
- `-i`*nº* para indicar un incremento distinto de 1
- `-p` continua la numeración sin reiniciar en cada página lógica
- `-v`*nº* para indicar el número inicial en cada página lógica

3.3.5 SÍMBOLOS DE PROCESAMIENTO CONDICIONAL (UTILIZAR VARIOS COMANDOS)

Igual que en Windows

Ejercicios:

Estudia los comandos de la shell. Para ello utiliza el comando `man` conforme vas realizando ejercicios del apartado "Ejercicios de comandos" (página 28). Describe la acción de cada uno de ellos, prueba los modificadores y recoge los más importantes en un chuletario siguiendo el formato que usaste para el chuletario de los comandos Windows:

Ejemplo:

ls [*patrón*] | muestra contenido directorios | -liaRhQr

3.3.6 MONTAJE DE DISPOSITIVOS

Para poder acceder al contenido de un dispositivo hay que montarlo, es decir, indicarle al sistema qué dispositivo es, que sistema de archivos utiliza y en qué directorio (punto de montaje) queremos tener acceso a su contenido.

Es importante saber que los datos de los dispositivos los maneja Linux en la swap y no sobre el dispositivo en si. Por ello, los cambios que hagamos en un disquete o pendrive no serán reales hasta que el sistema haga un proceso de volcado diferido (cosa que hace periódicamente) o desmontemos el mismo.

Para montar y/o desmontar dispositivos hay que entrar al sistema como *root* (si está la cláusula `nouser` en `/etc/fstab`).

Punto de montaje

Algunas versiones de Linux tiene preparados dos directorios para el montaje de dispositivos externos: `/floppy` y `/cdrom1` (o similares `/media/cdrom`)

En los sistemas más modernos, el montaje del pendrive y del CD suele ser automático al pinchar el pendrive o poner un CD. Sin embargo, estos automatismos que suelen ser cómodos para el usuario normal, no lo son para el manejo de los dispositivos desde el entorno de texto.

En el fichero `/etc/fstab` suele haber información sobre los dispositivos existentes y el sistema de archivos que utiliza cada uno de ellos.

Dispositivo

Si el ordenador tiene disquetera, esta aparecerá como `/dev/fd0`

El CD-R puede aparecer como `/dev/cdrom` (si es IDE) o como `/dev/sd*` (si es SATA)

Los pendrives son dispositivos USB por lo que aparecen como dispositivos `/dev/sd*`

Sistema de archivos

En el caso de la disquetera y del pendrive el sistema de archivos suele ser `vfat`, aunque en algunos pendrives puede ser `ntfs-3g`. Normalmente, sólo montará automáticamente los disquetes formateados con sistema `vfat` (desde Windows). Para montar un disquete formateado con sistema de archivos de Linux tendremos que usar la versión completa del comando `mount`.

El sistema de archivos de los CDs y DVDs suele ser `iso9660`

mount | umount

Para montar un dispositivo se usa la orden `mount` cuyo formato es:

`mount` `-t sist-archi dispositivo directorio [opciones]`

Con la opción `-t` le indicamos el sistema de archivos del dispositivo a montar: `vfat`, `ext3`, `ntfs-3g`, etc.

El dispositivo será el nombre con el que Linux denomina al mismo: `/dev/hda?` o `/dev/sda?` (disco duro IDE maestro), `/dev/hdb?` o `/dev/sdb?` (disco duro IDE esclavo), `/dev/fd0` (disquetera), `/dev/sd??` (pendrive en puerto USB), etc.

El directorio será el nombre del punto de montaje en el que queremos ver el contenido del dispositivo y debe existir previamente.

Por lógica, para montar un pendrive formateado con Windows tendríamos que usar la orden

```
mount -t vfat /dev/sdc1 /pendrive
```
[9]

Para montar un pendrive formateado con Linux tendríamos que usar la orden

```
mount -t ext3 /dev/sdc1 /pendrive
```

Con un cdrom tendríamos que usar la orden

```
mount -t iso9660 /dev/cdrom /cdrom
```

Para desmontar el dispositivo se usa la orden

```
umount dispositivo | directorio
```

4. TABLA CON ALGUNOS CÓDIGOS ASCII.

Para insertar los caracteres (sobre todo si el teclado no esta en español) mantener pulsada la tecla `Alt` y, con el teclado numérico, marcar el número del carácter deseado

Carácter	Código	Carácter	Código
Espacio	32	?	63
"	34	@	64
#	35	\	92
%	37	|	124
*	42	~	126
/	47	©	184
:	58		

[9] ver antes de pinchar el pendrive los dispositivos sd?? y comprobar después de pinchar para saber el valor real que debemos poner

Manejo de rutas

Utilizando el árbol de directorios de la página 2, escribe, indicando si es relativa o absoluta, la ruta que resultaría más cómoda para:

Siendo el directorio activo **Apuntes**:

Linux.pdf del directorio `Varios` _____

IPv6.doc _____

chipset.pps _____

Siendo el directorio activo **Comandos**:

Linux.doc del directorio `Comandos` _____

Linux.doc del directorio `Gestion_Informacion` _____

chipset.pps _____

Siendo el directorio activo **Complementario**:

Linux.doc del directorio `Comandos` _____

Redes.doc _____

Linux.pdf del directorio `Varios` _____

Bloque1.pps _____

Teniendo en cuenta que el comando `move archivo destino` cambia un fichero de ubicación:

cambia *IPv6.doc* al directorio `Redes` _____

cambia *Redes.doc* al directorio `Varios` _____

Teniendo en cuenta que el comando `md directorio` crea un nuevo directorio y siendo el directorio activo **Apuntes**:

Crea el directorio `ejercicios` como hijo de `Comandos` _____

Crea el directorio `linux` como hijo de `ejercicios` _____

Crea el directorio `redes` como hijo de `Varios` _____

Cambia *Redes.doc* al nuevo directorio _____

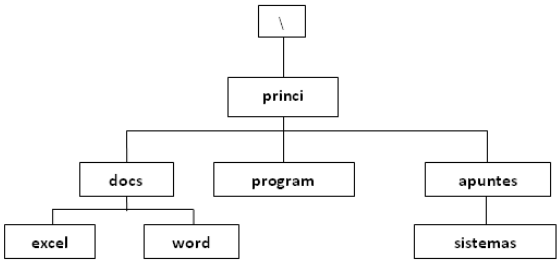
Crea el directorio `trabajos` como hijo del raíz _____

Cambia *SistemasNumericos.doc* al nuevo directorio

_____

Ejercicios de comandos Linux y Windows

1. Crea la siguiente estructura jerárquica de directorios.



2. Genera, copiándolos desde la consola, los siguientes archivos: *fichero.txt* y *fichero2.txt* (en `princi`), *tema1.doc* y *tema2.doc* (en `docs`), *doc1.xls* y *doc2.mdb* (en `sistemas`).
3. Copia todos los archivos generados en el punto anterior en `program`. Haz una copia de todo el directorio `princi` en `princi2` (con todos sus subdirectorios y archivos). Comprueba que se ha creado la copia exacta.
4. Estás en `sistemas`, sitúate en `program`. Hazlo de todas las formas que sepas.
5. Lista el contenido del directorio `princi`, que muestre primero los directorios y a continuación los archivos ordenados alfabéticamente por la extensión.
6. Lista el contenido del directorio `princi` ordenado por tamaño y los subdirectorios al principio. Introduce el parámetro necesario para ver la información de forma paginada.
7. Utilizando una ruta relativa, crea el directorio `nuevo` dentro de `word`.
8. Sitúate en `word` y cambia el nombre del directorio `nuevo` por `textos`. ¿de cuántas formas puedes hacerlo?
9. (sólo Windows) Copia los archivos del directorio `princi` en el directorio `docs`, pero con extensión *.bak*
10. Copia los archivos que en el nombre contengan un 2 del directorio `docs` al directorio `textos`.
11. Visualiza el contenido de los ficheros del directorio `textos`.
12. Copia el contenido de los ficheros en uno solo llamado *fusión.txt*.
13. Visualiza el contenido de este nuevo fichero.
14. Mueve el fichero *fusión.txt* al directorio `excel`.
15. Crea un fichero copiándolo desde la consola en el directorio anterior. Se llamará *prueba1.txt*.
16. Mueve el fichero *prueba1.txt* al directorio `nuevo` dentro de `princi`. El directorio `nuevo` no existe.
17. Visualiza el fichero *prueba1.txt*.
18. Elimina el directorio `nuevo` y su contenido. Indica todos los comandos que puedes usar.
19. Mueve el fichero *fusión.txt* al directorio `textos` pero con el nombre *union.doc*.
20. Visualiza atributos de ficheros del directorio `princi` y de sus subdirectorios.
21. Sitúate en `program` y visualiza atributos de los ficheros del directorio `textos`
22. Sitúate en `textos`. Visualiza los atributos de los ficheros con extensión *.txt*.
23. Copia los ficheros del directorio `sistemas` en el directorio `textos`.
24. (solo windows) Asigna el atributo de solo lectura a los archivos del directorio `textos`.
25. Intenta borrar los ficheros de este directorio.
26. Quita el atributo de sólo lectura a los archivos del directorio `textos`.
27. Visualiza los atributos del directorio `textos`.

28. Asigna el atributo oculto a los ficheros con extensión *.txt*.
29. Visualiza de nuevo los ficheros del directorio `textos`.
30. Quita los atributos de archivo y de lectura a los archivos con extensión *.bak*.
31. Quita el atributo de oculto de los ficheros con extensión *.txt*.
32. Modifica un archivo con extensión *bak*. Utiliza el editor `edit` o hazlo desde la consola con la orden `copy con >>`
33. Visualiza los atributos de los archivos con extensión *bak*.
34. Asigna el atributo de lectura a un archivo con extensión *.bak*.
35. Modifica archivo anterior con extensión *.bak*. Utiliza el editor `edit`. ¿qué ocurre? Modifícalo desde la consola con la orden `copy con >>` … Muestra su contenido ¿qué ocurre?
36. Copia un archivo *.bak* que no tenga el atributo de archivo, en el directorio `sistemas`. Visualiza los atributos del fichero copiado ¿qué ocurre?

37. Copia en el directorio `program` toda la estructura de ficheros y directorios que cuelga del directorio `apuntes`.
38. Visualiza ayuda del comando `format` (`mkfs`)
39. (solo windows) Cambia la etiqueta a un disco
40. Visualiza trayectorias de búsqueda de archivos ejecutables.
41. Observa y describe el resultado de la ejecución de las órdenes:
    42. `dir|sort`:
    43. `dir>more`:
    44. `sort>dir`:
    45. `more<dir`:
    46. `sort`:
    47. `more`:
48. Crea un fichero que se llame *ayuda-for* que contenga la ayuda del comando `format` (`mkfs`).
49. Visualiza el contenido de *ayuda-for* de forma paginada.
50. Visualiza todos los ficheros del directorio `\` que han sido creados o modificados hace dos días.
51. (windows) Explica que hace el siguiente comando: `for %V in (*.txt *.bas) do echo %V`.
52. Crea en `result` (hijo de `princi`) tres ficheros (*fich1*, *fich2*, *fich3*). El tercer fichero tiene que contener en la primera línea tu nombre, en la segunda tu fecha de nacimiento y en la tercera tu ciudad.
53. Visualiza las líneas que no contienen la edad del fichero anterior.
54. Visualiza las líneas de los alumnos de 19 años del fichero *fich3*.
55. Borra el directorio `result`. Indica las formas que tienes de hacerlo.
56. Indica la orden necesaria para preparar un disco y trabajar con él
57. Ejecuta la orden para chequear el disco
58. (linux) ¿Qué comando se usa para añadir el número de línea a un fichero?
59. (linux) ¿Qué comando permite combinar dos ficheros usando campos de unión?
60. (linux) ¿Qué comando permite borrar caracteres de un fichero?
61. (linux) ¿Con qué comando se pueden ordenar numéricamente el contenido de un fichero?
62. (linux) Copia el `archiv /etc/passwd` en tu *home* con el nombre `usuarios`
63. (linux) Divide el archivo `usuarios` en varios archivos que contengan cada uno de ellos 6 líneas

64. (linux) Muestra la línea de `usuarios` correspondiente al ultimo usuario que fue dado de alta en el sistema
65. (linux) Ordena alfabéticamente de la z a la a el fichero `usuarios`, numerara las líneas y finalmente divídelo en ficheros llamados `usuxx` que contengan 5 líneas cada uno usando tuberías
66. (linux) Cuenta las líneas y palabras del fichero `usuarios`, usando una única línea de comando
67. (linux) Crea un nuevo fichero llamado `minus` con el contenido de `usuarios` pero todo en minúsculas
68. (linux) Sustituye en `minus` los ; por guiones
69. (linux) Crea un enlace duro (`enlH`) y uno simbólico (`enlS`) al fichero `minus`. Observa los permisos de los tres archivos
70. (linux) Monta el pendrive. Copia los tres archivos del ejercicio anterior. ¿Qué ocurre?.
71. (linux) Crea en el pendrive el archivo `Nuevo`. En el mismo pendrive, crea un enlace duro y uno simbólico a `Nuevo`. Ve a tu *home* y crea un enlace duro y uno simbólico a `Nuevo` ¿Qué ocurre?
72. El comando `cut` sirve para extraer campos de un fichero. Observa los resultados obtenidos al ejecutar los siguientes comandos:
    a) `cut –c1 /etc/passwd`
    b) `cut –c1,5,10-20 claves`
    c) `cut -d: -f3- claves`
73. Observa el funcionamiento del comando paste de la siguiente forma:
    74. Crea un fichero (`nombres`) con el editor que contenga el nombre y apellidos de cinco personas (uno por línea)
    75. Crea otro fichero (`direcciones`) con las direcciones de esas cinco personas (una por línea)
    76. Prueba los siguientes comandos:

```
paste nombres direcciones
paste -d: nombres direcciones
paste -s nombres
paste -s nombres direcciones
```