TEMA 4: WINDOWS 10 PARTE I. COMANDOS

Índice

1. Introducción

Normalmente gestionamos los sistemas operativos desde los interfaces gráficos de usuario (IGU - GUI) de una forma visual, pero también podemos gestionar dichos sistemas desde la línea de comandos, usando para ello una pantalla de texto.

La línea de comandos tiene varias ventajas sobre el GUI, como pueden ser:

- Muchas órdenes de gestión del sistema operativo, que se consideran de muy bajo nivel o muy peligrosas, no son accesibles desde el GUI.
- El entorno de texto, es un sistema muy eficiente, podemos abrir sesiones remotas en nuestro equipo desde otras ubicaciones y usar una línea de comandos para dar órdenes al sistema controlado, podemos tener varias sesiones con entorno de texto concurrentes, etc.
- Podemos automatizar las órdenes usando los lenguajes de programación del propio sistema operativo. Estos programas por lotes se conocen como scripts, procesos por lotes o archivos batch y nos ofrecen muchas posibilidades.
- En caso de un error en algún dispositivo hardware del sistema informático, es muy probable que no podamos acceder al GUI, pero casi seguro que será posible acceder de algún modo a la línea de comandos.
- En caso de estar usando herramientas de recuperación de un sistema informático, para intentar corregir un problema de software importante, necesitaremos conocer el uso de la línea de comandos por que seguramente será lo único con lo que contemos.

Normalmente hablamos del intérprete de comandos como un shell. 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 del antiguo sistema operativo MS-DOS, el command.com. El shell de comandos de los sistemas operativos Windows utiliza el intérprete de comandos cmd.exe, que carga aplicaciones y dirige el flujo de información entre ellas, para

traducir los datos de entrada del usuario a un formato que el sistema operativo reconozca.

CMD no es el único shell de comandos que podemos usar en entornos Windows. Microsoft ha desarrollado otros shell que podemos instalar y usar. Así, tenemos por ejemplo, el **PowerShell**, antes llamado Nomad, basado a su vez en un shell llamado MSH, orientado a objetos y que dispone de muchos más comandos que el CMD. Es el actual sustituto de CMD y presenta potentes opciones de scripting (creación de procesos por lotes) y comandos renovados. Lo estudiaremos en próximos temas.

2. Ejecución y configuración del shell de comandos

Para ejecutar el shell de comandos de Windows, debemos **ejecutar** (Tecla Windows + R) el programa CMD.EXE o bien haciendo clic en el cuadro de búsqueda y tecleando *Símbolo del Sistema* o *cmd*.

OJO. En las nuevas versiones de Windows 10 el intérprete de comandos ha desaparecido del menú Inicio y de las opciones del menú contextual del menú Inicio (botón derecho del ratón sobre menú Inicio). En su lugar aparece el intérprete de comandos de PowerShell. Podríamos ejecutar también los comandos que veamos desde ese nuevo intérprete de comandos (que estudiaremos en temas posteriores) pero lo vamos a hacer desde el clásico para no confundirnos.

Para configurar el símbolo del sistema:

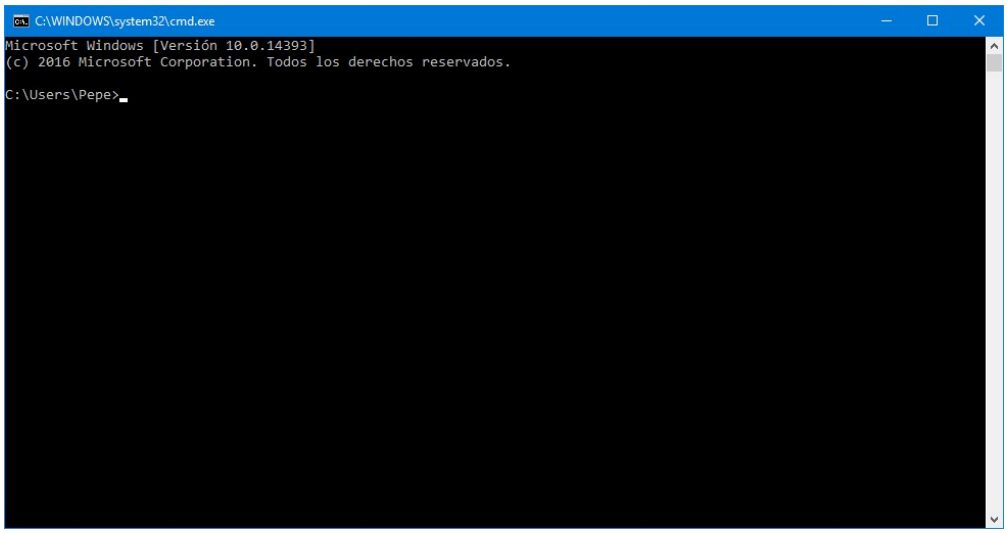
1. Abrimos Símbolo del sistema (cmd.exe).
2. Hacemos clic en la esquina superior izquierda de la ventana del símbolo del sistema y, a continuación, hacemos clic en Propiedades.
3. Hacemos clic en la ficha Opciones.

Desde aquí podemos modificar muchas opciones.

- En Historial de comandos, en Tamaño del búfer si escribimos 999 y, a continuación, en Número de búferes escriba o seleccione 5 mejoraremos el tamaño y el comportamiento del buffer de comandos (que nos permite acceder a lo escrito anteriormente con los cursores)

- En Opciones de edición, si activamos las casillas de verificación Modalidad de edición rápida y Modalidad de inserción, conseguiremos habilitar la función de copiar y pegar directamente en el shell de comandos. Para copiar simplemente seleccionamos con el ratón y pulsamos botón derecho del ratón. Para pegar, simplemente pulsamos botón derecho del ratón.
- También podemos modificar el alto y ancho de la pantalla, su posición automática, etc.

Al entrar en el intérprete de comandos nos aparece una ventana como la siguiente:



Nos informa por pantalla de la versión y del copyright y seguidamente nos muestra el **prompt** para poder introducir los comandos:

```
C:\Users\Pepe>
```

La información que muestra el prompt es configurable y se puede modificar. Por defecto informa del directorio activo, seguido del símbolo de mayor (>). Este símbolo es el que nos indica que Windows está listo para ejecutar un comando. A lo largo de los apuntes lo mostraremos delante de los comandos. Evidentemente no habrá que teclearlo.

El directorio activo es el lugar donde nos encontramos. De entrada Windows nos sitúa en nuestra carpeta personal. Para el usuario Pepe, nos encontramos en **C:\Users\Pepe**

Es muy importante conocer el directorio en el que nos encontramos en todo momento, pues para trabajar con ficheros que se encuentran en otras ubicaciones habrá que poner la ruta completa (absoluta o relativa) de estos. Veremos que también podemos cambiar el directorio activo cuando queramos y situarnos en otra ubicación.

Sabemos también que Windows no es *case sensitive*, es decir, no distingue entre mayúsculas y minúsculas. No tendremos problemas con este aspecto a la hora de teclear los comandos o los nombres de los ficheros.

Con lo que sí que vamos a tener problemas es con los nombres de los ficheros o directorios que tienen espacios en blanco en su nombre. El intérprete de comandos nos obligará a poner estos nombres entre comillas dobles. Ejemplo:

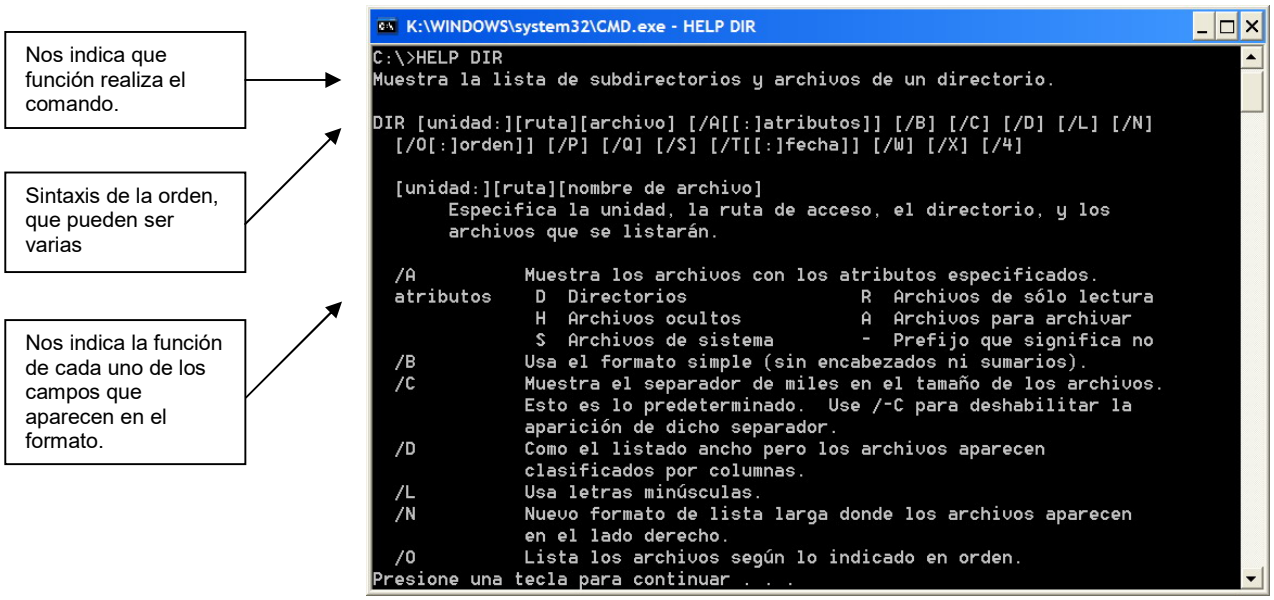
> El fichero cuyo nombre es (nótese que hay 3 espacios, dos en el directorio de su ruta y otro en el propio nombre del fichero):
> `C:\Users\Alumno\Apuntes de SIN\Tema 1.pdf`
>
> debería escribirse como sigue:
> `"C:\Users\Alumno\Apuntes de SIN\Tema 1.pdf"`
> o bien:
> `C:\Users\Alumno\"Apuntes de SIN"\"Tema 1.pdf"`
> o también:
> `C:\Users\Alumno\"Apuntes de SIN\Tema 1.pdf"`

Hay que señalar también que en las rutas de los archivos se utiliza la **barra invertida (\) o contrabarra** y para especificar los parámetros en los comandos se utiliza la **barra inclinada** (/).

3. Uso de la ayuda en el shell de comandos

En el caso de la línea de comandos, disponemos de una ayuda general accesible mediante la orden HELP. Si queremos ayuda específica sobre cualquier comando, podemos ejecutar HELP comando. También podemos acceder a la ayuda del comando escribiendo **`comando /?`**

Es muy importante saber interpretar correctamente las pantallas de ayuda. Existen una serie de convenciones comunes a todos los sistemas que debemos conocer.



La sintaxis aparece en el orden en que debe escribir un comando y los parámetros que lo siguen. La tabla siguiente explica cómo interpretar los diferentes formatos de texto.

Leyenda de formato

Formato	Significado
Cursiva o minúsculas	Información que debe suministrar el usuario
Negrita o mayúsculas	Elementos que debe escribir el usuario exactamente como se muestran
Puntos suspensivos (...)	Parámetro que se puede repetir varias veces en una línea de comandos
Entre corchetes [ ]	Elementos opcionales, pueden usarse o no.
Entre llaves {} opciones separadas por barras verticales |.	Conjunto de opciones de las que el usuario debe elegir sólo una. Ejemplo: {par | impar}

Ejemplo:

```
> DIR /?
```

```
DIR [unidad:][ruta][archivo] [/A[[:]atributos]] [/B] [/C] [/D] [/L] [/N]
  [/O[:]orden]] [/P] [/Q] [/S] [/T[[:]fecha]] [/W] [/X] [/4]
```

Las palabras que aparecen sin estar encerradas entre corchetes son palabras obligatorias al formato, es decir que no podemos escribir la orden sin usarlas. Si nos fijamos, solo la palabra DIR está libre, así que el formato mínimo de la orden seria DIR.

Todo lo que está encerrado entre corchetes indica que es optativo. Así por ejemplo, el modificador /A es optativo, pero veamos cómo está representado dicho modificador:

```
[/A[[:]atributos]]
```

Vemos que ahí varios niveles de integración de corchetes. Así, /A es optativo (está entre corchetes) y podemos poner /A sin poner nada más. Podemos poner también /A atributos si queremos, sin poner el símbolo : .Si lo deseamos podemos poner el formato completo que sería /A:atributos.

Lo que se consigue con /A o lo que significan atributos, lo tenemos en la misma ayuda de DIR un poco más abajo.

```
/A          Muestra los archivos con los atributos especificados.
atributos    D  Directorios                R  Archivos de sólo lectura
             H  Archivos ocultos           A  Archivos para archivar
             S  Archivos de sistema        -  Prefijo que significa no
```

Vemos aquí como /A nos sirve para mostrar archivos que cumplan con un determinado atributo. Y vemos como donde en la línea de formato pone atributos, debemos poner una de las siguientes letras: D R H A S. Vemos que también podemos poner el símbolo menos -, pero en este caso se nos indica que es un prefijo, por lo que podríamos poner –A, –S, etc.

4. Uso de varios comandos y símbolos de procesamiento condicional

Podemos ejecutar varios comandos desde una línea de comandos o secuencia de comandos si utilizamos 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 del comando que hay a la izquierda del símbolo de procesamiento condicional. Por

ejemplo, podemos ejecutar un comando solamente si el anterior causa un error. También podemos ejecutar un comando solamente si el anterior es correcto.

Podemos usar los caracteres especiales enumerados en la tabla siguiente para pasar varios comandos.

Carácter	Sintaxis	Definición
&	Comando1 & Comando2	CMD ejecuta el primer comando, y luego el segundo.
&&	Comando1 && Comando2	CMD ejecuta el primer comando, y si ese comando es correcto, entonces ejecuta el segundo. Si Comando1 falla, no se ejecuta Comando2.
||	Comando1 || Comando2	Comando2 solo se ejecuta si Comando1 es incorrecto o falla.
( )	(Comandos)	Se usa para anidar comandos. Se ejecutan primero los comandos que están dentro de los paréntesis que los que están fuera de los mismos)

Ejemplo:

```
> DIR F: || DIR D:
```

Se ejecuta el comando DIR F: y si da un error, entonces se ejecuta el comando DIR D:

5. Uso de comodines

Los comodines son caracteres del teclado como el asterisco (*) o el signo de interrogación (?) que se pueden utilizar para representar uno o más caracteres reales al buscar archivos o carpetas. A menudo, los comodines se utilizan en lugar de uno o varios caracteres cuando no se sabe el carácter real o no se desea escribir el nombre completo.

Asterisco (*)

Podemos utilizar el asterisco como sustituto de cero o más caracteres. Si buscamos un archivo que sabemos que comienza por "glos" pero no recordamos el resto del nombre

del archivo, escribimos lo siguiente:

```
> dir glos*
```

Con esto, buscaremos todos los archivos de cualquier tipo que comiencen por "glos", incluidos Glosario.txt, Glosario.doc y Glos.doc. Para limitar la búsqueda a un tipo de archivo específico, escribimos:

```
> dir glos*.doc
```

En este caso, buscaremos todos los archivos que comiencen por "glos" pero con la extensión .doc, como Glosario.doc y Glos.doc.

Signo de interrogación (?)

Podemos utilizar el signo de interrogación como sustituto de un único carácter en un nombre. Por ejemplo, si escribimos

```
> dir glos?.doc
```

Encontraremos los archivos Glosa.doc y Glos1.doc, pero no Glosario.doc.

6. Principales comandos

En el shell de comandos de Windows, existen cientos de comandos que pueden ser utilizados. Muchos de ellos se instalan directamente con Windows, mientras que otros especiales se instalan conjuntamente con otras herramientas. Veamos los más habituales:

Comando	Descripción	Ejemplo
VER	Muestra la versión del sistema operativo.	VER
Unidad:	Cambia la unidad activa	C: D: E: A:
HELP	Muestra una pequeña ayuda sobre los comandos	HELP HELP comando
DIR	Visualiza el contenido de un directorio	DIR C:\WINDOWS\
ECHO	Muestra mensajes de texto	ECHO HOLA MUNDO
FORMAT	Formatea una unidad (cuidado, no probar)	FORMAT G:
CHKDSK	Comprueba el estado de un disco	CHKDSK C:
LABEL	Cambia la etiqueta de un disco	LABEL D:
VOL	Muestra la etiqueta de un disco	VOL C:

CLS	Limpia la pantalla	CLS
TIME	Muestra y permite cambiar la hora	TIME
DATE	Muestra y permite cambiar la fecha	DATE
ATTRIB	Muestra o cambia los atributos de un archivo	ATTRIB FOTO1.JPG
COPY	Permite copiar ficheros	COPY C:\BOOT.INI E:\
MOVE	Mueve ficheros	MOVE C:\BOOT.INI E:\
DEL	Borra ficheros	DEL E:\WINDOWS*.JPG
REN	Renombra ficheros	REN E:\BOOT.INI E:\BT.INI
TYPE	Muestra el contenido de un fichero	TYPE FICHERO.EXT
MKDIR (MD)	Crea un directorio	MD E:\APUNTES
RMDIR (RD)	Borra directorios	RD E:\APUNTES
CHDIR (CD)	Cambia de directorio actual	CD E:\APUNTES
TREE	Muestra la estructura de directorios	TREE
CACLS	Muestra/modifica las listas de control de acceso	CACLS FOTO1.JPG
EXIT	Sale del símbolo de comandos (si es posible)	EXIT
XCOPY	Copy extendido. Dispone de modificadores exclusivos	XCOPY E:\ D:\ /E
SUBST	Le da un nombre de volumen a un directorio	SUBST J: E:\UTILES
FIND	Busca una cadena de caracteres en un fichero	FIND "CADENA" FICHERO.EXT
SORT	Recibe un fichero y lo devuelve ordenado	SORT NOMBRES.TXT

De cada uno de estos comandos podemos obtener ayuda, bien escribiendo HELP comando o escribiendo *comando /?*

Existen muchos más comandos, tanto internos como externos. (Se dice que un comando es interno cuando viene incluido en el propio CMD y se carga en memoria continuamente). Al menos los comandos que aparecen en la tabla de la página anterior, deben conocerse y usarse sin problemas.

- Ejemplo: Visualizar el contenido de un disco. Escribe `DIR` y Pulsa Intro

  Aparecerá un listado de archivos y carpetas, que contienen archivos en su interior, tamaño expresado en bytes, fecha, hora de última actualización (o

edición), de la unidad a la que le hemos hecho el DIR. Si el listado es muy largo (hay muchos archivos), veremos como la pantalla va muy rápida y no nos da tiempo a leerlo todo. Para remediar esto escribimos el DIR seguido de /P.

- Ejemplo: Listar archivos haciendo pausa. Escribe `DIR /P`

  Una vez la pantalla quede llena, nos pedirá que pulsemos cualquier tecla para continuar, y así hasta terminar listando todos los archivos y carpetas del disco. También hay otra forma de presentar los archivos por pantalla, visualizándolos a lo ancho.

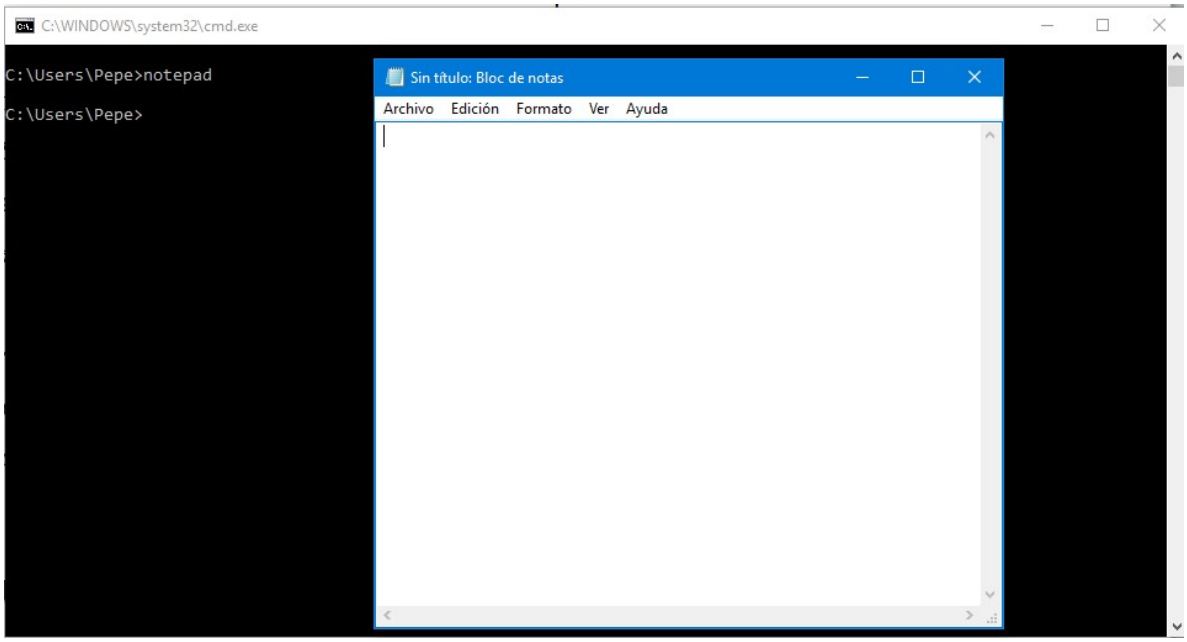
- Ejemplo: Listar archivos a lo ancho. Escribe `DIR /W`

  Si no cabe en la pantalla, pasará lo mismo que en los casos anteriores, pero esto tiene solución si hacemos servir la función de pausa.

- Ejemplo: Listar archivos a lo ancho con pausa Escribe `DIR /W /P`

  Se puede identificar una carpeta si al hacer el DIR hay unos archivos que lleven a su parte derecha, en vez del tamaño, un nombre: <DIR>, esto significa que esta es una carpeta que contiene, posiblemente, más archivos en su interior.

El intérprete de comandos de Windows 10 no incorpora ningún editor de textos puro. Nosotros utilizaremos el bloc de notas (**Notepad.exe**) para acceder al editor de archivos de texto de Windows.



Desplazarse por el sistema

Todos los directorios del sistema se encuentran dentro de la jerarquía que cuelga de la raíz de la unidad que los contiene (C: \, D: \ ...). La orden *cd* (change directory) permite cambiar de un directorio a otro. El camino del directorio actual se muestra siempre en la entrada de la línea de comandos (prompt).

La instrucción cd tiene la siguiente sintaxis:

```
> cd directorio
```

La especificación de este directorio puede ser relativa o absoluta. Para poder acceder a un camino de una unidad distinta de la unidad donde se encuentra el terminal en un momento determinado, hay que cambiar antes de unidad. Para hacer el cambio, hay que introducir el nombre de la unidad, seguido de dos puntos.

La orden cd se suele combinar con los dos puntos (..) para ascender un nivel en la jerarquía y las contrabarras como carácter separador de directorios (\).

Ejemplos de cambios de directorio.
Subir un nivel:

```
C:\Users\Mindeos> cd ..
```

Se accede por camino absoluto, a C:\Windows:

```
C:\Users> cd C:\Windows
```

Podríamos haber omitido el nombre de la unidad C: (puesto que nos encontramos en ella) y haber escrito **cd \Windows**

Se accede por camino relativo, el directorio C:\Windows\System32\drivers:

```
C:\Windows> cd System32\drivers
```

Se cambia de unidad:

```
C:\Windows\System32\drivers> D:
```

Se accede al directorio apuntes de la unidad D:

```
D:\> cd Apuntes
D:\Apuntes>
```

Acceder al directorio raíz de la unidad en que nos encontramos:

```
D:\Apuntes> cd \
D:\>
```

Listar el contenido de un directorio

Otra de las tareas que se llevan a cabo cuando se trabaja con directorios es visualizar el contenido o listarlo. La orden **dir** (directory) es la encargada de llevar a cabo esta operación y tiene la siguiente sintaxis:

```
dir ficheros_o_directorios
```

Una vez más, la especificación de los ficheros o directorios puede ser tanto absoluta como relativa. La orden dir permite el uso de los comodines que ya hemos visto.

A continuación pueden ver algunos ejemplos de ejecución de la instrucción dir:

Ejemplos de comodines

Se muestra el contenido del directorio actual:

```
C:\Users\Mindeos> dir
El volumen de la unidad C es OS
El número de serie del volumen es: 70B5-34EB

Directorio de C : \ Users \ Mindeos

19/02/2011 09:08 <DIR> .
19/02/2011 09:08 <DIR> ..
20/02/2011 10:51 <DIR> . VirtualBox
03/08/2010 17:46 <DIR> Contacts
 ...
19/02/2011 10:25 <DIR> VirtualBox VMs
0 archivos 0 bytes
16 dirs 402,295,152,640 bytes libres
```

A continuación se listan las palabras que siguen el patrón p?ta.txt :

```
C:\Users\Mindeos> dir p?ta.txt
El volumen de la unidad C es OS
El número de serie del volumen es: 70B5-34EB

Directorio de C :\Users\Mindeos
```

```
05/03/2011 10:06 0 pata.txt
05/03/2011 10:06 0 peta.txt
05/03/2011 10:10 0 pita.txt
05/03/2011 10:10 0 pota.txt
4 archivos 0 bytes
0 dirs 402,038,427,648 bytes libres
```

A continuación se listan los ficheros que siguen el patrón p*ta.txt:

```
C:\User\Mindeos> dir p*ta.txt
El volumen de la unidad C es OS
El número de serie del volumen es: 70B5-34EB

Directorio de C:\Users\Mindeos

05/03/2011 10:06 0 pata.txt
05/03/2011 10:06 0 peta.txt
05/03/2011 10:11 0 pelota.txt
05/03/2011 10:10 0 pita.txt
05/03/2011 10:10 0 pota.txt
5 archivos 0 bytes
0 dirs 402,038,468,608 bytes libres
```

A continuación se listan los archivos de extensión *.txt :

```
C:\Users\Mindeos> dir *.txt
El volumen de la unidad C es OS
El número de serie del volumen es: 70B5-34EB

Directorio de C:\Users\Mindeos

05/03/2011 10:14 0 gat.txt
05/03/2011 10:06 0 pata.txt
05/03/2011 10:06 0 peta.txt
05/03/2011 10:11 0 pelota.txt
05/03/2011 10:10 0 pita.txt
05/03/2011 10:10 0 pota.txt
6 archivos 0 bytes
0 dirs 402,038,677,504 bytes libres
```

Es posible visualizar también el contenido de un directorio distinto al que nos encontramos poniendo la ruta completa del mismo.

Para visualizar el contenido del directorio \windows sin necesidad de cambiar de directorio actual de trabajo:

```
C:\Users\Mindeos> dir \windows
```

Crear directorios

La orden que permite crear directorios en Windows es **mkdir** o **md** (make directory).
Tiene un uso y una sintaxis muy simple:
mkdir cami_directori
md cami_directori

Como siempre, el camino del directorio que se quiere crear puede ser relativo o absoluto. Veamos algunos ejemplos:
Ejemplos de caminos de directorios
Se crea un nuevo directorio, llamado practicas:

```
C:\Users\Mindeos> mkdir practicas
```

Se crea un nuevo directorio, llamado problemas:

```
C:\Users\Mindeos> md problemas
```

Cambiar el nombre de ficheros o directorios

Para cambiar el nombre de un archivo o de un directorio es necesario utilizar la orden rename o **ren** ('Cambiar el nombre'). Tiene la siguiente sintaxis:

```
rename nombre_antiguo nombre_nuevo
```

O también:

```
ren nombre_antiguo nombre_nuevo
```

Ejemplos de cambios de nombres de ficheros y directorios.
Se cambia el nombre del archivo pelota.txt a bola.txt:

```
C:\Users\Mindeos> rename pelota.txt bola.txt
```

También indicando la ruta si estuviera el fichero en un directorio distinto al actual:

```
C:\Users\Mindeos> rename \Users\Pepe\pelota.txt \Users\Pepe\bola.txt
```

Se cambia el nombre del directorio prácticas por laboratorio:

```
C:\Users\Mindeos> ren practicas laboratorio
```

Copiar ficheros y directorios

Una de las operaciones más típicas cuando se trabaja con sistemas de ficheros es copiar archivos y directorios. La instrucción que permite hacerlo es copy. Tiene la siguiente sintaxis:

```
copy origen destino
```

Esta orden sólo permite hacer copias de archivos. Cuando se hace una copia con destino a un directorio, copy mantiene el nombre original, en caso contrario, se le da el nombre de archivo que se haya indicado. Para duplicar directorios hay que utilizar la orden **xcopy**, que tiene la misma sintaxis que **copy**. Ambas órdenes aceptan el uso de los metacaracteres **?** y ***** .

Ejemplos de copia de archivos y directorios

Se hace una copia del archivo bola.txt con el nombre pelota.txt :

```
C:\Users\Mindeos> copy bola.txt pelota.txt
1 archivo(s) copiado(s).
```

Se copian todos los archivos de extensión *.txt en el directorio practicas :

```
C:\Users\Mindeos> copy *.txt practicas
bola.txt
got.txt
pata.txt
peta.txt
pelota.txt
pita.txt
pota.txt
7 archivo(s) copiado(s).
```

Se copia un fichero en un directorio destino distinto y con un nombre de fichero diferente del original:

```
C:\Users\Mindeos> copy pata.txt practicas\patata.txt
1 archivo(s) copiado(s).
```

Se copia un directorio completo con la orden **xcopy**:

```
C:\Users\Mindeos> xcopy practicas C:\practicas
C:\practicas especifica un archivo o un directorio como destino
(F=archivo , D=directorio)? d
practicas\bola.txt
practicas\got.txt
practicas\pata.txt
practicas\patata.txt
practicas\peta.txt
practicas\pelota.txt
practicas\pita.txt
practicas\pota.txt
8 archivo(s) copiado(s)
```

Mover ficheros y directorios

El orden move utiliza para mover archivos y directorios de un lugar a otro del sistema. La sintaxis de la orden move es muy similar a la de copy y también permite el uso de comodines:

```
move origen destino
```

Donde origen y destino pueden hacer referencia tanto a archivos como directorios. Cuando se mueven archivos se puede omitir el nombre del archivo que se quiere mover en el destino, si lo movemos a un directorio diferente, ya que se le asignará el mismo nombre que tenía el original. Cuando se mueven directorios es obligatorio indicar el nombre que tendrá el directorio en el destino (aunque sea el mismo que ya tenía, situado dentro de otro directorio).

Ejemplos de maneras de mover los archivos

Se mueve el archivo document.odt de directorio (al directorio Mis Documentos):

```
C:\Users\Mindeos> move document.odt Documents
Se han movido 1 archivos.
```

Igual, pero con una ruta absoluta. Esta vez el documento es abc.pdf.

```
C:\Users\Mindeos> move \Users\Mindeos\abc.pdf \Users\Mindeos\Documents
Se han movido 1 archivos.
```

Se mueve un directorio completo de sitio:

```
C:\Users\Mindeos> move practicas Documents\practicas
Se ha(n) movido 1 directorio(s).
```

Eliminar ficheros y directorios

Una de las funcionalidades que se requieren cuando se trabaja con archivos y directorios se la eliminación de estos. Hay dos grupos de comandos para llevar a cabo eliminaciones:

- **del** (delete) o erase: permiten la eliminación de archivos individuales.
- **rd** o rmdir (remove directory): permiten la eliminación de directorios, pero sólo es efectivo cuando los directorios están vacíos. Para poder eliminar directorios no vacíos hay que indicar la opción **/s**.

Estas instrucciones permiten la utilización de comodines, al igual que las anteriores.

Ejemplos de eliminación de archivos y directorios:

Se eliminan todos los archivos txt del directorio actual:

```
C:\Users\Mindeos> del *.txt
```

Se intenta eliminar el directorio que no está vacío practicas:

```
C:\Users\Mindeos> rd practicas
El directorio no está vacío.

C:\Users\Mindeos> rd practicas /s
practicas, ¿Estás seguro (S/N)? s
```

Mostrar el contenido de un fichero

Cuando trabajamos con archivos, muchas veces deseamos ver el contenido de un fichero de texto desde la consola, sin necesidad de abrir ningún programa. Para ello el intérprete de comandos nos proporciona el mandato **type**. Con él podemos ver el contenido de cualquier fichero.

Si bien podemos utilizar type con cualquier fichero, realmente sólo entenderemos la información de aquellos ficheros que contienen texto plano. Evidentemente no tiene sentido hacerlo con ficheros de imagen, vídeo, música o cualquier otro formato que no sea texto. Tampoco con ficheros .pdf, ni .doc, ni .odt, … Normalmente ficheros .txt o aquellos que teniendo cualquier otra extensión, su contenido sea texto plano.

La sintáxis es:

type nombre_de_fichero

Se pueden utilizar tanto rutas absolutas como relativas para identificar el fichero.

Ejemplos de mostrar el contenido de un archivo:

Mostrar el contenido del archivo pelota.txt:

```
C:\Users\Mindeos> type pelota.txt
Esto es el contenido del fichero pelota
Es un fichero de texto que se puede mostrar con type
```

Mostrar el contenido del archivo pelota.txt utilizando una ruta absoluta:

```
C:\Users\Mindeos> type \Users\Mindeos\pelota.txt
Esto es el contenido del fichero pelota
Es un fichero de texto que se puede mostrar con type
```

7. Redireccionamientos y tuberías

Cualquier software que ejecutemos en nuestro sistema informático, va a procesar una información que le llega desde una ENTRADA y va a enviar el resultado del proceso a una SALIDA. Si no indicamos nada, se supone que la entrada será desde el dispositivo por defecto de entrada (**stdin**) y la salida será al dispositivo por defecto de salida (**stdout**).

Normalmente en nuestros sistemas, stdin y stdout se refieren a la consola (a la que se referencia en entornos Windows como CON) que está formada por **el teclado como stdin y por el monitor como stdout**. Normalmente, además de stdout, nos encontraremos con otra salida que se llama stderror. Mientras por stdout salen los mensajes de salida normales, por stderror salen los mensajes de salida de error.

Con los redireccionamientos, podemos indicar a las órdenes que entrada, salida y salida de errores deben usar, evitando que usen las estándar. Estos redireccionamientos son los siguientes:

>	Redirecciona stdout. Es decir, nos permite indicar una salida para la orden que no sea el monitor. Normalmente un fichero.
2>	Redirecciona stderror. Es decir, nos permite indicar una salida para los errores de la orden que no sea el monitor.
<	Redirecciona stdin. Es decir, nos permite indicar una entrada para la orden que no sea el teclado. Por ejemplo un fichero de respuestas previamente creado.
>>	Igual que >, pero la salida de la orden se añade a la salida que indiquemos. Con > la salida de la orden reescribe la salida que indiquemos.
|	El indicador de tubería. Nos permite indicar que la entrada de una orden será la salida de otra orden. Es decir, el stdout de la 1ª orden, será el stdin de la 2ª orden.

Veamos algunos ejemplos de estas redirecciones y tuberías. Si escribimos `DIR` veremos como esta orden no nos pide nada (no usa stdin) y nos muestra unas líneas (stdout) por pantalla. Vamos a cambiarle stdout, para ello escribimos `DIR > SALIDA.TXT`. Veremos como por pantalla no nos sale nada, ya que hemos cambiado stdout. Si ahora miramos en el directorio, comprobaremos que se ha creado un fichero SALIDA.TXT que en su interior (`TYPE SALIDA.TXT`) contiene la salida de la anterior orden DIR.

¿Qué ocurriría si escribimos las siguientes órdenes?

```
> ECHO "HOLA MUNDO" > FICHERO1
> ECHO "ESTO ES UN EJEMPLO" > FICHERO1
```

Si ahora miramos el contenido de FICHERO1 (`TYPE FICHERO1`)veremos como solo contiene la ultima línea. Esto es así porque > siempre sobrescribe la salida. Para evitar esto escribimos:

```
> ECHO "HOLA MUNDO" > FICHERO1
```

```
> ECHO "ESTO ES UN EJEMPLO" >> FICHERO1
```

Veamos como funciona la redirección de stdin. Si escribimos la orden `TIME` veremos que esta orden si usa stdin, en concreto nos pide que por teclado introduzcamos la hora en formato HH:MM:SS y pulsemos INTRO para cambiar la hora. Bien, escribamos ahora lo siguiente:

```
> ECHO 15:00:00 > TIME
```

Si ahora escribimos TIME comprobaremos que ya no nos pide nada, pero que la hora no se ha cambiado. ¿Por qué? Muy simple, estamos enviando la salida de una orden como entrada de otra orden, cosa que no se puede hacer con las redirecciones. Hagamos lo siguiente:

```
> NOTEPAD HORA.TXT
```

Nos abrirá el editor de texto con un nuevo fichero que se llama HORA.TXT, dentro de este fichero escribid en la 1ª línea `15:00:00` y en la 2ª línea simplemente pulsad INTRO (dadle entonces a guardar y cerrar). Ahora escribimos la siguiente orden:

```
> TIME < HORA.TXT
```

Comprobamos como ahora si ha funcionado, la hora se ha cambiado a la deseada.

Veamos ahora la redirección para stderror. Si escribimos:

```
> MKDIR ONE TWO THREE TWO
```

El sistema creará los tres primeros directorios, pero nos dará un aviso de error, ya que no se ha podido crear el 4º directorio, ya que ya existe.

Escribimos ahora:

```
> MKDIR GUAN TU TRI TU > SALIDA.TXT
```

Veremos como el error sigue apareciendo, ya que hemos redireccionado stdout, pero no stderror. Escribimos por fin la línea correcta que seria:

```
> MKDIR UNO DOS TRES DOS > SALIDA.TXT 2> ERRORES.TXT
```

Veremos como ahora todo funciona bien. En SALIDA.TXT tendremos la salida normal de la orden, si la hubiera (stdout) y en ERRORES.TXT tendremos la salida de los errores de la orden (stderror).

Usamos la tubería (|) cuando queremos usar la salida de una orden como entrada de la siguiente. Repitamos el ejemplo anterior del echo y el time, pero esta vez con una tubería:

```
> ECHO 14:30:00 | TIME
```

Veremos como ahora si funciona perfectamente. Siempre que en una línea queramos usar la salida de una orden como entrada de la siguiente, debemos usar la tubería, no los redireccionamientos.

8. Otros comandos

Filtros

En todos los sistemas operativos, existen una serie de órdenes especiales conocidas como filtros. Estas órdenes están especialmente diseñadas para trabajar con tuberías, y nos permiten trabajar con la salida de una orden. Entre las principales que podemos encontrar en los sistemas Windows, tenemos:

SORT	Nos permite ordenar una salida alfabéticamente.
FIND	Nos permite filtrar una salida, haciendo que solo aparezcan las líneas que contengan una palabra, las que no contengan una palabra, que contemos las líneas que contienen una palabra, etc.
MORE	Nos permite obtener una salida por pantalla paginada. Es decir, cada vez que la pantalla se llene, nos pide que pulsemos una tecla antes de continuar escribiendo texto.

Así, por ejemplo, si escribimos `HELP` veremos que nos devuelve una gran cantidad de líneas, posiblemente más de las que seremos capaces de ver por pantalla. En este caso podemos escribir `HELP | MORE` para paginar la información.

El comando **sort** lee un fichero de entrada y lo muestra ordenado de manera alfabética por la salida estándar, sin alterar el contenido del fichero original.

Admite varios parámetros, como por ejemplo:

/R Muestra el fichero ordenado en orden inverso.

/+n donde n es un número que indica el carácter a partir del cual ordenará las líneas. /+3 indica que cada comparación debería comenzar en el tercer carácter de cada línea.

Ejemplo:

```
> sort alumnos.txt /r
```

Mostrará por la salida estándar el fichero alumnos ordenado de manera inversa. Este comando sería equivalente:

```
> type alumnos.txt | sort /r
```

El mandato **find** se utiliza para buscar patrones de texto en un fichero. En el mandato hay que especificar la cadena a buscar y el fichero dónde debe buscar. El resultado será que nos mostrará por la salida estándar aquellas líneas del fichero que contengan la cadena patrón. Admite también varios parámetros. Algunos de ellos son:

/I Ignora la diferencia entre mayúsculas y minúsculas entre el fichero y la cadena a buscar.

/V Muestra aquellas líneas del fichero que NO contienen la cadena patrón.

/C No muestra cada línea que contiene la cadena patrón, sino un número con el total de líneas que contienen la cadena patrón.

Ejemplo:

```
> find "Antonio" alumnos.txt /i
```

Mostrará por la salida estándar aquellas líneas del fichero alumnos que contengan la palabra "Antonio". No diferenciará entre mayúsculas y minúsculas. Este comando sería equivalente:

```
> type alumnos.txt | find "Antonio" /i
```

Como ejemplo, vamos a crear con el notepad un fichero tabulado con nombre FAVORITOS.TXT y escribiremos en él, por ejemplo, los nombres de varias páginas Web, sus direcciones y su temática, con este formato:

```
> notepad favoritos.txt
```

Dentro del fichero escribimos las siguientes líneas (utilizando el tabulador):

```
Youtube          www.youtube.com        vídeos
El País          www.elpais.es          periódicos
El Mundo         www.elmundo.es         periódicos
Google           www.google.es          buscador
Bing             www.bing.es            buscador
```

Si ahora escribimos:

```
> TYPE FAVORITOS.TXT | SORT
```

veremos como obtenemos la lista ordenada desde la primera columna, así que se ordenará por el nombre de la página.

Si escribimos:

```
> TYPE FAVORITOS.TXT | FIND "buscador"
```

Veremos como solo nos muestra las líneas donde aparezca la palabra buscador, con lo que es muy fácil filtrar el archivo.

Una línea como la siguiente :

```
> TYPE FAVORITOS.TXT | FIND "periódicos" > PRENSA.TXT
```

Nos crearía un fichero con nombre PRENSA.TXT que contendría todas las líneas de FAVORITOS.TXT donde aparezca la palabra periódicos.

Procesos

Existen dos comandos para trabajar con los procesos que se están ejecutando en el ordenador permitiendo al administrador de sistemas matar procesos que no funcionen bien o que se hayan quedado colgados desde la línea de comandos. Estos comandos son **tasklist** y **taskkill**.

El primero permite ver un listado de todos los procesos que se están ejecutando en ese momento en el sistema.

```
> tasklist

Nombre de imagen              PID Nombre de sesión Núm. de ses Uso de memor
========================= ======== ================ =========== ============
System Idle Process             0 Services                   0         24 KB
System                          4 Services                   0        532 KB
smss.exe                      468 Services                   0         72 KB
csrss.exe                     600 Services                   0      1.484 KB
wininit.exe                   644 Services                   0        612 KB
csrss.exe                     656 Console                    1      9.272 KB
services.exe                  688 Services                   0      2.520 KB
lsass.exe                     700 Services                   0      3.500 KB
lsm.exe                       708 Services                   0      1.780 KB
winlogon.exe                  784 Console                    1      1.188 KB
svchost.exe                   896 Services                   0      3.752 KB
explorer.exe                  588 Console                    1     66.192 KB
SynTPEnh.exe                 1520 Console                    1      1.420 KB
hkcmd.exe                    1576 Console                    1        992 KB
igfxpers.exe                 2136 Console                    1      1.628 KB
dlcxmon.exe                  2156 Console                    1      1.788 KB
memcard.exe                  2164 Console                    1      1.180 KB
AppleMobileDeviceService.    2208 Services                   0      1.356 KB
egui.exe                     2316 Console                    1      2.964 KB
sttray.exe                   2340 Console                    1      1.328 KB
BlueSoleilCS.exe             2440 Services                   0      1.832 KB
mDNSResponder.exe            2480 Services                   0      2.252 KB
dlcxcoms.exe                 2496 Services                   0      4.516 KB
ekrn.exe                     2620 Services                   0     30.676 KB
ehtray.exe                   2676 Console                    1        732 KB
WINWORD.EXE                  3112 Console                    1    123.408 KB
E_FINVLGE.EXE                5452 Console                    1         76 KB
cmd.exe                      4716 Console                    1      2.400 KB
conime.exe                   5568 Console                    1      3.748 KB
tasklist.exe                 5428 Console                    1      5.012 KB
WmiPrvSE.exe                 4840 Services                   0      6.264 KB
```

Con el segundo podemos finalizar procesos en ejecución de varios modos. O bien especificando el número de proceso que deseamos eliminar (PID) o bien filtrando por nombre de proceso, estado o consumo de CPU entre otros. Este comando puede incluso matar procesos de otras máquinas remotas.

Ejemplo:

```
> taskkill /PID 3112
```

Finalizaría el proceso cuyo PID es el 3112 (el Word del listado anterior).

Planificación de tareas

Existe un potente comando para la planificación de tareas denominado **schtasks**. Este comando sustituye al antiguo comando **at**. Se puede utilizar el comando **schtasks** con el fin de programar un comando, una secuencia de comandos o un programa para ejecutarse en una fecha y hora especificados. También podemos utilizar este comando para ver las tareas programadas existentes. Al igual que otros comandos, nos permite trabajar con equipos remotos.

Para utilizar el comando schtasks, el servicio *Programador de tareas* se debe estar ejecutando y debemos iniciar sesión como miembro del grupo local Administradores.

Lo normal es que el servicio *Programador de tareas* se esté ejecutando. Para comprobarlo tecleamos el comando:

```
> net start
```

Y comprobamos que aparece en la lista. En caso de que no estuviera, lo arrancamos:

```
> net start "task scheduler"
```

Se puede ejecutar **schtasks** para crear una tarea programada (parámetro /create), para modificar una tarea programada ya creada (parámetro /change), para ejecutarla inmediatamente (parámetro /run) o simplemente para consultar las tareas planificadas (/quuery).

El comando schtasks utiliza la sintaxis siguiente para crear una tarea programada (se muestran algunos parámetros):

```
SCHTASKS /Create [/S sistema [/U usuario [/P [contraseña]]]] /SC programa [/MO modif.] [/D día] [/M meses] /TN tarea /TR ejecución [/ST hora_inicio] [/RI intervalo] [/SD fecha_inicio] [/ED fecha_fin]
```

Algunos ejemplos:

```
> SCHTASKS /Create /SC DAILY /TN Copia_Seguridad /TR backup /ST 23:00
```

Ejecuta el comando backup.bat (que previamente debería existir) a las 11 pm de cada día de la semana. Nombra a la tarea como Copia_Seguridad.

```
> SCHTASKS /Create /tn "Copia" /tr c:\aps\miap.exe /sc once /st 23:59
```

Ejecuta esta media noche una tarea llamada copia que lanza un aplicación llamada miap

Atributos de ficheros

Existe un comando que muestra, establece o quita los atributos asignados a los archivos o directorios (archivos ocultos, de sólo lectura, del sistema, …). Este comando es **attrib**.

Si se utiliza sin parámetros, attrib muestra los atributos de todos los archivos en el directorio actual.
> attrib

La sintaxis completa de attrib es:

```
attrib [{+|-}r] [{+|-}a] [{+|-}s] [{+|-}h]] [<Drive>:] [<Path>] [<FileName>]
[/s [/d] [/l]]
```

Y algunos de sus parámetros significan:

{+|-}r Establece (+) o quita (-) el atributo de sólo lectura.

{+|-}a Establece (+) o quita (-) el atributo de modificado.

{+|-}s Establece (+) o quita (-) el atributo de archivo del sistema.

{+|-}h Establece (+) o quita (-) el atributo de archivo oculto.

/s Aplica attrib y las opciones de línea de comandos a los archivos coincidentes en el directorio actual y todos sus subdirectorios.

/d Aplica el comando attrib y las opciones de línea de comandos a los directorios.

Ejemplos:

```
> attrib + r informe.txt
```

Asigna el atributo de sólo lectura al fichero informe.txt

```
>  attrib - r f:\public\*.* /s
```

Quita el atributo de sólo lectura de archivos en el directorio public y sus subdirectorios en la unidad F

9. Variables de entorno

El sistema cuenta con sus propias variables, que toman valor cuando se inicia el Sistema. Si queremos ver dichas variables podemos usar la orden SET, que nos muestra una lista de variables ya definidas. Podemos definir nuestras propias variables sin ningún tipo de problemas, basta con poner:

```
> SET nombre_de_variable = valor
```

Es importante no dejar espacios ni delante ni detrás del símbolo =. Asi por ejemplo:

```
> SET EDAD=18
```

crea una variable con nombre EDAD y valor 18.

Si queremos acceder al contenido de la variable, encerramos dicha variable entre símbolos de %.

Ejemplo:

```
> SET NACIONALIDAD="Español"
> ECHO %NACIONALIDAD%
```

Las variables de entorno típicas de un sistema Windows, son las siguientes:

Variable	Tipo	Descripción
%ALLUSERSPROFILE%	Local	Devuelve la ubicación de perfil Todos los usuarios.
%APPDATA%	Local	Devuelve la ubicación en que las aplicaciones guardan los datos de forma predeterminada.
%CD%	Local	Devuelve la cadena del directorio actual.
%CMDCMDLINE%	Local	Devuelve la línea de comandos exacta utilizada para iniciar el Cmd.exe actual.
%CMDEXTVERSION%	Sistema	Devuelve el número de versión de Extensiones del procesador de comandos actual.
%COMPUTERNAME%	Sistema	Devuelve el nombre del equipo.
%COMSPEC%	Sistema	Devuelve la ruta de acceso exacta al ejecutable del shell de comandos.

%DATE%	Sistema	Devuelve la fecha actual. Utiliza el mismo formato que el comando date /t. Generado por Cdm.exe. Para obtener más información acerca del comando date, vea Fecha.
%ERRORLEVEL%	Sistema	Devuelve el código de error del último comando utilizado. Usualmente, los valores distintos de cero indican que se ha producido un error.
%HOMEDRIVE%	Sistema	Devuelve la letra de unidad de la estación de trabajo local del usuario conectada al directorio principal del usuario.
%HOMEPATH%	Sistema	Devuelve la ruta de acceso completa del directorio principal del usuario. Se establece según el valor del directorio principal. El directorio principal del usuario se especifica en Usuarios y grupos locales.
%HOMESHARE%	Sistema	Devuelve la ruta de acceso de red del directorio principal compartido del usuario. Se establece según el valor del directorio principal. El directorio principal del usuario se especifica en Usuarios y grupos locales.
%LOGONSERVER%	Local	Devuelve el nombre del controlador de dominio que validó la sesión actual.
%NUMBER_OF_PROCESSORS%	Sistema	Especifica el número de procesadores instalados en el equipo.
%OS%	Sistema	Devuelve el nombre del sistema operativo. En Windows 2000 se muestra el sistema operativo Windows NT.
%PATH%	Sistema	Especifica la ruta de acceso de búsqueda para los archivos ejecutables.
%PATHEXT%	Sistema	Devuelve una lista de extensiones de archivo que el sistema operativo considera como ejecutables.
%PROCESSOR_ARCHITECTURE%	Sistema	Devuelve la arquitectura de chip del procesador. Valores: x86 o IA64 (basado en Itanium).
%PROCESSOR_IDENTFIER%	Sistema	Devuelve una descripción del procesador.
%PROCESSOR_LEVEL%	Sistema	Devuelve el número de modelo del procesador instalados en el equipo.
%PROCESSOR_REVISION%	Sistema	Devuelve el número de revisión del procesador.
%PROMPT%	Local	Devuelve la configuración del símbolo del sistema del intérprete actual. Generado por Cmd.exe.
%RANDOM%	Sistema	Devuelve un número decimal aleatorio entre 0 y 32767. Generado por Cmd.exe.
%SYSTEMDRIVE%	Sistema	Devuelve la unidad que contiene el directorio raíz del sistema operativo de servidor de Windows (es decir, la raíz del sistema).
%SYSTEMROOT%	Sistema	Devuelve la ubicación del directorio del sistema operativo de servidor de Windows.
%TEMP% y %TMP%	Sistema y usuario	Devuelve los directorios temporales predeterminados que utilizan las aplicaciones disponibles para los usuarios conectados actualmente. Algunas aplicaciones requieren TEMP y otras requieren TMP.
%TIME%	Sistema	Devuelve la hora actual. Utiliza el mismo formato que el comando time /t. Generado por Cdm.exe. Para obtener más información acerca del comando time, vea Time.

%USERDOMAIN%	Local	Devuelve el nombre del dominio que contiene la cuenta de usuario.
%USERNAME%	Local	Devuelve el nombre del usuario que ha iniciado la sesión actual.
%USERPROFILE%	Local	Devuelve la ubicación del perfil del usuario actual.
%WINDIR%	Sistema	Devuelve la ubicación del directorio del sistema operativo.

Algunas de estas variables son especialmente importantes, ya que se nos permiten automatizar muchos procesos de Administración. Por ejemplo, si tenemos que ir al directorio Windows para retocar algunos ficheros y en nuestro servidor disponemos de varios sistemas operativos y varios volúmenes de datos, podemos perder mucho tiempo en buscar donde está situado. Pues un simple `CD %WINDIR%` nos llevaría al directorio de Windows sin posibilidad de error.

Otra variable que usaremos mucho cuando lleguemos al tema de Windows Server será la de %USERNAME%.

¿Cómo podríamos obtener mediante la orden ECHO por pantalla una línea como la siguiente?

```
Hola, usuario JOSE. Ahora mismo son las 13:17:06,45 del día 20/10/2015
y su directorio actual es C:\Users\Jose
```

Por último, una variable de entorno que contiene un valor muy importante es el PATH. El PATH es la ruta por defecto que debe buscar Windows para encontrar un comando, programa o fichero. Este camino puede ser uno o varios directorios. Es importante el orden en que se introduzcan los directorios en el path, pues Windows sigue ese orden para buscar los ficheros en los directorios y en caso de duplicidad prevalecerá uno sobre otro.

Un ejemplo del path sería:

```
>echo %path%
C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\Sys
tem32\WindowsPowerShell\v1.0\;C:\Users\Pepe\AppData\Local\Microsoft\Wi
ndowsApps;
```

Al ejecutar un comando o programa sin especificar dónde se encuentra, Windows irá a buscarlo en primer lugar a C:\Windows\System32, en caso de no encontrarlo lo buscará en el siguiente directorio del PATH, es decir, en C:\Windows, si tampoco está ahí

continuará por C:\Windows\System32\Wbem, y así sucesivamente. En caso de que no estuviera en ningún directorio del PATH informaría que no existe ese programa o comando.

Material elaborado a partir de:

- Administració de la Informació. Implantació de Sistemes Operatius. José Luis Antúnez Reales, Institut Obert Catalunya.
- Apuntes Sistemas Informáticos Monopuesto y Multipuesto. IES Romero Vargas.
- Apuntes Implantación de Sistemas Operativos. José Antonio Carrasco Díaz. IES Romero Vargas.
- Technet de Microsoft. https://technet.microsoft.com/es-es/
- Elaboración propia.