TEMA 6: WINDOWS. MODO COMANDO.

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 (CLI), usando para ello una pantalla de texto plano.

La línea de comandos (CLI) 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 de MS-DOS 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 MSH que no esta basado en texto sino en objetos y que dispone de muchos más comandos que el CMD. Basado en MSH esta disponible el Nomad, aún en versión Beta, que esta llamado a sustituir al CMD y que presenta potentes opciones de scripting (creación de procesos por lotes) y comandos renovados.

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 Inicio y escribir cmd o bien desde el menú Inicio Inicio>Programas>Accesorios>Símbolo del Sistema.

Para configurar el símbolo del sistema:

1. Abrimos Símbolo del sistema.

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.

3. Uso de la ayuda en el shell de comandos

Una de las principales habilidades que debe desarrollar un Administrador de Sistemas, consiste en usar correctamente la ayuda. Cualquier sistema que usemos contará con al menos un nivel de ayuda, que debemos saber buscar e interpretar. En el caso de la línea de comandos, disponemos de una ayuda general accesible mediante la orden HELP. Si queremos ayuda especifica sobre cualquier comando, podemos ejecutar HELP comando.

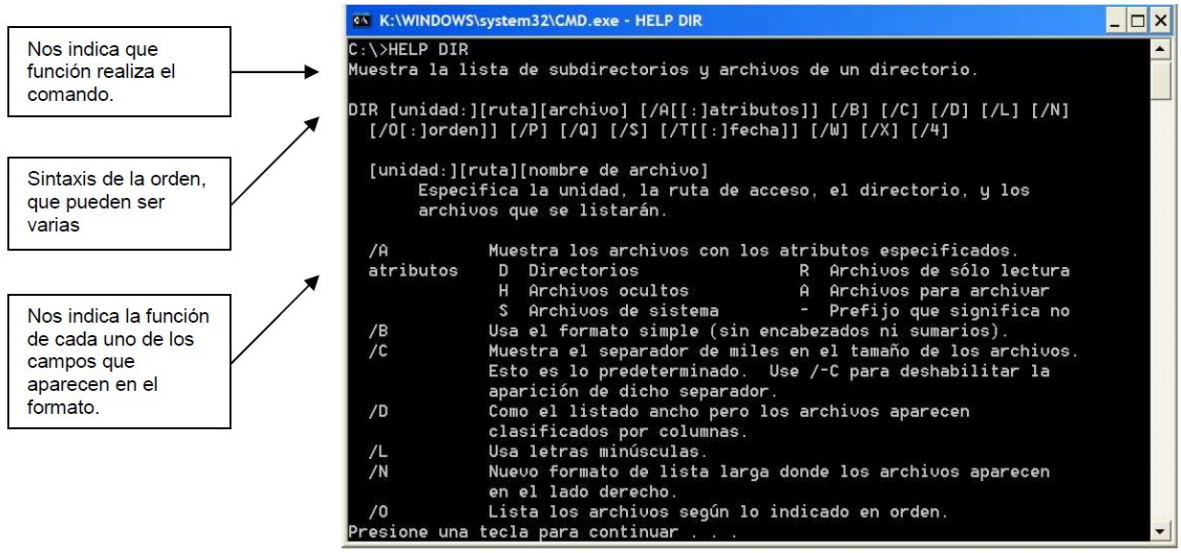
También podemos acceder a la ayuda del comando escribiendo *comando /?*

En caso de que la ayuda que obtengamos con HELP no nos sea suficiente, podemos acceder a la ayuda de Windows 7 a la que podemos llegar desde el botón Inicio, Ayuda y Soporte Técnico, donde en Buscar indicaremos el nombre de comando del que deseamos información.

Si tampoco aquí encontramos lo que buscamos, podemos acceder a Internet. Fuentes importantes son la Knowledge Base de Microsoft (entrad en http://www.microsoft.com/spain y allí seleccionad Knowledge Base) y el TechNet de Microsoft (http://www.microsoft.com/spain/technet/)

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.

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}

Vamos a insistir en lo que se ha explicado, para asegurarnos de que se entiende bien.

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

Veamos que información obtenemos de esta línea, y que significan los caracteres que ahí aparecen.

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 esta libre, así que el formato mínimo de la orden seria DIR.

Todo lo que esta encerrado entre corchetes indica que es optativo. Así por ejemplo, el modificador /A es optativo, pero veamos como 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 seria /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.

Si aprendemos a usar correctamente una pantalla de ayuda, entender lo que esta escrito en ella y lo que se nos quiere decir, habremos dado un paso de gigante para lograr ser Administradores de Sistemas.

4. Utilizar 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)

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:

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:

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

glos?.doc

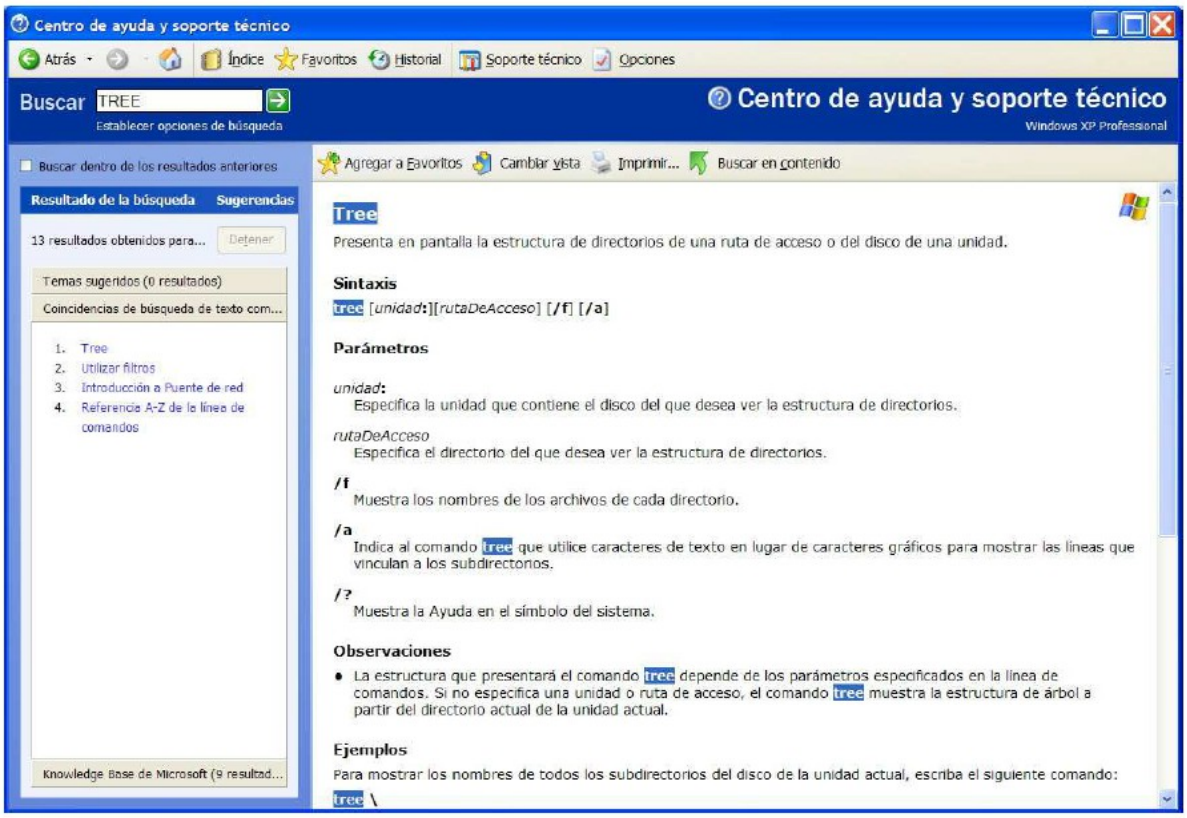
Encontraremos los archivos Glosa.doc y Glos1.doc, pero no Glosario.doc.

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.

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

Si esta ayuda no nos es suficiente, podemos acceder al centro de ayuda y soporte técnico de nuestro Windows 7 (Inicio – Ayuda y Soporte Técnico) y escribir el comando en el formulario de búsqueda.



Veamos ahora algunos de estos comandos:

6.1. Gestión de directorios

- **Comando: DIR**

  Función: Lista todos los ficheros y directorios de la ruta en la que nos encontramos. Mediante parámetros podemos modificar ese listado.

  Sintaxis: DIR [unidad\directorio\fichero]

  Parámetros: Algunos de los parámetros que se pueden utilizar para modificar el listado de archivos y directorios son los siguientes:

  - **/P** Muestra pantalla por pantalla el listado, para visualizar la pantalla siguiente basta con pulsar una tecla. Al pulsar una tecla se procesará el siguiente bloque de listado y así sucesivamente.
  - /**O** Ordena por el orden especificado:
  - **/ON** Ordena por nombre en formato de lista detallada
  - **/OE** Ordena por extensión
  - **/OS** Ordena por tamaño

- **/OD** Ordena por fecha
- **/OG** Ordena poniendo agrupados todos los directorios después de los ficheros.
- **/O-X** Ordena inversamente por el orden especificado por X
- **/S** Muestra los archivos del directorio especificado y todos sus subdirectorios.

- **Comando: CD**

  Función: Permite cambiar de un directorio activo a otro.

  Sintaxis: CD [unidad:]\[ruta]\[directorio]

  Observación: Si deseamos subir un nivel en el árbol de directorios, sólo es necesario escribir **cd ..**

- **Comando: MD ó MKDIR**

  Función: Crear directorios

  Sintaxis: MD [unidad\ruta\]<nombre>

  Observación: Si intentamos crear un directorio que ya estuviera creado nos da error.

- **Comando:** RD

  Función: Borra un directorio (sólo si se encuentra vacío).

  Sintaxis: RD [unidad\ruta\]<nombre>

  Parámetros: Los parámetros que se pueden utilizar con este comando son:
  - **/S** Elimina todo el directorio a borrar aunque no esté vacío, pero pide confirmación.
  - /**Q** No pide confirmación para eliminar un árbol de directorios cuando se utiliza junto con la opción /S.

6.2. Gestión de archivos

- **Comando: TYPE**

  Función: Ver el contenido de archivos de texto, haciendo un listado (no permite el uso de comodines).

  Sintaxis: TYPE [unidad:]\[ruta]\[directorio]\<archivo>

- **Comando: COPY**

  Función: Copia el fichero origen al fichero destino.

  Sintaxis: copy <fichero-origen> <fichero-destino>

- **Comando: REN o RENAME**

  Función: Renombra un fichero. Dará error si existe un fichero que tenga el mismo nombre dentro del mismo directorio. Con este comando se pueden utilizar los comodines del dir.

  Sintaxis: rename <nombre-actual> <nombre-nuevo>

- **Comando: MOVE**

  Función: Este comando mueve ficheros de un directorio a otro.

  Sintaxis: move [/Y] <origen> <destino> donde /Y es un parámetro que poniéndolo, el move moverá ficheros sin preguntar la confirmación de reemplazo a otros archivos que se puedan llamar de la misma forma al directorio de destino. En caso de no especificarse, MSDOS pedirá una confirmación de reemplazo de ficheros.

- **Comando: DEL ó ERASE**

  Función: Se encarga de borrar uno o varios archivos. Se pueden utilizar comodines.

  Sintaxis: del [unidad:]\[ruta]\[directorio]\<archivo>

  Parámetros: Algunos de los parámetros que se pueden utilizar con este comando son:

  - **/P** Pide confirmación antes de eliminar cada archivo.
  - /**F** Fuerza la eliminación de los archivos de sólo lectura.

- **Comando: FC**

  Función: Compara dos ficheros y verifica si las copias son exactas.

  Sintaxis: FC fichero1 fichero2

- **Comando: XCOPY**

  Función: Copia un directorio entero con subdirectorios y ficheros incluidos. Recibe el nombre del directorio a copiar y opcionalmente el directorio destino. Admite también una serie de parámetros.

  Sintaxis: xcopy origen [destino]

  Parámetros: Los parámetros que admite son:

  - /**P** : Pide confirmación de SI o NO antes de copiar cada archivo.
  - /**S** : Indica que la copia debe hacerse extensiva a todos los subdirectorios, exceptuando los vacíos.
  - /**E** : Igual que con el /**S**, sólo que copia también los directorios vacíos.
  - /**W** : Espera la confirmación antes de copiar los archivos.
  - /**V** : Verifica si el archivo se ha copiado correctamente.
  - /**C**: Continúa copiando aunque ocurran errores.
  - /**Q**: No muestra los nombres de los archivos mientras está copiando.

- **Comando: EDIT**

  Función: Editor de texto de MSDOS con una interfaz gráfica. Permite visualizar cualquier archivo que contenga texto.

  Sintaxis: edit [unidad:]\[ruta]\[directorio]\<archivo.ext (solo de texto)>

  Ejemplo: c:\>edit fichero.txt

Con esto se nos abriría el editor de texto del MSDOS con el archivo fichero.txt. En caso de que el nombre de archivo pasado como parámetro al edit no exista, el edit abrirá un archivo nuevo en blanco almacenado con ese nombre pero de forma temporal, con lo cual luego hay que guardarlo a través del menú que ofrece el edit.

Atributos de los ficheros

Los atributos permiten asociar a los ficheros características especiales. Estas características de los ficheros son:

1. **De sólo lectura (R):** Protege la escritura y el borrado de un fichero. El fichero sólo puede ser leído.
2. **De archivo (A) :** Sirve para saber si un determinado fichero ha sido o no

modificado.

3. **Oculto (H):** Sirve para ocultar un fichero. Por ejemplo algunos archivos del sistema son ocultos.

4. **De sistema (S) :** Sirve para identificar los archivos propios del sistema, los cuales sirven para cargar el SO.

Para visualizar o modificar los atributos de un fichero se utiliza el comando attrib.

- **Comando: ATTRIB**

  Función: Visualiza o modifica los atributos de un fichero.

  Sintaxis: attrib <fichero> <+/-></h/s/a/r> (modificará los atributos de un fichero determinado)

  Obs: Con + establecemos un atributo a un fichero y con – lo borramos. Si no damos ningún parámetro lo que hacemos es visualizar los atributos.

6.3. Discos

- **Comando: FORMAT**

  Función: Formateado, o borrado completo de un disco o disquete.

  Sintaxis: format <unidad:>

  Parámetros: Si se quiere realizar un formato rápido se le puede dar el parámetro /**Q**.

- **Comando: CHKDSK**

  Función: Comprueba el estado de un disco o disquete correspondiente a la unidad introducida como parámetro y muestra un informe de su estado.

  Sintaxis: chkdsk [unidad:] [fichero]

  Obs: Se puede utilizar con la opción /**F** ó /**R** que se encargan de encontrar los sectores dañados y recupera la información legible en el disco.

- **Comando: DISKCOPY**

  Función: Copia el contenido total de un disco o disquete, sirve para hacer copias de seguridad. La unidad origen y destino puede ser la misma, por ejemplo cuando copiamos un disquete.

  Sintaxis: diskcopy <unidad_origen:> <unidad_destino:>

- **Comando: LABEL**

  Función: Permite crear, cambiar o borrar el nombre de la etiqueta que tiene asignado un disco o disquete cuando éste es formateado. La etiqueta es un nombre asignado por el usuario para identificar el disco o disquete. Si ponemos sólo label, nos visualizaría el nombre de la etiqueta de la unidad actual. Si especificamos la etiqueta la asignaría a la unidad actual.
  Sintaxis: label [unidad:] [etiqueta de volumen]

- **Comando: VOL**

  Función: Muestra la etiqueta y el número de serie que se le es asignado al disco o disquete, si están especificados. Este número de serie no es modificable porque el sistema tiene que identificar de forma única el disco o disquete.
  Sintaxis: vol [unidad:]

6.4. Otros comandos

- **Comando: CLS**

  Función : Sus iniciales vienen de clean screen (limpiar pantalla). Su uso es muy simple, se introduce el comando se pulsa intro, y se limpia la pantalla.
  Sintaxis: cls

- **Comando: DATE**

  Función: Nos muestra en pantalla la fecha almacenada por el sistema. En una línea posterior nos facilita la posibilidad de cambiar la configuración de la fecha del sistema.
  Sintaxis: date

- **Comando: TIME**

  Función: Nos muestra en pantalla la hora almacenada por el sistema. En una lína posterior nos facilita la posibilidad de cambiar la configuración de la hora del sistema
  Sintaxis: time

- **Comando: COLOR**

  Función: Establece los colores de primer plano y fondo predeterminados de la consola.

  Sintaxis: color [atr]

- **Comando: EXIT**

  Función: Abandona el programa CMD.EXE (intérprete de comandos) o el archivo de comandos por lotes actual.

  Sintaxis: exit [/B] [codigo]

  Parámetros:

  - /B :especifica que se debe abandonar el archivo de procesos por lotes actual y no CMD.EXE.
  - código :especifica un número. Si se ha especificado /B, estable ERRORLEVEL con este número. Si abandona CMD.EXE, establece el código de salida del proceso con este número.

- **Comando: SHUTDOWN**

  Función: Apaga o reinicia el equipo.

  Sintaxis: shutdown

  Parámetros:

  - -s: Apaga el equipo
  - -r: Reinicia el equipo
  - -t xx: Establece el siguiente tiempo de espera expresado en segundos.
  - -c “Comentario”: Establece el siguiente comentario.

- **Comando: VER**

  Función: Muestra en pantalla la versión del sistema operativo.

  Sintaxis: ver

7. Redireccionamientos

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 esta 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 Standard. Estos redireccionamientos son los siguientes:

>	Redirecciona stdout. Es decir, nos permite indicar una salida para la orden que no sea CON (monitor).
2>	Redirecciona stderror. Es decir, nos permite indicar una salida para los errores de la orden que no sea CON (monitor).
<	Redirecciona stdin. Es decir, nos permite indicar una entrada para la orden que no sea CON (teclado).
>>	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 > PATATA. 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 PATATA que en su interior (TYPE PATATA) 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 veremos como solo contiene la ultima línea. Esto es asi 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:

EDIT 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 escribid 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.

Escribid ahora MKDIR GUAN TU TRI TU > SALIDA.TXT

Veremos como el error sigue apareciendo, ya que hemos redireccionado stdout, pero no stderror. Escribid 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. 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:

- MORE: Visualiza el resultado de una orden páginas a página si el resultado es mayor que la pantalla.

- SORT: Se encarga de ordenar los datos de entrada. Por defecto los ordena según la primera letra de cada fila de los datos de salida.

Sintaxis: SORT [/R] [/+numero]

/R: Indica a MSDOS que debe invertir el orden de la ordenación (de mayor a menor).

/+numero: Indica a MSDOS que en lugar de ordenar por el primer carácter, lo haga por el carácter

que ocupe la posición que se le especifique.

/T: Redirige la salida al fichero especificado después de esta opción.

- FIND: Localiza una cadena de caracteres dentro de un fichero. Este filtro recibe como entrada datos dispuestos en filas y devuelve sólo aquellas filas que contienen la secuencia de caracteres o la cadena que se ha especificado en la orden.

Sintaxis: FIND [parámetro] “cadena” fichero

/V: Muestra todas las líneas que no contienen la cadena.

/C: Muestra sólo el número de líneas que contienen la cadena.

/I: No distingue mayúsculas y minúsculas.

/N: Muestra las líneas de texto con la cadena y los números de esas líneas.

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 esta situado. Pues un simple CD %WINDIR% nos llevaría al directorio de Windows sin posibilidad de error.

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

¿Como pequeño ejercicio, como 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 09/11/2005
y su directorio actual es M:\Documents and Settings\Jose
```

10. Archivos por lotes

Dentro de un fichero podemos escribir una lista de comandos para ser ejecutados uno detrás de otro de forma automática. La forma más sencilla de crear un archivo por lotes es usando el programa **EDIT**. A este fichero habrá que ponerle de extensión **.BAT**, para que el sistema operativo lo identifique como script o fichero por lotes.

Para pasar parámetros o argumentos a los comandos y ficheros por lotes, usamos %1, para hacer referencia al primer parámetro posicional, %2 hará referencia al segundo, y así sucesivamente.

Dentro de los ficheros porlotes podemos usar una serie de comandos especiales para poder realizar ciertas funciones.

Comando	Función
CALL	Llama a otro fichero por lotes
ECHO	Muestra mensajes, o activa y desactiva el eco.
@	No muestra el eco en una línea.
FOR	Ejecuta un comando para cada archivo en un conjunto de archivos.
GOTO	Direcciona el intérprete de comandos a otra línea
IF	Ejecuta un proceso condicional.
PAUSE	Suspende el proceso y muestra un mensaje.
REM	Marca la línea como de comentario.
SHIFT	Desplaza los parámetros posicionales hacia la izquierda.

- **Comando REM:** Este comando nos sirve para poner comentarios en nuestros programas.

Ejemplo:

```
REM  Esto es un comentario.
```

- **Comando ECHO:** Si lo que queremos es escribir un texto en pantalla usaremos el comando ECHO, el cual también nos sirve para realizar saltos de línea. Si no queremos que salga la línea a ejecutar en pantalla, y que luego se ejecute, en la primera línea del fichero bat pondremos **echo off**.

Ejemplo:

```
ECHO Esto es un comentario que sale por pantalla.
ECHO.
ECHO La linea anterior hace un salto de linea.
```

- **Comando CALL:** Nos permite llamar a otro fichero por lotes. Cuando un fichero bat llama a un segundo fichero, se ejecutará este segundo fichero, volviendo al primer fichero donde se había quedado.

Uno.bat

```
ECHO Uso del parámetro CALL.
ECHO.
call dos.bat
ECHO otra vez estamos en uno.bat
```

Dos.bat

```
ECHO programa dos.bat.
ECHO.
Dir C:\windows\command /w
```

- **Comando IF:** Al igual que en programación, la sentencia IF sirve para evaluar una condición. La forma general de esta sentencia es: IF <cadena1> <operador> <cadena2><sentencia a ejecutar>

Ejemplo.bat

```
IF   ' %1 '  ==  '%1'          cls
IF '%1' NOT = '%2'             dir /p
IF NOT EXIST C:\DOS            MD C:\DOS
```

La sentencia IF también admite el formato **IF NOT**

- **Comando GOTO**: Casi de manera inseparable a la sentencia If usaremos la sentencia GOTO, que lo que hace es un salto hasta la parte del programa donde le hayamos creado la etiqueta a la que hagamos referencia. Las

etiquetas se crean usando : y a continuación el nombre de la etiqueta, por tanto para ir hasta ellas usaremos la sentencia GOTO seguida del nombre de la etiqueta.

```
Ejemplo2.bat
rem mejora sobre el ejemplo anterior.

IF   ' %1 ' == '1'                goto opcion1
IF   ' %1 ' == '%2'               goto opcion2
IF  EXIST C:\DOS\HELP.HLP   goto opcion3

:opcion1
cls
goto fin

:opcion2
dir /p
goto fin

:opcion3
ECHO El fichero existe.

:fin
echo programa finalizado.
```

Comando PAUSE: Nos sirve para detener la ejecución del programa hasta que el usuario pulse una tecla. Mostrará por pantalla el texto “Pulsa una tela para continuar ….”

Ejercicios resueltos:

Crea un archivo por lotes uno.bat que borre la pantalla, te diga que empieza el programa, muestre la fecha, el contenido del directorio actual, haga una pausa, borre de nuevo la pantalla, muestre el contenido del directorio raíz, haga otra pausa y diga que termina el programa.

Solución

```
@echo off
cls
echo Empieza
echo Directorio actual
dir
echo Fecha
```

```
date /t
pause
cls
echo Directorio \
dir \
echo Final
```

Comando FOR: Ejecuta un comando sobre un grupo de archivos.

Puede utilizarse en la línea de comandos o en un archivo bat:

a) en archivos BAT:

FOR %%variable IN (set) DO command [command-parameters]

b) en línea de comandos:

FOR %variable IN (set) DO command [command-parameters]

Parametros

%%variable %variable
: Representa una variable que será reemplazada por su valor. FOR reemplazará %%variable o %variable con la cadena de carácteres especificados en SET hasta que el comando especificado se haya ejecutado sobre todos los archivos. %%variable se emplea con FOR dentro de archivos batch, y %variable desde la linea de comandos.

(set)
: especifica uno o mas archivos de texto (o cadenas) que se procesaran con el comando. Necesita paréntesis.

command
: El comando que debe ejecutarse sobre cada archivo especificado en SET

parámetros de comando
: Podemos emplear el comando con cualquiera de sus parámetros habituales.

EjemploFor.bat

```
REM Visualiza el contenido de todos los archivos del directorio actual con extensión .txt
@echo off

for %%x IN (*.txt) DO type %%x
```

Comando SET: Muestra, establece o quita las variables de entorno de cmd.exe.

SET **[**variable=**[**cadena**]]**

variable Especifica el nombre de la variable de entorno.

cadena Especifica una serie de caracteres que se asignará a la variable.

Escriba SET sin parámetros para ver las variables de entorno actuales.

También podemos utilizar el comando SET de la siguiente forma:

SET /P variable=[promptString]

El modificador /P permite establecer el valor de una variable para una línea de entrada escrita por el usuario. Para referenciar a la variable en el programa se escribe %variable%

Ejemplo:

```
Echo “Introduzca su nombre”
Set /P nombre=Introduzca su nombre
Echo Buenos dias %nombre%
```