Introducción al batch. Comandos básicos.

Contenido

¿Qué es el lenguaje batch?

El lenguaje batch está formado por una serie de comandos MS-DOS que son guardados en un archivo de procesamiento por lotes normalmente con formato .bat. Este lenguaje nos puede servir para crear programas sencillitos usando simplemente programas de edición de texto como el bloc de notas.

Este lenguaje es muy famoso también porque era usado hace tiempo por los hackers para crear algunos virus muy sencillos pero a la vez de los peores, como por ejemplo uno que formatea el ordenador o que hace que se reinicie sólo al encenderse, pero estos virus ya no funcionan porque son bloqueados fácilmente por Windows.

Pues bien, abramos el bloc de notas, y vamos a empezar viendo los comandos que encontraremos en todo lenguaje batch. El lenguaje batch siempre empieza con el comando "@echo off" y termina con el comando "exit". De esta forma en la primera línea de nuestro programa vamos a tener el comando "@echo off", y cuando el programa ejecute el comando "exit", que no tiene por qué estar al final, el programa se cerrará.

Además le tenemos que poner un título a nuestro programa, que aparecerá en la barra de título de la ventana. Este título se pone con el comando "title título" (y donde pone título pondríamos el título del programa). A este primer programa le vamos a llamar "Primer programa", por lo que nuestro código va así:

```
@echo off
title Primer programa
exit
```

Comandos de texto (echo) y pause

Vamos a empezar con un programa que haga algo básico, y eso básico es mostrarnos frases, texto. Pues bien, para mostrar texto tenemos el comando **echo**:

Se puede utilizar de cinco formas:

ECHO	Indica si está activado o desactivado el eco de las órdenes.
ECHO ON	Activa el eco
ECHO OFF	Desactiva el eco
ECHO mensaje	Visualiza un mensaje en pantalla.
ECHO.	Visualiza una línea en blanco en pantalla.

De esta forma si queremos que nuestro primer programa tenga un primer párrafo en el que ponga "Este es el primer párrafo." y un segundo en el que ponga "Este es el segundo párrafo. Un poquito mas largo." tendríamos nuestro código así:

```
@echo off
title Primer programa
echo Este es el primer parrafo.
echo.
echo Este es el segundo parrafo. Un poquito mas largo.
exit /b
```

Ahora quiero recalcar dos cosas. La primera es que no podemos usar tildes ni eñes en el texto que se muestra en el programa. La segunda es que teniendo en cuenta la estructura del programa se ejecutará todo del tirón, llegará al comando **exit** sin pausa alguna y no te dará tiempo a leer el texto. Los comandos de pausa son los siguientes:

1. **pause** El comando pause pausa el programa hasta que pulses una tecla y te muestra un mensaje que dice "Presione una tecla para continuar".
2. **pause >nul** El comando pause >nul pausa el programa pero sin mostrar ningún mensaje de aviso.

Vamos a probar estos dos comandos, poniendo un pause >nul entre los dos párrafos y un pause después del segundo:

```
@echo off
title Primer programa
echo Este es el primer parrafo.
echo.
pause >nul
echo Este es el segundo parrafo. Un poquito mas largo.
pause
exit
```

Comentarios para el programador.

➢ **REM** Este comando se utiliza cuando queremos introducir un comentario que servirá como recordatorio o explicación para el programador. No se mostrará en pantalla. Servirá para explicar el código creado. Se puede poner en cualquier línea del código, tantas veces como sea necesario.

Programa de ejemplo:

```
@echo off
rem Programa: TIEMPO.BAT
rem Cometido: Mostrar al usuario la fecha y la hora del
rem sistema y ofrecerle la oportunidad de modificarla.
Date
time
```

Comando goto y etiquetas.

Muchas veces vamos a necesitar dividir nuestro programa en partes (a partir de ahora las llamaremos menús), y que dependiendo alguna condición vaya a un menú u otro. Más adelante veremos un ejemplo de para qué sirve esto.

➢ **GOTO Y ETIQUETAS** Para crear un menú pondremos en una línea el código :nombre (con los dos puntos), de forma que indicamos que ahí empieza un menú que se llama nombre. Luego usaremos el código goto nombre para ir al menú nombre. De esta forma vamos a hacer como ejemplo un programa que al empezar tenga un menú llamado frases en el que escriba dos frases, y luego tenga un comando goto frases, de forma que el programa vuelve atrás al menú frases y nunca acaba (esto se llama bucle). Este programa sería por ejemplo así:

```
@echo off
title Bucle de ejemplo
color 0A

:frases
echo 100101001001010010010 Creado por mi 100101001001010010010
echo 011010110110101101101 Creado por mi 011010110110101101101

goto frases
exit /b
```

Ejecutar el programa. Parámetros.

Para poder probar nuestro programa tenemos que ejecutarlo, para lo que hay que guardarlo en formato bat. Esto lo hacemos dándole a guardar como y poniendo el nombre que queramos terminado en .bat, por ejemplo "programa.bat". Entonces nos aparecerá el archivo con el icono de un engranaje, y ese será nuestro programa que podremos abrir para ver el resultado.
Para poder editar posteriormente nuestro programa le daremos un clic derecho y a editar.

Un parámetro es una información que acompaña a la invocación en la línea de comandos de un programa batch. Esa información puede ser variable. La forma de referenciar a esos parámetros es "%numero", como máximo un 9. Antes de la ejecución de un programa batch los parámetros se sustituyen por los valores.

• programa.bat pepe jose entonces %1 = pepe y %2 = jose
• %0 contiene el nombre del programa

Vamos a crear un programa nombre.bat que muestre el nombre y apellidos de un usuario. El nombre y los apellidos entran como parámetros en el fichero bat.

```
@echo off
title Programa con parámetros
echo Te llamas %1 %2 %3
echo.
pause >nul
```

Si el usuario no escribe parámetros el programa no los escribirá. Tendremos que mejorar el código de forma que compruebe si se han introducido parámetros y si no se han introducido explique al usuario que los debe escribir.

```
@echo off
title Programa con parámetros
If "%1"=="" goto explica
echo Te llamas %1 %2 %3
echo.
pause >nul
goto fin
:explica
cls
echo Para escribir su nombre escriba el nombre del programa bat y luego su nombre y
apellidos.
echo.
goto fin
:fin
echo Terminado.
exit /b
```

Los códigos de salida o códigos de retorno son un numerito que devuelve el sistema operativo tras la realización de algunas operaciones. Si el programa recibe un 0 como código de retorno significa que la operación ha ido bien, si recibe otro número suele ser que ha habido problemas. Los códigos de retorno se suelen evaluar con una variable llamada ERRORLEVEL que veremos más adelante.

Gestión de parámetros

➢ **SHIFT** Este comando se utiliza cuando queremos introducir mas de 9 parámetros, básicamente lo que hace es ir desplazando el contenido de cada contenedor de variables hacia la izquierda.
Este es un bucle que muestra los parámetros introducidos, pero se para cuando los ha mostrado todos.

```
@ECHO OFF
REM Muestra los parametros introducidos de 1 en 1 y sin limite de numero de
parametros
:INICIO
IF "%1" == "" GOTO FIN
REM si no metemos ningun parametro finaliza el programa
ECHO %1
SHIFT
REM con SHIFT desplazamos el contenedor de variables una posicion
REM solo visualiza los parametros introducidos de uno en uno y aunque sea mas de 9
GOTO INICIO
:FIN
ECHO Proceso terminado
```

Colores. Trabajando con variables.

Cambiar colores

➢ **COLOR** Antes de empezar con una de las herramientas más usadas de la programación como son las variables vamos a aprender a cambiar los colores tanto del texto en nuestro programa como del fondo. Para ello usaremos el código color numeroletra, y donde pone numero pondremos el color de fondo de la ventana del programa y donde pone letra el color del texto. Las correspondencias entres los números y letras y los colores son las siguientes:
0 = Negro 1 = Azul 2 = Verde
3 = Aguamarina 4 = Rojo 5 = Púrpura
6 = Amarillo 7 = Blanco 8 = Gris
9 = Azul claro B = Aguamarina C = Rojo
A = Verde claro E = Amarillo F = Blanco
D = Púrpura

Vamos a hacer por ejemplo que en nuestro primer programa al principio se ponga el fondo rojo y las letras amarillas, y que cuando vaya a aparecer el segundo párrafo aparezca el fondo negro y las letras color aguamarina.

```
@echo off
title Primer programa
color 4E
echo Este es el primer parrafo.
echo.
pause >nul
color 0B
echo Este es el segundo parrafo. Un poquito mas largo.
pause
exit
```

¿Qué es una variable?

Una variable es un elemento al que asignamos un nombre, por ejemplo var1, var2 o pepe, y al que asignamos también un valor, que puede ser numérico o con letras. Las variables se suelen usar para hacer operaciones entre valores numéricos, o para recordar datos que diga el usuario.

Por ejemplo, podemos hacer que el programa pregunte un número al usuario, que guarde ese número en una variable, que le pregunte otro y que lo guarde en otra variable, y que luego cree otra variable que sea la media de las otras dos variables. Esta sería la forma de hacer la media entre dos números cualesquiera.

Crear variables

➢ **SET** Para crear una variable usamos el comando set var=algo, siendo lo que está en rojo el nombre de la variable, y lo que está en verde el valor que le asignamos a la misma.

Algo importante es que cuando queramos hacer referencia a una variable tendremos que ponerla entre signos %. De esta forma si queremos mostrar en pantalla una variable que se llame var tendremos que poner el código echo %var%.
Al igual hay que tener en cuenta que si asignas un valor a una variable que ya existía esa variable tendrá el valor que se le asignó último.
Asignar un valor fijo a una variable no tiene mucha utilidad. Algo que tiene más utilidad es hacer una pregunta al usuario y lo que responda será el valor de la variable. Esto último se haría usando el comando set/p var=pregunta

De esta forma podemos hacer un programa que te pregunte el nombre y los dos apellidos y luego lo muestre todo seguido de la siguiente forma:

```
@echo off
title Nombre y apellidos
set/p nombre=Cual es tu nombre?
set/p apellido=Cual es tu primer apellido?
set/p apellido2=Cual es tu segundo apellido?
echo.
echo Tu nombre completo es %nombre% %apellido% %apellido2%
pause >nul
exit /b
```

Operaciones con las variables.

Por último vamos a aprender a hacer operaciones con las variables. Para hacer operaciones usamos el comando siguiente:
set/a var1= %var2% signo %var3%

Lo que está en rojo es la variable en la que guardará el resultado, las variables verdes son las que opera, y donde pone signo pondremos un + si queremos sumar, - para restar, * para multiplicar o / para dividir. Un ejemplo sería:
set/a var= %var1% / %var2%

Eso lo que hace es dividir las dos variables. También podemos hacer operaciones con una variable y un número, por ejemplo:
set/a var= %var1% * 2

Este último multiplica la variable var1 por dos.

Sabiendo esto vamos a hacer el programa que hace la media de dos números. Tenemos que hacer que le pregunte los dos números, y luego que los sume y acto seguido los divida entre dos (en las operaciones

podemos poner paréntesis para indicar el orden, en el ejemplo lo veréis mejor). De esta forma nuestro programa quedaría así:

```
@echo off
title Hacer media de dos numeros
set/p num1=Cual es el primer numero?
set/p num2=Cual es el segundo numero?
set/a media= (%num1% + %num2%) / 2
echo.
echo La media es %media%
pause >nul
exit
```

He de recalcar que en una variable numérica no puede haber números decimales, los quita directamente, por lo que si la media saliera por ejemplo 2,5 el programa te dirá que la media es 2.

Comandos if y else, for, choice y call.

Comandos if y else (establecer una condición)

Sintaxis de if:

IF EXIST fichero orden	Si existe el fichero se ejecuta la orden.
IF NOT EXIST fichero orden	Si no existe el fichero se ejecuta la orden.
IF cadena1==cadena2 orden	Si ambas cadenas son iguales se ejecuta la orden.
IF NOT cadena1==cadena2 orden	Si ambas cadenas son diferentes se ejecuta la orden.
IF ERRORLEVEL número orden	Si el código de salida del último programa es igual o superior al número, se ejecuta la orden.
IF NOT ERRORLEVEL número orden	Si el código de salida del último programa es inferior al número, se ejecuta la orden.

➢ **IF** El código if nos sirve para hacer que en el caso de que algo sea cierto o falso haga otra cosa. Por poneros un ejemplo, imaginaos que queremos hacer la división de dos números, pero si el denominador es 0 no podemos hacerla, puesto que algo entre 0 es infinito. Por lo tanto tenemos que verificar que el número de abajo no sea 0 para que no dé error, y si es 0 que le vuelva a preguntar el número. Este programa lo crearemos ahora de ejemplo.

Pues bien, el código de if tiene esta estructura:

if %var%==valor (comando) **else** (comando2)

Esto hace que si la variable llamada "var" tiene el valor "valor" ejecute el código "comando", y si no tiene ese valor que realice el valor "comando2".
También podemos eliminar la segunda parte si no la necesitamos y dejar el código de la siguiente forma: if %var%==valor (comando)
De esta forma vamos a hacer el programa de dividir un número entre otro:

```
@echo off
title Division de dos numeros
set/p uno=Escriba el numerador
```

```
:denominador
set/p dos=Escriba el denominador

if %dos%==0 (goto nosepuede) else (goto dividir)

:nosepuede
echo El denominador no puede ser cero.
goto denominador

:dividir
set/a tres=%uno%/%dos%
echo El resultado de %uno%/%dos% es %tres%
pause>nul
exit
```

Ahora vamos a recalcar un par de cosas. Primero que podemos sustituir el comando if por if not, de forma que realiza el comando si la condición no es verdad.

Lo segundo es que si os fijáis usamos == en vez de un solo =. Cuando comparamos dos cosas tenemos que usar siempre el doble signo ==.
Además de los signos igual podemos usar las siguientes cosas:

%var% == valor Verifica si var es igual al valor
%var% LSS valor Verifica si var es menor que el valor
%var% GTR valor Verifica si var es mayor que el valor
%var% LEQ valor Verifica si var es menor o igual al valor
%var% GEQ valor Verifica si var es mayor o igual al valor

Para finalizar vamos a hacer un ejemplo más que será un programa para hacer la media, pero esta vez no será entre dos números, sino que los elegirá el usuario. Este programa tendría la siguiente estructura:

```
@echo off
title Hacer media de dos numeros
:principio
set/p cant=Entre cuantos numeros quiere hacer la media?
set/a contador=%cant%
set total=0

if %cant% LSS 2 (goto mal) else (goto pregunta)

:mal
echo Debe indicar un numero mayor o igual que 2
goto principio

:pregunta
set/p num=Escriba uno de los %cant% numeros
set/a total=%total%+%num%
set/a contador=%contador%-1
if not %contador% == 0 (goto pregunta)

set/a media= %total% / %cant%
echo.
echo La media es %media%
pause >nul
exit
```

También podemos usar IF para comprobar si existen o no ficheros.
IF EXIST “nombre fichero” comando. Si existe ese fichero ejecutará la orden.

```
@echo OFF
REM Comando IF
REM Buscar si un fichero existe
```

```
IF NOT EXIST %1 GOTO NOEXISTE
ECHO El fichero %1 existe
ECHO Y el contenido de %1 es:
TYPE %1
GOTO FIN
:NOEXISTE
ECHO El fichero %1 no existe
:FIN
ECHO Programa terminado
```

IF ERRORLEVEL Cada orden externa de ms-dos genera un código de salida a su término indicando si pudo realizarse satisfactoriamente.
Generalmente un código de salida 0 indica que no hubo ningún problema y un código de salida superior hace referencia a diferentes errores.
Muchos ficheros por lotes necesitan saber si la orden anterior cumplió su cometido correctamente: para ello utilizan la orden If errorlevel.
Es muy importante recordar que la orden se ejecutará si el código de salida es igual o superior al especificado detrás de ERRORLEVEL.
A modo de ejemplo tenemos a continuación los códigos de salida de Xcopy:

Código	Significado
0	Los ficheros fueron copiados sin error.
1	No se encontraron ficheros para copiar.
2	El usuario presionó Ctrl+Pausa para suspender el proceso de Xcopy.
4	Ocurrió un error de inicio. No hay suficiente memoria o espacio en el disco, se introdujo un nombre de unidad no válida o se utilizó una sintaxis incorrecta en la línea de órdenes.
5	Ocurrió un error de escritura de disco.

Vamos a crear un fichero por lotes para copiar los ficheros de la unidad A: a la B: e informe del resultado de la copia.

```
@echo off rem Programa: COPIA-AB.BAT
xcopy /y a:\ b:\
if errorlevel 1 goto :Error
if errorlevel 0 goto :bien

:bien
cho ¡La copia fue correcta!
goto :Final

:Error
echo Se produjo un error durante la copia
:Final
echo Fin del proceso
```

En primer lugar, xcopy intenta realizar la copia de ficheros y devolverá un código de salida. Si se ha producido algún error el código será 1 o superior y entonces, el programa se desvía hasta la etiqueta :Error, muestra el mensaje y finaliza. Si la copia fue satisfactoria, el código de salida es 0. La segunda línea If mostrará el mensaje de éxito, saltando después a la etiqueta :Final y como no hay más líneas, termina el proceso.
En muchas ocasiones puede ser fuente de complicaciones que If errorlevel número se cumpla si el número es igual o mayor.
Para cumplirse exclusivamente si el código de salida es 5 -por ejemplo- podemos usar lo siguiente:

```
if errorlevel 5 if not errorlevel 6 dir
```

Esta compleja línea se traduce así: «Si el código de salida es 5 o superior pero inferior a 6 ejecutar Dir», es decir, si el código es 5 ejecutar Dir.

Llamada a otro subprograma.

➢ **CALL** Sirve para llamar a un programa batch (esclavo) desde otro programa batch (principal) de tal manera que una vez terminada la ejecución del programa esclavo se sigue con la ejecución del programa principal en el punto que se realizo la llamada CALL.

Gestión de menús.

➢ **CHOICE** Se utiliza para gestionar las opciones de un menú. Ofrece al usuario un conjunto de teclas para poder seleccionar las opciones de un menú que previamente se ha construido manualmente mediante sentencias ECHO, además permite escribir un mensaje aclaratorio pudiéndose omitir o no la relación de teclas sensibles. También es posible pasado un tiempo determinado establecer una opción por defecto. Cada tecla sensible permitida generara un código de retorno que podrá ser evaluado por el ERRORLEVEL.

CHOICE [mensaje] [/C:opciones] [/N] [/S] [/T:opción,segundos]

/C:opciones	Especifica las opciones posibles. Si el usuario pulsa la primera de las opciones, Choice devolverá un código de salida 1; si pulsa la segunda opción, Choice devuelve el código 2 y así sucesivamente. Si no se especifica este parámetro se asumen las opciones por defecto (SN).
/N	No muestra las opciones admitidas detrás del mensaje.
/S	Hace distinción entre mayúsculas y minúsculas. Si no se especifica este parámetro se toman como la misma opción.
/T:opción,segs	Toma la opción indicada si no se pulsa ninguna otra tecla en los segundos especificados.
mensaje	Contiene el mensaje mostrado al usuario pidiendo que introduzca una de las opciones admitidas.

Con la orden Choice y de una forma muy sencilla podemos crear menús con diferentes opciones:

```
@echo off
rem Programa: UTIL.BAT
:Menu
cls
echo UTILIDADES DE MS-DOS
echo -------------------- echo.
echo A. Anti-Virus
echo B. Backup
echo D. Defragmentar
echo E. Editor
echo S. Salir
echo.
choice ¿Qué utilidad desea comenzar? /c:abdes /n /t:s,15
if errorlevel 5 goto Salir
if errorlevel 4 goto Editor
if errorlevel 3 goto Defrag
if errorlevel 2 goto Backup
if errorlevel 1 goto Anti
if errorlevel 0 goto Menu
:Anti
MSAV
goto Menu
:Backup
MSBACKUP
```

```
goto Menu
Defrag DEFRAG
goto Menu
:Editor
EDIT
goto Menu
:Salir
echo.
```

Estructura de bucle.

➢ **FOR** Se utiliza para diseñar una estructura repetitiva. La sintaxis es la siguiente:

FOR %%variable IN (conjunto) DO orden
Esta orden repite la orden especificada para cada valor del conjunto. Conjunto es una lista de nombres de ficheros. En ella, se pueden establecer varios nombres separados por espacios y también, utilizar comodines.

Ejemplo:

for %%I in (juan.txt maria.txt *.dat) do type %%i

La variable %%I va tomando cada uno de los valores del conjunto y se los envía a la orden Type. En este ejemplo se visualizan en pantalla los ficheros JUAN.TXT, MARIA.TXT y todos los que tengan extensión DAT.

Variables predefinidas

Nosotros no podemos definir todos los nombres de variables que queramos, ya que algunos nombres ya vienen cogidos por nuestro sistema operativo y tienen su función correspondiente. En esta parte vamos a nombrar algunos de ellos, ya que puede que en algunos programas requiráis una lectura de los mismos. Estas variables son:

1. %ALLUSERSPROFILE% => Esta variable devuelve la localización del perfil de todos los usuarios.
2. %APPDATA% => Devuelve el lugar donde las aplicaciones guardan los datos por defecto (Normalmente la carpeta de Archivos de Programa).
3. %CD% => Devuelve el directorio en el que estás en ese momento (se explicará más adelante con sus comandos para cambiarse de directorio).
4. %CMDCMDLINE% => Muestra el comando exacto empleado para acceder al intérprete de comandos (cmd.exe).
5. %CMDEXTVERSION% => Devuelve la extensión de nuestro intérprete de comandos.
6. %COMPUTERNAME% => Devuelve el nombre del equipo.
7. %COMSPEC% => Devuelve la ruta de la shell de comandos.
8. %DATE% => Devuelve la fecha actual.
9. %ERRORLEVEL% => Devuelve el código de error del último comando ejecutado.
10. %HOMEDRIVE% => Devuelve la unidad en la que está el directorio en el que estás actualmente.
11. %HOMEPATH% => Devuelve la ruta completa a dicho directorio.
12. %LOGONSERVER% => Devuelve el nombre de nuestro servidor.
13. %NUMBER_OF_PROCESSORS% => Devuelve el número de procesadores instalados en el equipo.
14. %OS% => Devuelve nuestro sistema operativo, con la excepción de Windows 2000 y XP que lo devuelven como Windows_NT.
15. %PATH% => Devuelve la ruta a la carpeta de los ejecutables más importantes del sistema.
16. %PATHEXT% => Devuelve las extensiones de archivos que nuestro sistema considera ejecutables.
17. %PROCESSOR_ARCHITECTURE% => Devuelve la arquitectura del procesador.
18. %PROCESSOR_IDENTIFIER% => Devuelve la descripción del procesador.

19. %PROCESSOR_LEVEL% => Devuelve el número de modelo de procesador.
20. %PROCESSOR_REVISION% => Devuelve el número de revisión del procesador.
21. %PROGRAMFILES% => Devuelve la carpeta donde se guardan los programas (normalmente Archivos de Programa).
22. %RANDOM% => Devuelve un número al azar entre 0 y 32767.
23. %SYSTEMDRIVE% => Devuelve la unidad que contiene el directorio raíz del sistema.
24. %SYSTEMROOT% => Devuelve la carpeta de administración, que suele ser C://Windows
25. %TEMP% => Devuelve el directorio donde están los archivos temporales.
26. %TMP% => Igual que el anterior.
27. %TIME% => Devuelve la hora actual.
28. %USERNAME% => Devuelve el nombre del usuario actual.
29. %USERPROFILE% => Devuelve la ruta del directorio donde están los archivos del usuario actual.
30. %WINDIR% => Devuelve la ruta de la carpeta del sistema operativo

De esta forma nosotros podemos hacer por ejemplo un programa que nos muestre la hora actual, la fecha y el usuario que la ha visto de la siguiente forma:

```
@echo off
title Hora actual
echo El usuario %USERNAME% ha solicitado ver la hora
pause
echo La fecha es %DATE% y son las %TIME%
pause>nul
exit
```

Trabajando con procesos y carpetas

Trabajando con procesos

En este capítulo vamos a centrarnos en lo que sería los procesos y archivos de windows. Vamos a empezar centrarnos en la ejecución de programas y procesos, y vamos a ver los comandos más utilizados en batch para eso.

Primero vamos a ver el comando de taskkill /parámetro que se utiliza para matar un proceso, es decir, para terminar un proceso. Donde pone parámetros podemos poner cualquiera de los parámetros de la lista siguiente:

1. /S sistema : Especifica el sistema remoto al que conectarse.
2. /U usuario : Especifica el usuario en el que se terminará el proceso.
3. /P contraseña : Especifica la contraseña de dicho usuario. Si este parámetro no se pone y el usuario tiene contraseña nos la pedirá.
4. /F : Fuerza al proceso a cerrarse.
5. /PID Identidad : Especifica la identidad del proceso a cerrar.
6. /IM nombre : Especifica el nombre del proceso a cerrar.
7. /T : Termina el arbol de procesos del proceso indicado (es decir, todos los procesos iniciador por el).

Vamos a suponer por ejemplo que en alguna línea de nuestro programa queremos cerrar todas las ventanas abiertas del Internet Explorer forzosamente y también los procesos iniciados por el mismo, pues entonces esa línea sería así:

Taskkill /IM iexplore.exe /F /T

Además de finalizar procesos también podemos iniciar un programa o cualquier cosa, y esto se haría con el comando start programa, y donde pone programa podemos poner una de las siguientes cosas:

1. Un programa o archivo. En este caso el código sería por ejemplo start C:Windows\system32\mspaint.exe , que nos abriría el paint.

2. Una página web. En este caso lo pondríamos así: start http://www.tuwebdeinformatica.com . Esto es lo que usa por ejemplo los famosos virus adware.
3. Tu programa de envío de emails, para mandar un email a alguien: start contacto@hotmail.com

Trabajando con carpetas/directorios

El programa siempre va a estar localizado en una carpeta, de forma que si le decimos que cree un archivo (lo veremos en el siguiente capítulo) lo hará en la carpeta por defecto. El directorio que tendrá por defecto nuestro programa será la carpeta en la que se encuentre. Podemos leer el directorio en el que se encuentre localizado el programa con el comando CD. Además este comando puede tener atributos que nos permiten desplazarnos en el directorio:

Vamos a crear de ejemplo un programa que se vaya trasladando por las carpetas de Windows y nos muestre el lugar en el que se encuentre en cada momento:

```
@echo off
CD
pause>nul
CD C:Windows\system32
CD
pause>nul
CD ..
CD
pause>nul
CD..
CD
pause>nul
exit
```

Sólo tenemos que ejecutarlo, ir pulsando enter y observar cómo se va trasladando por las carpetas.