Programación BATCH – Procesamiento por lotes

Introducción

El procesamiento por lotes o programación batch se define como **archivos de texto sin formato**, guardados con la **extensión .bat** que contienen un **conjunto de comandos MS-DOS**. Cuando se ejecuta este archivo .bat, los **comandos contenidos son ejecutados** en grupo, de **forma secuencial**, permitiendo **automatizar diversas tareas**. **Cualquier comando de MS-DOS** puede ser utilizado en un archivo batch.

Hay que aclarar que batch o procesamiento por lotes no es un lenguaje de programación. Es un archivo de código que contiene **comandos del shell o intérprete de comandos de MS-DOS**, que en Windows también puede ejecutarse desde el CMD.

Hay dos maneras de ejecutar comandos batch:

- Desde el shell o intérprete de comandos directamente escribiendo el comando.
- Escribiéndolos en un archivo de texto con extensión .bat y luego ejecutar dicho archivo.

Mi primer programa BATCH

Comenzaremos creando el clásico "Hola Mundo", escribiendo en un archivo de texto el siguiente código.

```
@ECHO OFF
ECHO Hola Mundo
PAUSE
EXIT
```

Ahora guardemos el archivo como **nombre.bat** y lo ejecutamos. Nos aparecerá la pantalla que dirá "Hola Mundo", luego una línea abajo dirá "Presione una tecla para continuar". Pero, ¿qué hace cada instrucción?

- **ECHO** → Imprime un texto en pantalla, que es el texto que viene después (que pasamos como parámetro), que en este caso es "Hola Mundo". ECHO significa eco, por lo mismo @ECHO OFF elimina el eco, la repetición de la ruta en la que nos encontramos en cada línea código.

- **PAUSE** → Realiza una pausa la ejecución del código. Además, muestra el texto "Presione una tecla para continuar".

- **EXIT** → Cierra la ventana de comandos.

```
@ECHO OFF
ECHO Hola, cuando pulses la tecla, se borrará el contenido.
PAUSE
CLS
ECHO ¿Ves?
PAUSE
EXIT
```

Si quisiéramos averiguar más sobre las funciones, o conocer algunas nuevas por nuestra propia cuenta podemos escribir en la consola **el nombre del comando seguido de /?** y aparecerá toda la información necesaria. Podemos poner en práctica esto escribiendo las instrucciones en la línea de comando de MS-DOS.

- **@ECHO OFF** → Para eliminar el eco.
- **CLS** → Para borrar el contenido de pantalla.
- **COPY /?** → Para ver la ayuda de la función COPY.
- **PAUSE** → Crear una pausa a la espera de pulsar una tecla.
- **EXIT** → Cerrar la ventana.

Ahora vamos a crear un archivo batch que copie un archivo, luego lo mueva a otro directorio, borre el original y vuelva a copiar el copiado. En la **carpeta** donde vas a **crear el archivo .bat**, **crea un archivo de texto llamado copiame.txt**. En él, escribe lo que quieras, ya que será éste el archivo que copiaremos.

```
@ECHO OFF
ECHO Hola, copiaremos un archivo
PAUSE
COPY copiame.txt copiado.txt
MOVE copiado.Ttxt C:\copiado.tx
DEL copiame.txt
CD C:\
COPY copiado.txt copiado2.txt
EXIT
```

Guárdalo en la carpeta como **ejemplocopias.bat** y ejecútalo. Lo que ha hecho la cadena de instrucciones es más o menos lo siguiente:

1. Primero apagamos el eco, luego imprimimos en pantalla "Hola, copiaremos un archivo".
2. Pausamos el intérprete para que no continúe sin nuestro consentimiento.
3. A continuación, copiamos el archivo **copiame.txt** y su copia se llamará **copiado.txt**.
4. Después, la movemos a la carpeta raíz del disco C, y eliminamos el original para, finalmente, cambiar de directorio al disco C, copiar la copia y cerrar el batch.

Después de que ejecutemos el batch, si vamos al disco C:, veremos que hay dos nuevos archivos de texto: uno llamado **copiado.txt** y otro **copiado2.txt**. También si vamos a la carpeta del batch, veremos no está el archivo de texto original, **copiame.txt**.

Creación de nuevos archivos

Con ficheros batch podemos crear otros archivos. Para ello, tenemos que escribir lo siguiente:

```
ECHO TEXTO DE PRIMERA LINEA > nombre.extensión
ECHO TEXTO SEGUNDA LINEA >> nombre.extensión
ECHO TEXTO SIGUIENTE LINEA >> nombre.extensión
```

Al escribir ECHO seguido de un texto y la redirección, creamos un archivo con el nombre indicado. Si escribimos ECHO seguido del texto y doble redirección, se escribirá en la siguiente línea vacía del archivo.

```
ECHO Creando un archivo de texto > nuevo.txt
ECHO Esta es la segunda línea >> nuevo.txt
ECHO Y esta es la siguiente >> nuevo.txt
ECHO Esto está genial >> nuevo.txt
```

Personalizando el shell

El shell o intérprete de comandos puede personalizarse de varias formas. Con el comando **COLOR** podemos cambiar los colores de fondo y las letras. O con **TITLE** para cambiar el título de la barra de tareas.

- **COLOR AB** → Donde A es el color del fondo y B es el color de las letras.
- **TITLE título** → Escribimos la orden seguida del título deseado.
- **PAUSE** → Si queremos que al escribir PAUSE no salga el texto "Presione una tecla para continuar..." podemos escribir `PAUSE > NUL` y no aparecerá ningún texto. Entonces podemos anteponer un mensaje con el comando ECHO.

```
ECHO Pulsa cualquier tecla para continuar
PAUSE > NUL
```

Además, podemos jugar con los caracteres. Vemos un ejemplo donde se personalizan PAUSE y COLOR.

```
@ECHO OFF
COLOR 30
ECHO ==================================
ECHO =                                =
ECHO =          PERSONALIZADO         =
ECHO =                                =
ECHO ==================================
ECHO.
ECHO.
ECHO Esto está personalizado, para salir presiona una tecla.
PAUSE > NUL
EXIT
```

Mencionar que **ECHO.** (es decir, ECHO seguido de un punto) sirve para escribir una **salto de línea**.

Mensajes aclaratorios o comentarios

Una **tarea fundamental** al realizar un **desarrollo informático** consiste en la adecuada **documentación** del mismo. Los programadores suelen incluir numerosos comentarios en el código fuente de sus programas, de cara a facilitar **futuras revisiones de los mismos**.

La inclusión de comentarios en el código se consigue mediante el **comando REM.** Un comentario puede ser cualquier cadena alfanumérica de hasta 123 caracteres de longitud. Cuando el sistema operativo encuentra **una línea REM ignora su contenido**, continuando con la ejecución de la siguiente orden.

Otro comando perteneciente a esta categoría es ECHO, que sirve para mostrar mensajes al usuario o para visualizar por pantalla todas las órdenes incluidas en el fichero. ECHO puede tomar dos valores, **ON** y **OFF**. El valor por defecto es el primero (ON), en cuyo caso **se presentan las órdenes del fichero en el prompt** a medida que se vayan ejecutando. Si no queremos que esto suceda, tenemos que ponerlo a **OFF**.

Por otra parte, ECHO interpreta los caracteres "|", "<" y ">" como caracteres especiales (operadores de redirección), por lo que si queremos que aparezcan en un cierto mensaje tendremos que recurrir a **escribirlos entre dobles comillas**. Otro carácter problemático es "**%**", utilizado en los ficheros batch para **introducir parámetros**, que tiene que ser **tecleado dos veces cuando queramos verlo en pantalla**.

Como ECHO no se desactiva hasta que no lo ponemos a OFF, la primera línea de todo fichero batch debería visualizarse siempre. Pero esto no es así, ya que podemos utilizar el **comando @**. Si situamos dicho carácter justo en la **primera posición** de **cualquier línea**, ésta **quedará oculta aunque ECHO** esté a **ON**.

Utilización de parámetros

Los ficheros batch tienen la posibilidad de utilizar el indicador "**%N**" (donde **N es un número entre 0 y 9**) para **admitir parámetros de entrada**. Esto nos puede servir para **crear ficheros multiuso**. Podemos especificar **hasta 10 parámetros de entrada** (%0 %1 %2 hasta %9), de los cuales el "**%0**" **se sustituye siempre por el nombre del fichero .bat**.

Supongamos que hemos creado el fichero llamado **mueve.bat** que sirve para mover todos los ficheros con una determinada extensión desde la unidad C: hasta la D: cambiándolos de extensión:

```
COPY C:\*.%1 D:\*.%2
DEL *.%1
CLS
```

Un ejemplo de uso puede ser **MUEVE TXT DOC**, que realiza dos funciones diferentes: primero copia los ficheros TXT de la unidad C: a la D: cambiándoles la extensión y posteriormente borra los iniciales.

Un comando de utilidad a la hora de trabajar con parámetros es **SHIFT**, que sirve para **desplazar el contenido de un parámetro una posición a la izquierda**. Es decir, el parámetro **%0 tomaría el valor del %1**, el %1 tomaría el del %2, etc. Así sería posible, en cierto modo, trabajar con más de diez parámetros.

```
@ECHO OFF
ECHO El parámetro 1 es %1
ECHO El parámetro 2 es %2
ECHO El parámetro 3 es %3
SHIFT
ECHO Ahora el parámetro 1 es %0
ECHO Ahora el parámetro 2 es %1
ECHO Ahora el parámetro 3 es %2
ECHO Por último, el parámetro 4 es %3
PAUSE
```

```
C:/>MOSTRAR A B C D
El parámetro 1 es A
El parámetro 2 es B
El parámetro 3 es C
Ahora el parámetro 1 es A
Ahora el parámetro 2 es B
Ahora el parámetro 3 es C
Por último, el parámetro 4 es D
```

Terminando procesos

Cuando presionamos la combinación de teclas CTRL + ALT + SUPR se abre el **Administrador de tareas**. En la pestaña **Procesos** aparecen aquellos procesos que se están ejecutando en el equipo. Seleccionando alguno de ellos y pinchando en **Terminar proceso**, podemos finalizar su ejecución. Con fichero batch esto es muy fácil de realizar, usando la orden **TASKKILL**. Con **TASKLIST** podemos ver también el listado de todas las tareas en ejecución, incluidos los servicios.

Este comando no funciona en todas las versiones de MS-DOS, así que para ver si está disponible escribimos **TASKKILL /?**.

`TASKKILL /F /IM PROCESO.EXE` → El parámetro **/F** fuerza el término del proceso.

`TASKKILL /F /IM WMPLAYER.EXE` → Si está abierto el reproductor de Windows Media, se cerrará.

Iniciando procesos

Ahora vamos a abrir ese proceso que cerramos. Para abrir procesos se usa la función **START.**

`START PROCESO.EXE` → También puedes abrir páginas web con tu explorador predeterminado.

`START WMPLAYER.EXE` → Se abrirá el reproductor de música

`START WWW.GOOGLE.COM` → Se abrirá tu explorador de Internet la página de Google.

Definición de variables

Para crear una se escribe: **SET NOMBREVARIABLE=VALOR**, **sin espacios**. Pero, para **llamarla** (que sea sustituida por su valor), debemos escribir su nombre **entre los signos %**, es decir, para mostrar su valor.

```
@ECHO OFF
SET NOMBRE=Juan
ECHO %NOMBRE%
PAUSE
```

Para cambiar el valor solo debemos hacer esto: **SET Nombre=Nuevo Valor.**

Si queremos que la variable sea **dinámica**, o sea, **el usuario indica su valor**, utilizaremos el modificador **/P**:

```
SET /P NOMBRE=ESCRIBA SU NOMBRE:
```

La ejecución del fichero batch se pausará mostrando el mensaje que aparece en el código después del signo = de la orden SET, dejando que el usuario escriba algo.

También podemos hacer que la **variable** sea el **resultado de una operación aritmética** como una suma, resta, multiplicación, etc.

```
SET NUMERO1=2
SET NUMERO2=43
SET /A SUMA=%NUMERO1% + %NUMERO2%
ECHO %SUMA%
PAUSE
EXIT
```

Si ejecutamos este script nos devolverá la suma de 2 + 43, es decir, 45.

Uso de etiquetas

La lectura de los comandos que conforman un archivo tipo batch se produce de una manera lineal, pero existe una forma para que el fichero se salte líneas, o vuelva a ejecutar alguna anterior. Esto se logra con **etiquetas y la función GOTO**. Para crear una etiqueta solo debemos escribir:

:nombreEtiqueta

Dos puntos (:) seguido del nombre que le daremos a la etiqueta. De esta manera, el comando GOTO funciona escribiendo GOTO seguido del nombre de la etiqueta, como vemos en el siguiente ejemplo.

```
@ECHO OFF
GOTO :MiEtiqueta
ECHO ¿POR QUE ME SALTAN?
:MiEtiqueta
ECHO HOLA, ÉSTA ES LA ETIQUETA Y NOS SALTAMOS UNA PARTE DEL CÓDIGO.
PAUSE > NUL
EXIT
```

Estructura condicional IF

El comando **IF** se encarga de comparar cadenas, números o ficheros y directorios para saber si existen. Su sintaxis en el **caso de cadenas de texto** es: **IF %CADENA1% == %CADENA2% ORDEN_EJECUTAR**. Se puede interpretar de la siguiente manera: **si cadena1 es igual a cadena2 ejecutamos la orden** que aparece a continuación. También existe **NOT** para ver **si no son iguales**. Su sintaxis en ese caso sería la siguiente: **IF NOT %CADENA1%== %CADENA2% ORDEN_EJECUTAR**.

Para comprobar si un **archivo existe**, su sintaxis cambia: **IF EXIST ARCHIVO ORDEN_EJECUTAR**. También podemos comprobar que no existe: **IF NOT EXIST ARCHIVO ORDEN_EJECUTAR**.

Para comprobar si un **variable está definida**, su sintaxis cambia: **IF DEFINED VARIABLE ORDEN_EJECUTAR**. También podemos comprobar si no existe al igual que en el caso anterior.

Para realizar comparaciones de tipo aritmético (números) empleamos los siguientes comparadores:

- **EQU** → Si los números son iguales =.
- **NEQ** → Si los números son diferentes, desigualdad <>.
- **LSS** → Si el número es menor <.
- **LEQ** → Si el número es menor o igual <=.
- **GTR** → Si el número es mayor >.
- **GEQ** → Si el número es mayor o igual >=.

```
IF 5 GTR 4 ECHO Hola
```

En caso de que la **condición se compruebe cierta**, se escribirá “Hola” por pantalla. En caso contrario, no.

También podemos establecer que se ejecuten uno o varios comandos en caso de que la comparación sea evaluada a falso mediante la estructura **ELSE**. En el ejemplo a continuación, **en caso de existir** el fichero **fichero.txt** se muestra un mensaje por pantalla y se elimina. **En caso de NO existir**, se mostrará un mensaje.

```
IF EXIST fichero.txt (
   ECHO Eliminar fichero.txt
   DEL fichero.txt
) ELSE (
   ECHO No se encuentra el fichero.
)
```

Estructura condicional CHOICE

Ofrece al usuario una entrada de datos para que pueda escoger una opción (escoger una de las letras que se le ofrecen), y espera hasta que ésta tenga lugar.

El **modificador /M** permite especificar un **texto que aparecerá antes de la entrada de datos**. No hace falta entrecomillarlo, salvo que dicha cadena de texto incluya una barra (/).

El **modificador /C** es un parámetro opcional y **especifica las letras que indicarán las opciones del usuario**. Si se especifican separadas por comas, aparecerán entre corchetes seguidas de un interrogante. Si no se especifican, se usara YN (sí, no) por defecto.

```
CHOICE /C CD /M Seleccione [C] CD-ROM o [D] Disco duro
```

Para las opciones de CHOICE se utilizará la variable de entorno ERRORLEVEL, donde cada una de las opciones es un número, comenzando por 1. En el ejemplo anterior sería:

```
IF %ERRORLEVEL% EQU 1 ECHO Has seleccionado el CD.
IF %ERRORLEVEL% EQU 2 ECHO Has seleccionado el disco duro.
```

Estructuras de repetición – Bucle FOR

El comando FOR nos sirve para **repetir un comando varias veces**, en distintas variables. Su sintaxis puede variar según aquello que deseemos realizar. Por ejemplo: **FOR /L %%I IN (0, 1, 21) DO (ECHO %%I)**.

- El **modificador /L** es para especificar que nuestro **FOR se basa en un contador de números**.
- Se realiza un ECHO, que imprimirá números del 0 al 21.
- La variable **%%I hace referencia al nombre de la variable** que almacenara los datos.
- Con **IN (0, 1, 21)** especificamos que se repetirá del **0 al 21 y de 1 en 1**. Inicio, salto y fin.
- Finalmente, **DO (ECHO %%I)** será el comando o comandos que se ejecutarán en cada repetición del bucle FOR. En este caso, el comando ECHO que imprimirá la variable.

Otro uso que tiene también el b**ucle FOR es la capacidad de listar carpetas o archivo**. Para listar archivos:

```
@ECHO OFF
FOR %%X IN (A*) DO ECHO %%X
PAUSE
```

Este código, se **listan todos los archivos que comienzan con A**, ya que estamos utilizando el signo asterisco (*), que es un comodín. Otro ejemplo de uso comodines es listar archivos con extensión común, como el que aparece a continuación, que nos estaría **listando todos los archivos con la extensión .jpg**.

```
@ECHO OFF
FOR %%X IN (*.JPG) DO ECHO %%X
PAUSE
```

Si quisiéramos **listar todos los documentos** que están dentro de una carpeta contando los que están dentro de sus **subcarpetas, utilizaríamos /R**.

```
FOR /R %%X IN (A*) DO ECHO %%X
```

Para el **listado de carpetas** es casi idéntico al anterior, tan solo que al **indicar /D** solo se aplica a directorios. Si quisiéramos que se listaran los **directorios junto a los subdirectorios que contienen sería con /R**.

```
FOR /D %%X IN (A*) DO ECHO %%X
FOR /R /D %%X IN (A*) DO ECHO %%X
```

También podemos **leer un fichero separado por comas** (.csv o .txt) con el bucle FOR y extraer información. Las variables se generan de forma automática a partir de la que aparece en el bucle (%%G).

```
FOR /F "tokens=1,3 delims=;" %%G IN (clima.csv) DO ECHO %%G %%H
```

token = 1 (%%G)	token = 2 (ignorado)	token = 3 (%%H)
Enero		2
Febrero		15
Marzo		25

En un archivo separado por comas de la forma siguiente:

```
Enero;Nieve;2
Febrero;Lluvias;15
Marzo,Nubes y claro;25
```

Delims indica el delimitador, en este caso el ; (podría ser cualquiera, hasta varios) y **tokens** los ítems a leer.

Creación de menús

Ahora veremos la creación de menús. Habitualmente, se comenzamos indicando las instrucciones y las opciones disponibles:

```
@ECHO OFF
:MENU
CLS
ECHO Seleccione su opción tecleando el número correspondiente.
ECHO.
ECHO 1. Primera opción
ECHO 2. Segunda opción
ECHO 3. Salir
```

Luego, para continuar, nos ocuparíamos de cada una de las opciones:

```
SET /P VAR=
IF %VAR%==1 GOTO :PRIMERO
IF %VAR%==2 GOTO :SEGUNDO
IF %VAR%==3 GOTO EXIT
IF %VAR% GTR 3 ECHO ERROR
GOTO :MENU
```

Con esto, creamos una variable en la que se guardará la opción seleccionada para luego ser verificada por los condicionales. En el caso de que no exista dicha opción, se nos avisará. También, al principio tenemos **una etiqueta** para que sea posible volver al menú y la orden **CLS** para limpiar la pantalla. Para terminar creamos las respectivas etiquetas según las opciones.

```
:PRIMERO
CLS
COLOR A
ECHO Ésta es la primera opción
```

```
ECHO Presione una tecla para volver al menú
PAUSE > NUL
GOTO :MENU
:SEGUNDO
CLS
COLOR 1A
ECHO Ésta es la segunda opción
ECHO Presione una tecla para volver al menú
PAUSE > NUL
GOTO :MENU
```

Finalmente, el código quedaría de la siguiente manera:

```
@ECHO OFF
:MENU
CLS
ECHO Seleccione su opción tecleando el número correspondiente.
ECHO.
ECHO 1. Primera opción
ECHO 2. Segunda opción
ECHO 3. SALIR
SET /P VAR=
IF %VAR%==1 GOTO :PRIMERO
IF %VAR%==2 GOTO :SEGUNDO
IF %VAR%==3 GOTO EXIT
IF %VAR% GTR 3 ECHO ERROR
GOTO :MENU
:PRIMERO
CLS
COLOR A
ECHO Esta es la primera opción
ECHO Presione una tecla para volver al menú
PAUSE > NUL
GOTO :MENU
:SEGUNDO
CLS
COLOR 1A
ECHO Esta es la segunda opción
ECHO presione una tecla para volver al menú
PAUSE > NUL
GOTO :MENU
```

Cada sección puede tener su color e incluso podemos cambiar la barra de título.

Apagado y reiniciado del equipo

Con un fichero batch es posible apagar y reiniciar el ordenador, incluso programarlo para que se apague a la hora deseada. La sintaxis de este comando es: **SHUTDOWN -S -T TIEMPO -C "Comentario"**.

- Donde **-S significa que lo apagaremos**.
- La **opción -T indica los segundos** que tardará en apagarse.
- La opción **-C hace que se muestre un comentario** por pantalla.

Es posible cancelar el apagado antes de que el contador termine escribiendo SHUTDOWN -A o bien en un archivo de texto con extensión .bat, o bien en el mismo intérprete de comandos.

```
SHUTDOWN -S -T 999999 -C "ESTO SE ESTÁ APAGANDO…"
SHUTDOWN -A
```

Si en vez de escribir -S **escribimos -R**, el equipo se **reiniciará** en vez de apagarse. También se puede **forzar** el cierre de los programas, para que no dé tiempo a guardar nada utilizando la **opción -F**.

Otras instrucciones útiles

Vamos a ver otras funciones que pueden ser también de utilidad.

- **MSG** → Crear una alerta con un mensaje. Su utilización es la siguiente:
- **AT** → Esta opción se utiliza para programar una acción a cierta hora.

```
MSG * ÉsteEsMiMensaje
AT 20:00 MSG * SON LAS OCHO DE LA TARDE
```

Con esto, a las 20:00 h., aparecerá un mensaje que dirá que son las ocho de la tarde. También se puede programar un apagado o cualquier otra acción. Si la hora que ya pasó, se tomará como la del día siguiente.

Resumen de comandos para ficheros batch

Comando	Descripción
@	Evita que una línea aparezca por pantalla.
CALL	Llama a otro fichero batch.
ECHO	Visualiza en pantalla una secuencia de caracteres.
FOR	Repite un número determinado de veces un mismo proceso.
GOTO	Salta y ejecuta una nueva línea indicada por una etiqueta.
IF	Se utiliza para saltos condicionales.
PAUSE	Detiene momentáneamente la ejecución de un fichero.
REM	Introduce un comentario.
SHIFT	Modifica los parámetros de un fichero batch.
:ETIQ	Identifica una posición de salto.
%NUM	Introduce parámetros en el fichero.