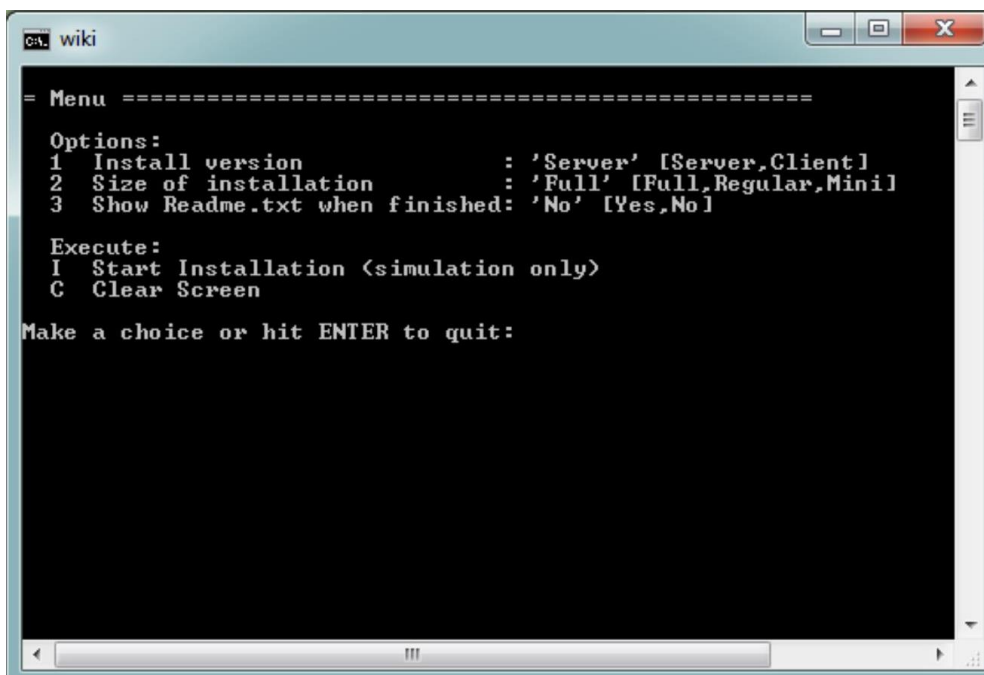


# Sistemas Informáticos

---

## Ficheros por lotes (archivos .bat)



```
= Menu =====  
  
Options:  
1 Install version           : 'Server' [Server,Client]  
2 Size of installation      : 'Full' [Full,Regular,Mini]  
3 Show Readme.txt when finished: 'No' [Yes,No]  
  
Execute:  
I Start Installation <simulation only>  
C Clear Screen  
  
Make a choice or hit ENTER to quit:
```

## Índice

1.	Introducción .....	2
2.	Concepto de fichero por lotes .....	2
3.	La orden ECHO .....	3
4.	La orden REM .....	5
5.	Gestión de parámetros .....	5
6.	La orden PAUSE .....	6
7.	La orden GOTO .....	7
8.	La orden IF .....	8
9.	La orden CHOICE .....	10
10.	Las variables del entorno .....	12
11.	Otras órdenes .....	12
12.	Soluciones de los ejercicios de autocomprobación .....	14
13.	Ejemplos resueltos de ficheros por lotes .....	15

# 1. Introducción

En muchas ocasiones los usuarios de MS-DOS repiten una determinada secuencia de órdenes día tras día, suponiendo una tarea bastante molesta e incómoda. Los *ficheros por lotes* pretenden automatizar una tarea rutinaria ejecutando una serie de órdenes definidas con antelación.

En este capítulo aprenderemos aquellas herramientas necesarias para *crear un fichero por lotes*. En el capítulo siguiente, se exponen y estudian útiles ejemplos que abarcan la totalidad del temario estudiado.

## 2. Concepto de fichero por lotes

*Un fichero por lotes es un fichero ASCII  
que contiene una serie de órdenes*

Como ejemplo, vamos a crear un fichero por lotes llamado TIEMPO.BAT. Para su confección podemos utilizar la orden **Copy** o el programa **Edit**.

```
C:\BATH>copy con tiempo.bat
date
time
^Z
1 archivo(s) copiado(s)
```

Si ahora ejecutamos el fichero recién creado, vemos como entran en funcionamiento las órdenes Date y Time.

```
C:\BATH>tiempo
```

```
C:\BATH>date
La fecha actual es Mar 15/08/1995
Escriba la nueva fecha (dd-mm-aa):
```

```
C:\BATH>time
La hora actual es 0:53:54,12
Escriba la nueva hora:
```

```
C:\BATH>
```

Todos los ficheros por lotes poseen unas características comunes:

- **Extensión.** Todos deben llevar obligatoriamente la extensión BAT.
- **Contenido.** Son ficheros de texto ASCII y, por consiguiente, pueden ser creados por Copy o Edit. Cada línea del fichero debe poseer una orden.
- **Ejecución.** Para hacerlo funcionar simplemente debemos teclear su nombre a continuación del símbolo del sistema. El fichero por lotes tomará entonces el control del ordenador.
- **Interrupción.** Podemos detener el procesamiento del fichero por lotes en cualquier momento presionando Ctrl+Pausa.

Todas las órdenes admitidas después del símbolo del sistema pueden introducirse también en un fichero por lotes. Además, existen una serie de órdenes diseñadas específicamente para estos ficheros:

<u>Orden</u>	<u>Breve descripción</u>
Echo	Controla el eco de las órdenes y visualiza mensajes.
Rem	Introduce comentarios.
Pause	Detiene temporalmente el desarrollo de un programa.
Goto	Desvía incondicionalmente el desarrollo de un programa.
If	Desvía condicionalmente el desarrollo de un programa.
Choice	Permite elegir entre unas opciones establecidas.
For	Repite una misma orden en un conjunto de ficheros.
Call	Llama a un fichero por lotes desde otro.
Shift	Desplaza el valor de los parámetros.

La siguiente tabla muestra aquellos símbolos empleados exclusivamente en los ficheros por lotes:

<u>Símbolo</u>	<u>Significado</u>
:etiqueta	Nombre de una etiqueta.
%número	Parámetro del fichero por lotes.
%variable%	Variable del entorno.
%%variable	Variable de la orden For.

### 3. La orden 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.

*El eco de una orden es el propio nombre de aquella escrito en la pantalla.* Al poner en marcha el fichero anterior, observará como aparece el nombre de cada orden en pantalla antes de ejecutarse. Al fichero del ejemplo podemos añadirle la línea ECHO OFF para desactivar el eco en lo sucesivo.

Para eliminar el eco de la propia orden ECHO OFF se antepone el *símbolo arroba* (@) al nombre de la orden. Veámoslo más claro en los ejemplos siguientes.

### *Programa:*

```
echo off  
date  
time
```

### *Ejecución:*

```
C:\BATH>tiempo
```

```
C:\BATH>echo off  
La fecha actual es Mar 15/08/1995  
Escriba la nueva fecha (dd-mm-aa):  
La hora actual es 0:53:54,12  
Escriba la nueva hora:
```

### *Programa:*

```
@echo off  
date  
time
```

### *Ejecución:*

```
C:\BATH>tiempo  
La fecha actual es Mar 15/08/1995  
Escriba la nueva fecha (dd-mm-aa):  
La hora actual es 0:53:54,12  
Escriba la nueva hora:
```

La orden Echo también permite mostrar mensajes al usuario del fichero. En el siguiente ejemplo se ha introducido el mensaje "Este es el fichero TIEMPO.BAT":

### *Programa:*

```
@echo off  
echo Este es el fichero TIEMPO.BAT  
date  
time
```

### *Ejecución:*

```
C:\BATH>tiempo  
Este es el fichero TIEMPO.BAT  
La fecha actual es Mar 15/08/1995  
Escriba la nueva fecha (dd-mm-aa):  
La hora actual es 1:10:22,72  
Escriba la nueva hora:
```

## 4. La orden REM

*Permite introducir comentarios internos en el código de un fichero por lotes. Las líneas precedidas por la palabra **Rem** serán ignoradas durante el funcionamiento del fichero por lotes. Se suele utilizar esta orden para introducir aclaraciones en el código del fichero.*

*Objetivo: facilitar su lectura y corrección de errores.*

```
@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
```

### Ejercicios de autocomprobación (ver soluciones al final)

1. Indica la diferencia entre las siguientes órdenes:

- ECHO Ficheros por lotes
- REM Ficheros por lotes

## 5. Gestión de parámetros

*Los parámetros son informaciones adicionales colocadas detrás del nombre de una orden. Si la mayoría de las órdenes de ms-dos admiten parámetros, también será posible gestionar parámetros en los ficheros por lotes.*

Vamos a confeccionar un fichero por lotes que borre dos ficheros introducidos como parámetros.

*Programa:*

```
@echo off
rem Programa: BORRA2.BAT
del %1
del %2
```

*Ejecución:*

```
A:\>borra2 juan.txt maria.txt
```

En la línea de órdenes, cada parámetro debe estar separado con un espacio en blanco del anterior. De la forma anteriormente explicada podemos gestionar hasta nueve de ellos (del %1 al %9).

Para referirnos a un parámetro introducido en la línea de órdenes del programa, debemos escribir el signo de porcentaje (%) seguido del número de parámetro.

El siguiente ejemplo copia los ficheros introducidos como parámetros al disquete de la unidad B:

### *Programa:*

```
@echo off
rem Programa: COPIAB.BAT
echo Se están copiando los ficheros %1, %2 y %3 a la unidad B:
copy %1 b:\
copy %2 b:\
copy %3 b:\
```

### *Ejecución:*

```
A:\>copiab juan.bak alberto.bmp marta.dbf
```

## **6. La orden PAUSE**

*Detiene temporalmente el desarrollo de un programa.* Cuando el dos encuentra una orden Pause en un fichero por lotes visualiza un mensaje en pantalla y espera una tecla para proseguir. También podemos presionar Ctrl+Pausa y así, interrumpir el desarrollo del fichero.

Presione cualquier tecla para continuar . . .

Para mejorar la presentación, podemos insertar un mensaje indicando el motivo de la detención del programa gracias a la orden Echo:

```
echo Inserte un disquete en la unidad B:
pause
```

Si no deseamos ver el mensaje de Pause, podemos redireccionarlo al dispositivo ficticio NUL.

```
echo Inserte un disquete en B: y pulse Enter
pause >nul
```

### **Ejercicios de autocomprobación (ver soluciones al final)**

2. ¿Qué hace la orden PAUSE >NUL ?

3. Escribe un fichero de procesamiento por lotes que pida un disco en la unidad A: antes de mostrar su directorio raíz.

## 7. La orden GOTO

Normalmente un fichero por lotes se desarrolla secuencialmente, desde la primera línea hasta la última. Sin embargo, *la orden Goto permite desviar la ejecución del programa hasta una etiqueta especificada como parámetro.*

GOTO [:]etiqueta

Las etiquetas deben ir precedidas de dos puntos (:) para diferenciarse de las órdenes. Admiten hasta ocho caracteres significativos. Esto último significa que la etiqueta `BALANCE\_DE\_AGOSTO' es idéntica a `BALANCE\_DE\_ENERO'.

El siguiente ejemplo muestra cómo se usan las etiquetas:

### *Programa:*

```
@echo off
ver
goto Final
vol
:Final
```

### *Ejecución:*

Versión MS-DOS 6.22

En este ejemplo, la orden Vol nunca se ejecutará: al llegar la orden GOTO FINAL, ms-dos salta hasta la etiqueta :Final y termina el fichero porque no hay más líneas.

Vamos a crear un programa para copiar varios disquetes desde la unidad A: al directorio actual.

```
@echo off
rem Programa: DEMO.BAT
echo *****
echo ** Este programa copia todos los ficheros **
echo ** de la unidad A: al directorio actual    **
echo *****
pause

:Proceso
copy a:\ .
echo Introduzca el siguiente disquete en A: y pulse Enter
pause >nul
goto Proceso
```

En primer lugar el programa muestra en pantalla su cometido y así, el usuario decide entre continuar (Enter) o abandonar (Ctrl+Pausa). Si pulsa Enter se copian todos los ficheros del directorio raíz de A: en el directorio actual.

Luego, se ofrece la posibilidad de introducir otro disquete. Si pulsamos Enter se ejecuta la orden GOTO PROCESO, desviando la ejecución del programa hasta la etiqueta :Proceso. En cambio, si pulsamos Ctrl+Pausa el programa finaliza, apareciendo el símbolo del sistema.



## Ejercicios de autocomprobación (ver soluciones al final)

4. ¿Qué hace el siguiente fichero por lotes?

```
@echo off
dir c:\
goto etiqueta2
:etiqueta1
    ver
:etiqueta2
```

## 8. La orden IF

*Desvía condicionalmente el proceso de ejecución de un fichero por lotes. Admite 6 sintaxis diferentes:*

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 EXIST

Un programador debe considerar todas y cada una de las situaciones posibles al ejecutarse su programa y, por tanto, debe evitar la aparición de mensajes de error inesperados en la pantalla.

Suponga el siguiente fichero por lotes:

```
@echo off
del %1
```

Si ahora lo hacemos funcionar introduciendo como parámetro un fichero inexistente, el ms-dos tomará el control y dará su correspondiente mensaje de error. Este fichero resulta más correcto si contiene las siguientes líneas:

```
@echo off
if not exist %1 echo ;Es imposible borrar un fichero que no existe!
if exist %1 del %1
De esta forma, la orden Del sólo funciona si el fichero existe.
```

### IF cadena1==cadena2

Encuentra su utilidad al trabajar con parámetros en un fichero por lotes.

En el fichero BORRA2.BAT hemos supuesto que el usuario siempre introduce dos parámetros. Sin embargo, se producirá un error si el usuario sólo introduce uno de ellos: la orden DEL %2 quedará transformada en DEL a secas y como Del no funciona sin parámetros, ms-dos mostrará una advertencia. Este problema se soluciona con lo siguiente:

```
@echo off
rem Programa: BORRA2B.BAT
if "%1"==" " echo Debe introducir uno o dos ficheros como parámetros.
if not "%1"==" " del %1
if not "%2"==" " del %2
```

La primera línea If avisa si no se ha introducido ningún parámetro. La segunda línea únicamente borrará el primer parámetro si se ha introducido. La tercera línea hace lo mismo pero con el segundo parámetro.

## 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 a:\ b:\
if errorlevel 1 goto Error
if errorlevel 0 echo ¡La copia fue correcta!
goto Final

:Error
echo Se produjo un error durante la copia

:Final
```

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.

## 9. La orden CHOICE

*Permite escoger una opción entre varias y, dependiendo de la opción elegida, devuelve un código de salida.*

Su sintaxis es:

**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.

El programa DEMO.BAT podemos mejorarlo sensiblemente si cambiamos las órdenes Pause por órdenes Choice.

```
@echo off
rem Programa: DEMOB.BAT
echo *****
echo ** Este programa copia todos los ficheros **
echo ** de la unidad A: al directorio actual **
echo *****
choice ¿Desea continuar?
if errorlevel 2 goto Final

:Proceso
copy a:\ .
choice Para continuar con otro disquete pulse C y para finalizar, F /C:FC
if errorlevel 2 goto proceso

:Final
```

En la primera orden Choice se toman las opciones por defecto S y N. `S' corresponde a un código de salida 1 y `N' a un código 2. En la segunda orden Choice se toman las opciones F y C. `F' corresponde a un código 1 y `C' a un código 2.

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.
```

Observe la orden Choice: el modificador /C indica las opciones admitidas. Si se pulsa la `A' se generará un código de salida 1 y así sucesivamente hasta la `S' que corresponde a un código 5. Gracias al modificador /N Choice no muestra las teclas admitidas detrás del mensaje. El modificador /T toma como opción por defecto la `S' si pasan 15 segundos sin pulsar ninguna tecla.

Observe, asimismo, cómo se ha comenzado en las líneas If por el errorlevel más alto: así se evitan conflictos. El código de salida 0 se obtiene si el usuario responde con Ctrl+Pausa al mensaje de Choice.

## Ejercicios de autocomprobación (ver soluciones al final)

5. Escribe un fichero de procesamiento por lotes para borrar el fichero introducido como parámetro. El programa debe comprobar previamente si se ha introducido algún parámetro y si el fichero existe.
6. ¿Para qué devuelven un código de salida las órdenes externas?
7. Escribe un fichero por lotes que nos pregunte si deseamos ver el directorio de A: o el de B:

## 10. Las variables del entorno

Se puede recuperar el valor de una determinada variable del entorno introduciendo ésta entre signos de porcentajes (%NombreVariable%).

Ejemplo:

```
Echo El valor de la variable PATH es %PATH%  
Echo El valor de la variable PROMPT es %PROMPT%
```

El siguiente ejemplo da el valor C:\DOS a la variable TEMP si no ha sido definida:

```
if "%TEMP%"==" " set TEMP=C:\DOS
```

## 11. Otras órdenes

Existen otras tres órdenes más diseñadas para su uso en ficheros por lotes. Como estas órdenes tienen un menor uso, se exponen resumidamente a continuación.

La orden FOR

Su sintaxis es:

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.

## La orden SHIFT

Se traduce al español por desplazamiento. *Mueve el valor de cada parámetro a la variable anterior.* Por ejemplo, si existen 3 parámetros (%1, %2 y %3) y se utiliza la orden Shift, el valor de %1 lo tomará %0, el valor de %2 lo tomará %1 y el valor de %3 lo tomará %2.

### *Programa:*

```
@echo off
rem Programa: DEMO2.BAT
echo El parámetro 1 es %1
shift
echo El parámetro 2 es %1
shift
echo El parámetro 3 es %1
```

### *Ejecución:*

```
C:\BATH>demo2 juan maria alberto
El parámetro 1 es juan
El parámetro 2 es maria
El parámetro 3 es alberto
C:\BATH>
```

## La orden CALL

Se utiliza para llamar a un fichero por lotes desde el interior de otro. Su sintaxis es:

CALL fichero [ParámetrosDelFichero]

En el siguiente ejemplo, el fichero PRG1.BAT llama a PRG2.BAT. Cuando la ejecución de PRG2.BAT termina, continua PRG1.BAT en la siguiente línea a Call.

### *Programa PRG1.BAT:*

```
@echo off
echo línea 1
call prg2.bat
echo línea 4
echo línea 5
```

### *Programa PRG2.BAT:*

```
@echo off
echo línea 2
echo línea 3
```

### *Ejecución:*

```
C:\BATH>prg1
línea 1
línea 2
línea 3
línea 4
línea 5
C:\BATH>
```

## 12. Soluciones de los ejercicios de autocomprobación

1. La línea ECHO muestra el mensaje "Ficheros por lotes" en pantalla y la línea REM es ignorada al funcionar el programa.
2. Espera la pulsación de una tecla. Como la salida de Pause está redireccionada al dispositivo nulo, no se mostrará ningún mensaje en la pantalla.
- 3.

```
@echo off
Echo Introduzca un disquete en la unidad A:
Pause
dir a:\
```

4. Muestra el directorio raíz de la unidad C: y luego la versión del sistema operativo. Como el nombre de las etiquetas sólo admite ocho caracteres significativos la orden GOTO ETIQUETA2 llevará a la primera etiqueta empezada por ETIQUETA, en este caso ETIQUETA1.
- 5.

```
@echo off
if "%1"==" " goto Error1
if not exist %1 goto Error2
del %1
echo El fichero %1 se ha borrado
goto Final
:Error1
    echo Debe especificar un fichero
    goto Final
:Error2
    echo El fichero %1 no existe
:Final
    echo.
```

6. Las órdenes externas devuelven un código de salida para ser utilizado en los ficheros por lotes. Posteriormente se podrá evaluar este código gracias a la orden If errorlevel.
- 7.

```
@echo off
echo A. Ver el directorio de A:
echo B. Ver el directorio de B:
choice Elija opción /c:ab
if errorlevel 1 if not errorlevel 2 dir a:\
if errorlevel 2 if not errorlevel 3 dir b:\
```

## 13. Ejemplos resueltos de ficheros por lotes

### Ejemplo 1: BUSCAR.BAT

Encontramos una aplicación de la orden Dir en la búsqueda de ficheros por el disco duro. Ocurre a menudo que creamos un fichero pero luego no lo encontramos en el directorio esperado.

Generalmente, el fichero se encontrará almacenado en otro directorio diferente. Podemos utilizar el siguiente programa para *buscar un fichero por todo el disco duro*.

#### *Programa:*

```
@echo off
rem Programa: BUSCAR.BAT
rem Cometido: Buscar el fichero o grupo de ficheros
rem especificado como parámetro en la unidad actual.

echo.
dir \%1 /b /s /p
echo.
```

#### *Ejecución:*

```
C:\BATH>buscar perdido.*
```

```
C:\PERDIDO.TXT
C:\PRUEBA\PERDIDO.DBF
C:\TRABAJOS\PERDIDO.TXT
```

```
C:\BATH>
```

Lo verdaderamente importante en este fichero es la orden Dir. El utilizar el símbolo de directorio raíz `\' antes del parámetro es para comenzar la búsqueda desde el directorio raíz a todos los subdirectorios. De esta forma no hará falta cambiar al raíz para hacer funcionar al programa.

El parámetro **%1** se sustituirá automáticamente por el fichero o grupo de ficheros a buscar. El parámetro **/b** presenta el resultado de la búsqueda en formato sencillo. El parámetro **/s** permite buscar el fichero en todos los subdirectorios y, finalmente, el parámetro **/p** muestra por pantallas el resultado de la búsqueda.

En el ejemplo se han buscado todos los ficheros con nombre PERDIDO y se han encontrado tres.



## Ejemplo 2: RELOJ.BAT

Permite mostrar la fecha y la hora del sistema. Este programa se apoya en las órdenes Date y Time de ms-dos y filtra la entrada y la salida de las órdenes.

Supón que sólo quieres ver la hora. Si ejecutas la orden Time, deberás pulsar Enter para no cambiar la hora actual.

```
La hora actual es 14:10:51,92
Escriba la nueva hora:↵
```

Este problema se soluciona si redirigimos la entrada de la orden desde un fichero, llamado RELOJ.TXT que contiene únicamente 2 bytes: un retorno de carro (Enter) y un código de fin de fichero (Ctrl+Z).

Como la salida de la orden contiene dos líneas y sólo nos interesa la primera, podemos filtrar la salida con la orden Find.

Para funcionar el fichero RELOJ.BAT, se necesita crear el fichero RELOJ.TXT de la siguiente forma: escribimos COPY CON RELOJ.TXT, pulsamos Enter dos veces, pulsamos Ctrl+Z y pulsamos finalmente Enter.

```
C:\BATH>copy con reloj.txt↵
↵
^Z↵
1 archivo(s) copiado(s)

C:\BATH>
```

A continuación tienes el código del programa y un ejemplo de su ejecución:

### *Programa:*

```
@echo off
rem Programa: RELOJ.BAT
rem Cometido: Mostrar la fecha y la hora actual en pantalla

if not exist c:\bath\reloj.txt goto Error
date <c:\bath\reloj.txt |find "actual"
time <c:\bath\reloj.txt |find "actual"
goto Final:

:Error
echo ;Falta el fichero RELOJ.TXT!

:Final
echo.
```

### *Ejecución:*

```
C:\BATH>reloj↵
La fecha actual es Mar 15/10/2021
La hora actual es 14:15:09,58

C:\BATH>
```

En primer lugar se comprueba la existencia del fichero RELOJ.TXT en el directorio C:\BATH. Si el directorio en el que reside es otro, debemos ajustar las órdenes convenientemente. Si el fichero RELOJ.TXT no existe, el programa no funcionará correctamente y, con el fin de evitarlo, se muestra un aviso y termina.

Si el fichero existe, se ejecutan las órdenes Date y Time. El operador `<' hace que la entrada de la orden sea el fichero RELOJ.TXT y el filtro Find hace aparecer la línea que contiene la palabra *actual*.

## Ejemplo 3: PROGRAMA.BAT

En muchas ocasiones, cuando un usuario recibe una aplicación desconocida, se ve obligado a buscar el fichero ejecutable que haga funcionar la aplicación. El siguiente fichero por lotes *busca todos los ficheros ejecutables (aquellos con extensión COM, EXE o BAT) y los visualiza en pantalla*.

### *Programa:*

```
@echo off
rem Programa: PROGRAMA.BAT
rem Cometido: Visualizar ordenados y con pausa en cada
rem pantalla todos los ficheros ejecutables del
rem directorio actual.

set fichtemp=%temp%\temporal.txt

if exist *.com dir *.com /b >%fichtemp%
if exist *.exe dir *.exe /b >>%fichtemp%
if exist *.bat dir *.bat /b >>%fichtemp%

type %fichtemp% |sort |more
del %fichtemp%
set fichtemp=
echo.
```

### *Ejecución:*

```
C:\WP60>programa↵
```

```
CV.EXE
INSTALL.BAT
MCV.EXE
QFIGENES.EXE
VAPINUL.COM
VMP.COM
WPINFO.EXE
WP.COM
WP.EXE
```

```
C:\WP60>
```

Este fichero por lotes necesita crear un fichero temporal para ir añadiéndole todos los ficheros con extensiones COM, EXE o BAT. Este fichero temporal debe ser creado en el directorio destinado a este fin (variable TEMP). Por tanto se crea otra variable llamada FICHTEMP que contiene el nombre del fichero (TEMPORAL.TXT) y su trayectoria.

Si no se utilizase el directorio definido en la variable TEMP para crear el fichero TEMPORAL, podría darse el caso de intentar escribir en un disquete protegido contra escritura y no poder completarse la función del programa.

La primera línea If crea el fichero temporal con el nombre de los ficheros con extensión COM. La segunda línea If añade al fichero temporal, aquellos ficheros con extensión EXE y la tercera línea If hace lo mismo con los ficheros BAT.

La orden Type muestra todos los programas (ahora grabados en el fichero TEMPORAL.TXT) ordenados y por pantallas.

Finalmente, la orden Del borra el fichero temporal creado y la orden Set elimina la variable del entorno creada.

## Ejemplo 4: BORRAR.BAT

Este fichero por lotes, desplaza un fichero o grupo de ficheros a un directorio llamado C:\BASURA. De esta forma, si luego necesitamos recuperar un fichero borrado, sólo hará falta buscarlo en el directorio C:\BASURA. Sin embargo, esto tiene un *problema*: el directorio C:\BASURA crecerá y crecerá de tamaño llenando nuestro disco duro de ficheros inservibles. Por tanto, es necesario eliminar periódicamente el contenido de este directorio.

Si te parece interesante la idea consulta la orden **Undelete**. Undelete detecta la orden Del de ms-dos y mueve, igualmente, los ficheros a un directorio. Lo más importante de Undelete es la posibilidad de eliminar (realmente) los ficheros más antiguos almacenados de una forma automática.

### *Programa:*

```
@echo off

rem Programa: BORRAR.BAT
rem Cometido: Mueve el fichero o grupo de ficheros
rem especificados a un directorio llamado
rem C:\BASURA. Si no existe, lo crea.

if "%1"==" " goto Error
if not exist %1 goto Error

xcopy %1 c:\basura\ >nul
if errorlevel 1 goto Error
del %1

echo A continuación se recuerda el estado del directorio C:\BASURA
DIR C:\BASURA |FIND "archivo(s) "

goto Final

:Error
echo ;No se pudo borrar %1!

:Final
echo.
```

## *Ejecución:*

```
C:\PRUEBA>borrar *.ini↵
```

```
A continuación se recuerda el estado del directorio C:\BASURA
5 archivo(s) 27.238 bytes
```

```
C:\PRUEBA>
```

La primera orden If detecta si no se ha introducido ningún parámetro, produciendo un aviso en este caso. La segunda orden If comprueba que el fichero para borrar exista, si no es así dará un mensaje de error.

La orden Xcopy mueve el fichero o ficheros especificados como parámetros al directorio C:\BASURA. La barra inclinada invertida detrás de BASURA se emplea para informar a Xcopy que BASURA es un directorio y no un fichero. Si el directorio no existe, Xcopy sabrá que es un directorio y lo creará automáticamente. La salida de la orden se ha redireccionado al dispositivo ficticio NUL. De esta forma, no aparecerá ningún mensaje producido por la orden en la pantalla.

La siguiente línea If comprueba el código de salida. Si se ha producido un error en la copia de los ficheros, los ficheros no se borrarán. Esto es una medida de seguridad para evitar la desaparición definitiva de los ficheros.

La orden Del borra los ficheros del directorio actual. Ya hemos mencionado el problema de este fichero por lotes. Como solución, se informa al usuario del número de ficheros y los bytes ocupados por todos ellos en el directorio C:\BASURA. De esta forma el usuario sabrá si debe borrar con la orden Del los ficheros del directorio BASURA.

## Ejemplo 5: COPIADIR.BAT

Este programa se basa en el método para copiar un directorio que no cabe en un único pendrive, en varios.

### *Programa:*

```
@echo off
echo Este programa copia todos los ficheros del
echo directorio actual a la unidad E:
echo (si no cabe en un sólo pendrive, se pedirán más)

if not exist *.* goto Error
attrib +a *.*

:Proceso
echo.
echo ** Inserte un nuevo pendrive en E: Pulse Enter para proseguir
pause >nul

xcopy *.* E:\ /m
if errorlevel 5 goto Error
if errorlevel 4 goto Proceso
if errorlevel 1 goto Error
if errorlevel 0 goto FinCopia

echo.
```

```
echo ;No se pudo realizar la copia!
goto final

:FinCopia
echo.
echo ;El proceso de copia se completó con éxito!

:Final
echo.
```

La primera línea If detecta si el directorio actual está vacío produciendo un error. La orden Attrib enciende el bit de archivar a todos los ficheros del directorio actual.

Seguidamente, COPIADIR.BAT pide un primer pendrive en la unidad E:, luego comienza la copia de todos los ficheros a la unidad E:. El parámetro /m desactiva el bit de archivar a los ficheros copiados.

Las siguientes líneas If comprueban el resultado de la orden.

Las posibilidades son las siguientes: **a)** No hay suficiente espacio en el pendrive para copiar todos los ficheros. En este caso, se pide otro y se reanuda la copia donde se quedó; **b)** Se presionó Ctrl+Pausa u ocurrió un error de escritura en el pendrive. Se produce un mensaje y finaliza; **c)** La copia fue correcta. Esto significa que todos los ficheros han podido copiarse en uno o más pendrives y por tanto, se ha cumplido el objetivo del programa.