I file batch

La shell è in grado di mandare in esecuzione comandi contenuti in file caratterizzati dalle estensioni COM, EXE e BAT. I file dei tipi COM ed EXE contengono codice in linguaggio macchina e, una volta caricati in memoria, possono essere immediatamente eseguiti.
I file di tipo **BAT**, invece, sono file di testo che contengono sequenze di comandi di shell, perciò l'interprete dei comandi li tratta come se rispondesse a input da tastiera.
I file di tipo BAT, spesso chiamati **file batch** (o anche **script**), costituiscono la maniera più pratica di automatizzare sequenze di comandi che vengono adoperate spesso oppure che sono lunghe da eseguire.
Nel primo caso si ottiene un risparmio di battute, e quindi di errori di battitura, mentre nel secondo l'uso di uno script permette di non attendere il termine di ciascun comando per impartire il successivo, ottimizzando il tempo totale di esecuzione della sequenza di comandi.
Inoltre il lavoro può essere eseguito in modo non interattivo (modalità *batch*) senza la necessità della presenza dell'utente.
Si voglia per esempio costruire un batch file di nome *Copia.bat* che svolga le seguenti funzioni: copiare alcuni file dal disco fisso all'unità E: (per esempio una memoria USB) e cancellare in seguito i file copiati per motivi di segretezza.

```
@ECHO OFF
CLS
ECHO Introdurre il disco nell'unità E:
PAUSE
COPY C:\Docum\Testo.doc E:\File1
COPY C:\Tabelle\Vendite.tab E:\File2
COPY C:\Lettere\Lettera.txt E:\File3
COPY C:\Fatture\Luglio E:\File4
DEL C:\Docum\Testo.doc
DEL C:\Tabelle\Vendite.tab
DEL C:\Lettere\Lettera.txt
DEL C:\Fatture\Luglio
```

Il comando può essere eseguito dalla linea comandi scrivendo:

```
Copia.bat
```

o, più semplicemente:

```
Copia
```

Con questo batch file, è possibile chiedere al sistema l'esecuzione dei comandi in esso contenuti nello stesso ordine con il quale sono stati scritti nel testo del file.
Si noti l'istruzione **@ECHO OFF** che serve a disattivare la ripetizione sullo schermo del comando in esecuzione; il simbolo **@** disattiva anche la visualizzazione del comando *echo off* stesso.
Il comando **CLS** cancella lo schermo e il comando **ECHO** seguente visualizza un messaggio informativo.
Il comando **PAUSE** sospende l'esecuzione che può essere riattivata premendo un tasto qualsiasi.
I comandi successivi eseguono la copia dei file dal disco fisso all'unità esterna e la successiva cancellazione dei file copiati.

Un secondo esempio mostra il testo di un batch file che serve a ripristinare i file su disco fisso:

```
COPY E:\File1 C:\Docum\Testo.doc
COPY E:\File2 C:\Tabelle\Vendite.tab
COPY E:\File3 C:\Lettere\Lettera.txt
COPY E:\File4 C:\Fatture\Luglio
```

Il file batch *Copia.bat* può essere migliorato introducendo le seguenti righe dopo la sequenza dei comandi COPY.

```
DIR E:\
PAUSE
```

Questo fa sì che il contenuto dell'unità E: venga visualizzato e l'esecuzione sospesa prima di cancellare i file sul disco fisso. Se si constata che uno o più file non sono stati copiati si può interrompere l'esecuzione e controllare che cosa sia andato male.
All'interno di uno script è possibile in qualche misura il controllo del flusso di esecuzione, vale a dire che si può alterare la sequenza con cui i comandi vengono eseguiti, a seguito del verificarsi o meno di certe condizioni. Per esempio si può controllare che un comando sia stato eseguito correttamente prima di passare al successivo, oppure si può verificare che un file esista prima di eseguire operazioni che lo riguardano.
Vediamo qualche esempio:

```
IF NOT EXIST Lettera.doc GOTO ERRORE
COPY Lettera.doc C:\Archivio
DEL Lettera.doc
GOTO FINE
:ERRORE
ECHO Errore: il file non esiste
:FINE
```

Quando la shell esegue questo file inizia con la verifica, richiesta con la prima riga, dell'esistenza di un file e prende una decisione di conseguenza: se il file non esiste (**IF NOT EXIST**) esegue il comando **GOTO ERRORE**, in modo che il prossimo comando da interpretare sia quello che segue l'etichetta *ERRORE*, indicata entro il file con *:ERRORE* (con i due punti iniziali). Questo comando è **ECHO *frase***, che causa la visualizzazione di *frase* sul video.
Se invece il file esiste, l'interpretazione prosegue con la riga successiva all'IF, che ricopia il file *Lettera.doc* nella directory *C:\Archivio*, quindi con **DEL Lettera.doc**, che cancella la versione originale del file; infine salta all'etichetta *FINE*, anch'essa indicata entro il file con il simbolo **:** (due punti) seguito dal nome dell'etichetta. Oltre a IF NOT EXIST esiste il controllo complementare, cioè **IF EXIST**.
Durante l'esecuzione dello script, tutti i comandi in esso contenuti vengono visualizzati: se si desidera evitare questo, si può introdurre nella prima riga il comando **@ECHO OFF** che appunto sopprime l'eco di ogni comando a video.
Desiderando ripristinarlo, si usa il comando **ECHO ON**.

Per alcuni comandi, è possibile controllare se la loro esecuzione è terminata in modo normale, oppure con errori, attraverso la struttura **IF ERRORLEVEL**. Questo è reso possibile dall'uso del codice di uscita del programma avente la seguente sintassi generale:

```
IF ERRORLEVEL n comando
```

dove *n* è un numero intero maggiore o uguale a zero.

Il nome **ERRORLEVEL** definisce una variabile predefinita della programmazione batch (**variabile di ambiente**).
Questa forma di controllo verifica se il codice di uscita è stato posto ad un numero pari a *n* o superiore, e in caso positivo la shell manda in esecuzione il comando specificato. Un valore 0 (zero) indica che il comando è terminato con successo, perciò in generale:

```
IF ERRORLEVEL 1 GOTO ERRORE
```

causa il salto all'etichetta *ERRORE* se il comando è terminato in modo anomalo; infatti qualsiasi codice di uscita diverso da zero viene rilevato da questo controllo.
Per determinare quale errore sia effettivamente accaduto, bisogna controllare tutti i codici in ordine decrescente, in modo che si abbia riscontro positivo solo al codice effettivamente trasmesso.

Se è noto che un certo comando può terminare con un codice compreso tra 0 e 5, occorre controllare i codici nell'ordine da 5 a 1: se il codice di uscita è 3, i controlli su 5 e 4 sono negativi, mentre è positivo quello su 3.

```
 ......
IF ERRORLEVEL 5 GOTO ERRORE5
IF ERRORLEVEL 4 GOTO ERRORE4
IF ERRORLEVEL 3 GOTO ERRORE3
IF ERRORLEVEL 2 GOTO ERRORE2
IF ERRORLEVEL 1 GOTO ERRORE1
:TUTTOOK
 ...
GOTO FINE
:ERRORE1
 ...
GOTO FINE
:ERRORE2
 ...
GOTO FINE
 ...
 ...
:ERRORE5
 ...
GOTO FINE
:FINE
```

Non tutti i comandi usano il codice di uscita, per esempio il comando DEL non lo adopera e se si cerca di cancellare un file inesistente, che costituisce evidentemente una situazione anomala, non si ha modo di verificarlo per mezzo di ERRORLEVEL.
Altri comandi usano più codici di uscita, per esempio FORMAT è in grado di comunicare diversi eventi anomali che possono verificarsi durante la sua esecuzione per mezzo di diversi codici di uscita: il loro uso consente di scrivere file batch più raffinati, in grado di prendere le decisioni giuste in ogni circostanza.

Gli script sono in grado di trattare **parametri di chiamata**, ovvero stringhe fornite in ingresso insieme al comando.
Per esempio, lo script seguente accetta in ingresso i nomi di due file e ne visualizza il contenuto, uno dopo l'altro e una pagina per volta.

```
@ECHO OFF
TYPE %1 | MORE
PAUSE
TYPE %2 | MORE
```

Dopo la visualizzazione del primo file, lo script si ferma e attende che l'utente prema un tasto per continuare (comando **PAUSE**).
Scrivendo un file *Scrivi.bat* contenente i comandi specificati, se si digita

```
Scrivi Primo.txt Secondo.txt
```

si ottiene l'output dei due file sul video.
Infatti %1 assume il valore corrispondente alla prima stringa che segue il nome del comando e %2 quello della seconda, perciò i due comandi entro lo script vengono espansi in:

```
TYPE Primo.txt | MORE
PAUSE
TYPE Secondo.txt | MORE
```

Il comando **SET** assegna un valore a una variabile (*SET nomevariabile = valore*). Per indicare il valore di una variabile si usa la notazione *%variabile%.*
Per controllare l'uguaglianza tra variabili, o tra variabili e stringhe, si usa l'operatore con il doppio segno == per il confronto:

```
IF %variab%==%variab2% comando
IF NOT %variab%==%variab2% comando
IF %variab%==stringa comando
IF NOT %variab%==stringa comando
```

Variabili delle quali si deve controllare il contenuto sono per esempio quelle passate allo script quando lo si esegue.

```
IF %1==%C% comando
IF NOT %2==PIPPO comando
```

Si osservi che viene usato solo un simbolo di percentuale per espandere il contenuto delle variabili passate allo script.
Se nel secondo caso lo script contenente il controllo viene lanciato senza almeno due stringhe, il controllo dà luogo ad un errore, perché %2 viene espanso in una stringa nulla, contenente zero caratteri, e quindi la riga assume l'aspetto seguente

```
IF NOT ==PIPPO comando
```

che contiene un errore di sintassi, perché non si può mettere il doppio segno == senza un argomento che lo preceda. È perciò opportuno usare la forma seguente:

```
IF NOT "%2"=="PIPPO" comando
```

In tal modo, se %2 è la stringa nulla, l'espansione genera la riga seguente:

```
IF NOT ""=="PIPPO" comando
```

che sintatticamente è corretto.

Usando questo tipo di controlli si possono creare script in grado di eseguire compiti diversi, in dipendenza da diversi valori di una stringa in ingresso.
Lo script che segue, che chiamiamo *Uti.bat*, richiede che vengano forniti due o tre parametri in ingresso, il primo dei quali determina il tipo di operazione da eseguire, i successivi sono nomi di file sui quali operare.

All'interno del testo sono state introdotte alcune righe di commento per spiegare le funzionalità del file batch. Le righe che iniziano con la parola chiave **REM** contengono i commenti.

```
@ECHO OFF
REM Gestione di file

REM Variabile per il controllo del tipo di errore
SET TIPO_ERR=0

IF "%1"=="c" GOTO COPIA
IF "%1"=="m" GOTO MUOVI
IF "%1"=="d" GOTO CANCELLA
IF "%1"=="" GOTO ISTRUZIONI

REM Istruzioni per copiare un file in un altro
:COPIA
IF "%3"=="" GOTO ISTRUZIONI
REM Se uso c devo dare anche 2 nomi di file

IF EXIST %3 GOTO ERRORE
SET TIPO_ERR=1
IF NOT EXIST %2 GOTO ERRORE
COPY %2 %3
GOTO FINE

REM Istruzioni per spostare un file in un altro
:MUOVI
IF "%3"=="" GOTO ISTRUZIONI
REM Se uso m devo dare anche 2 nomi di file

IF EXIST %3 GOTO ERRORE
SET TIPO_ERR=1
IF NOT EXIST %2 GOTO ERRORE
MOVE %2 %3
GOTO FINE

REM Istruzioni per cancellare un file
IF "%2"=="" GOTO ISTRUZIONI
REM Occorre un nome di file
:CANCELLA
SET TIPO_ERR=1
IF NOT EXIST %2 GOTO ERRORE
DEL %2
GOTO FINE
```

```
REM Se il comando viene eseguito con i parametri
REM sbagliati, vengono visualizzate le istruzioni per l'uso
:ISTRUZIONI
ECHO Istruzioni per l'uso del comando Uti:
ECHO Uti c File1 File2  copia File1 in File2 se File2 non esiste
ECHO Uti m File1 File2  muove File1 in File2 se File2 non esiste
ECHO Uti d File1  cancella File1
GOTO FINE
:ERRORE
IF %TIPO_ERR%==0 ECHO Attenzione, il File %3 esiste gia'
IF %TIPO_ERR%==1 ECHO Attenzione, il File %2 non esiste

:FINE
```

Come si può vedere dalle istruzioni per l'uso, lo script permette di copiare un file in un altro, cancellando o no l'originale, oppure di cancellare un file.

Vengono eseguiti alcuni controlli, di correttezza formale e di possibilità di esecuzione. Se la forma non è rispettata, cioè se i parametri non sono corretti, vengono visualizzate le istruzioni per l'uso, mentre se un'operazione non può essere eseguita compare un messaggio di errore.

Lo script riconosce due tipi di errore: l'esistenza del file di destinazione e la non esistenza del file da copiare o cancellare. La gestione è affidata alla variabile *TIPO_ERR*, che vale 0 all'inizio e durante il controllo su %3, e viene posta ad 1 subito prima del controllo su %2. In questo modo, se si ha effettivamente un errore si può controllare il valore di *TIPO_ERR* per visualizzare il messaggio corretto.

In sintesi, gli script permettono di eseguire sequenze di comandi con un certo grado di sicurezza in modo automatico, a condizione che si prevedano le circostanze anomale che possono verificarsi, soprattutto se lo script deve essere eseguito in modo *unattended*, cioè senza la sorveglianza continua di un operatore. La corretta gestione dei codici di uscita e delle condizioni di esistenza dei file può impedire di causare danni al sistema, o comunque di eseguire operazioni sbagliate, mentre l'uso delle variabili permette di scrivere script più versatili e potenti.

Il seguente esempio presenta un programma batch che può essere utilizzato per installare su disco fisso i programmi per la gestione della contabilità, prelevandoli dal CD che si trova nell'unità D:.
Il programma sarà registrato sullo stesso CD contenente il software con il nome *Installa.bat* e la sua esecuzione potrà essere attivata scrivendo da linea comando:

```
D:Installa
```

Il testo del file *Installa.bat* è il seguente:

```
@ECHO OFF
CLS
ECHO    INSTALLAZIONE SOFTWARE CONTABILITÀ
ECHO                CONTAB
ECHO    (VERS. 1.0)
ECHO ------------------------------------------------------
ECHO ------------------------------------------------------
```

```
ECHO Con questa installazione CONTAB
ECHO viene creato (o sostituito)
ECHO nella directory del disco fisso C:\CONTAB
ECHO se non vuoi proseguire premi CTRL + C
ECHO ------------------------------------------------
PAUSE
ECHO      prego attendere ....
C:
CD\
COPY D:con*.* C:\contab >NUL
COPY D:Inizio.bat C:\ >NUL
CD\
ECHO ------------------------------------------------
ECHO      Installazione terminata
ECHO ------------------------------------------------
ECHO ------------------------------------------------
ECHO PER INIZIARE, DIGITARE:  Inizio
ECHO ON
```

La ridirezione **>NUL** nel comando *Copy* serve ad eliminare la visualizzazione dei nomi dei file che vengono via via copiati dal CD.