

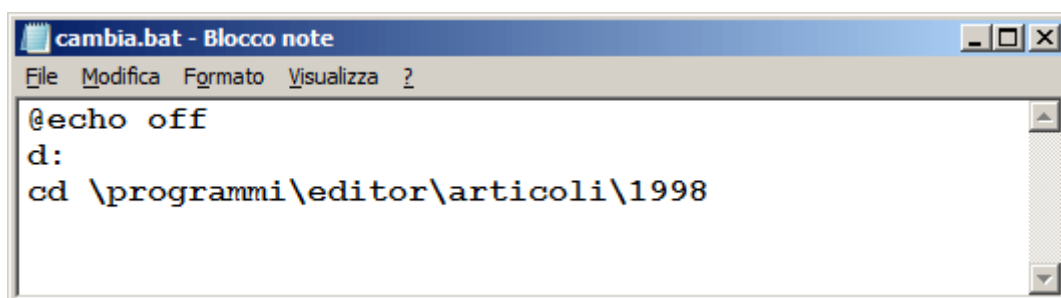
# MANUALE DOS - I FILES BATCH

---

## I FILE BATCH

---

Un **file batch** in dos è un file testuale contenente una sequenza di comandi DOS. L'obbiettivo dei files batch è l'automatizzazione di operazioni ripetitive. Benché i file batch DOS siano piuttosto limitati, è possibile scrivere semplici script per automatizzare alcune operazioni frequenti. Ad esempio se dovete cambiare spesso directory da c:\ a d:\programmi\editor\articoli\1998, la cosa più semplice è quella di scrivere un file batch così:



```
cambia.bat - Blocco note
File Modifica Formato Visualizza ?
@echo off
d:
cd \programmi\editor\articoli\1998
```

la prima riga **@echo off** serve solo per evitare di mostrare i comandi eseguiti, mentre il resto sono gli usuali comandi che daresti tramite tastiera. Chiamando questo file **cambia.bat**, basterà ora digitare sul prompt del DOS il comando **cambia** per cambiare directory, semplice no ? Quello che forse non sapete, è che c'è un metodo ancora più veloce per fare la stessa cosa:

```
subst e: d:\programmi\editor\articoli\1998
```

in questo modo il drive "e:" diventa sinonimo di "d:\programmi\editor\articoli\1998", ed invece dell'interminabile CD, è sufficiente un E: .

Questo trucco può essere utilizzato anche in altre occasioni, ad esempio a volte capita di volere avere una directory con un nome molto lungo (o più sottodirectory) nel PATH, ma sfortunatamente il DOS non permette di avere più di 79 caratteri, perciò basta usare il comando subst sui percorsi più lunghi.

Vediamo ora alcune istruzioni di base che ci consentiranno di costruire piccoli programmi:

## LE VARIABILI E I PARAMETRI

---

Le variabili servono a registrare dei valori acquisiti durante l'esecuzione di un programma. L'inizializzazione di una variabile (assegnamento di un valore iniziale) avviene mediante il comando **SET**

```
@echo off
:: Inizializzazione
set NOME=Marco
:: Stampa del valore
ECHO Il mio nome e': %NOME%
ECHO Il mio nome e': %nome% (si noti che
le variabili non sono CASE SENSITIVE)
:: Eliminazione della variabile
SET NOME=
```

Tutte le variabili definite sono visibili digitando sul prompt del DOS il comando **SET** . Il nome delle variabili non è case sensitive ovvero scrivere una variabile in maiuscolo e la stessa cosa di scriverlo in minuscolo. Esistono due opzioni del comando **SET** : **/A** per assegnare ad una variabile un'espressione numerica

```
@ECHO OFF
SET /A X=10
SET /A Y=3
SET /A Z=X+Y
ECHO %X% + %Y% = %Z%
SET /A Z=X-Y
ECHO %X% - %Y% = %Z%
SET /A Z=X*Y
ECHO %X% x %Y% = %Z%
SET /A Z=X/Y
ECHO %X% DIV %Y% = %Z%
SET /A Z=%X%%Y%
ECHO %X% MOD %Y% = %Z%
SET /A X+=1
ECHO INCREMENTO X = %X%
SET /A A=8,3
SET /A B=3,6
SET /A C=A+B
ECHO -----
ECHO i comandi batch non gestiscono numeri
decimali
ECHO -----
ECHO %A% + %B% = %C%
```

Che produce il seguente output:

```
E:\Users\Administrator\Desktop>operatori
10 + 3 = 13
10 - 3 = 7
10 x 3 = 30
10 DIV 3 = 3
10 MOD 3 = 1
INCREMENTO X = 11
-----
i comandi batch non gestiscono numeri
decimali
-----
8 + 3 = 13
E:\Users\Administrator\Desktop>
```

Senza l'opzione **/A** i dati vengono trattati come stringhe. Vediamo il seguente esempio:

```
@ECHO OFF
SET X=10
SET Y=3
SET Z=%X%+%Y%
ECHO %X% + %Y% = %Z%
SET Z=X+Y
ECHO %X% + %Y% = %Z%
```

la cui esecuzione produce:

```
E:\Users\Administrator\Desktop>No_SlashA
10 + 3 = 10+3
10 + 3 = X+Y
E:\Users\Administrator\Desktop>
```

L'altra opzione è **/P** che consente l'input da parte dell'utente: verrà trattata nel prossimo paragrafo. Utilizzando **/?** si ottiene un help esaustivo del comando **SET**.

Come si citava all'inizio un batch è un file testuale nel quale sono inseriti dei comandi dos. E' possibile rendere tale sequenza di comandi parametrica ovvero accanto al nome del file (durante la sua digitazione sul prompt) posso inserire una serie di stringhe che rappresentano i parametri del comando che intendo eseguire. Ad esempio è possibile, mediante la parametrizzazione, costruire un batch **Somma** il quale digitando **Somma 1 23 12** esegue la somma degli argomenti

```
E:\Users\Administrator\Desktop>Somma 1 23
12
La somma 1 + 23 + 12 = 36
E:\Users\Administrator\Desktop>
```

I parametri passati al comando sono leggibili nello script tramite le variabili predefinite **%1 %2 ... %9** (quindi sembra che non sia possibile passare più di 9 argomenti ma in realtà il problema è superabile!). A queste si aggiungono: **%0** e **%\*** che contengono rispettivamente il nome del file batch e l'elenco completo dei parametri. Analizziamo l'esempio sottostante:

```
@ECHO OFF
:: Visualizzo i parametri
ECHO Nome file batch : %0
ECHO Tutti i parametri : %*
ECHO primo parametro : %1
ECHO secondo parametro : %2
ECHO terzo parametro : %3
ECHO quarto parametro : %4
ECHO quinto parametro : %5
ECHO sesto parametro : %6
ECHO settimo parametro : %7
ECHO ottavo parametro : %8
ECHO nono parametro : %9
ECHO decimo parametro : %10 (Si noti che
da errore)
```

la sua esecuzione produce:

```

E:\Users\Administrator\Desktop>Elenca A B
C D E F G H I L M
Nome file batch      : Elenca
Tutti i parametri   : A B C D E F G H I L M
primo parametro     : A
secondo parametro   : B
terzo parametro     : C
quarto parametro    : D
quinto parametro    : E
sesto parametro     : F
settimo parametro   : G
ottavo parametro    : H
nono parametro      : I
decimo parametro    : A0 (Si noti che da
errore)

```

Il superamento dei 9 parametri verrà analizzato in seguito.

## INPUT ED OUTPUT

Per quanto riguarda l'output abbiamo visto che **ECHO frase** visualizza a video la frase passata come argomento. Il comando **cls** consente la pulizia del video. Inserendo come prima riga **@ECHO OFF** disabilitiamo la visualizzazione dei comandi batch inseriti nel file.

L'input può avvenire in diversi modi:

A) Come accennato il comando **SET /P** consente di acquisire dei valori da tastiera. L'opzione **/P** è attiva solo se sono installate le "**COMMAND EXTENSION**" (sono presenti da win2000). Per verificare se tali estensioni sono attive basta digitare sul prompt questo comando: **ECHO %CMDEXTVERSION%**. Se viene visualizzato il numero **2** allora sono attive.

```

@ECHO OFF
set INPUT=
set /P INPUT=Digita qualcosa:
echo Hai digitato: %INPUT%

```

la sequenza di queste istruzioni determina questo output:

```

E:\Users\Administrator\Desktop>Digita
Digita qualcosa: ciao
Hai digitato: ciao
E:\Users\Administrator\Desktop>

```

B) Altra modalità è utilizzare il comando **choice** (questo comando è presente in tutte le versioni di Windows tranne in windows XP!)

Il comando dos **choice** presenta (nelle versioni *Windows 2003 Server/Windows 2008 Server/Windows Vista/Windows 7*) la seguente sintassi:

```
E:\Users\Administrator\Desktop>choice /?
```

```
CHOICE [/C scelte] [/N] [/CS] [/T timeout  
/D scelta] [/M testo]
```

Descrizione:

Questa utilità consente all'utente di selezionare un elemento da un elenco di scelte e restituisce l'indice della scelta selezionata.

Elenco parametri:

/C scelte Specifica l'elenco delle scelte da creare.

L'elenco

predefinito è "YN".

/N Nasconde l'elenco delle scelte nel prompt.

Il messaggio prima che il prompt venga visualizzato e le scelte sono ancora abilitate.

/CS Abilita le scelte con distinzione tra maiuscole e minuscole da selezionare.

Per impostazione predefinita, l'utilità non effettua alcuna distinzione tra maiuscole e minuscole.

/T timeout Il numero di secondi di sospensione

prima che venga effettuata una scelta predefinita. I valori accettabili sono compresi tra 0 e 9999.

Se è stato specificato 0, non ci sarà sospensione e verrà selezionata la scelta predefinita.

/D scelta Specifica la scelta predefinita dopo nnnn secondi.

Il carattere deve essere compreso nell'insieme di scelte specificate dall'opzione /C e deve specificare nnnn con /T.

/M testo Specifica il messaggio da visualizzare prima del prompt. Se non specificato, l'utilità

```

visualizzerà
soltanto un prompt.

/?          Visualizza questo
messaggio della Guida
...

```

Nella versione **windows 98** il comando dos **choice** (versione disponibile per il download su questo sito) presenta una differente sintassi:

```

C:\WINDOWS\Desktop>choice /?
Attende che l'utente scelga da un insieme
di scelte.

CHOICE [/C[:]scelte] [/N] [/S] [/T[:]c,nn]
[ testo ]

/C[:]scelte  Specifica i tasti ammessi. I
predefiniti sono SN.
/N          Non visualizza le scelte e ?
alla fine della stringa di prompt.
/S          Distingue tra Maiuscole e
minuscole per quanto riguarda
            i tasti di scelta.
/T[:]c,nn   Scelta predefinita di c dopo
nn secondi.
testo       Stringa del prompt da
visualizzare.

ERRORLEVEL è impostato sull'ordine dei
tasti che l'utente preme per le sue
scelte.

```

**ATTENZIONE!** Per svolgere le esercitazioni, qualora il vostro PC abbia installato Windows XP, è necessario scaricare il comando **choice** (preso da windows 98) dal seguente link [CHOICE\\_win98.zip](#)

Il comando **choice** non gestisce tutti i caratteri ma solo i simboli alfabetici (a-z, A-Z), numerici (0-9) oppure i codice ascii compresi tra 128 e 254.

B.1) Vediamo un primo esempio che consente la lettura di un singolo carattere tra **A**, **B** e **C**.

compatibile con choice di Win Vista, 7, 2008Server, 2003Server	compatibile con choice di Win 98 sotto Windows
<pre> @ECHO OFF CHOICE /N /C:ABC /M "Digita la scelta (A, B, o C):" echo %errorlevel% </pre>	<pre> @ECHO OFF CHOICE /N /C:ABC "Digita la scelta (A, C):" echo %errorlevel% </pre>

L'opzione /C definisce i caratteri ammessi dal comando **choice** e restituisce nella variabile d'ambiente **%ERRORLEVEL%** la posizione, rispetto all'elenco definito con /C, del carattere digitato. Con questo batch di esempio, digitando il carattere **C**, vedremo che la variabile **%ERRORLEVEL%** viene posta al valore 3

```

E:\Users\Administrator\Desktop>Digita
Digita la scelta (A, B, o C): C

```

3

```
E:\Users\Administrator\Desktop>
```

La variabile d'ambiente **ERRORLEVEL** può essere settata in linguaggio C tramite l'istruzione **exit**. Questo meccanismo consente al programmatore la restituzione di un valore numerico che rappresenta lo stato di uscita (ad esempio se l'esecuzione si è conclusa senza errori) di un programma. Generalmente a valori diversi da 0 si abbinano esecuzioni incomplete o che hanno generato un errore. Immaginiamo quindi di costruire in C++ il seguente programma **Conta\_Arg.cpp**.

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
    int i;
    cout << "Report dal C++   : Nr.
argomenti: " << argc-1 << endl;
    // for (i=1; i <argc ; printf("%2d)
%s\n",i-1,argv[i++]));
    exit(argc-1); // restituisco in
ErrorLevel il nr di argomenti
}
```

Compiliamo ed eseguiamolo. Qui sotto un esempio di esecuzione. Si noti che **Conta\_Arg** si limita a stampare il numero di argomenti passati sulla linea di comando

```
C:\arch_def_1\Scuola\SW-
Usato\Prove>Conta_Arg
Report in C++   : Nr. argomenti: 0

C:\arch_def_1\Scuola\SW-
Usato\Prove>Conta_Arg Ave Student
Report in C++   : Nr. argomenti: 2
```

Per verificare il collegamento tra il comando **EXIT** ed **ERRORLEVEL** basta provare il seguente file batch: **TESTEXIT.BAT**

```
@ECHO OFF
ECHO ---ESECUZIONE 1 (Nessun argomento)
Conta_Arg
ECHO Valore ERRORLEVEL: %ERRORLEVEL%
ECHO ---ESECUZIONE 2 (Due argomenti)
Conta_Arg AVE STUDENT
ECHO Valore ERRORLEVEL: %ERRORLEVEL%
```

Analizzando la sua esecuzione notiamo che **ERRORLEVEL** assume il valore indicato come argomento nell'istruzione **EXIT** del nostro programma in C++.

```
C:\arch_def_1\Scuola\SW-
Usato\Prove>TestExit
---ESECUZIONE 1 (Nessun argomento)
Report dal C++   : Nr. argomenti: 0
Valore ERRORLEVEL: 0
---ESECUZIONE 2 (Due argomenti)
```

```
Report dal C++ : Nr. argomenti: 2
Valore ERRORLEVEL: 2
```

C) E' possibile crearsi un piccolo programmino di input utilizzando il codice assembler. Utilizzando la sequenza qui sotto costruiamo un file (ad esempio **leggi.asm**) sostituendo la sequenza **<invio>** con dei ritorni a capo effettivi

COMANDI	OUTPUT	COMMENTI
<b>A 100</b>	100	Inizio ad inserire le istruzioni assembler all'indirizzo 100
<b>MOV AH,08</b>		
<b>INT 21</b>		Legge da tastiera il carattere senza visualizzarlo
<b>CMP AL,0</b>		
<b>JNZ 010A</b>		Interrupt 21 - MS-DOS service
<b>INT 21</b>		Confronto AL con zero
<b>MOV AH,4C</b>		se inizia con zero leggo il secondo codice del carattere
<b>INT 21</b>		Interrupt 21 - MS-DOS service
<b>&lt;invio&gt;</b>		
<b>rcx</b>	CX 0000	Chiude il processo restituendo il codice ASCII
<b>e</b>		
<b>n LEGGI.COM</b>		
<b>w</b>	Writing 000E bytes	Interrupt 21 - MS-DOS service
<b>q</b>		
<b>&lt;invio&gt;</b>		

Successivamente sul prompt del DOS digitiamo questo comando:

**debug<leggi.asm**

Adesso nella cartella corrente dovreste trovare un file **leggi.com** di 14 byte. Il programma appena creato può essere utilizzato in questo modo

```
@ECHO OFF
echo Digita un carattere:
leggi
echo Il codice ASCII del carattere
digitato: %errorlevel%
```

come si nota il programma registra nella variabile **%ERRORLEVEL%** il codice ASCII del carattere digitato.

```
E:\Users\Administrator\Desktop>Digita
Digita un carattere: 2
Il codice ASCII del carattere digitato: 50
E:\Users\Administrator\Desktop>
```

questo metodo più complesso ha il vantaggio di poter utilizzare qualsiasi carattere come input.

## IF ELSE - ISTRUZIONE DI SELEZIONE

---

Nei batch file esistono tre **Condizioni** di riferimento indicate in questi esempi:



**IF [NOT] ERRORLEVEL numero istruzione**  
**IF [NOT] stringa1==stringa2 istruzione**  
**IF [NOT] EXIST NomeFile istruzione**

la condizione **ERRORLEVEL numero** risulta vera se il valore di **ERRORLEVEL** è uguale o maggiore di **numero**. Pertanto per verificare l'esatta uguaglianza di un valore, ad esempio 3 dovrei utilizzare questa formula: **IF ERRORLEVEL 3 IF ERRORLEVEL 4**. L'ordine con cui inserisco il controllo relativo alla variabile **ERRORLEVEL** deve quindi iniziare partendo dai valori più alti e poi in modo decrescente fino ai valori più bassi.

Se il sistema operativo ha attive le "**COMMAND EXTENSION**" è possibile utilizzare anche questa formulazione:

**IF [/i] stringa1 OPERATORE stringa2 istruzione**

dove l'**OPERATORE** di confronto è scelto tra una di queste possibilità

<b>EQU</b>	uguale
<b>NEQ</b>	diverso
<b>LSS</b>	meno di
<b>LEQ</b>	meno o uguale di
<b>GTR</b>	più grande di
<b>GEQ</b>	più grande o uguale

e **/i** indica che il confronto non sarà case sensitive. Questo switch è applicabile anche alla forma **stringa1==stringa2**.

La sintassi estesa per l'**IF** è la seguente:

```

IF Condizione (
    istruzioneIF
) ELSE (
    istruzioneELSE
)

```

ecco uno script di esempio che consente di valutare se ciò che digito non è un numero oppure valutare se si tratta di un positivo, negativo o zero.

```

@ECHO OFF
set STRINGA=
set /P STRINGA=Digita un numero:
SET /A NUMERO=%STRINGA%
if %NUMERO% EQU %STRINGA% (
    if %NUMERO% LSS 0 (
        ECHO %NUMERO% numero negativo
    )
)

```

```

    ) else (
        if %NUMERO% GTR 0 (
            ECHO %NUMERO% VALORE
POSITIVO
        ) else (
            ECHO %NUMERO% ZERO
        )
    )
) else (
    ECHO %STRINGA% NON E' UN NUMERO
)

```

esiste anche una forma compatta:

## IF Condizione (istruzioneIF) ELSE (istruzioneELSE)

ecco un esempio che legge qualcosa da tastiera e visualizza se ho scritto ciao o altro

```

@ECHO OFF
set INPUT=
set /P INPUT=Digita qualcosa:
if /I %INPUT% EQU CIAO (ECHO HAI SCRITTO
CIAO) ELSE (ECHO NON HAI SCRITTO CIAO)

```

## GOTO - ISTRUZIONE DI SALTO

---

L'istruzione GOTO consente di saltare con l'esecuzione ad un altro punto dello script. La sintassi generica è la seguente

### GOTO Etichetta

```

...
:Etichetta

```

Questa istruzione è molto utilizzata per costruire delle procedure iterative  
L'esempio successivo implementa un menu a tre voci

```

@ECHO OFF
:INIZIO
CLS
ECHO 1 - Menu uno
ECHO 2 - Menu due
ECHO 3 - Menu tre
ECHO 4 - Esci dal menu
CHOICE /N /C:1234 /M "Scelta: "

IF errorlevel 4 goto TERMINA
IF errorlevel 3 goto MNU3
IF errorlevel 2 goto MNU2

ECHO Esego il menu uno ...
GOTO FINE
:MNU3
ECHO Esego il menu tre ...
GOTO FINE
:MNU2

```

```
ECHO Eseguo il menu due ...
:FINE
ECHO.
PAUSE
GOTO INIZIO
:TERMINA
```

## FOR - ISTRUZIONE ITERATIVA

---

La sintassi generica del comando **for** è la seguente

```
FOR [/opzione] %%variabile IN (Insieme) DO (
    IstruzioniFOR
)
```

L'**Insieme** può essere un elenco di valori statico come in questo esempio:

```
@echo off
for %%c in (1 2 3 4 5 6 7 8 9 10 11 12) do
( echo valore letto: %%c )
```

oppure può fare riferimento a files e a cartelle contenute nella directory indicata dopo la parola chiave **IN** (se omessa fa riferimento alla cartella corrente) e che soddisfano un determinato pattern. Nell'esempio successivo vengono elencati i files che hanno estensione TXT della cartella corrente.

```
@echo off
for %%c in (*.txt) do ( echo %%c )
```

Se il sistema operativo ha attive le "**COMMAND EXTENSION**" è possibile utilizzare anche altre formulazioni del **for**:

**OPZIONE /R:**

L'opzione **/R** consente di condurre una ricerca, basata su un pattern, anche nelle sotto cartelle (ricorsione). Il batch successivo consente di conteggiare tutti i files di word (doc e dot) presenti nell'unità logica corrente.

```
@ECHO OFF
set /A N=0
for /r %%c in (*.do*) do ( set /A N=N+1 )
echo %N% files di WORD
```

**OPZIONE /D:**

L'opzione **/D** consente di estrarre le sole directory. Nel prossimo esempio vengono conteggiate tutte le cartelle (senza estensione) dell'unità logica corrente

```
@ECHO OFF
set /A N=0
for /d /r %%c in (*) do ( set /A N=N+1 )
echo %N% cartelle
```

## OPZIONE /L:

L'opzione /L presenta la seguente sintassi

```
FOR /L %%I IN (ValoreIniziale, Passo, ValoreFinale) DO IstruzioniFOR
```

lo script d'esempio seguente chiede un numero N e stampa la sequenza dei numeri interi da 1 a N

```
@ECHO OFF
SET /P N=Digita un numero:
FOR /L %%I IN (1, 1, %N%) DO Echo %%I
```

## OPZIONE /F:

Ulteriore opzione è /F che consente di analizzare in dettaglio il contenuto di uno/più files oppure l'output di un comando dos. La sintassi nel caso di un file è:

```
FOR /F "Opzioni" %%VARIABLE IN (NomeFile) DO IstruzioniFOR
```

mentre se è analizzato l'output di un comando DOS avremo:

```
FOR /F "Opzioni" %%VARIABLE IN ('ComandoDOS') DO IstruzioniFOR
```

Le "Opzioni" sono:

<b>eol=c</b>	Il simbolo <b>c</b> è da considerarsi come carattere di fine linea e dopo di esso il testo deve essere considerato come un commento
<b>skip=n</b>	Specifica quante linee del file o dell'output del comando, partendo dall'inizio, dovranno essere saltate
<b>delims=xxx</b>	Specifica l'insieme dei simboli da considerarsi come delimitatori. I delimitatori di default sono lo <b>spazio</b> ed il <b>tab</b> . Il comando di esempio (all'interno di un file batch) elenca i soli nomi delle variabili d'ambiente <pre>FOR /F "delims==" %%A IN ('set') DO ( ECHO %%A )</pre> oppure <pre>FOR /F "delims==" %%A IN ('set') DO ECHO %%A</pre>
<b>tokens=x,y,m-n</b>	indica quali elementi (individuati dai delimitatori) di ogni linea devono essere usati dal <b>FOR</b> . Questa opzione causa la creazione automatica di variabili aggiuntive. La forma <b>n-m</b> indica un intervallo di elementi che va dall'n-esimo e l'm-esimo token. Se l'ultimo carattere del tokens è un asterisco <b>*</b> allora nella

	<p>variabile aggiuntiva allocata verrà registrato la parte restante della linea. L'esempio seguente (inserito in un file batch) visualizza l'elenco dei soli valori che le variabili d'ambiente assumono</p> <pre><b>FOR /F "delims== tokens=1,2" %%A IN ('set') DO (ECHO Valore: %%B)</b></pre>
<b>usebackq</b>	<p>questa opzione indica che per l'analisi dell'output di un comando dos uso come delimitatore il carattere ascii 96 e non il singolo apice. Ad esempio:</p> <pre><b>FOR /F "usebackq tokens=*" %%r IN ('help') DO Echo %%r</b></pre> <p>al posto di</p> <pre><b>FOR /F "tokens=*" %%r IN ('help') DO Echo %%r</b></pre>

Vediamo ora un esempio che analizza l'output di un comando dos come il **DIR**:

```
E:\app>dir
Il volume nell'unità E è Volume
Numero di serie del volume: C252-3BA5

Directory di E:\app

31/08/2009  13.59    <DIR>      .
31/08/2009  13.59    <DIR>      ..
31/08/2009  13.59    <DIR>      admin
02/12/2009  11.24    <DIR>
cfgtoollogs
31/08/2009  13.59    <DIR>      DB
31/08/2009  13.44    <DIR>      diag
31/08/2009  13.06    <DIR>
product
                0 File                0 byte
                7 Directory  11.892.981.760
byte disponibili

E:\app>FOR /F "skip=4" %A IN ('dir') DO
@echo %A
31/08/2009
31/08/2009
31/08/2009
02/12/2009
31/08/2009
31/08/2009
31/08/2009
0
7

E:\app>FOR /F "tokens=1-3" %A IN ('dir')
DO @echo %B
volume
di
di
13.59
13.59
13.59
11.24
```

```
13.59
13.44
13.06
File
Directory
```

analizzando l'esempio si osservi:

- se il **FOR** è utilizzato direttamente sulla linea di comando la variabile iterativa viene indicata con un solo **%** (ad esempio scrivo **%A** e non **%%A**)
- inserendo il carattere **@** davanti ad ogni comando della parte iterativa evitiamo che l'output venga "sporcato" con la visualizzazione dei comandi stessi

Vediamo un'altro esempio che rielabora l'output del comando **help**:

```
E:\app>help
Per ulteriori informazioni su uno
specifico comando, digitare HELP nome
comando
ASSOC          Visualizza o modifica le
associazioni alle estensioni dei file.
ATTRIB         Visualizza o modifica gli
attributi del file.
BREAK          Attiva o disattiva il
controllo esteso di CTRL+C.
...
...
VOL            Visualizza l'etichetta di
volume e il numero di serie del disco.
XCOPY          Copia file e alberi di
directory.
WMIC           Visualizza le informazioni
relative a WMI all'interno
della shell dei comandi
interattivi.

Per ulteriori informazioni sulle utilità,
consultare il
riferimento alla riga di comando nella
Guida.
```

Immaginiamo di rielaborare l'output del comando **help** utilizzando in sequenza una serie di pipe verso il comando **find**. Si scriva quindi il seguente comando:

```
help|find /V "          "|find " "
```

```
E:\app>help|find /V "          "|find
"          "
ASSOC          Visualizza o modifica le
associazioni alle estensioni dei file.
ATTRIB         Visualizza o modifica gli
attributi del file.
BREAK          Attiva o disattiva il
controllo esteso di CTRL+C.
...
...
VOL            Visualizza l'etichetta di
volume e il numero di serie del disco.
```

```
XCOPY          Copia file e alberi di
directory.
WMIC           Visualizza le informazioni
relative a WMI all'interno
```

Analizzando l'help in linea del comando `find`

```
E:\app>find /?
Ricerca una stringa di testo in uno o più
file.

FIND [/V] [/C] [/N] [/I] [/OFF[LINE]]
"stringa" [[unità:]
[percorso]nomefile[ ...]]

/V          Visualizza le righe NON
contenenti la stringa specificata.
/C          Visualizza solo il conteggio
delle righe contenenti la stringa.
/N          Visualizza i numeri delle
righe visualizzate.
/I          Ignora maiuscole/minuscole
durante la ricerca della stringa.
/OFF[LINE] Non ignora i file in cui è
impostato l'attributo non in linea.
"stringa"  Specifica la stringa di testo
da cercare.
[unità:][percorso]nomefile
           Specifica uno o più file in
cui ricercare.

Se non viene specificato il percorso, FIND
cerca il testo digitato al prompt
dei comandi o reindirizzato da un altro
comando.
```

si osserva:

- che `find /V " "` mi consente di scartare le righe di output caratterizzati da 15 spazi consecutivi
- il successivo `find " " "` estrae, dalle righe filtrate, solo quelle che hanno almeno 4 spazi

Alla luce di quanto detto il `for` sottostante produce l'elenco dei nomi dei comandi presenti nell'help.

```
E:\app>FOR /F %A IN ('help|find /V "
|^find " "') DO @(echo %A)
ASSOC
ATTRIB
BREAK
...
...
VOL
XCOPY
WMIC

E:\app>
```

si noti:

- che la scritta `^|` indica che il simbolo di pipe va inserito come se fosse un normale carattere nella stringa di comando (eseguita poi dalla clausola **IN** del **for**) e non deve attivare un meccanismo di piping connesso al comando principale **for**.

Analizziamo ora un batch contenente l'istruzione appena analizzata:

```
@ECHO OFF
SETLOCAL ENABLEDELAYEDEXPANSION
SET COMANDI=
FOR /F %%A IN ('help^|find /V
"          "^|find " "') DO (
    SET COMANDI=!COMANDI! %%A
)
ECHO %COMANDI%
```

la sua esecuzione elenca in sequenza tutti i comandi DOS disponibili nel comando **help**.

```
E:\Users\Administrator\Desktop>ComandiDOS
ASSOC ATTRIB BREAK BCDEDIT CACLS CALL CD
CHCP CHDIR CHKDSK CHKNTFS CLS CMD COLO
R COMP COMPACT CONVERT COPY DATE DEL DIR
DISKCOMP DISKCOPY DISKPART DOSKEY DRIVE
RQUERY ECHO ENDLOCAL ERASE EXIT FC FIND
FINDSTR FOR FORMAT FSUTIL FTYPE GOTO GPR
ESULT GRAFTABL HELP ICACLS IF LABEL MD
MKDIR MKLINK MODE MORE MOVE OPENFILES PAT
H PAUSE POPD PRINT PROMPT PUSHD RD RECOVER
REM REN RENAME REPLACE RMDIR ROBOCOPY
SET SETLOCAL SC SHTASKS SHIFT SHUTDOWN
SORT START SUBST SYSTEMINFO TASKLIST TA
SKKILL TIME TITLE TREE TYPE VER VERIFY VOL
XCOPY WMIC
```

L'istruzione **SETLOCAL ENABLEDELAYEDEXPANSION** abilita l'aggiornamento della variabile **COMANDI** quando uso la sintassi **!COMANDI!** all'interno del ciclo **FOR**. In caso contrario la variabile **COMANDI** assume un valore solo nell'ultimo ciclo. Infatti eseguendo questo script

```
@ECHO OFF
SET COMANDI=
FOR /F %%A IN ('help^|find /V
"          "^|find " "') DO (
    SET COMANDI=%COMANDI% %%A)
ECHO %COMANDI%
```

l'output risulta differente:

```
E:\Users\Administrator\Desktop>ComandiDOS
WMIC
```

## UTILI SUGGERIMENTI

### 1) SCRITTURA SU FILE:

Alla base abbiamo i simboli di ridirezione `>` e `>>`. Nell'esempio creo un file **Testo.txt** contenente 4 righe

```
@ECHO OFF
```



```
ECHO Prima riga > TESTO.TXT
ECHO Seconda riga >>TESTO.TXT
ECHO Terza riga >>TESTO.TXT
ECHO -- Creato il %date% alle
%time% >>TESTO.TXT
ECHO Visualizza il FILE
ECHO -----
TYPE TESTO.TXT
ECHO -----
```

**2) ATTESA:**

Per sospendere l'esecuzione di uno script per un certo numero di secondi possiamo usare l'istruzione **CHOICE**

compatibile con choice di Win Vista, 7, 2008Server, 2003Server	compatibile con choice di Win 98 sotto Windows
<pre>@ECHO OFF ECHO Attendere 3 secondi (premi X per terminare) ... choice /C:X /N /T:3 /D:X &gt; NUL  oppure senza i :  choice /C X /N /T 3 /D X &gt; NUL</pre>	<pre>@ECHO OFF ECHO Attendere 3 secondi (premi X per t ... choice /C:X /N /T:X,3 &gt; NUL</pre>

**3) GENERAZIONE NUMERI CASUALI:**

La variabile **%RANDOM%** può essere utilizzata per generare dei numeri casuali compresi da 0 a 32.767. L'esempio proposto genera un numero intero compreso nell'intervallo[1,MAX\_N]

```
@ECHO OFF
SET /A MAX_N=10
SET /A N=%RANDOM%
SET /A N=%N%%MAX_N%+1
ECHO N=%N%
```

**4) AND:**

per controllare che due o più condizioni siano soddisfatte possiamo mettere in serie una serie di **IF**. L'esempio proposto controlla che un punto (x,y) stia nel primo quadrante del piano cartesiano (origine inclusa).

```
@ECHO OFF
set /P STRINGA=Digita X:
SET /A X=%STRINGA%
set /P STRINGA=Digita Y:
SET /A Y=%STRINGA%
if %X% GEQ 0 if %Y% GEQ 0 GOTO
LBL_1QDENTRO
ECHO (%X%,%Y%) NON E' NEL PRIMO QUADRANTE
GOTO LBL_FINE
:LBL_1QDENTRO
ECHO (%X%,%Y%) E' NEL PRIMO QUADRANTE
:LBL_FINE
```

**5) OR:**

per controllare che almeno una tra due o più condizioni sia soddisfatta possiamo mettere in successione, uno per linea, una sequenza di **IF**. L'esempio proposto controlla che un punto (x,y) non stia nel primo quadrante del piano cartesiano.

```

@ECHO OFF
set /P STRINGA=Digita X:
SET /A X=%STRINGA%
set /P STRINGA=Digita Y:
SET /A Y=%STRINGA%
if %X% LSS 0 GOTO LBL_1QFUORI
if %Y% LSS 0 GOTO LBL_1QFUORI
ECHO (%X%,%Y%) E' NEL PRIMO QUADRANTE
GOTO LBL_FINE
:LBL_1QFUORI
ECHO (%X%,%Y%) NON E' NEL PRIMO QUADRANTE
:LBL_FINE

```

## ESEMPI

### esempio 1

Vediamo ora un esempio di comando batch che utilizza l'istruzione IF. Creiamo un file **scelta.bat** ed incolliamo il contenuto del box sottostante.

```

@echo off
if "%1"==" " goto LBL_HELP
SET MATERIA=%1
if /i "%MATERIA%"=="SISTEMI" goto LBL_OK
if /i "%MATERIA%"=="INFORMATICA" goto
LBL_OK
goto LBL_ERRORE

:LBL_OK
@echo -----
---
@echo OK: HAI DIGITATO %MATERIA%!
@echo -----
---
goto LBL_FINE

:LBL_HELP
@echo -----
---
@echo HELP: DIGITA SISTEMI O INFORMATICA!
@echo -----
---
goto LBL_FINE

:LBL_ERRORE
@echo -----
---
@echo ERRORE! DIGITA SISTEMI O
INFORMATICA!
@echo -----
---
goto LBL_FINE

:LBL_FINE
SET MATERIA=

```

Per utilizzarlo sul prompt dei comandi digitiamo **scelta** con il nome di una materia (**che nel nostro caso deve essere "SISTEMI" o "INFORMATICA"**). La sequenza di **if**

genera dei salti a sottosezioni di codice a seconda del parametro digitato. In generale gli **if** vengono usati per saltare nel punto giusto del programma che gestisce quella particolare casistica.

## esempio 2

Analizziamo un ulteriore file batch che consente di elencare tutti i files con una certa estensione:

```
@echo off
if "%1"==" " goto LBL_HELP
@echo -----
---
@echo ELENCO DEI FILES CON ESTENSIONE %1
@echo -----
---
for %%i in (%1) do echo %%i
goto LBL_FINE

:LBL_HELP
@echo -----
---
@echo HELP: digitare ad esempio:
@echo          ricerca *.txt
@echo -----
---
:LBL_FINE
```

Il funzionamento è questo: l'istruzione iterativa di controllo **for** funziona sui files. In altre parole l'argomento **%1** viene utilizzato come argomento del comando **dir** e questo determina la creazione di un insieme di nomi di files che soddisfano il criterio specificato mediante le wildchars **\*** e **?**. La variabile **%%i** assume in sequenza tutti i valori (nomi di files) dell'insieme determinato da **%1**. Su ogni valore posso eseguire un'operazione (ad esempio la rinomina).

## esempio 3 - espansione del parametro iterativo

Possiamo modificare il comando batch appena proposto immaginando di aggiungere delle opzioni. Ad esempio se si vuole che il batch visualizzi un maggior numero di info relative ai files selezionati possiamo gestire un argomento facoltativo (in 2° posizione) **/E** la cui presenza implica una visualizzazione più estesa (mediante il procedimento di espansione del parametro iterativo: **~<lettera><parametro>**). Ad esempio presento oltre al nome del file anche una serie di info come: il percorso completo, la dimensione, la data di modifica, ...

```
@echo off
if "%1"==" " goto LBL_HELP
if /i "%2"==" /E" goto LBL_LUNGO

:LBL_BREVE
@echo -----
---
@echo ELENCO DEI FILES CON ESTENSIONE %1
@echo - modalita' breve
@echo -----
---
for %%i in (%1) do echo %%i
goto LBL_FINE
```

```

:LBL_LUNGO
@echo -----
---
@echo ELENCO DEI FILES CON ESTENSIONE %1
@echo - modalita' estesa /E
@echo -----
---
for %%i in (%1) do (
  echo -----
  echo FILE: %%i
  echo -----
  echo ~t Ultima modifica      : %%~ti
  echo ~f Full Path (long name^):
%%~fi
  echo ~s Full Path (short name^):
%%~si
  echo ~d Drive                 : %%~di
  echo ~p Percorso relativo     : %%~pi
  echo ~n Nome File (no est.^)  :
%%~ni
  echo ~x Estensione File       : %%~xi
  echo ~z Dimensione in byte    : %%~zi
  echo ~a Attributi File        : %%~ai
  echo ~a Attributi File        :
%%~$PATH:i
)
goto LBL_FINE

:LBL_HELP
@echo -----
---
@echo HELP: digitare ad esempio:
@echo        ricerca *.txt [/E]
@echo -----
---
:LBL_FINE

```

Se creo un file batch [elenca.bat](#) utilizzando queste istruzioni e poi lo eseguo senza e con l'opzione /E ottengo:

```

E:\Documents and
Settings\administrator>c:\elenca.bat *.txt
-----
ELENCO DEI FILES CON ESTENSIONE *.txt
- modalita' breve
-----
pippo.txt

E:\Documents and
Settings\administrator>c:\elenca.bat *.txt
/E
-----
ELENCO DEI FILES CON ESTENSIONE *.txt
- modalita' estesa /E
-----
-----
FILE: pippo.txt
-----
~t Ultima modifica      : 22/02/2010
09.43

```

```

~f Full Path (long name) : E:\Documents
and Settings\administrator\pippo.txt
~s Full Path (short name):
E:\DOCUME~1\administrator\pippo.txt
~d Drive : E:
~p Percorso relativo : \Documents
and Settings\administrator\
~n Nome File (no est.) : pippo
~x Estensione File : .txt
~z Dimensione in byte : 31
~a Attributi File : --a-----

```

## esempio 4

Ecco ora un esempio che consente la gestione di un semplice menu. Alla base il comando **CHOICE** che resta in attesa di un input da parte dell'utente.

La variabile **ERRORLEVEL** è utilizzata da **CHOICE** per indicare quale carattere è stato digitato dall'utente. Il comando **PAUSE** ferma l'esecuzione del comando batch fino al momento in cui l'utente batte invio. I singoli comandi presenti verranno illustrati in dettaglio durante le lezioni. L'opzione **/?** può essere utilizzata per approfondire la sintassi dei comandi presenti in questo batch file.

```

@echo off
set CONTA=

:~-----
:LBL_INIZIO
cls
@echo -----
---
@echo ESEMPIO MENU %CONTA%
@echo -----
---
@echo 1 - Menu A : Visualizza Ora
@echo 2 - Menu B : Visualizza Data
@echo 3 - Menu C : Azzera conteggio
@echo 4 - Exit
@echo -----
---
echo.
:: con Windows XP usare:
:: >> choice /C:12340 /N /T:0,1 >nul
:: con altre versioni di Windows usare:
:: >> choice /C:12340 /N /T:1 /D:0
>nul
choice /C:12340 /N /T:1 /D:0 >nul
:: Elencare i valori in modo decrescente
if errorlevel 5 goto LBL_INCREMENTA
if errorlevel 4 goto LBL_FINE
if errorlevel 3 goto LBL_AZZERA
if errorlevel 2 goto LBL_GIORNO
if errorlevel 1 goto LBL_ORARIO
goto LBL_INIZIO
:~-----
:~-----
:LBL_ORARIO
echo. |time|find "corrente"
pause

```

```

goto LBL_INIZIO
:-----

:-----

:LBL_GIORNO
echo.|date|find "corrente"
pause
goto LBL_INIZIO
:-----

:-----

:LBL_AZZERA
set CONTA=
goto LBL_INIZIO
:-----

:-----

:LBL_INCREMENTA
set CONTA=%CONTA%*
if "%CONTA%"=="*****" goto LBL_AZZERA
goto LBL_INIZIO
:-----

:LBL_FINE

```

l'output ottenuto è il seguente:

```

-----
ESEMPIO MENU *****
-----
1 - Menu A : Visualizza Ora
2 - Menu B : Visualizza Data
3 - Menu C : Azzera conteggio
4 - Exit
-----

```

si noti come il passare del tempo venga evidenziato con l'aggiunta di \* a fianco del titolo del MENU.

## esempio 5

**I files batch rappresentano un meccanismo primitivo di programmazione. Ecco come è possibile implementare una primitiva calcolatrice:**

```

@echo off
:-----

:LBL_AZZERA
set op=
set /A y=0
set /A x=0
set /A q=1
set /A m=0
set Frase="digita il numero o
l'operatore:-> "

:LBL_INIZIO
cls

```

```

echo ----- CALCOLATRICE [solo interi] ---
-----
echo Numeri      : 0 1 2 3 4 5 6 7 8 9
echo Operatori: S(+) D(-) P(x) F(:) U(=)
echo Azioni     : X (Uscita) - Z (Azzera)
echo -----
-----
echo Primo numero      = %x%
echo Operatore [SDPF] = %op%
echo Secondo numero    = %y%
echo Risultato [U]     = %m%
echo -----
-----
:: con Windows XP usare:
:: >> choice /C:1234567890SDPFUZX /N
%Frase%
:: con altre versioni di Windows usare:
:: >> choice /C:1234567890SDPFUZX /N
/M %Frase%
choice /C:1234567890SDPFUZX /N /M %Frase%
if errorlevel 17 goto LBL_FINE
if errorlevel 16 goto LBL_AZZERA
if %q%==3 goto LBL_INIZIO
if errorlevel 15 goto LBL_UGUALE
if errorlevel 14 set op=":"&set /A
q=2&goto LBL_INIZIO
if errorlevel 13 set op="x"&set /A
q=2&goto LBL_INIZIO
if errorlevel 12 set op="-"&set /A
q=2&goto LBL_INIZIO
if errorlevel 11 set op="+"&set /A
q=2&goto LBL_INIZIO
if errorlevel 10 goto LBL_NUMERO0
if errorlevel 9 goto LBL_NUMERO
if errorlevel 8 goto LBL_NUMERO
if errorlevel 7 goto LBL_NUMERO
if errorlevel 6 goto LBL_NUMERO
if errorlevel 5 goto LBL_NUMERO
if errorlevel 4 goto LBL_NUMERO
if errorlevel 3 goto LBL_NUMERO
if errorlevel 2 goto LBL_NUMERO
if errorlevel 1 goto LBL_NUMERO

:-----
:LBL_NUMERO0
if %q%==1 set /a x=x*10
if %q%==2 set /a y=y*10
goto LBL_INIZIO

:-----
:LBL_NUMERO
if %q%==1 set /a x=x*10+%errorlevel%
if %q%==2 set /a y=y*10+%errorlevel%
goto LBL_INIZIO

:-----
:LBL_UGUALE
if "%op%"==" " goto LBL_INIZIO
if %op%=="+" set /a m=x+y
if %op%=="-" set /a m=x-y
if %op%=="x" set /a m=x*y
if %op%==":" set /a m=x/y

```

```
set /a q=3
set Frase="Resetta la calcolatrice (Z) o
esci (X):-> "
goto LBL_INIZIO

:-----
:LBL_FINE
```

che produce il seguente output

```
----- CALCOLATRICE [solo interi] -----
-
Numeri      : 0 1 2 3 4 5 6 7 8 9
Operatori: S(+) D(-) P(x) F(:) U(=)
Azioni     : X (Uscita) - Z (Azzera)
-----
-
Primo numero      = 11
Operatore [SDPF]  = "-"
Secondo numero    = 23
Risultato [U]     = -12
-----
-
Resetta la calcolatrice (Z) o esci (X):->
```