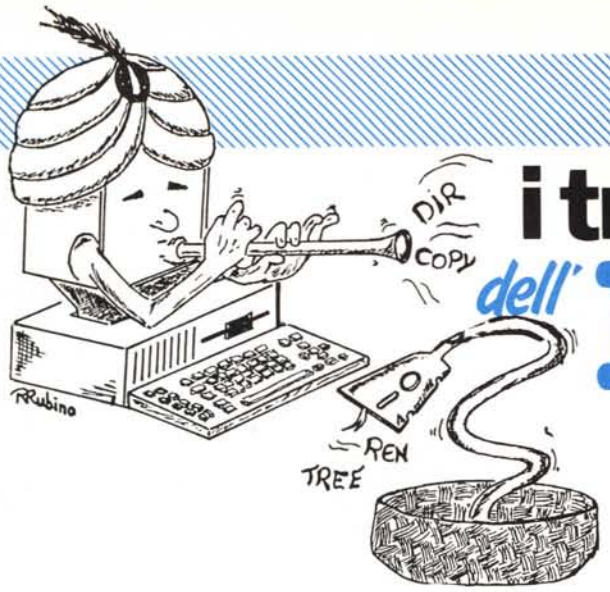


i trucchi dell' MS-DOS

a cura di Pierluigi Panunzi

La programmazione in batch

prima parte

Con il termine «batch» si intende la particolare modalità di esecuzione di uno o più comandi dell'MS-DOS, non da tastiera e perciò da parte dell'operatore, ma da un apposito file di comandi, contenente appunto un insieme di comandi da eseguire l'uno dopo l'altro. Tramite un batch-file si evita dunque, nei casi di sequenze di comandi da ripetere più volte, di dover impostare ogni volta i singoli comandi della sequenza direttamente dalla tastiera, evitando così eventuali errori di digitazione e soprattutto risparmiando tempo. In particolare un file siffatto può avere un filename qualsiasi, ma deve avere l'estensione «.bat»: per eseguirlo basta digitarne il filename senza estensione. Inoltre esiste una serie di comandi interni dell'MS-DOS creati appunto per la gestione dei batch-file, in generale per migliorare (ma non di tanto...) l'interfaccia verso l'utente. Dicevamo dunque che un batch file può in prima analisi contenere una sequenza di comandi da eseguire uno dopo l'altro: analizziamo un primo esempio.

Il batch-file «mle.bat», costituito dalle seguenti linee:

```
masm progr.asm;
link progr.obj;
exe2bin progr.exe progr.bin
```

consente di eseguire in sequenza le tre linee che lo compongono, così come se le avessimo digitate una dopo l'altra, aspettando il termine dell'elaborazione del comando precedente.

In questo caso «mle.bat» consente di assemblare il programma «prog.asm», per ottenere il file «progr.obj», sul quale operare tramite il linker, per ottenere il file «progr.exe» infine da convertire in «progr.bin» tramite il programma «exe2bin»: ecco che perciò (tralasciando il significato dei tre comandi appena impostati) dovendo ripetere i tre comandi un'altra volta basta semplicemente digitare «mle», per attivare appunto il batch-file.

Va subito notato che un batch-file di questo tipo non consente l'esecuzione di altro che quello descritto: in particolare la sua struttura rigida consente l'elaborazione del solo file «progr.asm». Come fare, se dobbiamo operare su di un altro file sorgente in assembler?

Quello che vogliamo ora è creare dunque un nuovo batch-file, stavolta il più generale possibile e che perciò ci consenta di effettuare le operazioni viste su di un file a nostra scelta e variabile volta per volta: tale file lo chiameremo «asmbin.bat» per ricordarci che,

a partire da un «.asm», genera un «.bin» (sempre che tutto vada bene e cioè che non ci siano errori nel sorgente assembler!).

Abbiamo dunque necessità di «parametrizzare» uno o più elementi posti all'interno del batch-file (nel nostro caso il nome del file su cui operare) e ciò si ottiene con l'introduzione, all'interno delle linee di comando, di uno o più «dummy parameters» (chiamati «%0», «%1», ... , «%9») laddove vorremo ottenere per esempio il nome del file: all'atto dell'esecuzione i parametri verranno sostituiti proprio con i valori che noi desideriamo.

Supponiamo perciò di voler parametrizzare il nome del file su cui operano in successione l'assemblatore, il linker ed il comando exe2bin.

Sapendo che il parametro %0 è riservato all'indicazione parametrica del nome del batch file stesso, usiamo il parametro %1 per creare il nostro «asmbin.bat»:

```
masm %1.asm;
link %1.obj;
exe2bin %1.exe %1.bin
```

Ecco che dunque al posto di tutte le ricorrenze del nome del file su cui si opera, apparirà il parametro «fittizio» %1. Ora per eseguire questo batch-file sul file sorgente chiamato «prova.asm», basterà impostare il comando

```
asmbin prova
```

In questo caso «asmbin», oltreché il nome del batch da attivare, è proprio il parametro 0 («%0»), mentre «prova» è il parametro «%1»: eventuali altri nomi elencati successivamente sulla linea di comando diventeranno automaticamente «%2», «%3», ecc. fino a «%9».

Per effetto di tale comando, dunque, tutti i simboli «%n» presenti all'interno del batch-file verranno sostituiti dai rispettivi parametri «attuali»: se ne mancherà qualcuno, non ci saranno problemi, e verrà assunto come valore effettivo la stringa nulla.

Nel nostro caso il primo parametro effettivo del comando così impartito, dato dalla stringa «asmbin», non andrà a sostituire alcun «%0», in quanto non usati all'interno del batch-file, mentre il parametro «%1» (per l'appunto «prova»), andrà a sostituire tutte le occorrenze del simbolo rispettivo.

Nel caso in cui noi digitassimo il comando

```
asmbin
```

dove manca la stringa corrispondente a «%1», allora per essa verrà assunto un valore nullo ed allora, senza alcuna segnalazione di errore da parte del «gestore dei batch» otterremmo l'esecuzione della sequenza di comandi

```
masm .asm;
link .obj;
exe2bin .exe .bin
```

che invece già dalla prima riga genererà errore, in quanto il masm cercherebbe un file chiamato «.asm»; lo stesso succederà sia per il link che per l'exe2bin, i quali non troveranno i file su cui operare.

Facciamo un altro esempio: supponiamo di voler creare un piccolo file di comando che ci consenta di ricercare se all'interno di una directory ci sono file aventi due estensioni a nostra scelta; nel caso di una estensione sola, possiamo usare comodamente il comando «dir», mentre per due estensioni saremmo costretti a ripetere due volte il comando (con grave dispendio di energie...) dapprima con un'estensione e subito dopo con l'altra estensione.

Il nostro batch-file (che chiamiamo stavolta «d2.bat») sarà dunque dato da:

```
dir *. %1
dir *. %2
```

ed ora per essere attivato richiede la presenza di due parametri nella linea di comando stessa: volendo dunque ricercare i file di tipo «.bat» e di tipo «.com» basterà impostare il comando

```
d 2 bat com
```

dove già si può vedere che il parametro impostato andrà a sostituire in tutto e per tutto il parametro %n, laddove sono perciò significativi eventuali blank.

Se ad esempio avessimo scritto, all'interno del batch-file:

```
dir *. %1
```

e cioè con lo spazio tra il punto e il simbolo «%», allora si otterrebbe una segnalazione di errore in quanto il comando diventerebbe ora

```
dir *. bat
```

nel quale «bat» non è uno switch ammesso dal comando «dir» (vedasi a tal proposito il n. 57 di MC).

In questo caso dunque abbiamo due parametri da introdurre, rispettivamente «%1» e «%2», e nel caso in cui nella stringa di comando ne poniamo uno solo, automaticamente verrà associato a «%1»: come dire che non c'è alcun modo di associare qualcosa solo a «%2» e non a «%1». Stesso discorso vale per più parametri successivi, che verranno comunque associati in sequenza.

Come ulteriore esempio supponiamo di volerci creare una serie di batch (strani!) che ci consentano di effettuare più comandi dell'MS-DOS «in una sola linea»: ogni batch avrà un nome che ci indicherà linea per linea il numero di parametri necessari per il suo corretto funzionamento.

Ci spieghiamo dunque con un esempio: supponiamo di creare il batch-file «22.bat», formato di due linee ognuna formata da due parametri, dato da

```
%1 %2
%3 %4
```

Questo batch dunque ci consente di eseguire due comandi ognuno formato da due stringhe!

Ecco che con il comando «in una sola linea»

```
22 dir pippo type pippo.txt
```

otterremo proprio l'esecuzione di

```
dir pippo
type pippo.txt
```

Ecco che perciò possiamo costruirci ad esempio il file «111.bat» formato da tre linee da un parametro l'una e costituito perciò da

```
%1
%2
%3
```

come pure possiamo crearci il file «131.bat», formato da

```
%1
%2 %3 %4
%5
```

che ad esempio potremo usare per eseguire i comandi

```
dir
exe2bin pippo.exe pippo.com
dir
```

con l'unica linea di comando

```
131 dir exe2bin pippo.exe pippo.com dir
```

Notare che in ogni caso l'MS-DOS accetta benissimo nomi di file che iniziano per numero.

Il tutto in questo caso ha una vaga reminiscenza «BASIChiana», quasi come se fossimo riusciti a mettere i numeri di linea a delle linee di comando, dove per giunta troviamo più comandi dell'MS-DOS separati solo da un blank!

Prima di passare ai comandi interni predisposti per l'uso nei batch-file, ricordiamo che all'interno di un batch può trovarsi la chiamata ad un altro batch, ma il guaio è che il «salto» all'altro batch è «senza ritorno», nel senso che eseguito il secondo batch il controllo non ritorna al «chiamante», ma semplicemente all'MS-DOS. Peccato!

Comunque ritorneremo nella prossima puntata su questo argomento e vedremo come si può aggirare l'ostacolo e poter dunque creare dei «nested-batch-file».

I comandi per i batch - echo

Gli ultimi 7 comandi interni dell'MS-DOS sono appunto comandi che si usano al 99% dei casi nei batch-file:

si tratta di comandi «echo», «for», «goto», «if», «pause», «rem» e «shift».

In questa puntata parleremo dei primi tre: iniziamo dal primo, «echo», che consente di abilitare e disabilitare l'eco sullo schermo di quanto il computer sta eseguendo, nonché di stampare sul video un messaggio.

Il comando in esame può assumere una delle quattro forme seguenti, di ognuna delle quali daremo il significato:

1) echo
2) echo on
3) echo off
4) echo <messaggio>

La prima forma è quella che consente di conoscere lo stato dell'eco e cioè consente di sapere se l'eco è o meno abilitato.

La seconda e la terza consentono rispettivamente di abilitare e di disabilitare l'eco sul video.

Per default l'eco è abilitata e cioè a mano a mano che viene eseguito un batch-file, sullo schermo vengono ripetute le linee di comando.

Infine la quarta forma consente di inviare il «<messaggio>» sullo schermo, ad esempio per indicare a quale punto del batch si è arrivati: praticamente è utilizzabile solo quando l'eco è disabilitata, allorché sullo schermo comparirà solamente il messaggio.

Viceversa, se l'eco è abilitata, comparirà dapprima il comando «echo <messaggio>» e sulla linea successiva il «<messaggio>», alquanto bruttino a vedersi...

Conviene dunque disabilitare sempre l'eco, se si vuole inviare un messaggio alla console con il comando «echo».

I comandi per i batch - for

Il comando «for» consente di eseguire un certo comando per ogni valore di un certo parametro, all'interno di un certo set di valori.

La sintassi del comando è la seguente:

```
for %%<ch> in <set> do <comando>
```

Ma vediamo subito con un esempio cosa succede con un comando del genere, prima ancora di definire i vari campi presenti nel comando stesso.

Supponiamo dunque di avere nel nostro dischetto dei file che si chiamano «file.com», «progr.com», «prova.com» e «testo.com», che desideriamo cancellare una volta arrivati ad un certo punto di un file di batch.

Potremmo perciò in prima analisi scrivere le quattro linee seguenti:

```
del file.com
del progr.com
del prova.com
del testo.com
```

oppure, sfruttando il comando «for»:

```
for %%i in (file progr prova testo) do del %%i.com
```

Vediamo dunque passo passo il significato del comando precedente: innanzitutto compare un parametro chiamato «%%i» e cioè con due «percento» a differenza dei «dummy parameter» già incontrati e questo perché in tal modo rimane un simbolo «%i» dopo che tutti i «dummy parameter» sono stati sostituiti.

Il parametro «%i» ora verrà a tutti gli effetti sostituito in ogni sua occorrenza dai valori posti all'interno delle parentesi (il «<set>») e queste sostituzioni avverranno una alla volta, ogni volta andando ad eseguire il comando posto dopo al «do», nel nostro caso «del %i.com».

Sostituendo dunque mentalmente i nomi contenuti tra parentesi ed eseguendo ogni volta il comando così ottenuto, si ottiene nient'altro che l'abbreviazione in una sola linea dei quattro comandi che avevamo visto in prima analisi.

Tornando dunque alla sintassi del comando «for» abbiamo dunque che «<ch>» è un qualsiasi carattere (ad eccezione di 0, 1, ..., 9 per evitare confusioni con i parametri già visti prima), «<set>» è l'insieme di valori assunti dal parametro, racchiusi tra parentesi e separati da un blank ed infine «<comando>» è il comando che si deve eseguire e che presumibilmente conterrà a sua volta all'interno il parametro «%%<ch>»: questo lo diciamo perché in realtà potremmo scrivere un comando del tipo

```
for %%i in (O y & Z) do dir
```

il quale effettua per quattro volte il comando «dir» indipendentemente dal valore assunto dal parametro «%%i», che infatti è stato posto a quattro valori «strani» ed inutili.

Un altro esempio che proponiamo è il seguente

```
for %%a in (dir break echo) do %%a
```

il quale permette di eseguire proprio i tre comandi posti all'interno del «<set>».

Altra possibilità che abbiamo è quella di porre all'interno del «<set>» o di «<comando>» dei «dummy parameter» che verranno dunque sostituiti con i valori attuali forniti nel comando di attivazione del batch: ritorniamo dunque un attimo al primo esempio di batch, che ci aveva condotto alla creazione di «asmbin.bat».

Tale batch ora può essere ulteriormente accorciato, sfruttando anche il fatto che il programma «exe2bin» non richiede esplicitamente il file di destinazione nel comando, in quanto per default genera un file avente per nome quello del sorgente e per estensione «.bin».

In particolare il nuovo «asmbin.bat» diventerà semplicemente:

```
for %%w in (masm link exe2bin) do %%w %1
```

e verrà attivato al solito con

```
asmbin nomefile
```

In realtà i lettori che conoscono bene il «masm» ed il «link» vedranno subito che il comportamento è leggermente differente nei due casi di batch vecchio e di batch nuovo, ma non ci interessa dato che desideriamo solo fare un esempio di utilizzazione del comando «for».

Terminiamo l'analisi del comando «for» dicendo che esso può essere lanciato anche interattivamente, ma in questo caso il doppio «percento» deve diventare singolo: «%%<ch>» in tutte le sue occorrenze deve diventare «%<ch>».

I comandi per i batch - goto

Questo comando è utilissimo per quei batch particolarmente lunghi ed in cui si deve saltare da un punto ad un altro: in tal caso si userà il comando in esame che ha la sintassi seguente:

```
goto <label>
```

dove «<label>» non è altro che l'etichetta della parte del programma a cui si deve saltare: per definire un'etichetta basta seguire la semplice sintassi

```
: <label>
```

Ad esempio supponiamo di dover saltare da un certo punto ad un altro, avente l'etichetta «alfa»:

```
...
goto alfa
...
...
:alfa
...
```

Diamo così appuntamento al prossimo numero dove troveremo, come detto, una prima applicazione veramente avanzata dei comandi dell'MS-DOS, per l'appunto la possibilità (assente a livello di «primitive» del sistema operativo) di chiamare un batch dall'interno di un «batch chiamante» e poi di ritornare senza intoppi al programma chiamante.

MC

Il grande interprete delle esigenze d'azienda.

M.I.D.A.© Un programma che comincia dove gli altri finiscono

La contabilità generale, ordinaria e forfettaria, il magazzino, la gestione degli ordini attivi e passivi (bolle e fatture), l'IVA, il saldo conto, il bilancio. Ecco alcuni dei problemi a cui M.I.D.A.© offre una risposta, una soluzione.
M.I.D.A.©. un programma che, contrariamente ad altri pacchetti software, permette di impostare non soltanto la contabilità generale, ma anche quella analitica.
M.I.D.A.© può fornire il saldo conto, ma non solo quello. Può gestire gli ordini, i solleciti.
Riclassificare i creditori in base alla loro capacità di pagamento, ai loro ritardi. E ancora può occuparsi dei conti correnti, consentire la chiusura del bilancio e l'immediata riapertura del nuovo esercizio.
M.I.D.A.© vi permetterà di organizzare la vostra produzione gestendo la distinta base dei prodotti; sarà strumento essenziale del vostro reparto finanziario al quale fornirà la situazione del portafoglio effetti, a quello amministrativo che supporterà nella fase di analisi di bilancio, a quello del personale con il calcolo automatico della ritenuta d'acconto dei collaboratori.

Un programma di contabilità forfettaria

Fino a oggi contabilità ordinaria e contabilità forfettaria, pur rappresentando spesso due facce di un identico problema, erano tenute ben distinte e separate. Occorreva procurarsi due diversi pacchetti, familiarizzarsi con due diverse procedure e, naturalmente, spendere il doppio. In M.I.D.A.© c'è invece una perfetta integrazione delle due esigenze. Ciò lo rende particolarmente adatto anche a chi, come i commercialisti, si trova a dover fronteggiare problematiche che interessano contemporaneamente piccole, piccolissime e medie aziende.

MODULI DISPONIBILI

Contabilità Generale
Magazzino e Fatturazione
Contabilità + Magazzino + Fatturazione
Modulo di Conversione da M.I.D.A.© ad altri ambienti
Analisi di Bilancio
Gestione Ritenuta d'acconto
Contabilità Analitica
Contabilità Finanziaria
Distinta Base
Gestione Ordini Clienti
Gestione Ordini Fornitori
Gestione Portafogli Effetti

Un programma per i vostri programmi

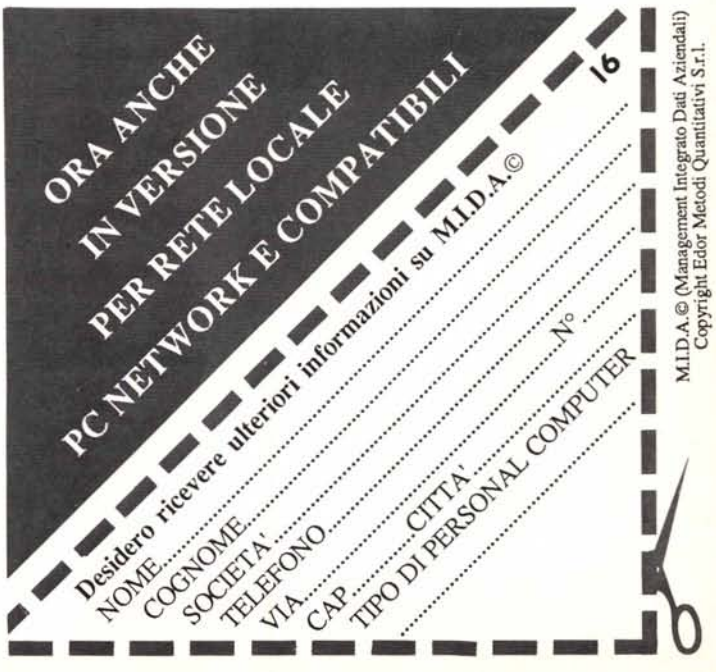
M.I.D.A.© è semplice da usare. Le sue prestazioni sono molto elevate, ma il suo utilizzo è semplicissimo grazie al procedimento di autodefinizione dei menu, grazie alla gestione dinamica e guidata dei tasti funzione, alla chiarezza delle sue istruzioni. Con lui avrete la possibilità di configurare un programma a misura delle vostre reali esigenze.
M.I.D.A.© può trasferire i propri dati dall'area contabile ad altre aree: fogli elettronici integrati (Lotus 1-2-3 e Symphony), word processing (Word e WordStar), database (dBaseIII), diagrammi e grafici (Chart).

Un programma economico

M.I.D.A.© è davvero un buon investimento, soprattutto un investimento a lungo termine che da subito vi permette di lavorare di più e di lavorare meglio. Costa meno di altri pacchetti software: soltanto L. 1.900.000 IVA esclusa (Contabilità Generale + Magazzino + Fatturazione). Ha al suo attivo il pregio di essere un prodotto J.*soft*, la società che distribuisce il più avanzato software internazionale.

Caratteristiche tecniche e operative di M.I.D.A.©

M.I.D.A.© è fatto per essere usato su tutti i personal computer, in ambiente MS-DOS e PC-DOS (IBM PC, XT, AT; OLIVETTI M24 e compatibili).
E' richiesta una memoria centrale minima di 192 Kbyte.
La procedura è totalmente uniforme sia per Winchester che per floppy.
I programmi occupano 2,5 Mbyte, mentre la dimensione degli archivi è limitata unicamente dalla capacità della memoria di massa.



J.soft
Soluzioni senza problemi

Viale Restelli, 5 - 20124 Milano
Tel. 02/6888228 - 683797 - 6880841/2/3

ASSOFT
Socio Fondatore