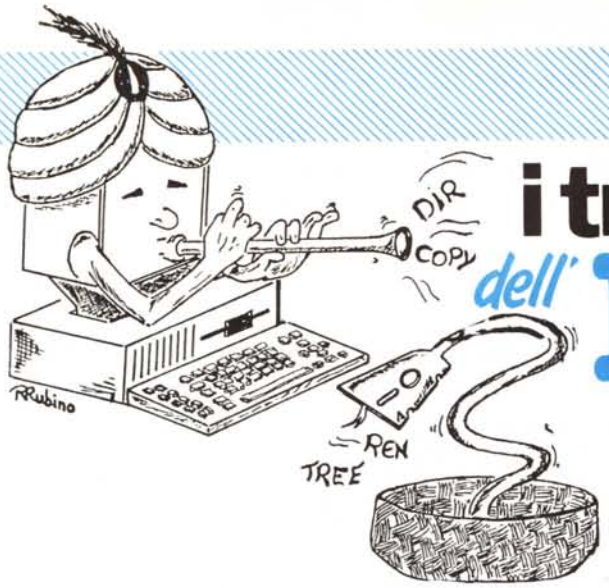

i trucchi dell' MS-DOS

a cura di Pierluigi Panunzi

I comandi interni: exe2bin

Come promesso nella scorsa puntata, ma anche ampiamente anticipato nelle puntate precedenti, ecco che questa volta parleremo in dettaglio del programma «exe2bin.exe», l'ormai ben noto programma che permette di convertire (laddove possibile) un programma avente l'estensione «.exe» in un programma (in genere più corto) avente l'estensione «.bin» o «.com» che dir si voglia.
La nostra analisi si spingerà addirittura andando a disassemblare il «nostro» programma ed osservando quali operazioni compie. Prima dunque di cominciare, consigliamo i lettori interessati di tenere sottomano le ultime due puntate della rubrica, nelle quali abbiamo spiegato in dettaglio le caratteristiche del «program header», quella zona posta all'inizio di un programma di tipo «.exe» e che come sappiamo serve al suo caricamento in memoria.

Il programma «exe2bin»

Iniziamo dunque l'analisi di questo programma ricordando innanzitutto che il suo nome deriva dalla sua funzione principale e cioè far passare da un file di tipo «exe» od un file di tipo «com», in inglese «from exe to bin».

Dal momento che i nomi dei programmi sotto MS-DOS possono essere al massimo di 8 caratteri, ecco che «fromexetobin» non poteva andare bene: eliminato il «from» si otteneva un «exetobin» che già rispondeva alle norme. Però per motivi a noi sconosciuti o solo semplicemente perché gli anglofoni amano i giochi di parole, ecco che quella particella «to» (che rendeva il nome forse difficile da pronunciare o brutto a vedersi...) è diventata subito un «2» che i suddetti anglofoni pronunciano allo stesso modo: il gioco è fatto.

Lasciamo dunque da parte queste considerazioni più o meno folkloristiche ed immergiamoci nell'analisi del programma: in tal modo possiamo vedere quali e quante delle informazioni contenute dal «program header» sono effettivamente utilizzate, almeno da un programma di utilità generale.

Analizzando poi in dettaglio l'«exe2bin» sono saltate fuori qua e là delle utilissime nozioni, in particolare quelle riguardanti la gestione attraverso l'MS-DOS dei file, per mezzo di chiamate a procedure molto potenti ma al tempo stesso facilissime da utilizzare: un discorso più ampio e dettagliato sarà oggetto delle prossime puntate (è una promessa!).

Iniziamo dunque dicendo che abbiamo analizzato, disassemblandolo, il programma «exe2bin.exe» per mezzo dell'omnipresente «debug.com», fornito in genere assieme al sistema operativo: nella nostra analisi siamo stati favoriti senza alcun dubbio dalla piccola dimensione del file da esaminare, meno di 2k byte, considerato poi che si tratta proprio di un file «.exe», dotato cioè di «header», che non è da disassemblare. Considerato che poi ci sono 9 messaggi di errore o diagnostici ecco che in realtà da disassemblare rimane poco più di 1k byte, ben alla portata di un esperto programmatore, magari abituato a «lavoracci» del genere. Eppoi, come vedremo, il programma in sé è alquanto semplice e lineare senza l'ombra di trucchetti difficili da smascherare: insomma proprio un bel programmino....

Prima di proseguire fermiamoci un istante alla sintassi della chiamata del programma in esame, che è data da

```
exe2bin [<drive>]<filename1>[.exe]
[<drive>] [<filename2>.<ext>]
```

dove:

— «drive» può più o meno essere presente laddove il programma da convertire sia in un altro drive diverso a quello di default (ma queste sono cose inutili da dirsi, tanto sono ben note) e laddove il file prodotto debba risiedere in un'altra unità: sappiamo però che l'MS-DOS prevede l'uso di un «path» completo ed all'occorrenza nessuno ci impedisce di utilizzarlo;

— «filename1» è il nome del programma da convertire, al quale possiamo o meno aggiungere l'estensione «.exe»;

— «filename2» è infine il nome del

programma prodotto da «exe2bin» (con o senza estensione), laddove non ci vada bene il nome fornito per default, dato da «filename1» con estensione «.bin», che poi potremmo sempre rinominare in «.com».

In particolare, un comando siffatto

```
exe2bin program
```

convertirà, se possibile, il file «program. exe» generando «program.bin» che per essere eseguito correttamente, deve essere rinominato in «program.com». Viceversa il comando

```
exe2bin program test.com
```

converte il file «program.exe» nel programma «test.com».

Per inciso, invece, il semplice comando

```
exe2bin
```

fornisce il messaggio d'errore «file not found», come era più che lecito aspettarsi, come pure se poniamo nel comando il nome di un file non immediatamente raggiungibile e cioè non presente nella directory di default: sappiamo che in questo caso si può usare comodamente il comando «path» per suggerire al sistema operativo il path da percorrere qualora non trovasse il file in esame nella directory di default.

Un altro inciso curioso prima di proseguire: l'«exe2bin»-izzazione del programma «exe2bin» stesso fornisce un deludente messaggio che suona «file cannot be converted», come dire che non potremo mai ottenere un file «exe2bin.com», a meno di non riscriverlo daccapo!

Detto questo, ci dovremo aspettare all'interno del file disassemblato un controllo della stringa di comando alla ricerca di uno o due nomi validi di programmi, ma andiamo per ordine.

Innanzitutto la prima operazione che «exe2bin» compie è il controllo che la versione di MS-DOS utilizzata sia maggiore o uguale alla 2.0, in quanto alcune caratteristiche del «program header» sono state introdotte solo a partire da tale versione: in caso contrario il programma si ferma subito, emettendo il messaggio «Incorrect DOS version»: per fare ciò viene utilizzata un'apposita chiamata al DOS, sulla quale ritorneremo in seguito.

In caso positivo (versione del DOS accettabile) il programma prosegue andando ad analizzare la stringa di comando per estrarre il nome del file da convertire: nel caso in cui sia presente anche il nome del file da generare anche questo nome viene estratto e memorizzato, per essere usato in seguito. Nel caso che il secondo nome manchi, allora il programma provvederà a duplicare il nome del primo file, aggiungendoci l'estensione «.bin»: il tutto come detto, in previsione dell'uso di tali stringhe contententi i nomi dei due programmi allorché verranno usate le chiamate al sistema operativo per gestire i file in questione.

Nel caso in cui i nomi dei file manchino completamente, sappiamo già che il programma emetterà il messaggio «file not found»: ciò viene fatto controllando la «lunghezza» della stringa rappresentante la cosiddetta «command string» e cioè quella parte della stringa di comando che si ottiene eliminando il nome del programma attivato, nella fattispecie eliminando «exe2bin».

Questa «command string» viene fornita all'interno della struttura chiamata «Program Segment Prefix» (PSP) della quale parleremo in seguito ed è immediatamente disponibile al programmatore in Assembler secondo modalità sulle quali torneremo in dettaglio: in particolare il primo byte della suddetta «command string» rappresenta appunto la lunghezza della stringa vera e propria (nella quale compaiono «blank», «tab» ed altri eventuali caratteri impostati dall'operatore all'atto dell'esecuzione): nel caso che tale lunghezza sia nulla allora vuol dire che il comando digitato era semplicemente «exe2bin» e ciò comporta l'arresto del programma con la diagnostica già vista.

Viceversa una stringa "reale" conterrà come detto anche eventuali caratteri non alfanumerici, che il programma provvede a filtrare e gestire opportunamente.

Create così due stringhe in memoria, contenenti i nomi completi dei due file «sorgente» e «destinazione», il programma è pronto a vedere se effettivamente esiste il primo programma, tentando di effettuare un'operazione di «OPEN FILE», per mezzo di una chiamata al sistema operativo.

Le chiamate al DOS: un primo assaggio

Non vogliamo affrontare in questa puntata l'analisi dettagliata di questo argomento, ma ne parleremo per parecchie puntate, anche in considerazione del fatto che tale argomento è trattato ampiamente dal voluminoso «DOS Technical Reference Manual»: ci limiteremo in questa sede a segnalare i fatti salienti del meccanismo, rimandando come detto alle prossime puntate.

Per effettuare una chiamata al DOS, cioè per ottenere un certo servizio (che va dall'output di una stringa sul video alla gestione dei file), si devono settare opportunamente i registri dell'8088 a seconda di quanto richiesto dal servizio stesso e poi deve essere attivato l'interrupt 21H, per mezzo dell'istruzione Assembler

```
INT 21H
```

per la quale rimandiamo alle prossime puntate della rubrica «Assembler 8086/88».

In particolare nel registro AH viene posto il «numero» relativo al servizio da attivare, mentre negli altri registri andranno ad esempio l'indirizzo di una stringa, il numero di byte da leggere da un file, ecc.

Ecco che ad esempio per effettuare l'output su schermo di un messaggio diagnostico, viene attivato il servizio numero «9» («Print string»), dopo aver caricato nella coppia DS:DX l'indirizzo della stringa da stampare (stringa che deve terminare con il carattere «$», il che ci riporta indietro al glorioso CP/M) e dopo avere messo in AH il valore 9: l'attivazione come detto avviene con l'istruzione «INT 21H».

Senza scendere troppo nei dettagli invece l'operazione di «Open File» si ottiene attivando il servizio «3DH», avendo preventivamente caricato in DS:DX l'indirizzo della stringa (termi-

nante stavolta con un byte nullo) contenente il nome del file da «aprire» ed avendo caricato in AL un certo valore, su cui non ci soffermiamo: una volta attivato questo servizio, otterremo dal sistema operativo una risposta positiva o negativa.

Una risposta negativa sarà contraddistinta dal flag di Carry (CF) settato ed in tal caso AX conterrà il motivo della risposta negativa (ad esempio «file inesistente»), mentre viceversa una risposta positiva avrà il Carry resettato ed in tal caso AX conterrà un numero detto «gestore del file» o «file handler», che il sistema operativo considera univocamente associato al file in questione: per tutte le operazioni successive «sul» file in questione basterà comunicare di volta in volta al DOS tale numero, al che il sistema operativo saprà quale file deve gestire. Questo perché è più semplice ed agevole portarsi appresso una word piuttosto che una stringa: d'altro canto così fanno pure i linguaggi ad alto livello (vedi ad esempio il ben noto «Turbo Pascal»)... oppure fanno così perché è il DOS a richiederlo?!

Evidentemente è vera quest'ultima ipotesi!

Continuiamo l'analisi...

Riprendiamo dunque l'analisi del programma «exe2bin» proprio dove l'avevamo abbandonata: abbiamo perciò generato le due stringhe contenenti i nomi dei due file da gestire.

Ora il programma andrà ad effettuare la chiamata del servizio «Open File» relativa al primo file: se il carry non è settato allora AX conterrà il già citato «gestore del file», che verrà scrupolosamente memorizzato per poter essere riutilizzato nel seguito; in caso negativo invece (ad esempio perché il file non esiste nella directory di default e non è stato attivato in precedenza alcun «path») il programma provvederà ad emettere il messaggio «file not found», sempre per mezzo di una chiamata al già citato servizio «Print String».

Trascurando volutamente di citare il numero e le altre caratteristiche delle chiamate ai servizi del DOS (in quanto ne riparleremo, state tranquilli...), chiamate che si susseguiranno nel corso del programma, ecco che troviamo subito la lettura del file da convertire ed in particolare la lettura dei suoi primi 26 byte, che dalle ultime due puntate della rubrica sappiamo trattarsi proprio del «program header».

La prima cosa che viene in mente da fare per vedere se il file in esame è effettivamente un «.exe» (e guardacaso il «nostro» fa proprio così!) è di testare la prima word dell'«header», che sappiamo valere 5A4DH: in caso negativo si uscirà dal programma con il solito messaggio «file cannot be converted», giustamente.

Di seguito il programma va a leggere la word n. 5, che rappresenta l'ampiezza dell'«header», ci somma il valore 1FH e va a vedere se il tutto è minore di 1000H: in caso contrario il tutto si ferma con l'emissione del messaggio «insufficient memory».

Non sappiamo dare una spiegazione immediata a questo fatto, dal momento che il «.exe» in esame è stato generato dal «link» che ne avrà sicuramente controllato l'ampiezza, per cui suona strano che si testi a questo livello la grandezza soprattutto perché con i 640k byte a disposizione (che salgono a 2M byte e più con le schede di espansione) sembra strano leggere che la memoria è insufficiente: comunque è evidente che ciò è stato fatto a ragion veduta ed è solo che a noi sfugge il significato immediato.

Sembra in definitiva che ci debba essere un'ampiezza limite per l'«header», fatto questo che non è documentato, e che non si può agevolmente gestire.

Per mezzo di un'altra chiamata al DOS, viene spostato il puntatore di lettura del file «.exe» dal punto in cui si trovava (al ventisettesimo byte) all'inizio del programma vero e proprio, posto subito dopo la fine dell'«header».Un successivo confronto dei valori contenuti nelle word 8, 9 e 12 (rispettivamente la variazione del contenuto del registro SS, il valore dell'offset da attribuire al registro SP e la variazione del contenuto del registro CS) controlla che tali valori siano esattamente nulli: infatti sappiamo che un file di tipo «.com» non deve avere uno «Stack segment» (e perciò le parole relative a SS ed SP devono essere nulle) e deve cominciare, a livello codice eseguibile, all'indirizzo 100H (per cui anche CS non può essere alterato, il che comporta che la word relativa dell'«header» deve essere nulla).

Inutile dire che basta che uno dei tre valori sia diverso da zero, che il «file cannot be converted».

Successivamente viene letta la word 11 (offset da attribuire al registro IP) e se tale valore non è nullo allora si vede se per caso è 100H: in caso negativo il file non può essere convertito, mentre se vale 100H allora viene spostato il puntatore del file in lettura di 100H posizioni in avanti rispetto all'attuale: inoltre viene letta la word 4 (numero di «relocation item») e se questa risulta diversa da 0, allora è impossibile la conversione.

Viceversa a questo punto abbiamo il puntatore del file 100H byte in avanti (nel caso in cui la word 11 conteneva tale valore) oppure proprio all'inizio del codice eseguibile, nel caso che la word 11 era nulla.

A questo punto il «nostro» si chiede ancora se la word 4 è diversa da zero, nel qual caso siamo nella condizione che IP è stato specificato (vale 0 o 100H). CS viceversa ha una variazione nulla, ma esistono dei «relocation item»: ciò vuol dire che le locazioni puntate dagli «item» dovrebbero essere alterate da un valore che invece è nullo. Ecco che dunque il nostro «exe2bin» effettua un'operazione particolare, con l'intento di conoscere qual è il valore con cui alterare gli elementi rilocabili: non sapendolo lo chiede all'operatore con il messaggio

"fix-ups needed - base segment (hex):"

al che bisogna inserire un valore esadecimale che rappresenta appunto lo scostamento da aggiungere al CS ed agli elementi rilocabili, in base alla zona di memoria in cui sappiamo che il programma dovrà essere caricato in seguito.

Si tratta ovviamente di un argomento molto delicato e difficile da capire di primo acchitto e probabilmente non ci capiterà mai l'occasione di affrontarlo nella realtà: possiamo supporre che ciò serva in quei casi in cui noi stiamo scrivendo un programma che dovrà risiedere poi su EPROM e perciò ad un indirizzo definito, dove ancora in fase di test abbiamo messo una RAM.

Il sistema operativo in questo caso non può caricare il programma dove vorrebbe lui, ma (si spera, in quanto il manuale non è del tutto esplicito in merito) dove vogliamo noi, e per giunta con gli elementi rilocabili corretti: non basterebbe infatti caricare il programma «dove-vuole-il-DOS» e poi spostarlo opportunamente dove vogliamo noi, in quanto così facendo sarebbero errati tutti i riferimenti rilocabili puntati dai «relocation item».

Sarebbe interessante a questo punto sapere se qualche lettore ha già avuto a che fare con situazioni del genere e come si è comportato: oppure, a voler essere cattivi, si prende un «.exe» qualsiasi e si azzerano «a mano» (con

un qualsiasi correttore-corruttore di file quali il già citato «pctools») le tre word in esame (la 8, la 9 e la 12) e si tenta la conversione vedendo quanto succede, cercando magari valori del «fix-up» tali da mandare in crash il sistema caricandoci sopra il nostro programma...

Continuando dunque l'analisi, troviamo ora, per mezzo di una chiamata al DOS, la lettura completa del file «.exe» in memoria, che ci permetterà poi di effettuare la correzione degli elementi rilocabili, seguendo quanto farebbe il sistema operativo stesso all'atto del caricamento in memoria del file «.exe», subito prima della sua esecuzione: anche in questo caso c'è la possibilità che ci capiti un errore, cioè nel caso in cui l'ampiezza effettiva del file sia minore dell'ampiezza indicata nell'«header». In tal caso il programma non si ferma, ma viene emesso un messaggio diagnostico:

«WARNING - Read error on EXE file
Amount read less than size in header»

che, se non altro, è particolarmente esplicativo.

Se tutto è andato bene, si va a vedere se la word 4 (il solito numero di «relocation item») è diversa da zero (altrimenti si salta al passo successivo): allora si entra in un loop da ripetere tante volte quanti sono gli «item».

In questo loop si sposta il puntatore di lettura del file in modo da puntare all'item corrente (che sappiamo essere due word indicanti l'«indirizzo» dell'elemento da modificare), si legge il contenuto di tali word e si va dunque a correggere l'elemento rilocabile puntato; infine si decrementa di un'unità la word n. 4 (in memoria, non quella vera del file!) e si ripete il tutto fino all'azzeramento del contatore.

A questo punto dunque abbiamo in memoria la versione corretta e convertita del file «.exe» e cioè abbiamo il programma «.com» vero e proprio: non resta altro che salvare il tutto in un file ed i giochi sono conclusi.

In base dunque al secondo nome fornito nella stringa di comando o in base ai «calcoli» interni, ecco che viene eseguita la chiamata di sistema operativo di «Create» (creazione) del file «convertito»: se per qualunque motivo tale operazione fornisce un errore (ad esempio se la directory del dischetto non lo permette) allora il programma si fermerà emettendo il messaggio «file creation error», dopodiché il manuale del DOS consiglia di effettuare un «chkdsk» per vedere di localizzare l'errore.

Invece in caso positivo si scrivono in un sol colpo tutti i byte rappresentanti il programma convertito «nel» file or ora creato, il tutto grazie ad un'altra chiamata al sistema operativo.

Un errore a questo punto potrebbe essere imputato ad uno spazio insufficiente sul dischetto ed infatti il messaggio emesso è «Insufficient disk space».

Terminata invece normalmente l'operazione di scrittura, ecco che con un'altra chiamata al sistema operativo si effettua la chiusura del file «convertito», operazione necessaria in quanto ci avevamo scritto sopra, mentre non è necessario effettuarla per il file semplicemente letto.

Così come termina questa puntata a parer nostro molto difficile da digerire, ma senza dubbio molto istruttiva, termina anche il programma. MC

Texas Instruments TI-95.
Tutto per tenere in mano la vostra professione.

Da molti anni Texas Instruments produce calcolatrici e da sempre conosce le esigenze di chi le utilizza. Per questo ha progettato uno dei più avanzati strumenti di calcolo mai esistiti: TI-95 Procalc.

Fino ad ora alta potenza significava tastiere complicate e sovraccariche. La TI-95 invece ha un design innovativo che permette l'utilizzo di un vasto linguaggio di programmazione con una tastiera semplificata e pratica da usare.

Questo vi permette di accedere facilmente, tramite menù, a più di 200 funzioni. E potete anche creare una serie praticamente illimitata di funzioni con l'esclusivo sistema di menù e tasti virtuali ridefinibili.

Lavorare con la TI-95 è più facile grazie a un display LCD a 2 linee e a una precisione di calcolo interna di 13 cifre.

Inoltre per adeguarla alle vostre esigenze potete suddividere gli 8K di RAM interni. La TI-95 ha fino a 7.200 passi di programma, 900 memorie dati o 6.200 bytes di memoria per memorizzare dati e programmi che vi servono di più.

Potete anche inserire un modulo di memoria aggiuntiva da 8K nel connettore per moduli.

Come software opzionale sono disponibili: Matematica e Statistica.

Se preferite una calcolatrice per programmi in Basic, Texas Instruments ha la TI-74 Basicalc che unisce le 70 funzioni di una calcolatrice scientifica ai 113 comandi di un computer programmabile in Basic. In un unico sistema portatile con le stesse dimensioni della Procalc.

Come vedete, alle calcolatrici programmabili Texas Instruments potete chiedere di tutto.

Tranne di fare di più.

Disponibili moduli preprogrammati (MAT-STAT) o da 8K RAM per archiviare dati.

Per favore inviatemi la documentazione tecnica sul TI-95 e sul TI-74.

Nome _____

Cognome _____

Azienda _____

Funzione _____

Indirizzo _____

Spedire a: TEXAS INSTRUMENTS ITALIA S.p.A., Viale Europa 40 - 20093 Cologno Monzese (MI) - Tel. 02/253001.

TEXAS INSTRUMENTS

MC