a cura di Pierluigi Panunzi

I «device driver»

prima parte

Già dal titolo i lettori più attenti arguiranno che non stiamo mantenendo le promesse fatte la scorsa puntata: avevamo infatti detto che avremmo iniziato a parlare della gestione da parte dell'MS-DOS delle unità di memorizzazione di massa... Invece no! Rimandando dunque l'appuntamento a... fra qualche puntata, parleremo in questa e nelle successive di un argomento che riteniamo molto interessante, un po' teorico (il che non guasta!), ma soprattutto parecchio pratico. Parleremo infatti dei cosiddetti «Installable Device Drivers», in gergo semplicemente «driver», i quali sono molto nominati nella letteratura riguardante l'MS-DOS, ma mai esplicitamente descritti: comunque svolgono un ruolo fondamentale, seppure nascosto. Fondamentalmente i driver sono dei programmi, ovviamente scritti in Assembler, che possono essere considerati delle «aggiunte» al sistema operativo e vengono caricati automaticamente all'atto del bootstrap. Prima di andare avanti vediamo dunque di riassumere cosa succede all'interno del computer all'atto del bootstrap da disco, sia subito dopo l'accensione (dopo i vari test dell'hardware del sistema), sia a seguito della pressione dei tasti «Ctrl-Alt-Del»

Il bootstrap

Facciamo una parentesi etimologica: lo sapete da dove deriva tale termine, comunemente usato da tutti i computeristi? Letteralmente la parola è formata dai due termini inglesi «boot» che significa «stivale, scarpone» e «strap» che significa «laccio».

È dunque finalmente chiarito che il termine significa «laccio degli scarponi», il che quadra bene con il fatto che da sempre i computer sono dotati di piedi e di relativa scarpa, ma senza lacci: i bravi softwaristi hanno pensato bene di sopperire alla mancanza...

A parte gli scherzi, il termine è in realtà una parte di una frase che descrive un'immagine impossibile di una persona che si «innalza tirandosi su per i lacci degli scarponi»: tutto questo agli anglofoni è piaciuto molto, tanto che hanno pensato bene di applicare questo concetto astratto dapprima all'elettronica (laddove in un amplificatore un apposito condensatore di «bootstrap» serve a fornire una buona dose di controreazione che fa «innalzare» il guadagno) e poi all'informatica.

In definitiva il concetto di «bootstrap» è associato ad un oggetto che utilizza i propri mezzi per innalzarsi, per procedere.

Certo che avrebbero potuto usare altri termini un po' più consoni e meno fantasiosi, ma sappiamo che fondamentalmente gli informatici vanno matti per i giochi di parole...

Chiusa dunque questa parentesi, vediamo cosa succede nel nostro computer:

— innanzitutto viene caricato da disco, sia esso un floppy disk che un hard disk, il cosiddetto «boot sector» (ariecoci con gli stivali!) che rappresenta nient'altro che un «loader» («caricatore») del sistema operativo vero e proprio;

— ceduto dunque il controllo a tale «loader», questi caricherà in memoria i due file di sistema IO.SYS e MSDOS-.SYS i quali contengono appunto l'MS-DOS;

— il controllo passa ora ai due file caricati e viene dunque testata la presenza sul disco del file chiamato CONFIG.SYS, il quale, se presente, contiene al suo interno informazioni su:

— eventuali «driver» presenti, grazie a linee contenenti istruzioni del tipo

DEVICE = nomefile.est

(dove «nomefile.est» è il nome completo del file che implementa il driver)

— dati di sistema, quali aggiunte all'environment, il numero di file gestibili, ecc., per mezzo di linee di testo contenenti statement del tipo

FILES = 20

— se dunque esistono dei driver, questi vengono caricati ed inizializzati (vedremo dopo cosa significa);

— successivamente viene caricato ed eseguito il COMMAND.COM il quale va a cercare se è presente un file AUTOEXEC.BAT, che, in caso positivo, viene eseguito.

A questo punto il controllo o rimane all'autoexec di cui sopra oppure passa all'utente, libero dunque di caricare ed eseguire i programmi desiderati.

In definitiva dunque il «loader» serve a porre in memoria il sistema operativo MSDOS, il quale poi cerca subito se vi sono delle «appendici» create dall'utente, appendici che appunto sono i «driver»: tutto quello che segue è poi di normale routine e dipenderà dal sistema operativo ed in piccola o grande

offset	L	descrizione
0000	4	puntatore al prossimo header
0004	2	attributi
0006	2	puntatore alla "device strategy routine"
0008	2	puntatore alla "device interrupt routine"
000A	8	nome/unita'

Figura 1 - Struttura di un «device header», cioè la testata che identifica un «device driver».

parte (a seconda dei casi) dai driver. A seconda dunque della funzione dei driver avremo perciò la possibilità di espandere a nostro piacimento il funzionamento del sistema operativo, per la soluzione di problemi che originariamente non erano stati toccati dai progettisti dell'MSDOS, il che non è poco, anzi è veramente notevole.

I driver più noti

Tra i driver già forniti insieme al sistema operativo ne troviamo un paio di ben noti: si tratta del celeberrimo ANSI.SYS e dell'utilissimo VDISK.SYS.

Il primo è un'estensione del DOS che consente di eseguire correttamente le cosiddette «sequenze di Escape», che permettono una gestione universale del video a partire da stringhe opportune di caratteri il cui primo carattere è appunto l'«escape»: si tratta di sequenze che permettono di cancellare lo schermo, settarne gli attributi (colore di sfondo e dei caratteri, formato dello schermo in termini di righe e colonne), di gestire il cursore (posizionamento, movimenti e salvataggio) e da ultimo di ridefinire il funzionamento di alcuni tasti della nostra tastiera.

Considerato che queste sequenze sono standardizzate (appunto dall'ANSI), ecco che in tal modo si è fatto un grosso passo avanti nella universalizzazione e portatilità del software da un computer all'altro.

L'altro driver, il VDISK.SYS, invece serve a creare nella memoria RAM del computer un disco virtuale, che può essere utilizzato a tutti gli effetti come unità a dischi, con tanto di directory, di settori e di tracce, aggiuntiva a quelle presenti nel sistema.

Altri driver sono poi fioriti a seguito dell'introduzione di nuovi dispositivi hardware aggiuntivi: un esempio è il micro-floppy da 3'', che richiede (oltre alla versione 3.3 dell'MSDOS) un driverino aggiuntivo.

Un altro esempio sono le unità a nastro di backup, dotate anche loro di un potente driver, che in alcuni casi può arrivare a parecchi kbyte di programma, magari superando l'ampiezza del sistema operativo stesso.

Un ultimo esempio, molto di moda in questi ultimi tempi, è dato dai lettori di CD, costituenti una memoria di massa di notevoli dimensioni; ebbene è inutile dire che assieme all'hardware, formato dal lettore vero e proprio nonché da una scheda eventuale da porre dentro al PC, viene fornito il relativo driver, in genere accompagnato dal software di gestione del sistema.

I driver

In definitiva dunque il driver deve essere inteso come parte integrante del DOS, ben diverso dunque da un qualsiasi programma applicativo, seppur nettamente specializzato e magari reso residente in memoria e perciò al sicuro da ingerenze da parte di altri programmi.

Questa è una differenza notevole oltreché formale: un programma applicativo deve essere «chiamato» per essere eseguito, viene allocato in memoria là dove il sistema operativo vuole, può magari essere interrotto (ad esempio premendo i tasti «Ctrl-Break») ed al termine verrà «ricoperto» da un altro programma che eventualmente lanciamo; inoltre, grazie ai potenti debugger presenti sul mercato, può essere debuggato anche in corso di esecuzione, verificato, riscritto, ricompilato e rilanciato.

Per quanto riguarda un driver, invece, il discorso è differente: viene lanciato automaticamente dal sistema e posto in memoria al riparo da altri programmi, rimane automaticamente residente in memoria e richiede una maggiore cura soprattutto in fase di sviluppo, dal momento che il suo funzionamento richiede dei particolari meccanismi non direttamente gestibili neppure con il miglior debugger in circolazione.

Infatti il driver deve possedere una certa «testata di identificazione», deve sottostare ad un certo protocollo di scambio dati e informazioni con il sistema operativo, ma a parte questo può far di tutto: chiamate al DOS, settaggio di interrupt vector, lancio di altri programmi, ecc. In definitiva creare un driver richiede da parte del programmatore una maggiore dose di cautela che non quando si ha a che fare con un qualsiasi programma.

La struttura di un «device driver»

In fin dei conti dunque un driver è un programma scritto in Assembler con specifici compiti e doveri: innanzitutto può essere indifferentemente un file di tipo «.COM» o «.EXE», ma deve possedere una particolare testata («header») che identifica il file come «device driver», ne definisce gli attributi e ne fornisce gli entry point.

Come prima regola nello scrivere un driver (che ritroveremo nel seguito), la sua prima istruzione non deve essere posta all'offset 0100H, cosa che invece è tassativa per i «.COM», ma viceversa deve iniziare dall'offset 0000H, a partire dal quale deve essere posto l'«header»: questo perché il driver non ha bisogno di un PSP («Program Segment Prefix»), del quale sappiamo invece che occupa i primi 256 byte di un programma caricato in memoria. Rimandiamo alle precedenti puntate della rubrica per i dettagli sul PSP, che però in questo contesto non ci serviranno minimamente...

Abbiamo parlato di «header»: in particolare si tratta di una struttura composta da 18 byte, posti come detto a partire da offset 0000H e fino ad offset 0011H, suddivisi in vari campi, secondo quanto riportato in figura 1, dove accanto all'offset troviamo la lunghezza del campo ed il significato del campo stesso.

In dettaglio si hanno i seguenti campi:

— all'offset 0000 verrà posto dal sistema operativo il puntatore (completo e perciò nella forma «segment:offset») all'«header» del prossimo device driver (ce ne possono essere infatti più di uno nel sistema): questo campo viene infatti riempito dal DOS all'atto del caricamento dei driver e viene posto con l'offset nella prima word ed il segment nella word più significativa (all'offset 0002), così come vuole la regola solita. In particolare l'utente deve porre questo campo al valore 0FFFFH: 0FFFFH, corrisponente a −1, e ci penserà appunto il DOS ad effettuare la concatenazione in caso di più driver.
— Sul campo «attributi» ritorniamo tra breve.
— I due puntatori successivi contengono l'offset di due entry point del driver stesso, rispettivamente della cosiddetta «device strategy routine» e della «device interrupt routine»: la prima è la routine che viene attivata quando il DOS deve «colloquiare» con il driver per comunicargli l'indirizzo di un buffer contenente in poche parole la funzione che il driver deve svolgere (ad esempio input di un carattere oppure output di un blocco, ecc.), mentre la seconda routine è quella che viene attivata subito dopo e che permetterà di eseguire la funzione richiesta.
— Il campo «nome/unità» serve a definire il nome del driver oppure il numero di unità gestite: per entrambe le possibilità chiediamo ai lettori di pazientare, dal momento che ne parleremo tra breve. Come promesso ora torniamo ad analizzare in dettaglio il campo «attributi»: esso consente di specificare le caratteristiche del driver e prevede l'uso di 8 dei 16 bit, secondo lo schema di figura 2 che commenteremo subito.

I «character device», come dice il nome, sono dei driver predisposti ad effettuare l'I/O seriale, un carattere per volta, così come fanno i dispositivi CON, AUX, e PRN: verso tali device si possono aprire dei «canali» attraverso i quali effettuare l'input o l'output.

Invece i «block device» sono dispositivi un po' più complessi in quanto consentono di gestire «blocchi» di dati: sono in genere preposti a gestire dischi fissi o floppy disk (ma non solo quelli...) ed in tal caso con un «blocco» si identifica in genere un settore.

Tali dispositivi, a differenza dei precedenti, non hanno un nome e non possono essere «aperti» come canali, ma viceversa sono contraddistinti da una lettera, proprio come le unità a dischi.

A seconda poi delle unità che costituiscono il driver, ecco che avremo altrettante lettere identificatrici di ogni unità: ad esempio il già citato VDISK.SYS si comporta in maniera da aggiungere una unità a quelle già presenti.

Infatti supponendo che il sistema abbia due floppy disk (A: e B:), ecco che l'unità aggiunta (un disco virtuale) avrà come lettera identificatrice la «C:», mentre nel caso sia già presente un disco rigido (universalmente posto come C:) allora diventerà «D:».

Nel caso di più driver di tipo «block» bisogna stare attenti che le lettere di identificazione possono arrivare solo fino a «Z:» e cioè si possono avere fino a 26 unità (sempre che ciò sia un problema!). Per quanto riguarda il bit 14, quello relativo alle «stringhe IOCTL», diciamo solo che tale bit serve ad indicare se il dispositivo può o meno accettare e perciò gestire delle particolari stringhe di controllo, attraverso un'apposita chiamata del DOS: non approfondiamo l'argomento più di così e ci riserviamo di parlarne in seguito con maggior dovizia di particolari. Mentre sul bit 13 sorvoliamo (ne parleremo più avanti), diciamo che il bit 11 si riferisce al fatto se il driver possa gestire o meno delle unità rimovibili (ad esempio i microfloppy) oppure no (i dischi rigidi).

Il bit 3 è riservato ad un eventuale driver di gestione del clock di sistema, nel qual caso deve trattarsi di un «character device» (quasi ovvio...) e deve prevedersi di settare il bit in questione.

Sul bit 2 possiamo calare un velo di omertà in quanto non appare ben chiaro a cosa serva dal momento che non è possibile assegnare il dispositivo NUL al nostro driver... Infine gli ultimi due bit (quelli meno significativi) servono ad indicare se il nostro driver, che deve essere di tipo «character», sarà il nuovo dispositivo standard di input o output.

bit	se vale «1»	se vale «0»
15	character device	block device
14	gestisce IOCTL	non gestisce IOCTL
13	blocco formato non IBM	blocco formato IBM
11	dispositivi rimovibili	dispositivi non rimovibili
3	clock device	no clock device
2	NUL device	no NUL device
1	standard output	no standard output
0	standard input	no standard input

Figura 2

Tiriamo le somme

Dopo tante informazioni conviene fermarsi un attimo, per rivedere alcuni punti magari ancora rimasti poco chiari.

Abbiamo più volte detto che un driver non è altro che un programma particolare, che non può essere eseguito su comando, ma che viceversa viene attivato automaticamente al bootstrap.

Un driver in genere viene scritto per integrare alcune nuove funzioni nel sistema operativo, funzioni che in via del tutto generale possono essere divise in due parti: da un lato le funzioni su singolo carattere e dall'altro funzioni su blocchi di dati (vettori, matrici o buffer, a seconda dei gusti). Tali funzioni possono, sempre in grandi linee, riferirsi all'input o all'output, allo stato in input e allo stato in output oppure ad altre ancora, tra le quali la funzione di inizializzazione. Per comprendere ancora meglio quanto detto, facciamo un esempio, puramente teorico: supponiamo di voler scrivere il driver di una scheda seriale, generando così un dispositivo chiamato SER.

Tale driver sarà di tipo «character», si chiamerà appunto «SER» e prevedibilmente potrà effettuare le operazioni di inzializzazione del tutto, di input e di output di un carattere ed infine di controllo dello stato in input ed in output.

Suddivideremo quindi questo driver in, perlomeno, cinque routine principali:
— una routine di inizializzazione, per mezzo della quale ad esempio settare le caratteristiche della linea seriale nonché eventuali parametri che ci serviranno in seguito. Come regola fondamentale, tale routine verrà effettuata una sola volta all'atto del caricamento del driver per poi non essere mai più interpellata.
— Una routine di input ed una di output, che provvederanno ad interfacciarsi adeguatamente con le routine di interrupt relative al particolare componente hardware che implementa la comunicazione seriale.
— Una routine di controllo dello stato di input ed una relativa allo stato di output, che servono a verificare la presenza di eventuali errori tanto in input quanto in output con lo scopo di prendere gli opportuni provvedimenti.

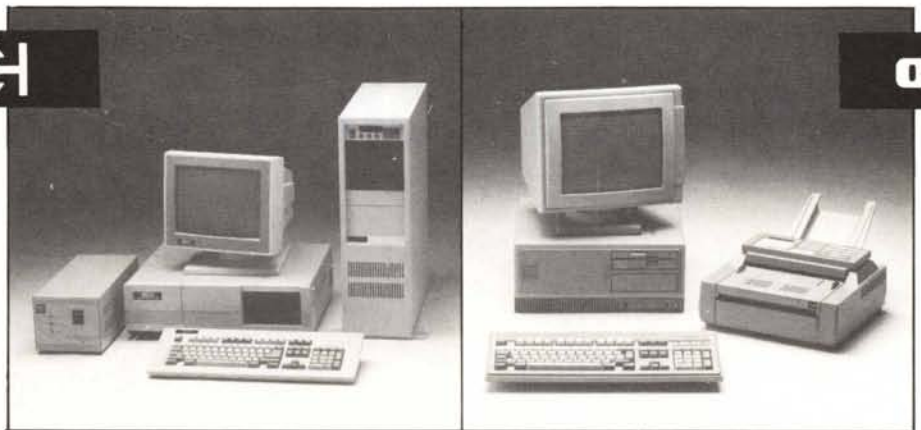
Fatto ciò siamo già a buon punto in quanto basta ora vedere in quale modo il DOS effettua la richiesta al driver di eseguire una certa funzione,

Lo spazio, il solito tiranno, ci costringe ad interrompere a questo punto il ragionamento, che verrà ripreso la prossima puntata.

SOLO I MIGLIORI.
PER VOI.

HTECH

olivetti



olivetti PRODEST

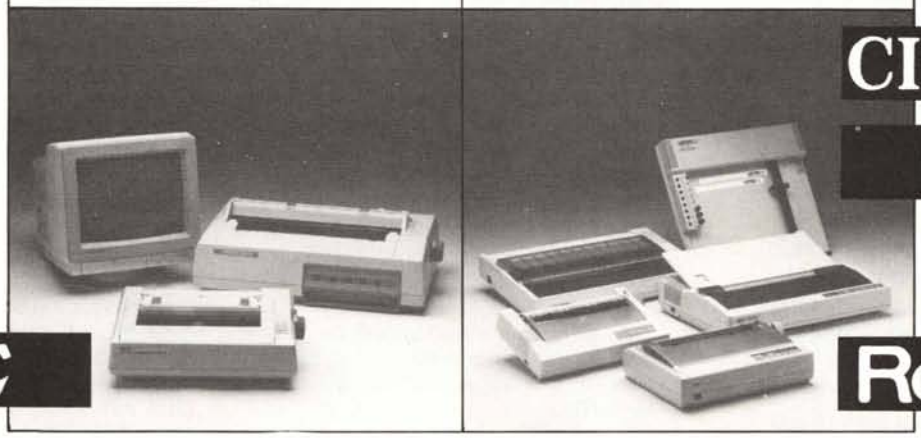
ATARI

CITIZEN

star

NEC

Roland



DISCOM

Discom, ovvero una delle più dinamiche società di distribuzione nate negli ultimi dieci anni. Discom si è imposta sul mercato grazie alla continuità del suo servizio, alla possibilità di offrire il prezzo migliore, alla capacità di scegliere sempre i prodotti vincenti, cioè i migliori, per voi.

00128 Roma - Via Marcello Garosi, 23
Telef. (06) 52.07.839-52.07.917 - Telex 620238 - Telefax (06) 52.05.433