di Pierluigi Panunzi

I «device driver»

quinta parte

Eccoci dunque alla quinta ed ultima parte (almeno per quanto riguarda la «teoria») di una serie di articoli riguardanti i «device driver»: dopo tanta teoria passeremo poi alla pratica andando ad analizzare un driver molto noto, quell'«ANSI.SYS» presente in tutti i dischi di sistema, ma che probabilmente non viene utilizzato dai programmatori, vuoi perché non è velocissimo, vuoi perché i manuali non sono ben chiari in proposito

La funzione NON DESTRUCTIVE INPUT NO WAIT

Si tratta senza dubbio della funzione con il nome più strano in assoluto: in definitiva si tratta di una normale funzione di Input, ma con una piccola differenza. In particolare dunque la funzione INPUT fornisce al DOS il successivo carattere ricevuto in input (in genere dalla tastiera, oppure come output da un programma), carattere che è di solito presente in un apposito buffer: con la funzione INPUT il carattere viene estratto dal buffer (ed eliminato da questo) e viene subito processato dal DOS stesso a seconda del suo valore. In questo caso è proprio il DOS a sapere che dovrà prendere dal driver un carattere per processarlo immediatamente.

Invece la funzione in esame fornisce ancora una volta il carattere successivo da un apposito buffer, ma non lo eliminerà dal buffer stesso: nel caso di chiamata alla funzione «non distruttiva» infatti, il DOS va solamente a leggere il carattere successivo, senza però elaborarlo immediatamente.

In tal modo il DOS può andare dapprima a dare un'occhiata al carattere successivo, prendere opportuni provvedimenti e poi andare a leggerlo definitivamente: è importante che nel driver si abbia la distinzione tra un input non distruttivo ed un input normale.

Facendo riferimento alla figura 2, vediamo che il carattere letto, ma non eliminato, deve essere messo nell'apposito campo posto all'offset ODH del Request Header, fermi restando i significati dei campi precedenti.

codice	funzione	
	block device	character device
00	INIT	INIT
01	MEDIA CHECK	NOP
02	BUILD BPB	NOP
03	IOCTL input	IOCTL input
04	INPUT	INPUT
05	NOP	NONDESTRUCTIVE INPUT
06	NOP	INPUT STATUS
07	NOP	INPUT FLUSH
08	OUTPUT	OUTPUT
09	OUTPUT with VERIFY	OUTPUT with VERIFY
0A	NOP	OUTPUT STATUS
0B	NOP	OUTPUT FLUSH
0C	IOCTL output	IOCTL output
0D	DEVICE OPEN	DEVICE OPEN
0E	DEVICE CLOSE	DEVICE CLOSE
0F	REMOVABLE MEDIA	NOP

Figura 1 - Ecco qui la solita tabella dove sono riportate tutte le funzioni relative ai drive di tipo «blocco» o di tipo «carattere».

Le funzioni INPUT STATUS ed OUTPUT STATUS

Abbiamo riunito insieme le due funzioni che riguardano lo stato di due operazioni di per sé contrastanti, in quanto, viste dalla parte del DOS hanno in comune la struttura del Request Header, (che vediamo in figura 3), il quale non è altro che l'header «nudo e crudo», senza aggiunte altri campi.

Per quanto concerne il funzionamento del driver nei due casi, tale funzione, o meglio la coppia di funzioni, deve fornire un valore per il bit «Busy» posto all'interno della parola di stato posta a sua volta nel Request Header.

A seconda però che si tratti di stato di input o di output, sempre e solo per un «character device», il driver deve funzionare in modo differente.

In particolare nel caso di input, il DOS si regola in maniera differente a seconda di come trova il bit di Busy:

— se tale bit è posto ad 1, allora significa che l'input sta avvenendo attraverso un dispositivo fisico, per cui il DOS deve attendere che il byte venga posto nel buffer e fornito (su richiesta di input);

— se tale bit è invece posto a 0 allora significa che vi sono già caratteri all'interno del buffer di input, dal quale poi potranno essere letti subito, con una funzione di input, da parte del DOS.

C'è da dire inoltre che il DOS, avendo a che fare con un «character device», si aspetta sempre che questo possieda al suo interno un buffer di input (quello che in gergo si chiama «type-ahead buffer»): per questo motivo i device che non prevedono di usare tale buffer dovranno porre tale bit a 0 (e poi fornire un carattere ben definito alla richiesta di input) per evitare che il DOS attenda all'infinito la lettura di un byte (perché trova il bit Busy posto ad 1) che il driver dovrebbe porre in un buffer che in realtà poi non esiste.

Ricapitolando: se il DOS trova il bit Busy settato, allora si aspetta che il

Comando	NONDESTRUCTIVE INPUT NO WAIT (5)	
indir.	tipo	campo
00	13 BYTE	REQUEST HEADER
0D	BYTE	Carattere letto dal buffer

Figura 2 - Struttura del Request Header nel caso di chiamata alla routine NONDESTRUCTIVE INPUT NO WAIT, per mezzo della quale il DOS può leggere il carattere successivo senza che quest'ultimo venga eliminato dal buffer di input.

Comandi vari		
indir.	tipo	campo
00	13 BYTE	REQUEST HEADER

Figura 3 - Struttura del Request Header nel caso di chiamata alle routine INPUT STATUS, OUTPUT STATUS, INPUT FLUSH, OUTPUT FLUSH, OPEN, CLOSE e REMOVABLE MEDIA: in tutti questi casi, infatti, la funzione non ha bisogno di campi aggiuntivi, ma deve solo settare alcuni campi all'interno del Request Header stesso, come è spiegato in dettaglio nell'articolo.

driver sta leggendo un dato che poi provvederà a porre in un buffer; non appena trova tale bit a 0 allora il DOS potrà leggere il byte. Ecco perciò spiegato il perché dunque il driver che non è dotato di buffer «deve» sempre porre a 0 tale bit Busy: in tal modo il DOS può subito leggere il dato, mentre viceversa attenderebbe all'infinito che il driver resetti tale bit, cosa che invece non avviene mai. Nel caso di funzione di output il bit Busy ha il seguente significato:
— se è posto ad 1 allora significa che è già in corso di esecuzione una vecchia richiesta, per cui il DOS deve attendere;
— se tale bit è invece nullo, allora la richiesta di output potrà essere onorata immediatamente.

In questo caso dunque il significato del bit Busy è proprio quello «canonico» e cioè relativo ad un soggetto che è «occupato» («busy», appunto) nell'esecuzione di una certa operazione e non ne può onorare una successiva se non al termine della precedente.

Le funzioni INPUT FLUSH ed OUTPUT FLUSH

Anche in questo caso, come nel precedente, è comodo analizzare due funzioni, di input e di output, contemporaneamente, in quanto il Request Header è ancora una volta lo stesso, ed uguale a quello del paragrafo precedente (si veda la figura 3): le funzioni in esame non devono aggiungere altri campi, ma ancora una volta dovranno settare opportunamente la parola di stato, posta appunto nel Request Header. In entrambi i casi (sia con la funzione di input che con quella di output) il DOS richiede al driver di terminare, cancellare, le relative richieste di input e di output eventualmente in corso di esecuzione («pending» in gergo), in particolare cancellando completamente il buffer di input (contenente magari caratteri digitati dalla tastiera) oppure quello di output contenente viceversa caratteri in attesa di essere inviati sullo schermo video.

Le funzioni DEVICE OPEN e DEVICE CLOSE

Queste funzioni sono presenti solo per versioni del DOS dalla 3.00 in su e consentono ad un driver relativo a dei cosiddetti «removable media» di conoscere lo stato delle varie operazioni sui file.

Per i dispositivi relativi a «caratteri» tali funzioni di open e di close consentono di inviare al dispositivo una particolare sequenza di inizializzazione (OPEN) e di «chiusura» (CLOSE): in tal modo il dispositivo verrà comunque a trovarsi in uno stato ben noto, ad esempio prima di ricevere in output dei caratteri o viceversa prima di emetterne altri. Due esempi possono essere dati dai driver di una stampante e di un dispositivo di input (ad esempio una tavoletta grafica): nel primo caso, con la funzione OPEN si può inviare alla stampante una sequenza di «escape» per settare il font grafico, le caratteristiche di stampa, ecc, e con la funzione CLOSE si può inviare un «form feed»; invece nel caso della tavoletta grafica con OPEN si potranno inviare comandi di setup e con CLOSE dei comandi di reset.

In entrambi i casi, dopo la OPEN i dispositivi sono sicuramente pronti, rispettivamente, a ricevere ed inviare dei byte.

Per quanto riguarda i «block device» la funzione di open può servire a mantenere aggiornato il conto di quanti file vengono aperti e di quanti vengono chiusi, semplicemente incrementando un contatore (all'inizio nullo...) per ogni OPEN e decrementandolo per ogni CLOSE: quando tale contatore è ritornato a 0 allora si potrà decidere se cancellare o meno gli ultimi buffer utilizzati (gli ultimi dati gestiti), presupponendo che successivamente il «media» può essere anche rimosso, rendendo così inutilizzabili le informazioni poste nei buffer.

La funzione REMOVABLE MEDIA

Siamo arrivati dunque all'utima funzione gestibile con un driver, almeno per ciò che riguarda le versioni 3.00 e 3.10 del DOS.

Anche in questo caso, la funzione ha una certa utilità nel caso che il driver abbia a che fare con «removable media», come fa capire il nome della funzione stessa: in particolare in questo caso il driver resetterà il bit Busy della parola di stato del Request Header (vedasi ancora una volta la figura 3), indicando così al DOS che il dispositivo gestito è di tipo rimovibile. In caso contrario invece il driver setterà il bit Busy, con il che il DOS potrà prendere opportuni provvedimenti.

Un esempio classico è dato dal famoso e famigerato FORMAT, che deve riconoscere, al di là di ogni dubbio ed indipendentemente dal nome dell'unità da formattare, se il dispositivo in questione è un disco rigido o un floppy disk, prendendo opportuni provvedimenti a seconda della risposta ottenuta: il FORMAT in questione può distinguere così situazioni in cui ad esempio il driver «C:» è un hard disk da quelle in cui è viceversa un RAM-disk, oppure un'ulteriore unità a dischetti aggiunta.

Un'occhiata alle sequenze ANSI

Abbiamo già promesso di analizzare «dall'interno» il driver ANSI.SYS, ma

nome	sequenza	significato
CUP	[#];[#]H	"CUrsor Position" : posizionamento del cursore
HVP	[#];[#]f	"Horizontal and Vertical Position" : posizionamento del cursore
CUU	[#]A	"CUrsor Up" : muove il cursore in alto
CUD	[#]B	"CUrsor Down" : muove il cursore in basso
CUF	[#]C	"CUrsor Forward" : muove il cursore a destra
CUB	[#]D	"CUrsor Backward" : muove il cursore a sinistra
DSR	6n	"Device Status Report" : segnala la posizione del cursore
SCP	s	"Save Cursor Position" : salva la posizione del cursore
RCP	u	"Restore Cursor Position" : ripristina la posizione del cursore
ED	2J	"Erase display" : cancella lo schermo (equivale al CLS)
EL	K	"Erase Line" : cancella dal cursore alla fine linea
SGR	#;[#]m	"Set Graphic Rendition" : setta caratteristiche dell'output
SM	#h	"Set Mode" : setta il modo video
RM	#l	"Reset Mode" : resetta il modo video
KKR	#;"s"p	"Keyboard Key Reassignment" : associa la stringa "s" ad un tasto
KKR	"c";"s"p	"Keyboard Key Reassignment" : associa la stringa "s" al tasto "c"

Figura 4 - Elenco delle sequenze ANSI gestibili da DOS attraverso il driver «ANSI.SYS»: tutte le sequenze riportate, come detto nell'articolo, devono essere precedute dal cosiddetto CSI e cioè la coppia di caratteri «escape» e «[», che contraddistinguono le sequenze ANSI da normali caratteri da inviare sul video.

prima di procedere nell'analisi desideriamo premettere alcune informazioni riguardo alle sequenze ANSI.

In particolare le sequenze ANSI sono delle particolari sequenze di caratteri che servono in genere a far compiere delle determinate operazioni al dispositivo che le riceve: si tratta delle sequenze che i computer (non î personal, ma quelli «grandi») inviano ai «terminali» per effettuare certe operazioni, in genere di cancellazione del video, di posizionamento del cursore, ecc.

Per chi non lo sapesse, diciamo infatti che i terminali non sono altro che dei dispositivi di I/O connessi a computer, che servono solamente ad introdurre caratteri tramite tastiera ed a visualizzarne tramite video: come suol dirsi si tratta di dispositivi «non intelligenti», nel senso che pedissequamente provvedono ad inviare quanto viene digitato dalla tastiera ed a mostrare quanto ricevuto dal computer, in generale tramite una linea seriale in «full duplex» e cioè con informazioni che viaggiano nei due sensi, asincronamente.

Il problema era dunque di far capire al terminale l'intenzione, ad esempio, di voler cancellare lo schermo video, senza essere costretti ad inviare come minimo una sequenza di 25 «carriage return — line feed» oppure di voler rappresentare una certa scritta sul video a partire da una certa posizione piuttosto che nella posizione corrente.

Ecco che perciò sono state inventate e codificate delle particolari sequenze di caratteri che vengono univocamente identificate da ogni terminale e che fanno eseguire certe operazioni ben determinate piuttosto che essere semplicemente rappresentate sul display.

In generale sono sequenze di caratteri che iniziano cone «Escape» (1BH) seguito da un carattere «[» (e tale coppia in gergo è denominata «CSI», dalle iniziali di «Control Sequence Indicator»), per proseguire con altri caratteri a seconda della funzione da eseguire: ogni sequenza ha poi un suo nome di tre caratteri che identifica rapidamente la funzione svolta.

Nel caso dell'MS-DOS in particolare si è mantenuta la possibilità di gestire tali sequenze (un limitato sottoinsieme) se non altro per compatibilità verso programmi che magari possono poi essere trasportati su mainframe.

Ecco che perciò per gestire tali sequenze è stato creato il driver «ANSI .SYS», che non è altro che il driver del dispositivo «CON» e che intercetta sequenze di caratteri inizianti per CSI onde poter eseguire correttamente la loro funzione.

Detto questo, vediamo che le sequenze ANSI sono formate da:

— un CSI

— uno o più valori numerici eventualmente separati da un carattere «;»

— una lettera identificatrice della funzione.

Nella tabella di figura 4 vediamo in particolare quali sono le sequenze ANSI riconosciute dall'MS-DOS: con il simbolo «#» indichiamo la presenza di un valore numerico (che può anche mancare, nel qual caso verrà indicato con «[#]».

Ad esempio, vogliamo spostare il cursore alla riga 12, alla colonna 40 (circa a metà schermo, cioè) e scrivere «MC»?

Semplice! La sequenza ANSI in questo caso sarà:

<esc>[12;40HMC

dove:

— con «<esc>» intendiamo il valore 27 decimale, 1BH esadecimale o «^[» (a seconda di come lo si vuole impostare)

— «[» è appunto il secondo carattere del CSI

— «12;40H» serve a spostare il cursore alla riga 12 e alla colonna 40: l'«H» è appunto il comando di posizionamento del cursore

— «MC» non è altro che la stringa di caratteri da inviare così com'è al video.

E se a questo punto vogliamo spostarci in su di cinque linee ed a sinistra di 7 colonne?

Presto detto!

La stringa in esame sarà; la seguente

<esc>[5A<esc>[7D

dove

— «<esc>[» è l'ormai ben nota CSI

— «5A» rappresenta che vogliamo spostarci in su («A») di «5» posizioni

— «<esc>[» è ancora una CSI in quanto deve essere ripetuta ogni volta che si vuole inviare una sequenza ANSI: come dire che le sequenze non sono concatenabili

— «7D» infine indica che si deve spostare a sinistra («D») il cursore di «7» posizioni.

Tornando un attimo alla «non concatenazione» di sequenze ANSI, una sequenza del tipo

<esc>[5A7D

non farebbe altro che scrivere i caratteri «7D» in una posizione sullo schermo 5 righe più in alto rispetto all'ultima posizione in cui abbiamo scritto qualcosa.

Concludiamo la puntata proponendo quanto scrive un manuale dell'MS-DOS versione 3.2 a riguardo della sequenza «DSR» riportata in figura 4:

DSR - Relazione di stato del dispositivo
ESC [6 n.

quando riceve la sequenza di escape DSR, il file di descrizione della console emette la sequenza RCP».

Qualche lettore ci ha capito o qualcosa?!

A risentirci dunque la prossima puntata, laddove tra l'altro spiegheremo il mistero nascosto dietro questa specie di «messaggio cifrato». MC

POSTAL COMPUTER

PC XT IBM COMPATIBILE
L. 750.000
SCHEDA MADRE 6/10 MHZ, 1 DRIVE 360K, SCHEDA CGA O HERCULUS, 256K ESPANDIBILE A 640K SU PIASTRA, TASTIERA AVANZATA 101 TASTI

PC XT IBM COMPATIBILE
L. 1.200.000
SCHEDA MADRE 6/10 MHZ, 1 DRIVE 360K, SCHEDA GRAFICA HERCULUS O CGA, 1 HARD DISK 20 MEGA, 256 ESPANDIBILE A 640K SU PIASTRA, TASTIERA AVANZATA 101 TASTI.

PC PHILIPS 9110
768K 1 DRIVE 5 1/4" e 1 DRIVE 3 1/2"
L. 1.230.000

NOVITÀ
CITIZEN 180 E
COMPLETA DI INTERFACCIA IBM O COMMODORE
L. 380.000

PC AT IBM COMPATIBILE
L. 1.890.000
SCHEDA MADRE 80286, 12 MHZ, O WAIT, 512K ESPANDIBILE A 1024K, 1 DRIVE 5,25" da 1.2 MB 1 HARD DISK DA 20 MB SCEDA HERCULES O CGA TASTIERA AVANZATA 101 TASTI.

TELEFAX MURATA M-1
L. 1.500.000

- COMPATIBILITÀ: G2 G3
- VELOCITÀ DI TRASMISSIONE 15 SECONDI
- APPARECCHIO TELEFONICO A TASTIERA INCORPORATO
- FOTOCOPIATORE
- RICEZIONE AUTOMATICA
- ROTOLO CARTA TERMICA 216 mm × 30 metri.
- OROLOGIO/CALENDARIO DIGITALE

HARD DISK SEAGATE 20 MB	L. 350.000
HARD DISK CONTROLDATA 40 MB	L. 680.000
HARD DISK CONTROLLER PER XT	L. 100.000
HARD DISK CONTROLLER PER AT	L. 220.000
SCHEDA GRAFICA SUPER E.G.A.	L. 300.000
SCHEDA MULTI I/0	L. 110.000
SCHEDA SERIALE	L. 40.000
SCHEDA PARALLELA	L. 35.000
SCHEDA PORTA JOYSTICK	L. 28.000
SCHEDA MADRE XT	L. 190.000
SCHEDA MADRE AT (12 MHZ 0 WAIT)	L. 650.000
TASTIERA AVANZATA 101 TASTI	L. 110.000
DRIVE 5,25 360KB	L. 140.000
DRIVE 5,25 1,2MB	L. 190.000
DRIVE 3,50 720KB	L. 190.000
DRIVE CONTROLLER	L. 49.000
CAVO PARALLELO	L. 15.000
DATA SWITCH A 2 PORTE	L. 60.000
MOUSE ANKO	L. 59.000
JOYSTICK I.B.M. ANKO	L. 45.000

STAMPANTI CITIZEN GRAFICA - NLQ

CITIZEN 120 D L. 360.000 120 CPS, SET. EPSON IBM 80 COL. TRATO IN TRAZIONE. FRIZIONE INTER. OPZIONALE IBM/COMMODORE	**CITIZEN MSP 50** L. 1050.000 250/300 CAR/SEC., 80 COL.
CITIZEN LSP 100 L. 550.000 -160 cps, 80 COL.	**CITIZEN MSP 55** L. 1.230.000 250/300 CAR/SEC. 136 COL.
CITIZEN MSP 10E L. 650.000 - 160 CAR/SEC, 80 COL.	**CITIZEN HQP 40** L. 1.160.000 - 24 AGHI, 200 CPS ALTISSIMA QUALITÀ
CITIZEN MSP 15E L. 680.000 160 CAR/SEC, 136 COL.	**CITIZEN HQP 45** L. 1.530.000 - 24 AGHI, 200 CPS ALTISSIMA QUALITÀ
CITIZEN MSP 40 L. 775.000 - 200/240 CAR/SEC, 136 COL.	**CITIZEN PREMIERE 35** L. 1.250.000 - MARGHERITA PROFESSIONALE, 35 CPS
CITIZEN MSP 45 L. 950.000 - 200/240 CAR/SEC, 136 COL.	**CITIZEN OVERTURE 110** * L. 3.600.000 - STAMPANTE LASER

TUTTI I PRODOTTI CITIZIEN SONO COPERTI DA CERTIFICATO DI GARANZIA DELLA VALIDITÀ DI DUE ANNI

OFFERTA MONITOR

PHILIPS			**Segue PHILIPS**		
MONITOR 8875 14" MULTISINK	L. 935.000	colore	MONITOR 7749 14" TTL		
MONITOR 8833 14" CGA	L. 450.000	colore	compatibile IBM sist. 2	L. 210.000	F/B
MONITOR 8802 14" COLORI	L. 360.000	colore	MONITOR 7513 12" TTL	L. 136.000	F/V
MONITOR 9043 14" EGA	L. 535.000	colore	MONITOR 7713 14" TTL	L. 183.000	
MONITOR 9053 14" EGA	L. 595.000	colore	**ANTAREX**		
MONITOR 9073 14" EGA	L. 680.000	colore	BOXER 14" P39 JAN DUAL	L. 190.000	F/V o F/B
MONITOR 7723 14" TTL	L. 192.000	F/A	BIM 12" PC DM 216B	L. 135.000	F/V
MONITOR 7743 14" TTL	L. 205.000	F/B	CT 9000 SHR EGA JAN	L. 670.000	colore
MONITOR 9082 14" VGA	L. 700.000	colore	CT 9000/L MR14 DIM 414	L. 430.000	colore

PREZZI SU RICHIESTA

GARANZIA 18 MESI

PREZZI IVA ESCLUSA
SPESE DI SPEDIZIONE ESCLUSE

TEL. 06/3652427/3652431

TELEFONATECI