di Tommaso Pantuso

VIC da zero +64

Un File per il video

(prima parte)

Abbiamo visto come codificare il contenuto della memoria video con delle linee di programma contenenti dei DATA e come utilizzarle poi in maniera autonoma. Dobbiamo ora vedere come conservare il contenuto del video su disco. Per far ciò, è bene aver chiara qualche nozione sui file sequenziali.

Gestiamo un file sequenziale

Il nostro scopo, in questa parte, è cercare di mettere tutti a loro agio su come progettare un programma che utilizzi un file sequenziale su disco. Naturalmente, come al solito, l'articolo è rivolto a tutti, anche quelli che si sono accostati da poco al software dei computer Cbm quindi, anche se non entreremo in tutti i dettagli tecnici riguardo alla creazione di file su disco, daremo tutte le informazioni necessarie a permetterne a tutti un buon uso.

Un file viene impiegato tutte quelle volte che si vuol trasferire una certa quantità di dati dal computer ad una periferica. L'esempio più immediato lo abbiamo quando vogliamo effettuare la stampa di un programma su carta: i dati vengono prelevati dalla memoria del computer e, opportunamente codificati, inviati alla stampante la quale provvede a riportarli, mediante una testa di scrittura, sulla carta. Le linee di programma, alla fine della stampa, sono così archiviate su un foglio e noi avremo la possibilità di consultarle a prescindere da ciò che sarà contenuto in seguito nella memoria del calcolatore.

È immediato constatare che un'operazione del genere può essere svolta solo unidirezionalmente, cioè solo dal computer verso la stampante e quindi è impossibile, una volta trasferiti i dati a quest'ultima, riprenderli in caso di necessità. Per poter far ciò, dobbiamo utilizzare un tipo di trasferimento simile, sempre mediante un file, su un altro tipo di periferica, l'unità a dischi, che archivierà i nostri dati su un supporto fisico (il disco) in modo da poterli riprendere quando se ne ha bisogno. Prescindendo per il momento dal trasferimento di dati costituiti da un programma, vediamo come sia possibile, usando un tipo di file strutturalmente molto maneggevole (almeno per certe cose), archiviare dati intesi come singoli elementi da trasferire singolarmente o a gruppi.

Qualche definizione. Un file è un insieme si dati strutturati a gruppi più o meno vasti ciascuno dei quali prende il nome di "record" o "registrazione". A sua volta, ogni record può essere suddiviso in più gruppi che prendono il nome di "campi". Il campo non è ancora l'unità fondamentale in quanto esso può ancora essere suddiviso in sottocampi, ma, per il momento, non abbiamo bisogno di una grossa mole di nozioni in tal senso per capire ciò che andiamo ad illustrare. Un file, per fare il solito esempio, potrebbe contenere il nome e i numeri di telefono di più persone. In questo caso, un singolo record sarebbe composto da

Nome + Numero telefonico

ed i singoli campi sarebbero:

campo 1= Nome
campo 2= numero di telefono.

Questa situazione è illustrata in una delle figure.

Le operazioni da compiere

1) Apertura. — La prima operazione da compiere è quella di comunicare al sistema che abbiamo intenzione di creare un file. Bisogna in pratica fornire un certo numero di parametri che identifichino il tipo di file che si vuol creare, l'operazione che si vuol compiere (lettura, scrittura o altro), un numero di identificazione detto "numero di file", la periferica che si vuol contattare, il nome da attribuire alla massa di dati che nella sua interezza andrà a posizionarsi su di essa, un numero che specifica il canale su cui si vuol operare. La sintassi del comando che sintetizza tutte queste opzioni, nella sua forma più semplice, è la seguente:

OPEN nf, np, cn, "0: nome, tipo, op"

dove:

nf —numero di file;
np —numero di periferica, nel caso del disco è 8;
cn —numero del canale: per quanto riguarda il trasferimento dei nostri dati va da 2 a 14. I numeri 1 e

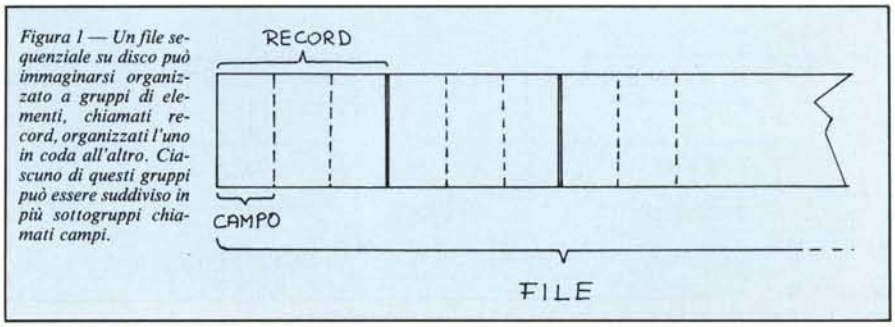

Figura 1 — Un file sequenziale su disco può immaginarsi organizzato a gruppi di elementi, chiamati record, organizzati l'uno in coda all'altro. Ciascuno di questi gruppi può essere suddiviso in più sottogruppi chiamati campi.

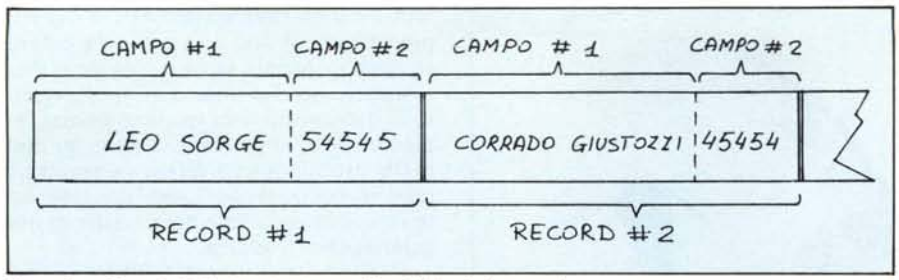

Figura 2 — Esempio di file in cui ogni record è suddiviso in due campi.

15 vengono impiegati per operazioni speciali;

nome —è il nome da attribuire al file

tipo —è il tipo di file: per un file sequenziale è S;

op —è l'operazione che si vuol compiere: R per lettura di dati, W per la scrittura ecc.

Per quanto riguarda il tipo di operazione, oltre a quelle di scrittura e lettura è possibile effettuarne delle altre, abbastanza sconosciute da chi usa il drive 1541 perché non sono evidenziate sul manuale e sui testi specializzati. Noi ci soffermeremo, tra breve, su una di esse: l'Append.

Intanto, date un'occhiata allo schema in una delle figure, quella in cui sono rappresentate, a blocchi, le varie operazioni da compiere nell'uso del file.

La prima cosa che bisogna fare, sia in lettura che in scrittura, è definire tutti i parametri caratteristici. Per fare un esempio, se vogliamo aprire un file sequenziale su disco in scrittura, contrassegnato dal numero 5, sul canale 5 e con il nome Pippo, scriveremo:

```
OPEN 5,8,5"0:PIPPO,S,W"
```

mentre, se vogliamo leggere, cambierà semplicemente l'ultimo parametro; dovremo cioè scrivere

```
OPEN 5,8,5"0:PIPPO,S,R"
```

2) Scrittura. — Una volta aperto il file (con il numero 5), volendo scrivere, dobbiamo iniziare il trasferimento dei dati dall'unità centrale verso la periferica, che nel nostro caso è il disk drive. Supponendo di voler trasferire una variabile A$ al file, scriveremo:

```
PRINT#5,A$
```

che equivale a dire "scrivi (Print) sul file identificato dal numero 5 (#5) la variabile A$". In questo caso il record è formato da una sola variabile che corrisponde ad un solo campo. Se abbiamo la necessità di inviare blocchi di più variabili, possono essere impiegati più modi. Ne illustreremo qualcuno con qualche esempio considerando, per semplicità, di voler trattare solo tre variabili contemporaneamente — A$, B$, C$ — che supporremo introdotte da tastiera con:

```
INPUT A$,B$,C$.
```

Un sistema, il più immediato, è quello di usare tre comandi di Print#, in tre linee di programma, nei seguenti termini:

```
PRINT A$
PRINT B$
PRINT C$.
```

Un altro utilizza una sola linea di programma e un carattere separatore, il chr$(13), che corrisponde al "carriage return", tra le variabili. La scrittura è la seguente:

```
PRINT#5 A$;CHR$(13);B$;CHR$(13);C$.
```

Il terzo sistema è, come scrittura, simile al precedente, solo che il carattere separatore, questa volta, è il chr$(44), che equivale alla virgola.

Naturalmente, a seconda del modo in cui vengono scritte, le variabili sono posizionate in modo diverso sul disco; di ciò comunque non ci occuperemo per il momento. Da un punto di vista della rilettura i primi due modi si equivalgono in quanto, con il primo sistema, i chr$(13) vengono inseriti automaticamente dopo la scrittura di ogni variabile. In ogni modo, altri esempi di scrittura potrete trovarli sul manuale fornito con il drive 1541.

Lettura — La lettura delle variabili inserite nel file può essere effettuata con due comandi:

```
INPUT#
```

e

```
GET#
```

La differenza tra essi è nel modo, più o meno frazionato, di rilevare una qualsiasi variabile contenuta nel file. In altre parole, se leggiamo dal file la parola "CASA", con INPUT# essa sarà rilevata nella sua interezza, cioè sarà letto un blocco formato da quattro caratteri mentre, se leggiamo con GET#, la parola in questione sarà introdotta come C+A+S+A. Non ci soffermiamo oltre su questi punti perché alla fine dell'articolo daremo un programmino dimostrativo che vi permetterà di esercitarvi con le operazioni di lettura e scrittura e potrete così constatare cosa effettivamente avviene nei vari casi.

Un'ultima cosa. Tutti gli elementi posti in un file sequenziale vengono sistemati, per così dire, in fila l'uno dietro l'altro. In coda all'ultimo elemento viene automaticamente posto un flag che segnala la fine del file. In fase di lettura quando viene incontrato il flag in questione, la variabile di stato del sistema, ST, assume il valore 64. Ci possiamo quindi basare sulla verifica di tale variabile per rilevare che il file è stato letto tutto. Ciò ci sarà spessissimo utile, in lettura. Un esempio lo troverete nel programma dimostrativo riportato in queste pagine.

Chiusura. — Dopo ogni operazione effettuata, il file va sempre chiuso e ciò può essere semplicemente ottenuto con:

```
CLOSE N
```

dove N, è il numero di identificazione del file che vogliamo chiudere.

Vogliamo aggiungere alcune cose che ri-

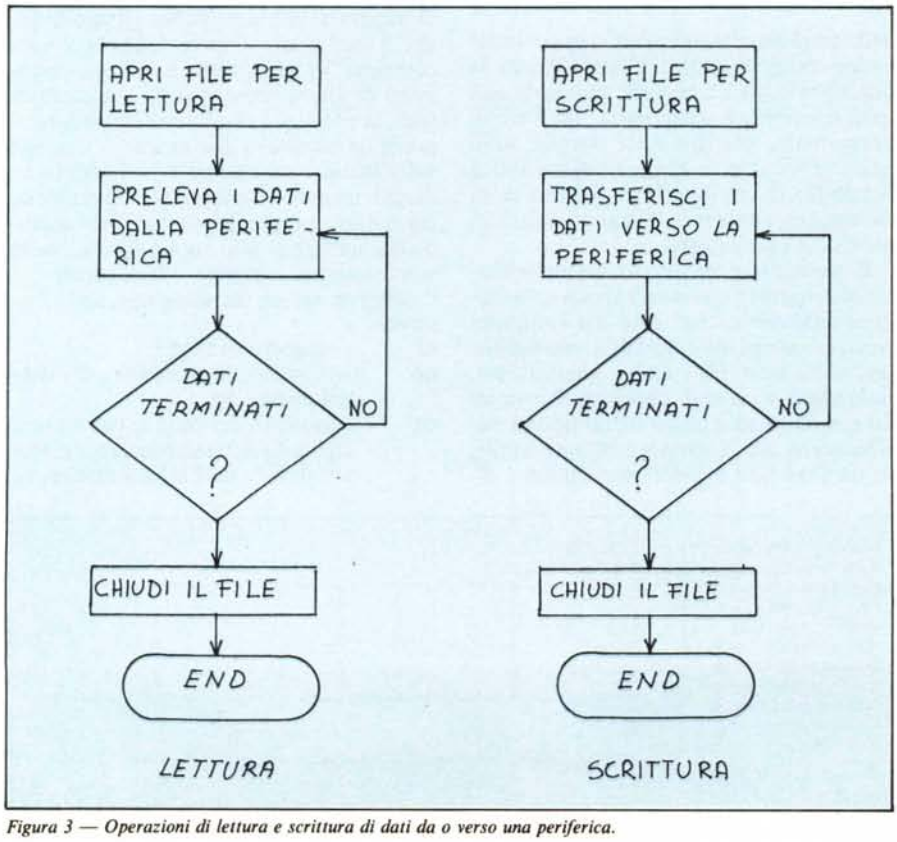

Figura 3 — Operazioni di lettura e scrittura di dati da o verso una periferica.

Figura 4 — In coda al file viene posto un segnalatore (End-Of-File). In lettura, la fine del file è segnalata dalla variabile di stato, ST, che assume il valore 64.

guardano la lettura. La prima è che con INPUT# non possono essere lette variabili composte da più di 79 caratteri perché, in caso contrario, verrebbe segnalato un errore (String too long). La seconda è che, se mettiamo nel file tre variabili separate da una virgola o da chr$(44) (le due cose sono leggermente diverse e lo vedrete facendo degli esperimenti), ciascun gruppo di tre variabili, da un punto di vista della lettura con INPUT#, viene visto come un record che ha come campi le tre variabili. In altre parole, se leggiamo con INPUT#5,A$, ci verrà restituita solo la prima variabile, con INPUT#5,A$,B$ leggeremo le prime due e così via.

Apertura flessibile

Se osservate la sintassi di apertura di file che vi abbiamo mostrato poco fa, vedrete un certo numero di parametri contenuti tra virgolette e precisamente:

```
"0:nome,S,R"
```

Noi possiamo anche non stabilire un formato fisso per quanto è contenuto tra virgolette, ma fare in modo di dargli una struttura tale da poter sostituire, di volta in volta ed a seconda delle opportune necessità, gli adeguati parametri. Spieghiamoci meglio.

Se non sappiamo a priori quale sarà il nome che daremo al nostro file, possiamo introdurre, nei parametri contenuti tra le virgolette, al posto di nome, la generica variabile N$ (potrebbe essere qualunque altra) che di volta in volta potremo sostituire secondo le esigenze (vedi programma dimostrativo). In tal caso, per fare un esempio, il formato di apertura tra virgolette diventa:

```
"0:"+N$+",S,R";
```

è facile osservare come vengono composte, mediante un'operazione di somma, le tre stringhe

```
"0:","N$" e ",S,R"
```

che, insieme, restituiscono l'esatta sintassi.

Naturalmente il discorso può essere ripetuto anche per gli altri parametri (vedi sempre programma dimostrativo).

Così, se, insieme al nome, vogliamo rendere più flessibile anche il tipo di operazione da compiere (lettura, scrittura o altro), potremo scrivere — facendo un esempio con i numeri che identificano il numero di file, la periferica ed il canale nel seguente modo:

```
OPEN5,8,5,"0:"+N$+",S,"+T$
```

dove T$ parametrizza il tipo di operazione da compiere. Per essere ancora più concreti, scriveremo in un programma le linee:

```
10 N$="PIPPO"
20 T$="R"
30 OPEN5,8,5,"0:"+N$+",S,"+T$
```

il programma vede la linea 30 come se fosse:

```
30 OPEN5,8,5,"0:PIPPO,S,R".
```

In ultimo, aggiungiamo una cosa molto importante. Se apriamo un file con un nome che già esiste sul disco, il sistema rivela un errore. Per sostituire ad un file un altro con lo stesso nome, useremo allora, insieme allo "0" che troviamo nella sintassi di apertura, il carattere "@" (chiocciolina) come abbiamo fatto nel programmino proposto.

Ne riparleremo. E vedremo, nel prossimo numero, come "salvare" il contenuto del video sul disco.

APPEND nei file sequenziali

Questa opzione non è conosciuta da molte persone che usano i drive 1541 perché non è presente nei manuali. Vediamo a cosa serve.

Se abbiamo introdotto dei dati in un file sequenziale ed un bel giorno ci accorgiamo di volerne aggiungere altri, non è necessario, come fa la maggior parte degli utenti, riportare tutto il file in memoria per stabilirne la fine, aggiungere i dati che occorrono e registrare di nuovo il file su disco. Questo perché è possibile aggiungere degli elementi in coda al file sequenziale, semplicemente utilizzando l'opzione di APPEND ottenuta introducendo una "A" dove prima ponevamo una "S" o una "R". È possibile quindi, in questo caso specifico, ovviare all'uso di file relativi nei quali, usando gli opportuni puntatori, siamo in grado di identificare l'esatta posizione di ogni record.

```
10 REMARK ---------------------------
15 REMARK ---- LETTURA SCRITTURA ----
20 REMARK ----                   ----
30 REMARK ----  APPEND FILE SEQ  ----
35 REMARK ---------------------------
40 REMARK
50 PRINT"□":POKE53280,0:POKE53281,0
60 PRINT"■":PRINTTAB(210)"1 SCRITTURA"
65 PRINT
70 PRINTTAB(10) "2 LETTURA":PRINT
75 PRINTTAB(10) "3 APPEND"
80 PRINTTAB(212)"SCEGLI"
90 GETA$:IF A$=""THEN90
100 IF A$="1"THENT$="W":GOTO150
110 IF A$="2"THEN300
115 IF A$="3"THENT$="A":GOTO150
120 GOTO90
125 REMARK ---------------------------
130 REMARK --- SCRITTURA / APPEND ----
135 REMARK ---------------------------
150 PRINT"□"
200 PRINTTAB(210)"← PER FINIRE":PRINT
205 INPUT"NOME FILE";N$
210 OPEN5,8,5,"@0:"+N$+",S,"+T$
220 INPUT"INPUT RECORD";F$
230 IFF$="←"THENCLOSE5:GOTO50
240 PRINT#5,F$
250 GOTO 220
255 REMARK ---------------------------
260 REMARK -------  LETTURA  ---------
265 REMARK ---------------------------
300 PRINT"□"
305 INPUT"NOME FILE";N$:PRINT"□"
310 OPEN6,8,6,"@0:"+N$+",S,R"
320 GET#6,F$
330 PRINTF$;
340 IFST=64THEN500
370 GOTO320
400 REMARK ---------------------------
500 PRINT"■ PREMI UN TASTO■"
510 GETA$:IF A$=""THEN510
520 CLOSE6:GOTO50
530 REMARK ---------------------------
```

Programma dimostrativo. Con le nozioni apprese nell'articolo non dovrebbe essere difficile comprendere questo programma nei suoi dettagli. Dopo il Run, comparirà un menu che chiede se vogliamo scrivere (creare il file), leggere o realizzare un Append. Naturalmente, l'Append può essere implementato solo su file già creati e quindi presenti sul disco.