Avete mai visto un programma scritto in C da Maurizio Mangrella? Sì, Mangrella, il nostro affezionato lettore che ci ha tanto cantato sull'uso delle librerie da Basic, ha mandato un programma scritto in C come esempio concreto di utilizzo della trackdisk.device.
Si sarà pentito? Non lo sappiamo né lo crediamo. Di seguito troverete infatti un altro programma dello stesso autore scritto alla vecchia maniera...

La trackdisk.device

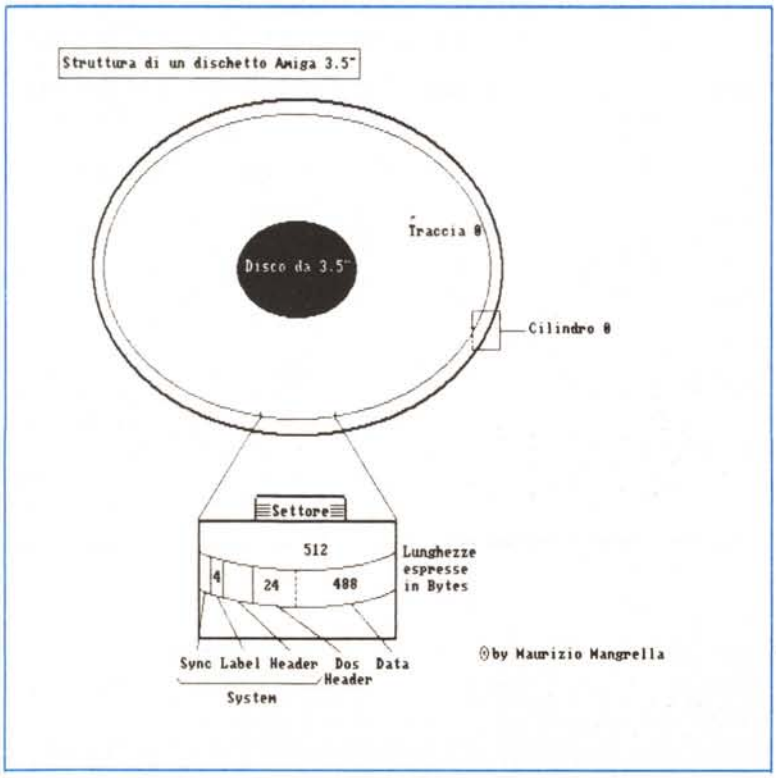
di Maurizio Mangrella - Eboli (SA)

La gestione delle unità a disco nell'Amiga non è affidata, come si potrebbe credere, alle sole routine di filing (Open(), Close(), Read(), Write(), etc.) ma ad un «organismo» più complesso, la trackdisk.device. Ne parlo per esteso (come potrete notare) perché... ne vale la pena!

Ma cosa è?

La trackdisk.device è un normale device, e come tale si comporta: tuttavia comprende alcuni comandi specifici che ne fanno un sistema di gestione dei dischi abbastanza completo ed efficace.

Forse non tutti sanno cosa è un device, perciò cercherò di spiegarlo. Un device è una parte del Sistema Operativo, che quest'ultimo provvede a caricare ogni qual volta ne ha bisogno: il relativo handler può essere collocato nell'ambito del KickStart o presente nella directory DEVS: del WorkBench. Se viene eseguito il comando BindDrivers del CLI (DOS 1.2) i device caricati da disco dopo il bootstrap risiedono in memoria fino a quando il S.O. non ha vitale necessità della memoria che occupano.



Un device è, in sostanza, solo un programma con il quale il S.O. si interfaccia tramite un normale port, attraverso cui vengono passati i comandi e gli argomenti organizzati in una struttura dati detta IORequest; il device stesso «avverte» il task chiamante della fine del compito assegnatogli tramite un altro port. La routine che si occupa di inviare una IORequest è la DoIO(), mentre quella che apre un device «partizionandolo» tra i vari task che lo richiedono è la OpenDevice().

I device, nei confronti del S.O., si comportano in maniera standard, usufruendo di un subset (anch'esso standard) di comandi (v. dopo); ciò rende semplice l'eventuale aggiunta di altri device, scritti all'uopo, e di «logical devices» ad essi associati: a tale scopo è sufficiente scrivere una opportuna MountList (v. relativo paragrafo).

Un device comprende, al suo interno, altre subunità, dette — ma guarda un po'... — Units: ad esempio, il timer.device comprende la UNIT_VBLANK (50 Hz) e la UNIT_MICROHZ (1 MHz). Le Units della trackdisk.device sono i vari drive collegati all'Amiga (bel colpo!).

Richiedendo un accesso alla trackdisk.device si fa concorrenza al Sistema Operativo, che se ne serve per la gestione del drive a basso livello.

Per aprire la trackdisk.device, dopo aver allocato un port con mp_SigTask puntante (!) al nostro Task ed aver allocato memoria per la IOExtTD (v. dopo) bisogna dare:

```
IOErr = OpenDevice(devName,unit,ioRequest,flags)
```

dove, nel nostro caso, devName punta al nome («trackdisk.device\0»), unit è il numero del drive (0, 1, 2 o 3), ioRequest è l'indirizzo della IOExtTD e flags... è meglio porli a 0. IOErr è un codice di errore (0 = tutto OK).

Il listato presentato in queste pagine è solo un esempio.

Come comprenderete agevolmente alla fine dell'articolo, si tratta di un breve programma che carica un settore e lo stampa sulla stampante in formato

Comando	Codice	Funzione
CMD_INVALID	0	Non accettato
CMD_RESET	1	Resetta il device
CMD_READ	2	Legge dal device
CMD_WRITE	3	Scrive al device
CMD_UPDATE	4	Aggiorna i dati del device
CMD_CLEAR	5	Inizializza i dati del device
CMD_STOP	6	Ferma il device
CMD_START	7	Fa partire il device
CMD_FLUSH	8	Forza la scrittura dei dati del device
CMD_NONSTD	9	Offset per comandi non standard

I comandi non standard della trackdisk.device sono :

Comando	Codice	Funzione
TD_MOTOR	9	Accende/spegne il motore del drive
TD_SEEK	10	Sposta la testina ad un dato settore
TD_FORMAT	11	Formatta un dato settore
TD_REMOVE	12	Cancella un dato settore
TD_CHANGENUM	13	Per ottenere il ChangeNum del disco (v. dopo)
TD_CHANGESTATE	14	???
TD_PROTSTATUS	15	???
TD_RAWREAD	16	Legge e scarica in una finestra RAW
TD_RAWWRITE	17	Scrive il contenuto di una finestra RAW
TD_GETDRIVETYPE	18	Per ottenere il tipo di drive
TD_GETNUMTRACKS	19	Per ottenere il numero di tracce disponibili
TD_ADDCHANGEINT	20	Setta un interrupt di cambio disco (v. dopo)
TD_REMCHANGEINT	21	Disabilita il suddetto interrupt
TD_LASTCOMM	22	???

Tabella 1

HEX. Lanciatelo con DiskPrint <drive> <cilindro> <settore> <faccia>, ad esempio: DiskPrint 1 40 5 1.

Comandi e funzionamento

I comandi standard di un device — lo ricordo — sono riportati in tabella 1.

La trackdisk.device adopera una forma particolare di IORequest, che è un'estensione della IOSdReq; nella sua forma completa assume il seguente aspetto:

```
struct IOExtTD {
  struct  Message  io_Message;
  struct  Device   *io_Device;
  struct  Unit     *io_Unit;
  UWORD   io_Command;
  UBYTE   io_Flags;
  BYTE    io_Error;
  ULONG   io_Actual;
  ULONG   io_Length;
  APTR    io_Data;
  ULONG   io_Offset;
  ULONG   iotd_Count;
  ULONG   iotd_SecLabel;
}
```

io_Command è il comando da indirizzare al device: settando il bit 15 (o, in C, or-ando con TDF_EXTCOM) si può chiedere al trackdisk handler di non effettuare il comando se, nel frattempo, il dischetto è stato cambiato (da notare che il trackdisk può effettuare molti comandi anche a drive vuoto).

io_Error riporta un eventuale codice di errore, secondo la seguente tabella:

Errore	Codice
Not Specified	20
No Sector Header	21
Bad Sector Preamble	22
Bad Sector ID	23
Bad header Sum	24
Bad Sector Sum	25
Too Few Sectors	26
Bad Sector Header	27
Write Protected	28
Disk Changed	29
Seek Error	30
No Memory	31
Bad Unit Number	32
Bad Drive Type	33
Drive In Use	34
POST / Reset	35

io_Actual riporta le risposte ai comandi di richiesta secondo lo schema di tabella 2.

io_Length è un dato immediato (0 o 1, tipicamente) o il numero di byte da trasferire alla destinazione io_Data (puntatore a UBYTE). Se io_Lenght è più lungo di un settore, verranno trasferiti più settori in ordine di cilindro (vedi dopo). io_Data (nota IMPORTANTISSIMA) deve puntare ad un'area di memoria CHIP (in quanto il trackdisk fa SEMPRE uso del DMA hardware), pena il blocco totale del computer . io_Offset è l'offset del settore desiderato; si calcola con

Offset = 512*(Sec+11*Head+11*2*Cyl)

dove 512 è il numero di byte per settore, 11 è il numero di settori per traccia, 2 è il numero di facce del disco, Sec è il settore (da 0 a 10), Head è la faccia (0 = Superiore, 1 = Inferiore) e Cyl è il cilindro (da 0 a 79 per i drive 3''1/2, da 0 a 39 per i driver 5'' 1/4). Di solito, però, i blocchi si numerano in maniera progressiva, da 0 a 1759 per un disco da 3''1/2 e da 0 a 879 per un disco da 5''1/4: questa è la numerazione di cui fa uso il mio programma DiskEd.

iotd_Count è il ChangeNum del disco, mentre iotd_SecLabel è la «label» di un dato settore, che dovrebbe essere un LONG int (di più non saprei dirvi).

Traccia o cilindro?

Normalmente la superficie di un dischetto è divisa in piste circolari concentriche, dette tracce: queste, a loro volta, sono divise in un certo numero di settori (sui quali insistono angoli al centro uguali). In molti casi, con il diminuire della lunghezza delle tracce proporzionalmente al raggio delle stesse, diminuisce anche il numero dei settori.

Nell'Amiga (come al solito) le cose vanno diversamente: la superficie del disco è divisa in cilindri (circolari e con-

Comando	Dato in io_Actual
TD_MOTOR	Vecchio stato del motore (0 = spento, 255 = acceso)
TD_CHANGENUM	ChangeNum del disco corrente (in base a questo la trackdisk decide se questo e' stato o meno cambiato)
TD_GETDRIVETYPE	Tipo di drive (1 = 3 pollici 1/2, 2 = 5 pollici 1/4)
TD_GETNUMTRACKS	Numero di tracce disponibili (40 o 80)

Tabella 2

centrici), e, ad ogni cilindro, corrispondono due tracce, una sulla faccia superiore e una su quella inferiore. Ne deriva che, su un disco 3" 1/2 DS DD, avremo 80 cilindri e 160 tracce. Idealmente un cilindro appare, al programmatore, come una integra unità composta di 22 settori (11 per sub-traccia): la distinzione tra le facce, a questo punto, diventa inutile. Così la formula dell'offset potrebbe essere riscritta in

```
/* An example MOUNTLIST file enabling a 5.25" disk to be mounted
   as DF2: and an interactive serial port mounted as AUX:
*/

/* If you only have 1 3.5" disk, change the name to DF1: and the Unit to 1 */
DF2:        Device = trackdisk.device
            Unit   = 2
            Flags  = 1
            Surfaces  = 2
            BlocksPerTrack = 11
            Reserved = 2
            PreAlloc = 11
            Interleave = 0
            LowCyl = 0  ;  HighCyl = 39
            Buffers = 5
            BufMemType = 3
#
```

Figura 2

```
/* BootBlock.h          Commodore Amiga, Inc. */
struct BootBlock {
 UBYTE    bb_id[4];
 LONG     bb_chksum;
 LONG     bb_dosblock;
};
#define BOOTSECTS        2
#define BBID_DOS         { 'D', 'O', 'S', '\0' }
#define BBID_KICK        { 'K', 'I', 'C', 'K' }
#define BBNAME_DOS       (('D'<<24)|('O'<<16)|('S'<<8))
#define BBNAME_KICK      (('K'<<24)|('I'<<16)|('C'<<8)|('K'))
```

Figura 1

Offset = 512*(Sec.+22*Cyl)

dove Sec varierà tra 0 e 21. Altra idiosincrasia (come direbbe il buon Giustozzi): i settori per traccia (indipendentemente dalla lunghezza di quest'ultima) sono sempre 11 e sempre di 512 byte ognuno; ciò richiede spesso che i dischetti siano affidabili (e nella maggior parte dei casi le tracce difettose di molti dischetti sono proprio le più interne).

Dunque, facciamo un po' di calcoli: 512 byte/settore per 11 settori/traccia per 160 tracce fa 901120 byte, cioè esattamente 880 Kbyte; a ciò si aggiunga il fatto che molti copiatori in commercio formattano i cilindri 81 e 82, oltre alla possibilità (hardware) di memorizzare più di 512 byte per settore, almeno nelle tracce più esterne. A svalutare tutte queste capacità è una certa «rigidità» del S.O., il quale, tanto per fare un esempio, salva i file in blocchi con 24 byte di header e 488 byte di dati, col risultato di dover effettuare due DMA per ogni settore (quando potrebbe essere effettuato un solo DMA di oltre 4 Gbyte...!). Il Fast File System (DOS 1.3) eliminerà di sicuro il difetto.

BootBlock

Un'altra device!

È la «bootblock.device», device assolutamente standard con la quale si ha accesso al solo bootblock di un disco: per la cronaca è il device al quale fa riferimento il comando Install del CLI. Ricordandovi che è stato citato nell'articolo «Attenti ai... VIRUS!!» in MC 69 (al quale vi rimando), riporto per completezza in figura 1 il file «devices/bootblock.h» della libreria Lattice.

MountList

Prima di concludere il lungo discorso, lasciatemi dire due parole sulla MountList. Questa è un semplice file di testo presente nella subdirectory DEVS: del WorkBench, e serve a installare eventuali logical device non presenti nel S.O. In figura 2 riportiamo un esempio di MountList.

A parte alcune specifiche tecniche (che vi invito a prendere per buone), notiamo che il device di un drive 5" 1/4 è il trackdisk.device (però!), a cui seguono parametri che dovremmo già conoscere (BufMemType = MEMF_PUBLIC, MEMF_CHIP, memoria CHIP!!). Per «montare» il device DF2: basta dare

Mount DF2:

da CLI.

Conclusioni

Nel ricordarvi che la Commodore-Amiga distribuisce un pacchetto di «Developer's Aid Software», comprendente

La trackdisk. device

```
/*  TrackDisk Demo     by Maurizio Mangrella  1988
 *   Questo demo, molto semplice, legge un settore da un disco
 *  e ne riversa il contenuto sulla stampante in formato HEX .
 *
 *  SINTASSI : DiskPrint <drive> <cilindro> <settore> <faccia>
 *
 *  ESEMPIO  : DiskPrint 1 40 0 1
 */
#include <stdio.h>
#include <exec/types.h>
#include <exec/nodes.h>
#include <exec/lists.h>
#include <exec/memory.h>
#include <exec/interrupts.h>
#include <exec/ports.h>
#include <exec/libraries.h>
#include <exec/io.h>
#include <exec/execbase.h>
#include <exec/devices.h>
#include <devices/trackdisk.h>

#define SPACE  32
#define DEL    127

#define DEVNAME  TD_NAME
#define PORTNAME "DiskPort"

#define TD_NUMSECS  NUMSECS
#define TD_NUMHEADS 2

struct IOExtTD *DiskReq;
struct MsgPort *DiskPort;
UBYTE *diskbuffer;
FILE *fp;

main(argc,argv)
  int argc;
  char *argv[];
{
```

```
int Drive,Cyl,Sec,Hd,ChngNum,Offset,Error;
int x,y;
UBYTE sigBit,Char;

if (argc != 5) {
  fprintf(stderr,"Uso : DiskPrint <Drive> <Cilindro> <Settore> <Faccia>\n");
  exit(20);
}

/*  Inizializza le strutture dati  */

if ((DiskPort = (struct MsgPort *)
  AllocMem(sizeof(struct MsgPort),MEMF_CLEAR)) == NULL) exit(20);
if ((DiskReq = (struct IOExtTD *)
  AllocMem(sizeof(struct IOExtTD),MEMF_CLEAR)) == NULL) CleanExit(20);
if ((diskbuffer = (UBYTE *)
  AllocMem(TD_SECTOR,MEMF_CHIP)) == NULL) CleanExit(20);

Drive = atoi(argv[1]);
Cyl = atoi(argv[2]);
Sec = atoi(argv[3]);
Hd = atoi(argv[4]);

DiskPort->mp_Node.ln_Type = NT_MSGPORT;
DiskPort->mp_Node.ln_Pri = 0;
DiskPort->mp_Node.ln_Name = (char *)PORTNAME;
DiskPort->mp_Flags = PA_SIGNAL;
if ((sigBit = (UBYTE)AllocSignal(-1)) == NULL) CleanExit(20);
DiskPort->mp_SigBit = sigBit;
DiskPort->mp_SigTask = (struct Task *)FindTask((char *)0);
AddPort(DiskPort);

DiskReq->iotd_Req.io_Message.mn_Node.ln_Type = NT_MESSAGE;
DiskReq->iotd_Req.io_Message.mn_Node.ln_Pri = 0;
DiskReq->iotd_Req.io_Message.mn_Node.ln_Name = (char *)0;
DiskReq->iotd_Req.io_Message.mn_ReplyPort = DiskPort;
Error = OpenDevice((char *)DEVNAME,Drive,DiskReq,0);

/*  Preleva il Change Num del disco nel drive  */

DiskReq->iotd_Req.io_Command = TD_CHANGENUM;
DoIO(DiskReq);
ChngNum = DiskReq->iotd_Req.io_Actual;
printf("Change Number attuale = %ld\n",ChngNum);

/*  Legge il settore  */

Offset = TD_SECTOR*(Sec+TD_NUMSECS*Hd+TD_NUMSECS*TD_NUMHEADS*Cyl);
DiskReq->iotd_Req.io_Offset = Offset;
DiskReq->iotd_Req.io_Command = (UWORD)ETD_READ;
DiskReq->iotd_Req.io_Data = (APTR)diskbuffer;
DiskReq->iotd_Req.io_Length = TD_SECTOR;
DiskReq->iotd_Count = ChngNum;
DoIO(DiskReq);

/*  Spegne il motore  */

DiskReq->iotd_Req.io_Length = 0;
DiskReq->iotd_Req.io_Command = TD_MOTOR;
DoIO(DiskReq);

/*  Stampa il settore  */

if ((fp = (FILE *)fopen("prt:","w")) == NULL) CleanExit(20);
fprintf(fp,"\nDump    Drive %ld    Cilindro %ld  Settore %ld  Faccia
%ld\n\n",Drive,Cyl,Sec,Hd);
for (y = 0;y < 32;y++) {
  fprintf(fp,"%04X: ",16*y);
  for (x = 0;x < 16;x++) {
    fprintf(fp,"%02X ",(int)diskbuffer[16*y+x]);
  }
  fprintf(fp,"  ");
  for (x = 0;x < 16;x++) {
    Char = (int)diskbuffer[16*y+x];
    if ((Char < SPACE)||(Char > DEL)) Char = (int)'.';
    fprintf(fp,"%c",(char)Char);
  }
  fprintf(fp,"\n");
}
CleanExit(0);
}  /*  Fine !!  */

CleanExit(ret)
  int ret;
{
  if (diskbuffer != NULL) FreeMem(diskbuffer,TD_SECTOR);
  if (DiskPort != NULL) {
    RemPort(DiskPort);

    FreeMem(DiskPort,sizeof(struct MsgPort));
  }
  if (DiskReq != NULL) {
    CloseDevice(DiskReq);
    DiskReq->iotd_Req.io_Message.mn_Node.ln_Type = 0xFF;
    DiskReq->iotd_Req.io_Device = (struct Device *)-1;
    DiskReq->iotd_Req.io_Unit = (struct Unit *)-1;
    FreeMem(DiskReq,sizeof(struct IOExtTD));
  }
  if (fp != NULL) fclose(fp);
  exit(ret);
}
```

un disk editor (DiskEd), vi informo che su MC-Link troverete DiskEd un programma utile — per davvero — con il quale potrete intervenire sui dischetti blocco per blocco, alla ricerca delle tastiere yankee o per stanare qualche virus. Come lo SMARTDisk, con la differenza di essere scritto in Basic (!!).

Le potenzialità che offre sono solite: inserimento di dati (che dovrete scrivere senza spazi, ad es. 4EAEFF40) o di stringhe a partire dalla posizione del cursore (che potrete posizionare col mouse), ricerca di dati o di una stringa sul disco, dump su stampante, ecc. Ogni volta che cambiate disco, selezionate «Cambio disco» dal menu «Progetto». Il settore caricato viene visualizzato in due metà («Byte 0-255» e «Bytes» 256-511). Potete richiedere informazioni al programma sulla natura del blocco selezionando «Info» dal menu Progetto (per questa feature mi sono servito delle preziose informazioni contenute nell'articolo «L'Amiga Filing System» di Luca Ceccatelli, MC 78). Se volete dargli una marcia in più, compilatelo con un compilatore «serio» (non con l'AC-Basic della AbSoft, ben lungi dall'essere perfetto!).

A presto!

ESCConverter

di Maurizio Mangrella - Eboli (SA)

Spesso si ha a che fare con word processor o text processor che non consentono un regolare utilizzo delle sequenze di ESCape e i caratteri CTRL-ati. Ad esempio, MicroEMACS — l'editor che sto usando in questo momento per scrivere questa documentazione — intercetta la pressione del tasto ESC o del tasto CTRL per suoi scopi: non diversamente si comportano — almeno nell'ambito IBM — «mostri sacri» come Word e WordStar.

Chi utilizza programmi come Word Perfect o ProWrite, o anche il semplice NotePad — a proposito, anche questo formatta il testo con complicatissime sequenze di ESCape per la scelta dei font, etc. — solo per effettuare una stampa grafica di ciò che appare in video, ovviamente, non ha problemi; ma per chi, come me, utilizza l'NLQ della stampante per scrivere testi e

possiede una stampante a colori — io ho l'MPS 1500 C della Commodore — le sequenze di ESCape sono pane quotidiano...

Voglio ricordare — e mi limito a questo, altrimenti il discorso si allunga troppo — che i driver per la stampante selezionabili da Preferences si comportano tutti allo stesso modo, riconoscendo una serie standard di ESCape sequence, dettate dallo standard ANSI X3.64: alcune di queste — ad esempio, quelle per cambiare il colore di scrittura — sono piuttosto complesse, ed inserirle attraverso un word processor sarebbe del tutto impossibile.

Per questo motivo ho scritto questo post-processor, il quale, attraverso comandi e direttive molto semplici da inserire nel testo, consente di formattare l'output con il massimo controllo dell'operatore.

I comandi

Tutto il «castello» si fonda sull'uso del carattere \ (backslash) normalmente poco usato nei testi, ed il cui uso è consentito dalla maggior parte — almeno credo — dei word processor.

Cominciamo con le sequenze di ESCape, ad esempio quella per scegliere di stampare in blu (con il driver Epson JX 80, che uso con la mia stampante):

ESC[36m

Nell'ambito del testo basta scrivere

\[36m

e sarà opportunamente trasformata dal post-processor.

I soliti incontentabili penseranno: «Ma come si fa a scrivere una \ ?» È semplice (alla maniera del C): basta scrivere \\ .

Volendo inserire il carattere CTRL - L (^L), che serve per rinfrescare l'attuale finestra di output, basta scrivere la sequenza

\'l oppure \'L

(tre caratteri invece di uno: la comodità si paga...). Inserite SOLO lettere dopo la \ ', altrimenti l'effetto sarà imprevedibile...

Le direttive

Sarebbe meglio dire «LA direttiva», in quanto praticamente è una sola... Battendo nel testo, ad un certo punto,

*—

si disattiva, da quel punto in poi, il controllo da parte del post-processor, cioè in altre parole, da lì fino al coman-

```
' ESC Sequences Converter     By Maurizio Mangrella  1988
' Questo programma può risultare utile a quanti usano
' spesso word processors o text editors che non consen -
' tono l'inserimento immediato di sequenze di ESCape o
' i caratteri CTRL-ati . Con questo post - processors
' A sufficiente scrivere, nell'ambito del testo, le op -
' portune sequenze \ .          c Maurizio Mangrella  1988
PALETTE 0,1,1,1
PALETTE 1,0,0,0
PALETTE 3,.76,.76,.76
CLS
PRINT "ESC Sequences Converter      By Maurizio Mangrella  1988"
LOCATE 4,1 : PRINT "Load File :"
LOCATE 6,1 : PRINT "Save File :"
LINE (91,22)-(572,34),1,bf
LINE (93,23)-(570,33),0,bf
LINE (91,38)-(572,50),1,bf
LINE (93,39)-(570,49),0,bf
FOR k = 0 TO 3
   xb = 136*k
   LINE (xb,66)-(xb+95,85),1,bf
   LINE (xb+2,67)-(xb+93,84),3,bf
NEXT k
LOCATE 10,1 : COLOR 1,3
PRINT PTAB(8)"Print Save";PTAB(144)"Print Load";PTAB(292)"Convert";PTAB(440)"Exi
t"
COLOR 1,0
WHILE 1
     WHILE MOUSE(0) <> 0 : WEND
     s = 0
     WHILE s = 0
          s = MOUSE(0)
          x = MOUSE(1)
          y = MOUSE(2)
     WEND
     IF y > 21 AND y < 35 THEN
          LINE (93,23)-(570,33),0,bf
          LOCATE 4,13 : INPUT "",ld1$
          IF ld1$ = "" THEN LOCATE 4,13 : PRINT ld$ : ELSE ld$ = ld1$
     ELSEIF y > 37 AND y < 51 THEN
          LINE (93,39)-(570,49),0,bf
          LOCATE 6,13 : INPUT "",sv1$
          IF sv1$ = "" THEN LOCATE 6,13 : PRINT sv$ : ELSE sv$ = sv1$
     ELSEIF y > 65 AND y < 86 THEN
          gn = x\136
          xb = 136*gn
          IF gn < 4 AND (x-xb) < 96 THEN
               ON (gn+1) GOSUB PrtSave,PrtLoad,Convert,Ending
          END IF
     END IF
WEND

PrtSave:
     IF sv$ = "" THEN RETURN
     OPEN sv$ FOR INPUT AS 1
     ln = LOF(1)
     st = ln\1500
     res = ln MOD 1500
     IF res THEN st = st+1
     OPEN "prt:" FOR OUTPUT AS 2
     FOR k = 1 TO st
        IF k <> st THEN l$ = INPUT$(1500,1) : ELSE l$ = INPUT$(res,1)
        PRINT #2,l$
     NEXT k
     CLOSE 2 : CLOSE 1
     RETURN


PrtLoad:
     IF ld$ = "" THEN RETURN
     OPEN ld$ FOR INPUT AS 1
     ln = LOF(1)
     st = ln\1500
     res = ln MOD 1500
     IF res THEN st = st+1
     OPEN "prt:" FOR OUTPUT AS 2
     FOR k = 1 TO st
        IF k <> st THEN l$ = INPUT$(1500,1) : ELSE l$ = INPUT$(res,1)
        PRINT #2,l$
     NEXT k
     CLOSE 2 : CLOSE 1
     RETURN

Convert:
     IF ld$ = "" OR sv$ = "" THEN RETURN
     LOCATE 15,1 : PRINT "Converting ..."
     OPEN ld$ FOR INPUT AS 1
     OPEN sv$ FOR OUTPUT AS 2
     ln = LOF(1)
     st = ln\1500
     res = ln MOD 1500
     IF res THEN st = st+1
     act% = 1
     FOR k = 1 TO st
        IF k <> st THEN l$ = INPUT$(1500,1) : ELSE l$ = INPUT$(res,1)
        ps = 1 : bl = 1
        WHILE bl
             bl = INSTR(ps,l$,"\")
             IF bl THEN
                  seq$ = MID$(l$,bl+1,1)
                  IF act% THEN
                       IF seq$ = "\" THEN
                            l$ = LEFT$(l$,bl-1)+MID$(l$,bl+1)
                       ELSEIF seq$ = "*" THEN
                            act1$ = MID$(l$,bl+2,1)
                            IF act1$ = "+" THEN act% = 1 : ELSE act% = 0
                            l$ = LEFT$(l$,bl-1)+MID$(l$,bl+3)
```

```
                    ELSEIF seq$ = "\" THEN
                        c$ = MID$(l$,bl+2,1)
                        MID$(l$,bl,1) = CHR$(ASC(UCASE$(c$))-64)
                        l$ = LEFT$(l$,bl)+MID$(l$,bl+3)
                    ELSE
                        MID$(l$,bl,1) = CHR$(22)
                    END IF
                ELSE
                    IF seq$ = "*" THEN
                        actfl$ = MID$(l$,bl+2,1)
                        IF actfl$ = "+" THEN act% = 1 : ELSE act% = 0
                        l$ = LEFT$(l$,bl-1)+MID$(l$,bl+3)
                    END IF
                END IF
            END IF
        ps = bl+1 : IF seq$ = "*" THEN ps = bl
        WEND
        PRINT #2,l$
    NEXT k
    CLOSE 2 : CLOSE 1
    LINE (0,112)-(111,119),0,bf
    RETURN

Ending:
    CLS
    PALETTE 0,0,0,1
    PALETTE 1,1,1,1
    PALETTE 3,1,.566,0
    SYSTEM
```

Listato del programma ESCConverter.

do *+ (che riabilita il controllo) il postprocessor lascerà il testo intatto, eliminando dal testo solo i *+ o *—.

Può essere utile per inserire, ad esempio, listati in C (linguaggio che fa larghissimo uso della \) o in un altro linguaggio di programmazione, senza essere costretti a dover correggere tutte le righe del programma.

ESC Converter lascia intatte le formattazioni introdotte dal word processor: si richiede solo che il testo sia in formato ASCII, altrimenti non lo si potrebbe stampare tramite il programma (ed inserirlo di nuovo nel wp per stamparlo sarebbe problematico a causa della presenza degli ESC). Talvolta le sequenze \ possono creare qualche problema a livello della disposizione dei caratteri sul video: in questo caso è meglio effettuare qualche prova, o «immaginare» l'output finale.

Il programma

Il programma si presenta con degli «pseudo-gadget», di cui ora vedremo il significato.

Load File e Save File sono i pathname dei file, rispettivamente, da caricare (e da convertire) e da salvare: clickando sui riquadri orizzontali si può inserire il nome opportuno. Premendo RETURN a vuoto si ottiene di nuovo la selezione precedente.

Print Save e Print Load servono a stampare sulla stampante (secondo i «dettami» di Preferences) il file indicato, rispettivamente, con Save File e Load File: può servire per vedere gli effetti della conversione.

Convert — è inutile dirlo — carica il Load File, lo converte e lo salva in Save File: conviene effettuare questa operazione utilizzando il RAM Disk per risparmiare tempo. La ricerca delle sequenze\ è ottimizzata grazie all'uso della funzione INSTR: in pratica il tempo impiegato è direttamente proporzionale al numero di \ presenti nel testo. Il mio programma carica il file di testo a blocchi di 1500 caratteri e li processa. Anche Convert non funziona se uno dei due pathname (o entrambi) non è stato inserito.

Exit serve ad uscire dal programma: questo non ha bisogno di spiegazioni.

```
\[2"zProva di stampa usando ESCConverter   Copyright Maurizio Mangrella  1988

\*-Stampiamo tante barrette : \\\\\\\\\\\\\\\\*+

Stampa in nero, \[35mrosso, \[36mblu, \[32mverde, \[31marancio, \[34mviola
\[30mStampa in NLQ o \[1"zdraft
Stampa normale, \[6wallargata o \[5w\[4wcondensata .
\[2"z\[3wStampa normale, \[6wallargata o \[5w\[4wcondensata .
\[1"z\[3wNormale, \[3mcorsivo, \[23m\[1mgrassetto, \[21m\[4msottolineato .

\[24mUn a\[31mr\[32mc\[33mo\[34mb\[35ma\[36ml\[35me\[32mn\[36mo \[30mdi colori !

\[2vAPICI \[1ve \[4vPEDICI
\[3v
Arrivederci!
```

```
Prova di stampa usando ESCConverter   Copyright Maurizio Mangrella  1988

Stampiamo tante barrette : \\\\\\\\\\\\\\\\

Stampa in nero, rosso, blu, verde, arancio, viola
Stampa in NLQ o draft
Stampa normale, allargata o condensata .
Stampa normale, allargata o condensata .
Normale, corsivo, grassetto, sottolineato .

Un arcobaleno di colori !

APICI e PEDICI

Arrivederci!
```

Demo delle possibilità di ESCConverter.

Variabili utilizzate

l$	Blocco di testo caricato dal Load File
ps	Posizione iniziale per la ricerca di una \
bl	Posizione di una \
seq$	Carattere seguente ad una \
actfl$	Il terzo carattere in una direttiva *+ o *—
act%	Flag: se 0, la ricerca è disabilitata, se 1 è attivata
ld$	Pathname del Load File
sv$	Pathname del Save File
ld1$,sv1$	Stringhe temporanee per il controllo, a livello di INPUT, di un eventuale RETURN a vuoto
length	Lunghezza del file
res	Lunghezza dell'ultimo blocco
st	(Steps) Numero di blocchi da caricare

Le subroutine

PrtLoad:	Stampa il Load File
PrtSave:	Stampa il Save File
Convert:	Converte il Load File e salva il risultato nel Save File
Ending:	Esce dal programma

Il caricamento di un file avviene sezionandolo in blocchi di 1500 caratteri. Nel caso il file sia lungo meno di 1500 byte o la sua lunghezza non sia esattamente divisibile per 1500, l'ultimo blocco (che potrebbe anche essere il primo...) viene caricato con la sua opportuna lunghezza (variabile res) per evitare un «Input past end».

Durante la conversione si fa largo uso della funzione MID$ in funzione di assegnazione, nella forma

```
MID$(1$,bl,1) = CHR$(27)
```

forse un po' contorta, ma certamente utilissima.