Questo mese ritorneremo a parlare di Dos, nella fattispecie delle funzioni Dos «per eccellenza» ovvero quelle che ci permettono di interagire con le unità a dischi. Inutile dirvi che il nostro prode (e affezionatissimo) Mangrella parla sempre e solo in termini di Basic, linguaggio al quale sembra essere davvero tanto affezionato.
Ovviamente non Basic nudo e crudo, ma Basic «espanso» dalla potente possibilità di utilizzare librerie esterne. È solo attraverso queste che si ha, da Basic, il controllo di tutta la macchina

AmigaDOS: le funzioni speciali

di Maurizio Mangrella - Eboli (SA)

Oltre alle classiche funzioni di Open &Close, Read&Write, Seek, etc., l'AmigaDOS offre alcune funzioni speciali. Alcune di esse sono state descritte in dettaglio da Dario de Judicibus sul numero 75 di MC, altre mi appresto a descriverle in accordo con le mie faticose «scoperte».

La directory

Come tutti ormai sapranno, l'Amiga gestisce un filing system gerarchico, in cui, cioè, sono presenti una directory principale (root) e varie subdirectory, le quali contengono file nettamente separati dall'ambiente principale.

Forse non tutti sanno (ma i sassi sì!) che l'Amiga non predispone, sui dischetti, cose come FAT (File Allocation Table) o FCB (File Control Block), o altre informazioni fisse inerenti le directory: ogni file ha un header che ne descrive le caratteristiche e l'allocazione nell'ambito del filing system. Per questo motivo la directory è piuttosta lenta da visualizzare (talvolta anche troppo), ma è protetta da accidentali «casini» — che, evidentemente, devono essere più frequenti di quanto ci si aspetterebbe!

I file direttamente eseguibili aggiungono un altro header, contenente informazioni sulla rilocabilità e sulla organizzazione delle tre sezioni di cui si compone: CODE, DATA e BSS (dati inutilizzati). Un programma utente ha accesso solo all'header dei file eseguibili, ma non a quello «di sistema» che fa le veci della directory: evidentemente per motivi di discurezza. Comunque è possibile accedere interamente al disco, senza limitazioni di sorta, tramite la trackdisk.device e la lettura/scrittura blocco per blocco.

Il file locking

Amiga è un sistema multitasking, e come tale ha bisogno di suddividere le risorse (tra cui il disco) tra i vari processi. Un processo può appropriarsi, in maniera esclusiva o condivisa, di un file o di una directory tramite la funzione Lock. La sintassi è

```
FLock& = Lock& (Name&,Mode&)
```

dove Name& punta (come al solito) al pathname del file o della directory (terminato con un CHR$(0)) e Mode& è il modo di accesso, che può essere −1 (EXCLUSIVE_LOCK o ACCESS_WRITE) o −2 (SHARED_LOCK o ACCESS_READ). Un lock di scrittura, si capisce, è esclusivo, mentre uno per la lettura è condiviso tra i vari processi. Siamo ancora lontani dal record locking della multiutenza, ma ci si può comunque accontentare. FLock& è il puntatore al lock (può essere 0 se l'operazione non è andata bene).

Perché vi ho parlato dei lock? Perché sono necessari quasi per ogni operazione sul disco: nella pratica, prima di operare su un dato file, molto spesso dovrete «lock-arlo».

Altre funzioni del DOS

Prima di parlarvi di DiskGest (il programma che accompagno a questa descrizione) eccovi alcune utili funzioni del DOS (tra parentesi il relativo comando CLI):

CreateDir (Name&) (MakeDir)
Crea una directory con il pathname puntato da Name& (al solito modo). Ritorna un lock di lettura associato alla nuova directory

CurrentDir (Lock&) (CD)
Rende la directory associata al lock

Lock& la Current Directory (non avrete più bisogno di specificare il pathname) e ritorna il lock associato alla Current Directory precedente

DeleteFile (Name&) (Delete)
Cancella il file/directory rispondente al nome Name&

DupLock (Lock&)
Duplica il lock Lock& (solo se e a sola lettura) e ritorna il nuovo lock

UnLock (Lock&)
Rimuove il lock Lock& (qualunque esso sia)

IoErr () (Why, Fault)
Ritorna l'ultimo errore secondo il codice usato anche dal CLI. Vale solo se si è effettivamente verificato un errore.

ParentDir (Lock&) (CD)
Ritorna il lock alla directory «genitrice» del file/directory associato a Lock&

SetComment (Name&,Comm&) (FileNote)
Setta il commento puntato da Comm& stringa terminata con un CHR$(0)) al file con pathname Name&

SetProtection (Name&,Mask&) (Protect)
Protegge un file secondo la maschera Mask& (vedi dopo)

DateStamp (Vect&) (Date)
Restituisce la data corrente nel vettore Vect& di tre long-words (vedi dopo). Ritorna l'indirizzo Vect&.

Informazioni sul disco e su un file

Possiamo chiedere informazioni tramite le routine Info e Examine, passando ad esse un opportuno lock e l'indirizzo di una struttura dati che la routine provvederà gentilmente a riempirci di informazioni utili. Per chiedere informazioni sul disco correntemente presente nel drive 0, possiamo dare:

```
DLock& = Lock& (SADD("DF0:"+CHR$(0)),
-2)
Id& = AllocMem& (36,65537&)
CALL Info (Dlock&, Id&)
```

(AllocMem () è una routine della exec.library). In pratica abbiamo «lock-ato» in lettura il disco nel drive 0, poi abbiamo creato la struct InfoData (figura 1) e abbiamo chiesto informazioni tramite la Info. Con questi tre comandi avremo ottenuto:

```
UniNum& = PEEKL (Id&+4)
Numero del drive
DState& = PEEKL (Id&+8)
Stato del disco
NBlock& = PEEKL (Id&+12)
Numero di blocchi
NBlUsd& = PEEKL (Id&+16)
Numero di blocchi utilizzati
BytBlk& = PEEKL (Id&+20)
Byte per blocco
DType$ = MKI& (PEEKL (Id&+24))
Tipo di disco
```

Il tipo di disco è una stringa di 3 o 4 caratteri (se sono tre, il quarto è CHR$(0)) che può essere DOS, NDOS (Non DOS), KICK (KickStart) o BAD (disco non funzionante).

In genere, comunque, Info si rifiuta di funzionare se il disco è NDOS o BAD.

DState& può valere 80 (WRITE_PROTECTED), 81 (VALIDATING, se il processo di validating è ancora in corso) o 82 (VALIDATED, se il disco è «buono» e non è protetto antiscrittura). Le altre informazioni si spiegano da sole.

Per avere informazioni su un file o una directory di nome Name$, daremo:

```
FLock& = Lock& (SADD(Name$+CHR$(0)),
-2)
Fib& = AllocMem& (260,65537&)
CALL Examine (FLock&,Fib&)
```

e otterremo:

Listato del programma DiskGest

```
'  DiskGest          By Maurizio Mangrella  1988
'  Questo programma consente di manipolare a pia -
' cimento (o quasi ...) i files di un dischetto .
'  Il programma fa uso delle routines del DOS, an -
' che quando un comando BASIC sarebbe stato equiva -
' lente .
'                c by Maurizio Mangrella  1988
SCREEN 1,640,256,3,2
WINDOW 1,"DiskGest          By Maurizio Mangrella  1988",,0,1
CLS : COLOR 1,0
PALETTE 0,1,1,1
PALETTE 1,0,0,0
PALETTE 2,.8,.4,.4
PALETTE 3,.2,.7,.1
PALETTE 4,.7,.7,.7
DECLARE FUNCTION Rename& LIBRARY
DECLARE FUNCTION DeleteFile& LIBRARY
DECLARE FUNCTION SetComment& LIBRARY
DECLARE FUNCTION SetProtection& LIBRARY
DECLARE FUNCTION IoErr& LIBRARY
DECLARE FUNCTION Lock& LIBRARY
DECLARE FUNCTION AllocMem& LIBRARY
LIBRARY "dos.library"
LIBRARY "exec.library"
DIM FileName$(200),Comment$(200),Func$(6),ProtFlag$(3)
DATA Rename,Delete,Protect,Comment,Info,Expand
DATA Read,Write,Execute,Delete
RESTORE
FOR k = 1 TO 6 : READ Func$(k) : NEXT k
FOR k = 0 TO 3 : READ ProtFlag$(k) : NEXT k
Id& = AllocMem&(36,65537&)
Fib& = AllocMem&(260,65537&)
PRINT "DiskGest   Gestione dei dischi        c by Maurizio Mangrella  1
988"
PRINT :PRINT "  Il programma consente di gestire i files e le directory
 di un disco"
PRINT "molto più facilmente che da CLI ."
LOCATE 7,1
PRINT "Inserire un disco nel drive 0 e premere un tasto"
GOSUB GetKey
Path$ = "DF0:"
Unit$ = Path$+CHR$(0)
ULck& = Lock&(SADD(Unit$),-2)
CALL Info(ULck&,Id&)
DState& = PEEKL(Id&+8)
WHILE DState& = -1 OR DState& = 81
     DState& = PEEKL(Id&+8)
     CALL Info(ULck&,Id&)
WEND
CALL Examine(ULck&,Fib&)
DName$ = "" : CALL GetName(Fib&,8,DName$)
Dtype$ = "" : CALL GetName(Id&,24,Dtype$)
NBlocks& = PEEKL(Id&+12)
NBlUsed& = PEEKL(Id&+16)
CLS
PRINT "Nome del disco  ";DName$
PRINT "Tipo  ";Dtype$;"    N° blocchi ";NBlocks&;";";NBlUsed&"usati
";
IF DState& = 80 THEN PRINT "- Write Protected";
PRINT
GOSUB VisDir
COLOR 1,4
LINE (0,136)-(607,143),4,bf : LINE (0,152)-(607,159),4,bf
LOCATE 22,1
FOR k = 1 TO 6
   boff% = 58+84*(k-1)
   LINE (boff%-4,166)-(boff%+67,177),1,b
   LINE (boff%-3,166)-(boff%+66,177),1,b
   toff% = boff%+4*(8-LEN(Func$(k)))
   PRINT PTAB(boff%)"          ";PTAB(toff%)Func$(k);
NEXT k
PRINT
```

```
struct InfoData {
    LONG    id_NumSoftErrors;
    LONG    id_UnitNumber;
    LONG    id_DiskState;
    LONG    id_NumBlocks;
    LONG    id_NumBlocksUsed;
    LONG    id_BytesPerBlock;
    LONG    id_DiskType;
    BPTR    id_VolumeNode;
    LONG    id_InUse;
};
```

Figura 1 - La InfoData Structure.

```
struct FileInfoBlock {
    LONG    fib_DiskKey;
    LONG    fib_DirEntryType;
    char    fib_FileName[108];
    LONG    fib_Protection;
    LONG    fib_EntryType;
    LONG    fib_Size;
    LONG    fib_NumBlocks;
    struct DateStamp fib_Date;
    char    fib_Comment[116];
};
```

Figura 2 - La FileInfoBlock Structure.

```
COLOR 1,4
LOCATE 18,1 : PRINT LEFT$(Path$,72)
WHILE 1
     GOSUB GetMouse
     IF x > 7 AND x < 20 THEN
          IF y > 39 AND y < 48 AND FilOff% > 0 THEN
               FilOff% = FilOff%-1
               COLOR 1,0 : SCROLL (24,40)-(598,119),0,8
               LOCATE 6,4
               EnType$ = LEFT$(FileName$(FilOff%+1),1)
               IF EnType$ = "D" THEN COLOR 2,0 : ELSE COLOR 1,0
               PRINT MID$(FileName$(FilOff%+1),2,72)
          ELSEIF y > 111 AND y < 120 AND FilOff% < (File%-10) THEN
               FilOff% = FilOff%+1
               COLOR 1,0 : SCROLL (24,40)-(598,119),0,-8
               LOCATE 15,4
               EnType$ = LEFT$(FileName$(FilOff%+10),1)
               IF EnType$ = "D" THEN COLOR 2,0 : ELSE COLOR 1,0
               PRINT MID$(FileName$(FilOff%+10),2,72)
          END IF
     ELSEIF y > 23 AND y < 32 THEN
          IF Path$ <> "DF0:" THEN
               IF INSTR(Path$,"/") = 0 THEN
                    Path$ = "DF0:"
               ELSE
                    spos% = LEN(Path$)  ' Cerca una / partendo dalla fi
ne
                    WHILE MID$(Path$,spos%,1) <> "/"
                         spos% = spos%-1
                    WEND
                    Path$ = LEFT$(Path$,spos%-1)
               END IF
               COLOR 1,4
               LOCATE 18,1 : PRINT Path$+SPACE$(76-LEN(Path$))
               GOSUB VisDir
          END IF
     ELSEIF y > 39 AND y < 120 THEN
          FileNum% = FilOff%+((y-40)\8)+1
          IF FileNum% <= File% THEN
               FileName$ = MID$(FileName$(FileNum%),2)
               IF Path$ <> "DF0:" THEN FileName$ = "/"+FileName$
               FileName$ = Path$+FileName$
               COLOR 1,4
               LOCATE 18,1 : PRINT FileName$+SPACE$(76-LEN(FileName$))
          END IF
     ELSEIF y > 167 AND y < 177 AND FileName$ <> "" THEN
          Comm% = ((x-50)\84)+1
          DosErr& = 0
          ON Comm% GOSUB Rename,Deletes,Protect,Comment,Info,Expand
     END IF
WEND
' Comandi
Rename:
     CALL InputName(FileName$,s$)
     FunErr& = Rename&(SADD(FileName$+CHR$(0)),SADD(s$+CHR$(0)))
     IF FunErr& = 0 THEN GOSUB ErrorHandle
     IF DosErr& = 0 THEN GOSUB VisDir
     RETURN
Deletes:
     COLOR 1,0
     LOCATE 24,1 : PRINT "Sei sicuro ? <S/N>"
     GOSUB GetKey : WHILE a$ <> "S" AND a$ <> "N" : GOSUB GetKey : WEND
     IF a$ = "S" THEN
          FunErr& = DeleteFile&(SADD(FileName$+CHR$(0)))
          IF FunErr& = 0 THEN GOSUB ErrorHandle
          IF DosErr& = 0 THEN GOSUB OmitFile
     END IF
     LINE (0,184)-(607,191),0,bf
     RETURN
Protect:
     COLOR 1,4
     LOCATE 20,1 : PRINT "Protezioni :";
     FLck& = Lock&(SADD(FileName$+CHR$(0)),-2)
     CALL Examine(FLck&,Fib&)
     CALL UnLock(FLck&)
```

(continua a pag. 262)

PMask& = PEEKL (Fib&+116)
Maschera di protezione
Entry& = PEEKL (Fib&+120)
Tipo (file/directory)
BySiz& = PEEKL (Fib&+124)
Lunghezza in byte (solo per i file)
BlSiz& = PEEKL (Fib&+128)
Lunghezza in blocchi (solo per i file)

A Fib&+8 troverete il nome del file/directory (max. 108 caratteri terminati da un CHR$(0)); a Fib&+144 il commento (max. 116 caratteri terminati dal solito CHR$(0)). A Fib&+132 troveremo la data del file (nel formato interno), che si compone di tre long-word:

Days& = PEEKL (Fib&+132)
Giorni trascorsi (da quando?)
Mins& = PEEKL (Fib&+136)
Minuti trascorsi dalla mezzanotte
Tick& = PEEKL (Fib&+140)
Cinquantesimi di secondo (ticks) nel minuto

Entry& è positivo se Name& è una directory, negativo se è un file.

La maschera di protezione è così organizzata:

— Bit 3 : READ
— Bit 2 : WRITE
— Bit 1 : EXECUTE
— Bit 0 : DELETE (l'unico riconosciuto attualmente)

Se uno di questi bit è settato la relativa operazione non è consentita. Il bit 4 (ARCHIVE) viene settato quando si apre un file in scrittura e resettato nel momento in cui esso viene regolarmente chiuso: può essere utile per correggere eventuali errori di «shut-down». Gli altri bit sono riservati.

La directory

Per ottenere l'elenco dei file presenti in una directory è sufficiente lockarla in lettura, esaminarla con Examine e, poi, chiamare ripetutamente la routine ExNext. Mi spiego:

DLock& = Lock& (SADD(Name$+CHR$(0)),−2)
CALL Examine (FLock&,Fib$)
Succ& = ExNext& (FLock&,Fib&)

Fino a quando non finiamo i file.

ExNext () riempie il FileInfoBlock (figura 2) esaminando le varie «entries» di una directory. Succ& vale −1 se tutto è

andato bene, 0 altrimenti (in questo caso è bene chiamare IoErr& () : se l'errore ritornato è 232, ovvero ER- ROR_NO-_MORE_ENTRIES, i file sono finiti). A proposito, ricordo solo che le routine DeleteFile, Examine, ExNext, Info, Rename, SetComment e SetProtection ritornano un valore di tipo BOOL (−1 (TRUE) riuscito, 0 (FALSE) non riuscito).

DiskGest

Il programma che vi presento fa uso di quanto esposto finora (anche quando non sarebbe strettamente necessario, vista la presenza di alcuni comandi Basic adatti allo scopo) per esaminare ,e modificare il contenuto di un disco.

All'inizio vi chiede di inserire un dischetto e di premere un tasto: a questo punto mostra le caratteristiche generali del disco e, dopo un po' di lavorio del drive, la directory. Potrete scorrere in essa clickando sulle freccette a sinistra dei nomi dei file, oppure sceglierne uno clickando sul suo nome. Se è una directory (in rosso) potrete visualizzarne il contenuto (clickando su Expand). I normali file sono scritti in nero.

Altre operazioni che potete svolgere sono: cambio di nome (Rename), cancellazione (Delete), protezione (Protect), commento (Comment), richiesta di informazioni (Info). Per sceglierne una, basta clickarci sopra. Con Rename potrete inserire il nuovo nome (facendo uso dei tasti e del BackSpace) e sistemarlo con un bel Return (dopo questa operazione il programma carica di nuovo la directory). Con Delete vi verrà richiesta conferma (Sei sicuro?), cui potrete rispondere con i tasti S e N. Con Protect compariranno le protezioni (bianco : attivata ; nero : non attivata) : clickandoci sopra potrete cambiarne lo stato. Per proteggere un file basta clickare di nuovo su Protect. Con Comment, come con Rename, potrete inserire il commento (a partire da quello eventualmente già presente) e sistemarlo con un Return. Qualunque problema verrà visualizzato sotto forma di «Operazione non effettuata — Errore xxx» dove xxx è il codice dell'errore.

Ultima nota: in qualunque meandro del vostro disk vi siate sperduti, potete tornare al livello immediatamente superiore clickando su «Parent Dir».

Note tecniche

Il programma è in Basic, e fa uso delle seguenti variabili:

(segue da pag. 261)

```
      Prot& = PEEKL(Fib&+116) AND 15
      FOR k = 0 TO 3
         toff% = 104+84*k
         ProtF& = Prot& AND (2^k)
         IF ProtF& THEN COLOR 0,4 : ELSE COLOR 1,4
         PRINT PTAB(toff%)ProtFlag$(k);
      NEXT k
      WHILE MOUSE(0) <> 0 : WEND
      x = 0 : y = 0
      WHILE y < 168 OR y > 175
           GOSUB GetMouse
           IF x > 103 AND x < 440 AND y > 151 AND y < 161 THEN
                k = (x-104)\84 : toff% = 104+84*k
                Prot& = Prot& XOR (2^k)
                ProtF& = Prot& AND (2^k)
                IF ProtF& THEN COLOR 0,4 : ELSE COLOR 1,4
                PRINT PTAB(toff%)ProtFlag$(k);
           END IF
      WEND
      PRINT : LINE (0,152)-(607,159),4,bf
      FunErr& = SetProtection&(SADD(FileName$+CHR$(0)),Prot&)
      IF FunErr& = 0 THEN GOSUB ErrorHandle
      RETURN
Comment:
      CALL InputName(Comment$(FileNum%),s$)
      Comment$(FileNum%) = s$
      FunErr& = SetComment&(SADD(FileName$+CHR$(0)),SADD(s$+CHR$(0)))
      IF FunErr& = 0 THEN GOSUB ErrorHandle
      RETURN
Info:
      FLck& = Lock&(SADD(FileName$+CHR$(0)),-2)
      CALL Examine(FLck&,Fib&)
      CALL UnLock(FLck&)
      SizByt& = PEEKL(Fib&+124)
      SizBlk& = PEEKL(Fib&+128)
      Prot& = PEEKL(Fib&+116)
      EnType$ = LEFT$(FileName$(FileNum%),1)
      COLOR 1,0
      LOCATE 24,1
      PRINT FileName$ : PRINT
      IF EnType$ = "D" THEN
           PRINT "Directory";
      ELSE
           PRINT "File  ";SizByt&;"bytes ";SizBlk&;
           IF SizBlk& = 1 THEN PRINT "blocco"; : ELSE PRINT "blocchi";
      END IF
      PRINT "   Protezioni : ";
      FOR k = 0 TO 3
         ProtF& = Prot& AND (2^k)
         IF ProtF& THEN PRINT LEFT$(ProtFlag$(k),1); : ELSE PRINT "-";
      NEXT k
      PRINT :PRINT
      IF LEN(Comment$(FileNum%)) THEN
           PRINT "; ";Comment$(FileNum%)
      ELSE
           PRINT "Nessun commento"
      END IF
      GOSUB GetKey
      LINE (0,184)-(607,223),0,bf
      RETURN
Expand:
      EnType$ = LEFT$(FileName$(FileNum%),1)
      IF EnType$ = "D" THEN Path$ = FileName$ : GOSUB VisDir
      RETURN
' Funzioni di uso generale
VisDir:
      CALL GetDir(Path$,File%)
      LINE (3,20)-(600,123),0,bf
      LINE (3,20)-(600,123),1,b
      LINE (4,20)-(599,123),1,b
      COLOR 2,0
      AREA (8,47) : AREA (12,40) : AREA (15,47) : AREAFILL
      AREA (8,112) : AREA (12,119) : AREA (15,112) : AREAFILL
      NumDisp% = File% : IF NumDisp% > 10 THEN NumDisp% = 10
      FilOff% = 0
      LOCATE 6,1
      FOR k = 1 TO NumDisp%
         EnType$ = LEFT$(FileName$(k),1)
         IF EnType$ = "D" THEN COLOR 2,0 : ELSE COLOR 1,0
         PRINT TAB(4)MID$(FileName$(k),2,72)
      NEXT k
      COLOR 3,0 : LOCATE 4,4 : PRINT "Parent Dir"
      RETURN
ErrorHandle:
      COLOR 1,4
      LOCATE 20,1 : PRINT SPACE$(76)
      FOR k = 1 TO 1000 : NEXT k
      DosErr& = IoErr&(0)
      LINE (0,184)-(607,191),0,bf
      IF DosErr& THEN
           COLOR 1,0
           LOCATE 24,1 : PRINT "Operazione non effettuata - Errore";DosE
rr&
           FOR k = 1 TO 2500 : NEXT k
      END IF
      RETURN
GetKey:
```

```
      a$ = ""
      WHILE a$ = "" : a$ = UCASE$(INKEY$) : WEND
      RETURN
GetMouse:
      ms = 0
      WHILE MOUSE(0) <> 0 : WEND
      WHILE ms = 0
            ms = MOUSE(0)
            x = MOUSE(1)
            y = MOUSE(2)
      WEND
      RETURN
OmitFile:
      FOR k = FileNum% TO File%-1
         FileName$(k) = FileName$(k+1)
         Comment$(k) = Comment$(k+1)
      NEXT k
      File% = File%-1
      LINE (24,40)-(598,119),0,bf
      IF FilOff% = (File%-9) AND File% > 10 THEN FilOff% = FilOff%-1
      NumDisp% = FilOff%+10
      IF NumDisp% > File% THEN NumDisp% = File%
      LOCATE 6,1
      FOR k = FilOff%+1 TO NumDisp%
         EnType$ = LEFT$(FileName$(k),1)
         IF EnType$ = "D" THEN COLOR 2,0 : ELSE COLOR 1,0
         PRINT TAB(4)MID$(FileName$(k),2,72)
      NEXT k
      RETURN
SUB GetName(Addr&,Offset%,Var$) STATIC
      Var$ = ""
      ch% = PEEK(Addr&+Offset%)
      WHILE ch%
            Var$ = Var$+CHR$(ch%)
            Offset% = Offset%+1 : ch% = PEEK(Addr&+Offset%)
      WEND
      END SUB
SUB GetDir(PathName$,File%) STATIC
      SHARED Fib&,DLck&,FileName$(),Comment$()
      IF DLck& THEN CALL UnLock(DLck&)
      DLck& = Lock&(SADD(PathName$+CHR$(0)),-2)
      CALL Examine(DLck&,Fib&)
      File% = 0
      DosErr& = 0
      WHILE DosErr& = 0 AND File% < 200
            File% = File%+1
            CALL ExNext(DLck&,Fib&)
            DosErr& = IoErr&(0)
            IF DosErr& = 0 THEN
                  CALL GetName(Fib&,8,FileName$(File%))
                  CALL GetName(Fib&,144,Comment$(File%))
                  EnType& = PEEKL(Fib&+120)
                  IF EnType& > 0 THEN
                        FileName$(File%) = "D"+FileName$(File%)
                  ELSE
                        FileName$(File%) = "F"+FileName$(File%)
                  END IF
            END IF
      WEND
      File% = File%-1
      END SUB
SUB InputName(Var1$,Var2$) STATIC
      Var2$ = LEFT$(Var1$,75)
      COLOR 1,4
      LOCATE 20,1 : PRINT SPACE$(76)
      LOCATE 20,1 : PRINT Var2$+"_";
      a$ = ""
      WHILE a$ <> CHR$(13) AND LEN(Var2$) < 75
            WHILE INKEY$ <> "" : WEND
            a$ = "" : WHILE a$ = "" : a$ = INKEY$ : WEND
            IF a$ = CHR$(8) THEN
                  IF LEN(Var2$) THEN
                        PRINT CHR$(8)+CHR$(8)+"_";
                        Var2$ = LEFT$(Var2$,LEN(Var2$)-1)
                  END IF
            ELSE
                  PRINT CHR$(8);
                  IF a$ <> CHR$(13) THEN
                        Var2$ = Var2$+a$
                        PRINT a$+"_";
                  END IF
            END IF
      WEND
      LINE (0,152)-(607,159),4,bf
      END SUB
```

FileName$()
Nomi dei file in una directory (fino a 200)
Comment$()
Commenti ai file (fino a 200)
ProtFlag$()
Nomi dei flag di protezione
File%
Numero di file trovati
FileNum%
Numero del file correntemente selezionato
FileName$
Nome completo del file selezionato
Path$
Pathname corrente
FilOff%
Offset per FileNum% (per lo scorrimento della directory)
Prot&, ProtF&
Maschere di protezione
BytSiz&, BlkSiz&
Grandezza del file (in byte e blocchi)
EnType$
Tipo di entry ("F" = File, "D" = Directory)
DState&
Stato del disco
DType&
Tipo del disco (DOS, NDOS, KICK, BAD)
NBlocks&
Numero di blocchi
NBlUsed&
Numero di blocchi utilizzati
Fib&
Indirizzo del FileInfoBlock
Id&
Indirizzo dell'InfoData
k, s$, ch%
Variabili di uso generale

In ogni item dell'array FileName$ (), il primo carattere rappresenta il tipo dell'entry (come per EnType$), mentre i restanti indicano il suo nome.

Subroutine e subprogram:

GetDir (Path$,fIle%)

Carica la directory Path$ e pone il numero di file trovati in File%

GetName (Addr&,Var$,off%)

Carica una stringa nella variabile Var$ a partire dall'indirizzo Addr&+off%

InputName (Var1$,Var2$)

Inserisce da tastiera una stringa in Var2$ partendo dal contenuto di Var1$

Rename:	Comando Rename
Deletes:	Comando Delete
Protect:	Comando Protect
Comment:	Comando Comment
Info:	Comando Info
Expand:	Comando Expand
VisDir:	Carica e visualizza una directory
ErrorHandle:	Visualizza un errore del DOS se si è verificato
OmitFile:	Cancella un file dalla directory
GetKey:	Attende la pressione di un tasto
GetMouse:	Preleva le coordinate del mouse