a cura di Valter Di Dio

software

MS-DOS

Il programma MELOLE permette di giocare contro un avversario che apprende dai propri errori. Il trucco consiste nel costruire un albero delle scelte via via eseguite durante una partita e cominciare a scartare l'ultima mossa che ha determinato una sconfitta. L'albero si assottiglierà ad ogni partita, lasciando alla fine solo mosse vincenti. Perché ciò sia sempre vero è necessario che il gioco rispetti delle regole abbastanza semplici in modo che l'albero delle scelte sia binario e non troppo grande. Pensare di applicare un simile procedimento al gioco degli scacchi, per esempio, pur essendo valido dal punto di vista teorico (l'albero degli scacchi può essere assottigliato delle mosse «perdenti») è assolutamente impossibile in pratica: l'albero completo degli scacchi è la cosa più vicina ad infinito che riesco ad immaginare!

Il secondo programma permette una cosa ufficialmente vietata: la chiamata di procedure figlie da un programma in Turbo Pascal. In realtà la colpa è del DOS che non è sempre rientrante, ma la maggior parte delle volte lo è (esiste pure un flag nella parte bassa della memoria che ci fa sapere quando è possibile rientrare nel DOS). Poter chiamare processi figli da un programma principale permette di creare delle procedure di lancio che accettano anche delle risposte da tastiera (cosa impossibile con i normali file BATCH). La procedura di lancio, ad esempio, chiede quale programma vogliamo eseguire (Wordstar, DB3, Lotus ecc.), chiama la procedura scelta e, alla fine di questa, riprende il controllo, impedendo all'utente di vedere il sistema operativo.

Melole

di Gianluca Amato - Siracusa

Cos'è Melole

Salve. Sono un ragazzo di 13 anni che ha appena finito la scuola dell'obbligo e si è iscritto all'Istituto Tecnico Industriale. Leggendo vari libri di informatica ho trovato molto interessante la macchinetta denominata Melole, che in pratica non è altro che un congegno in grado di imparare dai propri errori.

Nell'idea originale, attribuita a D. Michie, questa si basa su una rappresentazione di un modello concreto attraverso delle scatolette di fiammiferi, da cui il nome della macchina (Matchbox Educable Last One Loses Engine). Su ogni scatola di fiammiferi che rappresenta uno stato del modello in esame, supponiamo il gioco del pari, si mettono tanti fiammiferi quante sono le mosse possibili per quello stato. In questo modo si avrà una corrispondenza biunivoca fra ogni fase del gioco e ogni scatola di fiammiferi e fra ogni mossa e ogni fiammifero nella scatola.

Durante il gioco, nel turno di Melole si estrae casualmente un fiammifero e lo si mette fuori dalla scatola, eseguendo nel contempo la mossa corrispondente. Alla fine del gioco se vince Melole i fiammiferi si rimetteranno tutti dentro le proprie scatole, altrimenti si metterà da parte l'ultimo fiammifero preso, che in pratica rappresenta la mossa perdente. In questo modo abbiamo insegnato a Melole che se si dovesse trovare nella stessa situazione di prima, non deve più eseguire quella mossa. Se tutti i fiammiferi di una scatola vengono posti da parte allora occorrerà eliminare anche il fiammifero che ha portato a quella scatola.

All'inizio Melole giocherà in modo completamente casuale e quindi sarà molto facile da battere ma, perdendo ripetutamente, affinerà sempre più il suo gioco fino a quando diventerà imbattibile.

Melole e il gioco del pari

A questo punto ho pensato: «Perché non fare una versione computerizzata di Melole in modo da rendere tutti i processi automatici?». Mi risposi subito di sì e, occorrendomi un modello sul quale far lavorare la macchinetta, scelsi il gioco del pari.

La mia scelta fu dovuta principalmente al fatto che questo gioco, pur presentando regole molto semplici, non è affatto banale poiché non si basa su una rete combinatoria ma sequenziale (vedi articoli di Andrea de Prisco). Infatti è molto difficile trovare manualmente una strategia ottimale per questo gioco ma è molto facile per una macchina come Melole. Basta farla giocare diverse volte contro di noi

È disponibile presso la redazione, il disco con i programmi pubblicati in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 249.

1 Melole

```
program melole (input,output);

{ Programma MELOLE di Amato Gianluca, Siracusa. }

{ Questo programma simula il comportamento della macchinetta
  MELOLE ed è un tipico esempio di Intelligenza Artificiale.
  In questa versione esso serve a giocare al pari ma con
  opportune modifiche puo assolvere molti altri ruoli. }


  const max=11;
        maxt=6;

  type rectype=record
                 orig:integer;
                 ocol:integer;
                 omos:integer;
               end;
       arrtype=array[1..max,0..1,1..3] of boolean;

  var mov:arrtype;
      stato,pedine,cont,mieped,scel,tupped:integer;
      numpart,numvint:real;
      basta,visual:boolean;
      np:char;
      oldmov:array [1..maxt] of rectype;
      mrig,mcol,mmos:integer;

  procedure inizializza;
    var x,y:integer;
   begin
      numpart:=0; numvint:=0; mrig:=0; mcol:=0; mmos:=0; basta:=false;
      for x:=3 to max do
       for y:=1 to 3 do
          begin
            mov [x,0,y]:=true;  mov [x,1,y]:=true;
          end;
      for y:=0 to 1 do
        begin
          mov [1,y,1]:=true; mov [1,y,2]:=false; mov [1,y,3]:=false;
          mov [2,y,1]:=true; mov [2,y,2]:=true; mov [2,y,3]:=false;
        end;
    end;

  procedure computer;
    var mossa,x:integer;
        esci:boolean;
    begin
      visual:=false;
      if pedine=0 then exit;
      esci:=false;
      for x:=1 to 3 do
        if mov [pedine,stato,x]=true
          then esci:=true;
      if esci then
      begin
        repeat
          mossa:=trunc (random*3+1);
        until mov [pedine,stato,mossa]=true;
        oldmov [cont].orig:=pedine; oldmov [cont].ocol:=stato;
        oldmov [cont].omos:=mossa; cont:=cont+1;
      end
       else mossa:=1;
      write ('La mia mossa è  ',mossa);
      mieped:=mieped+mossa; stato:=mieped mod 2;
      visual:=true; pedine:=pedine-mossa;
    end;
```

3

```
      writeln; write ('Partite giocate ',numpart:3:0);
      write ('    Partite che hai vinto ',numvint:3:0);
      if numpart>0 then
        writeln ('    Percentuale vittorie ',round((numvint/numpart)*100),'%')
        else writeln;
      writeln; writeln ('Premi un tasto per continuare');
      repeat until keypressed;
      writeln; textcolor (Green);
    end;

  procedure partita;
  var primo,modif,conc:boolean;
      domanda:char;
  begin
    clrscr; primo:=false; modif:=false; conc:=false;
    textcolor (Red); writeln ('Programma MELOLE - Gioco del pari');
    textcolor (Green); writeln; writeln ('Regole di gioco:');
    writeln ('Ci sono in campo inizialmente ',max,' pedine.');
    writeln ('Il gioco consiste nell''estrarre tutte le pedine dal campo');
    writeln ('con mosse di 1, 2 o 3 pedine.');
    writeln ('Alla fine vincerà chi avrà un numero pari di pedine.');
    writeln; write ('Giochi per primo ? '); readln (domanda);
    if upcase(domanda)='S' then primo:=true;
    write ('Modifica utomatica strategia ? '); readln (domanda);
    if upcase(domanda)='S' then modif:=true;
    write ('Computer contro computer ? '); readln (domanda);
    if upcase(domanda)='S' then conc:=true;
    repeat
      clrscr; randomize; pedine:=trunc(random);
      pedine:=max; cont:=1; stato:=0;  mieped:=0; tueped:=0; writeln;
      mrig:=0; mcol:=0; mmos:=0;
      textcolor (Red); writeln; writeln ('Programma MELOLE - Gioco del pari');
      textcolor (Green); writeln;
      while pedine<>0 do
      begin
        if primo
          then giocatore (conc)
          else computer;
        writeln ('    Pedine in campo ',pedine,'     Pedine tue ',tueped);
        if primo
          then computer
          else giocatore (conc);
        if visual then
          writeln ('    Pedine in campo ',pedine,'     Pedine tue ',tueped);
      end;
      fine (modif); writeln;
      repeat
        textcolor (Red);
        write ('Nuova partita ? S)i, N)o, L)ista '); readln (np);
        textcolor (Green);
        np:=upcase(np);
        if np='L' then lista;
      until (np='N') or (np='S');
    until np='N';
  end;

procedure salva;
  var nomefile:string[8];
      movfile:file of arrtype;
      x,y:integer;
  begin
    clrscr; textcolor (Red); writeln ('Programma MELOLE - Salvataggio');
    textcolor (Green); writeln;
    write ('Immettere il nome del file strategia: '); readln (nomefile);
    assign (movfile,nomefile+'.str'); rewrite (movfile);
    write (movfile,mov);
    close (movfile);
    writeln; writeln('Premi un tasto per continuare');
```

2

```
procedure giocatore (conc:boolean);
  var mossa:integer;
  begin
    visual:=false;
    if pedine=0 then exit;
    repeat
      writeln;
      if conc
        then mossa:=trunc (random*3+1)
        else begin
          write ('Dual''è la tua ? ');
          read (mossa);
            end;
      if pedine-mossa<0
        then mossa:=0;
    until (mossa>0) and (mossa<4);
    if conc then write ('La tua mossa è  ',mossa);
    pedine:=pedine-mossa; visual:=true; tueped:=tueped+mossa;
  end;

procedure fine (modif:boolean);
  var x:integer;
      esci:boolean;
  begin
    cont:=cont-1; numpart:=numpart+1;
    if stato=0
      then writeln ('Ho vinto io')
      else
      begin
        repeat
          esci:=false;
          mrig:=oldmov [cont].orig; mcol:=oldmov [cont].ocol;
          mmos:=oldmov [cont].omos;
          if modif then
            mov [mrig,mcol,mmos]:=false;
          cont:=cont-1;
          for x:=1 to 3 do
            if mov [mrig,mcol,x]=true
              then esci:=true;
        until esci;
        writeln; textcolor (Red);
        writeln ('Hai vinto tu, ma mi vendicherò'); textcolor (Green);
        numvint:=numvint+1;
      end;
  end;

procedure lista;
  var x,y,z:integer;
  begin
    clrscr; writeln; textcolor (Red);
    writeln ('Lista regole e percentuali vittorie');
    textcolor (Green); writeln;
    writeln ('I---colonna= numero pedine in campo (1-11)');
    writeln ('II--colonna= pedine pari (0) o dispari (1)');
    writeln ('III-colonna= mossa da effettuare (1-3)'); writeln;
    for x:=1 to max do
      begin
        for y:=0 to 1 do
          for z:=1 to 3 do
            begin
              if (x=mrig) and (y=mcol) and (z=mmos) then
                textcolor (Red);
              write (x:2,y,z,'-',mov [x,y,z]:5,'  ');
              textcolor (Green);
            end;
        writeln;
      end;
    textcolor (Red);
```

4

```
    repeat until keypressed;
  end;

procedure carica;
  var nomefile:string[8];
      movfile:file of arrtype;
      x,y:integer;
  begin
    clrscr; textcolor (Red); writeln ('Programma MELOLE - Caricamento');
    textcolor (Green); writeln;
    write ('Immettere il nome del file strategia: '); readln (nomefile);
    assign (movfile,nomefile+'.str'); reset (movfile);
    read (movfile,mov);
    close (movfile);
    writeln; writeln('Premi un tasto per continuare');
    repeat until keypressed;
  end;
  procedure immetti;
    var x,y,z,xpos,ypos:integer;
        appoggio:char;
    begin
      clrscr; textcolor (Red);
      writeln ('Programma MELOLE - Immissione strategia');
      textcolor (Green); writeln;
      writeln ('I---colonna= numero pedine in campo (1-11)');
      writeln ('II--colonna= pedine pari (0) o dispari (1)');
      writeln ('III-colonna= mossa da effettuare (1-3)'); writeln;
      textcolor (Red); writeln ('Premi F o T per FALSE e TRUE');
      textcolor (Green); writeln;
      for x:=1 to max do
         begin
           for y:=0 to 1 do
             for z:=1 to 3 do
               begin
                 xpos:=wherex; ypos:=wherey;
                 repeat
                   gotoxy (xpos,ypos); write (x:2,y,z,'-');
                   read (kbd,appoggio); appoggio:=upcase(appoggio);
                 until (appoggio='T') or (appoggio='F');
                 if appoggio='T'
                    then mov [x,y,z]:=true
                    else mov [x,y,z]:=false;
                 write (mov[x,y,z]:5); write ('  ');
               end;
           writeln;
         end;
    end;

  begin
   inizializza;
   repeat
     clrscr; writeln;
     textcolor (Red); writeln ('Programma MELOLE');
     writeln ('di Amato Gianluca'); textcolor (Green);
     writeln; writeln ('1) Caricare strategia');
     writeln ('2) Salvare strategia'); writeln ('3) Iniziare partita');
     writeln ('4) Annullare strategia'); writeln ('5) Lista strategia');
     writeln ('6) Immetti strategia'); writeln ('7) Fine'); writeln;
     repeat
       write ('Quale scegli ? '); readln (scel);
     until (scel>0) and (scel<8);
     case scel of
       1: carica;
       2: salva;
       3: partita;
       4: inizializza;
       5: lista;
       6: immetti;
       7: basta:=true;
     end;
   until basta;
   clrscr;
  end.
```

fino a quando non riusciamo più a batterla, ed allora vuol dire che la strategia di gioco che sta adottando è quella ottimale.

Vediamo ora, per chi non lo conoscesse, come si gioca. Si mettono su un campo un certo numero di pedine iniziali, nel nostro caso 11, e con prelievi di 1, 2 o 3 pedine i due avversari tentano di esaurire quelle in campo in modo da averne ognuno un numero pari. Essendo però quelle iniziali dispari, uno solo dei due giocatori avrà un numero pari di pedine e quindi sarà vincitore.

Cosa fa il programma

Così, mano al mio fido IBM PC XT e al Turbo Pascal 3.0., ecco il frutto di un duro e paziente lavoro.

Una volta caricato ed eseguito, il programma mostra un menu comprendente 7 opzioni:

1) Caricare strategia
2) Salvare strategia
3) Iniziare partita
4) Annullare strategia
5) Lista strategia
6) Immetti strategia
7) Fine

Analizziamole ora ad una ad una cominciando dalla 3. Questa permette di fare una partita al gioco del pari. Verrà visualizzata una piccola spiegazione del gioco e verrà chiesto se si vuole giocare per primo, se si vuole la modifica automatica della strategia adottata da Melole ad ogni sconfitta e se si vuole far giocare il computer contro se stesso. A questo punto la partita comincerà e accanto ad ogni mossa effettuata dal computer o dal giocatore verranno visualizzati numero di pedine in campo e pedine del giocatore. Alla fine del gioco il computer proclamerà il vincitore e in caso di sconfitta di Melole, se è stata scelta la modifica automatica, la strategia adottata sarà perfezionata.

Da notare che se si sceglie l'opzione «computer contro se stesso», la seconda mossa del computer, cioè non di Melole, sarà generata casualmente e trattata come se fosse immessa da un giocatore umano.

Dopo la proclamazione del vincitore sarà fatta una richiesta a cui si può rispondere S, N o L rispettivamente per Si, No, Lista. Scegliendo S si farà una nuova partita, che comincerà da dopo la scelta delle opzioni. Quindi se si vuole cambiare qualcuna di queste occorrerà scegliere N per tornare al menu principale e riscegliere l'opzione 3. Se invece si batte L allora verrà visualizzata la strategia adottata attualmente, secondo uno schema che verrà spiegato dopo.

Tornando al menu principale, le opzioni 1 e 2 servono rispettivamente a caricare da disco una strategia e a salvare quella attuale. Verrà prima chiesto il nome del file da registrare o caricare senza estensione, che per default è STR, dopo di che verrà eseguita l'operazione.

L'opzione 4 serve invece ad inizializzare la strategia e quindi causa la perdita di tutte le conoscenze di Melole. La 5 serve a ottenere la lista della strategia anche da menu principale ed è identica alla stessa opzione durante la partita. La 6 serve ad immettere delle strategie manualmente secondo il solito schema dell'opzione Lista. Infine la 7 serve a uscire dal programma.

Cosa si può fare con Melole

Gli utilizzi di Melole sono i più svariati. Così come può già essere utilizzata per studiare la strategia ottimale del gioco del pari, puó essere utilizzato per far vedere agli scettici amici quanto è intelligente il proprio computer, o semplicemente per giocare e vedersi battere.

Lo studio delle strategie può avvenire in diversi modi. Si può creare una strategia a mano e poi verificarla e perfezionarla con Melole oppure si può fare in modo che Melole faccia tutto per voi, basta giocarci un po', o farla giocare contro il computer. Ancora è possibile studiare prima strategie con un numero di pedine iniziali basso, e poi si può alzare gradatamente il tiro usando come base le strategie già sviluppate. Cambiare il numero di pedine iniziali è molto semplice: all'inizio del programma sorgente si trovano infatti le due dichiarazioni di costanti MAX e MAXT. Ora, ammettendo di voler mettere 13 pedine iniziali, basta sostituire l'11 dopo MAX con 13, e il 6 in MAXT con la metà intera di 13 più 1, quindi 7.

Molto utile per lo studio di strategie con un gran numero di pedine iniziali è la possibilità di salvare la strategia ottenuta fino ad un certo punto per poi ricaricarla e perfezionarla quando più aggrada.

Ah, un consiglio: se volete ottenere strategie ben fatte fate giocare Melole o per prima o per seconda, ma non cambiate in continuazione perché in questo modo si otterrebbero strategie piuttosto strane. Comunque per chi vuol provare, il programma è lì pronto.

A questo punto qualcuno potrebbe chiedersi: «Ma non è possibile studiare nient'altro che il gioco del pari con questo programma?». Risposta: sì e no. No se si intende il programma così com'è senza alcuna modifica, Sì se invece siamo disposti a qualche lieve modifica al programma. Infatti la struttura rimane sempre questa, cambiano solo le parti legate al gioco del pari, il resto può essere lasciato intatto.

Per i più volenterosi, non io quindi, propongo un esercizio: modificare il listato in modo che Melole sia in grado di giocare all'esapedone. Per facilitare queste modifiche parlerò allora brevemente del modo di comportarsi di Melole.

Come funziona Melole

Il funzionamento di Melole si basa principalmente sui due array MOV e OLDMOV. Il primo è un array tridimensionale che viene utilizzato per contenere la strategia da adottare. Il primo indice di questo array va da 1 al numero massimo di pedine iniziali, il secondo vale 0 se Melole ha un numero pari di pedine, 1 altrimenti; infine il terzo contiene le mosse 1, 2 o 3 che Melole può effettuare.

Il contenuto di questo array è un valore booleano; se è TRUE vuol dire che la mossa z, con un numero x di pedine in campo e con lo stato y è possibile, se è FALSE invece no.

Il secondo array, OLDMOV, è stavolta a una dimensione, ma è composto da tanti elementi di tipo RECORD composti ognuno da 3 campi. Questo array contiene le mosse effettuate da Melole. Così per esempio, se la seconda mossa effettuata è stata 1, con 7 pedine in campo, un numero pari di pedine prese, allora la seconda posizione di OLDMOV conterrà nei suoi tre campi rispettivamente 7,0,1.

Appena si esegue il programma l'array MOV viene portato tutto a TRUE tranne in caso di mosse non possibili, come ad esempio la 1-0-3 poiché non è possibile prendere 3 pedine se in campo ce n'è solo una. Quando il computer deve far eseguire la mossa a Melole, cosa che avviene nella procedura COMPUTER, per prima controlla se con il numero di pedine e con lo stato attuale c'è qualche mossa possibile. Questo controllo è necessario perché è vero che quando una scatola si svuota si elimina il fiammifero che porta a quella scatola, ma è altrettanto vero che ad una scatola o nodo si può arrivare da tante parti e senza questo controllo il computer si potrebbe bloccare. Se il controllo ha avuto esito positivo il computer genera mosse casuali finché non ne trova una possibile, poi registra la mossa su OLDMOV e aumenta di uno il contatore che indica l'ultima mossa registrata.

Quando la partita finisce il controllo passa alla procedura FINE che se il

perdente è Melole, provvede a prelevare l'ultima mossa efettuata da OLD-MOV che mette in tre variabili da appoggio e rende FALSE la cella dell'array specificata. A questo punto controlla se nel nodo della mossa cancellata ci sono ancora mosse possibili e altrimenti cancella anche la mossa precedente ripetendo di nuovo il controllo. Se poi si sceglie la visualizzazione della lista strategia, allora il controllo passa alla procedura LISTA che visualizza tutte le celle dell'array MOV con il loro contenuto. Inoltre l'ultima mossa effettuata viene visualizzata in rosso grazie al confronto con le variabili d'appoggio suddette a proposito della procedura FINE.

Spawn

Child Processes in Turbo Pascal

di Felice Sobrero - Torino

Quasi tutti i compilatori «C» comprendono delle funzioni di libreria che permettono di caricare ed eseguire un programma esterno e poi rientrare o no nel programma chiamante.

In particolare il nuovissimo Turbo-C della Borland offre EXEC(...) per l'esecuzione senza ritorno di un programma che si sostituisce al chiamante, e SPAWN(...) per l'esecuzione di quello che viene definito «child process» e rientro nel «parent».

Purtroppo invece il Turbo Pascal prevede soltanto le procedure predefinite EXECUTE(filvar) e CHAIN(filvar), che possono solo passare il controllo senza ritorno ad altri programmi compilati con Turbo Pascal.

Vi propongo quindi di sperimentare la semplice funzione SPAWN, che ho preparato ad imitazione dell'omonima funzione Turbo-C, sfruttando i servizi del DOS v3.x che si attivano tramite l'interrupt 21h.

Faremo ricorso a tre servizi di questo interrupt, caricandone i codici in AH:

- $4A : modifica del blocco di memoria allocato.
- $4B : caricamento ed esecuzione di child process. (con AL = 0)
- $49 : liberazione di memoria allocata.

La funzione è stata collaudata su un clone AT con 512k di memoria e DOS Olivetti v3.1 e IBM v3.3. Come vedremo, con 640k di memoria è ancora meglio.

Spawn si aspetta di ricevere due parametri di tipo stringa:

1) un nome di file, obbligatorio. Se manca il pathname completo sarà cercato soltanto nella directory corrente, ignorando il PATH.

Spawn

```
{*****************************************************************************}
{*              Felice Sobrero per MC Microcomputer agosto 1987              *}
{*                          Funzione : SPAWN                                 *}
{*           ESECUZIONE DI CHILD PROCESSES DA TURBO PASCAL                   *}
{*****************************************************************************}

FUNCTION spawn (filename,argument : longstr) : integer;
TYPE  chip = record
              AX,BX,CX,DX,BP,DI,SI,DS,ES,FL : integer;
             end;
VAR
 registers: chip;
 infobuf  : array[1..7]  of integer;              { 7 words buffer per il Dos }
 vidsys   : byte absolute $0000:$0449;                  { modo del video ecc }
 vidold   : array[1..29] of byte;       { salvataggio variabili sistema video }
 color    : byte absolute $B800:$0000;                          { punta a CGA }
 mono     : byte absolute $B000:$0000;                { punta a mono-Hercules }
 display  : array[1..4000] of byte;                { buffer salvataggio VRAM }
 child    : longstr;                                          { nome del file }
 roof,i   : integer;
{---------------------------------------------------------------------------}
PROCEDURE store;
begin
 move(vidsys,vidold,29);                          { salva 29 bytes variabili video }
 if vidsys = 7 then move(mono,display,4000)              { salva display mono }
               else move(color,display,4000);                { oppure colore }
end;
{---------------------------------------------------------------------------}
PROCEDURE retrieve;
begin
 move(vidold,vidsys,29);                          { recupera variabili video }
 if vidsys = 7 then move(display,mono,4000)            { recupera display mono }
               else move(display,color,4000);                { oppure colore }
end;
{---------------------------------------------------------------------------}
BEGIN
roof := seg(HeapPtr^);              { indirizzo di segmento del puntatore Heap }
 with registers do begin
  BX := roof+1;                                  { nuova dimensione della memoria }
  ES := cseg;                                    { a partire dal CS-PSP del Turbo }
  AX := $4A00;                          { $4A = funzione "cambia memoria allocata" }
  msdos(registers);                                        { chiama Dos INT 21h }
  if odd(FL) then begin                                 { se bit 0 di Carry acceso }
                  spawn := AX;                            { codice di errore in AX }
                  exit;                                             { e per sicurezza }
                  end;                                  { si abbandona la funzione }
  store;                                          { salva memoria video e annessi }
  child := filename + chr(0);                    { converte nome in formato ASCIIZ }
  DX := ofs(child) + 1;                 { attenzione: child[0] == length(child) }

---

  DS := seg(child);                                      { indirizzo del nome file }
  BX := ofs(infobuf);                                    { indirizzo completo del }
  ES := seg(infobuf);                                          { buffer di 7 words }
  for i:= 1 to 7 do infobuf[i] := 0;       { prepara il buffer di comunicazione }
  infobuf[2] := ofs(argument);              { indirizzo completo della linea di }
  infobuf[3] := seg(argument);                  { comando passata come argomento }
  AX := $4B00;                               { $4B = funzione "carica ed esegui" }
  msdos(registers);                                        { chiama Dos INT 21h }
  if odd(FL) then spawn := AX           { se Carry acceso = codice errore in AX }
             else spawn := 0;                    { altrimenti nulla da segnalare }
  AX := $4900;                                { $49 = funzione "disalloca memoria" }
  ES := cseg;                                    { al punto in cui eravamo prima }
  msdos(registers);                                        { chiama Dos INT 21h }
 end;                                                                  { with }
retrieve;                                                { recupera memoria video ecc }
END;                                                                   { spawn }
{-------------------------------<<< FINE >>>-------------------------------}
```

```
{*****************************************************************************}
{*              Felice Sobrero per MC Microcomputer agosto 1987              *}
{*            PROGRAMMA DIMOSTRATIVO DELLA FUNZIONE SPAWN                    *}
{*                 DA UTILIZZARE IN FORMATO .COM                             *}
{*****************************************************************************}
PROGRAM demo;
CONST MESSAGE = 'Rientriamo in Turbo Pascal, stato=';
TYPE longstr  = string[80];                              { lunghezza non critica }
VAR ch      : char;
    stato,i : integer;
{$I spawn.pas }                                  { includiamo la nuova funzione }
begin
 clrscr;
 for i:=1 to 2000 do write(chr(177));
 gotoxy(1,1);
 writeln('Premete un tasto per uscire dal programma Turbo Pascal... ');
 repeat read(kbd,ch) until keypressed=false;
 stato := spawn('GWBASIC.EXE','PROVA.BAS');                      { primo esempio }
 writeln(MESSAGE,stato);
 delay(2000);
 stato := spawn('EDIT.EXE','PROVA.DOC');                       { secondo esempio }
 writeln(MESSAGE,stato);
 delay(2000);
 stato := spawn('DEBUG.COM','');                  { terzo esempio, senza parametro }
 writeln(MESSAGE,stato);
 writeln('Fine.');
end.
{-------------------------------<<< FINE >>>-------------------------------}
```

PROVA.BAS

```
10 FOR A=1 TO 350 : PRINT"BASIC ";: NEXT A
20 PRINT:PRINT:PRINT"Battere SYSTEM per tornare al Turbo Pascal"
```

PROVA.DOC

```
questo e' un piccolo file creato con EDIT.EXE
premere SHIFT-F1 per tornare al Turbo Pascal
```

2) una stringa di parametri da trasmettere al child process, sostituibile eventualmente con una stringa nulla.

Spawn restituisce un integer che rappresenta un codice diagnostico:

- 0 = l'operazione si è svolta regolarmente. (Carry Flag spento)
- 2 = file non trovato.
- 7 = controllo memoria alterato. (messaggio dalla funzione $4A)
- 8 = memoria insufficiente per ospitare il child process.
- 10 = environment non valido. (o troppo esteso)
- 11 = formato non valido. (formato EXE tale da confondere il DOS)

Analizziamo come agisce la funzione: — se possibile riserva memoria per l'intero ambiente Pascal. Se non riesce, cessa immediatamente. (Servizio $4A).

— Salta le variabili di sistema relative al video, e decidendo in base a queste salva anche a display-ram della scheda colore o mono. Vi prego di notare la velocità della MOVE(...) del Turbo Pascal! Per semplificare un po' le cose e risparmiare memoria, lasciamo perdere il caso del video grafico: supponiamo di essere in modo testo.

— Carica il nostro file sopra al blocco di memoria riservato e lo esegue, comunicandogli se del caso i parametri da noi stabiliti, sotto forma di una linea di comando. Gli altri byte di valore zero vedete in infobuf[..] servono a creare un nuovo PSP del tutto simile a quello del parent.

— Alla fine del child process recupera la memoria allocata e restaura il video con le sue variabili di sistema, situate nella memoria bassa.

Riprendiamo quindi il testo, gli attributi, la posizione del cursore per ogni pagina, il modo del video ecc.

— Il programma Pascal proseguirà dove era stato interrotto, ritrovando intatte le sue variabili e il suo environment.

Per individuare la zona occupata dal Pascal ho pensato di sondare HeapPtr, un puntatore predefinito che tiene conto anche dello heap nel caso si usino strutture dinamiche come liste, alberi ecc. Attenzione comunque a non invadere l'ultima parte della memoria alta riservata allo stack e al DOS che ci conserva anche la parte transiente di COMMAND.COM. Qualche volta è necessario ricaricarla alla fine del programma Pascal. In caso di sconfinamento si ottiene il codice di errore 7, il programma continua regolarmente ma all'uscita non riesce più a ricaricare l'interprete di comandi.

Vediamo in figura A delle mappe approssimative della situazione.

L'utilità di Spawn sta nel poter preparare programmi che ad un certo punto chiamano delle utility, o altri programmi applicativi in Basic compilato, o in «C», o chiamano l'interprete GWBASIC e poi proseguono il loro corso.

L'esecuzione di PCTOOLS è interessante per controllare tra le «System Info» la nuova allocazione di memoria che il DOS tiene riservata.

Vista la comodità e la compattezza del Turbo Pascal, si potranno preparare dei programmi menu-pilota che non fanno altro che smistare il lavoro ad altro software.

Ho provato anche a lanciare DB.COM che a sua volta carica DBIII-Plus.

Ci sono anche dei casi in cui Spawn non se la cava come previsto: spero che qualche altro lettore o qualcuno da MCmicrocomputer mi possa gentilmente dare dei suggerimenti:

1) Resta il problema di proteggere in modo sicuro al 100% la parte alta della memoria per il ricarico di COMMAND (v. quanto detto sul codice errore 7).

2) Non sempre possiamo eseguire correttamente i file .COM.

3) Se carichiamo COMMAND.COM ne abbiamo effettivamente una seconda copia funzionante, ma al momento dell'EXIT non rientriamo più nel programma chiamante, ma viene restituito il controllo al COMMAND.COM primario.

Una considerazione importante: Spawn non può funzionare con la compilazione «in memoria» del Turbo, come del resto non funzionerebbero la EXECUTE(filvar) e la CHAIN(filvar).

Bisogna compilare su disco in forma di .COM e poi far partire il programma. MC

Figura A

Bibliografia

Peter Norton
«PC-IBM: Guida del Programmatore»
Ed. Mondadori.

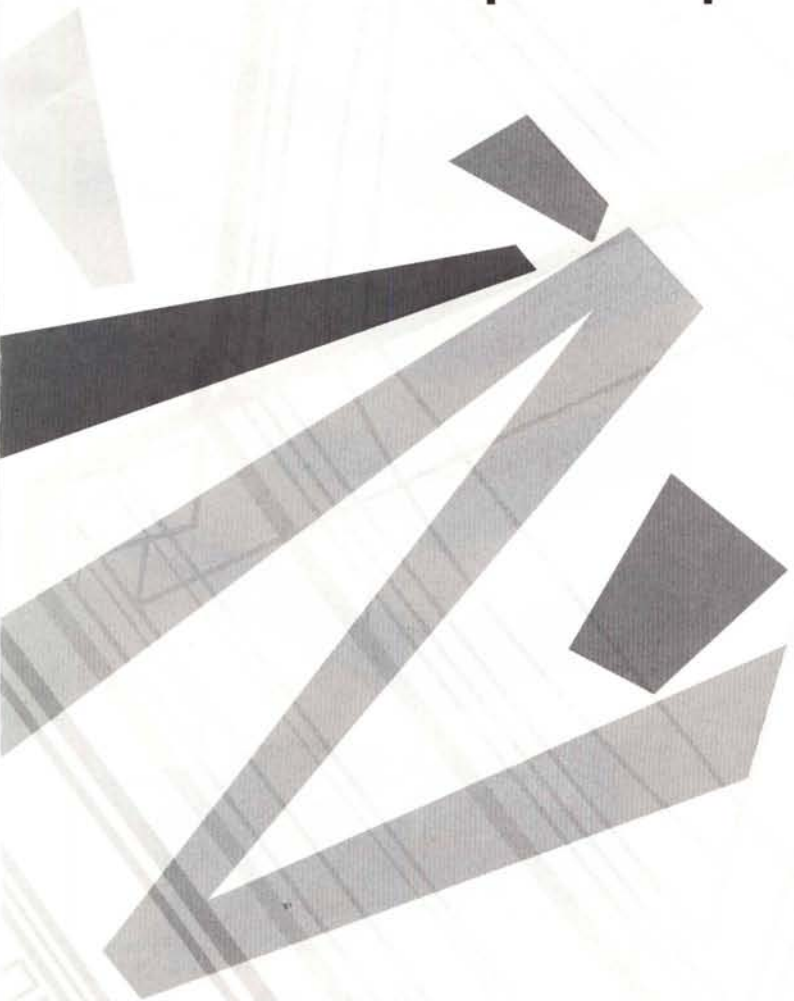
C. Vergoz Rekis
«Push» su Computer Language
di giugno 1987.

«MS-DOS System Programmer Guide»
OEM tech. lib. Olivetti.

RICORDI presenta:

Archimedes

La potenza del RISC nel personal computer più veloce del mondo



▷ Dalla Acorn di Cambridge, U.K., una nuova rivoluzione nell'informatica personale ▷ Archimedes, un computer (o meglio, un'intera serie) dalle altissime prestazioni ▷ Basato su un'unità centrale RISC (Reduced Instruction Set Computer) a 32 bit, Archimedes mette a vostra disposizione una potenza di calcolo finora sconosciuta nel campo dei personal computer ▷ Potenza per eseguire programmi in BBC BASIC a una velocità superiore a quella del linguaggio macchina di molti microcomputer tradizionali ▷ Potenza per accedere a diversi sistemi operativi, dall'ADFS all'MS-DOS* ad altri ancora ▷ Potenza per supportare linguaggi ad alto livello come C, FORTRAN, LISP, PROLOG, PASCAL (oltre a un BASIC formidabile) ▷ Potenza per generare un suono stereofonico di qualità digitale, e una grafica ad altissima definizione con migliaia di colori ▷ Potenza per collegare le più varie periferiche: digitalizzatori, interfacce MIDI, modem, eccetera ▷ Vincitore del Microcomputer Of The Year Award 1987 ▷ Archimedes, il personal computer più veloce del mondo, a un prezzo eccezionale: presso il vostro rivenditore o nei negozi RICORDI.

*MS-DOS è un marchio della Microsoft Corp.

Distributore esclusivo: G. RICORDI & C.
Settore Informatico
Via Salomone, 77
20138 MILANO
tel. 02/5082-315

DOPPIOUNI

Acorn
The choice of experience.
Un'azienda del gruppo Olivetti

Per maggiori informazioni, inviate questo coupon a G. RICORDI & C.
Settore Informatico, Via Salomone, 77, 20138 MILANO

Desidero avere maggiori informazioni su Archimedes

Nome: _____

Cognome: _____

Qualifica professionale: _____

Ditta, Ente o Scuola: _____

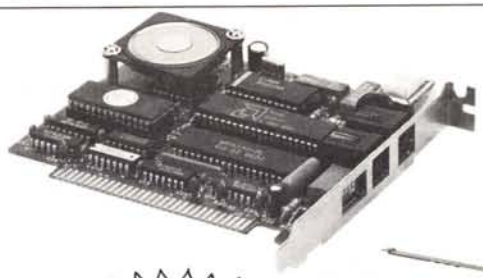
Indirizzo: _____

GVH linea computer

GVH - Via Della Selva Pescarola, 12/2 - 40131 Bologna - Tel. 051/6346181 r.a. - Telex 511375 GVH I

Distributore Link TECHNOLOGY CORP per l'Italia

MODEM PROFESSIONALI PER TUTTI I SISTEMI



SM 1200 BS - Versione interna su card corta.
Caratteristiche:
COMPATIBILITÀ Bell 212A-1200 bps asincrono, Bell 103-300 bps asincrono, CCITT V. 22-1200 bps asincrono, CCITT V. 21-300 bps asincrono - MODULAZIONE 1200 bps: DPSK, 300 bps: FSK - OPERAZIONI Full or half duplex - PORTA RS - 232C.

L. 199.000

SL 1-2-3 - Versione da esterno. Completo.
Caratteristiche:
COMPATIBILITÀ CCITT V. 21 BELL 103 300 bps, CCITT V. 22 BELL 212A 1200 bps, CCITT V. 23 (MODE 1) 600/75, 75/600 bps, CCITT V. 23 (MODE 2) 1200/75, 75/1200 bps, BELL 202 1200/5, 5/1200 bps - MODULAZIONE 1200 bps: DPSK, 0-1200 bps: FSK - OPERAZIONI Full or half duplex - PORTA EIA RS-232C/CCITT V.24

L. 299.000

SM 1200 S - Versione esterna. Completo di alimentazione.
Caratteristiche: uguali all'SM 1200BS

L. 235.000

SM 2400 - Versione esterna. Completo di alimentazione a 220 V.
Caratteristiche:
COMPATIBILITÀ CCITT V. 22 bis - 2400 bps asincrono/sincrono, CCITT V. 22-1200 bps asincrono/sincrono, Bell 212A-1200 bps asincrono/sincrono, Bell 103-300 bps asincrono - MODULAZIONE 2400 bps: 16 QAM, 1200 bps: DPSK, 300 bps: FSK - OPERAZIONI Full or half duplex - PORTA RS - 232C

L. 450.000

SL 2400 A il «TOP MODEM»!! Con 6 standard asincrono/sincrono.
Caratteristiche: uguali all'SM 2400 B + compatibilità sistema V23.

L. 590.000

SM 2400 B - Versione interna su card lunga.
Caratteristiche:
COMPATIBILITÀ CCITT V. 22 bis - 2400 bps asincrono, CCITT V. 22-1200 bps asincrono/sincrono, CCITT V. 23-1200/75 bps, 75/1200 bps asincrono, CCITT V. 21-3000 bps asincrono, Bell 212A-1200 bps asincrono/sincrono, Bell 103-300 bps asincrono - MODULAZIONE 2400 bps: 16 QAM, 1200 bps: DPSK, 1200/75 bps, 75/1200 bps: FSK, 300 bps: FSK - OPERAZIONI Full or half duplex - PORTA RS - 232C, V. 24

L. 399.000

Tutti i MODEM sono corredati di libretto d'istruzioni e software di gestione e sono compresi di **AUTODIAL/ANSWER** con comandi di **AT** estesi.

NUOVI ARRIVI: AT card clock 10 Mhz

AT versione card così composto: 512 K RAM 100 nS. Controller universale per Floppy drive e Hard disk. Corredato di Floppy da 1,2 Mbyte. Con scheda colore grafica (Hercules)+uscita stampante. Tastiera avanzata 101 tasti. Garantito 1 anno. L. 1.590.000

Hard disk e schede speciali sono fornite a richiesta.

Arrivano in continuazione articoli nuovi: fax portatili, schede speciali, tavolette grafiche, mouse, accessori vari. TELEFONATE!!

PREZZI IVA ESCLUSA **Si cercano rivenditori**