a cura di Valter Di Dio

Due programmi in C e uno in Basic, è da un po' di tempo che non si vede più in giro il Pascal; che fine hanno fatto i programmatori Wirth-like, un dubbio mi assale, che siano tutti passati al Turbo Basic? Comunque sia, il C è senza dubbio in forte rimonta, forse perché per scrivere piccole utility è senz'altro più potente. A proposito comunque di linguaggi, anche i vari DBIII, Paradox e 123 sono da considerarsi in un certo senso linguaggi (linguaggi dedicati?); perché allora non ci mandate le vostre migliori Macro in 123 oppure in DBIII o altro? Se sono interessanti prometto che verranno «trattate» alla stregua di qualsiasi altro software!

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. 247.

Whereis

di Stefano Minardi - Firenze

Si tratta di un programma interamente scritto in Basic (QuickBasic 4.5) ma che fa largo uso degli *interrupt* del DOS. Il suo scopo è quello di mostrare come sia possibile sfruttare da Basic le funzioni del DOS per ottenere risultati altrimenti impossibili e, nello stesso tempo, come gestire queste stesse funzioni del DOS. L'esercizio non è comunque fine a se stesso, dato che *qualcosa* il programma fa. *Whereis* consente infatti di localizzare su di un disco i *file* che soddisfano determinate specifiche (nome e/o estensione, con caratteri *jolly*, etc.). Sicuramente avrete già visto programmi del genere, ma questo si differenzia leggermente per il fatto che quando un *file* viene trovato, oltre al nome e al percorso, vengono visualizzate tutte le informazioni che si possono ottenere su di esso e cioè gli attributi (H=nascosto, S=sistema, R=solo lettura, V=etichetta, D=*directory*, A=archivio), data e ora di creazione/modifica e dimensioni — questo, fra l'altro, significa che è possibile modificare il programma in modo da inserire come specifiche di ricerca anche la data o la lunghezza, ecc. Va comunque osservato che l'estrazione di queste notizie dalla DTA penalizza abbastanza le prestazioni in termini di velocità, per cui chi sia interessato al programma per quello che fa — trovare un *file* — può desiderare una procedura più veloce e meno sofisticata: in questo caso sarà sufficiente modificare il parametro 'Ext%' nelle chiamate a *GETFILENAME* e modificare di conseguenza l'*output* dei *file* trovati nel modulo principale.

Il programma richiede:
- compilatore: QB 4.50;
- librerie: BCOM45.LIB, QB.LIB;
- linker: MS-LINK 3.69;
- S.O.: MS-DOS 3.0 o successivo;
- Hardware: PC MS-DOS compatibile

(non vengono utilizzate funzioni esclusive di un modello).

Vengono utilizzate le seguenti funzioni del DOS:

INT 21H f. 19H : Get Current Disk
INT 21H f. 1AH : Set DTA Address
INT 21H f. 2FH : Get DTA Address
INT 21H f. 30H : Get MS-DOS Version Number
INT 21H f. 4EH : Find First File
INT 21H f. 4FH : Find Next File
INT 21H f. 59H : Get Extended Error Information.

Bold.c

di Marco Arcioni - Roma

Questo programma è in grado di enfatizzare sulla stampante, le parole riservate dei tuoi sorgenti Pascal 3.0 o 4.0 o 5.0, C, Assembly, Cobol, GW-Basic.

Se A:\DETECT.PAS è un sorgente Turbo Pascal (qualsiasi versione), per stamparlo sulla porta LPT1: in modo enfatizzato, devi impartire al prompt del tuo PC:

C:\>Bold A:\DETECT.PAS /p > LPT1:

Se nessuna ridirezione è specificata, il file con i relativi codici di controllo, sarà stampato sul corrente device stdout (lo schermo). È quindi possibile trasmettere un file processato ad un altro PC, oppure alla tua stampante seriale scrivendo:

C:\>Bold A:\DETECT.PAS /p > COM1:

oppure ad un nuovo file con la seguente:

C:\>Bold A:\DETECT.PAS /p > A:\DETECT.NEW.

In modo analogo è possibile enfatizzare anche sorgenti in C, in Assembly, in GW-Basic o in Cobol usando i seguenti statement:

C:\>Bold A:\DETECT.C /c > LPT1:

per i sorgenti C, mentre questo per i file Assembly:

C:\>Bold A:\DETECT.ASM /a > LPT1:

Gli switch /p, /c, /a, /o, /b o /x indicano alla utility «bold» di utilizzare le corrette tavole, affinché si ottenga una corretta enfatizzazione del sorgente.

Se nessuna opzione è presente nella linea comandi, e se manca anche l'estensione al file, il programma cerca un file con estensione .PAS, in quanto lo switch di default è /p. Analogamente, se è specificata l'estensione, ma non lo

switch, il programma assume come linguaggio corrente quello dettatogli dall'estensione:

estensione	linguaggio
.PAS	Turbo Pascal V3.0 V4.0 V5.0
.C	ANSI C
.ASM	MASM
.BAS	GW-BASIC
.CBL	COBOL
.XXX	Linguaggio Custom (inizialmente vuoto)

Se nessuna estensione è specificata, il programma assume l'estensione specificatagli dallo switch (se è presente).

Vista la strutturalezza del programma, chiunque dispone di un minimo di esperienza nella programmazione in C, può apportare modifiche al programma, adattandolo ad altri linguaggi. L'unico accorgimento consiste nel porre in fase dichiarativa le parole riservate in ordine alfabetico, quindi aggiungere la parte di codice per l'elaborazione nella struttura case. Come esempio ho introdotto un sesto linguaggio Custom, senza parole riservate, avente switch /x ed estensione '.XXX'.

Setmenu

di Daniele Bufarini - Rieti

La semplicità d'uso di queste routine, costituisce anche il loro punto di forza: la possibilità di poter realizzare menu in un qualsiasi formato deriva dal fatto che, per ogni voce del menu, oltre ad altre informazioni, si passano le coordinate relative al loro posizionamento sullo schermo; in questo modo con una successiva chiamata ad un'altra routine, non si fa altro che gestire le opzioni così impostate.

Come usare le routine nel proprio programma

Usare le due routine Setmenu e Menuto nel proprio programma è molto facile.

```
/* definisce i codici dei tasti */

#define NOTAVAILABLE 255
#define BS                              8
#define TAB              9
#define FORMFEED        12
#define CR              13
#define ESC             27
#define SHTAB           NOTAVAILABLE +  15
#define HOMEKEY         NOTAVAILABLE +  71
#define ENDKEY          NOTAVAILABLE +  79
#define UPKEY           NOTAVAILABLE +  72
#define DOWNKEY         NOTAVAILABLE +  80
#define PGUP            NOTAVAILABLE +  73
#define PGDN            NOTAVAILABLE +  81
#define LEFTKEY         NOTAVAILABLE +  75
#define INSKEY          NOTAVAILABLE +  82
#define RIGHTKEY        NOTAVAILABLE +  77
#define DELKEY          NOTAVAILABLE +  83
#define CTRLLEFTKEY     NOTAVAILABLE + 115
#define CTRLRIGHTKEY    NOTAVAILABLE + 116
#define CTRLEND         NOTAVAILABLE + 117
#define F1              NOTAVAILABLE +  59
#define F2              NOTAVAILABLE +  60
#define F3              NOTAVAILABLE +  61
#define F4              NOTAVAILABLE +  62
#define F5              NOTAVAILABLE +  63
#define F6              NOTAVAILABLE +  64
#define F7              NOTAVAILABLE +  65
#define F8              NOTAVAILABLE +  66
#define F9              NOTAVAILABLE +  67
#define F10             NOTAVAILABLE +  68

#define TRUE (0 == 0)
#define FALSE !TRUE
#define BUFFER 1000
#define NOCURSOR 0x2000

struct option {
        struct option *next;
        struct option *prev;
        unsigned int y;
        unsigned int x;
        unsigned int color;
        char *string;
        unsigned int y_mess;
        char *message;
};
typedef struct option Option;

int getkey (void);
void setcursor (unsigned int shape);
unsigned int getcursor (void);
void writef (int y, int x, int color, char *s, ...);
int intchr (int *arr, int c);
int setmenu (unsigned int y, unsigned int x, unsigned int color, char *string,
                                unsigned int y_mess, char *message);
int menuto (int *num, unsigned int *exitkey, int wrapon);
```

Listato della routine SETMENU.H eseguita con il programma Bold.c.

```
#include "setmenu.c"

main()
{
  int opzione, tasto;

  clrscr();

  setmenu (10, 12, 11, "Opzione 1", 24, "Questa e' la prima opzione");
  setmenu (11, 12, 11, "Opzione 2", 24, "Questa e' la seconda opzione");
  setmenu (12, 12, 11, "Inserimento valori", 24, "Opzione numero 3");
  setmenu (14, 12, 13, "Fine", 24, "Fine lavoro");

  {
    int exitkey[2];
    exitkey[0] = CR; exitkey[1] = ESC; exitkey[2] = '\0';
    tasto = menuto (&opzione, exitkey, TRUE);
  }
  writef (23, 10, 4, "Scelta: %d\nTasto premuto: %d", opzione, tasto);
}
```

Figura 1

1) Bisogna innanzitutto disporre o del codice oggetto delle routine, sia sotto forma di file .obj che .lib, o del file setmenu.c. da includere con l'istruzione del preprocesore #include "setmenu.c". Nel caso si utilizzi il modulo oggetto delle routine e la versione integrata del Turbo C, si deve utilizzare un file .prj che nella sua forma più semplice potrebbe apparire così:

nomefile.c
setmenu.obj

dove nomefile.c è il nome del vostro programma.

2) Effettuare tante chiamate alla routine Setmenu quante sono le voci del menu da visualizzare; i parametri da passare alla funzione Setmenu sono:

setmenu (y, x, color, string, y1, message)

y ed x sono le coordinate Y ed X dell'opzione;
'color' è il colore dell'opzione;
'string' è il relativo testo;
y1 è la coordinata y di message;
'message' è una stringa associata all'opzione che viene visualizzata centrata nella riga y1 con lo stesso colore di string (ovviamente se in 'message' viene passata una stringa nulla, "", non viene visualizzato nessun messaggio).

3) Fare una chiamata alla funzione Menuto con i seguenti parametri:

menuto (opzione, exitkey, wrapon)

'opzione' è un puntatore ad intero e conterrà, dopo la chiamata alla funzione, l'opzione scelta;
'exitkey' è un puntatore ad un array di interi che contengono i tasti di uscita dal menu. Come tasti di uscita si possono ridefinire anche quelli usati internamente dalla funzione, che sono: UpKey o LeftKey, per passare dall'opzione attuale a quella precedente; DownKey o RightKey per passare all'opzione successiva; HomeKey per andare sull'opzione iniziale; EdKey per andare sull'ultima opzione. L'ultimo elemento dell'array deve essere '\0'.
'wrapon' è una variabile booleana che, se settata a TRUE, consente di passare dall'ultima opzione alla prima e viceversa, mentre se è settata a FALSE non consente tale operazione.

La funzione Menuto ritorna in un intero il tasto che si è premuto per uscire dal menu.

In figura 1 vediamo un esempio di utilizzo delle funzioni.

```
/*  Setmenu.c
        Versione 1.01
        By Daniele Bufarini Copyright 1989
        All Rights reserved */


#include <stdio.h>
#include <stdarg.h>
#include <dos.h>
#include <dir.h>
#include <conio.h>
#include <ctype.h>
#include "setmenu.h"

int getkey(void)
{
        int key, lo, hi;

        key = bioskey(0);
        lo = key & 0x00FF;
        hi = (key & 0xFF00) >> 8;
        return((lo == 0) ? hi + 255 : lo);
}

void setcursor(unsigned int shape)
{
 union REGS reg;

 reg.h.ah = 1;
 reg.x.cx = shape;
 int86(0x10, &reg, &reg);
}

int attr(int fg, int bg)
{
        return((bg << 4) + fg);
}

int center(char *string, int width)
{
        int len = strlen(string);

        if (len > width)
                return (1);
        else
                return ((width - len) / 2);
}

unsigned int getcursor(void)
{
 union REGS reg;

 reg.h.ah = 3;
 reg.h.bh = 0;
 int86(0x10, &reg, &reg);
 return(reg.x.cx);
}

void writef(int y, int x, int color, char *s, ...)
{
        va_list argptr;
        char str[BUFFER];
        register char far *scrptr = (char far *) 0xb8000000;
        register int i;
        int len;

        scrptr += ((y - 1) * 160) + ((x << 1) - 2);

        va_start (argptr, format);
        vsprintf (str, s, argptr);
        len = strlen(str);
        for (i = 0; i < len; i++, scrptr++) {
                *scrptr = *(str + i);
                *(++scrptr) = color;
        }
        va_end (argptr);
}

intchr(int *arr, int c)
{
        register int i;

        for (i = 0; *(arr + i) != '\0'; i++)
                if (*(arr + i) == c)
                        return (i + 1);
        return (NULL);
}

Option *s_head = NULL, *s_tail = NULL;

setmenu(unsigned int y, unsigned int x, unsigned int color, char *string,
                unsigned int y_mess, char *message)
```

```
{
        Option *ptr, *p_prev = NULL;

        for (ptr = s_head; ptr; ptr = ptr->next)
                p_prev = ptr;

        if ((ptr = (Option *) calloc (sizeof (Option), 1)) == NULL)
                return (NULL);

        if (p_prev == NULL)
                s_head = ptr;

        if (p_prev)
                p_prev->next = ptr;
        ptr->prev = p_prev;

        ptr->y = y;
        ptr->x = x;
        ptr->color = color;
        ptr->string = string;
        ptr->y_mess = y_mess;
        ptr->message = message;

        s_tail = ptr;
        return (TRUE);
}

menuto(int *num, unsigned int *exitkey, int wrapon)
{
        Option *ptr, *p_prev;
        unsigned char ok = TRUE, oldcursor = getcursor();
        int key, x_mess, end, i = 0;

        for (ptr = s_head; ptr; ptr = ptr->next) {
                writef(ptr->y, ptr->x, ptr->color, ptr->string);
                i++;
        }

        ptr = s_head;
        end = i; i = 1;
        setcursor (NOCURSOR);

        do {
                writef (ptr->y, ptr->x, attr(BLACK, LIGHTGRAY), ptr->string);
                writef (ptr->y_mess, x_mess = center(ptr->message, 80), ptr->color, ptr->message);
                key = getkey();
                writef (ptr->y, ptr->x, ptr->color, ptr->string);
                writef (ptr->y_mess, x_mess, BLACK, ptr->message);

                if (intchr(exitkey, key) != NULL)
                        ok = FALSE;
                else if (key == DOWNKEY || key == LEFTKEY) {
                        if (ptr->next != NULL) {
                                ptr = ptr->next;
                                i++;
                        }
                        else if (ptr->next == NULL && wrapon == TRUE) {
                                ptr = s_head;
                                i = 1;
                        }
                }
                else if (key == UPKEY || key == RIGHTKEY) {
                        if (ptr->prev != NULL) {
                                ptr = ptr->prev;
                                i--;
                        }
                        else if (ptr->prev == NULL && wrapon == TRUE) {
                                ptr = s_tail;
                                i = end;
                        }
                }
                else if (key == HOMEKEY) {
                        ptr = s_head;
                        i = 1;
                }
                else if (key == ENDKEY) {
                        ptr = s_tail;
                        i = end;
                }
        } while (ok);

        for (ptr = s_tail; ptr; ptr = p_prev) {
                p_prev = ptr->prev;
                free (ptr);
        }
        s_head = NULL; s_tail = NULL;

        *num = i;
        setcursor(oldcursor);
        return (key);
```

Listato della routine SETMENU.C eseguite con il programma Bold.c.

Breve descrizione delle routine

Le routine principali che compongono questa utility sono:

— void writef (int y, int x, int color, char *string, ...)
questa funzione scrive direttamente sullo schermo a partire dalle coordinate y,x con colore 'color' la stringa 'string' che segue il formato della printf();

— int intchr (int *array, int c)
questa funzione è identica alla strchr e cerca nell'array puntato da 'array' il valore c; se lo trova ritorna la sua posizione più 1, altrimenti NULL;

— int getkey (void)
questa funzione legge un tasto, tramite il Bios, e lo ritorna; se il tasto è uno di quelli che hanno il doppio codice ritorna 255 + il valore del secondo codice.

— unsigned int getcursor (void)
questa funzione ritorna in una variabile di tipo unsigned la forma attuale del cursore mediante una chiamata al Bios;

— void setcursor (unsigned int shape)
questa funzione setta la forma attuale del cursore come indicato dalla variabile shape;

— int setmenu (int y, int x, int color, char *string, int y1, char *message)
questa funzione salva in una lista dinamica doppiamente "linkata" (ldl) tutti i parametri relativi all'opzione passatagli; tiene aggiornati anche due puntatori globali, head e tail, che puntano rispettivamente alla testa e alla coda della lista, usati anche dalla Menuto.

Ritorna TRUE se c'è stato sufficiente spazio per allocare un nuovo nodo in memoria, altrimenti ritorna FALSE.

— int menuto (int *opzione, int *exitkey, int wrapon)
questa è la funzione che si occupa di gestire il menu contenuto nella 'ldl' costruita da setmenu, simulando, se wrapon è TRUE, una 'ldl' circolare.

Ritorna il tasto che si è premuto per uscire dal menu.

Queste routine fanno anche uso di alcuni define che si trovano nel file setmenu.h, dove si trova anche la dichiarazione della struct option, che contiene le informazioni relative alle singole opzioni.

Un'idea per sfruttare queste funzioni sarebbe realizzare un gestore di help context-sensitive tipo Borland o ciò che la fantasia di ognuno meglio suggerisce.