Questa volta abbiamo cercato di pubblicare anche i listati, frugando tra il software pervenuto alla ricerca di programmi interessanti ma «corti!». Il primo programma è un comando che cerca le stringhe presenti in un qualsiasi file; generalmente servirà per trovare i testi dentro un file; eseguibile. Il secondo programma permette di installare procedure residenti in modo molto semplice; di questo programma non possiamo pubblicare né il listato principale né un secondo programma di utilità, ma si può già lavorare con quello pubblicato. Infine una routine che permette di dividere un file su due dischi (qualcuno potrebbe dire che esiste già il backup, ma avete mai provato poi a fare il restore su un computer con un dos non compatibile?)

Strings

di Egidio Casiraghi - Milano

Ecco un piccolo programma che in molti casi può rivelarsi utile. Non è una novità, infatti ho semplicemente riscritto per il DOS una utility esistente sotto Unix. Il programma permette di estrarre le eventuali stringhe ASCII presenti in un file che non sia di solo testo. Ad esempio è possibile estrarre tutti i messaggi presenti in un eseguibile scrivendo "strings nomefile". Il comando accetta, oltre al nome del file, un altro argomento col quale si specifica la minima lunghezza delle stringhe da considerare. Quindi se voglio conoscere le stringhe di lunghezza minima 7, presenti nel file acad.exe scriverò: strings acad.exe 7. Se viene omesso il secondo argomento o se esso è scorretto (non è un numero oppure è <=0) viene assunto come default 4. Nel caso che il file specificato non esiste o non è leggibile, il programma segnala quale errore si è verificato. Se non viene specificato alcun argomento viene stampata la sintassi del comando.

Come si può notare dal listato, il programma è scritto in C ed ho usato funzioni standard in modo da permettere la compilazione tramite qualsiasi compilatore.

È evidente che il programma potrà essere poi personalizzato per riconoscere un set più o meno esteso di caratteri. (Per coloro che dispongono del compilatore C 5.0 Microsoft consiglio l'uso dell'opzione -Ox che aumenta sensibilmente la velocità di esecuzione).

È disponibile, presso la redazione, il disco con gli ultimi due programmi pubblicati in questa rubrica. Le istruzioni per l'acquisto e l'elenco degli altri programmi disponibili sono a pag. 247.

Strings

```
Listato del Programma  STRINGS
#include <stdio.h>

char *aaa="Program created by Casiraghi Egidio";
main(argc,argv)
int argc;
char **argv;
{
        FILE *fd;
        char *buffer;
        unsigned count;
        int minlen;

        if(!strlen(argv[1]))
        {
 fprintf(stderr,"usage:strings  <filename> [minimum lenght]\n");
                exit(-1);/*Non sono stati specificati argomenti*/
        }

        if((fd=fopen(argv[1],"rb"))==NULL)
        {
                perror("strings");
                exit(-1);/*Non e' possibile aprire il file*/
        }
    /*Verifica la presenza del secondo argomento*/
      if(argc>1)
                if((minlen=atoi(argv[2]))<=0)
                        minlen=4;
    /*Alloca spazio per il buffer di lettura */
      if((buffer=(char *) calloc(8192,sizeof(char)))==NULL)
      {
                fprintf(stderr,"calloc failed\n");
                fclose(fd);
                exit(-1);/*Memoria insufficiente */
      }

      while(!feof(fd))
      {
                count=fread(buffer,sizeof(char),8192,fd);
                strings(buffer,count,minlen);
      }
        fclose(fd);
        exit(0);
}
#define START 1
#define ALPHA 2
/*Ricerca all'interno del buffer le stringhe di lunghezza  minima
specificata e le stampa */
strings(buf,len,minlen)
int len,minlen;
char *buf;
{
        int k,n;
        char *start,c;
        int status,pos;

        status=-1;n=0;
        for(k=0;k<len;k++)
        {
                c=buf[k];
                switch(status)
                {
                case START:
                        if(c<32 || c>126)
                                break;
                        status=ALPHA;
                        start=buf+k;
                        n=1;
                        break;
                case ALPHA:
                        if(c>31 && c<126)
                        {
                                n++;
                                break;
                        }
                        if(n>=minlen)
                        {
                                buf[k]='\0';
                                printf("%s\n",start);
                        }
                        status=START;
                        break;
                }
        }
}
```

```
10  REM ************************************************************************
20  REM *                              INT-SHOW                                  *
30  REM *   Programma per la visualizzazione dei vettori d'interrupt 0..255      *
40  REM *                                          1988 by Davide Nardella       *
50  REM ************************************************************************
100 CLS:KEY OFF:DEF SEG=0
110 START.COUNTER=0:STOP.COUNTER=349
120 GOSUB 190
130 IN$=INKEY$
140 IF IN$="1" THEN START.COUNTER=0:STOP.COUNTER=351:Z=0:GOSUB 190:GOTO 130
150 IF IN$="2" THEN START.COUNTER=352:STOP.COUNTER=703:Z=88:GOSUB 190:GOTO 130
160 IF IN$="3" THEN START.COUNTER=704:STOP.COUNTER=1023:Z=176:GOSUB 190:GOTO 130
170 IF IN$=CHR$(27) THEN CLS:END
180 GOTO 130
190 CLS
200 C$="[1] Int 0..Int 87  [2] Int 88..Int 175 [3] Int 176..Int 255  [Esc] Fine"
210 LOCATE 25,5:PRINT C$;:Y=2:X=4
220 A$="Int    Segm Offs   Int    Segm Offs"
230 LOCATE 1,4:PRINT A$:LOCATE 1,42:PRINT A$
240 FOR COUNTER=START.COUNTER TO STOP.COUNTER STEP 4
250 POINTER=COUNTER
260 GOSUB 440
270 OFFS$=VALUE$:POINTER=COUNTER+2
280 GOSUB 440
290 ADDR$=VALUE$+":"+OFFS$
300 INT.DEC$=STR$(Z):INT.DEC$=RIGHT$(INT.DEC$,(LEN(INT.DEC$))-1)
310 INT.HEX$=HEX$(Z)
320 WHILE LEN(INT.DEC$)<3
330 INT.DEC$="0"+INT.DEC$
340 WEND
350 WHILE LEN(INT.HEX$)<2
360 INT.HEX$="0"+INT.HEX$
370 WEND
380 ADDR$=INT.DEC$+" "+INT.HEX$+" "+ADDR$
390 LOCATE Y,X:PRINT ADDR$;
400 IF Y=23 THEN Y=2:X=X+19:ELSE Y=Y+1
410 Z=Z+1
420 NEXT COUNTER
430 RETURN
440 LSB.VALUE$=HEX$(PEEK(POINTER))
450 IF LEN(LSB.VALUE$)=1 THEN LSB.VALUE$="0"+LSB.VALUE$
460 MSB.VALUE$=HEX$(PEEK(POINTER+1))
470 IF LEN(MSB.VALUE$)=1 THEN MSB.VALUE$="0"+MSB.VALUE$
480 VALUE$=MSB.VALUE$+LSB.VALUE$
490 RETURN
```

INT-SHOW - Con questo programma in GWBASIC è possibile esaminare i vettori di interrupt di un qualsiasi computer MS-DOS per scoprire i punti di inizio dei programmi residenti ed eventualmente per scoprire programmi che si installano da soli (anche i virus).

Installatore

di Davide Nardella - Taranto

Rileggendo gli scorsi numeri di MC, abbiamo buona prova di come sia utile certe volte installare in memoria dei programmi residenti, vuoi per intercettare certi interrupt o per riservarci una parte di memoria in cui passare dei dati a programmi che prevedono una riallocazione del Data Segment. Per fare questo ho costruito dei programmi di supporto che mi permettono una rapida messa a punto di codice residente. E sono: DEVELOPE.ASM (1), che è un installatore di codice residente, con esso è sufficiente scrivere «ciò che vogliamo rimanga», battezzarlo STANDP.INC, assemblare il tutto e «servire caldo». Esso in esecuzione ci dice quale procedura abbiamo installato, l'interrupt rivettorizzato, l'interrupt tampone (usato per riporre il vecchio vettore) l'indirizzo in

```
;*
I_mask_register     equ     0021h
Clock_off           equ     0001h
Set_Int_Vector      equ     0025h
Clock_Interrupt     equ     001Ch
Keep_Int_num_fun    equ     0035h
Psp_Amount          equ     0100h
I_com_area_Segment  equ     004Fh
Already_stand       equ     000Fh
Ret_To_Caller       equ     004Ch
I_dos_fcall         equ     0021h
I_Dos_Exit_Res      equ     0027h
;                           Macro di servizio di Rivectorize
Store       Macro   Interrupt
                    mov     dx,offset User_Area
                    mov     ax,cs
                    push    ds
                    mov     ds,ax
                    mov     al,Interrupt
                    mov     ah,Set_Int_Vector
                    Int     I_Dos_Fcall
                    pop     ds
            EndM
;                           Rivettorizza il nuovo interrupt
Rivectorize Macro   Intr
            Local   User_Clock
                    mov     al,Intr
                    cmp     al,Clock_Interrupt
                    je      User_Clock
                    Store   Intr
User_Clock:         in      al,I_Mask_Register
                    mov     Current_Status,al
                    mov     al,Clock_off
                    out     I_Mask_Register,al
                    Store   Intr
                    mov     al,Current_Status
                    out     I_Mask_Register,al
            EndM
;                           Rivettorizza il vecchio interrupt
Shift       Macro   Old_Int,New_Interrupt
                    mov     ah,Keep_Int_num_fun
                    mov     al,Old_int
                    int     I_Dos_fcall
                    mov     ax,es
                    push    ds
                    mov     ds,ax
                    mov     dx,bx
                    mov     ah,Set_Int_Vector
                    mov     al,New_Interrupt
                    int     I_Dos_fcall
                    pop     ds
            EndM
;                           Stabilisce il segm. corrente per ds
Set_Seg     Macro   Work_Area
                    mov     ax,Work_Area
                    mov     ds,ax
                    mov     es,ax
            EndM
;  Segmento codice
.List
Main        Segment 'Code'
            assume cs:Main,ds:Main,es:Main
            Current_Status  db ?
            jmp     Install
;                               Procedura esterna utente
User_Area:  Include STANDP.INC
;                               Procedura di install
Install:            mov     bx,I_Com_Area_segment
                    mov     es,bx
                    cmp     byte ptr es:[Proc_Number],Already_stand
                    jne     Inst_proc
                    mov     ah,Ret_To_Caller
                    int     I_Dos_Fcall
;
Inst_proc:          mov     byte ptr es:[Proc_Number],Already_Stand
                    Shift   Interrupt_num,User_Interrupt
                    Rivectorize Interrupt_num
                    Set_Seg Main
                    mov     dx,offset Install
                    add     dx,Psp_Amount+2
                    int     I_Dos_Exit_Res
;
Main        EndS
            End
;
;           EndList
```

Versione ridotta di DEVELOPE.ASM, identico nella struttura e nel funzionamento. Quando richiamato non mostra sul video i messaggi di output (nome e parametri). Anche qui non è possibile installare più di una procedura con lo stesso identificatore, ma non avremo nessun messaggio di ritorno.
Questo programma è il vero caricatore, dopo aver sviluppato e collaudato con DEVELOPE le proprie procedure (o tabelle) non è necessario che sul video appaiano i fatti nostri, e potremo inserire il tutto in un «discreto» file batch. Tutti i commenti al programma e le considerazioni tecniche salvo ovvie eccezioni sono identici a quelli di DEVELOPE.

memoria dal quale comincia la nostra area e la sua lunghezza. Possiamo installare fino a sedici procedure sfruttando l'ICA (Area di comunicazione fra processi) come area flag, se tentiamo di installare due volte la stessa procedura avremo un messaggio d'errore. Il file .INC dovrà quindi contenere tre costanti obbligatorie: numero della procedura, interrupt da rivettorizzare, interrupt parcheggio (se gli ultimi due saranno uguali, il programma emetterà messaggio di allocazione tabella). P_LOADER.ASM è identico; in meno ha i messaggi di output, cioè lo utilizzeremo quando il nostro lavoro sarà collaudato e di tutti i messaggi di stato non importa niente a nessuno. DISCARD.PAS (1) ci permette di «ripulire» l'ICA per non dover stare a resettare ogni volta, ma attenzione, non elimina fisicamente le procedure dalla memoria. INT-SHOW.BAS è un programmino monitor che ci visualizza tutti i vettori degli interrupt (da 0 a FF) in formato umano segm:offs. Per ciò che riguarda la struttura, ho utilizzato svariate macro nel programma .ASM che rendono il tutto più comprensibile e funzionale e sono abbastanza generiche da poter essere incluse altrove. Per ovvi motivi non è possibile scrivere un caricatore Basic per i suddetti, e ciò non penso sia un gran male (i caricatori Basic uccidono ogni umana virtù di un programma L.M.). I programmi .ASM li ho compilati con il Macroassembler/2 IBM ma hanno funzionato tranquillamente anche con un 4.0 Microsoft, per DISCARD ho utilizzato il Turbo Pascal 4.0 Borland (anche il 3.0 va bene), INTSHOW è scritto in GWBASIC e compilato con il QuickBasic 3.0 Microsoft, il tutto sotto MS-DOS 3.30. Sebbene, come già detto, possiamo installare sedici procedure in memoria, gli interrupt utenti (da parcheggio) sono solo otto (60..67), per recuperarne altri, possiamo sfruttare quelli da 80h a 85h riservati al Basic, il quale, da prove effettuate, non utilizza affatto (... però attenzione). Per allocare in memoria tabelle più grandi di 64k bisogna sostituire l'int. 27h con il 21h funzione 31h, badando che in dx sia specificato il size dell'area in paragrafi e non in byte (1 paragrafo = 16 byte).

(1) *(di cui non pubblichiamo il listato in quanto eccessivamente lungo).*

Bibliografia

IBM DOS 3.30 Technical Reference
IBM Macroassembler/2 1.00 Language Reference
BORLAND Turbo Pascal 4.0 Reference Manual

Sega-File

di Buson Aldo - Magenta (MI)

Bella idea i floppy da 3" e 1/2 vero? Peccato che molti hanno ancora quelli vecchi che contengono solo 360k. Questa considerazione potrebbe restare fine a se stessa se non ci fossero dei furboni in giro che sbattono sui loro floppy nuovi nuovi dei file da 500 e più kbyte. E chi ha il 5" e 1/4? Se non ha anche un disco rigido ci fa una croce sopra altrimenti un rimedio c'è. Basta prendere questi file troppo grandi e «segarli» in due su altrettanti floppy da 360k, per poi ricomporli su hard disk. È il classico uovo di Colombo che ho pensato bene di trasformare in una frittata di poche righe scritte in Turbo C. Il programma (io l'ho chiamato «Sega-File» ma è vivamente consigliato un RENAME) lavora in modo molto semplice. Supponiamo di avere un file chiamato CATALOGO che vogliamo dividere in CAT1 e CAT2. Si sceglie l'opzione «dividi» con -d- e si inseriscono il nome del mega-file (è chiamato così nel programma) CATALOGO, seguito dalla lettera del drive in cui si trova, e dei semi-file CAT1 e CAT2 seguiti dal drive su cui andranno scritti. Poi basta scegliere le dimensioni dei due semi-file ed il gioco è fatto. Attenzione solo a non rimuovere il floppy che contiene CATALOGO quando i messaggi sul video chiedono di sistemare i floppy destinati a contenere CAT1 e CAT2. Adesso possiamo portarci a casa i due floppy con CAT1 e CAT2 e riunirli su HD scegliendo l'opzione «riunisci». Vengono ancora richiesti i nomi del mega-file che stiamo ricostruendo e dei due semi-file coi rispettivi drive di destinazione e provenienza.

E dopo tutta questa fatica rilassiamoci pure con l'ultimo gioco piratato nonostante le sue dimensioni... In realtà si tratta di un lavoro minimo che richiede però la presenza di un amico che ha sia il 3" e 1/2 che il 5" e 1/4. Comunque nessuno vieta di usare «Sega-File» per trasferire dati da un hard disk ad un altro. Una nota per chi riuscisse a mettere le mani sul listato. La compilazione va fatta in modo small. Inoltre si può variare SIZE in modo da avere più libertà nella scelta delle dimensioni dei file parziali. Così come l'ho definita, SIZE consente di selezionarne le dimensioni a passi di 50k alla volta. Attenti però che SIZE determina anche la dimensione del buffer e quindi se è troppo piccola rallenta il lavoro.

Sega-File

```
#include <conio.h>
#include <fcntl.h>
#include <alloc.h>
#include <stdio.h>
#include <io.h>
#include <dir.h>
#include <stdlib.h>
#include <ctype.h>
#define SIZE 51200L                    /*dimensione buffer                */

char file1[13],file2[13],file3[13];    /* nomi file                       */
char d1,d2,d3;                         /* drive di provenienza            */
unsigned long int l1,l2,l3;            /* lunghezza file                  */
int fh1,fh2,fh3;                       /* file handler                    */
char *buffer;                          /* buffer SIZE Kbyte  (malloc)     */

void dividi(void);
void riunisci(void);
void chiedinomi(void);
void chiedi(char[],char *);
void errore(char);
```

3

```
setdisk(d3);
do if((fh3=open(file3,O_BINARY))==-1) errore(2);
while(fh3==-1);
copia(fh3);
}

void copia(fh)                         /*append a megafile                */
int fh;
{
char d;
l1=filelength(fh);                     /*legge n.byte da copiare*/
for(d=l1/(unsigned)SIZE;d;d--)         /*copia semifile                   */
  {
   read(fh,buffer,(unsigned)SIZE);
   write(fh1,buffer,(unsigned)SIZE);
  }
l1-=((l1/(unsigned)SIZE)*(unsigned)SIZE);   /*copia rimanenza             */
read(fh,buffer,(unsigned)l1);
write(fh1,buffer,(unsigned)l1);
}
```

2

```
void sceglisezioni(void);
void copia(int);

main()
  {
  char risp[2];
  puts("\nSEGAFILE --BAsoft--\n");
  if((buffer=   ar *)malloc((unsigned int)SIZE))==0) /*controlla se il     */
    {                                                /*buffer è disponibile*/
    puts("\nNon c'è memoria a sufficienza...");
    exit(1);
    }
  printf("Vuoi dividere o riunire dei file (d/r) ? \b");
  scanf("%1s",risp);
  if(tolower((int)*risp)=='d') dividi();
  if(tolower((int)*risp)=='r') riunisci();
  }

void dividi()                                    /* divide in due un file*/
  {
  char n;
  chiedinomi();
  printf("\nSe il mega-file è nel drive %c: premi un tasto\n",d1+'a');
  getch();
  setdisk(d1);                                   /* apre megafile        */
  do if((fh1=open(file1,O_BINARY))==-1) errore(0);
  while(fh1==-1);
  l1=filelength(fh1);
  sceglisezioni();                               /*chiede come dividerlo */
  printf("\n\nSe %c: è pronto per il semi-file1 premi un tasto",d2+'a');
  getch();
  setdisk(d2);                                   /*apre semifile1        */
  do if((fh2=_creat(file2,0))==-1) errore(1);
  while(fh2==-1);
  for(n=l2/(unsigned)SIZE;n;n--)                 /*scrive semifile1      */
    {
    read(fh1,buffer,(unsigned)SIZE);
    write(fh2,buffer,(unsigned)SIZE);
    }
  printf("\n\nSe %c: è pronto per il semi-file2 premi un tasto",d3+'a');
  getch();
  setdisk(d3);                                   /*apre semifile2        */
  do if((fh3=_creat(file3,0))==-1) errore(2);
  while(fh3==-1);
  for(n=l3/(unsigned)SIZE;n;n--)                 /*scrive semifile2      */
    {
    read(fh1,buffer,(unsigned)SIZE);
    write(fh3,buffer,(unsigned)SIZE);
    }
  l3-=(l3/(unsigned)SIZE)*(unsigned)SIZE;        /*aggiuge parte mancante*/
  read(fh1,buffer,(unsigned)l3);
  write(fh3,buffer,(unsigned)l3);
  }

void riunisci()                                  /*ricostruisce megafile */
  {
  chiedinomi();
  printf("\nPrepara il disco del semi-file1 (%c:) e premi un tasto",d2+'a');
  getch();
  setdisk(d2);                                   /*apre semifile2        */
  do if((fh2=open(file2,O_BINARY))==-1) errore(1);
  while(fh2==-1);
  printf("\n\nPrepara il disco del mega-file (%c:) e premi un tasto",d1+'a');
  getch();                                       /*apre megafile         */
  setdisk(d1);
  do if((fh1=_creat(file1,0))==-1) errore(0);
  while(fh1==-1);
  copia(fh2);                                    /*copia semifile1 in megafile*/
  printf("\n\nPrepara il disco del semi-file2 (%c:) e premi un tasto",d3+'a');
  getch();
```

4

```
void chiedinomi()
  {
  printf("\nNome mega-file ? ");
  chiedi(file1,&d1);
  printf("\nNome primo semi-file ? ");
  chiedi(file2,&d2);
  printf("\nNome secondo semi-file  ? ");
  chiedi(file3,&d3);

  }

void chiedi(nome,drive)                          /*input nomi e drive    */
char nome[],*drive;
  {
  char d[2],n;
  for(n=13;n>1;n--) printf("\b");                /*fa arretrare cursore  */
  scanf("%12s",nome);                            /*chiede nome file      */
  do
    {
    printf("\nChe drive ? (a,b,c...) :\b\b");   /*chiede drive          */
    scanf("%1s",d);
    *d=tolower((int)*d);
    }
  while((*d<'a')||(*d>'e'));                     /*se sbagliato riprova  */
  *drive=*d-'a';
  }

void sceglisezioni()                             /*scegli dim. semifile  */
  {
  char c;
  l2=(unsigned)SIZE;
  l3=l1-(unsigned)SIZE;
  printf("\nIl mega-file è di %ld byte",l1);
  puts("\nPer scegliere le dimensioni dei semi-file usa <-,-> e RET");
  do
    {
    printf("Semi-file1 %06ld semi-file2 %06ld\015",l2,l3);
    c=getch();
    if(!c)
      {
      switch(getch())
        {
        case 'K': if(l2>(unsigned)SIZE)          /*se <- file1 cala file2 sale*/
                    {
                    l2-=(unsigned)SIZE;
                    l3+=(unsigned)SIZE;
                    }
                  break;
        case 'M': if(l2<(l1-(unsigned)SIZE))     /*se -> file2 cala file1 sale*/
                    {
                    l2+=(unsigned)SIZE;
                    l3-=(unsigned)SIZE;
                    }
        default:;
        }
      }
    }
  while(c!=13);                                  /*se RET esci           */
  }

void errore(n)                                   /*se non trova file     */
char n;
  {
  char risp[2],*mess[]={"mega-file","semi-file1","semi-file2"};
  printf("\nNon riesco ad aprire il %s\n",mess[n]);
  for(;;)                                        /*visualizza un messaggio*/
    {
    printf("Abbandono o riprovo ? (a/r) \b");    /*e chiede il da farsi  */
    scanf("%1s",risp);
    if(tolower((int) *risp)=='r') return;
    if(tolower((int) *risp)=='a') exit(1);
    }
  }
```

da 2 a 64 terminali con il tuo personal

G.I.C.A.

PACCHETTO DI CONTABILITÀ GENERALE, CONTABILITÀ DI MAGAZZINO,
GESTIONE ORDINI, BOLLETTAZIONE E FATTURAZIONE, VERAMENTE INTEGRATO.

Servizi forniti: CORSI DI ISTRUZIONE E AVVIAMENTO PROCEDURA, LINEA DIRETTA TELEFONICA.

G.I.C.A. È DISPONIBILE IN VERSIONE DOS E XENIX, ANCHE SU **SYSTEM/2 IBM**

CERCASI CONCESSIONARI PER ZONE LIBERE

D.M.C. S.r.l.

S.S. Tiberina 3/bis - tel. 075/8510262-8510463
(06011) CITTÀ DI CASTELLO (PG)

DISTRIBUTORI AUTORIZZATI:

BG	M.T.C. - Via Camozzi n. 106 - 035/236606 - Bergamo
BO	D.M.C. - Viale Indipendenza, 54 - 051/211306 - Bologna
BS	COMPUTER SHOP srl - Via Aria Libera n. 24 0364/534934 - Darfo Boario Terme
FI	I.Q.N.P. - Via Reginaldo Giuliani n. 137 - 055/4360975 Firenze
LI	FORM ITALIA snc - Via Grande, 32 - 0586/889408 Livorno
PG	C.D.I. - Via dei Priori n. 80 - 075/62585 - Perugia
PS	D.M.C. - Via Mazzini n. 7/B - 0722/331069 Fermignano
ROMA	NUMERICA srl - Via di Bertinoro n. 6 - 06/423007 Roma
VI	GESTIONI SOFTWARE ITALIA - Via Milano n. 66 0444/322115 - Vicenza
VR	GESTIONI SOFTWARE ITALIA - Via Leoncino n. 35 045/8010044 - Verona