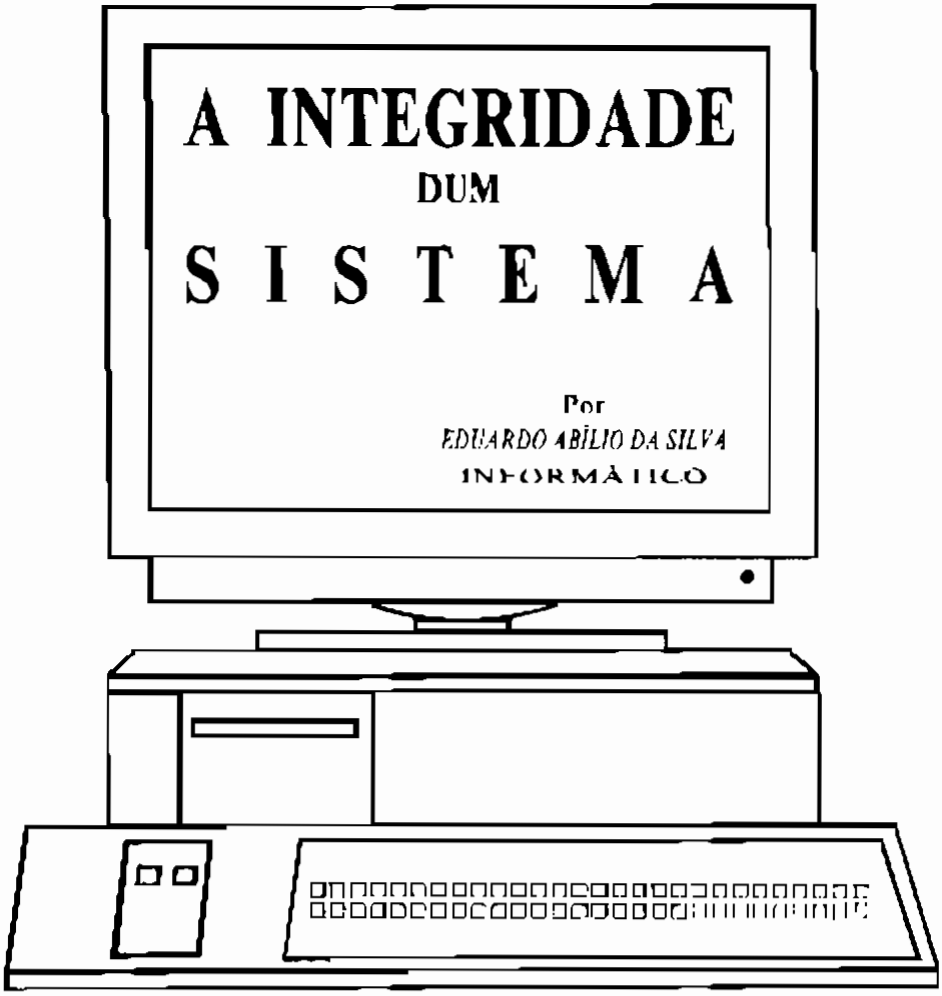
A INTEGRIDADE
DUM
S I S T E M A

Por
EDUARDO ABÍLIO DA SILVA
INFORMÁTICO



EDIÇÃO DO AUTOR
1991

A INTEGRIDADE DUM SISTEMA

A Integridade dum Sistema...

Não pensem que estou a falar de Política! Em Política não há integridade possível. Tentei criar o Partido dos Incorruptos e não consegui arranjar assinaturas suficientes para o legalizar.

Estou, sim, a falar de Informática. Se não sabem o que é um Sistema Informático, vou tentar explicar: Pensem numa máquina simples que trabalhe automaticamente. Para trabalhar deverá ser preciso uma sucessão de operações bem definidas. Portanto terá de ser programada.

Um programa dar-lhe-á a sequência das instruções necessárias para o seu bom funcionamento. Se o programa for bom poderá até optar por alteração de operações consoante as necessidades.

Hoje em dia a maior parte das máquinas, mesmo as máquinas humanas, passaram a ser programadas, ou antes, geridas por outras máquinas a que chamam computadores.

Um Sistema Informático será portanto constituído por um computador, pelo menos. Ou por vários computadores interligados.

Para um computador trabalhar precisa de um programa ou um conjunto de programas.

Pode chamar-se Sistema Informático a este conjunto formado pelo computador ou computadores e o programa ou conjunto de programas.

Também se pode chamar Sistema **Informático** a um conjunto de programas que trabalham em quaisquer computadores e resolvem um assunto.

Mas, para mim, a este Sistema chamo com mais propriedade uma Aplicação **Informática**.

Um programa para um computador consta essencialmente de duas partes. Um conjunto de instruçóes normalmente em circuito fechado e um conjunto de dados.

Este circuito fechado pode abrir-se de uma maneira normal (fim do programa) ou anormal (erro).

Actualmente os programas tendem a trabalhar bem instalados na memória do computador. Só em casos excepcionais e de grande complexidade deixam de estar residentes (em tempo partilhado com opção de prioridades).

Já talvez me alarguei demasiado sobre a Informática sem dizer nada que todos não saibam...

Vamos agora analisar um problema grave na era actual dos computadores.

O problema dos **VÍRUS**. Ligo o que vou expor a computadores pessoais para uma melhor exposição, pois já não há ninguém que **os** não conheça. Mas, mesmo em computadores de grande porte e com meios mais sofisticados de controlo, o problema **su-**

bsiste. Em Sistemas completamente abertos, como os sistemas normais de controlo dos computadores pessoais, então toma amplitudes verdadeiramente catastróficas.

No .princípio, os Vírus talvez não passassem de brincadeiras ou mesmo vingançazinhas de muito bons e geniais programadores que não viram se calhar o seu mérito reconhecido.

Actualmente transformaram-se num negócio. Talvez seja má língua minha, mas não me admirava nada que os fabricantes de produtos ANTIVÍRUS tenham algumas equipas, evidentemente secretas, a fabricá-los. Qualquer dia vendem o antídoto antes do vírus ser detectado.

Os produtos antivírus são às vezes de uma ingenuidade fantástica, o que não os impede de serem vendidos a preços exorbitantes, pelo menos neste País, o que acho um autêntico roubo.

Uma outra situação extremamente grave é a de um vírus poder danificar equipamentos caros no próprio "HARDWARE". Isto é inconcebível pois os fabricantes não tomam providências nenhumas para evitar estes estragos que num bom equipamento nunca deveriam ser permitidos. A qualquer Fábrica será fácil pôr no mercado equipamentos cujo "hardware" esteja protegido contra hipotéticas alterações do seu funcionamento normal. Mas, é claro que, se o equipamento se estragar sem causa imputável ao seu fabrico, melhor é para o vendedor. Por exemplo, num simples receptor de rádio a pilhas que também permita a ligação a corrente contínua exterior, é raro encontrar um que no

circuito de ligação (geralmente uma ficha de n Volts) esteja protegido com um simples díodo contra a ligação de polaridade invertida. Se o desgraçadinho do Zé se engana ao ligá-lo à bateria do automóvel, pum... e lá vai o aparelho para a sucata.

Mas voltemos aos Vírus. Como disse, transformou-se num negócio. Custa-me a acreditar que cabeças mais privilegiadas que a minha (de reles programador) não tenham descoberto uma solução para, o que eu chamo, a Integridade dum Sistema!!!

Provavelmente o que vou expor não é novidade nenhuma para meia dúzia. No entanto talvez não convenha pôr isto a claro.

Passo a exemplificar a minha ideia (que com certeza não é inédita) com programas preparados para um Computador Pessoal, o que não a invalida para aplicação em qualquer outro Computador.

Um programa consta de instruções e dados armazenados num ficheiro (absoluto ou executável). Não é mais que um conjunto de palavras do Computador (em alguns computadores um conjunto de "bytes"). Este conjunto tem um determinado comprimento.

Não será difícil arranjar um programa que possa testar os outros programas. Ou, pelo menos, os ficheiros onde estão guardados e donde são recolhidos para serem executados.

Vamos ver vários programas para organizarem e testarem a Integridade dum Sistema. Pelo menos a integridade dos programas executáveis desse Sistema.

1 - O programa GI:

Gera o ficheiro onde são guardados os dados que servirão para testar a integridade do programa. Este ficheiro deve ser organizado duma maneira prática de rápido acesso para encontrar um registo. Deveria ser sequencial indexado e estar sempre organizado.

A chave do registo será o nome do ficheiro onde está guardado o executável e terá duas variáveis de comprimento suficiente para guardarem o número de palavras (comprimento) do executável e a soma das palavras do executável (ou qualquer outro algoritmo de teste).

2 - O programa MI:

Manipula o ficheiro anterior. Actualiza, cria e elimina registos.

3 - O programa TI:

Testa um programa executável. Estamos mesmo a ver já como. Este programa lê o ficheiro onde está guardado o executável, conta o número de palavras (ou "bytes") do executável e soma para uma variável, previamente inicializada a zero, essas palavras (ou "bytes").

Se estes dois valores coincidirem com os previamente guardados é natural que o ficheiro executável (ou não executável - dados constantes, por exemplo) mantenha a sua integridade.

Se um só "bit" do ficheiro testado tiver sido alterado, a soma não baterá certa e será avisado o utilizador.

Será preciso uma manipulação muito hábil (alteração de duas palavras conjugadas - subtrair a uma o que se acrescenta a outra) para alterar a integridade do ficheiro (e isto se o algoritmo de teste for a soma). Pelo menos, um executável não resistiria a isso.

Após passar neste teste pode ser executado o programa sem perigo.

4 - O programa XI:

Faz o mesmo que o programa anterior e manda executar o programa se o teste estiver certo. Caso contrário avisa o utilizador.

Este programa não passa de um preciosismo visto que, na maioria dos computadores, pode ser substituido com vantagem por um programa em "batch".

Agora vamos ver mais um programa muito interessante:

O programa AUTOTI:

Um programa executável que testa a sua integridade. Este programa lê o ficheiro onde está guardado e testa-se a si próprio. Em vez de ler o ficheiro onde está guardado pode fazer o teste após o carregamento em memória manipulando essa memória, o que tornaria o teste muito mais rápido. Se o teste estiver errado dá uma mensagem de erro e indica o comprimento e a soma. Esta soma (ou outro algoritmo) deve ser posta no programa. Pode ser posta numa variável (módulo 256) ou em números ASCII (módulo 10) e ajustada com o subterfúgio duma cadeia de caracteres alteráveis na zona de dados (constantes) do programa. Há compiladores que introduzem a data, hohora, minuto e segundo na compilação o que torna o acerto através da compilação quase impossível. Neste caso o número certo da .soma será corrigido na cadeia de caracteres directamente no ficheiro que guarda o executável.

Com o TI e o AUTOTI há um acréscimo de tempo de processamento. No entanto talvez este acréscimo seja compensatório.

Evidentemente não pode anular os Vírus, mas pode detectá-los e evitar a sua proliferação, pois os programas testados manterão a sua integridade e, caso contrário, terminarão em erro, avisando o utilizador. Este sistema de controlo também preservará a integridade dum Sistema de "Software" caso esse Sistema seja projectado com um AUTOTI.

Penso que todos os projectos de "Software" futuros terão em conta um AUTOTI ou qualquer teste semelhante, pois as vantagens são evidentes, além duma maior protecção dos direitos de autor.

Vejamos agora um projecto de controlo dum Ambiente DOS nos Computadores Pessoais:

No "PATH" deve existir acesso aos ficheiros TI ou, então, estes devem ser explícitos.

Conteúdo do ficheiro AUTOEXEC.BAT:

```
.....................................
PATH.................c:\bat.........
.....................................
c:\ti.exe c:\ibmbio.com
c:\ti.exe c:\ibmdos.com
c:\ti.exe c:\command.com
c:\ti.exe c:\config.sys
c:\ti.exe c:\autoexec.bat
```

Conteúdo do ficheiro `c:\bat\tp.bat`

```
c:\ti.exe c:\tp\turbo.exe
c:\ti.exe c:\tp\turbo.hlp
c:\ti.exe c:\tp\turbo.tp
c:\ti.exe c:\tp\turbo.tpl
c:\tp\turbo.exe
```

Evidentemente, foram executados os programas:

```
c:\mi c:\ibmbio.com
c:\mi c:\ibmdos.com
c:\mi c:\command.com
c:\mi c:\config.sys
c:\mi c:\autoexec.bat

c:\mi c:\tp\turbo.exe
c:\mi c:\tp\turbo.hlp
c:\mi c:\tp\turbo.tp
c:\mi.exe c:\tp\turbo.tpl
```

Para um controlo efectivo de vírus não deverá ser omitida a extensão nos executáveis para evitar a execução de um programa .com em vez de um `.exe`, pois um vírus pode lembrar-se de fabricar um programa .com que mande executar o respectivo programa `.exe`. Pelo menos na `feitura` de programas "batch" este cuidado deve ser tido em conta.

```
{**************************************}

   {****** PROGRAMA GI ******}

program GI;

          { Gerar ficheiro f_f. }

uses crt;

const

cbuf = 32768;
          { Comprimento do "buffer" de
            f_tf. }

ling = 'Desenvolvido em TURBO PASCAL '+
       '6.0 Copyright (c) Borland 1988';
```

```
type

 rec_f = record
    nome_f : string[12];
    comp_f : longint;
    cont_f : longint;
    test_f : longint;
    end;



var

 f_tf : file;
           { f tf ficheiro a testar a
             integridade. }

 f_f : file of rec f;
           { f_f ficheiro para guardar o
nome (nome_f), comprimento (comp_f) e
soma de "bytes" ou outro algoritmo de
teste (cont_f) do ficheiro f_tf e
quaisquer outros items para o teste à
vontade do utilizador como, por exemplo,
algoritmo de teste (test_f), atributos
do ficheiro f_tf, data do ficheiro f_tf,
etc. }

 r_f : rec_f;
           { Para escrever rec_f. }

 bl : array [1..cbuf] of byte;
           { Para "buffer" de f_tf. )

 fl : longint;
           { Para ciclos e "seek". }
 tfl : word;
           { Para "blockread". }
```

```
 sl : string[255];
          { Para guardar nomes dos
            ficheiros f_f e f_tf. }

 s2 : string[255];
          { Idem. }

 i1 : longint;
          { Para ciclos. }
```

```
procedure apito_f;
begin
sound(333);
delay(999);
nosound;
halt;
end;
```

```
procedure abrir_f;
begin
if paramstr(2)='' then
  s1:='C:\TI.TI'
 else
  s1:=paramstr(2);
  {$I-}
filemode:=0;
assign(f_f,sl);
reset(f_f);
close(f_f);
```

```
  {$I+}
if ioresult=0 then
  begin
  writeln('Execução terminada por já',
  ' existir o ficheiro ',s1);
  apito_f;
  end;
  {$I-}
assign(f_f,sl);
rewrite(f_f);
  {$I+}
if ioresult<>0 then
  begin
  writeln('Ficheiro ',s1,
  ' inacessível!!!');
  apito_f;
  end;
end;



procedure esc_f;
begin
seek(f_f,f1);
write(f_f,r-f);
end;


procedure trat_nome_tf;
begin
          { Tirar em s2, igual a s1,
qualquer "path" ficando só o nome do
ficheiro (até 12 caracteres).
A existência do ficheiro sl já foi
validada em abrir_tf.
No ficheiro f_f só se guardam os nomes
dos ficheiros sem "path".
Desta maneira o teste de integridade
```

serve para um determinado ficheiro esteja onde estiver, mesmo duplicado, e evita ambiguidades, pois não permite ser aplicado em ficheiros diferentes que tenham o mesmo nome. }

```
s2:=s1;
while pos(':',s2)<>0 do
  delete(s2,1,pos(':',s2));
while pos('\',s2)<>0 do
  delete(s2,1,pos('\',s2));
for i1:=1 to length(s2) do
 s2[i1]:=upcase(s2[i1]);
while length(s2)<12 do
  s2:=s2+' ';
          { Guarda em f f o nome
em letras maiúsculas sempre com 12
caracteres a fim de o encontrar quando
procurado. }
end;



procedure abrir_tf;
begin
sl:=paramstr(1);
if length(sl)=0 then
  begin
 writeln('Não deu nome ao ficheiro!!!');
  apito_f;
  end;
  {$I-}
filemode:=0;
assign(f_tf,sl);
reset(f_tf,1);
  {$I+}
```

```
if ioresult<>0 then
  begin
  writeln('Ficheiro ',s1,
    não existe!!!');
  apito_f;
  end;
if filesize(f_tf)=0 then
  begin
  writeln('Ficheiro ',s1,' vazio!!!');
  apito_f;
  end;
trat_nome_tf;
end;



procedure ler_tf;
begin
blockread(f_tf,b1,cbuf,tf1);
end;



procedure tratar;
begin;
r_f.nome_f:=s2;
r_f.cont_f:=0;
r_f.comp_f:=0;
val(paramstr(3),i1,tf1);
if (tf1<>0) or (i1<0) or (i1>3) then
  r_f.test_f:=0
 else
  r f.test_f:=i1;
ler-tf;
while tf1>0 do
  begin
  for i1:=1 to tfl do
    begin
```

```
            { Seguem várias hipóteses de
testes diferentes, e podem ser
acrescentadas outras quaisquer.
No entanto é suficiente o teste de soma
que assume por omissão. }

    case r_f.test_f of
      0 : r_f.cont_f:=r_f.cont_f+b1[i1];
      1 : r_f.cont_f:=r_f.cont_f-b1[i1];
      2 : if odd(r_f.comp_f) then
            r_f.cont_f:=r_f.cont_f+b1[i1]
              else
            r_f.cont_f:=r_f.cont_f-b1[i1];
      3 : if not odd(r_f.comp_f) then
            r_f.cont_f:=r_f.cont_f+b1[i1]
              else
            r_f.cont_f:=r_f.cont_f-b1[i1];
      end;
      inc(r_f.comp_f);
    end;
    ler_tf;
  end;
end;
```

```
begin
checkbreak:=false;
abrir f;
abrir_tf;
r_f.comp_f:=1;
            { Escreve no registo zero em
     comp f o número do último registo
     carregado em f_f (comprimento do
     ficheiro f_f). }
```

```
f1:=0;
esc f;
tratar;
f1:=1;
esc_f;
writeln('*nreg ',f1,'nfic ',r_f.nome_f,
        ' cfic ',r_f.comp_f,' nbfic ',
         r_f.cont_f,'*   fim GI *');

close (f_f);
close (f_tf);

end.



{**************************************}
```

```
{****************************************}

   {****** PROGRAMA MI ******}

program MI;

            { Manusear f f. Inserir,
              apagar e modificar registos
              e m f_f. }

uses crt;

const

cbuf = 32768;
            { Comprimento do "buffer" de
              f±f . )

ling = 'Desenvolvido em TURBO PASCAL '+
       '6.0 Copyright (c) Borland 1988';
```

```
type

 rec_f = record
    nome_f : string[12];
    comp_f : longint;
    cont_f : longint;
    test_f : longint;
    end;


var

 f_tf : file;
           { f_tf ficheiro a testar a
             integridade. }

 f_f : file of rec_f;
           { f_f ficheiro para guardar o
nome (nome_f), comprimento (comp_f) e
soma de "bytes" ou outro algoritmo de
teste (cont_f) do ficheiro f_tf e
quaisquer outros items para o teste à
vontade do utilizador como, por exemplo,
algoritmo de teste (test_f), atributos
do ficheiro f_tf, data do ficheiro f_tf,
etc. }

 r_f : rec_f;
           { Para escrever rec_f. }

 r_g : rec_f;
           { Para guardar rec_f. }

 nfic : integer;
           { Para guardar o número de
```

```
registos, exceptuando o registo zero,
existentes em f_f correspondendo cada
registo a um ficheiro f_tf.
Este número nfic é guardado no registo
zero de f_f em comp_f. }

 bl : array [1..cbuf] of byte;
           { Para "buffer" de f_tf. }

 fl : longint;
           { Para ciclos e "seek". }

 tf1 : word;
           { Para "blockread". }

 sl : string[255];
           { Para guardar nomes dos
             ficheiros f_f e f_tf. }

 s2 : string[255];
           { Idem. )

 i1 : longint;
           { Para ciclos. )

 gl : longint;
           { Para guardar. }

 inf : integer;

 sup : integer;
```

```
procedure apito_f;
begin
sound(333);
delay(999);
nosound;
halt;
end;


procedure abrir_f;
begin
if paramstr(2)='' then
  s1:='C:\TI.TI'
 else
  sl:=paramstr(2);
  {$I-}
filemode:=2;
assign(f_f,sl);
reset(f_f);
  {$I+}
if ioresult<>0 then
  begin
  writeln('Ficheiro ',s1,
  ' inacessível!!!');
  apito_f;
  end;
end;



procedure ler_f;
begin
seek(f_f,f1);
read(f_f,r_f);
end;
```

```
procedure esc_f;
begin
seek(f_f,f1);
write(f_f,r_f);
end;



procedure procura_a;
begin
          { O ficheiro f_f deve estar
sempre organizado. Os registos deste
ficheiro devem estar ordenados
sequencialmente pela chave nome_f para
uma fácil pesquisa pelo programa TI. }

f1:=0;
ler_f;
          { No registo zero em comp_f
está o número do último registo
carregado em f_f (comprimento do
ficheiro f_f). }

inf:=1;
sup:=r_f.comp f;
       { comprimento do ficheiro f_f. }
end;



procedure procura_b;
begin
while sup>=inf do
  begin
  f1:=(sup+inf) div 2;
  ler_f;
  if s2>r_f.nome_f then
    inc(inf)
```

```
  else if s2<r_f.nome_f then
    dec(sup)
  else
    begin
    g1:=f1;
    exit;
    end;
  end;
end;



procedure procurar;
begin
procura_a;
nfic:=sup;
g1:=nfic;
procura_b;
end;



procedure procura;
begin
procurar;
if s2=r_f.nome f then
  exit;
inc(g1);
f1:=0;
          { Escreve no registo zero
em comp f o número do último registo
carregado em f_f (comprimento do
ficheiro f_f). }

r_f.comp_f:=g1;
nfic:=g1;
esc_f;
end;
```

```
procedure dec_f;
begin
f1:=0;
dec(nfic);
r_f.comp_f:=nfic;
esc_f;
end;



procedure org_fic;
begin
          { O ficheiro f_f deve estar
sempre organizado. Os registos deste
ficheiro devem estar ordenados
sequencialmente pela chave nome f para
uma fácil pesquisa pelo programa TI. }

if f1=nfic then
  begin
  dec_f;
  exit;
  end;
g1:=f1+1;
for f1:=g1 to nfic do
  begin
  ler_f;
  dec(f1);
  esc_f;
  inc(fi);
  end;
dec_f;
end;
```

```
procedure del_fic;
begin
procurar;
if s2=r f.nome_f then
  begin-
  r_f.nome_f:='                II
  esc_f;
  writeln('Eliminado o ficheiro ',s1);
  org_fic;
  apito_f;
  end;
writeln('Não existe o ficheiro ',s1);
end;



procedure trat_nome_tf;
begin
          { Tirar em s2, igual a s1,
qualquer "path" ficando só o nome do
ficheiro (até 12 caracteres).
A existência do ficheiro s1 já foi
validada em abrir_tf.
No ficheiro f_f só se guardam os nomes
dos ficheiros sem "path".
Desta maneira o teste de integridade
serve para um determinado ficheiro
esteja onde estiver, mesmo duplicado, e
evita ambiguidades, pois não permite ser
aplicado em ficheiros diferentes que
tenham o mesmo nome. }

s2:=s1;
while pos(':',s2)<>0 do
  delete(s2,1,pos(':',s2));
while pos('\',s2)<>0 do
  delete(s2,1,pos('\',s2));
```

```
for i1:=1 to length(s2) do
 s2[i1]:=upcase(s2[i1]);
while length(s2)<12 do
  s2:=s2+' ';
            { Guarda em f f o nome
em letras maiúsculas sempre com 12
caracteres a fim de o encontrar quando
procurado. }
end;



procedure abrir_tf;
begin
s1:=paramstr(1);
if length(s1)=0 then
  begin
 writeln('Não deu nome'ao ficheiro!! !');
  apito_f;
  end;
  {$I-}
filemode:=0;
assign(f_tf,s1);
reset(f_tf,1);
  {$I+}
trat_nome_tf;
if ioresult<>0 then
  begin
            { Se o ficheiro dado para
incluir no teste de integridade não
existir e ainda estiver no ficheiro f_f
será eliminado deste ficheiro f_f. )

  del fic;
  apito f;
  end;
if filesize(f_tf)=0 then
  begin
```

```
  writeln('Ficheiro ',s1,' vazio!!!');
  apito_f;
  end;
end;



procedure ler_tf;
begin
blockread(f_tf,b1,cbuf,tf1);
end;



procedure tratar;
begin;
r_f.nome_f:=s2;
r_f.cont_f:=0;
r_f.comp_f:=0;
val(paramstr(3),i1,tf1);
if (tf1<>0) or (i1<0) or (i1>3) then
  r_f.test_f:=0
 else
  r f.test_f:=i1;
ler-tf;
while tf1>0 do
  begin
  for i1:=1 to tfl do
    begin

           { Seguem várias hipóteses de
testes diferentes, e podem ser
acrescentadas outras quaisquer.
No entanto é suficiente o teste de soma
que assume por omissão. }
```

```
    case r_f.test_f of
      0 : r_f.cont_f:=r_f.cont_f+b1[i1];
      1 : r_f.cont_f:=r_f.cont_f-b1[i1];
      2 : if odd(r_f.comp_f) then
            r_f.cont_f:=r_f.cont_f+b1[i1]
            else
            r_f.cont_f:=r_f.cont_f-b1[i1];
      3 : if not odd(r_f.comp_f) then
            r_f.cont_f:=r_f.cont_f+b1[i1]
            else
            r_f.cont_f:=r_f.cont_f-b1[i1];
      end;
      inc(r_f.comp_f);
    end;
    ler_tf;
  end;
end;


procedure ordenar;
begin
          { O ficheiro f_f deve estar
sempre organizado. Os registos deste
ficheiro devem estar ordenados
sequencialmente pela chave nome_f para
uma fácil pesquisa pelo programa TI. }

f1:=nfic;
ler_f;
r_g:=r_f;
for f1:=nfic-1 downto 1 do
  begin
  ler_f;
  if r_f.nome_f<r_g.nome_f then
    begin
    inc(f1);
    r_f:=r_g;
    esc_f;
```

```
    exit;
    end;
  inc(f1);
  esc_f;
  dec(f1);
  end;
r_f:=r_g;
f1:=1;
esc_f;
end;




begin
checkbreak:=false;
abrir_f;
abrir_tf;
procura;
tratar;
f1:=g1;
            { Escreve registo procurado,
 já existente ou não no ficheiro f_f. }

esc_f;
writeln('*nreg ',f1,' nfic ',r_f.nome_f,
        ' cfic ',r_f.comp_f,' nbfic ',
        r_f.cont_f,'  *   fim MI *');

ordenar;
close (f_f);
close (f_tf);

end.

{****************************************}
```

```
{**************************************}

    {****** PROGRAMA TI ******}

program TI;

            { Testar integridade do
              ficheiro f_tf. }

uses crt;

const

cbuf = 32768;
            { Comprimento do "buffer" de
              f_tf. }

ling = 'Desenvolvido em TURBO PASCAL '+
       '6.0 Copyright (c) Borland 1988';
```

```
type

 rec_f = record
    nome_f : string[12];
    comp_f : longint;
    cont_f : longint;
    test_f : longint;
    end;



var

 f_tf : file;
           { f tf ficheiro a testar a
             integridade. }

 f_f : file of rec f;
           { f_f ficheiro para guardar o
nome (nome_f), comprimento (comp_f) e
soma de "bytes" ou outro algoritmo de
teste (cont_f) do ficheiro f_tf e
quaisquer outros items para o teste à
vontade do utilizador como, por exemplo,
algoritmo de teste (test-f), atributos
do ficheiro f_tf, data do ficheiro f_tf,
etc. }

 r_f : rec_f;
           { Para escrever rec_f. }

 bl : array [1..cbuf] of byte;
           { Para "buffer" de f_tf. }

 fl : longint;
           { Para ciclos e "seek". }
```

```
  tf1 : word;
              { Para "blockread". }

  s1 : string[255];
              { Para guardar nomes dos
                ficheiros f_f e f_tf. }

  s2 : string[255];
              { Idem. }

  i1 : longint;
              { Para ciclos. }

  inf : integer;

  sup : integer;

  c1 : longint;
              { Para contar número de
        "bytes" (comprimento de f_tf). }

  nbl : longint;
              { Para somar número de
       "bytes" de f_tf ou gerar qualquer
        outro algoritmo de teste. }

  ctfl : longint;
            { Para comprimento de f_tf. }



procedure apito;
begin
sound(900);
delay(300);
nosound;
end;
```

```
procedure repetir;
begin
writeln('***FIM DE TI POR ERRO***');
repeat
  begin
  apito;
  delay(3000);
  end;
until keypressed;
            { Pode optar por fazer
"reset" do sistema pois alguma anomalia
existe com possibilidade de vírus. }
            {
procedure resetsystem;
begin
inline ($ea/$5b/$e0/$00/$f0);
end;
            }
halt;
end;



procedure abrir_f;
begin
if paramstr(2)='' then
  s1:='C:\TI.TI'
 else
  s1:=paramstr(2);
  {$I-}
filemode:=0;
assign(f_f,s1);
reset(f_f);
```

```
  {$I+}
if ioresult<>0 then
  begin
  writeln
  ('Erro 6. Não existe o ficheiro ',s1);
  repetir;
  end;
end;



procedure ler f;
begin
seek(f_f,f1);
read(f_f,r_f);
end;



procedure trat_nome_tf;
begin
          { Tirar em s2, igual a s1,
qualquer "path" ficando só o nome do
ficheiro (até 12 caracteres).
A existência do ficheiro s1 já foi
validada em abrir_tf.
No ficheiro f_f só se guardam os nomes
dos ficheiros sem "path".
Desta maneira o teste de integridade
serve para um determinado ficheiro
esteja onde estiver, mesmo duplicado, e
evita ambiguidades, pois não permite ser
aplicado em ficheiros diferentes que
tenham o mesmo nome. }

s2:=s1;
while pos(':',s2)<>0 do
  delete(s2,1,pos(':',s2));
```

```
while pos('\',s2)<>0 do
  delete(s2,1,pos('\',s2));
for i1:=1 to length(s2) do
 s2[i1]:=upcase(s2[il]) ;
while length(s2)<12 do
  s2:=s2+' ';
            { Guarda em f f o nome
em letras maiúsculas sempre com 12
caracteres a fim de o encontrar quando
procurado. }
end;



procedure abrir-tf;
begin
s1:=paramstr(1);
if length(sl)=0 then
  begin
  writeln
('Erro 3. Não deu nome ao ficheiro!!! ');
  repetir;
  end;
 {$I-}
filemode:=0;
assign(f_tf,sl);
reset(f_tf,1);
 ($I+)
if ioresult<>0 then
  begin
  writeln
  ('Erro 4. Não existe o ficheiro ',s1),
  repetir;
  end;
if filesize(f_tf)=0 then
  begin
  writeln('Erro 5. Ficheiro ',s1,
  ' vazio!!!');
```

```
  repetir;
  end;
trat_nome_tf;
end;



procedure ler_tf;
begin
blockread(f_tf,bl,cbuf,tf1);
end;



procedure procura_a;
begin
          { O ficheiro f f deve estar
sempre organizado. Os registos deste
ficheiro devem estar ordenados
sequencialmente pela chave nome f para
uma fácil pesquisa pelo programa TI. }

f1:=0;
ler_f;
          { No registo zero em comp_f
está o número do último registo
carregado em f_f (comprimento do
ficheiro f_f). }

inf:=l;
sup:=r_f.comp_f;
       { Comprimento do ficheiro f_f. }
end;
```

```
procedure procura b;
begin
while sup>=inf do
  begin
  f1:=(sup+inf) div 2;
  ler_f;
  if s2>r_f.nome_f then
    inc(inf)
  else if s2<r_f.nome_f then
    dec(sup)
  else
    exit;
  end;
writeln
('Erro 2. Não existe o ficheiro ',
s1,' na lista de teste!! !');
repetir;
end;


procedure procurar;
begin
procura_a;
procura_b;
end;


procedure verificar;
begin
if (r_f.comp_f=ctf1) and (r_f.comp_f=c1)
 and(r_f.cont_f=nb1) then
   exit;

writeln
('Erro 1. Integridade do ficheiro ',s1,
' alterada!!! ');
writeln('ctf1 ',ctf1,' c1 ',c1,' nbl ',
nbl);
```

```
writeln('*nreg ',f1,' nfic ',r_f.nome_f,
        ' cfic ',r_f.comp_f,' nbfic ',
         r_f.cont_f,' *  fim TI *');
repetir;
end;



procedure tratar;
begin
c1:=0;
nb1:=0;
ler tf;
whiie tfl > 0 do
  begin
  for i1:=1 to tfl do
    begin
          { Seguem várias hipóteses de
testes diferentes, e podem ser
acrescentadas outras quaisquer.
No entanto é suficiente o teste de soma
que assume por omissão. }

      case r_f.test_f of
        0 : nb1:=nb1+b1[i1];
        1 : nb1:=nb1-b1[i1];
        2 : if odd(c1) then
              nb1:=nb1+b1[i1]
            else
              nb1:=nb1-b1[i1];
        3 : if not odd(c1) then
              nb1:=nb1+b1[i1]
            else
              nb1:=nb1-b1[i1];
      else
```

```
        begin
        writeln('Erro 7. Ficheiro de',
        ' teste estragado!! !');
        repetir;
        end;
      end;
    inc(c1);
    end;
  ler_tf;
  end;
end;
```

```
begin
normvideo;
checkbreak:=(false);
abrir f;
abrir_tf;
ctf1:=filesize(f_tf);
procurar;
tratar;
verificar;
writeln('*nreg ',f1,' nfic ',r_f.nome_f,
        ' cfic ',r_f.comp_f,' nbfic ',
         r_f.cont_f,' *  fim TI *');

close (f_f);
close (f_tf);

end.
```

```
{**************************************}
```

```
{**************************************}

    {****** PROGRAMA XI ******}

{$M $4000,0,0}

program XI;

            { Testar integridade do
            ficheiro f_tf e executá-lo. }

uses crt,dos;

const

cbuf = 32768;
            { comprimento do "buffer" de
              f_tf. }
```

```
ling = 'Desenvolvido em TURBO PASCAL '+
       '6.0  Copyright (c) Borland 1988';


type

 rec_f = record
    nome_f : string[12];
    comp_f : longint;
    cont_f : longint;
    test_f : longint;
    end;



var

 f_tf : file;
           { f tf ficheiro a testar a
             integridade. }

 f f : file of rec_f;
           { f_f ficheiro para guardar o
nome (nome_f), comprimento (comp_f) e
soma de "bytes" ou outro algoritmo de
teste (cont_f) do ficheiro f_tf e
quaisquer outros items para o teste à
vontade do utilizador como, por exemplo,
algoritmo de teste (test_f), atributos
do ficheiro f_tf, data do ficheiro f_tf,
etc. }

 r_f : rec_f;
           { Para escrever rec_f. }

 b1 : array [1..cbuf] of byte;
           { Para "buffer" de f_tf. }
```

```
f1 : longint;
          { Para ciclos e "seek". }

tf1 : word;
          { Para "blockread". }

s1 : string[255];
          { Para guardar nomes dos
            ficheiros f_f e f_tf. }

s2 : string[255];
          { Idem. }

i1 : longint;
          { Para ciclos. }

inf : integer;

sup : integer;

c1 : longint;
          { Para contar número de
       "bytes" (comprimento de f_tf). }

nbl : longint;
          { Para somar número de
       "bytes" de f_tf ou gerar qualquer
        outro algoritmo de teste. )

ctfl : longint;
         { Para comprimento de f_tf. }
```

```
procedure apito;
begin
sound(900);
delay(300);
nosound;
end;



procedure repetir;
begin
writeln('***FIM DE XI POR ERRO***');
repeat
  begin
  apito;
  delay(3000);
  end;
until keypressed;
           { Pode optar por fazer
"reset" do sistema pois alguma anomalia
existe com possibilidade de vírus. }
           {
procedure resetsystem;
begin
inline ($ea/$5b/$e0/$00/$f0);
end;
           }
halt;
end;



procedure abrir_f;
begin
if paramstr(2)='' then
  s1:='C:\TI.TI'
 else
  s1:=paramstr(2);
```

```
  {$I-}
filemode:=0;
assign(f_f,s1);
reset(f_f);
  {$I+}
if ioresult<>0 then
  begin
  writeln
  ('Erro 6. Não existe o ficheiro ',s1);
  repetir;
  end;
end;



procedure ler_f;
begin
seek(f_f,f1);
read(f_f,r_f);
end;



procedure trat_nome_tf;
begin
          { Tirar em s2, igual a s1,
qualquer "path" ficando só o nome do
ficheiro (até 12 caracteres).
A existência do ficheiro s1 já foi
validada em abrir_tf.
No ficheiro f_f só se guardam os nomes
dos ficheiros sem "path".
Desta maneira o teste de integridade
serve para um determinado ficheiro
esteja onde estiver, mesmo duplicado, e
evita ambiguidades, pois não permite ser
aplicado em ficheiros diferentes que
tenham o mesmo nome. }
```

```
s2:=s1;
while pos(':',s2)<>0 do
  delete(s2,1,pos(':',s2));
while pos('\',s2)<>0 do
  delete(s2,1,pos('\',s2));
for i1:=1 to length(s2) do
 s2[i1]:=upcase(s2[i1]);
while length(s2)<12 do
  s2:=s2+' ';
          { Guarda em f f o nome
em letras maiúsculas sempre com 12
caracteres a fim de o encontrar quando
procurado. )
end;



procedure abrir_tf;
begin
sl:=paramstr(1);
if length(sl)=0 then
  begin
  writeln
('Erro 3. Não deu nome ao ficheiro!! !');
  repetir;
  end;
 {$I-}
filemode:=0;
assign(f_tf,s1);
reset(f_tf,1);
 {$I+}
if ioresult<>0 then
  begin
  writeln
  ('Erro 4. Não existe o ficheiro ',sl);
  repetir;
  end;
```

```
if filesize(f_tf)=0 then
  begin
  writeln('Erro 5. Ficheiro ',s1,
  ' vazio!!!');
  repetir;
  end;
trat_nome_tf;
end;



procedure ler_tf;
begin
blockread(f_tf,b1,cbuf,tf1);
end;



procedure procura_a;
begin
           { O ficheiro f_f deve estar
sempre organizado. Os registos deste
ficheiro devem estar ordenados
sequencialmente pela chave nome f para
uma fácil pesquisa pelo programa TI. }

f1:=0;
ler_f;
           { No registo zero em comp_f
está o número do último registo
carregado em f_f (comprimento do
ficheiro f_f). }

inf:=1;
sup:=r_f.comp_f;
       { Comprimento do ficheiro f_f. }
end;
```

```
procedure procura_b;
begin
while sup>=inf do
  begin
  f1:=(sup+inf) div 2;
  ler_f;
  if s2>r_f.nome_f then
    inc(inf)
  else if s2<r_f.nome_f then
    dec(sup)
  else
    exit;
  end;
writeln
('Erro 2. Não existe o ficheiro ',s1,
' na lista de teste!!!');
repetir;
end;



procedure procurar;
begin
procura a;
procura_b;
end;



procedure verificar;
begin
if (r_f.comp_f=ctf1) and (r_f.comp_f=c1)
 and(r_f.cont_f=nb1) then
  exit;
```

```
writeln
('Erro 1. Integridade do ficheiro ',s1,
' alterada!!!');

writeln('ctf1 ',ctf1,' c1 ',c1,' nbl ',
nbl);
writeln('*nreg ',f1,' nfic ',r_f.nome_f,
        ' cfic ',r_f.comp_f,' nbfic ',
         r_f.cont_f,' *  fim XI *');
repeat;
end;



procedure tratar;
begin
c1:=0;
nbl:=0 ;
ler_tf;
while tfl > 0 do
  begin
  for i1:=1 to tf1 do
    begin
          { Seguem várias hipóteses de
testes diferentes, e podem ser
acrescentadas outras quaisquer.
No entanto é suficiente o teste de soma
que assume por omissão. }

      case r_f.test_f of
        0 : nb1:=nb1+b1[i1];
        1 : nb1:=nb1-b1[i1];
        2 : if odd(c1) then
              nb1:=nb1+b1[i1]
            else
              nb1:=nb1-b1[i1];
        3 : if not odd(c1) then
              nb1:=nb1+b1[i1]
```

```
                else
                 nb1:=nb1-b1[i1];
      else
        begin
        writeln('Erro 7. Ficheiro de',
        ' teste estragado!! !');
        repetir;
        end;
      end;
    inc(cl);
    end;
  ler_tf;
  end;
end;



begin
normvideo;
checkbreak:=(false);
abrir_f;
abrir_tf;
ctf1:=filesize(f_tf);
procurar;
tratar;
verificar;
writeln('*nreg ',f1,' nfic ',r_f.nome_f,
        ' cfic ',r-f.comp_f,' nbfic ',
         r_f.cont_f,' *  fim XI *');
close (f_f);
close (f_tf);

swapvectors;
exec('c:\command.com','/c '+s1);
swapvectors;

end.
{**************************************}
```

```
{**************************************}

   {****** PROGRAMA AUTOTI ******}

program AUTOTI;

            { Autoteste. }

uses crt;

const

cbuf = 32768;
            { Comprimento do "buffer" de
              f_tf. }

ling = 'Desenvolvido em TURBO PASCAL '+
       '6.0  Copyright (c) Borland 1988';

s1 = 'AUTOTI.EXE';
```

```
comp_f : longint = 6656;

cont_f : longint = 650900;

s2 : string[12] = '12345678 0  ';
                  {123456789012}
```

{ Inicialmente em comp_f e cont_f está um número qualquer. Pode ser zero. Em s2 estará, por exemplo, '1234567890 '. Também pode ter outro conteúdo à escolha do utilizador. O programa é compilado e executado. Dá logo o valor real de comp_f e um valor aproximado de cont f. Põem-se esses valores em-comp_f e cont_f. Compila-se e executa-se outra vez o programa. Ajusta-se o valor de cont_f para um valor aproximado de nbl. O valor de cont_f será ajustado em s2 por substituição de caracteres ASCII. Se se substituir, por exemplo, '1' por ' ' subtrai-se 49-32 ou seja 16 a nbl. Se se substituir ' ' por '9' junta-se 57-32 ou seja 25 a nbl. Enfim, com um pouco de paciência obter-se-á o valor pretendido. }

```
var

 f_tf : file;
```

{ f tf ficheiro a testar a integridade. }

```
b1 : array [1..cbuf] of byte;
          { Para "buffer" de f_tf. }

tfl : word;
          { Para "blockread". }

i1 : longint;
          { Para ciclos. }

c1 : longint;
          { Para contar número de
       "bytes" (comprimento de f_tf). }

nbl : longint;
          { Para somar número de
       "bytes" de f_tf ou gerar qualquer
        outro algoritmo de teste. }

ctfl : longint;
         { Para comprimento de f_tf. }
```

```
procedure apito;
begin
sound(900);
delay(300);
nosound;
end;
```

```
procedure repetir;
begin
writeln('***FIM DE AUTOTI POR ERRO***');
repeat
  begin
  apito;
  delay(3000);
  end;
until keypressed;
            { Pode optar por fazer
"reset" do sistema pois alguma anomalia
existe com possibilidade de vírus. )

procedure resetsystem;
begin
inline ($ea/$5b/$e0/$00/$f0);
end;
            }
halt;
end;



procedure abrir_tf;
begin
 {$I-}
filemode:=0;
assign(f_tf,sl);
reset(f_tf,1);
 {$I+}
            { Os testes seguintes têm
razão de existir por poder ser alterado
o nome do programa. }
if ioresult<>0 then
  begin
  writeln
  ('Erro  3. Não existe o ficheiro ',sl)
  repetir;
```

```
  end;
if filesize(f_tf)=0 then
  begin
  write('Erro 2. Ficheiro ',s1,
        ' vazio!!!');
  repetir;
  end;
end;



procedure ler tf;
begin
blockread(f_tf,b1,cbuf,tf1);
end;



procedure verificar;
begin
if (comp_f=ctf1) and (comp_f=c1) and
(cont_f=nb1) then
  exit;
  writeln
  ('Erro 1. Integridade do ficheiro ',
  s1,' alterada!!!');
  writeln('    ctf1 ',ctf1,' c1 ',c1,
  ' nbl ',nb1);
  writeln(' *nfic ',s1,' cfic ',comp_f,
  ' nbfic ',cont_f,' *  fim AUTOTI *');
repetir;
end;
```

```
procedure tratar;
begin
c1:=0;
nbl:=0;
ler_tf;
while tf1 > 0 do
  begin
    for i1:=1 to tfl do
      begin
      nb1:=nb1+b1[i1];
      c1:=c1+1;
      end;
    ler_tf;
  end;
end;



procedure testar;
begin
abrir_tf;
ctf1:=filesize(f_tf);
tratar;
verificar;
close (f_tf);
   { A instrução seguinte é opcional. }
writeln('*nfic ',s1,' cfic ',comp_f,
   ' nbfic ',cont_f,' *   fim AUTOTI *');
end;
```

```
begin
testar;

writeln('Inicio do programa ',s1);
writeln('Instruções do programa ',s1);
writeln('.................');
writeln('.................');
writeln('.................');
writeln('.................');
writeln('.................');
writeln('Fim do programa ',s1);

end.
```

```
{**************************************}
```

****

123456*89012 *** **PROGRAMA AUTOTICB** ***

```
IDENTIFICATION DIVISION.
PROGRAM-ID. AUTOTICB.
AUTHOR. EDUARDO ABILIO DA SILVA.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FIC
    ASSIGN  "AUTOTICB.EXE"
    ORGANIZATION SEQUENTIAL
    ACCESS SEQUENTIAL.
```

```
DATA DIVISION.
FILE SECTION.
FD  FIC
    BLOCK 32768
    LABEL RECORD OMITTED.
01  R-FIC          PIC 9(2) COMP.

WORKING-STORAGE SECTION.
01  CONST-W        PIC X(20)
    VALUE '9999999999
#           12345678901234567890'.
01  COMPI-W        PIC 9(10)
    VALUE 40736.


01  CONTI-W        PIC 9(10)
    VALUE 4286200.
01  COMP-W         PIC 9(10)
    COMP VALUE 0.
01  CONT-W         PIC 9(10)
    COMP VALUE 0.


PROCEDURE DIVISION.
INIC.
    OPEN INPUT FIC.

LER.
    READ FIC        END GO FIM1.
    ADD R-FIC       TO CONT-W.
    ADD 1           TO COMP-W.
    GO LER.
```

```
FIM1.
    CLOSE FIC.
    IF COMP-W NOT = COMPI-W
       OR CONT-W NOT = CONTI-W
       DISPLAY 'Integridade  de '
       'AUTOTICB.EXE alterada!!!'
       DISPLAY COMP-W ' i '
       COMPI-W ' ' CONT-W ' i '
       CONTI-W
       GO FIM2.

INICIO.
    DISPLAY 'Inicio  do programa '
    'AUTOTICB.EXE'.
    DISPLAY 'Instruções do '
    'programa AUTOTICB.EXE.
    DISPLAY '.................'.
    DISPLAY '.................'.
    DISPLAY '.................'.
    DISPLAY '.................'.
    DISPLAY '.................'.
    DISPLAY 'Fim do programa '
    'AUTOTICB.EXE'.


FIM2.
       STOP RUN.
```

****

I N F O R M A Ç Õ E S

Os programas são expostos em Pascal (desenvolvidos em TURBO PASCAL 6.0 Copyright (c) Borland 1988), mas será fácil a qualquer um fazê-los noutra linguagem de programação. Se precisar de alguma ajuda pode contactar-me.

Se tiver tirado umas fotocopiazitas é porque o livro tem algum valor para si. Não se esqueça do autor e envie o que achar justo, ou peça um original, a

Eduardo Abílio da Silva
Rua Fernando de Oliveira, 2 - 2º direito
Santo António dos Cavaleiros
2670 Loures
PORTUGAL
Telefone (01)9883714

Preciso muito de dinheiro. Quem não precisa? Embora tenha o suficiente para mim não tenho o suficiente para os outros.

Eu vendia a minha alma ao Diabo, mas nem o diabo ma quer.

C O R R I G I R

procedure procura_b;

Onde está:	Deveria estar:

inc(inf)	begin
	inf:=f1;
	inc(inf);
	end

dec(sup)	begin
	sup:=f1;
	dec(sup);
	end

procedure repetir;

Onde está:	Deveria estar:

procedure repetir;
begin
	procedure repetir;
	var ch : char;
	begin
	while keypressed do
	ch:=readkey;

Í N D I C E