更完美的硬盘加锁程序

王咏武　（无锡市工商银行电脑处）

摘要:在本刊 1993 年第八期上登载的＜＜一种实用的硬盘双重加锁程序＞＞一文的基础上,更详尽地论述了微机的自举原理,介绍了一种更完美的硬盘加锁技术,并附有程序,使硬盘软加锁达到实用阶段。

本刊 1993 第八期上登载的＜＜双重加锁＞＞一文中介绍了一个硬盘双重加锁程序,很有实用性,但文中提出的两种解决办法存在缺陷:

第一,是修改 COMMAND.COM 程序,使它结束前能清除 DOS 标识符,这个办法是否能实现我还不能肯定,关机前是否要执行 EXIT 命令呢?而且对于一般用户来讲,这个办法就更不可能被采用了。

第二,是关机前执行文中提供的 PASSWORD.EXE 程序,这嗇了操作的麻烦,如果用户偶然忘记执行即关机,其它用户用软盘启动机器,可对硬盘随意操作,口令形同虚设。

本文提供的程序,成功地解决了上述问题,关机前不要执行任何程序,并可方便地增、删和修改口令。实现了用硬盘启动需回答口令,用软盘启动则不能对硬盘进行任何操作,成功地保护了硬盘上的数据,唯一的代价是占用了 IK 内存。

一、微机自举原理

1.DOS 磁盘组织

一个可启动的软盘必须具有以下要素:在 0 道 0 面 1 扇区(对应于逻辑 0 扇区)必须具有一个 BOOT 区,内含引导程序,随后是双备份的 FAT 表和根目录表,再其后是数据区,数据区上必须包含两个隐含文件(IBMBIO. COM 和 IBMDOS. COM)以及一个外壳程序 COMMAND.COM。

硬盘略有不同,在 0 柱 0 面 1 扇区是一主 BOOT 区,内含主引导程序,以及用以指明硬盘划分情况的分区表。随后是 16 个隐含扇区(DOS3.0 以上),未使用,由于本程序使用到了这些隐含扇区,因此本程序只适用于 DOS3.0 及其以上版本,这 17 个扇区 DOS 无法访问,在 DOS 分区的第一个扇区有一个和软盘类似的 BOOT 区(对应于逻辑 0 扇区),其后就和软盘类似,包括 FAT 表,根目录表和数据区。

2.软、硬盘自举原理

当系统复位或上电时,CPU 从地址 OFFFFOH 开始执 ROM-BIOS 中的自检引异程序,首先进行硬件自检,然后判断 A 驱动器门是否关上,如果确认关上,则读软盘 BOOT 程序至内存 0:7C00 处,并跳转至 0:7C00 处继续执行,BOOT 程序应在软盘根目录上寻找 IBMBIO.COM 和 IBMDOS.COM 两个隐含程序,并将它们调入内存,由它们继续完成系统初始化,最后 COMMAND.COM 被调入内存,显示 A＞提示符。至此,系统成功地启动了。

当 ROM 引导程序检测到软驱中无软盘,而硬盘已连接时,它就把硬盘主 BOOT 程序调至内存 0:7C00 处,并交出控制,主 BOOT 程序根据偏移 7DBEH 至 7DFDH 的分区表确定活动 DOS 分区的起始,终止位置以及大小信息,然后找到活动分区的 BOOT 程序,再将其读入内存 0:7C00 处,并把控制交给分区 BOOT 程序,分区 BOOT 程序继而在 DOS 分区上寻找 IBMBIO.COM 和 IBMDOS.COM 两个隐含文件,并调入内存,随后的启动过程便和软盘基本相同了。

值得注意的一点是,无论是软盘启动,还是硬盘启动,随后的启动过程中都会再去读硬盘 0 柱 0 面 1 扇区,利用其中的分区信息来形成硬盘物理扇区和逻辑扇区的对应关系。如果读到的 DOS 标识符为零,便会认为硬盘上无 DOS 分区。因此利用这一点,使软盘启动后访问到的

分区信息为全零。而硬盘启动后访问到的分区信息是正常信息,这样就形成了下述的加锁原理。

二、加锁原理

本程序利用了硬盘上的隐含扇区 0 柱 0 面 2 扇区,把 0 柱 0 面 1 扇区的主 BOOT 程序搬至第二扇,1 扇区写入我所编写的新的引导程序 BOOT. ASM, 其中对应分区表的偏移 1BE～1FD 处全部置 0。这样当从软盘启动时,系统读到的硬盘分区信息为全零, 会认为无 DOS 分区,从而无法使用硬盘,但硬盘启动后系统同样也读不到正常的分区信息了,这个问题又如何解决呢?

＜＜双重加锁＞＞一文中是采用确认口令后再将正确信息写回第一扇, 这就造成了上述提到的缺陷 ,即关机前必须再执行程序抹掉正确信息。

我编写的主引异程序是这样解决的:驻留内存,链接 INTB 中断向量,新的 INTB 程序暗中把系统读 1 扇区的功能调用改成读 2 扇区,即可读到原先保留在内的正确分区表。

具体实现如下:确认口令正确后,读内存 0:413 中的一个字,其中存放的是系统基本内存大小,以 K 为单位。将其减 1 后再放回 0:413,系统内存将减小 1K, 系统将不会再分配最后 1K 内存,然后把自已搬移至内存高端最后 1K 内存,更改 INTB 中断向量, 使之指向新 INTB 程序,然后将读入硬盘 0 柱 0 面 2 扇区真正的引导程序,开始启动系统。以后, 当系统去读 1 扇区分区信息时,链入的 INTB 程序将获得控制,它先判断是否是读硬盘 0 柱 0 面 1 扇区,若是,则改成读 0 柱 0 面 2 扇区,然后转原 INTB 处理,不是则直接转原 INTB 处理,这样,系统将被欺骗,读出 2 扇区中的正确分区信息,确认 DOS 分区后,硬盘正常启动。

三、具体程序

附上 BOOT.ASM,DISKPS.C,SEC-RD.C 三个程序,需在 BORLANDC++或低版本的 TURBOC 上用巨型模式编译连接,本程序在 BC++环境下编译通过,在几台长城机、紫金机上试运行正常。加锁后,PCTOOLS8.0,CPAV,NORTON 等需读主 BOOT 扇区的软件仍工作正常。

DISKPS.C 包括 MAIN 主模块,它能判断硬盘是否已加锁,如果已加锁,当前是从 C 盘启动还是从 A 盘启动的,将根据不同情况作不同处理,实现增、删、修改口令。

SEC-RD.C 包括 SEC-RD()函数,该函数根据入口参数实现硬盘某个扇区的读写。

BOOT.ASM 包括一个远程数据段,即主引导程序,供 MAIN 函数把它写入 0 柱 0 面 1 扇区。

执行只需不带参数的 DISKPS 即可。

四、结束语

本程序是第一版,还有不少地方仍需改进,列出如下以供大家参考。

1.本程序启动时,只判断一次口令,错误即死机, 应增加判断次数至两次到三次比较合理。

2.删除和改变口令前,本程序未确认原口令,因此本程序不能流传。可增加相应程序段来增强功能。

3.口令未加密,采用 ASCII 码形式直接写入 0 柱 0 面 1 扇区,有经验的人员读出 1 扇即可看到口令,应相应增加加密程序段,而在启动程序 BOOT.ASM 中增加解密程序段。

程序 1:

```
#include＜stdio.h＞        / * DISKPS.C * /
#include＜dos.h＞
#define READ 0
#define WRITE 1
extern char boot;
extern char message[];
extern char passwd[21];
char password[21];
char password1[21];
char buf[0x200];
main()
{
  int i,no  d＝0;
  char * s;
  unsigned int13cs, * mem;
  printf("Disk Password System 1.0!＼nCopywrite by wang
```

```
yong wu!\n\n");
  if (sec rd(buf,0x81,1,READ)==-1)
brintf("No d:driver!\n\n");no d=1;} * no d=1 读 D 盘错
*
i f(sec rd(buf,0x80,1,READ)==-1)
Printf("No c:driver!\n\n");exit(-1);}  * 读 C 盘错,退出 *
m mem=MK FP(0,0x413);
i nt13cs=(*mem<<6)-0x7c0;  * 求最后 1K 段址 *
i f(!strcmp(MK FP(int13cs,FP OFF(message)),message))
  / * 判断是否已驻留内 * /
{printf("Disk system is in 0 sector....\n\n");
    printf("    1.del password\n");
    printf("    2.change password\n");
    printf("    0.exit\n\n");
  printf("Please input chioce:");      * 显示菜单 *
  switch(getche())
    {case'0':break;
     case'1':sec rd(buf,0x80,1,READ);sec rd(buf,0x80,1,
          WRITE); * 读出第二扇,写回第一扇,删除 *
            if(!on d)
          {sec rd(buf,0x81,1,READ);sec rd(buf,0x81,1
          ,WRITE);} * D 盘存在,对 D 盘操作 *
            break;
     case'2':while(password[0]==0)
       {printf("\nPlease input password(1-20char):");
       gets(password);
       printf("\nPlease re input password(1-20char):");
       gets(passwordl);
       if(strcmp(password,passwordl))password[0]=0;
              }
          for(i=0;i<=20;i++)passwd[i]=password[i];
        sec rd(boot,0x80,1,WRITE);
          break;
    }
  }
else if(!strcmp(buf+FP OFF(message) -0x7c00 ,message))
 * 未驻留内存情况下,判断是否已加口令 *
  {printf("Disk system is in 0 sector....\n\n");  * 即是由
     软盘启动的情况 *
   printf(" 1.del password\n");
   printf(" 2.change password\n");
   printf(" 0.exit\n\n");
   printf("Please input chioce:");
   switch(getche())
     {case'0':break;
      case'1':sec rd(buf,0x80,2,READ);sec rd(buf,0x80,1,
          WRITE);  * 读第二扇,写回每一扇,删除 *
            if(!on d)
          {sec rd(buf,0x81,2,READ);sec rd(buf,0x81,1
          ,WRITE);}
            break;
      case'2':while(password[0]==0)
        {printf("\nPlease input password(1-20char):");
        gets(password);
        printf("\nPlease re input password(1-20char):");
        gets(passwordl);
        if(strcmp(password,passwordl))password[0]=0;
        }
      for(i=0;i<=20;i++)passwd[i]=password[i];
      sec rd(boot,0x80,1,WRITE);
        break;
      }
  }
else
  {printf("    1.add password\n");* 未加过口令情况 *
    printf("    0.exit\n\n");
    printf("Please input chioce:");
    switch(getche())   * 显示菜单 *
   {case'0':break;
  case'1':sec rd(buf,0x80,2,READ); * 读 C 盘第一扇 *
          while(password[0]==0
    {printf("\nPlease input password(1-20char):");
     gets(password);
   printf("\nPlease re input password(1-20char):");
     gets(passwordl);
     if(strcmp(password,passwordl))password[0]=0;
    }
   for(i=0;i<=20;i++)passwd[i]=password[i];
   sec rd(but,0x80,2,WRITE);   * 写入第二扇 *
   sec rd(&boot,0x80,1,WRITE);   * 写远程数据段至
   C 盘第一扇 *
   if (!no d)   * D 盘存在 *
```

```
        sec rd(but,0x81,1,READ);
        sec rd(but,0x81,2,WRITE);    * D 盘第一扇写入
      第二扇 *
        for(i=0;i<0x200;i++)buf[i]=0;
        sec rd(but,0x81,1,WRITE);   * 全零写入 D 盘第
一扇,完成增口令 *
          }
      }
    }
}
```

程序 2:

```
cr        equ 0dh ;BOOT,ASM
if        equ 0ah ;回车和换行
          public  boot, passwd, message
FARDATA segment word public'FAR DATA'      ;远 程
          数据段
          assume cs; FARDATA,DS: FARDATA
          org 7c00h
boot:     xor ax,ax
          mov ds,ax
          mov es,ax
          cli
          mov ss,ax
          mov sp,07c00h
          sti
          mov bx,offset  message
display:  mov al,[bx]
          cmp al,0
          jz   input
          mov ah,0eh      ;oh表示显示结束
          int 10h
          inc bx      ;显示版本信息
          cmp  display      ;及输入口令提示
input:    mov bx,offset  passwd
input 1   1:mov ah,00
          int 16h      ;输入口令
          cmp al,cr
          jz   cmpend      ;按回车后停止比较
          cmp aal,[bx]
          jz   next1
          mov  err,0ffh      ;和原口令比较
nest1:    mov al,'X'      ; err=OFFh口令错误
          mov ah, 0eh
          int 10h
          inc bx      ;显示'X'
          cmp bx,offset  passend
          jz   error
          cmp  input1      ;输入超过二十个字符,错
cmpend:mmov al,[bx]
          cmp al,0      ;按回车
          jnz  error      ;判断原口令是否也结束
          mov al, err      ;未结束,错,转错误处理
          cmp al,0
          jnz  error
right:    mov bx,offset  rightm      ; err不等于0口令有错
loop:     mov al,[bx]
          cmp al,0
          jz   nest2
          mov ah,0eh
          int 10h
          inc bx      ;显示口令正确信息
          jmp  loop
next2:    mov ax,es:[413h]      ;取内存大小
          dec ax ;减1K
          mov es:[413h],ax      ;放回413
          sh1 ax ,cl
          sub ax,7c0h
          mov  newcs,ax      ;求最后1K段址
          mov es,ax
          mov di,7c00h
          mov si,di
          mov cx,200h
          cld
          rep movsb      ;自身搬移至最后1K
          jmp dword ptr  newip      ;跳到最后1K继续执行
conti:    :push es
          pop ds
          xor ax,ax
          mov es,ax
          mov ax,es;[13h * 4]
```

```
        mov  old13ip,ax
        mov ax,es:[13h * 4+2]
        mov  old13cs,ax        ;保存原INT13中断向量
        cli
        mov ax,offset  int13
        mov es:[13h * 4],ax
        mov ax,cs
        mov es:[13h * 4+2],as        ;置新INT13中断向量
        sti        ;使之指向 INT13
        mov ax,0201h
        mov bx,7c00h
        mov cx,1h
        mov dx,80h       ;读第1扇,实际是
        int 13h          ;读第二扇
        jmp dword ptr  j07c00    ;控制交真正主引导程序
error:   mov bx,offset  errm
loop1:   mov alc[bx]
        cmp sl ,0
        jz   here
        mov ah,0eh
        int 10h          ;显示错误信息
        inc bx
        jmp  loop1
here:    jmp  here
int13:   cmp ah,02h       ;死循环
        jnz  exit       ;新INT13中断
        cmp dx,80h      ;是否读
        jz   next
        cmp dx,81h       ;是否读C或D盘
        jnz  exit
next:    cmp cx,1h        ;是否读C或D盘0柱0面1扇
        jnz  exit
        mov cx,2h        ;是改成读2扇
exit:    jmp dword ptr cs:  old13ip        ;转原INT13处理
message  db  cr,lf,cr,lf,'Disk Password System 1.0'
        db  cr,lf,'Copywrite by wang yong wu'
        db  cr,lf,cr,lf,;Please input password:',00h
rightm   db  cr,lf,cr,lf,'Right!!',cr,lf,00h
errm     db  cr,lf,cr,lf,'Error!!',cr,lf,00h
passwd   db  'wang-system'       ;口令
         db  10 dup(0)
passend  equ ¥
newip    dw  offset  conti
newis    dw  0
j07c00   dw  7c00h
         dw  0h
err      db  0
old13ip  dw  ?
old13cs  dw ?
         org 7c00h+lfeh      ;中间包括分区表初始化成零
         db  55h,0aah        ;主引导记录标志
FARDATAends
         end
```

程序 3:

```
#include<stdio.h>    * SEC  RD.C *
#include<dos.h>
#define READ 0
#define WRITE 1
sec  rd(char * bufrd,int hd,int sec,int foag)
{
    struct SREGS wseg;
    union rEGS wreg;
    segread(&wseg);
    wseg.es=FP  SEG(bufrd); * bufrd 是读写缓冲区指针 *
    switch(flag)   * flag 读写标志 *
       {case READ:wreg.x.ax=0x0201;
                  break;
        case WRITE:wreg.x.ax=0x0301;
                  break;
        }
    wseg.x.bx=FP  OFF(bufrd);
    wseg.x.cx=sec;    * SEC 读写的扇区号 *
    wseg.x.dx=hd;     * hd:0x80 是 C 盘 0X81 是 D 盘 *
    int86x(0x13,&wreg,&wreg,&wseg);
    if (wreg.x.cflag)return-1;    * 进位标志若置位,有错 *
}
```