利用微机磁盘划分结构恢复主引导扇区

张振滨　　(辽阳市辽化信息中心)

近几年来,经常迂到由于病毒的发作破坏了主引导扇区,导致计算机不能引导启动的情况。从一些杂志介绍这类问题的解决方法上看,一是将过去备份的主引导扇区用中断INT13H写回硬盘;二是找一台同机型、同磁盘类型的好机器,用INT13H将好机器上的主引导扇区通过软盘写到坏机器的主引导扇区位置上。但是,这两种方法存在着问题,一是备份主导扇区的软盘可能被损坏,二是在周围找不到同类的机器,即使找到了,而磁盘分区可能不相同,仍然无法进行恢复。于是,就提出一个问题:在没有外界条件下,能否利用被破坏的计算机本身磁盘结构特征来推导出一个和原来一样的主引导扇区?笔者通过对十几种型号、二十余台计算机的实际分析对比,找到了解决这一问题的方法,并编写了程序。

一、基本原理

任何一台微机的主引导扇区都是由引导记录和磁盘分区表两部份组成的。不同机器的引导记录都相同,而磁盘分区表却不同。因此,我们可以把恢复主引导扇区归为恢复磁盘分区表。由于磁盘分区表与该磁盘划分的结构有着密切的关系,因此,我们只要对磁盘划分的结构(不能被病毒破坏)进行分析,即搞清楚磁盘是如何划分的,再通过这种关系,便可反推导出原来的磁盘分区表,从而生成一个和原来一样的主引导扇区,这就是解决这一问题的基本原理。

二、方法和步骤

先来分析一下主引导扇区的结构及其内容含义(如图1)。

```
2865:0000  FA 33 C0 8E D0 BC 00 7C-8B F4 50 07 50 1A FB FC  .3.......P.P...
  * * *                      *  *  *                          * * *
2865:00D0  69 6E 67 20 73 79 73 74-65 6D 00 00 00 00 00 00  ing system......
  * * *                      *  *  *                          * * *
            起始柱面号(03H)                    自举标志(00H)
起始扇区(02H)   DOS系统标志(04H)  可用扇区数(0CH～0FH)   起始磁头号(01H)

2865:01B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 80 01  ................
2865:01C0  01 00 01 03 1A B3 1A 00-00 00 06 49 00 00 00 00  ...........I....
2865:01D0  01 B4 05 03 DA 23 20 49-00 00 80 FD 00 00 00 00  .....# I........
2865:01B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
2865:01B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 55 AA  ..............U.

 终止磁头号(05H)               相对扇区号(08H～0BH)
      终止扇区号(06H)   终止柱面号(07H)     引导记录扇区有效标志
```

图1　主引导扇区映象

从图 1 中可以看到,该扇区从 0000H 到 00D9H 为主引导记录,而从 01BEH 到 01FDH 的 64 个字节为磁盘分区表。通常对单一 DOS 引导的操作系统来说,其磁盘分区表只需前 32 个字节。前 16 字节是主 DOS 分区记录,后 16 字节是扩展 DOS 分区记录。每条记录结构含义都一样。

自举标志字节(00H)内容对主 DOS 分区记录为 80H,对扩展 DOS 分区记录为 00H。

起始磁头号(01H)的取值,对主 DOS 分区是 01H,对扩展 DOS 分区是 00H。

起始扇区号(02H 字节的低 6 位)的取值均为 01H。

起始柱面号(03H 字节为低 8 位,02H 字节的高 2 位为该柱面号的高 2 位)的取值,对主 DOS 分区来说为 0H,对扩展 DOS 分区来说为主 DOS 分区终止柱面号加 1。

系统标志字节(04H)的取值内容,对主 DOS 分区记录而言,是由该分区的可用扇区数(0CH～0FH 双字)的大小决定的。如果可用扇区数大于 FFFFH 时,取值为 06H,否则,当该分区的总柱面数(终止柱面号加 1)小于等于 1/2 最大柱面数(所含扇区总数小于等于 FFFFH 的,且为最大的那个柱面数)时,该系统标志字节取值为 01H,反之为 04H。系统标志字节对扩展 DOS 分区而言取值均为 05H。

分区的终止磁头号(05H)的取值均为总磁头数减 1。

分区的终止扇区号(06H 字节的低 6 位)的取值均为每磁道的扇区数。

分区的终止柱面号(07H 字节为低 8 位,06H 字节的高 2 位为该柱面号的高 2 位)的取值,对主 DOS 分区来说为待寻求的数,对扩展 DOS 分区来说为磁盘的总柱面数。

相对分区号(08H～0BH)取值,对主 DOS 分区是每磁道扇区数(从 CMOS 中得到),对扩展 DOS 分区则是主 DOS 分区中的可用扇区数与每磁道的扇区数之和。

可用扇区数(0CH～0FH),对主 DOS 分区来说,它等于该分区的柱面数(该分区的终止柱面号加 1)与总磁头数(从 CMOS 中得到)和每磁道扇区数之积后,再减去每磁道扇区数。就是说知道了主 DOS 分区的终止柱面号即可算得。对扩展 DOS 分区来说,它等于磁盘总柱面数(从 CMOS 中得到)减去主 DOS 分区的终止柱面号再与总磁头数和每磁道扇区数之积。

从上述分析的结果可以看出,只要能够找到主 DOS 分区的终止柱面号或扩展 DOS 分区的起始柱面号,就可以通过上述各部分取值与之存在的关系得到一份磁盘分区表。为找到扩展 DOS 分区区域的第一个扇区,我们将主面号从 1 开始逐个递增,同时对每个柱面的 0H 号磁头,第一扇区进行比较。比较的内容是 1FEH 字的内容与 AA55H 值相等,同时 1C0H 字的内容与当前该扇区所在的柱面、扇区值相等。如果同时满足了这两个条件,而且是第一次满足,则当前的柱面号即为所要寻找的扩展 DOS 分区的起始柱面号。于是,原磁盘分区表就可以推导出来。

如果遍历所有磁盘柱面仍没有同时满足上述两个条件,则说明该磁盘根本就没有扩展 DOS 分区。全部磁盘空间均为主 DOS 分区。这种磁盘分区表很容易推导。

三、软件实现

从上述介绍的恢复主引导扇区的方法和步骤来看,用手工来推导是很困难的,而且容易出错。因此,笔者用汇编语言将其编写成 RECABOOT.ASM,并编译为 RECABOOT.EXE 来运行,最长时间不超过 30 秒钟,效率是相当高的。RECABOOT.ASM 程序清单附后。

四、结束语

用本文介绍的方法编成的 RECABOOT.ASM 程序,无论是在什么类型的微机上,无论是被什么样的引导型病毒破坏的,只要是 DOS 操作系统,其主引导扇区均可恢复。而且,用户机器内原有的任何文件和数据都保持不变。为了考核本程序,在辽化公司十几个单位机器的主引导扇区遭病毒破坏时,发挥了其应有的作用,得到了用户的肯定。

RECABOOT.ASM清单

```
    name creaboot
        title 'creat boot.'
cseg    segment para public'CODE'
        assume cs:cseg,ds:data,es:data,ss:stack
RECABOOT proc far
        push ds
        xor ax,ax
        mov ax,data
        mov es,ax
        mov ds,ax
recaboot1:mov ah,08h      ;读CMOS
        and ah,0fh
        mov dl,80h
        int 13h
        jc recaboot4
        mov eendc n,cx
        mov al,cl
        and al,3fh
        mov sect n,al
        mov al,ch
        rol cx,1
        rol cx,1
        mov ah,ch
        and ah,03
        inc ax
        mov cyl n,ax
        mov head n,dh
        call LOCABOOT
        mov ah,ebbz
        cmp ah,0h
        jnz recaboot4
        call LOCAEB
        jmp recaboot3
recaboot3:mov ax,0301h
        mov bx,offset boot buffer
        mov cx,0001h
        mov dx,0080h
        int 13h
        pop ds
        mov ax,4c00h
        int 21h
recaboot4:mov dx,offset msg2
        mov cx,msg2 length
        mov bx,2h
        mov ah,40h
        int 21h
        pop ds
        mov ax,4c01h
        int 21h
RECABOOT endp
LOCABOOT proc near
        xor ax,ax
        mov ccyl n,ax
local:  mov ax,ccyl n     ;找扩展DOS分区的起始柱面号
        inc ax
        mov ccyl n,ax
        mov ch,1
        call ROL6
        mov cx,ax
        call READEB
        jc loca5
        mov ax,exboot+1feh
        cmp ax,0aa55h
        jnz local
        cmp cx,exboot+1c0h
        jnz local
        mov ebbz,00
        jmp loca2
loca5:  mov ebbz,01
        clc
loca2:  ret
LOCABOOT endp
LOCAEB proc near          ;推导新的磁盘分区记录
        mov ax,ccyl n
        dec ax
        mov ch,sect n
        call ROL6
        mov boot buffer+1c4h,ax
        mov al,sect n
        mov mc6,al
        mov al,head n
        mov mc3,al
        mov ax,exboot+1c0h
        mov boot buffer+1d0h,ax
        mov md2,05h
        call CMULS
        mov boot buffer+1d6h,ax
        mov boot buffer+1d8h,dx
        mov cl,sect n
        sub ch,ch
        sub ax,cx
        jnc locaeb3
        dec dx
locaeb3:mov boot buffer+1cah,ax
        mov boot buffer+1cch,dx
        cmp dx,0
        jz locaeb4
        mov mc2,06h
        jmp locaeb5
locaeb4:call GET MAXS
        call SYS BZ
locaeb5:mov ax,exboot+1cah
        add boot buffer+1dah,ax
        jnc locaeb6
        inc boot buffer+1dch
        clc
locaeb6:mov ax,exboot+1cch
        add boot buffer+1dch,ax
        xor ah,ah
        mov al,sect n
        add boot buffer+1dah,ax
        jnz locaeb7
        inc boot buffer+1dch
        clc
```

```
locaeb7:mov cx,exboot+1d0h
        cmp cx,0
        jz locaeb8
        call READEB
        jnc locaeb5
locaeb8:mov ax,exboot+1c3h
        mov md3,al
        mov ax,exboot+1c4h
        mov boot_buffer+1d4h,ax
        ret
LOCAEB endp
READEB proc near          ;读一个扇区
        mov ax,0201h
        mov bx,offset exboot
        mov dx,0080h
        int 13h
        ret
READEB endp
ROL6 proc near
        mov cl,6h
        rol ah,cl
        add ah,ch
        xchg ah,al
        ret
ROL6 endp
CMULS proc near           ;求扇区数
        mov ax,ccyl_n
        mov cl,head_n
        inc cl
        sub ch,ch
        mul cx
        mov cl,sect_n
        mul cx
        ret
CMULS endp
SYS_BZ proc near          ;求系统标志
        mov ax,maxc_n
        mov dx,0h
        mov cx,02h
        div cx
        cmp dx,0h
        jz sb1
        inc ax
sb1:    cmp ccyl_n,ax
        jb sb2
        mov mc2,04h
        jmp sb3
sb2:    mov mc2,01h
sb3:    ret
SYS_BZ endp
GET_MAXS proc near        ;求<=0FFFFH的最大扇区数
        xor dx,dx
        xor cx,cx
        xor ah,ah
        mov al,head_n
        inc al
        mul sect_n
        mov hms_n,ax
        mov ax,0ffffh
        mov cx,hms_n
        div cx
        mov maxc_n,ax
        mul cx
        mov maxs_n,ax
        ret
GET_MAXS endp
cseg ends
data segment para public'DATA'
    maxc_n dw 1 dup(0)
    maxs_n dw 1 dup(0)
    ccyl_n dw 1 dup(0)
    head_n db 1 dup(0)
    sect_n db 1 dup(0)
    hms_n dw 1 dup(0)
    boot_buffer           ;引导记录
    db FA,33,C0,8E,D0,BC,00,7C,8B,F4,50,07,50,1F,FB,FC
    db BF,00,06,B9,00,01,F2,A5,EA,1D,06,00,00,BE,BE,07
    db B3,04,80,3C,80,74,0E,80,3C,00,75,1C,83,C6,10,FE
    db CB,75,EF,CD,18,8B,14,8B,4C,02,8B,EE,83,C6,10,FE
    db CB,74,1A,80,3C,00,74,F4,BE,8B,06,AC,3C,00,74,0B
    db 56,8B,07,00,B4,0E,CD,10,5E,EB,F0,EB,FE,BF,05,00
    db BB,00,7C,B8,01,02,57,CD,13,5F,73,0C,33,C0,CD,13
    db 4F,75,ED,BE,A3,06,EB,D3,BE,C2,06,BF,FE,7D,81,3D
    db 55,AA,75,C7,8B,F5,EA,00,7C,00,00,49,6E,76,61,6C
    db 69,64,20,70,61,72,74,69,74,69,6F,6E,20,74,61,62
    db 6C,65,00,45,72,72,6F,72,20,6C,6F,61,64,69,6E,67
    db 20,6F,70,65,72,61,74,69,6E,67,20,73,79,73,74,65
    db 6D,00,4D,69,73,73,69,6E,67,20,6F,70,65,72,61,74
    db 69,6E,67,20,73,79,73,74,65,6D
    db 0e4h dup(?)
    db 80,01              ;磁盘分区表
    mc0 dw 1 dup(?)
    mc2 db 1 dup(?)
    mc3 db 1 dup(?)
    mc4 dw 1 dup(?)
    mc6 db 2 dup(?)
    mc8 dw 4 dup(?)
    md0 dw 1 dup(?)
    md2 db 1 dup(?)
    md3 db 1 dup(?)
    md4 dw 6 dup(?)
    me0 db 1Eh dup(?)
    db 55,AA
    exboot dw 256 dup(?)
    cends_n db 1 dup(0)
    chead_n db 1 dup(0)
    ccyl_n dw 1 dup(0)
    eendc_n dw 1 dup(0)
    ebbz db 1 dup(0)
    msg2      db 0dh,0ah,'Missing parameters.',0dh,0ah
    msg2_length equ $-msg2
    msg11     db 0dh,0ah,'Recall mainboot is success.',0dh,0ah
    msg11_length equ $-msg11
data ends
stack segment para stack'STACK'
      db 64 dup(?)
stack ends
      end     recaboot
```