PC Computing
电子&电脑

1995年合订本（上）

《电子&电脑》编辑部 编

電子工業出版社

电子&电脑

1995年合订本（上）

《电子与电脑》编辑部 编

電子工業出版社

内 容 简 介

本刊是一本享誉海内外的电脑月刊，她荟萃《PC/Computing》、《Windows Sources》、《PC Week》等世界著名电脑刊物的精华，以及国内计算机工作者撰写的大量文章。她融信息与技术于一体，信息量大，实用性强。

合订本汇集了本刊第1～4期的全部技术内容，删除了广告和实用性不强的部分。主要栏目有："专题报导"(PC/Computing评出的94年最有价值的产品；Windows超级指南，移动计算环境——笔记本机与桌面系统和网络的连接、笔记本机的发展趋势、对13种笔记本机的测试报告、便携机的选购与配置，Windows NT Server3.5，Windows桌面出版系统；"产品评测"；"实践与技巧"；"软件之窗"；"网络天地"；"开发与应用"；"技术讲座"（Word 6.0 for Windows的工作环境、Word 6.0文档的录入与编辑、Word 6.0的图表与页面设置，软件的加密和解密技术之三、四、五）；"新品世界"——1994年世界上推出的各种软件、网络、外设、多媒体等最新产品。该合订本内容翔实，具有长期参考和保存价值，适合于广大计算机工作者阅读。

电子&电脑

1995年合订本（上）

《电子&电脑》编辑部　编

*

电子工业出版社出版

电子工业出版社发行　各地新华书店经销

中国科学院印刷厂印刷

北京富国电子信息有限公司排版

开本：787×1092毫米1/16　印张：30.5　字数：736千字

1995年5月第1版　1995年5月第1次印刷

印数：8000 册　定价：30.00 元

刊号：ISSN 1000-1077
CN11-2199/TN

目　录

• 专题 •

·实践与技巧·

·技术讲座·

·开发与应用·

·新品世界·

软件的加密&解密技术 之一

杨道沅

[编者按]从本期开始，本刊特请著名加密技术专家北京化工大学计算机系杨道沅教授开办“软件的加密与解密技术”讲座，讲解加密原理、磁盘结构、指纹制作技术、跟踪与反跟踪技术、解密技术，并剖析目前流传广、影响大的几种加密系统。本讲座采用连载的形式，预计连续刊登六期。

第一章 绪论

我国计算机是在 1985 年开始有较大发展的。当时，苹果机（Apple II）大量上市，随之而来带入了丰富的软件，用户在此基础上，进行各种软件开发。为了捍卫开发者的利益，软件的加密解密问题自然就提到了议事日程上。

典型的初期的Apple II被加密软件有由李存庆研制的CPDBASE，采用了额外磁道的加密方法，这对于刚刚接触到计算机的普通中国人来说，是一个了不起的开始。

由于用户可以编程直接控制磁头的动作，所以 Apple II 机 COPY 软件非常丰富，功能强大的 COPY 软件有 LOCKSMITH、BACK IT UP、CRAZY COPY、NIBBLE AWAY.....所以，我国初期的 Apple II 加密软件没有一个可以挡得住各类 COPY 软件的攻击。

1986年，IBM PC 机大量涌入中国市场， 电子部六所在此基础开发了CCDOS 2.0， 为PC 机在我国扎根打下了坚实的基础。随之而来的还是软件开发，软件加密问题。

最初进入我国的加密系统是PROLOK.EXE 激光加密盘和数据加密系统 FILE LOCK 激光加密盘。中国人怀着神秘的心情来跟踪它，分析它，经历了一次又一次的失败，终于敲开了它的大门。其中，国防科工委指挥技术学院研制了多种 PROLOK 加密盘的解密软件，北京信通公司也推出了改进后的激光加密盘，上海、河北也推出了类似的加密系统。技术人员从中学习到了许多反跟踪的知识、激光指纹的特征。这是我国加密解密技术的启蒙学习，在目前流行的许多加密系统中，还能看到它的影子。

以后，又从国外传入了PROTECT加密盘，我们惊讶地看到了它在加密过程中熟练地利用了 DOS 的各种功能和使用技巧，使我们大开眼界。

与此同时，我国的加密技术也从仿制过渡到独立开发新产品的阶段，由科海公司张汉亭先生推出的“虎符”加密盘取得了良好的销售效果。周志农先生对他的自然码进行加密，充分利用了DOS与DEBUG.COM传递给运行程序环境的不同来进行反跟踪。本人开发的、由信通公司代理的 LOCK89加密系统，吸收了上述各加密系统的优点，并把密码运算放在DOS的向量区0:0至0:47FH 区，在当时取得了良好的社会效果。

同时进行的指纹研究也取得了长足的进步。一开始，人们只是模仿激光孔法，电子部六所和河北某高校都先后掌握了激光孔指纹制作法。接着，人们从解密dBASE III 中学会了隐含 ID 法。由夏东涛先生提

出了“磁道间隙”法，周志农先生研制了用磁铁制作“弱位法”。有趣的是，这种方法生成的弱位区和正常区之间没有明确的界限，存在一段模糊区，我们无法复制它。以后，各种指纹制作方法都相继问世。同时，人们也把目光投向了硬指纹的制作，纷纷推出加密卡、软件狗。继而，由金盾公司推出了带有 CPU 的加密卡。

在1990年前后的几年里，人们开始注意加密系统的人机界面，用C语言和汇编语言混合编程的加密系统相继出现，大都采用下拉式菜单，使用户感到友好亲切，同时增加了加密系统本身被解密的难度。

到1991年，SOFT-ICE 调试软件传入我国，它充分利用 386 微机的硬件优势，以保护模式的零级来跟踪运行在虚拟8086模式的被加密软件。各种加密系统都纷纷落马，几乎无一幸免，这真是加密界的一次地震级的大灾难。

但是，人们很快从惊异中苏醒过来，跟踪它、分析它、了解它的堆栈使用过程，分析了它必需走过的轨迹，从而找出对抗它的方法。目前流行的 LOCK93、BITLOCK、KEYMAKE等加密系统，都能有效地对抗 SOFT-ICE 的跟踪。但是，当这些加密系统的研制者集中力量在虚拟8086模式对抗SOFT-ICE时，却忽略了对抗工作在实地址方式下的跟踪软件。所以，在实地址方式下去解密这些加密系统，反而是一件容易的事。

伴随着解密技术的日新月异发展，专用加密软件显示出越来越强的重要性。专用加密软件是相对通用加密软件而言的，用户在市场上买到的各种加密软件、加密卡、软件狗都属于通用加密软件，它们的共同特点是对被加密软件一无所知，只能对被加密软件加一层外壳。这样，外壳和被加密软件之间就有一条明确的分界，解密专家能找到这条分界线，剥掉外壳，解开程序。专用加密系统则不同，它由加密专家和用户互相合作，把加密程序穿插在用户的程序中，融合为一体，没有任何界限，使解密专家望而生畏。

加密与解密的斗争从软件的商品化开始，必将伴随着商品化的进程日趋激烈。没有解不开的加密系统，也没有难不倒的解密跟踪工具。一个好的加密系统，只要能顶住解密专家一年的进攻，就是一项了不起的成绩，因为到此时，被加密软件已到了版本升级的时候了。

加密与解密的斗争正向纵深发展.....

第二章 与加密解密技术相关的 DOS 知识

从事加密解密的工作几乎用到了 DOS 的所有知识，大部分知识对许多人来说是陌生的，由于篇幅的限制，这里只能列出最主要的也是关系最密切的几种。

（1）.EXE 文件的结构

当一个源程序被编译、链接后，都生成.EXE文件，每个.EXE文件由文件头和紧随其后的文件体组成。

文件头部包含文件的控制和再定位信息，其结构如下：

16进制位移　本字（word）的含义

00-01　4DH，5AH 表示本文件是一个.EXE文件的标志
02-03　文件体mod 512（即文件长度除512后的剩余部份）
04-05　文件页数=>（文件头+文件体）/512后的商
06-07　再定位表项目数，也是文件体中须再定位的字（word）数，它们是一一对应的
08-09　用16个字节长为一节来表示的文件头的长度
0A-0B　被装入程序的尾部以上需要的最小节数（16个字节为一节）
0C-0D　被装入程序的尾部以上需要的最大节数（16个字节为一节）
0E-0F　栈段寄存器SS的位移（以节为单位）
10-11　程序开始运行时SP的值
12-13　文件字（word）的检查和
14-15　程序开始运行时IP的值
16-17　指令段寄存器CS的位移（以节为单位）
18-19　文件第一个再定位项的位移（此位移从文件头的0字节开始计算）
1A-1B　覆盖号（对程序的常驻部分，为零）

```
DS:0000 4D 5A 8F 01 24 00 05  00-20 00  34 03 34 03 8C 04
DS:0010 00 30 1E 6E 70 14 10  00-1E 00  00 00 01 00 0C 00
DS:0020 49 04 7E 14 00 00 57  15-00 00  00 00 6E 03 D5 09
DS:0030 73 03 00 00 00 00 00  00-00 00  00 00 00 00 00 00
DS:0040 00 00 00 00 00 00 00  00-00 00  00 00 00 00 00 00
```

图2.1程序头部例（下文的 file_head => ds：0000）

由图2.1可知，从 02 至 05 可得，本程序全长=

（024H*512）+018FH。

从 06-07 可得， 有 5 个再定位项。

从18-19可得，再定位项从 01EH开始，每两个字（word）形成一个地址指针，指向程序体中需要再定位的字（word）。

从08-09可得，程序头部长=020H*16。

从10-11知，程序开始时SP =3000H。

从14-15知，程序开始时IP =1470H。

从图2.2可见 0E-0F，16-17 字节的含义：

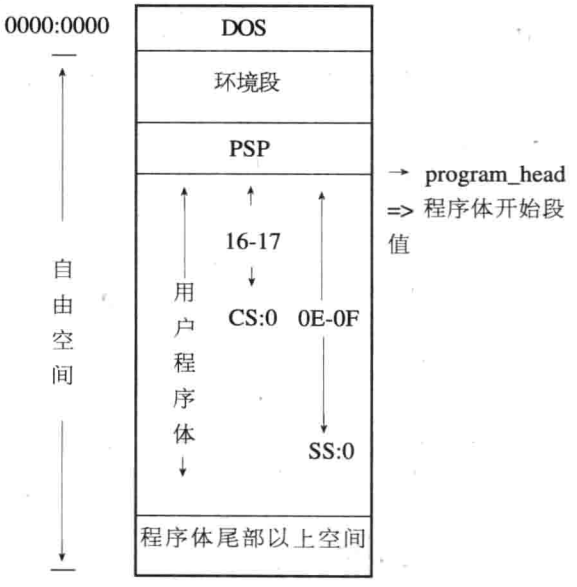

图2.2 .EXE 型用户程序在内存分布图，及程序头部中0E-0F，16-17 所代表的偏移量

（2）.COM 型文件结构

.COM 型文件是用 DOS 盘中的 EXE2BIN.EXE 对 .EXE 文件（此 .EXE 文件必须是从100H 的地址开始执行，且无数据段，无堆栈段，文件体小于 64K）处理后得到的，没有头部，只有一个其长度不得超过 64K 的文件体。文件开始执行时的首地址是 IP=100H，SP=0FFEEH。

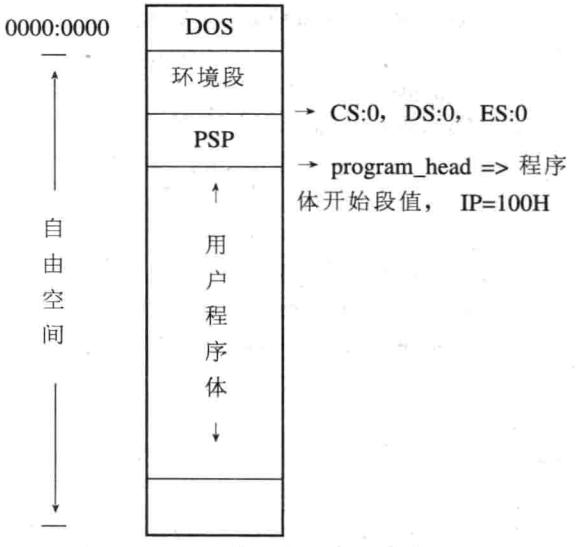

图2.3 .COM 型用户程序在内存分布图

（3） 文件装配程序在加密系统中的位置及作用

一个文件在 DOS 状态下运行时，是由 DOS 的装配程序来装配并转去执行的，用户不需要去关心它。而在加密系统中，加密系统的研制者必须自己编写一段装配程序，来装配被加密程序，并把控制权交给它，图2.4及图2.5说明了装配程序的位置及作用。

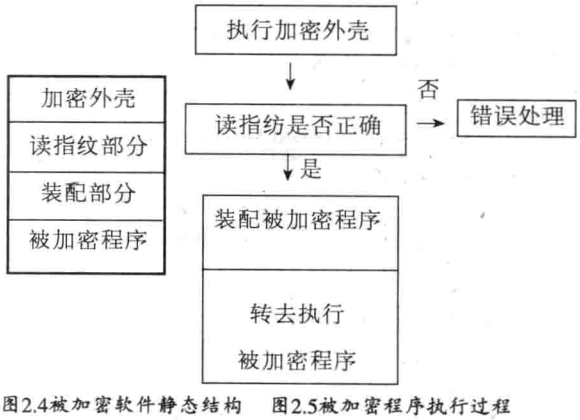

图2.4被加密软件静态结构 图2.5被加密程序执行过程

（4）.EXE和 .COM文件装配例

本例首先判断是.EXE文件还是.COM型文件，而后分情况进行装配，并转去执行。

```
    cmp word ptr file_head,5A4DH;是.EXE文件吗?(图2.1
                                中DS:0=5A4DH?)
    jnz COM_file                ;是 .COM文件，转走
    mov si，offset file_head    ;取.EXE文件头部首地
                                址，见图2.1
                                ;由此，DS:SI指向程序
                                头部的 4DH 字节
    mov cx，word ptr [si+6]     ;取再定位项数
    add si，word ptr [si+18h]   ;得再定位项首地址
    mov bp,word ptr program_head;取内存中程序体段地
                                址，见图2.2、图2.3
    jcxz ok                     ;再定位项为0则转
again:lodsw                     ;取一个再定位项的第
                                一个字
    mov di，ax                  ;送 DI，当作偏移量
    lodsw                       ;取一个再定位项的第
                                二个字
    add ax，bp                  ;加程序体段址
    mov es，ax                  ;送段寄存器
                                ;由此， ES:DI指向程
                                序体的一个字
    add word ptr es:[di]，dx    ;再装配程序体中的一
                                个字
    loop again                  ;循环至完
  ok:mov si，offset file_head   ;取.EXE文件头部首地址
    cli
    mov ax，word ptr [si+0eh]   ;取 SS 段偏移量
    add ax，bp                  ;加程序体在内存中段值
    mov ss，ax                  ;装配好SS
    mov sp，word ptr [si+10h]   ;装配好SP
    sti
    mov ax，word ptr [si+16h]   ;取 CS段偏移量
    add ax，bp                  ;加程序体在内存中段值
    push ax                     ;压栈，准备转出
    mov ax，word ptr [si+14h]   ;取IP值
    push ax                     ;压栈，准备转出
    sub bp，10h                 ;使BP指向PSP
    mov ds，bp                  ;送DS
    mov es,bp                   ;送ES
go_go:xor ax，ax
    mov  bx，ax                 ;初始化各寄存器
    mov cx，ax
    mov dx，ax
    mov si，ax
    mov di，ax
    mov bp，ax
    cld
    sti
    retf                        ;转去执行程序


COM_file:                       ; .COM 文件处理
    mov bp,word ptr program_head;取内存中程序体段地
                                址，见图2.2、图2.3
    sub bp，10h                 ;调整 BP，指向 PSP
    push bp                     ;压栈，弹出后送至 CS
    mov ax，100h
    push ax                     ;压栈，弹出后 100H 送
    至 IP
    mov ds，bp                  ;送DS
    mov es，bp                  ;送ES
    cli
    mov ss，bp                  ;送SS
    mov sp，0ffeeh              ;0ffeeh送SP
    jmp go_go                   ;转去执行.COM 程序
```

（5） **硬盘分区结构**

一个硬盘可以分为几个区域，这是为方便用户或安装几个不同的操作系统。

DOS 3.3 用一个字（word）来代表一个扇区，每个扇区为 512 字节。所以，DOS 3.3最多能管理 33 MB 的分区，而 DOS 5.0 以上版本无此限制。

由 FDISK 对硬盘分区，在硬盘的 0 柱面， 0 面，0 磁头的第一个扇区上建立起一个含分区链表的硬盘分区扇区。本区一开始是引导代码，从 1BEH 开始，每 16 个字节代表一个分区或分区链表，其格式如图2.6所示。

1BEH	boot id	H	S	CYL
1C2H	syst ind	H	S	CYL
1C6H	low word		high word	
1CAH	low word		high word	
1CEH	:		:	
1FEH	055H	0AAH		

图2.6 BOOT 区中分区链表

图2.6中，各项义意如下：

1BEH+00 （boot id） 00H不可自举分区，80H可自举分区（活动分区）

1BEH+01 （H） 本分区开始磁头号

1BEH+02 （S） 开始扇区号

1BEH+03 （CYL） 开始柱面号

1BEH+04 （syst ind） DOS 系统指示字节，

01->本分区FAT表每项占12 bit（1.5字节）

02->xenix操作系统

04->本分区 FAT 表每项占16 bit（2字节）

05->当前分区表为链接项

06->DOS 5.0扩展用，代表分区容量超过33M

1BEH+05（H） 本分区结束磁头号

1BEH+06（S） 结束扇区号

1BEH+07（CYL） 结束柱面号

1BEH+8至1BEH+0B 首扇区的相对扇区号（相对于本分区范围内的第一个物理扇区，即本区的分区扇区），首扇区即本分区的 BOOT 扇区，是用DEBUG的L命令能访问到的扇区。

1BEH+0C至1BEH+0 本分区的总扇区数

1FEH字里的0AA55H 是自举指示字符，来标识一个有效的自举记录。

这里，相对扇区号->相对于本分区的第一个物理扇区（即分区扇区）的依次编号。逻辑扇区号 ->以本分区 BOOT 为0扇区，依次编号。

我们不能用INT 21H或 DEBUG.COM 的L命令读出分区扇区，只能用 INT 13H 读出它们。

```
MOV DL，80H
MOV DH，头号
MOV BX，6000H
MOV ES，BX
XOR BX，BX
MOV CL，扇区号
MOV CH，柱面号
MOV AX，201H
INT 13H
INT 20H
```

```
1B0                                                    80 01
1C0 01 00 04 0F 26 6A 26 00-00 00 FA FD 00 00 00 00
1D0 01 6B 05 0F A6 A9 20 FE-00 00 A0 55 05 00 00 00
1E0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1F0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 55 AA
```

图2.7 C:盘分区扇区中的分区表

```
1B0                                                    00 01
1C0 01 6B 04 0F 26 D5 26 00-00 00 FA FD 00 00 00 00
1D0 01 D6 05 10 66 40 20 FE-00 00 20 FE 00 00 00 00
1E0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1F0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 55 AA
```

图2.8 D:盘分区扇区的分区表

由图2.7 C:盘分区表知，1BEH 为80H表示它为活动分区，本分区从1头1扇0柱面开始，0FH头26H扇区6AH柱面结束。本分区的 BOOT 扇区的相对扇区号是 26H，表明隐含扇区数为 26H，0柱面0头下的26H扇区保留，用户用INT 21H访问不到这些扇区，本区有0FDFAH个扇区。

1CEH为链接项，它开始于6BH 柱面0头1扇区，按链接表的指定，把6BH柱面头1个扇区用INT 13H读入内存，即是D:盘的分区扇区，由图2.8可见，1BEH 至1CDH为D:盘分区表项，1CEH至1DDH为本分区表的链接项，它指出下个分区扇区位于0D6H柱面0头1扇区。继续下去，可得到所有分区的分区表。

（6）程序段前缀（PSP）

每个程序装入内存后，它的前部都有一个PSP，如图2.2、图2.3所示。PSP由100H个字节组成，包含许多重要的信息，各字节含义如下所示。

字节	内容
00-01	指令 INT 20H
02-03	内存大小（以节为单位）
04	保留

05-09	DOS功能调用入口地址
0A-0D	INT 22H 入口地址
0E-11	INT 23H 入口地址
12-15	INT 24H 入口地址
16-17	父进程 PSP 段地址，若无父进程，放当前 PSP 段地址 。
18-2B	文件句柄表
2C-2D	环境段段地址
2E-31	栈切换存储区，当进程对 DOS 栈操作时，用于保存进程的 SS:SP。
32-33	句柄计数，打开最多文件个数。
34-37	指向文件句柄表地址的长指针
38-4F	保留
50-51	INT 21H
52	RETF
53-54	保留
55-5B	文件控制块的扩充，是文件控制块 FCB #1 的扩充域。
5C-6B	格式化的参数区 1，其格式同标准的未打开的 FCB，即 FCB #1。
6C-7B	格式化的参数区 1，其格式同标准的未打开的 FCB，即 FCB #2。
7C-7F	保留
80	命令行长度（第一种用途）
81-FF	命令行参数区（第一种用途）
80-FF	默认的盘传输区（第二种用途）

与 PSP 有关的功能调用：

功能	用途
26H	创建 PSP 块。
50H	设置当前PSP，BX包含一个有效的PSP段地址，以后对 DOS 的调用引用PSP数据，例如文件句柄使用新PSP。
51H	由BX获取当前PSP段地址，功能与62H同。
55H	复制PSP，功能与26H 相似，DX中包含新PSP的段地址。
62H	由BX获取当前PSP段地址。

（7）程序环境块

一般说，一个程序开使执行时，它的PSP前面有一个程序环境块（见图2.2、图2.3），由PSP中2CH处指出环境块段地址，它包含了DOS的环境复制，有PATH，COMSPEC，PROMPT设置和用SET命令设置的变量的地方。环境变量的一般格式是 NAME=strings。环境块一般格式如图2.9所示。

```
COMSPEC=C:\COMMAND .COM 0
INCLUDE=C:\C600\INCLUDE 0
LIB=C:\C600\LIB 0
ECHO=OFF 0
PROMPT=$P$G 0
PATCH=C:\DOS;C:\C600;C: \PCTOOLS 0
0
0001
C:\PCTOOLS\PCTOOLS.EXE 0
unused
```

图2.9 环境块例图

由上例中可见，每项都以 0 字节为结束，字 0001 下面的一行是正在执行的文件及路径。

（8）文件控制块（FCB）结构

INT 21H 的功能调用 0FH～29H（由 MS-DOS 第一版引入）用于与 FCB连用以创建、修改和删除文件。DOS 和应用程序用 FCB 参数确定文件的地址、名字、大小和其它有关信息。FCB 及扩充的 FCB 结构如下所示：

字节	功能
00	驱动器号 打开之前：0 -> 缺省驱动器， 1-> A:,2 -> B:,..... 关闭以后： 1 -> A:， 2 -> B:,..... 打开期间用实际驱动器号代替 0。
01-08	文件名，左边对齐，尾部补空格。
09-0B	文件扩充名，左边对齐，尾部补空格。
0C-0D	相当于文件头的当前块号，从零开始计算（由打开功能调用置零），一块由128个记录组成，当前块号和当前记录字段一起用于顺序的读和写。
0E-0F	按字节计算的逻辑记录长，由打开功能设置为 80H，也可自行设置。
10-13	按字节计算的文件长度。
14-15	月/日/年

16-1F　保留
20　当前块内当前记录号
21-24　相对文件头的相对记录号,从零开始计算。

说明：未打开的FCB由FCB前缀（如果使用的话）、驱动器号、文件名和扩充名组成，而打开的FCB，由打开功能调用来填充其余字段。

扩充的文件控制块（用offset FCB表示普通 FCB 的首地址）

offset FCB减7　0FFH 标志字节，指出这是一个扩充的FCB
offset FCB减6到offset FCB减2　保留
offset FCB减1　文件属性字节

在DOS功能调用中，无论使用打开的FCB或未打开的FCB，都可以使用普通的或扩充的FCB。

使用 FCB 的方法来处理文件来源于DOS 1.0，现在，人们都喜欢用“路径\文件名字串”的命令INT 21H的3DH、3FH、40H、3EH来打开、读/写、关闭文件以求简捷省事，但是，由上面的FCB块结构可见，用FCB来处理文件可以得到更多的信息，如文件的大小、产生日期、当前块等等，是加密解密中的重要手段之一，下例说明它的用法。

```
mov ah, 29h              ;分析文件名
mov al, 1                ;跳过分隔符
mov si, offset fname1    ;取文件名首地址
mov di, offset fcb1      ;取空白 FCB 首地址，
                          存放符合 FCB 格式的
                          分析结果。
int 21h
or al, al
jnz name_err             ;若坏文件名，则转出
 .
 .
mov ah, 0fh              ;打开文件
mov dx, offset fcb1      ;指向存放刚才分析结果
                          的 FCB
int 21h
or al, al
jnz no_file              ;打开失败则转出
 .
 .
mov ah, 1ah              ;设立盘传输区
 .
mov dx, offset buffer ;送传输区地址
int 21h
mov ah, 14h              ;顺序读
mov dx, offset fcb1
int 21h
cmp al, 1
je file_end              ;文件结束则转
cmp al, 3
je file_end              ;文件尾则转
or al, al
jnz bad_read             ;坏文件则转
 .                       ;成功读入数据
 .
file_end: mov ah, 10h    ;关闭文件
mov dx, offset fcb1
int 21h
mov ax, 4c00h            ;程序结束
int 21h
fname1 db 'my_name.dat', 0
fcb1 db 37 dup (0)
buffer db 200h dup (?)
```

(9) DOS 内存链

用户程序装配在自由内存空间（见图2.2），我们把它叫作暂存程序区（即 TPA ->Transient Program Area），由于用户程序可以驻留在 TPA，所以内存可以被分成多个区域。

由图2.10可见，TPA被分成许多个内存控制块

address	type	owner	size	
0A00：0	4D	0008	1600	
0A01：0	DOS 占有块			
2001：0	4D	2013	0010	
2002：0	进程 2013 占有块			
2012：0	4D	2013	0500	
2013：0	进程 2013 占有块			
2513：0	5A	0000	7AEC	
2514：0	自由块			

2.10 DOS 的内存控制块图

（MCB），MCB 的第 0 个字节 4D 为开头的链接块和以 5A 为开头的最后块，第 1，2 个字节指出该内存块的主人（owner），采用指出主人的 PSP 段地址的方来实现，本字节若为 0000，则表示本 MCB 是自由的，第 3，4 个字节指出该 MCB 块大小（size）。

DOS 用 INT 21H 的 AH=52H 来返回指向 DOS 的内部值列表的指针，该指针返回在ES：BX 中，由 ES：[BX－2] 所指的字处，就是第一个 MCB 的段地址，从该点开始，链中下一个MCB 的段地址通过对当前 MCB 段地址加当前块的大小（以段），再加一，以这种方法可以遍历整个 MCB 链。

图2.10中，第二个MCB段地址=0A00H+1+1600H=2001H，第三个MCB段地=2001H+1+0010H=2012H。

由以上所学到的各种知识，我们可以轻易地查到任何一个 MCB属于那一个程序。首先从 MCB 的第 1，2 字节找到它主人的 PSP，再从 PSP 的 2CH 处找到它的环境段地址，从环境段的 0，0001 （见图2.9）的数据结构的后面查出主人的名字（程序名）。

（10） **由程序判断CPU 的类型**

加密程序必须有能力判断本计算机的 CPU 的类型，以决定采用那一种反跟踪措施，下面给出一个实际程序，供读者采用。

```
.model small
.386
.code
begin:mov ax, @data
    mov ds, ax
    mov es, ax
    pushf               ;FLAG 送 BX
    pop bx
    and bx, 0fffh       ;清除 12-15 位
    push bx
    popf                ;回送到 FLAG
    pushf
    pop ax              ;再送 FLAG 到 AX
    and ax, 0f000h      ;如果 12-15 位被设置，则是
                        8086。
    cmp ax, 0f000h
    jz is_8086
    or bx, 0f000h       ;设置 FLAG 的 12-15 位
    push bx
    popf                ;送到 FLAG
    pushf
    pop ax
    and ax, 0f000h      ;如果 12-15 被清除，则是
                        80286。
    jz is_80286
    mov edx, esp        ;保存当前栈指针
    and esp, not 3      ;调整栈以避免 AC 位（eflags
                        的18位）故障
    pushfd              ;EFLAGS 压栈
    pop eax             ;得EFLAGS值
    mov ecx, eax        ;保存原 EFLAGS
    xor eax, 40000h     ;取反 AC 位
    push eax            ;COPY 到 EFLAGS
    popfd
    pushfd
    pop eax             ;eax 得到新 EFLAGS 值
    xor eax, ecx        ;观察 AC 位的变化
    shr eax, 18         ;将eax中的AC位移到第0位
    and eax, 1          ;去掉无关位
    push ecx            ;恢复 EFLAGS
    popfd
    mov esp, edx        ;恢复esp
    cmp eax, 0          ;eax=0 -> 80386,  eax=1 ->
                        80486
    jz is_80386         ;eax=0，是 80386，则转
is_80486:
    mov dx, offset cpu80486
    jmp done
is_80386:
    mov dx, offset cpu80386
    jmp done
is_80286:
    mov dx, offset cpu80286
    jmp done
is_8086:
    mov dx, offset cpu8086
done:mov ah, 9          ;显示结果
    int 21h
```