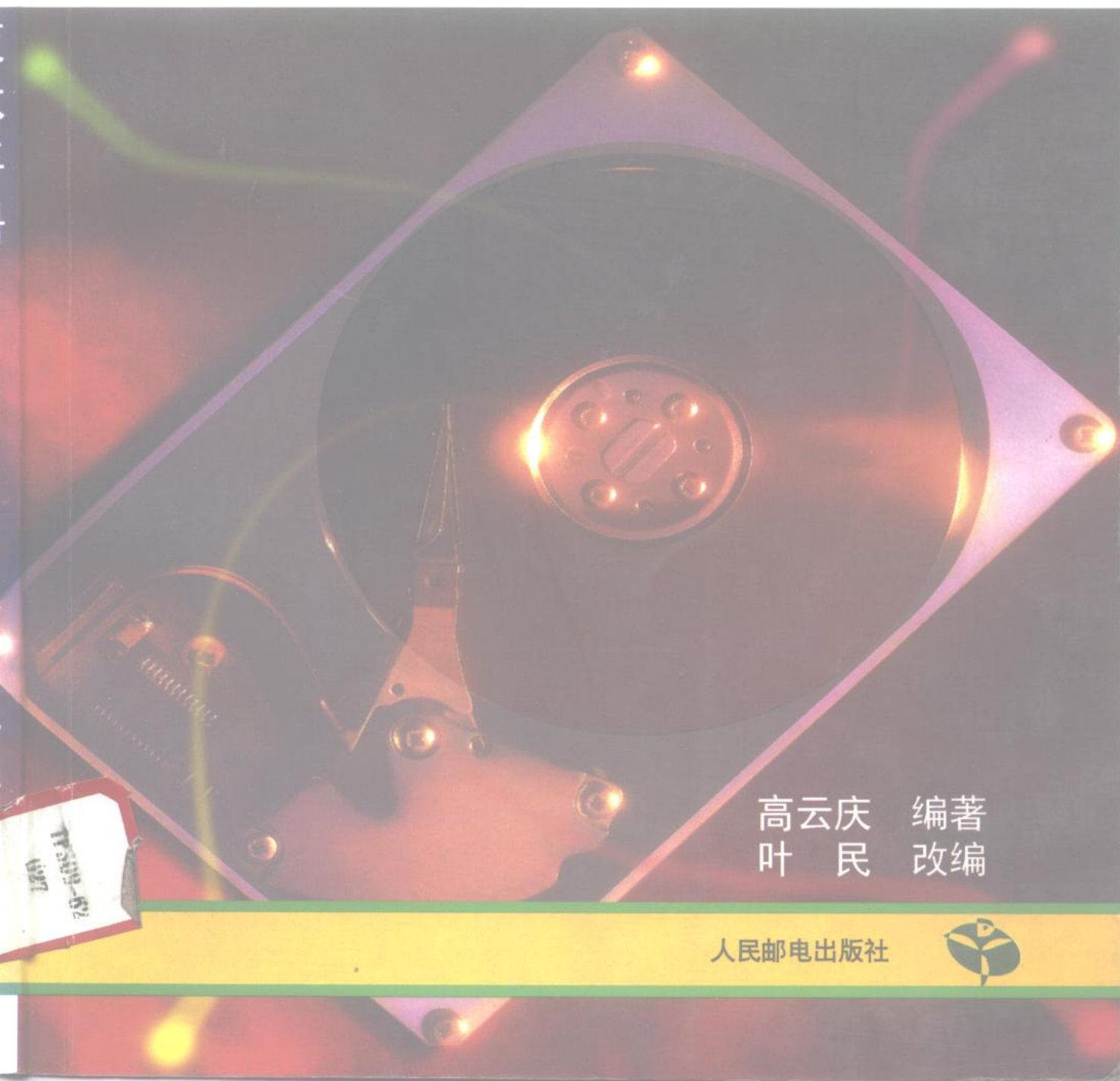


GOTOP

计算机技术入门提高精通系列丛书

硬盘保护 技术手册



高云庆 编著
叶 民 改编

人民邮电出版社



计算机技术入门提高精通系列丛书

硬盘保护技术手册

高云庆 编著
叶 民 改编

人民邮电出版社

计算机技术入门提高精通系列丛书

硬盘保护技术手册

高云庆 编著

叶 民 改编

责任编辑 万东旭

*

人民邮电出版社出版发行

北京朝阳门内南竹杆胡同 111 号

北京密云春雷印刷厂印刷

新华书店总店科技发行所经销

*

开本: 787×1092 1/16 1996 年 3 月 第 一 版

印张: 20 1996 年 3 月 北京第 1 次印刷

字数: 467 千字 印数: 1—5 000 册

ISBN 7-115-05999-3/TP·271

图字: 01-95-198 号

定价: 30.00 元

硬盘是计算机用户保存数据的主要设备,如何避免硬盘受到他人恶意访问和计算机病毒的侵入,是所有用户关心的问题。本书全面剖析了从裸机上电启动到操作系统引导的全过程,介绍了在 DOS 环境下系统保护的各种方法。

本书分为三部分:第一部分基础篇,主要介绍保护程序设计的技巧、MASTER BOOT 扇区、BOOT SECTOR 的结构;第二部分提高篇,介绍扇区程序的编写,针对保护程序的每一具体功能分别做了详细讲解;第三部分研究篇,对于硬盘保护方法进行了总结、归纳,并介绍了两个典型硬盘锁程序。通过本书,读者不仅可以了解计算机基本原理,还可以掌握 DOS 环境下硬盘保护的各种方法,书中列举的大量编程技巧可供读者参考。

本书适用于广大计算机用户,及大、中专院校计算机专业师生阅读。

本书原版书名为《硬碟保护技术手册》,1995 年 3 月出版。



在计算机技术飞速发展的今天,为了进一步向全社会普及计算机知识,提高计算机应用人员的技术水平,使计算机在各个领域发挥更大作用,也为了促进海峡两岸计算机技术图书的交流;台湾碁峰资讯股份有限公司对我社独家授权陆续组织出版该公司的部分计算机技术书籍。这些书基本覆盖了当前最常用的各类计算机软、硬件技术,并紧随世界上计算机技术的飞速发展,不断有所更新。在写作特点上,这些书内容深入浅出、实用性强,在台湾很受读者欢迎。

在组织出版过程中,我们请有关专家在尊重原著的前提下,进行了改编,并对有关图文进行了核对和精心制作。

由于海峡两岸在计算机技术名词的称谓上差异较大,改编者依照有关规定和大陆习惯用法进行了统一整理。

对原书文字叙述中由于海峡两岸不同的语言习惯而造成的差异,我们的处理原则是只要不会造成读者理解上的歧义,一般没做改动,以尊重原著写作风格。另外改编时对原书的一些差错及疏漏之处做了订正。

由于本书改编和出版时间紧张,如有差错和疏漏,敬请读者指正。

人民邮电出版社

1995.9



本书为台湾群峰资讯股份有限公司独家授权的中文简化字版本。本书专有出版权属人民邮电出版社所有。在没有得到本书原版出版者和本书出版者书面许可时,任何单位和个人不得擅自摘抄、复制本书的一部分或全部以任何形式(包括资料和出版物)进行传播。

本书原版版权属群峰资讯股份有限公司。

版权所有,侵权必究。

基础篇

●第一章 程序编写基础.....	3
1-1 本书名词解释.....	3
1-2 汇编语言程序编写说明.....	4
1-3 工具程序的使用.....	9
●第二章 硬盘 MASTER BOOT 扇区分析	13
2-1 程序流程	16
2-2 程序分析	16
2-3 MASTER BOOT 程序清单	22
2-4 另一种写法	24
2-5 硬盘分区表的格式及其说明	28
2-6 MASTER BOOT 扇区安装程序	30
2-7 C 语言编写的安装程序	33
●第三章 引导扇区 (BOOT SECTOR)程序的编写	37
3-1 程序流程	38
3-2 程序数据格式	38
3-3 程序分析	40
3-4 程序编译方式	44
3-5 程序清单——IBMBOOT.ASM	44
3-6 引导扇区安装程序——FIXBOOT.ASM	49

提高篇

●第四章 热键式硬盘锁	59
4-1 程序原理	59
4-2 使用说明	60
4-3 程序的编写	60
4-4 安装程序——XLOCKLD.ASM	64
4-5 C 语言编写的安装程序	66
●第五章 密码式硬盘锁	71
5-1 程序原理	71
5-2 使用说明	73
5-3 程序的编写	74
5-4 安装程序——XLOCK2LD.ASM	79
5-5 C 语言编写的安装程序	83
●第六章 硬盘防病毒程序	89

6-1	程序原理	89
6-2	使用说明	90
6-3	程序清单	91
6-4	安装程序	94
6-5	C 语言编写的安装程序	97
6-6	程序改进	100
●第七章	硬盘隐藏程序	115
7-1	程序原理	115
7-2	使用说明	117
7-3	程序清单——TBLOCK. ASM	118
7-4	安装程序清单——TBLOCKLD. ASM	121
7-5	编码程序清单——ENCODE. ASM	123
7-6	C 语言编写的安装程序	124
7-7	C 语言编写的编码程序——CENCODE. C	127
●第八章	隐藏式硬盘锁	129
8-1	程序原理	129
8-2	读取字符串输入过程	129
8-3	编码程序	131
8-4	使用说明	131
8-5	程序的编写	132
8-6	程序清单	132
8-7	安装程序——XLOCK3LD. ASM	138
8-8	C 语言编写的安装程序	142
●第九章	多重操作系统启动程序	147
9-1	多重操作系统启动程序概述	147
9-2	程序原理	147
9-3	程序流程	149
9-4	使用说明	150
9-5	程序清单	150
9-6	多重操作系统启动程序的安装程序清单	154
9-7	C 语言编写的安装程序	156
●第十章	软盘隐藏程序	161
10-1	软盘隐藏程序概述	161
10-2	程序原理	161
10-3	安装程序的编写	164
10-4	使用说明	164
10-5	程序清单——BOOT1. ASM	164
10-6	安装程序清单——HDBOOT. ASM	167
●第十一章	软盘启动管理器	171

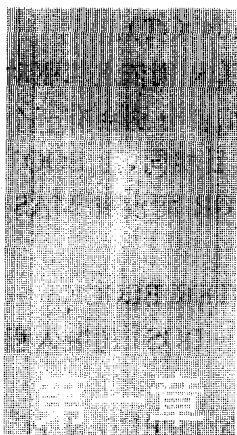
11-1	软盘启动管理器概述	171
11-2	程序原理	171
11-3	软盘启动流程	172
11-4	KEY DISK 保护原理概述	174
11-5	INT 1EH 原理	175
11-6	利用 BIOS 格式化特殊扇区	176
11-7	格式化特殊扇区程序清单——FMTDSK. ASM	177
11-8	BOOT MANAGER 程序清单	178
11-9	安装程序的编写	181
11-10	使用说明	181
11-11	安装程序清单——BOOTMAN. ASM	182

研究篇

●第十二章	硬盘锁程序编写综述	189
12-1	DOS 看不到的硬盘扇区	189
12-2	中断服务程序的编写	189
12-3	如何让软盘启动不能进入硬盘	191
12-4	如何做到仅写保护第 1 逻辑驱动器(锁 C 盘不锁 D 盘)	191
12-5	不通过 DOS 而使程序驻留的技巧	193
●第十三章	硬盘密码锁	197
13-1	硬盘密码锁功能简介	197
13-2	硬盘密码锁的使用	198
13-3	程序分析	201
13-4	程序编译方法	214
13-5	装入程序(HDPASS.C)分析	214
13-6	修改密码程序(SETPASS.C)清单	225
13-7	撤除密码程序(HDREMOVE.C)清单	229
●第十四章	区域写保护式硬盘锁	233
14-1	功能简介	233
14-2	区域写保护式硬盘锁的使用	234
14-3	程序分析	237
14-4	程序编译方法	248
14-5	HDLOCK.C 安装程序分析	249
14-6	SETPASS.C 修改密码程序清单	260
14-7	HDREMOVE.C 撤除密码程序清单	263
●第十五章	硬盘锁、写保护卡破解	269
15-1	保护原理分析	269
15-2	破解原理分析	270
15-3	单步中断服务程序分析	272

15-4 取得 INT 13H 原始入口地址程序(INT 13ADR. ASM)清单	273
15-5 硬盘写保护卡破解程序	276
附录 A 本书内各程序使用说明	279
附录 B 工具源程序清单	291
B-1 BINTODAT.C 源程序清单及使用说明	291
B-2 BINTOCAT.C 源程序清单及使用说明	293
附录 C INT 13H 驱动器中断功能	295
附录 D PARTITION TABLE 格式	305

基 础 篇



程序编写基础

1-1 本书名词解释

为避免读者对本书中的用词有所疑惑,造成阅读的困扰,在此就本书中的用词做一简单说明:

主引导扇区(MASTER BOOT SECTOR):硬盘的第1扇区,位于硬盘第0道第0面第1扇区,它是IBM PC读硬盘时所最先读取的扇区,包含了一小段装入程序(LOADER)及一个硬盘分区表(PARTITION TABLE)。这个扇区有些人称之为MBR(MASTER BOOT RECORD),也有人因为这个扇区中包含分区表(PARTITION TABLE),称之为分区扇区。在本书中一律称之为MASTER BOOT(主引导扇区),在以后的各章中所谓MASTER BOOT就是指这个扇区。在“基础篇”中我们会对这个扇区做深入的探讨,它也是“提高篇”的程序基础。

因为主引导扇区是独立在DOS之外用来装入操作系统的扇区(所使用的操作系统可以不是DOS,可能为其它的操作系统,如XENIX),所以不能用DOS的扇区读取命令来读取它,在DEBUG中也没有相对应的命令来读取它,所以必须使用BIOS的INT 13H功能调用编写一个程序去读取它。在本书的“基础篇”中会详加解释的。

引导扇区(BOOT SECTOR):指的是软盘的第1扇区以及硬盘DOS分区开始的第1扇区。许多人会把这个扇区和硬盘的MASTER BOOT混淆。引导扇区(BOOT SECTOR)指的是逻辑驱动器的第1个扇区,假如一个硬盘分割成C、D两个逻辑驱动器,那么在C盘及D盘中各会有一个引导扇区(虽然D盘不可以用来启动,但

是它还是会有这个扇区),然而硬盘却只能有一个主引导扇区(MASTER BOOT)。

简而言之,在硬盘中,MASTER BOOT 只能有一个,而 BOOT SECTOR 则依逻辑驱动器的数目而定。假如一台硬盘分割成 C、D、E 三个逻辑驱动器,那么它们就分别会有一个 BOOT SECTOR。换句话说,MASTER BOOT 的数目和物理驱动器数目相同,而 BOOT SECTOR 的数目则和逻辑驱动器数目相同。软盘则只有 BOOT SECTOR 而没有 MASTER BOOT。

引导扇区包含用来描述这个逻辑驱动器的 BPB (BIOS PARAMETER BLOCK——BIOS 参数块),以及用来装入 IO.SYS (MS-DOS) 或 IBMIO.SYS(PC-DOS) 的装入程序。

由于引导扇区是定义在 DOS 层次的,因此一般的 DOS 工具程序,如 DEBUG 及 PC-TOOLS 都可以访问到这个扇区。

1-2 汇编语言程序编写说明

1. 程序的编译

本书所有程序都可以在 MASM 及 TASM 中编译,作者大部分是使用 TASM 作为编译工具,它们两者的汇编及链接方式如下:

(1) 以 XLOCK.ASM 说明扇区程序的编译法(见第四程序)。

• TASM

TASM XLOCK

TLINK XLOCK

EXE2BIN XLOCK——产生 XLOCK.BIN

BINTODAT XLOCK.BIN XLOCK.OUT——转换成汇编语言格式数据文件

• MASM

MASM XLOCK;

LINK XLOCK;

EXE2BIN XLOCK——产生 XLOCK.BIN

BINTODAT XLOCK.BIN XLOCK.OUT——转换成汇编语言格式数据文件

其中 BINTODAT.EXE 为作者自行编写的工具程序,用来将二进制文件转换成汇编语言格式的数据文件,因为 XLOCK.ASM 为一扇区程序,不能在 DOS 下执行,必须将它写入 MASTER BOOT 中才可以在启动的时候执行。为了程序编写方便,在本书中都是将它转换成汇编语言的数据文件,然后再写一个装入程序。至于 BINTODAT.EXE 的源程序及详细用法请见附录。

(2) 以 XLOCKLD.ASM 说明 COM 执行文件的编译法(见第四程序)。

• TASM

TASM XLOCKLD

TLINK/t XLOCKLD——t 必须为小写

• MASM

```

MASM XLOCKLD;
LINK XLOCKLD;
EXE2BIN XLOCKLD.EXE XLOCKLD.COM

```

如果读者没有 EXE2BIN 这个程序 (TASM 不提供), 也可以用 DEBUG 程序自行做转换的工作, 其用法如下:

```

C>DEBUG XLOCK.EXE
-N XLOCK.BIN —— 取文件名为 XLOCK.BIN
-R CX
200 —— 设置文件大小 (DEBUG 以 BX: CX 表文件大小)
-W CS: 0 —— 由 CS: 0 开始写入
-Q —— 离开

```

(3) 由于以上操作频繁, 因此作者建议各位读者可以写一个批处理文件完成以上的工作。作者提供一个简单的批处理文件:

```

• RUNBIN.BAT
TASM %1
TLINK %1
EXE2BIN %1
BINTODAT %1.BIN %1.OUT

```

```

• RUNCOM.BAT
TASM %1
TLINK/t %1

```

如此一来前面的过程我们只要改成下面的用法即可:

```

RUNBIN XLOCK
RUNCOM XLOCKLD

```

使用 MASM 的读者只要将上面的 TASM 改成 MASM, TLINK 改成 LINK 即可正常执行了。

2. 汇编语言的特殊技巧

在本书中使用了少部分在汇编语言中鲜为人知的技巧, 我们在此一一说明, 读者也可以体会汇编语言的威力:

(1) 控制权切换的方法。

汇编语言的控制权切换指令有下面几个:

- JMP 指令群 (JMP, JNZ, JE.....)
- CALL 指令
- INT 指令

其中 JMP 及 CALL 指令大多使用相对地址, 也就是说链接程序在碰到 JMP 及 CALL 指令时必须将转移的相对地址算出, 然后才给定它的值。(JMP 及 CALL 可以使用绝对地址, 稍后再加说明, 此处先说明一般使用符号地址做转移的情形)。

我们举一个简单的例子来说明:

程序 TEST.ASM

```

.MODEL TINY
.CODE
ORG 100H
BEGIN:
    PUSH CS
    POP DS
    JMP @1
@1:
    JMP @2
    NOP
    NOP
@2:
    MOV AH,4CH
    INT 21H
    END BEGIN

```

显示如下:

```

C > DEBUG
-n test.com
-l
-u 100 110
3809:0100 0E      PUSH    CS
3809:0101 1F      POP     DS
3809:0102 EB01    JMP     0105 ← 机器码 EB 01
3809:0104 90      NOP
3809:0105 EB03    JMP     010A ← 机器码 EB 03
3809:0107 90      NOP
3809:0108 90      NOP
3809:0109 90      NOP
3809:010A B44C    MOV     AH,4C
3809:010C CD21    INT     21
3809:010E E9EEFE  JMP     EEFE
-q

```

TEST.ASM 是一个简单的程序,用来查看 JMP 指令所编译出来的机器码。我们由 DEBUG 的结果可以看出第一个转移指令 JMP @1 所编译出的机器码是 EB 01,而第二个转移指令 JMP @2 所编译出来的机器码是 EB 03。由这里我们可以很简单地看出,JMP 指令的机器码是 EB,而后面接的数字则是程序指针至新地址所必须加上的位移值。

同样地,读者可以用同样的方式来对 CALL 指令做检查,其结果应该和上面类似。

现在让读者思考一个问题:如果要使程序转移到固定地址时,应该如何来完成?我们将问题简单化,如果要让程序转移至 0:600 的地址,应该怎么做才可以呢?

前面说过,因为 JMP 指令是使用相对地址,因此在不同的地方使用 JMP 指令转移到同一地址会产生不同的机器码,因为其相对地址不同;就好像虽然同样是乘飞机到北京,但是由上海到北京和由广州到北京的票价就不同一样。我们写一个程序来验证这个说法:

程序 TEST2.ASM

```
.MODEL TINY
.CODE
ORG 100H
BEGIN:
    PUSH CS
    POP DS
SHANGHAI:
    JMP BEIJING
    NOP
    NOP
GUANGZHOU:
    JMP BEIJING
    NOP
    NOP
BEIJING:
    MOV AH,4CH
    INT 21H
    END BEGIN
```

显示如下:

C > DEBUG

-n test2.com

-l

-u 100 110

3809:0100 0E	PUSH	CS
3809:0101 1F	POP	DS
3809:0102 EB08	JMP	010C
3809:0104 90	NOP	
3809:0105 90	NOP	
3809:0106 90	NOP	
3809:0107 EB03	JMP	010C
3809:0109 90	NOP	
3809:010A 90	NOP	
3809:010B 90	NOP	
3809:010C B44C	MOV	AH,4C
3809:010E CD21	INT	21

-q

同样是 JMP 10C,但是一个机器码为 EB 08,
另一个为 EB 03,因为位移不同。