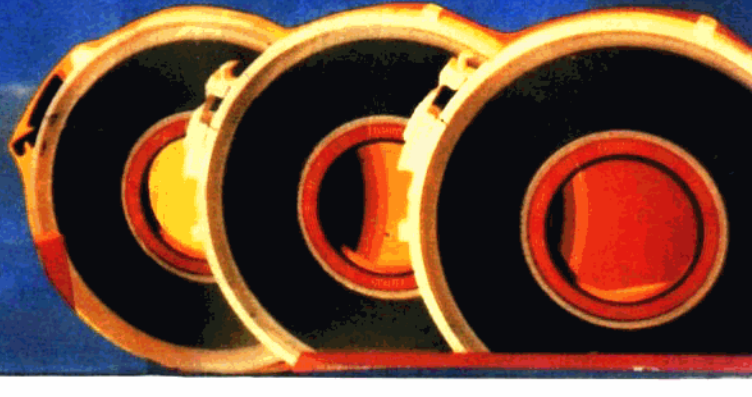
HOPE

高级DOS程序设计

——MS和PC—DOS环境下的内存驻留程序、中断和磁盘管理

北京希望电脑公司



高级DOS程序设计

严红岩

北京希望电脑公司

一九九一年九月

■北京市新闻出版局
准印证号：3575—91575
■订购单位：北京8721信箱资料部
■邮　　码：100080
■电　　话：2562329
■乘　　车：320、332、302路车至海淀黄庄下车
■办公地点：希望公司大楼101房间

目　录

第一部分　磁盘分析 …… (1)

第一章　磁盘简介 …… (2)
程序设计要点 …… (3)

第二章　EXPLORER 概貌 …… (4)
程序设计要点 …… (5)

第三章　读命令及磁盘 I/O …… (6)
读命令 …… (6)
磁盘读写 …… (7)
DOS 缓冲区 …… (9)
EXPLORER. PAS …… (9)
程序设计要点 …… (14)

第四章　扇区分析 …… (15)
使用这个程序 …… (22)
程序设计要点 …… (23)

第五章　引导记录 …… (24)
引导记录和 IBM PC 兼容机 …… (25)
BOOT. PAS …… (25)
程序设计要点 …… (27)

第六章　文件分配表 …… (28)
12 位和 16 位 FAT 表 …… (28)
FAT. PAS …… (30)
程序设计要点 …… (34)

第七章　根目录 …… (35)
ROOT. PAS …… (36)
改变目录信息 …… (40)
程序设计要点 …… (42)

第八章　文件 …… (43)
FILE. PAS …… (43)
子目录 …… (52)

使 FILE. PAS 能处理子目录 …… (52)
程序设计要点 …… (55)

第九章 删除文件 …… (56)
ERASED. PAS …… (56)
程序设计要点 …… (68)

第十章 分区表 …… (69)
用 BIOS 读分区表 …… (69)
PART. PAS …… (70)
程序设计要点 …… (72)

第十一章 磁盘问题及技巧 …… (73)
磁盘技巧 …… (74)
程序设计要点 …… (74)

第十二章 改变 DOS 内部命令 …… (75)
NEWCMMDS …… (75)
程序设计要点 …… (81)

第二部分 BIOS 和 DOS 中断及其实用程序 …… (82)

第十三章 中断和汇编程序设计简介 …… (83)
为什么使用中断? …… (83)
中断和实用程序 …… (83)
汇编语言程序的结构 …… (84)
编程须知 …… (84)
程序设计要点 …… (85)

第十四章 输出,屏幕控制,正文和图形 …… (86)
选择视频页 …… (87)
确定屏幕方式和活动页 …… (87)
清屏 …… (87)
打印字符 …… (88)
显示字符串 …… (90)
控制光标 …… (91)
从屏幕读字符 …… (92)
图形 …… (92)
程序设计要点 …… (94)

第十五章　输入：键盘、光笔和鼠标器 …… (95)
给键盘缓冲区增加键 …… (106)
读字符串 …… (106)
光笔 …… (107)
鼠标器 …… (107)
程序设计要点 …… (112)

第十六章　PSP 和参数传送 …… (113)
程序设计要点 …… (115)

第十七章　磁盘文件 …… (116)
打开文件 …… (120)
读、写以及定位 …… (125)
记录文件数据 …… (126)
关闭文件 …… (127)
传送和重命名文件 …… (127)
删除文件 …… (128)
改变文件的属性、日期和时间 …… (128)
设备的输入/输出控制 …… (130)
程序设计要点 …… (134)

第十八章　终止和一个程序实例 …… (135)
一个程序实例：MOVE …… (135)
程序设计要点 …… (139)

第十九章　目录 …… (140)
建立和删除目录 …… (140)
当前目录 …… (141)
搜索目录中的文件 …… (141)
DIR2：一个目录搜索实用程序 …… (143)
程序设计要点 …… (148)

第二十章　存储器 …… (149)
常规存储器 …… (149)
扩充存储器 …… (151)
扩展存储器 …… (151)
程序设计要点 …… (157)

第二十一章　磁盘扇区和驱动信息 …… (158)
磁盘信息 …… (161)

程序设计要点…… (163)

第二十二章　子程序和覆盖…… (164)
程序设计要点…… (166)

第二十三章　处理中断的中断…… (167)
程序设计要点…… (167)

第二十四章　系统和设备信息…… (168)
设备信息…… (171)
程序设计要点…… (173)

第二十五章　其它中断…… (174)
程序设计要点…… (175)

第三部分　内存驻留实用程序…… (176)

第二十六章　内存驻留程序的组成…… (177)
中断…… (177)
中断的类型…… (179)
使用中断表…… (180)
内存驻留程序如何工作…… (180)
不驻留部分…… (181)
例子:VIDEOTBL …… (182)
驻留部分…… (184)
中断处理程序…… (184)
链接…… (185)
cli 和 sti …… (185)
可重入…… (185)
准备处理…… (187)
解决热引导…… (188)
通讯中断…… (189)
处理部分…… (190)
退出中断处理程序…… (190)
程序设计要点…… (191)

第二十七章　实例:PROTECT 程序…… (193)
如何进行格式化…… (193)
PROTECT 如何保护磁盘 …… (193)
PROTECT(代码)源程序 …… (194)

程序设计要点…… (200)

第二十八章　程序协同操作及其它问题…… (201)
把键盘作触发器使用…… (201)
检查使用中断 9 的触发键…… (201)
检查使用中断 16h 的触发键…… (204)
与其它键盘触发程序共存…… (205)
取代中断 9 …… (205)
使用时钟中断的程序…… (205)
写弹出程序…… (206)
何时弹出…… (206)
检查屏幕状态并保存屏幕…… (207)
转换屏幕…… (208)
往屏幕写…… (208)
准备退出…… (208)
弹出程序与多任务处理环境…… (208)
使用 DOS 和多任务程序 …… (209)
多任务执行系统…… (209)
扩展存储器…… (210)
去活，卸载和 AT 机中要注意的问题 …… (210)
AT 要注意的问题 …… (211)
程序设计要点…… (211)
小结…… (212)

第一部分　磁盘分析

这部分，将对磁盘及其引导记录，FAT 表、目录、子目录、文件以及分区进行研究。从中你可了解到它们的功能，怎样去使用它们，及各种不同类型的磁盘的特点。通过设计一个对磁盘进行检测和修改的实用程序，你可以了解一些对磁盘编程的技巧。

这一部分的结尾是一个修改 DOS 内部命令的程序。

了解磁盘的工作原理在许多方面对你会很有帮助。如果你操作磁盘时犯了某种失误，比如不小心删除了一个文件，重新格式化了硬盘，或毁坏了一个目录，你就能知道怎样去恢复这些数据。这些知识可帮助你开发或利用那些使用磁盘的应用程序，比如数据库或加密系统。

第一章　磁盘简介

你每次使用计算机，其实都是在使用磁盘。从磁盘引导，从磁盘装载应用程序，再把结果存到磁盘上。本章将讲述信息在磁盘上是怎样存贮和组织的，这将有助你编写那些充分利用磁盘资源的功能强大的应用程序。

通常，把磁盘看成是文件的集合。你可以运行程序文件，编辑数据文件，还可以拷贝、删除文件，但磁盘能做的远不止这些。

实际上，磁盘上包含有更多的信息，磁盘必须识别它自己是什么类型的磁盘，包含什么文件，这些文件存在什么位置，哪里有空间来存贮更多的文件。这些信息是以四种基本结构存贮的：分区表，引导记录，文件分配表和目录。用户基本上是看不到这四种结构的。磁盘的物理结构为面、磁道和扇区。首先，看看磁盘的物理结构。

磁盘是由磁介质组成的圆盘，分为软盘和硬盘。硬盘是一类这样的圆盘：每一个圆盘有顶和底两面。正在使用的磁盘上的那一面被称作面。磁头指的是做读工作的电子机械设备，它是一个在磁盘上来回移动的小电磁体。软磁盘有两个面。硬盘的每一圆盘通常都有两个面。

磁盘的每一面进一步分成许多同心环，叫作**磁道**或**柱面**。磁头可在磁盘上的任何磁道上移动。由于磁盘的旋转，磁头就可以读到存贮在磁盘上的不同信息。

每一道又分成许多扇区。扇区是能从磁盘上读或写的最小信息块。一旦读扇区，就可以检查单独的字节。一个扇区一般是 512 字节长。

在一个磁盘上有许多扇区需要管理，因此，扇区又组成簇。簇是相邻扇区的集合。根据磁盘的类型，每簇有 2—8 个扇区不等。图 1.1 是磁盘的示意图。

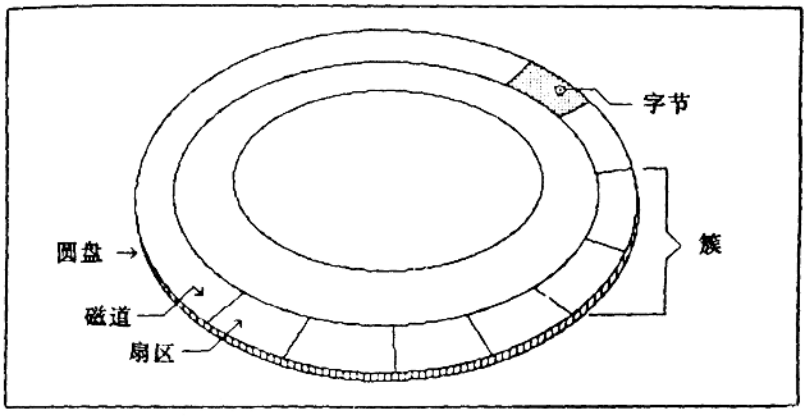


图 1.1　磁盘

上面这些解释听起来很复杂，但你却勿需担心。因为 DOS 已为你做了绝大部分工作，你一般不用考虑簇或扇区，几乎可以不用过问道和面。

绝大多数时间里，你是使用一个操作系统(DOS)，但有一些 PC 用户还需使用 CP/M 或 XENIX 系统。如果这些用户有一个硬盘，他们肯定想用它存贮所有这些操作系统的文件，而这些操作系统操作磁盘的方式与 DOS 不兼容。为使一个硬盘能被一个以上的操作系统所使用，硬盘最多可分成四个区，每一区可以使用一个不同的操作系统。

分区表包含的信息有：每一个分区的位置和大小；如果计算机是以硬盘引导的，那么将使用哪一个区(即活动区)；哪一个区用 DOS 作为其操作系统。这最后一点很重要。你在后面几章将了解到，DOS(和实用程序)要求在磁盘的一些地方找到某种类型的信息。这些预先定义的位置可能在硬盘的任何地方，这就取决于分区是如何建立的了。DOS 利用分区表来确定所使用的分区从哪开始，并把此开始位置作为访问磁盘时的偏移地址。这样，就总能找到那些特别信息，并且对 DOS 和用户双方，分区都是透明的。

当打开一台计算机时，ROM 里的一个程序首先尝试从 A 驱动器(不管 A 驱动器中是否有盘)引导系统。如果 A 驱动器中没有磁盘，那么这个 ROM 里的程序就把控制权转给分区表之前的一个程序。(在一个无硬盘的系统里，就启动 ROM BASIC)。该程序读分区表，如果找到的信息有效，就寻找活动分区引导记录的位置，并移交控制权。

记住，分区表只能在硬盘里找到。

引导记录是磁盘中的第二个重要部分。它包含装载和启动操作系统的程序，还包含一张描述磁盘的表：大小、类型、操作系统、格式以及有关文件分配表和目录的重要信息。DOS 靠这些信息确定怎样对磁盘进行读写。

数据存放在簇中的文件里。簇经常不是顺序出现的，一个文件的第一部分可能在靠近磁盘前面某个地方，而其余部分则可能在磁盘的中间或末尾部分。**文件分配表**(FAT)确定每个文件使用的簇，磁盘的哪一部分是未用的，磁盘的哪一部分因为物理故障而不能使用。文件分配表非常重要，所以在磁盘上常有两个拷贝。没有文件分配表，磁盘上的所有数据将被随意组织而毫无意义。

目录是最后一个组织结构。它包含文件的名字、大小、生成时间，以及 DOS 怎样才能找到文件。

程序设计要点：

- 磁盘包括面、磁道和扇区。
- 扇区组成叫做簇的组织单元。
- 磁盘的五个基本部分是分区表、引导记录、文件分配表、目录和数据区。

第二章　EXPLORER 概貌

本章将详细介绍磁盘的工作原理。你不仅可以了解磁盘的结构和使用方法，还可实际测试一下。为此，将用 Pascal 编写一个磁盘检测程序，由于将讨论磁盘的不同特征，所以要在程序里增加些模块来检测这些特征。

图 2.1 显示了 EXPLORER 的主菜单。

```
----------------------MAIN MENU-----------------
F1:      Dump and Modify Sectors
F2:      Examine Boot Record
F3:      Cluster by Cluster FAT Dump
F4:      Examine Root Directory
F5:      Examine File
F6:      Get Back Erased File
F7:      Examine Partition

F10:     Exit
```

图 2.1　EXPLORER 的主菜单

该程序的全部代码都以文本形式给出。它是用 Turbo Pascal 3.0 和 Turbo Pascal 5.0 写成的。从代码可看出程序在 Turbo Pascal 5.0 下运行，去掉一些注释，就可在 3.0 版本下运行。有关两种版本的区别，已在文本中给出了。如果你使用的是另一种版本的 Pascal，则程序需稍微做些修改。

因为 EXPLORER 程序很长，所以一次只能写一部分，然后用 include($I)命令和主模块一起编译。主模块叫做 explorer.pas。

Turbo Pascal 5.0 像 C 一样有几种不同的整数格式。这些格式为带符号和不带符号，2 个或 4 个字节长。Turbo Pascal 3.0 只使用两字节的整数，但有时不免要处理 4 字节的长整数。为使程序与 Turbo Pascal 3.0 或 Pascal 的其它版本兼容，本书中的程序假定整数是两字节的，带符号的值，也就是大于 32K 的整数是负数，大于或等于 64K 的值返转。但这样造成了一些混乱，为解决这个问题，有时你得把整数值作为浮点数值，执行算术运算，然后再把浮点数回整数。如果你使用的是 Turbo Pascal 5.0，可把这些部分改成使用长整数，避免混乱的选型。

本书将用到二进制、十进制、十六进制编码。例如，二进制数将写成 1001111b，十进制是 14，十六进制数可写成 $15 或 15h。

程序设计要点：

- 编写一个叫做 EXPLORER 的磁盘检测程序。
- 把 EXPLORER 程序分成一系列的模块来编写。对应每一模块，都在 explorer. pas 中增加一个 include($I)命令，然后编译 explorer. pas。

第三章　读命令及磁盘 I/O

在这一章将从 explorer. pas 开始编写 EXPLORER。explorer. pas 是 EXPLORER 的基础。在 explorer. pas 里，要为整个程序设置常量和类型说明，以及主菜单的进程命令。对磁盘进行读写的基本程序也要放在 explorer. pas 里。

为了获得你需要的灵活性和控制权，你编写的程序要从键盘读命令并执行磁盘 I/O。磁盘 I/O 程序可能比后面四章的程序更复杂，因为它要用汇编语言编写。

如果不熟悉汇编语言程序设计，你可能会觉得这些程序相当混乱。为使程序功能更强，必须使用 BIOS 和 DOS 中断调用。如果这些你也不熟悉，那么你就略过并使用本章末尾的程序。你也许不能完全理解它，但至少你知道它的功能。

读命令

为使 EXPLORER 易于使用，看起来也专业化，必须做到能处理功能键和光标键。但 readln 不能处理这些键，你必须要用一个 BIOS 中断调用(将在第二部分详细讨论)才行。

BIOS 中断是存贮在 ROM 里的子程序，执行最基本的与计算机硬件的交互。例如，有向屏幕上打印字符，检测光笔位置和往打印机送信息的 BIOS 中断，也有从键盘上读击键的中断。每个中断都使用一个特定的数值和参数组。

Pascal 中调用中断的方法是，用 intr 命令传送中断号和参数。参数用下列数据结构来设置：

```
result = record

    ax,bx,cx,dx,bp,si,di,ds,es,flags: integer;
  end;
```

每个整数变量代表一个 8086 寄存器(若你对 8086 寄存器不熟悉，不要着急，这个概念在这并不至关重要，书中会告诉你怎样去设置参数)，有时不使用 ax，bx，cx 或 dx，而使用 ah，al，hl，ch，cl，dh 或 dl。这就涉及到相应寄存器的高位或低位字节。例如：ah＝ax/256；al＝ax％256。

读键盘用中断 16h，用 AH＝0 调用。中断程序等待一个击键键入，有一击键后返回，AX 是击键代码。若 AL 不为 0，则该键是一个标准字母键或数字键，AL 中是它的 ASCII 码。若 AL 为 0，AH 中是一个功能键或光标键的码。

中断 16h 和其返回码将在第 15 章详细讨论。现在只检测一下调用中断 16h 后，其返回值的类型。例如，可从主过程看到，假如键入 F10 键，中断将返回 AX＝4400h。

读键盘的 Pascal 程序如下：

```
regs.ax := 0;              {regs is of type result.  AX = 0 means
                            read keyboard}
intr($16,regs);  {wait for a key and return the code}
if regs.ax and $ff = 0 then ... {key is a function or cursor
                                 key}
```

你可以用下列语句检查 AX 的值：

```
case (regs.ax) of .....
```

explorer. pas 程序列在本章的末尾。可从中找到读键盘的程序。

磁盘读写

对磁盘进行读写也要使用中断。这次使用 DOS 中断。DOS 中断同 BIOS 中断非常相似，只是 DOS 中断程序不是 ROM 里的一部分。它们是在计算机被引导时，才装载上的，就程序而言，没有什么区别。

用中断 25h 和 26h 进行扇区读写，用逻辑扇区号访问扇区。磁盘的第一个扇区编号为 0，第二个为 1，以此下去，直到没有扇区剩下为止。

有时用物理扇区号访问扇区。在物理编号系统里，必须说明面和磁道。每个磁道包含本磁道上从 1 到最后一个扇区的所有扇区。DOS 中断则使之变得非常容易，DOS 只考虑什么时候改变磁道或面。只需告诉其一个扇区号，DOS 就会为你找到那个扇区。(相反，BIOS 磁盘中断则使用面、磁道、扇区参考系统。)

进行磁盘读写操作时，几乎都使用逻辑扇区号。

调用中断 25h 时，需要给出四个参数：

AL＝驱动器号(0＝A，1＝B，2＝C，以此类推)
CX＝要读的扇区号
DS:BX＝放置读出数据的缓冲区地址(DS 为缓冲区的段地址，BX 是偏移量)。

中断 26h 使用相同的参数，只是 CX 为要写的扇区号，DS:BX 指向能找到的要写数据的位置。

如果你能象读键盘时所做的一样用 intr 命令调用这些参数，那将很方便。但这两个中断有些麻烦，不象其它的中断，这两个中断在返回调用程序前不能复位，而是留下了一些数据——一个栈状态标记。如果你使用 intr 命令，在读写过程结束之前一切运行良好。但，当它想返回调用它的过程时，就将发生冲突。

为避免这个问题，你必须用 inline 功能在 explorer. pas 里增加一个小汇编语言程序(如果你使用的是 Turbo Pascal，那你一定要用一个相等同的命令。变量进栈的格式可能会有些不同。查阅你的编译员手册)。仔细观察从磁盘读的过程。往磁盘写的过程基本上是相同的。

你要生成一个叫做 DiskRead 的函数来读磁盘。该函数要使用驱动器号，开始扇区号，要读的扇区号和将用来存贮信息的缓冲区的指针。如果磁盘读失败，它就返回一个错误码。用 inline 功能调用 DOS 中断，检查错误，复位并返回。

仔细注意 Turbo Pascal 和汇编语言之间的接口处理过程。在 Turbo Pascal 手册中(2. 0 和 3. 0 版本)有关这点的叙述是错误的，各 Turbo Pascal 版本不同，与汇编语言的接口也不同。

当调用一个函数时，该函数把它的参数送到栈。数值参数作为数据传送，变量参数和数组作为指针传送。Turbo Pascal 3. 0 中数组也是作为指针传送的，而不管是用数值还是变量传送的。Turbo Pascal 5. 0 不是这样。所以当你转换程序使其在 Pascal 5. 0 版本下运行时，要注意这种不同。不同类型的数据用不同的格式传送。字节和整数作为字传送，指针作为偏移量和段传

送，其它类型的传送情况，详见 Turbo Pasacl 手册。

函数说明之后，是用于存放函数结果的栈。整数型、布尔型、字节型、字符型其结果是一个字。然后，参数从说明的左边压入右边，接着压入三个以上的字(注：在 Turbo Pascal 手册中没有记载这些)。最后调用 inline 程序。

读参数首先要保存 bp 寄存器，把其压入栈。然后，把 sp 赋给 bp 并加 8 以指向参数区(刚压入的 bp 用了 2 个字节，刚才压入栈的三个字用了 6 个字节)。与 bp 相加得到下一个参数。记住，因为参数是从左向右压入栈的，所以当你与 bp 相加时要从右到左计数。

这个函数需要传送一个字节、两个整数值和一个指针。功能结果是一个整数。因此，压入 bp 后，将 sp 赋给 bp，再加 8，结果如下：

bp　指向指针偏移量值

bp＋2　指向指针段值

bp＋4　指向一整数

bp＋6　指向一整数

bp＋8　指向一个字节(以字来存贮的)

bp＋10　指向函数结果

记住在你的 inline 程序中要说明你压入栈的任何值。加 8 是因为你压入了 bp，Turbo Pascal 压入了三个附加字。

为了返回函数结果要把你想返回的值放入与返回值相对应的栈单元。

Turbo Pascal 5.0 里，栈调用略有不同。返回值存在 bp－6 里，而不是 bp＋10 里。

可在 inline 程序中修改任何 8086 寄存器。退出之前，你必须恢复 bp，sp，ds 和 ss。

下面看看从磁盘读一个扇区的汇编语言程序。该程序读参数，调用 DOS 磁盘读中断，检查错误，清栈，返回错误码(若没有错误返回 0)。注：程序结尾不需要 ret 指令，在 Turbo Pascal 函数中用 end 代替。为了在 Pascal 程序中使用汇编语言程序，需汇编该程序，并从 .LST 文件中读汇编值。下面这段 explorer.pas 完成这个功能。

```
; Reads sectors from disk using DOS interrupt 25h.  Assumes used
; within a function xxx(drive: byte; sector,no_sects: integer;
;      var buffer:array): integer
; Drive is a number indicating the disk drive, where 0=A, 1=B, ...
; Sector is the starting sector, no_sects tells how many sectors to
; read, and buffer is the disk buffer where the data will be stored.
;
; Returns a zero if the read was successful, otherwise returns a
; number telling an error code.

        push    bp              ;save bp register
        mov     bp,sp           ;get pointer to stack
        add     bp,8            ;point to parameters
        push    ds              ;save ds register
        mov     bx,[bp]         ;offset of disk buffer
        mov     ax,[bp+2]       ;segment of disk buffer
        mov     ds,ax           ;point to buffer for interrupt
        mov     cx,[bp+4]       ;number of sectors to read
        mov     dx,[bp+6]       ;first sector to read
        mov     al,[bp+8]       ;drive
        push    bp              ;save bp because it is destroyed
                                ;by the interrupt
        int     25h             ;issue the DOS disk read
                                ;interrupt
        jnc     ok              ;if there is no carry, the read
                                ;was successful
```

```
                popf                    ;Was no good--pop disk flags
                pop     bp              ;restore bp
                                        ;Use error code for return val
                mov     [bp-6], ax      ; for Turbo 5.0
;               mov     [bp+10],ax      ; for Turbo 3.0
                jmp     cont
ok:             popf                    ;Was good read--pop disk flags
                pop     bp              ;restore bp
                xor     ax,ax           ;Use 0 for error code and use
                mov     [bp-6],ax       ;it as the function return
                                        ;value
;               mov     [bp+10],ax      ; for Turbo 3.0
cont:           pop     ds              ;restore ds
                pop     bp              ;restore bp
                                        ;and continue with the code in
                                        ;the function
```

DOS 缓冲区

为了加速访问时间，DOS 在内存缓冲区中存贮一些磁盘数据。这样，DOS 可直接从内存而不是从磁盘读数据。当用 DOS 写扇区中断写磁盘时，直接修改了磁盘，但不必修改内存缓冲区。因此，在磁盘发生此变化时，观察内部缓冲区的程序将"看"不到这个变化，这样非常危险，特别是对那些从非移动介质中恢复删除文件的程序。

你可在调用中断 26h 后，复位内部文件缓冲区，这样可以很容易地避免这个问题。用 DOS 中断 21h，功能 13 完成这个任务：

操作：复位内部磁盘缓冲区
版本：DOS 1.0—
中断号：21h
功能号：13
调用值：AH=13

DiskWrite 功能在进行了磁盘写操作之后调用该中断。

EXPLORER. PAS

你已能读键盘命令，进行磁盘读写操作，现在来看看整个 explorer. pas 程序。

注意，磁盘读写程序是怎样用 inline 命令插入的。还要注意键盘读线。

大多数常量用来处理窗口位置。在后面的模块中将更多地用到窗口。注意，磁盘缓冲区已足够大，一次可读入九个扇区。其原因将在第 5 章讨论。

还要注意全局变量用来指定被检测的驱动器和扇区，这样你就可以在一个模块中指定一个扇区，然后在另一个模块中对它进行修改。

这段程序在 Turbo Pascal 5.0 下运行。有些部分用{$IFDEF Turbo 3}语句隔开。要在 Turbo Pascal 3.0 下运行该程序，需删除{$IFDEF}行，和所有和 Turbo Pascal 5.0 程序，这部分程序在一个{$ELSE}和一个{$ENDIF}之间。

```
{.....................DISK EXPLORER.........................
.
. This is a program to explore disks.  It has commands for
. decoding the partition, boot record, FAT, and directory,
```