微型小型计算机操作系统应用技术与软件加密技术汇编

中国科学院成都计算机应用研究所情报室

前　言

《微型、小型计算机操作系统应用技术与软件加密技术汇编》一书是为了提高系统软件的编制能力和应用技巧以及提高应用软件编制的效率和提高软件产品的保密性等而编纂的。我们希望它能成为计算机应用工作者，获取成果捷径不可少的工具。

为此，我们尽可能搜集了近两年来国内有关的期刊、资料、会议文献等论文，优选编制成册的，该书内容丰富，有较高的实用价值，是一本难得的内部参考性的情报资料。

该书重点是微型、小型计算机操作系统中的应用实践和软件加密技术应用实践。共分上下两册。

上册内容有：　系统分析、系统移植、汉字操作系统及打印驱动程序的开发与改进、开发应用等。

下册内容有：　加密、解密原理；加密、解密技术；文件的恢复等

在资料的汇集中，因时间关系未一一征求作者的意见，有不妥的地方，敬请作者原谅。并对作者积极支持此工作表示衷心感谢。

由于水平有限及时间关系，搜集的资料还不很完全，错误难免，敬请读者指正。

该书在编审的过程中受到了高树清，罗正伦，吴院尘，单永涛，罗淳等同志的大力帮助和支持；同时也受到了我室孟晓玲、张薇薇、梁年、郑茂才、于川等同志的支持，表示感谢。

该书由杨明芳、周永培统编

编者

87、10

微型、小型计算机操作系统应用技术与软件加密技术汇编

上　册

目　录

系统分析

系统设计与移植

汉字操作系统及打印驱动程序的开发与改进

开 发 应 用

IBM PC的系统分析

谢志良　严　汉　刘国平　范良根　刘殿周

（上海交通大学）

一、系统结构及性能概要

IBM PC包括两个主要部件：系统部件和键盘。还有许多选件：可接两个5.25英寸软盘驱动器及其适配器，单色或彩色显示器及其适配器，每秒80字符的点阵打印机及其适配器，异步通讯适配器和游戏控制适配器等。

系统部件是整个PC的核心，以Intel8088为CPU，在8088旁边还有一个"辅助处理器"插座，可插入数值数据处理器Intel 8087。在合适的软件管理下，8087能加快浮点数据处理以及指数函数、超越函数的速度。系统板上有5个62端的扩充槽，供系统扩充用。

系统板上有48KB ROM（或EPROM）的位置，共有6个插座，每个可插8KB的芯片。系统配有40KB ROM，其内容是盒带操作系统、MS BASIC解释程序、开机自测试程序、I/O驱动程序及软盘引导装入程序等。ROM的存取时间是250ns，周期时间375ns。

最小系统配置是16KB RAM，而留有48KB的RAM插座[1]，还可利用扩充槽加上存储卡，系统最大存储容量可达256KB。存储器使用的是16K×9的芯片，存取时间为150ns，周期时间300ns，所有存储器进行奇偶校验。

总线有4个DMA通道，其中一个用来刷新系统的动态存储器，三个用于存储器和外设之间的DMA传输。DMA传输需5个时钟（1.05ms），共有8级中断，其中用户可用六级。

IBM PC还可配上扩充柜，装两台10MB的硬盘机。

开机时，系统先完成14种测试，发现错误立即报告，包括测试8088、内部ROM、主存、显示适配器、键盘、盒式录音机和磁盘系统。存储器测试包括对用户区域的五种不同的读/写测试，各采用不同的测试比特模式。由于这个原因，IBM PC的启动时间要五秒到一分半钟，长短取决于机器所具有的存储器容量。例如测试一个具有64KB RAM和软盘机的PC，约18秒完成初始测试，约25秒完成引导装入。

图1示出了IBM PC的系统结构框图。

二、ROM BIOS系统简介

固化在ROM中的基本I/O系统（BIOS）为系统部件中的主要I/O设备提供了设备一级的控制，使汇编程序员能进行块（对于盒带和软盘）或字符（对于显示器、通讯、键盘和打印机）级的I/O操作，而不需关心设备地址及其操作特性。另外，BIOS还提供了一些服务程序，如时间、存储器配置等。

BIOS的目的是提供一个系统操作的界面，使用户程序对硬件更改和改进透明。这样便于在系统部件上增加新设备。

访问BIOS功能是通过8088软件中断来实现的，每个BIOS入口通过其自己的中断便可获得（图2）。软件中断10H到1AH各

注(1)：现在的IBM PC采用64K RAM芯片，所以系统板上的容量可达256KB。

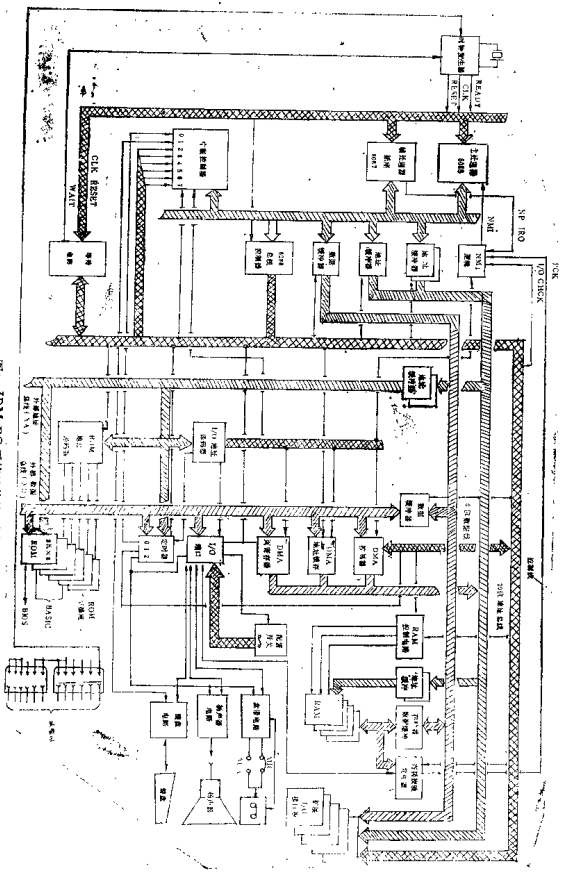

图1 IBM PC系统结构的框图

中断号	中断名	BIOS 初始化
0	被零除	无
1	单 步	无
2	无屏蔽	NMI-INT(F000:E2C3)
3	断 点	无
4	溢 出	无
5	打印屏幕	PRINT-SCREEN(F000:FF54)
6	没 用	
7	没 用	
8	时 间	TIMER-INT(F000:FEA5)
9	键 盘	KB-INT(F000:E987)
A 8256	没 用	
B 中断向量	没 用	
C	没 用(保留通迅)	
D	没 用	DISK-INT(F000:EF57)
E	软 盘	
F	没 用(保留打印机)	
10	显 示	VIDEO-IO (F000:FO65)
11	设备检查	EQUIPMENT(F000:F84D)
12	存储器	MEMORY-SIZE-DETERMINE(F000:F841)
13 BIOS	软 盘	DISKETTE-IO (F000:EC59)
14	通 迅	RS232-IO (F000:E739)
15 入 口	盒 带	CASSETTE-IO (F000:F859)
16	键 盘	KEYBOARD-IO(F000:E82E)
17	打印机	PRINTER-IO (F000:EF02)
18	盒 带BASIC	(F600:0000)
19	引 导	BOOT-STRAP (F000:E6F2)
1A	时 间	TIME-OF-DAY(F000:FE6E)
1B 用户提供的	键 盘	无
1C 子 程 序	计 时	DUMMY-RETURN(F000:FF53)
1D BIOS	显示初时化	VIDEO-PARMS(F000:F0A4)
1E	软盘参数	DISK-DASE(F000:EFC7)
1F 参 数	显示图形字符	无

图 2 中断向量表

访问BIOS的不同模块。例如，若要确定当前系统中可用的存储量，可使用 INT 12H 便会调用BIOS中的存储器大小确定程序，并把相应的值还回给调用程序。

送到BIOS程序的参数及BIOS返回的值

都是通过80H寄存器传递的。例如上例中存储器大小程序的返回结果放在AX寄存器中返回。

如果一个BIOS程序有多种可能的操作，则用AH寄存器传送参数指出所希望的操作。

例　设置时间值所需的代码如下

```
MOV  AH, 1   ；功能是"设置时间"
MOV  CX, HIGH-COUNT；设定当前时间
MOV  DX, LOW-COUNT
INT  1AH     ；设置时间
```

若要读出时间，则可用如下代码：

```
MOV  AH, 0；功能是"读时间值"
INT  1AH,    ；读时间
```

系统规定，除了AX寄存器和标志寄存器外，BIOS程序保护有其它寄存器，这些寄存器只有用于向调用程序返回值时才会被修改。

三、IBM PC的MS-DOS初步分析

IBM PC的操作系统DOS是由Microsoft公司开发的，主要由以下三段程序组成：

（1）引导记录，驻留在每只盘的第一扇区，当启动系统时它自动地收入内存，并负责装入DOS的其余部分。引导记录放在所有盘上，若把非DOS盘插在驱动器A启动系统时，将给出错误信息。

（2）ROM BIOS接口模块，其文件名为IBMBIO.COM，提供了与ROM BIOS设备子程序的低级接口。

（3）DOS程序本身，文件为IBMDOS.COM，它为用户程序提供了高级接口，包括文件管理子程序，磁盘子程序的加锁和去锁以及许多内部功能子程序。当用户程序调用这些功能子程序时，子程序通过寄存器和控制块内容接受对设备输入输出的高级信息，然后把对设备控制的命令，再换成对IBMBIO的一个或多个调用，以完成请求。

1. 磁盘分配

磁盘每面分为40道，每道划分为8个扇区，每扇区为512字节。

0道1扇区放引导记录，0道2—3扇区是两份相同的文件分配表FAT。0道4—7扇区是目录区，0道8扇区之后是数据区。

磁盘中含有DOS的拷贝时，IBMBIO.COM放在0道8扇区到1道3扇区，IBMDOS.COM放在1道4扇区到2道8扇区，它们非在特定的数据区位置，以便系统启动时申请到记录装入内存。

2. 文件分配表FAT

FAT建立文件和扇区的关系，DOS用它为文件分配磁盘空间，一次分配一个扇区。DOS规定0道6扇区开始为相对扇区，相对扇区号为000，001，002，…。FAT给每一相对扇区留一个12位的项，其开始两项映射到目录区的最后两个扇区，指示目录大小和格式说明。FAT的第三项开始，每项由三个16进制字符组成：000表示该扇区未被使用且有效；FFF表示文件的最后扇区；×××（任何其它的16进制字符）表示文件的下一扇区相对扇区号。文件的第一扇区的相对扇区号放在文件目录中。

每一驱动器最后使用的FAT的内容存放在内存中，一旦磁盘空间的状态变化时，把它分别写到0道2和3扇区。

3. 磁盘目录

0道4—7扇区为目录区，总共2048字节。目录区分为64个目录项，每项32字节。因此，一个磁盘最多可放64个可变长度文件。每个目录项格式如下：

0	7 8	10 11	12	23 24	25 26	27 28	31(字节)
文件名	文件扩展名	文件属性	保留	文件建立或最后修改的日期	起始扇区号	文件大小字节数	

4. **磁盘传送区DTA**

DTA（亦称缓冲区）是内存区，供DOS存放磁盘读写的数据。DTA可在内存的任何位置，由程序设置。

5. **DOS初始化**

当系统启动后，引导记录读入内存，控制转移给它。它检查目录，看首先列出的两个文件是否为IBMBIO.COM和IBMDOS.COM，如不是，则发错误信息，如果是，引导记录就从绝对磁盘扇区中读此两文件到内存，然后转移到IBMBIO.COM的第一个字节。IBMBIO.COM的开头是JUMP指令，转移到其本身的初始化代码。初始化代码决定设备状态、复位磁盘系统，初始化设备、设置低编号的中断向量。接着它重新定位IBMDOS.COM在段×'B0'，调用DOS的第一个字节。

DOS的开头也是一条JUMP指令，转移到其初始化代码，初始化其内部工作表，决定FAT日录和数据缓冲区的内存位置，中断向量×'20'至×'26'，以最少的有效段为COMMAND.COM建立一个程序前缀，然后返回到IBMBIO.COM。初始化的最后任务是把COMMAND.COM装入DOS初始化所指定的位置，然后把控制转给COMMAND的第一个字节。

6. **文件控制块FCB**

在涉及文件处理（打开、关闭、建立、删除、重新命名等）过程中，都要与FCB打交道。FCB由标准FCB（37字节）和扩展FCB（7字节）组成。扩展文件控制块用来建立或寻址具有特殊属性的磁盘文件。图3为FCB的格式。其中阴影部分由DOS填入，不能修改，其余由应用程序填入。

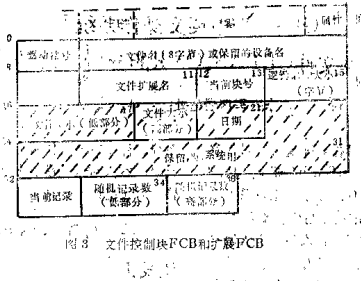

图3 文件控制块FCB和扩展FCB

7. **DOS存储分配**

DOS的存储分配如图4所示。图中的地址形式是：段：偏移量，例如0060:0000表示内存绝对地址×'0600'。

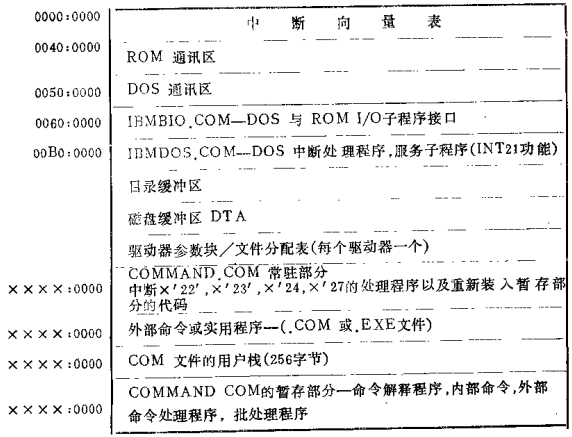

图 4 DOS的存储分配

文献出处:《微计算机应用》1985年 1 期12—17页

GD—OS中英文操作系统字表文件GDBAS.OVR的分析与改错

广东省军区通信处 刘岩川

GD—SO中英文操作系统是中国科学院广州电子研究所专门为IBMPC 研制、生产的中英文操作系统软件。由于该系统采用了LBC—220中英文电脑简单易学、输入快捷的笔形码汉字输入方式,而赢得了广大用户的欢迎。在我们军区范围内使用也比较广泛。但是在使用中发现一些字在使用笔形码输入方式时敲不出来或是张冠李戴。经过分析是字表文件GDBAS.OVR(早期GD—OS系统软件没有该文件而为BXMVT)中存在错误,本人对这些错误进行了修改。现将改错的方法提供给大家参考。

一、GD—OS 中英文操作系统浅析

整个GD—OS中英文操作系统分装在两张5″软盘上,一张为系统盘,另一张字库

盘。它们的文件名及作用如下：

1. **系统盘**

COMMAND.COM	
GDOS.BAT	批处理启动GD—OS
ACB.EXE	为汉字库开辟内存
GDLIK.EXE	汉字输入与显示控制
GDBAS.OVR	汉字键盘输入码与汉字机内码转换字表
PR9.EXE	9针打印机汉字打印驱动
PR24.EXE	24针打印机汉字打印驱动
CZDEF.EXE	定义词组
CZLIK.EXE	连接词组
GENCC.EXE	造字

2. **字库盘：**

CCB1	国际码第一级汉字库（不含1～15区）
CCB2	国际码第二级汉字库与图形符号

3. GDOS.BAT批处理文件内容为：

ACB.EXE	
GDLIK.EXE	
PR9.EXE	（根据用户需要进行调整）
CZLIK.EXE	（根据用户需要进行调整）

二、GDBAS.OVR字表文件分析与改错

1.**作　用**：

GDBAS.OVR操有一个专为笔形码汉字输入方式而设计的汉字机内码字表文件，当用户调用笔形码汉字输入方式（ALT—F4）时，就把该文件调入内存。

2. GDBAS.OVR **汉字机内码的形式**：

GD—OS中英文操作系统采用的是国标码（GB2312—80）即2字节ASCII码作为汉字机内码，且两位ASCII码字节全部高8位置1.例如汉字“啊”的机内码为：B0A1.

它与汉字国标区位码的数学转换过程为：

16（区位码的区号，10进制）	01（区位码的位号10进制）
↓（10进制转换为16进制）	↓（10进制转换为16进制）
10（16进制）	01（16进制）
↓（＋20H）	↓（＋20）
30H	21H
↓（＋80H）	↓（＋80H）
B0H	A1H

3.GDBAS.OVR **汉字机内码表的排序方法与改错**：

GDBAS.OVR汉字机内码表的每个汉字机内码在内存中占用了4个字节。前两个字节的内容是标识符，后两个字节的内容是汉字机内码。这些汉字机内码排序的方法是以其笔形码，大小顺序排的。最开头放笔形码为0的汉字，最后放笔形码为“777”的汉字，例如表的开头为：

是、口　吕、品、器、唱、粤顶……

表的末尾是：

……获、耪、枸、戆、彗。

这样把相邻笔形的汉字放在一起，不用打出某一个汉字的全码，可以很方便地把该字扫描出来。这就是在笔形码输入汉字时打空格键也可以找出汉字的原因。

在机器上可以用8088汇编语言调试程序DEBUG.COM把这个字表显示在屏幕上，作法是：

首先把GDBAS.OVR这个字表文件与DEBUG.COM调试程序拷贝到一张事先准备好的软盘上。在GD—OS系统状态下键入：

①A>DEBUG GDBAS.OVR

②用DEBUG的“R”指令查看

GDBAS.OVR放在内存库中的地址。

```
AX=0000 BX=0000 CX=ACC0
DX=0000 SP=FFEE BP=0000
SI=0000 DI=0000 DS=54B3 ES=54B3
SS=54B3 CS=54B3 IP=0100 NV UP
DI PL NZ NA PO NC 54B3: 0100
0000 ADD [BX+SI], AL DS:
0000=CD
```

可见，GDBAS.OVR这个字表文件被放在偏移地址为DS:100至DS:ABC0这一段内存中。

③用显示内存指令"D"查字表文件的内容。

```
—d4270 ffff
54B3:4270 00 00 00 00 00 00 00 00—A1
    A3 00 00 BF DA 00 00....↑#..? Z.
54B3:4280 C2 C0 00 00 C6 B7 87 00—
    F6 BE 10 00 B3 AA 26 20 B@..
    F7..v〉..3#&
54B3:4290 F2 A6 26 40 F0 CA 26 60—
    B6 F5 40 00 BA C8 43 10 r&&Cpj
    &'6u@.: HC
54B3:42A0 CF 1F 9 60 00 E0 qC 50 00
    —BA D9 20 00 D1 CA 25 00 Oy'
    .'\P.:Y.Qj%.
54B3:42B0 E0 C5 52 00 BF DE 52 10
    —C6 F7 70 00 D6 E4 62 00 'ER.
    ?AR.Fwp.Vdb.
54B3:42C0 C2 EE 00 00 D4 EB 20 00—
    CE B3 30 00 DF C8 30 00 Bn..
    Tk.N90.H0
54B3:42D0 E0 D0 70 00 E0 AF 80 00
    —C1 A8 [illegible] C5 D5 00 0[illegible] Fp./.
    .[illegible]..HI..
```

从内存中可见：

汉字机内码	汉字
A1A3	是
BFDA	口
C2C0	吕
C6B7	品
F6BE	器
B8AA	唱
F2A6	颚
⋮	⋮
⋮	⋮

若想找出某一个汉字机内码在内存中的位置，可以使用DEBUG的搜索指令"S"。

由于该字表文件有七千多个汉字的机内码，文件长度为44224字节，在编写该字表文件的8088汇编语言程序时，总免不了要出现一些错误，而且一些错误在用户大量输入汉字时才会发现，例如："暱"字，笔形码应为"04451"，但在输入"04451"后却出现"皱"字，显然是一个错误。类似还有象：罗、秦、晏、侏等一百多个字，这些字在笔形码汉字输入方式时敲不出来。修改时只要一一换算出这些汉字的机内码，并找出放该汉字机内码的内存地址，然后使用DEBUG的修改指令"E"就可以很方便的进行修改。除此之外还可以把一些常用的标点符号如：逗号、句号、引号等放入笔形码表中去，例如：

笔形码	标点符号
3	,
4	、
0	。
33	"
66	"

修改后的字表文件GDBAS.OVR，经过运行证明是可行的，给使用者带来方便。若GD—OB（彩色显示器配置也可）的用户需要修改后的字表文件GDBAS.OVR，我们将提供方便。

文献出处：《电脑与微电子技术》1986年3期18—19页

PCDOS3.0版中的修改和使用注意

Ray Duncan 覃余生译

IBM公司在推出PC/AT计算机的同时宣布了PC DOS3.0版磁盘操作系统。与2.0版相比，新版本包括了一些新增的或修改过的功能调用，但占用的内存较大，在带有硬盘的PC/AT上它占48K字节。因为DOS技术手册至今仍只有极少数人有，因此借这篇短文概括介绍一下PC DOS 3.0中所作的重大修改。

PC DOS 3.0最为人注意的一点是，PC DOS的开发者Microsoft公司退居幕后。事实上，在有关的报导和各种DOS手册上"Microsoft"这一字已完全销失。只在盘片的标签上还能找到非常小的"Microsoft"的印字。

现在回过头来讨论PC DOS 3.0。PC DOS 3.0对紧急错误处理程序中断（Int 24H）作了小修改，增加了用Ah寄存器传送的某些新的状态信息，而且在返回时增加了一个新的DOS响应。（应用程序可以要求使违法的系统调用失效，不理采、重执或终止程序。）有 错误中断的文字说明，以前是相当粗糙的，现在虽还有些模糊，但已经变得可以理解了。

Int 2FH是DOS 3.0上新增加的。它本质上是应用软件和打印假脱机之间的接口。一个用户程序可以把文件加入到打印队列，注销文件或检查等待文件的假脱机表。

在Int 21H总标题下(DOS功能调用)有以下三个功能作了修改。

功能38H（取出或设置国家名称）对功能38H作了如下所述的根本性的扩充：

1. 可以设置国家名的代码和询问国家名
2. 有总数大于255个的国家名代码
3. 可以在数据块内送回更多的有关国家名的特定的限定符和格式等信息。

功能3DH（打开文件）增强了3DH的功能，使其可以支持多任务和联网环境。当一个文件被打开后，应用程序可以指定该文件是否可以为运行在同一网络节点上的一个子进程所"继承"，或者可以为运行在同一个网络节点或另一个网络节点上的另一个作业所"共享"（即被独立地打开）。

功能44H（输入/输出设备控制）该功能新增加了两种功力，询问某一块设备是否有可卸的介质；当发生"争用共享文件"时设置重执次数和规定两次重执间的延迟时间。

Int 21H还有下述五种新增的功能调用。

功能59H（取详细出错信息）在另一功能送回错误码后，应用程序可通过调用这一功能来取出有关故障。故障的类别（暂时性故障、系统内在故障、硬件故障，等等）和应用程序宜作出的响应（包括重执，延迟后重执，中止）等较详细信息。

功能5AH（建立暂存文件）由应用程序把一个路径串（Path String）传送给功能5AH，功能5AH产生一个唯一的文件名字符串；在指定的子目录内建立文件，并回送一个完整的路径和文件说明串。此后该文件可以被应用程序打开，并可以用任何一种希望的方式来使用它。当应用程序结束时，不自动删除该文件。

功能5BH（建立文件）除了当文件已经存在时失效这一点外，它与DOS 2.0中的

3CH（建立文件）完全一样。另一方面，功能3CH以同一文件名把先前已经有的文件截成长度为零，然后返回成功码。

功能5CH（闭锁/解锁文件访问）功能5CH为一个作业对取得的文件中某一部分的专有访问权（即便该文件正在被其它作业所共享）提供一般的控制机构。

功能62H（取程序段前缀地址）其功能是不言而喻的。它方便了程序人员写EXE类型的程序。

从PC DOS3.0可清楚地看出，尽管PC DOS 3.0或它以后的产品最后将完全融入TopView或IBM公司其它的专利产品中，PC DOS操作系统继续在向结构复杂化和功能加强的方向发展。为了达到顺利使用的目的，以下提出若干指导原则，以便利程序员处理即将来临的多任务和联网的新环境。

·用Extended File（扩充文件）功能（3CH-42H）来打开、关闭、读和写文件，而不是用老的“FCB”（文件控制块）类型的调用。这种调用是从CP/M中引用来的。Extended File功能的加强提供更好的出错信息，并支持分层的文件结构和文件共享。

·用“修改存贮块分配”调用来释放应用程序不用的任何存贮空间，同时，必须检查程序段的前缀，以求出分配给应用程序的总的内存容量，并且不要越界访问（即便已查明实际存在内存容量更大）。

·不要直接地访问断矢量，而应分别用DOS的功能调用25H和35H来设置和取出中断矢量的内容。这对Top View和对PC/AT中的80286微处理器是有意义的，80286可以在保护模式下支持多个中断矢量表。

·在你的程序取得控制权后，用功能30H取出DOS的功能号。如果程序运行于DOS 3.X下，当一个DOS功能失效时可以调用功能59H，以取得更详细的出错信息和建议的操作。总之，使用紧急出错处理程序的中断能力，使你的程序可以更宽恕硬件故障。

·用Exec调用产生其它任务或装入重叠；不要自行建立程序段前缀。

·如有必要，可以直接向显示缓冲器写入数据，但必须通过调用ROM中的BIOS来完成所有工作模式的改变。要使所有显示代码访问一个包含有缓冲器段地址的变量。这样，对一个程序进行修改使之适于在Top View下工作的任务，可大大简化。（在Top View中分配一个“影象缓冲器”）。

·如要向一个标准设备写入文件，则尽可能发送字符串。而不要送单个字符。要利用缓冲和DOS驱动程序中的快速传送逻辑。反之，当DOS已经在你的应用程序和设备间建立了多层缓冲时，就不要把程序资源浪费在记录或字符的内部组块和解块上。

·在退出应用程序前，必须十分小心地关闭所有文件控制块和（或）处理程序。如果文件的某些区域被闭锁的话，那末在关闭该文件前要将它们解锁。

·通过Int 21H的功能4CH，以返回代码终止你的程序，而不是利用以前常用的功能0或Int 20H。调用进程或批子命令IF和ERRORLEVEL可以查询返回码。

·如写入一个常驻驱动程序，可用Int 21H的功能31H来终止和保持驻留，而不用以前惯用的Int 21H。这样可允许传送返回信息，而且允许驻留程序大于64K字节。

文献出处：《微型计算机》 1986年3期79—80页

UNIX操作系统低级调度的可靠性分析

宋 建 云

UNIX操作系统短小精悍而功能又完备，已博得了普遍的赞誉。近年来，国内的UNIX用户迅速增加，UNIX更是愈来愈多地引起了人们的注意。许多人对此进行了一系列研究。但是对UNIX进程调度（低级调度）的分析评价方面，某些结论是值得商榷的。

大家知道，进程调度是整个操作系统的核心，这不仅仅是因为CPU管理的优劣直接影响着操作系统的运行效率，处理机利用的任何一点浪费都将使整个计算机系统资源的利用率大大下降，而更主要的是因为进程调度描述了系统中全部并发活动的进行，关系到整个系统的工作正常与否。

因此"UNIX操作系统的模块化设计"[2]（以下简称"模块化)一文及《UNIX的操作系统分析报告》[3]中对现今UNIX的低级调度的可靠性的质疑，就不可避免地会引起用户们的关注。经过细致地分析讨论，下面仅就这个问题谈一点看法，供大家在分析研究UNIX操作系统时参考。

UNIX进程调度程序swtch（）与一般操作系统的进程调度不太一样，大多数操作系统的进程调度并不从属于某个进程，而是作为所有进程的转接站看待，但是UNIX的swtch（）却是0#进程的一部分出现，甚至调用swth（）象调用任何其它子程序一样方便，它不封锁中断，也不保留处理机状态（PS），因此也难免造成"在UNIX系统中，进程之间的接口不清晰"。但仅是根据以上两点便认定，UNIX的swtch（）将使"一个进程对另一进程产生隐蔽影响"，并不是十分合适的。

让我们先来看一看一个设想的进程转换模型（参图1）：spl5（）、spl6（）分别置当前处理机优先级为5级和6级，setrun（P_1）、setrun（P_2）分别唤醒进程1和进程2。"模块化"一文认为，如果进程2先

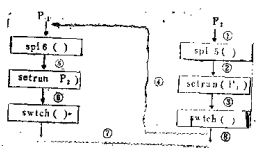

图1 假想的进程转换模型

执行，控制路线如图中标号所示，那么"进程2优先级为5，而到了⑧时优先级就被改成了6，如果进程1先执行，则效果不同"。而且进一步认为，"并发程序的这种执行效果，在本文上看不见，而与进程运行的相对速度有关。这些操作系统失败的主要原因"（着重号为引者所加），于是在文章的后半部分提出了改造UNIX的一个方案。

然而这种叙述是不确切的，在swtch（）的执行中（参见图二），savu（）和retu（）二个汇编子程序是必定要执行的。重要而容易忽略的是，执行save（）、retu（）（或者aretu（））时，为了屏蔽中断，都要修改处理机优先级，做完后开中、即无论调用它们时CPU优先级是多少，返回时优先级总是为0；因此即便上述的假想模型存在的话，进程2执行到⑧时优先级决不会调6而只能是0，进程1先执行结果也是一样。当然，高优先级的进程经过一次出让处

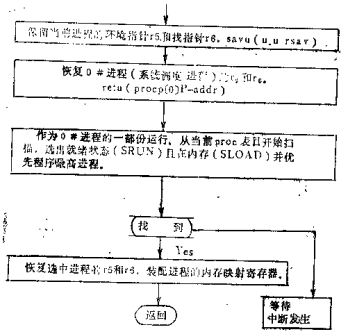

图2 swtch()的工作流程

理机（用是swtch()）后，就把自己的优先级降成0了，这样的系统仍然是不可靠的，但如果我们在更广阔的背景下来看对swtch()的调用关系，就会发现在UNIX的制约下，上述的进程转换模型是不可能存在的。对swtch()的调用大致可分四种情况讨论：

发生了中断或捕俘并在相应的处理完成以后其先前态为用户态的进程就检查是否有比现进程更优先的在等待（参见 xrap() 查询runrun标志），若有则调swtch()将投入运行，这时进程优先级可能发生转换。但是我们可以发现，这里的swtch()还处在中断或捕俘的总控制程序的环境中，swtch()返回后马上恢复中断现场并退出断，swtch()对ps的影响被抵消了。（而且从另一方面看，当前被投入的进程是用户态的，优先级为0级，因此总是安全的）。

现进程因某种原因需要睡眠等待，则调用sleep()，由sleep()再调用swtch()。这几乎是调用swtch()的最经常的形式，而且它可能在任何优先级下被使用。但很显然，UNIX设计者对此已有所考虑，sleep()中调用swtch()前是保留ps的，最后恢复ps再退出sleep()，因此这种情形下使用swtch()也是没有副作用的。

现行进程希望扩大用户区或要求连入一正文段时（参见expand()和xalloc()程序），有可能调用swtch()。expand()和xalloc()与sleep()不同，不保留ps，但经过仔细检查可以看到expand()除在系统初启时被main()调用（这时不会调用swtch()）外，总是被一些系统调用程序直接或间接地调用，xalloc()也是同样。容易理解，用户只要申请操作系统服务，就必须得通过系统调用程序，而系统调用是以0优先级运行的（参见tsap()）。

现在进行调用exit()终正自身，或在子进程（现行进程）要求父进程跟踪时发现了软中断信号，调用stop()自动进入停止状态，放弃处理机。但最终我们还是可以确认，它们的运行环境也是0优先级。

综上所述，可以认为UNIX的swtch()在使用上是相当节制的，虽然它分离开来看或许有一些不可靠的因素，但从整个系统看可靠性是保证的。它的设计者为了增加系统的可靠性，在保证系统效率的前提下已把并发性降到了很低的程度。核心态进程除非由于等待资源，否则不会插入别的进程的代码（中断也不导致交出处理机），甚至一般父子进程也不并发操作，一个常见的例子就是shell命令的执行过程。

关于swtch()的可靠性问题，我们提出了上述的看法，错误疏漏之处谨请大家指正。

文献出处《计算机研究与发展》1985年2期64—65页