联想计算机丛书之一

LEGEND

DOS操作系统内核剖析

（下册）

第二部分　控制进程

周利华　李凤华　编著

西安电子科技大学出版社

DOS 操作系统内核剖析

（下册）

第二部分　控制进程

周利华　李凤华　编著

西安电子科技大学出版社
1991

内 容 提 要

作者对DOS操作系统的几个较高版本的核心程序进行了完整、详细的分析，同时结合从事DOS操作系统开发和改造的实践经验，全面地介绍了DOS操作系统的设计思想及其具体实现，为读者提供了DOS操作系统内核的完整信息。

《DOS操作系统内核剖析》分上、下册出版。本书系下册，又分为第一部分文件系统及第二部分控制进程，由三章正文和五个附录组成，详细介绍了DOS操作系统的文件系统、控制进程和命令处理器，并提供了IBMDOS.COM、COMMAND.COM的源程序注释清单，在附录B中给出了包括保留功能调用在内的所有DOS功能调用的详细信息；此外，为了读者方便地阅读源程序，附录E提供了DOS的三个核心文件的源程序索引。

本书适合于计算机系统软件开发人员、微机开发和应用人员参考，也可作为大专院校"操作系统"、"系统程序设计"等有关课程的教学参考书，本书也可供从事微机的加/解密和计算机病毒防治等方面的技术人员参考。

DOS操作系统内核剖析(下册)

第二部分　控制进程

周利华　李凤华　编著

西安电子科技大学出版社出版

通县兰空印刷厂印刷

开本 787×1092　1/16　印张 51 14/16　字数 1 555 千字

1991年5月第1版　1991年5月第1次印刷　印数 1—3 000

ISBN7—5606—0154—5/TN·0050　　定价：20 元

目　录

第二部分　控制进程

第七章　控制进程

操作系统最基本、最核心的任务是对程序的管理与控制，但在单任务的 DOS 操作系统中，程序的管理与控制要比多任务、多用户等操作系统简单得多，这主要是因为它不涉及到进程之间的调度和进程因调度而需要对其运行状态、环境信息进行保存。在 DOS 操作系统中，一个程序一旦被装入内存并获得控制后，它几乎控制了系统的全部资源，除当前运行程序及其调用的程序之外，在它结束之前，其它先前被加载到内存的程序不被执行。DOS 操作系统的加载是通过 EXEC 功能调用 4BH 来实现的，它供系统程序或应用程序（称其为父进程）把另一个程序（称其为子进程）加载到内存中，由父进程决定子进程是否执行，并且父进程还能决定子进程加载到内存的位置（如：装入覆盖等）。当子进程结束之后，控制将返回到父进程。

对 EXEC 功能调用来说，发出请求调用者是父进程，被加载的程序是子进程，它们之间的父子关系是由调用次序决定的，因而子进程也可以用 EXEC 功能调用加载属于它的子进程，如此下去，形成执行程序的单链形式的控制结构。

命令处理器（COMMAND. COM）就是通过 EXEC 功能调用来加载应用程序或系统程序（EXE 文件或 COM 文件）的，但是 COMMAND. COM 本身就是一个可执行程序，它也可被应用程序或它自己加载执行，因而在应用程序的控制之下，允许使用 DOS 的命令，如 DOS 内部命令 DIR、COPY、TYPE 等；或通过它来执行 DOS 的外部命令，如："command/c cmd"，其中 cmd 为一个存在的 DOS 命令名说明串。用户熟悉的字处理程序和数据库管理程序就是利用 EXEC 功能调用去运行其它程序（如拼写检查程序）的或允许用户在不中止主程序的情况下列文件目录、复制文件或重命名文件等。

由于父进程加载子进程是要求子进程去完成一定的工作，因而它们之间就存在着运行环境的继承和传递、程序的结束处理等。本章详细地讨论上述问题，并介绍面向调试工具、TSR 程序使用的功能调用。

§7.1　可执行文件结构

在 DOS 操作系统下，可执行文件有 3 种：COM、EXE 和 BAT 文件。BAT 文件（批文件）是由命令处理器处理的一系列命令集合的文件，它的每一项（每一个命令串）由命令处理器解释并调用相应的处理程序进行处理（有关 BAT 文件的批处理将在 § 8.3 节介绍），而 COM 文件和 EXE 文件是可以由 EXEC 功能调用加载并予以执行。因此可执行文件实质上只有两种结构：COM 和 EXE 文件。

7.1.1　COM 文件结构

COM 文件实际上是二进制程序代码在内存中的映象，它总是紧跟在程序段前缀（简称 PSP）后装入，加载过程中不进行段重定位，因而它具有如下特征：

• 该程序只能有一个段（即代码段、数据段、堆栈段共用同一段地址）；

• 此类程序的总长度不超过 64KB；

• 程序头必须预留 100HB 的空间供本程序的 PSP 使用，且在偏移 100H 处必须是一条可执行指令；

• 该程序的子进程必须具有近过程属性(NEAR)。

COM 文件结构紧凑、装入速度快，它不需访问几个段空间，但它受到 64KB 的使用空间限制，因而只适宜小程序；亦可作为一个大型应用程序的引导程序，由它利用 EXEC 功能调用装入主程序。现在大多数 COM 文件都是一些老程序(或许是从 CP/M 移植过来的)，或者是一些小的实用程序。PC 机上的第一代语言通常只能产生 COM 文件，而现在大多数编译程序都支持大的存贮模式，并产生 EXE 文件。

7.1.2 EXE 文件结构

为了突破 COM 文件的长度不超过 64KB，以及程序执行入口地址偏移固定为 100H 的限制，尤其是当可执行程序处于多用户实时任务的环境下，多用户 DOS 要求将一个大程序的不同段分别加载到内存的不同区域，而指定某一个或几个纯代码段为其它任务共享。EXE 文件结构能解决 COM 文件的限制，其文件长度仅受到当前可用内存空间的限制。

大程序以及需要分配和释放存贮空间的程序被构造为 EXE 文件，EXE 文件的重要特性是它可以使用尽可能多的内存段，即代码段、数据段和堆栈段可以使用不同的内存段，在装入时可以重定位。

EXE 文件的结构特征可归纳如下：

• EXE 文件的长度仅受到当前内存可用空间的限制，它允许建立若干个不同名的代码段、数据段、附加数据段、堆栈段；

• 程序中各个子进程的属性既可为 NEAR，也可为 FAR，随段内和段间调用而定；

• 程序执行入口地址随具体应用由编程者确定。

• 由若干个不同的目标模块连接生成的 EXE 文件，可按应用要求将每一模块内代码段、数据段、附加数据段，以及堆栈段取相同名字或不同名字。但要保证：其中只有主模块指明程序执行入口的起始标号，并至少有一个具有“STACK”属性的堆栈段。

与 COM 文件被成块地装入不同，DOS 的 EXEC 功能调用必须得到一些有关 EXE 文件的信息才能正确地装入 EXE 文件并为它分配所需的存贮空间。每个 EXE 文件都有一个 EXE 文件头(如表 7.1 所示)，其中关于重定位程序代码段、数据段、堆栈大小、最小运行空间，以及它将最多使用多大存贮空间的信息都记录在 EXE 文件头中。整个 EXE 文件结构如图 7.1 所示。

表 7.1

```
EFH    Struc
    EFH_Signature        DW   4D5AH/5A4DH   ;EXE 文件特征标志
    EFH_LengthInSector   DW   ?             ;文件映象长度除以 512 的余数，即文件占用的最后扇
                                            ;区长度
    EFH_Pages            DW   ?             ;包括 EXE 文件头在内的文件页长度
    EFH_RelocationItems  DW   ?             ;重定位表的项数
```

```
EFH_HeaderLen       DW  ?       ;EXE 文件头的节长度
EFH_MinSpace        DW  ?       ;被装入程序上方所需最小内存节数
EFH_MaxSpace        DW  ?       ;被装入程序上方所需最大内存节数
EFH_SS              DW  ?       ;被装入模块中堆栈段的相对段值,它由重定位因子调
                                ;整,参阅图 7.3
EFH_SP              DW  ?       ;被装入模块取得控制权时,SP 寄存器值
EFH_CRC             DW  ?       ;文件中所有字的负累加和
EFH_IP              DW  ?       ;被装入模块取得控制权时,IP 寄存器值
EFH_CS              DW  ?       ;被装入模块中代码段的相对段值,它由重定位因子调
                                ;整,参阅图 7.3
EFH_FirstRelocation DW  ?       ;重定位表的第一个重定位项在 EXE 文件中的位移(从
                                ;文件头开始,以字节为单位)
EFH_OverlayNum      DW  ?       ;覆盖号(0:程序驻留部分)
EFH Ends
```

注:一节为 16 字节

注:一页为 512 字节

图 7.1 EXE 文件结构

EXE 文件的程序代码可以使用功能调用 49H 或 4AH 释放不需要的内存空间。因而,EXE 文件虽然多附加了一个 EXE 文件头,但这给它带来了更强的灵活性。值得注意的是:EXE 文件头的大小是 512 字节的整数倍,至少占一个扇区单位(512 字节)长度。

在早期的 DOS 中,灵活的内存分配和释放并不显得重要,因为通常内存中只有一个程序。但从 DOS 2.0 版本开始,内存常驻程序的出现改变了这种情况,现在,绝大多数 DOS 环境下运行的大型程序都应按需要释放或申请内存空间。但一个扰乱整个内存的程序将会使那些需要根据情况分配内存的常驻程序处于冻结状态。

§7.2 环 境 块

我们在使用计算机时经常设置命令检索路径(PATH 命令)、改变系统提示符(PROMPT 命令)等,这些信息是决定命令处理器工作方式所需要的。同时,在一些系统软件中常常需要建立一些路径说明,如 Microsoft C4.0 编译程序要求在首次编译前运行一个批文件,其批文件的参考格式如下:

```
path=\bin
set tmp=\tmp
set lib=\lib
set include=\include
```

此外，尽管大型系统软件或应用软件功能齐全，但用户在使用过程的某一段时间内只会使用其中一部分功能，因而它们往往是一些程序集，在执行过程中根据使用的需要而确定加载对应的程序并执行它，以便完成相应的工作。父进程加载子进程时，经常需要给子进程传递一些信息（如设置工作路径等）。我们称上述信息为环境，这个环境指明了可由相关程序使用的特定参数。

“环境块”是 DOS 用于父进程与子进程之间传递环境信息的一种机制。环境块不仅可被逐级继承，而且也可被逐级设置新的；同时，如果任何一个进程“结束且驻留”，环境块又处于静止不变的状态。不论系统处于哪一个层次运行，DOS 都为该层次的程序设置一个“环境块”，也就是在加载程序本身之前，先申请一个存放环境串信息的内存块。

DOS 允许每一个环境块的最大容量为 32KB，环境块的内容对操作系统本身没有任何影响；甚至组成环境块的信息仅为被加载程序利用，对 DOS 无任何意义的关键字或参数组成的 ASCIIZ 字符串集，然而被加载的应用程序却可编程检测并解释它们。但是环境块的管理、复制是由 DOS 内核完成的，它在内存中的位置是随机的，为了能让被加载的子进程知道它存放的位置，将它的段地址存于程序段前缀 PSP+2CH 处（参阅 § 7.3 的表 7.2），这一规定使得系统程序和应用程序能方便地访问它。

7.2.1 环境块信息

环境块信息是由一系列以 0 为结尾的 ASCIIZ 字符串组成的，每个这样的 ASCIIZ 字符串对 DOS 或应用程序有特殊意义，整个环境块也是以 0 结尾的。每个 ASCIIZ 字符串的形式如下：

```
name=parameter
```

等号左边是环境变量的名字，右边是该环境变量的串参数。通常，环境块含有以下 3 个环境串：

```
'path=',0
'comspec=c:\command.com',0
'prompt=',0
```

上述 3 个环境变量：PATH、COMSPEC 和 PROMPT 是 DOS 保留的（有关这 3 个环境变量的功能实现请参阅 COMMAND.COM 程序清单），其中：PATH 环境变量规定了命令执行时使用的命令检索路径，因此，当键入一个 DOS 外部命令后，若键入的 DOS 命令在指定的目录或当前目录下未找到对应的 DOS 命令文件，则 DOS 知道到那一个目录路径去查找此命令。COMSPEC 环境变量总是在 DOS 启动时设置并定义命令处理器的路径和名字（通常的命令处理是 COMMAND.COM，但也可以使用另外的命令处理器），它指明了命令处理器所在的位置。当 DOS 外部命令结束时，若命令处理器的暂驻程序代码被覆盖，则常驻程序代码知道到何处再次装入命令处理器的暂驻程序代码。PROMPT 环境变量指定了系统提示符的形式说明符。

若用户未在 AUTOEXEC.BAT 批文件(或其它已执行过的批文件)或随后的命令行上输入命令 PATH、PROMPT,则 DOS 对它们选择缺省值,即命令检索路径为当前目录,系统提示符是驱动器名和“>”。一旦在命令行上使用了 PATH、PROMPT 命令,则它们都会被命令处理器自动地加入到环境块中。

此外,DOS 还允许在命令行上,通过输入 SET 命令向环境内存块追加新设置的环境串,或者删除、修改原有的环境串(包括 DOS 保留的 PATH、COMSPEC、PROMPT 三个环境串)。方法如下:

```
set name=parameter
```

如果:

(1) name 是新建的,则相应的环境串追加到环境内存块中;

(2) name 是原有的,则以等号右边的新串参数代替原串参数,并且整个环境串被调整到环境块的最后;

(3) 只输入 name 且已存在,则原有的环境串被删除。

既然,在 DOS 命令级上允许设置环境串信息,以便在加载一个应用程序时,环境块由子进程继承,从而达到传递信息的目的;那么,当该应用程序处于父进程的地位,并通过 EXEC 功能调用加载另一个子进程时,也就允许设置环境串信息传递给被加载的子进程。由父进程在 EXEC 功能调用的参数块(参阅附录 B.093、B.094)中设置一个环境块的段地址变量,这有两种选择方法:

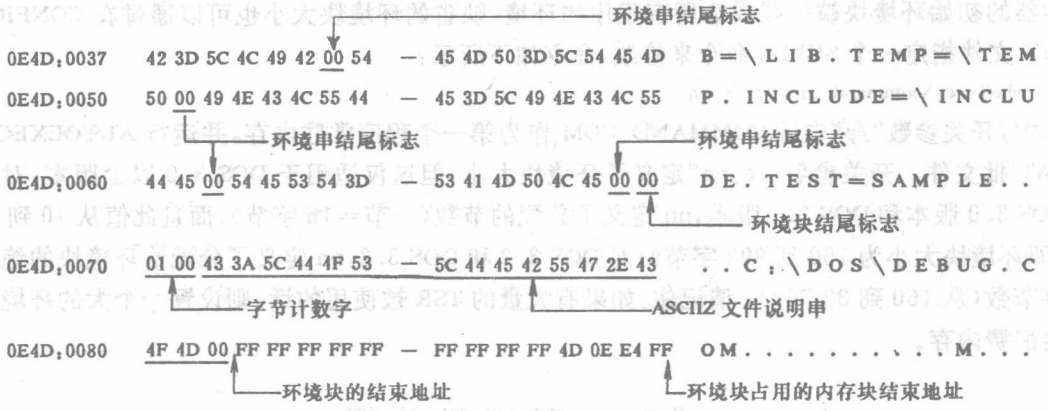
(1) 若该变量为 0,则表示父进程的所有环境串信息复制给子进程,由其全部继承并不再添加新的环境串信息;

(2) 若该变量指向一个新建的环境块,则该环境块内含有的环境串信息被子进程继承,并传递给子进程,如同在命令行上键入 SET 命令。

充分地利用环境变量,用户可确定在一个程序运行时特殊应用的数据文件被装在什么地方,使用此方法,你不必给那些将被你的应用程序引用的文件指定目录路径名,这一特性可用来支持那些被单个程序引用的唯一的多个数据集,批文件能够修改环境变量,指出所期望的数据。在 Mircosoft C4.0 中,功能 GETENV 提供了环境变量的引用。对其它语言,你还必须找出环境空间以安装那些感兴趣的变量。

零字节紧跟在所定义的所有环境串最后,对于 DOS 2.X 版本,环境块在此结束。但从 DOS 3.0 开始,环境块的最后是一个所谓字节计数字,然而,你将发现这个值总是“1”。下一个是指定现行应用程序的包括完整路径名的 ASCIIZ 文件说明串,这个 ASCIIZ 文件说明串在打印常驻内存程序的存贮器映射图时被使用(参阅上册 § 5.4.4 节提供的 MCB.ASM 程序)。一个环境块实例如下:

```
-D E4D:0 9F
0E4D:0000   43 4F 4D 53 50 45 43 3D  -  41 3A 5C 43 4F 4D 4D 41   COMSPEC=A:\COMMA
0E4D:0010   4E 44 2E 43 4F 4D 00 50  -  41 54 48 3D 43 3A 5C 3B   ND.COM.PATH=C:\;
                                  └──── 环境串结尾标志
0E4D:0020   43 3A 5C 42 49 4E 3B 43  -  3A 5C 44 4F 53 3B 44 3A   C:\BIN;C:\DOS;D:
0E4D:0030   3B 00 50 52 4F 4D 50 54  -  3D 24 70 24 67 00 4C 49   ;.PROMPT=$p$g.LI
            └──────── 环境串结尾标志 ────────┘
```



7.2.2 在批文件中使用环境变量

环境变量的特性很容易被用到批文件中，例如，假定你已在 DOS 命令行下发出了命令："set ver=sample"，那么，变量 ver 将在批文件中通过括在两个百分号%内定义，并被命令处理器的暂驻程序代码中 FormBatcmd 子程序解释，以便用 ver 环境变量的串参数取代%ver%。下面的命令行：

```
if %ver%=="sample" type help.txt | more
```

用在批文件中，以便通过 DOS 的"MORE"过滤器输出一个"帮助文件"。管道"|"操作符指示 DOS 将第 1 个命令(TYPE HELP.TXT)的标准输出改为第 2 个命令(MORE)的输入(参阅 §8.2 节)。

批文件的另一个使用技巧是以下面的方式来使用环境变量的，即在一个批文件内允许选择调用子进程，当这些子进程结束后，控制返回到批文件中的由环境变量指定的标号，这是一个与 GOTO 语句配合使用的典型例子，下面的批文件描述了这一技巧。

```
set label=one                    ;设置环境变量
goto subroutine                  ;控制转到 subroutine
:one                             ;标号
set label=two                    ;修改环境变量的串参数
goto subroutine                  ;控制转到 subroutine
:two                             ;标号
goto end                         ;退出批文件
:subroutine                      ;标号
echo inside of subroutine        ;提示信息
goto %label%                     ;控制转到环境变量的串参数定义的标
                                 ;号处
:end
```

环境变量增加了批文件的编程范围，特别是 DOS 3.3 增加了一个批文件将另一个批文件作为子进程来调用的功能(参阅 §8.3 节)，使批处理语言更接近真正的编程语言。

7.2.3 扩展环境块空间

环境块的大小随其内容变化而扩大或缩小，它占用的内存空间以节为单位，最大可扩至

32KB。但当设置环境变量时，可能会遇到如下出错信息："Out of environment space"，命令处理器的初始环境块被设置成为所有程序的环境，缺省的环境块大小也可以通过在 CONFIG.SYS 文件指定一个 SHELL 命令来修改，命令如下所示：

```
shell=c:\command.com /p /e:nn
```

其中，开关参数"/p"告诉 COMMAND.COM 作为第一个程序常驻内存，并运行 AUTOEXEC.BAT 批文件。开关参数"/e:nn"定义了环境块大小，但这仅适用于 DOS 3.0 以上版本，对于 DOS 3.0 版本和 DOS 3.1 版本，nn 定义了分配的节数(一节=16 字节)，而且此值从 10 到 62(即环境块大小为 160 到 992 字节)；对 DOS 3.2 和 DOS 3.3，nn 定义了分配给环境块的绝对字节数(从 160 到 32 768)。请记住：如果有大量的 TSR 被使用的话，则设置一个大的环境块会浪费内存。

§7.3 程序段前缀

环境块只包含本级程序运行时周围所处的系统状况和某些信息，但父进程所在的位置，传送给被加载程序利用的参数，甚至环境块在内存中的地址等都需要一个控制数据结构(称为程序段前缀，简称为 PSP)告诉被加载程序，它决定了子进程将如何工作，以及如何返回到父进程。PSP 的结构如表 7.2 所示，各字段表示的意义在本节稍后详细介绍。

表 7.2

```
PSP         Struc
  PSP_INT20H              DW      0CD20H
  PSP_EndOfMemAllocated   DW      ?
  PSP_Reserved1           DB      ?
  PSP_CallToDOS           DB      5  Dup(?)
  PSP_INT22HEntry         DD      ?
  PSP_INT23HEntry         DD      ?
  PSP_INT24HEntry         DD      ?
  PSP_ParentPSP           DW      ?
  PSP_SFTIndexTable       DB      20  Dup(?)
  PSP_EnvironPtr          DW      ?
  PSP_UserSP              DD      ?
  PSP_SFTIndexSize        DW      20
  PSP_SFTIndexTablePtr    DD      ?
  PSP_Reserved2           DD      -1
  PSP_Reserved3           DB      20  Dup(?)
  PSP_INT21H              DW      0CD21H
                          DB      0CBH
  PSP_Reserved4           DB      2  Dup(?)
  PSP_ExtFCB1             DB      7  Dup(?)
  PSP_FCB1                DB      16 Dup(?)
  PSP_FCB2                DB      20 Dup(?)
```