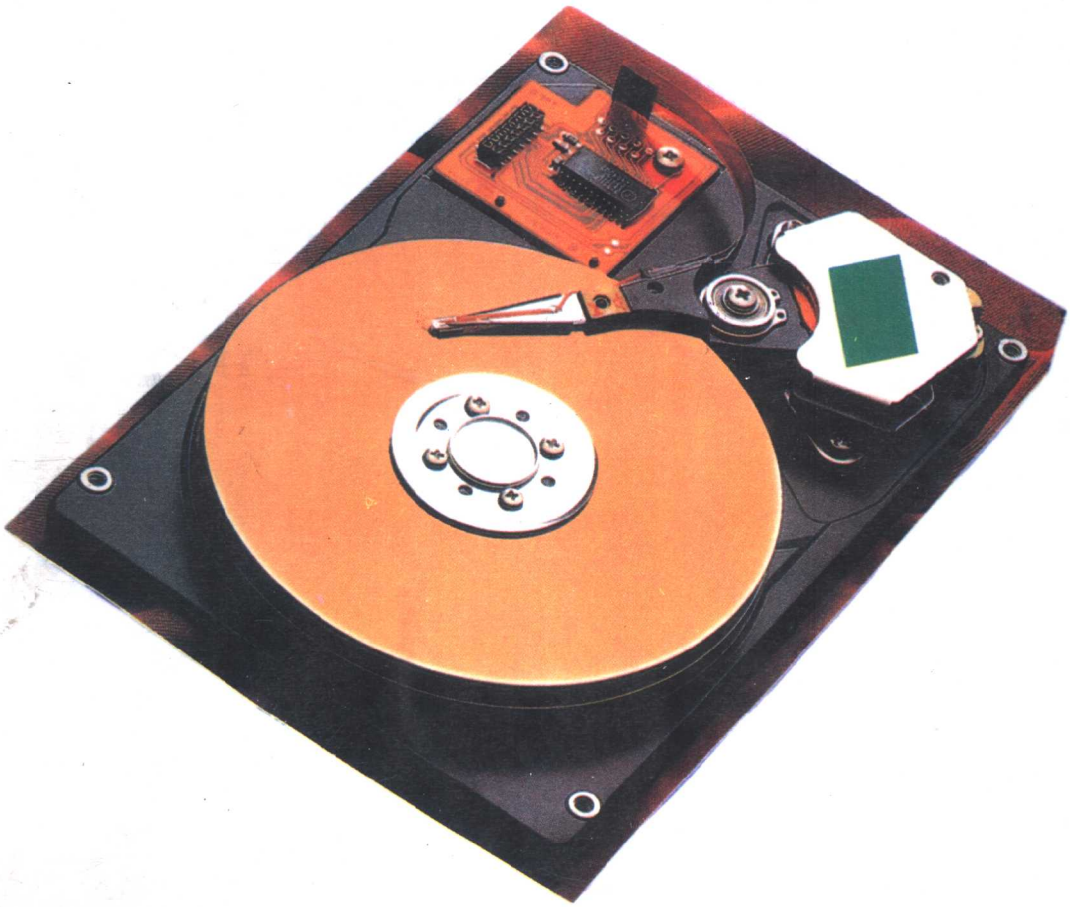
学苑出版社



DOS的全面剖析与编程启示

黄 山 万玉丹 编著

微机操作系统系列丛书

DOS 的全面剖析与编程启示

黄　山　万玉丹　　编著
刘良观　章立生　希　望　　审校

学苑出版社
1993.

(京)新登字 151 号

内 容 提 要

本书对 DOS 3.30 及 DOS 5.0 的内部结构进行了全面剖析，并给出了 DOS 各个重要环节的流程图，包括 DOS 初始化流程，设备驱动程序的流程，DOS 内核各功能的流程以及 COMMAND 的流程等。同时讨论了可为编程提供的帮助，提出了大量的观点、模型及编程实例。该书是作者数年深研 DOS 内部结构的心血结晶，相信对计算机软、硬件、技术人员、大专院校师生是一本不可多得的参考书。

欲购本书者，请与北京 8721 信箱联系，邮政编码 100080，**电话** 2562329。

微机操作系统系列丛书

DOS 的全面剖析与编程启示

编　　著：黄　山　万玉丹
审　　校：刘良观　章立生　希　望
责任编辑：徐建军
出版发行：学苑出版社　　**邮政编码**：100032
社　　址：北京市西城区成方街 33 号
印　　刷：兰空印刷厂
开　　本：787×1092　1/16
印　　张：52.875　　**字　数**：1218 千字
印　　数：1—5000 册
版　　次：1993 年 12 月北京第 1 版第 1 次
ISBN 7-5077-0804-7/TP・15
本册定价：78.00 元

学苑版图书印、装错误可随时退换

前　言

历史悠久的DOS已经经历了十多年的发展历程，而程序员及用户对DOS的认识也逐步深入。如果说，在DOS发展的早些时期，有关DOS的技术资料只限于介绍如何使用DOS命令，以及利用INT 21H API编程；那么，近一段时期则越来越深入到DOS内部，揭示了大量未公开的DOS内部数据结构和保留的功能调用；此外，DOS的源程序清单及详尽注释也已公开；以上这些方面的工作将人们对DOS的认识推进到前所未有的深度，从而对微型机上的编程技术也产生了极大的影响。

本书的作者曾经耗费数年时间，对DOS 3.30及DOS 5.0进行了全面剖析(见参考文献【3】)；同时，作者也充分参考了已出版的大量介绍DOS内部奥秘的上乘之作，结合自己的工作，编著成此书。本书的特点是：

首先，给出了DOS各个重要环节的流程图。虽然完整的程序清单足以包罗万象，但经过提炼、概括，将使得读者更能把握其脉络和精髓。所以，本书对于DOS的初始化流程、DOS设备驱动程序的流程、DOS内核(INT 21H)各功能的流程以及COMMAND的流程等，均根据DOS的源程序进行了精心的总结。

其次，在浩如烟海的DOS技术文献中，由于专业名词及转译等原因，存在着不少晦涩难懂或模糊不清的地方。比如设备驱动程序的所谓"部件号"，文献上用词就相当含糊，使得不少人以为"部件号"就是逻辑驱动器号，其实并非如此。本书对于类似这些不太透彻或常常误解的概念均着重进行了一些辨析，阐述了其来龙去脉。相信读者在读到散见于本书各章节的上述知识点后，顿会发现自己长期以来竟存在着不少并不十分正确的概念。这些知识点均是通过直接分析DOS源程序并加以总结而来，因而具有相当的可靠性，叙述也更清晰。

再次，本书在全面剖析DOS的基础上，还从多个侧面探讨了这些知识对于我们编写应用程序必然会带来的巨大影响。在本书中各个章节，几乎都在介绍了DOS的内部结构之后，紧接着就讨论了可为编程提供的帮助，提出了大量的观点、模型及方法。其中，本书有5个方面的问题是作者独立思考提出来的：

1. 中断驱动的字符设备驱动程序的编程；尤其是它对于高层程序编程风格的重大影响，以及由此引出的多个线索的程序设计；

2. 用汇编语言及C语言编写双态设备驱动程序，所谓双态设备驱动程序是指既能用"device="命令安装，又能在命令行方式直接安装和卸载的EXE文件型设备驱动程序；

3. 讨论了长期不为人所熟知的"重定向接口"规范，并对MS－NET的实现原理进行了探讨；此外还研究了可安装的文件系统及其妙用；

4. 图形模式下弹出的TSR的编程；

5. 利用SDA切换技术改编的可跟踪DOS本身的增强型调试器。

以上几个方面的问题都是异常新颖而独特的，这些问题的提出充分说明，在了解了DOS内部奥秘后，将有多少有意义的工作值得去做！

除此之外，本书还提供大量编程实例，如DOS内部数据和内部接口的监测器等，都是很有参考价值和实用价值的。

本书在整体编排上也是颇具特色的：在本书的绪论中，将DOS的总体结构划分成三个层次，即最低层为IO.SYS层；中间层为MSDOS.SYS内核层，以及与之平行的IFS层；最高层为COM-

MAND 层。本书的第一篇至第四篇即按照这种分层方式，逐层展开进行详细阐述；在第五篇介绍了 DOS 5.0 和 DOS 6.0 的新特性，第六篇是一些技巧较高的编程实例。而每一篇或按流程、或按功能分别讲述，其间穿插了大量的编程思路和有启发意义的方法，供读者思考、实现和运用。这样，本书既有总的线索，同时又覆盖了非常广泛的知识面，包容了相当多的知识点，其中有不少是让读者感觉一新的概念和技巧，相信读者在读完全书后，对此就会有较深的体会。

由于本书是一本在广度和深度方面都达到相当程度的 DOS 类参考书，因此要求读者应具有一定的基础知识，主要是熟悉 DOS 的中断调用，并有一定的汇编语言的编程经验。读者基础越扎实，就越能更好地理解本书所讲述的内容。不过，本书也力图作到深入浅出，以期对大多数读者都有所帮助。

那么，应该怎样阅读本书呢？如果你希望通过参考这本书，达到对 DOS 的更深层了解的话，那么，你就应该循序渐进。因为本书各章节之间常常是相互关联的，很多地方可以说是环环相扣。跳读某些章节虽无不可，但却有可能忽略某些基础性的细节问题，影响你对重要方面的理解。所以至少应从头到尾阅读一遍，然后你可以选择你的重点；或者让本书中众多的知识点能留在记忆里，以备随时回来参考。

总之，作者对本书的要求是：尽可能站在一个更高、更深的角度，从各个方面对 DOS 的实现原理进行总结，并注重探讨在实际编程中的应用。作者衷心希望能通过本书激发起大家进一步深入研究 DOS 及其编程技术的热情，欢迎读者与我们一起讨论有关问题，欢迎对本书提出批评指正或建议，以便使得本书更深入，更完善。

本书的绪论、第一篇、第二篇(8 章，9 章，11 章 11.2 节)、第三篇、第四篇由武汉大学计算机科学系万玉丹编写，第二篇、第五篇、第六篇由黄山编写。我们愿意借此机会与全国的同行们切磋和交流。

最后，作者要感谢学苑出版社和北京希望电脑公司的大力支持，使本书得以面世。在本书成书过程中，武汉大学计算机科学系刘良观教授审阅了书稿，杨令鹏、林宏轩、李志峰同志及崔宝秋、宁强、姚虎、周红松等同学给予了帮助，刘宏程同志提供了部分程序，他们的意见给我们以很多启发。素雯、范增杰等同志为本书录入、排版、制图付出辛勤劳动，这里一并致谢。

作　者
1993 年 8 月于
武汉珞珈山
(430072)

关于 DOS 的 100 个问题的启示

有了全书的总目录，为什么还要有这么一节呢？

目录只是本书的总线索；在本书各章节中，分布着不少有意义的问题，由于服从整体上编排的需要，所以在目录中看不到这些问题；但它们却是有着独特参考价值的重要组成部份，也是本书的特色之所在。因此我们另辟一节，选取了 100 个问题，它们的答案可到各章节中找到。相信这份索引式的问题清单能引起读者的兴趣。

1. 虽然人们已熟知 DOS 的层次式总体结构，却遗漏了一个重要的方面，即重定向接口和 IFS。本书绪论中给出了一份最完整的 DOS 总体结构图，那位长期湮没无闻的 DOS 成员终于开始呈现在眼前。

2. 本书的重点是 DOS，但在第一篇 1.1 节顺便介绍了“扩展 ROM”芯片的格式。扩展 ROM 在设计汉卡、防病毒卡等方面意义重大，它能使得适配器在 ROM－BIOS 自检过程中获得控制权。

3. 以前的 DOS 版本对系统盘的要求很严格：IO. SYS 与 MSDOS. SYS 必须是头两个文件，且顺次连续存放在用户区的开头几簇中。到了 DOS 3. 30“连续存放”的限制已放宽，即可以是象普通文件那样簇号不连续。这一点得益于“IO. SYS 重装入程序”，请参见第一篇 1. 4 节。

4. 常看到 DOS 3. 30＋下硬盘被划分为 C，D，E，……多个逻辑盘，那么 DOS 是怎样掌管硬盘空间，又是如何支持这么多盘符的呢？原来，这一切都建立在 FDISK 划分的“分区表链表”的基础上，请参见第一篇 1. 5. 2 小节。

5. 你想知道为什么 DOS 3. 30 受硬盘分区小于 32 兆字节的限制吗？为什么 DOS 5. 0 能突破这一限制呢？第一篇 1. 5. 2 小节对这个问题作了一个带根本性的分析，原来瓶颈在设备驱动程序的请求头。

6. 想必大家一直都认为 CONFIG. SYS 的“FILES＝”命令限定了可同时打开的文件数，实际上并非如此。准确地说，FILES 规定的是 DOS 内核留出的 SFT 数，这跟可同时打开的文件数之间并不能划等号的，参见第一篇 1. 6. 3 小节。

7. DOS 的 CONFIG. SYS 有一个不为人所熟知的命令“STACKS＝”，在高档微机上使用这一命令也许会使你的程序运行起来更少出错。它的原理见第一篇 1. 6. 3 小节。

8. 不知你想过这样一个问题没有：当 VDISK 报告虚拟盘盘号，它怎么知道刚好是那个盘号没有被软驱和硬盘占用呢？还有，你可以给 VDISK. SYS 附加参数行，指定虚拟盘的盘体大小等等，那么，VDISK 又是如何得知参数行的呢？这与设备驱动程序的 00H 功能有关，请参考第一篇 2. 2 节。

9. 在 DOS 文献中，提到设备驱动程序的请求头结构时，曾使用过“部件号”(Unit Number)这个概念；那么，什么叫一个部件？部件号是否就是指逻辑驱动器的编号？第一篇 2. 2 节将告诉你：两个概念不是一回事，部件号另有其含义。

10. 难以想象，DOS 竟然自己为病毒程序大开方便之门！它支持 INT 2FH/AH＝13H 功能，让应用程序(当然病毒也不会客气)将自身的一段代码秘密地串接到 INT 13H 服务程序之中，而不必修改 INT 13H 入口。第一篇 2. 2. 5 小节提醒用户注意这个关键点。

11. VDISK. SYS 虚拟盘可以安装在扩充内存之中。那么，在这种情况下是如何实现虚拟盘的

呢？更重要的是，当在扩充内存中重复安装 VDISK 时，不会有冲突，这说明它一定有一套分配扩充内存的方法，那么这种方法又是什么呢？请见第一篇 3.3.1 小节。

12. 都知道当只有一个单软盘驱动器时，A、B 盘合用一个驱动器；没想到 DRIVER.SYS 可以人为地产生出这种不同盘符合用一个驱动器的现象，并且还有着其特殊用途呢。请见第一篇 3.3.2 小节。

13. DOS 随系统盘一起提供了 display.sys 及 printer.sys 设备驱动程序，但很多用户却不太明白这两个程序的真正用途。这要看了本书第一篇 3.3.3 小节，第二篇 9.1 节有关内容及 9.3 节后才能完全明白。

14. DOS 初始化过程中，在处理 CONFIG.SYS 时的一个特点，而今成了在启动过程中"动态选择安装"设备驱动程序的契机。详见第一篇 3.4.1 小节。

15. 你想编写出既能在 device 下安装，又能在命令行方式下安装和卸载的设备驱动程序来吗？那么你就应该读一读第一篇 3.4.3 小节，它将带给你一个未曾听说的新概念：双态的设备驱动程序；以及随之一起的独特编程方法。

16. 人们常批评 DOS 没有为异步通信设备提供一个中断驱动方式下的版本，致使 AUX 设备名形同虚设，根本完成不了异步通信。现在本书试图弥补上这一缺陷，请见第一篇 4.2 节。

17. 一个意外的收获：中断驱动的 AUX 设备使得 COMMAND 的 ctty 命令（改变控制台）获得了新的生命力，真的能派上用场了！参见第一篇 4.2.3 小节。

18. 鼠标器虽然简单，然而其设备驱动程序的设计却相当难，有点没想到吧？鼠标器支持的 INT 33H 中断提供了那么多的子功能，初看起来似乎很复杂，其实最关键的却不在这里，而是在硬中断服务程序，请详见第一篇 4.3 节对这一问题的分析。

19. 有没有想过现在的磁盘读/写方式也许可以来一个根本性的改变，即你将不必等待磁盘于完，就可以转到别的工作上去，两方面同时进行呢？这是一个大胆的设想，也并非实现不了，然而真要这样做了，DOS 内核大概就要一片大乱了。请参考第一篇 4.4 节对这个问题的完整分析。

20. 一直以为"修改内存分配"是将一个内存块缩小，以便释放一些空间给其他程序使用。实际上，这一功能也可以将一个内存块动态扩大，只要还有相禽的空闲空间的话。请见第二篇 8.1 节对这一问题的见解。

21. INT 21H/AX＝4B03H 功能可以装入一个"覆盖文件"，那么，什么是一个"覆盖"？在第二篇 8.4.1 小节作了非常透彻的论述，并与 Microsoft C 语言的"覆盖模块"作了比较。

22. PSP＋0AH－0DH 字段对于子进程返回到的地址有决定意义。文献上称该字段保存着 INT 22H 中断向量，这种说法太庞统，也不确切。要想了解这一重要字段的来龙去脉，请见第二篇 8.4.2 小节。

23. PSP＋50H－53H 称为"DOS 功能调度程序"。你想知道它对于多任务程序设计的潜在意义吗？在第二篇 8.4.2 小节，就此提出了一个将 DOS 扩充成多任务的工作模型。

24. 相信大家头脑中有一个根深蒂固的观念，即可有 EXE 和 COM 文件才是可执行文件。令人吃惊的是，在 INT 21H/AX＝4B00H 功能的服务程序中，却从未判过文件名后缀，而只是根据最开头为"MZ"判明是否为 EXE 文件，否则一律视作 COM 文件，想来内核对文件名后缀并无要求……原来，这全是 DOS 外壳（COMMAND）硬性附加上去的要求啊！不信，你可以看看第二篇 8.5 节的 4BH 功能程序清单，就不得不相信这一点。

25. 在 EXE 文件头中有一个字段规定了"程序上方所需的最大内存"，这个字段一向不被重视，未能发挥作用。实际上它是有着潜在意义的，可以解决单个进程独占全部可用内存的问题。第二篇

8.5 节在程序清单的注释中穿插讲述了这个问题。

26. 覆盖文件一般是一次性安装好；但也有可能先暂装到某个位置，最终却是搬移动另一个位置上加以执行。这就要用到 4B03H 功能参数块中的第 2 个字段。遗憾的是，在 DOS 3.30 中有一个"BVG"，使得这种装入覆盖的方式用不通；到了 DOS 5.0 才纠正过来，请见第二篇 8.5 节 4BH 功能程序清单中的注释。

27. 虽然 INT 21H 的 26H 和 55H 功能都可以创建 PSP，但两者之间却有着细微、然而重要的差别，决定了其使用的场合；关于这方面的阐述请见第二篇 8.6 节。

28. 字符设备可以以文本方式和二进制方式之两种方式之一进程工作。那么，文本方式与二进制方式各自到底有什么含义，又有什么区别呢？请见第二篇 9.1 节，定能得到相当详尽的答案。

29. 如果你想弄清楚系统中哪一个驱动器是被 SUBST 过的驱动器，可以调用 INT 21H/AX=4409H。在第二篇 9.1 节，讲述了这一未见诸于文献的功能。

30. 为什么使用 mode 命令可以改变码页，从而改变了屏幕上的字符集点阵呢？第二篇 9.1 节指出只有 EGA/VGA 及 LCD 才从硬件上提供支持字符集点阵的改变。同时，还讨论了与码页相关的 mode chcp 命令，display.sys 等几方面之间的关系。

31. 如果不论是由于单软驱，还是安装了 DRIVER.SYS，导致了多个盘号共用同一驱动器的情况；那么，如何掌握好"当前驱动器"的切换，避免突然出现换盘提示信息就很重要。第二篇 9.1 节在最后讨论了这一问题。

32. 中断驱动的字符设备的出现，强烈地冲击了现有的编程风格，促进程序向多条线索并行的方向发展。在新的编程方式下，原来并不重要的"取输入状态"、"取输出状态"等功能现在成为必不可少的了。第二篇 9.2 节讨论了这种新的编程模式。

33. DOS 外部命令中有一些程序，如 NLSFUNC，SHARE 等，它们本身被 DOS 内核所调用，同时它们也需要调用一些内核功能；怎样解决这种重入式的矛盾呢？DOS 内部服务 INT 21H/AX=12H 功能解决了这一难题，提供了一条直通 DOS 内核的"快班车"。请见第二篇 9.3 节、9.4 节有送论述。

34. DOS 的 SHARE.EXE 外部命令非常独特，它直接将其子程序的入口地址填到 DOS 内核数据区中！这也使得 SHARE 的安装过程是不可逆转的，即不可能再卸载了。第二篇 11.2 节对这一问题作了讲解。

35. 在网络中共享硬盘资源的方案很多；最简单的想法是：让每个站点都能直接访问共享硬盘的逻辑扇区不就得了？这时候，共享硬盘就成了每个站点的虚拟盘。初看起来，这种方法似乎简便易行，实际上它有许多严重的问题。参看第三篇 1.4 节对这一问题的论述，会使你深深地认识到网络中文件系统的重要性。

36. 常听说的网络传输中的数据报和会晤是怎么回事呢？二者有什么区别呢？第三篇 2.2.2 小节对此问题进行了解说。

37. 可曾想到，重定向接口的潜力巨大，尤其是它可以构筑起与 DOS 截然不同的文件系统，同时又能允许 DOS 访问其上的文件。在第三篇 4.2 讨论了一个 phantom 驱动器的例子，证实了重定向接口的存在及其意义。

38. 利用重定向接口可以完成不少意外的工作，比如实现硬盘分区的保密即为其中之一，在第三篇 4.3 节给出了基本思路。

39. COMMAND 外壳使用了/P 开关后即可永久驻留，不被卸掉；那么，COMMAND 成为这种"顶层进程"的密诀何在呢？这就牵涉到 PSP 中的重要字段也与 INT 21H/AX=4CH 等功能的流程

有关。请参考第四篇 1.1 节。

40. 虽然 COMMAND 的命令行方式早已为大家所熟悉，但却未必能列举出全部的合法命令行格式来。第四篇 2.1.1 小节对命令行格式作了归纳，划分为 10 种格式。

41. 看来有些奇特的改向操作其实实现起来非常简单，只是将 JFT 中的 SFT 序号交换一下即可，因此在你的程序中甚至可以如法炮制。见第四篇 2.2.1 小节。

42. 管道操作需要分解成几条改向操作命令，逐步加以完成。第四篇 2.2.2 小节论述了管道操作的执行过程。

43. 可曾听说，有一种介于内部命令和外部命令之间的命令，只需要在第一次从磁盘上装入并执行，以后就再也不必读磁盘，直接从内存中执行。这种特殊命令称为"可安装的命令"，它是以 INT 2FH/AH=0AEH 接口为实现基础的，见第四篇 2.3 节。

44. DOS 3.30 的 dir 命令不能对隐含文件、只读文件、系统文件列目录。其实这只是由于匹配时所用的属性不包括上述文件所致。要改进也非常容易，只要改动一个字节就够了，请见第四篇 2.4.2 小节。

45. 即便是 DOS 的老用户，有时也会发现还有自己未曾用过的命令形如：COMMAND 遵循的是"子目录优先原则"，即任何 ASSIIZ 事来后先判断是否为一条路径名，不成功才视作文件名、根据这种原则，可以有"dir 目录名"、"copy 子目录 1 子目录 2"、"del 子目录"等形式，与带有"*.*"的格式是等价的。怎么样，有点没想到吧？第四章 2.4、2.5、2.6 节将会告诉你一些可能未见过的用法。

46. 当一个批处理命令使用 CALL 命令调用另一个批处理时，还可以返回继续执行；若不用 CALL 则将"一去不复返"。那么，CALL 命令到底起着一种什么样的作用呢？请见第四篇 2.9.3 小节的介绍。

47. 文本文件也可能感染上病毒，并可传播开去。在第四篇 2.9.2 小节介绍了一个精巧的"批处理病毒"，专门感染批处理文件。其病毒体也不过是一些命令行文本串。由这个例子可以看出，批处理文件也可能成为病毒的携带者。

48. 如果你不想使用 COMMAND 的错误显示信息，而由自己来提供错误信息；或者要为新的错误码提供相应的错误信息，你都可以通过接管 INT 2F/AH=05 功能做到这一点。请参考第四篇 3.1 节对该功能的介绍。

49. 调用 INT 2EH 可以借助于 COMMAND 的暂驻部份完成一些内、外部命令，甚至可安装命令也可被执行。但这一中断隐含着一个必要条件。如果你发现调用失败，可能是该条件不满足所致，见第四篇 3.2 节。

50. COMMAND 并非一个完美无缺的产品，有很多地方都可以改进。在第四篇第四章对此进行了讨论，提出了不少思路，相信对大家有启发。

51. DOS 有哪些内部数据结构，它们之间的关系如何，LOL（表之表）在其中将演什么角色

答：DOS 内核所用的内部数据结构及其组织形式如下：

LOL （表之表）	单一数据结构
SFT （文件打开表）	链组
SCBSFT （FCB 所对应的文件打开表）	数组
CDS （当前目录结构）	数组
DPB （磁盘参数表）	单向链
DBUF （盘缓冲区）	单向链

MCB （内存控制块）　　（自 DOS 5.0 起是双向链）

PSP （程序段前缀）　　准链式

LOL 是这些内部数据结构的“参家婆”，在 LOL 中记录有以上各数据结构的链首地址或数组开始地址。

52. DOS 内部数据区的布局如何，如何观察之

DOS 使用了大量的内部数据区，对于 DOS 3.3 而言，DOS 内部数据区是从 DKS:0～DKS:1406H，DKS 是 DOS 内核段址的英文缩写，它可以通过调用 DOS 的 52H 功能获得，52H 功能返回的 ES 寄存器是 DKS 段址。

在本文中附有 DOS 3.3、DOS 5.0(DOS 6.0)的 DOS 内部数据区清单。可供读者参考。

53. 调用 DOS INT21H 的方法有哪几种

答：INT 21H 的调用方式有以下四种：

1） 直接发出 INT 21H

2） 近调用 PSP+5H

3） 远调用 PSP+60H

4） 近调用 0000:00C0H。

54. DOS 内部所使用的内部错误码及其关系如何

答：请见第二篇第二章。

55. DOS 文件系统是分几层实现的？其总体关系如何

答：DOS 文件系统是分三层实现的，其层次关系请见第二篇第三节。

56. 一个句柄可对应多个 SFT 吗

答：多个进程的同一句柄号完全可以对应不同的 SFT，但同一进程的同一句柄只能对应一个 SFT。

57. 一个 SFT 表可对应同一进程的多个句柄吗

答：可以。当用 DOS 的句柄复制功能(45H、46H)时，则再分配一个句柄，但 SFT 并不再分配，只是将 SFT 表中的打开文件计数字段加一。

58. 字符设备也有 SFT 吗

答：有。DOS 可为字符设备分派一个句柄号，当然也分配了一个对应的 SFT 表，DOS 预先打开的 0～4 号句柄就均是字符设备，它们正用于 0～2 号 SFT。

59. 在程序结束时忘记关闭已打开文件，将会出现意料之外的结果吗

答：不会，只要是用 DOS 的 00H、4DH 功能和 INT 20H 退出，DOS 会代劳此事，将所有未关闭的文件关闭。

60. 受保护的 FCB_SFT 是怎么回事

答：由于 FCB 所对应的 SFT 表组空间有限，因此，DOS 每给一个 FCB 文件分配 SFT 时，若 FCB_SFT 已用完，DOS 会野蛮地将最先打开的 FCB_SFT 占用，因此，用户可设立受保护的 FCB_SFT 个数，以保护最先打开的受保护的 FCB 文件对应 FCB_SFT 空间不被占用。

61. 当对许多 FCB 文件同时进行读写时，为什么特别慢

答：这是因为 FCB_SFT 表组少，且易被占用，因此每个 FCB 读写功能都先检查自己的 FCB_SFT 是否也被占用，若占用就再次读取目录项等有关信息再次建立一个 FCB_SFT，这就相当于一个文件打开动作，是相当费时间的。

62. 孤儿 SFT 是怎么回事

答：当SFT打开某文件为它弹出时使用，在调用DOS的驻留退出功能后，由于DOS驻留退出功能不象一般退出功能那样会自动关闭未关闭的文件，因此，该文件一直打开。当将该SFT释放时，若未关闭该文件，则该文件的SFT表就一直保持打开，但此时该SFT没有使用者，故就叫孤儿SFT。

63. “文件打开太多”的错误原因是什么

答：文件打开太多对应错误码是0004，当DOS在SFT不够用或者句柄用完时，返回的错误。

64. SFT链组的个数受什么限制，可以动态增减码

答：CONFIG.SYS语句中的FILES＝M命令，决定SFT链组中SFT个数，当m＜8时，DOS将SFT链组置为8个，应用程序可以动态扩大SFT链组个数，只需实现方法是：

在应用程序空间开辟一块SFT用内存区，找到SFT链组第二组的SFT控制头域，将其指针指向应用程序空间中的SFT缓冲区，注意，SFT缓冲区头两个字节应为FFFFH，标志SFT链完。

以上方法须访问且改变DOS内部数据区，应十分小心，我还没看见哪个应用程序使用此方法，但从DOS SFT链的设计上看，是支持这种方法的，笔者曾测试通过。

65. 句柄数可扩展吗

答：可以，DOS INT 21H的67H功能就是扩展句柄个数、缺省句柄个数为20个。

66. 提交文件是怎么回事

答：提交文件就是将文件所有的改动刷新到盘上，但不关闭文件，不释放SFT。而关闭文件则多做了一件关闭文件的事。由于打开文件操作费时很长，这样提交文件的话就避免了关闭文件以后再次打开的麻烦。

67. 打开文件和移动文件指针操作哪个快

答：DOS在打开文件时，须访问目录区，FAT区十分费时，但移动文件指针只须将SFT中的文件R/W指针改变一下即可，因此打开文件费时大于移动文件指针。

68. 因句柄打开的SFT和FCB_SFT是在一起的吗

答：因句柄打开而建立的SFT是在SFT链组中，而FCB功能建立的的SFT是在FCB_SFT数组中，二者根本不在一块儿。

69. Subst驱动器是怎么回事

答：Subst驱动器是以一个盘号替换一个实际的路径，请见第二篇第四章CDS一节。

70. Join驱动器是怎么回事

答：Join驱动器则是将某个实际盘号作为另一盘号的某一路径使用，详见第二篇第四章CDS一节。

71. 系统实际配置的块设备数怎么计算?

答：系统实际配置的块设备数＝软盘驱动器数(若只有一个软盘驱动器，则此值为2，即单驱盘用)＋硬盘分区数＋虚拟盘的个数。

72. DOS 3.3的32M分区的瓶颈是什么

答：在DOS 3.3中，逻辑扇区号用16位表示，则每个分区最多只能有216个扇区，因此每个分区最多只能有32M字节。

但自DOS 5.0起，DOS用32位表示逻辑扇区号，则突破了32M DOS硬盘分区的限制。

73. “Break＝ON”是怎么回事

答：内部命令“Break＝ON”其实是用DOS的33H功能实现的，33H功能改变了DOS内部一个标志，使得INT 21H进入时，所有的文件功能均需调用检测CTRL_C的过程，从而使得CTRL_C

有效。

而 DOS 的 1～0CH 功能在实现时多处检测 CTRL _ C，即使“Break＝OFF”，也不能屏蔽 DOS 的 1～0CH 功能对 CRTL _ C 的检测。

总之“Break”标志只对大于 0CH 功能的 DOS 调用有意义。

74. DOS 内核调用哪几个中断

答：DOS 内核中若检测到输入 CTRL _ C 则发出 INT 23H，若检测到出现严重错误，则调用 INT 24H，DOS 在 1－0CH 功能实现中多处调用 INT28H。另外，DOS 在进出临界区时，还调用了 INT 2AH，DOS 还调用 IFS 接口，INT 2FH 的 11XXH 功能组。

75. DOS —定不能重入吗

答：自 DOS 5.0 起，可通过 SDA 交换技术重入 INT21H，但必须截获 INT 2AH，以避免重入 DOS 临界区。

76. SDA 是怎么回事

答：DOS 内核中所有关键的内部服务和三个内部栈十分重要，正是它们才使得 INT 21H 不能重入。DOS 提供了 5D0AH 接口，可以获知此片 DOS 内部数据区的大小及位置，这样应用程序可以通过保存及恢复这片区域而重入 INT21H 了。这片区域就叫做 SDA(Swap In-DOS Area)。

77. INT 23H 有几种处理策略

答：有三种策略：

1. 若想用键入 CTRL _ C 而导致当前进程流产，则 INT 23H 应以 STC，RETF 返回。

2. 若想重复一次 INT 21H 调用，则应以 IRET 或 CLC，RETF 返回。

3. 若想使本次 INT21H 接着执行，则应该调查栈针，以 ADD SP，＋6H，RETF 返回。

78. 为什么 INT24H 服务程序中还可使用 DOS 的 1－0CH 功能

答：由于 INT 24H 是由 DOS 发出。因此，INT24H 再使用 INT 21H 的 1～0CH 功能就重入多一次 DOS，但是由于 DOS 的三个内部栈分配是：

一个供 1～0CH 功能使用

一个供大于 0CH 的功能使用

一个供出错时，也就是 INT 24H 使用 1～0CH 功能时使用。

这样，就解决了阻止 DOS 重入的内部栈问题。并且，1～0CH 功能使用的内部数据变量均是一次性的，不会影响到下一次 INT21H 的使用。因此也不存在阻止 DOS 重入的内部变量问题。

1～0CH 功能使用的内部变量均是一次性的，这些变量没有资格列入 DOS SDA 数据区中。

79. INT28H 为什么叫空闲中断

答：当 DOS 等待按键时，会发出 INT28H，此时应用程序可接管 INT28H 而干自己的事情。

80. 文件属性的档案位是怎么回事

答：文件若被修改后，DOS 就将档案位置 1，当用 Backup 和 Xcopy 外部命令进行备份存档时，若使用了/M 参数，则只对已经修改的(即档案位为 1 的)文件备份，并将此位清为 0，因此，就可以只备份修改后的文件了。

81. DOS 如何得知块设备的设备头地址的，为什么不能替换常驻的块设备，DOS 又如何寻址字符设备的设备头，为什么字符设备可被替换掉

答：DOS 内核是通过遍历设备头链，匹配设备名而寻址到某字符设备的 DHD 的。由于是从后向前遍历匹配；因此若可安装的设备驱动程序的设备名也叫 CON，则会先匹配到此 CON 设备，而短路了原 CON 设备，因此字符设备可能完全替换掉。

但是，对于块设备就完全不同了，DOS 内核是从每个块设备部件的 DPB 中得到块设备头的地址，因此常驻块设备是无法替换掉的。

82. DOS 盘缓冲区的设计用意是什么

答：由于 DOS 经常读写一扇区中的某些个别信息，如访问目录扇区中的 FOT，FAT 扇区中的簇链，因此，DOS 先将扇区读到一个缓冲区中，在缓冲区中进行修改后，DOS 也不会立即将缓冲区写盘，直到关闭文件时才将所有 DBuf 的变化写盘。

83. FASTOPEN 是怎么回事

答：由于打开文件费时很多，特别是对一个深层路径的文件打开时，需读多次目录扇区和 FAT 扇区，因此，可以通过外挂的 DOS 扩展器－fastopen 来跟踪某文件，某路径目录扇区的有关信息，而加快了文件打开速度。

84. 为什么要引入文件原语

其引入理由有四点，详见第二篇第五章文件原语一节。

85. DOS 盘缓冲区策略是什么

答：DOS 盘缓冲区的使用策略是 LRU 算法，即近来最少使用 DBuf 被刷新。

另外，DOS 其它有限数据资源的使用策略是：FCB _ SFT 采用先入先出策略，自 DOS 5.0 加入的先行缓冲区也采用 LRU 算法。

86. DOS 为什么经常发出介质检查设备驱动器调用

答：由于软盘驱动器可使用 1.2M 和 36K 两种格式的软盘，因此，DOS 必须判断曾是否已更换，若已换盘，则记录盘中 FAT 文件系统的有关信息的 DPB 应该重新建立，否则 DOS 在用 1.2M 格式的 DPB 信息对 360K 软盘操作时，其 FAT、根目录扇区计算就会出错。因此紧接着在换盘后 DOS 又调用设备驱动器执行重建 DPB 的操作。

87. INT22H 有什么意义

答：INT22H 只对于用 4B01H 加载不执行的功能才有意义，子进程在执行完毕用 4CH 功能返回时，应该返回到 INT22H 的控制之下，这样，进程若接管了 INT22H，就可以在子进程执行完毕后重新获得控制。在第六篇调试器接口一章有 INT22H 更详细的解释。

88. DOS 5.0 的先行缓冲区是怎么回事

答：DOS 5.0 引入先行缓冲区的用意是：由于 DOS 经常读写多个连续扇区，因此当 DOS 发现读写某扇区时，与前一次读写扇区的 LSN 号只差一，则在本次读入时就预先读入后几个扇区到先行缓冲区中。

89. DOS 5.0 的 DOS＝HIGH"是怎么回事

答：由于 80286 及以上机器可以在 A20 地址线有效后，可以寻址 FFFF：0000～FFFF：FFFFH 这附加的 65520 字节，"DOS＝HIGH"就是将 MSDOS.SYS 和 IO.SYS 中的大部分代码和数据区放到 M 以上的这部分之间中而使得常规内存增加。

90. 扩充内存和扩展内存谁的存取速度快

答：对于 80386 系统而言，由于扩充内存只涉及到保护挂式的分段功能，而扩展内存还使用了保护挂式下的分页功能，故 DMS 使用 TEXMS 慢。

91. 为什么要用阴影内存(Shadow RAM)

答：由于 RAM 中比 ROM 中程序运行得快，因此可将 ROM DIOS 或其它 ROM 中的内容拷贝到 RAM 的同样地址区中，并且进行硬件地址寻址的切换，以后访问 FFFF 的内容就访问到 RAM 中而不是 ROM 中了，这样使用的内内存就叫阴影内存。

92. C 语言启动代码(CQ. ASM)怎么用

答:所谓 C 语言启动代码,就是一段配置好 C 环境的代码,形成一个规范的 MAIN 调用:MAIN(Int argc,Char #argv),有无C 语言启动代码使用之剖析清见第六篇第一章。用C 语言编写TSR,设备驱动程序,都涉及对 C 语言代码的修改。

93. ErrorLevel 的用法是什么

答:在 DOS 批处理命令中,有一个 ErrorLevel,其用法是:IF ErrorLevel 50……

上句话含义是如果用 4CH 功能退出程序时的退出码≤50 则 2F 条件为真。

94. 可翻译程序是什么意思

答:由于不同国家的字符集和许多信息串的表区习惯不同,因此必须在编程时考虑到这一点,而很容易地适合于多个国家使用。

95. 怎样才是中断过滤程序

答:中断过滤程序是从 STOIN 读入信息,在进行一些处理后,向 STOOUT 输出的程序。

其用法常见是:(以 MORE 中断过滤程序为例)

1)DIR :MORE

2)DMORE<DIR

96. 命令解释器的功能是什么

答:命令解释器就是象 COMMAND. COM 那样的程序,由它负责接受处理命令行。

用户可以开发自己的命令解释器产品以代替 COMMAND. COM,但要做到与 COMMAND. COM 全兼容,有关内容详见第六篇第四章。

97. 为什么 DOS 5. 0"DOS=HIGH"时要加入检正码

答:由于象 PKLite. Exepack 之类的对执行文件压缩的软件的一个疏忽,没有考虑到当 DOS=HIGH 时,EXE 文件的加载地址在 64K 以下,因此,DOS 为检正该错误而在内核中设计了这段检正码。

98. 命令行历史机制是怎么回事

答:命令行历史机制就是可以方便地得到以前的命令行串,DOS 5. 0 的 DOSKEY. COM 就是支持命令行历史机制的 TSR,其编程接口是 INT 2FH 的 43XXH 功能。

99. 病毒—截获某个中断向量吗

答:不一定,有的病毒甚至于不驻留内存,即使驻留内存的病毒也不一定要截获中断向量。如 DIR—2 病毒通过截获 DOS 调用块设备驱动程序接口而干坏事的。

100. 本书研究 DOS 内核的工具有哪些

答:有四种:剖视设备驱动程序接口的 BIOSSPY。剖视中断调用接口的 Intrspy、监测 DOS 内部数据结构的 DOSSPY 和可跟踪 DOS 代码本身的调试器。

除 Intrspy,其它都是由本人研制的,在本书中有设计思想和程序清单。

绪 论

1. DOS 发展历程的回顾

磁盘操作系统 DOS(Disk Operating System)是美国 Microsoft(微软)公司的著名产品,最初是为 IBM 的个人计算机配置的操作系统。DOS 自 1981 年问世以来,一直是 PC 系列微型计算机及其兼容机上使用最广泛的操作系统。随着微机技术的不断进步,DOS 也不断向前发展,从最初的 DOS 1.0 直到最新的 DOS 5.0,6.0。尤其是众多的 MS－DOS 程序员们围绕着它开发出了数以万计的应用程序,使得 DOS 最终成为微机上最重要的软件支撑环境。这里,我们对 DOS 的发展历程作一番回顾:

1981 年 10 月,DOS 1.00 问世,标志着 DOS 时代的开端,它是由 CP/M 操作系统改进而来,为首次推出的 IBM PC 微机设计的。

1982 年 10 月,DOS 1.00 及 1.10 推出,支持双面软盘。但 DOS 1.10 由于与以后版本有很多根本性差异,功能较差,因而已基本上不再使用。

1983 年 3 月,DOS 2.00 推出,支持硬盘,并大量引进了 UNIX 操作系统的一些特点,如树形目录结构,输入/输出重定向,管道操作等。

1983 年 6 月,DOS 2.10 推出,支持半高型软盘驱动器,及 IBM 的 PCjr 等。

1984 年 8 月,为支持 1.2MB 高密软驱和以 Intel 80286 芯片为 CPU 的 PC/AT 机,推出了 DOS 3.0;

1984 年 11 月,DOS 3.10 推出,以支持 IBM 的网络系统 PC Network。在该版本中,首次出现了"重定向接口"(Redirect Interface),并提供了 Microsoft Network 服务程序来实现网络系统的资源共享。

1986 年 3 月,为支持容量为 720KB 或 1.44MB 的 3.5 英寸软盘驱动器,推出了 DOS 3.20。

1987 年 4 月,推出了 DOS 3.30,该版本改进了硬盘分区表的结构,引进了分区表链表,因而可支持容量超过 128MB 的大容量硬盘,但每个 DOS 分区仍受到 32MB 的容量限制。此外,DOS 3.30 还改正了以前版本中的错误,成为 DOS 最完善的一个版本,迄今仍然被广泛使用着。

此后,Microsoft 于 1988 年 6 月推出了 DOS 4.0。该版本做了不少工作,比如硬盘分区突破了 32MB 的限制这一点就比较好。但总的说来,DOS 4.0 是一个失败的操作系统,这主要是由于它的错误较多,占用的内存也大,且与以前版本兼容得不好。另外,它的 DOS SHELL 也很差,不能算是一个真正的图形用户接口。

而此时,DOS 正一度面临着 OS/2 等操作系统的挑战。然而由于 OS/2 迟迟出台,而且在发展过程中自身存在种种缺陷,终于未能迅速取代 DOS。虽然 DOS 本身的确存在着很多不足之处,如内存受 640KB 限制,缺乏多任务机制,用户界面太简单等,但许多 DOS 扩充程序(如 Quarterdeck 公司的 DESQView 及 Phar Lap 公司的 386/DOS－Extender 等)的出现,使得 MS－DOS 程序员可以开发出运行在 Intel 80286,80386,80486 保护模式下的应用程序,以及实现多任务功能。这就使得 DOS 的生命期得以大大延长。

1990 年 5 月,Microsoft 公司推出了 Windows 3.0,这是微机操作系统发展过程中的一个里程

碑。Windows 3.0 从技术上看有两大特点，一是使得 DOS 管理的内存空间有了迅速扩大，实现了多道应用程序的操作环境；二是从根本上改进了原来评价很差的 DOS SHELL，提供了一个丰富的图形交互式界面。"DOS＋Windows"被誉为新一代微机操作系统。

1991 年 5 月，Microsoft 公司推出了 MS－DOS 5.0。该版本有很多非常好的改进，除了硬盘 DOS 分区突破了 32MB 限制以外，还可以将 DOS 自身的代码及数据尽可能地装入扩充内存，为用户留出更多的基本内存空间；增加了一些很实用的 DOS 外部命令；提供了一个用于文件及目录操作的图形外壳等等。1993 年 3 月，Microsoft 公司还推出了 MS－DOS 6.0，新增加了存贮器优化(MemMaker)及硬盘空间倍增(DoubleSpace)等功能。

Windows 3.0 及 DOS 5.0 都取得了很大成功，加之丰富的应用软件资源，使得 DOS 仍处于重要的基础地位，在未来的微型计算机世界中将继续发挥其重大作用。这种情况就要求我们对于作为应用软件支柱的 DOS 应该有一个深入的了解，不仅要知道如何使用 DOS 提供的功能，还要深入到其内部去，剖析它的数据结构和实现流程。只有这样，才能使得我们对 DOS 的认识更向前迈进一大步，进入一个全新的 DOS 境界。这对于我们更好地利用这一支撑环境为开发各种各样的微机应用软件服务，以及做好系统维护，应付计算机病毒的侵袭等工作都是极为重要的。

本书将主要围绕有代表性的 MS－DOS 3.30 来进行讨论，因为 DOS 3.30 不仅代码简洁、高效，而且与高版本的 DOS 5.0 及 6.0 相比，最基本的功能都具备，各个分支下的实现流程也没有大的差别。另外 DOS3.30 的稳定性更好(已有不少关于应用软件与 DOS 5.0 冲突的消息)，因此足以代表 DOS 系列的核心技术，而且也最适合作为探询 DOS 内部奥秘的一个版本。此外，为兼顾高版本，在本书最后一篇还对 DOS 5.0 和 DOS 6.0 作了大量对比分析，着重阐述了它们的新功能、新特性，及其实现方法。

2. MS－DOS 的层次式总体结构

完整的 DOS 实际上包括四个部分，它们是：

(1)引导扇区 BOOT： 通常位于软盘或硬盘分区的逻辑 0 扇区，用于检测 DOS 系统文件 IO.SYS 和 MSDOS.SYS 是否存在，并装入 IO.SYS 开始启动 DOS。

(2)IO.SYS： 是 DOS 中贴近硬件的部份，负责完成 MS－DOS 的初始化过程的主要工作。提供常驻的设备驱动程序，使得高层程序能以标准接口访问硬件。

(3)MSDOS.SYS： DOS 内核部份。提供字符设备 I/O、文件系统、内存管理、进程调度、网络共享等一系列功能，为应用程序服务。

(4)COMMAND.COM：DOS 外壳，它给用户提供了一个行命令式的界面，接收键入的命令行并解释执行。

不同的微型计算机，硬件状况及设备配置各不相同。而 DOS 作为系统软件，能适应各种不同的情况，给高层的应用程序提供了一个统一的"工作平台"。DOS 的层次式结构确保了它能完善地实现上述要求，见图 1。图中实线框内的虚线表示各部分之间融合在一起，并不能截然分开。双线箭头表示高层对低层的层次间调用，而单线箭头则仅表示一般的调用关系。另外，IFS 与 MS－NET 使用的都是重定向接口，但 IFS 却有着更深的含义，以后将进行论述。

在图 1 中我们看到 DOS 呈现出鲜明的层次式结构，即整个系统从低到高依次由硬件层、BIOS 层、DOS 内核层和 DOS 外壳层构成。本书以后的各篇也大体上按这种结构逐层加以讨论。

DOS 的层次结构使得每一高层都以标准接口与其低层通讯，比如 COMMAND.COM 及应用软件按 INT 21H 的文档规范调用 DOS 内核功能；而内核则按设备驱动程序请求头 RH(Request Header)的标准格式调用 BIOS 层。层次结构使得高层不必关心低层的实现细节，只要接口不变，高

层就可以不变。这就使得低层可以独立地工作，并将高层与具体硬件隔离开来，使得DOS具有良好的适应性。比如微型计算机OEM们可以根据它们生产的兼容机的具体情况来修改IO.SYS，从而很容易地使得DOS及建筑在其上的庞大应用软件资源在这些微机上也可运行。

DOS的整体设计思想是成功的，值得我们在设计系统软件及应用软件时加以参考借鉴。当然，这种层次也并非是绝对的，很多应用程序为了加快I/O速度等目的，绕开了DOS直接调用ROM－BIOS的功能。但从可移植性的角度来看，我们还是要提倡尽量使用DOS功能。

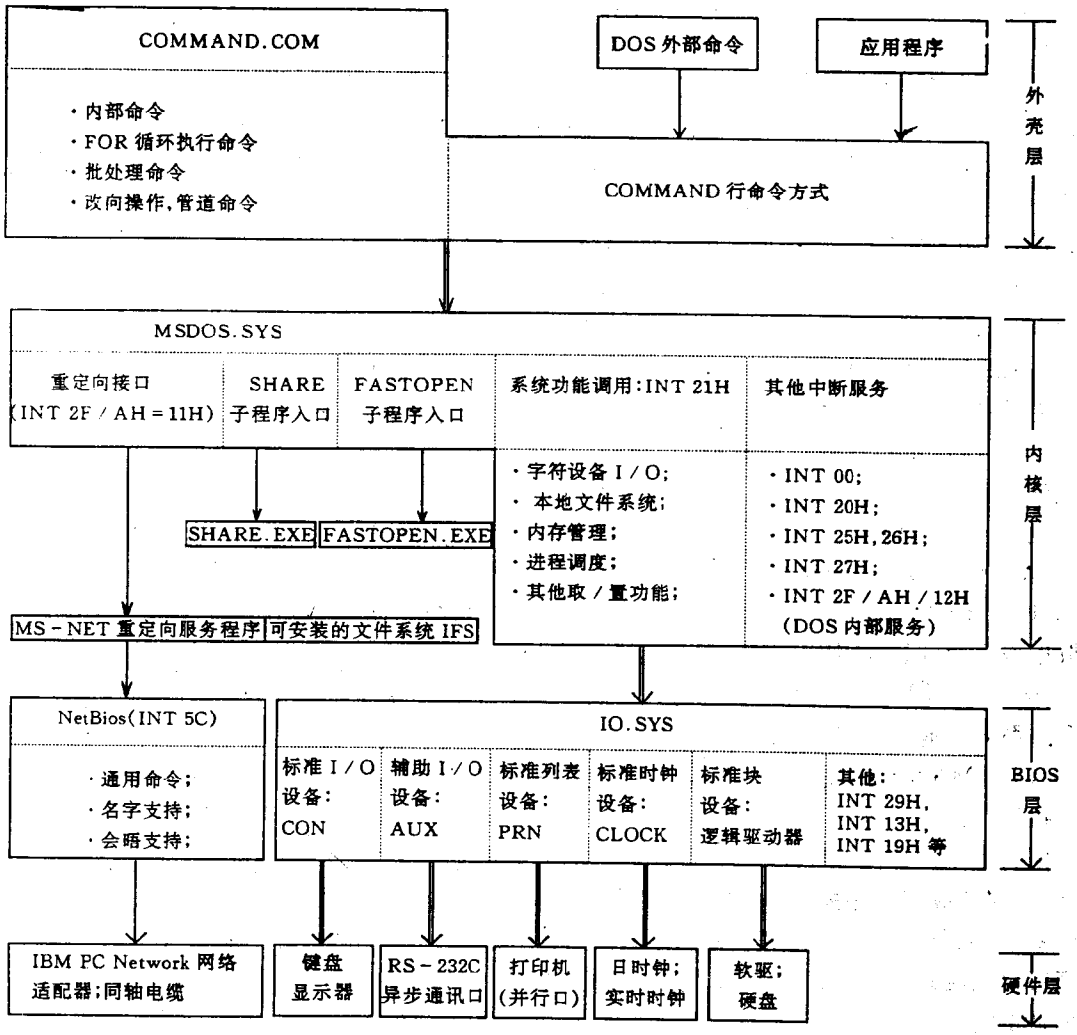


图1　局域网DOS的层次式结构