MS-DOS 4.0

磁盘操作系统

四川省计算机公司

前　言

本资料是美国 Microsoft 公司于1986年12月推出的。为了及时吸收国外软件开发的新成果，我们组织编译出版了这本资料，供从事计算机系统软件开发，计算机软件教学以及推广应用等各界人士参考使用。

MS—DOS4·0磁盘操作系统，除保留并部分增强以前版本的功能外，还新增了前后台管理功能，多任务(多进程)运行管理功能，网络支持功能，以及一些新的DOS命令。该操作系统功能比较强，适合于GW286、0530等中高档微机系统运行使用。

在这本资料中，除了从使用的角度系统地介绍了 MS—DOS 4·0的各种操作命令外，还深入一步地介绍了有关它的编程环境，系统调用、设备驱动程序、外部接口等涉及系统内部结构的某些问题。这对深入了解 MS—DOS操作系统本身、以及应用软件的开发都是有所帮助的。

本书第一、二、三、四、五、六章由葛玫同志翻译，第七、八、十、十一章由刘晓洁同志翻译，第九章及附录由廖华楷同志翻译，第十二、十三、十四章由彭忠义、杜国棉、刘开民等同志翻译，全书由黄安南、彭忠义二同志负责校核与审译。

由于时间仓促，水平有限，错误之处在所难免。敬请各界批评指正，我们将不胜感激！

四川省计算机公司 MS—DOS4·0 编译组

一九八七年十二月

目　录

第一章　MS—DOS4.0编程环境

1·1 引　　言

MS—DOS4.0是适用于8086系列计算机的多任务磁盘操作系统。它能运行一个前台应用程序和多个后台应用程序。

随着个人计算机和办公室自动化软件系统的日益复杂，多任务处理功能成为一种必须条件。多任务系统允许后台任务的建立(如假脱机打印)，从而改善了系统的性能。而且，将运行网络文件盘和进行系统进程的邮件传递置为后台功能，计算机便可以进入网络工作环境。

1·2　必要的条件

在MS—DOS3·2上设计的程序在MS—DOS4·0上也能运行。此外，MS—DOS4.0的较强的应用程序接口（API）功能使新的用户能够享用它的多任务特性，因此，对最终用户来说，MS—DOS4.0提供了新的有效的功能。

在MS—DOS4.0系统下应用程序的执行在前台区和后台区都可以完成。在前台区运行的程序就象 MS—DOS3.2 中的各种应用程序一样，可以直接与计算机和用户进行相互作用，但是后台区的使用却局限于那些按MS—DOS4·0后台环境而特别开发的应用程序。

MS—DOS4·0 与MS—DOS3·2 所适用的计算机的机型是完全相同的，不过为最大限度地使用MS—DOS4·0的功能，这些计算机应当提供中断——驱动I／O方式。

1·3　8086结构上的限制

MS—DOS4·0力图在8086系列微机上运行MS—DOS3·2 的各种应用程序。8086 cpu 和286实地址方式都没有保护和硬件再定位功能，这两种功能在多任务环境中十分有用。除了处理芯片以外，8086系列的微机没有对处理机芯片增加外部硬件存储保护，所以它们不能防备故障，也不能防止被损坏程序对其它程序或DOS本身的破坏。

而286在与8086二进制不兼容的方式下可提供保护和硬件再定位功能。因为，不可能将这些286的新的特性传送到老的应用程序，故在实地址方式下运行程序时，MS—DOS4·0 在8086及286微机上均没有保护措施。（也不能防止或恢复以实地址方式运行的程序所破坏的任务。）

1·4　定　　义

本文将用到下列定义：

Task　——“任务”，一次独立的程序执行。多个任务可以同时存在且通过任务数据

结构有各自的DOS考贝。

Process——“进程”，与“任务”可互换使用。

Concurrency ——“并发性”，同时运行多个应用程序的能力。

Device Driver——“设备驱动程序”，专为硬件编制的例行程序。通过它将DOS和被DOS系统调用的硬件相连。每个设备驱动程序都有一个策略入口点以便开始操作，同时还有一个中断入口点，在操作结束时被调用。

Interrupt ——“中断”，需要立即服务的外部事件，使8086暂时从它的正常执行中中断。在某些情况下，中断服务程序将不受时间片的约束，但典型的中断服务程序都是经过精心选择的，它们能使8086尽可能快地回到正常执行中去。

Detached Task——“被分离的任务”，即那些在后台环境运行的“任务”。

Blocked Task ——“被锁定的任务”，只有当DOS请求或某一设备条件满足后才能继续执行的任务。

Runnable Task——“可运行任务”，与“被锁定任务”相对应，不需经DOS请求或某一设备条件满足，调度程序即可选择”可运行任务”进行运行。

Task Switch ——“任务转换”，DOS系统挂起某一任务而恢复另一个任务的执行。为了进行任务转换，DOS系统必须对内部数据结构进行转换，每秒钟内这种转换可进行多次。

Exec ——“执行”，一个任务（母任务）调用另一个任务（子任务）的能力。所有由命令解释程序所调用的应用程序都是它的子进程。母任务既可以被暂停执行，直到子任务执行完毕为止，也可以继续执行而不受子任务的影响（至少在开始时是如此）。

Shell ——“外壳程序”，控制一系列程序运行的一个命令解释程序。

Foreground Partition ——“前台区”，前台运行那些直接与计算机和用户相互作用的应用程序。任何时候前台都只能有一个程序运行。

Background Partition ——“后台区”，后台区运行那些专在MS—DOS4·0上开发的程序。这些程序利用MS—DOS所提供的服务功能与计算机和用户进行相互作用。在后台内几个不同程序能够同时运行。

1·5 DOS接口

MS—DOS功能是通过INT 21H进入的。DOS调用控制着计算机的每一部分。一个程序唯有通过DOS调用才能和用户、其它程序或设备进行相互作用。

对程序设计者来说，DOS是可重新进入的。因为设计后台任务，设计者可不考虑其它任务所带来的限制。然而，对一个给定的任务来说，DOS系统一直是与之同步的：如果一个任务还在DOS系统中获得技术上的控制（如直接接受一次硬件中断），那么这个任务便不需再

呼叫DOS。同样，这种限制使设备驱动程序仅在初始化时间或借助辅助功能时才呼叫DOS。

1·6 设备驱动程序

1·6·1 多任务设备驱动程序

虽然大多数现有的MS—DOS3·2设备驱动程序均可在MS—DOS4·0上运行，但是它们影响了系统的性能。为此所有的设备驱动程序最好都按MS—DOS4·0所提供的特性来重写，这些特性包括：

- 锁定任务：当等待一个I／O操作完成时，MS—DOS4·0 可从运行队列中移去一个任务。
- 重叠I／O：当这种磁盘操作发生时，其它可运行的任务也被启动以使cpu处于忙碌状态。由于多个任务在文件系统中能够同时活化，这些任务甚至能够产生磁盘请求。
- 排队请求：由于在文件系统中多个任务可以进入活化状态，设备驱动程序可将多个请求进行排队。设备驱动程序能随意地按最有效的方法对请求进行重新排列。

1·6·2 控制台驱动程序

控制台设备驱动程序分为两个部分：屏幕显示和键盘。由于这些驱动程序日益复杂，较为理想的方法还是分别替换它们。

1·7. 健盘中止

键盘中止通过键盘操作（如CTRL—C）来实现。在通常的操作中，当前命令解释程序（称为“Shell”，一般为Command．com程序）启动 SIGINTR信号。如果用户键入CTRL—C，外壳程序“shell”便接收SIGINTR信号并发出（通过系统调用）一个SIGTERM 信号给当前运行的进程。在该进程已产生多个子进程的情况下，它便将SIGTERM信号发给当前运行进程的命令子树，直到所有的子进程结束后，命令外壳程序才会提示用户。

过程可能从下述三种方式来影响键盘中止的实现：

- 首先，进程可能不希望随意被删除，也许因为它们需要清扫一些锁定的文件或者需要恢复某个数据库。在这种情况下，当信号被接收后，程序中止SIGTERM信号，服务码被清扫然后再退出进程。
- 其次，程序可通过将键盘设备转入原始方式来影响SIGINTR（即 CTRL—C ）的处理。这是由屏幕编辑等程序完成的，它们将CTRL—C作为一个数据字符来看待，在退出之前，程序会将键盘恢复到非原始状态
- 第三，运行程序本身也可以是一个外壳程序。例如，编辑程序、 Multiplan 程序等等。这类程序确实可用 CTRL—C 进行报警，但却不会因此而被删除，它们会发出

Set—Signal—Handler系统调用来中止SIGTNTR信号。这就自动地挂起了母外壳程序对SIGINTR信号的响应。当这个新的外壳程序退回母外壳程序时，母外壳程序便又自动地恢复了对这些信号的响应能力。

1·8 内部结构

每一个任务都包括了一个Per Task Data Area (PTDA) 和一个程序段，这种设计结构被称为多逻辑内核触发。这种技术（同样也用于XENIX）大大地简化了内核结构，且有助于与以前的版本保持兼容。为了保证其简单、完整，整个MS—DOS的内核都被认为是不能中断的临界代码区。如果每次只可能有一个任务在内核中活化，这本身就造成了一个非最优的环境。将临界区退入到设备驱动程序中（这就要求在调用设备驱动程序之前将所有DOS数据结构适当地修改），如果一个任务留在设备驱动程序外面，那么多个任务即可同时在内核中活化。由于从进入内核到检索设备驱动程序的时间很短，所以为实现程序设计的简单明了，这点代价还是值得的。

如上所述，任务程序段与其它MS—DOS3·2下的任何程序具有同样的格式（256个字节标题，加上实际程序），而PTDA是为DOS而设的数据区，它包含所有任务专用信息。该区域只受DOS控制，任务本身却无法使用它，每一个任务都有一个单独的PTDA。

唯有前台应用程序才能使用8087协处理器。

1·9 硬件条件

能支持MS—DOS的最小系统是带有定时器的8086／8088计算机（即处理器、存储器、视频I／O等），事实上，这个最小系统中还应包括中断—驱动式I／O和对磁盘的DMA（即直接存储器存取）功能。

MS—DOS4·0的设计是为了充分支持并行I／O。并行I／O就是说系统能同时响应n个不同的I／O请求。虽然每次只能处理一个主要的I／O请求，但每个任务都有其各自的待处理请求。并行I／O对磁盘设备来说是十分有利的。磁盘设备的待处理请求将按其在磁盘上的位置进行排序，这样就减少了查找的时间，同时增加了总的I／O处理能力。为了最大限度地利用这一功能，硬件系统应保证每台设备都能使用并行I／O。这包括给每台设备配备合理的中断线路，对每个设备单独的DMA通道和并行磁盘查找。所有的硬件中断都是由相应的设备驱动程序来处理的，当I／O操作完成后，也是由这个设备驱动程序来通知DOS系统。

1·10 上托包 (Popup Package)

1·10·1 概 述

在MS—DOS4·0下的后台应用运行环境对程序有几条限制。其中之一就是后台应用只能访问屏幕和键盘，而且只有当后台应用已获得对屏幕和键盘的控制时，后台应用才能进行

"写屏幕"或"读键盘"的操作。上托包可用于使后台应用程序得到对屏幕和键盘的控制。

上托屏幕使后台应用程序能和用户开始交流，用户也可能希望同后台应用程序建立联系。这可以通过使后台应用程序挂上INT 9H来实现。（只有当后台应用程序确实不是即将终止时，它才能挂上INT 9H。）

通过挂上INT 9H，后台应用程序将等待一个用户键入的"热键"。键入此键时，应用程序的INT 9H处理程序将用某种进程间通讯的形式或一个标志字将所发生的一切通知它。通常，它的调度事例便是打开一个上托屏幕。在中断时间里，是不会打开任何上托屏幕的。

上托包由终止并保持常驻程序构成。终止和保持常驻程序被认为是在前台区运行的，因而可以对屏幕、键盘及所有中断向量进行访问。在上托包中建立终止并保持常驻程序的另一原因是因为它没有PTDA，这就节约了2K的存储空间。

当上托包被启动时，使用屏幕的任务被"冻结"，直到后台进程释放屏幕为止。上托包通过对任务发出Freeze系统调用来达到这个目的。如果当Freeze命令发出时，使用屏幕的程序在内核（程序）中被封锁，Freeze命令也不会返回，它会等到Freeze命令实施以后再返回。只有当程序的封锁被解除后，Freeze才会被实施。

在使用上托屏幕时，只有使用屏幕的任务被"冻结"，所有其它的任务（其中包括调用POPUP的那个）都继续被调度。（注意：后台应用程序可以优先于前台应用程序，但却不能优先于其它后台应用程序。）

为了使用上托包(POPUP)，上托应用程序将调用CheckPU，PostPU，SavePU和RestorePU等操作。CheckPU检查POPUP包是否已被装入。PostPU冻结和恢复使用屏幕的任务并打开或关闭上托屏幕。SavePU将当前屏幕及键盘的状态和内容存入上托包的缺省存储区或存入由任务指定的存储区。RestorePU则依据缺省存储区或任务指定存储区中的信息恢复屏幕及键盘的状态和内容。在PostPU要求的打开或关闭上托屏幕之间，SavePU和RestorePU可以被重复地调用。参看下节"使用注意事项"中有关内容。

如果使用屏幕的任务是以图形方式占用屏幕，POPUP则进行清屏。在恢复屏幕图象时，POPUP则进行"写入"屏幕操作，而其它的"写入"屏幕操作都必须由上托应用程序来进行。这样上托应用程序被允许了最大的灵活性，它可以决定其自身如何使用屏幕，它可以自己"清屏"，在屏幕上画一个框进行写入、修改屏幕已有的内容、改变屏幕颜色或是其它的什么操作。

POPUP必须在运行任何设置INT 8H、INT 9H、INT 10H、INT 16H、INT 1CH和INT 21H向量的前台应用之前运行。只有这样POPUP才能记录到INT 8H、INT 9H、INT 10H、INT 16H、INT 21H等向量的缺省设置内容。当启动POPUP时，被保存的向量将和当前向量进行互换以确认POPUP和使用上托屏幕的应用程序系统能顺利地进行访问。当上托屏幕关闭以后，向量将被重新设置为POPUP启动时的值。参看下一节"使用注意事项"中有关内容。

Microsoft开发的POPUP包将只支持CGA、EGA和MPDA，增加对任何其它图形适配器的支持留给OEM来完成。不管用的是什么型号的图形适配器，Microsoft的POPUP包都只保存当屏幕处于文本方式时所用到的那部分屏幕信息。这就意味着使用POPUP的后台

应用程序只能在文本方式下使用屏幕，这样作就使需用于保存一个屏幕的RAM决不大于13K字节。不使用POPUP的前台应用程序则可以以任何方式使用屏幕。

Microsoft的POPUP包也包含了另一个由Microsoft公司开发的称为EGAINT 10那部分。EGAINT 10包括一个INT 10H向量处理程序，当使用EGAINT 10上托包的附加功能时，它可以映射EGA中的只写寄存器。如果使用屏幕的任务不是以CGA兼容的方式来使用EGA，那么任务就必须用EGAINT 10的功能来使POPUP能够正确地保存和恢复屏幕。Microsoft的大多数从非CGA兼容方式来使用EGA的应用程序都借助于EGAINT 10。

1·10·2 使用注意事项

1）某些情况下，上托应用并不需要用户对其在上托屏幕上所显示的信息作出反应。这种时候，应用程序会在适当时间间隔之后关闭屏幕，以确保中断了的应用程序被排挤的时间尽量地短。

2）当保存一个上托屏幕时，POPUP将在返回上托应用程序前将屏幕置于文本方式之下。

3）因为在将来的MS—DOS版本中，调用CheckPU、PostPU、SavePU和RestorePU的实际操作方法有可能改变，所以所有POPUP包的功能都将通过一个调用接口来访问。用了调用接口后，当上托应用程序改为新的MS—DOS版本后，只需改变调用接口就行了。参看“MS—DOS4·0系统调用”一章，其中给出了高级语言中调用接口的一个例子。

4）当PostPU中的waitflag = 2时，可以有一个以上的进程同时使用POPUP，这时POPUP将冻结当前上托应用程序以便新的上托程序能够运行。如果POPUP必须冻结一个上托程序以使第二个上托程序运行，那么第二个上托应用程序必须为POPUP提供存储区来保存当前屏幕及键盘信息。如果第二个程序做不到这点，SavePU及RestorePU一经检查出即会发出出错信息。如果没有其它的应用程序被中断，新的上托程序就会占用POPUP的缺省存储区。

5）POPUP必须清除INT 9H向量以确保其它的上托程序不会干扰使用上托屏幕的应用程序。由于新的上托应用程序被认为是正常运行的，所以没有必要非把它们置于INT 9H键之外，它们可以仍然等待其“热键”。当后台应用希望等待其“热键”——即使上托屏幕是打开的，它也必须在POPUP包装入之前启动。

6）POPUP仅仅保存了屏幕状态信息的最小值，因此，上托应用程序在进行屏幕输出时只使用单一的文本方式显示页。

7）前台应用也可以使用POPUP包，这样一来，应用程序就可以明确地编写为在前台区或者在后台区运行。

第二章　MS—DOS 4·0新的系统调用

2·1 进程控制

新的系统调用

在MS—DOS4·0中，有三个新的或者说增强了的系统调用，它们可用于启动和控制任务或进程的执行。

Exec	启动新的任务。可以是同步的任务也可以是异步的任务。Exec也可以为前台进程装入复盖程序。
AExec	启动新的异步任务。这个调用不能用窗口程序将其中止。
Wait/Cwait	母进程（发送端）等待子进程结束，同时检索从子进程中返回的代码。
Kill	终止一个进程。

新的特点

MS—DOS4·0的一个主要功能是多任务处理。MS—DOS4·0最多能处理32个任务。一个被操作系统处理，其余31个为可用任务。

在MS—DOS3·2中，程序建立一个子任务，但在母任务能重新开始运行之前子任务必须结束运行。而在MS—DOS4·0中，母任务既能够等待子任务完成，又能够继续它本身的运行。子任务能享有其母任务的环境——包括打开的文件描述和共享存储区指针。这种功能与管道功能一起，使程序能够调用其它程序来完成其一部分工作。这种设计方式被称为Software Tools结构。比如说，一个对文件进行排序的程序并不需要包括一个排序软件包，它可以将系统中的排序实用程序作为一个子进程来调用。

这种结构中一个值得注意的关键问题就是影响进程的很多操作同样也会影响由指定进程产生的进程分支。例如，如果一个当前运行程序X调用了排序程序作为一个子任务，并假定X处于“冻结”状态，叫排序子任务也将处于“冻结”状态。虽然，建立X的任务或外壳程序对排序程序一无所知。事实上，外壳程序或是X都不知道排序或许又调用了另一个子进程。这条规律的一个例外就是前台进程的终止。由于在任何时刻仅有一个前台进程执行，所以只有正在运行的前台进程被终止，它的原始进程才不会受影响。

每一个进程及其它的所有分支被称为一个命令子树。

2·2 并行支持

“并行”意为操作系统可一次进行几个任务中的任何一个，随意在任务间转换。人就是并行处理器的一个明显的例子。一个人一次只能干一件事情，但他可以在几个进行的工作中任意挑选。

新的系统调用

Freeze	停止一个程序的执行，一直到它明显地重新启动为止。
Resume	恢复执行一个“冻结”的程序。

新的特点

当前台程序正在运行时，为了使后台程序能够在屏幕上显示信息并接收用户的输入，MS—DOS4·0为后台程序提供了一种功能，即在允许前台程序继续运行之前，“冻结”当前前台程序，以便后台能够使用屏幕和键盘。这一功能是借助上托包来实现的。

为了避免与前台相互干扰，后台进程在运行中必须有下述限制：

- 除非后台进程事先保存了屏幕内容并用POPUP的功能打开了一个上托屏幕，否则它们便不能在控制台设备上进行读、写。
- 除了INT 9H和2FH以外，后台进程不能设置任何中断向量。如果进程确需要查找某个关键字，可以设置INT 9H向量。
- 后台进程不能使用8087协处理器和仿真浮点，只能用调用接口。
- 后台进程可以在紧急时分配存储区，但不提倡这样做。更好的办法是在开始时即分配所有所需的存储区，以避免出现空间存储碎片（即存储区中出现未用满部分），同时也避免和前台任务发生冲突。许多程序进行存储区分配采用二个步骤。首先它们发生一个Alloc系统调用查找出自由存储区内最大的存储块，然后，再发出一个Alloc系统调用分来配这个存储块。如果在这两个步骤之间出现一个上下文转换开关（也叫关联开关），且如果后台任务分配了存储区，那么，前台任务的第二个 Alloc系统调用可能失效。
- 后台进程必须应用CritErr系统调用使所有INT 24H 硬件错误失效，按照约定，DOS自动地将后台进程产生的硬件错误失效。
- 欲在MS—DOS4·0后台环境下并行运行的程序必须用LINK 4 进行连接。LINK 4 以一种新的可执行文件格式来产生程序，这种格式以前的MS—DOS版本不能支持。
- 后台应用程序必须由Exec子功能 4 系统命令或AExec命令来调用。AExec 调用避免了用窗口程序来对Exec进行仿真。
- 后台进程只能发出异步的Exec调用，为模拟同步Exec命令，后台进程必须先执行一个异步Exec命令，再调用一个CWait。
- 后台进程比前台进程有更高的优先级。前台进程有可能被后台进程排挤。但与cpu 相关的那些后台进程不可能排挤前台进程，它们会暂时让位于cpu，以保证其它进程继续被调度。
- INT 11H、12H、21H，INT2AH和INT 2FH可由后台进程在任何时间发出。INT 10H和INT 16H却只能在进程已打开一个上托屏幕后才能发出。除此之外，后台进程不能发出任何其它的中断请求。
- 后台进程不能使用逻辑驱动器 B，因为这将使BIOS为进程产生控制台I/O。
- 后台进程不能复盖其代码段。