第 5 章　设备管理

本章是操作系统中最为繁琐的一部分，在这一章中介绍设备管理的任务和功能，数据传送的控制方式，缓冲技术，设备分配技术及 I/O 进程控制。

- 数据传送的 4 种方式：程序直接控制方式、中断控制方式、DMA 方式及通道控制方式
- 高速缓存和缓冲技术的概念及 4 种常见的缓冲技术：单缓冲、双缓冲、环形缓冲及缓冲池
- 设备分配的概念及 3 种设备分配技术：独享分配、共享分配及虚拟分配
- I/O 进程控制

5.1　设备管理概述

在计算机系统中，设备管理是指对数据传输控制和对除中央处理机、主存储器之外的所有其他设备的管理。

除中央处理机、主存储器之外的所有其他设备称为外部设备。

5.1.1　设备类型

外部设备的种类繁多，用途各异，现存的各种仪器设备均有可能作为计算机系统的外部设备。依据不同的方式可对设备有不同的分类方法，下面是几种常见的分类方法。

1. 按操作特性分类

按这种方法可把外部设备分为存储设备和输入/输出（I/O）设备。存储设备是计算机用来存储信息的设备，如磁盘、光盘、磁带等；I/O 设备包括输入设备和输出设备两类。输入设备的作用是将外部带来的信息输入计算机，如键盘、鼠标等。输出设备的作用是将计算机加工好的信息输出到外部，如显示器，打印机等。

2. 按传输的信息特点分类

按这种方法可将外部设备分为字符设备和块设备。字符设备处理的信息是以字符为单位组织的，如打印机、键盘等。块设备处理的信息是以块为单位组织的，如磁带、磁盘等。

3. 按系统和用户的观点分类

按这种观点可将外部设备分为系统设备和用户设备两种。系统设备是指操作系统生成时已登记在系统中的标准设备，如键盘、磁盘等。用户设备是指操作系统生成后，用户定义的

非标准设备，如绘图仪，游戏手柄等，这时需要用户安装设备驱动程序。

4. 按使用角度的观点分类

按这种观点可将外部设备分为独占设备和可共享设备。独占设备是指在作业执行期间只允许一个作业独占使用的设备，如打印机、输入机等。可共享设备是指当多个作业同时执行时，这些作业可同时使用的设备，值得注意的是，“同时使用”的含义是指一个作业尚未撤离，另一个作业即可使用，但同一时刻只有一个作业能够启动设备，允许它们交替地启动设备，比如磁盘。

5.1.2 设备管理的任务和功能

1. 设备管理的任务

在操作系统中设备管理的任务有以下四个。

（1）提高设备的利用率。外部设备经济价值在整个计算机系统中占有相当大的比重，如何有效地使用这些设备是操作系统管理中的首要任务。尽管现代计算机外部设备的工作速度有一定程度提高，但与中央处理机的速度相比仍太慢。为提高设备的利用率，除合理分配和使用外部设备外，应努力提高设备与 CPU 的并行程度，与此相关的技术有通道技术、中断技术和缓冲技术。

（2）设备独立性。设备独立性是指将用户使用的设备与机器中进行 I/O 操作的物理设备分离开来。用户不用关心具体的物理设备，只需按习惯为所需的设备起一个逻辑名字，在用户程序中仅使用逻辑设备名即可。

设备独立性有以下两种类型：

1）独立于同类设备的具体设备号。如系统中有相同类型的多个设备，则不论使用其中的哪个设备都行。即与给定设备类型中的哪一台设备供其使用无关。

2）独立于设备类型。如程序要求输入信息，则不论从什么设备上输入均可，对输出也一样。即用户程序与设备类型无关。

（3）字符编码的独立性。各外部设备的字符编码方式会有所不同，为减轻用户编程时的负担应统一使用内部字符码。这就要求设备管理中应有适应于各设备的字符编码的变换机构。

（4）设备处理的一致性。种类繁多的外设特性各不相同，其差别主要有以下几点：

1）速度。在不同的设备之间传输数据，传输速率可能有几个数量级的差别。如键盘输入和光盘输入速度相差甚远。

2）传送单位。有的设备以字符为单位传递信息，有的设备以块为单位传递信息。

3）允许的操作。不同的设备有着不同的特性和操作方法。如磁带能反绕，卡片机不能倒退，磁盘能随机读写等。

4）出错条件。根据所用的设备的不同，出错条件也不同。如奇偶校验错误，打印机无纸错误等。

为了简便和避免出错，应用统一的方法来处理所有的设备。为了做到这一点，应将设备的特性与处理它们的程序分开，使之只与设备本身紧密联系。这样，可使某一类设备共用一个设备处理程序。

2. 设备管理的功能

为了完成上述任务，设备管理应具有以下功能：

（1）监视系统中所有设备的状态。系统中存在许许多多设备，这些设备在系统运行期间处于各自不同的状态，为了对设备实施分配和控制，系统必须要在任何时间内都能快速地跟踪设备状态。设备状态信息保留在设备控制块（DCB）中，DCB 能动态地记录设备状态的变化及有关信息。

（2）设备分配。在多用户多进程中，系统必须决定进程何时取得一台设备，使用多长时间，使用完毕后如何收回等问题，具体的设备分配方法将在 5.4 节介绍。

（3）设备控制是设备管理的另一功能，它包括设备驱动和设备中断处理，具体的工作过程是在设备处理的程序中发出驱动某设备工作的 I/O 指令后，再执行相应的中断处理。

5.1.3　设备控制块（DCB）

1. DCB 结构

DCB 是设备管理的重要数据结构（见图 5.1），通常每台设备的一组数据集中存放于主存中的某一区域内组成的数据块称为设备控制块。当设备装入系统时，DCB 被创建。DCB 主要包括以下内容：

（1）设备名。它是设备的系统名，即设备的物理名。

（2）设备属性。它描述设备的特性和类型。

（3）设备状态。它描述设备现行的状态。

（4）设备在 I/O 总线上的地址。它反映现行设备在 I/O 总线上的具体地址。

（5）等待队列指针。它存放要求使用该设备的队列的首指针。

设备名
设备属性
设备状态
设备在 I/O 总线上的地址
等待队列指针

图 5.1　DCB 的结构

2. 设备转换表

对设备的请求最终要转换成对设备的 I/O 操作，系统可以通过操作码检索“设备转换表”找到相应的设备地址。该转换表包含设备特定的 I/O 例行程序地址，不具备相应操作的设备在其例行程序地址上要填“-1”。

5.2　数据传送控制方式

随着计算机技术的发展，数据传送控制方式也在不断地发展。当引入中断机制后，数据传送从最简单的程序直接控制方式发展为中断控制方式；DMA 控制器的出现，使数据传送的传输单位从字节扩大到数据块；而通道控制方式的出现，使得数据传送能独立进行而无需 CPU 干预，CPU 可从繁杂的数据传送控制中解脱出来，以便更多地进行数据处理，通道和 CPU 的并行工作无疑会明显提高工作效率。下面就数据传送的四种控制方式分别进行描述。

5.2.1 程序直接控制方式

程序直接控制方式是指由程序直接控制内存或 CPU 和外围设备之间进行信息传送的方式。通常又称为“忙－等”方式或循环测试方式。

在数据传送过程中，必不可少的一个硬件设备是 I/O 控制器，它是操作系统软件和硬件设备之间的接口，它接收 CPU 的命令，并控制 I/O 设备进行实际的操作。

I/O 控制器有两个寄存器：控制状态寄存器和数据缓冲寄存器。控制状态寄存器有几个重要的信息位：启动位、完成位、忙位等。“启动位”置 1，设备可以立即工作；“完成位”置 1，表示外设已完成一次操作；“忙位”表示设备是否处于忙碌状态。

数据缓冲寄存器是进行数据传送的缓冲区。当输入数据时，先将数据送入数据缓冲寄存器，然后由 CPU 从中取走数据。反之，当输出数据时，先把数据送入数据缓冲寄存器，然后及时由输出设备将其取走，进行具体的输出。

下面讲述程序直接控制方式的工作过程。由于数据传送过程中输入和输出的情况比较类似，下面只给出输出数据时的工作过程。

（1）把一个启动位为“1”的控制字写入该设备的控制状态寄存器。

（2）将需输出数据送到数据缓冲寄存器。

（3）测试控制状态寄存器中的“完成位”，若为 0，转（2），否则转（4）。

（4）输出设备将数据缓冲寄存器中的数据取走进行实际的输出。

程序直接控制方式虽然比较简单，也不需要多少硬件支持，但它存在以下明显的缺点。

（1）CPU 利用率低，CPU 与外围设备只能串行工作。由于 CPU 的工作速度远远高于外围设备的速度，使得 CPU 大量时间处于等待和空闲状态，CPU 利用率大大降低。

（2）外设利用率低，外设之间不能并行工作。

5.2.2 中断控制方式

一个进程占有处理器运行时，由于自身或外界的原因使运行被打断，让操作系统处理所出现的事件，到适当的时候再让被打断的进程继续运行，我们称这个进程运行被中断了，引起中断的事件称为中断源，对出现的事件进行处理的程序称为中断处理程序。

对于不同的计算机，它们的中断源不尽相同，但从中断事件的性质来说，可以分为强迫性中断事件和自愿性中断事件。前者不是正在运行的进程所期待的，而是由于外部的请求或某些意外事件而使正在运行的进程被打断；后者是正在运行的进程所期待的，是正在运行的进程执行一条“访管指令”请求系统调用为其服务所引起的中断。

为了克服程序直接控制方式的缺点，提高 CPU 的利用率，应使 CPU 与外设并行工作，这种方式要求在 I/O 控制器的控制状态寄存器中存放相应的“中断允许位”。

中断控制方式下的数据的输入按以下步骤进行：

（1）进程需要数据时，将允许启动和允许中断的控制字写入设备控制状态寄存器中，启动该设备进行输入操作。

（2）该进程放弃处理机，等待输入的完成。操作系统进程调度程序调度其他就绪进程占用处理机。

（3）当输入完成时，输入设备通过中断请求线向 CPU 发出中断请求信号。CPU 在接收

到中断信号之后，转向中断处理程序。

（4）中断处理程序首先保护现场，然后把输入缓冲寄存器中的数据传送到某一特定单元中去，同时将等待输入完成的那个进程唤醒，进入就绪状态，最后恢复现场，并返回到被中断的进程继续执行。

（5）在以后的某一时刻，操作系统进程调度程序选中提出的请求并得到获取数据的进程，该进程从约定的内存特定单元中取出数据继续工作。

此方式下的输出操作与输入操作基本类似。在中断控制方式中，CPU 在执行其他的进程时，假如那个进程也要求输入或输出操作，CPU 也可以发出启动不同设备的启动指令和允许中断指令，从而做到设备与设备间的并行操作以及设备和 CPU 间的并行操作。

尽管中断控制方式与程序直接控制方式相比，CPU 的利用率大大提高且能支持外设间的并行操作，避免了 CPU 循环测试控制状态寄存器这一工作，但它仍存在许多问题。其中最大的缺点是：每台设备输入/输出数据时，相应的中断 CPU 的次数也会增多，这会使 CPU 的有效计算时间大大减少，为解决这一问题，又产生了 DMA 控制方式和通道控制方式。

5.2.3　DMA 方式

DMA 方式又称直接存储器访问（Direct Memory Access）方式。其基本思想是在外设和主存之间开辟直接的数据交换通路。

在 DMA 方式中，I/O 控制器有比前两种方式更强的功能。DMA 控制器除有控制状态寄存器和数据缓冲寄存器外，还包括传送字节计数器和内存地址寄存器等。DMA 控制器可用来代替 CPU 控制内存和外设之间进行成批的数据交换。

DMA 方式的特点是：

（1）数据传送的基本单位是数据块。即 CPU 与 I/O 设备之间，每次传送的至少是一个数据块。

（2）所传送的数据是从设备送内存，或者相反。

（3）仅在传送一个或多个数据块的开始和结束时，才需中断 CPU，请求干预，整块数据的传送是在 DMA 控制器控制下完成的。

从 DMA 方式的特点可以看出，DMA 方式较之中断控制方式成百倍地减少了 CPU 对 I/O 控制的干预，进一步提高了 CPU 的使用效率，同时也提高了 CPU 与 I/O 设备的并行操作程度。

在 DMA 方式下，DMA 控制器与 CPU，主存及 I/O 设备之间的关系如图 5.2 所示。

DMA 方式下的数据输入处理过程如下：

（1）当某一进程要求设备输入数据时，CPU 把准备存放输入数据的内存基址及要传送的字节数据分别送入 DMA 控制器中的内存地址寄存器和传送字节计数器。

（2）将控制状态寄存器中的数据允许位和启动位置“1”，启动设备进行成批的数据输入。

（3）该进程进入等待状态，等待数据输入的完成，操作系统进程调度程序调度其他进程占用 CPU。

（4）在 DMA 控制器的控制下，按内存地址寄存器中的内容把数据缓冲寄存器的数据源源不断地写入到相应的主存单元，直至所有的数据全部传送完毕。

（5）输入完成时，DMA 控制器通过中断请求线发出中断信号，CPU 收到后转中断处理程序进行善后处理。

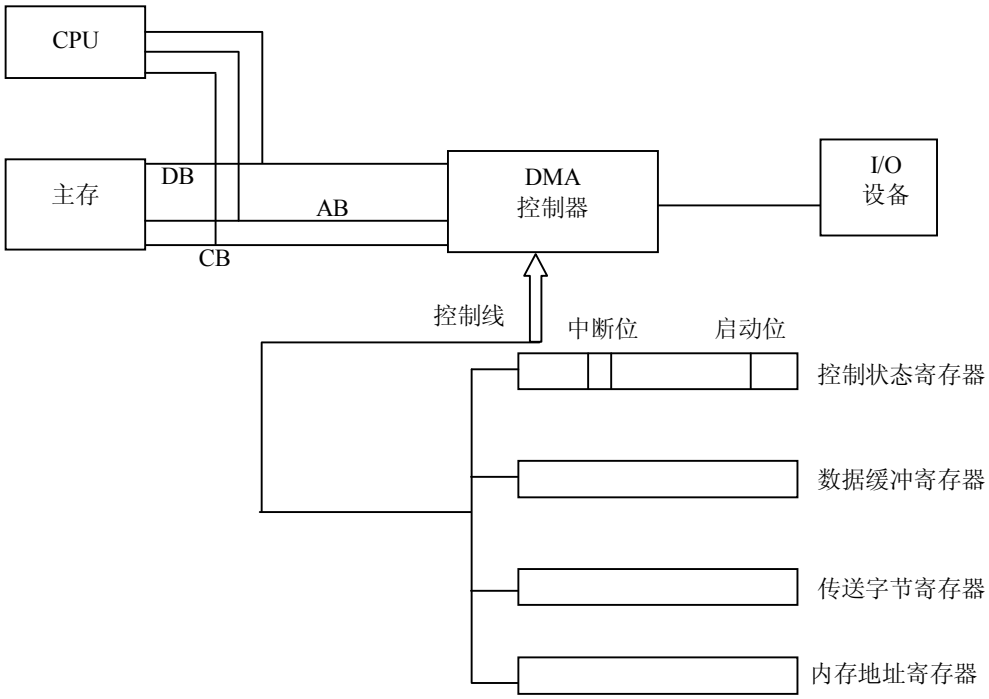

图 5.2 DMA 控制器与其他部件的关系

（6）中断处理结束时，CPU 返回被中断进程处执行。

（7）当操作系统进程调度程序调度到该进程时，该进程按指定的内存基址和实际传送的数据对输入数据进行加工处理。

虽然 DMA 方式比前两种方式有明显的进步，但它仍存在一定的局限性。首先，DMA 方式对外设的管理和某些操作仍由 CPU 控制；另外，多个 DMA 控制器的同时使用可能会引起内存地址的冲突，同时也是不经济的。为了克服以上的毛病，于是就出现了通道控制方式。

5.2.4 通道控制方式

如何实现输入/输出的操作呢？这需要计算机系统中的硬件技术和软件技术密切配合。为了使物理特性不同的外围设备能够用统一的标准接口连接到系统中，计算机系统引入了独立系统的通道结构。通道的出现使计算机系统的性能得到提高，它把中央处理器从繁忙的输入/输出操作中空闲出来，为计算机系统中各个部件能并行工作创造条件。

通道控制方式与 DMA 方式相类似，也是一种内存和设备直接进行数据交换的方式。与 DMA 方式不同的是，在通道控制方式中，数据传送方向存放数据的内存基址及传送的数据块长度均由一个专门负责输入/输出的硬件——通道来控制。另外，DMA 方式每台设备至少需要一个 DMA 控制器，而通道控制方式中，一个通道可控制多台设备与内存进行数据交换。

通道是一个独立于 CPU 的专门负责输入/输出控制的处理机，它和设备控制器一起控制设备与内存直接进行数据交换。它有自己的通道指令，这些指令受 CPU 控制启动，并在操作结束时向 CPU 发出中断信号。

每条通道指令应包含以下的内容：

（1）操作码：它规定指令所执行的操作，如读、写等。

（2）内存地址：标明数据传送时内存的首址。

（3）计数：表示传送数据的字节数。

（4）通道程序结束位 R_0，表示通道程序是否结束。R_0=1 表示本条指令是最后一条指令。

（5）记录结束标志 R_1，表示所处理的记录是否结束。R_1=1 表示这是处理某记录的最后一条指令。

下面给出一个由两条通道指令所构成的简单程序。该程序是将内存中不同地址的数据写成多个记录。

操作	R_0	R_1	计数	内存地址
WRITE	0	0	80	1420
WRITE	0	1	170	2120

其中，这两条指令是将单元 1420～1499 中的 80 个字符和单元 2120～2289 中的 170 个字符写成一个记录。

在通道控制方式中，通道有三种不同的类型，即字节多路通道、选择通道和数组多路通道。由这三种通道组成的数据传送结构如图 5.3 所示。

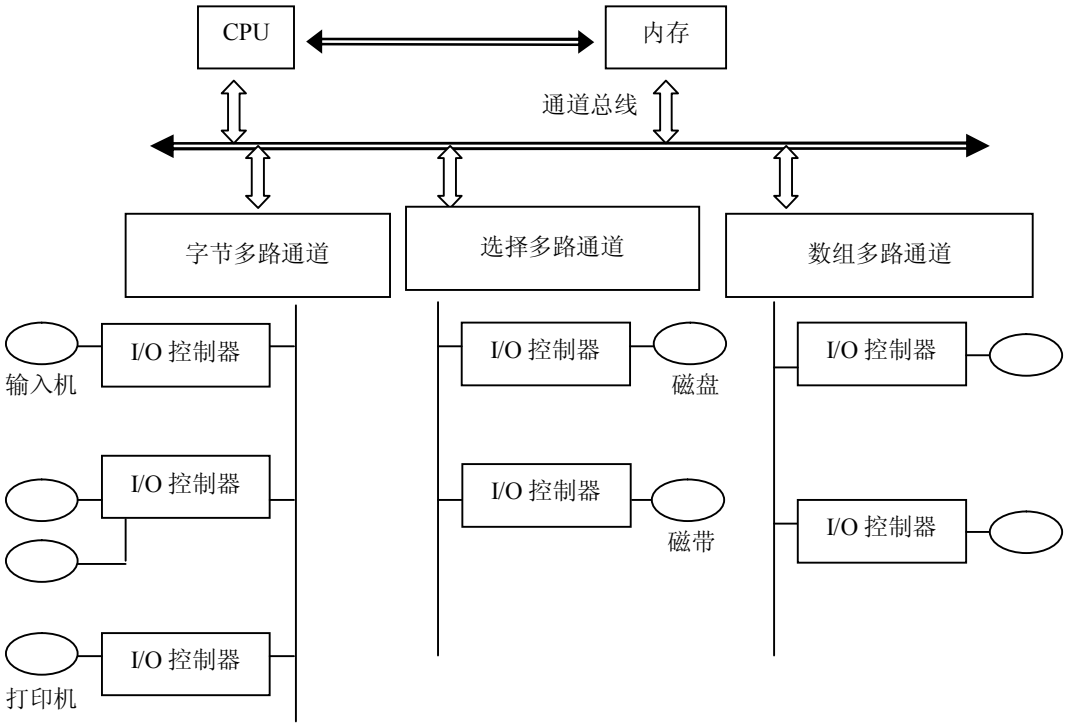

图 5.3　通道方式的数据传送结构

字节多路通道以字节为单位传送信息，它可以分时地执行多个通道程序。主要用来连接大量低速设备，如终端、卡片机等。

选择多路通道一次只能执行一个通道程序，只有执行完一个通道程序后才能执行另一个

通道程序，所以它一次只能控制一台设备进行 I/O 操作。但它具有传送速度快的特点，因而用来连接高速外部设备，如磁盘机等。

数组多路通道以分时方式执行几个通道程序，同时以块为单位传送数据，所以它具有字节多路通道的分时操作及选择多路通道连接较高速外设的特点。一般连接中速设备，如磁带机等。

通道控制方式的数据传输过程如下：

（1）当进程要求设备输入时，CPU 发指令指明 I/O 操作、设备号和对应通道。

（2）对应通道收到 CPU 发来的启动指令后，读出内存中的通道指令程序、设置对应设备的控制状态寄存器的初值。

（3）设备按通道指令的要求，把数据送往内存指定区域。

（4）若传送结束，I/O 控制器通过中断请求线发中断信号请求 CPU 做中断处理。

（5）中断处理结束后，CPU 返回到被中断进程处继续执行。

（6）当进程调度程序选中这个已得到数据的进程后，才能进行加工处理。

与前面几种方式相比，通道控制方式有更强的 I/O 处理能力。有关 I/O 的工作委托通道去做，当通道完成了 I/O 任务后，向 CPU 发中断信号，请求 CPU 处理。这样就使 CPU 基本上摆脱了 I/O 控制工作，大大提高了 CPU 的工作效率及与外设间的并行工作程度。

5.3 I/O 应用接口

为了便于 I/O 设备按统一的标准方式使用，就要讨论一下操作系统的组织技术和接口。例如，应用程序为什么打开磁盘上的文件不需要知道什么是磁盘，设备增加到计算机中为什么不必中断操作系统。为了解决这样的问题，必须从不同的 I/O 设备中抽象出一些通用类型。每个通用类型都可以通过一组标准函数（即接口）来访问。具体的差别被内核模块（设备驱动程序）所隐藏，这些设备驱动程序一方面可以被专门的设备使用，另一方面也提供了一组标准的接口。图 5.4 说明了内核中与 I/O 相关的部分是如何按层次进行组织的。

设备驱动程序层的作用是为内核 I/O 子系统隐藏设备控制器之间的差别，同样，I/O 系统调用接口包装设备行为是为应用程序隐藏了硬件差异。将 I/O 子系统和硬件分离简化了操作系统开发人员的任务，也有利于硬件制造商。硬件制造商可以设计新的硬件设备并使其与现有控制器接口相兼容，或为操作系统编写新的设备驱动器以与新硬件连接。新的设备可以与计算机连接，而不需要等待操作系统厂商开发支持代码。

各种设备之间是有很大差异的，但是对于应用程序的访问而言，多数差别都被操作系统所隐藏。因此设备也就被分成少量的几种传统类型，由此产生的设备访问方式被证明十分有效而被广泛应用。主要的访问方式有块 I/O、字符流 I/O、内存映射访问和网络套接字。

1. 块设备接口

块设备接口规定了访问磁盘驱动器和其他基于块设备所需的各个方面。一般的设备应知道 read 和 write 命令的含义，随机访问设备还应明白 seek 命令是用来描述下次传输那个块的。应用程序通常通过文件系统接口访问设备。操作系统将块设备当做一个简单的线性块数组来访问。read、write 和 seek 描述了块设备的基本特点，因此应用程序就不必关心这些设备的底层差别了。

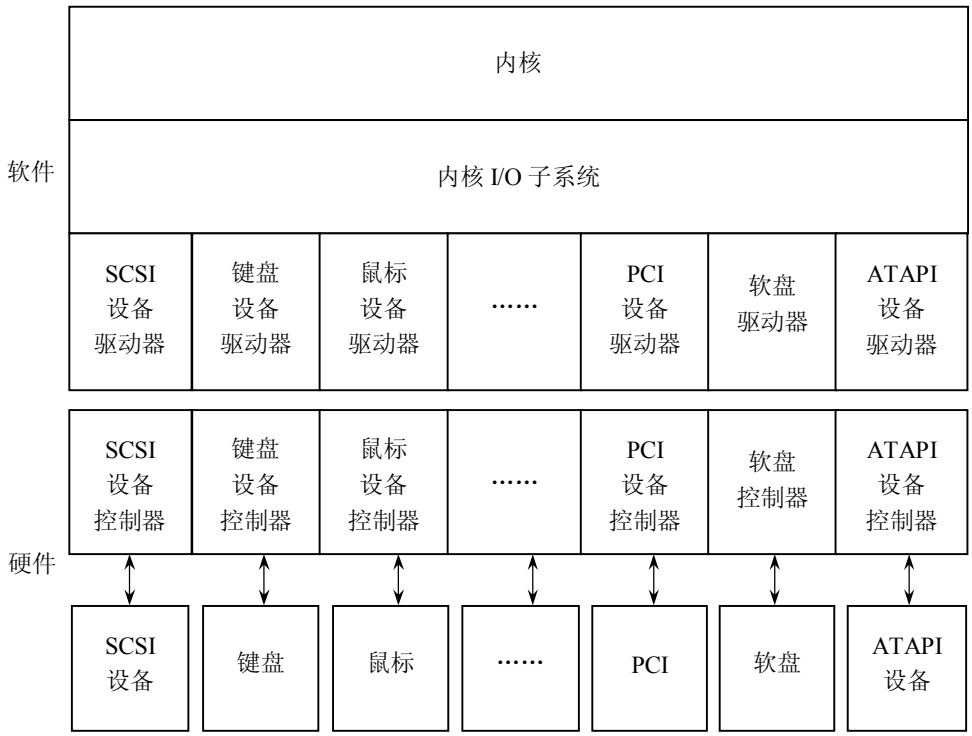

图 5.4　内核 I/O 结构

2. 字符流接口

字符流接口的基本系统调用使得应用系统可以 get 或 put 字符，键盘就是这种设备。在此接口上，还可以构造库以提供具有缓冲和编辑功能的按行访问。例如，当用户输入一个退格键，之前的字符可以从输入流中被删除。

3. 内存映射文件访问

内存映射文件访问是块设备驱动程序之上的。内存映射接口没有提供 read 和 write 命令，而提供了通过内存中的字节数组访问磁盘存储。文件映射到内存的系统调用会返回一个字符数组的虚拟内存地址，该字符数组包含了文件的一个拷贝。因为数据传输采用了与虚拟内存访问相同的机制，所以内存映射文件访问比较高效。由于访问内存映射文件和内存读写一样简单，因此内存映射也有益于程序员。提供虚拟内存的操作系统通常也为内核服务提供映射接口。例如，要执行程序，操作系统将可执行的程序映射到内存中，并切换到可执行程序的入口地址。

4. 网络套接字

由于网络 I/O 的性能和访问特点和磁盘 I/O 有很大区别，大多数操作系统提供的接口是网络套接字接口，如 UNIX 和 Windows NT。套接字好比电源插座可以和任何电器上的插头相连。套接字接口的系统调用可以让应用程序创建一个套接字，并用其监听远程应用程序创建的套接字，然后通过连接两个套接字发送和接收数据。为了实现服务器的功能，套接字接口还提供了 select 函数，用于管理一组套接字。调用 select 可以知道哪个套接字有接收的数据要处理，哪个套接字有空间可以接收数据。使用 select 就不必再使用轮询和忙等待来处理

网络 I/O 了。套接字封装了基本的网络功能，大大加快了使用网络硬件和网络协议的分布式应用程序的创建。

5.4 缓冲技术与高速缓存

在现代操作系统中，设备与内存交换数据时，为提高 I/O 速度和设备的利用率及并行程度，大都需要借助缓冲技术来实现。缓冲可分为硬件缓冲及软件缓冲两种。硬件缓冲可用硬件缓冲器来实现，但由于成本太高，除一些关键部位采用外，一般情况下不采用硬件缓冲器。软件缓冲是一种应用广泛的缓冲机制，它由缓冲区和缓冲区管理两部分组成。缓冲区是指在 I/O 操作时用来临时存放输入输出数据的一块存储区域。下面介绍的是软件缓冲技术。

5.4.1 缓冲的引入

引入缓冲的主要目的有以下几点。

1. 缓和处理机和 I/O 设备间速度不匹配的矛盾

众所周知，处理机的速度大大高于 I/O 设备的速度，而通常的程序都是时而计算时而产生输入/输出的。如对于输出，若没有缓冲，则程序在输出数据时必然会由于打印机速度跟不上，而使 CPU 停下来等待；在计算阶段，打印机会空闲等待。在设置缓冲区后，计算进程可把数据首先输出到缓冲区，然后继续执行，打印机则可以从缓冲区取出数据慢慢打印。

2. 减少对 CPU 的中断次数

如从外设来的数据仅用一位缓冲来接收，则每收到一位数据时便中断一次，倘若设置一个 8 位的缓冲寄存器，则可使中断次数降为 1/8。很明显，用缓冲技术可减少对 CPU 的中断次数。

3. 提高 CPU 和 I/O 设备之间的并行性

引入缓冲后，在输入数据时，输入设备可将数据输入到缓冲区存放，与此同时 CPU 再进行计算工作。输出数据时，CPU 也在进行计算工作，而同时输出设备将缓冲区的数据取出慢慢打印。很明显，缓冲的引入可以提高 CPU 和 I/O 设备的并行性。

从上可以看出，缓冲技术在操作系统中所起的作用是比较明显的，下面就几种不同类型的缓冲——单缓冲、双缓冲、环形缓冲、缓冲池分别做以下描述。

5.4.2 单缓冲

单缓冲是操作系统提供的一种最简单的缓冲形式。每当一个进程发出一个 I/O 请求时，操作系统便在主存中为之分配一缓冲区，该缓冲区用来临时存放输入/输出数据。

在单缓冲形式下，数据输入的情形是这样的：当进程要求数据输入时，操作系统先控制外设将数据送往缓冲区存放，然后进程从缓冲区中取出数据继续运行。采用单缓冲方式可以缓和 CPU 和外设速度之间的矛盾，同时也可以使 CPU 和外设并行工作，但是它不能使设备和设备之间通过单缓冲达到并行操作。

单缓冲方式由于只有一个缓冲区，这一缓冲区在某一时刻能存放输入数据或输出数据，但不能既是输入数据又是输出数据，否则在缓冲区中的数据会引起混乱，所以此缓冲区可以认为是临界资源，不允许多进程同时访问它。在单缓冲方式下解决输入及输出的情形是这样

的：当数据输入到缓冲区时，输入设备在工作，而输出设备空闲；而当数据从缓冲区输出时，输出设备在工作，输入设备空闲。从上可以看出单缓冲方式不能通过缓冲区解决外设之间的并行问题，为解决此问题必须引入双缓冲。

5.4.3　双缓冲

解决外设之间并行工作的最简单的办法是设置双缓冲。在双缓冲方案中，具体的做法是为输入或输出操作设置两个缓冲区 buffer1 和 buffer2。当进程要求输入数据时，首先输入设备将数据送往缓冲区 buffer1，然后进程从 buffer1 中取出数据进行计算，在进程从 buffer1 中取数的同时，输入设备可向缓冲区 buffer2 送入数据。当缓冲区 buffer1 中的数取完时，进程又可从 buffer2 中提取数据，与此同时输入设备又可将数据送往 buffer1。如此交替使用 buffer1 和 buffer2 可使 CPU 和外设的并行程序进一步提高。当进程要求输出时的处理与输入时类似。

在双缓冲方式下解决外设并行工作的方案是这样的：输入时将数据送往缓冲区 buffer1，然后进程从 buffer1 中提取数据进行计算，输出时将数据送往缓冲区 buffer2，输出设备从 buffer2 中取出数据慢慢输出，与此同时输入设备又可以将数据送往 buffer1，进程从 buffer1 中提取数据进行计算。显然，在此方式中，输入设备和输出设备可能并行工作。

双缓冲方式和单缓冲方式相比，虽然双缓冲方式能进一步提高 CPU 和外设的并行程度，并能使输入设备和输出设备并行工作，但在实际系统中很少采用这一方式，这是因为在计算机系统中的外设很多，又有大量的输入和输出，同时双缓冲很难匹配设备和 CPU 的处理速度。因此现代计算机系统中一般使用环形缓冲或缓冲池结构。

5.4.4　环形缓冲

环形缓冲技术是在主存中分配一组大小相等的存储区作为缓冲区，并将这些缓冲区链接起来，每个缓冲区中有一个指向下一个缓冲区的指针，最后一个缓冲区的指针指向第一个缓冲区，这样 n 个缓冲区就成了一个环形。此外，系统中有个缓冲区链首指针指向第一个缓冲区。环形缓冲区结构如图 5.5 所示。

环形缓冲用于输入（输出）时除 START 指针指向第一个缓冲区外，还需要有两个指针 In 和 Out。对输入而言，先从设备接收数据到缓冲区中，In 指针指向可输入数据的第一个空缓冲区，当进程运行过程中需要数据时，从环行缓冲区中取一个装满数据的缓冲区从中取出数据，Out 指针指向可提取数据的第一个满缓冲区。

系统初启时，指针初始化为 Start=In=Out。当输入时，数据输入到 In 指针指向的缓冲区，输完后，In 指针指向下一个可用的空缓冲区；进程从缓冲区提取数据时，提取 Out 指针所指的缓冲区的内容，操作完后，Out 指针指向下一个满缓冲区。

在环形缓冲这一方案中，为保证并行操作必须有这样一种约束条件，即 In≠Out。在一般情形下，Out<In。当 Out 即将赶上 In 时，进程从缓冲区提取数据这一操作必须等待，当 In 即将赶上 Out 时，从设备输入数据这一操作也必须等待。

虽然，环形缓冲可解决双缓冲中存在的一些问题，但当系统较大时，会有许多这样的环形缓冲，不仅要耗费大量的内存空间，而且其利用率也不高。为提高缓冲区的利用率，目前广泛采用缓冲池技术。

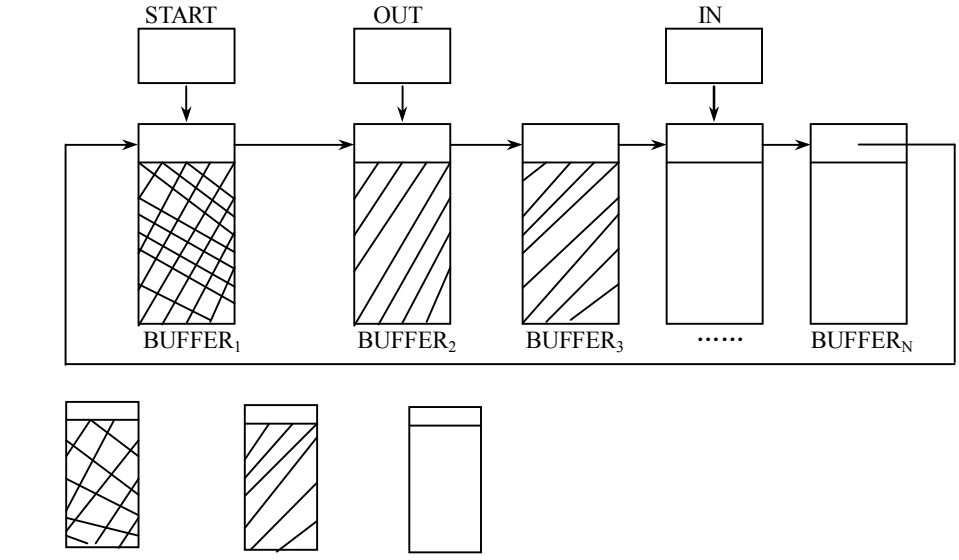

图 5.5 环形缓冲区结构

5.4.5 缓冲池

从自由主存中分配一组缓冲区即可构成缓冲池。在缓冲池中每个缓冲区的大小可以等于物理记录的大小，它们作为公共资源被共享，缓冲池既可用于输入，也可用于输出。下面从以下几个方面简单介绍一下缓冲池技术。

1. 缓冲池的组成

缓冲池中的缓冲区一般有以下三种类型：空闲缓冲区、装输入数据的缓冲区和装输出数据的缓冲区。为管理上的方便，系统将同类型的缓冲区连成队列，于是就有以下三个队列：①空缓冲区队列 emq。队首指针为 F(emq)，队尾指针为 L(emq)；②装输入数据的输入缓冲队列 inq。队首指针为 F(inq)，队尾指针为 L(inq)；③装输出数据的输出缓冲队列 outq。队首指针为 F(outq)，队尾指针为 L(outq)，如图 5.6 所示。

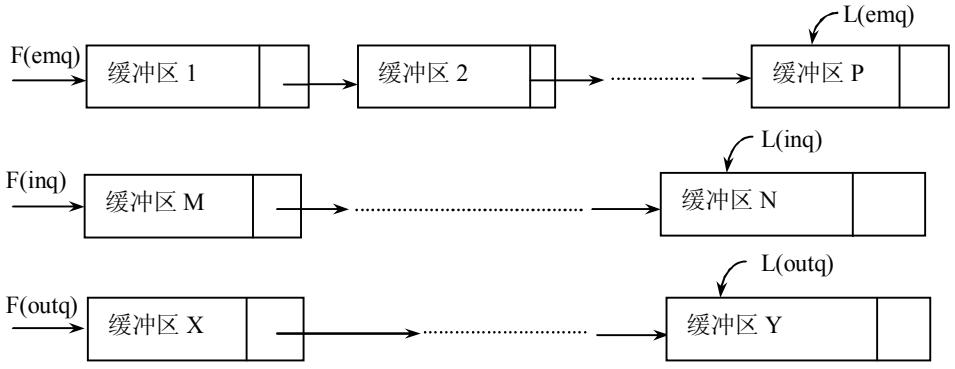

图 5.6 缓冲区队列缓冲区

除上述三个队列外，还应具有四种工作缓冲区：①用于收容输入数据的缓冲区 hin；②用于提取输入数据的缓冲区 sin；③用于收容输出数据的缓冲区 hout；④用于提取输出数据的缓冲区 sout，如图 5.7 所示。

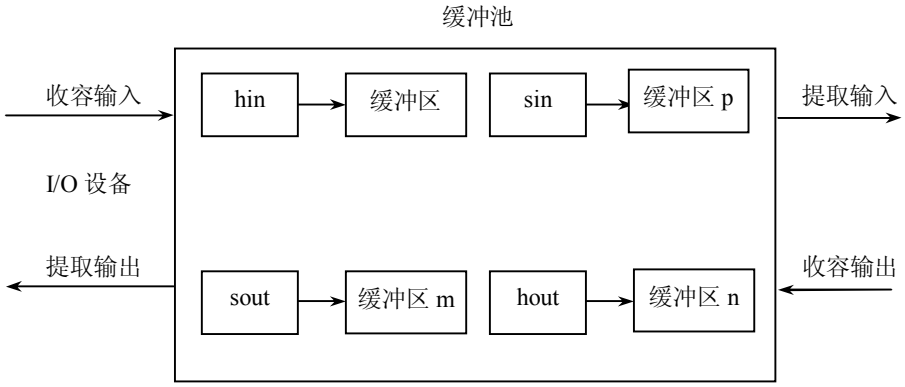

图 5.7　缓冲池的工作缓冲区

2. 缓冲池的工作方式

缓冲区可以在收容输入、提取输入、收容输出和提取输出四种方式下工作。

（1）收容输入。当输入进程需要输入数据时，系统从 emq 队列的队首摘下一空缓冲区，把它作为收容工作缓冲区 hin。然后，将数据输入其中，再将它挂在队列 inq 的末尾。

（2）提取输入。当计算进程需要输入数据进行计算时，系统从输入队列队首取得一缓冲区作为提取输入工作缓冲区 sin，计算进程从中提取数据，当进程用完该缓冲区数据后，再将它挂到空缓冲区队列 emq 末尾。

（3）收容输出。当计算进程需要输出数据时，系统从空缓冲队列 emq 的队首取得一空缓冲区，将它作为收容输出工作缓冲区 hout。hout 输出数据后，再将它挂在队列 outq 末尾。

（4）提取输出。当要进行输出操作时，从输出队列 outq 的队首取得一缓冲区，作为提取输出工作缓冲区 sout。当数据提取完毕后，再将该缓冲区挂在空缓冲区队列 emq 的末尾。

5.4.6　高速缓存

高速缓存（cache）是可以保留数据拷贝的高速内存。高速缓存拷贝的访问速度要比原始数据访问更为高效。例如，一个正在运行的进程的指令既存储在磁盘上，也存储在物理内存上，还被复制到 CPU 的二级和一级的高速缓存中。缓冲和高速缓存的差别是缓冲只保留数据仅有的一个现存拷贝，而根据定义，高速缓存只是提供了驻留在其他地方的数据的一个高速拷贝。

高速缓存和缓冲是两个不同的功能，但有时同一块内存区域也可以同时用于这两个目的。例如，为了文件的操作和有效调度磁盘 I/O，操作系统在内存中开辟缓冲区来保留磁盘数据。这些缓冲也可用作高速缓存，以改善对某些文件的 I/O 操作效率，这些文件可以被多个程序所共享或者被快速地写入和重读。当内核收到 I/O 请求时，内核首先检查高速缓存以确认文件是否在内存中。如果是，物理磁盘 I/O 就可以避免或延迟。而且，几秒内的磁盘写也可以累积到缓冲区中，这样就收集到大量的传输以允许高速的写调度。

5.5 设备分配技术

在现代操作系统中要求设备为其服务的进程数量总是多于设备数量，所以往往有多个进程同时要求占有某个设备的使用权，而这个设备一次只能分配给一个进程使用，到底分配给哪一个进程，这就涉及到设备分配问题。

根据设备的特性可把设备分为独享设备、共享设备和虚拟设备三种。下面就这三种类型的设备分别讨论它们的分配技术。

5.5.1 独享设备的分配

所谓独享设备是指这类设备被分配给一个作业后，被这个作业所独占使用，其他任何作业不能使用，直到该作业释放该设备为止。常见的独享设备有打印机、光电输入机等。通常独享设备在使用前或使用过程中需要人工干预，如为打印机装纸、从打印机上取下打印结果等。

针对独享设备，系统一般采用静态分配方式。即在一个作业执行前，将它所需要使用的这类设备分配给它，当作业结束撤离时，才将分配给它的独享设备收回。静态分配方式实现简单，而且绝对不会发生死锁，但采用静态分配方式进行设备分配时会造成设备的利用率不高。

5.5.2 共享设备的分配

所谓共享设备是指允许多个用户共同使用的设备。如磁盘、磁鼓等设备，可由多个进程同时进行访问。设备的共享有两层含义：一是指对设备介质的共享，如磁盘上的各扇区，多个用户可以把信息存于同一设备的不同扇区上，这种共享一般是以文件的形式存放的，有关存储介质的分配及管理方面的问题将在 6.9 节中介绍；二是指对磁盘等驱动器的共享，多个用户访问这些设备上的信息是通过驱动器来实现的。对磁盘、磁鼓等设备采用共享分配，可将这些设备交叉分给多个用户或多个进程使用，很明显提高了设备的利用率。

对共享设备的分配一般采用动态分配这一方式，所谓动态分配是指在进程执行过程中，当进程需要使用设备时，通过系统调用命令向系统提出设备请求，系统按一定的策略给进程分配所需设备，进程一旦使用完毕就立即释放。显然，这种分配方式提高了设备的利用率，但是容易出现死锁，因此，在选择分配方法时应极力避免死锁的发生。

常见的设备分配方法有两种。

1. 先来先服务

当多个进程对某一设备提出 I/O 请求，或者是在同一设备上进行多次 I/O 操作时，该算法是根据进程对设备提出请求的先后次序，将这些进程排成一个设备请求队列，当设备空闲时，设备分配程序总是把设备首先分配给队首进程，队首进程就可以在该设备上进行 I/O 操作。

2. 优先级高者优先

优先级高者是指发出 I/O 请求命令的进程，如果进程的优先级高，则它的 I/O 请求的优先级也高，这有利于进程尽快运行完毕尽早释放它所占用的资源。因此，优先级高者优先算法是把进程请求某设备的 I/O 请求命令按进程的优先级排成队列，当设备空闲时，设备分配程序将该设备分配给队首进程，队首进程就可以进行 I/O 操作。如进程的优先级相同，则按先来先服务的方法进行排队。

5.5.3　虚拟设备的分配与 Spooling 技术

1. 虚拟设备的分配

系统中独享设备的数量有限，且独享设备的分配往往采用静态分配这一方式，使得许多进程会因为等待某些独享设备而处于等待状态，而分得独享设备的进程，在其整个运行期间往往占有这个设备，而不是经常使用这些设备，因而使这些设备的利用率很低。为克服这一缺点，可通过共享设备来模拟独享设备以提高设备利用率及系统效率，于是就有了虚拟设备。

所谓虚拟设备是指代替独享设备的那部分存储空间及有关的控制结构。对虚拟设备采用的是虚拟分配，其过程是：当进程中请求独享设备时，系统将共享设备的一部分存储空间分配给它。进程与设备交换信息时，系统把要交换的信息存放在这部分存储空间，在适当的时候对信息做相应的处理。如打印时，把要打印的信息送到某个存储空间中，在打印机空闲时将存储空间上的信息送到打印机上打印出来。

2. Spooling 技术

通过共享设备来模拟独享设备所采用的操作是假脱机操作，即在联机情况下外部设备同时操作所使用的技术称之为 Spooling 技术。

要实现联机方式下的外设同时操作，必须有一个完整的系统，这一系统称之为假脱机系统（Spooling 系统）。下面介绍 Spooling 系统。

3. Spooling 系统的组成

（1）输入井和输出井。输入井和输出井是在外存上开辟的两大存储空间。其中，输入井用于存放 I/O 设备输入的数据，输出井用于存放用户程序的输出数据。

（2）输入缓冲区和输出缓冲区。输入缓冲区用于暂存由输入设备送来的数据，以后再送到输入井中。输出缓冲区用于暂存从输出井送来的数据，以后再送给输出设备输出。

（3）输入进程和输出进程。输入进程模拟脱机输入时的外围控制机，它将数据从输入机经过输入缓冲区送到输入井中。当 CPU 需要数据时，直接从输入井中提取数据到内存。输出进程模拟脱机输出时的外围控制机，它把用户要求输出的数据从内存送到输出井；当输出设备空闲时，将输出井中的数据经过输出缓冲区送往输出设备输出。图 5.8 是 Spooling 系统组成的示意图。

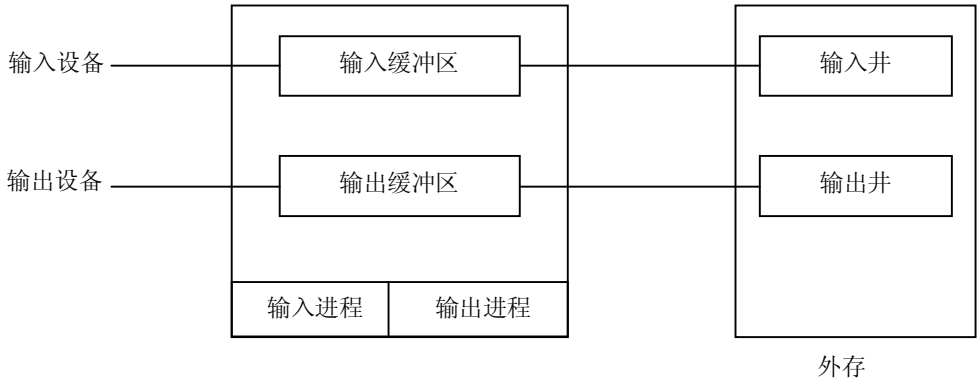

图 5.8　Spooling 系统的组成

4. Spooling 系统的特点

（1）提高了 I/O 速度。从对低速 I/O 设备进行的 I/O 操作变为对输入井或输出井的操作，如同脱机操作一样，提高了 I/O 速度，缓和了 CPU 与低速 I/O 设备速度不匹配的矛盾。

（2）设备并没有分配给任何进程。在输入井或输出井中，分配给进程的是一个存储区和建立一张 I/O 请求表。

（3）实现了虚拟设备功能。多个进程同时使用一独享设备，而对每一进程而言，都认为自己独占这一设备，不过，该设备是逻辑上的设备。

5.6 I/O 进程控制

从用户进程的 I/O 请求开始，给用户进程分配设备和启动 I/O 操作完成之后响应中断，一直到进行善后处理为止的整个系统控制过程称为 I/O 进程控制。下面介绍 I/O 控制过程。

5.6.1 用户进程的输入输出请求

用户进程的输入输出请求包括：申请进行 I/O 操作的逻辑设备名、要求的操作、传送数据的长度和起始地址等。典型的用户进程的输入输出请求可通过下述通用的系统调用命令来实现：

doio(dev,mode,bmount,bddr)

其中：dev 表示执行 I/O 的逻辑设备名；mode 表示操作的类型，如读、写等；bmount 表示传输的数目；bddr 表示数据传送的地址。

当用户发出请求 I/O 传输的系统调用命令时，设备管理模块中有一设备驱动程序处理该命令，下面讨论设备驱动程序。

5.6.2 设备驱动程序

1. 设备驱动程序的功能

设备驱动程序的主要功能有以下几点：

（1）实现逻辑设备到物理设备的转换。

（2）检查 I/O 请求的合法性，了解 I/O 设备的状态，传递有关参数，设置设备的工作方式。

（3）发出 I/O 命令，启动相应的 I/O 设备，完成相应的 I/O 操作。

（4）及时响应中断请求，并根据中断类型调用相应的中断处理程序进行处理。

2. 设备驱动程序的处理过程

不同的设备有不同的设备驱动程序，但设备驱动程序大都可分为两部分，除有能驱动 I/O 设备工作的驱动程序外，还有设备中断处理 I/O 完成后的工作程序。

设备驱动程序的主要任务是驱动设备，但在驱动之前有一些准备工作，只有在完成所有的准备工作后，才向设备控制器发送一条启动命令。设备处理程序的处理过程如下：

（1）将逻辑设备转换为物理设备。当逻辑设备打开时，在相应的逻辑设备描述器中记录了该逻辑设备与实际物理设备之间的联系。

（2）I/O 请求的合法性检查。对于输入输出设备在某一时刻只能进行输入或输出操作，如设备不支持这次 I/O 请求，则认为这次 I/O 请求非法。对于磁盘、磁带之类的设备，虽然是

读写设备，但在某一时刻也只能进行其中的一项操作，如规定的是写操作，则读操作是非法的。

（3）检查设备的状态。启动某设备的前提是该设备处于空闲状态，因此启动设备之前，必须从设备控制器的状态寄存器中读出该设备的状态，只有该设备处于空闲状态时，才能启动该设备控制器，否则应等待。

（4）传送参数。许多设备除应向其控制器发出启动命令外，还必须传送相应的参数。如在启动磁盘读写之前，应先将传送的字节数、数据应达到的主存起始地址送入控制器的相应寄存器中。

（5）启动 I/O 设备。完成上述的准备工作后，驱动程序向控制器的命令寄存器中传送相应的控制命令。驱动程序发出 I/O 命令后，基本的 I/O 是在设备控制器的控制下进行的。通常 I/O 操作要完成的工作很多，需要一定时间，此时驱动程序进程把自己阻塞起来，直到中断到来才将它唤醒。

5.6.3　中断处理程序的处理过程

I/O 设备在设备控制器的控制下完成了 I/O 操作后，控制器便向 CPU 发出一中断请求，CPU 响应后便转向中断处理程序。中断处理程序的处理过程如下：

1. 唤醒被阻塞的驱动程序进程

无论是什么类型的中断，当中断处理程序开始执行时，都必须去唤醒阻塞的驱动程序进程。

2. 保护被中断进程的现场

将被中断进程的现场信息保留在相应的中断栈中。

3. 分析中断原因、转入相应的设备中断处理程序

对中断源进行测试，找出本次中断的 I/O 设备，将该中断处理程序的入口地址装入到程序计数器中，使处理机转向中断处理程序。

4. 进行中断处理

执行相应的中断处理程序进行中断处理。如是正常完成中断，驱动程序进行中断处理；如是异常结束，则根据发生异常的原因做相应的处理。

5. 恢复被中断进程的现场

中断处理程序完成后，可将保存在中断栈中的被中断进程的现场信息取出装入到相应的寄存器中。

5.6.4　I/O 调度

上面我们介绍了一个 I/O 申请的处理过程，但当有一组 I/O 操作申请时，将如何确定一个好的顺序来执行这些请求呢？应用程序所发布的系统调用的顺序并不一定总是最好的选择，I/O 调度可以改善系统整体性能，能在进程之间公平地共享设备，还可以减少 I/O 完成所需的平均等待时间。这里用一个简单的例子说明这个问题。假设磁头位于磁盘的开始处，有 3 个应用程序都发布了对该磁盘的阻塞读调用。第一个应用程序要读磁盘结束部分的块，第二个应用程序要读磁盘开始部分的块，第三个应用程序要读磁盘开始部分的块。操作系统如果按照 2、3、1 的顺序处理读调用，可以减少磁头移动距离，提高 I/O 效率。按这种方法来重新安排 I/O 申请服务的就是 I/O 调度的核心。

操作系统开发人员是通过为每个设备设置一个请求队列来实现调度的。当应用程序发布了阻塞 I/O 系统的调用时，该请求就被加到相应设备的队列上。I/O 调度可以重新调整队列的顺序来改善系统总体效率和应用程序的平均响应时间。通过操作系统的 I/O 调度，可以使所有的应用程度都得到比较好的服务；也可以给予那些对延迟很敏感的请求较优先的服务。

本章从设备的分类、设备管理的任务和功能出发，对设备和中央处理器之间的数据传送的控制方式、缓冲技术、设备分配技术、I/O 进程控制以及设备驱动程序进行了比较充分的介绍和讨论。

常见的设备和 CPU 之间数据传送的控制方式有 4 种，它们是直接控制方式、中断控制方式、DMA 方式和通道控制方式。程序直接控制方式比较简单而且不需要多少硬件支持，但 CPU 和外设只能串行工作且 CPU 要花大量的时间进行循环测试。中断控制方式虽然在一定程度上解决了上述问题，但由于一次数据传输中断次数很多，使得 CPU 要花费较多的时间处理中断，且能够并行操作的设备总数也受到中断处理的时间限制。DMA 方式和通道控制方式较好地解决了上述问题。这 2 种方式采用了外设和内存直接交换数据的方式，只有在一段数据传送结束时，这 2 种方式才发出中断信号请求 CPU 做善后处理，从而大大减少了 CPU 的工作负担。这 2 种方式的区别是，DMA 方式要求 CPU 执行设备驱动程序启动设备，给出存放数据的内存地址以及操作方式和传送字节长度等，而通道控制方式则是在 CPU 发出 I/O 启动命令后，由通道指令来完成这些工作。

I/O 应用接口实际是在不同设备上进行抽象得出来的一些通用的设备类型。这样就隐藏了具体设备的底层特性，更有利于各种设备与计算机相连。我们主要介绍了四种比较常见的接口类型：块 I/O、字符流 I/O、内存映射访问和网络套接字。

缓冲是为了匹配设备和 CPU 的处理速度，进一步减少中断次数及提高 CPU 和设备之间的并行性而引入的。缓冲有硬缓冲和软缓冲。我们主要介绍 4 种软缓冲技术：单缓冲、双缓冲、环形缓冲及缓冲池。单缓冲是最简单的一种缓冲技术，实现起来简单，但它不能解决外设之间的并行问题。双缓冲虽然能解决上述问题，但由于系统中的外设很多，又有大量的输入和输出，很难匹配设备和 CPU 的处理速度。环形缓冲可解决上述问题，但当系统较大时，会有很多环形缓冲区，不仅耗费大量的内存空间，且利用率不高，因此引入缓冲池技术。需要注意的是由于缓冲区是临界资源，因此对缓冲区或缓冲队列的操作必须互斥。高速缓存只是提供了驻留在其他地方的数据的一个高速拷贝，和缓冲技术是有差别的。

针对设备分配，介绍了动态分配及静态分配。就设备的 3 种类型：独享设备、共享设备及虚拟设备介绍了独享分配、共享分配及虚拟分配，还介绍了设备分配的 2 种常见的算法，然后介绍的是 Spooling 技术。

I/O 进程控制是对整个 I/O 操作的控制，包括对用户进程 I/O 请求命令的处理，启动通道指令或驱动程序进行真正的 I/O 操作，以及分析中断原因、响应中断等。设备驱动程序是驱动物理设备和 DMA 控制器或 I/O 控制器等直接进行 I/O 操作的子程序集合。它们负责设置相应设备有关寄存器的值、启动设备进行 I/O 操作，指定操作的类型和数据的流向等。当有一组 I/O 操作同时申请时，可通过 I/O 调度以最佳的效率执行。

一、选择题

1．通道是一种（　　）。

A．I/O 端口　　B．数据通道

C．I/O 专用处理机　　D．软件工具

2．采用 Spooling 技术的目的是（　　）。

A．提高独享设备的利用率　　B．提高主机效率

C．减轻用户编程负担　　D．提高程序的运行速度

3．采用假脱机技术，将磁盘的一部分作为公共缓冲区以代替打印机，用户对打印机的操作实际上是对磁盘的存储操作，用以代替打印机部分是指（　　）。

A．独占设备　　B．共享设备

C．虚拟设备　　D．一般物理设备

4．如果 I/O 设备与存储设备进行数据交换不经过 CPU 来完成，这种数据交换方式是（　　）。

A．程序查询　　B．中断方式

C．DMA 方式　　D．无条件存取方式

5．在操作系统中，下列（　　）指的是一种硬件机制。

A．通道技术　　B．缓冲区

C．Spooling 技术　　D．内存覆盖技术

6．在操作系统中，用户在使用 I/O 设备时，通常采用（　　）。

A．物理设备名　　B．逻辑设备名

C．虚拟设备名　　D．设备牌号

7．为了使多个进程能有效地同时处理输入和输出，最好使用（　　）结构的缓冲技术。

A．缓冲区　　B．闭缓冲区环

C．单缓冲　　D．双缓冲区

8．利用虚拟设备达到输入输出要求的技术是（　　）。

A．利用外存作为缓冲，将作业与外存交换信息和外存与物理设备交换信息两者独立起来，并使它们并行工作的过程

B．把 I/O 要求交给多个物理设备分散完成的过程

C．把 I/O 信息先放在外存，然后由一台物理设备分批完成 I/O 要求的过程

D．把共享设备改为某作业的独享设备，集中完成 I/O 要求的过程

9．主机与输入、输出设备之间进行数据交换的方式包括程序控制方式、程序中断方式和直接存储器存取方式等。在程序控制方式中，对于输出过程，准备就绪指的是（　　）。

A．输出缓冲器已空　　B．输出缓冲器已有数据

C．输出设备已等待工作　　D．输出设备正在工作

10．为了提高设备分配的灵活性，用户申请设备时应指定（　　）号。

A．设备类相对　　B．设备类绝对　　C．相对　　D．绝对

二、简答题

1．设备可分为哪几种类型？
2．数据传送有哪几种方式？
3．为什么要引入缓冲技术？常用的缓冲技术有哪些？
4．什么是独享设备？什么是共享设备？什么是虚拟设备？
5．对独享设备、共享设备、虚拟设备分别是采用什么分配方式？
6．提高设备与设备、设备与 CPU 并行性的途径有哪些？
7．假脱机系统由哪些部分组成？其特点有哪些？
8．设备驱动程序的处理过程是怎样的？
9．中断处理程序的处理过程是怎样的？