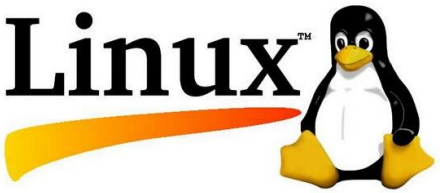
操作系统



中国矿业大学计算机学院

4

设备□管理

目 录

CONTENTS

4.1 设备管理概述

计算机系统中有各种设备，通常情况下除了CPU和内存以外的所有其他硬件资源统称为计算机外部设备，简称设备。

设备管理主要指除CPU和内存之外的设备的分配、控制、管理和回收。

4.1.1 设备分类

按服务功能分类

（1）存储类设备。通常以存贮大量信息和快速检索为目标，也称外存或后备存储器、辅助存储器，是计算机系统用以存储信息的主要设备，如U盘、光盘等。

（2）输入/输出类设备。主要完成把外界信息输入计算机，或者把运算结果从计算机输出的功能，例如键盘、显示器、打印机、音响、摄像头、扫描仪。

（3）通信类设备。这类设备主要完成计算机和外界的通信过程，如网卡、红外设备、蓝牙设备等。

4.1.1 设备分类

按每次信息交换的单位分类

（1）字符设备。指以单个字符为单位来传送信息的设备，如字符显示终端、键盘、打印机、异步通讯接口等。

（2）块设备。指以数据块为单位来组织和传送数据的设备，属于有结构设备，如磁盘、摄像头等。磁盘输入输出以一个扇区为基本单位,摄像头(或图形屏幕)抓取(或显示)以一帧为单位。

4.1.1 设备分类

按使用特征分类

（1）独占设备。这类设备在用户作业的整个运行期间必须为此用户所独占，才能保证传送信息的连贯性。独占性是设备本身的属性。

（2）共享设备。这类设备通常指磁带、磁盘一类的存取设备。这里的共享是指多个用户进程运行期间可以交替地使用它们，对它们进行读写。

（3）虚拟设备。为了将慢速的独占设备改造成多个用户可共享的设备，以提高设备的利用率、提高系统进程并行的程度，可借助于假脱机技术（Simultaneously Peripheral Operation On Line，SPOOLing）进行模拟。模拟独占设备的那部分共享设备的空间称为虚拟设备。

4.1.2 设备管理的目标、功能和结构

设备管理的目标

（1）提高使用效率。要尽量提高CPU和外设之间以及外设与外设之间的并行度，均衡系统中各设备的负载，最大限度地发挥所有设备的潜力，以使操作系统获得最佳效率。

（2）提供便捷的界面。所谓便捷，一方面是指用户能独立于具体设备的复杂物理特性而方便地使用设备；另一方面是指对多种不同设备尽量能有统一的操作方式。

4.1.2 设备管理的目标、功能和结构

设备管理的功能

（1）设备的分配与回收。在多道程序环境下，多个用户或进程往往同时要求使用同一类或同一台设备。操作系统一方面根据进程的请求分配设备，另一方面在进程使用设备结束后，回收设备，以便重新分配。

（2）缓冲区管理。为了实现低速的输入/输出设备与高速处理器之间的协调工作，一般都在内存中开辟一块存储区作为设立缓冲区，使CPU和设备通过缓冲区传送数据，从而使设备与设备之间、设备与CPU之间的工作协调起来。

4.1.2 设备管理的目标、功能和结构

设备管理的功能（二）

（3）设备控制和中断处理。根据用户提出的I/O要求，在未设置通道的系统中，由设备管理软件对设备I/O请求做必要的处理。

（4）实现虚拟设备。为了实现多进程并发对独占设备的需求，设备管理实现了虚拟设备功能，将一条独占的物理设备变为多个逻辑设备，从而能够接受多个进程对设备的请求。

4.1.2 设备管理的目标、功能和结构

设备管理的结构

设备管理的结构描述了设备、设备控制器、设备驱动程序与应用程序之间的逻辑关系。

设备一般包括其机械部件和电子部件。为了设计和制造的模块性和通用性，一般设备将这两部分分开。电子部件称为设备控制器或适配器，机械部件则为物理设备，图**4.1**描述了设备控制器和设备的分离。

4.1.2 设备管理的目标、功能和结构

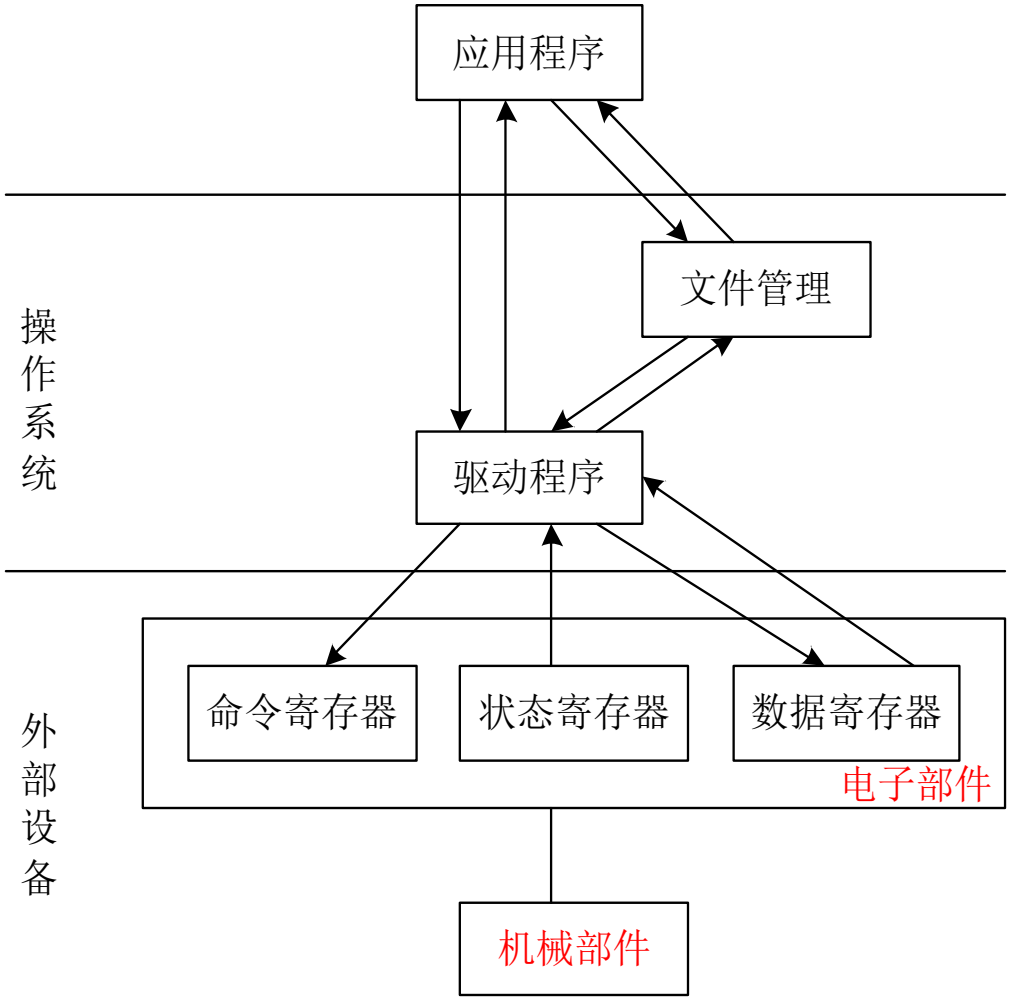


图4.1设备管理的逻辑结构

4.1.2 设备管理的目标、功能和结构

设备控制器

设备控制器起着承上启下的作用，是物理硬件和逻辑软件的桥梁，主要由三个部分构成。

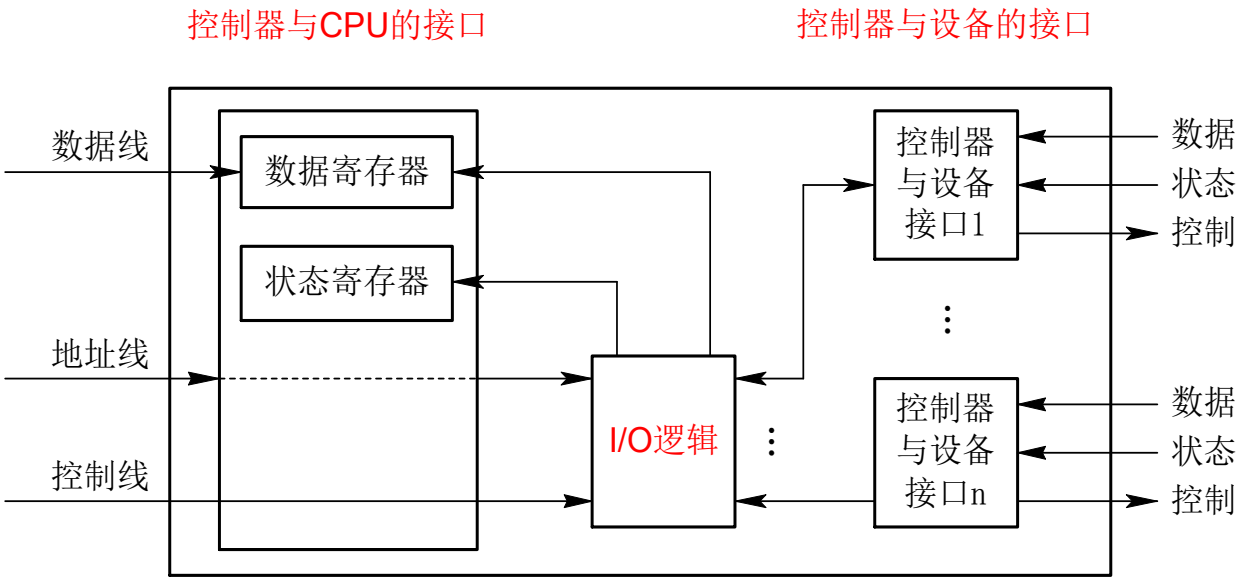


图4.2 设备控制器的构成

4.1.2 设备管理的目标、功能和结构

设备控制器三部分的功能

控制器与CPU的接口，主要用于通过数据线、地址线、控制线实现设备控制器与**CPU**之间的通信。

控制器与设备的接口。在一个控制器中有一个或多个设备接口，一个接口连接一台设备，在每个接口中都有数据、控制和状态三种类型的信号。

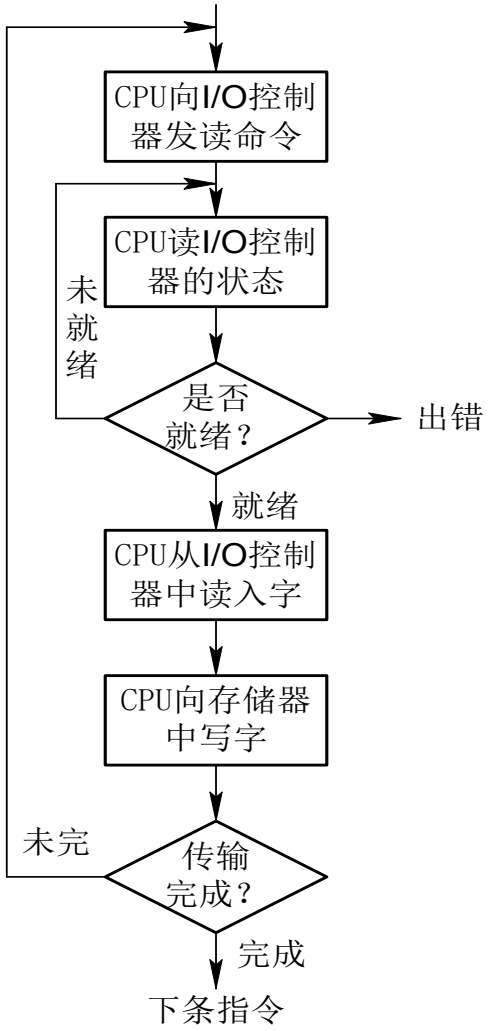
I/O逻辑。主要用于对**I/O**的控制。

4.2 设备控制方法

按照I/O 控制器与CPU 之间联系方式的不同，可把I/O 设备的控制方式分为四类：查询方式、中断方式、DMA方式和通道方式。

它们的主要差别在于中央处理器和外围设备并行工作的方式不同，并行工作的程度不同。中央处理器和外围设备并行工作有重要意义，它能大幅度提高计算机效率和系统资源的利用率。

4.2.1 程序循环查询方式

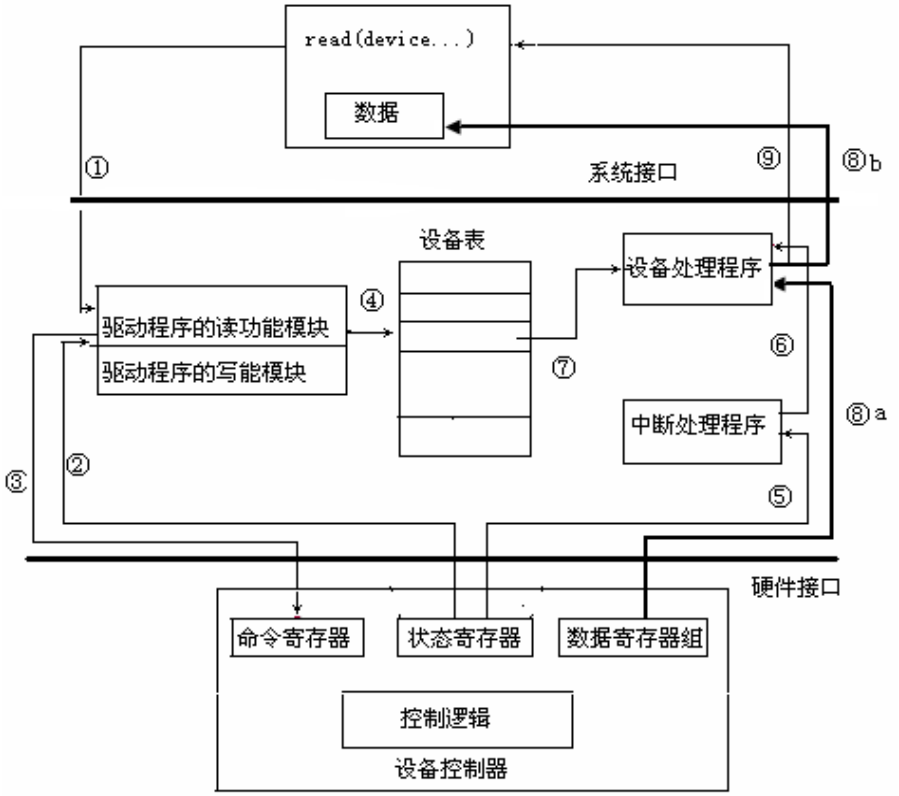


CPU要不断的发送I/O测试指令用以测试设备控制器的忙/闲（**busy**）标志位，若设备不忙，则主存与外部设备交换一个字符或一个字。若设备忙，则**I/O**测试指令不断对它进行测试，直到设备空闲为止。

由于**CPU**的速度远高于**I/O**设备的速度，使得**CPU**绝大部分时间都处于等待**I/O**完成的循环测试之中。显然，这对**CPU**是极大的浪费。但是，它的控制简单，在**CPU**速度慢、要求不高的场合下常被采用。

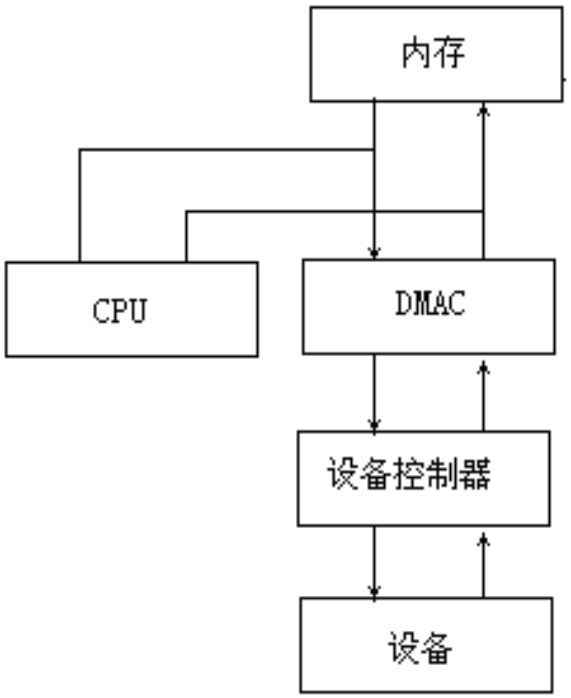
4.2.2 中断驱动方式

使用中断机制的系统中，由硬盘读入到内存的请求执行步骤示例见图**4.4**。



引入中断机构是为了消除设备驱动程序不断地轮询控制器状态寄存器的开销，当**I/O**操作结束后，由设备控制器“自动地”通知设备驱动程序。

4.2.3 直接内存访问方式



目前个人计算机块设备传输系统中，都普遍采用了DMA（Direct Memory Access）直接内存存取方式，通过DMA控制器DMAC控制从内存向设备输入输出,其原理如图4.5所示。

4.2.3 直接内存访问方式

三个特点

（1）数据传输的基本单位是数据块，即每次传送至少一个数据块。

（2）所传送的数据是从设备直接送入内存，或者直接读出内存的。

（3）在传输时CPU参与更少，仅在传送一个或多个数据块的开始和结束时，才需CPU干预，整块数据的传送是在控制器的控制下完成的。

4.2.4 通道方式

I/O通道方式是DMA方式的发展，它可进一步减少CPU的干预，即把对一个数据块的读(或写)为单位的干预，减少为对一组数据块的读(或写)及有关的控制和管理为单位的干预。

有专门用于I/O的处理单元。在进行I/O操作时，接受CPU的委托，独立地执行自己的通道程序来实现内存与外设之间的数据传输。

使CPU从对I/O设备的繁忙直接控制中解脱出来，极大地提高了CPU与外设并行工作的程度，从而更有效地提高整个系统的资源利用率。

4.2.4 通道方式

通道分类

（1）字节多路通道

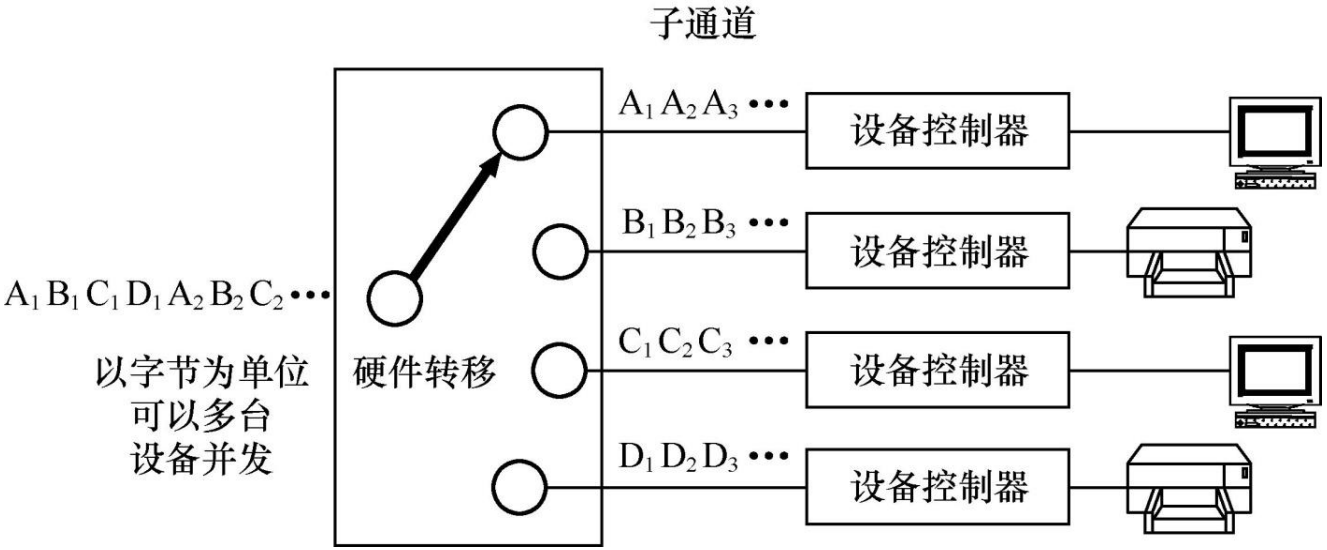
每个通道可以连接8、16甚至更多的子通道。子通道所连接的I/O设备以字节为单位经通道与内存交换数据。在一个子通道传送一个字节后，立即让位于另一个子通道传送下一个字节。

（2）选择通道

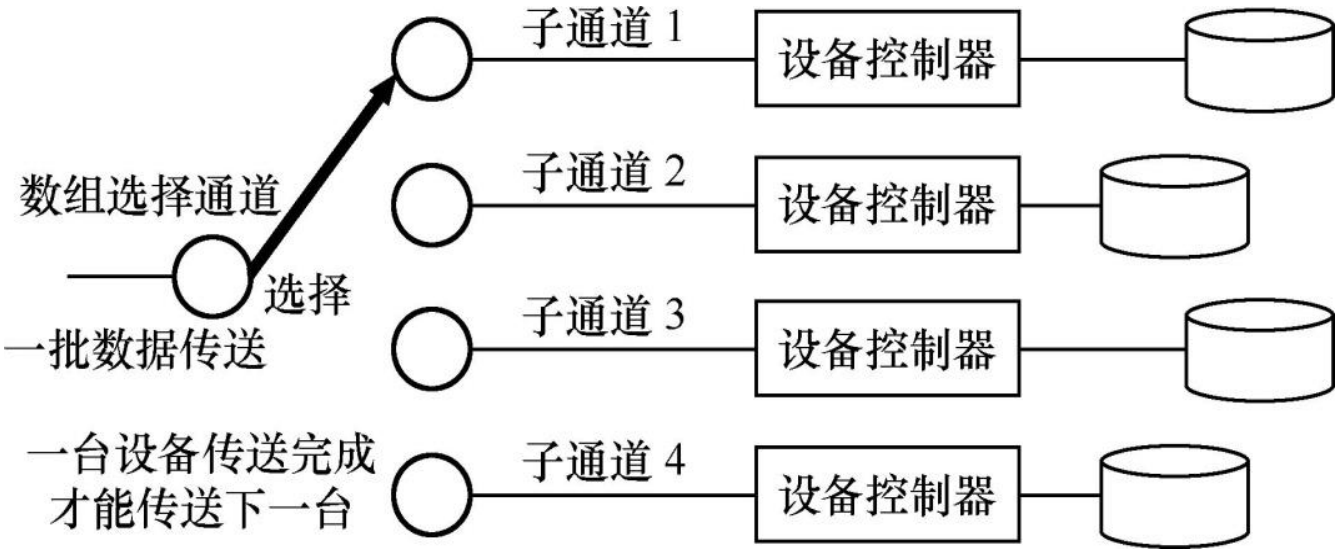
选择通道每次传送一批数据，传送速率可很高。由于选择通道只有一个分配型子通道，虽然这个子通道可以连接多台设备，但每次只能把子通道分配给一台设备使用。

（3）成组多路通道

它具有多个非分配型子通道，每个子通道连接一台中、高速I/O设备，因而通道所连接的几个设备可并行工作，而且每台设备的数据传送都是按成组方式进行的。

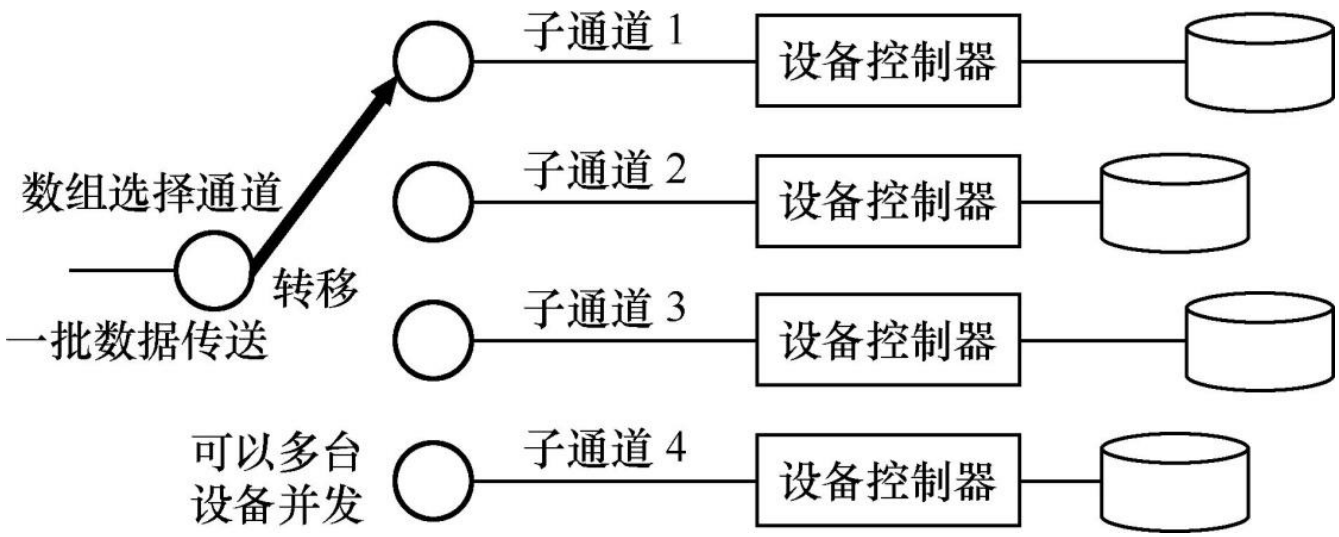

子通道
$A_1 B_1 C_1 D_1 A_2 B_2 C_2 \cdots$
以字节为单位
可以多台
设备并发
硬件转移
$A_1 A_2 A_3 \cdots$
$B_1 B_2 B_3 \cdots$
$C_1 C_2 C_3 \cdots$
$D_1 D_2 D_3 \cdots$
设备控制器
设备控制器
设备控制器
设备控制器

字节多路通道


子通道 1
设备控制器
子通道 2
设备控制器
数组选择通道
选择
子通道 3
设备控制器
一批数据传送
一台设备传送完成
才能传送下一台
子通道 4
设备控制器

选择通道

- 成组多路通道:结合了选择通道传送速度高和字节多路通道能进行分时并发传送多个设备数据的优点。一台设备执行一条通道指令，一次输送一批数据，多个通道程序并发，通道之间自动转接。如图所示。



成组多路通道

4.2.4 通道方式

通道工作方式

通道I/O操作由两种命令实现控制：CPU的I/O指令和通道本身提供的通道程序。

CPU的I/O指令的功能一般包括有：清除、启动、停止、查询等功能，除了操作码之外，I/O指令中还有通道地址和设备地址域。I/O指令属特权指令，只能由操作系统使用。

通道程序一般有读、写、查询、控制和转移等功能。

在通道I/O工作过程中，CPU对通道的通信是向通道发出启动、查询和停止通道指令；而通道向CPU的通信则采用中断方式向CPU汇报。

4.2.4 通道方式

通道工作方式

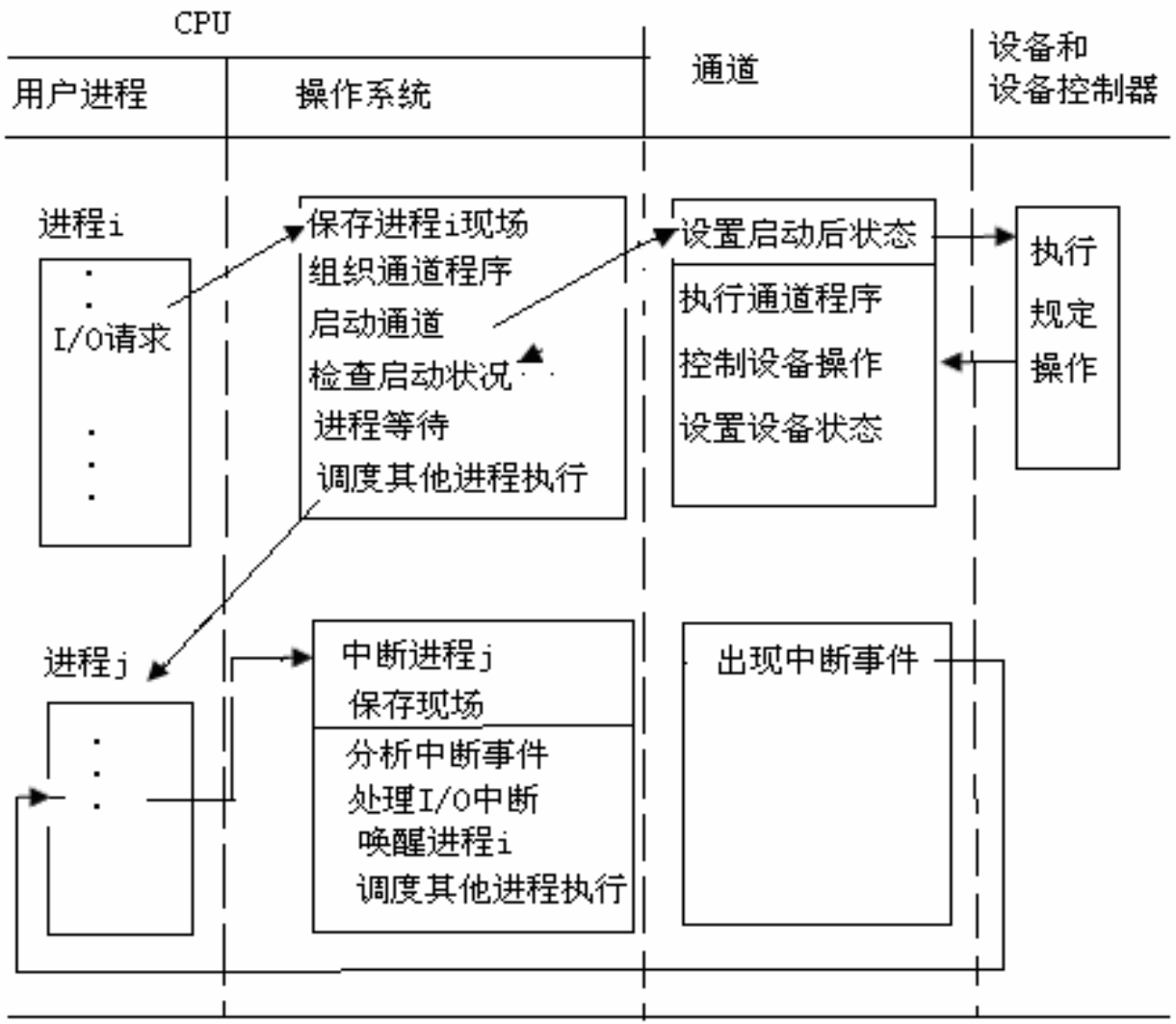


图4.6 通道与CPU之间通信示意图

4.3 缓冲技术

缓冲技术的作用

（1）它能改善中央处理器与外围设备之间速度不匹配的矛盾，提高CPU 和I/O 设备的并行性。

（2）它能减少I/O 对CPU 的中断次数和放宽对CPU 中断响应时间的要求。

（3）缓冲技术还能协调逻辑记录大小与物理记录大小不一致的问题。

4.3 缓冲技术

缓冲的分类

根据缓冲性质划分，缓冲有硬件缓冲和软件缓冲之分。

硬件缓冲是以专用的寄存器作为缓冲器。软缓冲缓冲即在操作系统的管理下，在内存中划出若干个单元作为缓冲区。软件缓冲的好处是易于改变缓冲区的大小和数量，缺点是占据一部分内存空间。

根据缓冲区个数的多少和结构，缓冲可分为：单缓冲、双缓冲、多缓冲、循环缓冲与缓冲池。

4.3.1 单缓冲

单缓冲指当一个进程发出I/O请求时，操作系统便在主存中为之分配一个缓冲区，用来临时存放输入／输出数据。它是操作系统提供的一种简单的缓冲形式。

缓冲的分类

在数据的输入或者过程中，在某一时刻该缓冲区只能存放输入数据或输出数据，而不能既是输入数据又是输出数据，否则会引起缓冲区中数据的混乱。对缓冲区来说，信息的输入和输出实际上是串行工作的。

4.3.1 单缓冲

无缓冲与单缓冲比较

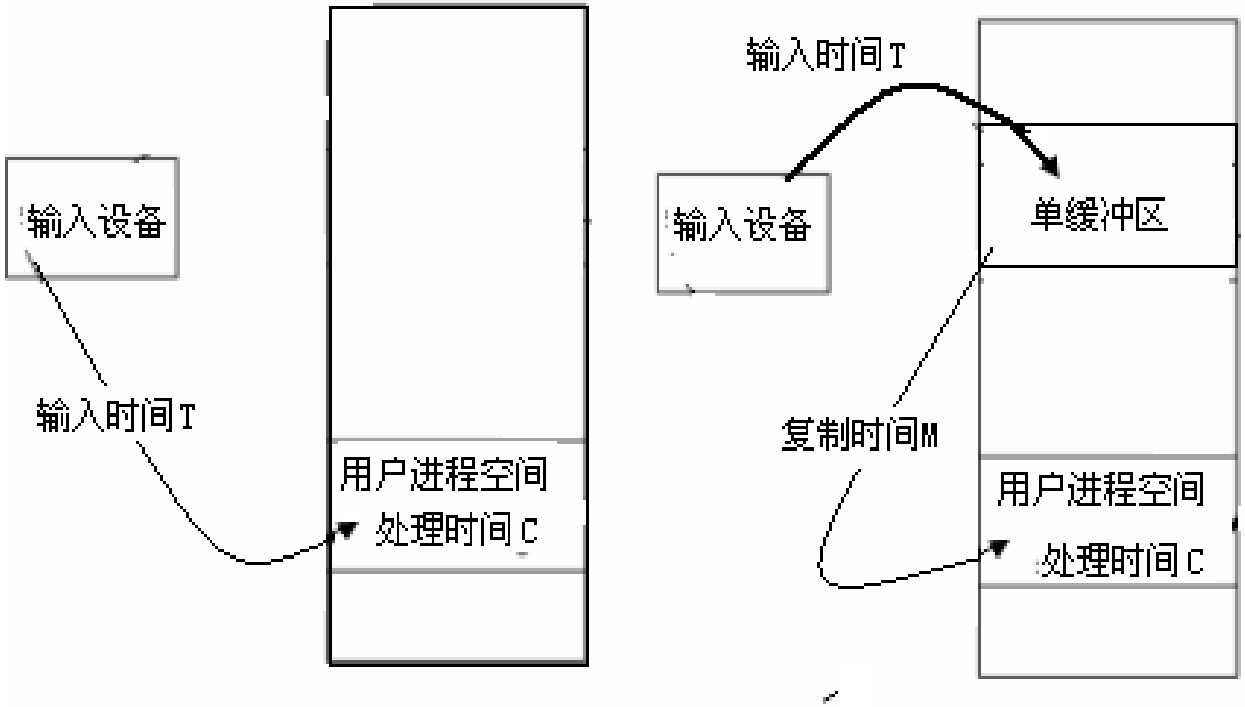
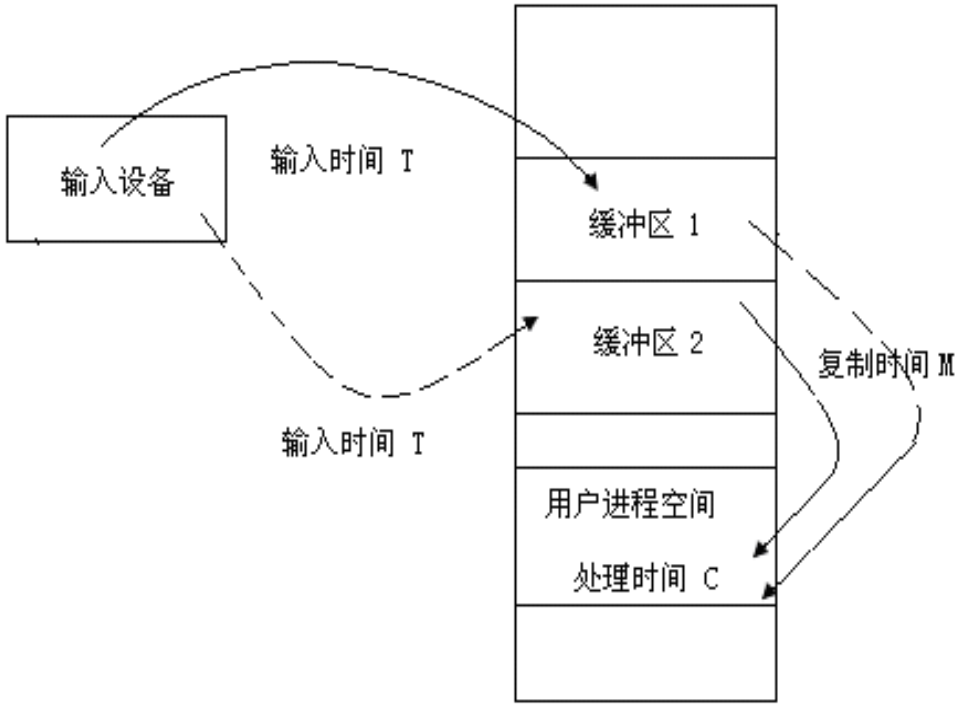


图4.7　无缓冲区(左)与单缓冲区输入(右)

4.3.2 双缓冲



双缓冲指在操作系统中为某一设备设置两个缓冲区，当一个缓冲区中的数据尚未被处理时可使用另一个缓冲区存放从设备读入或读出的数据，以此来进一步提高**CPU**和外设的并行程度。

4.3.2 双缓冲

双缓冲的工作原理

在输入数据时，首先填满缓冲区1，操作系统可从缓冲区1把数据送到用户进程区，用户进程便可对数据进行加工计算；与此同时，输入设备填充缓冲区2。

当缓冲区1空出后，输入设备再次向缓冲区1输入。操作系统又可以把缓冲区2的数据传送到用户进程区，用户进程开始加工缓冲区2的数据。

这样两个缓冲区交替使用，使CPU和I/O设备的并行性进一步提高。仅当两个缓冲区都空或者都满，进程还要提取数据或者写入数据时，它才被迫等待。

4.3.3 多缓冲

一、循环缓冲

为了使缓冲区资源得到充分利用，操作系统从主存区域中分配一组缓冲区，如图**4.10**所示，根据用途分为：输入缓冲区组和输出缓冲区组。

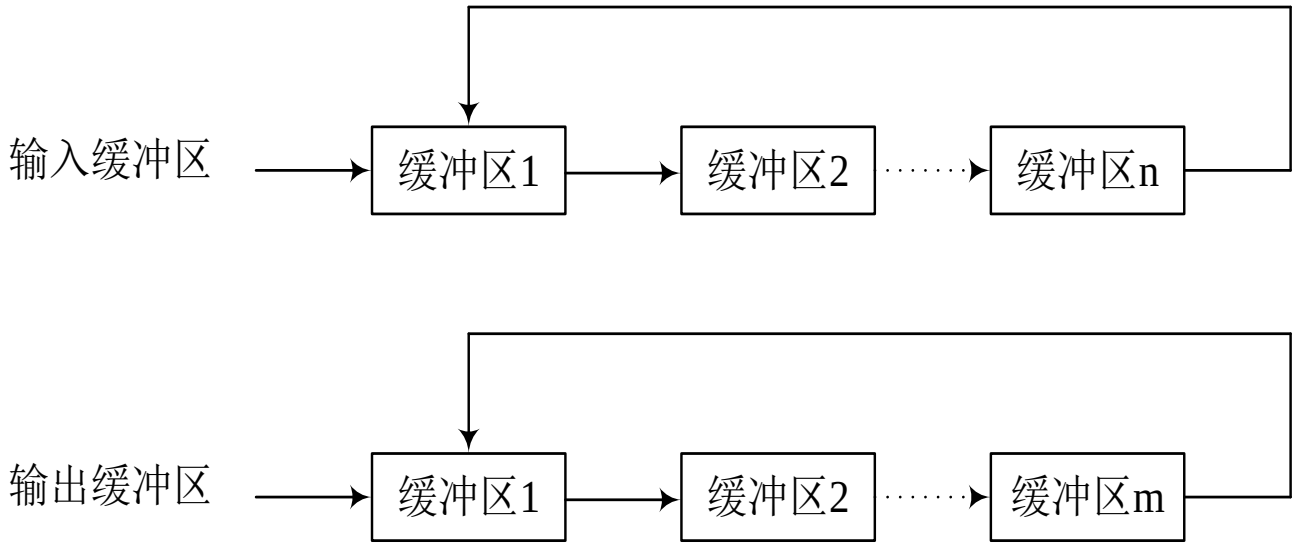


图4. 10 循环缓冲

4.3.3 多缓冲

一、循环缓冲

每个缓冲区有一个链接指针指向下一个缓冲区，最后一个缓冲区指针指向第一个缓冲区，组成了循环缓冲。每个缓冲区的大小可以等于物理记录的大小。

这种循环式的多缓冲区是系统的公共资源，并由系统统一分配和管理，可供各个进程共享。为了管理各缓冲区，进行统一调度和管理，操作系统中通常要设立专门的机构来管理它们。

4.3.1 多缓冲

二、缓冲池

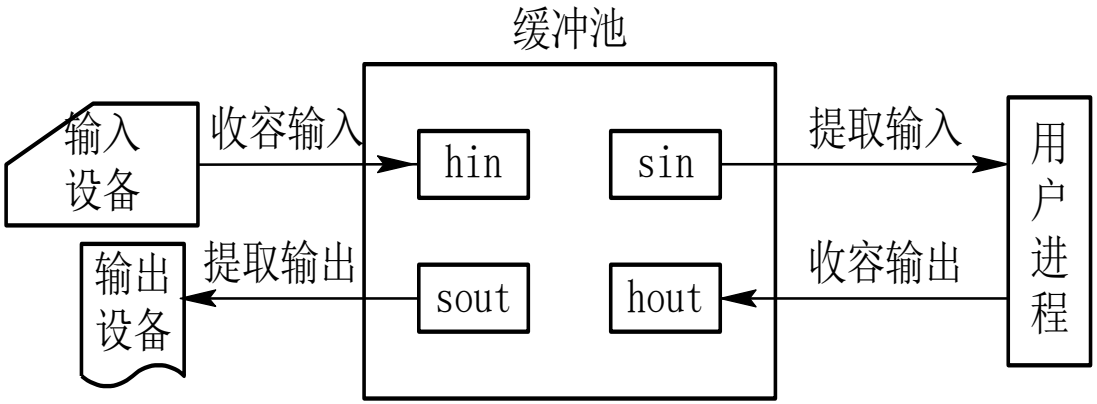
缓冲池由内存中的一组缓冲区构成,各缓冲区之间并不一定采用循环链表的方式进行链接，操作系统与用户进程将轮流地使用各个缓冲区，以改善系统性能。

缓冲池中多个缓冲区可供多个进程使用，既可用于输出又可用于输入，是一种现代操作系统经常采用的一种公用缓冲技术。

4.3.3 多缓冲

缓冲池中的缓冲区一般按照内容被组织成三个队列：空闲缓冲区队列**emq**、输入缓冲区队列**inq**、输出缓冲区队列**outq**。

每个缓冲区根据其当前的工作性质不同，分为四种状态：收容输入、提取输入、收容输出、提取输出。



对缓冲池的管理，有两个基本操作**Getbuf过程**和**Putbuf过程**。**Getbuf（type）**用于从**type**所指定的队列的队首摘下一个缓冲区；**Putbuf（type，number）**用于将由参数**number**所指示的缓冲区挂在**type**队列上。

4.4 输入输出软件

输入输出软件是实现输入输出管理的软件部分，其目标是改善输入输出设备的效率，实现统一标准的输入输出设备管理方式。

这些软件通常被组织成一种层次结构，自底向上分别为中断处理程序、设备驱动程序、设备独立性程序、用户层软件。

处于底层的软件用来屏蔽输入／输出硬件的细节，从而实现上层的设备无关性，高层软件则主要为用户提供一个统一、规范、方便的接口。

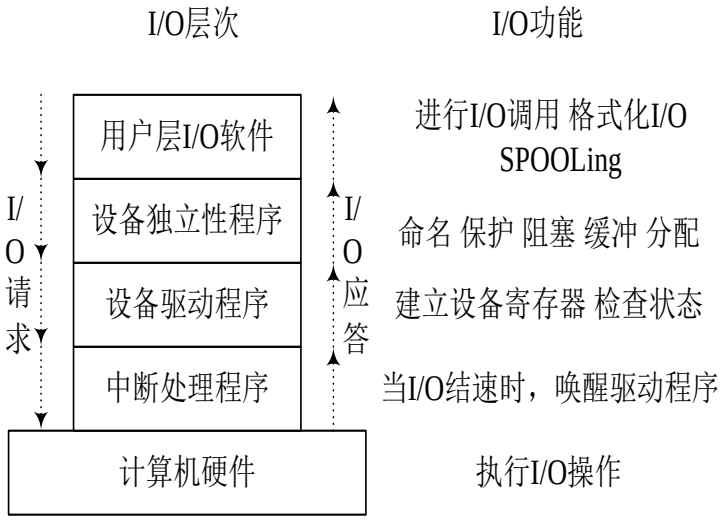


图4.12 I/O软件的层次

4.4.1 中断处理程序

凡是涉及到输入/输出设备开始、结束或者异常时，一般都会发生中断信号。

当一个中断信号发生时，若能及时得到相应，则正在运行的进程将被挂起，直到I/O 操作结束并发出一个中断请求，CPU响应后便转向中断处理程序，然后解除相应进程的阻塞状态。

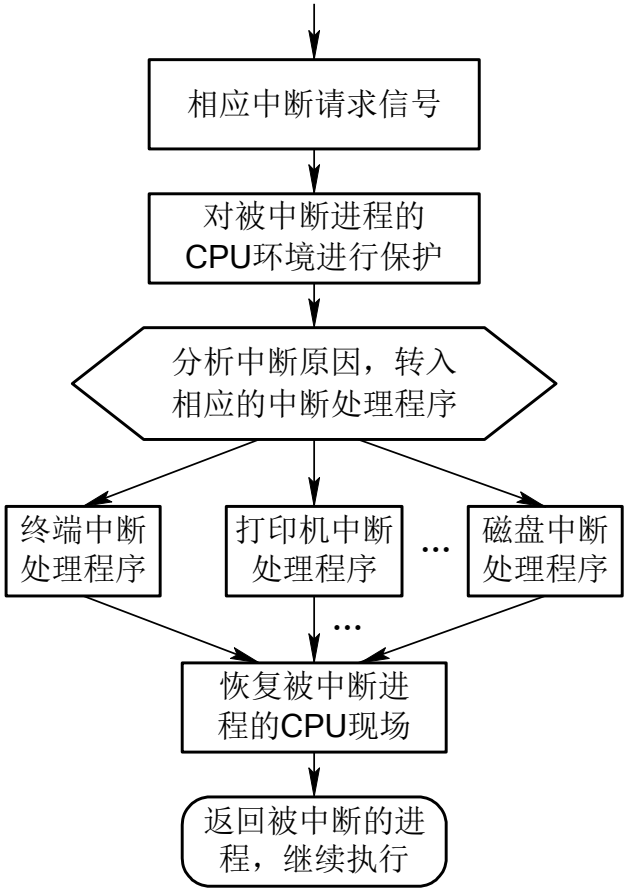


图4.13 中断处理流程

4.4.2 设备驱动程序

所谓设备驱动程序是指驱动物理设备和DMA控制器或I/O控制等直接进行I/O操作的程序集合。

不同类型的设备应有不同的设备驱动程序，设备驱动程序主要负责启动指定设备，即负责设置与相关设备有关的寄存器的值，启动设备进行I/O操作，指定操作的类型和数据流向等。

4.4.2 设备驱动程序

设备驱动程序的处理过程

(1) 接收由**I/O**进程发来的命令和参数，并将命令中的抽象要求转换为具体要求。

(2) 检查用户**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，如果设备空闲，便立即启动**I/O**设备去完成指定的**I/O**操作；如果设备处于忙碌状态，则将请求者的请求挂在设备队列上等待。

(4) 及时响应由控制器或通道发来的中断请求，并根据其中断类型调用相应的中断处理程序进行处理。对于设置有通道的计算机系统，驱动程序还应能够根据用户的**I/O**请求，自动地构成通道程序。

(5) I/O完成后，由通道（或设备）产生中断信号。**CPU**接到中断请求后，如果条件符合（中断优先级高于运行程序的优先级），则响应中断，然后转去执行相应的中断处理程序，唤醒因等待**I/O**完成而睡眠的进程，调度用户进程继续运行。

4.4.2 设备驱动程序

与一般的操作系统程序和应用程序不同，设备驱动程序由于与物理设备有关，具有以下特性：

设备驱动程序的特点

(1) 驱动程序指在请求**I/O**的进程与设备控制器之间的一个通信和转换程序。它的主要任务是接受上层软件发来的抽象要求并将设备控制器发来的信号传送给上层。

(2) 驱动程序与设备控制器和**I/O**设备的硬件特性紧密相关，因而对不同类型的设备应配置不同的驱动程序。

(3) 驱动程序与**I/O**设备所采用的**I/O**控制方式紧密相关。

4.4.3 设备独立性程序

所谓设备独立性，也称为设备无关性，是指在用户程序中不要直接使用物理设备名（或设备的物理地址），而只能使用逻辑设备名。

逻辑设备是实际物理设备属性的抽象，它不限于某类具体设备。逻辑设备究竟和哪一个具体的物理设备相对应，还要由系统根据当时的设备忙、闲情况来决定或由系统管理员指定。

4.4.3 设备独立性程序

在应用程序中使用逻辑设备名来请求使用某类设备，而系统在实际执行时，使用物理设备名。

当然系统必须根据如图**4.14**所示的对应关系将逻辑设备名转换成物理设备名，进一步找到物理设备的驱动程序地址。

逻辑设备名	物理设备名	驱动程序入口地址
/dev/tty	3	1024
/dev/printer	5	
⋮	⋮	

逻辑设备名	系统设备表指针
/dev/tty	3

(a

硬　　件	设备文件名
IDE硬盘	/dev/hd[a-d]
SCSI/SATA/USB硬盘	/dev/sd[a-p]
光驱	/dev/cdrom或/dev/hdc
软盘	/dev/fd[0-1]
打印机（25针）	/dev/lp[0-2]
打印机（USB）	/dev/usb/lp[0-15]
鼠标	/dev/mouse

图4

4.4.3 设备独立性程序

引入设备独立性这一概念，使得用户程序可使用逻辑设备名，而不必使用物理设备名，有以下优点：

(1) 使得设备分配更加灵活，提高了设备的利用率。当多用户多进程请求分配设备时，系统可根据设备当时的忙闲情况，合理调整逻辑设备名与物理设备名之间的对应情况，以保证设备的充分使用。

(2) 可以实现**I/O**重定向。所谓**I/O**重定向是指可以更换**I/O**操作的设备而不必改变应用程序。

4.4.3 设备独立性程序

设备驱动程序是一个与硬件（或设备）紧密相关的软件，为了实现设备独立性，就必须在驱动程序之上设置一层与设备无关的软件，其主要功能如下：

(1) 向用户层软件提供统一接口。

(2) 设备无关程序负责将设备名映射到相应的设备驱动程序。

(3) 设备保护。

(4)协调不同设备数据块的差异。

(5)差错控制。

4.4.4 用户层软件

用户层的**I/O**软件实际上就是面向设备具体应用的软件，它是**I/O**系统软件的最上层软件。

它面向程序员，当接收到用户的I/O指令后，把具体的请求发送到设备无关的I/O软件，进行进一步的处理。

针对I/O设备的用户层软件主要包含用于I/O操作的库函数和和前文提到的SPOOLing假脱机系统。

4.5 设备分配与回收

设备分配和回收的任务是按照一定的算法将设备有关资源分配给申请进程，在进程使用完毕后还要负责收回相关设备资源。

4.5.1 设备信息描述

设备信息描述表格

- 系统设备表SDT（System Device Table）
- 设备控制表DCT（Device Control Table）
- 控制器控制表COCT（Controller Control Table）
- 通道控制表CHCT（Channel Control Table）

4.5.1 设备信息描述

系统设备表SDT

整个系统中设置惟一的**系统设备表SDT**，记录了系统中全部设备情况，并为每个设备设置了一个表项。

SDT的每个表项主要包括：设备类型、设备标识符、指向DCT的指针以及驱动程序入口。

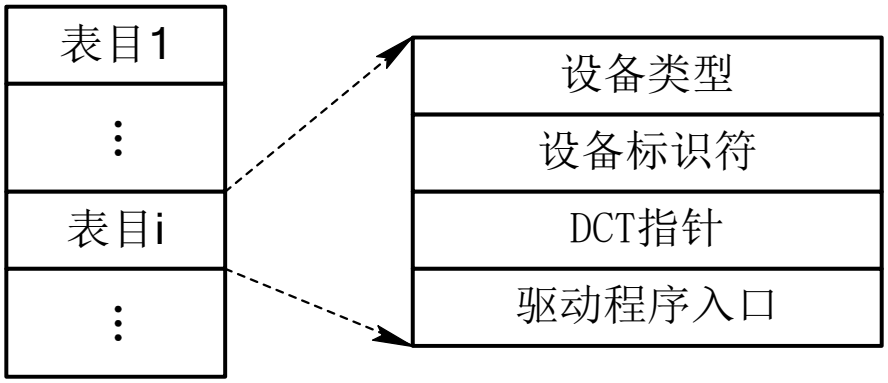


图4.15 系统设备表

4.5.1 设备信息描述

设备控制表DCT

设备控制表DCT： 用来记录系统内任一台设备的特性、设备和I/O控制器的连接情况以及设备的分配和使用情况。

DCT在系统生成时或在该设备和系统连接时创建，但表中的内容则可根据系统执行情况动态修改。

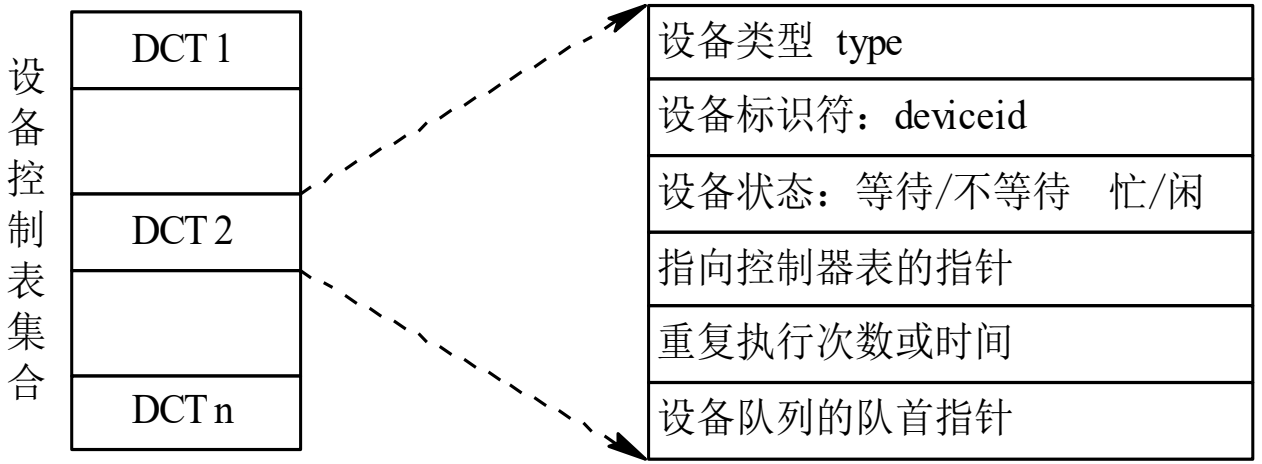


图4.16 设备控制表

4.5.1 设备信息描述

控制器控制表COCT

系统为每个控制器都设置了一个COCT，记录I/O控制器的使用情况以及所连接的通道情况。

主要表项包括：控制器标识符、控制器的状态、与控制器相连接的通道表指针、控制器队列的队首指针、控制器队列的队尾指针，各相应项意义与DCT类似。

4.5.2 设备分配策略

按照设备本身使用属性，设备分成独占设备、共享设备和虚拟设备三类，相应的设备分配策略就叫独占方式、共享方式和虚拟方式。

独占方式就是把一台设备固定地分配给一个用户或进程，直到它运行结束。这种方式对用户来说是方便的，管理起来也简单，但资源往往造成浪费。因为用户程序或进程运行过程中不会自始至终都使用像打印机这类设备。

共享方式是指设备可以在多个用户（或进程）“交替”使用，即一个进程需要时，便申请它，获得后使用它，用完就释放它。其他进程也如此方式使用。磁盘、磁带就是可共享方式分配、使用的设备。

4.5.2 设备分配策略

虚拟方式：为了提高独占设备的利用率，提高进程并行程度，引入了虚拟设备技术。虚拟设备技术就是利用快速、共享设备（例如磁盘）把慢速、独占设备模拟成为同类物理设备。

例如，一个用户需要一台打印机，这时系统分配给它的不是一台物理打印机，而是一磁盘文件，用户进程输出，向该磁盘文件输出，用户进程结束，系统把该磁盘文件传输给打印机进程，依次排队，打印输出。从用户看来，每个用户都感到是系统为自己提供了一台物理打印机。

计算机的屏幕也是独占设备，图形界面的窗口是虚拟屏幕，使用窗口可以使不同应用程序共享屏幕。

4.5.3 SPOOLing技术

早期，为了缓和CPU的高速性与I/O设备的低速性之间的矛盾而引入了脱机输入输出技术，该技术是利用专门的外围控制机，实现低速I/O设备与高速磁盘的数据传输。

当系统中引入了多道程序技术后，可以利用其中的一道程序，来模拟脱机输入输出时的外围控制机功能，把低速I/O设备上的数据传送到高速磁盘上；或者把数据从磁盘传送到低速输出设备上。

这样，便可在主机的直接控制下，实现脱机输入输出功能。此时的外围操作与CPU对数据的处理同时进行，这种在联机情况下实现的同时外围操作称为**SPOOLing**，或称为假脱机系统。

4.5.3 SPOOLing技术

SPOOLing系统的组成

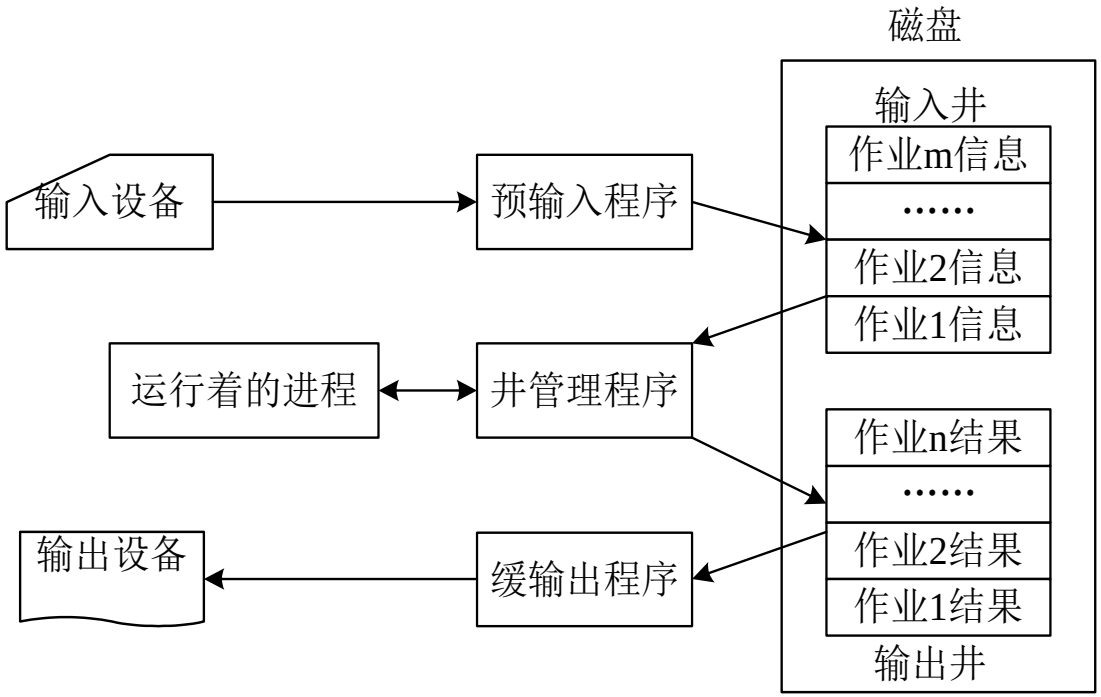


图4.19 SPOOLing系统结构

输入井模拟假脱机输入，用于收容输入的数据。

输出井模拟脱机输出，用于收容用户程序的输出数据。

预输入程序模拟脱机输入时的外围控制机，将输入设备的输入信息送到输入井，当相关进程需要输入数据时，直接从输入井读入到内存中的用户程序区。

4.5.3 SPOOLing技术

SPOOLing系统的组成

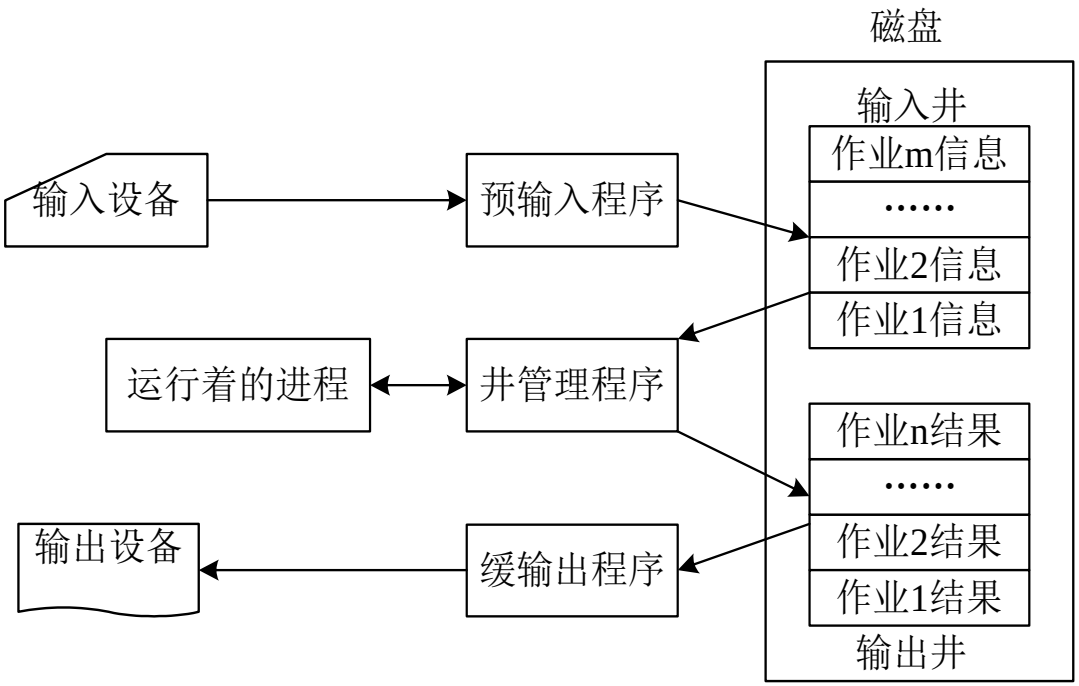


图4.19 SPOOLing系统结构

缓输出程序模拟脱机输出时的外围控制机，把用户要求输出的信息从用户程序区送到输出井，待输出设备空闲时，将输出井中的信息送到输出设备上。

井管理程序负责管理输入井与输出井的协调工作。

4.5.3 SPOOLing技术

共享打印机

创建一个特殊的守护进程以及一个特殊的目录——SPOOLing目录。

当用户进程请求打印输出时，SPOOLing系统同意为它打印输出，但并不真正立即把打印机分配给该用户进程，而只为它做两件事：

①由输出进程在输出井中为之申请一个空闲磁盘块区，并将要打印的数据送入其中。

② 输出进程再为用户进程申请一张空白的用户请求打印表，并将用户的打印要求填入其中， 再将该表挂到请求打印队列上。

4.5.3 SPOOLing技术

共享打印机

实际上，当进程把该文件放到SPOOLing系统中之后就可以认为打印过程已经完成，虽然打印机还没有进行该文件的打印，因此SPOOLing也称为打印的“假脱机”过程。

而整个的打印作业由该守护进程进行处理，只有该守护进程能够真正使用打印机设备文件。

4.5.3 SPOOLing技术

SPOOLing技术的特点

（1）提高了I/O的速度，缓和了高速的处理器与低速输入输出设备之间的矛盾。

（2）将独占设备改造为共享设备，提高了设备的利用率。

（3）实现了虚拟设备功能，将物理的单个设备变换为多个对应的逻辑设备。

4.5.4 设备分配算法

（1） 先来先服务算法。系统允许多个进程请求同一个设备，也允许一个进程请求多个设备。系统按照进程对某设备提出I/0请求的先后顺序，将对应进程组织成一个设备请求队列。当设备空闲时，设备分配程序总是把此设备首先分配给队首进程。

（2） 优先级高者优先算法。对有I/O的进程按照其优先级的高低进行排列，当有一个新进程要加入设备请求队列中时，根据进程的优先级插在适当的位置。保证在该设备空闲时，系统能从I/O请求队列的队首取下一个具有最高优先级进程，并将设备分配给它。

4.5.5 设备分配与回收过程

设备分配过程是逐步由抽象信息表格到具体物理设备的过程，首先使用了管理逻辑设备信息的系统设备表，继而查找设备控制表，然后通过设备控制器和通道启动设备，传输信息。

4.5.5 设备分配与回收过程

4

设备分配

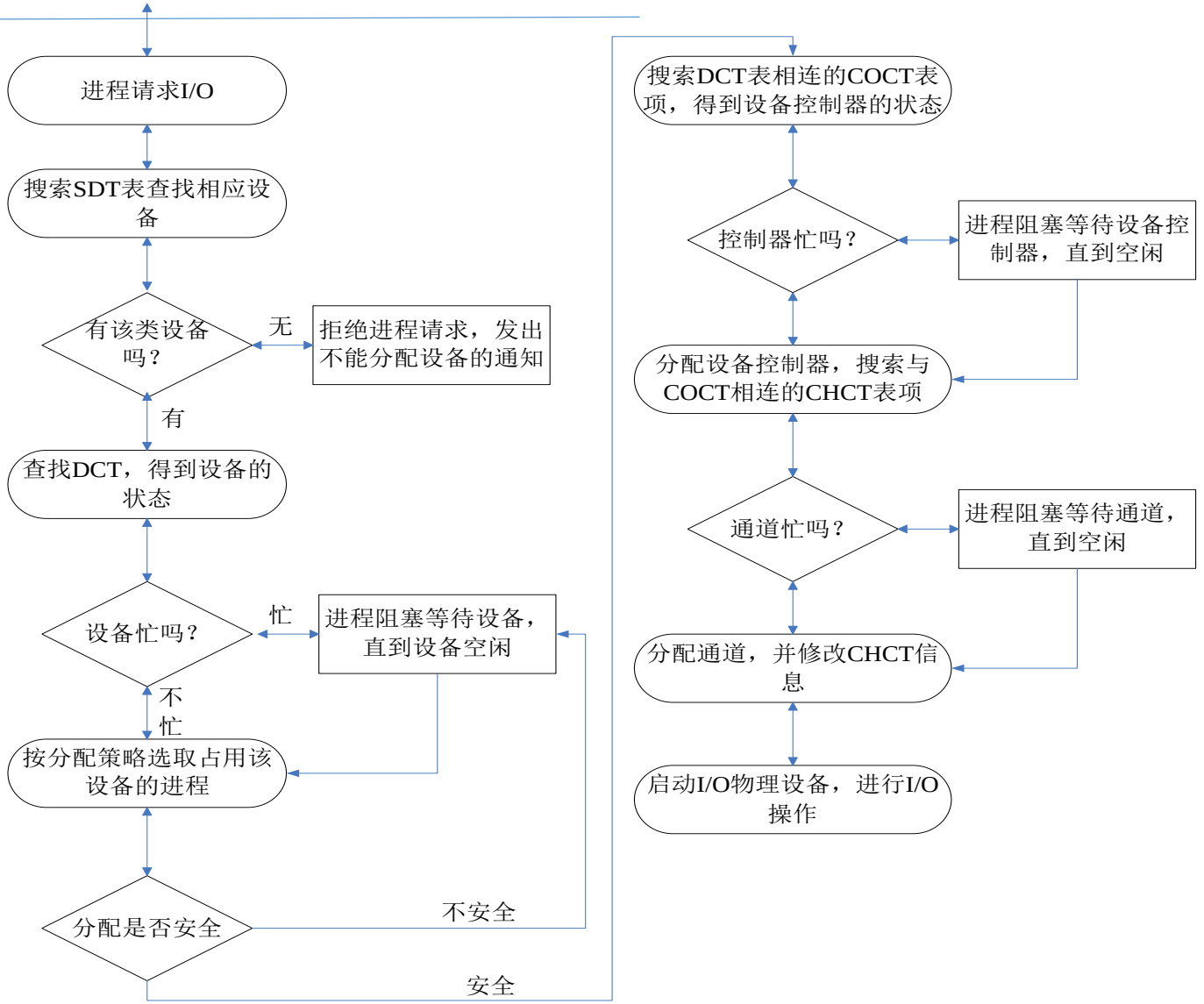


图4.20 设备分配过程

4.5.5 设备分配与回收过程

设备回收

当某一作业（或进程）使用完设备后，则需“释放设备”，设备回收过程是设备分配过程的逆过程。

回收时，要请求操作系统依次修改与设备有关的通道控制表、设备控制器控制表、设备控制表和系统设备表，主要是修改其中的状态信息，以使设备能够及时被下一个进程使用。

4.6 Windows 7中的I/O设备管理

在输入输出设备管理方面，Windows系列操作系统的基本原理上与其他操作系统是相通的。

但在实现技术和细节上，Windows系列操作系统有着自身的特点。从windows 2000、XP直到windwos 7都在不断地优化和改进。

4.6.1 WindowsI/O系统软件层次结构

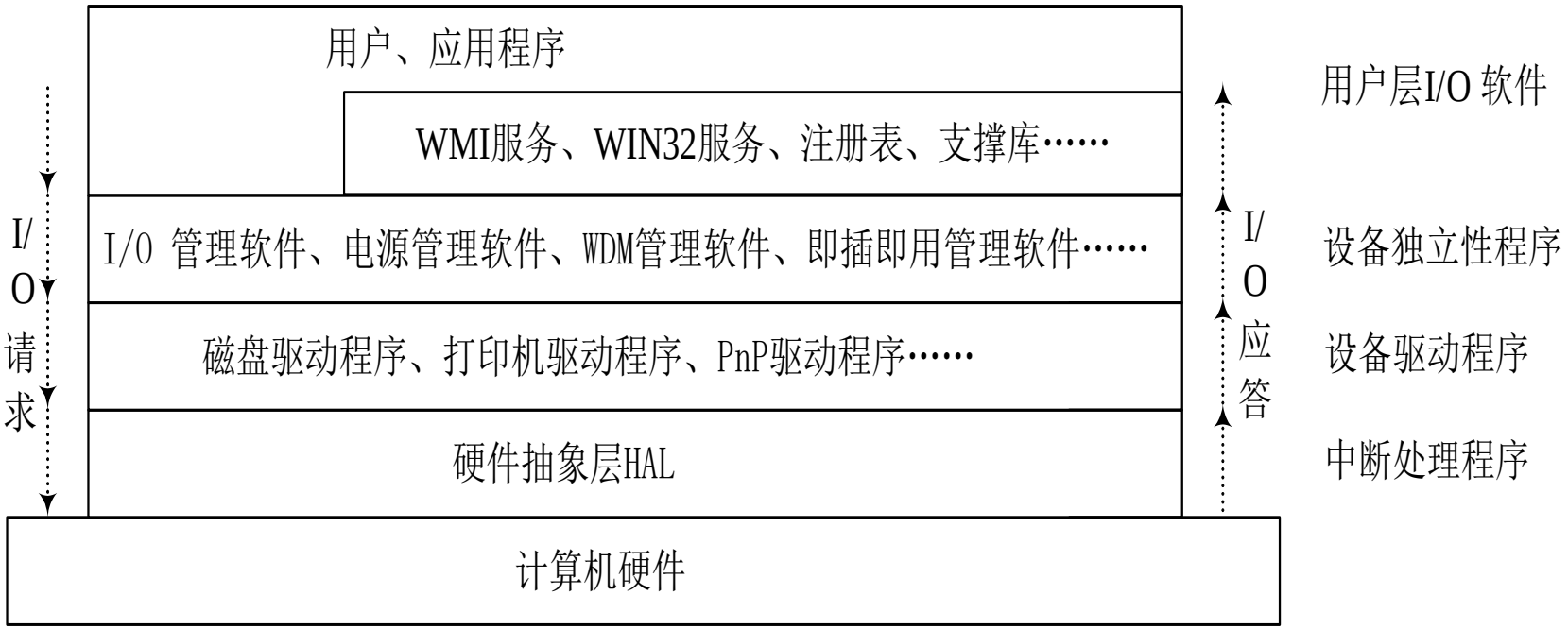


图4.21 Windows I/O系统层次

4.6.1 WindowsI/O系统软件层次结构

硬件抽象层（HAL:Hardware Abstract Layer）实际上中断处理程序，它把上层的各种驱动程序与下层的多种硬件设施隔离开来，以确保在Windows支持的硬件体系结构的计算机中能够移植。

磁盘驱动程序、打印机驱动程序、PnP驱动程序等所在层次是设备驱动程序层，主要是为对应的设备提供一个I/O接口，结合具体的硬件完成设备独立性程序与中断处理程序之间的信息传递和设备驱动工作。

4.6.1 WindowsI/O系统软件层次结构

I/O管理软件把用户层应用程序或者系统组件连接到各种虚拟的、逻辑的或者物理的设备上，并且在逻辑上定义了一个适用于设备驱动程序的基本框架。

电源管理软件通过与I/O管理软件的协同工作来检测整个系统或者某个硬件设备，完成不能电源状态的转换。

WDM管理软件，是Windows驱动程序模型（WDM: Windows Driver Model）的缩写，提供WMI（Windows Management Instrumention）服务的支持程序，允许下层的驱动程序使用这些程序与用户层的WMI服务通信。

即插即用（PnP: Plug and Play）管理软件通过与I/O管理软件、相关驱动程序的协同工作来检测硬件设备的添加、删除或者相应设备的硬件资源分配情况。

4.6.1 WindowsI/O系统软件层次结构

在用户层的I/O软件由两个层面来构成。

一部分是用户程序。

另外一部分是可以向最终用户或应用程序提供二次开发功能的相应服务或软件，比如**WMI**服务、**WIN32**服务、注册表以及各种支撑库等。

4.6.2 I/O系统的数据结构

文件对象

在Windows XP/7中，文件作为对象来处理，它是两个或多个用户态线程可共享的系统资源，文件对象作为设备资源的管理方式，有时称为虚拟文件，是指用于I/O的所有源或目标，它们都被当做文件来处理(例如文件、目录、管道和邮箱)。

所有被读取或写入的数据都可以看作是直接读写到这些虚拟文件的流。它提供基于主存储器的共享物理资源的表示。系统中所有的I/O 操作都通过虚拟文件执行，隐藏了I/O 操作中具体硬件设备的实现细节，为应用程序提供了一个统一的接口。

4.6.2 I/O系统的数据结构

驱动程序对象

在进行I/O操作时，当线程首先打开文件对象对应的句柄，I/O管理软件为为文件对象选择合适的驱动程序对象。驱动程序对象在系统中代表一个独立的驱动程序，并且为I/O记录每个驱动程序的调度例程的入口。

当驱动程序被加载到系统中时，系统将创建一个驱动程序对象，然后它调用驱动程序的初始化例程，该例程把驱动程序的入口填放到该驱动程序对象中。初始化例程还创建用于每个设备的设备对象，这样就使设备对象脱离了驱动程序对象。

4.6.2 I/O系统的数据结构

设备对象

设备对象在系统中代表一个物理的、逻辑的或虚拟的设备并描述了它的特征。

设备对象以**DEVICE_OBJECT**结构来描述，它是驱动程序对象数据结构中的一个成员。

4.6.2 I/O系统的数据结构

I/O请求包

I/O请求包（IRP:I/O Request Package）是I/O系统用来存储处理I/O请求所需信息的数据结构。当有进程或者线程调用I/O服务时，I/O管理软件就构造一个IRP来表示在整个系统I/O进展中要进行的操作。

IRP 由两部分组成：固定部分(也称作标题)和若干个堆栈单元。固定部分信息包括请求的类型和数据大小、指向缓冲区的指针和相关状态信息等等。IRP 的堆栈单元包括功能码、与功能相关的参数和指向调用者文件对象的指针。

4.6.3 I/O的处理过程

I/O类型

(1)同步I/O与异步I/O

同步I/O：当调用者启动I/O设备之后，等待设备执行数据传输并在I/O完成时收到一个状态码，然后程序就可以访问被传输的数据。通常的I/O操作大都以同步方式实现。

异步I/O：允许应用程序发布I/O请求，在设备传输数据的同时，应用程序不必等待，可以继续执行其他工作。在传输期间调用者不能访问来自I/O的数据，当设备完成数据传输时，将有相应的信号状态产生。

4.6.3 I/O的处理过程

（2）快速I/O

快速I/O 是Windows提供的一种特殊输入输出机制，它允许I/O系统不产生IRP 而直接到文件系统驱动程序或高速缓存管理器去执行I/O 请求。这种机制为特定的环境下加快I/O过程而提供了可能。

(3)分散I/O与集中I/O

从虚拟内存的分散着多个缓冲区读取数据并写到磁盘上文件的一个集中的连续区域里，这种I/O过程，读取时是分散开的，而写时则是集中的，反之也可以。这种特殊类型的高性能I/O，它被称作“分散/集中”(scatter/gather)，可通过Win32 的ReadFileScatter 和WriteFileScatter 函数来实现。

(4) 映射文件I/O

映射文件I/O是指把磁盘中的文件视为进程的虚拟内存的一部分。程序可以把文件作为一个大的内存数组来访问，而无需做缓冲数据或执行磁盘I/O的工作，它需要借助虚拟内存管理功能，通过使用Win32 的CreateFileMapping和Map ViewOfFile 函数来实现。

4.6.3 I/O的处理过程

（1）I/O库函数经过语言的运行库时转换成对子系统DLL的调用，I/O管理软件根据调用命令生成一个I/O请求包，包括：指向文件对象的指针、所执行的操作和参数，初始化堆栈单元；

（2）子系统DLL 调用I/O 管理器的相关服务。

（3）I/O系统服务调用对象管理程序，检查给定的文件名，再搜索名字空间，把控制权交给I/O管理软件寻找文件对象。

（4）驱动程序询问安全子系统，确定线程的存取权限。如果不允许，就出错返回；否则，由对象管理器把所允许的存取权限和返回的文件句柄连在一起，返回用户态线程，之后线程用文件句柄对文件实施操作。

4.6.3 I/O的处理过程

（5）I/O管理软件以IRP的形式定位设备对象，找到设备驱动程序，将I/O请求传送给设备驱动程序，驱动程序启动设备，执行I/O操作。

（6）设备完成指定的操作，请求I/O中断，转入设备驱动程序的中断处理程序进行中断处理。

（7）I/O管理软件调用I/O完成中断处理，将完成状态返回给调用线程。

（8）对与异步I/O，I/O管理软件将控制权返回给调用线程，使得调用线程与I/O操作并行执行。

根据I/O请求所指向设备种类和进行I/O类型的不同，I/O处理的过程会有一定的差异性，但原理上是一致的。

4.6.4 Windows7 设备管理新特性

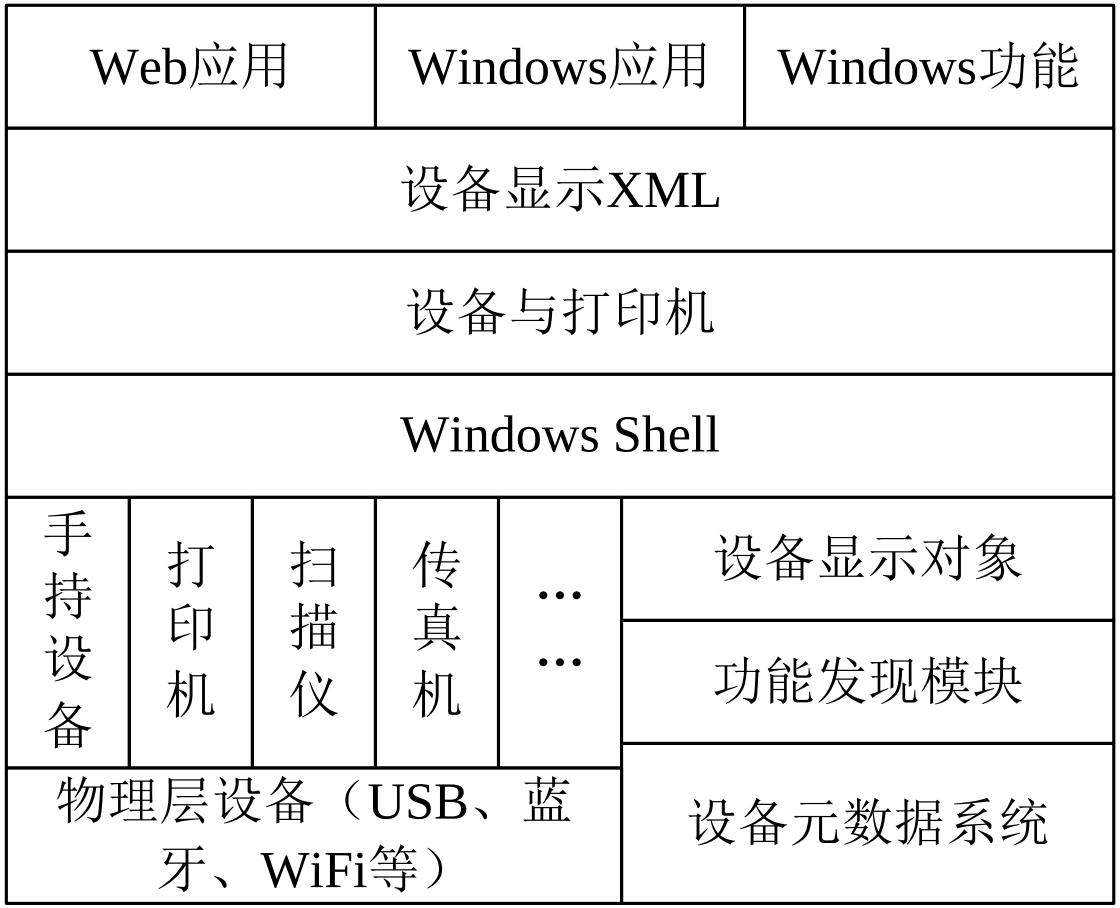


图4.23 Windows 7设备系统结构图

设备元数据系统、设备显示对象、设备与打印机、**Windows**功能模块是**Windows 7**新增加的模块。

4.6.4 Windows7 设备管理新特性

4

设备元数据系统

在**Windows 7** 中，设备元数据系统 **(Device Metadata System)** 为设备制造商定义和分发设备元数据包提供了一个端到端的流程，体现了“**Devices and Printers**”元数据包的资源流程。

设备元数据包包含表示设备属性及其功能的设备体验 **XML** 文档，以及支持该设备的应用程序和服务。

4.6.4 Windows7 设备管理新特性

Device Stage

Device Stage技术是**Windows 7**新增加的一个设备解决方案，它为用户提供了一种与 **Windows 7** 中的合格设备进行交互的新方法，可以将其看成是一个增强版的即插即用技术。

它包含一个直观的界面，用户可以很方便地查找和使用其设备的应用程序和服务。**Device Stage** 也为特定的合格设备提供了一个多功能版的 **AutoPlay**。

对于便携设备，**Device Stage** 通过新的设备服务支持行业标准的媒体传输协议 **(Media Transfer Protocol, MTP) 1.0**，这些新的设备服务在 **Windows 7** 中启用一组新功能。

4.6.4 Windows7 设备管理新特性

4

设备容器

“设备容器”是**Windows 7**中提出的一个概念，能有效、便捷地识别多功能设备。 可以将一个或多个设备功能分组到相应设备容器中，相当于把每个功能当作标签放入标签盒，这样更接近用户对物理设备及其功能的理解。

“设备容器”功能是借由新的随插即用特征“**ContainerID**”来达成，它是一个全局唯一识别码。

4.6.4 Windows7 设备管理新特性

4

设备容器

通过“设备容器”这一功能，使得**Windows 7**在设备管理方面具有以下优势：

(1)将设备功能按组归类到一个代表物理设备的容器

(2) 允许**Windows**决定各功能间如何关联

(3) 关联的功能可被呈现为一个设备

(4) 在以更自然的方式呈现设备的同时，保持了现有的设备节点模式

4 小结

小结

设备管理主要指对除CPU和内存之外的硬件资源的管理，有时称作输入输出设备管理。设备管理目的是尽量提高设备与设备、设备与CPU的并行性，提高系统的效率，使用户更加方便地使用设备。

常用的设备和内存之间的数据传送控制方式有四种：程序循环查询方式、中断驱动方式、直接内存访问方式和通道方式。

引入缓冲，包括单缓冲、双缓冲和多缓冲，能在一定程度上改善CPU和I/O设备之间速度不匹配，减少I/O对CPU的中断次数，缓冲还可以协调逻辑记录大小与物理记录大小不一致的问题。

4 小结

小结

输入输出过程中需要输入输出软件的参与，一般分为四层：用户层输入输出软件、设备独立性程序、设备驱动程序、中断处理程序。

设备分配与回收是设备管理中需要解决的问题，不仅要及时查找、更新相应数据结构中的参数信息，还要考虑设备的属性及分配是否安全，避免出现死锁现象。为了提高设备利用率，操作系统通常采用虚拟设备技术，**SPOOLing**系统就是典型的实例。

Windows 7作为新一代操作系统，采用了新一代**XML**架构，引入了元数据系统、**Devicestage**和设备容器等新技术，提供了更加人性化的设备管理界面。


THANKS
@CUMTIS