第6章 设备管理

由于设备种类繁多，设备特性和操作方式差异大，因此，设备管理是一项较繁琐和复杂的工作。

任务：设备无关性；优化I/O，实现最大并行性。

主要内容

6.1 设备管理概述

6.2 I/O系统

6.3 缓冲技术

6.4 独占设备的分配

6.5 磁盘管理

6.6 设备处理

6.7 虚拟设备

6.8 Linux设备管理

6.9 本章小结

本章要点

- 本章首先介绍设备管理的基本功能，介绍了I/O系统，缓冲技术，接着讨论独占型设备的分配和回收，并以磁盘为例讨论共享型设备的分配和回收。
- 本章重点掌握以下要点：
  - 了解设备管理的主要功能；I/O控制方式以及缓冲区技术；
  - 理解设备的分配和回收技术；
  - 掌握磁盘组织与管理，磁盘调度算法以及虚拟设备的实现思想和假脱机技术（SPOOLing）。

6.1 设备管理概述

- 现代计算机系统中配置了大量不同类型的外围设备，包括用于实现信息输入、输出和存储功能的设备以及相应的设备控制器，在有的大中型计算机中还设有输入/输出通道。在计算机系统中，通常把外围设备又称为I/O设备，这些设备的物理特性和操作方式有很大区别，在运行速度、控制方式、数据表示以及传送单位上存在着很大的差异。因此，计算机系统对外围设备的管理，是操作系统中最具有多样性和复杂性的部分。

6.1 设备管理概述

- **设备分类**
- 20世纪80年代以后，由于个人计算机、工作站以及计算机网络等的发展，外围设备开始走向多样化、复杂化和智能化。可以从不同角度对外部设备进行分类：
  - 按照工作特性可把它们分成存储设备和输入/输出设备两大类
  - 根据设备的使用性质可将设备分成独占设备、共享设备和虚拟设备三种
  - 按照数据传输的方式可将设备分为串行设备和并行设备

6.1 设备管理概述

- **设备的管理功能**
- 现代计算机系统要方便用户使用，为用户提供使用外围设备的统一界面、尽可能地提高输入/输出设备的使用效率，发挥系统的并行性。
  - ① 实现对外围设备的分配与去配
  - ② 实现外围设备的启动
  - ③ 实现对磁盘的驱动调度
  - ④ 实现设备处理
  - ⑤ 实现虚拟设备

6.2 I/O系统

- I/O设备及其接口线路、控制部件、通道以及管理软件统称为I/O系统。
- 主存与外围设备之间的信息传输操作，称为I/O操作。
- 多道程序设计技术引入后，I/O操作能力成为计算机系统综合处理能力及性能价格比的重要因素。

6.2.1 I/O系统结构

- 典型的输入/输出系统具有四级结构：主机、通道、设备控制器和输入/输出设备。

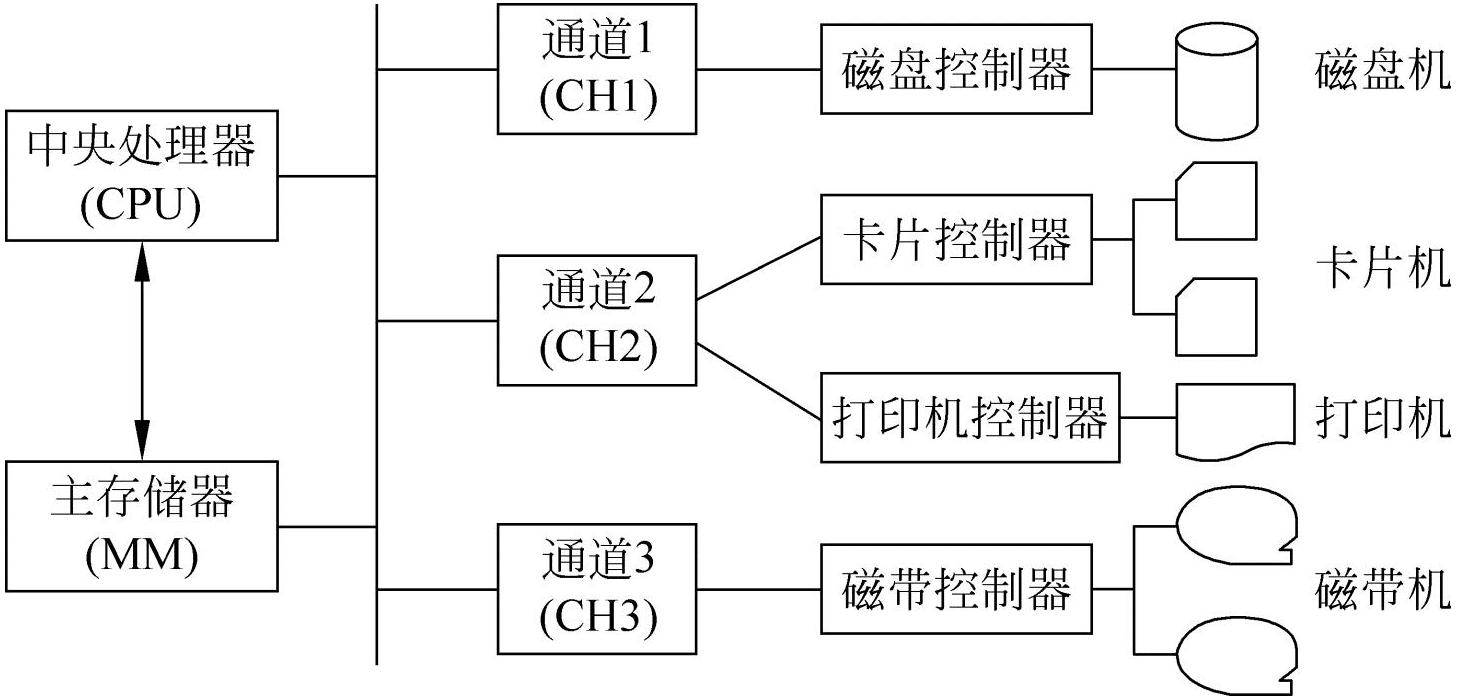


图6-1 输入/输出系统四级结构

6.2.1 I/O系统结构

1. I/O设备

- I/O设备的种类繁多，其重要性能指标有：数据传输单位、数据传输速率和设备的共享属性等。

- (1)按传输速率分类
  - 第一类是低速设备，其传输速率仅为每秒钟几个字节到数百个字节，如键盘、鼠标等设备；
  - 第二类是中速设备，其传输速率在每秒钟数千个字节到数万个字节，如行式打印机、激光打印机等；
  - 第三类是高速设备，其传输速率在每秒钟数十万个字节到数十兆字节，如磁带机、磁盘机、光盘机等。

6.2.1 I/O系统结构

(2)按信息交换的单位分类

- 第一类是块设备，以块为单位与主存交换信息，属于有结构设备，如磁盘(每个盘块的大小为0.54KB)、磁带等。
- 基本特征：传输速率较高，通常每秒钟为几兆位；
  可寻址，即允许对指定的块进行读/写操作；在I/O操作时常采用直接存储器访问(DMA)方式。
- 第二类是字符设备，以字符为单位与主存交换信息，属于无结构设备。字符设备种类繁多，如交互式终端、打印机等。
- 基本特征：传输速率较低；不可寻址；在I/O操作时，常采用中断驱动方式。

6.2.1 I/O系统结构

(3)按设备的共享属性分类

- 第一类是独占型设备，在一段时间内只能被一个作业独占使用，例如，输入机、磁带机和打印机等。独占型设备通常采用静态分配方式。
- 第二类是共享型设备，在一段时间内允许几个作业同时使用，例如，磁盘，对共享型设备允许多个作业同时使用，即一段时间内多个作业可以交替地启动共享设备，但在每一时刻仍只有一个作业占用。
- 第三类是虚拟设备，通过虚拟技术用共享型设备来模拟独占型设备的工作。

6.2.1 I/O系统结构

2. 设备控制器

- （1）接口线路
  - 通常，外围设备并不是直接与CPU进行通信，而是与设备控制器通信。在设备与设备控制器之间有一个接口，通过数据线、控制线和状态线传输数据、控制和状态三种类型信号。

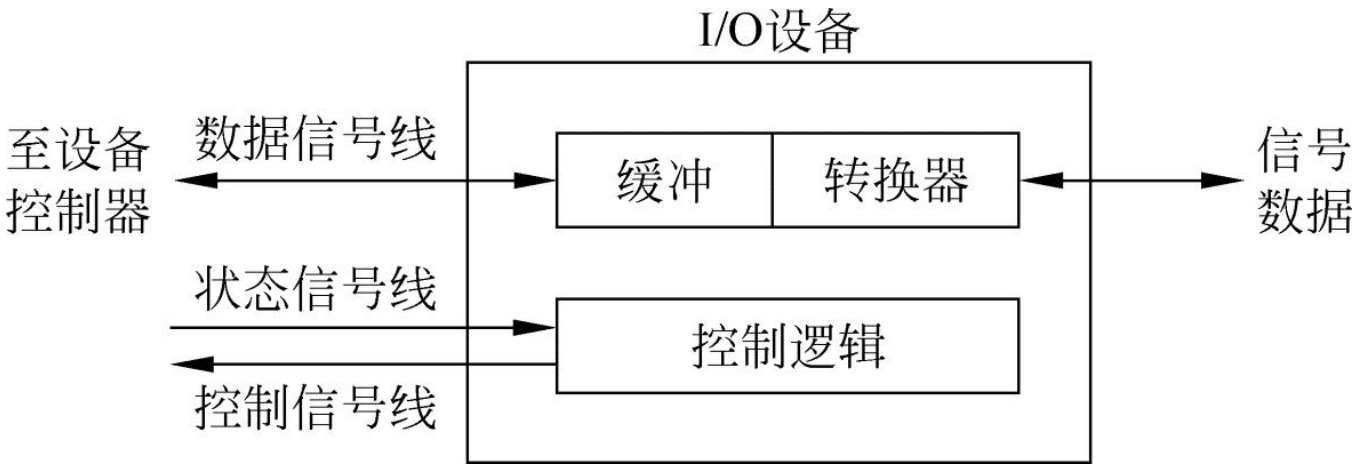


图6-2　设备与控制器间的接口

6.2.1 I/O系统结构

- 数据信号线
  - 数据信号线用于设备和设备控制器之间数据信号的传送。
- 控制信号线
  - 控制信号线作为设备控制器与I/O设备之间控制信号的传送通道。
- 状态信号线
  - 状态信号线用于传送指示设备当前状态的信号。设备的当前状态有正在读、正在写、设备已完成等。

6.2.1 I/O系统结构

- 设备控制器位于CPU与设备之间，控制一个或多个I/O设备，以实现I/O设备和主机之间的数据交换。设备控制器既要与CPU通信，又要与设备通信，由它接受从CPU发出的命令，并控制I/O设备的工作，是CPU与I/O设备之间的接口，能有效地将CPU从设备控制事务中解脱出来。

- 设备控制器分为两类：控制字符设备的控制器和控制块设备的控制器。设备控制器是一个可编址设备，它含有多少个设备地址，就可以连接多少个同类型设备，并且为它所控制的每一个设备分配了一个地址。

6.2.1 I/O系统结构

- （2）设备控制器的基本功能

① **接受和识别命令**：识别CPU向控制器发出的多种不同命令。

② **数据交换**：CPU与控制器、控制器与设备之间的数据交换。

③ **表示和报告设备的状态**：记录外围设备的工作状态。

④ **地址识别**：每个设备有一个唯一的地址。

⑤ **数据缓冲**：解决高速设备与低速设备之间不匹配的问题。

⑥ **差错控制**：对传送的数据进行差错检测。

6.2.1 I/O系统结构

- （3）设备控制器的组成
  - 设备控制器一般由**设备控制器与CPU接口**、**设备控制器与设备接口**以及**I/O逻辑**三部分组成。

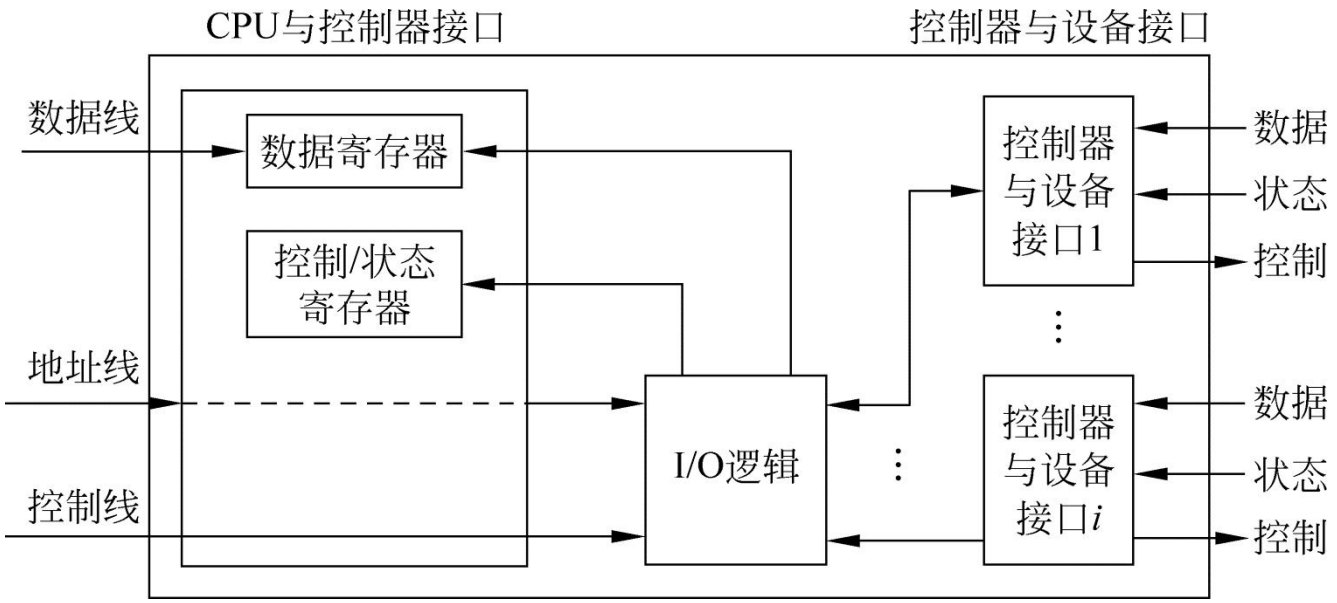


图6-3　设备控制器间的组成

6.2.1 I/O系统结构

① **设备控制器与CPU的接口**。该接口通过数据线、地址线和控制线实现CPU与设备控制器之间的通信。

② **设备控制器与设备的接口**。一个设备控制器可以有一个或多个设备接口，一个接口连接一台设备，在每个接口中都存在**数据**、**控制**和**状态**三种类型的信号。

③ **I/O逻辑**。设备控制器中的I/O逻辑用于实现对设备的控制。通过一组控制线与CPU交互，CPU利用该逻辑向控制器发出I/O命令；I/O逻辑对收到的命令进行译码。当CPU要启动一个设备时，一方面将启动命令发送给控制器；同时通过地址线把地址发送给控制器，由控制器的I/O逻辑对收到的地址进行译码，再根据所译出的命令对所选设备进行控制。

6.2.1 I/O系统结构

3. 通道

- 通道又称**输入/输出处理机**。它具有执行I/O指令的能力，并通过执行通道程序来控制I/O操作，完成主存储器和外围设备之间的信息传送。采用通道技术主要解决了输入/输出操作的独立性和各部件工作的并行性，实现了外围设备与CPU之间的并行操作，通道与通道之间的并行操作，各个通道上的外围设备之间的并行操作，提高了整个系统效率。
- **具有通道装置的计算机系统，主机、通道、设备控制器和设备之间采用四级连接，实施三级控制**。通常，一个中央处理器可以连接若干通道，一个通道可以连接若干个控制器，一个控制器可以连接若干台设备。

6.2.1 I/O系统结构

- 根据信息交换方式的不同，通道可分为三种类型：字节多路通道、数组选择通道和数组多路通道。
- （1）字节多路通道（Byte Multiplexor Channel）
  - 是一种按字节为单位以交叉方式工作的通道。它通常含有许多非分配型子通道，其数量可达数百个，每一个子通道连接一台I/O设备，并控制该设备的输入/输出操作，这些子通道按时间片轮转方式共享主通道。
  - 字节多路通道主要用于连接大量的低速外围设备。

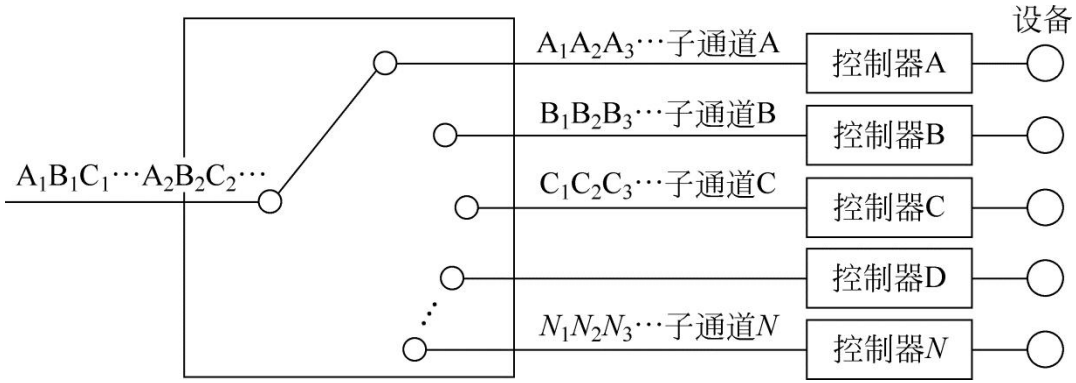


图6-4　字节多路通道的工作原理

6.2.1 I/O系统结构

- （2）数组选择通道（Block Selector Channel）
  - 数组选择通道以块为单位成批传送数据。它只含有一个分配型子通道，在一段时间内只能执行一道通道程序， 控制一台设备进行数据传送， 致使当某台设备占用该通道后，便一直独占使用， 即使无数据传送，通道被闲置， 也不允许其它设备使用该通道， 直至设备释放该通道。
  - 数组选择通道可以连接多台高速设备，每次传送一批数据，传送速度高，但通道的利用率很低，如磁带机、磁盘机等设备。

6.2.1 I/O系统结构

- （3）数组多路通道（Block Multiplexor Channel）
  - 数组多路通道含有多个非分配型子通道，以分时方式同时执行几道通道程序，因而数组多路通道既具有很高的数据传输速率，又能获得令人满意的通道利用率。数组多路通道的实质是对通道程序采用多道程序设计技术的硬件实现。

6.2.1 I/O系统结构

- 由于通道的成本高，在系统中通道数量有限，这往往成为I/O的瓶颈，造成整个系统的吞吐量降低。
  - 如图6-5所示单通路I/O系统，为了驱动设备1，必须连通控制器1和通道1，若通道1已被其他设备（如设备2，设备3或设备4）所占用或存在故障，则设备1无法启动，这就是由于通道不足而造成输入/输出操作中的“瓶颈”现象。

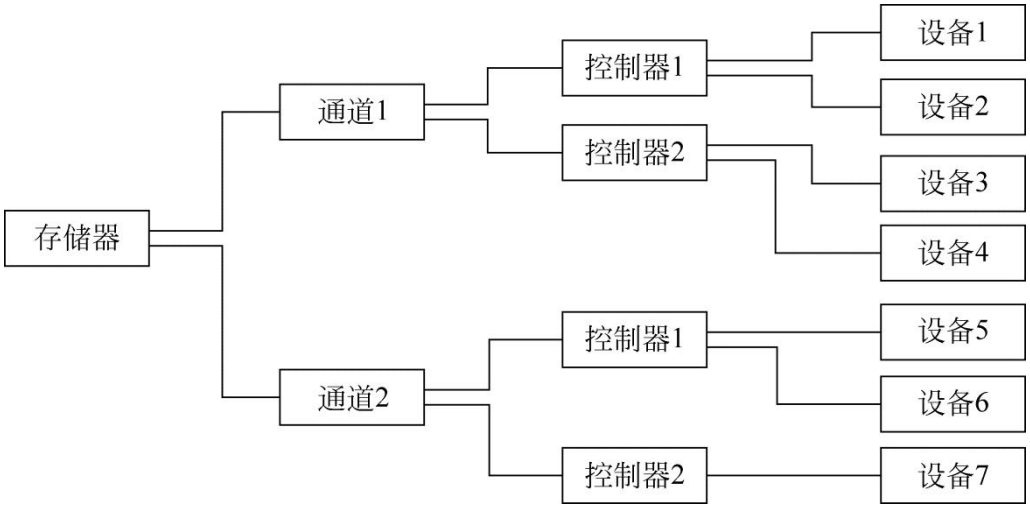


图6-5 单通路I/O系统

6.2.1 I/O系统结构

- 解决“瓶颈”问题的最有效办法，便是增加设备到主机之间的通路而不增加通道，即把一个设备连接到多个控制器上，而一个控制器又连接到多个通道上，实现多路交叉连接，即使个别通道或控制器出现故障时，也不会使设备和存储器之间没有通路。多通路方式不仅解决了“瓶颈”问题，而且提高了系统的可靠性。

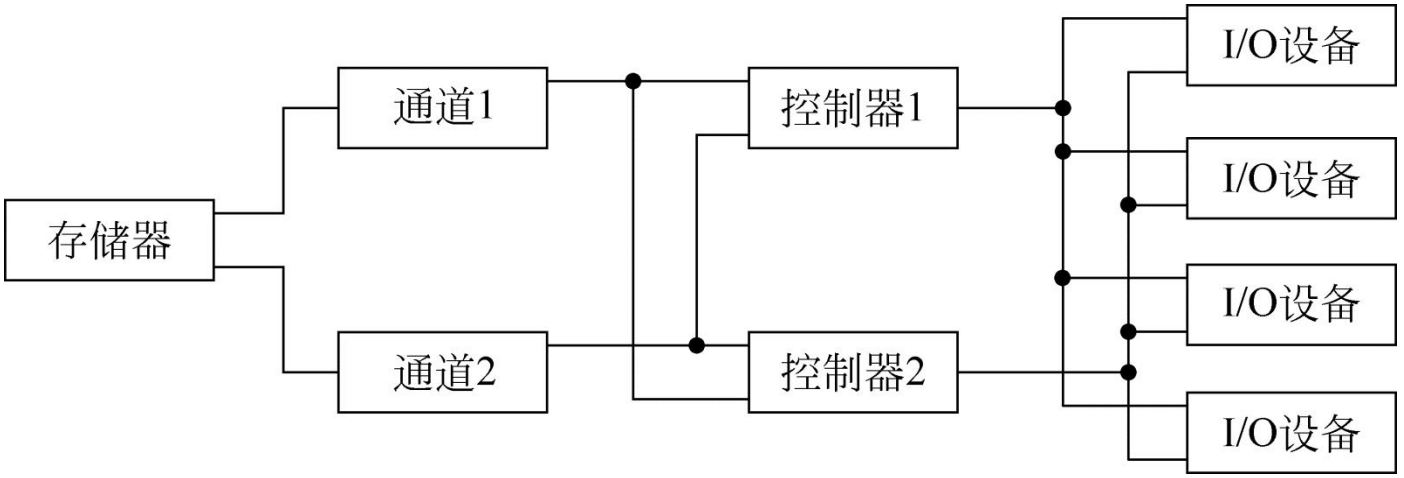


图6-6 多通路I/O系统

6.2.1 I/O系统结构

- 在一个计算机系统中，由于外围设备种类繁多，为了获得更高的输入/输出效率，可能同时存在多种类型的通道。

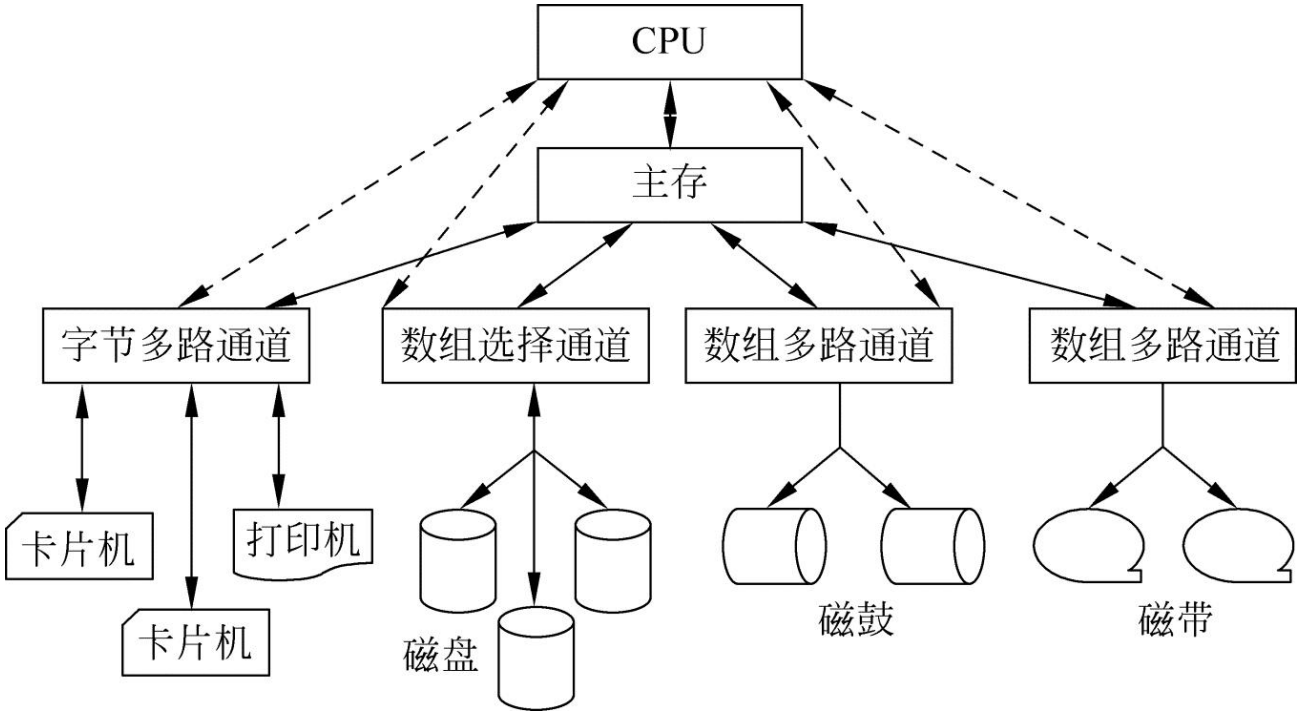


图6-7 IBM-370系统结构

6.2.1 I/O系统结构

3. 总线系统

- 计算机系统中的各个部件，如中央处理器、存储器以及各种I/O设备通过总线实现各种信息的传递。
- 总线的性能通过总线的**时钟频率**、**带宽**和相应的总线**传输速率**等指标来衡量。随着计算机的CPU和主存速率的提高，字长的增加，以及新型设备的推出，不断地推动着总线的发展。

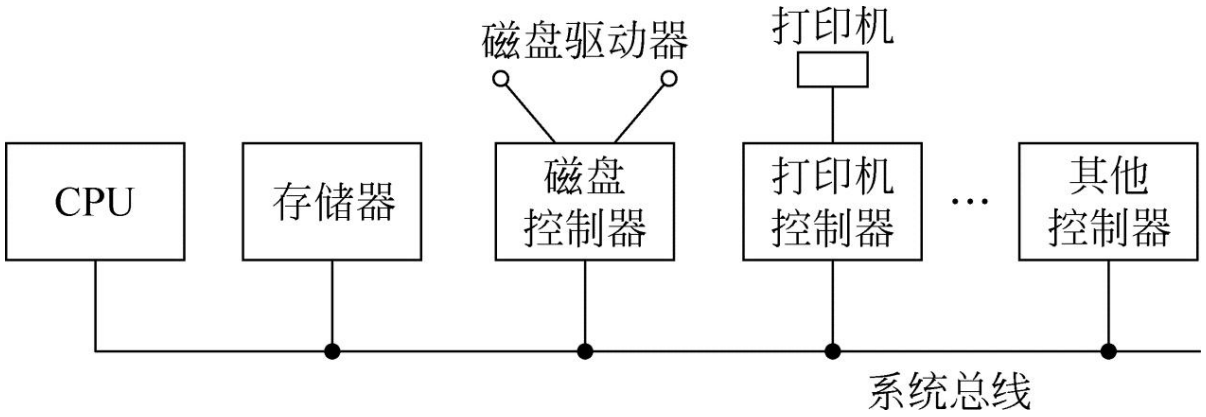


图6-8 总线型I/O系统结构

6.2.1 I/O系统结构

- ISA（Industry Standard Architecture）总线
  - 最早的PC总线是IBM公司1981年在PC/XT电脑采用的系统总线，它基于8bit的8088 处理器，被称为PC总线或者PC/XT总线。
  - 1984年，IBM 推出基于16-bit Intel 80286处理器的PC/AT 电脑，系统总线也相应地扩展为16bit，并被称呼为PC/AT 总线。而为了开发与IBM PC 兼容的外围设备，行业内便逐渐确立了以IBM PC总线规范为基础的ISA（工业标准架构：Industry Standard Architecture ）总线。
- EISA（Extended Industry Standard Architecture）总线
  - EISA总线是1988年由Compaq等9家公司联合推出的总线标准。它是在ISA总线的基础上使用双层插座，在原来ISA总线的98条信号线上又增加了98条信号线，也就是在两条ISA信号线之间添加一条EISA信号线。在实用中，EISA总线完全兼容ISA总线信号。

6.2.1 I/O系统结构

- VESA（Video Electronics Standard Association）总线
  - 1992年由60家附件卡制造商联合推出的一种局部总线，简称为VL（VESAlocalbus）总线。它的推出为微机系统总线体系结构的革新奠定了基础。该总线系统考虑到CPU与主存和Cache的直接相连，通常把这部分总线称为CPU总线或主总线，其他设备通过VL总线与CPU总线相连，所以VL总线被称为局部总线。它定义了32位数据线，且可通过扩展槽扩展到64位，使用33MHz时钟频率，最大传输率达132MB/s，可与CPU同步工作。是一种高速、高效的局部总线，可支持386SX、386DX、486SX、486DX及奔腾微处理器。

6.2.1 I/O系统结构

- PCI（Peripheral Component Interconnect）总线
  - 是当前最流行的总线之一，它是由Intel公司推出的一种局部总线。它定义了32位数据总线，且可扩展为64位。PCI总线主板插槽的体积比原ISA总线插槽还小，其功能比VESA、ISA有极大的改善，支持突发读写操作，最大传输速率可达132MB/s，可同时支持多组外围设备。PCI局部总线不能兼容现有的ISA、EISA、MCA（microchannelarchitecture）总线，但它不受制于处理器，是基于奔腾等新一代微处理器而发展的总线。
- PCI-Express
  - 在经历了长达10年的修修补补，PCI 总线已经无法满足电脑性能提升的要求，必须由带宽更大、适应性更广、发展潜力更深的新一代总线取而代之，这就是PCI-Express 总线。

6.2.2 I/O控制方式

- 为了有效地实现物理I/O操作，必须通过硬件和软件技术，对CPU和I/O设备的职能进行合理的分工，以调节系统性能和硬件成本之间的矛盾。

- 选择和衡量I/O控制方式有如下3条原则：
  ① 数据传输速度足够高，能满足用户的需要但又不丢失数据；
  ② 系统开销小，所需的处理控制程序少；
  ③ 能充分发挥硬件资源的能力，使I/O设备尽可能忙，而CPU等待时间尽可能少。

6.2.2 I/O控制方式

- I/O设备的控制方式分为四类：
  ① 直接程序控制方式；
  ② 中断驱动控制方式；
  ③ 直接存储器访问(DMA)控制方式；
  ④ 通道控制方式。

- I/O控制方式发展的目标是尽量减少主机对I/O控制的干预，把主机从繁杂的I/O控制事务中解脱出来，更多地进行数据处理，提高计算机效率和资源的利用率。它们之间的主要差别在于CPU与外围设备并行工作的方式不同，并行工作的程度不同。

6.2.2 I/O控制方式

- **直接程序控制方式**

- **由用户进程直接控制主存或CPU和外围设备之间的信息传送。通过输入/输出指令或询问指令测试I/O设备的忙/闲标志位，决定主存储器与外围设备之间是否交换一个字符或一个字。**

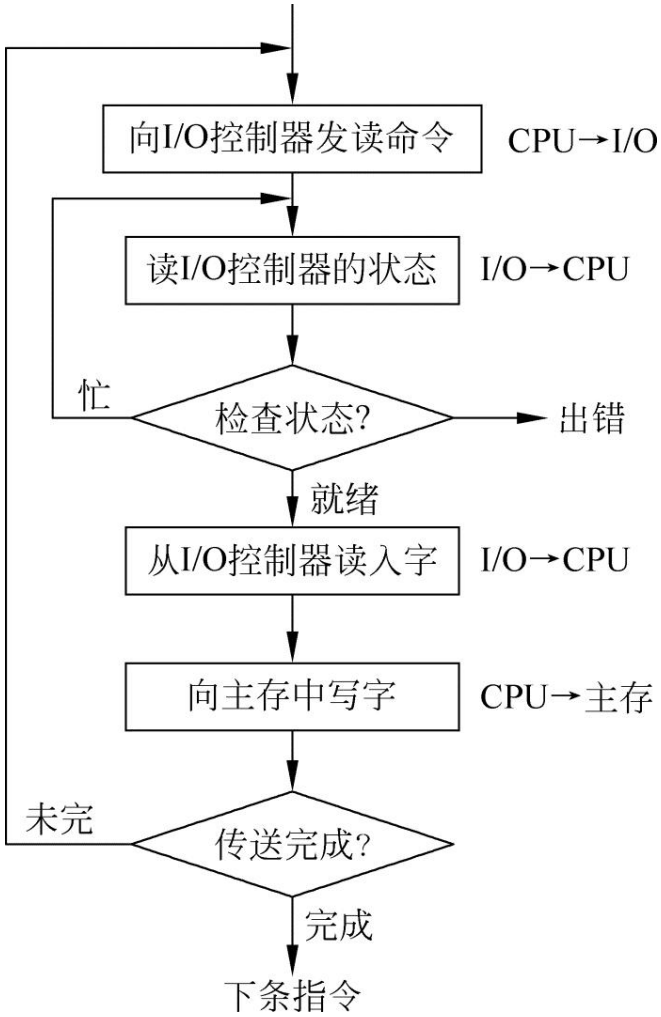


图6-9 直接程序控制方式流程

6.2.2 I/O控制方式

- 直接程序控制方式虽然简单，不需要多少硬件的支持，但由于高速的CPU和低速的I/O设备之间的速度不匹配，因此，CPU与外围设备只能串行工作，使CPU的绝大部分时间都处于等待是否完成输入/输出操作的循环测试中，造成CPU的极大浪费，外围设备也不能得到合理的使用，整个系统的效率很低。

- 直接程序控制方式只适合于CPU执行速度较慢，且外围设备较少的系统。

6.2.2 I/O控制方式

- **中断驱动控制方式**
- **中断机制引入后，外围设备仅当操作正常结束或异常结束时才向中央处理机发出中断请求。在I/O设备输入每个数据的过程中，由于无须CPU干预，一定程度上实现了CPU与I/O设备的并行工作。仅当输入/输出完一个数据时，才需CPU花费极短的时间做中断处理。**

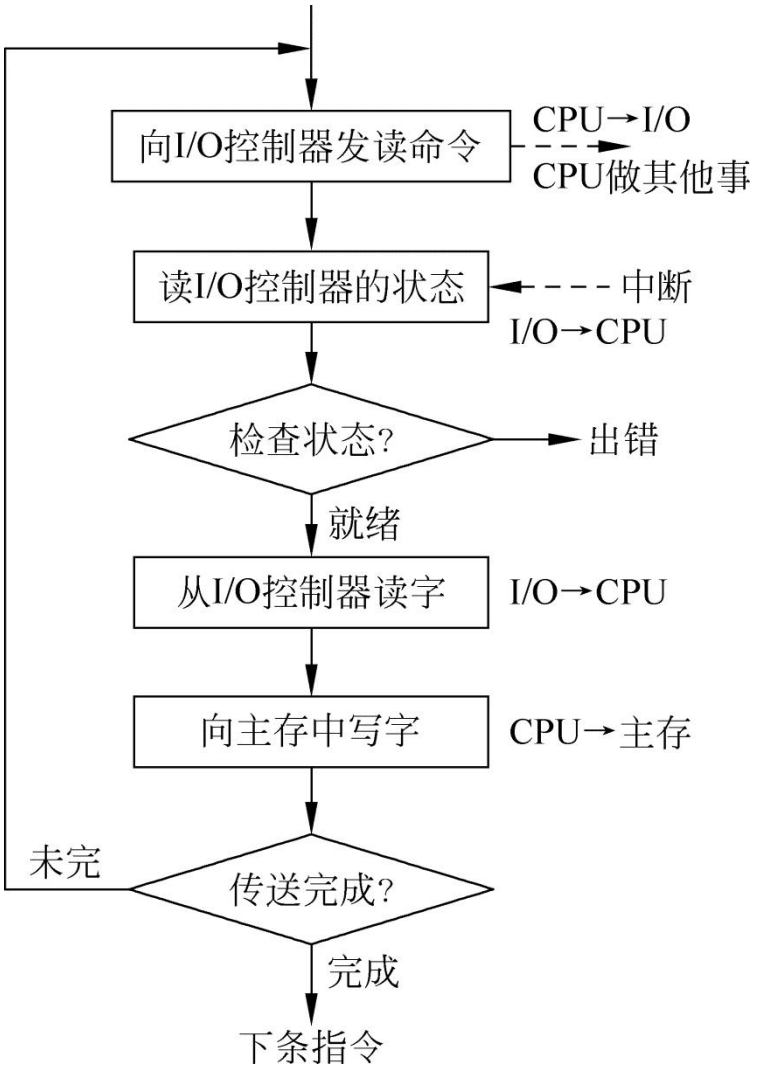


图6-10 中断驱动方式流程

6.2.2 I/O控制方式

- 中断驱动方式可以成百倍地提高CPU的利用率。但是由于输入/输出操作直接由CPU控制，每传送一个字符或一个字，都要发生一次中断，仍然占用了大量的CPU处理时间，可以通过为外围设备增加缓冲寄存器存放数据，大大减少中断次数。

6.2.2 I/O控制方式

- **直接存储器访问控制方式**
- 又称DMA（Direct Memory Access）方式。为了进一步减少CPU对输入/输出的干预，防止并行操作设备过多CPU来不及处理或因速度不匹配而造成的数据丢失现象，而引入DMA控制方式。
- 它不仅设有中断机构，而且增加了DMA控制机构。在DMA控制器的控制下，采用窃取或挪用总线控制权，占用CPU的一个工作周期把数据缓冲器中的数据直接送到主存地址寄存器所指向的主存区域中，在设备和主存之间开辟直接数据交换通道，成批地交换数据，而不必CPU的干预。

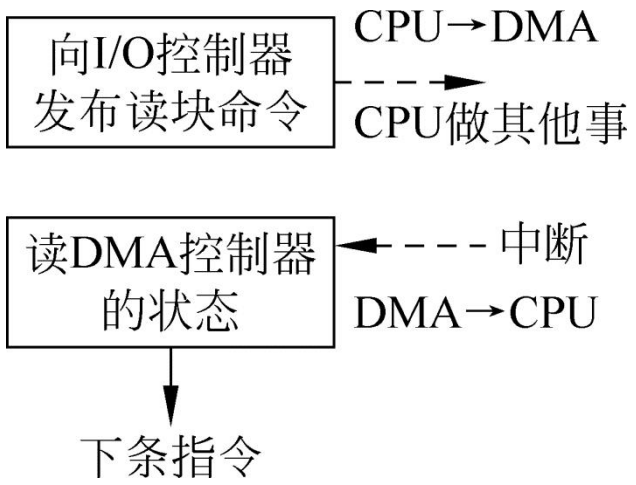
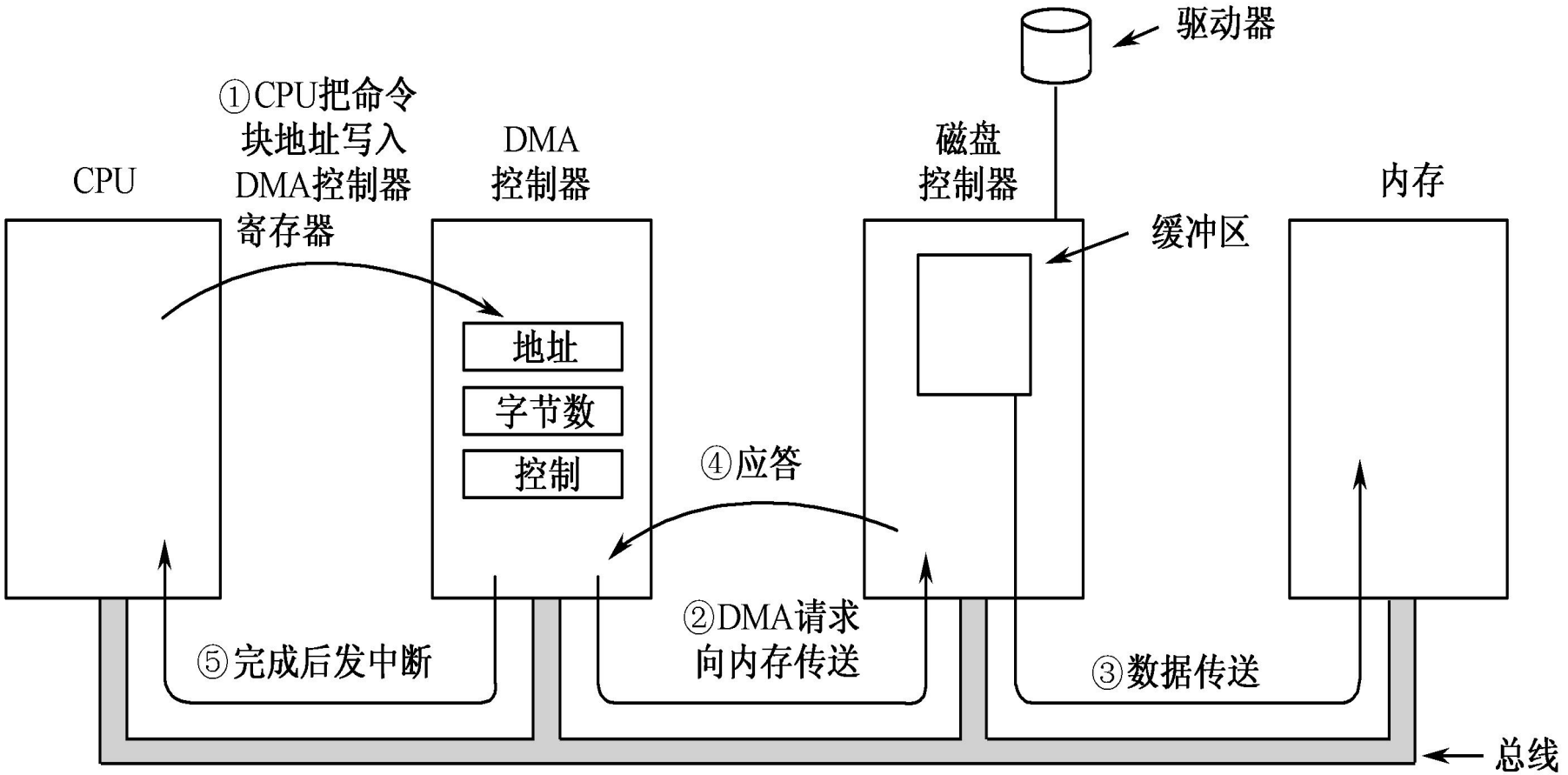


图6-11 DMA控制方式流程


驱动器
①CPU把命令
块地址写入
DMA控制器
寄存器
CPU
DMA
控制器
磁盘
控制器
内存
缓冲区
地址
字节数
控制
④应答
②DMA请求
向内存传送
⑤完成后发中断
③数据传送
总线

6.2.2 I/O控制方式

- DMA方式的特点：

① 数据传输以数据块为基本单位；

② 所传送的数据从设备直接送入主存，或者从主存直接输出到设备上；

③ 仅在传送一个或多个数据块的开始和结束时，才需CPU的干预，而整块数据的传送则是在控制器的控制下完成。

6.2.2 I/O控制方式

- 在DMA控制器中设置四类寄存器实现主机与控制器之间成块数据的直接交换：
  ① 命令/状态寄存器CR：用于接收从CPU发来的I/O命令或有关控制信息， 或设备的状态。
  ② 内存地址寄存器MAR：输入时，存放把数据从设备传送到内存的起始目标地址；输出时，存放由内存到设备的内存源地址。
  ③ 数据寄存器DR：暂存从设备到内存或内存到设备的数据。
  ④ 数据计数器DC：存放本次CPU要读或写的字(节)数。

- DMA控制方式线路简单，价格低廉，适合高速设备与主存之间的成批数据传输，但其功能较差，不能满足复杂的输入/输出要求。

6.2.2 I/O控制方式

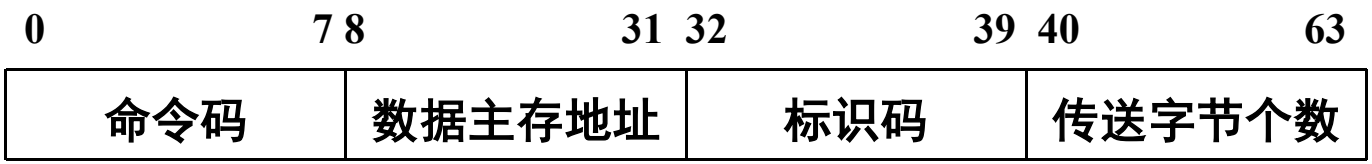
- **通道控制方式**

- (1)通道控制方式的引入
- 直接程序控制方式和中断控制方式适合于低速设备的数据传输；DMA方式适合于高速设备，但一个DMA只能控制少量同类设备。

- 通道控制方式将对一个数据块的读(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，可以进一步减少CPU的干预程度。
- 实现CPU、通道和I/O设备三者的并行操作，从而更加有效地提高整个系统的资源利用率。

6.2.2 I/O控制方式

- (2)通道指令和通道程序
- 通道程序规定了通道进行一次输入输出操作应执行的操作及顺序，由一系列通道指令（Channel Command Word ——CCW）所构成。通道程序在进程要求数据传送时由系统自动生成，通道指令一般包含被交换数据在主存中占据的位置、传送方向、数据块的长度以及被控制的I/O设备的地址信息、特征信息等。



IBM系统通道指令格式

6.2.2 I/O控制方式

- 程序的功能：要求在新的一页第4行的位置打印输出一行信息："Operating System"。
- 假定用户要求输出的信息已经存放在主存L单元开始的区域中，连空格在内共16个字符。组织好的通道程序在主存K单元开始的区域中。其中，命令码07表示"对折页线"(即走到新的一页开始)，命令码EF表示"走纸3行"(即走纸到第4行的位置)，命令码F9表示"打印一行信息"，标识码60表示有后续命令。最后一条命令要求打印从L单元开始的16个字符，打印结束后本操作结束。

主存地址	命令码	数据主存地址	标识码	传送字节个数
K	07	000000	60	0001
K+8	EF	000000	60	0001
K+16	F9	L	00	0010

6.2.2 I/O控制方式

- (3)I/O通道的启动与结束

- 当进程提出I/O请求后，操作系统首先分配通道和外设，然后按照I/O请求编制通道程序并存入主存，将其起始地址送入通道地址寄存器(CAW)，接着CPU发出“启动I/O”指令启动通道工作，启动成功后，通道逐条执行通道程序中的通道指令，控制设备实现I/O操作。

6.2.2 I/O控制方式

- CPU启动通道后，通道的工作过程如下
  - ① 从主存固定单元取出通道地址寄存器(CAW)，根据该地址从主存中取出通道指令，通道执行通道控制字寄存器(CCW) 中的通道命令，将 I/O地址送入CCW，发出读、写或控制命令，并修改CAW 使其指向下一条通道指令地址。
  - ② 控制器接收通道发来的命令之后，检查设备状态，若设备不忙，则告知通道释放CPU，开始I/O操作。执行完毕后，如果还有下一条通道指令，则返回①，否则转③。
  - ③ 通道完成I/O操作后，向CPU发出中断请求，CPU根据通道状态字了解通道和设备的工作情况，处理来自通道的中断。
- 可见，只是在I/O操作的起始和结束时，通道向CPU发出I/O中断申请，把产生中断的通道号、设备号存入中断寄存器，同时形成通道状态字汇报情况，等待处理。CPU用极短的时间参与控制管理工作，其他时间则处理与I/O无关的操作。

6.3 缓冲技术

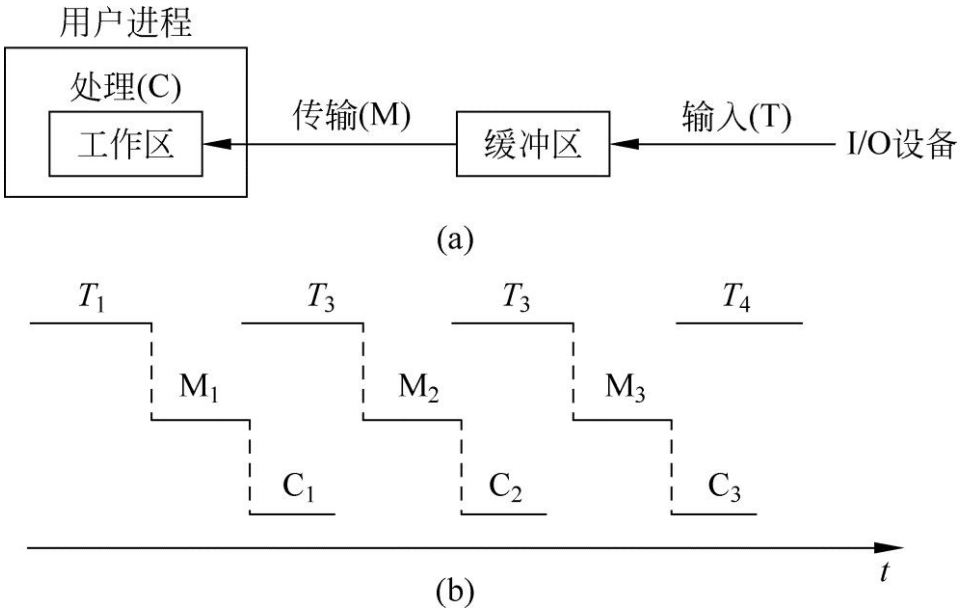
- 为了缓解CPU与外围设备之间速度不匹配和负载不均衡的问题，为了提高CPU和外围设备的工作效率，增加系统中各部件的并行工作程度，在现代操作系统中普遍采用了缓冲技术。
- 缓冲管理的主要职责是组织好缓冲区，并提供获得和释放缓冲区的手段。
- 缓冲区不仅适用于CPU与I/O设备之间，凡数据到达率与数据离去率不一致的地方，都可通过缓冲区来解决它们间的不匹配矛盾。比如，在网络通信中，可利用缓冲技术解决发送方与接收方之间速度不匹配的问题。

6.3.1 缓冲的引入

- **缓和CPU与I/O设备间速度不匹配的矛盾**
  - 高速的CPU与慢速I/O设备之间存在着速度差异很大，CPU是以微秒甚至微毫秒时间量级高速工作，而I/O设备则一般以毫秒甚至秒时间量级的速率工作。在不同阶段，系统各部分的负载往往很不平衡。
- **减少对CPU中断频率，放宽对CPU中断响应时间限制**
  - 在数据通信中，如果仅有一位数据缓冲接收数据，则必须在每收到一位数据时便中断一次CPU，进行数据的处理，否则缓冲区内的数据将被新传送来的数据冲掉。若设置一个具有8位的缓冲器，则可使CPU被中断的频率降低为原来的1/8。这样减少了CPU的中断次数和中断处理时间。
- **提高CPU和I/O设备之间的并行性**
  - 缓冲的引入可显著提高CPU与I/O设备之间的并行操作程度，提高系统的吞吐量和设备的利用率。

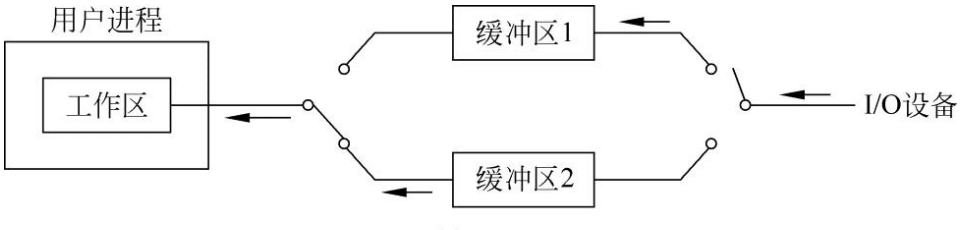
6.3.2 单缓冲

- 单缓冲是在设备和处理器之间设置一个缓冲器，由输入和输出设备共用。设备和处理器交换数据时，先把被交换数据写入缓冲器，然后需要数据的设备或处理机从缓冲器中取走数据。
- 由于缓冲器属于临界资源，所以输入设备和输出设备以串行方式工作，这样尽管单缓冲能匹配设备和处理机的处理速度，但是设备和设备之间并不能通过单缓冲达到并行操作。

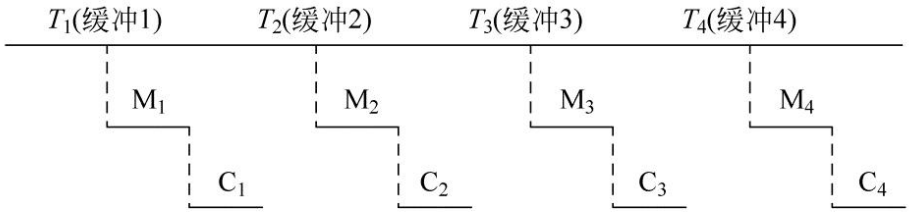


6.3.3 双缓冲

- 双缓冲机制又称为缓冲对换。双缓冲是为输入和输出设备设置两个缓冲区的缓冲技术。在设备输入数据时，可以把数据放入其中一个缓冲区中，在进程从缓冲区中取数据使用的同时，将输入数据继续放入另一个缓冲区中。当第一个缓冲区的数据处理完时，进程可以接着从另一个缓冲区中获得数据，同时，输入数据可以继续存入第一个缓冲区，仅当输入设备的速度高于进程处理这些数据的速度，两个缓冲区都存满时，造成输入进程等待。



(a)



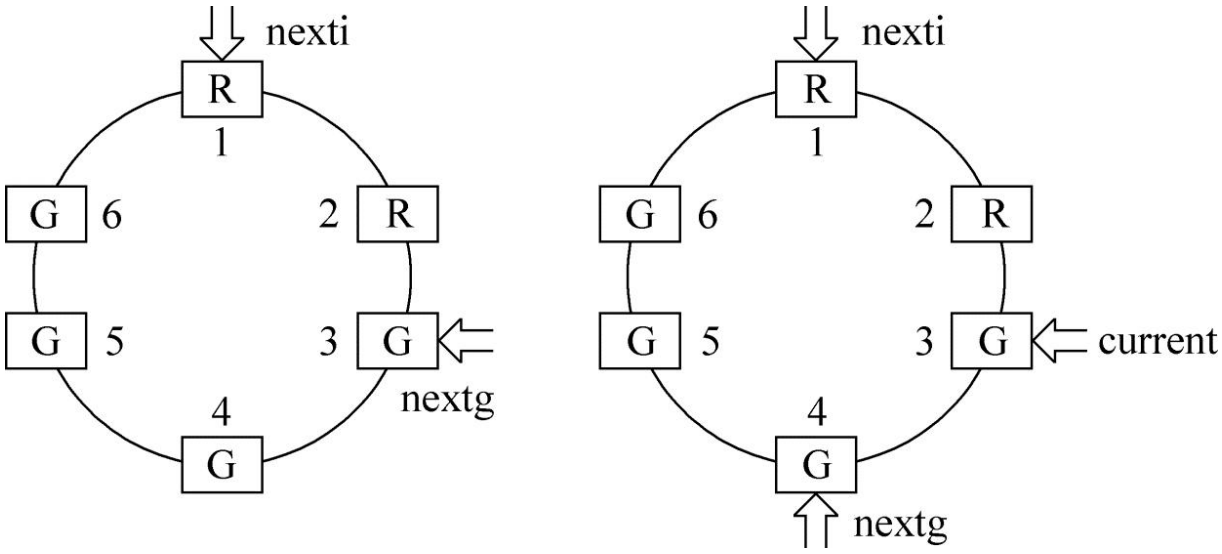
(b)

6.3.3 双缓冲

- 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，但在输入/输出设备和处理进程速度不匹配时仍不能适应。
- 双缓冲只是一种说明设备和设备、CPU和设备之间并行操作的简单模型，由于计算机系统中的外围设备较多，而双缓冲也难以匹配设备和处理器的处理速度，所以，双缓冲并不能用于实际系统中的并行操作。
- 现代计算机系统中一般使用多缓冲或缓冲池结构。

6.3.4 多缓冲

- 系统从主存中分配一组缓冲区组成多缓冲。
- 多缓冲中的缓冲区是系统的公共资源，可供各个进程共享，并由系统统一分配和管理。多个缓冲区组织成循环缓冲形式，对于用作输入的循环缓冲，通常是提供给输入进程或计算进程使用，输入进程不断向空缓冲区输入数据，而计算进程则从中提取数据进行计算。



6.3.5 缓冲池

- 一组多缓冲仅适用于某个特定的I/O进程和计算进程，当系统较大时，需要设置若干组多缓冲，这不仅消耗大量的贮存空间，而且其利用率不高。为了提高缓冲区的利用率，公用缓冲池被广泛使用，它由多个可共享的缓冲区组成。

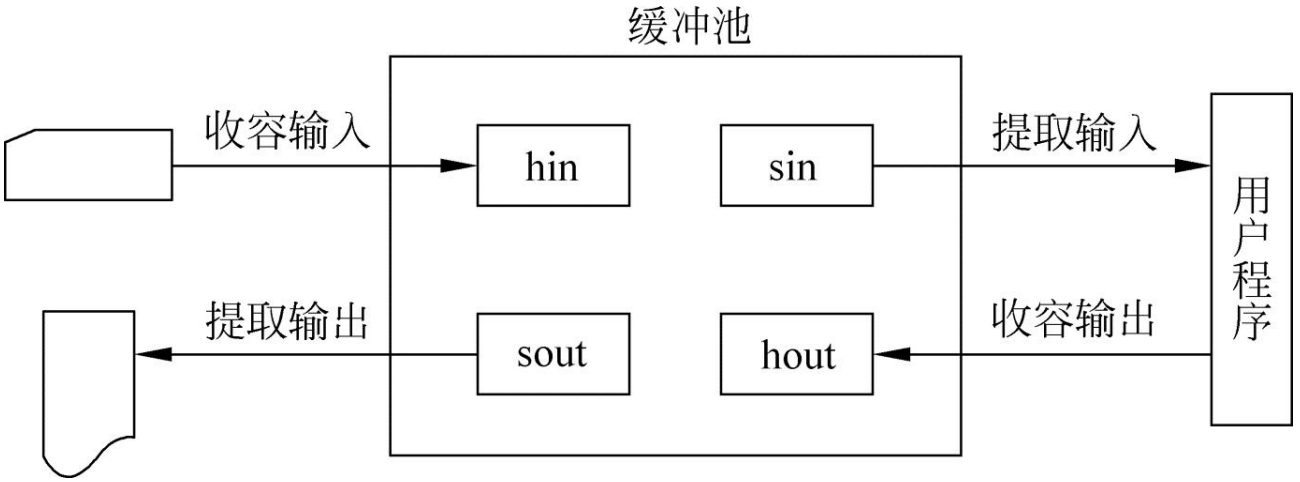
- 对于既可用于输入又可用于输出的公用缓冲池，按其使用状况可以分成三种类型的缓冲区：
  ① 空(闲)缓冲区；
  ② 装满输入数据的缓冲区；
  ③ 装满输出数据的缓冲区。

6.3.5 缓冲池

- 为了便于管理，可将相同类型的缓冲区链成一个队列，于是可形成以下三个队列：
  - ① 由空缓冲区所链成的空缓冲队列emq。
  - ② 由装满输入数据的缓冲区所链成的输入队列inq。
  - ③ 由装满输出数据的缓冲区所链成的输出队列outq。

6.3.5 缓冲池

- 在缓冲池中，有四种工作缓冲区，分别工作在收容输入、提取输入、收容输出和提取输出四种工作方式下。
  - ① 用于收容设备输入数据的工作缓冲区hin；
  - ② 用于提取设备输入数据的工作缓冲区sin；
  - ③ 用于收容CPU输出数据的工作缓冲区hout；
  - ④ 用于提取CPU输出数据的工作缓冲区sout。



6.4 独占设备的分配

- 在多道程序设计系统中，不允许用户直接启动外围设备，而必须由系统进行统一分配。当进程向系统提出I/O请求时，只要是可能和安全的，设备分配程序按照一定的策略，将设备分配给请求用户（进程），在有的系统中，还应分配相应的控制器和通道。

6.4.1 设备的逻辑号和绝对号

- 计算机系统中配置了各种不同类型的外围设备，每一类型外围设备可以有若干台。为了对设备进行管理，计算机系统为每一台设备确定一个编号，以便区分和识别，这个编号称为设备的绝对号（不可更改）。
- 在多道程序设计系统中，由于用户无法知道当前计算机系统中设备的使用状态，因此，一般用户不直接使用设备的绝对号，用户可以向系统说明所要使用的设备类型。至于实际使用哪一台设备，由系统根据该类设备的分配情况来决定。为了避免使用时产生混乱，用户可以在程序中对自己要求使用的若干台同类型设备给出编号。由用户在程序中定义的设备编号称为设备的逻辑号（可以更改）。

6.4.2 设备的独立性

- 设备的独立性也称设备的无关性，指应用程序独立于具体使用的物理设备，能有效地提高操作系统的可适应性和可扩展性。用户编制程序时，不必指明特定的设备，而是在程序中使用“设备类、相对号”定义的逻辑设备，程序执行时系统根据用户指定的逻辑设备转换成与其对应的具体物理设备，并启动该物理设备工作。
- 用户在编制程序时使用的设备与实际使用哪台设备无关，这种特性称为“设备的独立性”。
- 具有设备独立性的计算机系统，在设备分配时具有：
  ① 设备分配灵活性强。
  ② 设备分配适应性强，易于实现I/O重定向。

6.4.2 设备的独立性

- 设备驱动程序是一个与硬件（或设备）紧密相关的软件。为了实现设备的独立性，往往需要在设备驱动程序之上设置一层软件，称为设备独立性软件。其主要功能包括：
- （1）执行所有设备的公有操作
  - 公有操作包括：对独立设备的分配与回收；将逻辑设备名映射为物理设备名；对设备进行保护，禁止用户直接访问设备；缓冲管理；差错控制。
  - 由于在I/O操作中的绝大多数错误都与设备无关，所以I/O操作主要由设备驱动程序处理，而设备独立性软件只是处理那些设备驱动程序无法处理的错误。
- （2）向用户层（或文件层）软件提供统一的接口
  - 虽然各种设备内部的具体操作各不相同，但它们向用户提供的接口却是相同的。

6.4.3 独占设备的分配

- 设备分配方式有两种，即静态分配和动态分配。
- 静态分配方式是在用户作业开始执行之前，由系统一次性分配该作业所要求的全部设备、控制器和通道。一旦分配之后，这些设备、控制器和通道就一直为该作业所占用，直到该作业被撤销。
- 静态分配方式不会出现死锁，但是设备的使用效率低。独占型设备的分配采用静态分配策略。

6.4.3 独占设备的分配

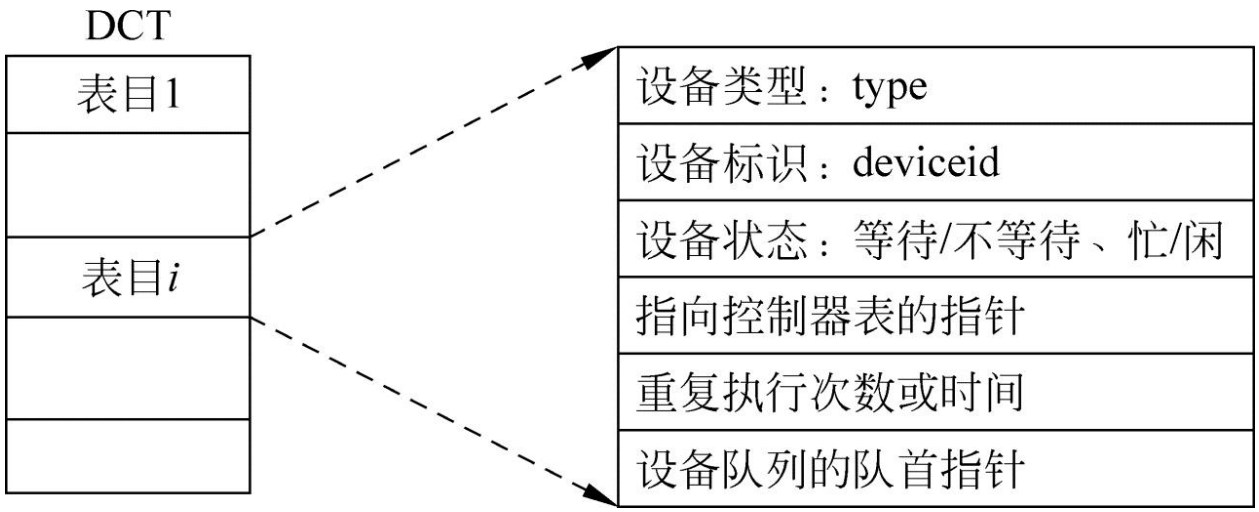
- **动态分配在进程执行过程中根据执行的需要进行分配**。当进程需要设备时，通过系统调用指令向系统提出设备请求，由系统按照事先规定的策略给进程分配所需要的设备、控制器和通道，一旦使用完毕，便立即释放。
- 动态分配方式有利于提高设备的利用率，但如果分配策略使用不当，则有可能造成进程死锁。

6.4.3 独占设备的分配

- 为了实现设备分配，系统设置了**设备控制表**、**控制器控制表**、**通道控制表**和**系统设备表**等数据结构，记录相应设备或控制器的状态以及对设备或控制器进行控制所需要的信息。

6.4.3 独占设备的分配

- **1.设备控制表**
- **系统为设备配置了一张设备控制表(DCT, Device Control Table)，用于记录设备的基本情况，包括设备类型、设备标识符、设备状态(好/坏，已/未分配，等待/不等待通道)、设备队列队首指针、与设备连接的控制器指针等。**



6.4.3 独占设备的分配

- **2.控制器控制表**
- **系统为每一个控制器设置一张控制表，称为控制器控制表(COCT, COntroller Control Table)，用于记录本控制器的使用状态以及与通道的连接情况等。**

控制器标识符：controllerid
控制器状态：忙/闲
与控制器链接的通道表指针
控制器队列的队首指针
控制器队列的队尾指针

6.4.3 独占设备的分配

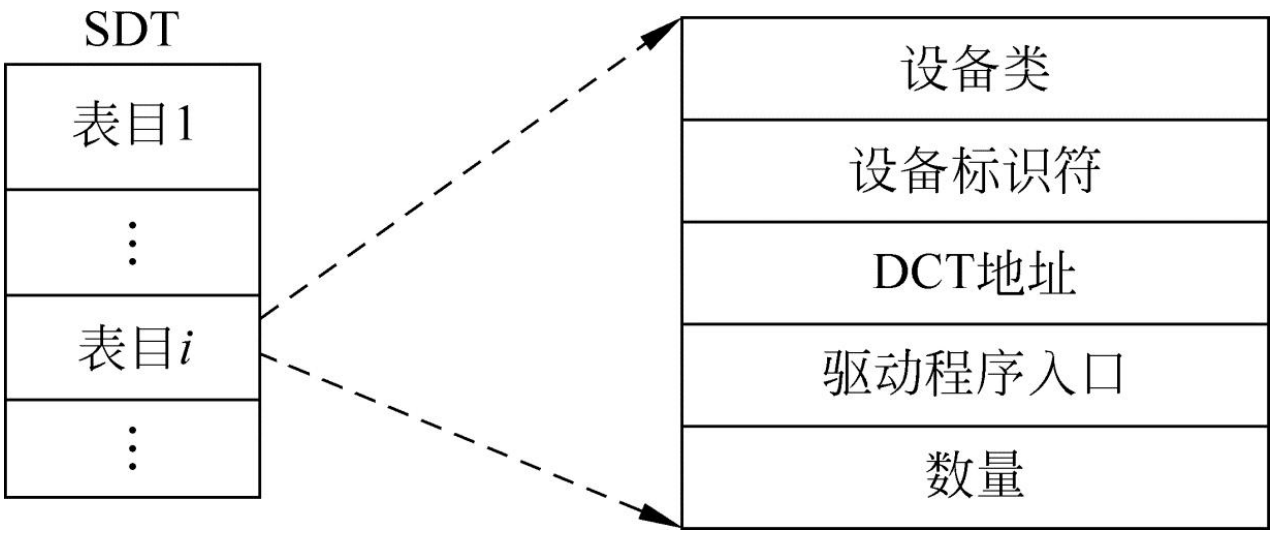
- **3.通道控制表**
- **系统为每一个通道配置了一通道控制表(CHCT, CHannel Control Table)。**

通道标识符：charmelid
通道状态：忙/闲
与通道链接的控制器表首址
通道队列的队首指针
通道队列的队尾指针

6.4.3 独占设备的分配

- **4.系统设备表**

- **系统设备表(SDT, System Device Table)又称为设备类表。整个系统一张系统设备表，用于记录已被连接到系统中所有类型设备的基本情况，包括设备类型、设备标识符、拥有同类型设备的数量、DCT及设备驱动程序的入口地址等。**



6.4.3 独占设备的分配

- **5.逻辑设备表(LUT)**
- **为了实现设备的独立性，系统还必须设置一张逻辑设备表(LUT, Logical Unit Table)，用于将应用程序中使用的逻辑设备名映射为物理设备名。该表的每个表目包括逻辑设备名和物理设备名。**

逻辑设备名	物理设备名	驱动程序入口地址	系统设备表指针
/dev/tty	3	1024	3
/dev/printer	5	2046	5
…	…	…	…

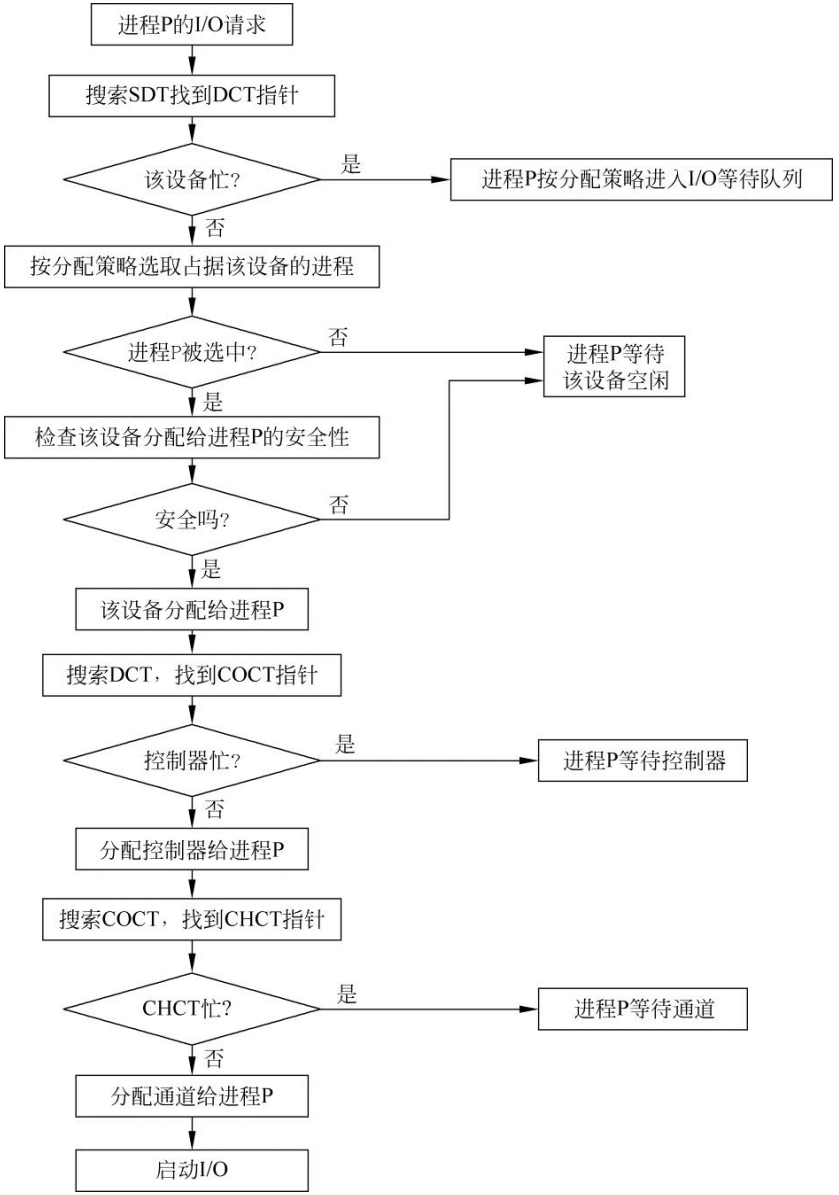
6.4.3 独占设备的分配

- 在多用户系统中，系统为每个用户设置一张LUT，并将该表放入进程的PCB中。当进程用逻辑设备名请求分配I/O设备时，系统为之分配相应的物理设备，并在LUT中建立一个表目，填上应用程序中使用的逻辑设备名和系统分配的物理设备名。
- 当进程提出I/O请求后，首先根据I/O请求中的设备名查找SDT，查找该类设备的DCT；再根据DCT中的设备状态字段，按照一定的算法，选择一台“好的且尚未分配的”设备进行分配，分配后修改设备类表中现存台数。把分配给作业的设备标志改成“已分配”且填上占用该设备的作业名和作业程序中定义的逻辑号，否则，将该进程的PCB插入设备等待队列。

6.4.3 独占设备的分配

- 当系统将设备分配给请求进程后，再到DCT中找到与该设备连接的COCT，从COCT的状态字段中判断出是否可以将该控制器分配。若不可以，则将该进程的PCB挂在该控制器的等待队列上。
- 通过COCT可找出与该控制器连接的CHCT。根据CHCT内的状态字段，判断出该通道是否可以进行分配。若忙，则将该进程的PCB挂在该通道的等待队列上。
- 显然，进程只有在获得设备、控制器和通道三者之后，才能启动设备进行I/O操作。

6.4.3 独占设备的分配


进程P的I/O请求
搜索SDT找到DCT指针
该设备忙?
是
进程P按分配策略进入I/O等待队列
否
按分配策略选取占据该设备的进程
进程P被选中?
否
进程P等待
该设备空闲
是
检查该设备分配给进程P的安全性
安全吗?
否
是
该设备分配给进程P
搜索DCT，找到COCT指针
控制器忙?
是
进程P等待控制器
否
分配控制器给进程P
搜索COCT，找到CHCT指针
CHCT忙?
是
进程P等待通道
否
分配通道给进程P
启动I/O

6.5 磁盘管理

- 磁盘存储器是一种高速、大容量的随机存储设备。现代计算机系统均配置了磁盘存储器，并以它为主存放大量的文件和数据。
- 磁盘分类
  - 软磁盘
  - 硬磁盘：固态硬盘、机械硬盘

磁盘概述

- **1956年：IBM向客户交付第一台磁盘驱动器RAMAC 305，可存储5MB数据，每MB成本为10000美元。它有2个冰箱那样大，使用50个24英寸盘片。**

磁盘概述

- 1980年：IBM发布了当时首个存储容量以GB为单位的磁盘，其大小和一台电冰箱大小差不多，重量为250kg，出售价格为40000美元。
- 1980年：希捷科技公司发布首个大小为5.25英寸的磁盘，容量为5MB。

磁盘概述

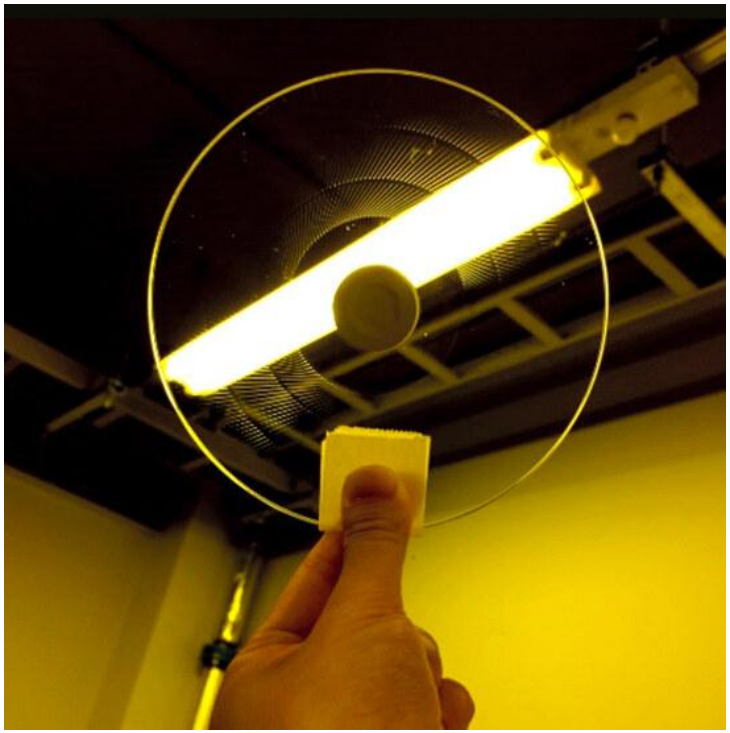
- 1983年：Rodime宣布推出了当时首个3.5英寸的磁盘——RO352，它包括有两个盘片，存储容量可达10MB。
- 1988年：Prairie Tek宣布推出了220磁盘，这是首个2.5英寸的磁盘，主要是针对初生的笔记本电脑市场推出的。220磁盘使用了两个盘片，存储容量可达20MB。 Connor
- 1990年：Western Digital发布了其首个3.5英寸的Caviar(鱼子酱) IDE磁盘。
- 1992年：希捷科技公司首次向外界展示了其2.5英寸的磁盘，在当时给了人们极大的震撼。成功的推出了存储容量为2.1GB的Barracuda(酷鱼)，这是首个采用7200r/min转速马达的磁盘。
- 1994年：Western Digital成功研发出Enhanced IDE，这是一个改良版的磁盘接口，并打破了当时528MB存储容量上限的束缚。EIDE同样也允许配置光驱和磁盘驱动器。

磁盘概述

- 1996年：希捷科技公司宣布推出了其Cheetah(捷豹)系列磁盘，这是首个采用10000r/min转速马达的磁盘。
- 1997年：IBM宣布推出了首个采用巨磁阻磁头(GMR)的磁盘——Deskstar 16GP Titan，在三个直径为3.5英寸的盘片上可装配16.8GB的存储容量。
- 1998年：IBM宣布推出了Microdrive(微磁盘)，这是当时世界上最小的磁盘，一个单一的1英寸盘片的容量可达340MB。
- 2000年：希捷科技公司发布了首款采用15000r/min转速马达的磁盘——Cheetah X15。
- 2003年：Western Digital推出了首个10000r/min的 SATA磁盘——Raptor(猛禽)，存储容量为37GB。
- 2004年：东芝宣布推出了世界上首款0.85英寸的磁盘MK2001MTN，在一个单一的盘片上，存储容量可达2GB。

磁盘概述

- 2015年11月：日本东京理科大学等组成的研究团队4日宣布研发出新技术，可制造新型磁盘，容量约是DVD的400倍。这种“全息图记忆”技术能将数据立体刻入，可用于长期保存影像等大型数据的设备。
- 一块与DVD相同大小的磁盘里可以保存2TB的数据。



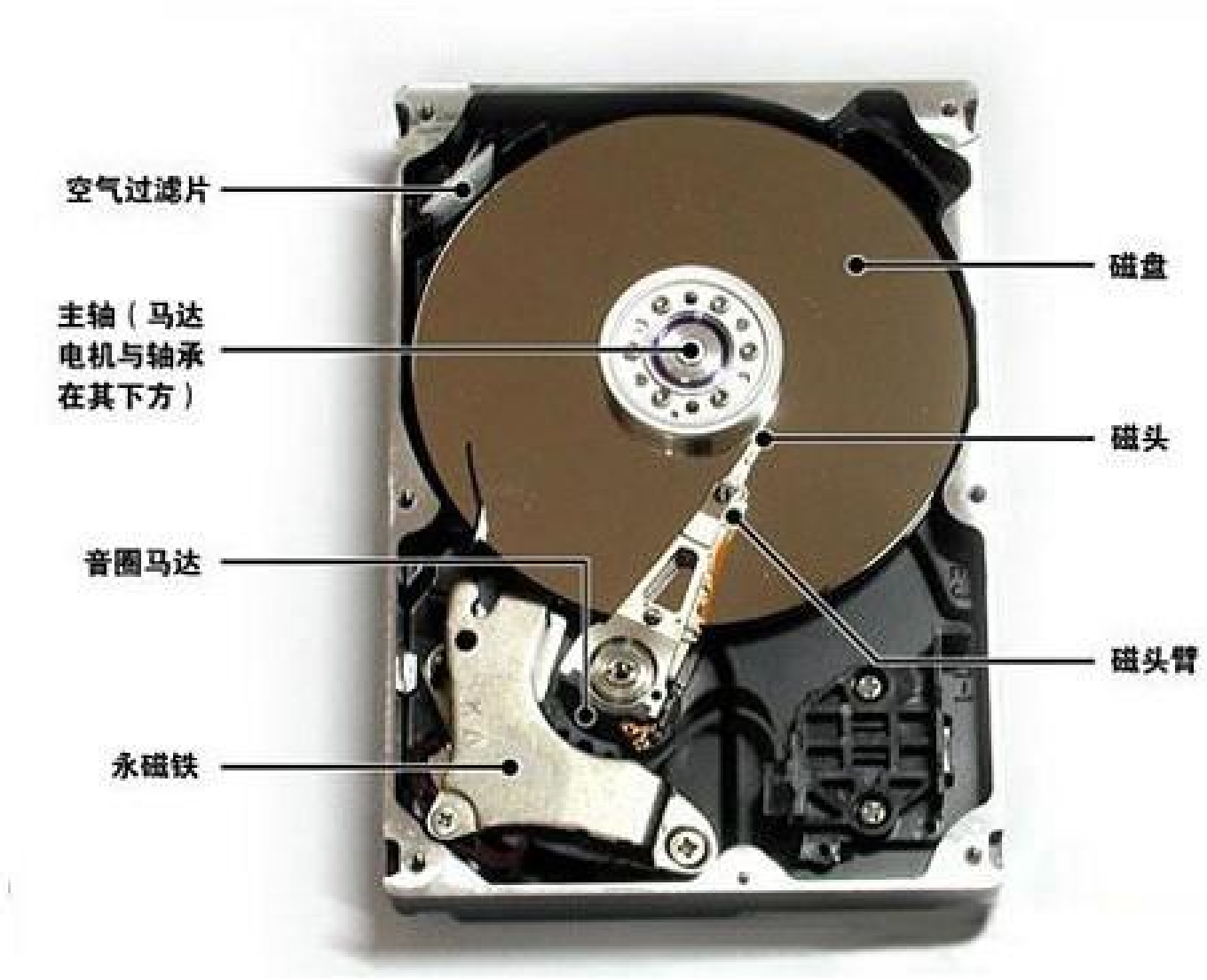
磁盘概述-固态硬盘

- 固态硬盘（Solid State Drives），简称固盘，固态硬盘（Solid State Drive）用固态电子存储芯片阵列而制成的硬盘，由控制单元和存储单元（FLASH芯片、DRAM芯片）组成。
- 1970年，StorageTek公司(Sun StorageTek)开发了第一个固态硬盘驱动器。
- 1989年，世界上第一款固态硬盘出现。
- 2006年3月，三星率先发布一款32GB容量的固态硬盘笔记本电脑，
- 2007年6月，东芝推出了其第一款120GB固态硬盘笔记本电脑。
- 2008年9月，忆正MemoRight SSD的正式发布，标志着中国企业加速进军固态硬盘行业。

磁盘概述-固态硬盘

- 2010年2月，镁光发布了全球首款SATA 6Gbps接口固态硬盘，突破了SATAII接口300MB/s的读写速度。
- 2010年底，瑞耐斯Renice推出全球第一款高性能mSATA固态硬盘并获取专利权。
- 2015年8月1日，特科芯推出了首款Type-C接口的移动固态硬盘。该款SSD提供了最新的Type-C接口，支持USB接口双面插入。
- 2016年1月1日，中国存储厂商特科芯发布了全球首款Type-C指纹加密SSD。

6.5.1 磁盘结构



6.5.1 磁盘结构

- 在微机上配置的温盘（温彻斯特，Winchester，机械硬盘）和软盘一般采用移动磁头结构。
- 一个盘组中所有盘片被固定在一根旋转轴上，沿着一个方向高速旋转。每个盘面配有一个读/写磁头，所有的读/写磁头被固定在移动臂上同时移动。

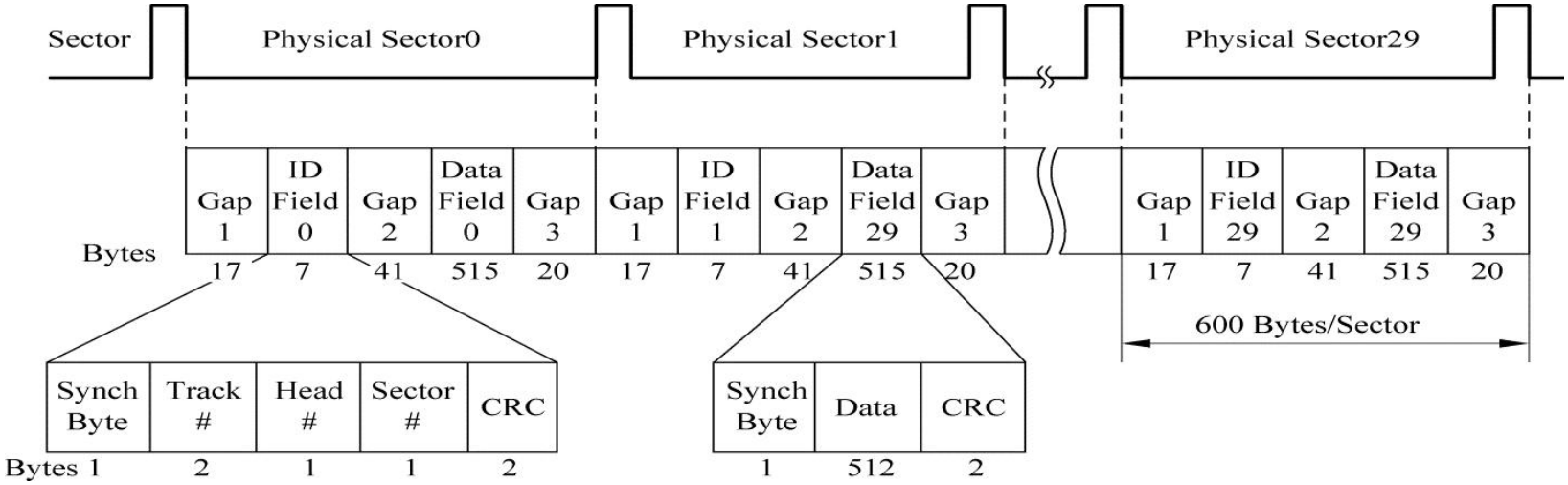


图6-23　机械硬盘的格式化（磁道）

6.5.1 磁盘结构

- 在微机上配置的温盘和软盘一般采用移动磁头结构。
- 一个盘组中所有盘片被固定在一根旋转轴上，沿着一个方向高速旋转。每个盘面配有一个读/写磁头，所有的读/写磁头被固定在移动臂上同时移动。

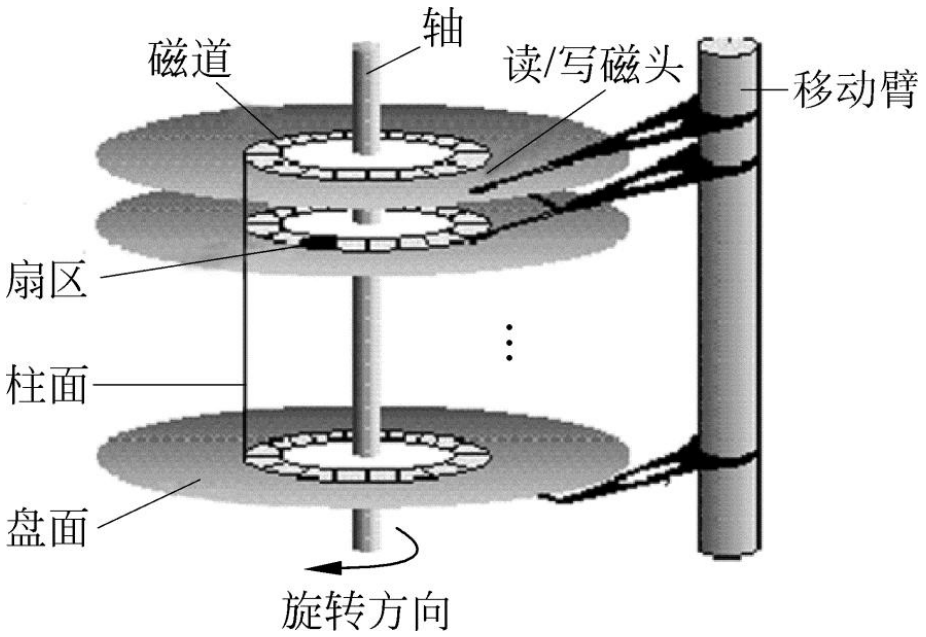
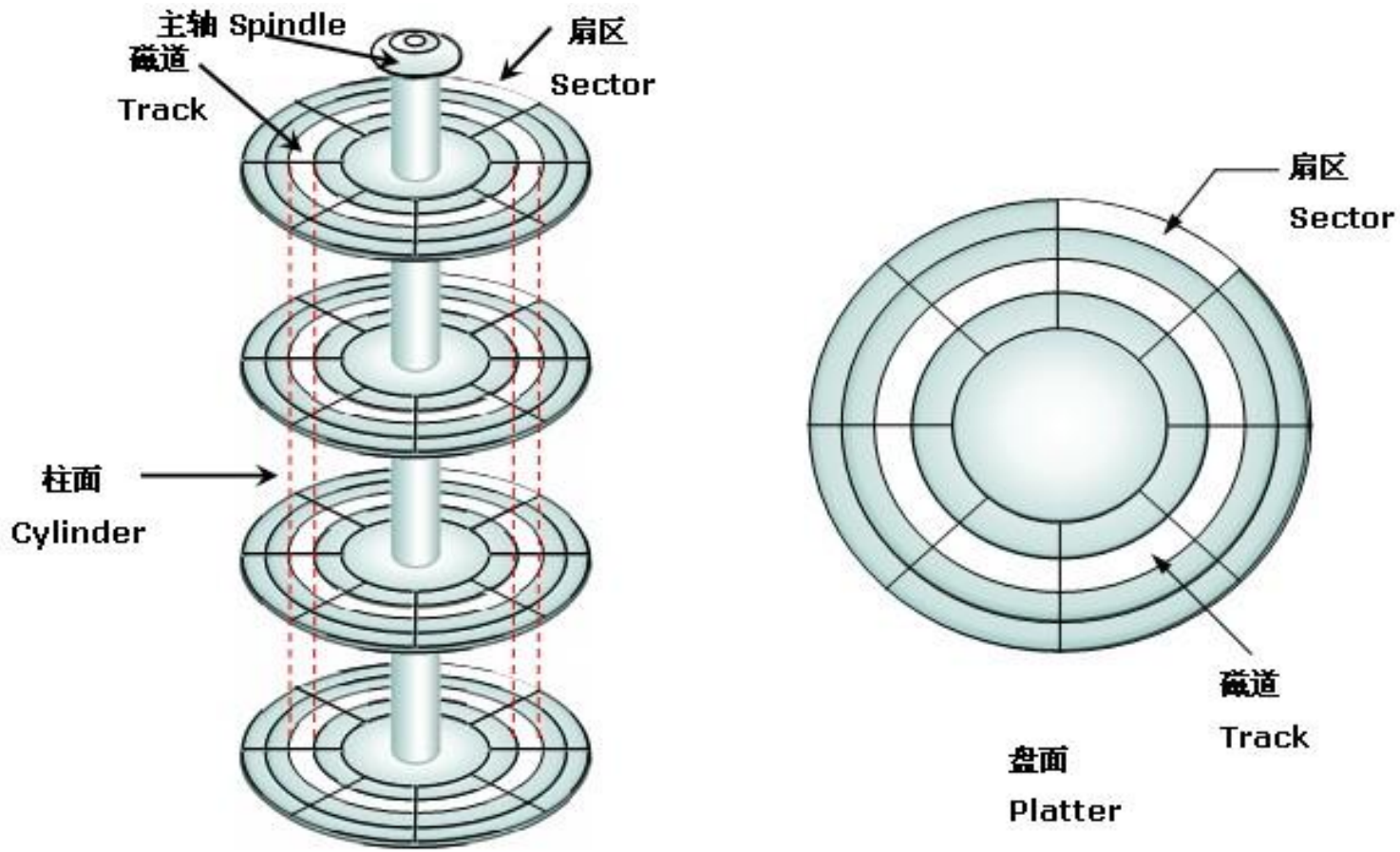


图6-24 磁盘结构

6.5.1 磁盘结构



- 硬盘容量=柱面数C*磁头数H*扇区数S*512B

6.5.1 磁盘结构

- 将磁头按从上到下次序编号，称为磁头号。每个盘面有许多磁道，磁头位置下各个盘面上的磁道处于同一个圆柱面上，称这些磁道组成了一个柱面。每个盘面上的磁道从0开始，由外向里顺序编号，通过移动臂的移动，读/写磁头可定位在任何一个磁道上，可见，移动磁头仅能以串行方式进行读/写。当移动臂移到某一个位置时，所有的读/写磁头处在同一个柱面上，盘面上的磁道号即为柱面号。每个盘面被划分成若干个扇区，沿与磁盘旋转相反的方向给个扇区编号，称为扇区号。**系统存放信息时，按柱面顺序存放**。
- 磁盘存储空间的位置可以由三个参数决定：**柱面号**、**磁头号**和**扇区号**（每个参数均从“0”开始编号）。磁盘空间的盘块按柱面（从0号柱面开始）、磁头、扇区顺序编号。

6.5.1 磁盘结构

- 假定在磁盘存储器中用t表示每个柱面上的磁道数，用s表示每个磁道上的扇区数，则第i柱面号、j磁头号、k扇区号所对应的块号b可用如下公式确定：

$$b=k+s\times(j+i\times t)$$

- 根据块号也可以确定该块在磁盘上的位置。在上述假设下，每个柱面上有：$s\times t$个磁盘块（扇区），为了计算第p块在磁盘上的位置，可以令$d=s\times t$，则有：
  - i柱面号$=[p/d]$
  - j磁头号$=[(p \bmod d)/s]$
  - k扇区号$=(p \bmod d \bmod s)$
- 第p块在磁盘上的位置就可以由i、j、k三个参数确定。

6.5.2 磁盘空间的管理

- 磁盘空间被操作系统和其他用户共享，大量的信息存放在磁盘上，系统需要对磁盘空间进行分配与回收管理。
- 磁盘空间的管理办法（第7章文件管理介绍）主要有：
  1. 空闲块表法
  2. 空闲块链法
  3. 位示图法
  4. 成组链接法

6.5.3 驱动调度

- 磁盘是目前使用最典型而又最广泛的一种块设备。任何一个对磁盘的访问请求，应给出访问磁盘的存储空间地址：柱面号、磁头号和扇区号。在启动磁盘执行输入/输出操作时，先把移动臂移动到指定的柱面，再等待指定的扇区旋转到磁头位置下，最后让指定的磁头进行读写，完成信息传送。
- 启动磁盘完成一次输入/输出操作所花的时间包括：
  - 寻找（寻道）时间：磁头在移动臂带动下移动到指定柱面所花的时间。
  - （旋转）延迟时间：指定扇区旋转到磁头下方位置所需的时间。
  - 传送时间：由磁头进行读/写，完成信息传送的时间。

6.5.3 驱动调度

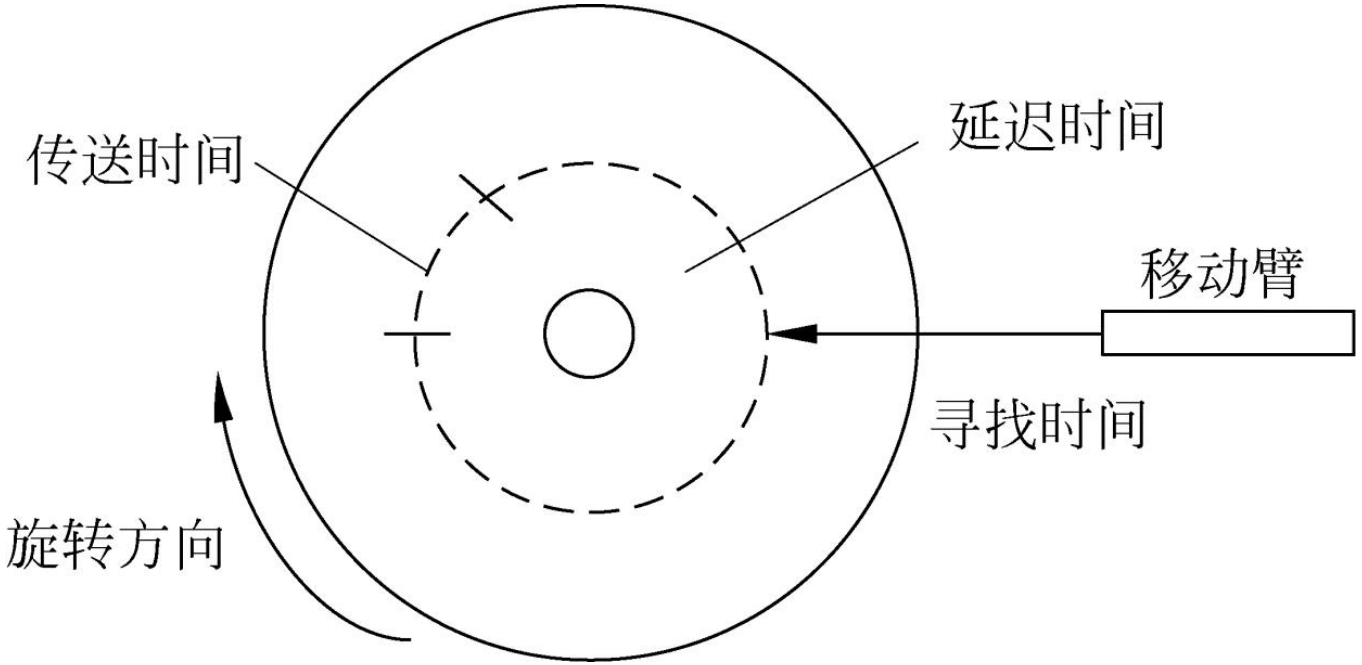


图6-25 访问磁盘的操作时间

6.5.3 驱动调度

- 磁盘属于共享型设备，在多道程序设计系统中，同时有若干个访问者请求磁盘执行输入/输出操作，为了保证信息的安全，系统在任一时刻只允许一个访问者启动磁盘执行操作，其余的访问者必须等待。一次输入/输出操作结束后，再释放一个等待访问者。
- 为了提高系统效率，降低若干个访问者执行输入/输出操作的总时间（平均服务时间），增加单位时间内输入/输出操作的次数，系统应根据移动臂的当前位置选择寻找时间和延迟时间尽可能小的那个访问者优先得到服务。
- 由于在访问磁盘时间中，**“寻找时间”是机械运动时间**，通常在几十毫秒时间量级。因此**设法减小寻找时间是提高磁盘传输效率的关键**。

6.5.3 驱动调度

- 系统采用一定的调度策略来决定各个请求访问磁盘者的执行次序，这项工作称为磁盘的“驱动调度”，采用的调度策略称为“驱动调度算法”。

- 对磁盘来说，驱动调度先进行“移臂调度”，以尽可能地减少寻找时间，再进行“旋转调度”，以减少延迟时间。

6.5.3 驱动调度-移臂调度

- 根据访问者指定的柱面位置来决定执行次序的调度称为“移臂调度”，移臂调度的目标是尽可能地减少输入/输出操作中的寻找时间。

- 常用的移臂调度算法有先来先服务调度算法、最短寻道时间优先调度算法、电梯调度算法、单向扫描算法和双向扫描算法等。

6.5.3 驱动调度-移臂调度

- （1）先来先服务调度算法FCFS

- 先来先服务调度算法是一种最简单的移臂调度算法。该算法只是根据访问者提出访问请求的先后次序进行调度，并不考虑访问者所要求访问的物理位置。

- 此算法简单且公平。但由于未对寻道进行优化，移动臂来回地移动，致使寻道时间可能比较长。
- 先来先服务算法仅适合于磁盘I/O请求数目较少的场合。

6.5.3 驱动调度-移臂调度

- （1）先来先服务调度算法FCFS
- 【例】现在读写磁头正在53号柱面上执行输入/输出操作，而访问者请求访问的柱面顺序为：98，183，37，122，14，124，65，67。FCFS调度共移动640个柱面。

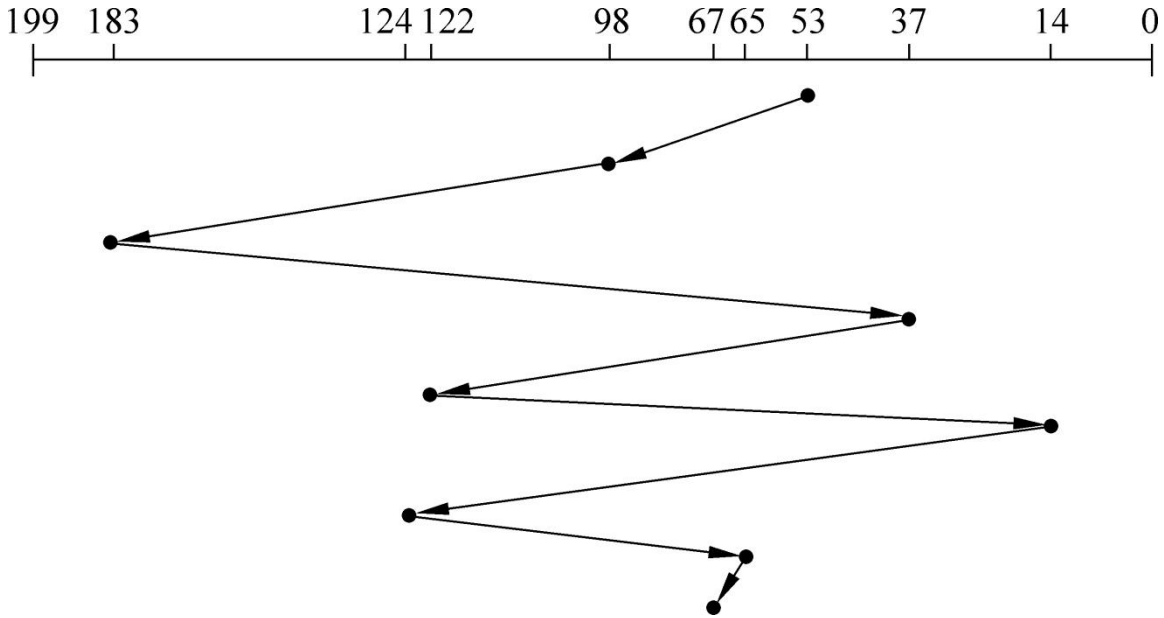


图6-26 先来先服务调度示意图

6.5.3 驱动调度-移臂调度

- （2）最短寻找时间优先调度算法SSTF

- 最短寻找时间优先调度算法总是从若干请求访问者中挑选与当前磁头所在的磁道距离最近，使每次寻道时间最短的那个请求进行调度，而不管访问者到达的先后次序。

- 该算法与先来先服务算法相比，大幅度地减少了寻找时间，从而缩短了为各请求访问者服务的平均时间，提高了系统效率。但它并未考虑访问者到来的先后次序，可能存在某进程由于距离当前磁头为之较远，致使该进程的请求被大大的推迟，即发生“饥饿”现象。

6.5.3 驱动调度-移臂调度

- （2）最短寻找时间优先调度算法SSTF
- 图6-27给出了采用最短寻找时间优先算法决定访问者执行输入输出操作的次序，读写磁头总共的移动距离为236个柱面。

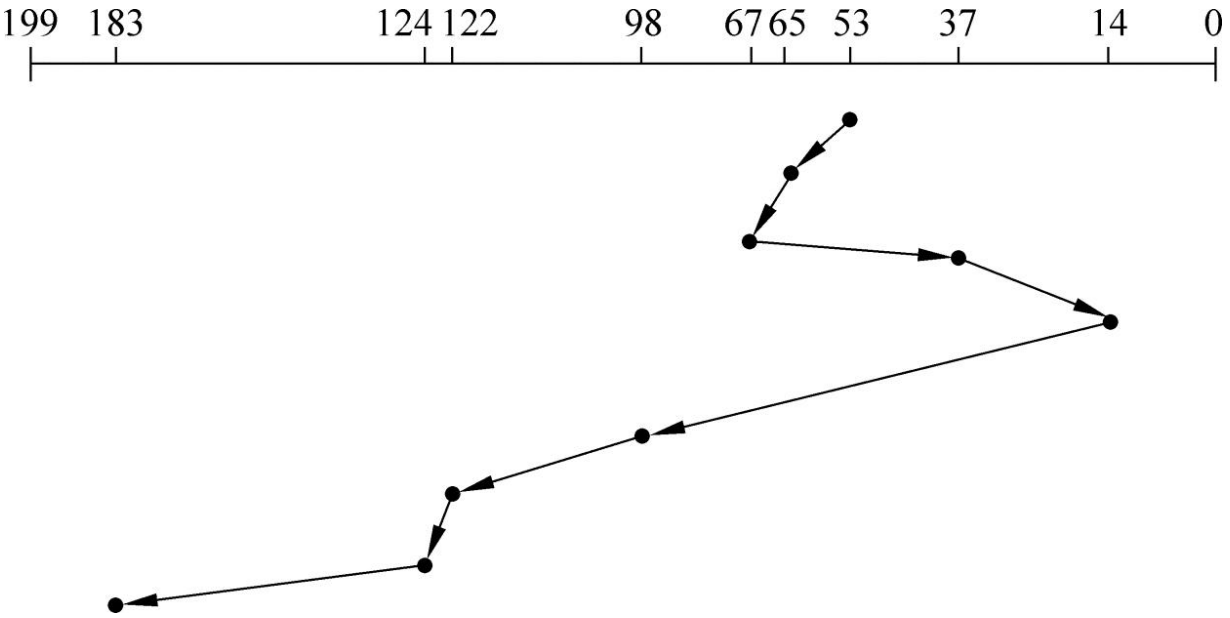


图6-27　最短寻找时间优先调度示意图

6.5.3 驱动调度-移臂调度

- （3）单向扫描调度算法

- 单向扫描调度算法不论访问者的先后次序，总是从0号柱面开始向里扫描，依次选择所遇到的请求访问者；移动臂到达盘面的最后一个柱面时，立即带动读/写磁头快速返回到0号柱面，返回时不为任何请求访问者服务，返回0号柱面后再次进行扫描。

- 该算法虽然考虑了移动臂的移动距离问题，但由于存在一趟空扫描，系统的效率并未得到大的提高。

6.5.3 驱动调度-移臂调度

- （3）单向扫描调度算法
- 图6-28给出了采用单向扫描算法决定访问者执行输入输出操作的次序，读写磁头总共的移动距离为382个柱面。

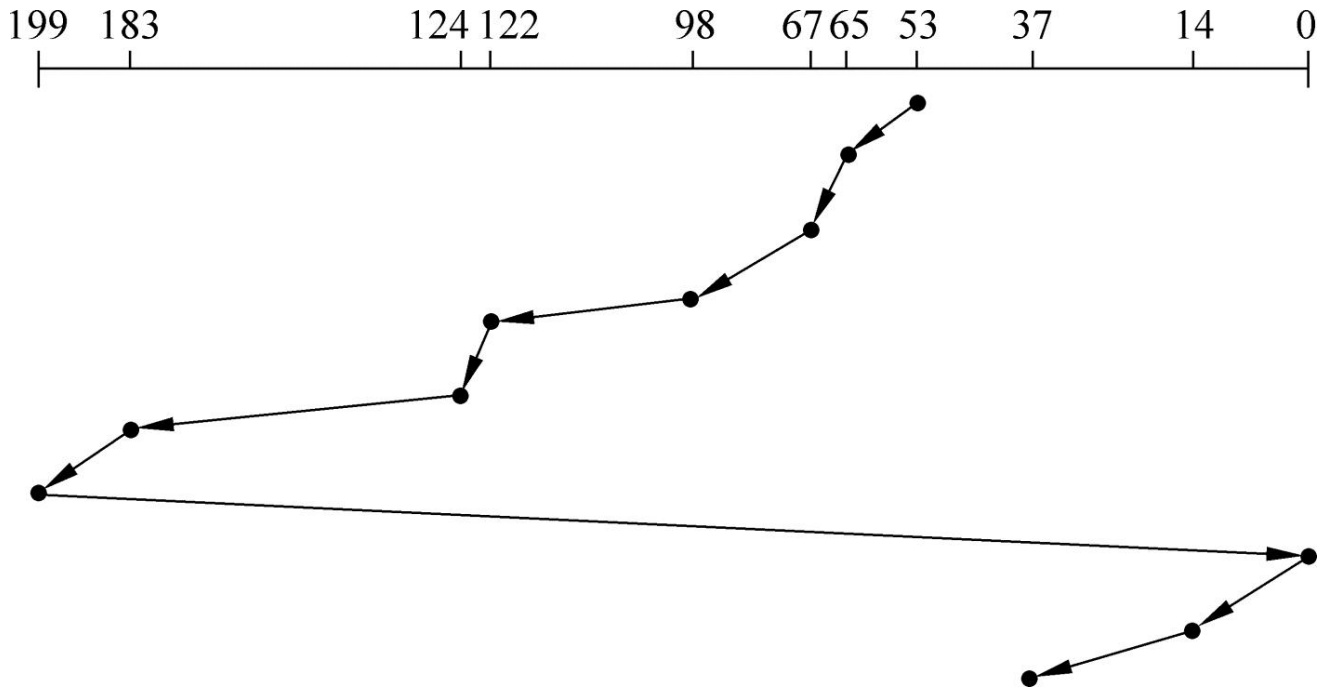


图6-28　单向扫描调度示意图

6.5.3 驱动调度-移臂调度

- （4）双向扫描调度算法

- 双向扫描调度算法从0号柱面开始向里扫描，依次选择所遇到的请求访问者；移动臂到达最后一个柱面时，调转方向从最后一个柱面向外扫描，依次选择所遇到的请求访问者。

- 双向扫描算法解决了单向扫描算法中的一趟空扫描问题，减少了寻找时间，提高了系统的访问效率，但在每次扫描过程中必须从最外磁道扫描到最内磁道，有可能存在部分空扫描。

6.5.3 驱动调度-移臂调度

- （4）双向扫描调度算法
- 图6-29给出采用双向扫描算法决定访问者执行输入/输出操作的次序，读/写磁头总共的移动距离为321个柱面。

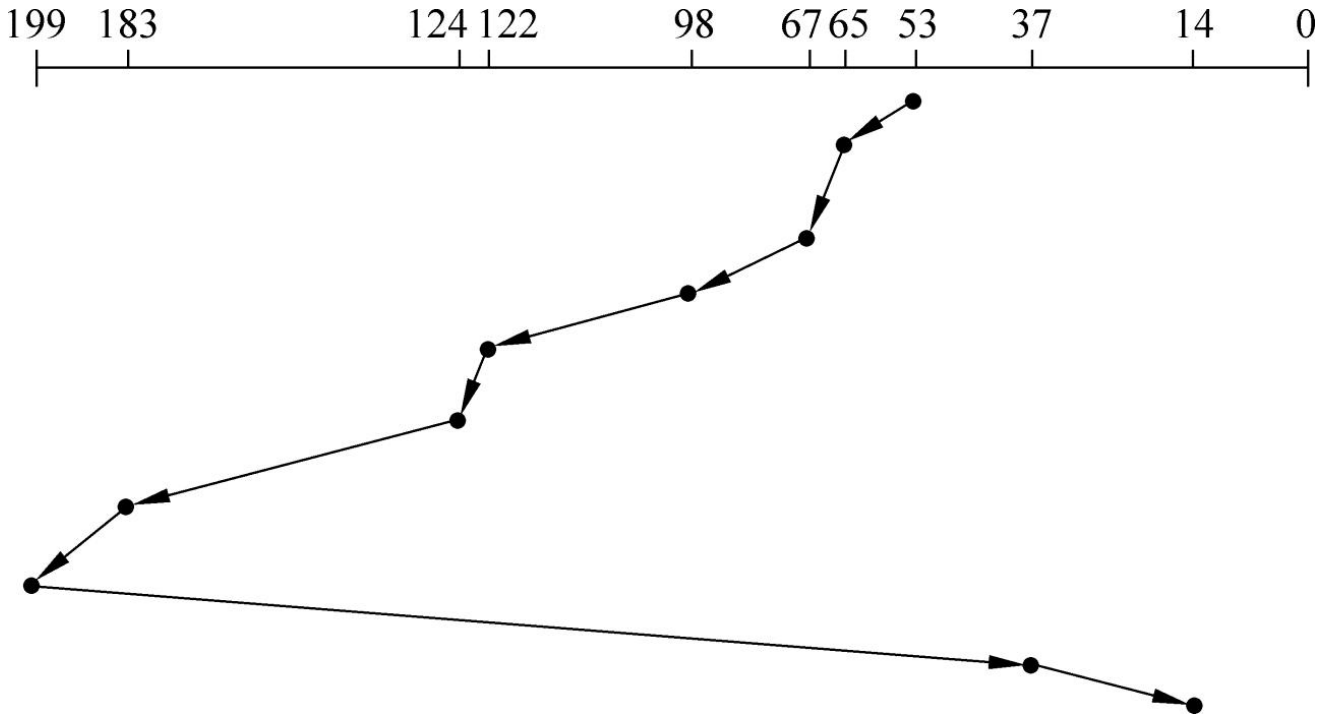


图6-29　双向扫描调度示意图

6.5.3 驱动调度-移臂调度

- （5）电梯调度算法

- “电梯调度”算法不仅考虑到请求访问者的磁头与当前磁头之间的距离，更优先考虑的是磁头当前的移动方向。总是从移动臂当前位置开始，沿着移动臂的移动方向选择距离当前移动臂最近的那个访问者进行调度，若沿移动臂的移动方向再无访问请求时，则改变移动臂的方向再选择，其目的是尽量减少移动臂移动时所花的时间。

6.5.3 驱动调度-移臂调度

- （5）电梯调度算法
- 由图6-30可以看出，若当前移动臂由里向外移动时，读写磁头共移动了208个柱面的距离，若当前移动臂由外向里移动时，则读写磁头共移动了299个柱面的距离。

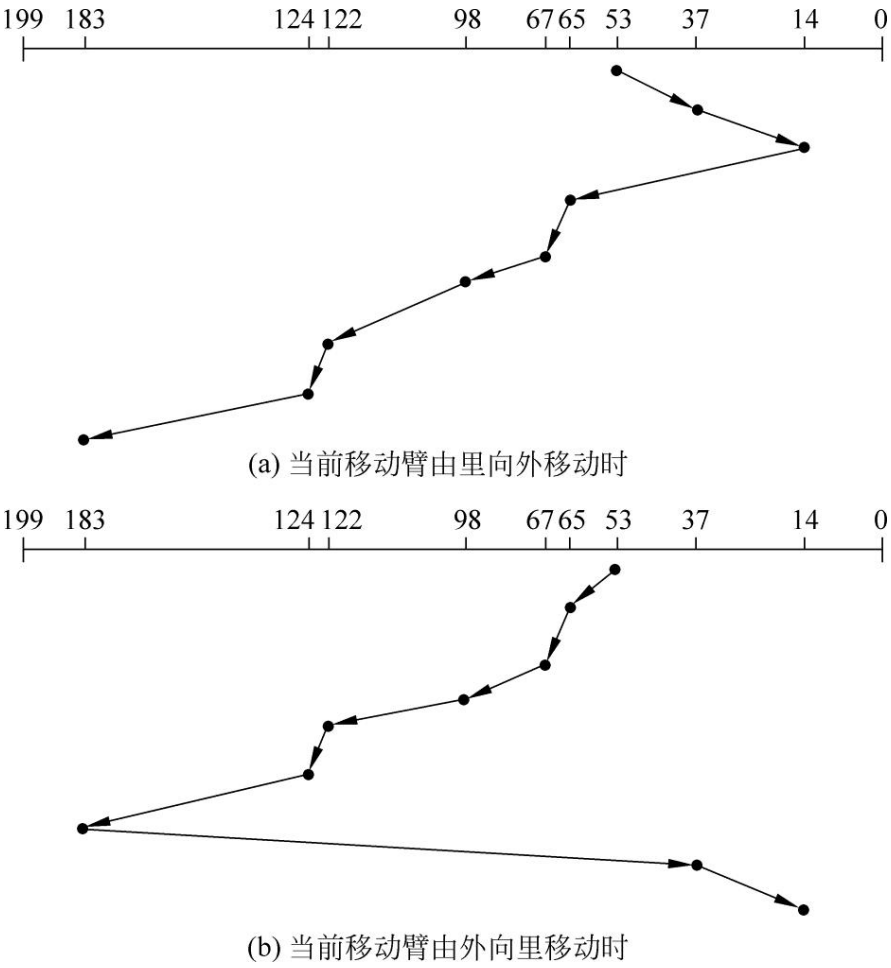


图6-30　电梯调度示意图

6.5.3 驱动调度

- “电梯调度”与“最短寻找时间优先”都是以尽量减少移动臂移动时所花的时间为目标
- “最短寻找时间优先”不考虑移动臂的当前移动方向，总是选择距离当前读/写磁头最近的那个柱面的访问者，可能会导致某个进程发生“饥饿”现象，移动臂来回改变移动方向；
- “电梯调度”算法沿着移动臂的移动方向选择距离当前读/写磁头最近的那个柱面的访问者，仅当沿着移动臂的移动方向无等待访问者时，才改变移动臂的方向。速度相对较慢。
- 电梯调度算法是一种简单、实用且高效的调度算法，能获得较好的寻道性能，又能防止“饥饿”现象，但是实现时需要增加开销，除了要记住读写磁头的当前位置外，还必须记住移动臂的移动方向，被广泛应用于大、中、小型计算机和网络的磁盘调度。

6.5.3 驱动调度-旋转调度

- 在一次移臂调度将移动臂定位到某一柱面后，允许进行多次旋转调度。旋转调度是指选择延迟时间最短的请求访问者执行的调度策略。

- 进行旋转调度时应分析下列情况：
  ① 若干等待访问者请求访问同一磁道上的不同扇区；
  ② 若干等待访问者请求访问不同磁道上的不同编号的扇区；
  ③ 若干等待访问者请求访问不同磁道上具有相同编号的扇区。

6.5.3 驱动调度-旋转调度

- 对于前两种情况，旋转调度总是让首先到达读/写磁头位置下的扇区先进行传送操作。
- 对于第三种情况，这些扇区同时到达读/写磁头位置下，旋转调度可任意选择一个读/写磁头进行传送操作。
- 【例】有4个访问请求者，他们的访问要求如下表示，可能的执行顺序：1-2-4-3或1-3-4-2。

请求次序	柱面号	磁头号	扇区号
1	5	4	1
2	5	1	5
3	5	4	5
4	5	2	8

6.5.3 驱动调度-旋转调度

- **影响I/O操作时间的因素**
- **记录在磁道上的排列方式会影响I/O操作的时间。**
- **【例】某系统，在对磁盘初始化时，把每个盘面分成8个扇区，有8个逻辑记录被存放在同一个磁道上供处理程序使用。处理程序要求顺序处理这8个记录，每次请求从磁盘上读一个记录，然后对读出的记录要花5ms的时间进行处理，以后再读下一个记录进行处理，直至8个记录全部处理结束。假定磁盘的转速为20ms/周，现把这8个逻辑记录依次存放在磁道上。**

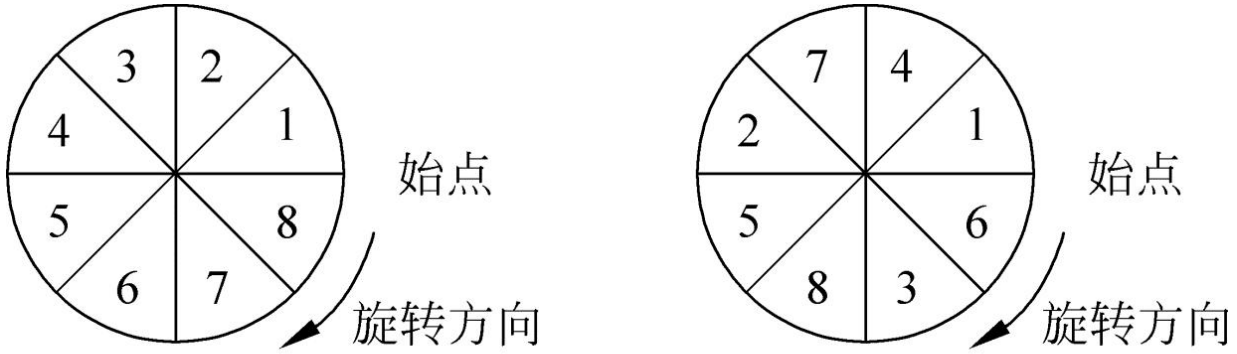


图6-31　记录的优化分布

6.5.3 驱动调度-旋转调度

- 在不知道当前磁头位置的情况下（寻道时间为10ms，旋转延迟时间为15ms），处理这8个记录所要花费的时间为：
  - 8×(2.5+5)+10+7×15=175(ms)

- 如果把这8个逻辑记录在磁道上的位置重新安排一下，当读出一个记录并处理后，读/写磁头正好位于顺序处理的下一个记录位置，可立即读出该记录，不必花费等待延迟时间。于是，处理这8个记录所要花费的时间为：
  - 10+8×(2.5+5)=70(ms)

6.5.3 驱动调度-旋转调度

- 记录的优化分布有利于减少延迟时间，从而缩短了输入/输出操作的时间。因此，对于一些能预知处理要求的信息采用优化分布可以提高系统的效率。
- 扇区的编号方式也会影响I/O操作的时间。**一般常将盘面扇区交替编号，磁盘组中不同盘面错开命名。**

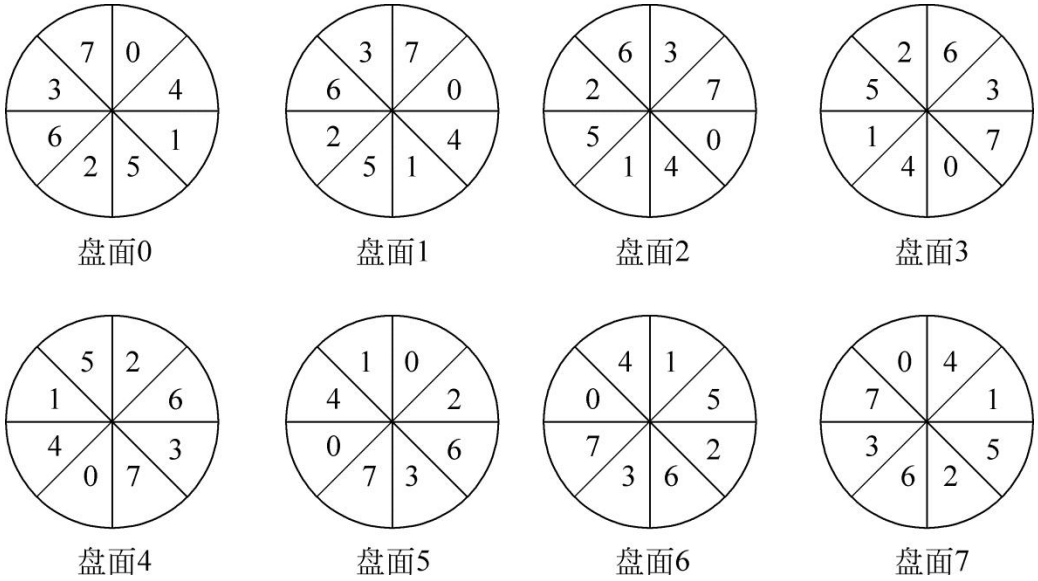
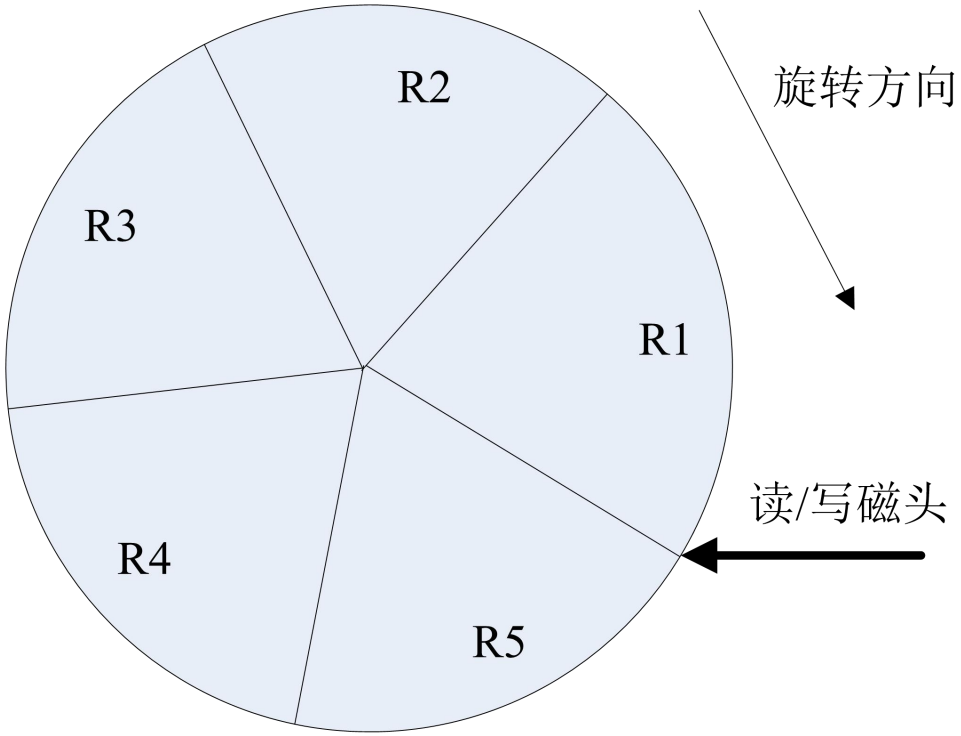


图6-32 磁盘组扇区编号

磁盘优化示例

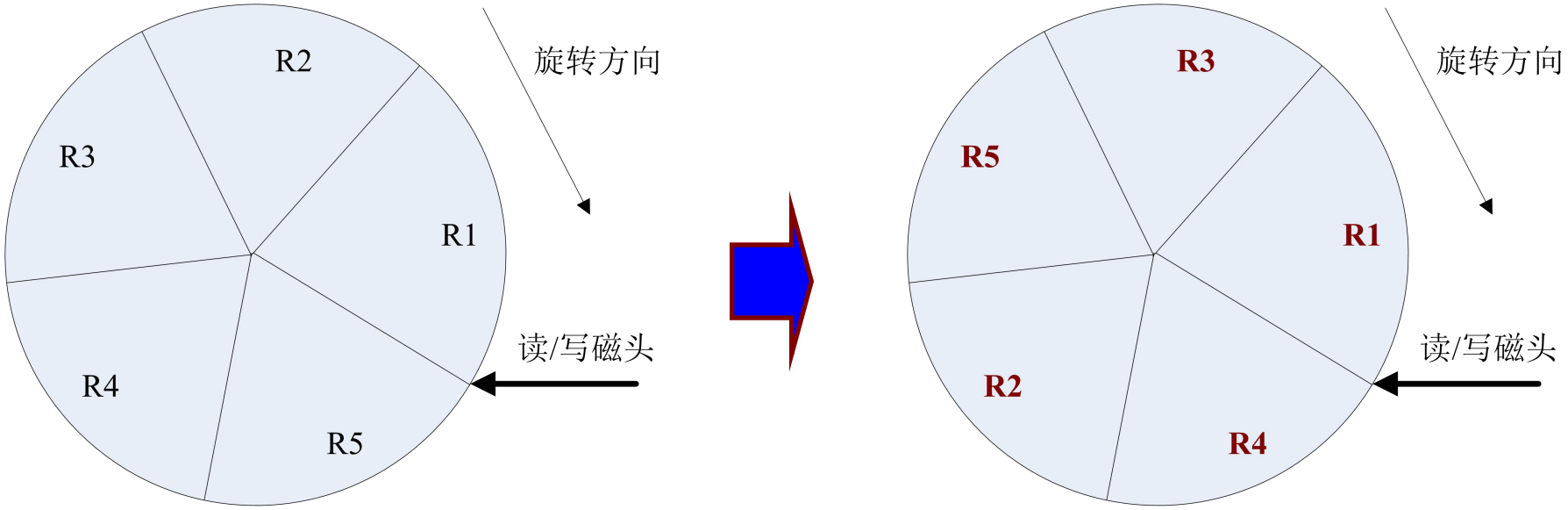
- 某磁盘磁道分成5个扇区(0～4)，每个扇区存放一个逻辑记录。一个用户文件有5个记录：R1~R5，被顺序存放在一个磁道上。假定磁盘旋转一周的时间是 20ms，每个记录读出后需6ms的时间处理。假设寻道时间为0ms，试问：(1)顺序读出5个记录并进行处理，共需多少时间？(2)给出一种在磁盘上安排记录的策略，使整个时间尽可能少。

由于每处理一个记录需要6ms，磁盘将转过1.5个扇区。顺序读取和处理5个记录，共需时间：4+(4+20)×4+6=106ms。



由于每处理一个记录需要6ms, 磁盘将转过1.5个扇区。为使时间减少，应减少旋转延迟，即把下一个要读取的记录放在处理完前一个记录后距磁头最近之处。读取和处理5个记录共需时间：

(4+6+2)×4+(4+6)=58ms



6.5.4 提高磁盘I/O速度的方法

- 磁盘I/O的速度已经成为计算机系统的瓶颈。为提高磁盘I/O的速度通常为磁盘设置高速缓存，它能显著减少等待磁盘I/O的时间。
- 磁盘高速缓存并非主存和CPU之间增设的一个小容量高速存储器，而是指利用主存中的存储空间，暂时存放从磁盘中读出的一系列盘块中的信息。
- 高速缓存是一组在逻辑上属于磁盘，物理上是驻留在主存的盘块。高速缓存在主存中可分为两种形式，第一种是在主存中开辟一个单独的存储空间来作为磁盘高速缓冲，其大小是固定的，不容易受到应用程序的影响；第二种是把当前所有未利用的主存空间作为一个缓冲池，供请求分页系统和磁盘I/O时共享，这种情况下缓存的大小显然不再是固定的。

6.5.4 提高磁盘I/O速度的方法

- **提前读**：在读当前盘块时，提前把下一个盘块数据也读入磁盘缓冲区。
- **延迟写**：执行写操作时，缓冲区的数据不立即写回磁盘，以备本进程或其他进程访问。
- **虚拟盘**：用主存空间去仿真磁盘，又称RAM盘。

6.6 设备处理

- 具有通道结构的计算机从启动外围设备到完成I/O操作，没有考虑不同类型物理设备的特性，采用统一的方法进行处理。这种不考虑具体特性（实际上设备特性已隐含在通道程序中)的处理方法称为设备处理的一致性。
- 采用设备处理一致性技术使得输入/输出操作既简单又不易出错。

6.6 设备处理

- 设备处理程序是输入/输出进程与设备控制器之间的通信和转换程序，它驱动物理设备和DMA控制器或I/O控制器等直接进行输入/输出操作。
- 它将输入/输出请求转换后，发送给设备控制器，启动设备执行，同时将设备控制器中记录的设备状态和输入/输出操作完成的情况传送给输入/输出的请求者，起着上传下达的作用；由于设备驱动程序与输入/输出设备的硬件特性密切相关，因此对不同类型的设备需要配置不同的驱动程序，而设备驱动程序中的一部分必须用汇编语言书写；此外驱动程序与输入/输出设备所采用的I/O控制方式紧密相关，在不同的I/O控制方式下，驱动程序启动设备以及中断处理的方式也不同。

6.6 设备处理

- 为了实现I/O进程与设备控制器之间的通信，设备驱动程序应具有以下功能。
  - 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。
  - ⑤ 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。

6.6.1 设备驱动程序的处理过程

- 设备驱动程序的处理过程如下。
- (1)预置设备
- 系统在初始或启动设备传输时，预置设备的初始状态，其中包括以下几步。
  - ① 将上层软件对设备的抽象要求转换为具体要求。
  - ② 检查I/O请求的合法性。
  - ③ 读出和检查设备的状态。
  - ④ 传送必要的参数。
  - ⑤ 工作方式的设置。

6.6.1 设备驱动程序的处理过程

- **(2)启动I/O设备**
- 在完成预置工作后，设备驱动程序可以向控制器中的命令寄存器传送相应的控制命令，负责启动设备的传送。对于具有通道的I/O系统，还将形成通道指令，并启动相应通道。

- **(3)设备中断处理**
- 负责处理设备(或通道)发出的各种中断，如一次I/O完成时的结束中断，I/O传输过程中的故障中断等。

6.6.2 设备的中断处理

- 设备中断是外围设备（通道）和中央处理器协调工作的一种手段。
- 设备（通道）借助I/O中断请求中央处理器进行干预，中央处理器根据产生的I/O中断事件了解输入/输出操作的执行情况。I/O中断事件或由于设备（通道）工作引起，或由外界的原因产生。
- 对于不同的中断事件，操作系统采用不同的处理方法。

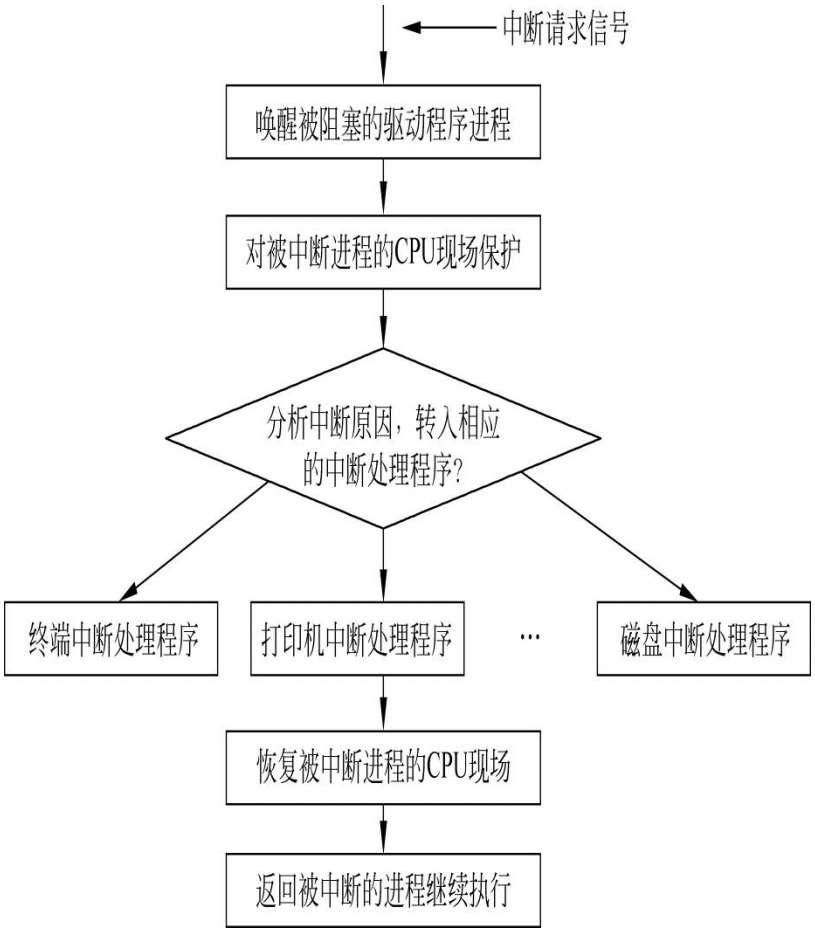


图6-33　中断处理流程

6.6.2 设备的中断处理

- **1. 操作正常结束**
  - 当通道状态字（CSW）中有通道结束、控制器结束和设备结束时，表示已完成一次I/O操作，形成输入/输出操作正常结束的中断事件。
- **2. 操作异常结束**
  - 当在I/O传输过程中出现设备故障或设备特殊情况时，形成操作异常结束的I/O中断事件。

① 设备故障

② 设备特殊情况

6.7 虚拟设备

- 对独占型设备采用静态分配方式，往往不能充分利用设备，不利于提高系统效率。具体表现在
  - ① 占有独占设备的作业只有一部分时间在使用它们，在其余时间里这些设备处于空闲状态。在设备空闲时不允许其他作业去使用它们，因此，不能有效地利用这些设备。
  - ② 当系统只配有一台独占设备时，就不能接受两个以上要求使用独占设备的作业同时执行，不利于多道并行工作。
  - ③ 这些独占设备大都是低速设备，在作业执行中往往由于等待这些设备的信息传输而延长了作业的执行时间。

- 为此，现代操作系统提供了虚拟设备的功能来解决这些问题。

6.7.1 脱机外围设备操作

- 早期采用脱机外围设备操作技术来解决上述系统效率不高的问题。使用两台外围计算机和一台主计算机，其中一台外围计算机专门负责把一批作业信息从读卡机上读取并记录到输入磁盘上，然后，把含有输入信息的输入磁盘人工地移动到主处理器上，在多道程序环境下，每个作业执行时不再启动输入机读取信息，而是让作业从磁盘上读取各自的信息，把作业运行的结果写入到输出磁盘上，最后把存有输出结果的输出磁盘移动到另一台外围计算机上打印输出。

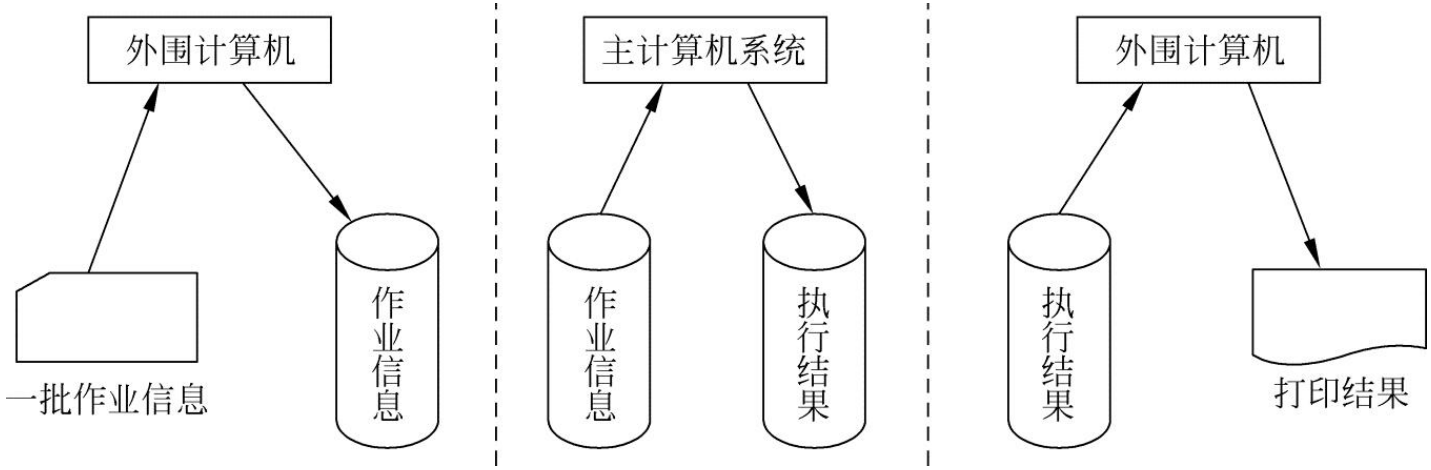


图6-34 脱机外围设备操作

6.7.1 脱机外围设备操作

- 两台外围计算机并不进行计算，只是将低速I/O设备上的数据从一台外围计算机传送到高速磁盘上，或者相反，而这种操作是独立于主计算机的，不在主计算机的直接控制下进行，因此称为“脱机外围设备操作”。

- 由于脱机外围设备操作中，主计算机只是专门进行计算，而与慢速外围设备打交道的工作交由外围计算机负责，在一定程度上提高了主计算机的效率，但使用多台外围计算机，增加了系统的成本；在主计算机与外围计算机之间来回移动磁盘，增加了操作人员的手工操作，既费时又增加了出错的机会，增加了每个作业的周转时间。

6.7.2 联机同时外围设备操作

- 现代计算机系统有足够的功能和大容量的磁盘，具有中央处理器与通道的并行工作能力，可以在执行计算的同时进行联机外围操作。事实上， 当系统中引入了多道程序技术后，完全可以利用其中的一道程序，来模拟脱机输入时的外围计算机功能，把低速I/O设备上的数据传送到高速磁盘上；再用另一道程序来模拟脱机输出时外围计算机的功能，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，我们把这种在联机情况下实现的同时外围设备操作称为SPOOLing，或称为假脱机操作。

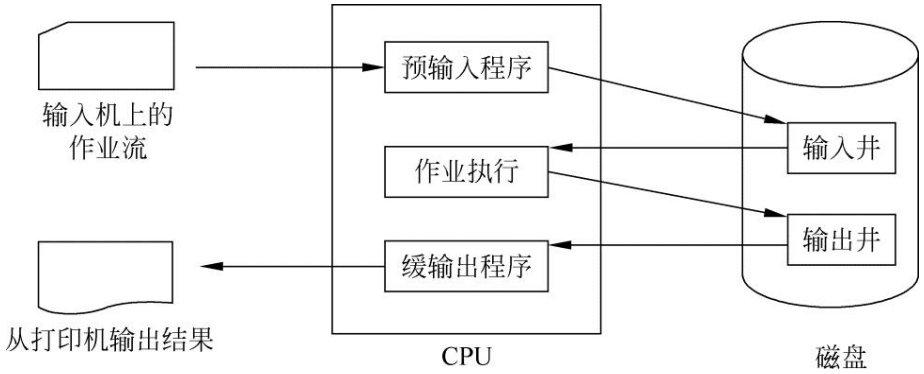


图6-35 联机同时外围设备操作

6.7.2 联机同时外围设备操作

- SPOOLing技术是对脱机输入、输出系统的模拟。它必须建立在具有多道程序功能的操作系统上，需要有高速的、大容量的随机存储器支持。
- 在磁盘上划出专用存储空间，称为“井”，用以存放作业的初始信息和执行结果。为了便于管理把“井”分为“输入井”和“输出井”。“输入井”中存入作业的初始信息，“输出井”中存放作业的执行结果。
- 操作系统中实现联机同时外围设备操作功能的部分也称为SPOOLing系统，SPOOLing系统主要由三部分程序组成，即“预输入”程序、实现输入井读和输出井写的“井管理”程序和“缓输出”程序。

6.7.2 联机同时外围设备操作

- （1）预输入程序
- 把一批作业组织在一起形成作业流，由预输入程序把作业流中每个作业的初始信息由输入设备输入到输入井保存，并填写好输入表以便在作业执行中要求输入信息时，可以随时找到它们的存放位置，以备作业调度。
- 系统拥有一张作业表用来登记进入系统的所有作业的作业名、状态、预输入表位置等信息。每个用户作业拥有一张预输入表用来登记该作业的各个文件的情况，包括设备类、信息长度及存放位置等。

- 输入井中的作业有四种状态：输入状态；收容收态；执行状态；完成状态。

6.7.2 联机同时外围设备操作

- （2）井管理程序
- 作业执行过程中要求启动输入机(或打印机)读文件信息(或输出结果)时，操作系统根据作业请求，调出井管理程序工作，不必再启动I/O设备，而转换成从磁盘的输入井读信息(或把结果写入输入井)。

- 井管理程序包括井管理读程序和井管理写程序两部分。
- 当作业请求从输入机上读文件信息时，就把任务转交给井管理读程序，从输入井中读出信息供用户使用。
- 当作业请求从打印机上输出结果时，就把任务转交给井管理写程序，把产生的结果保存到“输出井”中。

6.7.2 联机同时外围设备操作

- （3）缓输出程序
- 缓输出程序负责查看输出井中是否有等待输出的结果信息，若有，则启动打印机把作业的结果文件打印输出。当一个作业的文件信息输出完毕后，将它占用的井区回收以供其他作业使用。
- SPOOLing系统提高了I/O速度，缓和了CPU与低速的I/O设备之间速度不匹配的矛盾；增加了多道程序的道数，增加了作业调度的灵活性，将独占型设备改造为共享型设备。宏观上，虽然十多个进程在同时使用一台独占型设备，而对每一个进程而言，它们都认为自己独占了一个设备，实现了将独占型设备变换为若干台对应的逻辑设备的功能，即实现了虚拟设备功能。

6.7.3 Spooling应用例子

- Spooling是在多道程序系统中处理独占设备的一种方法，这种技术已经被广泛应用于许多设备和场合：早期的读卡机、穿卡机；现在的打印机、网络等。
- 打印机Spooling守护进程
  - 禁止用户直接使用打印机设备文件；创建特殊进程-守护进程，以及一个特殊的目录-SPOOLing目录，存放待打印文件。
- 网络通信Spooling守护进程
  - 网络上传文件时，使用网络守护进程，发送文件前，先将其放在特定目录下，然后由守护进程将其取出并发送出去。

6.9 本章小结

- 设备管理的功能主要有：外围设备的分配和去配，外围设备的启动，磁盘的驱动调度，设备处理以及虚拟设备。

- 按照I/O控制功能的强弱以及和CPU之间联系方式的不同，可以把I/O控制方式分为四类：直接程序控制方式、中断驱动控制方式、直接存储器访问(DMA)控制方式和通道控制方式。其中通道具有执行I/O指令的能力，并通过执行通道程序来控制I/O操作，完成主存和外围设备之间的信息传送。通道技术实现了外围设备与CPU之间、通道与通道之间以及各个通道上外围设备之间的并行操作，提高了整个系统效率。

6.9 本章小结

- 为了缓解CPU与外围设备之间速度不匹配和负载不均衡的矛盾，提高CPU和外围设备的工作效率和系统中各部件的并行工作程度，现代操作系统普遍采用缓冲技术。常见的缓冲机制有单缓冲机制、能实现双向同时传送数据的双缓冲机制以及能供多个设备同时使用的公共缓冲机制等。

6.9 本章小结

- 现代计算机系统具有设备的独立性，使得设备分配灵活性强，适应性强，易于实现I/O重定向。独占型设备往往采用静态分配方式。系统通过设置设备控制表、控制器控制表、通道控制表和系统设备表等数据结构，记录相应设备或控制器的状态以及对设备或控制器进行控制所需要的信息实现设备的分配。共享型设备的分配则更多地采用动态分配方式。磁盘属于共享型设备，启动磁盘完成一次I/O操作所花费的时间包括：寻找时间、延迟时间和传送时间。移臂调度的目标是尽可能地减少I/O操作的寻找时间。常用的移臂调度算法有先来先服务调度算法、最短寻道时间优先调度算法、电梯调度算法、单向扫描算法和双向扫描算法等。旋转调度是指选择延迟时间最短的请求访问者执行的调度策略。记录在磁道上的排列方式、盘组中扇区的编号方式等都会影响I/O操作的时间。通过优化记录的分布，交错编排盘面扇区号等方式可以达到减少延迟时间的目的。

6.9 本章小结

- 设备驱动程序中包括了所有与设备相关的代码，它把用户提交的逻辑I/O请求转化为物理I/O操作的启动和执行，如设备名转化为端口地址、逻辑记录转化为物理记录、逻辑操作转化为物理操作等，它对其上层的软件屏蔽所有硬件细节，并向用户层软件提供一个一致性的接口，如设备命名、设备保护、缓冲管理、存储块分配等。
- 为了提高独占设备的使用效率，创造多道并行工作环境，在中断和通道硬件的支持下，操作系统采用多道程序设计技术合理分配和调度各种资源，实现联机同时外围设备操作。SPOOLing技术将一个物理设备虚拟成多个虚拟(逻辑)设备，用共享型设备模拟独占型设备，实现了虚拟设备功能。SPOOLing系统主要由三部分组成，即“预输入”程序、“井管理”程序和“缓输出”程序。