第5章 设备管理

知识要点

- 掌握
  - I/O硬件原理：I/O控制方式；
  - I/O硬件原理：I/O软件层次和设备独立性；
  - I/O调度和磁盘驱动调度算法；
  - 虚拟设备的原理。
- 了解
  - 缓冲技术、缓冲区高速缓存。

第五章 设备管理

- 5.1 I/O硬件原理
- 5.2 I/O软件原理
- 5.3 驱动调度技术
- 5.4 设备分配
- 5.5 虚拟设备

设备管理的目标

- 提高设备的利用率
  - 提高CPU与I/O设备之间的并行操作程度，主要利用：中断技术、DMA技术、通道技术、缓冲技术。
- 方便、统一的操作界面
  - 方便：用户能独立于具体设备的复杂物理特性之外而方便地使用设备
  - 统一：指对不同的设备尽量使用统一的操作方式，例如各种字符设备用一种I/O操作方式。这就要求用户操作的是简便的逻辑设备，而具体的I/O物理设备由操作系统去实现，这种性能常常被称为设备的独立性。

设备管理功能

- 设备分配
  - 按照一定的算法把I/O设备、及其相应的设备控制器和通道分配给用户（进程），对未能分配到设备的进程，将其插入等待队列中。
- 缓冲区管理
  - 为解决CPU与I/O之间速度不匹配的矛盾，在它们之间配置了缓冲区。设备管理程序需要负责管理缓冲区的建立、分配和释放。
- 实现物理I/O设备的操作
  - 对具有通道的系统，生成专门的通道指令启动通道，对指定设备进行I/O操作，并能响应通道的中断请求。
  - 对未设置通道的系统，直接驱动设备进行I/O操作。

设备管理的主要工作

- 外围设备中断处理
- 缓冲区管理
- 外围设备的分配和去配
- 外围设备驱动调度
- 虚拟设备及其实现

5.1 I/O硬件原理

- I/O系统定义
- I/O控制方式
- 设备控制器

I/O系统

- I/O系统：I/O设备及其接口线路、控制部件、通道和管理软件的总称。
- I/O操作：计算机的内存和外围设备介质之间的信息传送操作。

I/O设备分类

- 按照I/O操作特性
  - 输入型设备、输出型设备、存储型设备。
- 按照I/O信息交换的单位
  - 字符设备、块设备。
- 输入型外围设备和输出型外围设备一般为字符设备，与内存进行信息交换的单位是字节。
- 存储型外围设备一般为块设备
  - 顺序存取存储设备：只有当前面的存储物理块被存取以后，才能存取后面的物理块，如磁带。
  - 直接存取存储设备：对任何一个物理块存取不必进行事先顺序搜索，如磁盘。存取任何一个物理块所需的时间几乎不依赖于此信息的位置。

设备的物理特性差异

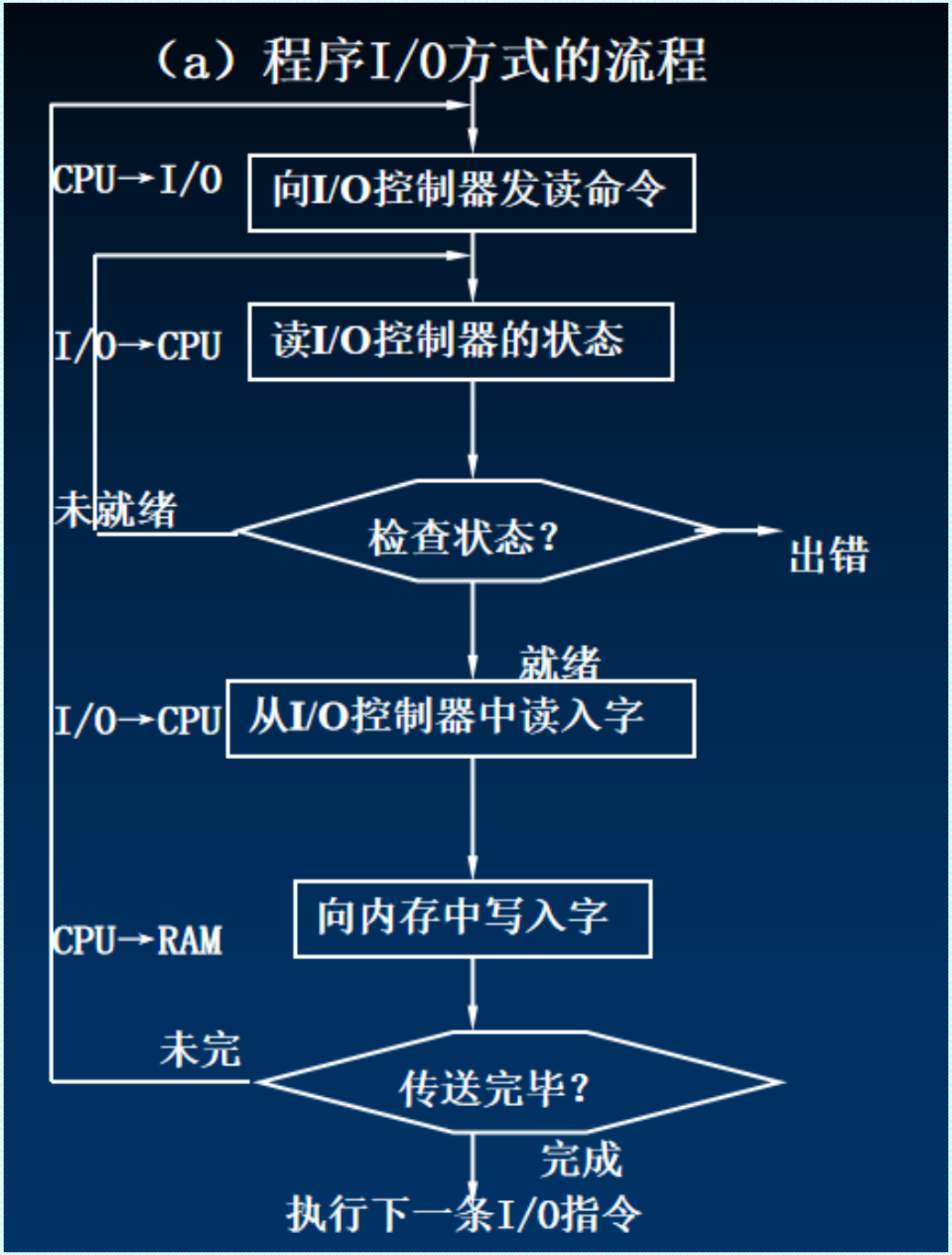
- 数据传输率
- 数据表示方式
- 传输单位
- 出错条件

5.1.2 I/O控制方式

- 按照I/O控制器功能的强弱，以及和CPU间联系方式的不同，对I/O设备的控制方式分类：
  - 轮询方式
  - 中断方式
  - DMA方式
  - 通道方式
- 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。

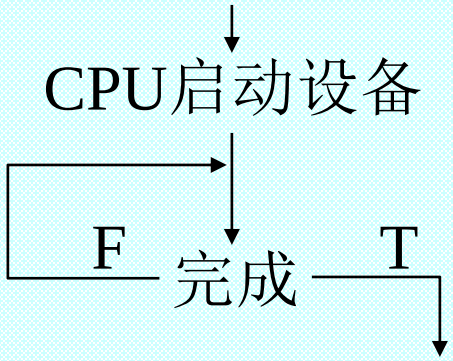
1. 轮询方式

- 程序I/O（Programmed I/O）方式或忙-等待方式。
- 早期的计算机系统中，没有中断机构，处理机对I/O设备直接进行控制。
- 使用查询指令测试设备控制器的忙闲状态位，决定内存和设备是否能交换数据。
- 轮询方式使用：
  - 1)查询指令：查询设备是否就绪；
  - 2)读/写指令：当设备就绪时，执行数据交换；
  - 3)转移指令：当设备未就绪时，执行转移指令转向查询指令继续查询。


（a）程序I/O方式的流程
CPU→I/O
向I/O控制器发读命令
I/O→CPU
读I/O控制器的状态
未就绪
检查状态?
出错
就绪
I/O→CPU
从I/O控制器中读入字
CPU→RAM
向内存中写入字
未完
传送完毕?
完成
执行下一条I/O指令

1. 轮询方式

- 几个设备同时要求I/O，可对每个设备都编写I/O数据处理程序，轮流查询这些设备的状态位，当某个设备准备好允许I/O数据时，就调用这个设备的I/O程序处理数据传输，否则依次轮询下个设备是否准备好。
- 特点：
  - 在外设进行数据处理时，CPU只能等待(忙等待），消耗大量处理机时间；
  - 处理机与设备串行工作；
  - 管理简单，可在要求不高的场合被采用。

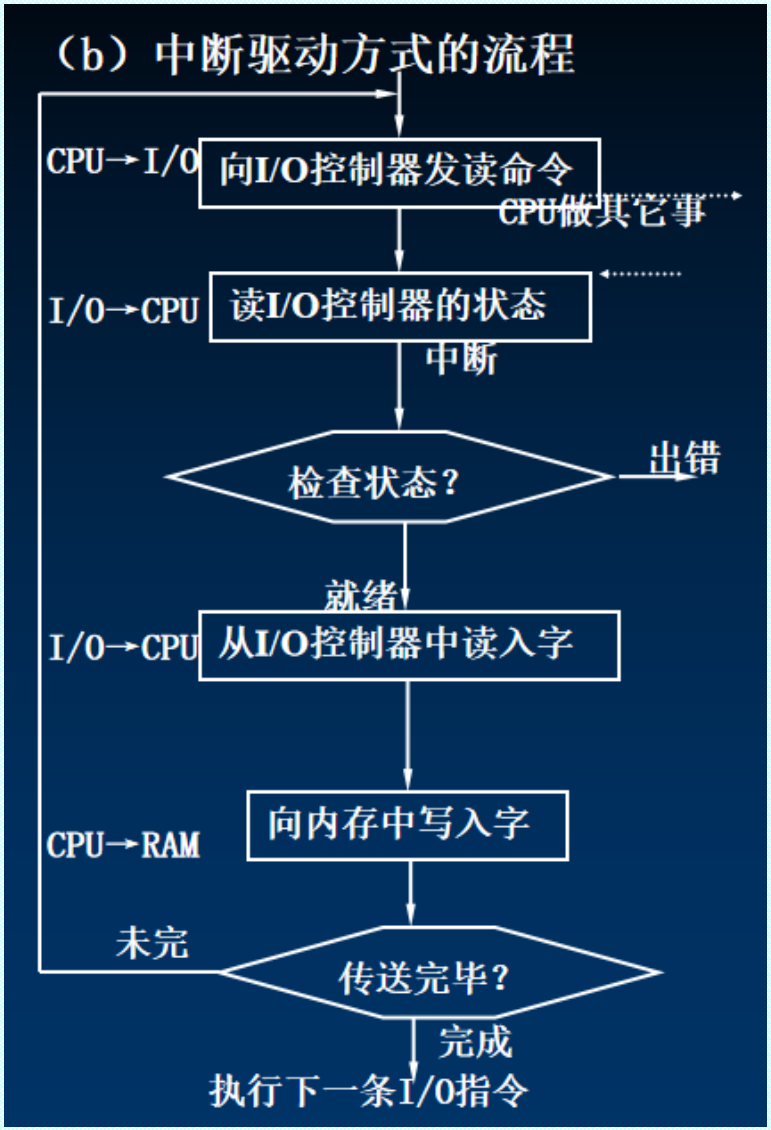


2. 中断方式

- 中断方式要求CPU与设备控制器及设备之间有中断请求线，控制器的状态寄存器有相应中断允许位。
- CPU与设备间数据传输过程：
  - 1)进程发出启动I/O指令，CPU加载控制信息到设备控制器的寄存器，然后，进程继续执行或放弃CPU等待设备操作完成；
  - 2)设备控制器检查状态寄存器，按照I/O指令要求，执行相应I/O操作，一旦传输完成，设备控制器通过中断请求线发出I/O中断信号；
  - 3)CPU收到并响应I/O中断后，转向处理该设备的I/O中断例程执行；
  - 4)中断处理例程执行数据读取操作，将I/O缓冲寄存器的内容写入内存，操作结束后退出中断处理程序，返回中断前的执行状态；
  - 5)进程调度程序在适当时刻恢复得到数据的进程执行。

2. 中断方式

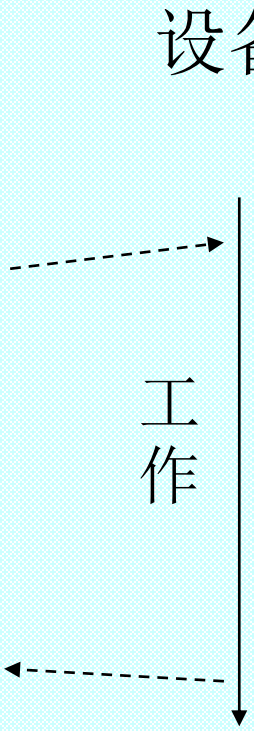
- 在现代计算机系统中，对I/O设备的控制，广泛地采用中断驱动方式。
- 即当某进程要启动某个I/O设备时，便由CPU向相应的设备控制器发出一条I/O命令，然后立即返回继续执行原来的任务。
- 设备控制器便按照该命令的要求去控制I/O设备。此时，CPU与I/O设备处于并行工作状态。



2. 中断方式

CPU

- 计算
- 启动设备
- 计算
- …
- 计算
- 中断处理
- 计算



- 优点：在外设进行数据处理时，CPU不必等待，可以继续执行该程序或其他程序。
- 缺点：每次I/O都要CPU的干预，CPU每次处理的数据量少（通常不超过几个字节），只适于数据传输率较低的设备。

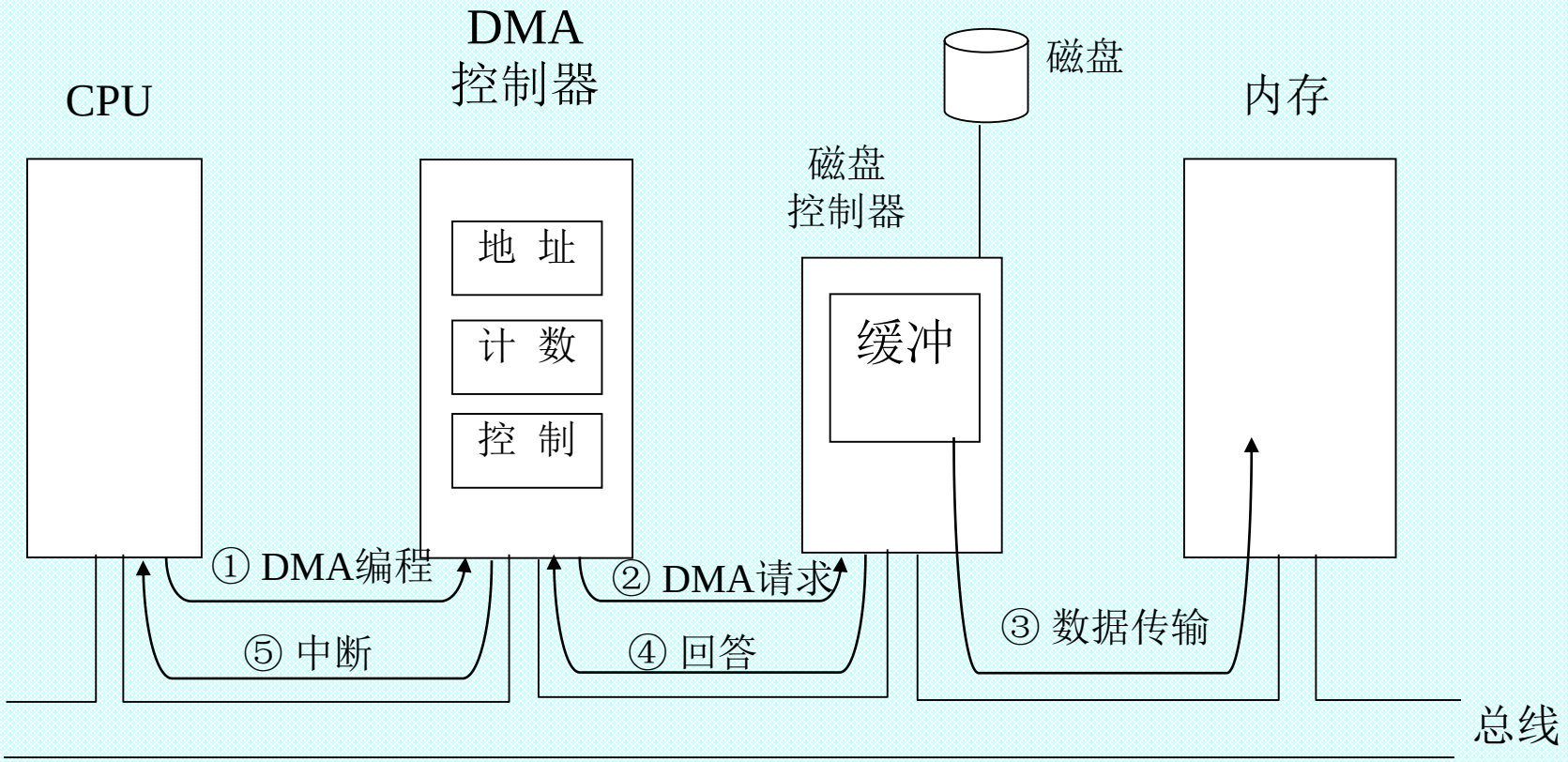
3. DMA方式

- 中断驱动方式以字（节）为单位进行数据传送，每完成一个字（节）的传送，控制器产生一次中断。系统需占用CPU进行现场的保存和和恢复。特别是高速直接存储设备的出现，中断处理开销过大，CPU利用率急剧下降。
- 如果I/O设备能直接与内存交换数据而不占用CPU，利用率还可提高，由此产生了直接存储器存取DMA方式。
- DMA方式需以下设施：
  - (1)内存地址寄存器
  - (2)字计数器
  - (3)数据缓冲寄存器或数据缓冲区
  - (4)设备地址寄存器
  - (5)中断机制和控制逻辑

3. DMA方式

- DMA控制器组成
  - DMA控制器与块设备的接口
  - I/O控制逻辑
  - 主机与DMA控制器的接口
    - 命令/状态寄存器CR：接收从CPU发来的I/O命令或有关的控制信息，或设备的状态。
    - 内存地址寄存器MAR：存放数据从设备传到内存的目的地址，或由内存到设备的内存源地址。
    - 数据寄存器DR：暂存设备和内存间交换的数据。
    - 数据计数器DC：存放本次CPU要读/写的字（节）数。

3. DMA方式


DMA
控制器
磁盘
CPU
内存
磁盘
控制器
地 址
计 数
控 制
缓冲
① DMA编程
② DMA请求
③ 数据传输
④ 回答
⑤ 中断
总线

3. DMA方式

- DMA工作过程
  - (1)CPU通过设置DMA控制器实现DMA编程，启动磁盘控制器并测试设备。
  - (2)DMA控制器向磁盘控制器发出读请求，并将内存地址放在地址总线上；
  - (3)磁盘控制器将字节传送到内存指定单元；
  - (4)磁盘控制器向DMA控制器发送应答；DMA控制器将内部地址寄存器加1同时将计数器减1；
  - (5)重复(2)-(4)，当计数器为0时，DMA控制器向CPU发出中断信号。

3. DMA方式

- DMA特点
  - DMA与内存间采用字传送，DMA与设备间可能是字位或字节传送，所以，DMA中还要设置数据移位寄存器、字节计数器等硬件逻辑。
  - DMA设有中断机制和DMA传输控制机制，若出现DMA与CPU同时经总线访问内存，CPU总把总线占有权让给DMA(称“周期窃用”)，让设备和内存之间交换数据，不再需要CPU干预，减轻CPU的负担，每次传送数据时，不必进入中断系统，能进一步提高CPU资源的利用率。
  - 优点：CPU只需干预I/O操作的开始和结束，而其中的一批数据读写无需CPU控制，适于高速设备。

3. DMA方式

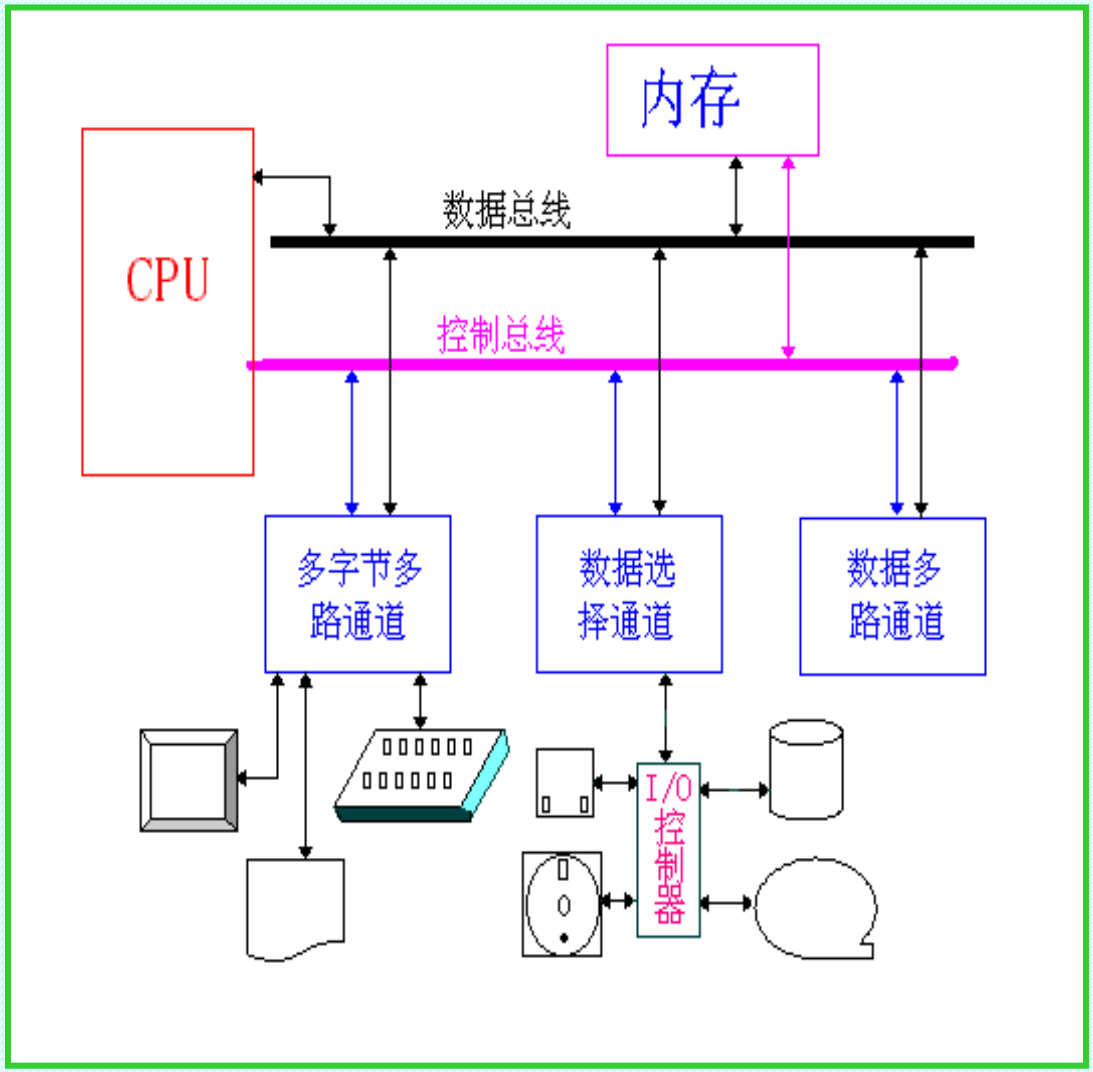
- DMA方式与中断方式比较
  - 数据传输的基本单位是一个连续的数据块。
  - 内存与设备之间直接数据传送，不用CPU的干预。
  - 仅在传送一个或多个数据块的开始和结束时，才需CPU干预，整块数据的传送是在控制器的控制下完成的。
- DMA方式较之中断驱动方式，减少了CPU对 I/O控制的干预，进一步提高了CPU与I/O设备的并发程度。

4. 通道方式

- DMA方式比中断驱动方式已显著地减少了CPU的干预，即由以字（节）为单位的干预减少到以数据块为单位的干预。但是CPU每发出一条I/O指令，只能读（或写）一个连续的数据块。当需要一次性读多个离散的数据块并将它们分别传送到不同的内存区域，或者相反时，则需要CPU分别发出多条I/O指令，通过多次中断处理才能完成。这种机制不能满足复杂的I/O操作要求。
- 为获得CPU和外围设备间更高的并行工作能力，让种类繁多、物理特性各异的外围设备能以标准的接口连接到系统中。在大、中型计算机系统中，普遍采用由专用的I/O处理机来接受CPU的委托，独立执行自己的通道程序来实现I/O设备与内存之间的信息交换，这就是通道技术。

4. 通道方式

- 通道：是计算机系统中专门用于I/O的处理机。
- 在大、中、小型计算机中一般称为通道，结构如图所示



4. 通道方式

- 通道程序
  - 通道程序由一系列通道指令（或称为通道命令）构成。
  - 通道指令与一般的机器指令不同，每条指令中包含的信息较多，包括操作码、内存地址、计数（读或写数据的字节数）、通道程序结束位P和记录结束标志R。
  - 通道通过执行通道程序，与设备控制器共同实现对I/O设备的控制。
- 采用通道后的I/O操作过程：
  - 用户执行I/O请求时，通过访管指令进入管理程序，由CPU通过管理程序组织一个通道程序，并启动通道。
  - 通道执行CPU为它组织的通道程序，完成指定的数据输入输出工作。此时，CPU可执行其他任务并与通道并行工作。
  - 通道程序结束后向CPU发中断请求。CPU响应这个中断请求后，调用管理程序对中断请求进行处理。

4. 通道方式

- 通道类型：
  - 由于外围设备的种类较多，且其传输速率相差很大，所以通道也具有多种类型。根据信息交换方式，可以把通道分成以下三种类型。

1. 字节多路通道（Byte Multiplexor Channel）

- **以字节为单位传送数据；**
- **通常含有多个（8，16，32）非分配型子通道，每一个子通道连接一台I/O设备。这些子通道按时间片轮转方式共享主通道。一个子通道完成一个字节的传送后，立即让出字节多路通道（主通道），给另一个子通道使用。**
- **适用于连接低速或中速设备，如打印机、终端等。**

4. 通道方式

- 通道类型：

2. 数组选择通道（Block Selector Channel）

- 以块（数组）为单位进行数据传输；
- 可以连接多台I/O设备，只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。即当某台设备一旦占用了该通道，就被它独占，直至该设备传送完毕释放该通道为止。
- 适于连接高速设备（如磁盘机、磁带机），但这种通道利用率较低。

4. 通道方式

- 通道类型：

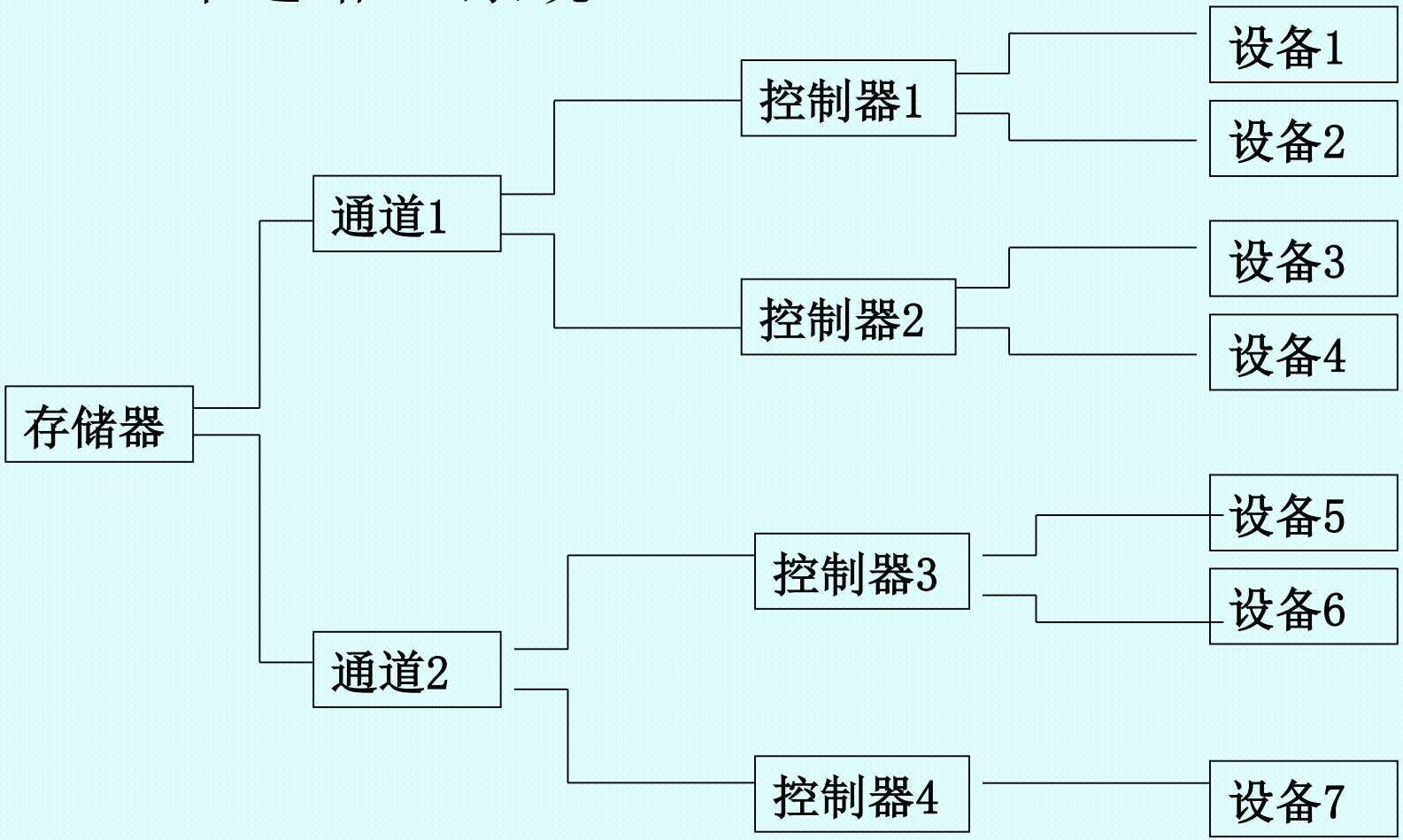
3.数组多路通道（Block Multiplexor Channel）

- 以块（数组）为单位进行数据传输；
- 具有多个非分配型子通道，可以连接多台高、中速的外围设备；
- 结合了数组选择通道的高传输速率和字节多路通道的分时并行操作优点；
- 具有数据传输速率高和分时操作不同设备的优点，能获得令人满意的通道利用率。

4. 通道方式

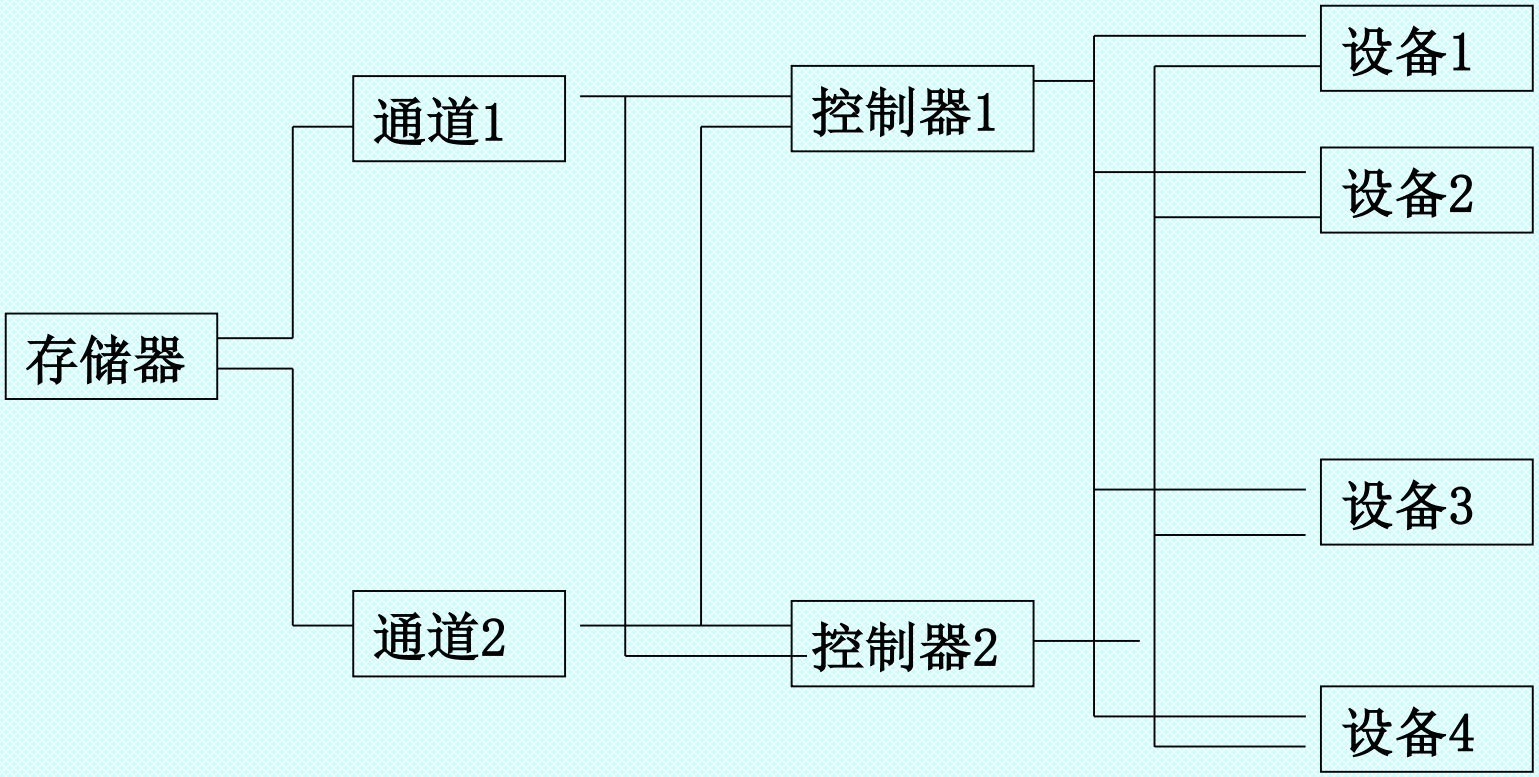
“瓶颈”问题

1. 单通路I/O系统



4. 通道方式

2. 多通路I/O系统



5.1.3设备控制器

- 设备控制器：I/O设备的电子部件，又称适配器，是可插入主板扩充槽的印刷电路板。
- 引入设备控制器的原因
  - 操作系统与控制器打交道，微机和小型机采用单总线模型，实现CPU和控制器间的数据传送，中、大型机则采用多总线结构和多通道方式，以提高并行操作程度。
  - 如果没有控制器，复杂操作必须由操作系统来解决，引入控制器后，通过传递简单参数就可进行I/O操作，大大简化系统的设计，有利于计算机系统对各类控制器和设备的兼容性。

设备控制器

传统的设备=机械部分+电子部分

将电子部分从设备中分离出来作为一个独立部件，就是控制器。

设备控制器

扩展插槽	
显卡插槽	2条PCI-E 16X
PCI 插槽	3条PCI 插槽,2条PCI-E 1X
IDE 插槽	一个IDE插槽
FDD 插槽	一个FDD，接软驱
SATA接口	6个SATAII接口,支持RAID 0,1
I/O接口	
USB 接口	10个USB 接口
其他内部插口	IEEE 1394接口 ESATA接口
外接端口	音频接口

设备控制器功能和结构(1)

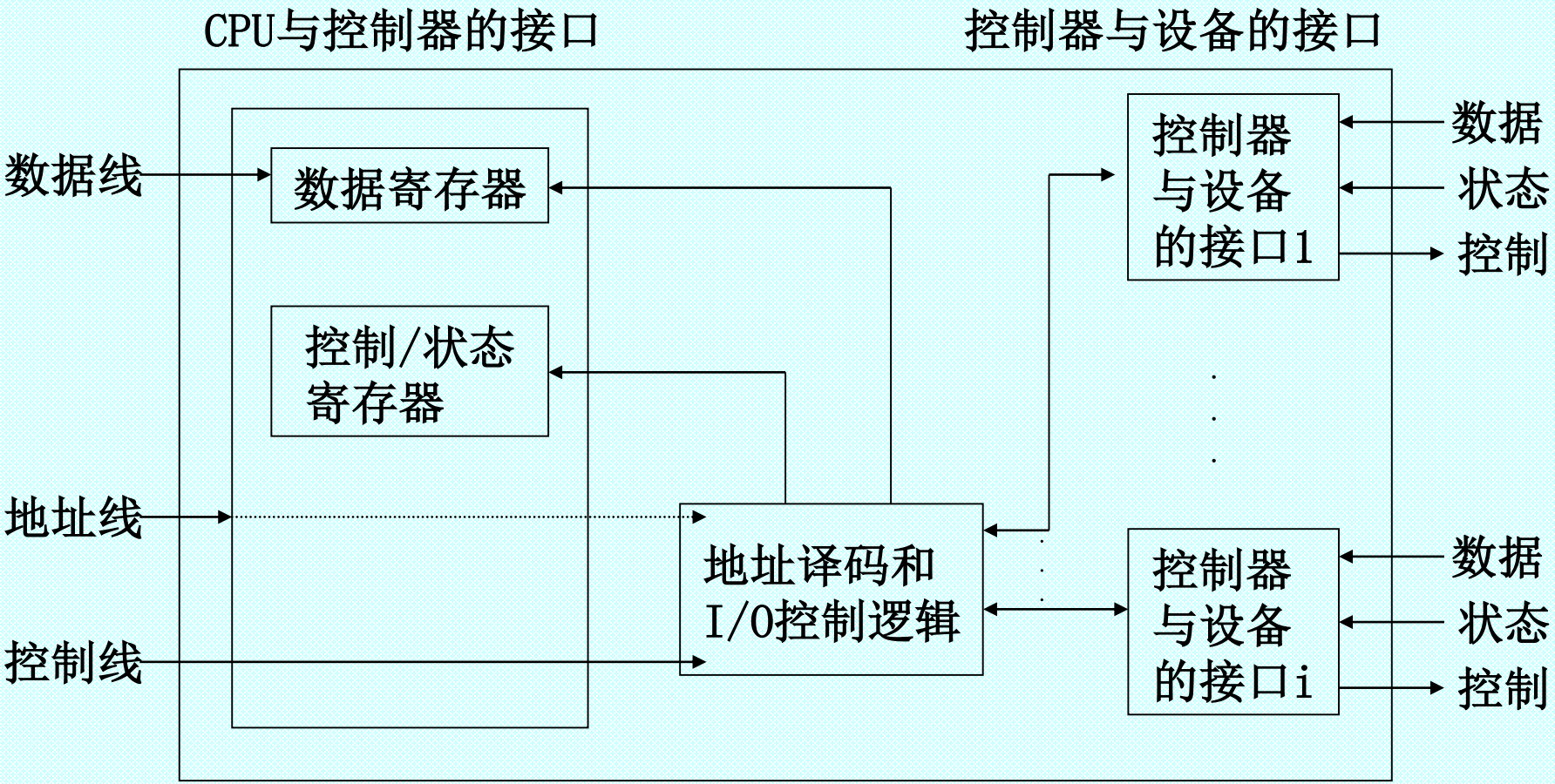
- 设备控制器是CPU和设备之间的接口，负责接收从CPU发来的命令，控制I/O设备操作，实现内存和设备之间的数据传输。
- 设备控制器是一个可编址设备，当它连接多台设备时，则应具有多个设备地址。

设备控制器功能和结构(2)

- 设备控制器主要功能：
  - 接收和识别CPU或通道发来的命令
  - 实现数据交换,包括设备和控制器间的数据传输
  - 发现和记录设备及自身的状态信息，供CPU处理
  - 设备地址识别
- 设备控制器组成
  - 命令寄存器及译码器
  - 数据寄存器
  - 状态寄存器
  - 地址译码器

设备控制器

设备控制器的组成



设备控制器组成

- 设备控制器与处理机的接口
- 设备控制器与设备的接口
- I/O逻辑

设备控制器

设备与控制器之间的接口

- 数据信号
  - 输入：设备—>控制器
  - 输出：控制器—>设备
- 控制信号

  读、写或执行等操作信号
- 状态信号

  指示设备的当前状态

5.2 I/O软件原理

- 5.2.1 I/O软件设计目标和原则
- 5.2.2 I/O中断处理程序
- 5.2.3 I/O设备驱动程序
- 5.2.4 独立于设备的操作系统I/O软件
- 5.2.5 用户空间的I/O软件

5.2.1 I/O软件设计目标和原则

- I/O软件总体设计目标:
  - 高效率。
  - 通用性。
- I/O软件总体设计要考虑的问题：
  - 设备无关性：程序与具体物理设备无关。
  - 出错处理：屏蔽错误，不让高层软件感知。
  - 同步/异步传输：支持同步（阻塞）和异步（中断驱动）两种工作方式。
  - 缓冲技术：数据处理速率不匹配，数据处理大小不一致。
  - 独占性外围设备和共享性外围设备：分配和共享。

I/O软件组织成四个层次

- I/O中断处理程序。
- I/O设备驱动程序。
- 独立于设备的操作系统I/O软件。
- 用户空间的I/O软件。

5.2.2 I/O中断处理程序

- I/O中断的类型和功能
  - 通知用户程序I/O操作沿链推进程度
  - 通知用户程序I/O操作正常结束
  - 通知用户程序发现的I/O操作异常
  - 通知程序外围设备上重要的异步信号
- I/O中断的处理原则
  - 操作正常结束：唤醒等待进程，并置为就绪态。
  - 操作发生故障或特殊事件的中断处理
  - 人为要求而产生的中断处理
  - 外围设备的异步信号处理

5.2.3 设备驱动程序(1)

- 设备驱动程序：I/O 系统高层与设备控制器之间的通信程序，是与设备密切相关的代码。
- 主要任务：
  - 接收用户提交的逻辑I/O请求，转化为物理I/O操作，发送给设备控制器，启动设备去执行。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。
  - 将设备控制器发来的信号传给上层软件。

5.2.3 设备驱动程序(2)

- 设备驱动程序主要功能：
  - 设备初始化：检查并预置设备和控制器以及通道的状态。
  - 执行设备驱动例程：启动设备，进行数据传输；生成通道指令和通道程序，启动通道工作。
  - 执行中断处理例程：响应设备、控制器和通道发出的中断请求，调用相应的中断处理程序进行处理。

5.2.3 设备驱动程序(3)

- 设备驱动程序特点
  - 是设备无关软件和设备控制器之间通信和转换程序。
  - 与设备控制器和I/O设备的硬件特性紧密相关。不同类型的设备应配置不同的设备驱动程序。
  - 与I/O设备所采用的I/O控制方式紧密相关。
  - 由于驱动程序与硬件紧密相关，因而其中的一部分必须用汇编语言编写。
  - 驱动程序应允许可重入，一个正在运行的驱动程序常会在一次调用完成前被再次调用。

设备驱动程序（UNIX）

在UNIX系统中，每类设备都有一个驱动程序，用它来控制该类设备。任何一个驱动程序通常都包含了用于执行不同操作的多个函数，如打开、关闭、启动设备、读和写等函数。为使核心能方便地转向各函数，系统为每类设备提供了一个设备开关表，其中有该类设备的各函数的入口地址，它是核心与驱动的接口。如下图所示。

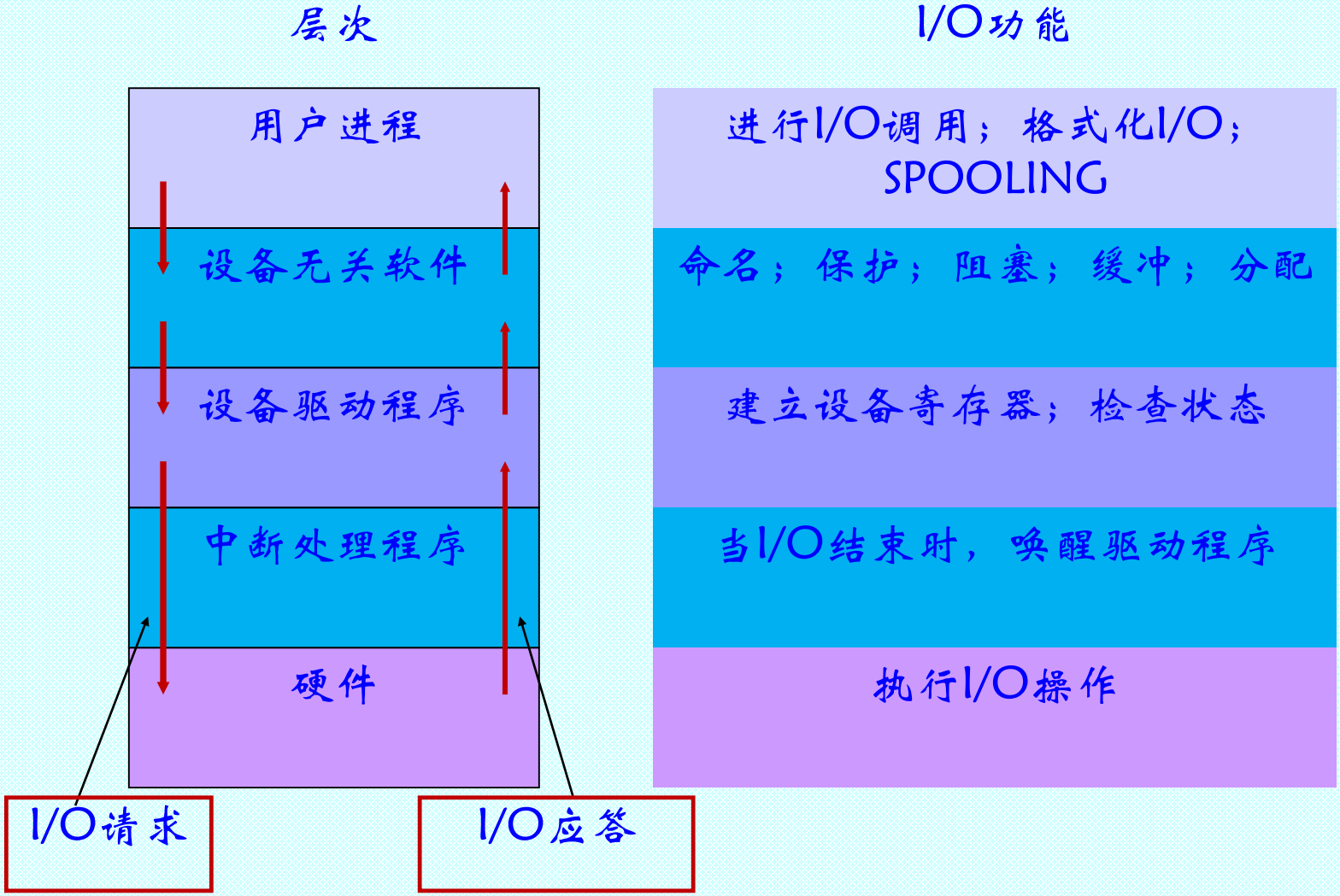
5.2.4 独立于设备的I/O软件(1)

- 设备无关软件完成的功能：
  - 提高设备驱动程序统一接口：方便添加设备驱动程序。
  - 设备命名和设备保护：所有设备抽象为文件，用设备文件来表示设备。
  - 提供独立于设备的块大小：隐藏不同设备的物理数据块大小的差异，向高层软件提供大小统一的逻辑数据块。
  - 缓冲区管理：建立内核缓冲区、数据复制。
  - 块设备的存储分配：实现块设备共享。
  - 独占性外围设备的分配和回收：由系统进行统一的分配和回收处理。
  - 错误报告：发现错误、就近逐层处理错误、提示错误。

5.2.5 用户空间的I/O软件

- 小部分I/O系统软件放在用户应用层
  - 库函数（与应用程序链接）
  - 假脱机技术（虚拟设备）
- 库函数实现的 I/O系统调用
  - I/O系统调用通常先是库函数调用：库函数与调用程序连接在一起，被嵌入在运行时装入内存的二进制程序中。
  - count=write(fd，buffer，nbytes)；
- 非库函数实现的 I/O系统调用
  - spooling系统

I/O系统各层软件及其功能



I/O操作执行步骤

1）进程对已打开文件的文件描述符执行读库函数；

2）独立设备I/O软件检查参数正确性。高速缓存中有要读的信息块，从缓冲区直接读到用户区，完成I/O请求；

3）若数据不在缓冲区，执行物理I/O，实现将设备逻辑名转换成物理名，检查对设备操作的权限，将I/O请求排队，阻塞进程且等待I/O完成；

4）内核启动设备驱动程序，分配存放读出块的缓冲区，准备接收数据，且向设备控制寄存器发启动命令，或建立DMA传输，启动I/O；

5）设备控制器操作设备，执行数据传输；

6）DMA控制器控制一块传输完成，硬件产生I/O结束中断；

7）CPU响应中断，转向磁盘中断处理程序。

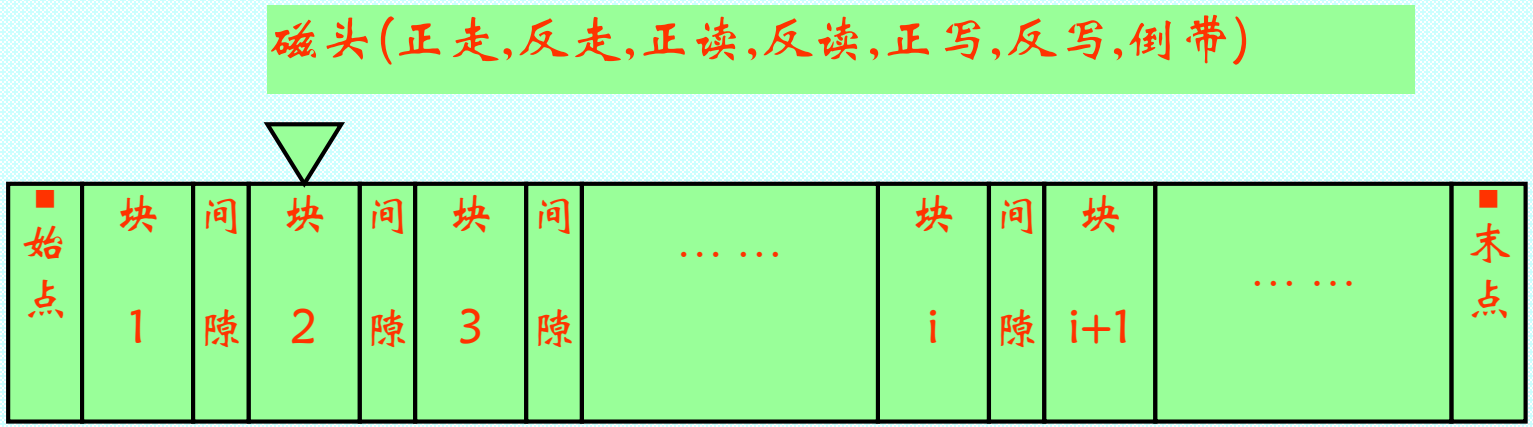
8）当应用进程被再次调度执行时，从I/O系统调用的断点恢复执行。

5.3 驱动调度技术

- 驱动调度：系统根据调度策略，按最佳次序执行要求访问的诸多请求。
- 驱动调度能减少为若干个I/O请求服务所需的总时间，提高系统效率、除了I/O请求的优化排序外，信息在辅助存储器上的排列方式，存储空间分配方法都能影响存取访问速度。

5.3.1 存储设备的物理结构(1)

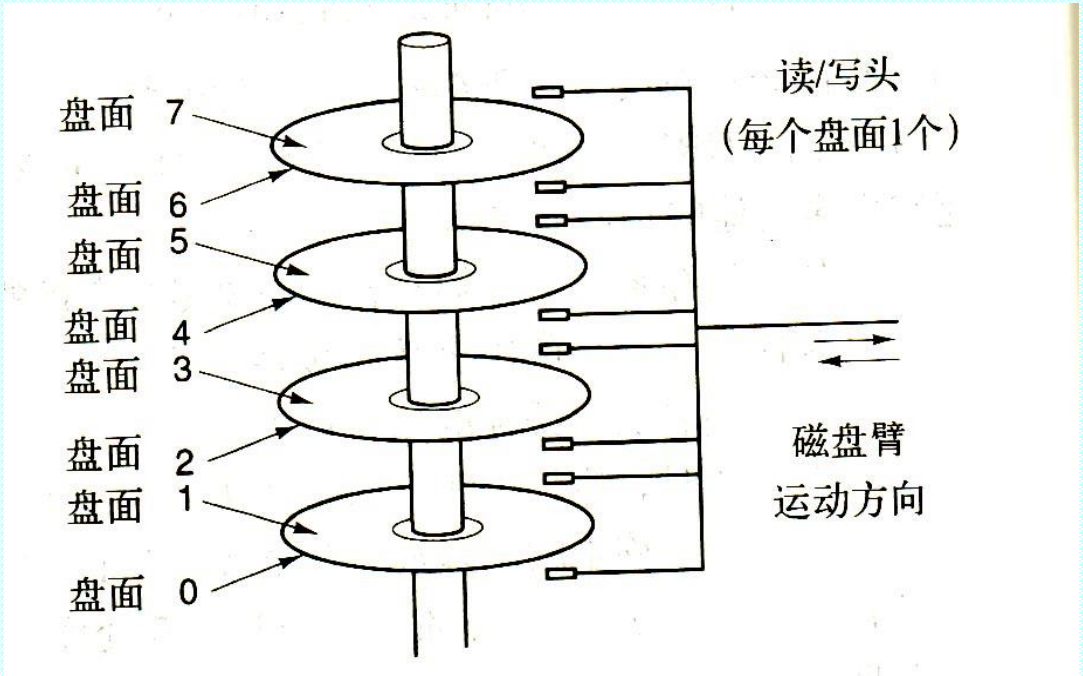
- 顺序存取存储设备：存取信息时，只能按存储单元的位置，顺序地一个接一个地进行存取的存储器。严格依赖信息的物理位置进行定位和读写的存储设备；
- 具有存储容量大、稳定可靠、卷可装卸和便于保存等优点。

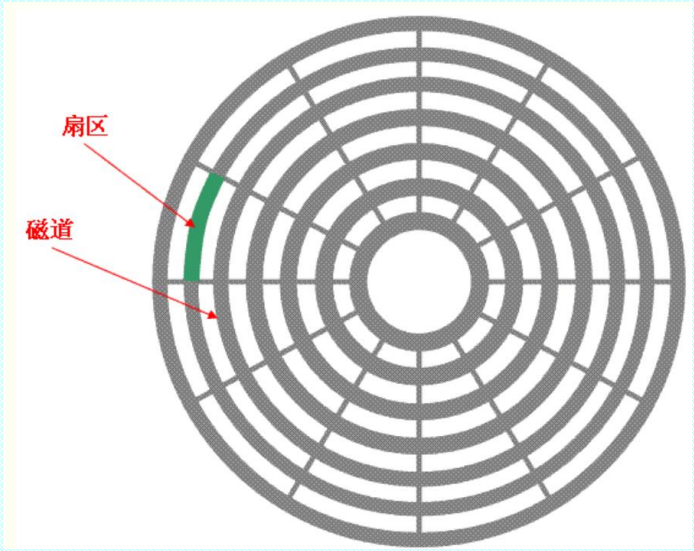


存储设备的物理结构(2)

- 直接存取存储设备：每个物理记录有确定的位置和唯一的地址。信息存取所需的时间几乎不依赖于此信息的位置。
- 磁盘是一种直接(随机)存取存储设备。
- 访问磁盘记录参数：柱面号、磁头号、块号

移动头磁盘结构





数据的组织

- 盘片（Platter）
  - 磁盘最基本的组成部分是由坚硬金属材料制成的涂以磁性介质的盘片，不同容量硬盘的盘片数不等。每个盘片有两面，都可记录信息。
- 磁道 (Tracks)
  - 盘片表面以盘片中心为圆心，不同半径的同心圆称为磁道。
- 扇区(Sectors)
  - 盘片被分成许多扇形的区域，每个区域叫一个扇区，硬盘每个扇区可存储512字节信息。FAT32模式下，每个扇区的容量为4KB。每个扇区的大小相当于一个盘块。
- 磁头(Heads)
  - 每个盘片的每一面都有一个读写头（read-write head），用于读取相应盘面的数据。习惯用磁头号来区分。

数据的组织

- 柱面 (Cylinders)
  - 不同盘片相同半径的磁道所组成的圆柱称为柱面。磁道与柱面都是表示不同半径的圆，在许多场合，磁道和柱面可以互换使用。
- 容量磁盘
  - 扇区，磁道（或柱面）和磁头数构成了硬盘结构的基本参数，通过这些参数可以得到硬盘的容量，计算公式为：

存储容量＝磁头数×磁道（柱面）数×每道扇区数

×每扇区字节数

1.44M =2×80×18×512

磁盘的类型

1.固定头磁盘

- 每条磁道上都有一个读/写磁头，所有的磁头被装入一个磁臂
- 通过这些磁头可以访问所有磁道，并进行并行读写
- 主要用于大容量磁盘

2.移动头磁盘

- 每个盘面仅有一个磁头，被装入一个磁臂中
- 为能访问盘面上的所有磁道，该磁头必须移动以进行寻道
- 只能串行读/写，致使I/O速度较慢
- 结构简单，广泛应用中、小型磁盘，微机上的硬盘和软盘，都采用移动磁头结构

磁盘访问时间

- 寻道时间（seek time）T_s
  - 将磁头从当前位置移到指定磁道所经历的时间。一般为2－30毫秒，平均约为10毫秒。

$$T_s = m*n+s$$

s：磁盘的启动时间,大约3ms；

m：每移动一条磁道所经历的时间，对一般磁盘：m＝0.3ms，对高速磁盘：m<＝0.1ms；

n：移动的磁道数目；

磁盘访问时间

- 旋转延迟时间（rotational latency time）Tr
  - 指定扇区移动到磁头下所经历的时间。

    Tr=1/2r （平均情况下，需要旋转半圈）
  - r—磁盘以秒计的旋转速度
  - 一个7200（转/每分钟）的硬盘，则旋转延迟时间为：60×1000÷7200÷2＝4.17毫秒。
  - 一个5400（转/每分钟）的硬盘，旋转延迟时间为：60×1000÷5400÷2＝5.56毫秒。
  - 一个300/600（转/每分钟）软盘，平均旋转延迟时间为：60×1000÷300÷2＝100毫秒，60×1000÷600÷2＝50毫秒。

磁盘访问时间

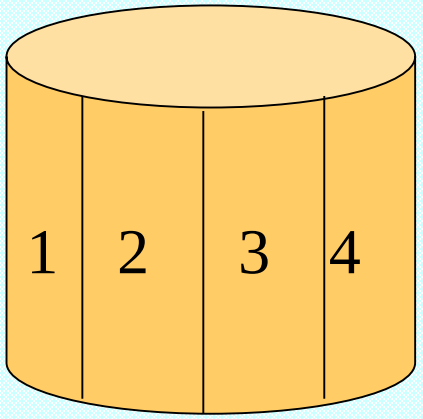
- 传输时间Tt
  - 数据从磁盘读出，或向磁盘写数据所经历的时间，约为零点几个毫秒,可以忽略不计。

    $Tt = b/rN$
  - b：读写的字节数
  - r：磁盘以秒计的旋转速度
  - N：一条磁道上的字节数
- 访问时间

  $Ta=Ts+Tr+Tt=(m*n+s)+1/2r+b/rN$

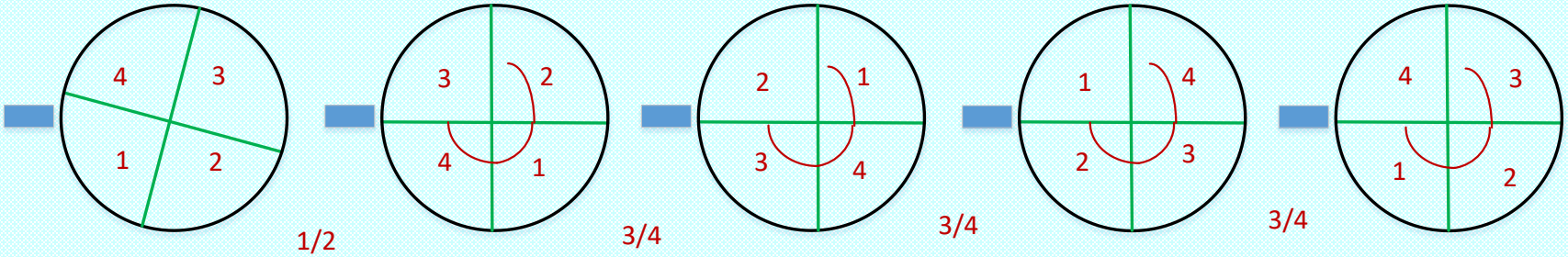
5.3.2循环排序(1)

- 考虑磁道保存4个记录的旋转型设备，假定收到四个I/O请求。
  - 请求次序　记录号
  - （1）　读记录4
  - （2）　读记录3
  - （3）　读记录2
  - （4）　读记录1



循环排序(2)

- 多种I/O请求排序方法
  - 方法1：按照I/O请求次序读记录4、3、2、1，平均用1/2周定位，再加上1/4周读出记录，总处理时间等于3周，即60毫秒。（1/2+1/4+3*3/4）
  - 方法2：如果次序为读记录1、2、3、4。总处理时间等于1.5周，即30毫秒。
  - 方法3：如果知道当前读位置是记录3，则采用次序为读记录4、1、2、3。总处理时间等于 1 周，即20毫秒。



5.3.3 优化分布(1)

- 考虑10个逻辑记录A，B……，J被存于旋转型设备上，每道存放10个记录，安排如下：

  物理块　　逻辑记录

  1-10　　A-J　(A B C D E …..)

- 处理10个记录的总时间

  10毫秒(移动到记录A的平均时间)+ 2毫秒(读记录A)+4毫秒(处理记录A)+9×[16毫秒(访问下一记录) +2毫秒(读记录)+4毫秒(处理记录)] ＝214毫秒

优化分布(2)

信息优化分布

- 处理10个记录的总时间为：

  10毫秒(移动到记录A的平均时间)+10×[2毫秒（读记录）×4毫秒（处理记录）]=70毫秒

物理块	逻辑记录
1	A
2	H
3	E
4	B
5	I
6	F
7	C
8	J
9	G
10	D

5.3.4 交替地址

- 每个记录重复记录在设备的多个区域，读相同的数据，有几个交替地址，也称为多重副本或折迭。
- 成功与否取决于下列因素：数据记录总是读出使用，不需修改写入；数据记录占用的存储空间总量不太大；数据使用极为频繁。

5.3.5 搜查定位(1)

- 移臂调度有若干策略

(1)“先来先服务” 算法

(2)“最短查找时间优先”算法

(3)“扫描”算法

(4)“分步扫描”算法

(5)“电梯调度”算法

(6)“循环扫描”算法

- 举例：假如磁盘机共有200个柱面，编号0至199，考虑依次到达下列柱面访问请求序列：

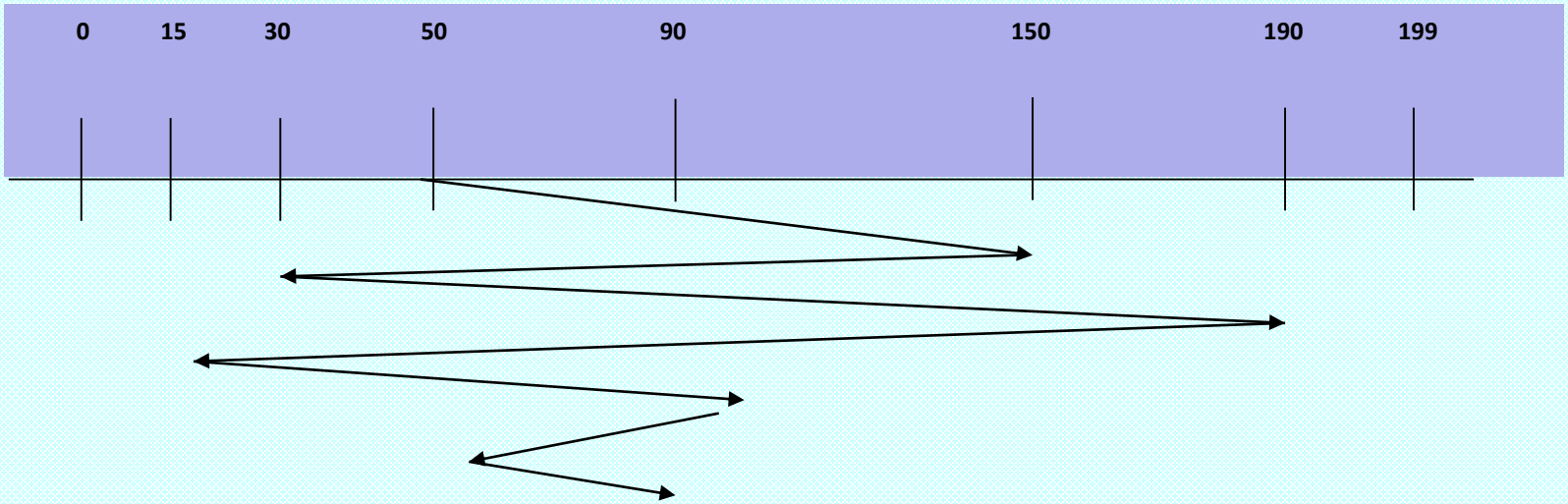
  150，30，190，20，100，55，90

- 同时假设磁头当前处于50号柱面位置，且正在向柱面号大的方向移动。

(1)“先来先服务”算法

- 磁盘臂是随机移动的，不考虑各I／O请求间的相对次序和移动臂当前所处位置，进程等待I／O请求时间会很长，寻道性能较差。
- 移动臂移动柱面总数=710。

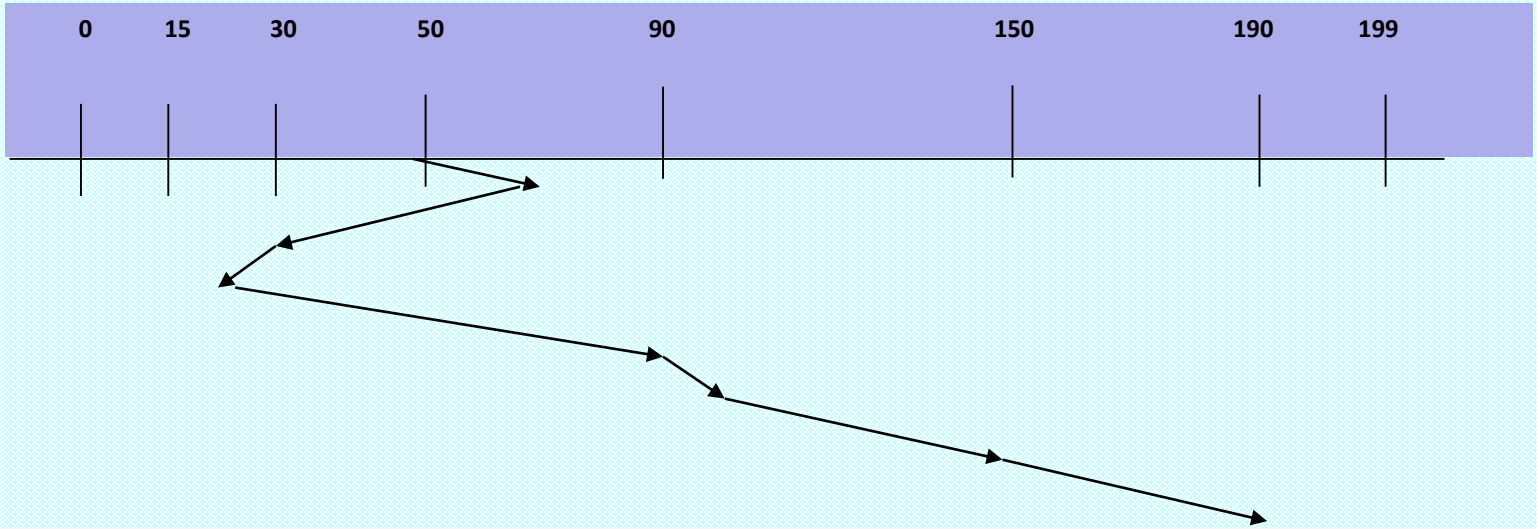
50 150，30，190，20，100，55，90



(2)“最短查找时间优先”算法

- 考虑I／O请求之间的区别，总是先执行查找时间最短的请求，与FIFO 算法相比有较好寻道性能。
- 移动臂移动柱面总数=210。

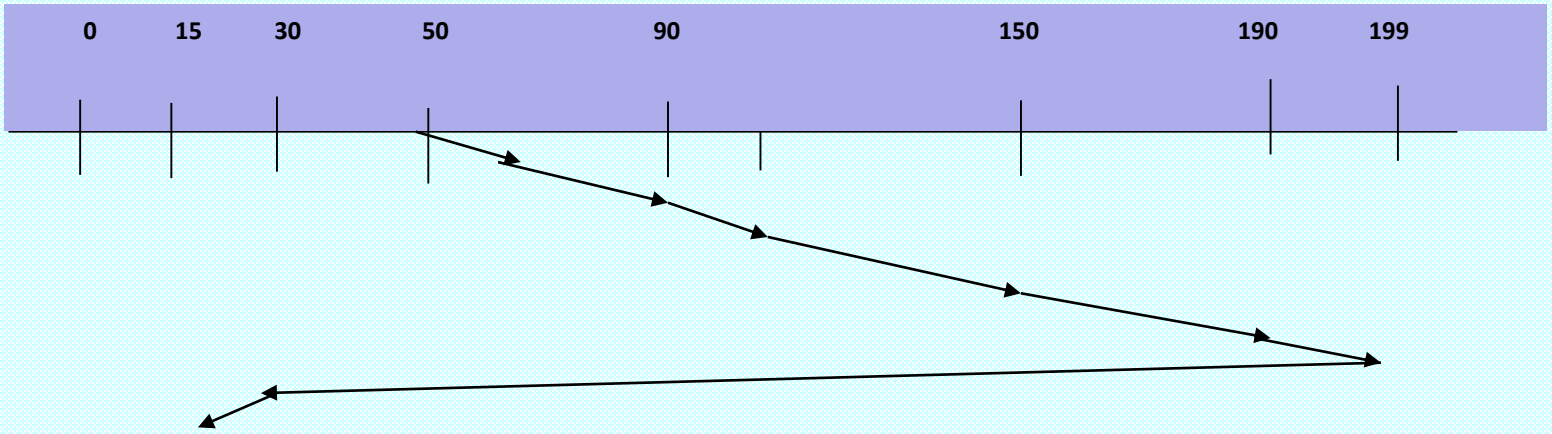
50 150，30，190，20，100，55，90



(3)“扫描”算法

- 磁盘臂每次沿一个方向移动，扫过所有柱面，遇到最近的I/O请求便进行处理，直到最后一个柱面后，再向相反方向移动回来。
- 移动臂移动柱面总数=328。

50 150，30，190，20，100，55，90



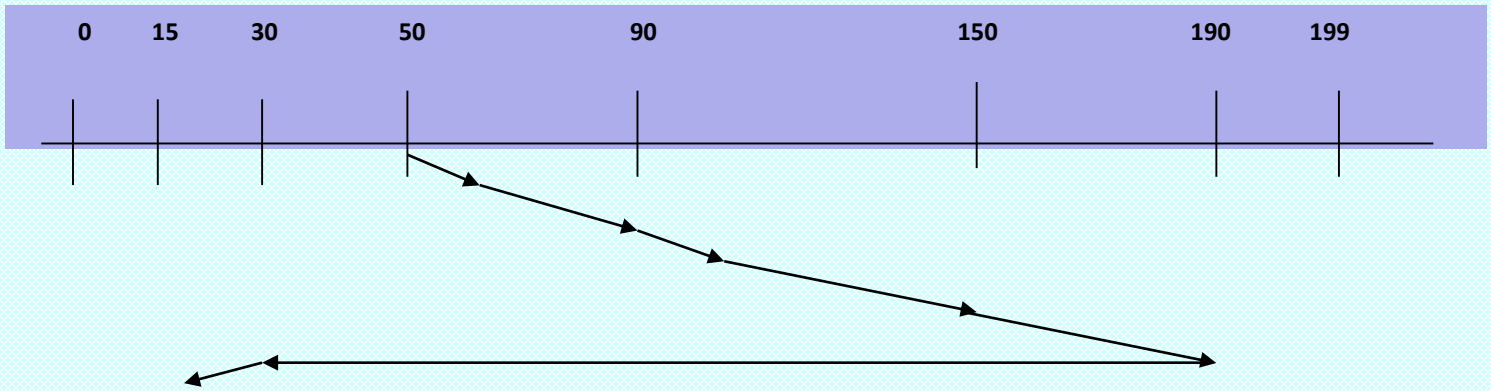
(4)“分步扫描”算法

- 进程重复请求访问同一柱面会垄断设备，造成“磁臂粘性”，导致其他柱面访问请求长时间得不到服务，采用“分步扫描”算法可以避免这类问题。
- 具体做法是：将I／O请求分为长度为N的子队列，按FIFO算法依次处理每个子队列，而每个子队列采用扫描算法，处理完一个后再服务下一个子队列，以避免出现磁臂粘住现象。这种调度算法能保证每个I/O请求的等待时间不致太长，当 N 值很大时，接近于“扫描”算法性能；当N＝ 1 时，接近于 FIFO算法性能。

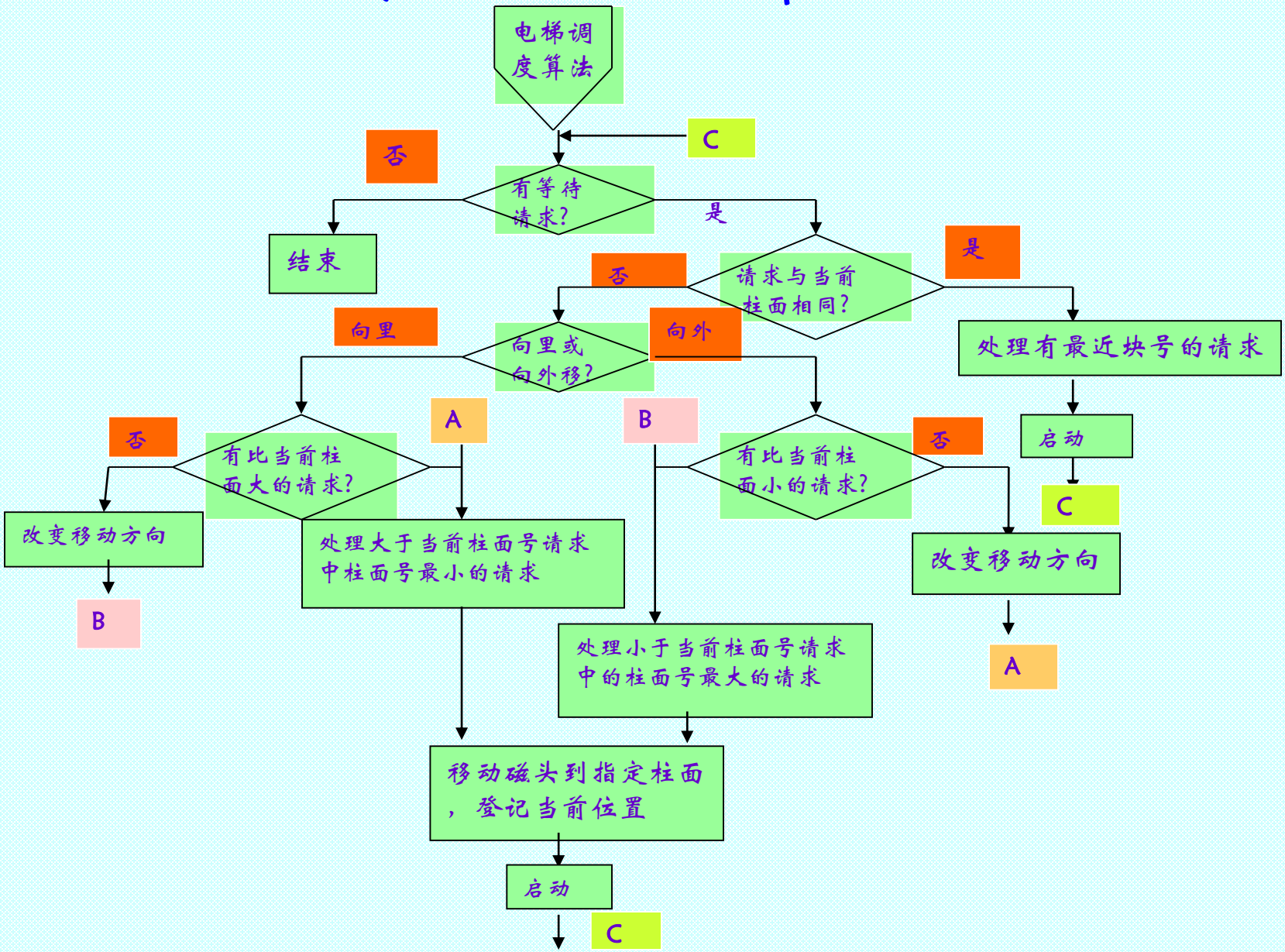
(5)“电梯调度”算法

- “电梯调度”算法 （elevator algorithm）又称LOCK算法，是如扫描算法的一种改进，无访问请求时，移动臂停止不动，有访问请求时，移动臂按电梯规律移动。
- 移动臂移动柱面总数=310。

50 150，30，190，20，100，55，90



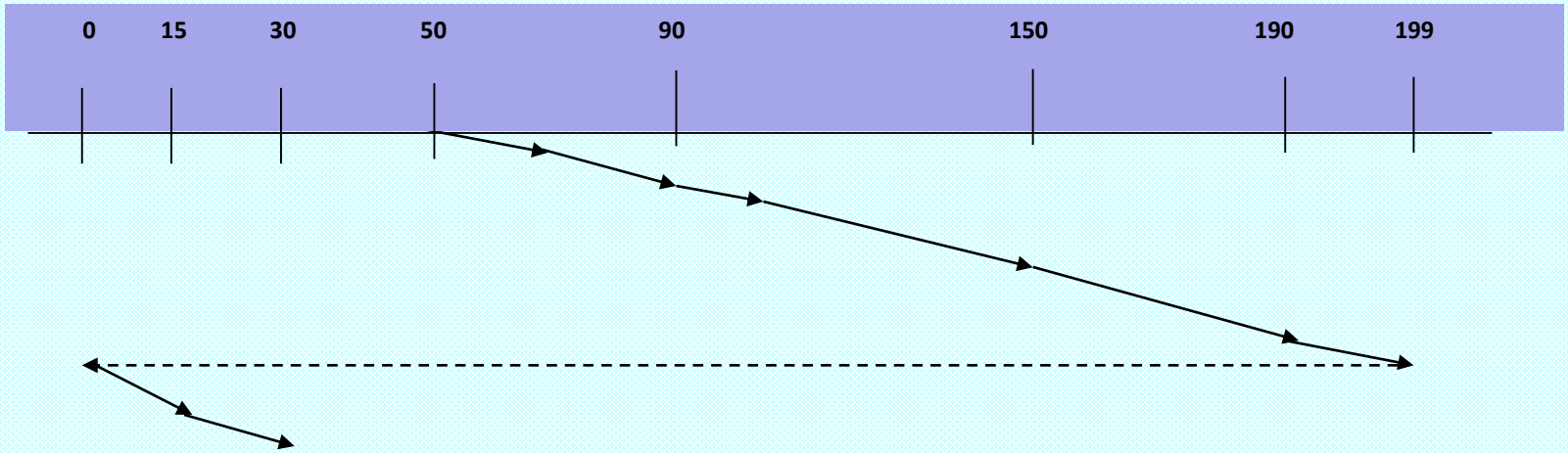
“电梯调度”算法流程


电梯调度算法
C
有等待请求?
否
是
结束
请求与当前柱面相同?
是
否
处理有最近块号的请求
向里或向外移?
向里
向外
启动
C
有比当前柱面大的请求?
否
A
B
有比当前柱面小的请求?
否
改变移动方向
B
处理大于当前柱面号请求中柱面号最小的请求
处理小于当前柱面号请求中的柱面号最大的请求
改变移动方向
A
移动磁头到指定柱面，登记当前位置
启动
C

“循环扫描”算法

- 为适应有大量柱面均匀分布的存取请求进入系统而设计的扫描方式。移动臂总是从 0 柱面至最大号柱面顺序扫描，然后，直接返回 0 柱面重复进行，归途中不再提供服务，构成一个循环，缩短处理新来请求的最大延迟。
- 移动臂移动柱面总数=378。

50 150，30，190，20，100，55，90



5.4 设备分配

- 5.4.1 设备独立性
- 5.4.2 设备分配及其数据结构

5.4.1 设备独立性

- 用户不指定特定设备，指定逻辑设备，使得用户作业和物理设备独立开来，
- 通过其它途径建立逻辑设备和物理设备之间对应关系，这种特性为“设备独立性”。
- 好处--用户与物理的外围设备无关，系统增减或变更外围设备时程序不必修改；易于对付输入输出设备的故障。

设备独立性

设备独立性带来以下两方面的好处：

- 设备分配时的灵活性
  - 当进程以逻辑设备名请求某类设备时，如果一台设备已经分配给其它进程或正在检修，此时系统可以将其它几台相同的空闲设备中的任一台分配给该进程，只有当此类设备全部被分配完时，进程才会被阻塞。
- 易于实现I/O重定向
  - 用于I/O操作的设备可以更换，重定向即而不必改变应用程序。

设备独立性

- 设备独立性软件

为了实现设备的独立性，必须在驱动程序之上设置一层软件，称为设备独立性软件，其主要功能有以下两个方面：

  - 1. 执行所有设备的公有操作
  - 2.向用户层（或文件层）软件提供统一的接口

- 逻辑设备表

为了实现逻辑设备名到物理设备名的映射，系统必须设置一张逻辑设备表LUT（Logical Unit Table），能够将应用程序中所使用的逻辑设备名映射为物理设备名，并提供该设备驱动程序的入口地址。

5.4.2 设备分配(1)

- 从设备的特性来看，可以把设备分成独占设备、共享设备和虚拟设备三类：
- 相应的管理和分配外围设备的技术可分成：
  - 独占方式、共享方式和虚拟方式。
- 常用的I/O设备分配算法：
  - 先请求先服务，优先级高者先服务等。
- 此外，在多进程请求I/O设备分配时，应防止因循环等待对方所占用的设备而产生死锁，应预先进行检查。

设备分配中的数据结构

数据结构有：系统设备表(SDT)、设备控制表(DCT)、控制器控制表(COCT)、通道控制表(CHCT)。

- 1、系统设备表SDT

在整个系统中，有一张系统设备表（SDT），用于记录系统中全部设备的信息。每个设备占一个表目，其中包括设备类型、设备标识符、设备控制表指针及设备驱动程序的入口地址等表项。

- 2、控制器控制表COCT

系统为每一个控制器都配置了一张用于记录本控制器情况的控制器控制表。

设备分配中的数据结构

- **3、通道控制表CHCT**

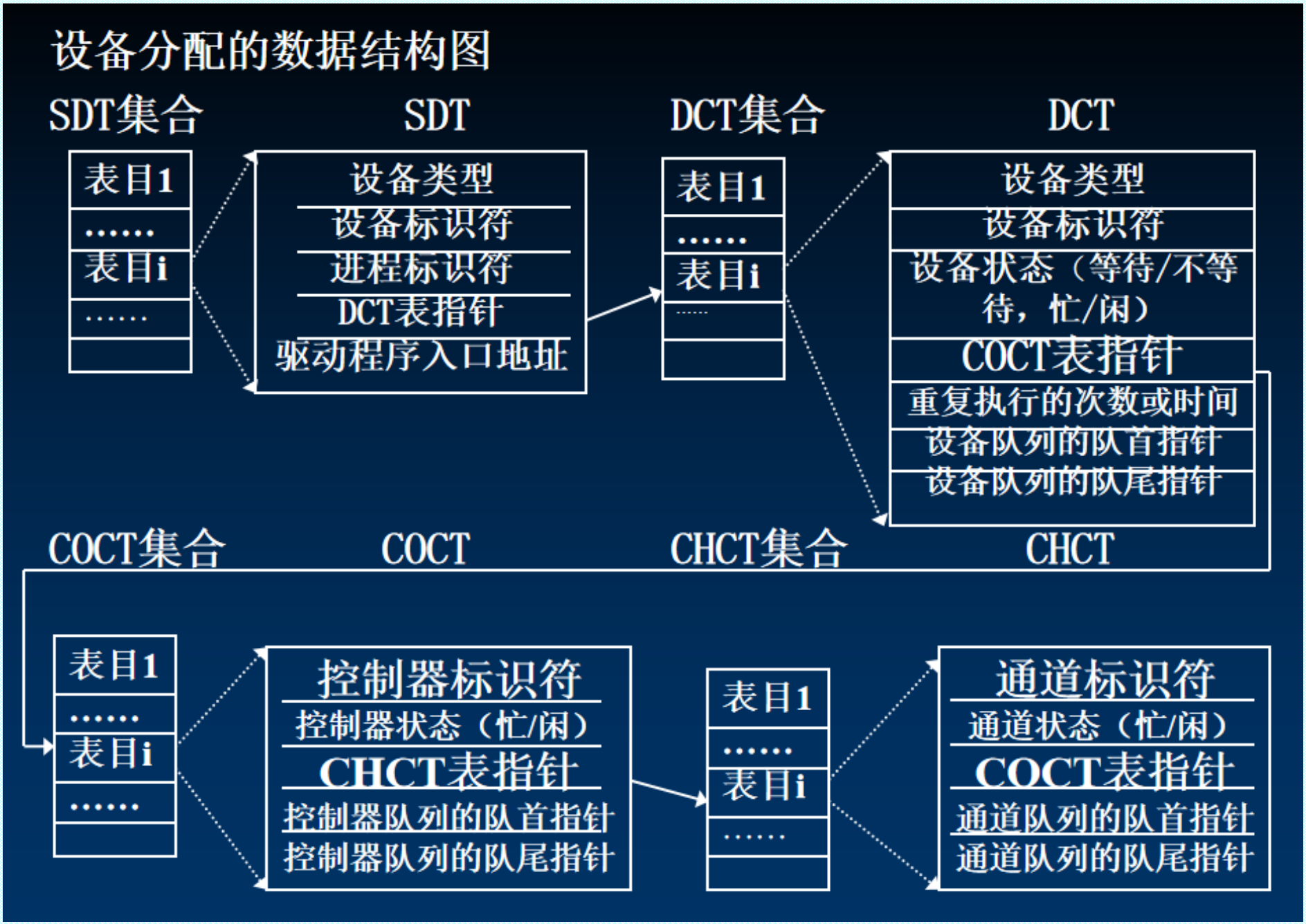
系统为每一个通道都配置了一张用于记录本通道情况的通道控制表。

- **4、设备控制表DCT**

系统为每一个设备都配置了一张设备控制表，用于记录该设备的情况。表中除了有用于指示设备类型的字段和设备标识符字段外，还应有下列字段：

  - 设备队列队首指针、设备状态、与设备连接的控制器表指针、重复执行次数。

设备分配的数据结构图


SDT集合
SDT
DCT集合
DCT
表目1
……
表目i
……
设备类型
设备标识符
进程标识符
DCT表指针
驱动程序入口地址
表目1
……
表目i
……
设备类型
设备标识符
设备状态（等待/不等待，忙/闲）
COCT表指针
重复执行的次数或时间
设备队列的队首指针
设备队列的队尾指针
COCT集合
COCT
CHCT集合
CHCT
表目1
……
表目i
……
控制器标识符
控制器状态（忙/闲）
CHCT表指针
控制器队列的队首指针
控制器队列的队尾指针
表目1
……
表目i
……
通道标识符
通道状态（忙/闲）
COCT表指针
通道队列的队首指针
通道队列的队尾指针

设备分配的策略

- 根据设备的固有属性而采取的策略
  - 独享方式

独享方式是指将一个设备分配给某进程后，便一直由它独占，直至该进程完成或释放该设备为止，系统才能将该设备分配给其它进程使用。这种分配方式是对独占设备采用的分配策略。它不仅往往造成设备利用率低，而且还会引起系统死锁。

设备分配的策略

- 共享方式

共享方式是指将共享设备（磁盘）同时分配给多个进程使用。但是这些进程对设备的访问需进行合理的调度。

- 虚拟方式

虚拟方式是指通过高速的共享设备，把一台慢速的以独占方式工作的物理设备改造成若干台虚拟的同类逻辑设备，这就需要引入SPOOLing技术。虚拟设备属于逻辑设备。

设备分配的策略

设备分配算法（与进程的调度算法相似）

- 先来先服务

  当多个进程同时向某一设备提出I/O请求时，该算法就根据对该设备提出请求的先后次序将这些进程排列成一个设备请求队列，设备分配程序把设备首先分配给队首进程。

- 优先级高者优先

  对优先权高的进程所提出的I/O请求赋予高优先权，在形成设备队列时，将优先级高的进程排在设备队列前面，先得到分配。而对于优先权相同的I/O请求，则按先来先服务原则排队分配。

设备分配的策略

设备分配中的安全性

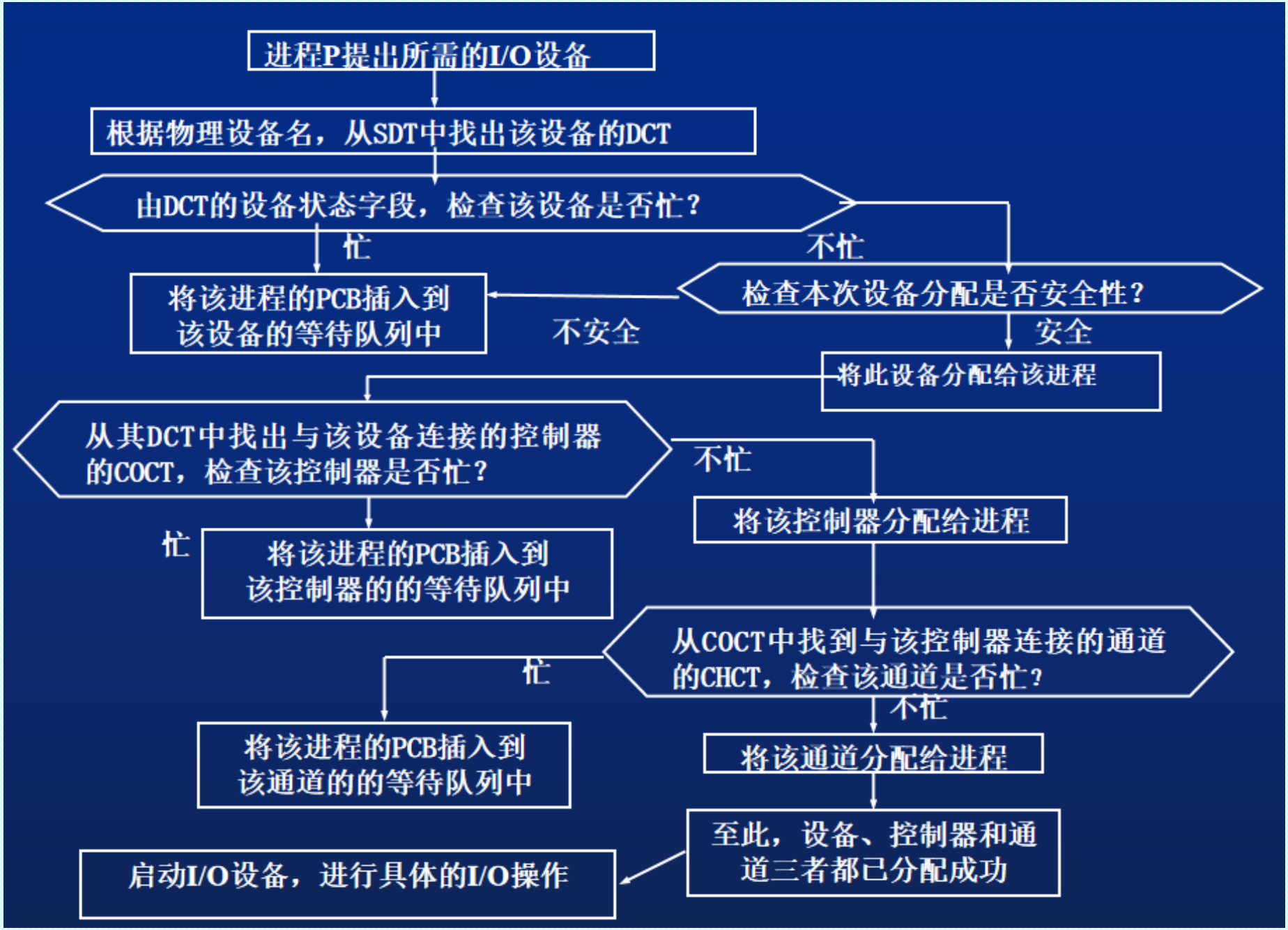
1. 安全分配方式

每当进程发出I/O请求后，便进入阻塞状态，直到其I/O操作完成时才会唤醒。

2. 不安全分配方式

进程在发出I/O请求后仍继续运行，需要时又发出第二个I/O请求、第三个I/O请求等。

设备分配程序

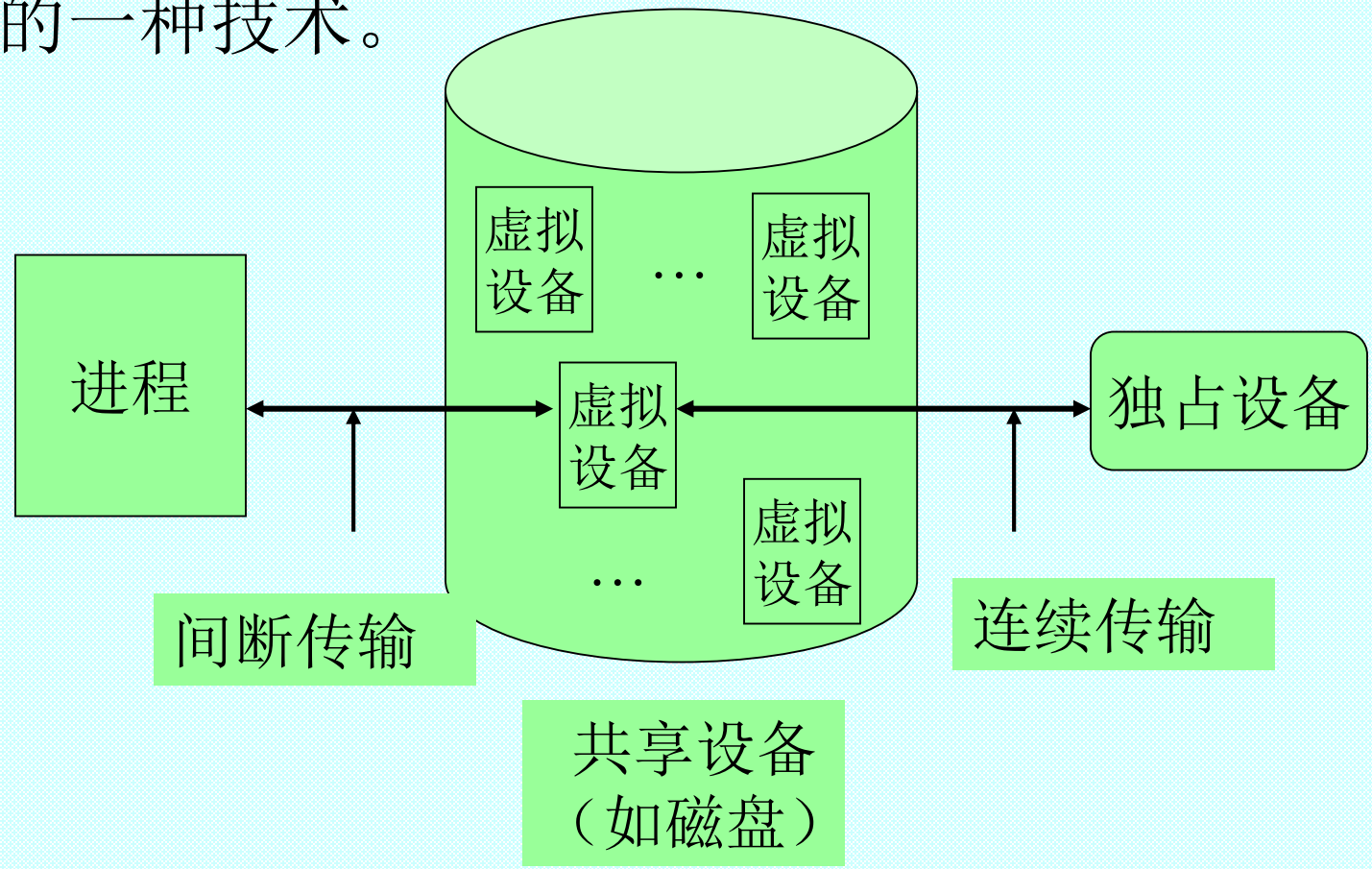

进程P提出所需的I/O设备
根据物理设备名，从SDT中找出该设备的DCT
由DCT的设备状态字段，检查该设备是否忙？
忙
不忙
将该进程的PCB插入到该设备的等待队列中
不安全
检查本次设备分配是否安全性？
安全
将此设备分配给该进程
从其DCT中找出与该设备连接的控制器的COCT，检查该控制器是否忙？
不忙
忙
将该进程的PCB插入到该控制器的的等待队列中
将该控制器分配给进程
从COCT中找到与该控制器连接的通道的CHCT，检查该通道是否忙？
忙
不忙
将该进程的PCB插入到该通道的的等待队列中
将该通道分配给进程
至此，设备、控制器和通道三者都已分配成功
启动I/O设备，进行具体的I/O操作

5.5 虚拟设备

- 5.5.1 问题的提出
- 5.5.2 SPOOLING设计和实现
- 5.5.3 SPOOLING应用

5.5.1问题的提出

- SPOOLing技术：是用一类物理设备模拟另一类物理设备的技术，是使独占设备变成共享设备的一种技术。



SPOOLing技术

早期批处理系统中使用的虚拟技术是以脱机方式工作的。为了缓和CPU和I/O设备之间的速度不匹配的问题。利用专门的外围控制机将低速I/O设备上的数据传送到高速磁盘上，或者相反。当多道程序设计的分时系统出现后，SPOOLing技术就孕育而生，它将一台独占设备改造成可以共享的虚拟设备。

SPOOLing技术

1.什么是 SPOOLing技术

当多道程序程序技术出现后，就可以利用一道程序，来模拟脱机输入时的外围控制机的功能，即把低速I/O设备上的数据传送到高速的磁盘上；

再用另一道程序来模拟脱机输出时外围控制机的功能，即把数据从磁盘传送到低速I/O设备上。

这样，便在主机的直接控制下，实现脱机输入、输出功能。

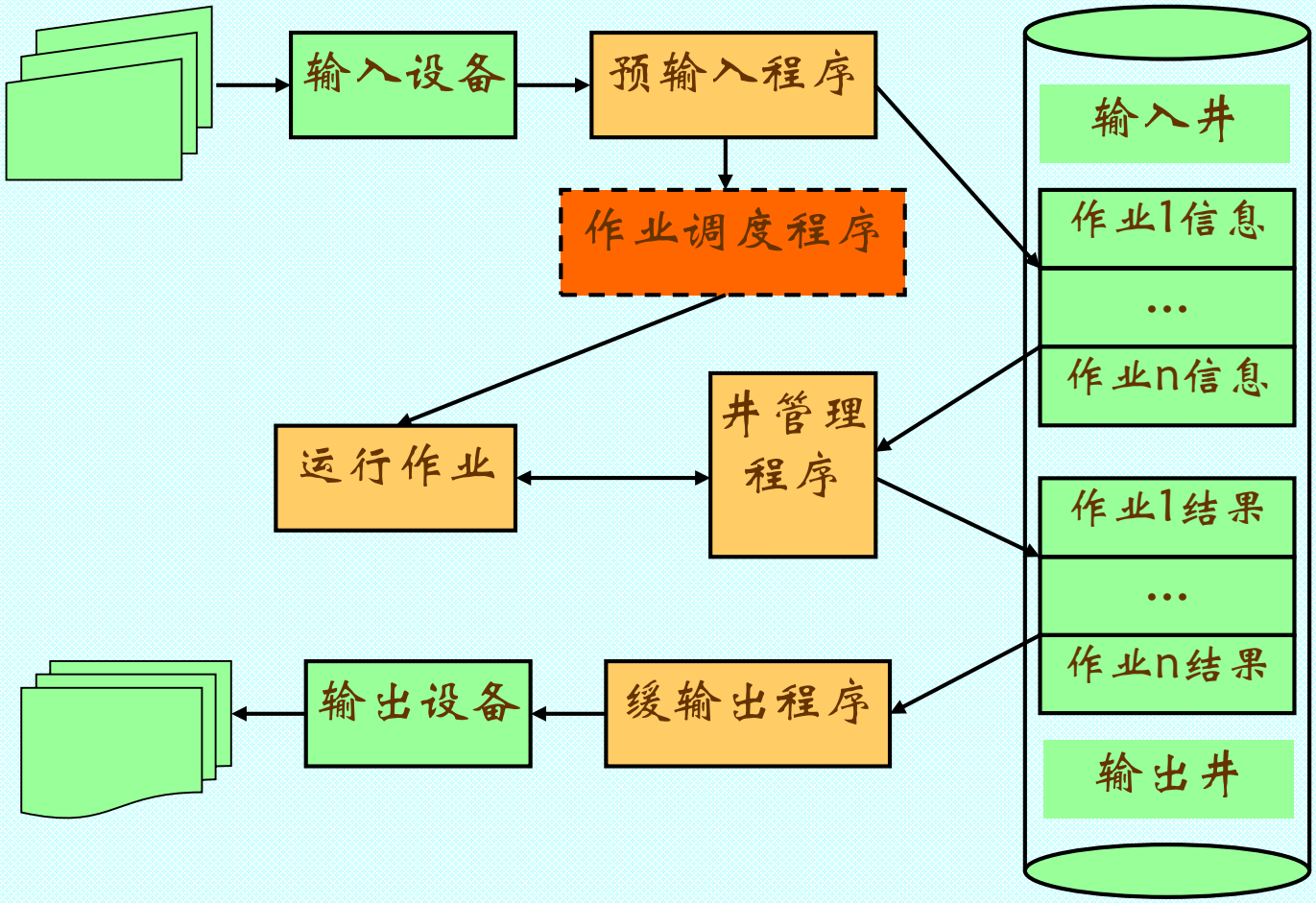
所以，我们把这种在联机情况下实现的同时与外围设备联机操作的技术称为SPOOLing（Simultaneous Peripheral Operation On Line），或称为假脱机技术。

- 虚拟设备的主要条件
  - 硬件方面：有大容量磁盘，中断机构和通道装置支撑，使CPU与外设设备可以并行工作的能力；
  - 软件方面：多道程序设计技术，合理分配处理器，实现联机的外围设备同时操作 。

5.5.2 斯普林系统的设计和实现(1)

- “井”是用作缓冲的存储区域，采用井的技术能调节供求之间的矛盾，消除人工干预带来的损失。
- “预输入程序”
- “缓输出程序”
- “井管理程序”

斯普林系统的设计和实现(2)



SPOOLING组成和结构

斯普林系统的设计和实现(3)

- 输入井中作业状态
  - 输入状态：
  - 收容收态：
  - 执行状态：
  - 完成状态：

斯普林系统的设计和实现(4)

- SPOOLING数据结构
  - 作业表：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。
  - 预输入表：每个用户作业有一张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。
  - 缓输出表：每个用户作业拥有一张包括作业名、作业状态、文件名、设备类、数据起始位置、数据当前位置等。

斯普林系统的设计和实现(5)

- 井文件空间的管理(1)
  - 连接方式：输入的信息被组织成连接文件，这种方式的优点是数据信息可以不连续存放，文件空间利用率高。

斯普林系统的设计和实现(6)

- 井文件空间的管理(2)
  - 计算方式：假定磁盘井文件空间，每个磁道存放100个80字节记录，每张卡片为80个字节，若每个柱面有20个磁道，则一个柱面可存放2000张卡片信息。第n张卡片信息被存放在：

    磁道号＝卡片号n /100

    记录号＝（卡片号n）% 100
  - 用卡片号n除以100的整数和余数部分分别为其存放的磁道号和记录号。

斯普林系统的设计和实现(7)

- Spooling应用例子
  - (1)打印机spooling守护进程
  - (2)网络通信spooling守护进程

SPOOLing技术应用

- 输入进程SP_i和输出进程SP_o
  - 进程SP_i

    模拟脱机输入时的外围控制机，将用户要求的数据从输入机通过输入缓冲区再送到输入井。当CPU需要输入数据时，直接从输入井读入内存。
  - 进程SPo

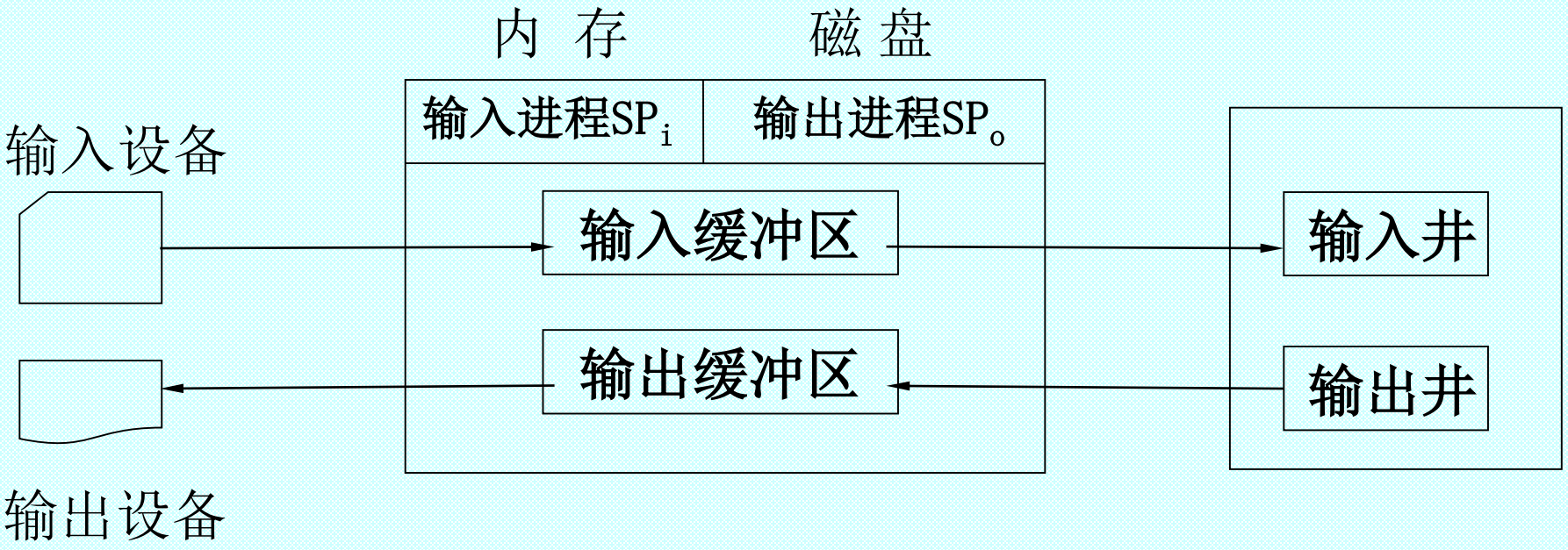
    进程模拟脱机输出时的外围控制机，把用户要求输出的数据，先从内存送到输出井，待输出设备空闲时，再将输出井中的数据，经过输出缓冲区送到输出设备上。

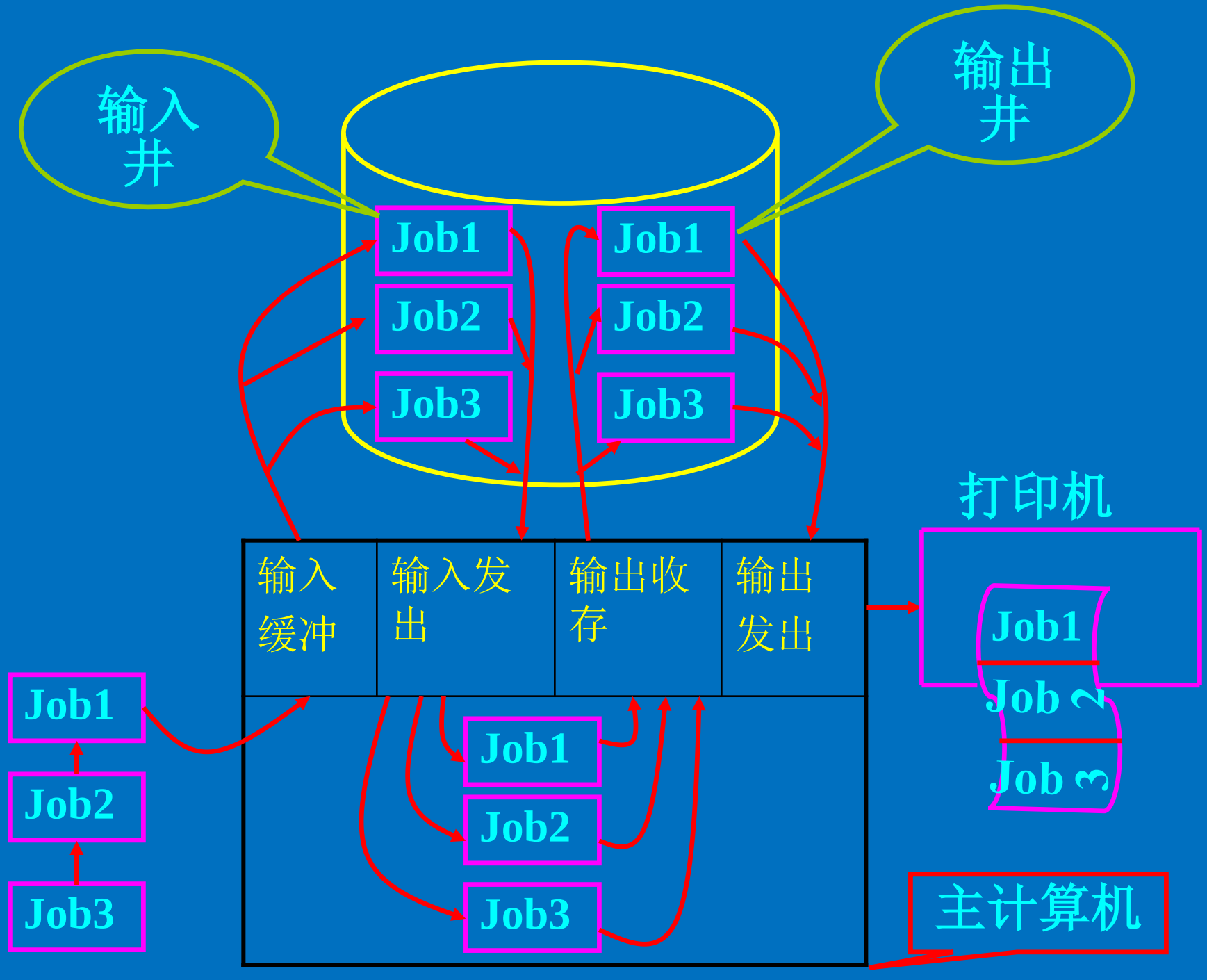
SPOOLing技术应用

- 输入缓冲区和输出缓冲区

在内存中开辟两个缓冲区：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，再传送到输入井
- 输出缓冲区用于暂存从输出井送来的数据，再传送给输出设备


内 存
磁 盘
输入进程SPi
输出进程SPo
输入设备
输入缓冲区
输出缓冲区
输入井
输出井
输出设备


输入井
输出井
Job1
Job2
Job3
Job1
Job2
Job3
打印机
输入缓冲
输入发出
输出收存
输出发出
Job1
Job2
Job3
Job1
Job2
Job3
Job1
Job 2
Job 3
主计算机

SPOOLing技术应用

共享打印机

打印机虽然是独享设备。但是通过SPOOLing技术，可以将它改造为一台可供多个用户共享的设备。共享打印机技术已被广泛地用于多用户系统和局域网络。当用户进程请求打印输出时，SPOOLing系统并不是真正把打印机分配给该用户进程，而由输出进程为他在输出井中申请一个存储空间，并将要打印的数据以文件的形式存放于此。各进程的输出文件形成一个输出队列，由输出SPOOLing系统控制这台打印机进程，依次将队列中的输出文件打印出来。

SPOOLing系统的特点

- 提高I/O的速度
- 将独占设备改造为共享设备
- 实现虚拟设备的功能