第六章 输入输出系统

6.1 I/O系统的功能、模型和接口

- I/O系统管理的主要对象
  - I/O设备和相应的设备控制器
- I/O系统管理最主要的任务
  - 完成用户提出的I/O请求
  - 提高I/O速率
  - 提高设备利用率
  - 方便更高层进程使用设备

- **I/O系统的基本功能**

① 隐藏物理设备的细节

② 与设备的无关性

方便用户

③ 提高CPU和I/O设备的利用率

④ 控制I/O设备

提高利用率

⑤ 确保正确共享设备

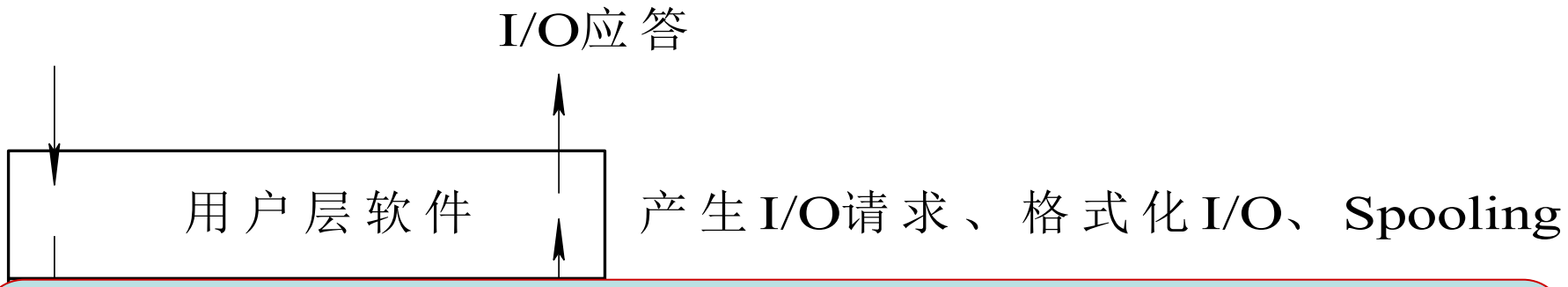
⑥ 错误处理

方便设备共享

I/O系统的层次结构和模型

- 由于I/O设备种类繁多，物理特性各异，并涉及到大量与数字逻辑运算有关的细节，造成对设备的操作和管理非常复杂和琐碎。
- 要求用户掌握I/O系统的原理，显然是不现实的，也会限制计算机的推广和应用。
- 所以，设法消除或屏蔽设备硬件内部的低级处理过程，为用户提供一个简便、易用、抽象的逻辑设备接口，保证用户安全、方便地使用各类设备，是设计I/O软件的一个重要原则。

- I/O软件负责设备的输入/输出，它涉及的面非常宽，往下与硬件有密切关系，往上又与用户直接交互。
- 它与进程管理、存储器、文件管理等都存在着一定的联系，即它们都可能需要I/O软件来实现I/O操作。
- 为使十分复杂的I/O软件能具有清晰的结构，更好的可移植性和易适应性，目前在I/O软件中已普遍采用层次式结构。
- 每一层都利用其下层提供的服务，完成输入/输出功能中的某些子功能，并屏蔽这些功能实现的细节，向高层提供服务。



•用户层软件：与用户交互的接口，用户可直接调用

• 设备独立性软件：负责实现与设备驱动器的统一接口、

• 设备驱动程序：与硬件直接相关，负责具体实现系统对

• 中断处理程序：保存被中断进程的CPU环境，转入相应的处理程序进行处理，处理完后再恢复被中断进程的现场后返回继续执行被中断的进程

I/O系统的层次结构及功能

• I/O系统接口

- I/O系统与高层之间的接口中，根据设备类型不同，进一步分成若干个接口

  ① 块设备接口

  ② 流设备接口（字符设备接口）

  ③ 网络接口

① 块设备接口

- 块设备管理程序与高层之间的接口
- 反映了大部分磁盘、光盘存储器的本质特征
- **块设备**
  - 数据存取和传输以数据块为单位的设备（磁盘）
- **隐藏磁盘的二维结构**
  - 将所有扇区统一编号，隐藏了[磁道号,扇区号]的二维结构
- **将抽象命令映射为低层操作**
  - 将上层发来的抽象命令映射为设备能识别的具体操作

② 流设备接口

- 流设备管理程序与高层之间的接口
- 反映了大部分字符设备的本质特征
- 字符设备
  - 数据存取和传输以字符为单位的设备（键盘）
- **get、put操作**
  - 字符设备不可寻址，只能采用顺序存取方式
- **in-control指令**
  - 通用指令，以统一方式处理不同类型的字符设备
- 大多数属于独占设备，互斥方式共享

③ 网络通信接口

- 现代OS大都提供面向网络的功能
  - 使计算机能够连接到网络
  - 使计算机能与网络上其他计算机通信

6.2 I/O设备和设备控制器

- I/O设备一般由两部分组成

① 执行I/O操作的机械部分

- 一般认为的I/O设备

② 执行控制I/O的电子部件

- 设备控制器或适配器

1. I/O设备的类型

- I/O设备的类型繁多，从OS观点看，其重要的性能指标有:
  - ① 设备使用特性
  - ② 数据传输速率
  - ③ 数据的传输单位
  - ④ 设备共享属性

  ……
- 因而，可以从不同角度对它们进行分类

① 按设备的使用特性分类

- 存储设备：也称外存或后备存储器、辅助存储器。
  - 该类设备存取速度较内存慢，但容量比内存大得多，相对价格也便宜
- 输入/输出设备：分为输入设备、输出设备和交互式设备。
  - 输入设备：键盘、鼠标、扫描仪、视频摄像、各类传感器等。
  - 输出设备：打印机、绘图仪、显示器、数字视频显示设备、音响输出设备等。
  - 交互式设备：集成上述两类设备，利用输入设备接收用户命令，通过输出设备显示执行的结果

② 按传输速率分类

- 低速设备：传输速率仅为每秒钟几个字节至数百个字节
  - 键盘、鼠标器、语音的输入和输出等
- 中速设备：传输速率在每秒钟数千个字节至数十万个字节
  - 行式打印机、 激光打印机等
- 高速设备：传输速率在数百个千字节至千兆字节
  - 磁盘机、光盘机等

③ 按信息交换的单位分类

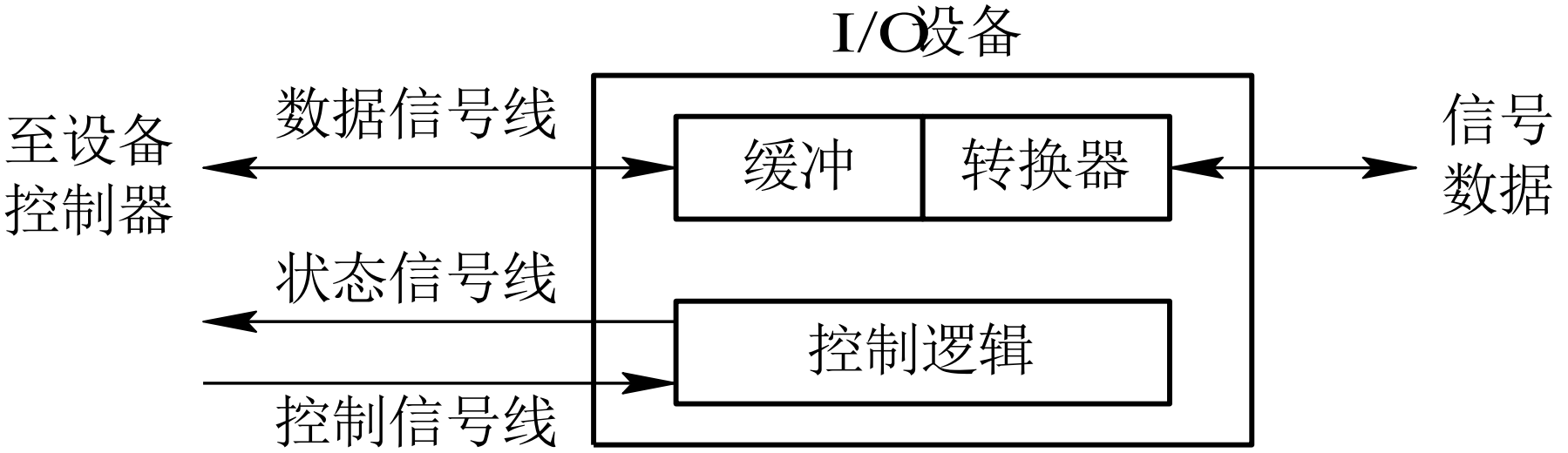
- 块设备：用于存储信息
  - 由于信息的存取总是以数据块为单位，故而得名
  - 属于有结构设备。典型的块设备是磁盘，每个盘块的大小为512 B～4 KB
  - 基本特征：速率高，可寻址，DMA方式
- 字符设备：用于数据的输入和输出。
  - 基本单位是字符，故称为字符设备。
  - 属于无结构类型。
  - 基本特征：速率低，不可寻址，中断驱动方式

④ 按设备的共享属性分类

- 独占设备：指在一段时间内只允许一个进程访问的设备，即临界资源
  - 系统一旦把这类设备分配给了某进程后，便由该进程独占，直至用完释放
  - 注意：独占设备的分配有可能引起进程死锁
- 共享设备：在一段时间内允许多个进程“同时”访问的设备
  - 显然，共享设备必须是可寻址的和可随机访问的设备。典型的共享设备是磁盘
- 虚拟设备：通过虚拟技术将一台独占设备变换为若干台逻辑设备，供若干个进程“同时”使用

2. 设备与控制器之间的接口

- 通常，I/O设备并不是直接与CPU进行通信，而是与设备控制器通信
- 在I/O设备中含有与设备控制器的接口，在该接口中有三种类型的信号，各对应一条信号线



设备与控制器之间的接口

① 数据信号线

- 用于设备和设备控制器之间传送数据信号。
- 对输入设备而言，外界输入的信号转换成数据，通常先送入缓冲器中，当数据量达到一定的比特(字符)数后，再从缓冲器通过一组数据信号线传送给设备控制器。
- 对输出设备而言，将从设备控制器经过数据信号线传送来的一批数据先暂存于缓冲器中，经转换后再逐个字符地输出。

② 控制信号线

- 是设备控制器向I/O设备发送控制信号时的通路
- 信号规定了设备将要执行的操作，如读操作、写操作，或移动磁头等操作

③ 状态信号线

- 用于传送设备当前状态的信号
- 设备的当前状态有正在读/写；设备已读/写完成，并准备好新的数据传送

3. 设备控制器

- 设备控制器的基本功能

① 接收和识别命令

- **CPU**向控制器发送多种不同的命令，设备控制器应能接收并识别这些命令。
- 为此，在控制器中应具有相应的控制寄存器，用来存放接收的命令和参数，并对所接收的命令进行译码。
- 例如，磁盘控制器可以接收**CPU**发来的**Read**、**Write**、**Format**等**15**条不同的命令，而且有些命令还带有参数；
- 相应地，在磁盘控制器中有多个寄存器和命令译码器等

② 数据交换

- 指实现CPU与控制器之间、控制器与设备之间的数据交换。
- 对于前者，是通过数据总线，由CPU并行地把数据写入控制器，或从控制器中并行地读出数据；
- 对于后者，是设备将数据输入到控制器，或从控制器传送给设备。
- 为此，在控制器中须设置数据寄存器

③ 标识和报告设备的状态

- 控制器应记下设备的状态供CPU了解
- 例如，仅当设备处于发送就绪状态时，CPU才能启动控制器从设备中读出数据
- 为此，在控制器中应设置一状态寄存器，用其中的每一位来反映设备的某一种状态
- 当CPU将该寄存器的内容读入后，便可了解该设备的状态

④ 地址识别

- 就像内存中的每一个单元都有一个地址一样，系统中的每一个设备也都有一个地址，而设备控制器又必须能够识别它所控制的每个设备的地址。
- 此外，为使CPU能向(或从)寄存器中写入(或读出)数据，这些寄存器都应具有唯一的地址
- 控制器应能正确识别这些地址。为此，在控制器中应配置地址译码器

⑤ 数据缓冲

- 由于I/O设备的速率较低而CPU和内存的速率很高，故在控制器中必须设置一缓冲器。
- 在输出时，用此缓冲器暂存由主机高速传来的数据，然后才以I/O设备所具有的速率将缓冲器中的数据传送给I/O设备；
- 在输入时，缓冲器则用于暂存从I/O设备送来的数据，待接收到一批数据后，再将缓冲器中的数据高速地传送给主机

⑥ 差错控制

- 设备控制器兼管对由I/O设备传送来的数据进行差错检测。
- 若发现传送中出现了错误，便向CPU报告，然后CPU将本次传送来的数据作废，并重新进行一次传送。
- 这样，便可保证数据输入的正确性

4. 设备控制器的组成

① 设备控制器与处理机的接口

- 用于实现CPU与设备控制器之间的通信。
  共有三类信号线: 数据线、地址线、控制线
- 数据线通常与两类寄存器相连接

  ①数据寄存器

  – 用于存放从设备送来的数据(输入)或从CPU送来的数据(输出)；

  ②控制/状态寄存器

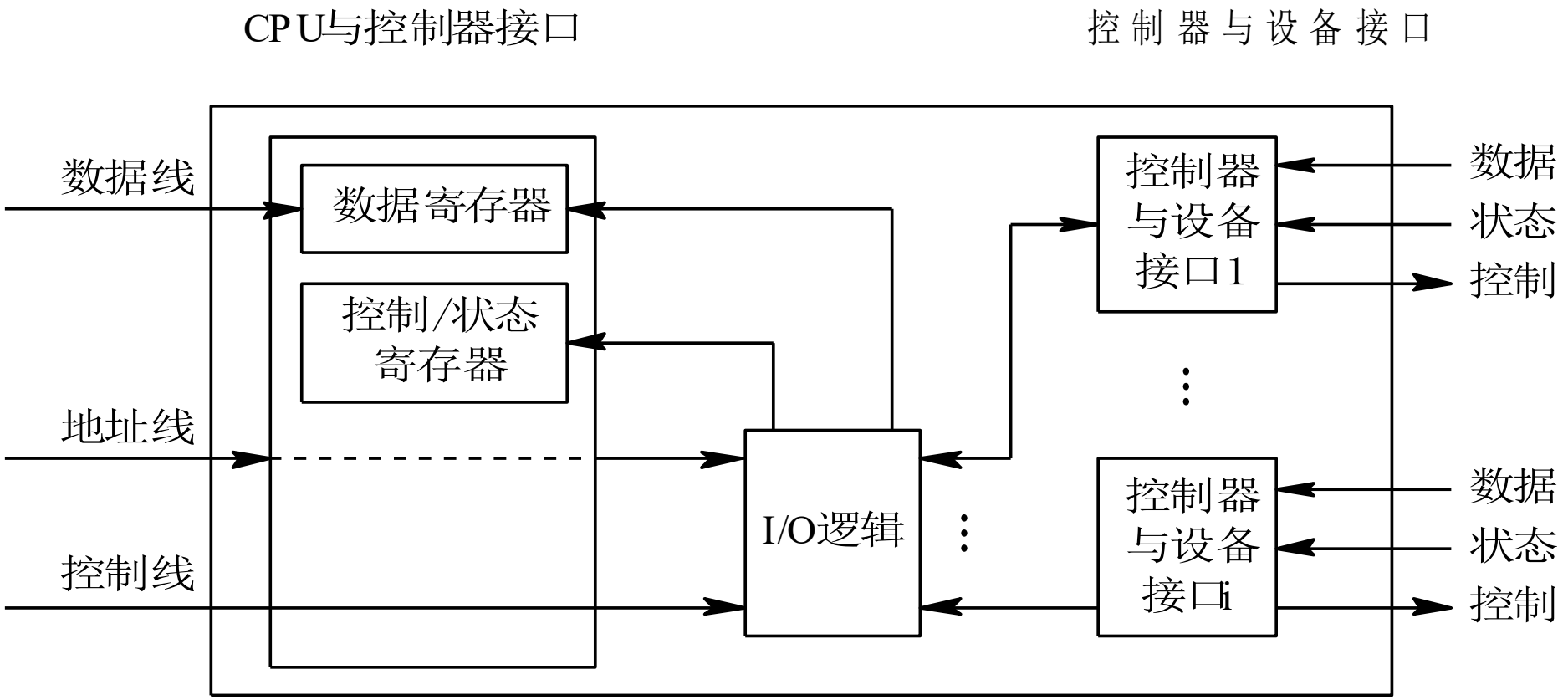
  – 用于存放从CPU送来的控制信息或设备的状态信息

② 设备控制器与设备的接口

- 在一个设备控制器上，可以连接一个或多个设备
- 相应地，在控制器中便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
- 控制器中的I/O逻辑根据CPU发来的地址信号去选择一个设备接口

③ I/O逻辑

- 在设备控制器中的I/O逻辑用于实现对设备的控制。
- 它通过一组控制线与CPU交互，CPU利用该逻辑向控制器发送I/O命令；I/O逻辑对收到的命令进行译码。
- 每当CPU要启动一个设备时，一方面将启动命令发送给控制器；另一方面又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。


CPU与控制器接口
控制器与设备接口
数据线
地址线
控制线
数据寄存器
控制/状态寄存器
I/O逻辑
控制器与设备接口1
控制器与设备接口i
数据
状态
控制
数据
状态
控制

设备控制器的组成

6.3 中断机构和中断处理程序

- 中断在OS中有着特殊且重要的地位

① 实现多道程序的基础

- 没有中断，不可能实现多道程序系统
- 因为进程之间的切换要通过中断来实现

② 实现设备管理的基础

③ 支持提高CPU利用率、CPU和I/O设备并行执行

- 中断
  - CPU对I/O设备发来的中断信号的一种响应
    - CPU暂停正在执行的程序，保留CPU环境后，自动转去执行该I/O设备的中断处理程序。
    - 执行完毕后，再回到中断点，继续执行原来的程序
  - 由于中断是由外部设备引起，故又称外中断

- 陷入
  - 由CPU内部事件引起的中断（溢出、程序出错等）
  - CPU也暂停正在执行的程序，保留CPU环境后，自动转去执行该陷入事件的处理程序。执行完毕后，再回到中断点，继续执行原来的程序
  - 由于中断是由CPU内部事件引起，故又称内中断或陷入
- 中断与陷入的主要区别：信号来源不同

- 中断向量表和中断优先级
  - 中断向量表
    - 通常为每种中断配以相应的中断处理程序，并把该程序在内存的位置（程序入口地址）集中放在一个表里（中断向量表），每个地址占一项
    - 当设备发来中断请求信号时，由中断控制器确定该请求的中断号，根据该中断号查找中断向量表，找到中断处理程序的入口地址，这样便可转入中断处理程序执行
  - 中断优先级
    - 为方便处理多个中断信号，系统利用优先级进行控制

- 中断处理程序的处理过程

① 测定是否有未响应的中断信号

② 保护被中断进程的CPU环境

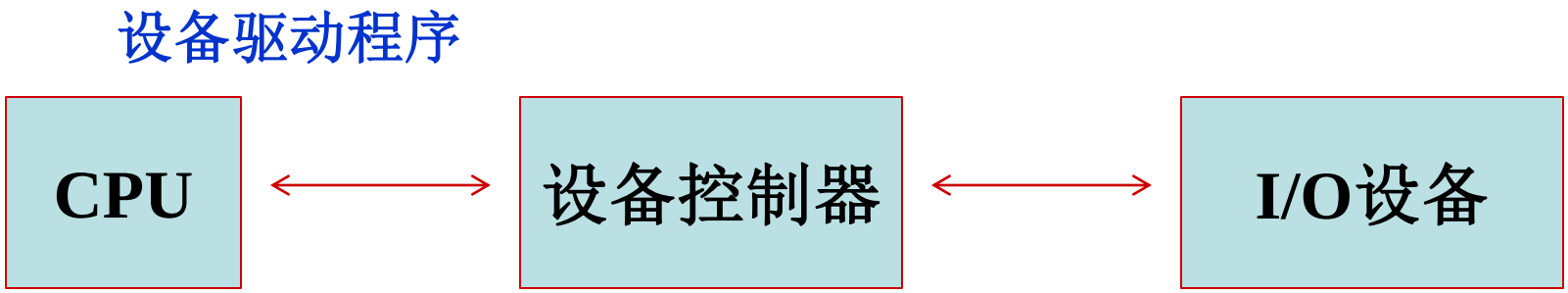
③ 转入相应的中断处理程序

④ 中断处理

⑤ 恢复CPU的现场并退出中断

6.4 设备驱动程序

- 又称为设备处理程序，I/O系统高层与设备控制器之间的通信程序



- 设备驱动程序的主要任务：
  - 接收上层发来的抽象I/O要求，如read或write，把它转换为具体要求后，发送给设备控制器，启动设备去执行
  - 它也将由设备控制器发来的信号传送给上层软件
- 由于设备驱动程序与硬件密切相关，故应为每一类设备配置一种驱动程序

1. 设备驱动程序的功能

① 接收由进程发来的I/O命令和参数，并将命令中的抽象要求转换为具体操作 (例如，将磁盘块号转换为磁盘的盘面、 磁道号及扇区号)

② 检查I/O请求的合法性，了解设备的状态，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，立即启动设备完成操作；如果设备忙碌，则将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，调用相应的中断处理程序

2. 设备驱动程序的特点

- 与一般程序的差异：

① 负责在请求I/O的进程与设备控制器之间进行通信和转换。

② 与控制器和I/O设备的硬件特性紧密相关，因而不同类型的设备的驱动程序一般不同。

③ 与I/O设备所采用的I/O控制方式紧密相关。(中断还是DMA)

④ 因为与硬件紧密相关，因而其中的一部分必须用汇编语言书写。

⑤ 允许重入：正在运行的驱动程序通常会在一次调用完成前被再次调用。

3. 设备处理方式

- 根据设备处理时设置进程的情况分类：

① 为每一类设备设置一个进程，专门用于执行这类设备的I/O操作。

② 在整个OS中设置一个I/O进程，专门用于执行OS中所有各类设备的I/O操作。(或分别设置输入、输出进程)

③ 不设置专门的设备处理进程，只为各类设备设置相应的设备处理程序(模块)，供用户进程或OS进程调用。

4. 设备驱动程序的处理过程

① 将抽象要求转换为具体要求

② 对服务请求进行校验

③ 检查设备的状态

④ 传送必要的参数

⑤ 启动I/O设备

- 具体的I/O操作在设备控制器的控制下进行
- CPU和设备可以并行

5. 对I/O设备的控制方式

- I/O控制方式有几个发展阶段

① 使用轮询的可编程I/O方式（程序控制方式）

② 使用中断的可编程I/O方式（中断驱动方式）

③ 直接存储器访问方式（DMA方式）

④ I/O通道控制方式

① 使用轮询的可编程I/O方式

- 早期的计算机系统中，CPU对设备的控制采取程序轮询的控制方式，或称为忙—等待方式。
- 基本思想

1. CPU发出一条I/O指令启动输入设备输入数据时，要同时令状态寄存器中的忙/闲标志busy置1，并循环检查busy的值。
2. 当busy=1时，表示尚未完成一个字(符)的输入，此时CPU应继续对busy进行检查，直至busy=0，表明已完成数据的输入。
3. 于是，CPU将数据寄存器中的数据取出，送入内存中，这样便完成了一个字(符)的I/O。
4. 接着再去启动读下一个数据，并置busy=1。如此反复下去。

② 使用中断的可编程I/O方式

- 现代计算机系统中，都毫无例外地引入了中断机构，因此对I/O设备的控制，广泛采用中断驱动方式
- 基本思想
  - 进程要启动I/O设备工作时，由CPU向该设备控制器发出一条I/O命令，然后便可立即返回继续执行原来的任务。
  - 设备控制器按照命令要求去控制指定设备。此时，CPU与I/O设备并行操作。
  - 设备处理完数据后，便产生一个中断信号。此时，CPU便转而处理该信号。

- 在I/O设备处理每个数据的过程中，由于无需CPU干预，因而可使CPU与I/O设备并行工作。
- 仅当处理完一个数据时，才需CPU花费极短的时间去做些中断处理。
- 可见，这样可使CPU和I/O设备都处于忙碌状态，从而提高了整个系统的资源利用率及吞吐量。

③直接存储器访问(DMA)方式

- 虽然中断驱动方式比程序控制方式更有效，但它仍是以字(节)为单位进行I/O的，每当完成一个字(节)的I/O时，控制器便要向CPU发出一次中断。
- 换言之，采用中断驱动方式时的CPU是以字(节)为单位进行干预的。如果将这种方式用于块设备的I/O，显然效率极低。
- 例如，为了从磁盘中读出1 KB的数据块，CPU会被中断 1K次。
- 为了进一步减少CPU对I/O的干预而引入了DMA方式

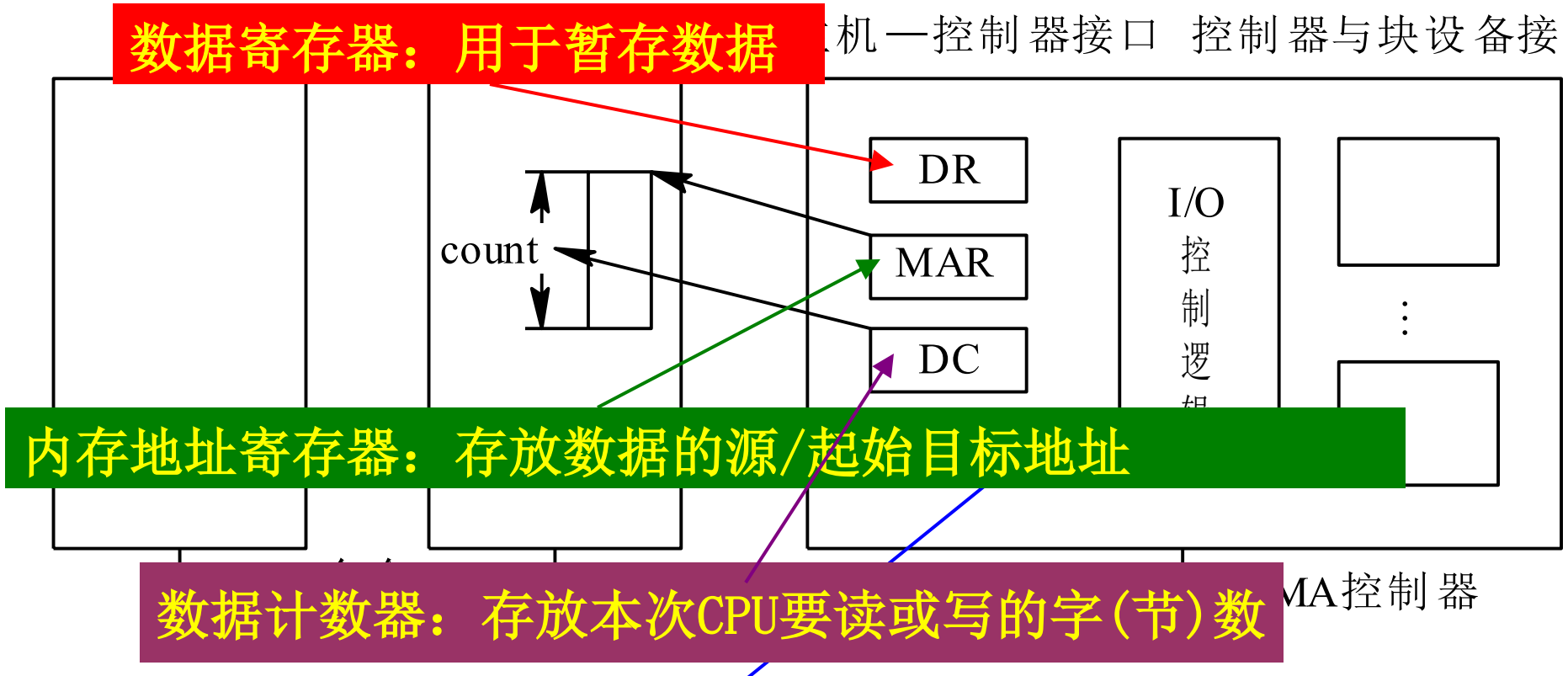
- DMA方式的特点
  - ① 传输的基本单位——数据块。CPU与I/O设备之间每次传送至少一个数据块
  - ② 所传送的数据从设备直接送入内存，或者相反
  - ③ 仅在传送一个或多个数据块的开始和结束时，才需CPU干预。传送是在控制器的控制下完成的。
  - 可见，DMA方式较之中断驱动方式，又成百倍地减少了CPU对I/O的干预，进一步提高了CPU与I/O设备的并行程度。

- **DMA控制器的组成**
  - **DMA控制器由三部分组成**

    ① 主机与DMA控制器的接口

    ② DMA控制器与块设备的接口

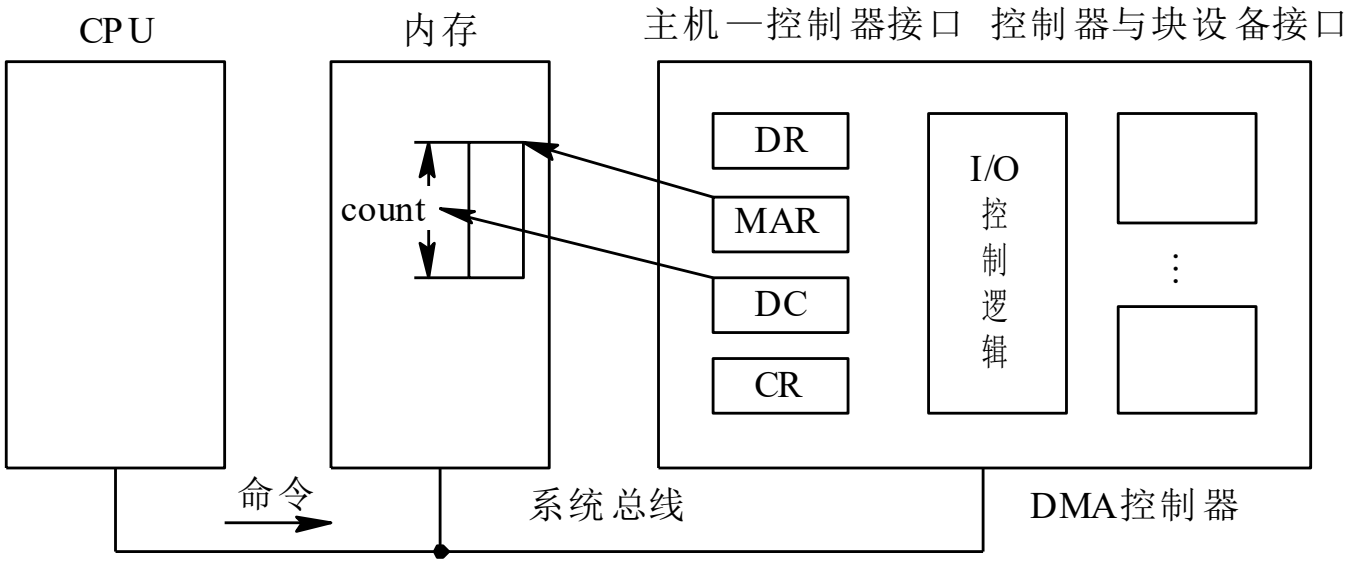
    ③ I/O控制逻辑


数据寄存器：用于暂存数据
机—控制器接口 控制器与块设备接口
DR
count
MAR
DC
I/O
控
制
逻
辑
内存地址寄存器：存放数据的源/起始目标地址
MA控制器
数据计数器：存放本次CPU要读或写的字(节)数
命令/状态寄存器：接收从CPU发来的I/O命令
或有关控制信息，或设备状态

DMA控制器的组成

DMA工作过程

- 以从磁盘读入数据为例：
  - ① 从磁盘读入数据块时，CPU将一条读指令送至磁盘控制器的命令寄存器(CR)中。
  - ② 将内存起始目标地址送入内存地址寄存器（MAR）中；将欲读取字(节)数送入数据计数器(DC)中
  - ③ 将磁盘的源地址送至DMA控制器的I/O逻辑中。
  - ④ 然后，启动DMA控制器进行数据传送，以后，CPU便可去处理其它任务。
  - ⑤ 此后，整个数据传送过程便由DMA控制器进行控制。

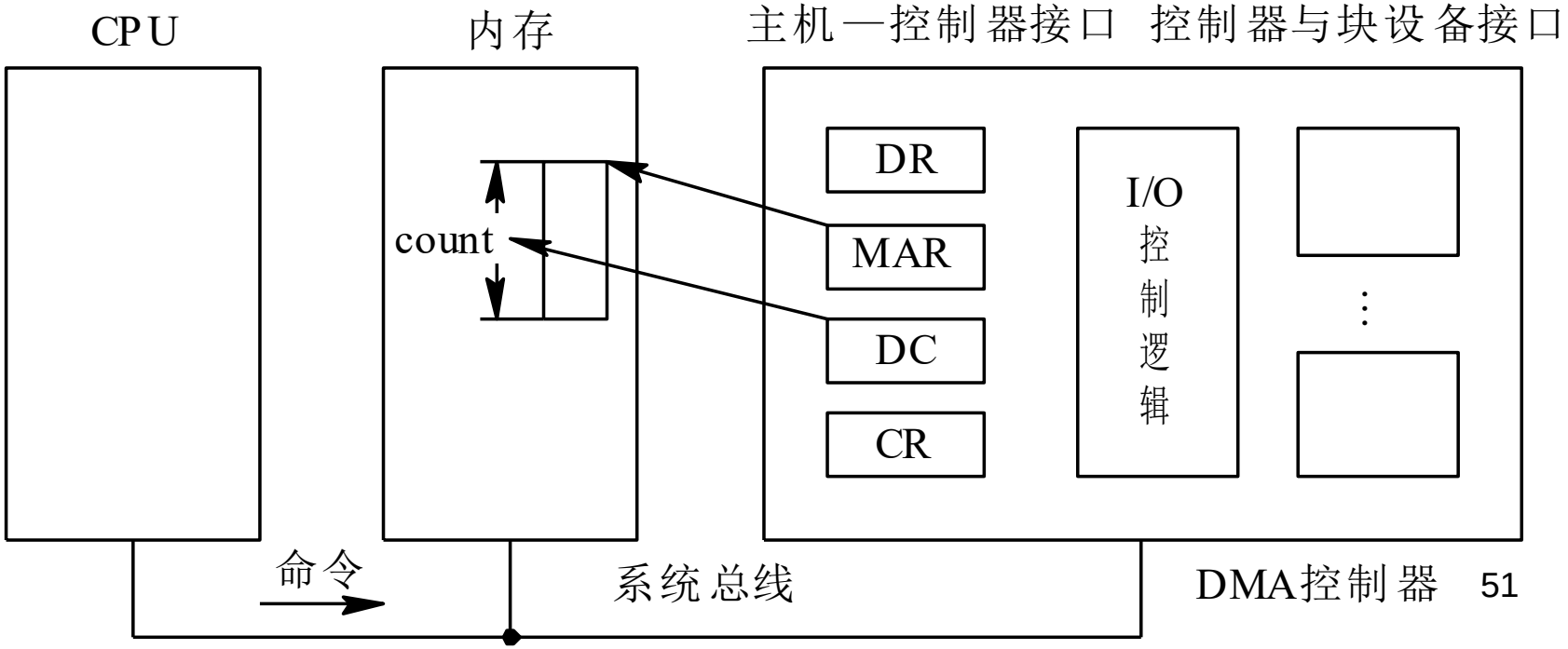


⑥ 当DMA控制器已从磁盘读入一个字(节)的数据并送入数据寄存器(DR)后，将该字(节)传送到指定的内存单元中。

⑦ 接着便对MAR内容加1，将DC内容减1。

⑧ 若减1后DC内容不为0，表示传送未完，便继续传送下一个字(节)；

⑨ 否则，由DMA控制器发出中断请求。



- **DMA方式与中断方式的主要区别**

① 中断的时机不同

- 中断方式是在数据寄存器满后，发中断请求，CPU进行中断处理
- DMA方式是在所要求传送的数据块全部传送结束时要求CPU进行中断处理，大大减少了CPU进行中断处理的次数

② CPU控制方法不同

- 中断方式的数据传送是由CPU控制完成的
- DMA方式是在DMA控制器的控制下不经CPU控制完成的

④ I/O通道控制方式

- **I/O通道控制方式的引入**
  - 虽然DMA方式比中断方式已经显著地减少了CPU的干预，但CPU每发出一条I/O指令，也只能去读(或写)一个数据块。
  - 当我们需要一次去读多个数据块，且将它们分别传送到不同的内存区域，或者相反时，则须由CPU发出多条I/O指令，并进行多次中断处理才能完成。
  - 在这种情况下，DMA方式会影响系统的效率和吞吐量
  - I/O通道方式是DMA方式的发展，它可进一步减少CPU的干预

- 因此，在CPU和设备控制器之间增设通道，其主要目的
  - 使一些原来由CPU处理的I/O任务转由通道来承担，从而把CPU从繁杂的I/O任务中解脱出来。
- 通道是一种特殊的处理机，特点：

① 指令类型单一，主要局限于对I/O操作

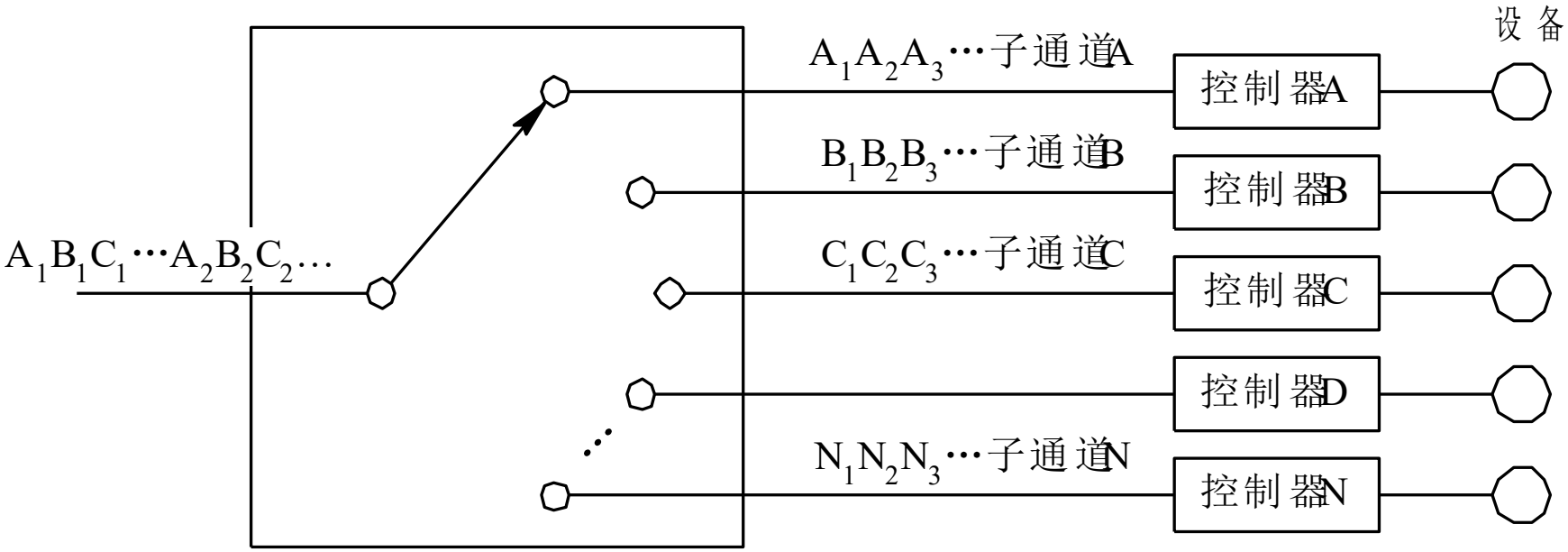
② 没有自己的内存，通道程序放在内存里

- 在设置了通道后，CPU只需向通道发送一条I/O指令。
- 通道在收到该指令后，便从内存中取出本次要执行的通道程序。
- 然后执行该通道程序。
- 仅当通道完成了规定的I/O任务后，才向CPU发中断信号。

- 通道类型

① 字节多路通道

- 子通道连接一个设备，按时间片共享主通道(以字节为单位)



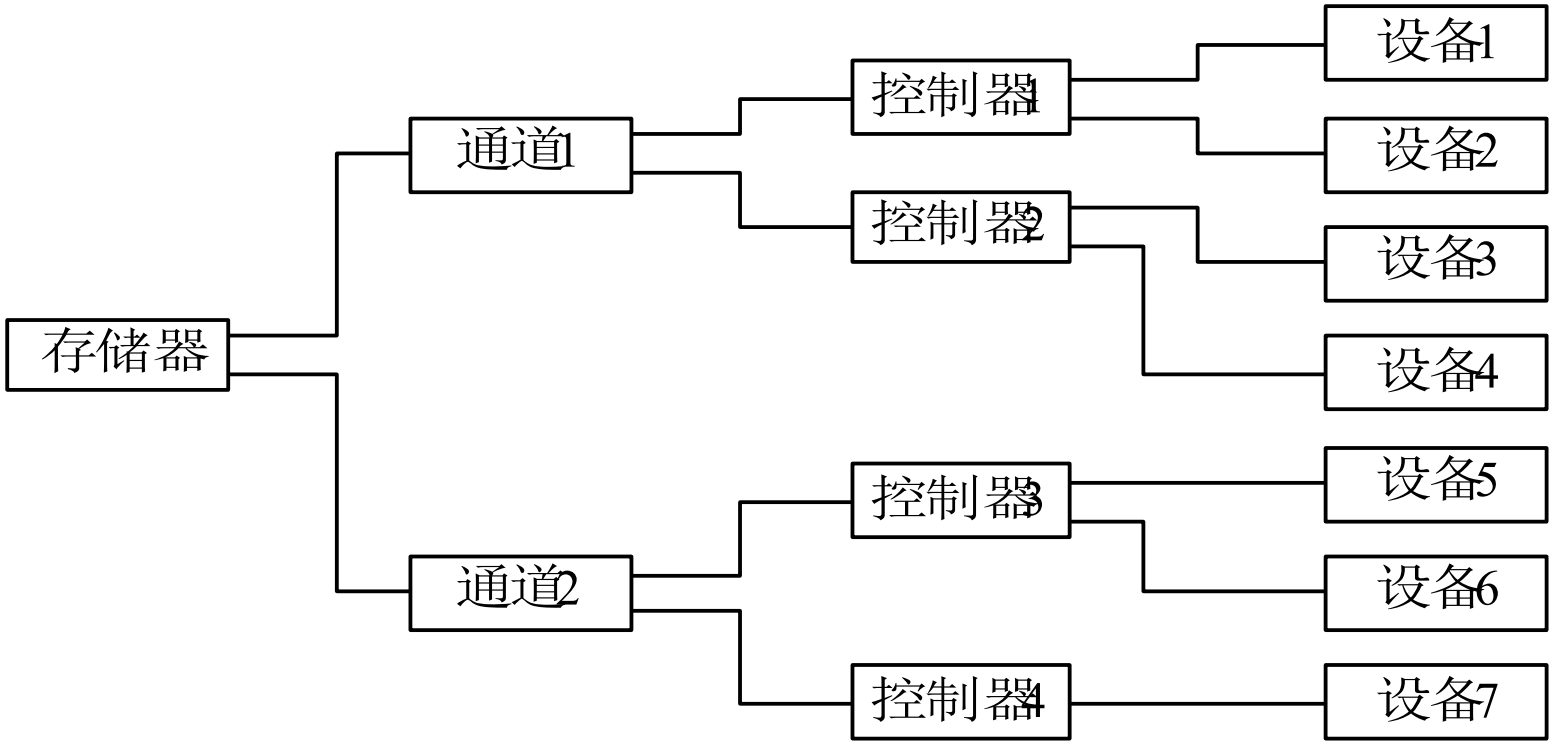
②数组选择通道

- 只有一个子通道，一段时间内只能执行一道程序，控制一台设备，利用率低。

③数组多路通道

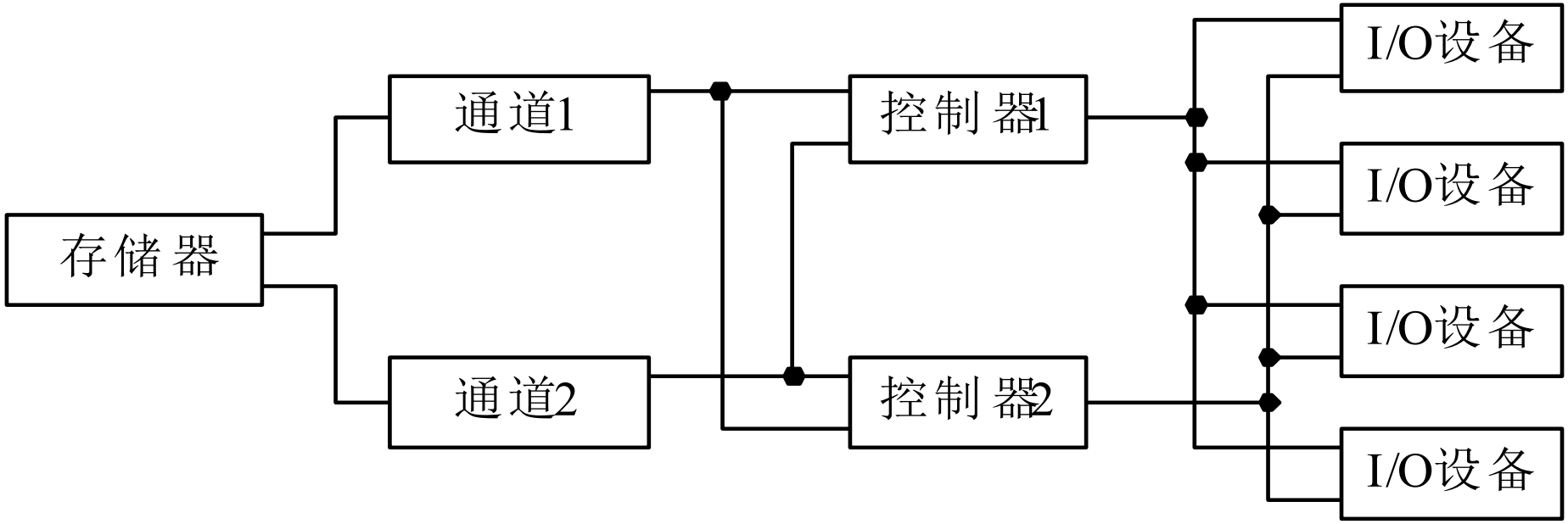
- 数组选择通道+字节多路通道。（广泛用于多台高、中速设备）。

- “瓶颈”问题
  - 通道价格昂贵，配备不可能太多，这造成了I/O瓶颈。



– 解决办法：增加通路

- 即，一个设备连接多个控制器，一个控制器连接多个通道。
- 这解决了瓶颈问题，提高了系统可靠性



多通路I/O系统

- 通道程序
  - 通道程序是由一系列通道指令构成的
  - 通道指令与一般的机器指令不同，在它的每条指令中都包含下列诸信息：
    ① 操作码：规定指令所执行的操作，如读/写/控制等
    ② 内存地址：标明数据送入内存(读操作)和从内存取出(写操作)时的内存首址
    ③ 计数：表示指令所要读(或写) 的字节数
    ④ 通道程序结束位P：表示通道程序是否结束。P=1表示本指令是通道程序的最后一条指令
    ⑤ 记录结束标志R：R=0表示本指令与下一条指令所处理的数据是同属于一个记录；R=1表示这是处理某记录的最后一条指令

- 写操作(从内存取数据)

多条指令操作的内存写到一起
通过R的值来识别

操作	P	R	计数	内存地址
Write	0	0	80	813
Write	0	0	140	1034
Write	0	1	60	5830
Write	0	1	300	2000
Write	0	0	250	1850
Write	1	1	250	720

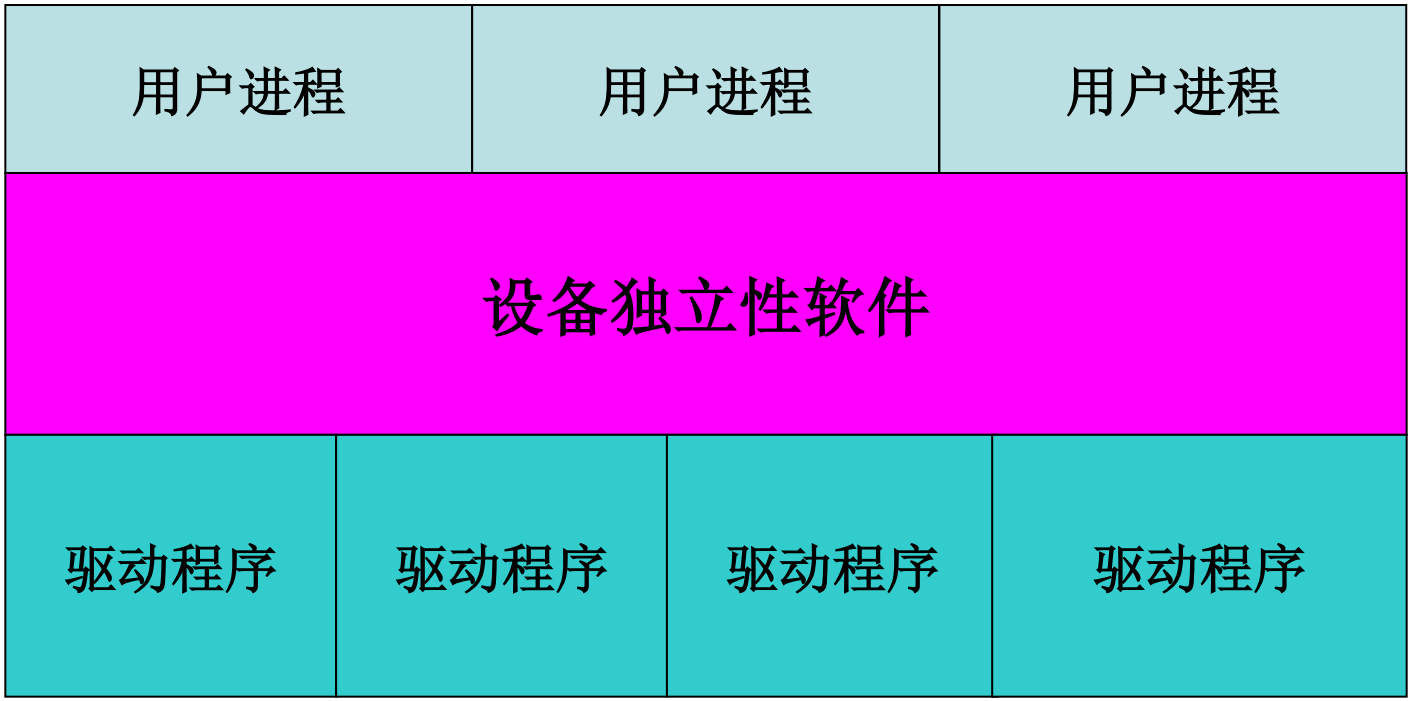
6.5 与设备无关的I/O软件

- 为方便用户和提高OS的可适应性和可扩展性，在现代OS的I/O系统中，都无一例外的增加了与设备无关的I/O软件，以实现设备独立性（设备无关性）
- 引入的目的
  - 可使应用程序中所用的设备，不局限于某个具体的物理设备。灵活！

1. 与设备无关软件的基本概念

- 早期OS中运行的应用程序如何使用设备：
  - 在应用程序中，直接使用具体设备名称，这会使应用程序与具体的设备直接相关。
  - 缺点：
    - 当该设备被占用或拆除，即使系统中还有空闲的同类设备，应用程序也无法使用。
    - 非常不灵活，也不利于提高设备利用率。

- 为实现应用程序与设备的无关性，引入逻辑设备和物理设备两个概念
  - 逻辑设备：抽象的设备名，如/dev/printer
  - 物理设备：具体的设备
- 在应用程序中，使用逻辑设备名请求某类设备；OS在实际执行时，使用物理设备名。


用户进程
用户进程
用户进程
设备独立性软件
驱动程序
驱动程序
驱动程序
驱动程序

- 设备独立性的好处

  a) 设备分配时的灵活性

  - 使用逻辑设备名称，进程申请时，OS可把任何一台同类设备分配给它

  b) 易于实现I/O重定向

  - 用于I/O操作的设备可以更换，而无需修改应用程序

- 为实现设备无关性，OS中必须配置一张逻辑设备表，以便实现逻辑设备名到物理设备名的转换

2. 设备无关性软件的功能

① 设备驱动程序的统一接口

② 缓冲管理

③ 差错控制

④ 对独占设备的分配与回收

⑤ 独立于设备的逻辑数据块

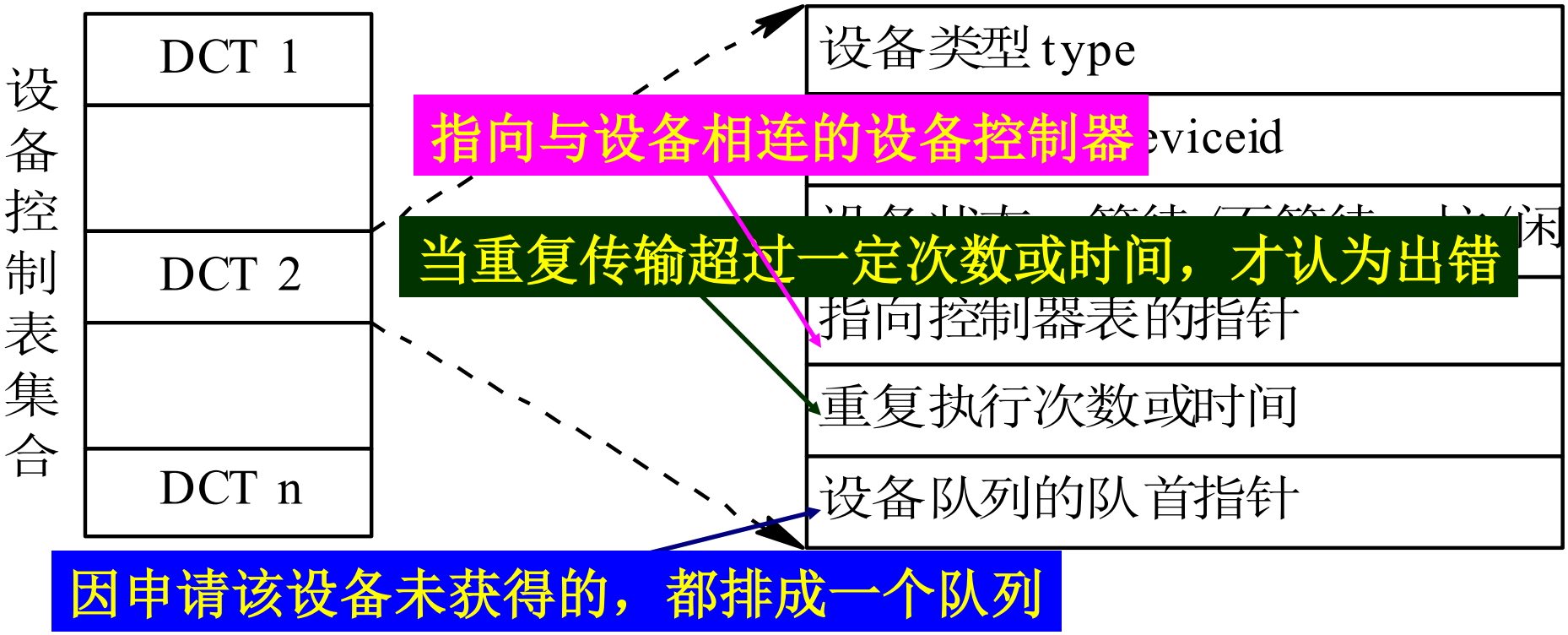
3. 设备分配

- 为防止设备被无序使用，它们必须由OS统一分配
- 进程要使用设备时，不是直接使用设备，而是向OS提出申请。
- 每当进程提出分配设备的申请，只要是安全和可能的，OS便按一定的策略，进行分配
- 为了便于分配，OS必须维护与设备相关的信息

- 设备分配中的数据结构

① 设备控制表(DCT)

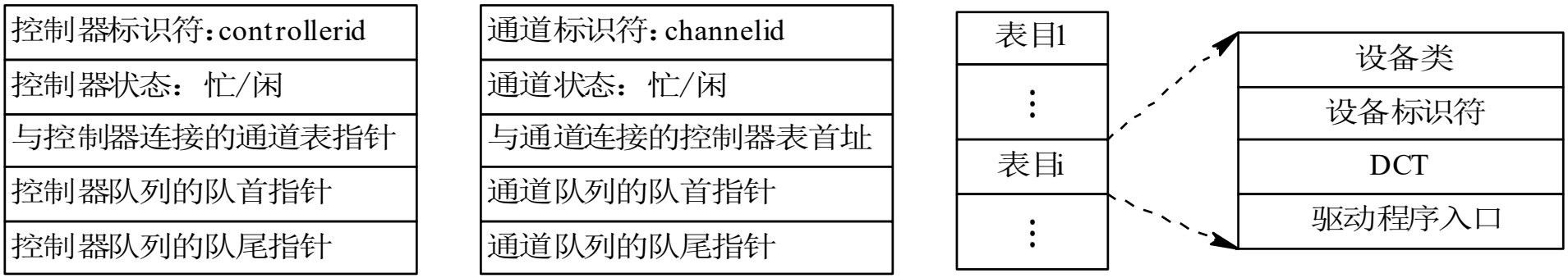
- OS为每一个设备都配置了一张设备控制表，用于记录该设备的情况



设备控制表DCT

② 控制器控制表、通道控制表和系统设备表

- 控制器控制表(COCT)
  - OS为每一个控制器设置一张记录其情况的控制表
- 通道控制表(CHCT)
  - 每个通道都配有一张控制表
- 系统设备表(SDT)
  - 这是系统范围的数据结构，记录了系统中全部设备的情况。
  - 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、设备驱动程序的入口等表项



(*a*) 控制器表COCT　　(*b*) 通道表CHCT　　(*c*) 系统设备表SDT

COCT、 CHCT和SDT表

- 设备分配时应考虑的因素
  - 为使OS有条不紊地工作，OS在分配设备时，应考虑这样几个因素:

    ① 设备的固有属性

    ② 设备分配算法

    ③ 设备分配时的安全性

① 设备的固有属性

- 设备的固有属性分成3种:
  - 独占性
    - 指这种设备在一段时间内只允许一个进程独占，此即“临界资源”
  - 共享性
    - 指这种设备允许多个进程“同时”共享
  - 可虚拟性
    - 指设备本身虽是独占设备，但经过某种技术处理，可以把它改造成虚拟设备。
- 对上述的独占、共享、可虚拟三种设备应采取不同的分配策略

② 设备分配算法

- 与进程调度的算法有些相似，但相对简单，通常只采用以下两种算法：
  - 先来先服务
    - 有多个进程对同一设备提出I/O请求时，根据提出请求的先后次序，将它们排成一个设备请求队列，设备分配程序总是把设备首先分配给队首进程
  - 优先级高者优先
    - 将优先级高的进程排在设备队列前面，而对于优先级相同的I/O请求，则按先来先服务原则排队

③ 设备分配中的安全性

– 从进程运行的安全性考虑，设备分配有以下两种方式。

① 安全分配方式

- 进程发出I/O请求后，便进入阻塞状态，直到其I/O操作完成时才被唤醒。
- 这种方式下，一旦进程已获得某种设备后便阻塞，使该进程不可能再请求任何资源，而在它运行时又不保持任何资源。
- 优点：这种方式破坏了造成死锁的四个必要条件之一的“请求和保持”条件，从而使设备分配是安全的。
- 缺点：进程进展缓慢，即CPU与I/O设备是串行工作的

② 不安全分配方式

- 在这种方式中，进程发出I/O请求后仍继续运行，需要时又可发出其他I/O请求。
- 仅当进程请求的设备已被另一进程占用时，才进入阻塞状态。
- 优点：进程可同时操作多个设备，使其推进迅速
- 缺点：分配不安全，可能导致死锁
- 因此，在设备分配程序中，还应再增加一个功能，以用于对本次设备分配是否会发生死锁进行安全性计算，仅当计算结果说明分配是安全的情况下才进行设备分配。

- 独占设备的分配程序

① 基本的设备分配程序

a) 分配设备

- 首先根据I/O请求中的物理设备名，查找系统设备表(SDT)，从中找出该设备的DCT，再根据DCT中的设备状态字段，可知该设备是否正忙。
- 若忙，便将请求I/O进程的PCB挂在设备队列上；
- 否则，便按照一定的算法计算本次设备分配的安全性。
- 如果不会导致系统进入不安全状态，便将设备分配给请求进程；
- 否则，仍将其PCB插入设备等待队列。

b) 分配控制器

- OS把设备分配给请求进程后，再到其DCT中找出与该设备连接的控制器的COCT，从中的状态字段可知该控制器是否忙碌。
- 若忙，便将进程PCB挂在该控制器的等待队列上；否则，便将之分配给进程

c) 分配通道

- 在该COCT中又可找到与该控制器连接的通道的CHCT，从中的状态信息，可知该通道是否忙碌。
- 若忙，便将进程PCB挂在该通道的等待队列上；否则，将之分配给进程

- 只有在设备、 控制器和通道三者都分配成功时，这次的设备分配才算成功

② 设备分配程序的改进

- 仔细研究上述基本的设备分配程序后可以发现:

  a) 进程是以物理设备名来提出I/O请求的;

  b) 采用的是单通路的I/O系统结构，容易产生“瓶颈”现象。

- 为此，应从以下两方面对基本的设备分配程序加以改进，以使独占设备的分配更灵活，提高分配的成功率。

  a) 增加设备的独立性

  b) 考虑多通路情况

- **改进方法:**
  - 程序中使用逻辑设备名。（这样，只要有一个该类设备可用，OS便进一步计算分配该设备的安全性。）
  - 仅当所有设备(控制器、通道)都忙时，分配才算失败，才把进程挂在相应的等待队列上。只要有一个设备(控制器、通道)可用，系统便可将它分配给进程。

4. 逻辑设备名到物理设备名映射的实现

- 逻辑设备表LUT
  - 用于实现逻辑设备名到物理设备名的映射
  - 进程用逻辑设备名请求设备时，OS为之建一个LUT表项。
  - 在其中填上该逻辑设备名和对应的物理设备名。
  - 以及该物理设备的驱动程序的入口地址。

- LUT可以有两种设置方法：

  a）系统中设置一张统一的表

  b）为每个用户设置一张表

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	2046
⋮	⋮	⋮

(*a*)

所有用户必须使用不同的逻辑设备名，在多用户环境下很难做到。

逻辑设备名	系统设备表指针
/dev/tty	3
/dev/printer	5
⋮	

(*b*)

可以实现每个用户使用相同的逻辑设备名表示不同的物理设备。

逻辑设备表

6.6 用户层的I/O软件

1. 系统调用与库函数

① 系统调用

- 应用程序可以通过系统调用间接地调用OS中的I/O过程，对I/O设备进行操作

② 库函数

- 在程序中使用库函数的方式调用系统调用

2. SPOOLing技术

- 目的
  - 可以将一台物理设备虚拟为多台逻辑设备，以允许多个用户共享一台物理设备

- 20世纪50年代，为缓和CPU的高速性和I/O设备的低速性之间的矛盾，引入脱机输入/输出技术
  - 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CPU需要输入数据时，就可以直接从磁盘上读取
  - 极大提高了输入速度
  - 反之，也可提高输出数据时的速度

- SPOOLing技术的实质是模拟脱机输入/输出
  - 用一个进程将设备输入的数据暂存到磁盘上
  - 用另一个进程把暂存磁盘上的输出数据传给设备
  - 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，或称为假脱机技术

- **SPOOLing系统的组成**
  - **SPOOLing系统必须建立在具有多道程序功能的OS上**
  - 应有高速随机外存（通常为磁盘）的支持

- **SPOOLing系统主要有以下四部分：**

① 输入井、输出井

- 磁盘上的两块存储区，用于暂存输入、输出的数据

② 输入缓冲区、输出缓冲区

- 位于内存中，作用是缓和CPU、设备之间的速度差异

③ 输入进程、输出进程

- 输入进程：利用输入缓冲区为中介，把输入设备的数据存入输入井
- 输出进程：设备空闲时，将输出井中的数据利用输出缓冲区送入设备

④ 井管理程序

- 控制任务与井之间信息的交换

- 利用SPOOLing技术实现共享打印机
  - 打印机是经常用到的输出设备，属于独占设备。
  - 利用SPOOLing技术，可将之改造为一台可供多个用户共享的设备，从而提高设备的利用率，也方便了用户。
  - 共享打印机技术已被广泛地用于多用户系统和局域网络中。

- 组成部分

① 磁盘缓冲区（输出井）

② 打印缓冲区（输出缓冲区）

③ 假脱机管理进程

④ 假脱机打印进程

- 进程请求打印时，假脱机打印系统并不是立即把打印机分配给该进程，而是由假脱机管理进程做以下两件事：

  ① 在磁盘缓冲区中申请一块空闲磁盘区，并将要打印的数据存入其中；

  ② 为该进程申请一张空白的请求打印表，并将打印请求填入其中；再将该表挂到假脱机文件队列上。

- 如果还有其他进程要求打印，系统仍可接受其请求，并同样为它们做上述两件事。

- 如果打印机空闲，假脱机打印进程从假脱机文件队列取出队首的打印表
- 根据表中的要求将数据，从磁盘缓冲区传送到打印缓冲区，再由打印机进行打印。
- 打印完后，该进程再查看队列中是否还有等待打印的请求表。
- 若有，再取出队首的表，并根据要求打印，如此下去。直至队列为空，该进程才将自己阻塞起来。仅当下次再有打印请求时，该进程才被唤醒。

- 打印守护进程
  - 改进上述假脱机打印系统，为打印机建立一个打印守护进程，它是允许使用打印机的唯一进程。
  - 原来假脱机管理进程的功能分为两部分：
    ① 一部分由打印守护进程完成
    ② 另一部分由请求打印的进程自己完成
- 需要将独占设备改造为可供多个进程共享的设备时，都可以为之设置一个守护进程（**daemon**）。
- 守护进程是允许使用该独占设备的唯一进程。

- **SPOOLing系统的特点**

① 提高了I/O的速度

② 将独占设备改造为共享设备

③ 实现了虚拟设备功能

6.7 缓冲区管理

- 在现代OS中，为实现与I/O设备的数据交换，都使用了缓冲区作为数据传输媒介。
- 缓冲区是一个存储区域
- 可以用专门的寄存器组成，但成本较高，容量有限
- 更多时候是在内存中开辟空间作为缓冲区。

1. 缓冲的引入

① 缓和CPU与I/O设备间速度不匹配的矛盾

- CPU速度远高于I/O设备
- 没有缓冲区时，传输数据时，由于设备速度跟不上会使CPU停下来等待。
- 设置缓冲区后，数据可以暂存在缓冲区中，以缓和CPU和设备的速度不匹配矛盾。
- 事实上，凡在数据到达速率与发送速率不同的地方，都可设置缓冲区，以缓和它们之间速率不匹配的矛盾。

② 减少对CPU的中断频率

- 在远程通信系统中，缓冲区满时，CPU必须中断一次以处理缓冲区中的数据。
- 而且如果CPU不及时响应，则会造成后续数据冲掉当前缓冲区数据的问题。
- 所以增加缓存区，则放宽了CPU中断响应时间的限制。

③ 解决生产者、消费者之间数据粒度（数据单元大小）不匹配问题

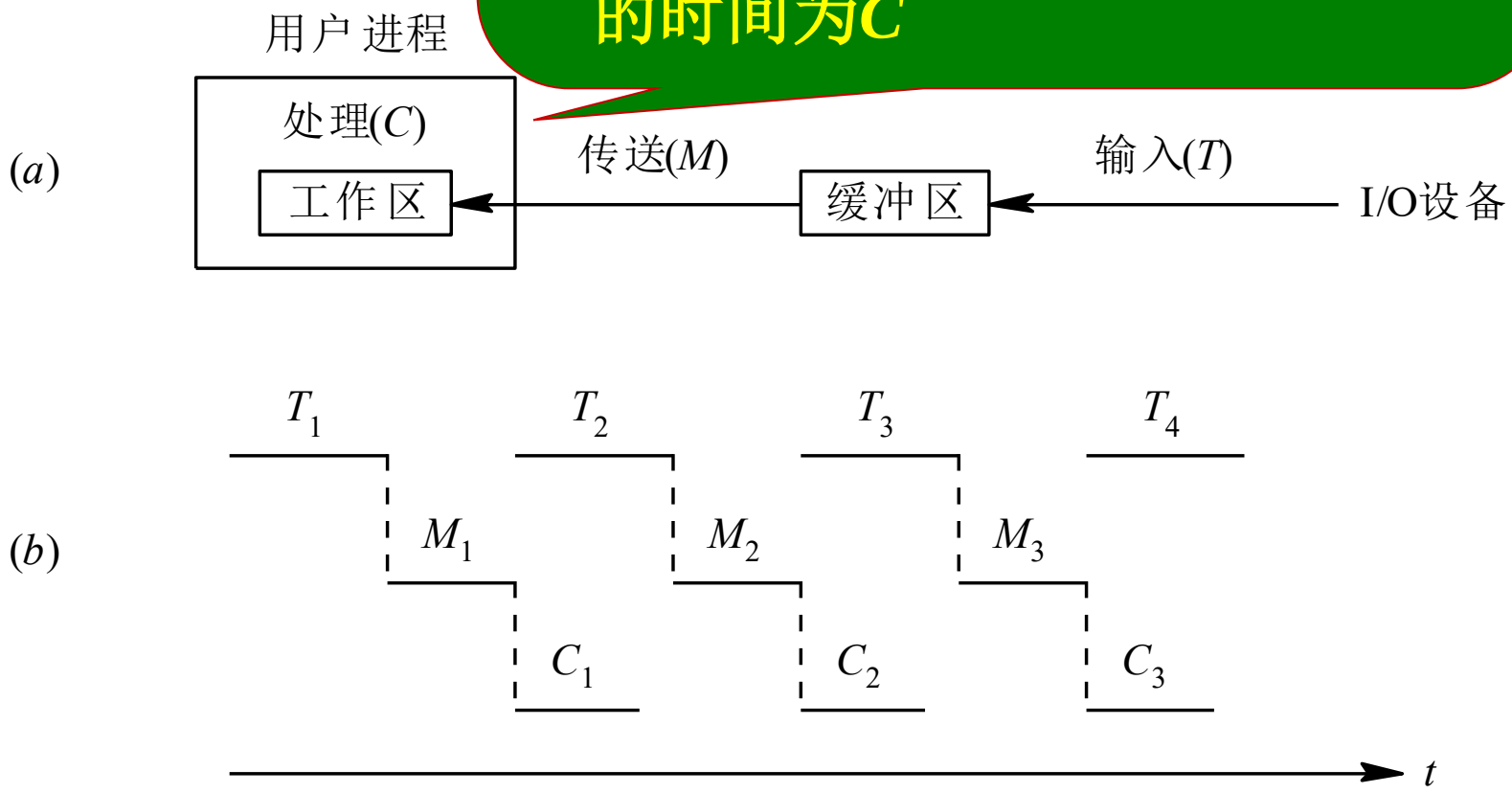
④ 提高CPU和I/O设备之间的并行性

- 可显著地提高CPU和I/O设备间的并行操作程度，提高系统的吞吐量和设备的利用率。

在块设备输入时，假定从磁盘把

由于*T*和*C*可以并行，
T>*C*时，系统对每一块数据的处理时间为*M*+*T*，反之为*M*+*C*，
故可把系统对每一块数据的处理时间表示为：Max(*C*, *T*)+*M*

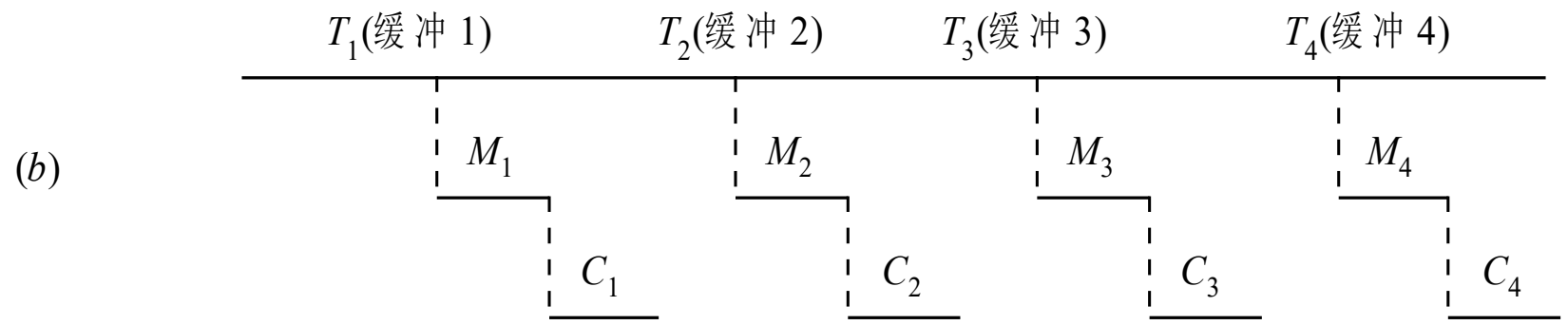
的时间为*C*



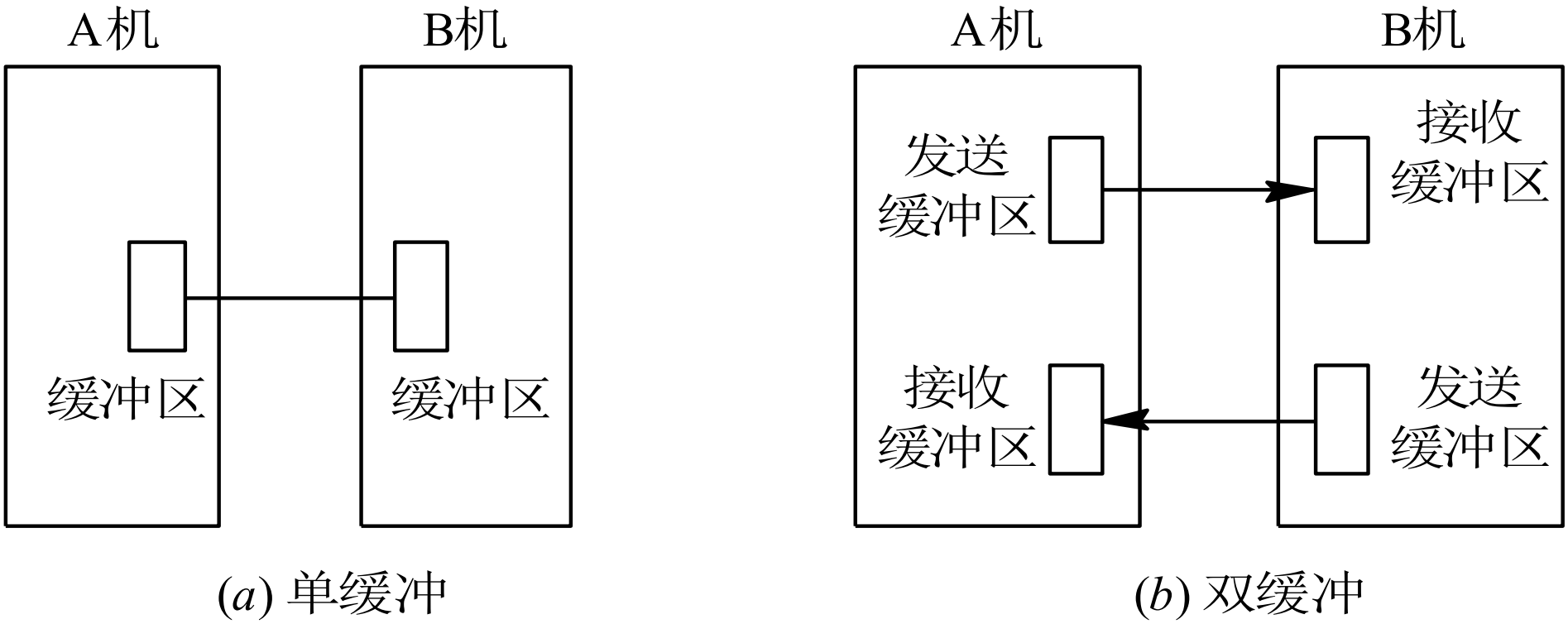
单缓冲工作示意图

设备输入时，先将数据送入缓冲区1，装满后转向缓冲区2
此时OS可从缓冲区1中读取数据，并送入用户进程。

系统处理一块数据的时间可以粗略地认为是Max(C，T)
如果C<T，可使块设备连续输入；
如果C>T，可使CPU不必等待设备输入。



双缓冲工作示意图



(*a*) 单缓冲　　(*b*) 双缓冲

双机通信时缓冲区的设置

3. 循环缓冲（环形缓冲）

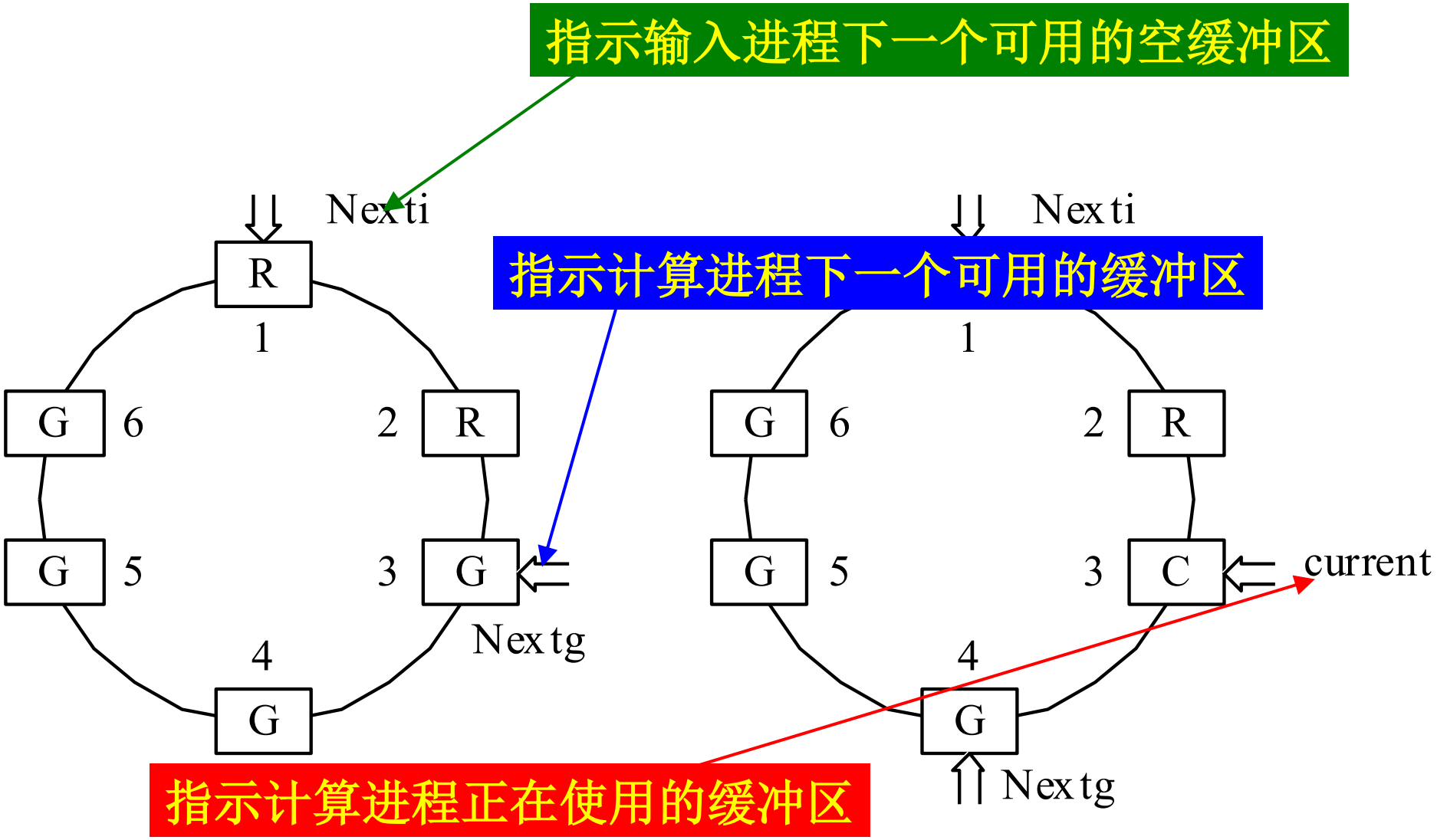
① 循环缓冲的组成

a) 多个缓冲区

- 在循环缓冲中包括多个缓冲区，每个缓冲区的大小相同
- 作为输入的多缓冲区可分为三种类型：
  - 空缓冲区R
  - 已装满数据的缓冲区G
  - 计算进程正在使用的工作缓冲区C

b) 多个指针

- 作为输入的缓冲区可设置三个指针
  - **Nexti:** 指示输入进程下次可用空缓冲区**R**
  - **Nextg:** 指示计算进程下一个可用缓冲区**G**
  - **Current:** 指示计算进程正使用的缓冲区**C**


指示输入进程下一个可用的空缓冲区
Nexti
R
1
G
6
2
R
G
5
3
G
Nextg
4
G
指示计算进程下一个可用的缓冲区
Nexti
1
G
6
2
R
G
5
3
C
current
4
G
Nextg
指示计算进程正在使用的缓冲区

循环缓冲

② 循环缓冲区的使用

- 计算进程和输入进程可利用两个过程使用循环缓冲区:

a) Getbuf过程

- 计算进程要使用缓冲区的数据时，将**Nextg**指向的缓冲区提供给进程使用，将其改为正在使用缓冲区，令**current**指向该区的第一个单元，再令**Nextg**指向下一个**G**缓冲区。**(**输入进程，类似**)**

b) Releasebuf过程

- 读取完缓冲区的数据后，由计算进程调用。将缓冲区释放，将之改为空缓冲区**R**。**(**输入进程，类似**)**

③ 进程同步

- 使用输入循环缓冲，指针**Nexti**和指针**Nextg**将不断地沿着顺时针方向移动，这样就可能出现下述两种情况:

a) Nexti指针追赶上Nextg指针

- 表示输入数据的速度大于处理数据的速度，已无空缓冲区可用。
- 此时，输入进程应阻塞，直到计算进程把某个缓冲区中的数据读取完，使之成为空缓冲区**R**，并调用**Releasebuf**过程将它释放时，才将输入进程唤醒。
- 这种情况被称为系统受计算限制

b) Nextg指针追赶上Nexti指针

- 表示输入数据的速度低于处理数据的速度，此时无装满数据的缓冲区供计算进程使用。
- 这时，计算进程只能阻塞，直至输入进程又装满某个缓冲区，并调用**Releasebuf**过程将它释放时，才去唤醒计算进程。
- 这种情况被称为系统受**I/O**限制

4. 缓冲池

- 前面讲的缓冲区都是专用缓冲，是为完成某个任务专门开辟的。因此，任务较多时，开销会很大，而且利用率低。
- 可以设立公用缓冲池，池中设立多个缓冲区，为多个进程共享，以提高利用率。
- 缓冲区、缓冲池的区别：
  - 缓冲区：单个内存区域或一组内存区域组成的链表
  - 缓冲池：包含一个管理的数据结构、一组操作函数的管理机制，用于管理多个缓冲区

① 缓冲池的组成

- 为管理方便，一般将相同类型的缓冲区链成一个队列，于是形成以下三个队列:

  ① 空缓冲队列 **emq**

  ② 输入队列 **inq**：装满输入数据的缓冲区组成

  ③ 输出队列 **outq**：装满输出数据的缓冲区

- 此外，还应具有四种工作缓冲区：

  ① 用于接收输入数据的工作缓冲区

  ② 用于提取输入数据的工作缓冲区

  ③ 用于接收输出数据的工作缓冲区

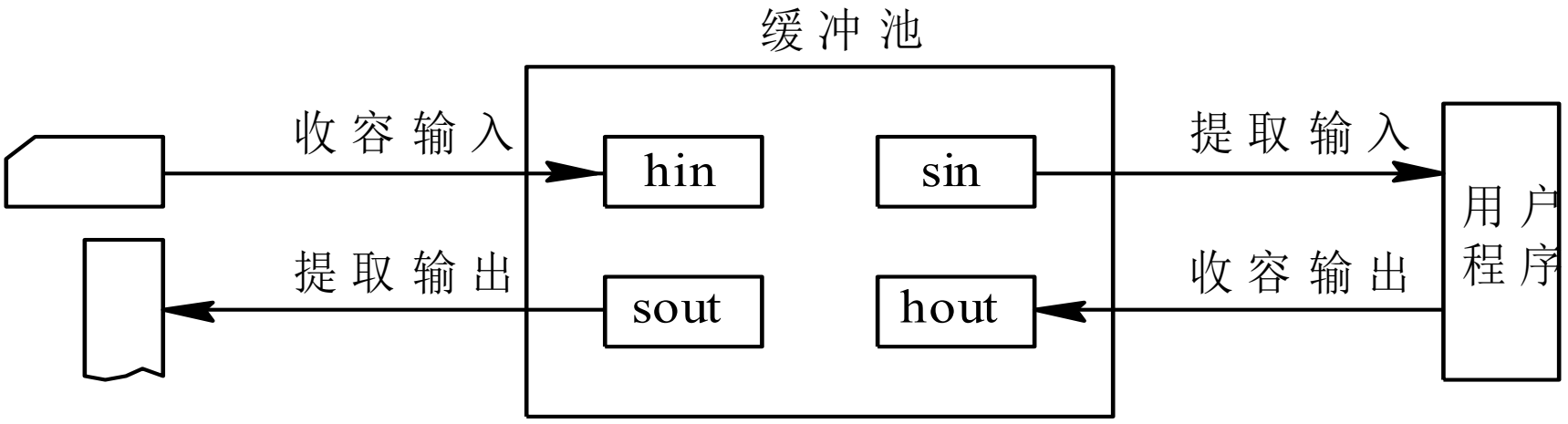
  ④ 用于提取输出数据的工作缓冲区

② Getbuf过程和Putbuf过程

- 因为队列属于临界资源，所以在Getbuf和Putbuf两个操作缓冲池的函数中加入信号量操作

③ 缓冲区的工作方式

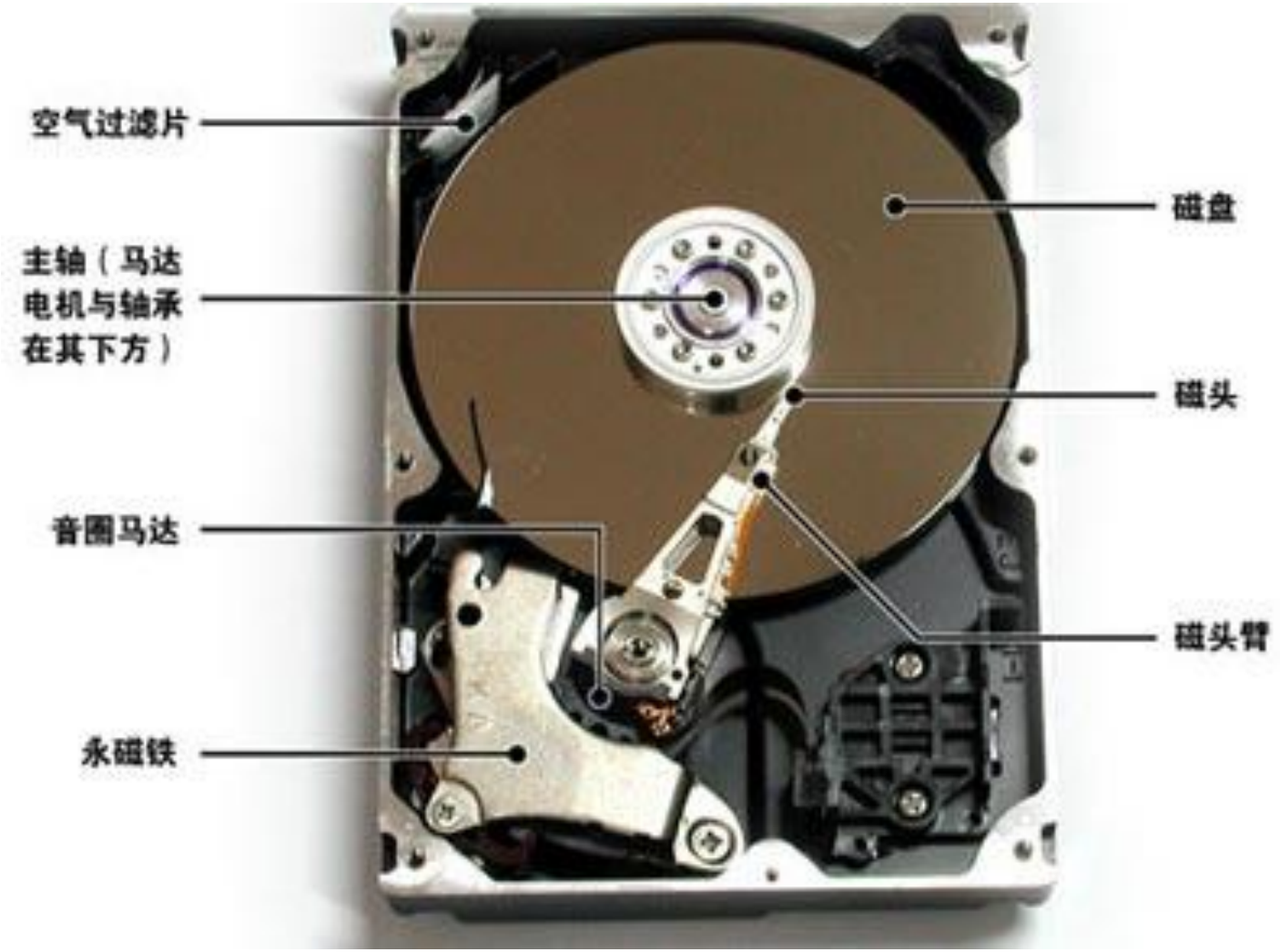
- 缓冲区可以工作在收容输入、提取输入、收容输出和提取输出四种工作方式下



缓冲区的工作方式

6.8 磁盘存储器的管理

- 温彻斯特硬盘内部结构



- **硬盘发展史**

① 1956年，IBM发明第一个磁盘存储系统 IBM 305 RAMAC。

（只有5MB，却有50个24英寸盘片）

② 1973年，IBM研制成功一种新型硬盘 IBM 3340。

（几个涂有磁性材料的同轴金属盘片，和可移动磁头共同密封在一个盒子里）—— 硬盘的祖先“温彻斯特硬盘”，简称“温盘”。

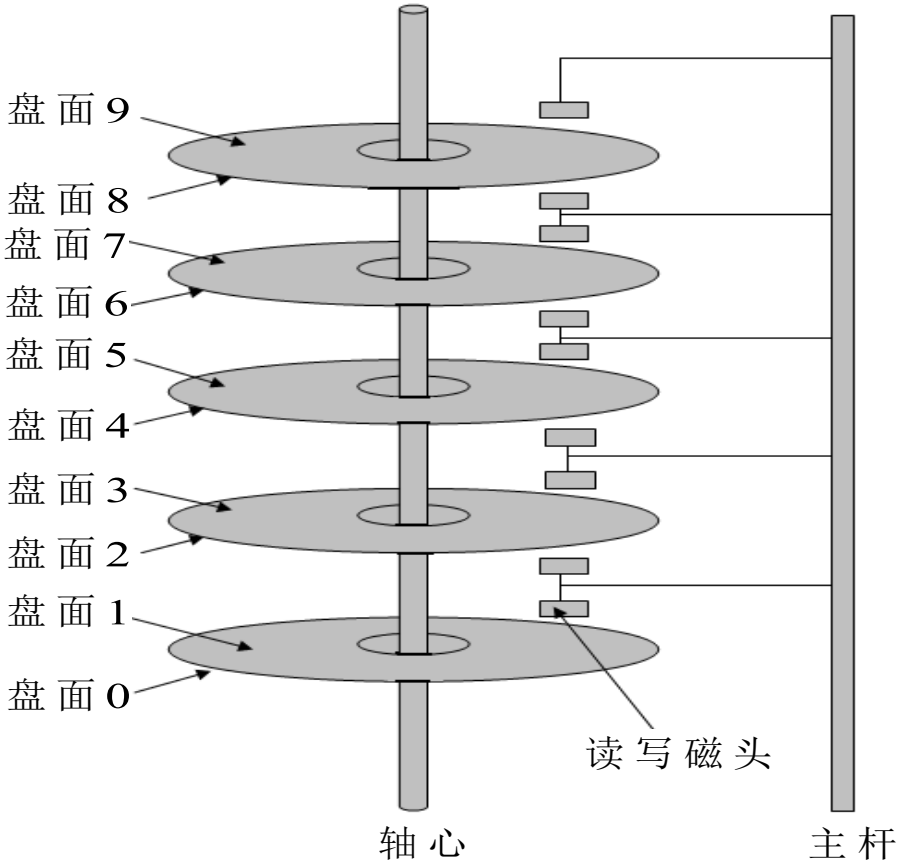
③ 1980年，希捷(Seagate) 制造出个人电脑上第一块温盘。

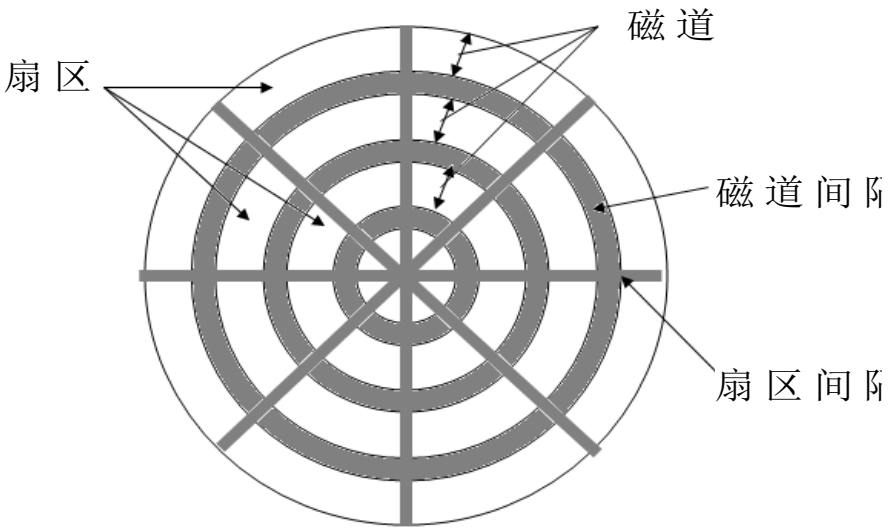
（5MB）

- **目前几乎所有的机械式硬盘都以“温彻斯特”技术为基本原理。**

1. 磁盘性能简述

- 磁盘的组织和格式
  - 磁盘包括一或多个物理盘片。
  - 每个盘片分一个或两个存储面。
  - 每个面被组织成若干同心环，这种环称为磁道(track)，磁道之间留有间隙(gap)。为使处理简单，磁道上可存储的字节数目是相同的。
  - 磁道又被逻辑上划分成若干扇区(sector) 。扇区之间保留一定的间隙。
  - 一个扇区称为一个盘块(或数据块) 。


盘面9
盘面8
盘面7
盘面6
盘面5
盘面4
盘面3
盘面2
盘面1
盘面0
读写磁头
轴心
主杆


磁道
扇区
磁道间隔
扇区间隔

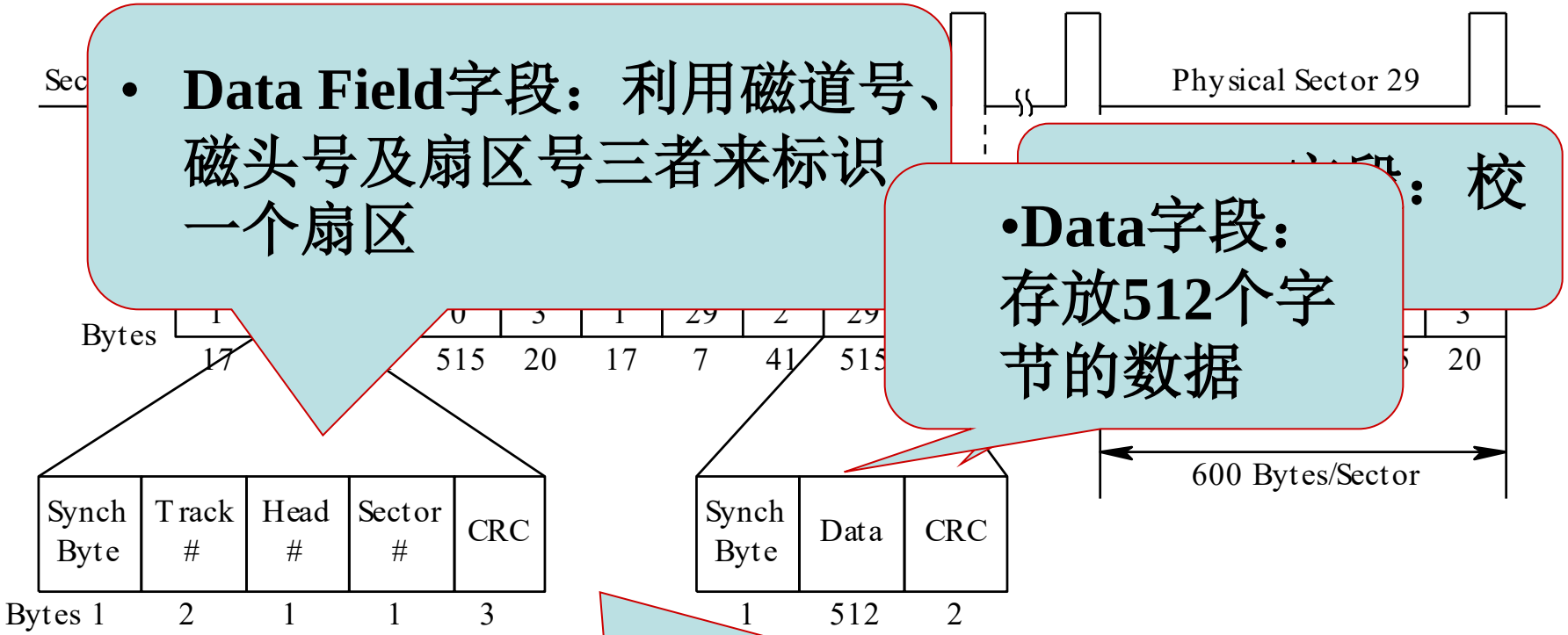
磁盘的结构和布局

- 磁盘上存储的数据块数目是由扇区数、磁道数以及磁盘面数所决定的。
- 例如，一个**10 GB**容量的磁盘
  - 有**8**个双面可存储盘片，共**16**个存储面**(**盘面**)**
  - 每面有**16,383**个磁道，**63**个扇区**(**盘块**)**

- 为充分利用磁盘外面磁道的存储能力，现代磁盘不再把内外磁道划分为相同数目的扇区
- 而是利用外层磁道容量较大的特点，将盘面划分成若干环带，同一环带内的所有磁道具有相同的扇区数。
- 显然，外层环带的磁道拥有更多的扇区。

- 为减少这种磁道和扇区在盘面分布的几何形式变化对驱动程序的影响，大多数现代磁盘都隐藏了这些细节，向OS提供虚拟几何的磁盘规格，而不是实际的物理几何规格。

- 为在磁盘上存储数据，必须先将磁盘低级格式化
- 图中给出一种温切斯特盘中一条磁道格式化的情况



- **Data Field字段：利用磁道号、磁头号及扇区号三者来标识一个扇区**

•Data字段：存放512个字节的数据

•每条磁道含有30个固定大小的扇区，每个扇区容量为600个字节，其中512个字节存放数据，其余的用于存放控制信息

2. 磁盘的类型

- 对磁盘可以从不同的角度进行分类。最常见的有：
  - 硬盘、软盘
  - 单片盘、多片盘
  - 固定头磁盘、活动头(移动头)磁盘

  …

- 下面仅对固定头磁盘和移动头磁盘做些介绍

① 固定头磁盘

- 每条磁道都有一读/写磁头，所有磁头都装在一刚性磁臂上。通过这些磁头可并行访问所有磁道，有效提高了磁盘的I/O速度。
- 这种结构主要用于大容量磁盘

② 移动头磁盘

- 每一个盘面仅配一个磁头，也被装在磁臂上。为能访问该盘面上的所有磁道，该磁头必须能移动。因此，移动磁头仅能以串行方式读/写，致使速度较慢
- 由于其结构简单，仍广泛用于中小型磁盘。
- PC机上配置的温彻斯特硬盘都采用这种结构

3. 磁盘访问时间

①寻道时间T_s

- 指磁臂(磁头)移动到指定磁道上所经历的时间
- 该时间是启动磁臂的时间s与磁头移动n条磁道所花费的时间之和， 即

$$T_s = m \times n + s$$

- 其中，m是一常数，与磁盘驱动器的速度有关。
- 寻道时间随寻道距离的增加而增大

② 旋转延迟时间T_r

- 扇区移动到磁头下面所经历的时间
- 不同的磁盘类型中，旋转速度至少相差一个数量级。如软盘为300 r/min，硬盘一般为7200～15 000 r/min，甚至更高。

③传输时间T_t

- 指读/写磁盘所经历的时间
- T_t的大小与每次所读/写的字节数b和旋转速度有关：

$$T_t = \frac{b}{rN}$$

•其中，r是磁盘每秒钟的转数；N是一条磁道上的字节数， 当一次读/写的字节数相当于半条磁道上的字节数时，T_t与T_r 相同。因此访问时间T_a可表示为：

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

4. 磁盘调度算法

① 先来先服务(FCFS)

- 一种最简单的磁盘调度算法。
- 它根据进程请求访问磁盘的先后次序进行调度。
- 优点：公平、简单，且每个进程的请求都能依次地得到处理，不会出现某一进程的请求长期得不到满足的情况。
- 缺点：由于未对寻道进行优化，致使平均寻道时间可能较长。
- 该算法仅适用于请求磁盘I/O的进程数目较少的场合。

（从 100 号磁道开始）	
被访问的下一个磁道号	移动距离（磁道数）
55	45
58	3
39	19
18	21
90	72
160	70
150	10
38	112
184	146
平均寻道长度：55.3	

平均寻道长度=
移动距离之和 / 寻道次数

FCFS调度算法举例

② 最短寻道时间优先(SSTF, Shortest Seek Time First)

- 该算法首先满足“要求访问的磁道与当前所在磁道之间距离最短”的进程，以使每次的寻道时间最短。
- **SSTF**算法的平均磁头移动距离明显低于**FCFS**，因而**SSTF**较之**FCFS**有更好的寻道性能，故过去曾一度被广泛采用。
- 但这种算法不能保证平均寻道时间最短。

总是优先访问离当前磁头最近的磁道

（从 100 号磁道开始）	
被访问的下一个磁道号	移动距离（磁道数）
90	10
58	32
55	3
39	16
38	1
18	20
150	132
160	10
184	24
平均寻道长度：27.5	

SSTF调度算法举例

- 进程“饥饿”现象
  - SSTF算法虽然能获得较好的寻道性能，但可能导致某个进程发生“饥饿” 现象。
  - 因为只要不断有新进程的请求到达，且其所要访问的磁道与当前磁道之间的距离较近，新进程的I/O请求必然优先满足。
  - 致使老进程的请求长期无法得到满足

③ 扫描(SCAN)算法

- 它是对SSTF算法的改进，以防止老进程出现“饥饿”现象。
- 该算法不仅考虑到要访问的磁道与当前磁道之间的距离，更优先考虑的是磁头当前的移动方向。
- 例如，当磁头正在自里向外移动时，SCAN算法所考虑的下一个访问对象，应是要访问的磁道既在当前磁道之外，又是距离最近的。
- 这样自里向外地访问，直至再无更外的磁道需要访问时，才将磁头换向为自外向里移动。
- 由于该算法的磁头移动规律颇似电梯的运行，因而又常称之为“电梯调度算法”。

不仅考虑磁道的距离，更考虑磁头移动方向。

(电梯调度算法)

（从100#磁道开始，向磁道号增加方向访问）	
被访问的下一个磁道号	移动距离（磁道数）
150	50
160	10
184	24
90	94
58	32
55	3
39	16
38	1
18	20
平均寻道长度：27.8	

SCAN调度算法举例

- SCAN算法既能获得较好的寻道性能，又能防止“饥饿”现象，故被广泛用于大、中、小型机器和网络中的磁盘调度。
- 但SCAN也存在这样的问题：
  - 假设当磁头刚从里向外移动而越过了某一磁道时，恰好又有一进程请求访问此磁道。
  - 这时，磁头继续从里向外，而该进程必须等待，等磁头再次从外向里扫描时，该进程的请求才可能得到处理，致使该进程的请求被大大推迟。

④ 循环扫描(CSCAN)算法

- 为减少SCAN算法造成的这种延迟，CSCAN算法规定磁头只做单向移动。
- 例如，总是自里向外移动，当磁头移到最外的磁道并访问完后，磁头立即返回到最里的要访问磁道，亦即将最小磁道号紧接着最大磁道号构成循环，进行循环扫描。

磁头单向移动。
总是从里向外访问，
或总是从外向里访问

（从100#磁道开始，向磁道号增加方向访问）	
被访问的下一个磁道号	移动距离（磁道数）
150	50
160	10
184	24
18	166
38	20
39	1
55	16
58	3
90	32
平均寻道长度：35.8	

CSCAN调度算法举例

本章小结

- 了解OS处理用户 I/O请求的基本过程。
- 理解通道、缓冲、设备独立性的概念
- 理解I/O 控制方式及设备驱动程序、设备分配的数据结构及分配程序
- 理解设备分配技术、设备管理程序功能
- 掌握缓冲技术、SPOOLing系统
- 掌握磁盘调度算法

练习题

1. 缓冲技术中的缓冲池位于__A__中

A. 内存　B. 外存　C. ROM　D. 寄存器

2. 缓冲引入的目的之一是__D__

A. 提高I/O设备的执行速率

B. 提高CPU的执行速率

C. 节省内存

D. 改善CPU和I/O设备速度不匹配的情况

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

A. 循环缓冲　B. 单缓冲　C. 双缓冲　D. 缓冲池

4. 通过硬件和软件的功能扩充，把原来独立的设备改造成能为若干用户共享的设备，这种设备称为__D__

A. 存储设备　　B. 系统设备

C. 用户设备　　D. 虚拟设备

5. C 算法是设备分配常用的一种算法

A. 短作业优先　　B. 首次适应

C. 先来先服务　　D. 最佳适应

6. SPOOLing技术提高了 A 的利用率

A. 独占设备　B. 共享设备　C. 文件　D. 内存

7. 按 D 分类，将设备分为块设备和字符设备

A. 共享特性　B. 操作特性

C. 从属关系　D. 信息交换单位

8. 下列算法中，用于磁盘调度算法的是

C

A. 时间片轮转法　　B. LRU

C. SSTF　　D. 优先级高者优先

1. 缓冲区是OS为提高CPU和I/O设备的并行性，在磁盘上开辟的存储区 （ X ）

   在内存里开辟的区域

2. 在采用DMA方式进行I/O操作时，只在传送一个数据块的开始和结束时，才需要CPU干预 （ V ）

3. 在磁盘访问时间中，寻道时间所占比例是最少的 （ X ）

   寻道花费的时间最长

4. 缓冲技术是借助外存的一部分作为缓冲区
（ × ）

借助内存

5. 按照所属关系进行分类，设备可分为独占设备、共享设备和虚拟设备 （ × ）

按照共享属性

6. 键盘、鼠标、打印机等以字符为单位组织和处理信息的设备，称为字符设备；磁盘等以块为单位组织和处理信息的设备，称为块设备
（ √ ）

7. 通道是一种传输数据的数据通道 (　✗　)

通道是一种I/O专用处理器

1. 在磁盘I/O时，若干进程申请操作磁盘，它们依次请求访问某磁道号，其顺序如下：

160、 39、18、 55、 90、38、 58、184 、150

如果当前磁头停在100号磁道，并准备向磁道号增加的方法(由里向外)扫描。请分别用SSTF、SCAN和CSCAN算法计算平均寻道长度

答案：同书上的算法举例