第13章 设备管理

设备管理

- 在计算机系统中，除了CPU和内存之外，其他的大部分硬件设备称为**外部设备**。
- 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：计算机的主存和外围设备的介质之间的信息传送操作。

设备管理的任务和功能

- 设备管理的主要任务是：
  - 响应I/O请求，为用户提出的I/O请求服务
  - 分配I/O设备，对I/O请求分配相应设备
  - 管理I/O设备，提高可靠性、安全性、利用率
  - 提供I/O设备接口，为用户提供友好访问接口

- 为了完成上述任务，设备管理应具备以下4个功能：

设备管理功能

- 设备分配
  - 根据用户的I/O请求，为之分配设备。包含控制器和通道。
- 设备处理
  - 负责启动设备及I/O操作完成时的中断处理。
- 缓冲管理
  - 为缓和CPU与I/O速度不匹配的矛盾常设置缓冲区。缓冲管理负责缓冲区的分配和释放及有关管理工作。
- 设备独立性
  - 又称设备无关性，是指用户编制程序时所使用的设备与物理设备无关。

设备独立性有两种类型：
1.独立于同类设备的具体设备号
2.独立于设备类型

目录

1.设备分类

按设备的使用特性分类：

- 存储设备：
  - 用来保存各种信息
  - 如磁盘，磁带等。
- 传输设备：
  - 用于传输数据
  - 如网卡、Modem
- 人机交互设备：
  - 向CPU传输信息或输出经过CPU加工处理信息的设备。
  - 如键盘、显示器和打印机等。

按设备的共享属性分类

- 独占设备
  - 在一段时间内只允许一个用户进程使用的设备。
- 共享设备
  - 在一段时间内允许多个进程使用的设备。
- 虚拟设备
  - 指通过虚拟技术将一台独占设备改造成若干台逻辑设备，供若干个用户进程同时使用。通常把这种经过虚拟技术处理后的设备称为虚拟设备。

按信息交换单位分类

- 块设备
  - 处理信息的基本单位是字符块。
  - 一般块的大小为512B～4KB，如磁盘、磁带等是块设备。
  - 一般存储型外围设备为块设备

- 字符设备
  - 处理信息的基本单位是字符。
  - 如键盘、打印机和显示器是字符设备。
  - 一般I/O型设备为字符设备，与主存交换的单位为字节。

2.一点硬件常识

- I/O设备与计算机的交互是通过信号来实现的

- Port 端口
  - 设备连接点

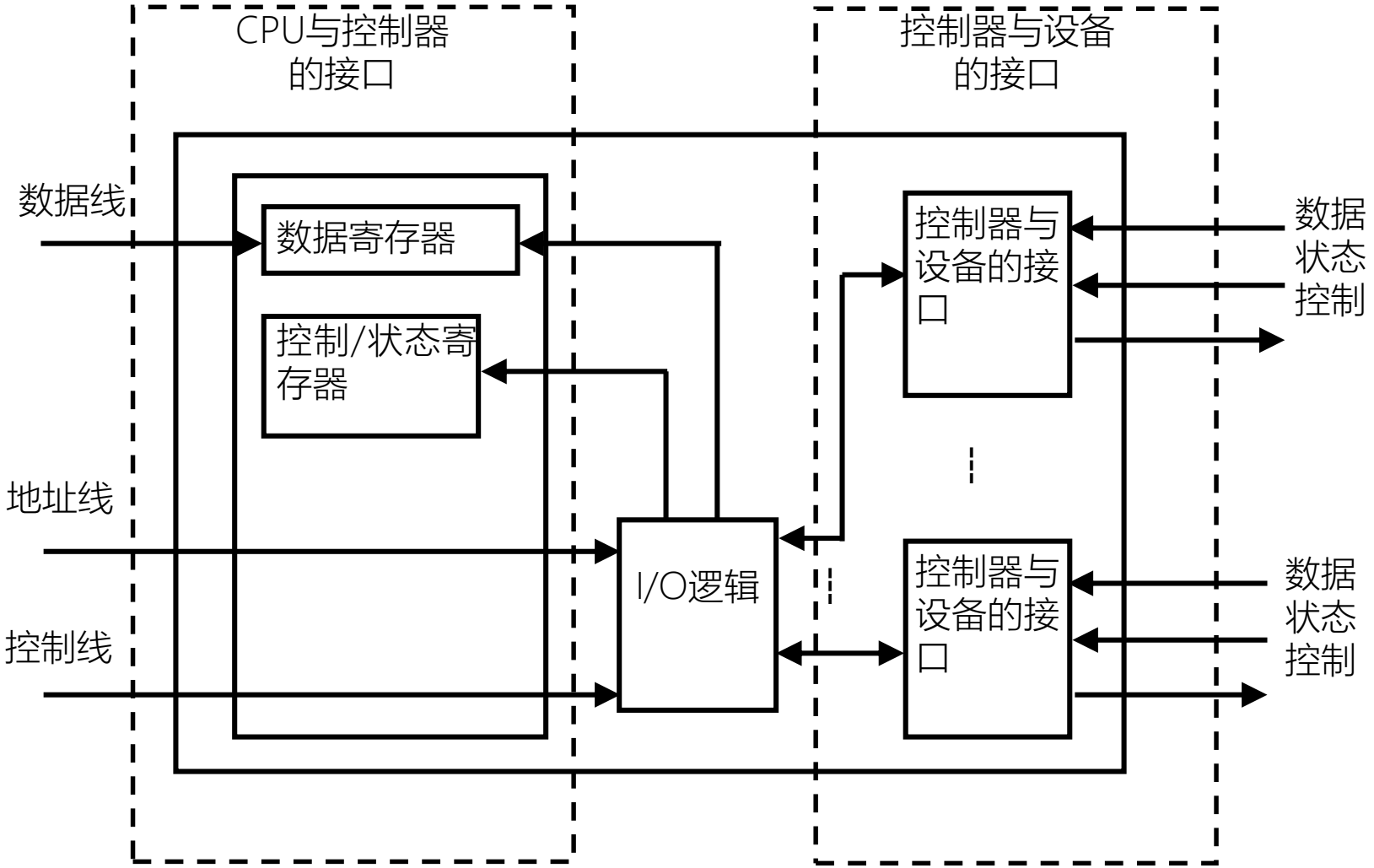
- Bus 总线
  - 一个或者多个设备可以使用一组共同的线
  - 一组**线**和一组严格定义的可以描述在线上传输信息的**协议**
  - 典型的总线：PCI bus(Peripheral Component Interconnect), PCI Express (PCIe)
  - 扩展总线：依附于总线之下，连接相对较慢的设备，如SCSI, Small Computer System Interface

设备控制器

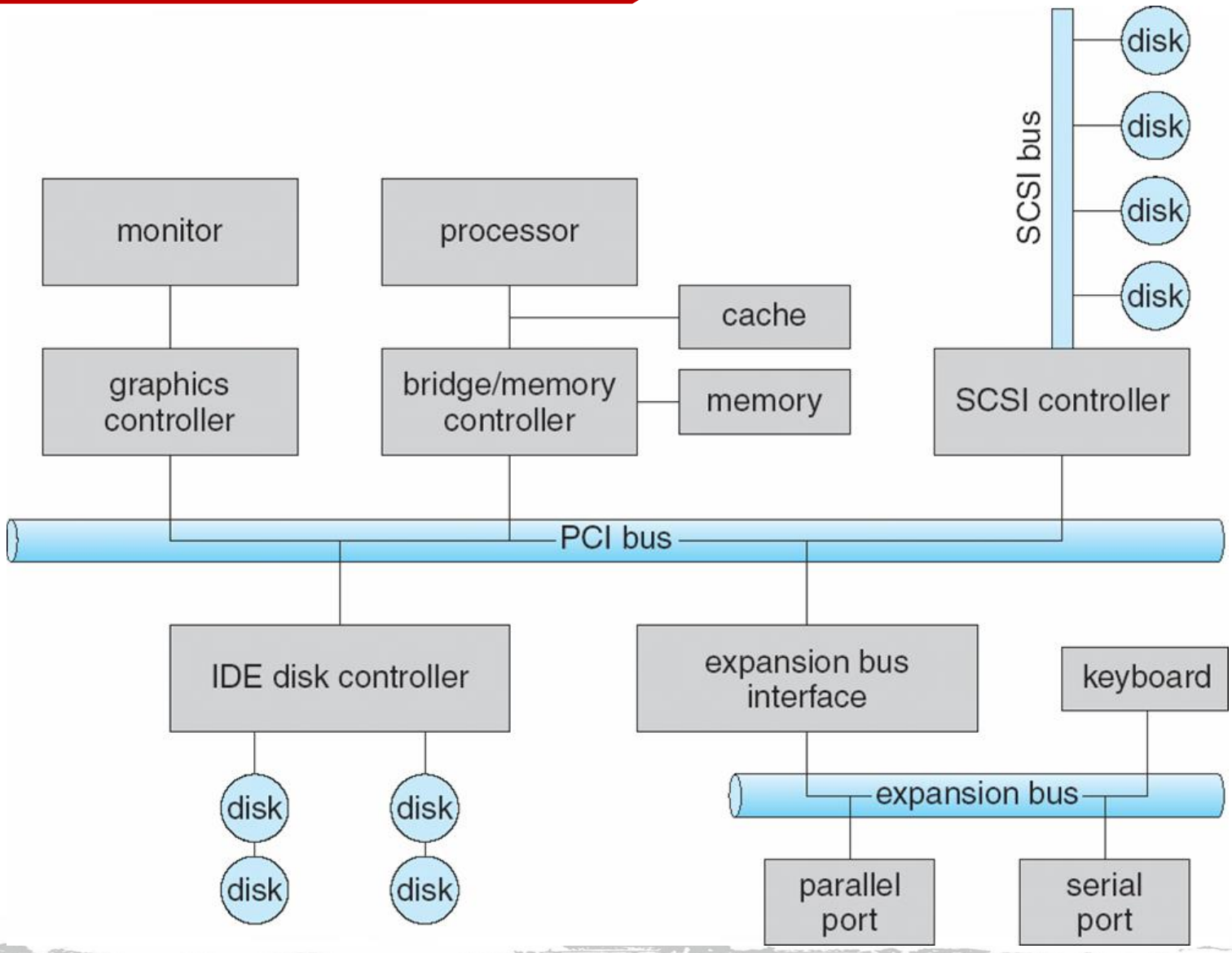
- 设备控制器，Controller (host adapter)
  - 操作管理端口、总线、设备的电子器件

- 设备控制器由三部分组成：
  ① 设备控制器与CPU的接口：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：实现设备与设备控制器之间的通信。
  ③ I/O逻辑：实现对设备的控制，它负责接收命令、对命令进行译码、再根据译出的命令控制设备。

设备控制器组成图


CPU与控制器
的接口
控制器与设备
的接口
数据线
数据寄存器
控制/状态寄
存器
地址线
控制线
I/O逻辑
控制器与
设备的接
口
控制器与
设备的接
口
数据
状态
控制
数据
状态
控制

一个典型的PC总线结构



3.处理器与控制器的交互方式

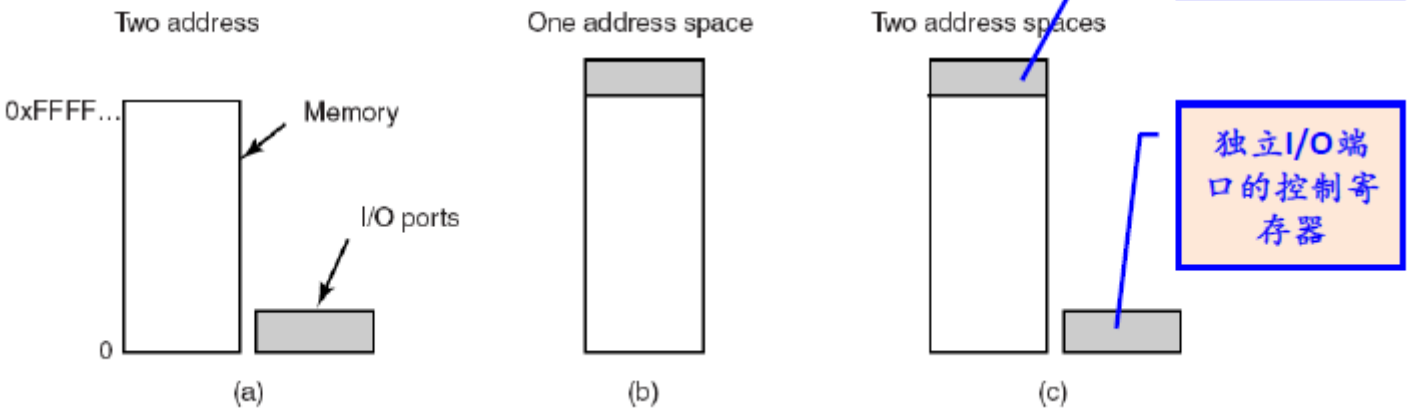
- I/O 指令：处理器与控制器的控制桥梁
- 设备所包含的寄存器：
  - 数据输入寄存器：被主机读取，实现获取数据
  - 数据输出寄存器：被主机写入，实现发送数据
  - 状态寄存器：包含主机可读取的位，标识状态
  - 控制寄存器：可被主机向设备发送命令或改变设备状态
- 通信方式
  - Direct I/O方式：使用I/O指令向指定端口传输数据
  - Memory-mapped I/O
    - 设备数据和命令寄存器被映射到处理器内存空间
    - 处理器通过标准内存访问实现对设备控制器读写
    - 如较大内存空间：显存

I/O端口地址

- I/O端口地址：接口电路中每个寄存器具有的、唯一的地址，是个整数
- 所有I/O端口地址形成I/O端口空间(受到保护)

I/O指令形式与I/O地址是相互关联的， 主要有两种形式：

- 内存映像编址（内存映像I/O模式）
- I/O独立编址（I/O专用指令）



I/O独立编址

- 分配给系统中所有端口的地址空间完全独立，与内存地址空间无关
  - 使用专门的**I/O**指令对端口进行操作
- 优点
  - 外设不占用内存的地址空间
  - 编程时，易于区分是对内存操作还是对**I/O**端口操作
- 缺点：
  - **I/O**端口操作的指令类型少，操作不灵活
- 例子：**8086/8088**，分配给**I/O**端口的地址空间**64K**，**0000H~0FFFFH**，只能用**in**和**out**指令进行读写操作

PC上的一些I/O端口分布

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

内存映像编址

- 分配给系统中所有端口的地址空间与内存的地址空间统一编址
  - 把I/O端口看作一个存储单元，对I/O的读写操作等同于对内存的操作
- 优点
  - 凡是可对内存操作的指令都可对I/O端口操作
  - 不需要专门的I/O指令
  - I/O端口可占有较大的地址空间
- 缺点
  - 占用内存空间

目录

I/O控制方式

- I/O控制方式发展过程中始终贯穿的宗旨是尽量减少主机对I/O控制的干预。
- 常用的输入/输出控制方式有下述几种：
  - 轮询方式
  - 中断方式
  - DMA方式
  - 通道方式

1 轮询方式 Polling

- 早期计算机系统中无中断机构，设备控制采用程序直接控制方式，也称为程序直接控制方式
- 由CPU代表进程给I/O模块发I/O命令，进程进入忙等待，直到操作完成才继续执行
- 几个状态/控制寄存器的状态位：
  - 状态寄存器
    - busy bit：1表示正在工作中，0表示空闲
    - error bit：1表示故障，0表示设备i/o成功，无故障
    - write bit：数据寄存器被写
    - command-ready bit：有命令需要控制器执行，命令就绪

主机与控制器的握手过程

① 主机不断地读取状态寄存器，直到busy bit位被清零
② 主机设置write bit为1，并向数据输出寄存器写一个字节
③ 主机设置command-ready bit为1，即命令就绪状态
④ 当设备控制器发现command-ready bit为1时，设置busy bit为1
⑤ 控制器读取控制寄存器，当看到write bit为1，就从数据输出寄存器中读取一个字节，并向设备执行i/o操作
⑥ 当传输完成，控制器清除command-ready，error bit，busy bit

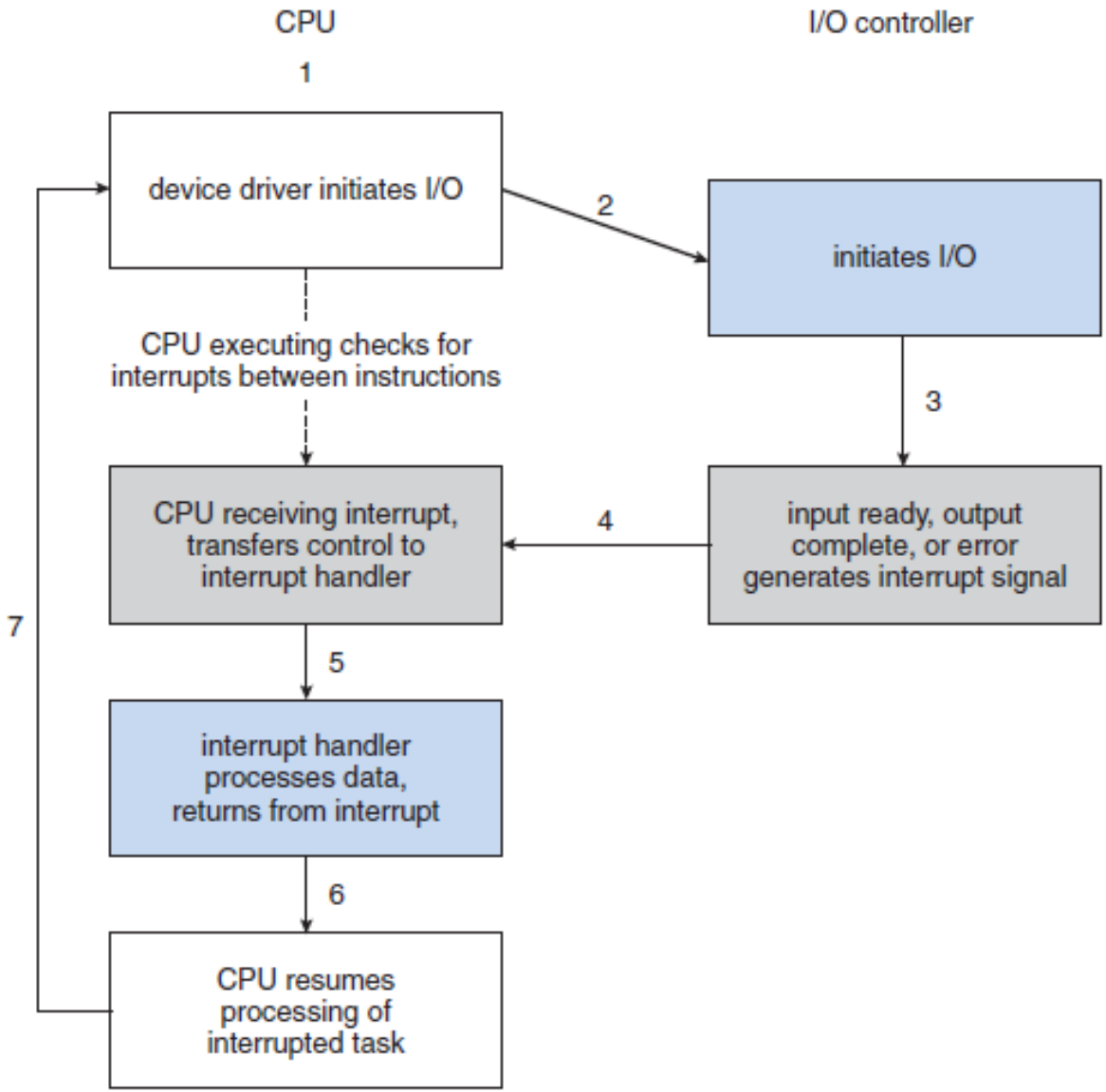
当第一步检查时采取的是 busy-wait 模式

- 如果控制器和设备够快，方法可行
- 如果设备比较慢，则效率低下

2 中断控制方式

- 现代计算机系统中对设备的控制广泛采用了中断控制方式。
- 为了减少设备驱动程序不断地询问控制器状态寄存器的开销 。**I/O**操作结束后，由设备控制器主动通知设备驱动程序。
- CPU 硬件具有一条中断请求线IRL， Interrupt-request Line
- CPU在每执行完一条指令后，都会花费极短时间检测IRL
- 当CPU检测到IRL后，会跳转中断处理程序

中断方式工作过程

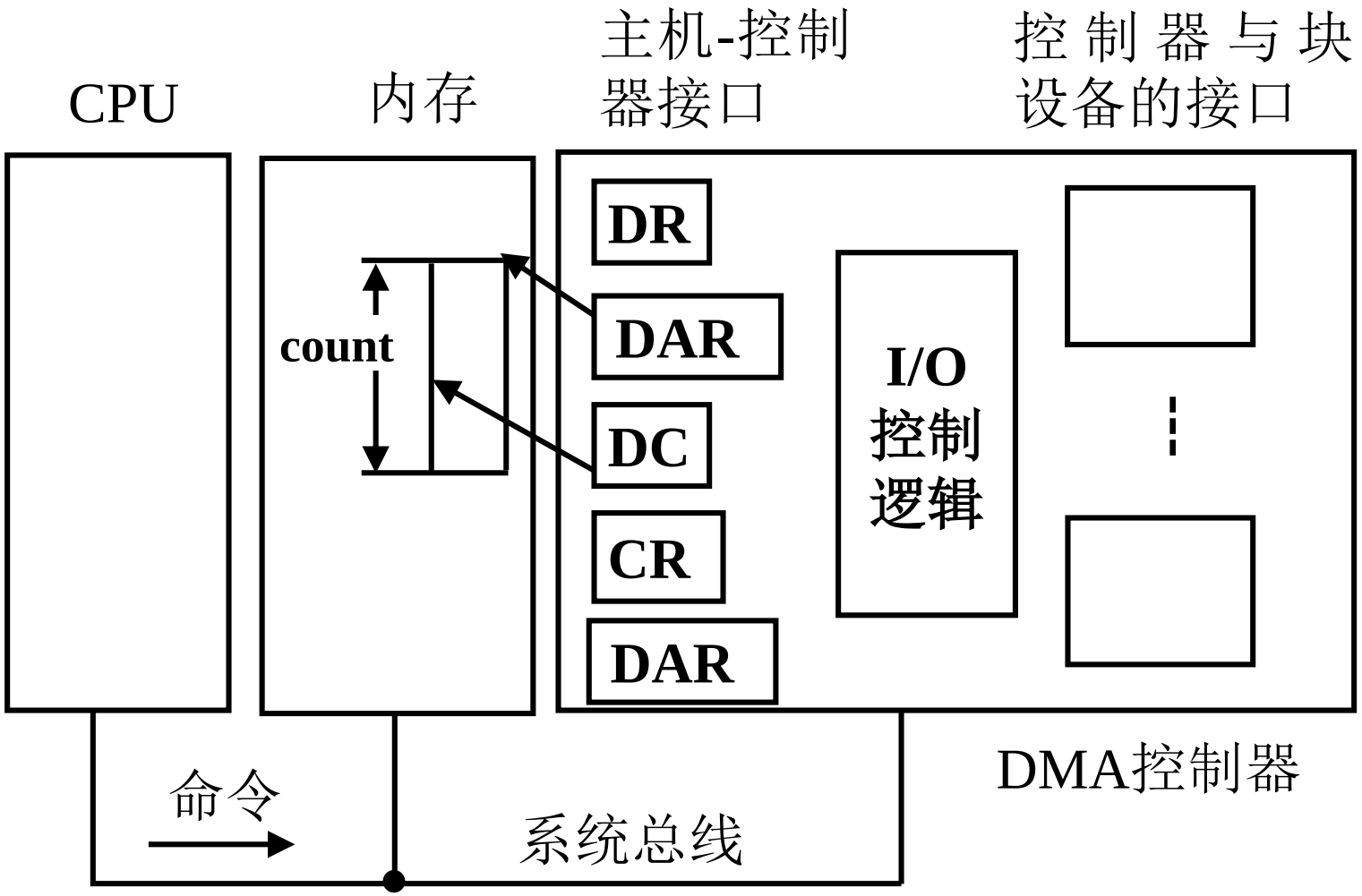

CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
7
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

3 DMA控制方式

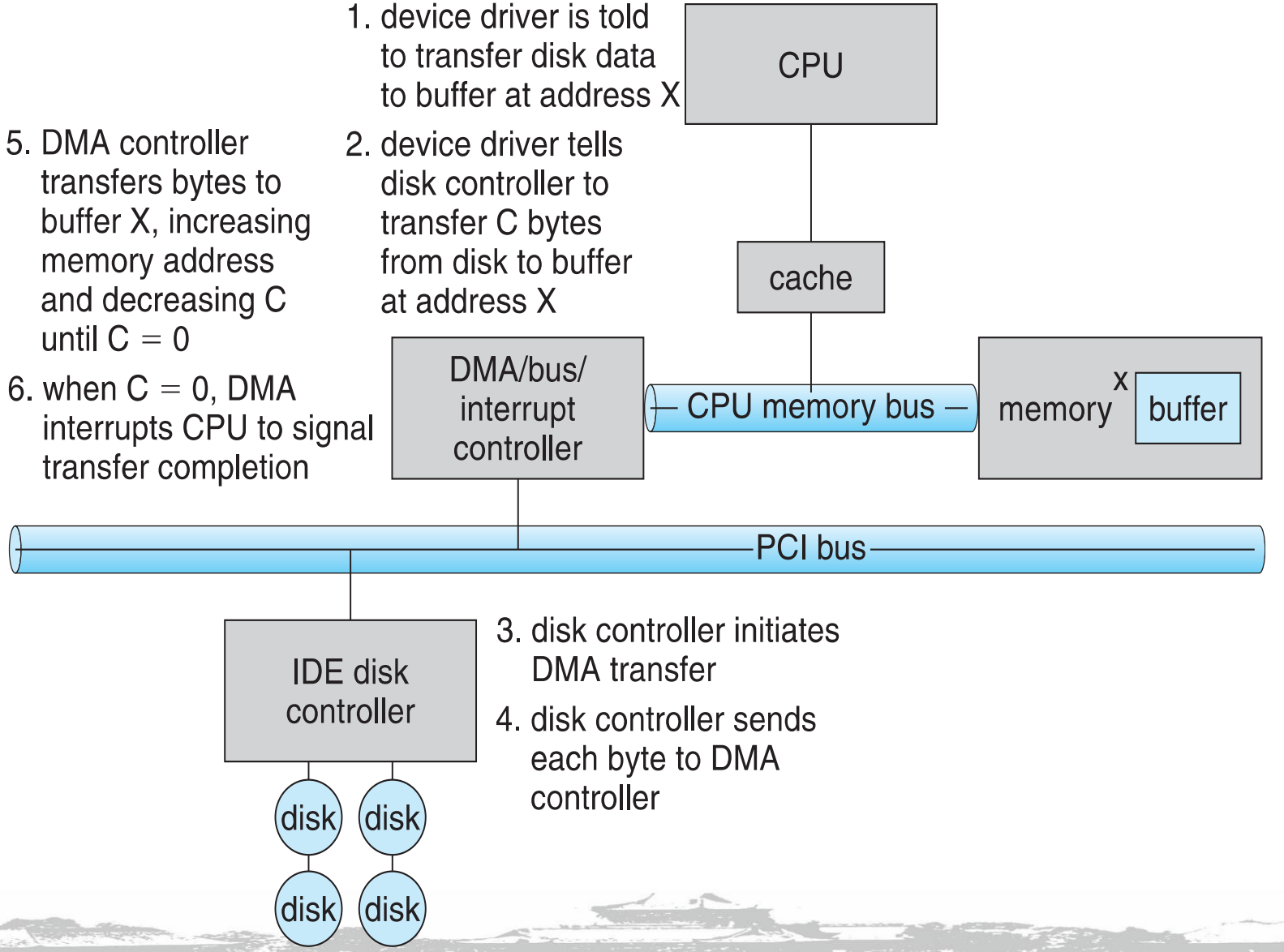
- 如果I/O设备能直接与主存交换数据而不占用CPU，CPU的利用率还可提高，这就出现了直接存储器存取DMA方式。

- DMA控制方式的思想
  - 在外设与内存之间开辟直接的数据交换通路，在控制器的控制下，设备和内存之间可以成批地进行数据交换。

DMA控制器的组成


CPU
内存
主机-控制器接口
控制器与块设备的接口
count
DR
DAR
DC
CR
DAR
I/O控制逻辑
DMA控制器
命令
系统总线

DMA方式工作过程


1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion
CPU
cache
DMA/bus/ interrupt controller
CPU memory bus
memory
x
buffer
PCI bus
IDE disk controller
disk
disk
disk
disk

DMA方式与中断方式的主要区别

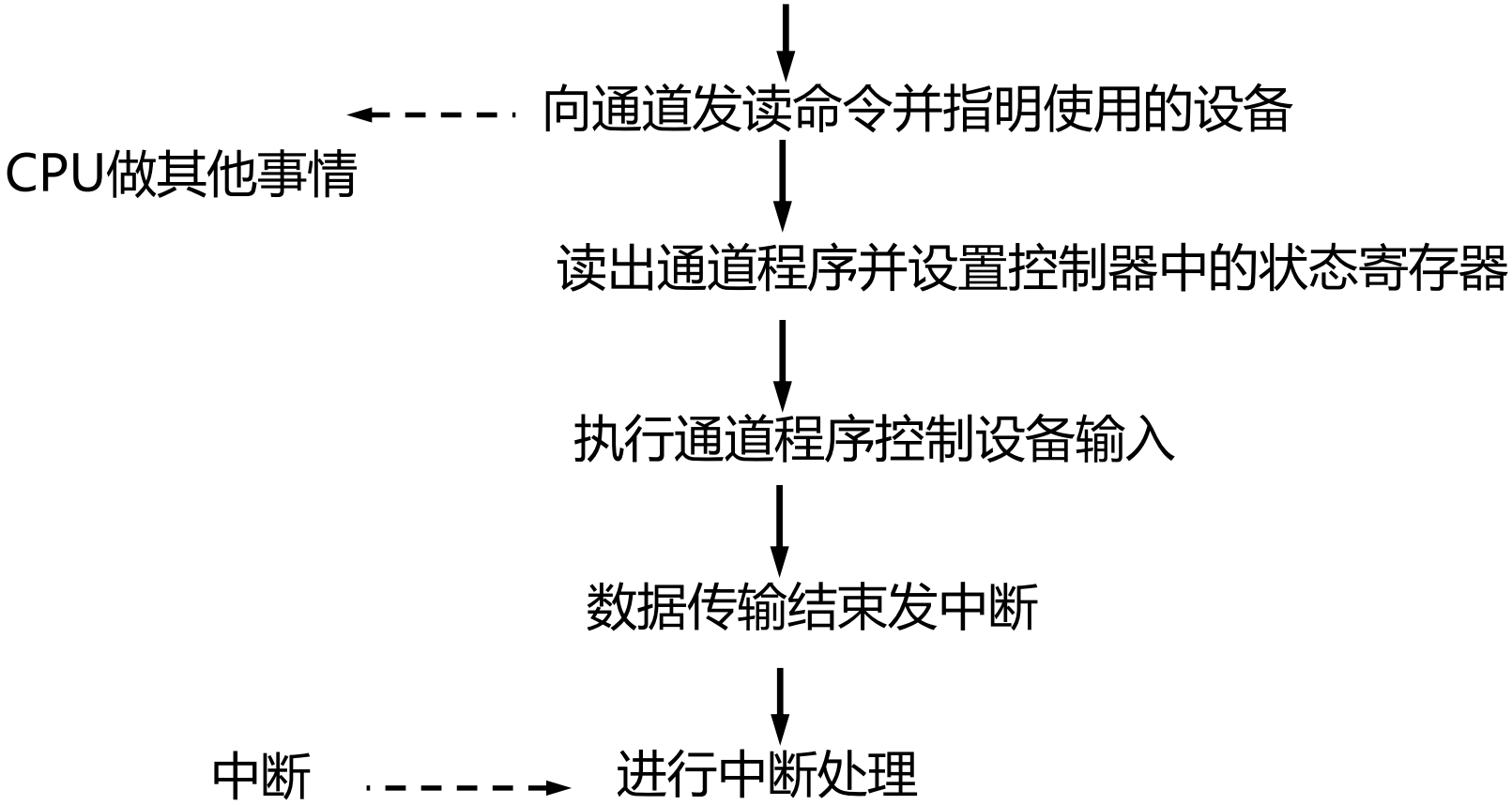
- 中断CPU的时机
  - 中断控制方式在每个数据传送完成后中断CPU
  - DMA控制方式是在所要求传送的一批数据全部传送结束时中断CPU

- 控制方
  - 中断控制方式的数据传送是在中断处理时由CPU控制完成
  - DMA控制方式是在DMA控制器的控制下完成

4 通道控制方式

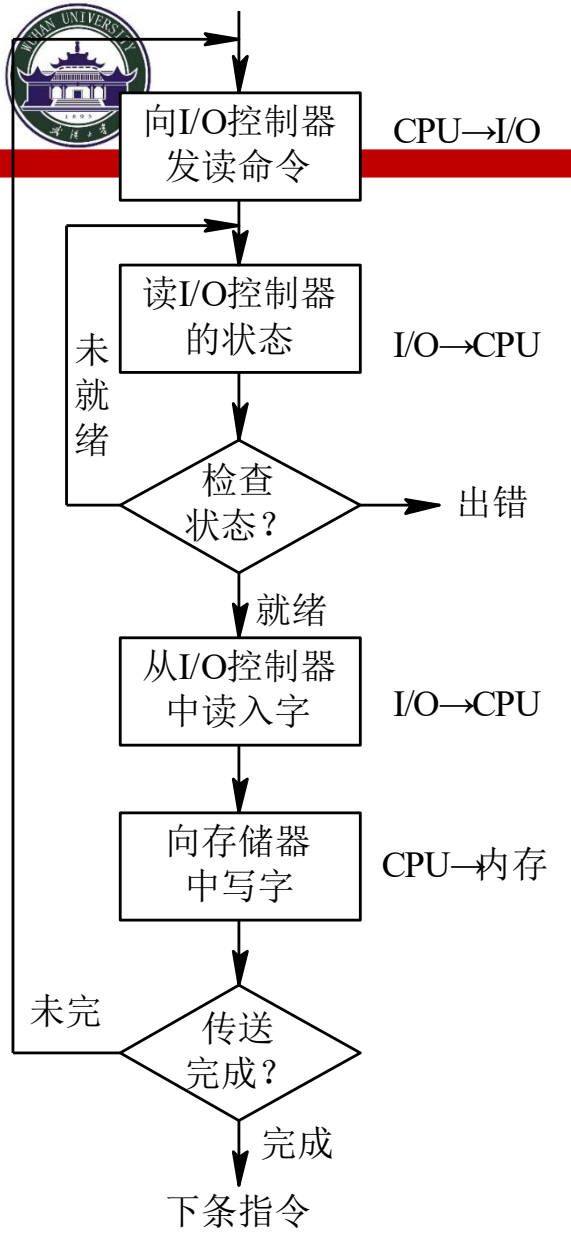
- 通道控制方式是DMA控制方式的发展，它所需的CPU干预更少。
- 通道控制方式与DMA方式类似，也是一种以内存为中心，实现设备与内存直接交换数据的控制方式。
- 在通道控制方式中，CPU只需发出启动指令，指出要求通道执行的操作和使用的I/O设备，该指令就可以启动通道并使该通道从内存中调出相应的通道程序执行。

通道方式工作过程

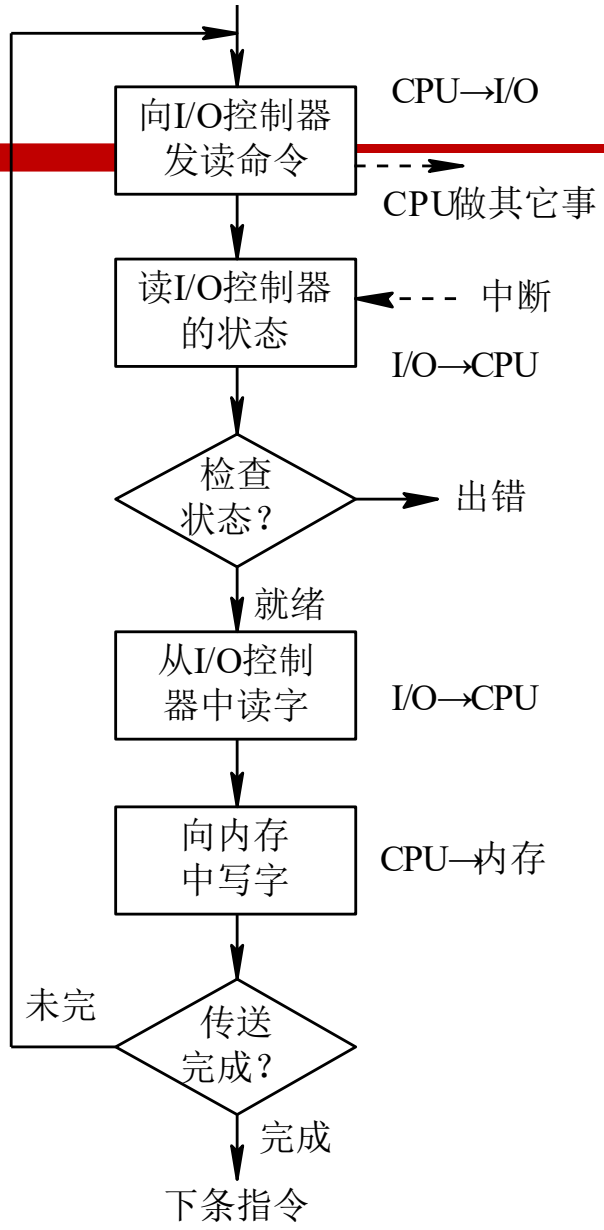

向通道发读命令并指明使用的设备
CPU做其他事情
读出通道程序并设置控制器中的状态寄存器
执行通道程序控制设备输入
数据传输结束发中断
中断
进行中断处理

I/O通道

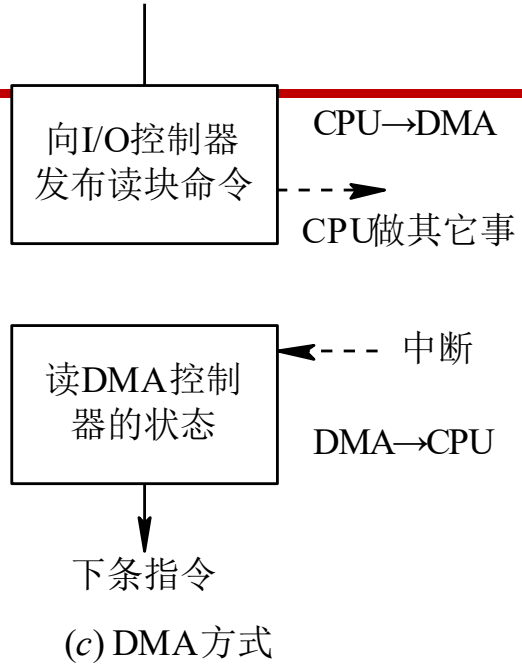
- I/O通道是专门负责输入/输出工作的处理机。它具有执行I/O指令的能力，并通过执行通道程序来控制I/O操作。
- 通道有自己的指令系统，该指令系统比较简单，一般只有数据传送指令、设备控制指令等。
- 通道程序：通道指令构成的程序。
- 通道与一般处理机的区别：
  - 指令类型单一
  - 没有内存



(*a*) 程序I/O方式



(*b*) 中断驱动方式



(*c*) DMA方式

第13章 设备管理

13.3 I/O软件原理

目录

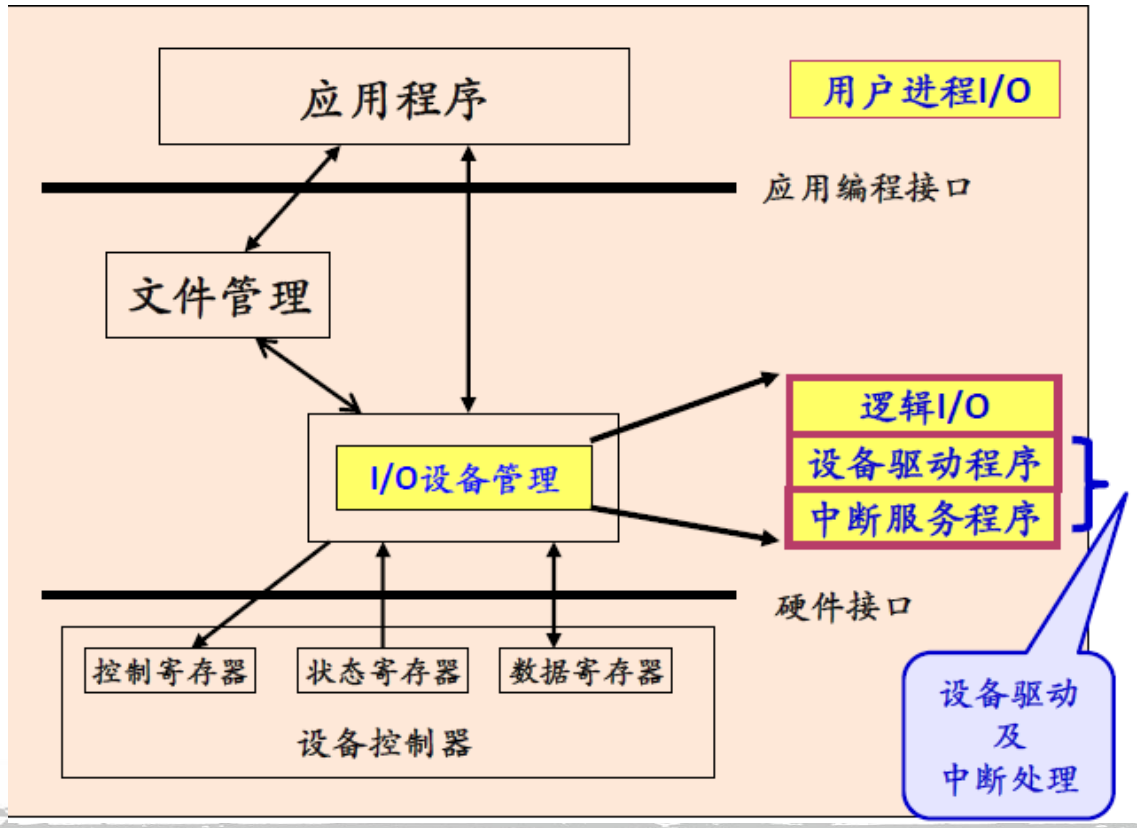
- I/O软件设计的基本思想
  - 将设备管理软件组织成一种层次结构
  - 低层软件与硬件相关
  - 高层软件则为用户提供一个友好的、清晰而统一的接口。

I/O软件的设计目标和原则

- 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
  - 出错处理
  - 同步/异步传输
  - 独占性外围设备和共享性外围设备

I/O软件的层次

- I/O中断处理程序
- 设备驱动程序
- 与设备无关的操作系统I/O软件
- 用户层I/O软件



1. 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
  - 操作正常结束处理
  - 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

2. 设备驱动程序

- 所有与设备相关的代码放在设备驱动程序中。
- 设备驱动程序与设备密切相关，故应为每一类设备配置一个驱动程序。
- 设备驱动程序是I/O进程与设备控制器间的通信程序。

设备驱动程序的任务

- 设备驱动程序的主要任务是：
  - 接收来自上层的与设备无关软件的抽象I/O请求
    - 如果请求得不到满足，则将请求挂在等待队列中
  - 把它转换成设备控制器可以接收的具体命令
    - 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
      - 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
  - 将这些命令发送给设备控制器
  - 监督命令的执行

驱动程序发出命令后的处理方式

- 设备驱动程序发出命令后，有两种处理方式：
  - 驱动程序阻塞，直到中断信号将其唤醒。
  - 设备驱动程序不阻塞，用于操作时间很短的情况。

例：磁盘驱动程序功能

- 计算请求块在磁盘的位置
  - 如将磁盘逻辑块号转换成柱面号、磁头号、扇区号。
- 检查驱动器的电机是否正常运转，磁头是否定位在正确位置。
- 向控制器发送I/O命令
  - 若设备空闲则启动设备完成I/O操作，否则将请求块挂在设备等待队列上。
- 响应由控制器或通道发来的中断请求，并调用相应的中断处理程序。
  - 若正常则将数据传送到与设备无关的软件层，否则报告错误。

3. 与设备无关的操作系统I/O软件

- 也称为内核I/O子系统
- I/O软件的一部分（如设备驱动程序）与设备相关，其他部分则与设备无关。
- 设备驱动程序与设备无关软件之间的界限随操作系统的不同而异
- 基本任务
  - 实现一般设备都需要的I/O功能，并向用户层软件提供一个统一的接口。

与设备无关软件的功能

- 与设备无关软件通常应实现的功能包括：
  - 设备命名
  - 设备保护
  - 提供与设备无关的逻辑块
  - 缓冲管理
  - 存储设备的空间管理
  - 独占设备的分配与释放
  - 错误处理

设备命名

- 与设备无关软件负责把设备的符号名映射到相应的设备驱动程序上。
  - 例如，在UNIX系统中，像/dev/tty00这样的设备名唯一确定了一个特殊文件的i节点，这个i节点包含了主设备号和次设备号。主设备号用于寻找对应的设备驱动程序，而次设备号提供了设备驱动程序的有关参数，用来确定要读写的具体设备。

设备保护

- 对设备进行必要的保护，防止无授权的应用或用户的非法使用
  - 例如，在大多数大型计算机系统中，用户进程对I/O设备的直接访问是完全禁止的；
  - 在UNIX系统中，使用存取权限来保护系统中的I/O设备。

提供与设备无关的逻辑块

- 不同磁盘可采用不同的扇区尺寸，与设备无关软件应向高层软件提供大小统一的逻辑块。
- 例如，可以将若干扇区合并成一个逻辑块。这样较高层软件只与抽象设备打交道，不考虑物理扇区的尺寸而使用等长的逻辑块。

缓冲

- 对于常见的块设备和字符设备，一般都使用缓冲区。
  - 对块设备，硬件一般一次读写一个完整块，而用户进程是按任意单位读写数据的。如果用户进程只写了半块数据，则操作系统通常将数据保存在内部缓冲区，等到用户进程写完整块数据才将缓冲区的数据写到磁盘上。
  - 对字符设备，当用户进程输出数据的速度快于设备输出数据的速度时，也必须使用缓冲。

存储设备的空间分配

- 在创建新文件并向其中写数据时，通常要为该文件分配新的存储块。
- 为完成空间分配工作，操作系统需为每个磁盘设置一张空闲磁盘块表或位示图，因查找一个空闲块的算法与设备无关，可以将该算法放在与设备无关的软件层中处理。

独占设备的分配和释放

- 独占设备在任何时刻只能被单个进程使用，因此操作系统应对设备的使用请求进行检查，并根据设备状态决定是接收请求还是拒绝请求。
- 一个简单的处理方法是要求进程直接通过OPEN打开设备特殊文件来提出请求。若设备不能用，则OPEN失败。关闭独占设备的同时释放该设备。

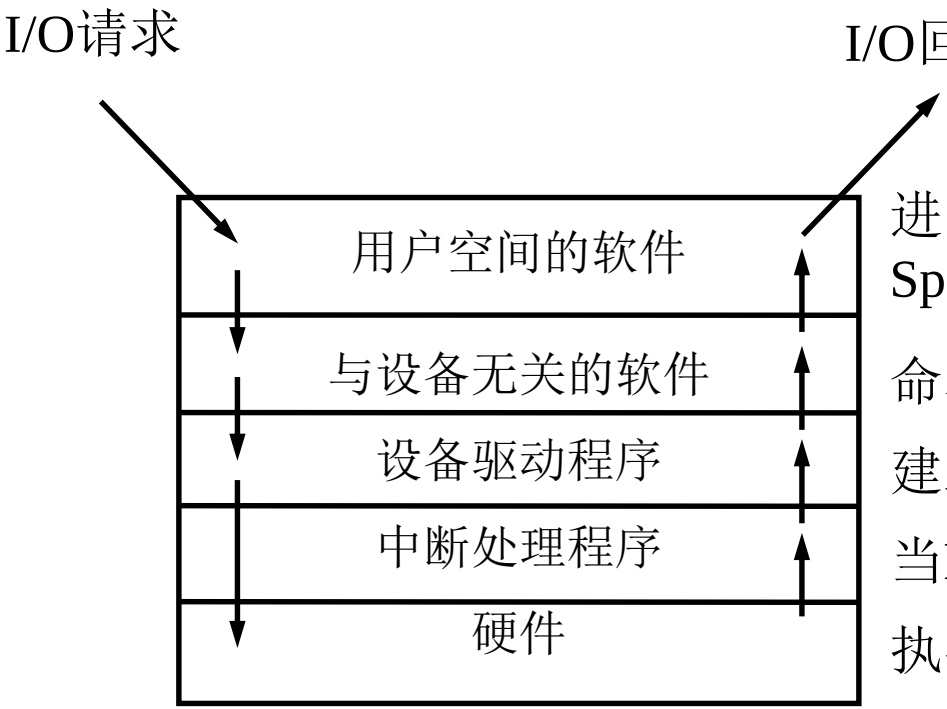
出错处理

- 一般出错处理由设备驱动程序完成。大多数错误与设备相关，因此只有驱动程序知道应如何处理（比如：重试）。
- 但有些错误不是设备造成的，如因磁盘块受损而不能读，驱动程序将尝试重读一定次数，若仍有错误则通知与设备无关软件。
- 若在读用户文件时出现错误，操作系统会将错误信息报告给调用者。若在读一些关键系统数据时出现错误，比如位示图，操作系统则打印错误信息，并向系统管理员报告相应错误。

4. 用户空间的软件

- 一般来说，大部分I/O软件都在操作系统中，但仍有一小部分由与用户程序链接在一起的库函数、甚至运行于内核之外的程序构成。
  - 如函数printf完成输出及输出信息的格式化，标准I/O库中包含许多涉及I/O的过程，它们是作为用户程序的一部分运行的。
- 非库例程实现的 I/O系统调用
  - 如spooling系统

5. I/O系统的层次结构小结



I/O功能

进行I/O调用；格式化I/O；Spooling

命名，保护，阻塞，缓冲，分配

建立设备寄存器；检查状态

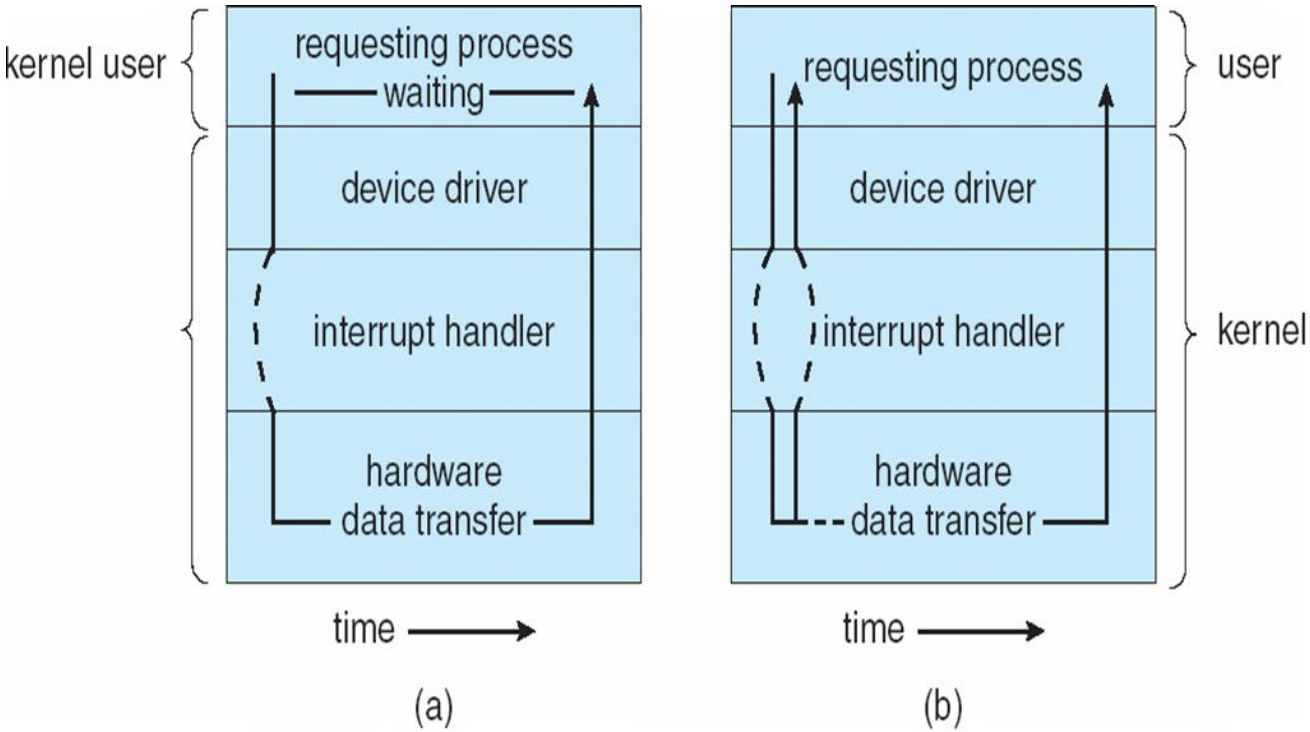
当I/O结束时唤醒驱动程序

执行I/O操作

6. 非阻塞I/O与异步I/O

- 阻塞I/O – 进程挂起直到I/O完成
  - 容易使用与理解
- 非阻塞I/O - I/O调用后立即返回
  - 场景：UI、视频播放时的数据处理
  - 通过多线程编程实现：有的线程阻塞，有的线程继续执行
  - 非阻塞read / write：能够返回等于或者少于请求的字节数
  - Socket中select()：能快速找到有效的i/o接口socket，然后继续read /write实现传输
- 异步I/O – 当I/O执行时，进程继续执行
  - 当I/O完成时，I/O子系统才会通知进程
  - 异步read/write调用：立即返回

两类I/O方法



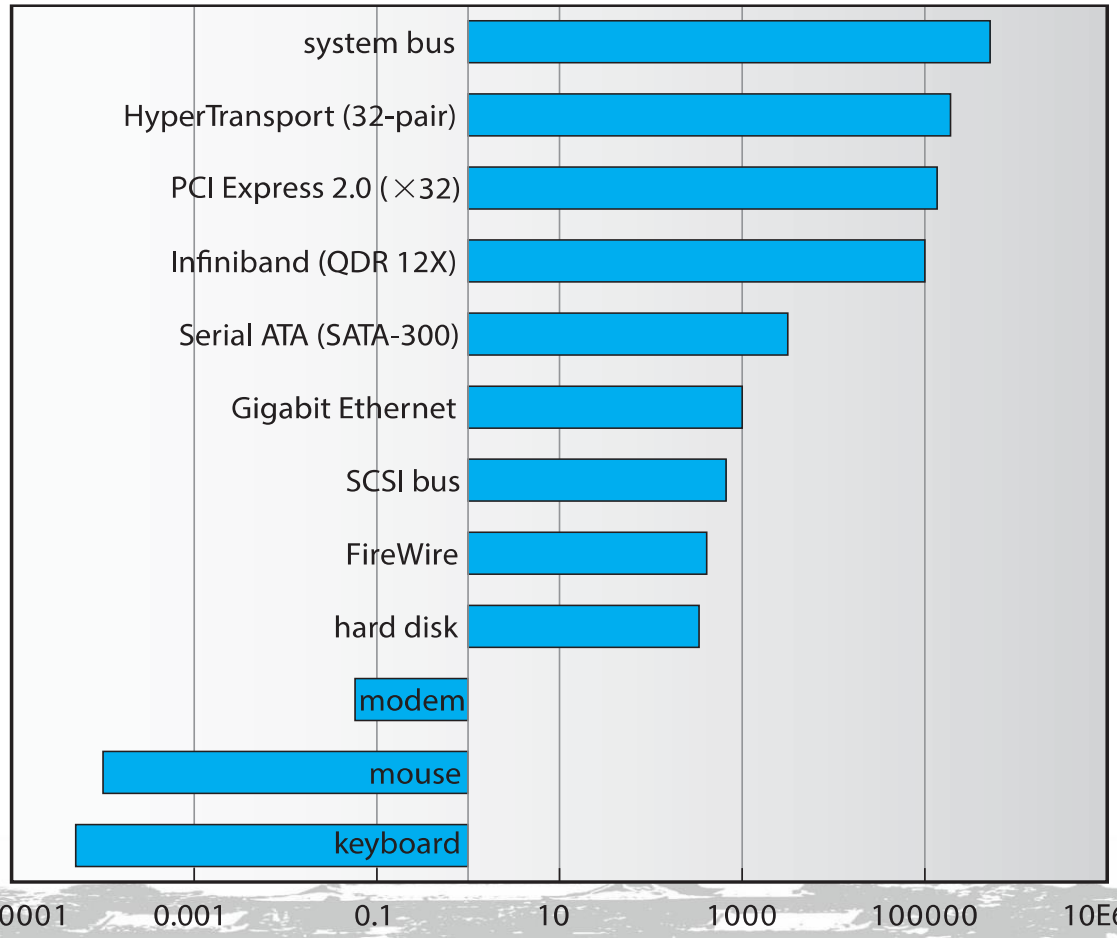
同步I/O

异步I/O

目录

为什么需要缓冲

- 引入缓冲的主要原因：
  ① 缓和CPU与I/O设备间速度不匹配的矛盾



为什么需要缓冲

- 引入缓冲的主要原因：
  - ② 协调逻辑记录大小与物理记录大小不一致
    - 网络传输中，消息分段与重组
  - ③ 支持“复制语义”
    - 场景：write()写磁盘操作，系统调用返回时，应用程序已经改变了缓冲区
    - 原则：操作系统要保证写入磁盘的数据是系统调用时的数据，而不要考虑缓冲区之后的变化
    - 方法：将应用程序缓冲区复制到内核缓冲区，系统调用访问的是内核缓冲区数据

✓ 提高**CPU**与**I/O**设备之间的并行性
✓ 减少了**I/O**设备对**CPU**的中断请求次数，放宽**CPU**对中断响应时间的要求

缓冲技术实现基本思想

① 进程执行写操作输出数据时，向系统申请一个缓冲区，若为顺序写请求，则不断把数据填到缓冲区，直到被装满。此后，进程继续它的计算，系统将缓冲区内容写到I/O设备上。

② 进程执行操作输入数据时，向系统申请一个缓冲区，系统将一个物理记录的内容读到缓冲区，根据进程要求，把当前需要的逻辑记录从缓冲区中选出并传送给进程。

③ 在输出数据时，只有在系统还来不及腾空缓冲而进程又要写数据时，它才需要等待；

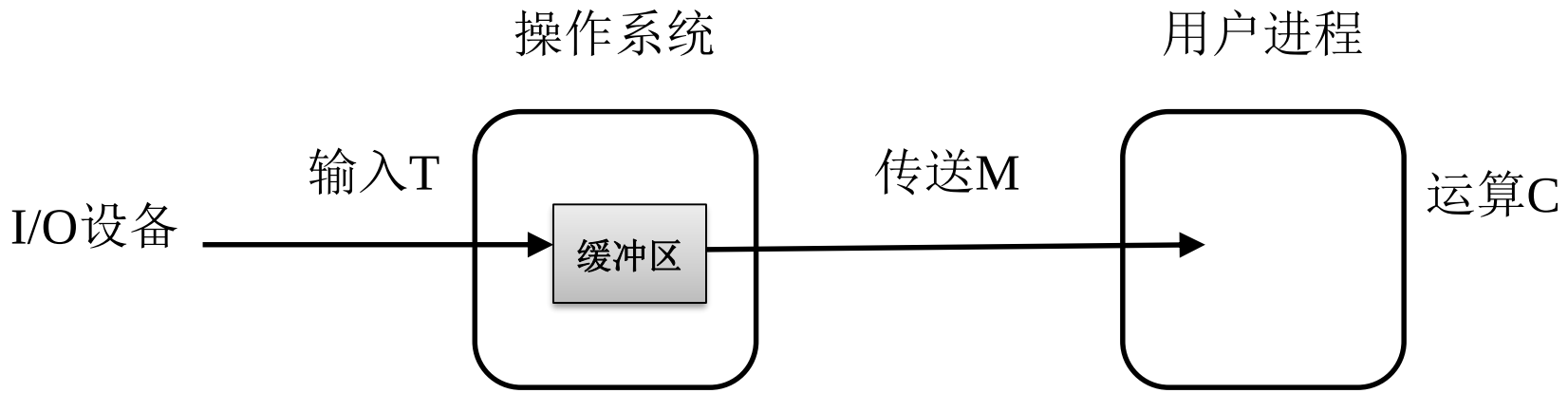
④ 在输入数据时，仅当缓冲区空而进程又要从中读取数据时，它才被迫等待。

缓冲的实现方法

- 硬件缓冲器：
  - I/O控制器中的数据缓冲寄存器，但成本太高。
- 软件缓冲：
  - 一片内存区域，用来临时存放输入输出数据。
- 缓冲技术分为：
  ① 单缓冲
  ② 双缓冲
  ③ 循环缓冲
  ④ 缓冲池
  ⑤ 缓冲区高速缓存

1 单缓冲

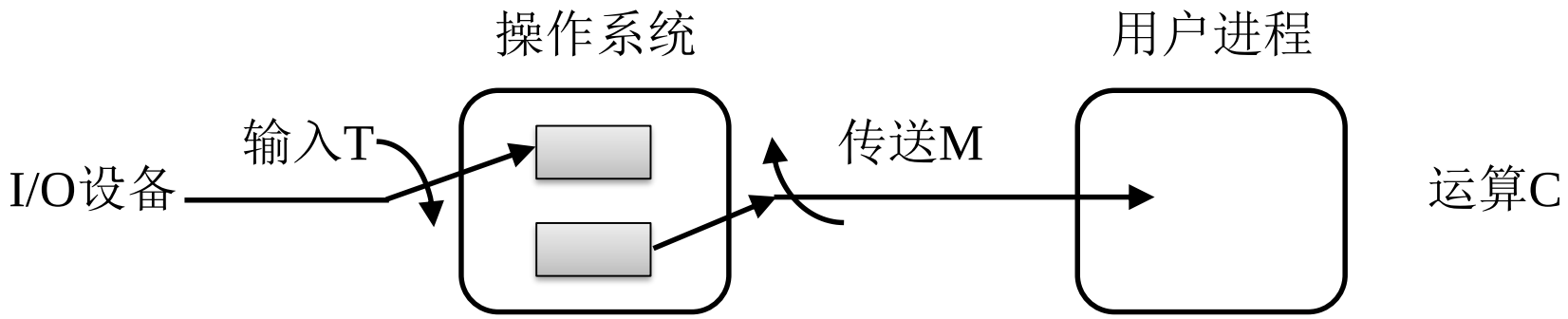
- 单缓冲是在设备和处理机之间设置一个缓冲区。



系统处理大量数据时，一块数据的处理时间是多少？

2 双缓冲

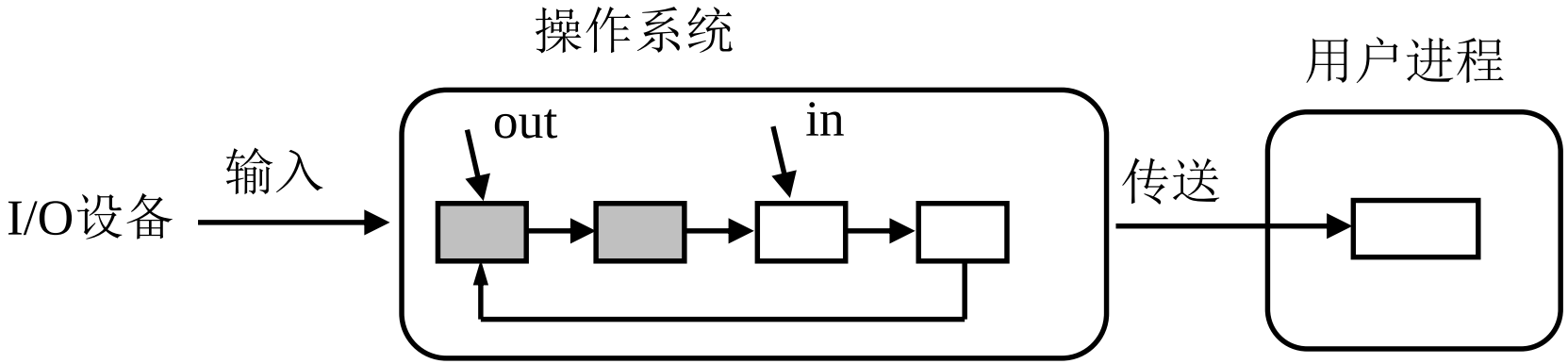
- 引入双缓冲，可以进一步提高处理机与设备的并行操作程度。



系统处理大量数据时，一块数据的处理时间是多少？

3 循环缓冲

- 若输入输出速度与数据处理速度相当，则双缓冲能获得较好的效果；若速度相差较大，则可以通过增加缓冲区的数量来改善性能。



4 缓冲池

- 循环缓冲适用于合作进程，当系统较大且共享缓冲区的进程较多时，这要消耗大量内存。目前广泛使用的是公用缓冲池。
- 缓冲池由多个缓冲区组成，其中的缓冲区可供多个进程共享，既能用于输入又能用于输出。

目录

设备分配的任务

- 当进程提出I/O请求时，设备分配程序便按照一定的策略为其分配设备，同时还应分配相应的控制器和通道，以保证CPU与设备之间的通信。

1 设备分配中的数据结构

- 设备分配依据的主要数据结构有：
  - 设备控制表
  - 控制器控制表
  - 通道控制表
  - 系统设备表

设备控制表（DCT）

- 系统为每个设备配置一张设备控制表，用于记录设备的特性及与I/O控制器连接的情况。

设备控制表

设备类型
设备标识符
设备状态：忙/闲
指向控制器表的指针
设备等待队列指针
…

控制器控制表（COCT）

- 控制器控制表也是每个控制器一张，它反映I/O控制器的使用状态以及和通道的连接情况。

控制器控制表

控制器标识符
控制器状态：忙/闲
指向通道表的指针
控制器等待队列指针
…

通道控制表（CHCT）

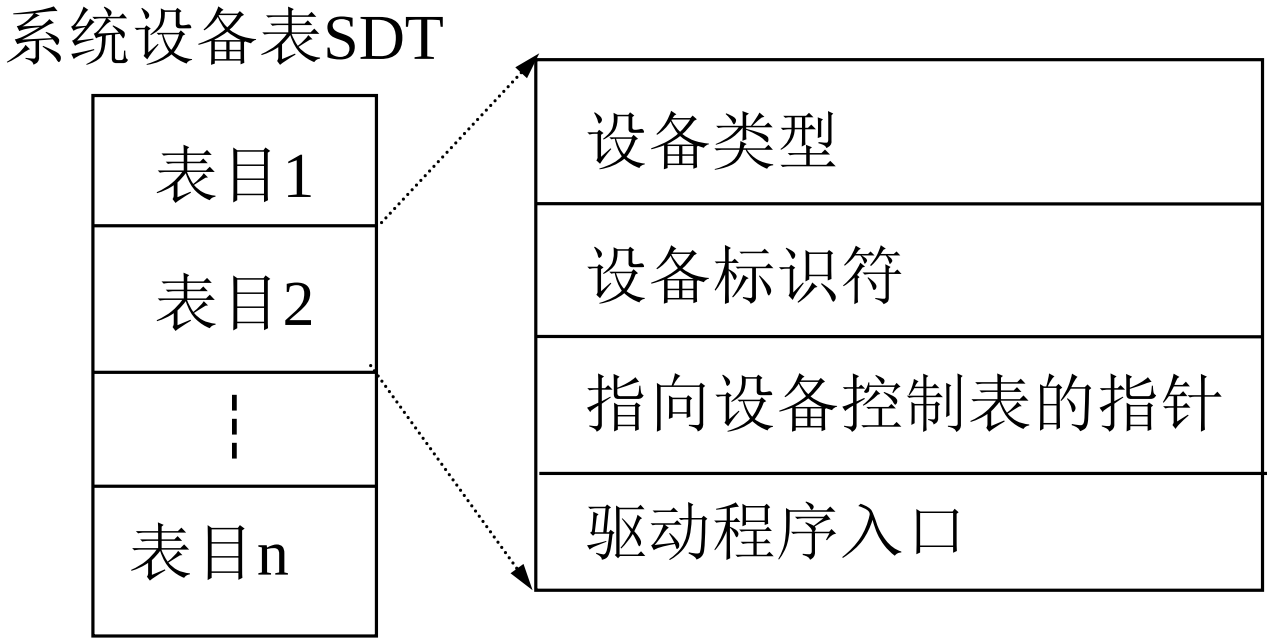
- 每个通道都配有一张通道控制表，它反映通道的使用状态。

通道控制表

通道标识符
通道状态：忙/闲
通道等待队列指针
…

系统设备表（SDT）

- 系统设备表整个系统一张，它记录了系统中所有物理设备的情况，每个物理设备占一个表目。



2 设备分配策略

- 系统在进行设备分配时，应考虑以下因素：
  - 设备的使用性质
  - 设备分配算法
  - 设备分配的安全性
  - 设备独立性

（1）设备的使用性质

- 按共享属性设备有三种类型：
  - 独占：这种设备在一段时间内只允许一个进程独占。
  - 共享：这种设备允许多个进程同时共享。
  - 虚拟：设备本身虽是独占设备，但经过某种技术处理后可改造成虚拟设备，使之成为共享设备。

根据设备性质的不同分配方式

- 独享分配
  - 在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后，系统才能再将该设备分配给其他进程使用。
- 共享分配
  - 将设备同时分配给多个进程使用。但这些进程对设备的访问需要进行合理调度。
- 虚拟分配：针对虚拟设备
  - 当进程申请独占设备时，系统给它分配共享设备上的一部分存储空间；
  - 当进程要与设备交换信息时，系统就把要交换的信息存放在这部分存储空间中；
  - 在适当的时候，将设备上的信息传输到存储空间中或将存储空间中的信息传送到设备。

（2）设备分配算法

- 设备分配通常只采用以下两种算法：
  - 先来先服务：根据进程对某设备发出请求的先后次序，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：按对某设备提出I/O请求的进程优先级由高到低排队，对优先级相同的I/O请求，按先来先服务的算法排队，设备分配程序总是把设备首先分配给队首进程。

(3) 设备分配的安全性

- 设备分配的安全性是指在设备分配中应保证不发生进程死锁。
- 设备有两种分配方式：
  - 静态分配
  - 动态分配

静态分配

- 静态分配：用户作业开始执行前，由系统一次分配该作业所要求的全部设备、控制器和通道。一旦分配，就一直为该作业所占用，直到该作业被撤消为止。

- 静态分配方式不会出现进程的死锁，但设备的使用效率低。

动态分配

- 动态分配：在进程执行过程中根据需要进行设备分配，一旦使用完之后便立即释放。
- 动态分配方式有利于提高设备利用率，但分配算法使用不当有可能造成进程死锁。
- 动态分配又分为：
  - 安全分配
  - 不安全分配

安全分配

- 安全分配：每当进程发出I/O请求后便进入阻塞状态，直到I/O操作完成时才被唤醒并释放资源。
- 特点：设备分配安全，但进程进展缓慢。

不安全分配

- 不安全分配：进程发出I/O请求后继续运行，需要时又可发出第二个I/O请求、第三个I/O请求等。仅当进程所请求的设备被其他进程占用时才进入阻塞状态。
- 特点：进程推进迅速，但可能死锁。

（4）设备独立性

- 设备独立性：
  - 又称设备无关性，是指用户编制程序时使用的设备与实际使用的物理设备无关。
  - 逻辑设备：用户程序中使用的设备。
  - 物理设备：计算机系统中存在的设备。
- 设备独立性的优点
  - 设备分配灵活：不指定物理设备，只使用逻辑设备，映射关系由系统决定，提高资源使用
  - 易于实现I/O重定向：可以更换设备，而不必修改程序

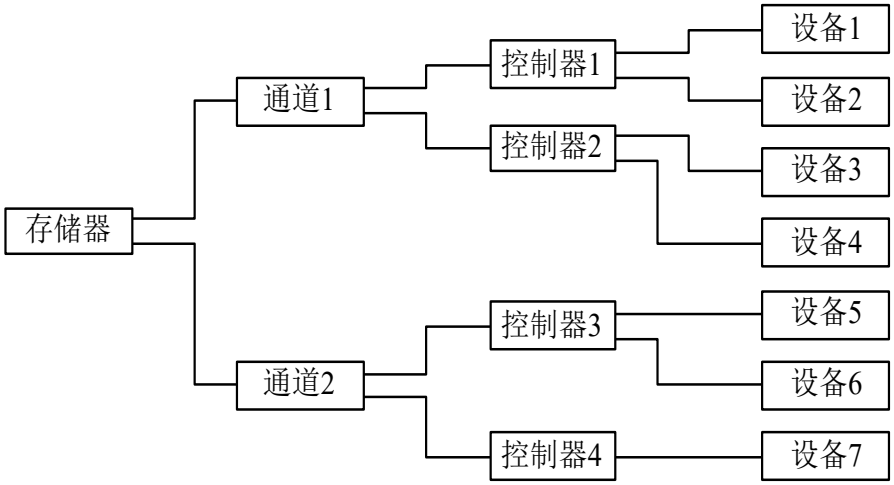
设备独立性的实现

- 为实现设备独立性，应在用户程序中使用逻辑设备名请求某类设备，而在程序实际执行时使用物理设备名。为此系统应具有将逻辑设备名转换为物理设备名的功能。

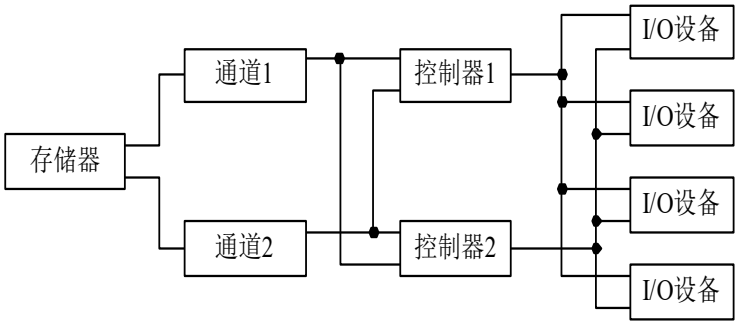
- 可以采用逻辑设备表LUT(Logical Unit Table)实现变换。

逻辑设备名	物理设备名	驱动程序入口地址
/dev/tty	3	1024
/dev/print	5	2046
……	……	……

3.设备分配程序



- 单通路I/O系统



- 多通路I/O系统

3.设备分配程序

单通路设备分配

- 单通路设备分配程序假定用户使用物理设备名，采用单通路的I/O系统结构。
- 当进程提出I/O请求时，系统的设备分配程序按下述步骤进行：
  - 分配设备
  - 分配控制器
  - 分配通道

分配设备

- SDT→DCT
  - 根据物理设备名查找SDT(系统设备表)，从中找到该设备的DCT(设备控制表)。

- DCT中检查设备状态，并考虑分配
  - 根据DCT中的设备状态字段查看设备忙否，若忙则将进程插入设备等待队列；
  - 否则计算本次设备分配的安全性，若安全则进行分配，否则仍将该进程插入设备等待队列。

分配控制器

- DCT→COCT
  - 设备分配后，再到DCT中找到与该设备相连的COCT(控制器控制表)，
- COCT中检查控制器状态
  - 从该表的状态字段中可知该控制器是否忙碌。
  - 若忙，则将进程插入该控制器的等待队列；否则将该控制器分配给进程。

分配通道

- COCT→CHCT
  - 从COCT中找到与该控制器连接的CHCT(通道控制表)，
- CHCT中检查状态
  - 从该表的状态字段中可知该通道是否忙碌。
  - 若忙，则将进程插入该通道的等待队列；否则将该通道分配给进程。

- 待以上操作完成后，将相应的设备、控制器、通道分配给进程后，便可以启动I/O设备实现I/O操作。

多通路的设备分配

- 单通道设备分配的问题
  - 进程按照物理设备名提出I/O请求，当指定的物理设备被占用，而其他同类设备为空时，不可复用

- 为了提高系统的灵活性和可靠性，通常采用多通路的I/O系统结构。
  - 用户使用逻辑设备名提出请求
  - 对于设备、控制器、通道均多次申请

目录

Spooling系统

- Spooling技术是将独占设备改造为共享设备的技术。
- Spooling是Simultaneous Peripheral Operating On-Line的缩写，意思是外部设备同时联机操作，又称假脱机操作。
- 用一道程序模拟脱机输入时的外围控制机功能，把低速输入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的外围控制机功能，把数据从磁盘传送到低速输出设备上。
- 采取预输入、缓输出、以及井管理程序来实现

Spooling系统的组成

- Spooling系统由三部分组成：
  - 输入井和输出井
  - 输入缓冲区和输出缓冲区
  - 输入进程和输出进程

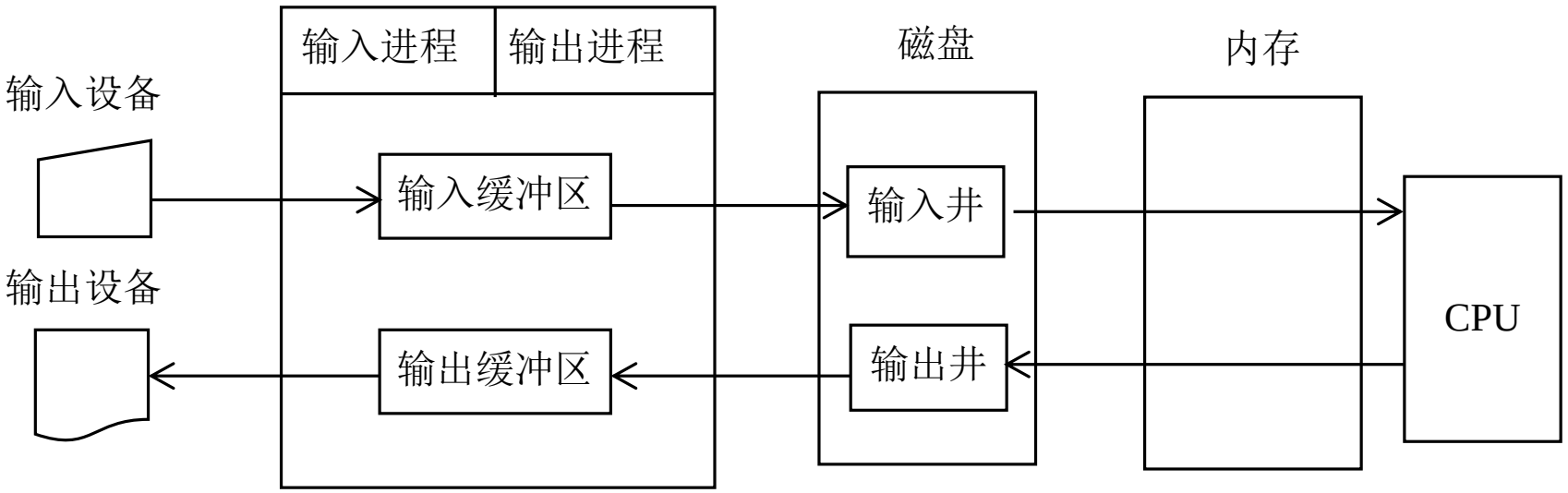
- 输入井和输出井
  - 输入井和输出井是磁盘上的两个存储区域。
  - 输入井收容I/O设备输入的数据。
  - 输出井收容用户程序的输出数据。

Spooling系统的组成

- 输入缓冲区和输出缓冲区
  - 输入缓冲区和输出缓冲区是内存中的两个缓冲区。
  - 输入缓冲区用于暂存由输入设备送来的数据，以后再传送到输入井；
  - 输出缓冲区用于暂存从输出井送来的数据，以后再传送到输出设备。

- 输入进程和输出进程
  - 输入进程模拟脱机输入时的外围控制机，将用户要求的数据从输入设备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；
  - 输出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系统组成 图示



Spooling应用——打印机

- 打印机的SPOOLing守护进程
  - 对于打印机，由于进程可能打开设备而长期不使用，则导致大量浪费，因此采用了SPOOLing技术。
  - 应用程序将待打印文件存放在打印目录
  - 打印机守护进程(Daemon)，是唯一具有使用打印机设备文件的进程，当打印机空闲，便启动守护进程进行打印

练习

- 某系统中，如果从磁盘读取一块数据到到缓冲区的时间为T，从缓冲区把数据取到用户区时间为 M，进程对数据进行运算时间为C。那么在单缓冲和双缓冲的情况下，处理大量数据时，一块数据的处理时间是多少？