第五章 设备管理

- I/O 硬件原理
- I/O 软件原理
- 具有通道的I/O系统
- 缓冲技术
- 驱动调度技术
- 设备分配
- 虚拟设备

本章要点

- 设备管理的功能、模型
- I/O子系统的组成
- I/O控制
- 设备的分配
- 设备的类型
- I/O缓冲技术
- 虚拟设备和SPOOLING系统
- 磁盘的管理

5.0 引言

- 计算机的外围设备分为两类：
  - 存储型设备：作为主存的扩充，以存储大量持久性信息和快速检索为目标
  - I/O型设备：把外界信息输入计算机，把计算结果输出，完成人机交互

5.0 引言（续）

- 设备管理的主要功能在于克服设备和CPU速度不匹配
  - 设备中断处理
  - 设备的分配和去配
  - 设备驱动调度
  - 缓冲区管理
  - 虚拟设备及其实现
- 设备管理的基本手段在于系统将所有设备都定义为文件

5.1 I/O 硬件原理

- I/O 系统
- I/O 控制方式
- 设备控制器

- I/O 系统
- I/O 控制方式
- 设备控制器

5.1.1 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：
  - 计算机的主存和外围设备的介质之间的信息传送操作

5.1.1：分类

- 按照I/O特性，I/O设备划分为：
  - 输入型设备
  - 输出型设备
  - 存储型设备
    - 顺序存取存储设备：顺序存取存储设备严格依赖信息的物理位置进行定位和读写，如磁带
    - 直接存取存储设备：直接存取存储设备的重要特性是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，如磁盘

5.1.1：分类（续）

- 按照I/O信息交换的单位可分为：
  - 字符设备
    - 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
  - 块设备
    - 存储型外围设备一般为块设备

- I/O 系统
- I/O 控制方式
- 设备控制器

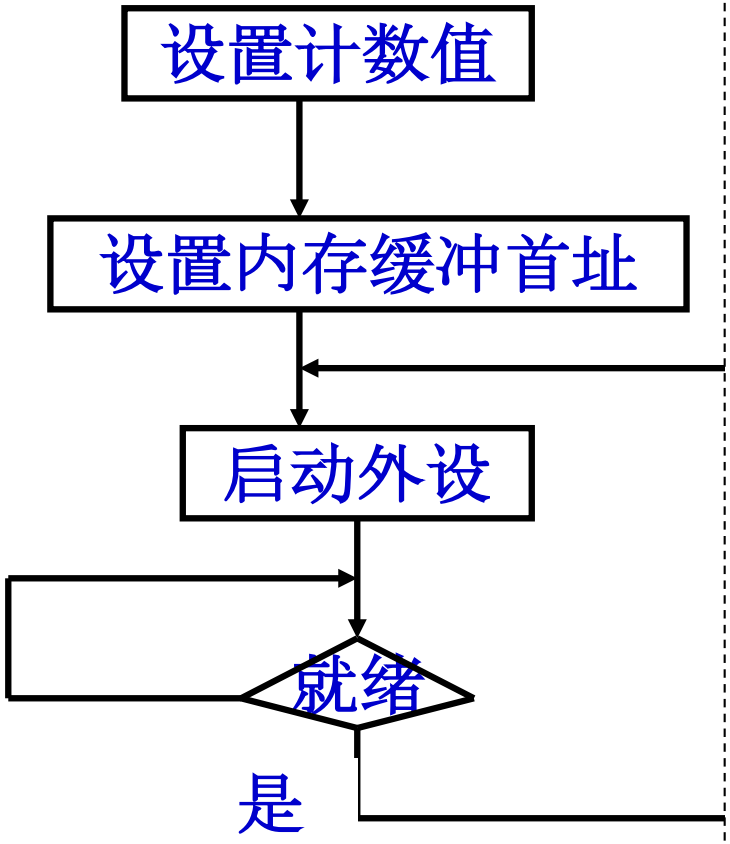
5.1.2 I/O控制方式

- 按照I/O控制器功能的强弱，以及和CPU之间联系方式的不同，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：中央处理器和外围设备并行工作的方式和程度不同

5.1.2 轮询方式

- 轮询方式又称程序直接控制方式，使用查询指令测试设备控制器的忙闲标志位，确定主存和设备是否能交换数据

5.1.2：轮询过程

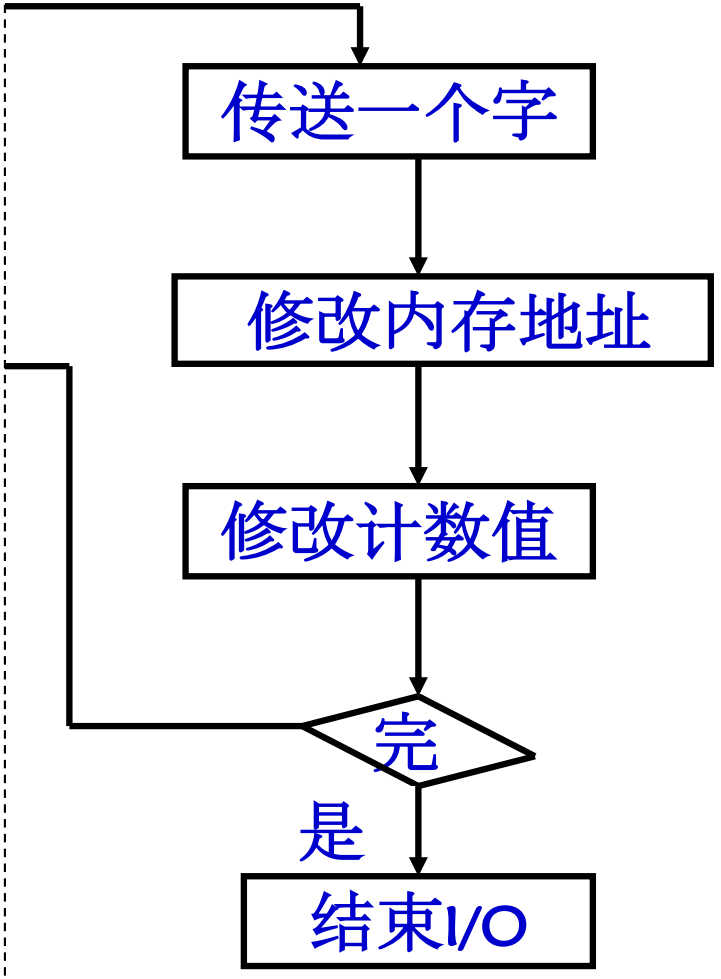


1. 如时正运行的程序需从设备读入一批数据，则该程序设置交换字节数和数据读入主存的起始地址，然后向设备发出查询指令，设备控制器便把状态返回给CPU。
2. 如果I/O操作忙或未就绪，则重复测试过程，继续查询

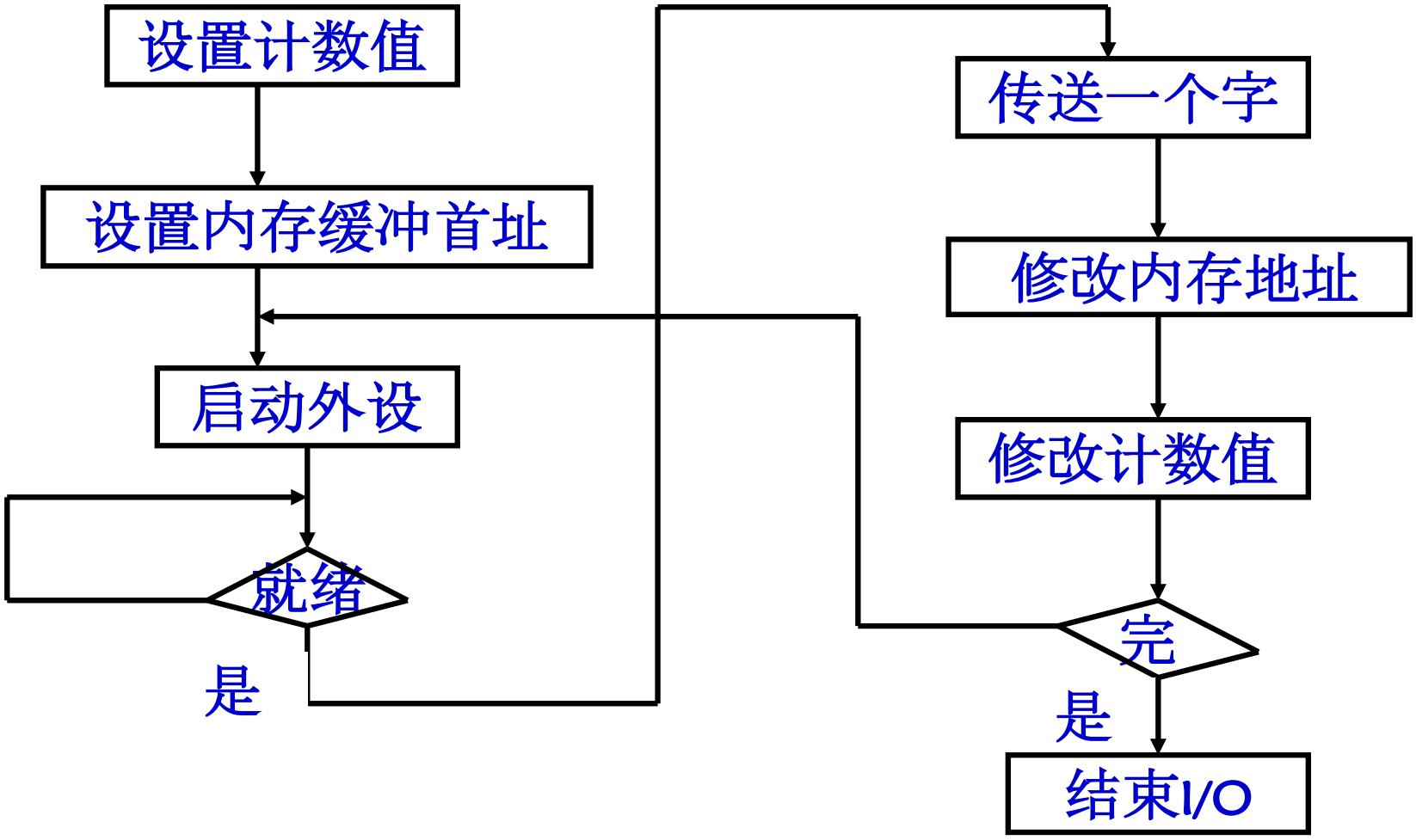
5.1.2：轮询过程

3. 否则开始数据传送，CPU从I/O接口（控制器寄存器）读取一个字，再用存储指令保存到主存

4. 如果传送尚未结束，再次向设备发出查询指令，直到全部数据传送完成



5.1.2：轮询过程



5.1.2：轮询指令

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

5.1.2：轮询的弊端

- 一旦CPU启动I/O设备，便不断查询I/O设备的准备情况，终止原程序的执行，浪费CPU时间
- I/O准备就绪后，CPU参与数据传送工作，而不能执行原程序
- CPU和I/O设备串行工作，使主机不能充分发挥效率，设备也不能得到合理使用，整个系统效率很低

5.1.2：中断方式

- 硬件要求：
  - ①CPU与设备控制器及设备之间存在中断请求线
  - ②设备控制器的状态寄存器有相应的中断允许位

5.1.2：中断过程

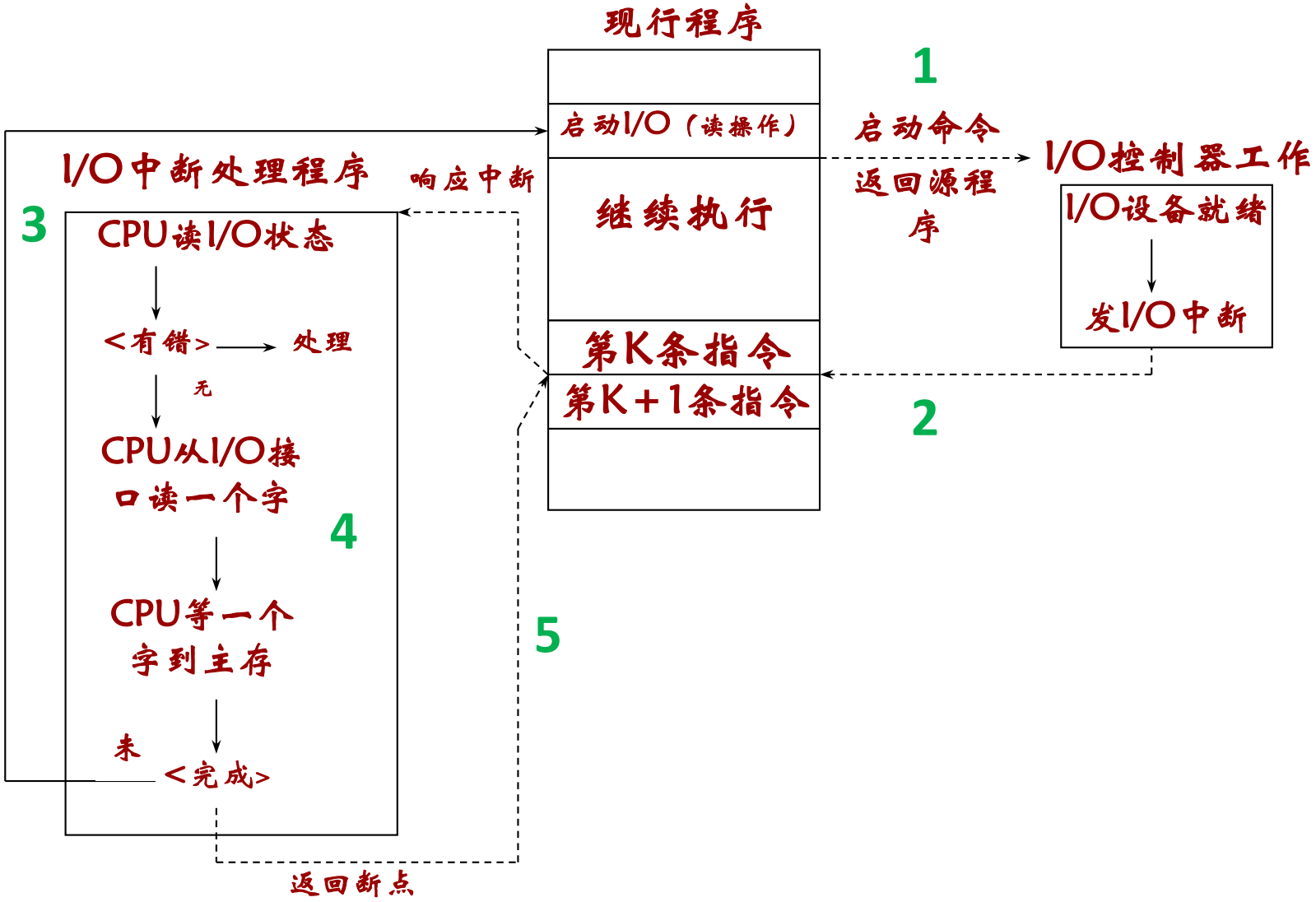
1. 进程发出启动I/O指令后，控制信息加载到设备控制器寄存器后，进程继续执行不涉及本次I/O数据的任务，或放弃CPU等待I/O操作完成

2. 设备控制器检查寄存器的内容，按照I/O指令的要求执行。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。

5.1.2：中断过程

3. CPU收到并响应I/O中断后，转向设备的I/O中断处理程序执行
4. 中断处理程序执行数据读取操作，将I/O缓冲寄存器的内容写入主存，操作结束后退出
5. 进程调度程序在适当时刻将得到数据的进程恢复

5.1.2：中断过程


现行程序
1
启动I/O（读操作）
启动命令
返回源程序
I/O控制器工作
I/O设备就绪
发I/O中断
继续执行
第K条指令
第K+1条指令
2
I/O中断处理程序
响应中断
3
CPU读I/O状态
<有错>
处理
无
CPU从I/O接口读一个字
4
CPU等一个字到主存
未
<完成>
5
返回断点

5.1.2：中断过程（续）

- CPU启动I/O设备后，不必查询I/O设备是否就绪，继续执行现行程序
- 直到在启动指令后的某条指令，响应了I/O中断请求，CPU才转至I/O中断处理程序执行

5.1.2：中断过程（续）

- 中断处理程序中，CPU全程参与数据传输操作，它从I/O接口读一个字(字节）并写入主存，如果I/O设备上的数据尚未传送完成，转向现行程序再次启动I/O设备，重复上述过程；否则，中断处理程序结束后，继续从K+1条指令执行
- I/O操作直接由CPU控制，每传送一个字符或字，要发生一次中断，仍然消耗大量CPU时间

5.1.2：中断过程（续）

- 中断处理程序中，CPU全程参与数据传输操作，它从I/O接口读一个字(字节）并写入主存，如果I/O设备上的数据尚未传[大小取决于设备控制器的数据缓冲区]再次启动I/O设备，重复上述过程[]结束后，继续从K+1条指令执行
- I/O操作直接由CPU控制，每传送一个字符或字，要发生一次中断，仍然消耗大量CPU时间

5.1.2：中断方式的优势

- 程序中断方式I/O，不必忙于查询I/O准备情况
- CPU和I/O设备可实现部分并行
- 与程序查询的串行工作方式相比，使CPU资源得到较充分利用

5.1.2：中断方式的缺陷

- 如果设备控制器的数据缓冲区较小，则传送过程中发生中断的次数较多，会耗用大量CPU时间。
- 若系统配备多种设备，都采用中断方式，则有可能由于中断次数过多而造成CPU来不及处理，数据丢失。

5.1.2：DMA方式

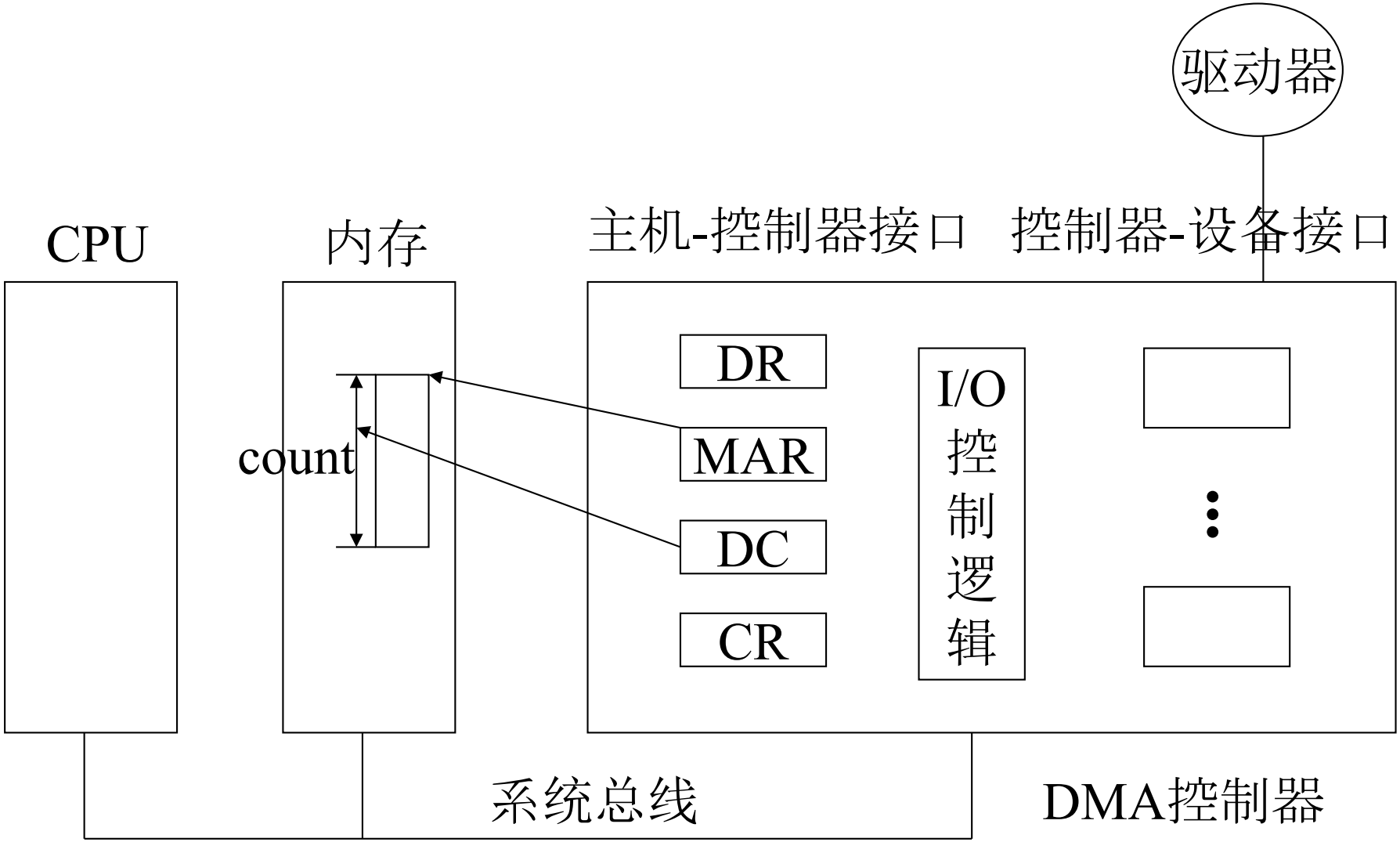
- 如果I/O设备能直接与主存交换数据而不占用CPU,CPU的利用率还可提高,这就出现了直接存储器存取 DMA（Direct Memory Access）方式
- 需要的逻辑部件
  1. 主存地址寄存器(MAR Memory Address Register):存放主存中需要交换的数据的地址.DMA传送前,由程序送入首地址,在DMA传送中,每交换一次数据,把地址寄存器内容加1
  2. 字计数器(DC):记录传送数据的总字数,每传送一个字,字计数器减1
  3. 数据缓冲寄存器或数据缓冲区(DR):暂存每次传送的数据

5.1.2：DMA方式

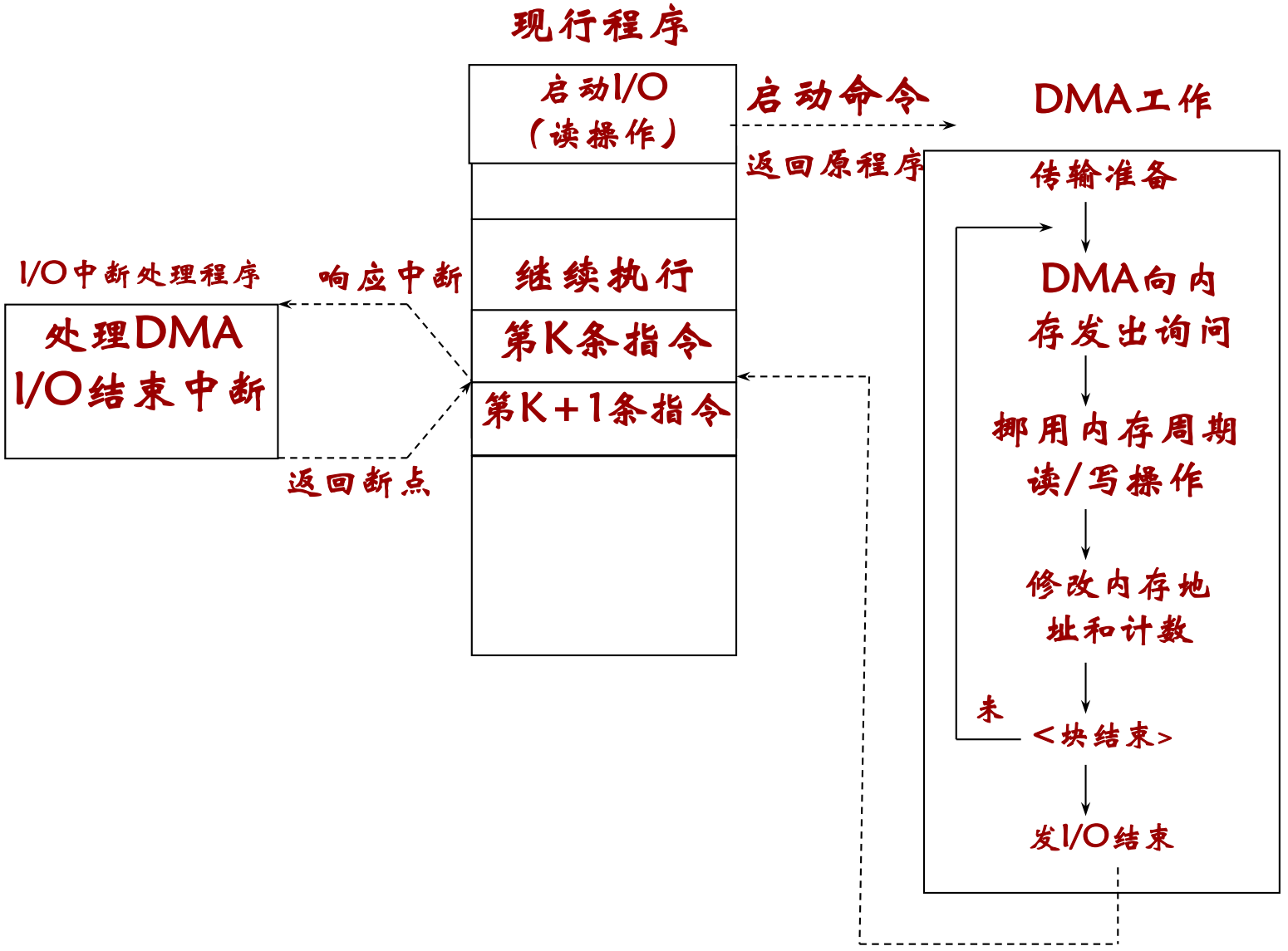
- 需要的逻辑部件

4. 设备地址寄存器：存放I/O设备信息，如磁盘的柱面号、磁道号、块号等
5. 中断机制和控制逻辑：用于向CPU提出I/O中断请求和保存CPU发送来的I/O命令及管理DMA的传送过程

5.1.2：DMA控制器


驱动器
CPU
内存
主机-控制器接口
控制器-设备接口
DR
MAR
DC
CR
count
I/O
控
制
逻
辑
系统总线
DMA控制器

5.1.2：DMA 过程


现行程序
启动I/O
（读操作）
启动命令
DMA工作
返回原程序
传输准备
继续执行
DMA向内
存发出询问
I/O中断处理程序
响应中断
处理DMA
I/O结束中断
第K条指令
第K+1条指令
挪用内存周期
读/写操作
修改内存地
址和计数
未
<块结束>
返回断点
发I/O结束

5.1.2：通道方式

- 为获得CPU和外围设备间更高的并行工作能力，也为了让种类繁多，物理特性各异的外围设备能以标准的接口连接到系统中，计算机系统引入了自成独立体系的通道结构
- 通道也叫输入输出处理器. 通道是独立于CPU的专门负责数据输入/输出传输工作的处理机，对外部设备实现统一管理，代替CPU对输入/输出操作进行控制，从而使输入，输出操作可与CPU并行操作

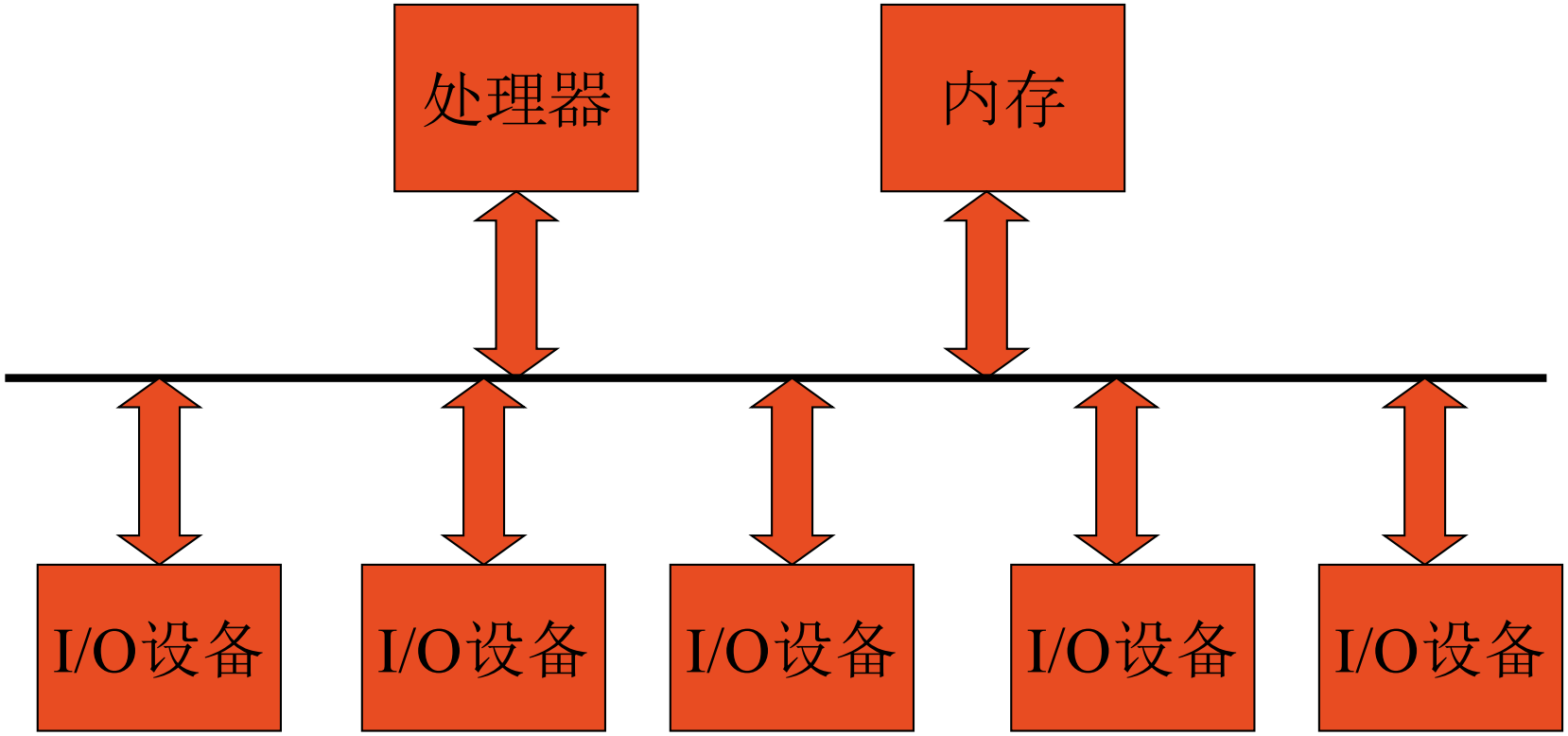
5.1.2：通道方式

- 通道负责管理设备与内存之间的数据传送的一切工作
- 能完成主存和外围设备间的信息传送，与CPU并行地执行操作。通道技术解决了I/O操作的独立性和各部件工作的并行性
  - 外围设备和CPU能实现并行操作
  - 通道和通道之间能实现并行操作
  - 各通道上的外围设备也能实现并行操作，达到提高整个系统效率这一根本目的

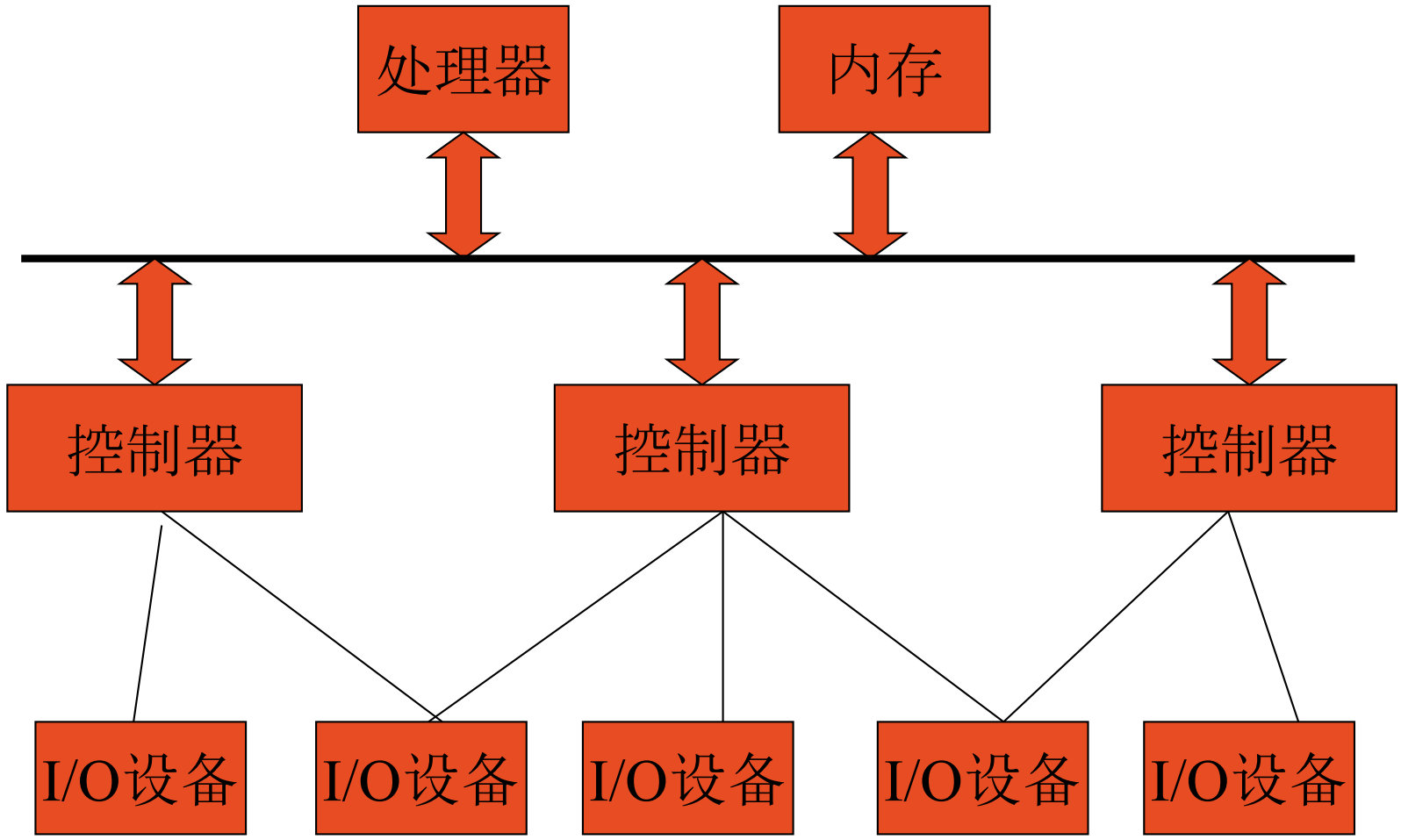
5.1.2：通道I/O过程

1. CPU在执行主程序时遇到I/O请求，它启动指定通道上选定的外围设备，一旦启动成功，通道开始控制外围设备进行操作
2. CPU就可执行其他任务并与通道并行工作，直到I/O操作完成。通道发出操作结束中断时，CPU才停止当前工作，转向处理I/O操作结束事件

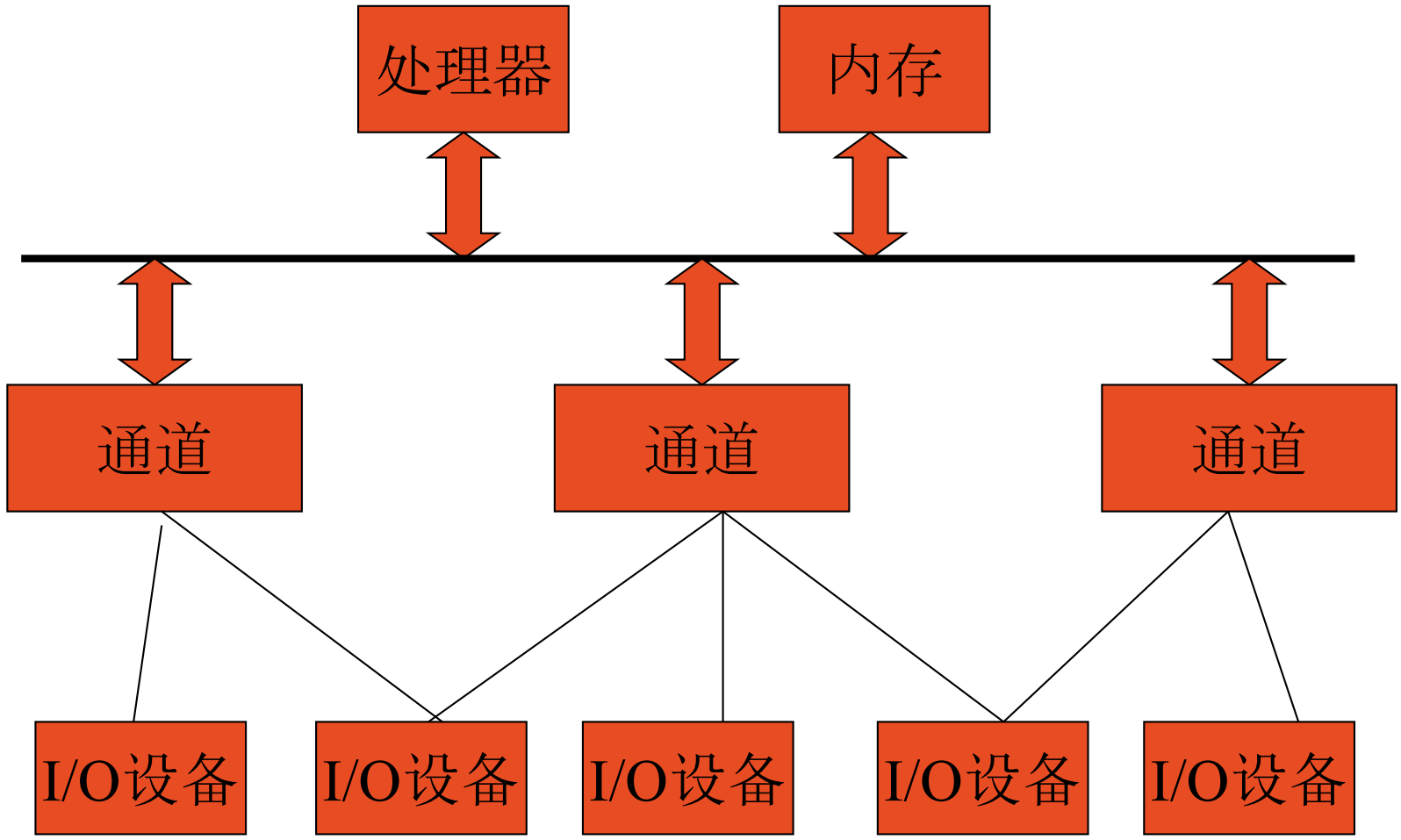
轮询或中断



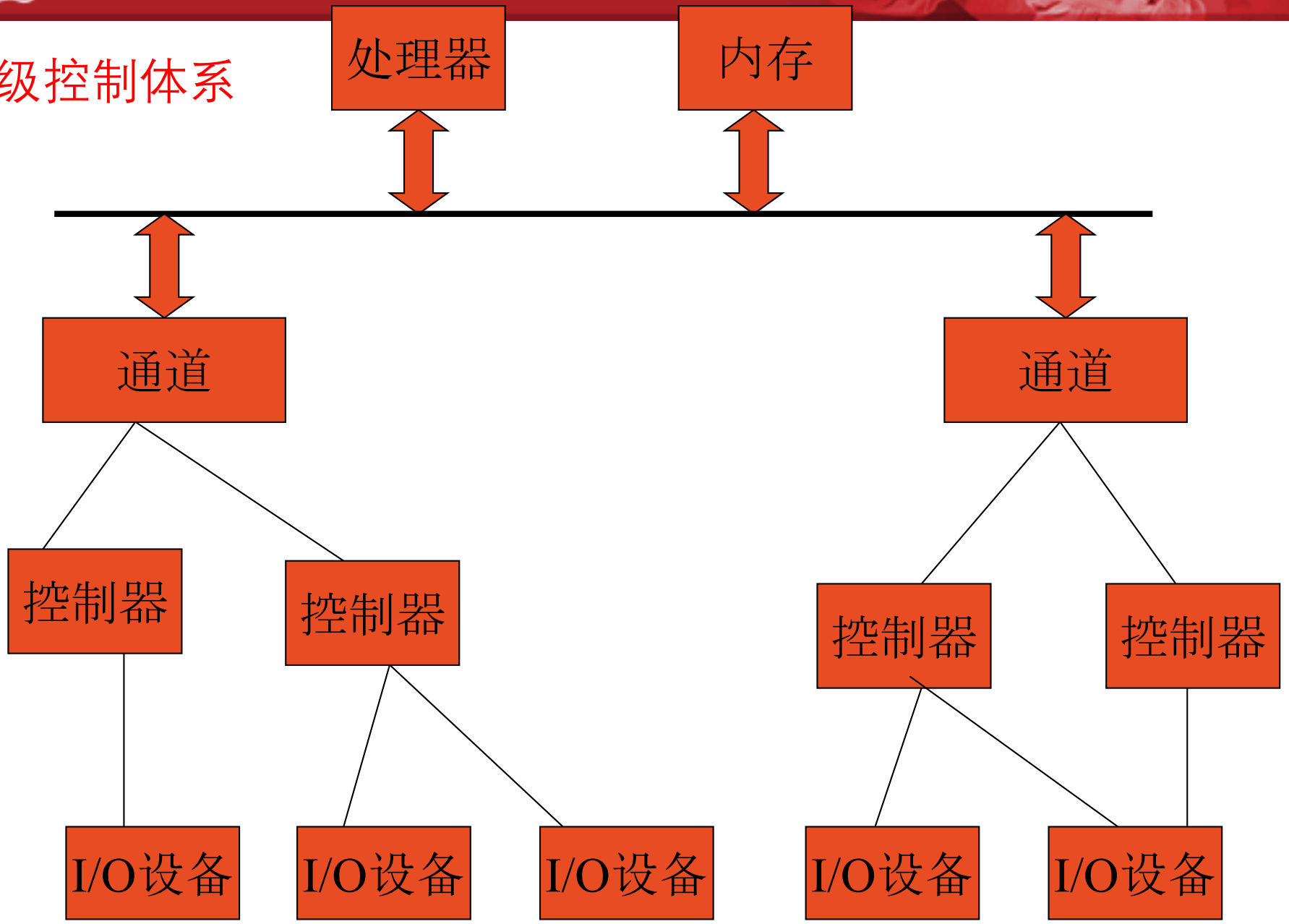
DMA



通道



三级控制体系



5.1.2：通道类型（1）

- 字节多路通道 （慢速设备）

以字节为单位传输信息，它可以分时地执行多个通道程序。当一个通道程序控制某台设备传送一个字节后，通道硬件就转去执行另一个通道程序，控制另一台设备传送一个字节信息。

一个通道程序相当于一个进程，通道程序之间是顺序执行关系，不是并发执行关系

5.1.2：通道类型（2）

- 选择通道（快速设备）

选择通道以分时方式执行多道通道程序，每次执行一个通道程序可以控制一台设备传送一批数据。数据传输完成后，选择通道再执行另一个通道程序，控制另一台设备传输一批数据

5.1.2：通道类型（3）

- 数组多路通道（高速设备）

数组多路通道先为一台设备执行通道程序中的一部分通道指令，传输一批数据，然后自动转接，为另一台设备执行一部分通道指令，传输另外一批数据

减少外设申请使用通道时的等待时间

允许多个设备同时工作，但只允许一个设备进行传输型操作，而其他设备进行控制型操作；

- I/O 系统
- I/O 控制方式
- 设备控制器

5.1.3 设备控制器

- I/O设备包括一个机械部件和一个电子部件。为了达到设计的模块性和通用性，一般将其分开
  - 电子部件称为设备控制器或适配器，在PC中，它常常是插入主板扩充槽的印刷电路板
  - 机械部件则是设备本身

5.1.3（续）

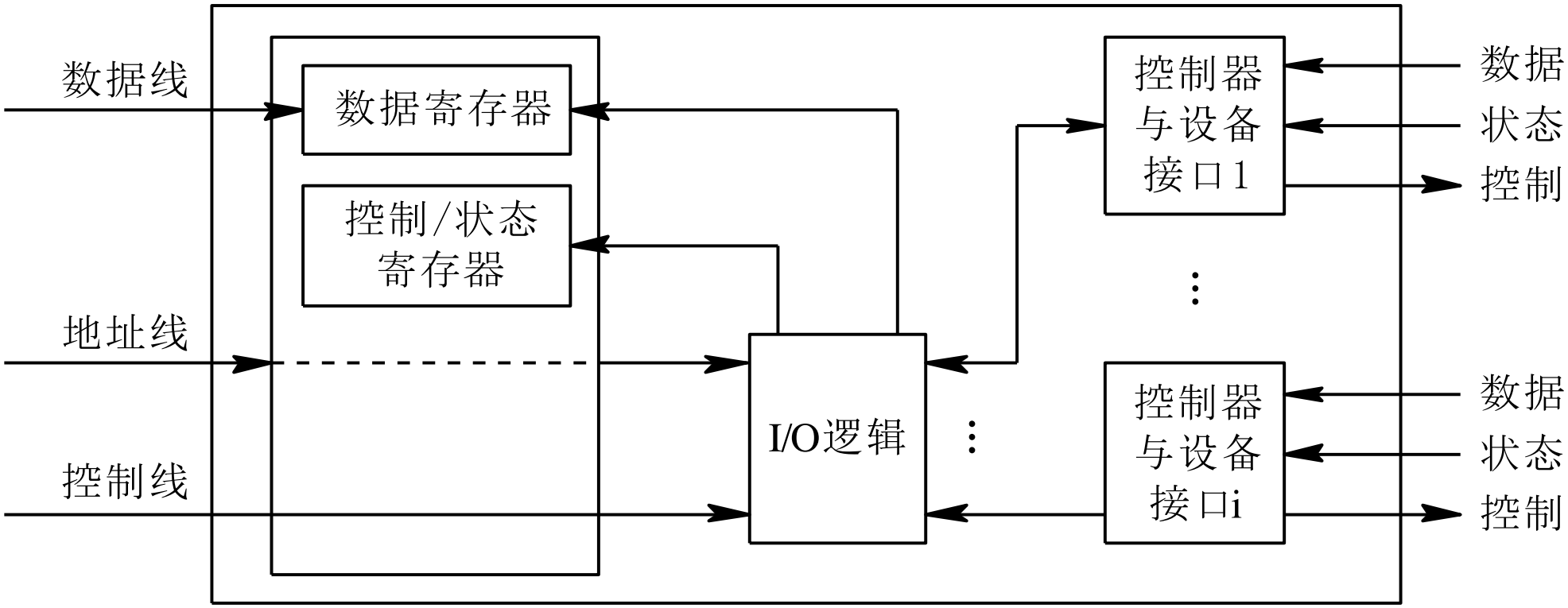
- 操作系统基本上与控制器打交道，而非设备本身
- 多数PC的CPU和控制器之间的通信采用单总线模型，CPU直接控制设备控制器进行I/O
- 大型主机则采用多总线结构和通道方式，以提高CPU与输入输出的并行程度

5.1.3：设备控制器组成

- 命令寄存器及译码器：接收CPU向设备发来的控制命令
- 数据寄存器：存放CPU与外设之间交换的数据
- 状态寄存器：反映外设的工作状态
- 地址译码器：识别同一控制器连接的不同外设
- 用于对设备操作进行控制的I/O逻辑

5.1.3：接口部分结构

CPU与控制器接口



5.1.3：设备控制器功能

1. 接收和识别CPU或通道发来的命令

2. 实现数据交换，包括设备和控制器间的数据传输

3. 发现和记录设备及自身的状态信息，供CPU处理使用

4. 设备地址识别

5.2 I/O 软件原理

- I/O 软件的设计目标和原则
- I/O 中断处理程序
- I/O 设备驱动程序
- 独立于设备的I/O软件
- 用户空间的I/O软件

- I/O 软件的设计目标和原则
- I/O 中断处理程序
- I/O 设备驱动程序
- 独立于设备的I/O软件
- 用户空间的I/O软件

5.2.1 I/O软件设计目标、原则

- 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：屏蔽低层错误
  - 同步（阻塞）—异步（中断驱动）传输
  - 缓冲技术

5.2.1：I/O软件组织层次

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


层次
I/O应答
I/O功能
用户进程
进行I/O调用；格式化I/O；SPOOLING
I/O请求
设备无关软件
命名；保护；阻塞；缓冲；分配
设备驱动程序
建立设备寄存器；检查状态
中断处理程序
当I/O结束时，唤醒驱动程序
硬件
执行I/O操作

- I/O 软件的设计目标和原则
- I/O 中断处理程序
- I/O 设备驱动程序
- 独立于设备的I/O软件
- 用户空间的I/O软件

5.2.2 I/O中断处理程序

- 当一个进程请求I/O操作时，该进程将被挂起，直到I/O操作结束并发生中断。当中断发生时，中断处理程序执行相应的处理，并解除相应进程的阻塞状态
- 中断层的主要工作有：
  - 处理中断信号
  - 修改进程状态等

5.2.2 处理原则

- 操作正常结束：释放等该设备或通道的进程
- 操作异常结束：设备本身的故障、通道的故障、通道程序错、启动命令错
- 人为要求而产生的中断
- 外围设备上来的异步信号

- I/O 软件的设计目标和原则
- I/O 中断处理程序
- I/O 设备驱动程序
- 独立于设备的I/O软件
- 用户空间的I/O软件

5.2.3 设备驱动程序

- 设备驱动程序的工作是把用户提交的逻辑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

5.2.3：程序功能

1. 设备初始化
   - 预置设备和控制器及通道的状态
2. 执行设备驱动例行程序
   - 负责启动设备，进行数据传输，启动通道工作
3. 执行中断处理程序
   - 负责处理设备和控制器及通道发出的各种中断

5.2.3：处理过程

- 驱动程序发出I/O命令后的动作，有两种可能性：
  - 驱动程序阻塞，直到中断信号到达时才被唤醒。因为I/O操作通常需要花费一定的时间才能完成
  - I/O操作没有任何延迟，驱动程序无需阻塞。例如向屏幕上输出一些信息

5.2.3：程序后程

- I/O操作完成后驱动程序的工作：
  1. I/O操作完成，驱动程序被唤醒后，便检查有无错误，若一切正常，则驱动程序将数据传送给上层的设备无关软件，
  2. 向它的调用者返回一个关于错误报告的状态信息。
  3. 若请求队列中有别的请求则它选中一个进行处理，若没有则它阻塞，等待下一个请求

- I/O 软件的设计目标和原则
- I/O 中断处理程序
- I/O 设备驱动程序
- 独立于设备的I/O软件
- 用户空间的I/O软件

5.2.4：独立于设备的I/O软件

- 基本功能：
  1. 设备命名和设备保护
  2. 提供与设备无关的块尺寸
  3. 缓冲技术
  4. 设备分配和状态跟踪
  5. 错误处理报告

5.2.4（1）设备命名和设备保护

- 设备都被视为文件，有相应的名字和支持与文件相关的所有系统调用，如open，read，stat等
- 设备保护检查设备是否有权访问所申请的设备
- I/O指令为特权指令，用户通过系统调用间接使用
- 设备文件依赖inode来实现

5.2.4（2）：与设备无关的块尺寸

- 操作系统为每个磁盘都配备一张记录空闲块的表或位示图
- 分配空闲块的算法独立于设备，可在高于设备驱动的程序处理
- 不同磁盘的扇区大小可能不同，必须屏蔽这一事实，向高层软件提供统一的数据块尺寸。

5.2.4（3）：缓冲技术

- 消除填满速率和清空速率的影响
- 块设备和字符设备的不一致
- 提供数据和消耗数据速度不一致时

5.2.4（3）：缓冲技术

- 缓冲区的实现
  - 可通过在主存建立缓冲区
  - 通常在内核开辟，数据在缓冲区缓冲后，在用户缓冲区和设备之间传输
- 缓冲区涉及大量复制操作，影响I/O性能

5.2.4（4）：设备分配

- 根据设备的物理特性，系统制订分配策略
  - 静态分配
  - 动态分配
  - 虚拟分配

5.2.4（4）：设备状态跟踪

- 系统检查相应设备的使用状态，来决定是否接受请求
- 一种简单的处理办法，用OPEN打开设备相应文件

5.2.4（5）：错误处理和报告

- I/O经常出错，多数错误是与硬件紧密相关，应尽可能在底层处理
- 通常驱动程序知道如何处理错误
- 如果处理不成功，报告给独立于设备的I/O软件

- I/O 软件的设计目标和原则
- I/O 中断处理程序
- I/O 设备驱动程序
- 独立于设备的I/O软件
- 用户空间的I/O软件

5.2.5 用户空间的I/O软件

1. 库例程
2. SPOOLing 软件（Simultaneous Peripheral Operation On-Line）
   - 外部设备联机并行操作
   - 于慢速字符设备如何与计算机主机交换信息

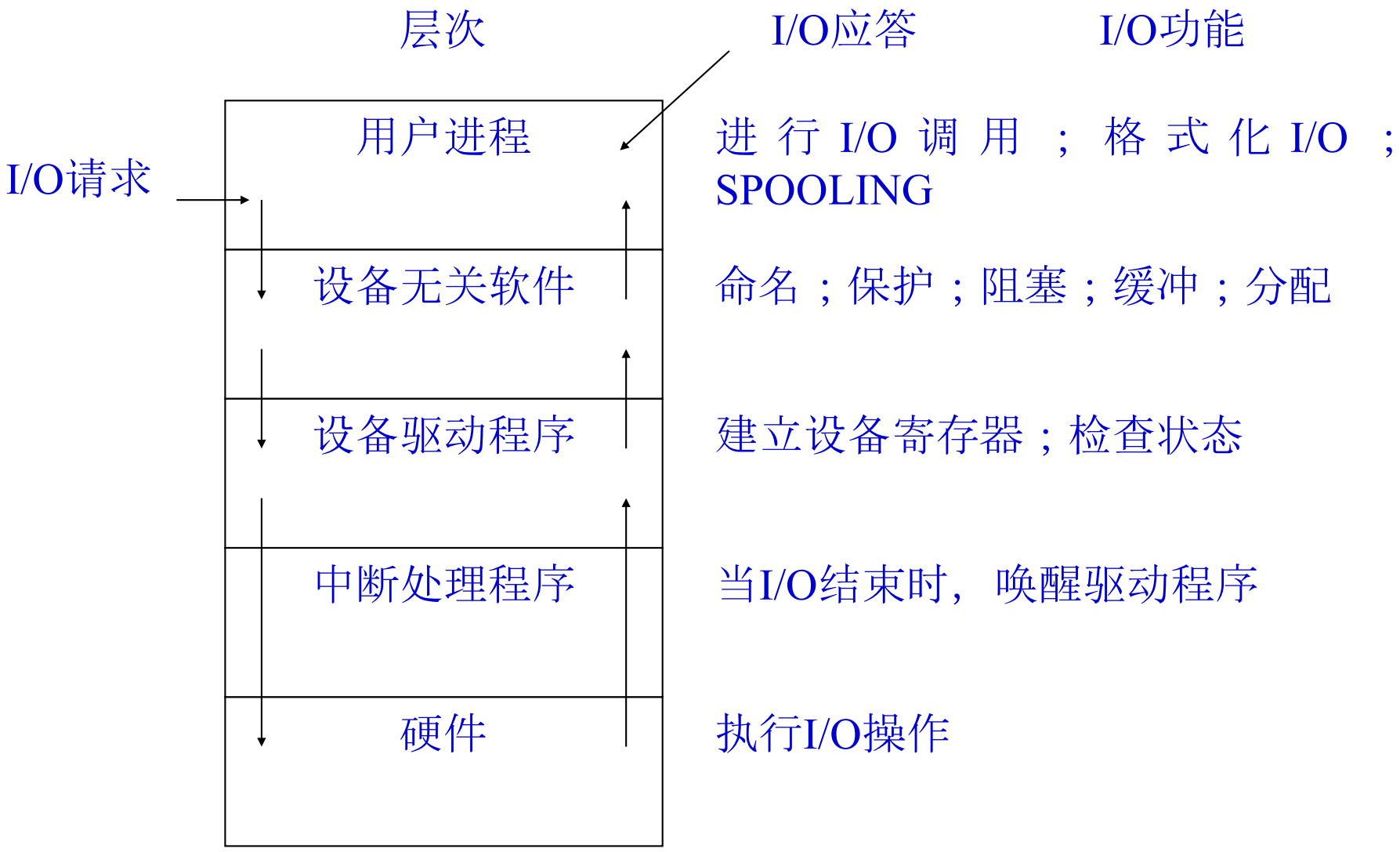
5.2.5（1） 库函数

- 大部分I/O软件属于操作系统,但是有一小部分是与用户程序链接在一起的库例程(库函数).
- I/O系统调用通常先是库例程调用
- 如C语言中的I/O函数：printf()、scanf()、read()、write()等

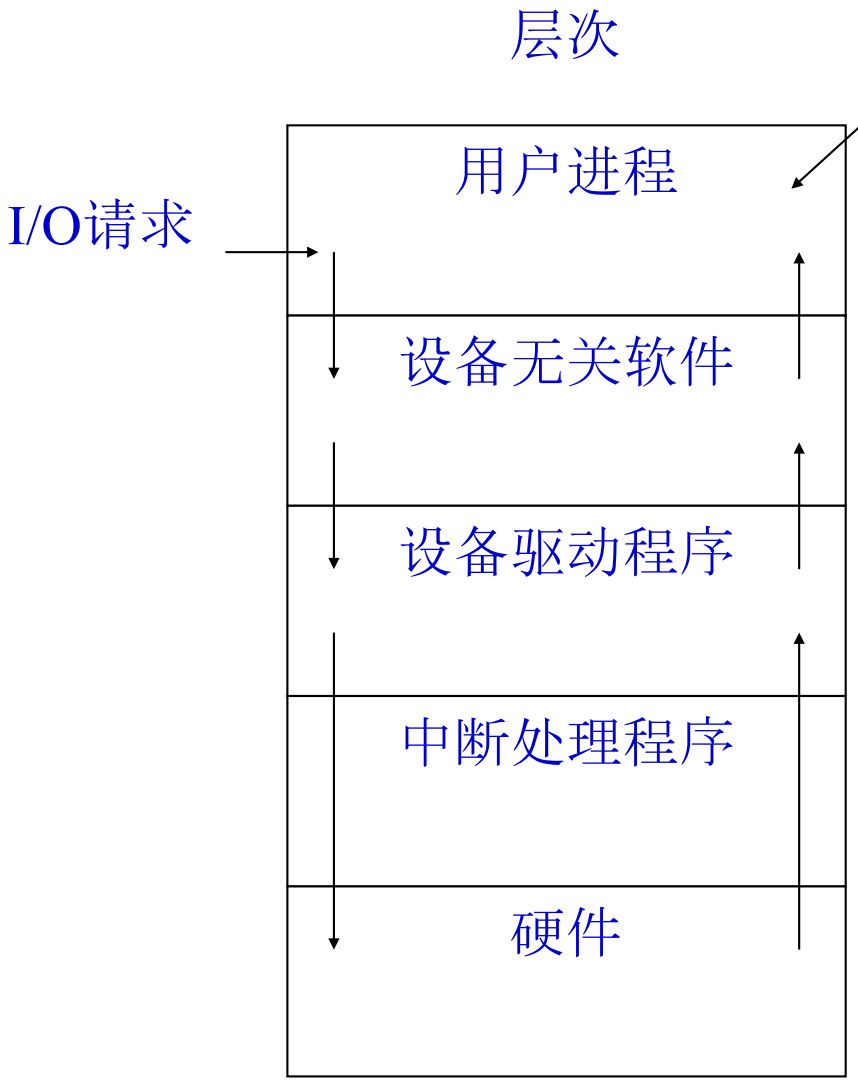
5.2.5（2）SPOOLing

- SPOOLing又称为假脱机操作，它是在多道程序环境下，利用其中一道程序来模拟脱机输入时的外围控制机的功能，把低速I/O设备上的数据传送到高速磁盘上；再利用另一道程序来模拟脱机输出时外围控制机的功能，把数据从磁盘传送到低速输出设备上。此时的外围操作与CPU对数据的处理同时进行。这种在联机的情况下实现的同时外围操作称为SPOOLing

5.2：小结


层次
I/O应答
I/O功能
I/O请求
用户进程
进行I/O调用；格式化I/O；SPOOLING
设备无关软件
命名；保护；阻塞；缓冲；分配
设备驱动程序
建立设备寄存器；检查状态
中断处理程序
当I/O结束时，唤醒驱动程序
硬件
执行I/O操作

5.2：小结

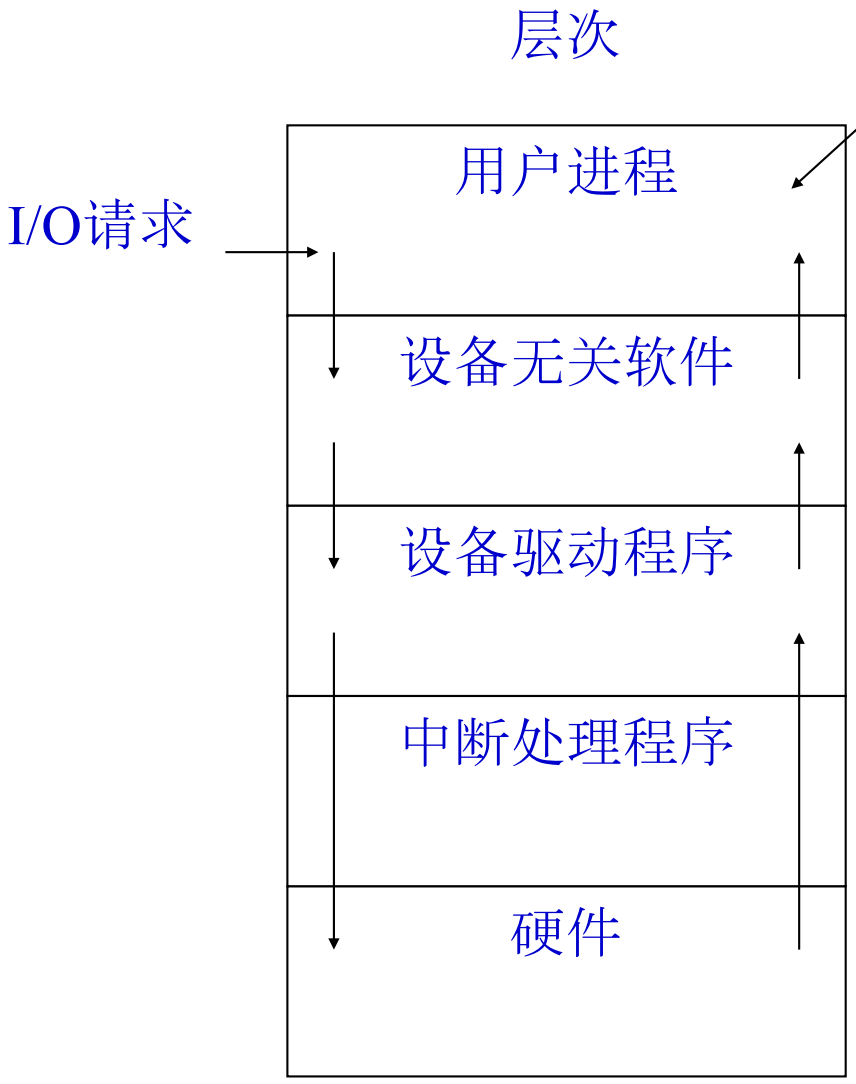


1）进程对已打开文件的文件描述符执行读库函数；

2）独立设备的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；

5.2：小结



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。

5.4 缓冲技术

- 单缓冲
- 双缓冲
- 多缓冲
- 缓冲区高速缓存

5.4：引入缓冲技术的目的

- 改善中央处理器与外围设备之间速度不匹配的矛盾
- 协调逻辑记录大小与物理记录大小不一致
- 提高CPU和I/O设备的并行性

5.3：缓冲技术实现基本思想

- 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
- 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

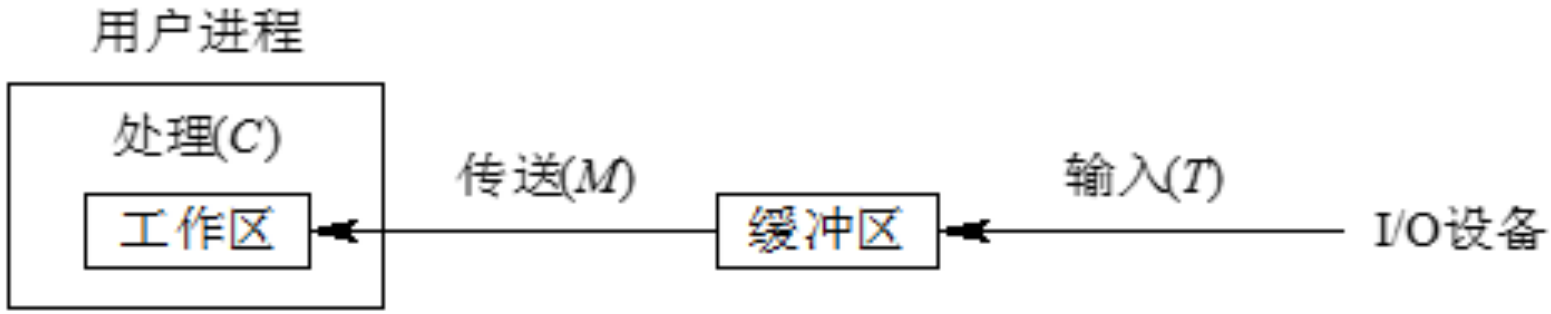
5.3：缓冲技术实现基本思想

- 进程执行读操作输入数据时，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
- 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

- 单缓冲
- 双缓冲
- 多缓冲

5.3.1 单缓冲

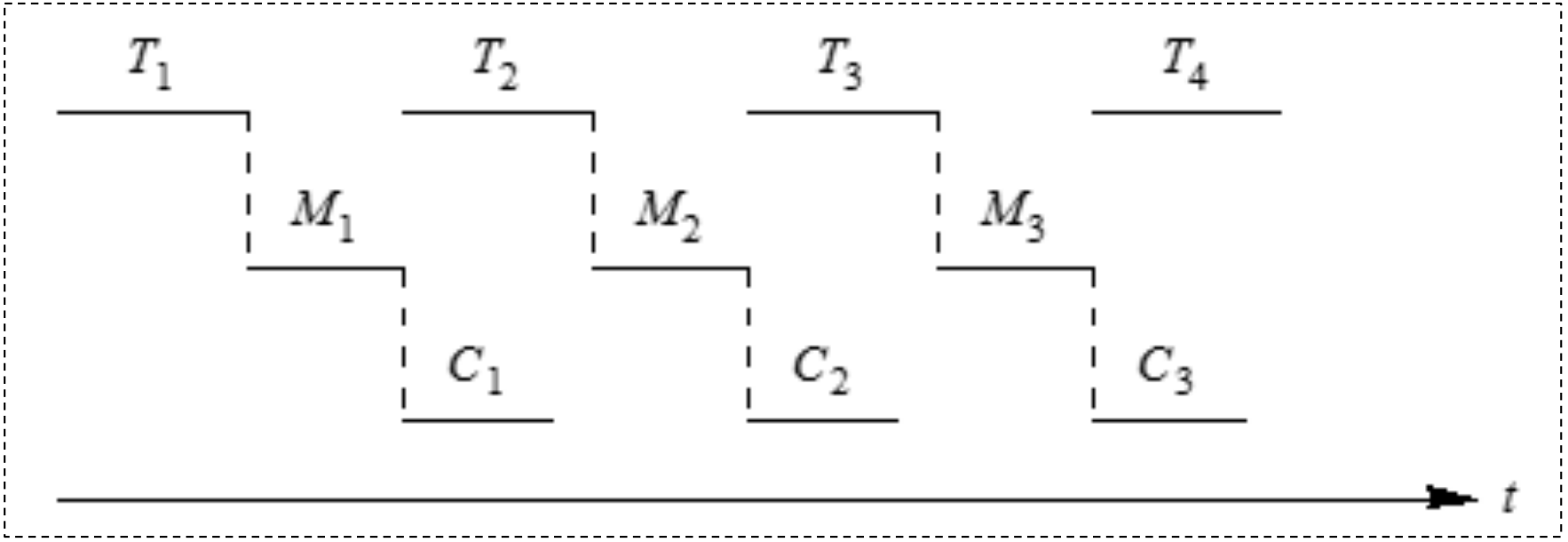
- 当用户进程发出请求时，OS在主存的系统区开设一个缓冲区。



设备	输入设备	内存	CPU
时间	T	M	C

5.3.1 单缓冲

- 块设备单缓冲机制的输入工作过程：
- 磁盘数据 → 缓冲区 → 用户区 → 处理数据
  - 设用时分别为T、M、C
  - 数据处理时间约为：max[C, T]+M
  - 不用缓冲时的处理时间为： T+C



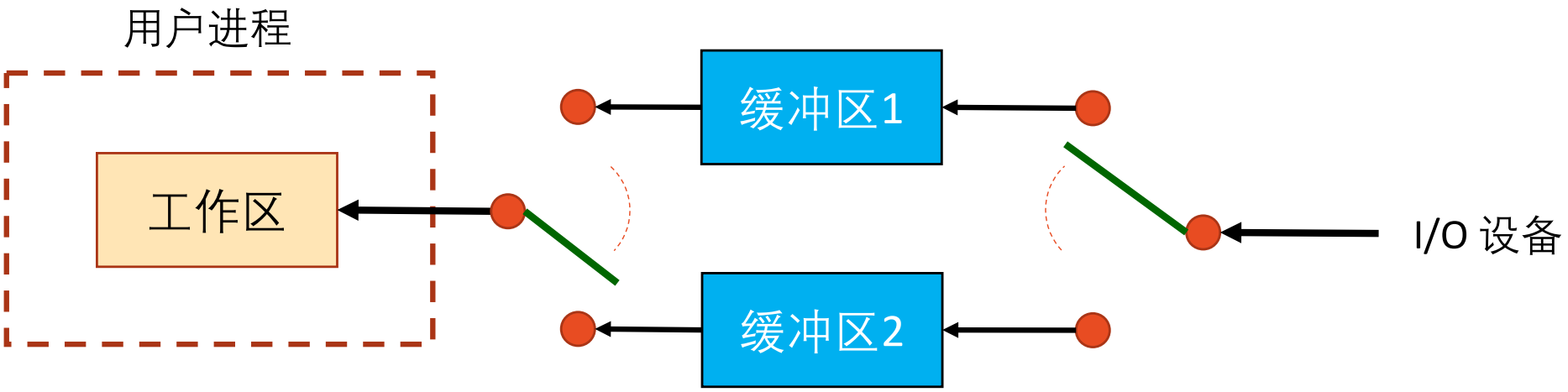
5.3.1 单缓冲

- 块设备单缓冲机制的输入工作过程：
- 磁盘数据 → 缓冲区 → 用户区 → 处理数据
  - 设用时分别为T、M、C
  - 数据处理时间约为：max[C, T]+M
  - 不用缓冲时的处理时间为： T+C
- 字符设备单缓冲机制的工作过程：
  - 同上，只是缓冲区小些

- 单缓冲
- 双缓冲
- 多缓冲

5.3.2 双缓冲

- 为了实现I/O的并行工作，引入了双缓冲
  - 磁盘数据 → 缓冲区1 → 用户区 → 处理数据
  - 磁盘数据 → 缓冲区2 → 用户区 → 处理数据

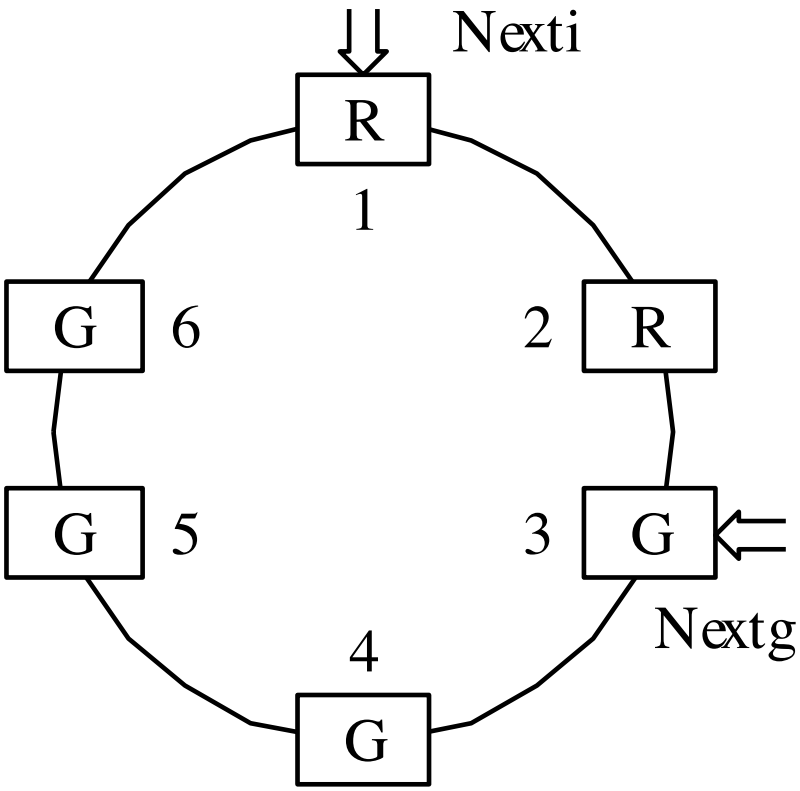


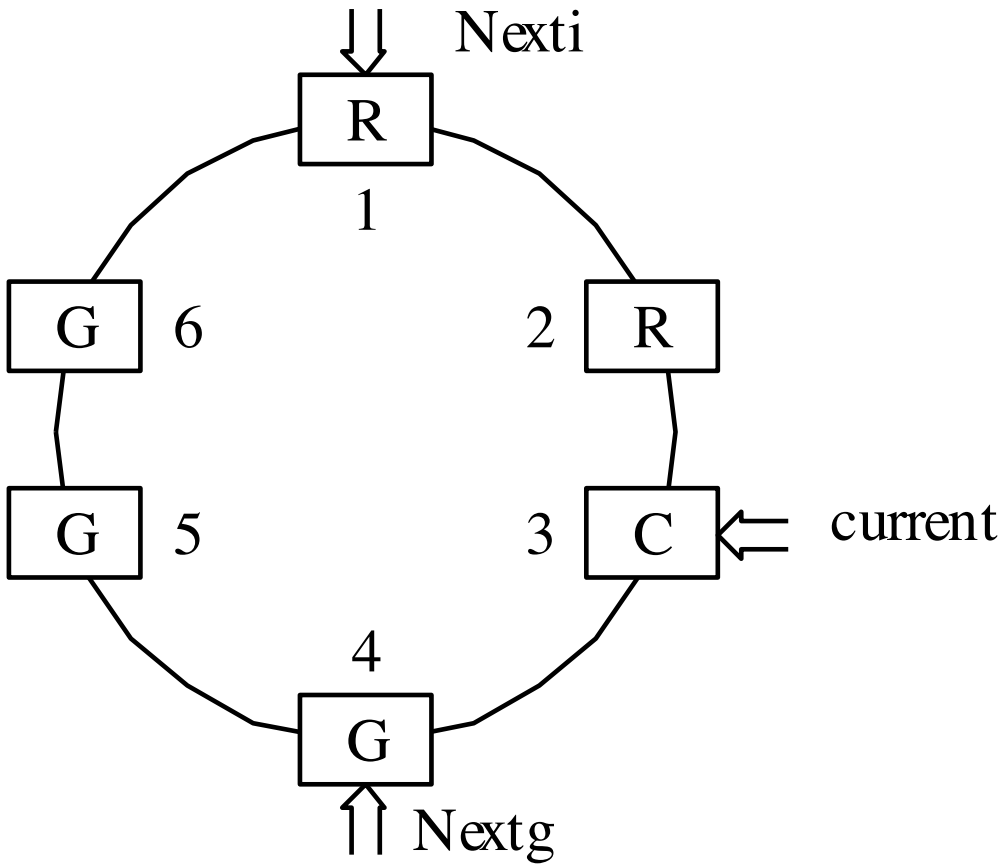
- 若C < T，则：所需时间为：T
- 若C > T，则：所需时间为：M + C

- 单缓冲
- 双缓冲
- 多缓冲

5.3.3 多缓冲

- 多缓冲：操作系统从自由主存区域中分配一组缓冲区组成循环缓冲(用链接指针)，每个缓冲区的大小等于物理记录的大小
- 使用对象：多缓冲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
- 用途分为：输入缓冲区，处理缓冲区和输出缓冲区
- 管理机制：缓冲区自动管理程序管理


Nexti
R
1
G
6
2
R
G
5
3
G
Nextg
4
G


Nexti
R
1
G
6
2
R
G
5
3
C
current
4
G
Nextg

循环缓冲

5.3.3 缓冲区高速缓存

- 为避免数据项的重复产生，内核建立了一个数据缓冲区高速缓存，专门用于保存最近使用过的磁盘数据块

5.3.3 缓冲区高速缓存

- 1）当请求从指定文件读写数据时，给定磁盘号和块号，查询是否在缓存中，如果在，则从对应的缓冲区获得数据；类似的，向磁盘写入数据时，也可以暂存缓存。
- 2）当文件关闭或撤销时，需要采用一定的策略解决释放缓冲区的重用的问题
- 3）提供一组高速缓存操作，如写缓存，为文件驱动程序实现读写文件数据。

5.4 驱动调度技术

- 存储设备的物理结构
- 循环排序
- 优化分布
- 交替地址
- 搜查定位
- 独立磁盘冗余阵列
- 提高磁盘I/O速度的方法

5.4 驱动调度技术

- 对于磁盘，同时会有若干个输入输出请求来到并等待处理，系统必须采用一种调度策略，能按最佳次序执行要求访问的各个请求，这就叫驱动调度，使用的算法叫驱动调度算法
- 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

5.4（续）

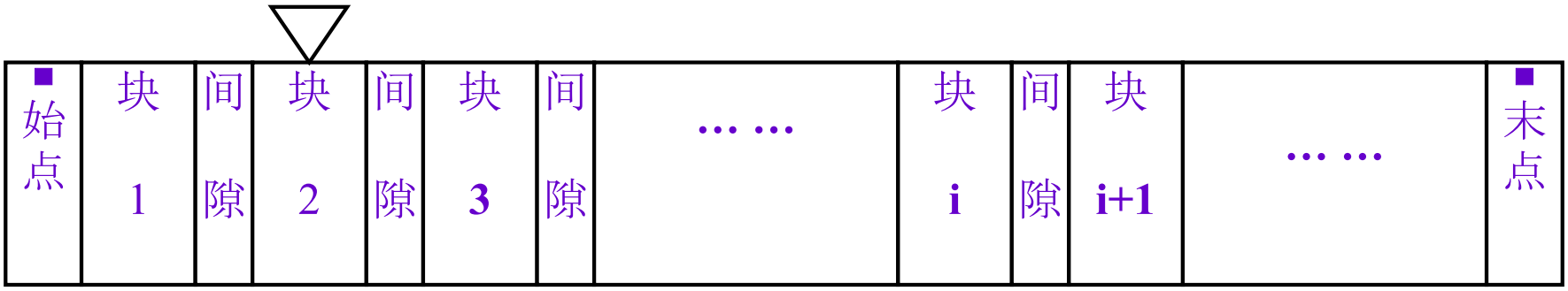
- 驱动调度（算法）：指磁盘调度（算法），调度磁盘的磁头臂移动和盘片转动
- 先进行移臂调度，再做转动调度
- 影响存取访问速度的因素：I/O请求的优化排序，信息在辅助存储器上的排列方式，存储空间分配方法等
  - 5.4.5(搜查定位)：有先来先服务、最短查找时间优先、扫描法、循环扫描法、分步扫描法、电梯调度算法等

- 存储设备的物理结构
- 循环排序
- 优化分布
- 交替地址
- 搜查定位
- 独立磁盘冗余阵列
- 提高磁盘I/O速度的方法

5.4.1 存储设备的物理结构

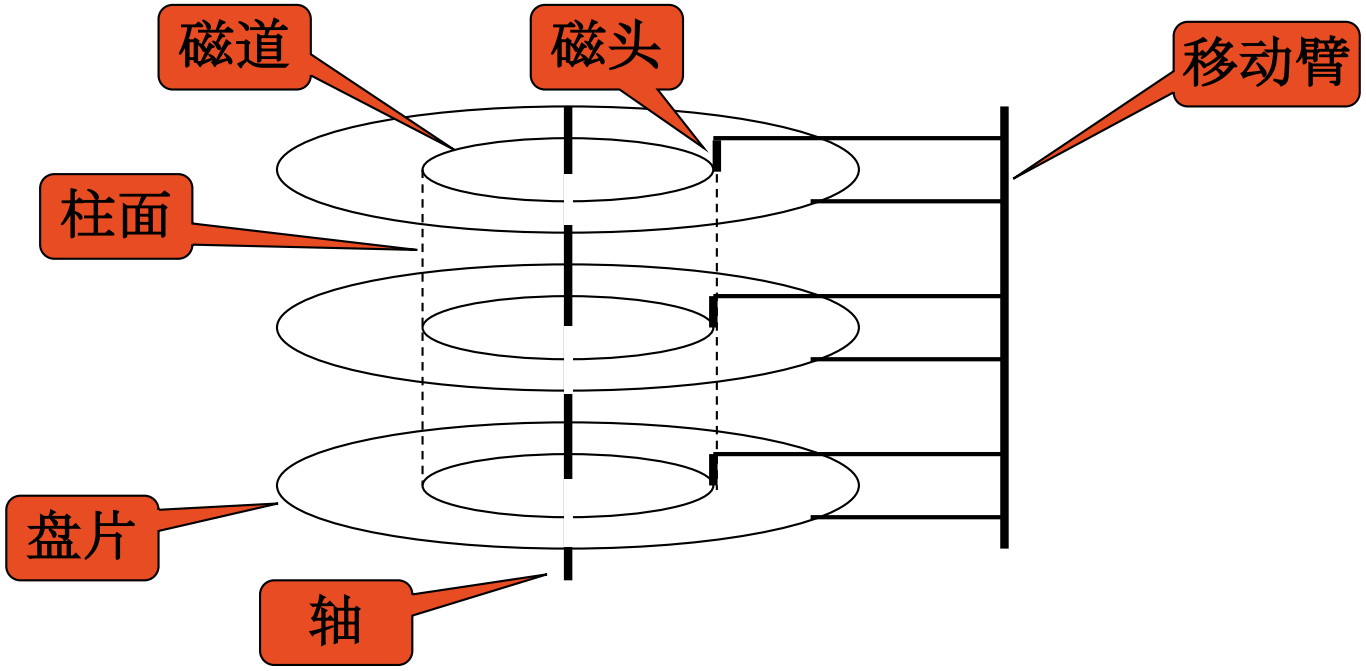
- 顺序存取存储设备是严格依赖信息的物理位置进行定位和读写的存储设备，如磁带，其特点是要读取靠后的信息，必须先读靠前的信息
  - 严格依赖信息的物理位置进行定位和读写的存储设备
  - 具有存储容量大、稳定可靠、卷可装卸和便于保存等优点

磁头(正走,反走,正读,反读,正写,反写,倒带)



5.4.1（续）

- 磁盘是一种直接(随机)存取存储设备。每个物理记录有确定的位置和唯一的地址，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



5.4.1：直接存取存储设备

- 磁盘记录参数：
  - 柱面号：与磁道号等价
  - 磁头号：一般为0
  - 块号：扇区号，磁盘中的最小物理单位

簇：操作系统进行文件存储管理的单位，由一个或多个扇区组成，是系统的逻辑概念。

5.4.1：直接存取存储设备

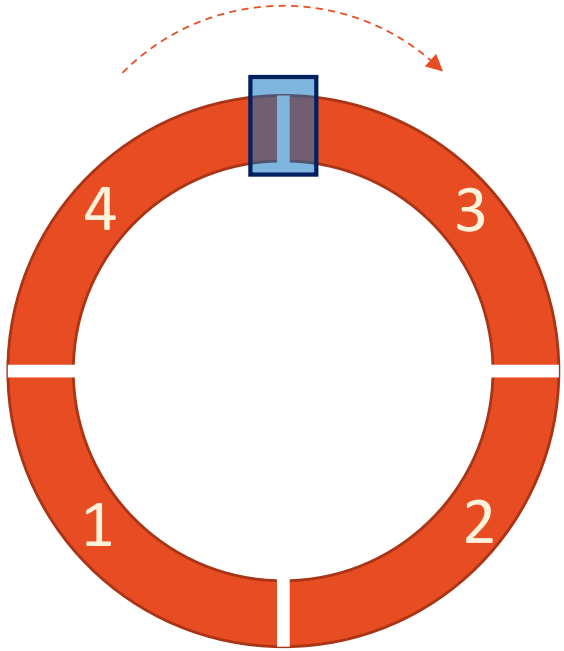
- 磁盘访问操作过程：根据信息的参数，移动磁头、搜索数据块、对数据块进行读写
- 查找时间：磁盘根据给出的柱面号，控制臂横向移动，使读写磁头到达指定柱面所需时间，即移动磁头时间。平均约20ms
- 搜索延迟：被访问信息块旋转到读写磁头下所需要的时间。平均约10ms
- 传送时间：是读写磁头对数据块读写所需要的时间。平均约10ms

- 存储设备的物理结构
- 循环排序
- 优化分布
- 交替地址
- 搜查定位
- 独立磁盘冗余阵列
- 提高磁盘I/O速度的方法

5.4.2 循环排序

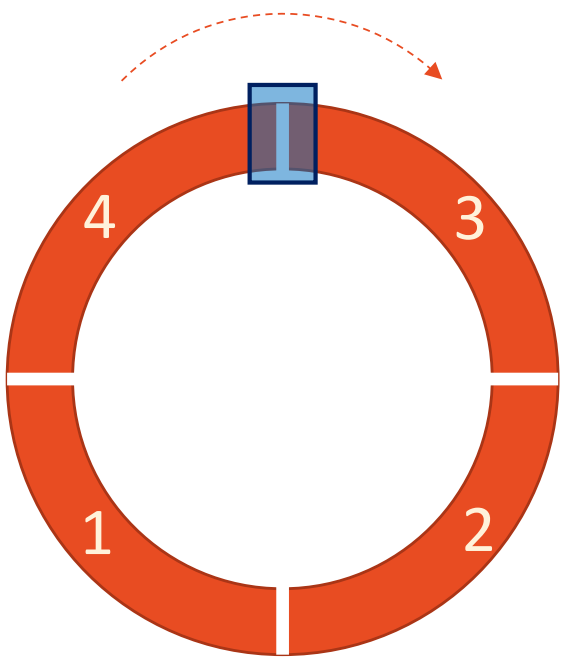
- 考虑在某磁道上保存有4个记录，盘片旋转一周耗时20ms，假定收到四个I/O请求：

请求次序	记录号
1	读取记录4
2	读取记录3
3	读取记录2
4	读取记录1

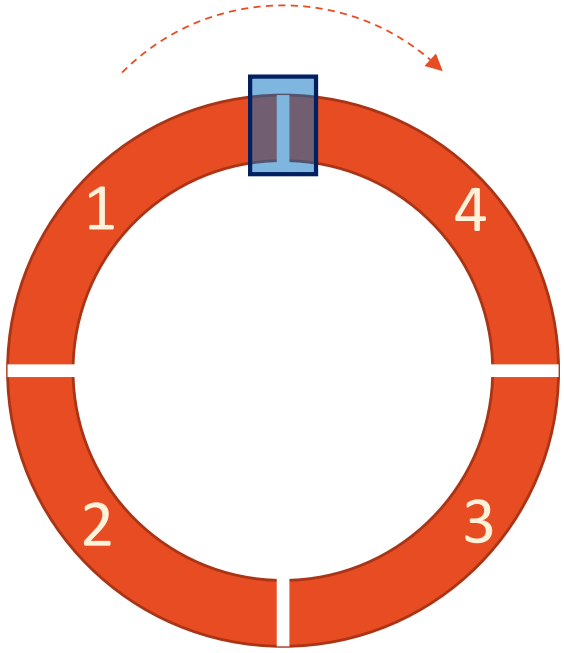


5.4.2 循环排序

- **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1/2+1/4+3*3/4=3周，即60毫秒
- 读取记录4：平均旋转时间1/2 ＋ 读取时间 1/4 周
  - 此时磁盘已旋转到记录1



t

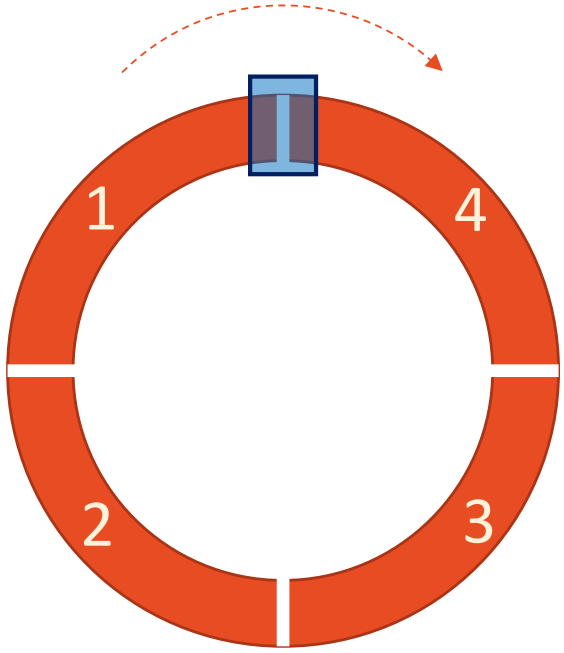


t + 1

5.4.2 循环排序

- **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1/2+1/4+3*3/4=3周，即60毫秒
- 读取记录4：平均旋转时间1/2 ＋ 读取时间 1/4 周
  - 此时磁盘已旋转到记录1

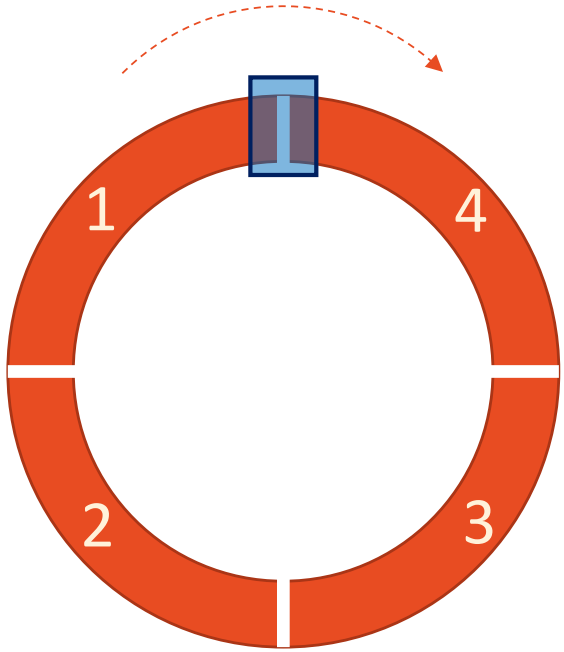
读取记录3：旋转2/4周定位
＋ 再旋转1/4周读取
读取记录2：旋转2/4周定位
＋ 再旋转1/4周读取
读取记录1：旋转2/4周定位
＋ 再旋转1/4周读取



t + 1

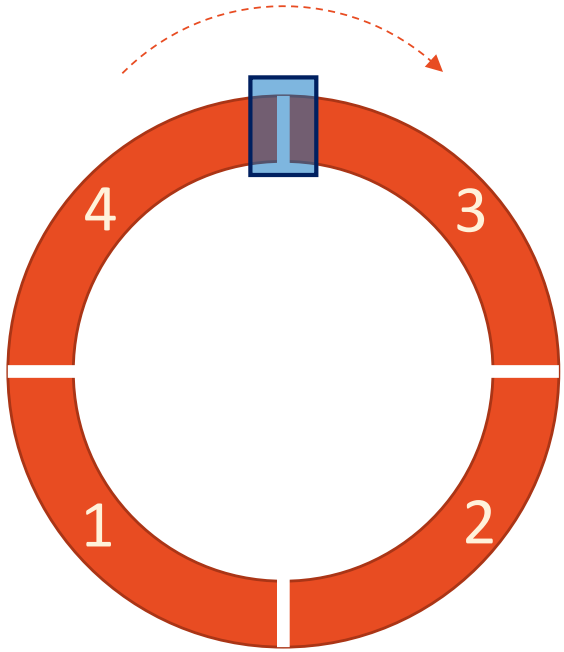
5.4.2 循环排序

- **方法2：**按照**数据存储方向**读记录1、2、3、4，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1/2+1/4+3*1/4=1.5周，即30毫秒
- 读取记录1：平均旋转时间1/2 ＋ 读取时间 1/4 周
  - 此时磁盘已旋转到记录2
- 读取记录2：旋转1/4周读取
- 读取记录3：旋转1/4周读取
- 读取记录4：旋转1/4周读取



5.4.2 循环排序

- **方法3：**如果可以知道当前读位置是记录3，则可采用次序为读记录4、1、2、3。总处理时间等于 1 周，即20毫秒
- 读取记录4：读取时间 1/4 周
  - 此时磁盘已旋转到记录1
- 读取记录1：旋转1/4周读取
- 读取记录2：旋转1/4周读取
- 读取记录3：旋转1/4周读取

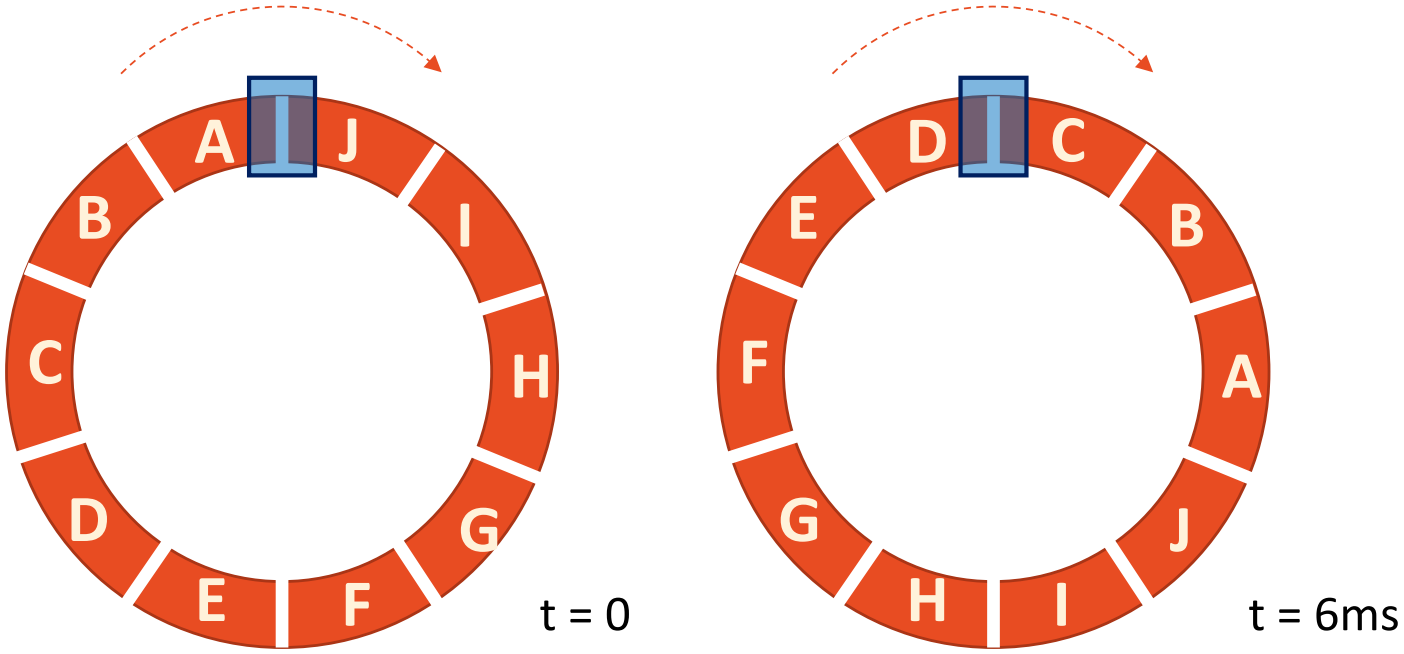


- 存储设备的物理结构
- 循环排序
- 优化分布
- 交替地址
- 搜查定位
- 独立磁盘冗余阵列
- 提高磁盘I/O速度的方法

5.4.3 优化分布

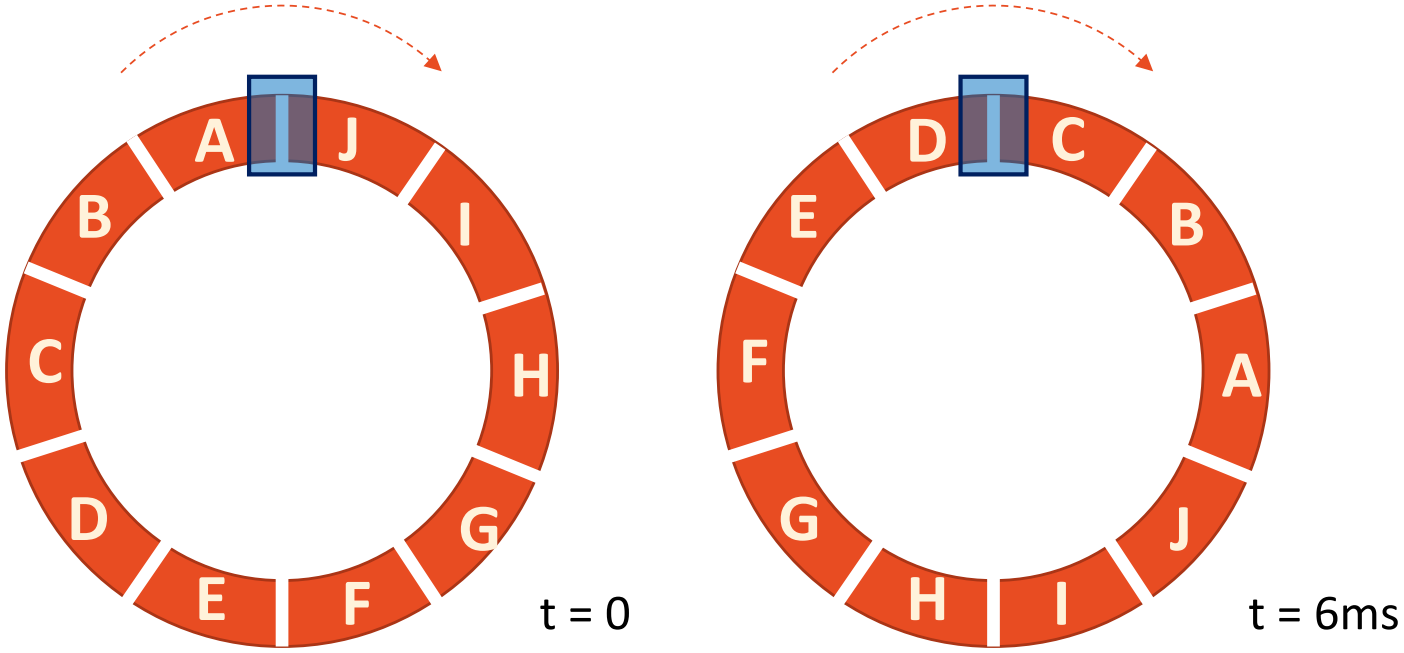
- 信息在存储空间的排列方式会影响存取等待时间。考虑10个逻辑记录A，B……，J被存于旋转型设备上，每道存放10个记录。
- 假定旋转速度为20ms/周，处理程序读出每个记录后花4ms进行处理.
- 需经常顺序读取记录进行处理

5.4.3 优化分布



1.读出记录A需要旋转20/10=2ms,
2.处理完记录A之后磁盘又旋转了B、C两个扇区
- 因为处理每个记录需4ms时间
- 因此将转到记录D的开始

5.4.3 优化分布



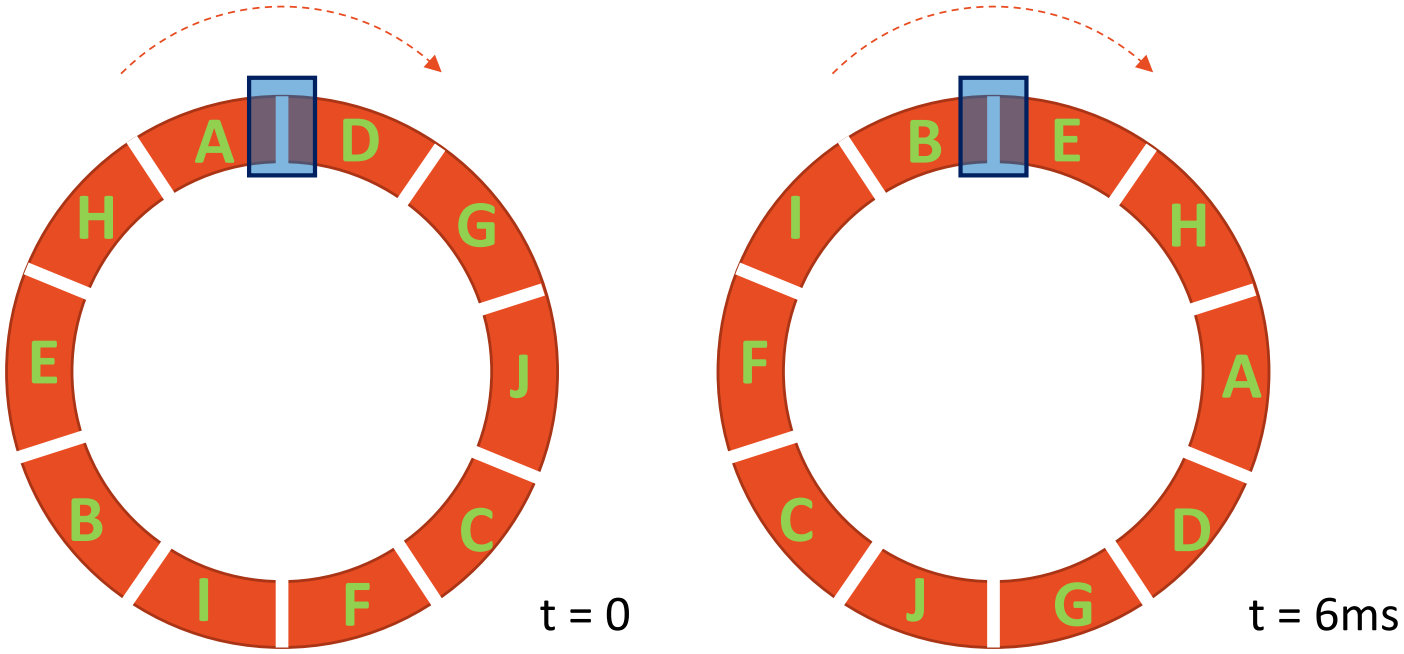
总处理时间：

10ms + 2ms + 4ms // 读取A需要的时间

+ 9* (16ms + 2ms + 4ms) // 读取其它记录需要的时间

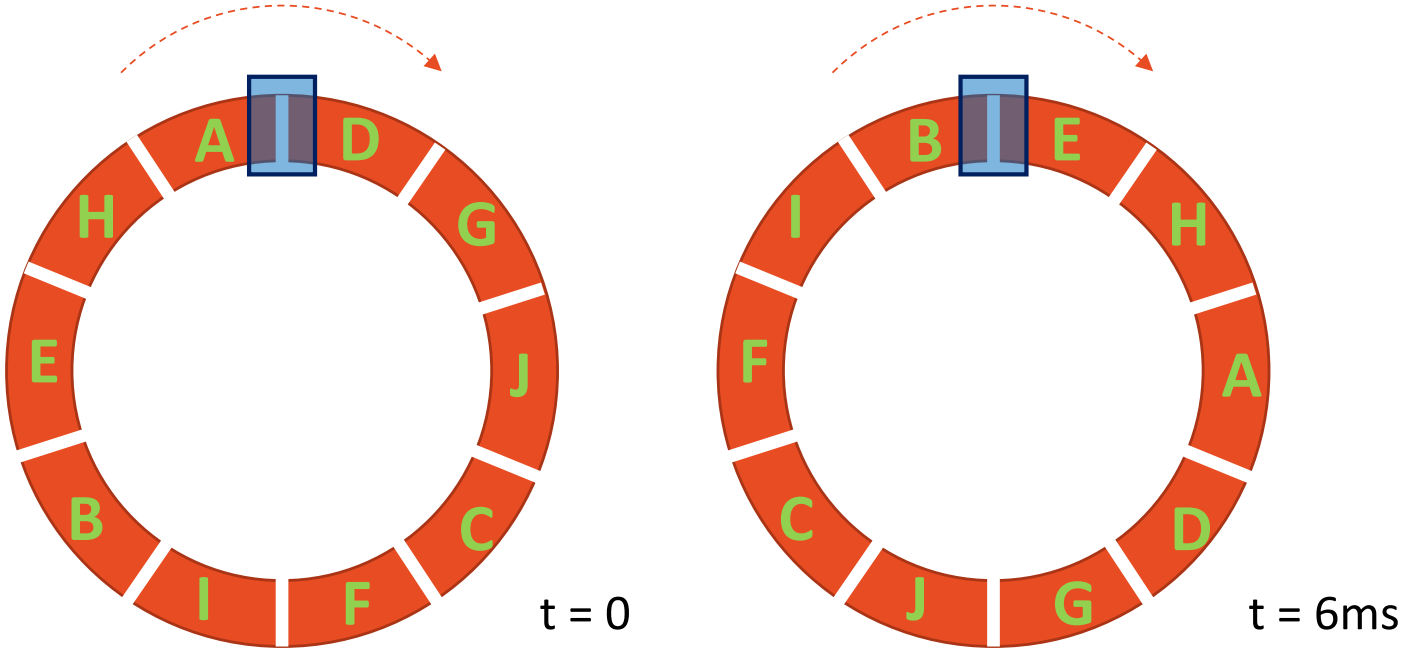
= 214ms

5.4.3 优化分布



读出记录A并处理结束后,恰巧转至记录B的位置可以立即就可读出并处理.

5.4.3 优化分布



总处理时间：

10ms + 2ms + 4ms // 读取A需要的时间

+ 9* (2ms + 4ms) // 读取其它记录需要的时间

= 70ms

- 存储设备的物理结构
- 循环排序
- 优化分布
- 交替地址
- 搜查定位
- 独立磁盘冗余阵列
- 提高磁盘I/O速度的方法

5.4.* 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

- 存储设备的物理结构
- 循环排序
- 优化分布
- 交替地址
- 搜查定位
- 独立磁盘冗余阵列
- 提高磁盘I/O速度的方法

5.4.4 搜查定位

- 除旋转延迟外，还存在搜查寻道延迟
- I/O请求需要3个参数：柱面号，磁道号和物理块号
- 如以下的5个磁盘访问请求：

柱面号	磁道号	物理块号
7	4	1
7	4	8
7	4	5
40	6	4
2	7	7

对于磁盘设备，应考虑使移动臂的移动时间最短的一些调度策略

5.4.4：移臂调度策略

1. “先来先服务”算法
2. “最短寻道（查找）时间优先”算法
3. “电梯调度”算法
4. “循环扫描”算法
5. “分步扫描”算法

5.4.4（1）：先来先服务

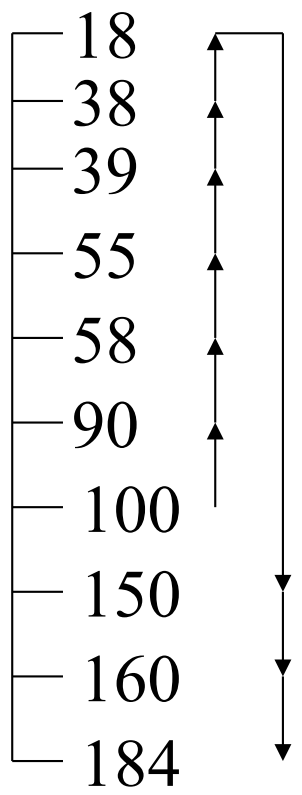
- 随机移动
- 例：假定磁头处于100号磁道
- 寻道时间周期长，性能较差

（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	

5.4.4（2）：最短查找时间

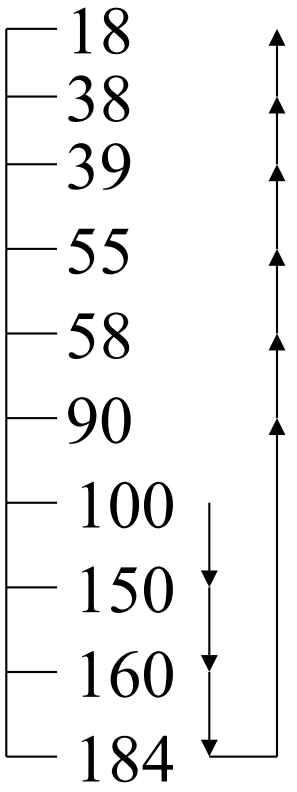
- 优先满足距离磁头最近的访问请求。
- 例：假定磁头处于100号磁道
- 某一个进程的请求长期得不到满足，产生饥饿

（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	



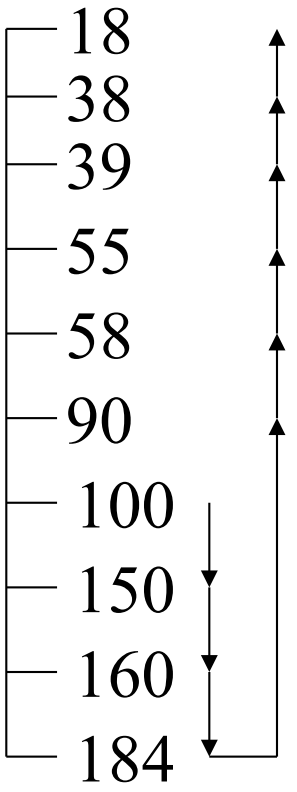
5.4.4（3）：电梯调度

- 选择沿臂的移动方向最近的柱面，如果同一柱面上有多个请求，还需进行旋转优化
- 如果这个方向没有访问请求时，就改变臂的移动方向，处理所遇到的最近的I/O请求，非常类似于电梯的调度规则



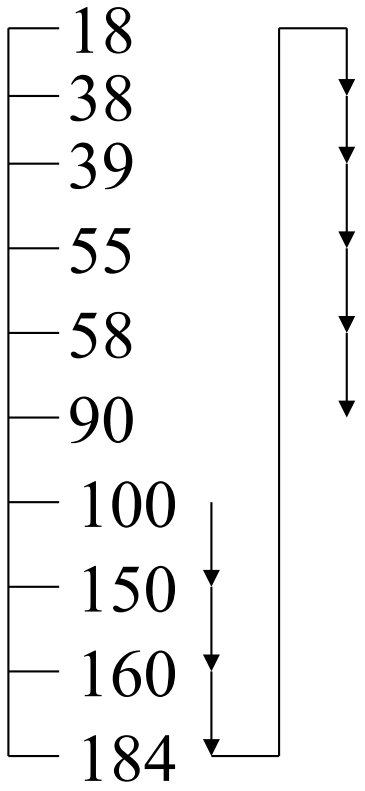
5.4.4（3）：电梯调度

- 电梯算法存在一种现象：当磁头刚从里向外越过某个磁道时，恰好又有一个进程请求访问该磁道时，必须等待，磁头移到最外层然后再返回来处理了其他磁道后才能处理它，导致该进程的请求被长时间延迟



5.4.4（4）：循环扫描

- 磁头单向移动，例如规定磁头自里向外移动时才访问磁道，磁头移动到最外磁道访问后，立即返回到最里的欲访问磁道



5.4.4（5）：分步扫描

- 将磁盘请求队列分成若干个长度为N的子队列，磁盘调度按先来先服务算法依次处理这些队列。当正在处理某子队列时，如果又出现新的磁盘I/O请求，便将新请求进程放入其它队列。每个队列按电梯算法处理

- N很大时性能接近电梯算法
- N很小时蜕化为FCFS算法

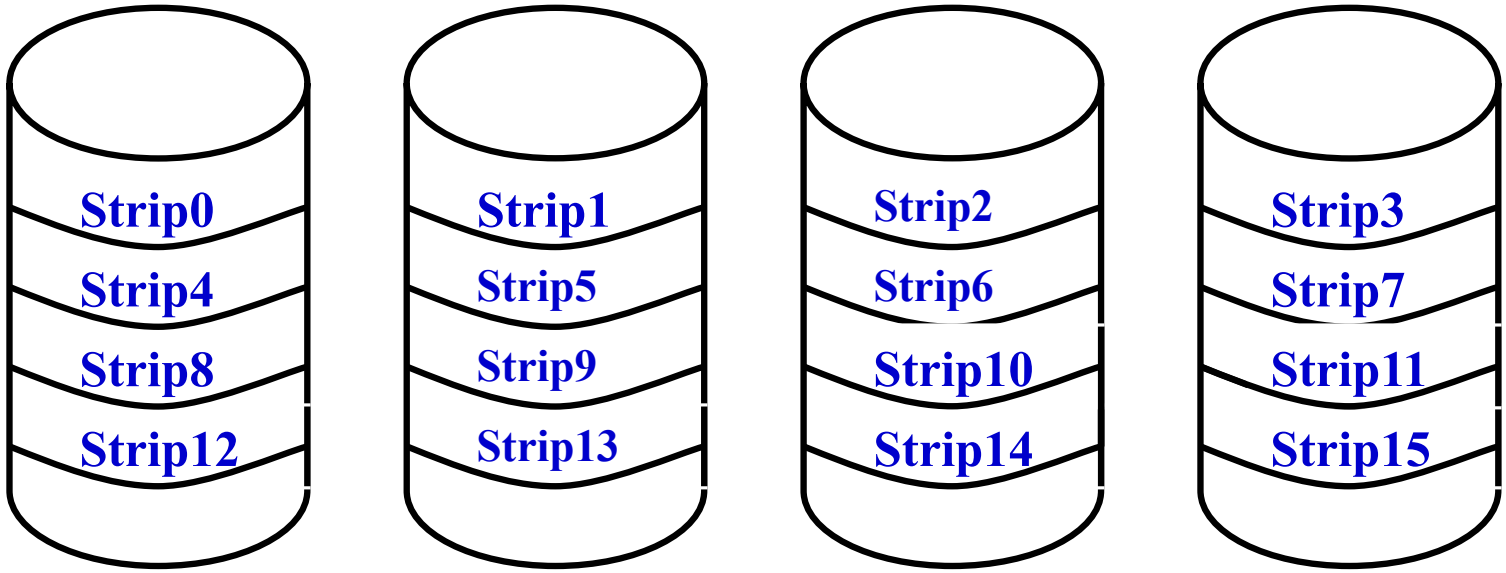
- 存储设备的物理结构
- 循环排序
- 优化分布
- 交替地址
- 搜查定位
- 独立磁盘冗余阵列
- 提高磁盘I/O速度的方法

5.4.5 独立磁盘冗余阵列

- 目的：通过在不同磁盘上维护冗余数据来增加容错性，把数据分布到小的独立单元，可并行工作存取以提高性能
- 策略：用一组容量小、独立的、可并行工作的磁盘驱动器组成阵列来代替单一的大容量磁盘，再增加冗余技术，数据分布存储，单个I/O请求能并行地从多个磁盘驱动器同时存取数据
- 组织方式：各种类型的RAID，主要差别在于冗余信息数量和容错级别，及冗余信息是集中在一个磁盘还是分散在多个磁盘上

5.4.5：RAID 0

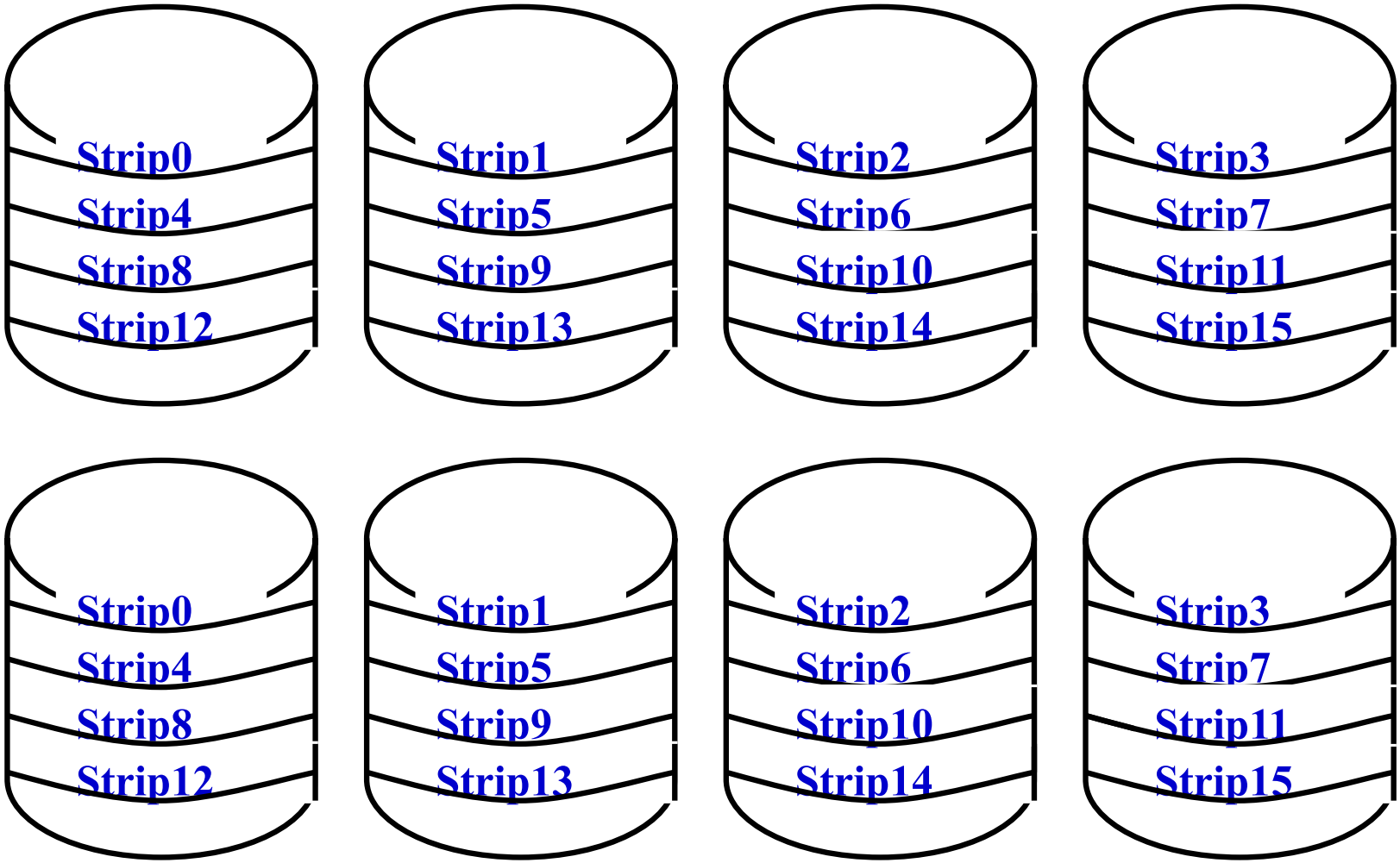
- 数据划成条块被分布存储在横跨阵列中的所有磁盘上
- 逻辑上连续的数据条块，在物理上可被依次存储在横向相邻的磁盘驱动器上
- 通过阵列管理软件进行逻辑地址空间到物理地址空间的映射
- 不存储校验信息
  - 具有最好的读写性能和最低的成本
  - 磁盘的利用率是100%
  - 安全性最低


Strip0
Strip4
Strip8
Strip12
Strip1
Strip5
Strip9
Strip13
Strip2
Strip6
Strip10
Strip14
Strip3
Strip7
Strip11
Strip15

RAID0 无冗余磁盘

5.4.5：RAID 1

- 双份数据，每个盘都有一个包含相同数据的镜像盘
- ①读请求能通过包含相同请求数据中的任何一个磁盘提供服务，其中的一个所花查找和搜索时间最少
- ②写操作时，要求改写对应的两个数据子块，可采用并行操作，写操作的性能由并行操作中较慢的一个决定
- ③一个驱动器出现故障，数据可以从镜像盘获得
- 最好的安全性、最低的利用率

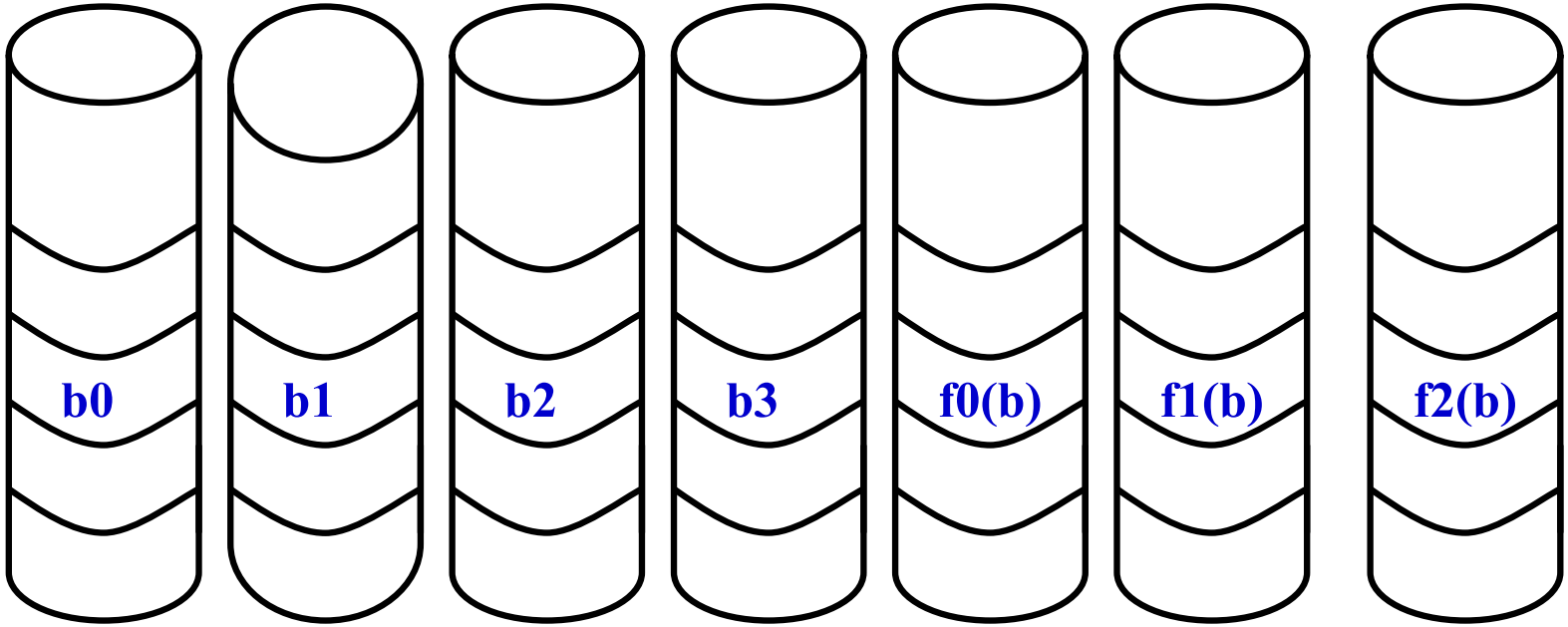


RAID Level 1 (Mirrored)

5.4.5：RAID 2

- 利用 Hamming Code 对数据进行编码并分区为独立的比特，并将编码分别写入不同的磁盘中。
- Hamming Code不仅可以检错，还可以纠错。
- 但是RAID 2方案的数据整体容量大，至少需要3台以上的磁盘。

5.4.5: RAID 2

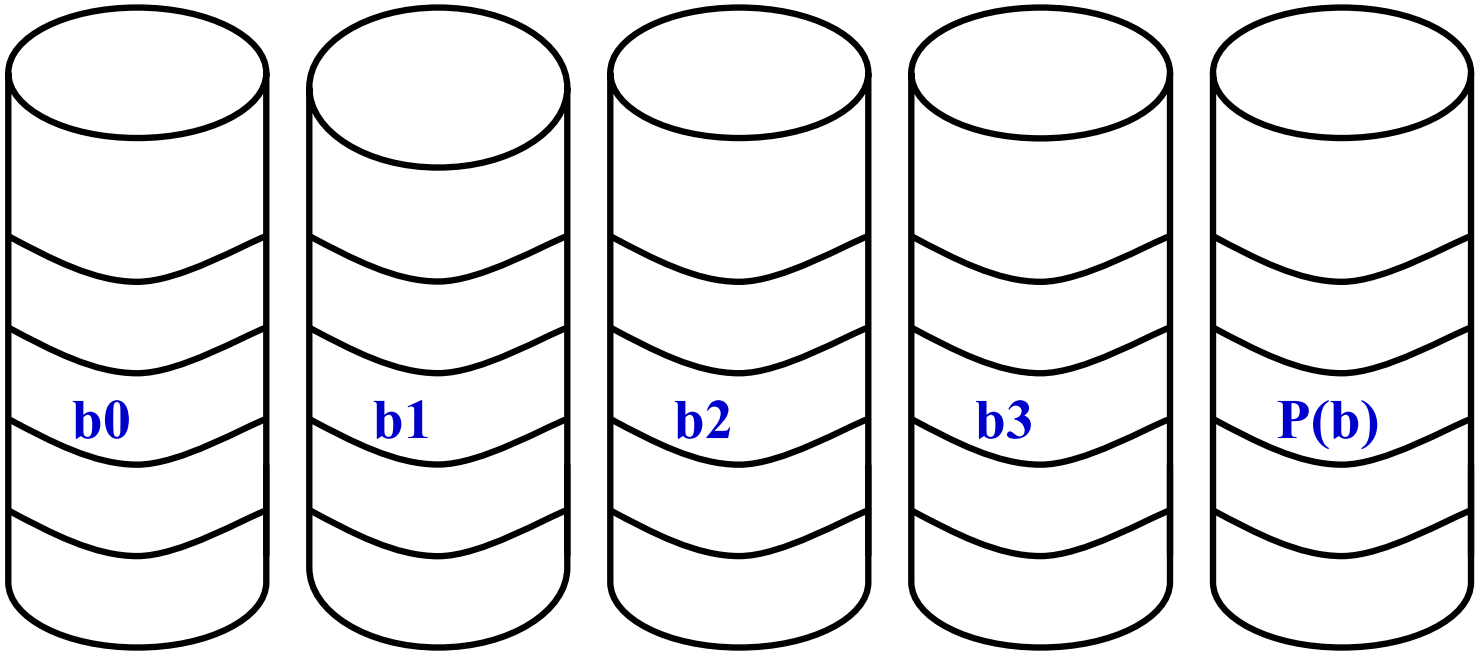


RAID Level 2 (Redundancy through Hamming Code)

5.4.5：RAID 3

- 采用Bit－interleaving（数据交错存储）技术，通过编码再将数据比特分割后分别存在硬盘中。
- 比特检查结果存储于同一个硬盘中。
- 数据比特分散在不同的硬盘上，取一小段数据都可能需要所有的硬盘工作。

5.4.5：RAID 3

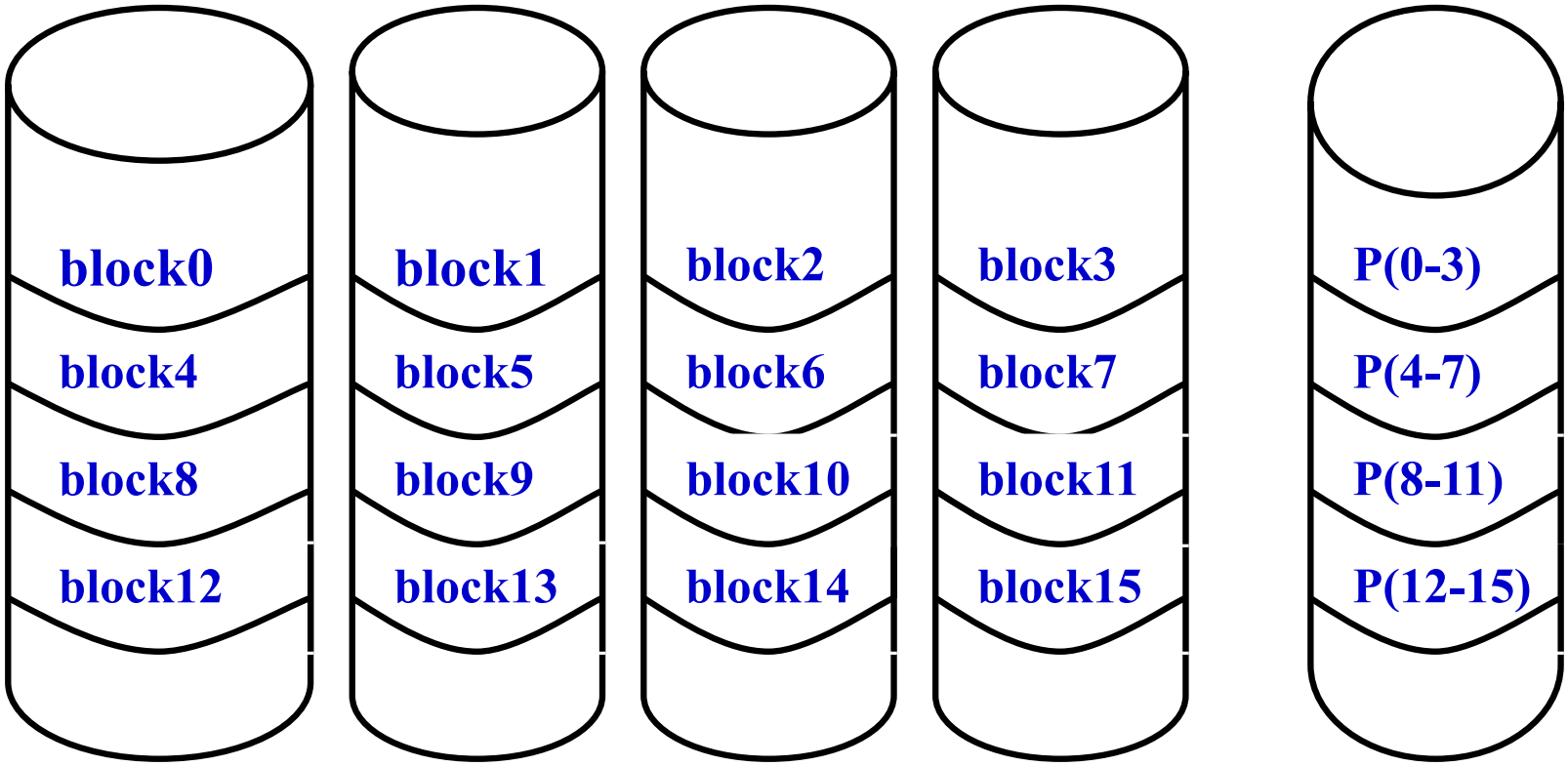


RAID Level 3 (Bit interleaved Parity)

5.4.5：RAID 4

- 采用区块交错存储（Block interleaving），在分区时以区块为单位存储在多个磁盘中。
- 每次的数据访问都必须从同比特检查的那个硬盘中取出对应的同比特数据进行核对，提高了检查磁盘的使用频率，导致易损坏。

5.4.5: RAID 4

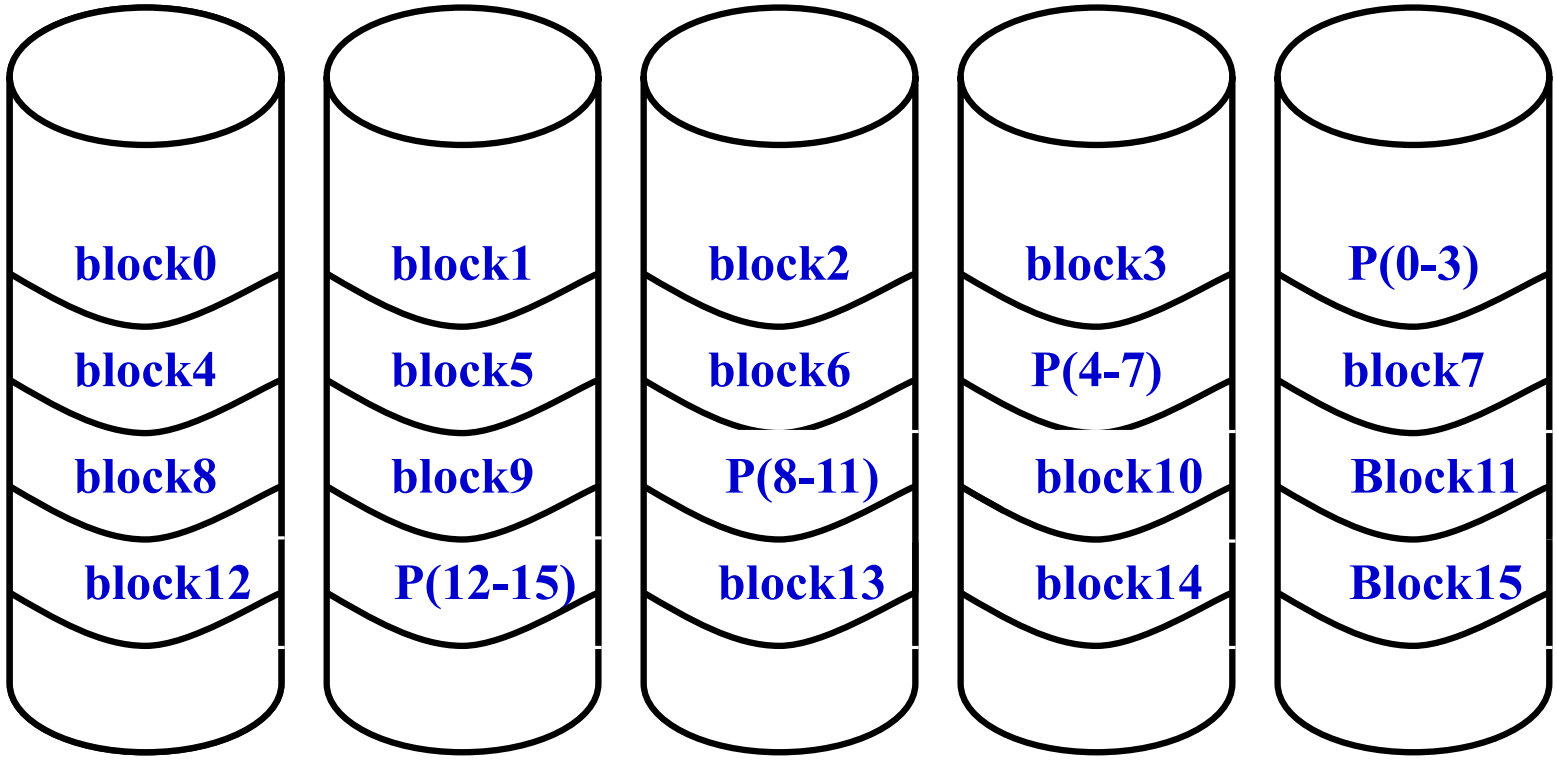


RAID Level 4 (Block level Parity)

5.4.5：RAID 5

- RAID 5 兼顾存储性能、数据安全和存储成本，使用度广。
- 将数据和相对应的奇偶校验信息存储到各个磁盘上。一旦发生磁盘损坏，可以利用剩下的数据和奇偶校验信息进行恢复。
- 至少需要3块磁盘

5.4.5: RAID 5



RAID Level 5 (Block level Distributed parity)

- 存储设备的物理结构
- 循环排序
- 优化分布
- 交替地址
- 搜查定位
- 独立磁盘冗余阵列
- 提高磁盘I/O速度的方法

5.4.6 提高磁盘I/O速度的方法

- 提前读：在读当前块的同时，将下一个盘块中的数据也读入缓冲区
- 延迟写：本应写回磁盘的缓冲区中的数据不久之后可能还会再被访问，因而不立即将其写回磁盘
- 虚拟盘：利用内存空间仿真磁盘，又称为RAM盘。虚拟盘中的数据在掉电或系统重启动以及发生故障时会丢失

作业

- 应用题：5，10，15，16