操作系统

Operating Systems

三种常见设备接口类型

- 字符设备
- 块设备
- 网络设备

三种常见设备接口类型

- 字符设备
  - 如: 键盘/鼠标, 串口等
- 块设备
- 网络设备

三种常见设备接口类型

- 字符设备
  - 如: 键盘/鼠标, 串口等
- 块设备
  - 如: 磁盘驱动器、磁带驱动器、光驱等
- 网络设备
  - 如: 以太网、无线、蓝牙等

设备访问特征

- 字符设备
- 块设备
- 网络设备

设备访问特征

- 字符设备
  - 访问特征
    - 以字节为单位顺序访问
  - I/O命令
    - get()、put()等
    - 通常使用文件访问接口和语义
- 块设备
- 网络设备

设备访问特征

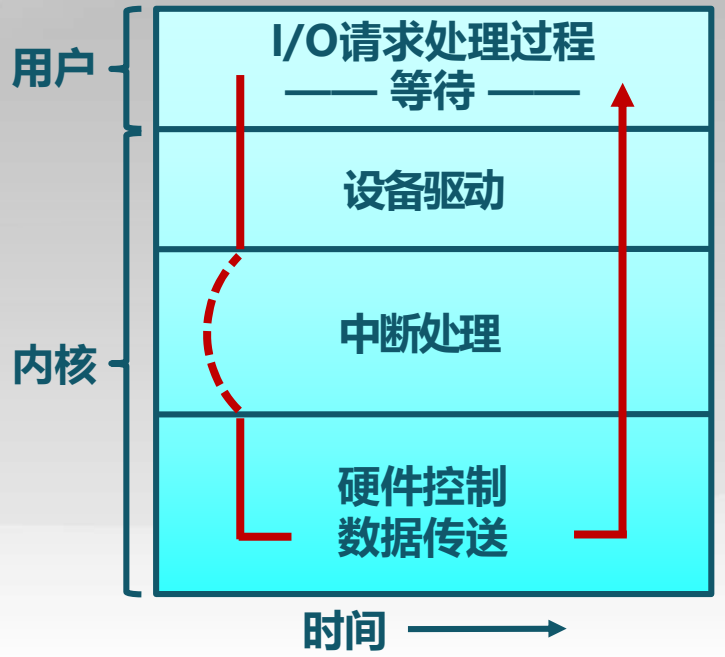
- 字符设备
- 块设备
  - 访问特征
    - 均匀的数据块访问
  - I/O命令
    - 原始I/O或文件系统接口
    - 内存映射文件访问
- 网络设备

设备访问特征

- 字符设备
- 块设备
- 网络设备
  - 访问特征
    - 格式化报文交换
  - I/O命令
    - send/receive 网络报文
    - 通过网络接口支持多种网络协议

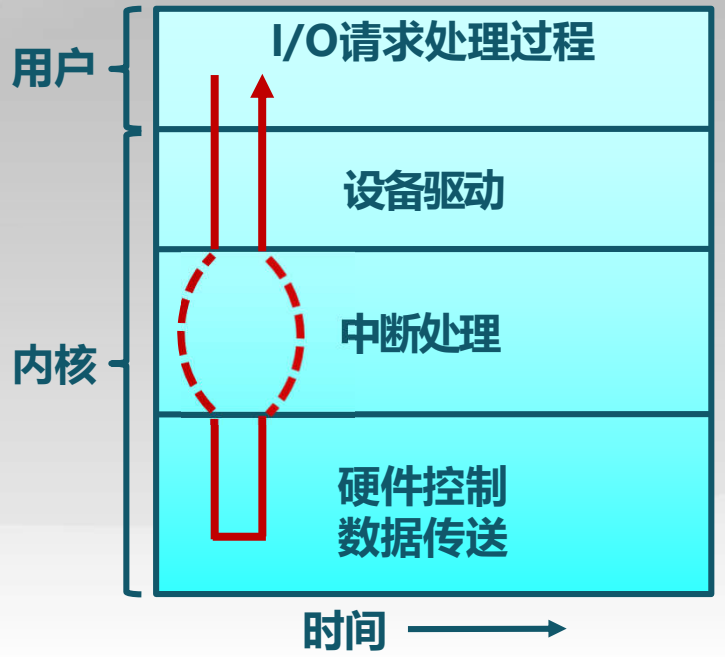
同步与异步I/O

- 阻塞I/O: “Wait”
  - 读数据(read)时，进程将进入等待状态，直到完成数据读出
  - 写数据(write)时，进程将进入等待状态，直到设备完成数据写入处理



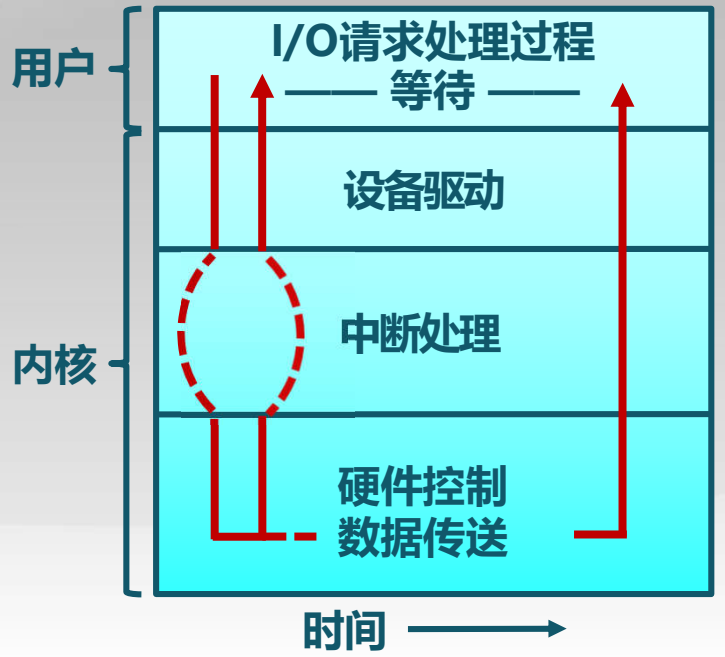
同步与异步I/O

- 非阻塞I/O: “Don’ t Wait”
  - 立即从read或write系统调用返回，返回值为成功传输字节数
  - read或write的传输字节数可能为零



同步与异步I/O

- 异步I/O: “Tell Me Later”
  - 读数据时，使用指针标记好用户缓冲区，立即返回；稍后内核将填充缓冲区并通知用户
  - 写数据时，使用指针标记好用户缓冲区，立即返回；稍后内核将处理数据并通知用户



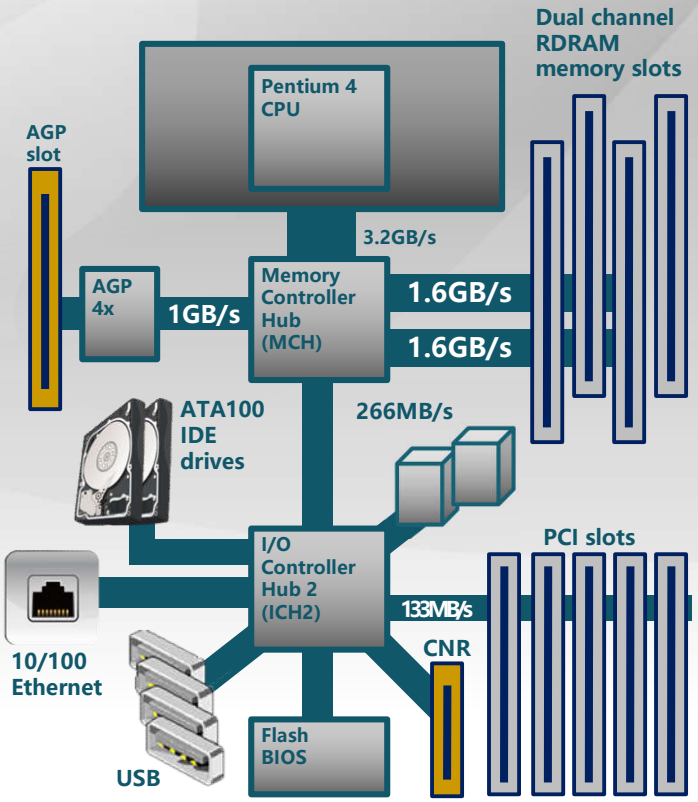
操作系统

Operating Systems

操作系统

Operating Systems

I/O 结构: 一个实际例子



北桥

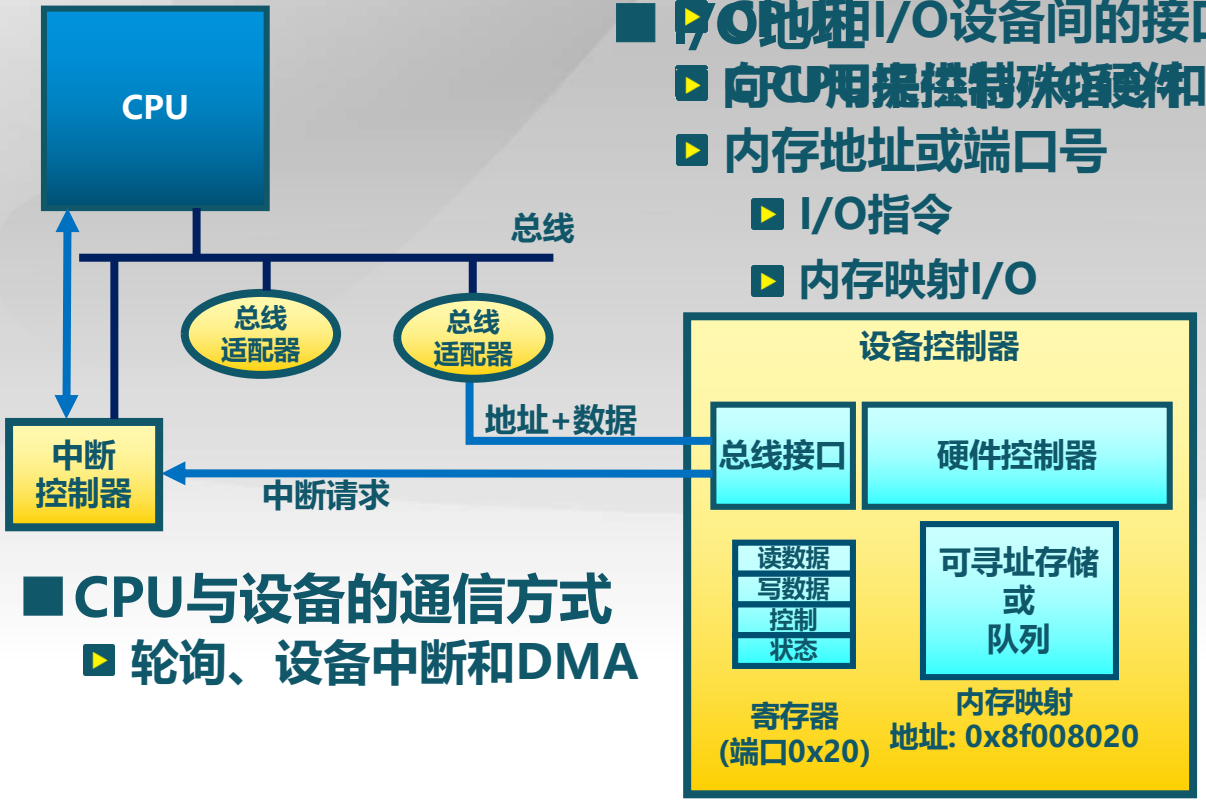
- 内存
- AGP/PCI-Express
- Built-in display

南桥

- ATA/IDE
- PCI总线
- USB/Firewire总线
- Serial/Parallel接口
- DMA 控制器
- Interrupt控制器
- RTC, ACPI, BIOS, ...

CPU与设备的连接

- 设备控制器
  - I/O地址：I/O设备间的接口
    - 向CPU用数据控制特殊指令和寄存器
    - 内存地址或端口号
      - I/O指令
      - 内存映射I/O

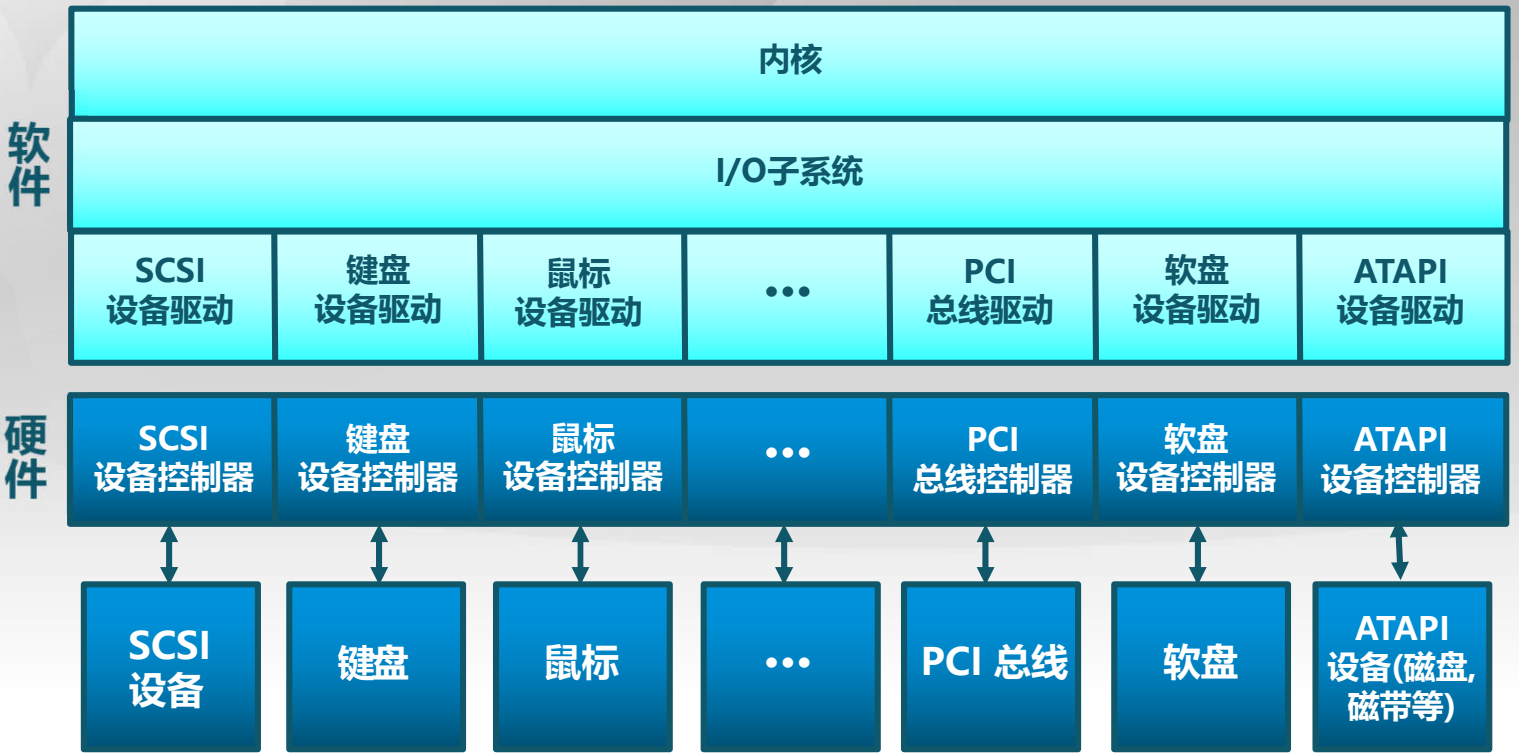


- CPU与设备的通信方式
  - 轮询、设备中断和DMA

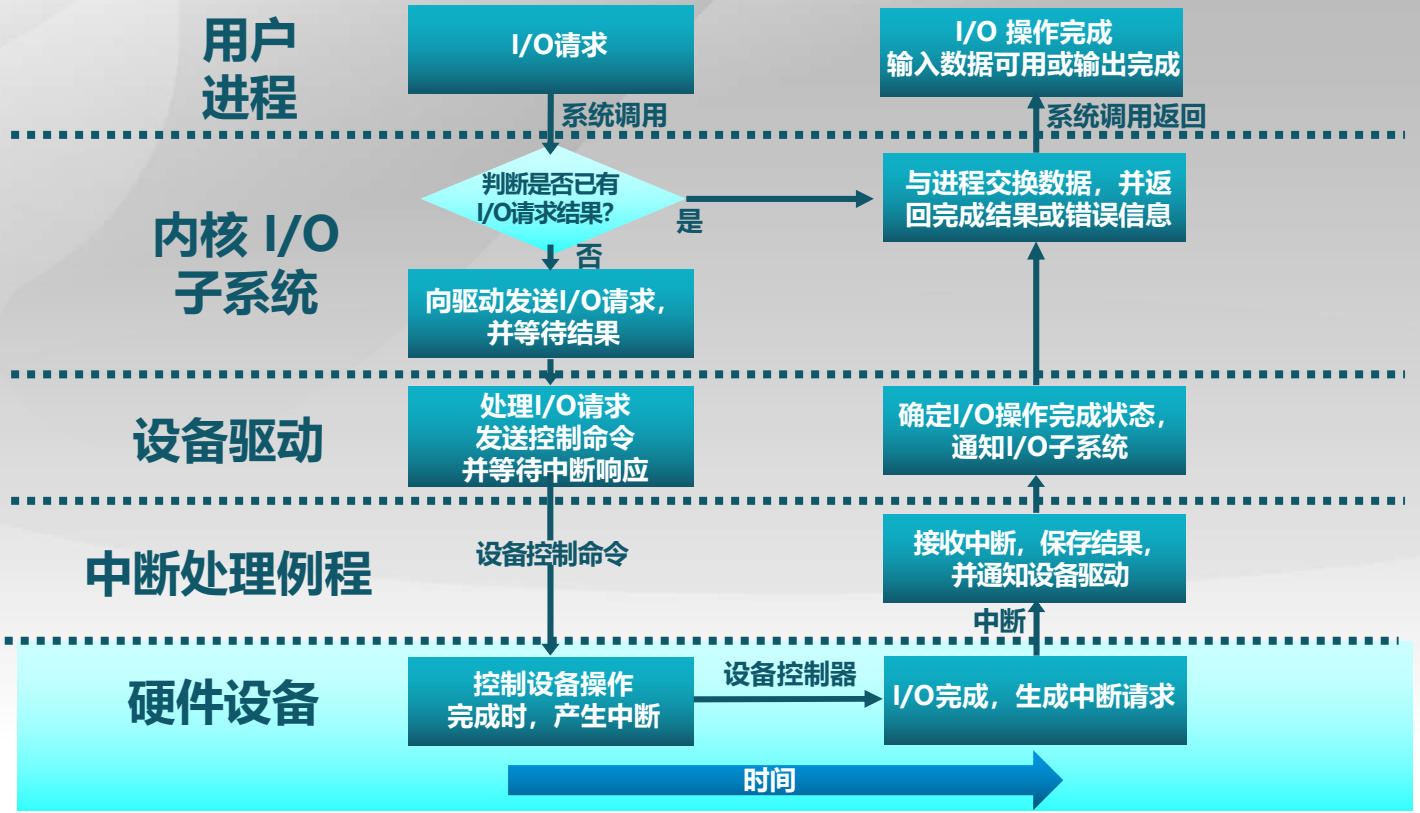
I/O 指令和内存映射I/O

- I/O指令
  - 通过I/O端口号访问设备寄存器
  - 特殊的CPU指令
    - out 0x21,AL
- 内存映射I/O
  - 设备的寄存器/存储被映射到内存物理地址空间中
  - 通过内存load/store指令完成I/O操作
  - MMU设置映射，硬件跳线或程序在启动时设置地址

内核I/O结构


软件
内核
I/O子系统
SCSI 设备驱动
键盘 设备驱动
鼠标 设备驱动
...
PCI 总线驱动
软盘 设备驱动
ATAPI 设备驱动
硬件
SCSI 设备控制器
键盘 设备控制器
鼠标 设备控制器
...
PCI 总线控制器
软盘 设备控制器
ATAPI 设备控制器
SCSI 设备
键盘
鼠标
...
PCI 总线
软盘
ATAPI 设备(磁盘, 磁带等)

I/O请求生存周期


用户进程
内核 I/O 子系统
设备驱动
中断处理例程
硬件设备
I/O请求
系统调用
判断是否已有I/O请求结果?
是
否
向驱动发送I/O请求，并等待结果
处理I/O请求
发送控制命令
并等待中断响应
设备控制命令
控制设备操作完成时，产生中断
设备控制器
I/O完成，生成中断请求
中断
接收中断，保存结果，并通知设备驱动
确定I/O操作完成状态，通知I/O子系统
与进程交换数据，并返回完成结果或错误信息
系统调用返回
I/O 操作完成
输入数据可用或输出完成
时间

操作系统

Operating Systems

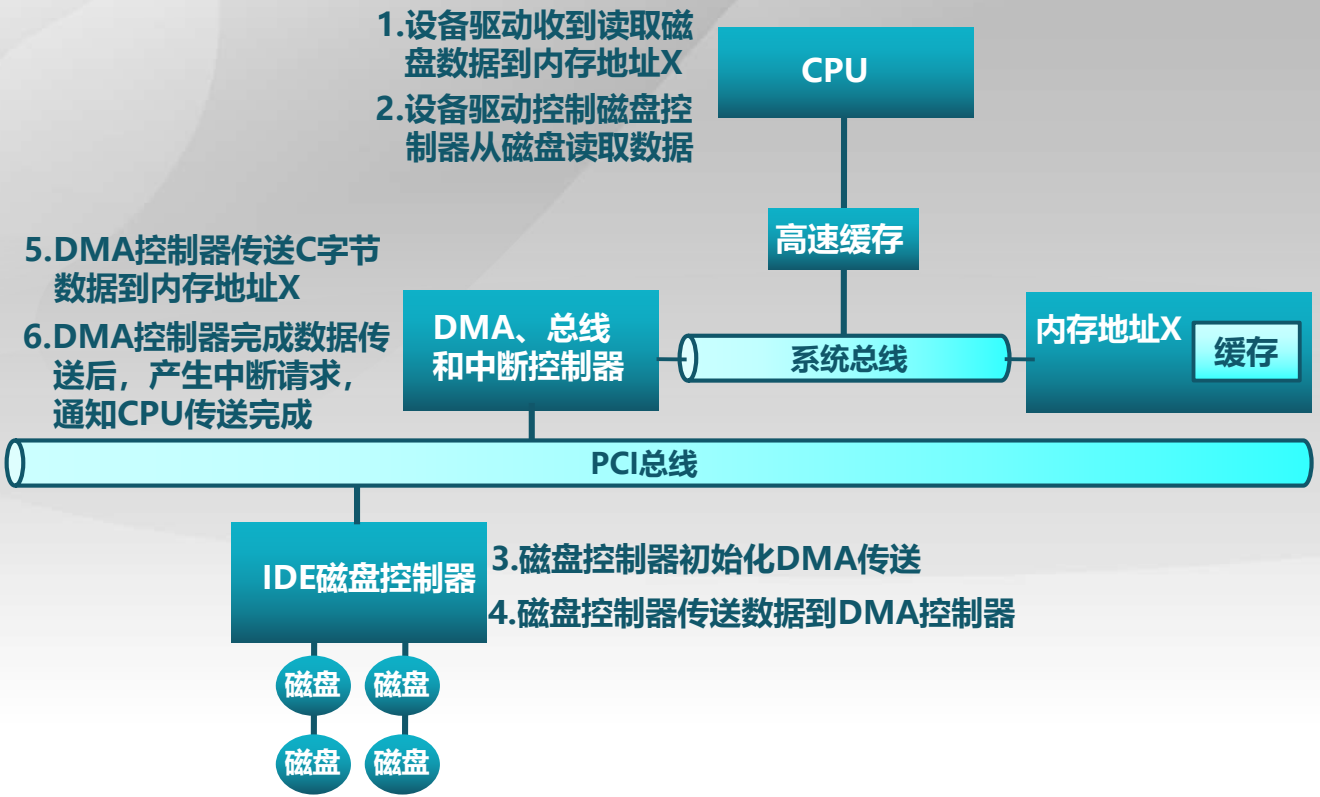
操作系统

Operating Systems

CPU与设备控制器的数据传输

- 程序控制I/O(PIO, Programmed I/O)
  - 通过CPU的in/out或者load/store传输所有数据
  - 特点
    - 硬件简单，编程容易
    - 消耗的CPU时间和数据量成正比
  - 适用于简单的、小型的设备I/O
- 直接内存访问(DMA)
  - 设备控制器可直接访问系统总线
  - 控制器直接与内存互相传输数据
  - 特点
    - 设备传输数据不影响CPU
    - 需要CPU参与设置
  - 适用于高吞吐量I/O

通过直接I/O寻址读取磁盘数据的步骤



I/O 设备通知操作系统的机制

- 操作系统需要了解设备状态
  - I/O操作完成时间
  - I/O操作遇到错误
- 两种方式
  - CPU主动轮询
  - 设备中断

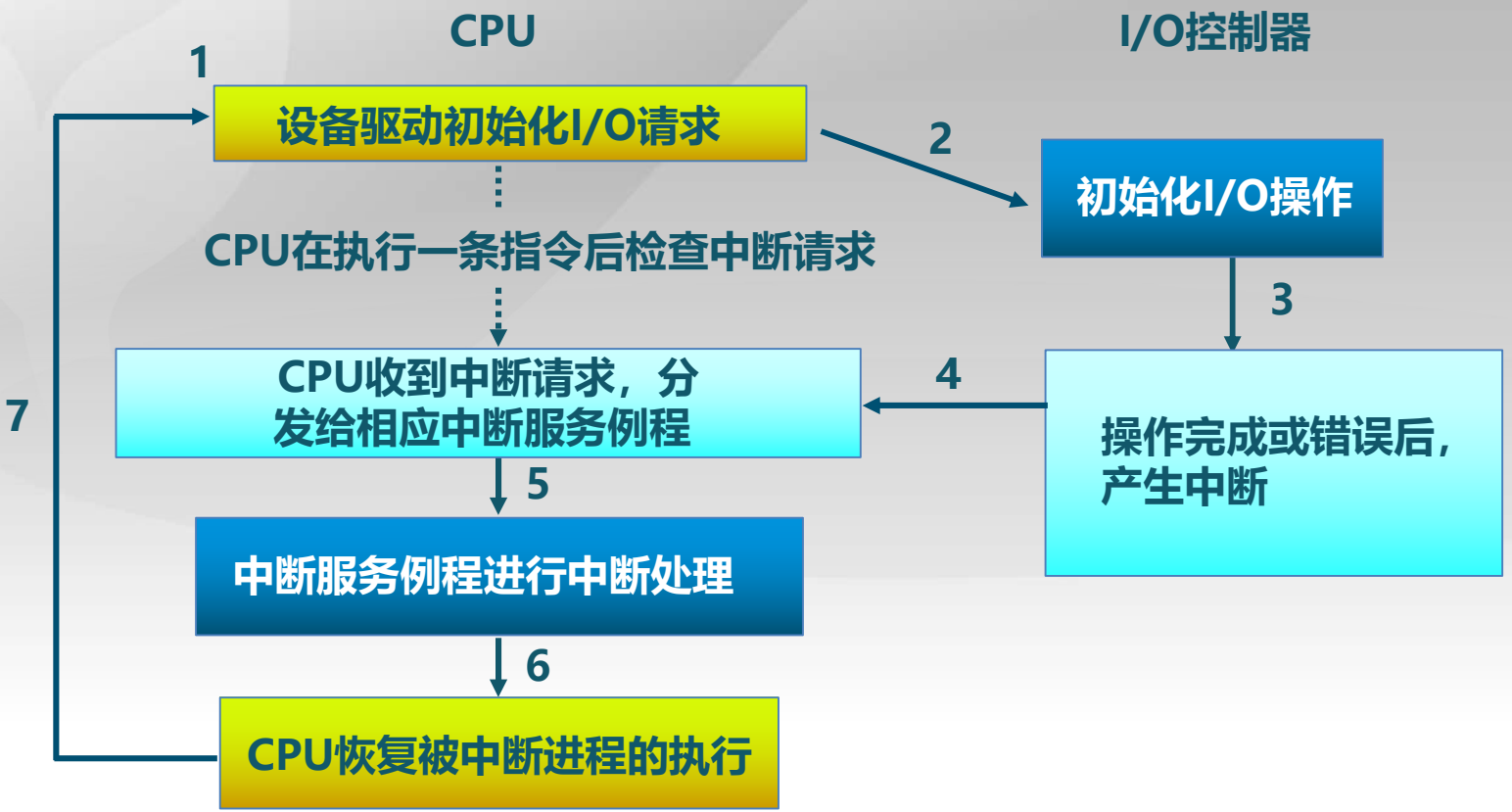
轮询

- I/O 设备在特定的状态寄存器中放置状态和错误信息
- 操作系统定期检测状态寄存器
- 特点
  - 简单
  - I/O操作频繁或不可预测时，开销大和延时长

设备中断

- 设备中断处理流程
  - CPU在I/O之前设置任务参数
  - CPU发出I/O请求后，继续执行其他任务
  - I/O设备处理I/O请求
  - I/O设备处理完成时，触发CPU中断请求
  - CPU接收中断，分发到相应中断处理例程
- 特点
  - 处理不可预测事件效果好
  - 开销相对较高
- 一些设备可能结合了轮询和设备中断
  - 如：高带宽网络设备
    - 第一个传入数据包到达前采用中断
    - 轮询后面的数据包直到硬件缓存为空

设备中断I/O处理流程


CPU
I/O控制器
1
设备驱动初始化I/O请求
2
初始化I/O操作
CPU在执行一条指令后检查中断请求
3
CPU收到中断请求，分
发给相应中断服务例程
4
操作完成或错误后，
产生中断
7
5
中断服务例程进行中断处理
6
CPU恢复被中断进程的执行

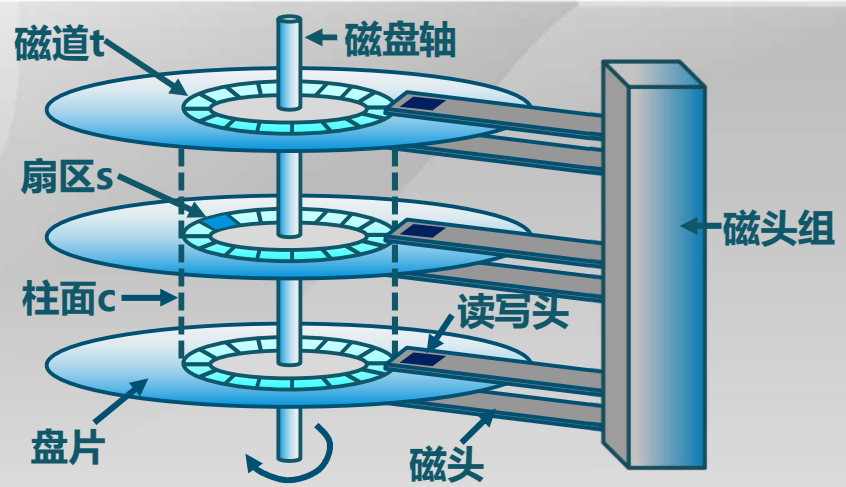
操作系统

Operating Systems

操作系统

Operating Systems

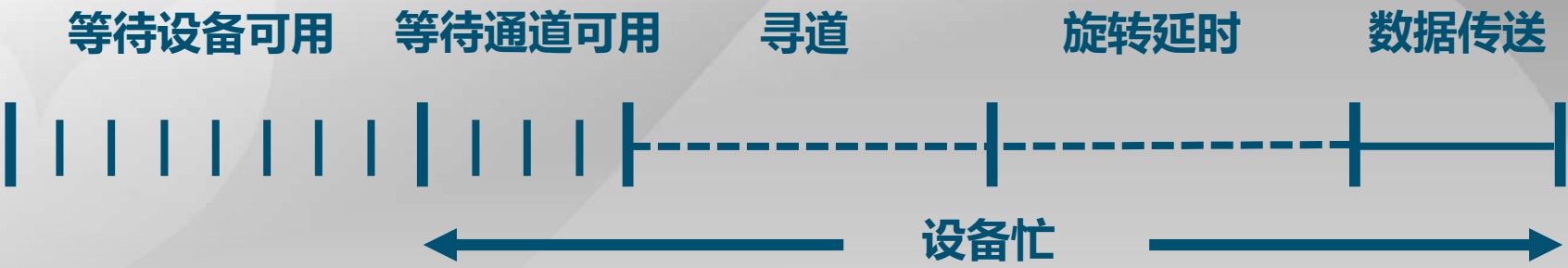
磁盘工作机制和性能参数



- 读取或写入时，磁头必须被定位在**期望的磁道**，并从所**期望的柱面和扇区**的开始
- 寻道时间
  - 定位到期望的磁道所花费的时间
- 旋转延迟
  - 从零扇区开始处到达目的地花费的时间

平均旋转延迟时间=磁盘旋转一周时间的一半

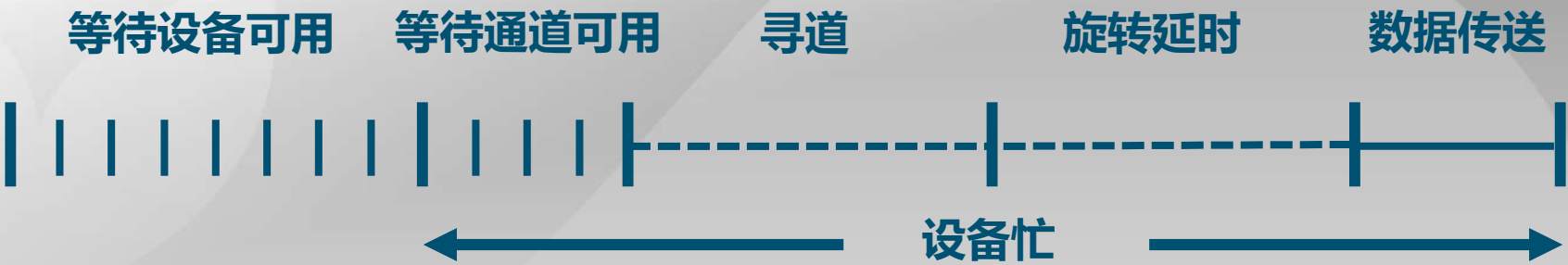
磁盘I/O传输时间



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

访问时间

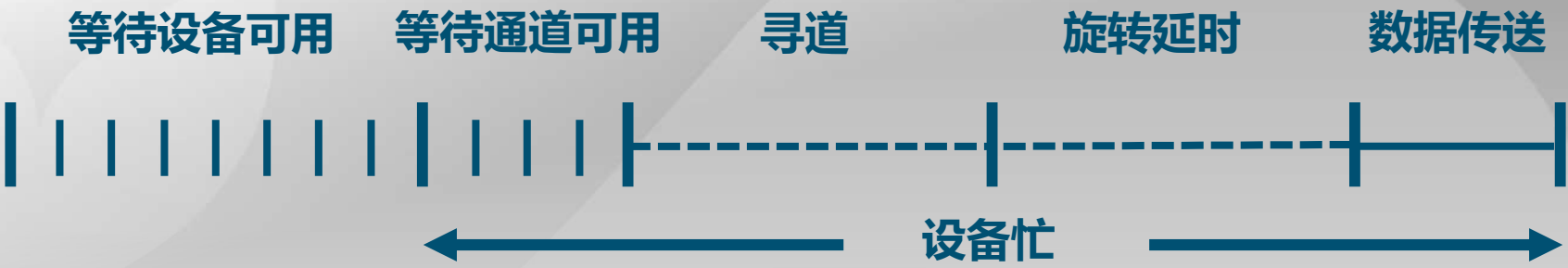
磁盘I/O传输时间



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

寻道时间

磁盘I/O传输时间

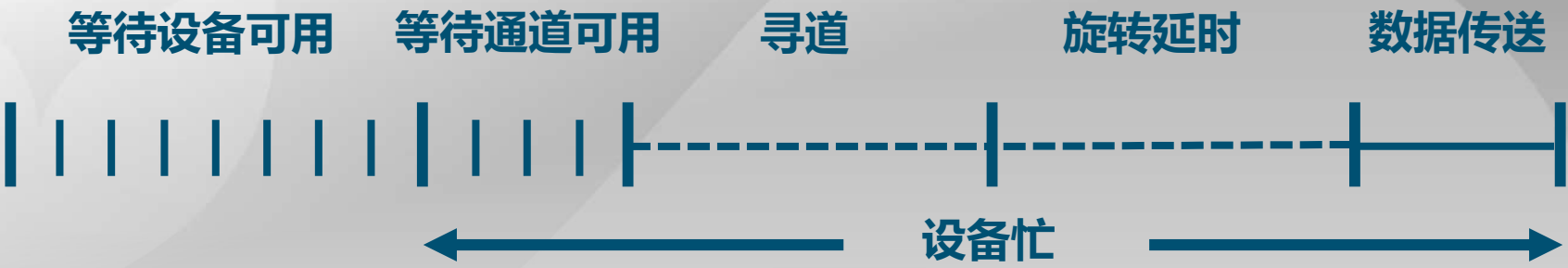


$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

旋转延迟

1/r =旋转一周的时间

磁盘I/O传输时间



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

传输时间

b = 传输的比特数
N = 磁道上的比特数
r = 磁盘转数

磁盘调度算法

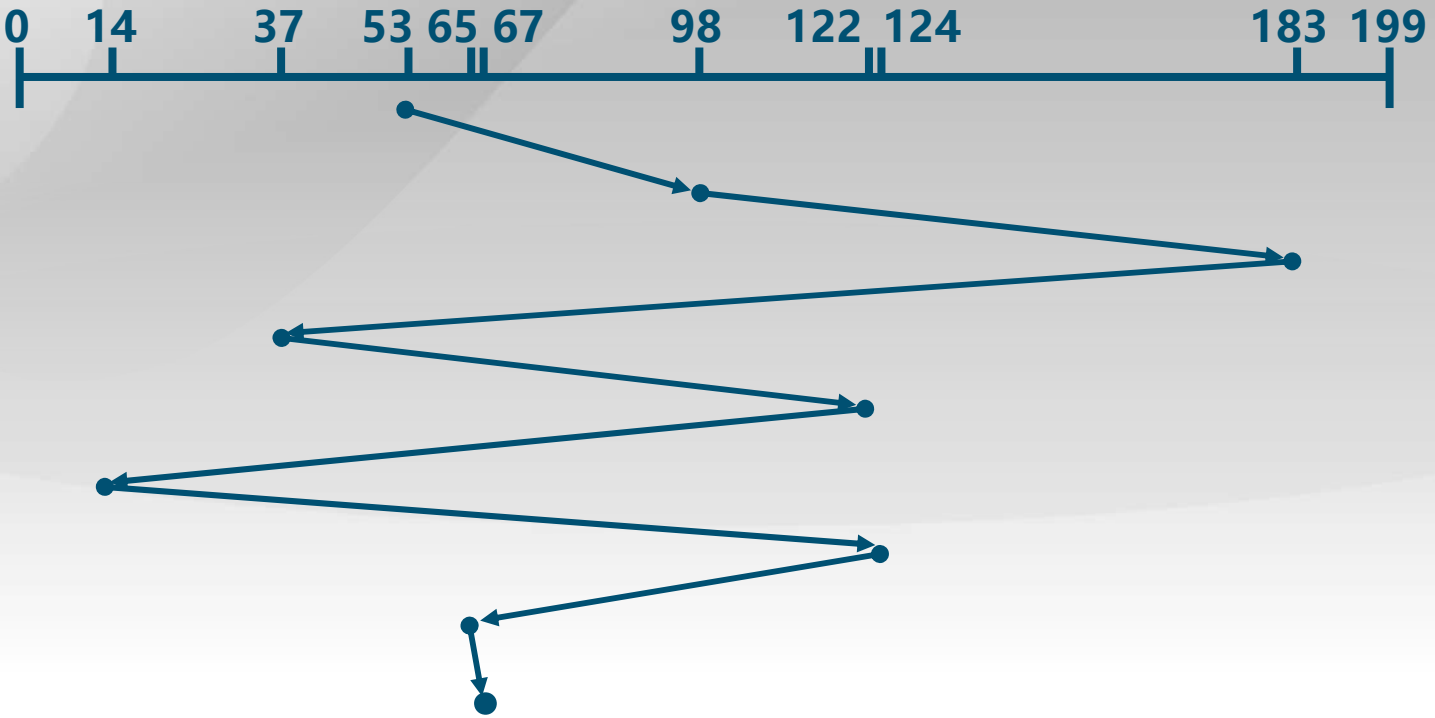
- 通过优化磁盘访问请求顺序来提高磁盘访问性能
  - 寻道时间是磁盘访问最耗时的部分
  - 同时会有多个在同一磁盘上的I/O请求
  - 随机处理磁盘访问请求的性能表现很差

先进先出(FIFO)算法

- 按顺序处理请求
- 公平对待所有进程
- 在有很多进程的情况下，接近随机调度的性能

FIFO算法示例

磁盘访问序列 = 98,183,37,122,14,124,65,67　　初始磁头位置：53



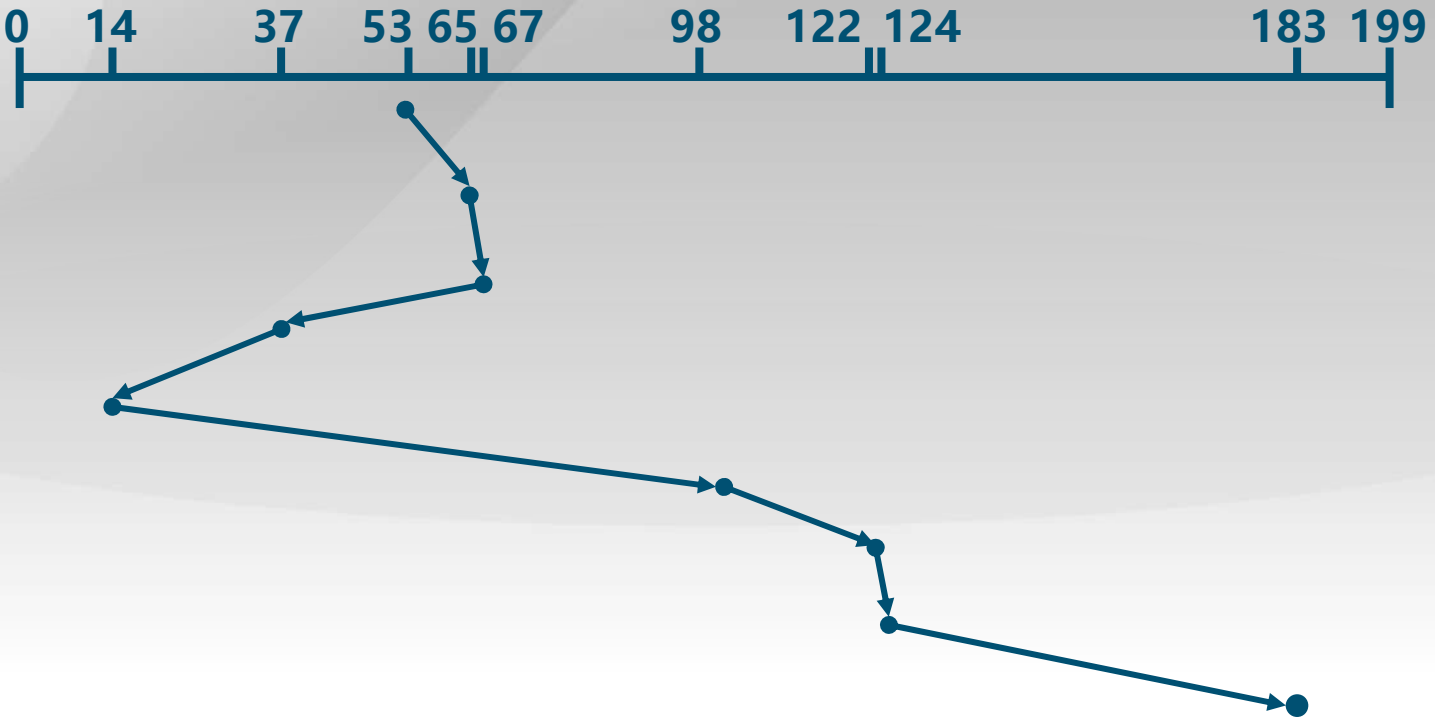
合计磁头移动距离=45+85+146+85+108+110+59+2=640

最短服务时间优先(SSTF)

- 选择从磁臂当前位置需要移动最少的I/O请求
- 总是选择最短寻道时间

SSTF算法示例

磁盘访问序列 = 98,183,37,122,14,124,65,67　　初始磁头位置：53



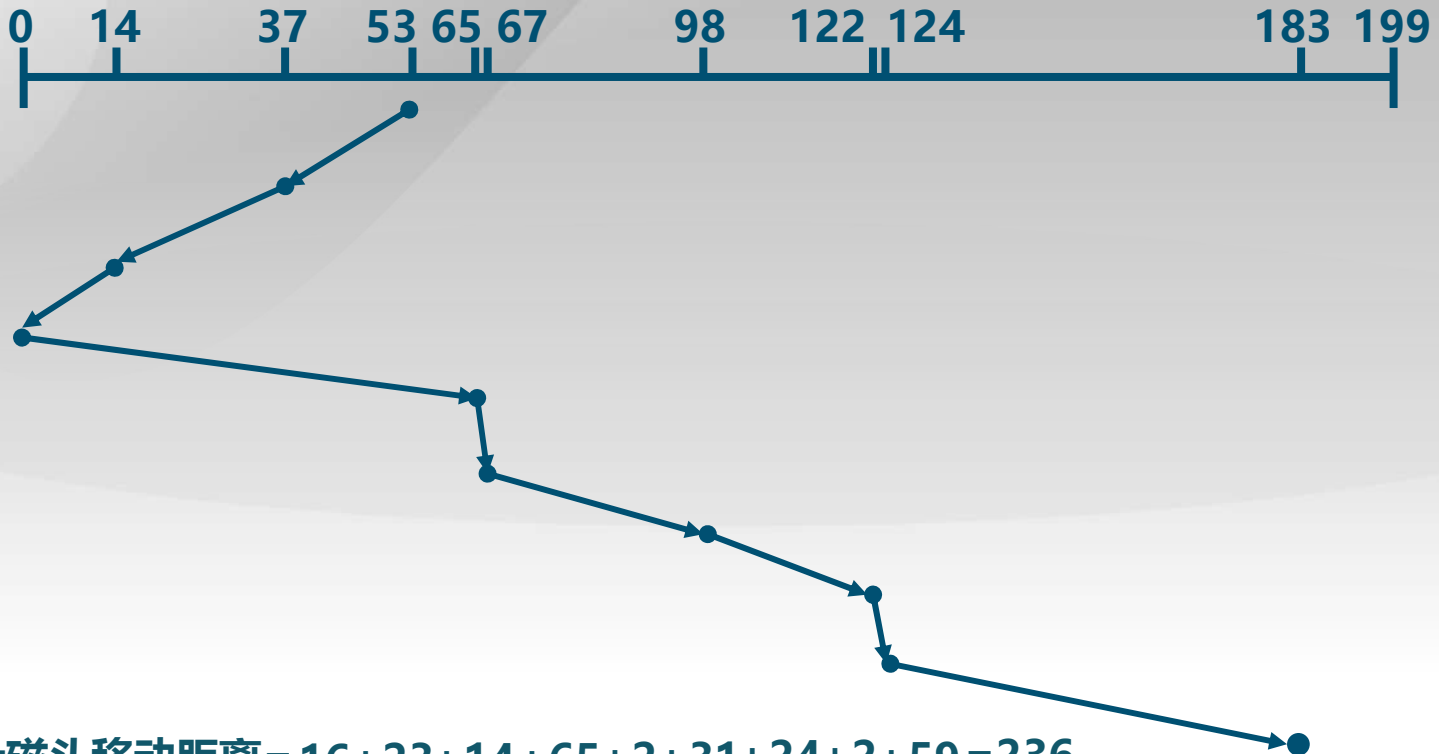
合计磁头移动距离=12+2+30+23+84+24+2+59=236

扫描算法(SCAN)

- 磁臂在一个方向上移动，访问所有未完成的请求，直到磁臂到达该方向上最后的磁道
- 调换方向
- 也称为电梯算法(elevator algorithm)

SCAN算法示例

磁盘访问序列 = 98,183,37,122,14,124,65,67　　初始磁头位置：53



合计磁头移动距离=16+23+14+65+2+31+24+2+59=236

循环扫描算法(C-SCAN)

- 限制了仅在一个方向上扫描
- 当最后一个磁道也被访问过了后，磁臂返回到磁盘的另外一端再次进行

C-LOOK算法

- 磁臂先到达该方向上最后一个请求处，然后立即反转，而不是先到最后点路径上的所有请求

N步扫描(N-step-SCAN)算法

- 磁头粘着(Arm Stickiness)现象
  - SSTF、SCAN及CSCAN等算法中，可能出现磁头停留在某处不动的情况
  - 如：进程反复请求对某一磁道的I/O操作
- N步扫描算法
  - 将磁盘请求队列分成长度为N的子队列
  - 按FIFO算法依次处理所有子队列
  - 扫描算法处理每个队列

双队列扫描(FSCAN)算法

- FSCAN算法是N步扫描算法的简化
  - FSCAN只将磁盘请求队列分成两个子队列
- FSCAN算法
  - 把磁盘I/O请求分成两个队列
  - 交替使用扫描算法处理一个队列
  - 新生成的磁盘I/O请求放入另一队列中
    所有的新请求都将被推迟到下一次扫描时处理

操作系统

Operating Systems

操作系统

Operating Systems

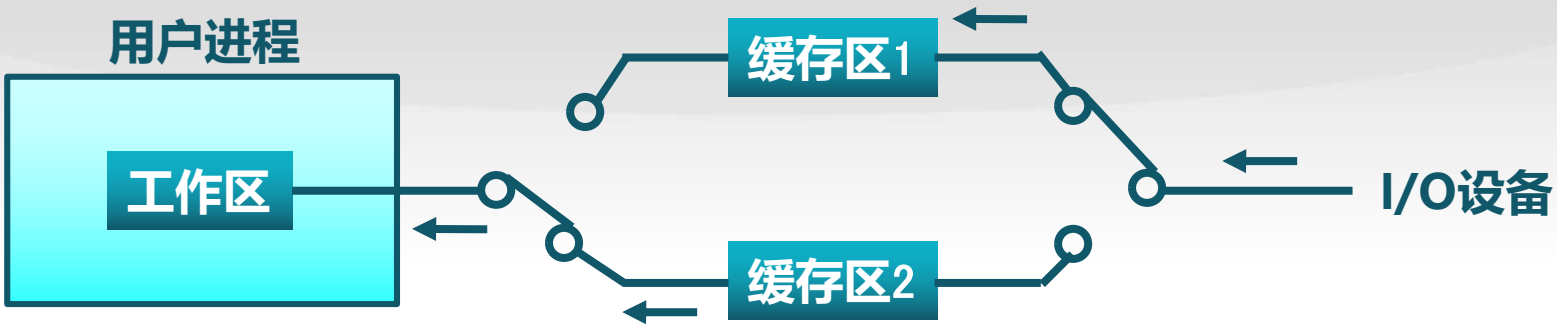
磁盘缓存

- 缓存
  - 数据传输双方访问速度差异较大时，引入的速度匹配中间层
- 磁盘缓存是磁盘扇区在内存中的缓存区
  - 磁盘缓存的调度算法很类似虚拟存储调度算法
  - 磁盘的访问频率远低于虚拟存储中的内存访问频率
  - 通常磁盘缓存调度算法会比虚拟存储复杂

单缓存与双缓存

单缓存(Single Buffer Cache)

双缓存(Double Buffer Cache)



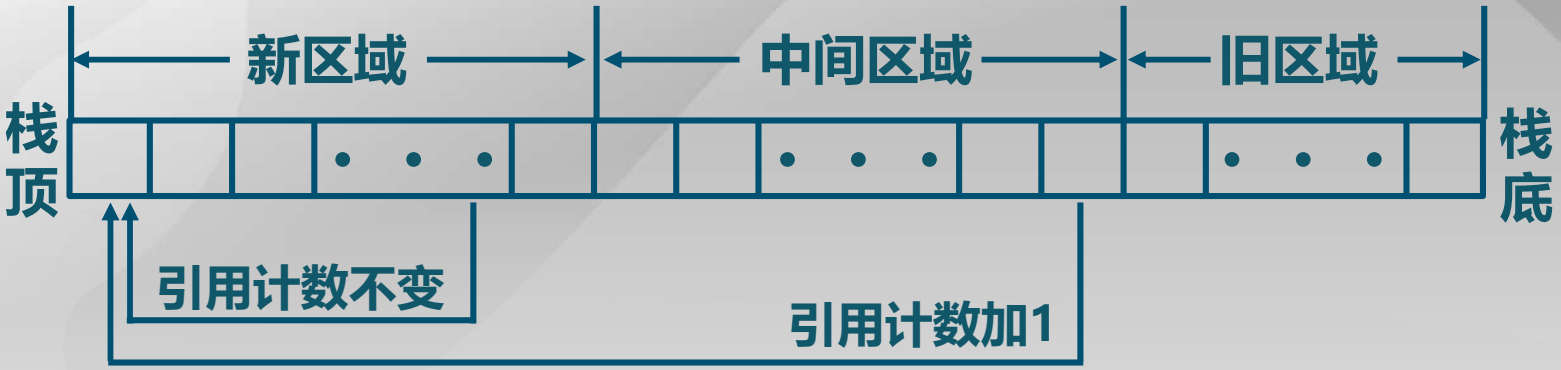
访问频率置换算法(Frequency-based Replacement)

- 问题
  - 在一段密集磁盘访问后，LFU算法的引用计数变化无法反映当前的引用情况
- 算法思路
  - 考虑磁盘访问的密集特征，对密集引用不计数
  - 在短周期中使用LRU算法，而在长周期中使用LFU算法

访问频率置换算法(Frequency-based Replacement)

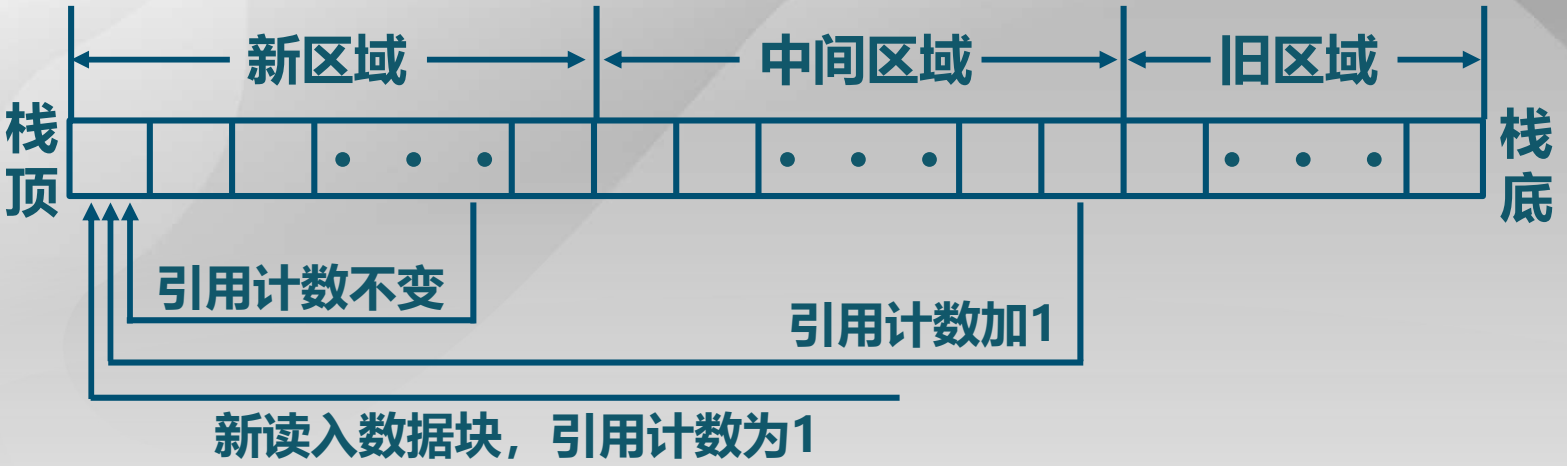
- 把LRU算法中的特殊栈分成三部分，并在每个缓存块增加一个引用计数
  - 新区域(New Section)
  - 中间区域(Middle Section)
  - 旧区域(Old Section)

访问频率置换算法(Frequency-based Replacement)



- 栈中缓存块被访问时移到栈顶；如果该块在新区域，引用计数不变；否则，引用计数加1
  - 在新区域中引用计数不变的目的是避免密集访问对引用计数不利影响
  - 在中间区域和旧区域中引用计数加1是为了使用LFU算法

访问频率置换算法(Frequency-based Replacement)



- 栈中缓存块被访问时移到栈顶；如果该块在新区域，引用计数不变；否则，引用计数加1
- 未缓存数据块读入后放在栈顶，引用计数为1
- 在旧区域中引用计数最小的缓存块被置换
  - 中间区域的定义是为了避免新读入的缓存块在第一次出新区域时马上被置换，有一个过渡期

小结

- I/O特点
- I/O结构
- I/O数据传输
- 磁盘调度
- 磁盘缓存

操作系统

Operating Systems