2021.6.2 wzh

操作系统引论

异常(exception): 陷阱(trap)和中断(interrupt)

类别				原因	返回行为	例子
异常	异步	中断(interrupt)	可屏蔽中断	来自 I/O 设备的信号	总是返回到下一条指令	所有的 IRQ 中断
			不可屏蔽中断			电源掉电和物理存储器奇偶校验
	同步	陷阱 (trap)		程序内部有意设置	总是返回到下一条指令	系统调用、信号机制等(通过中断指令实现)
		故障 (fault)		潜在可恢复的错误	返回到当前指令	缺页异常、除 0 错误、段错误
		终止 (abort)		不可恢复的错误	不会返回	硬件错误

系统引导

内存管理

存储管理基础

- ELF目标代码文件是由section组成的，而可执行文件的内容是由segment组成的
- 链接器将相同类型的section合并成segment
- ELF文件中：
  - .text 保存可执行文件的操作指令（由系统从可执行文件中加载）
  - .data 保存已初始化的全局变量和静态变量（由系统从可执行文件中加载）
  - .bss 保存未初始化的全局变量和静态变量（不在可执行文件中，由系统初始化）
- 执行程序的过程:
  - shell调用fork()系统调用，创建一个子进程
  - 子进程调用execve()加载program
    - ELF中Type为Load的segment是需要被加载到内存中的
- 一个segment在文件中的大小是小于等于其在内存中的大小

基础内存管理

- 闲置空间管理：位图表示法，链表表示法

单道程序

- 单道程序：静态地址翻译（在程序运行之前就计算出所有的物理地址）

多道程序

- 方法：固定（静态）式分区分配，程序适应分区；可变（动态）式分区分配，分区适应程序
- 固定式分区：
  - 把内存分成若干个固定大小的连续分区（分区大小不等，分区大小相等）
- 可变式分区：
  - 分区的边界可以移动，即分区大小可变
  1. 基于顺序搜索的分配算法：**适用于小型系统**（注意，找到后只切割出需要的空间，其他的依然留在空闲分区表中）
  - 首次适应算法(First Fit)：优先利用内存低地址的部分空闲分区，但是出现很多外碎片
  - 下次适应算法(Next Fit)：存储空间的利用更加均衡，不会导致小的空闲区集中在存储器的一端，但是会导致缺乏大的空闲分区
  - 最佳适应算法(Best Fit)：保留了大的空闲分区，留下了很多小碎片
  - 最坏适应算法(Worst Fit)：大作业往往得不到满足
  2. 基于索引搜索的分配算法：**适用于大中系统**
  - 快速适应算法：
    - 优点：查找效率高，仅需要根据程序的长度，寻找到能容纳它的最小空闲区链表，取下第一块进行分配即可。该算法在分配时，不会对任何分区产生分割，所以能保留大的分区，也不会产生内存碎片。
    - 缺点：在分区归还主存时算法复杂，系统开销较大。在分配空闲分区时是以进程为单位，一个分区只属于一个进程，存在一定的浪费。空间换时间。
- 伙伴系统：介于固定分区和可变分区之前的动态分区技术
  - 伙伴：在分配存储块时，将一个大的存储块分裂成两个大小相等的小块，这两个小块就称为伙伴
- 碎片：内存中无法被利用的存储空间
  - 内部碎片：分配给作业的存储空间中，未被利用的部分，如固定分区中存在的碎片
  - 外部碎片：系统中无法利用的小的空闲分区，如分区与分区之间的碎片。**外部碎片才是造成内存系统性能下降的主要原因，外部碎片可以被整理后清除（紧凑技术）**
- 分区保护：防止一个作业破坏其他作业或系统
  - 界限寄存器方法
  - 存储保护键方法
- 覆盖与交换技术：多道程序环境下用来扩充内存的方法
  - 可以解决小内存运行大作业的问题

页式存储管理

- 作业包括程序，数据，操作说明
- 进程包括程序，数据集合
- 纯分页系统：不具备页面对换功能，不支持虚拟存储器功能

段式存储管理

- 段表：记录段与内存位置的对应关系
- 访问一个字节的数据需要访问两次内存（段表一次，内存一次）
- 逻辑地址由段号和段内地址组成
- **可重入代码：允许多个进程同时访问的代码，不允许任何进程对他进行修改**

段页式存储管理

- 用分段方法分配和管理虚拟存储器，分页方法分配和管理实存储器

虚拟存储管理

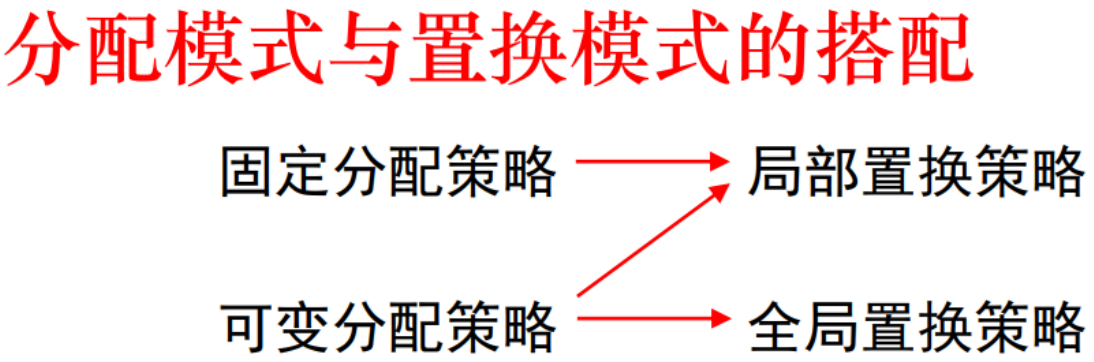
- 虚拟存储技术特征：离散性，多次性（多次调入内存），对换性（允许换入换出），虚拟性（从逻辑角度访问存储器，不考虑物理内存上的可用空间数量）
- 交换分区：一段连续的按页划分的磁盘空间，在物理内存不够的情况下，操作系统先把内存中暂时不用的数据，存到交换分区
- 页面置换策略：
  - 最优置换(OPT)
  - FIFO:
    - Belady现象：分配的缺页增多，但是缺页率反而提高了
    - 改进的FIFO: Second Chance, Clock
  - 最近最久不用算法(LRU):
    - Aging(LRU简化算法)
- 进程的工作集；当前正在使用的页面的集合
- 进程的驻留集：虚拟存储系统中，每个进程驻留在内存的页面集合，或进程分配到的物理页框集合

分配模式

- 页面分配策略：
  - 固定分配策略：为每个活跃的进程分配固定数量的页框，即驻留集尺寸不变
  - 可变分配策略

置换模式

- 页面置换策略：
  - 局部置换：系统在进程自身的驻留集中判断当前是否存在空闲页面，并在其中进行置换
  - 全局置换策略：在整个内存空间中判断有无空闲页面，并允许从其他进程的驻留集中选择一个页面换出内存



- 抖动问题：
  - 随着驻留内存的进程数目增加，处理器的利用率先上升后下降
  - 消除与预防：
    - 局部置换策略
    - 引入工作集算法
    - 预留部分页面
    - 挂起若干进程

进程与线程

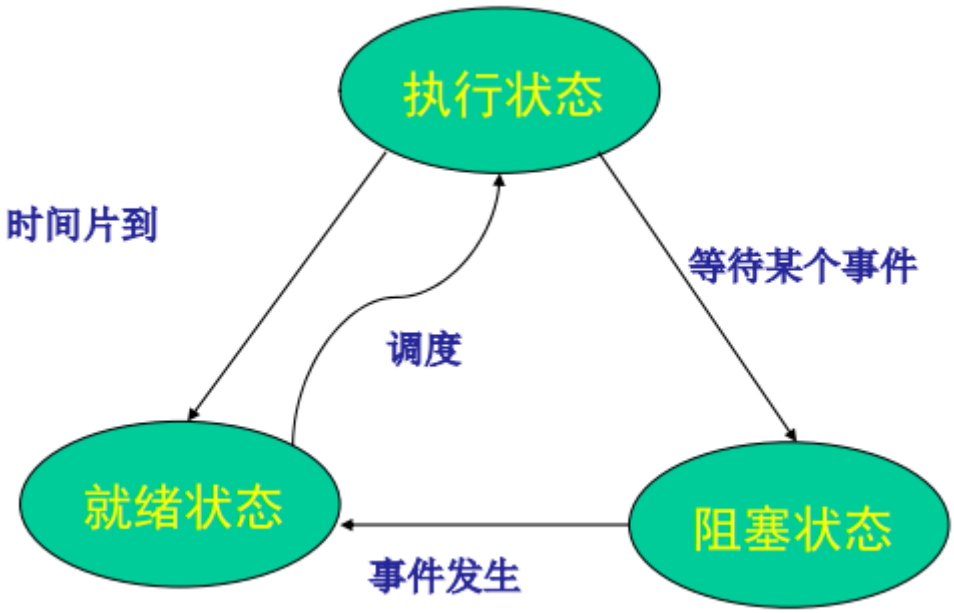
进程与线程

- 并发(concurrent)：指若干个程序同时在系统中运行，这些程序的执行在时间上是重叠的
- 并行(parallel)
- 竞争：多个进程在读写一个共享数据时结果依赖于它们执行的相对时间
- 竞争条件：多个进程并发访问和操作同一数据且执行结果与访问的特定顺序有关，称为竞争（发生）条件
- Bernstein条件：

Bernstein条件：

- 两个进程S1和S2可并发，当且仅当下列条件同时成立：
  - $R(S1) \cap W(S2) = \Phi$
  - $W(S1) \cap R(S2) = \Phi$
  - $W(S1) \cap W(S2) = \Phi$

进程

- 进程的定义和特征：
  - 进程是程序在一个数据集合上运行的过程，他是系统进行资源分配和调度的一个独立单位
  - **进程的组成包括程序、数据、进程控制块**
- 进程的状态：
  - 就绪，执行，阻塞状态
  - 

线程

- 进程包括了两个概念：资源拥有者和可执行单元
  - 进程：资源拥有者
  - 线程：可执行单元
- 引入线程的目的：
  - 减小进程切换的开销
  - 提高程序内的并发程度
  - 共享资源
- 线程共享进程数据的同时，有自己私有的栈

线程的实现方式

- 用户级线程：通过library模拟的线程
- 内核级线程：kernel有多个分身
  - 支持内核线程的操作系统称为多线程内核
- 两种线程的比较：
  - 内核级线程是操作系统内核可感知的，而用户级线程是操作系统内核不可感知的
  - 用户级线程的创建、撤销和调度不需要操作系统内核的支持，是在语言或用户库这一级处理的。而内核支持线程的创建、撤销和调度都需要操作系统内核提供支持
  - 用户级线程执行系统调用的时候，导致所属的进程被中断，而内核级线程只导致该线程中断
- 有些系统同时支持用户线程和内核线程，产生了不同的多线程模型：Many-to-One, One-to-One, Many-to-Many

进程同步

- **临界资源：一次只允许一个进程访问的资源**
- **临界区：每个进程中访问临界资源的那段代码**
- 进程互斥（间接制约关系）：两个及以上的进程，不能同时进入关于同一组共享变量的临界区域，否则可能发生与时间有关的错误，这种现象称为进程互斥
- 进程同步（直接制约关系）：系统中各进程之间能有效共享资源和相互合作，从而使程序的执行具有可再现性的过程称为进程同步

基于忙等待的互斥方法：

- 软件方法：Dekker算法，Peterson算法，面包店算法
- 硬件方法：中断屏蔽，使用test and set指令，使用swap指令
  共性问题：

  1.忙等待：浪费CPU时间

  2.优先级反转：低优先级进程先进入临界区，高优先级进程一直忙等

基于信号量的同步方法

- 信号量的定义：一个确定的二元组(s,q)，其中s是一个具有非负初值的整型变量，q是一个初始状态为空的队列

- 二元信号量：取值仅为0和1，主要实现互斥
- 一般信号量：初值为可用物理资源的总数，用于进程间的协作同步问题

- 强信号量：进程从被阻塞队列释放时采取FIFO

- 弱信号量：没有规定进程从阻塞队列中移除顺序

```
P(S):
while S <= 0 do skip;
S = S -1;

V(S):
S = S + 1;
```

- 信号量集机制：包括AND型信号量集和一般信号量集
  - AND型信号量集：
    - 基本思想：将进程所需的全部共享资源一次全部分配给它，待该进程使用完后一起释放
  - 一般信号量集：
    - 基本思想：SP(S1, t1(测试值), d1(占用值))

进程间通信(Inter-Process-Comm)

- 低级通信：只能传递状态和整数值，包括进程互斥和同步采用的信号量和管程机制
- 高级通信：适用于分布式系统，基于共享内存的多处理机系统，单处理机系统，能够传递任意数量的数据，可以解决进程的同步问题和通信问题，主要包括：**管道，共享内存，消息系统**
- 无名管道只能用于具有亲缘关系的进程，有名管道克服了这一点

进程调度

- CPU调度：CPU调度的任务是控制协调多个进程对CPU的竞争，也就是按照一定的策略，从就绪队列中选择一个进程，并把CPU的控制权交给被选中的进程。
- 调度的类型：高级调度，中级调度，低级调度
  - 高级调度（作业调度）：从用户工作流程的角度，一次提交若干个作业，对每个作业进行调度。（接纳多少个作业、接纳哪些作业）
  - 中级调度（内外存交换）：从存储器资源的角度，将进程的部分或者全部换出到外村，将当前所需要的部分换入内存，
  - 低级调度：从CPU资源的角度，**分为抢占式和非抢占式**：
    - 非抢占式：一旦处理器分配给一个进程，它就一直占用处理器直到该进程自己阻塞或者时间片用完时才让出处理器
    - 抢占式：
      - 时间片原则
      - 优先权原则
      - 短作业（进程）优先原则
- 周转时间：作业从提交到完成所经历的时间
- 响应时间：用户输入一个请求到系统**首次**给出相应的时间
- 吞吐量：单位时间内所完成的作业数
- 带权周转时间：周转时间/服务时间（执行时间）

批处理系统的调度算法

- 先来先服务(FCFS)，最短作业优先(SJF)，最短剩余时间优先(FRTF)，最高响应比优先(HRRF)

- 先来先服务：非抢占方式
  - 有利于长作业，不利于短作业
  - 有利于CPU繁忙的作业，不利于I/O繁忙的作业
- 短作业优先：非抢占
  - 比FCFS改善了平均周转时间和平均带权周转时间，缩短了作业的等待时间
  - 提高了系统的吞吐量
  - 对长作业不利
  - 难以估计执行时间，导致调度性能下降
  - 未能依据紧迫程度划分优先级
- 最短剩余时间：抢占
  - 大量的短作业会导致长作业饥饿
- 最高响应比：选择作业响应比RP值最大的执行：
  - RP = (已等待时间+要求运行时间)/要求运行时间
  - 计算时机：每当调度一个作业运行的时候

交互式系的调度算法

- 时间片轮转(RR)，多级队列(MQ)，多级反馈队列(MFQ)

- 时间片轮转
- 优先级算法：
  - 平衡各个进程对响应时间的要求，适用于作业调度和进程调度，可分为抢占式和非抢占式
  - 静态优先级和动态优先级
- 多级队列算法：不同队列可有不同的优先级、时间片长度、调度策略等；在运行过程中还可改变进程所在队列。如：系统进程、用户交互进程、批处理进程等。
- 多级反馈队列算法：**WARNING!!!如果进程执行时有新进程进入较高优先级的队列，则抢先执行新进程，并把被抢先的进程投入原队列的末尾。**

死锁

- 由于资源占用互斥，当某个进程提出资源申请之后，使得一些进程在无外力协助的情况下，永远分配不到必须的资源而无法执行
- 死锁发生的原因：
  - 竞争资源
  - 并发执行的顺序不当
- 死锁发生的四个必要条件：
  - 互斥条件
  - 请求和保持条件
  - 不剥夺条件
  - 环路等待条件
- 活锁：指任务或执行者没有被阻塞，由于某些条件一直重复尝试
  - 与死锁的区别：处于活锁的实体是在不断改变状态的，而死锁的实体表现为等待。活锁有可能自行解开。**通过先来先服务的策略解决**

- 饥饿：某些进程可能由于资源分配策略的不公平导致长时间等待。当等待时间给进程推进和响应带来明显影响时，称发生了进程饥饿，当饥饿到一定程度的进程所赋予的任务即使完成也不再具有实际意义时称该进程被饿死

处理死锁的基本方法

- 不允许死锁发生
  - 预防死锁（静态）
  - 避免死锁（动态）：在资源分配之前检查
- 允许死锁发生
  - 检测与解除死锁
  - do nothing
- 死锁预防：
  - 针对互斥条件：允许进程同时访问某些资源
  - 针对请求和保持条件：实行资源预先分配的策略，只有当系统能够满足当前进程的全部资源时，一次性地分配资源
    - 弊端：
      1. 通常不可能知道它需要的全部资源
      2. 资源利用率低
      3. 降低进程地并发性
  - 针对不剥夺条件：允许进程强行从占有者那里剥夺
    - 实现困难，降低系统性能
  - 打破循环等待条件：实行资源有序分配
    - 限制了进程对资源的请求，编号困难，增加系统开销
    - 为了按顺序申请，暂时不用的资源也要提前申请，降低了并发性
- 避免死锁
  - 安全序列：系统是安全的，是指系统中的所有进程能够按照某种次序分配资源且依次执行完毕
  - 分清**需求**和**最大需求**
  - 银行家算法：
    - 特点：
      - 允许互斥、部分分配和不可抢占，可以提高资源利用率
      - 要求事先说明最大资源需求，现实中很困难
- 检测死锁
  - 资源分配图(RAG)：圆圈表示进程，矩形表示一类资源，矩形中的小圈代表每个资源
    - RAG中有环不一定有死锁
    - 封锁进程：指某个进程由于请求超过了系统中现有的未分配资源数目的资源，而被系统封锁的进程
    - 化简：不断分配释放资源，使进程成为孤立结点
    - **死锁定理**：系统中某时刻为死锁状态的充要条件是该时刻系统的RAG是**不可完全化简**的
- 死锁解除
  - 两种方法：撤销进程，剥夺资源
    - 撤销进程：使全部死锁的进程夭折，按照某种顺序逐个撤销进程，直到有足够的资源可用
    - 剥夺资源：使用挂起/激活，挂起一些进程，剥夺它们的资源，以后再解除

设备管理

I/O硬件组成

- I/O端口地址：内存映像编址，I/O独立编址

I/O控制方式

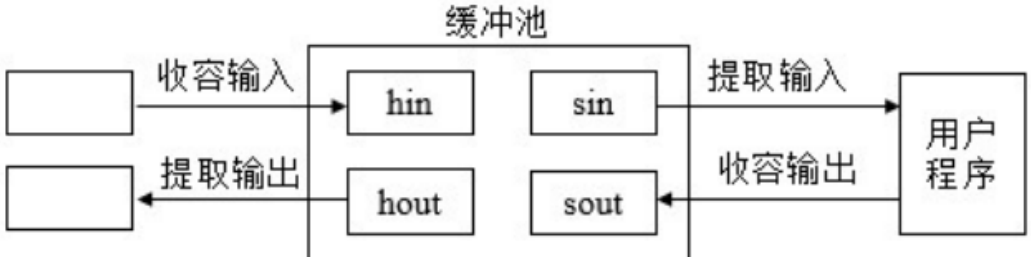
- 程序控制I/O（轮询或查询方式I/O）：由CPU向I/O模块发出指令，然后开始忙等
- 中断驱动：I/O操作结束以后，由设备控制器主动通知设备驱动程序，而不是轮询
- DMA：直接存储器访问方式，由一个专门的控制器来完成数据从内存到设备或者从设备到内存的传输
    - 步骤
        - 由程序设置DMA控制器中的若干寄存器值（如内存始址，传送字节数），然后发起I/O操作
        - DMA控制器完成内存与外设的成批数据交换
        - 在操作完成时由DMA控制器向CPU发出中断
    - 优点：
        - CPU只需要干预I/O操作的开始和结束，而后续成批的数据读写无需CPU控制，适用于高速设备
    - 缺点：
        - 占用CPU时间
        - 每个设备占用一个DMA控制器，当设备增加时，需要增加新的DMA控制器
- 通道：通道是一个特殊功能的处理器，有自己的指令和程序专门负责数据输入输出的传输控制
    - 优点：
        - 相比DMA进一步减少了CPU干预
    - 缺点：
        - 贵
    - I/O通道分类：
        - 字节多路通道；以字节为单位交叉工作，给一个设备传送一个字节以后，立即转去为另一个设备传送字节。适用于连接打印机，终端等中低速I/O设备
        - 数组选择通道：以“组方式”工作，每次传送一批数据，当一个I/O请求处理完以后，服务另一个。适用于连接磁盘、磁带等高速设备
        - 数组多路通道：综合了以上两种，与通道连接的设备可以并行

- 中断控制方式在每个数据完成传送以后中断，而DMA是在一批数据完成以后中断

I/O软件组成

- 设备独立性：逻辑设备和物理设备
    - 好处：
        - 设备分配时的灵活性
        - 易于实现I/O重定向
    - 逻辑设备表LUT(Logic Unit Table)，用于逻辑设备名和物理设备名之间的映射
        - 每个项目包括，逻辑设备名，物理设备名，设备驱动程序的入口
        - 设置方式：整个系统、每个用户
- 设备驱动程序：不能使用标准C库

I/O缓冲管理

- 缓冲技术可以提高外设利用率：
  - 匹配CPU与外设的不同处理速度
  - 减少CPU的中断次数
  - 提高CPU和I/O设备之间的并行性
- 专用缓冲：适用于特定的I/O进程和计算进程
  - 单缓冲：每当用户进程发出一个I/O请求时，OS便在主存中分配一个缓冲区
  - 双缓冲
  - 环形缓冲
- 缓冲池：为了方便管理，将相同类型的缓冲区链成队列
  - 空缓冲队列，输入队列，输出队列
  - 缓冲区可以工作在收容输入、提取输入、收容输出、提取输出四种工作方式下。
  - 

I/O设备管理

- I/O设备分配：由于外设资源的优先，需要解决进程间的外设共享问题，以提高外设资源的利用率
  - 两种做法：
    - 在进程间切换使用外设
    - 通过虚拟设备把外设和应用进程隔开
  - 设备控制表DCT(Device Control Table)：每个设备一张，描述设备特性和状态，
- SPOOLing技术：用户空间的I/O软件
  - SPOOLing：假脱机或虚拟设备技术，可把独享设备转变为具有共享特征的虚拟设备，提高设备利用率
  - 在多道程序系统中，专门利用一道程序（SPOOLing程序）来完成对设备I/O操作
  - SPOOLing程序和外设进行数据交换：实际I/O
  - 应用程序和SPOOLing程序交换数据：虚拟I/O
  - SPOOLing系统的组成：
    - 输入井和输出井：这是在磁盘上开辟的两个大存储空间。输入井是模拟脱机输入时的磁盘设备，用于暂存I/O设备输入的数据，输出井是模拟脱机输出时的磁盘，用于暂存用户程序和输出数据。
    - 输入缓冲区和输出缓冲区：缓和CPU与磁盘之间速度不匹配，在内存中开辟的两个缓冲区，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，以后再传送给输出设备
    - 输入进程SPi和输出进程SPo

I/O性能问题

- 两个途径：
  - CPU利用率尽可能不被I/O降低：
    - 缓冲，异步I/O

- CPU尽可能摆脱I/O
  - DMA，通道

磁盘管理

- 磁盘访问时间：寻道时间、旋转延迟时间、传输时间
  - 寻道时间：启动磁盘时间s，磁头移动磁道数n。`T=m*n+s`
  - 旋转延迟时间：`T=1/2r`
  - 传输时间：每次读写字节数b，旋转速度r，磁道上的字节数N。`T=b/rN`。典型N值为512B

磁盘调度算法

- 先来先服务(FCFS)，最短寻道时间优先算法(SSTF)，扫描算法(SCAN)，循环扫描算法(CSCAN)，LOOK，C-LOOK
- 先来先服务算法：简单，公平，效率不高
- 最短寻道时间优先：改善了磁盘的平均服务时间，可能产生饥饿
- 扫描算法：当有访问请求时，磁头按一个方向移动，在移动过程中对遇到的访问请求进行服务，然后判断该方向上是否还有访问请求，如果有则继续扫描；否则改变移动方向，并为经过的访问请求服务，如此反复。
  - 克服了最短寻道优先的缺点，但两侧磁道的被访问频率低于中间磁道
- 循环扫描算法：类似示波器扫描电压工作
- **采用SCAN算法和C-SCAN算法时磁头总是严格地遵循从盘面的一端到另一端，显然，在实际使用时还可以改进，即磁头移动只需要到达最远端的一个请求即可返回，不需要到达磁盘端点。**
- LOOK，C-LOOK

磁盘空间管理

- 位图，空闲表法，空闲链表法，成组链接法
- 成组链接法：
  - 空白块号登记不占用额外空间
  - 节省时间
  - 采用后进先出的栈结构思想
  - 

RAID(廉价冗余磁盘阵列)

级别	说明	特点	磁盘数量	最少磁盘数量
0	条带化	速度加倍，可靠性减半，容量不变	n	2
1	镜像	速度不变，可靠性加倍，容量减半	n*2	2
0+1	组内条带化、组间镜像	速度加倍，可靠性加倍，容量减半	n*2	4
1+0	组内镜像，组间条带化	速度加倍，可靠性加倍，容量减半	2*n	4
2	海明码校验	校验独立存储，数据位交叉（条带化），自动纠正1个盘错	n+log(n)	7*
3	奇偶校验	校验独立存储，数据位交叉（条带化）容1块盘错	n+1	3
4	奇偶校验	校验独立存储，数据块交叉，容1块盘错	n+1	3
5	奇偶校验	校验交叉存储，数据块交叉，容1块盘错	n+1	3
6	奇偶校验	校验交叉存储，数据块交叉，容2块盘错	n+2	4

- 条带化：一个字节块可能存放在多个数据盘上
  - 优点：并行存取，性能好，磁盘负载均衡
  - 缺点：可靠性、不同IO请求需要排队
- 镜像：数据完全拷贝一份
  - 优点：可靠性
  - 缺点：存储开销
- 校验：数据通过某种运算（异或）得出，用以检验该组数字的正确性
  - 优点：可靠性，快速恢复
  - 缺点：开销

提高I/O速度的主要途径

- 选择性能好的磁盘
- 并行化
- 采用适当的调度算法
- 设置磁盘高速缓冲区

文件系统

- 文件是指一组带标识（文件名）的、在逻辑上有完整意义的信息项的序列
  - 信息项：构成文件内容的基本单位（单个字节，或多个字节）
- 文件包含两部分：
  - 文件体：文件本身的内容
  - 文件说明：文件存储和管理的相关信息

- 文件的逻辑结构：
  - 流式文件结构：构成文件的基本单位是字符，文件是有逻辑意义，无结构的一串字符的集合
  - 记录式文件结构：文件由若干记录组成，可以按记录进行读写、查找等操作，每个记录有自己的内部结构。按照逻辑结构分为：顺序文件（链式，顺序存储），索引文件，索引顺序文件
  - 树形结构

- **目录是由文件说明索引组成的用于文件检索的特殊文件**
- 目录内容：目录的内容是文件属性信息:
    - 基本信息：文件名，别名数目
    - 文件类型：多种划分方式
    - 地址信息：文件存放位置，文件长度
    - 访问控制信息：文件所有者，权限
    - 使用信息：创建时间，最后一次读/写时间和用户
- 目录分类：单级文件目录，二级文件目录，多级文件目录
    - 一级文件目录：根目录下就是文件
    - 二级文件目录：根目录下增加每个用户的目录
    - 多级目录
- 文件系统：
    - 定义：操作系统中与文件管理有关的软件和被管理的文件以及实施管理所需要的数据结构的总体
    - 目的：为系统管理者和用户提供对文件的透明存取
- 文件系统模型的三个层次：
    - 文件系统的接口
    - 对象操作管理的软件集合
    - 对象及其属性

- 文件控制块与文件属性：
    - 文件控制块(FCB)：保存管理文件所需的所有有关信息
    - 信息：
        - 基本信息：文件名，物理位置，文件逻辑结构（记录，流式），文件物理结构（顺序，索引）
        - 访问控制信息：文件所有者，访问权限
        - 使用信息：创建时间，上次修改时间，当前使用信息
- 文件物理结构：文件在存储介质上的存储方式。**连续结构，索引结构，串联结构**
    - 连续结构：
        - 优点：
            - 结构简单，易于实现，不需要空间
            - 支持顺序存取和随机存取，存取速度快
            - 连续存取时速度较快
        - 缺点：
            - 文件长度一经固定，不易改变
            - 不利于文件的动态增加和修改
        - **适用于变化不大的顺序访问的文件**
    - 串联（链接）文件结构：串联文件结构是按顺序由串联的块组成的，即文件的信息按存储介质的物理特性存于若干块中。每个物理块的最末一个字(或第一个字)作为链接字，它指出后继块的物理地址。链首指针存放在该文件目录中。文件的结尾块的指针为“^”，表示文件至本块结束。
        - 优点：
            - 空间利用率高
            - 文件动态扩充和修改容易
            - 顺序存取效率高
        - 缺点：
            - 随机存取效率太低

        - 可靠性差
        - 指针占空间
    - 索引结构：一个文件的信息存放在若干个不连续物理块中。系统为每个文件建立一个专用数据结构：索引表，并将这些物理块的块号存放在该索引中。

- 索引文件：系统为每个文件建立逻辑块号与物理块号的对照表，称为文件的索引表。文件由数据文件和索引表构成。这种文件称为索引文件。
    - 索引表位置：文件目录中，文件的开头
    - 索引表大小：固定，非固定
- 索引文件在存储区占两个区：索引区和数据区
    - 访问索引文件需要两步：
        - 查文件索引号
        - 由物理块号获得信息
- 索引表的组织：
    - 链接模式：一个盘块一个索引表，多个索引表链接起来
    - 多级索引：将一个大文件的所有索引表地址放在一个索引表中
    - 综合模式：直接索引和间接索引结合