操作系统

一、概论

1.OS的定义

控制和管理计算机的**软/硬件资源**，合理**组织计算机工作流程**，方便用户使用的程序集合。

```
软硬件资源管理者
用户使用软/硬件的接口
系统本身是扩展机/虚拟机
```

目标：方便，有效，可扩充，开放

2.分时系统特征：

多路性，交互性，独立性，及时性

3.OS功能

处理机管理（进程），存储管理，设备管理，文件管理，用户接口

4.OS特征：

（1）**并发**：多个事件在同一时间间隔发生

（2）**共享**：资源可供多个并发进程使用（互斥共享/同时访问）

（3）虚拟：将物理实体映射为逻辑实体

（4）异步：进程执行的顺序，时间，速度不可知

并发和共享是最基本特征，且互为条件

5.OS结构设计

（1）模块化OS：

优：灵活，效率高

劣：复杂，不利于修改

（2）分层式OS：

优：高层对低层单向依赖；便于扩充

劣：运行效率低

（3）微内核结构：

C/S模式、对象模式

二、作业与接口

1.接口：

作业级接口、程序级接口、命令级接口

2.作业管理

1.概念：

用户在一次事务处理种要求OS做的工作的总称

2.作业组成：

数据，程序，作业说明书

三、进程

1.概念：

具有独立功能的程序在某个数据集合上的一次运行活动，是系统进行分配资源和调度的独立单位

2.特征：

- 动态性
- 独立性：具有自己的独立PCB
- 异步性
- 并发性
- 交互性

3.程序与进程之间的区别

（1）程序是静态的，进程是动态的

（2）进程=程序+数据+PCB

（3）进程和程序彼此都是多对多的关系

（4）进程是暂时的，程序是永久的

4.PCB

① 概念：

系统感知进程存在的唯一标志，系统设置的专门的数据结构

进程与PCB一一对应

② 内容：

- 进程标识符
- 进程控制信息
- 进程调度信息
- 处理机状态

```
进程描述信息：
    进程标识符，唯一的整数
    进程名
    用户标识符
进程控制信息
所拥有的资源和使用情况
CPU现场保护结构
```

为什么说PCB是进程存在的唯一标志：

1.包含了进程的描述信息和控制信息

2.是进程的动态特征的集中反映

3.系统根据PCB感知某一进程的存在

③ PCB表组织方式：

链表，索引表，线性表

④ 进程的状态：

- **运行状态**：正在CPU上运行
- **就绪状态**：无CPU可以运行，但可以立即调度给其他CPU运行
- **阻塞状态**：无法运行
- 创建状态：进程刚创建，但还不能运行
- 结束状态：进程已结束运行
- 挂起状态：换到外存

    阻塞挂起：进程在外存并等待某事件的出现

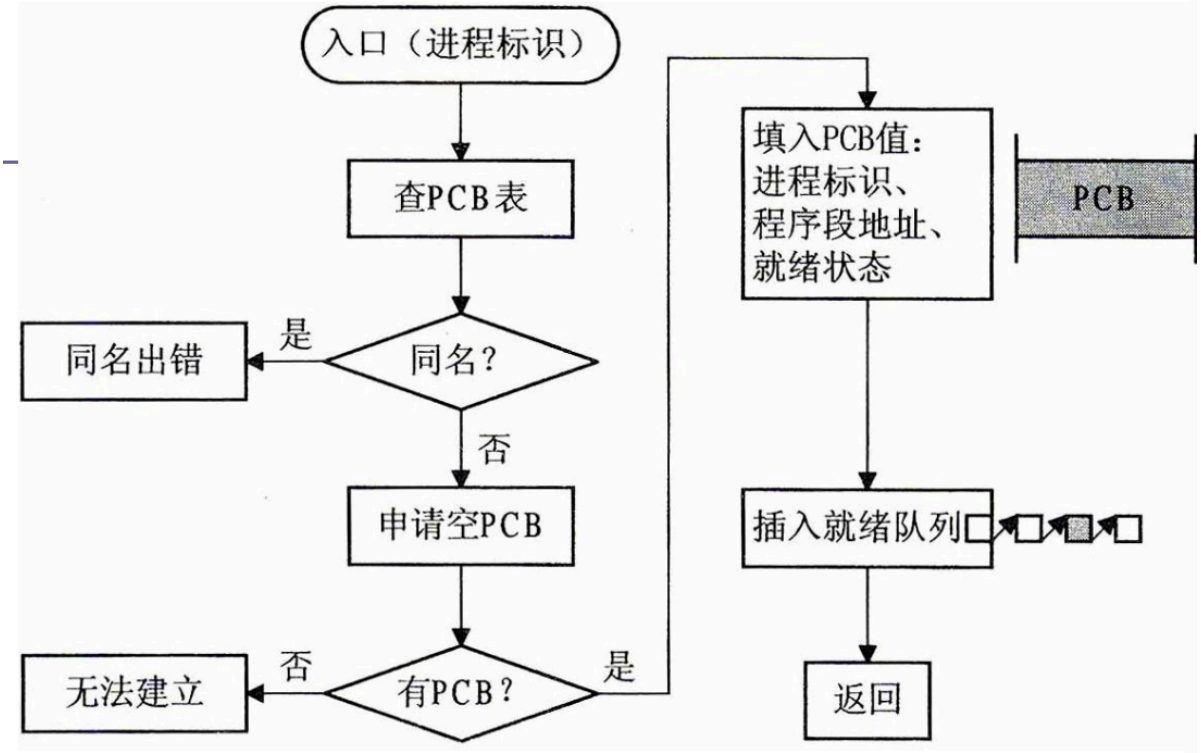
    就绪挂起：进程在外存，但只要进入内存即可运行

5.进程控制

OS通过原语进行控制，执行期间不能中断

```
进程创建原语
进程撤销原语
进程状态转换原语
```

① 进程的建立



创建事件：

用户登陆，作业调度，提供服务，应用请求

创建步骤：

1. 申请空白PCB，获得内部标识
2. 填写PCB参数
3. 初始化PCB：设置状态为就绪，将PCB插入到就绪队列中

② 进程的撤销中止

创建事件：

正常结束，异常结束，外界干预

撤销进程策略：

- 撤销指定进程
- 撤销该进程及其所有子孙进程

过程：

1. 找到相应PCB
2. 若处于执行状态，则停止并设置重新调度标志
3. 撤销所有子孙进程
4. 释放被撤销进程的资源
5. 释放PCB

6. if调度标志为真->重新调度

③阻塞与唤醒

进程阻塞

过程：

找到相应进程的PCB

if 执行->保护现场，改变状态为等待，停止运行，将PCB插入等待队列

if 就绪->修改状态为等待，移除就绪队列，加入等待队列

进程唤醒

过程：

1. 在等待队列中寻找相应PCB，移除队列
2. 设置状态为就绪，将PCB插入就绪队列
3. 等待调度

6.进程调度

目标：高CPU利用率，大吞吐量，快响应时间，公平平衡，策略强制执行

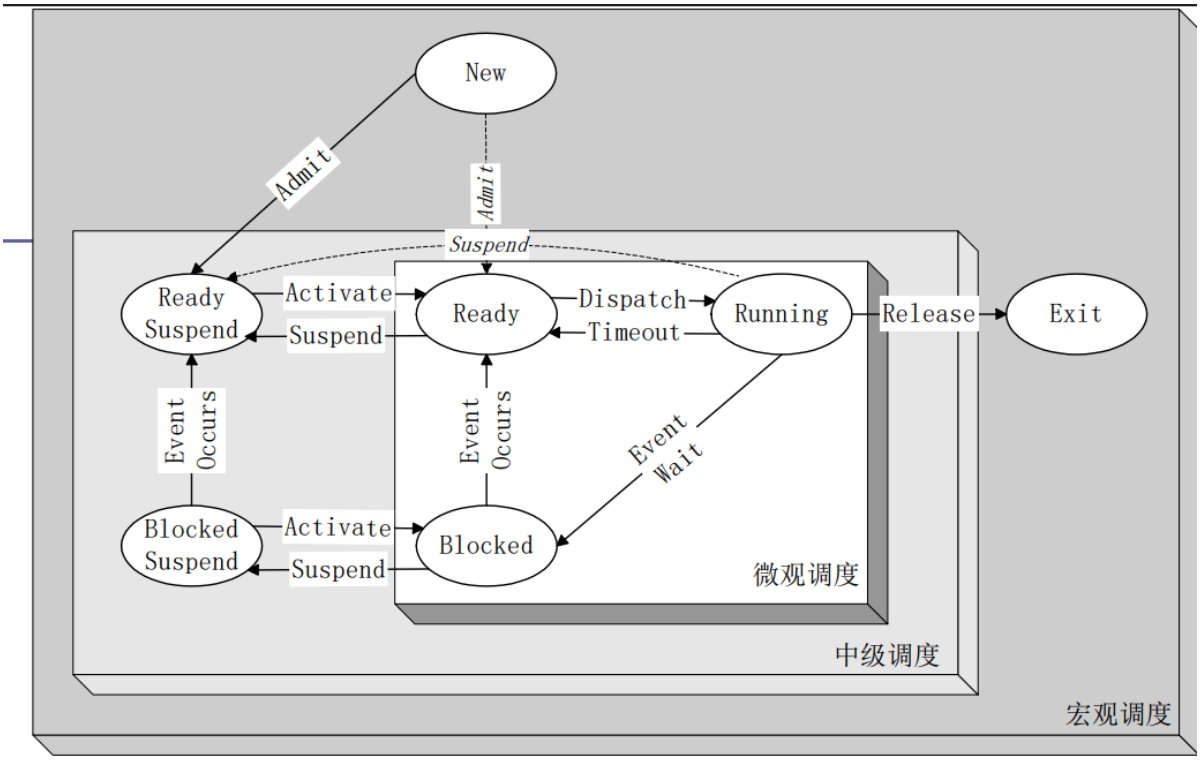
任务：协调进程对CPU的竞争

1.类型：

高级调度：作业调度

中级调度：内外存

低级调度：进程调度



2.性能准则

2.1面向用户

- 周转时间：作业从提交到完成的时间
- 公平性：不因为作业或进程本身的特性使得时间变长
- 截止时间
- 响应时间

2.2面向系统

- 吞吐量：单位时间内完成的作业数
- 处理机利用率
- 各种设备的均衡利用

2.3调度算法的调度性能准则

3.调度时机

- 运行结束
- 时间片到
- 进程提出I/O请求
- 执行原语操作
- 有更高级的进程进入就绪队列

4.调度方式

- 非剥夺调度
- 剥夺调度

5.调度算法

先来先服务FCFS

有利于长作业，CPU繁忙的作业，简单

不利于I/O繁忙

短作业优先SJF

平均周转时间短，系统吞吐量高

对长作业不利，调度性能受影响，高优先级任务可能一直无法执行

- 最短剩余时间优先SRT：允许当前进程剩余时间更短的进程来抢占
- 最高响应比优先HRRN：响应比R=(等待+要求执行)/要求执行时间

时间片轮转算法RR

进程按照FCFS排列，给每一个进程固定时间t执行

响应时间T=N进程数目*q时间片

基于优先级的调度算法

为每一个进程设置一个优先数，每次选择优先级最高的运行

优先级类型：

静态优先级

动态优先级

多级队列算法

多级反馈队列算法

特点：

短作业优先

I/O进程优先

运算行进程有较长的时间片

6.实时调度

①实现条件：

1. 提供必要的信息
2. 可调度的实时系统

- 给定m个周期性事件(periodic events)
- 事件i的周期为P_i，需要的处理时间为 C_i
- 可调度的标准(schedulable)

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

ec3

3. 采用抢占式调度机制
4. 具有快速切换机制

②算法分类

- 非抢占式调度算法

  非抢占式轮转调度算法

  非抢占式优先调度算法

- 抢占式调度算法

  基于时钟中断

  立即抢占

③优先级反转：

高优先级的进程所需的资源被低优先级进程占用导致的阻塞

解决方式：

```
优先级继承：
所有使用到高优先级进程所需资源的进程，继承高优先级直到用完竞争资源，再回到原来的优先级。
```

④ 常用的实时调度算法

1. 静态表调度算法
2. 静态优先级调度算法
3. 动态分析调度算法
4. 无保障动态调度算法
5. 最早截止期优先算法（EDF）——非抢占调度
6. 最低松弛度优先算法（LLF）——抢占调度

7.多处理机调度

①与单处理机区别：

1. 注重整体运行效率
2. 调度算法更多样
3. 调度单位——线程
4. 访问OS数据结构时进行互斥访问控制

②对称式多处理系统SMP

集中控制

- 静态分配（每个CPU设置一个序列）
- 动态分配（采用一个公共就绪序列，那个CPU空闲就分配到哪个CPU上）

分散控制

- 自调度（每个处理机选择合适的进程执行）——简单、处理机利用率高

③非对称式处理系统ASMP

主-从处理机系统，主处理机管理公共就绪队列，分配给从处理机，每个处理机有固定分工

④成组调度

将一个进程中的一组线程，每次分派时同时到一组处理机上执行

优点：提高线程执行并行度，减少阻塞，提高系统吞吐量；减少调度次数和算法开销

⑤专用处理机调度

为进程的每一个线程分配固定CPU

缺点：线程阻塞时，造成CPU的闲置

优点：线程执行时不需切换，减小开销，推进速度更快

7.UNIX进程

①两个特殊进程：

0#进程：内核程序创建，负责进程的调度与对换

1#进程：所有用户的祖先进程

②UNIX的PCB：

四部分：

- proc结构
- User结构
- 系统区表
- 本进程区表（pprt）

③进程映像组成：

- 用户级上下文
- 寄存器级上下文
- 系统级上下文

④调度算法：动态优先级多级反馈循环调度算法

1. 给进程分配时间片
2. 时间片结束时计算优先级
3. 用优先数作比较选出高优先级的进程
4. 调度高优先级进程开始运行
5. 被抢夺了CPU的进程反馈到相应的优先队列中

⑤计算公式

$$P_pri = p_CPU/2 + PUSER + P_nice + NZERO$$

PUSER，NZERO—是优先数基值；

Pcpu—是每个进程最近一次使用CPU的时间；

Pnice—是用户设置的进程优先数偏移值。

8.线程及其管理

1.线程的概念：TCB

进程内一个相对独立的、可调度的执行单元，是CPU的一个调度单位

进程时资源分配单位，线程是CPU调度单位

2.与进程的区别：

```
进程（不是可执行实体）：
    资源分配单位和CPU调度单位
    自己拥有资源（彼此之间资源独立）
    被调度分派在处理器上运行的单元体
    时空开销大，并发度限制高
    具有就绪阻塞执行三种状态
线程（可执行实体）：
    CPU调度单位
    具有就绪阻塞执行三种状态
    只拥有必不可少的资源
    并发时空开销少
    可通过增加线程提高并发度
————————————————————————————————————————————————————————
引入线程的好处
    创建新线程，中止线程花费时间少
    两个线程切换花费时间少，上下文切换时间快
    线程间通信方便
```

总结:

调度（线程切换/创建时间），资源，时空开销，并发性

3.多线程机制

- 多个线程服务于一个进程，并互相共享进程资源、数据和文件
- 线程可修改进程中数据项，多个线程修改时仅第一个拥有修改权限，其他有只读权限
- 地址空间相同

4.多进程OS

①属性：

- 资源分配的单位
- 拥有多个线程
- 不是可执行实体

②实现方式

- **内核线程KLT**

  内核维护

  优点：核心可调度同一进程的多个线程；阻塞在线程一级，彼此之间不影响；可以在多个处理器上运行同一进程的不同线程

  缺点：速度慢

- **用户线程ULT**

  无需内核维护，由应用进程完成

  优点：可运行在任何操作系统上；无需核心

  缺点：容易阻塞；同一个进程的线程只能在一个处理器上运行

5.线程的状态

①状态：

- 执行
- 就绪
- 阻塞

②线程的创建与中止

创建：

先执行初始化线程（1个）——创建若干线程

中止

- 线程结束后退出
- 运行错误等强行终止

9.作业、程序、进程、线程

- 作业：用户向计算机提交的任务实体
- 程序：有序指令集合
- 进程：系统分配资源的基本单位
- 线程：处理器调度的基本单位

10.进程间的通讯

1.进程间关系

进程之间相互依赖又相互制约——由于信息传递和进程间资源共享的占用

这种关系表现为同步和互斥

2.进程的同步

概念：

假设有两个进程，一个进程到达了一个点，只有当另一个进程完成某些操作后才能继续进行下一步的运行（进程的执行顺序的规定）

实现：

- 等待：**wait（消息名）**，等待到消息名为**true**时继续执行
- 发送：**signal（消息名）**，向合作的进程发送需要的消息名为true

3.进程的互斥

概念：

多个进程不能使用同一个资源

4.同步互斥的关系

11.临界资源与临界区

临界资源：

一次仅允许一个进程访问

临界区：

每个程序中访问临界资源的一段代码

- 设计临界段的原则：
    1. 每次至多一个进程处于临界段
    2. 有限时间内只让一个进程进入临界段
    3. 进程在临界段内的时间有限

```
进入区
  |
临界区
  |
退出区
```

12.同步机制遵循的准则

- 空闲让进
- 忙则等待
- 有限等待
- 让权等待

13.软件方法解决互斥

（1）软件

实现过于复杂，需要较高的编程技巧

无需硬件、OS和程序设计语言的支持

处理开销大, 容易出错

Peterson算法：先修改、后检查、后修改者等待

（2）硬件

1.中断禁用

使用关中断原语、开中断原语

关中断：进程在临界区执行时不响应中断

```
关中断原语
    |
  临界区
    |
开中断原语
```

缺点：

- 代价高：限制了处理器交替执行各进程的能力
- 不能用于多处理器结构：关中断不能保证互斥

2.专门的机器指令

Test-and-Set指令

为每个临界资源设置一个全局布尔变量lock，并赋初值false，表示资源空闲。在进入区利用TS进行检查：有进程在临界区时，重复检查；直到其它进程退出时，检查通过；当进程进入后会将lock值设为TRUE表示资源正在被使用，使用完后会将其设为false。

swap指令

为每个临界资源设置一个全局布尔变量lock，其初值为false，在每个进程中再利用一 个局部变量key，初值为true，重复交换key与lock的值，直到key的值为false后代表可以占用临界资源。

3.硬件的优缺点

缺点：

- 等待要耗费CPU时间，不能实现"让权等待"
- 可能导致"饥饿"，有的进程可能一直无法进入临界区
- 可能死锁。

（3）锁

0表示资源可用，1表示资源占用，用户通过原语进行操作。

- lock（w）：上锁
- unlock（w）：开锁

（4）进程的交互关系

- 互斥：多个进程不能同时使用同一个资源
- 死锁：多个进程互不相让
- 饥饿：一个进程一直都得不到资源

14.信号量和P、V原语

1.信号量

一种数据结构，由**整形变量V，指针变量S**构成

- 初始化指定一个非负整数值——空闲资源总数
- 非负值——空闲资源数；负值——等待临界区进程数
- 信号量的值只能被P、V原语改变

①信号量定义

- 必须置一次且只能设置一次初值
- 初值应该大于等于零
- 只能执行P、V操作

```
struct semaphore
{
    int value;
    pointer_PCB queue;
}
//声明
semaphore s;
```

②物理意义

P(s)：申请一个资源，**wait（消息名）**，等待到消息名为**true**时继续执行

V(s)：释放一个资源，**signal（消息名）**，向合作的进程发送需要的消息名为true

```
s>0：有空闲资源
s<0：无空闲资源，处于等待队列
s=0：无资源可用
```

初始化：将Value设置成一个非负整数

只能通过初始化和P、V操作来对信号量进行操作

P、V操作必须成对出现：互斥操作时处于同一进程（实现互斥时要将s设置成1）；同步操作时处于不同进程；同步P操作一定在互斥P操作之前

③优缺点

优点：简单，表达能力强

缺点：容易出现死锁，解决复杂同步问题时不方便，不安全

15.经典的进程同步问题

OSL9

16.进程通信

概念：指进程之间可**直接以较高的效率传递较多数据的信息交换方式**(进程间进行信息交换)

1.进程通信类型：

①共享存储器系统

- 基于共享数据结构
- 基于共享存储区

同步互斥由进程自己负责

②消息传递系统

消息：一段文本

进程间的数据交换以**消息**为单位，程序员利用系统的**通信原语**实现通信

- 直接通信（消息缓冲通信）：传递数据块
- 间接通信（信箱）

③管道通信——pipe（半双工通信，双边单向）

字符流方式写入读出

先进先出顺序

管道：连接一个读进程和一个写进程的文件——pipe文件

17.死锁

①资源：

物理资源：内存、CPU、I/O设备

逻辑资源：文件、信号量

- 可重用资源：一次只能供一个进程安全的使用，不会因为使用耗尽
- 可消费资源：可以创建可以销毁，无数目限制

②死锁的定义

一组进程中，**每个**进程都**无限等待**被该组进程中另一进程所占有的资源，因而**永远无法得到资源**

③死锁的产生原因

- 资源不足导致的资源竞争
- 并发执行顺序不当

④死锁的必要条件

- 互斥：在一段时间内某资源只能由一个进程占有
- 请求和保持条件：占有一个资源，同时又请求新的被其他进程占有的资源，并且不释放已经占有的资源
- 不可抢占：资源只有在使用完后才能被释放
- 循环等待

⑤处理方法

1. **忽略此问题不做任何处理**
2. **不允许出现死锁**
    - 死锁预防：限制如何**申请**资源

      方式：在系统设计时确定资源分配算法

      做法：破坏必要条件之一
    - 死锁避免：对于一个申请，进程是否应该等待

确保安全状态——找到安全序列（可以不唯一）

安全状态一定没有死锁，不安全状态不一定有死锁

3. **允许系统出现死锁后排除**
   - 死锁检测
   - 死锁恢复
4. **银行家算法**

Ⅰ.设置数据结构

- Available[M]：每一类资源可利用的数目
- Max[i,j]：进程i需要j类资源最大数目为Max[i,j]
- Allocation[i,j]：i已经分得j类资源的数目为Allocation[i,j]
- Need[i,j]：进程i还需要j资源Need[i,j]个
- $$Request_i[j]：需要Request_i[j]个j资源$$

$$Need[i,j]+Allocation[i,j]=Max[i,j]$$

Ⅱ.算法过程

```
step1：
    if(Request_[j]≤Need[i,j])
        goto step2
step2:
    if(Request_i[j]≤Available[j])
        goto step3
    else  等待
step3:
    Available[j]=Available[j]-Request_i[j];
    Allocation[i,j]=Allocation[i,j]+Request_i[j];
    Need[i,j]=Need[i,j]-Request_i[j];
step4:
    执行安全性算法，若系统处于安全状态，将资源分配给P_i
```

```
安全性算法：
step1：设置两个工作向量
    Work：
        表示系统可提供给进程继续运行所需的各类资源数目，含有m个元素
        在执行安全算法开始时，Work=Available
    Finish：
        表示系统是否有足够的资源分配给进程，使之运行完成。
        开始时先做Finish[i]=false
        当有足够资源分配给进程时,Finish[i]=true
step2：
    在进程中寻找：
        ①Finish[i]=false;
        ②Need[i,j]≤Work[j];
    找到了，goto step3;else goto step4
step3:
    work[j]=work[i]+Allocation[i,j];
    Finish[i]=true;
    goto step2;
```

```
step4:
    判断出处于安全状态
```

Ⅲ.优缺点：

优点：

- 限制少
- 非剥夺

缺点：

- 必须事先声明每个进程请求的最大资源
- 考虑的进程必须是无关的
- 进程的数量保持固定不变，且分配的资源数目必须是固定的
- 在占有资源时，进程不能退出

5. **死锁检测**
   - 资源分配图中没有环路——无死锁

- 死锁状态的充分条件：当且仅当资源分配图是不可完全简化的
- **资源分配图的化简**：

  资源类（资源的不同类型）：用方框表示

  资源实例（每个资源类中）：用方框中的黑圆点（圈）表示

  进程：用圆圈中加进程名表示

  分配边：资源实例→进程的一条有向边

  申请边：进程→资源类的一条有向边

```
找一个非孤立点进程结点且只有分配边，去掉分配边，将其变为孤立结点
再把相应的资源分配给一个等待该资源的进程，即将某进程的申请边变为分配边
重复以上步骤，若所有进程成为孤立结点，称该图是可完全简化的，否则称该图是不可完全简化的。
```

6. **死锁解除**
   1. 重新启动
   2. 撤销进程
   3. 剥夺资源
   4. 进程回退

四、存储管理

内存

- 系统区：存放操作系统
- 用户区：存放用户数据

存储管理的目的：

- 充分利用内存
- 方便用户使用
- 解决程序空间比实际内存空间大的问题

1.存储系统组织与四大功能

Ⅰ.组织

- 高速缓存Cache：少量，昂贵，快速，易变
- 内存RAM：全部中等，一边
- 磁盘：大量，链家，低速，不易变

Ⅱ.四大功能

1. **存储空间的管理，分配和回收**
2. **地址再定位**

   ```
   名空间->地址空间->存储空间
   ```

   构成：名空间，物理地址（可直接寻址），逻辑地址（通过**地址再定位**转换到物理地址）

   再定位方法：动态、静态
3. **存储共享和保护（多个进程共用内存相同区域）**
4. **存储器扩充**

2.分区存储管理

分区方式：

- 固定分区
- 动态分区
- 分区分配算法

①固定分区

将内存划分为若干个固定大小的分区

优点：易于实现，开销小

缺点：限制并发执行程序数目，分区总数固定；存在内碎片

数据结构：分区表

②动态分区

在装入程序时按照其初始要求分配，在执行过程中通过系统调用改变分区大小

优点：无内碎片

缺点：有外碎片

③动态分区分配算法

- 分区分配算法：空闲分区大于等于程序奥求，分割成两部分，一部分给程序占用，空一部分“**空闲**”，从内存低端到高端
- 分区释放算法：将相邻空闲分区合并成一个空闲分区

```
First Fit（FF最先匹配法）
    按照分区先后次序，从头查找
Next Fit（NF下次匹配法）
    按照分区先后次序，从上次分配的分区开始查找
Best Fit（BF最佳匹配法）
    找到空闲分区与程序要求相差最小的分区
Worst Fit（WF最坏匹配法）
    找最大的空闲分区
```

碎片问题的解决

- 多重分区
- 重定位分配：集中碎片

3.覆盖与交换

Ⅰ.覆盖技术

①原理

一个进程的几个代码段，按照时间先后占用公共的内存空间

②缺点：

编程复杂度增加；编译时间增长

Ⅱ.交换技术

①原理

暂停进程，将进程地址全部保存到外存交换区，将外存中就绪的进程读入到内存并送进就绪队列（swap in/out）

②优点：

- 增加并发运行程序数目
- 编写程序时不影响程序结构

③缺点

- 增加处理机开销
- 没有考虑地址访问的统计特性

Ⅲ.对比

1. **相同点**

   进程的程序和数据主要放在外存，当前需要执行的部分放在内存，内外存之间进行信 息交换

2. **不同点**

   - 交换发生在进程或作业之间，而覆盖发生在同一进程或作业内
   - 与覆盖技术相比，交换技术不要求用户给出程序段之间的**逻辑覆盖结构**

4.分页存储管理

Ⅰ.基本思想

- 主存分成多个固定大小的内存块
- 作业按照主存块大小分页
- 连续的页存放在离散的块中，逻辑上相邻的页，物理上不一定相邻

Ⅱ.页表

登记页号和块的对应关系——一种数据结构（包含**页号，块号**，其他...）

系统为每个进程建立一个页表，页表的长度和首地址存放在该进程的进程控制块（PCB）中

Ⅲ.快表（Cache的页表）

页号，内存块号，标识位，淘汰位

内存的页表称为慢表

5.虚拟存储器解决内存小，作业大，作业多的矛盾

定义：具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的存储系统

Ⅰ.实现方式

操作系统统一管理各级存储器；内存中只存放当前要执行的程序部分，其余的保存在外存上，OS根据需要随机地将需要的部分对换到内存执行

Ⅱ.物质基础

- 二级存储结构：内存、外存
- 动态地址转换：OS同一编址内外存

Ⅲ.虚空间大小

- 逻辑大小：可寻址范围
- 实际大小：内外存兑换区

Ⅳ.好处

- 大程序
- 大用户空间
- 高并发
- 易于开发

Ⅴ.技术特征

- 不连续性
- 部分交换
- 虚拟扩充
- 多次对换

Ⅵ.虚拟管理三大策略

- 调入策略——把哪部分装入主存
- 放置策略——放在主存什么地方
- 淘汰策略——主存不足时，把哪部分淘汰出主存

6.请求分页存储管理

Ⅰ.基本原理

目标：实现小内存大作业

实现方法：作业运行时，只将当前的一部分装入内存，其余的放在辅存，一旦发现访问的页不在主存中， 则发出缺页中断，由OS将其从辅存调入主存，如果内存无空块，则选择一个页淘汰（根据需求载入所需页面）

基本问题

- 怎样发现页不在内存

  扩充页表，增加中断位（0—在内存，1—不在内存）、辅存位置

- 如何处理缺页

  增加引用位（0-未访问过，1-访问过）、修改位（0-写回辅存、1-不写回辅存）

Ⅱ.页表表项

页号、中断位、内存块号、外存地址、访问位（选择淘汰哪一页）、修改位

Ⅲ.缺页中断处理

页表中访问的页不在内存，即产生缺页中断

发生缺页中断的处理：

1. OS接收中断信号，调出缺页中断处理程序
2. 将缺页进程挂起
3. if 有空闲块——>分配空闲块

   else ——>淘汰这一页，写回外存

Ⅳ.页面置换算法

目的：选择内存中哪个物理页面被调换

页面锁定：必须常驻内存的进程

实现：在页表中加上锁定标志位

常见的页面置换算法：

- 先进先出（FIFO）——缺页率75%
- 最近最久未使用置换算法（LRU）——缺页率60%
- 最佳页面算法（OPT）——缺页率最低
- 最不经常使用（LFU）——缺页率75%

影响缺页次数的因素：

- 分配给进程的物理块数目
- 页的大小
- 程序编程方法
- 页面淘汰算法

Ⅴ.常驻集

虚拟页式管理中给进程分配的物理页面数目

常驻集越小，缺页率越高；常驻集大到一定数目时，缺页率不再明显下降

Ⅵ.缺页中断率

缺页中断率=F/（S+F）=失败访问次数/成功+失败

Ⅶ.请求分页管理方案的评价

优点：

- 提供了虚存管理方式，作业地址空间不再受实存容量的限制
- 更有效的利用了主存，方便于多道程序运行，方便了用户

缺点：

- 为处理缺页中断，增加了处理机时间的开销，用**时间的代价换取了空间的扩大**
- 可能因作业地址空间过大或程序数目过多等**造成系统抖动**，为此采取措施会增加的系统的复杂度。

7.分段存储管理

Ⅰ.基本思想

- 用户程序划分：按程序自身的逻辑关系划分为若干个程序段，每个程序段都有一个段名，且有一个段号
- 内存划分：内存空间被动态的划分为若干个长度不相同的区域，称为物理段，每个物理段由起始地址和长度确定
- 内存分配：以段为单位分配内存，每一个段在内存中占据连续空间，但各段之间可以不连续存放

数据结构：

- 进程段表：描述组成进程地址空间的各段，每段有段基址和段长度
- 系统段表：系统内所有占用段
- 空闲段表：内存中所有空闲段

Ⅱ.段地址映射

段地址映射的数据结构有**段表、段表首址指针和段表的长度**，段表首址指针和段表长度存放在进程自己的PCB中

每一个进程有一个段表

映射过程：

1. 程序地址字送入虚地址寄存器VR中
2. 取出段号S和段内位移W

3. 根据段表首址指针找到段表，查找段号为S的表目，得到该段的首地址
4. 把段首地址与段内位移相加，形成内存地址送入MR中，并以此地址访问内存

分段保护：

- 越界检查
- 存取控制检查
  - 只读
  - 只执行
  - 读/写
- 环保护机构
  - 一个程序可以访问驻留在相同环或较低特权环中的数据
  - 一个程序可以调用驻留在相同环或较高特权环中的服务

Ⅲ.优缺点

优点：

- 消除了内碎片
- 通过请求分段存储管理方式提供了大量虚存
- 允许动态增加段的长度
- 便于动态装入和链接
- 便于程序共享
- 便于存储保护

缺点：

- 进行地址变换和实现内存靠拢要花费处理机时间
- 在辅存上管理可变长度的段比较困难

分段与分页区别：

	分段	分页
划分	按照逻辑	按照物理空间
地址空间	二维	一维
	面向用户	用户不可见
长度	由用户决定	固定且相等
共享	简单	难

8.段页式存储管理

- **用户程序划分**：按段式划分（对用户来讲，按段的逻辑关系进行划分；对系统讲，按页划分每一段）
- **内存划分**：按页式存储管理方案
- **内存分配**：以页为单位进行分配

地址映射：

- **段表：**记录每一段的页表始址和页表长度
- **页表：**记录了逻辑页号与内存块号的对应关系（每一段有一个，一个程序可能有多个页表）
- **地址变换**：先查段表，再查该段的页表

9.虚拟存储管理其他问题

Ⅰ.虚拟存储的调入策略

- 请求调页(demand paging)：只调入发生缺页时所需的页面
- 预调页(prepaging)：在发生缺页需要调入某页时，一次调入该页以及相邻的几个页

Ⅱ.清除策略

- 请求清除(demand cleaning)：该页被置换时才调出，把清除推迟到最后一刻
- 预清除(precleaning)：该页被置换之前就调出，因而可以成批调出多个页面

Ⅲ.常驻集大小的确定方式

- 固定分配(fixed-allocation)：常驻集大小固定
- 可变分配(variable-allocation)：常驻集大小可变，按照缺页率动态调整

置换范围(replacement scope)：被置换的页面局限在本进程，或允许在其他进程

- 局部置换(local replacement)：容易进行性能分析
- 全局置换(global replacement)：更为简单，容易实现，运行开销小

固定分配不能进行全局置换！

Ⅳ.页面调度策略——了解

- 取页策略：NT采用按进程需要进行的请求取页和按集群方法进行的提前取页
  - 集群方法：在发生缺页时，不仅装入所需的页，而且装入该页附近的一些页
- 置页策略：在线性存储结构中，简单地把装入的页放在未分配的物理页面即可
- 淘汰策略：采用局部FIFO置换算法。在本进程范围内进行局部置换，利用FIFO算法把驻留时间最长的页面淘汰出去

五、文件系统概述

1.基本要求

- 能够存储大量信息
- 长期保存信息
- 能够共享信息

把信息以文件的形式存储在介质上

文件通过OS进行管理

2.文件的定义

文件是赋名的数据项、赋名有关联的信息单位 (记录)的集合

①构成

- 名称
- 标识符
- 类型
- 位置
- 大小
- 保护
- 其他标识
- 文件体

②类型

- 按照性质用途分类：
  - 系统文件
  - 库文件
  - 用户文件
- 存取控制属性
  - 只读
  - 读写
  - 只执行
- 组织形式和处理方式
  - 普通文件
  - 目录文件
  - 特殊文件
- UNIX文件分类
  - 普通文件
  - 目录文件
  - 特殊文件

③结构

逻辑结构

- 无结构（流式）文件
- 有结构（记录式）文件
- 记录式文件按照组织方式：
  - 顺序文件
  - 索引文件
  - 索引顺序文件

物理结构

- **连续结构**：一个逻辑文件的信息存放在存储器上的相邻物理块中
- **链接结构**
  - 隐式：在每个物理块中设置一指针，指向该文件的下一 个物理块号，文件的末尾块存放结束标记“NULL”
  - 显式：FAT（文件分配表）：将盘块中的链接字按盘块号的顺序集中起来，构成文件分配表FAT
- **索引文件**：为文件建立一张索引表，索引 表按记录关键字排序，本身是顺序文件
  - **超级超级超级重要考点**：①要会根据多层索引、混合索引的结构计算出文件的最大长度（Key：各级索引表最大不能超过一个块）；②要能自己分析访问某个数据块所需要的读磁盘次数（Key：FCB中会存有指向顶级索引块的指针，因此可以根据FCB读入顶级索引块。每次读入下一级的索引块都需要一次读磁盘操作。另外，要注意题目条件——顶级索引块是否已调入内存）

		How?	目录项内容	优点	缺点
顺序分配		为文件分配的必须是连续的磁盘块	起始块号、文件长度	顺序存取速度快，支持随机访问	会产生碎片，不利于文件拓展
链接分配	隐式链接	除文件的最后一个盘块之外，每个盘块中都存有指向下一个盘块的指针	起始块号、结束块号	可解决碎片问题，外存利用率高，文件拓展实现方便	只能顺序访问，不能随机访问。
	显式链接	建立一张文件分配表(FAT)，显式记录盘块的先后关系（开机后FAT常驻内存）	起始块号	除了拥有隐式链接的优点之外，还可通过查询内存中的FAT实现随机访问	FAT需要占用一定的存储空间
索引分配		为文件数据块建立索引表。若文件太大，可采用链接方案、多层索引、混合索引	链接方案记录的是第一个索引块的块号，多层/混合索引记录的是顶级索引块的块号	支持随机访问，易于实现文件的拓展	索引表需占用一定的存储空间。访问数据块前需要先读入索引块。若采用链接方案，查找索引块时可能需要很多次读磁盘操作。

④文件存取方法

- 顺序存取
- 直接存取
- 按键存取

3.文件系统

操作系统中负责管理相关文件信息的软件机构

①管理对象：

- 文件
- 目录
- 磁盘空间

②基本功能：

- 文件的结构及有关存取方法
- 文件的目录结构和有关处理
- 文件存储空间的管理
- 文件的共享和存取控制
- 文件操作和使用

③功能模块：

- 文件的分块存储
- 文件定位
- 外存存储空间管理
- 外存设备访问和控制
- I/O缓冲和调度

④接口

- 命令接口
- 程序接口

4.文件目录

①目标：

能方便而迅速地对目录进行检索， 从而准确地找到所需文件

②**文件控制块FCB：**

是操作系统为管理文件而设置的数据结构， 存放了为管理文件所需的所有有关信息

文件控制块与文件一一对应，是**文件存在的标志**

内容

- 基本信息
    - 文件名
    - 物理位置
    - 逻辑结构
    - 物理结构

- 存取控制信息
- 使用信息

③文件目录

把所有FCB组织在一起，即文件控制块的有序集合

目录项：构成文件目录的项目（一个FCB）

目录文件：保存管理目录的文件

④简单目录

1. 一级目录：整个目录组织是一个线性结构，系统中的 所有文件都建立在一张目录表中
    - 优点：
        - 结构简单、 清晰, 便于维护和查找
        - 可实现按名存取
    - 缺点：
        - 查找速度慢
        - 不允许重名：文件名和文件体有一一对应关系
        - 不便于实现文件共享
2. 二级目录：在根目录下，每个用户对应一个目录
    - 优点：
        - 提高了检索目录的速度
        - 在不同的用户目录中， 可以使用相同的文件名
        - 不同用户还可使用不同的文件名来访问系统中 的同一个共享文件
    - 缺点：
        - 缺乏灵活性，特别是不能反映现实世界 中多层次的关系
3. 多级目录：多级目录结构由根目录和各级目录组成，为管理上的方便，除根目录外，其它各级目录均以文件的形式组 成目录文件

    各级**目录文件**称**中间结点**，用**方框**表示。**数据文件**称为**叶结点**，用**圆圈**表示

    路径名：从根结点开始，经过一个或多个中间结点，到达某个叶结点的一条路径，称这条路径为文件的路径名，是文件的唯一标识。

    名由根目录和所经过的目录名和文件名以及分隔符组成，通常使用**分隔符 /**
    - 绝对路径：由根目录开始的路径名
    - 相对路径：从当前工作目录开始的路径名
4. 索引节点：改进的多级目录，提高目录检索速度
    - 符号文件目录和基本文件目录共同构成
5. 查找一个文件的平均访盘次数：（N+1）/2次

⑤目录操作

- 创建目录
- 删除目录
  - 不删除非空目录
  - 可删除非空目录：子目录同时被删除
- 改变目录
- 移动目录
- 链接操作
- 查找操作
  - 线性检索法
  - Hash检索法：系统利用用户提供的文件名，将它变换为文件目录的索引值，再利用该索引值到 目录中去查找

5.文件系统的使用

1. 建立文件

   建立文件FCB
2. 删除文件
3. 打开文件
4. 关闭文件
5. 指针定位
6. 读文件
7. 写文件

6.文件存储空间管理

典型的存储介质：磁盘，光盘，磁带，U盘

访盘请求：

- 寻道
- 旋转延迟
- 数据传输

①存储空间分配

- 预分配：创建时(这时已知文件长度) 一次分配指定的存储空间
- 动态分配：需要存储空间时 才分配（创建时无法确定文件长度）

②存储单位：簇

③文件存储分配的数据结构

- 连续分配：只需记录第一个簇的位置，适用于预分配方法，通过紧缩(compact)将外存空闲空间合并成连续的区域

- 链式分配：在每个簇中有指向下一个簇的指针。 可以通过合并(consolidation)将一个文件的各个簇连续存放
- 索引分配：文件的第一个簇中记录了该文件的其他簇的位置

④空闲空间管理

1. **空白文件目录**

   辅存上的一片连续的空闲区, 可视为一个空白文件, 系统设置一张空白文件目录来记录辅存上所有空闲块的信息。每个表目存放一个空白文件的信息, 包括该空白文件第一个空闲块号、 空闲块个数、该文件所有空闲块号等信息，适用于**连续文件结构**

   - 缺点：
     1. 如果文件太大, 那么在空白文件目录中将没 有合适的空白文件能分配给
     2. 经过多次分配和回收, 空白文件目录中的小 空白文件越来越多, 很难分配出去, 形成碎片

2. **空白块链**

   把所有的空闲块链接在一起, 形成一个空闲块链表。释放和分配空白快都可以在链首处进行。

   优缺点：

   - 可实现不连续分配
   - 节省了存储开销
   - 系统开销大
   - 对于大型文件系统, 空闲链将会太长

3. **位视图**

   用一串二进制位反映磁盘空间分配使用情况, 每个物理块对应一位, 分配物理块为1，否则为0

   申请物理块时，可以在位示图中查找为0的位，返回对应物理块号；归还时；将对应位转置0

   - 磁盘分配：
     1. 顺序扫描位示图，从中找出一个或一组其值为“0”的二进制位(“0”表示空闲)
     2. 将所找到的一个或一组二进制位， 转换成与之相应的盘块号。假定找到的其值为“0”的二进制位，位于位示的第i行、第j列，则其相应的盘块号应按下式计算： b=n(i-1)+j，其中n代表每行的位数
     3. 修改位示图， 令map[i,j]=1

     位示图：每个二进制位对应一个盘块。在本例中，“0”代表盘块空闲，“1”代表盘块已分配。位示图一般用连续的“字”来表示，如本例中一个字的字长是16位，字中的每一位对应一个盘块。因此可以用（字号，位号）对应一个盘块号。当然有的题目中也描述为（行号，列号）
     重要重要重要：要能自己推出盘块号与（字号,位号）相互转换的公式。
     注意题目条件：盘块号、字号、位号到底是从0开始还是从1开始
     如本例中盘块号、字号、位号从0开始，若n表示字长，则...

     (字号, 位号)=(i, j) 的二进制位对应的 盘块号 b = ni + j

   - 磁盘回收：
     1. 将回收盘块的盘块号转换成位示图中 的行号和列号。转换公式为：

        i=(b-1)DIV n+1

        j=(b-1)MOD n+1

2. 修改位示图。 令map [i,j] =0

做例题

4. **文件卷**

- 磁盘分区：把一个物理磁盘的存储空间 划分为几个相互独立的部分
- 文件卷：在同一个文件卷中使用同一份管理数据进行文件分配和外存空闲空间管理，而在不同的文件卷中使用相互独立的管理数据
- 格式化：在一个文件卷上建立文件系统

7.文件共享

Ⅰ.定义

文件共享：一个或一部分文件由事先规定的某些用户共同使用

Ⅱ.目的

- 节省存储空间
- 进程间通过文件交换信息

Ⅲ.文件共享的方式

- 同名共享
- 异名共享——使用各自不同的文件名访问一个文件
  - 基于索引点的共享方法：硬链接（ln -d）
  - 基于符号链的共享方法：软链接（ln -s）

六、I/O系统

1.概述

Ⅰ.设备分类

- 按传输速率
  - 低速
  - 中速
  - 高速
- 按信息交换
  - 字符设备
  - 块设备
- 按资源管理方式
  - 独占型
  - 共享型
  - 虚拟设备
- 按外部设备的从属关系
  - 系统设备
  - 用户设备

Ⅱ.设备管理的目标

1. 实现设备的独立性
2. 提高设备的利用率
3. 设备统一管理

Ⅲ.设备管理的功能

- 隐藏I/O细节
- 保证设备无关性
- 对I/O设备进行控制
- 提高处理机和 I/O 设备的利用率
- 确保对设备的正确共享
- 处理错误

Ⅳ.设备管理的数据结构——设备控制块DCB

设备控制块DCB

- 设备名
- 设备属性
- 设备状态
- 设备I/O总线地址
- 等待队列指针

2.硬件特点

Ⅰ.设备组成

- 物理部分：设备本身
- 电子部分：设备控制器

Ⅱ.设备控制器完成的工作

- 接收和识别命令
- 数据交换
- 标志和报告设备的状态
- 地址识别
- 数据缓冲区
- 差错控制

Ⅲ.端口编址

①I/O端口地址

接口电路中每个寄存器具有的、唯一的地址——int

②端口编址方法

- 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
  - 优点
    - 凡是可对存储器操作的指令都可对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
- 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，主机使用专门的I/O指令对端口进行操作
  - 优点
    - 外部设备不占用内存地址空间
    - 易于区分是对内存操作还是对I/O端口操作
  - 缺点
    - 对I/O端口操作的指令类型少，操作不灵活

3.I/O控制方式

Ⅰ.循环查询I/O方式

浪费大量CPU时间

Ⅱ.I/O中断方式

CPU利用率大大提高，但数据缓冲寄存器每满一次都要中断一次，如果设备较多时，中断次数会很多，使CPU的计算时间大大减少

Ⅲ.直接存储器存取方式（DMA）

直接存储器访问

Ⅳ.I/O通道控制方式

在CPU的控制下独立地执行通道程序，对外部设备的I/O操作进行控制，以实现内存与 外设之间成批的数据交换

- 字节多路通道
- 数据选择通道
- 数据多路通道

Ⅴ.DMA与中断方式的主要区别

1. 中断时机
   - 中断方式是在**数据缓冲寄存器满**后，发中断请求， CPU进行中断处理
   - DMA方式则是在所要求传送的**数据块全部传送结束**时要求CPU进行中断处理
2. 数据传输

- 中断方式的数据传送由**CPU**控制完成
- DMA方式是在**DMA控制器**的控制，不经过CPU控制完成的

4.I/O系统的软件组成

把I/O软件组织成多个层次

```
  用户层软件
       |
设备独立性软件
       |
 设备驱动程序
       |
 中断处理程序
       |
      硬件
```

5.缓冲技术

Ⅰ.原因

- 缓和CPU与I/O设备间**速度不匹配**的矛盾
- **减少对CPU的中断频率**，放宽对CPU中断响应时间的限制
- 提高CPU和I/O设备之间的**并行性**
- 解决**数据粒度不匹配**的问题

Ⅱ.缓冲技术

- 硬件缓冲:容量小
- 软件缓冲:由缓冲区和对缓冲区的管理两部分组成
  - 单缓冲
    - **最简单**的一种缓冲形式,当进程发出一I/O请求时，OS为之分配一缓冲区
    - 对于输入：设备先将数据送入缓冲区，OS再将数据传给进程
    - 对于输出：进程先将数据传入缓冲区，OS再将数据送出到设备
    - **先让数据进入缓冲区,再让OS送出**
  - 双缓冲
    - 加快输入输出速度
    - 设置**两个缓冲区buf1和buf2**,读入数据时，首先输入设备向buf1填入数据，然后进程从buf1提取数据,在进程从buf1提取数据的同时，输入设备向buf2中填数据,当buf1取空时，进程又从buf2中提取数据，与此同时输入设备向buf1填数，如此交替使用两个缓冲区，使CPU和设备的并行操作的程度进一步提高。
  - 环形缓冲
    - 将多个大小相等的存储区作为缓冲区，并将这些缓冲区链接起来
  - 缓冲池
    - 由内存中一组大小相等的缓冲区组成属于系统资源，由系统进行管理

    1. 空缓冲区

2. 装满输入数据

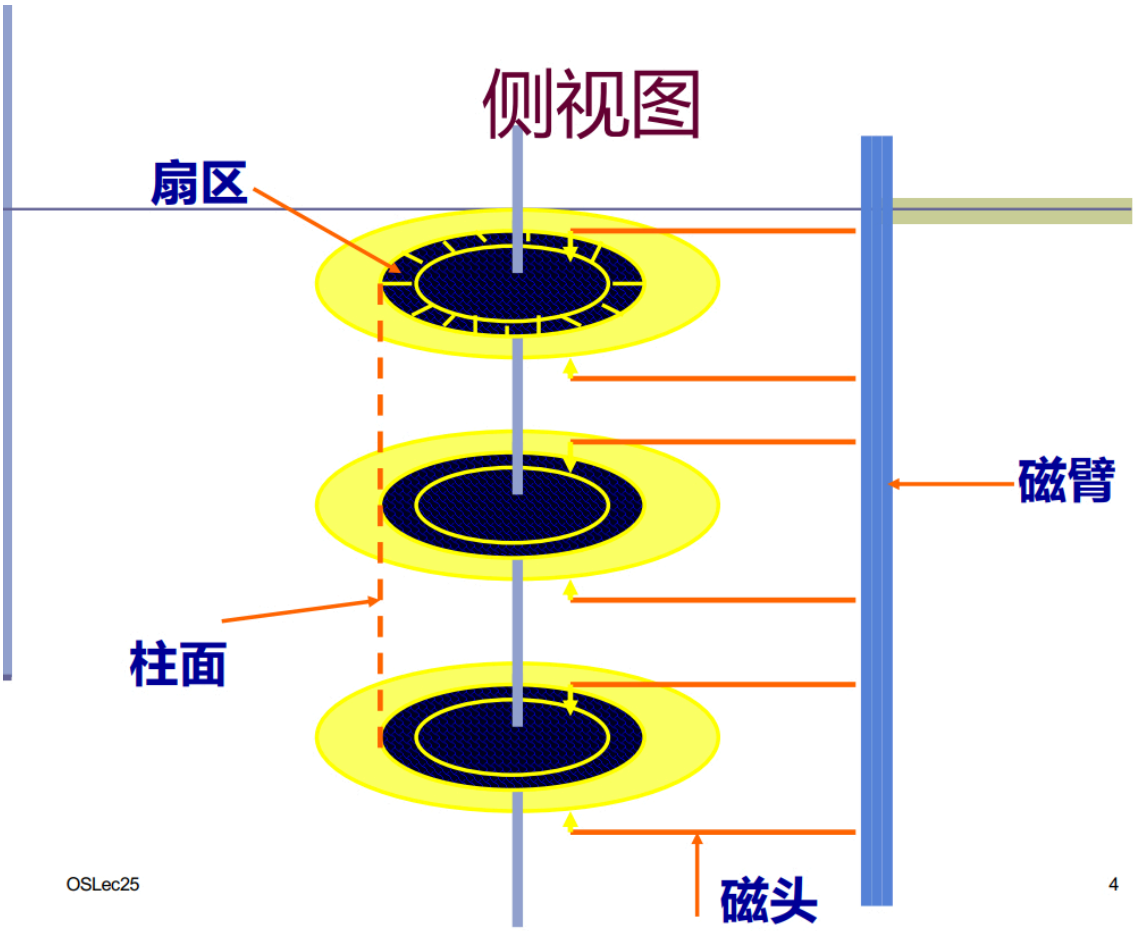
3. 装满输出数据

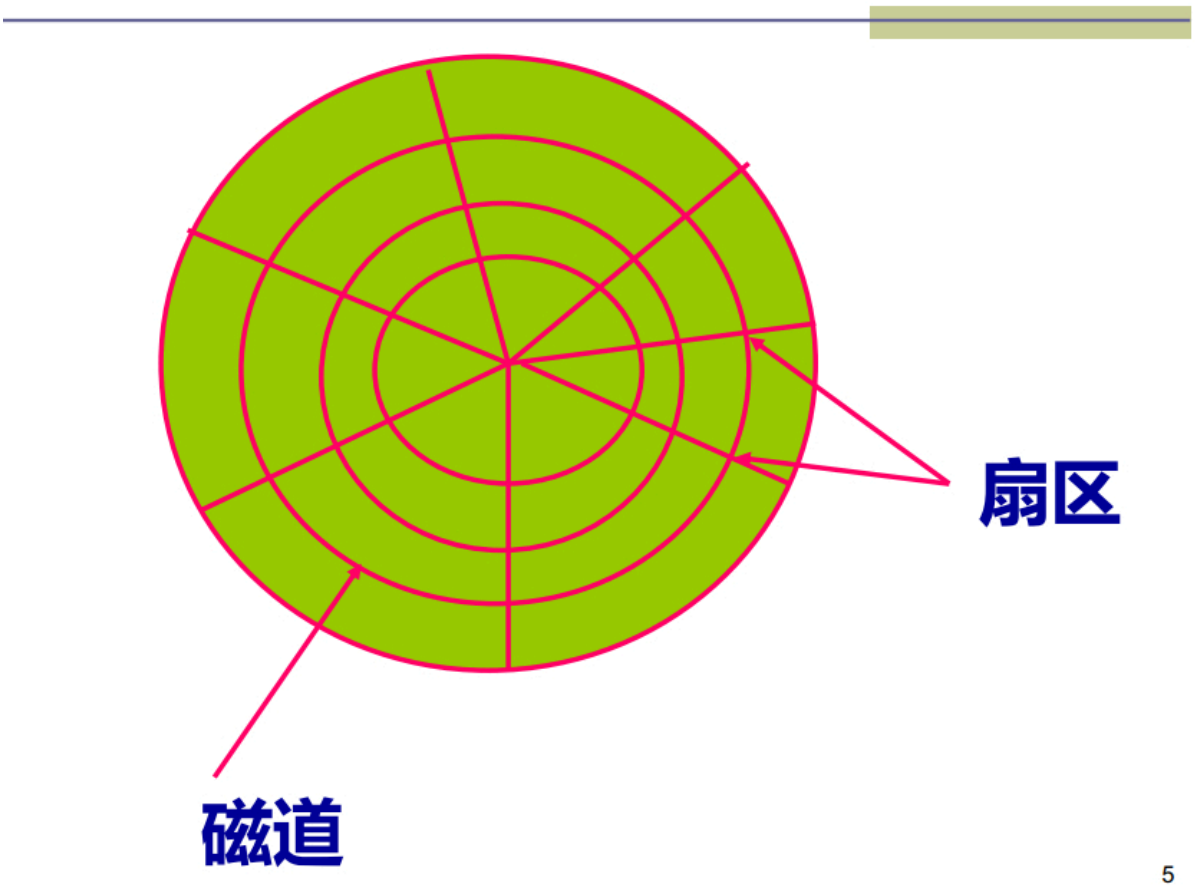
6.磁盘驱动调度

Ⅰ.磁盘分类

- 固定头磁盘
- 移动头磁盘

扇区大小:512字节





Ⅱ.磁盘访问过程

1. 寻道：磁头移动定位到指定磁道

   启动磁头臂 s，跨越一个磁道耗时 m，总共跨越 n 条磁道

$$T_s = m * n + s$$

2. 旋转延迟：等待指定扇区从磁头下旋转经过

   r为磁盘每秒的转数

$$T_\xi = 1/2r$$

3. 数据传输：数据在磁盘与内存之间的实际传输

   r为磁盘每秒的转数，b 为读/写的字节数，每个磁道上字节数为 N

$$T_t = b/rN$$

Ⅲ.磁盘调度算法

- 先来先服务FCFS
  - 简单，公平
  - 效率不高，相邻两次请求可能会造成最内到最外的柱面寻道，使磁头反复移动，增加了服务时间，对机械也不利
- 最短寻道时间优先SSTF——最常用
  - 优先选择距当前磁头最近的访问请求进行服务， 主要考虑**寻道优先**

    - 改善了磁盘平均服务时间，简单有效，性价比好
    - 造成某些访问请求长期等待得不到服务
- 扫描算法
    - 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
    - 类似于电梯
- 单向扫描调度算法CSCAN
    - 总是从0号柱面开始向里扫描，移动臂到达最后一个柱面后，立即带动读写磁头快速返回到0号柱面，返回时不为任何的等待访问者服务，返回后可再次进行扫描
    - **单向扫描**

Ⅳ.提高I/O速度的方法

- 磁盘高速缓存
    - 在内存中开辟一个单独的存储空间作为磁盘高速缓存
    - 把所有未利用的内存空间变为一个缓冲池，供分页系统和磁盘I/O共享
- 优化数据分布
    - 优化物理块的分布
        - 物理块连续分布
        - 增加物理块大小等可以减少磁头的移动
    - 优化索引结点的分布
        - 将索引结点放在中间位置
        - 将磁道分组，每组都有索引结点和文件数据
- 提前读
- 延迟写
- 虚拟盘——RAM盘

7.设备分配

Ⅰ.设备分配方法

- 静态分配
    - **作业级运行**
    - 当一个作业运行之前由系统一次分 配满足需要的全部设备，这些设备一直为该作业占用， 直到作业撤消
    - **不会出现死锁，利用率低**
- 动态分配
    - **进程级运行**
    - 当进程需要使用设备时， 通过系统调用命令向系统提出设备请求，系统按一定的分配策略给进程分配所需设备，一旦使用完毕立即 释放
    - **可能出现死锁**

Ⅱ.设备分配算法

- 先请求先服务
- 优先级高的优先服务

Ⅲ.设备分配技术

1. **设备分类：**
    - 独占设备
    - 共享设备
    - 虚拟设备
2. **分配技术**
    1. 独享分配：进程申请独占设备时，系统把设备分配给这个进程，直到进程释放设备
    2. 共享分配：当一个进程要请求某个设备时，系统按照某种算法立即分配相应的设备给请求者，请求者使用完后立即释放
    3. 虚拟分配：当用户(或进程)申请独占设备时，系统给它分配共享设备的一部分存储空间，当程序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，在适当的时候再将存储空间的信息传输到相应的设备上去处理

8.SPOOLing系统（假脱机）

在联机情况下实现的同时外围操作

Ⅰ.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

Ⅱ.工作原理

- 作业执行前**预先**将程序和数据**输入到输入井**中
- 作业运行后，使用数据时，**从输入井中取出**
- 作业**执行**不必直接启动外设输出数据，只需**将这些数据写入输出井中**
- 作业全部**运行完毕**，再由**外设输出全部数据和信息**

Ⅲ.好处

- 实现了对作业输入、组织调度和输出的统一管理
- 外设在CPU直接控制下，与CPU并行工作

Ⅳ.特点

- 提高了I/O速度
- 将独占设备改造为共享设备
- 实现了虚拟设备功能