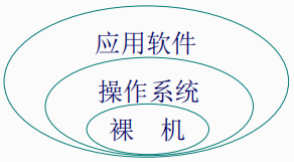
操作系统基础知识

操作系统功能、类型和层次结构

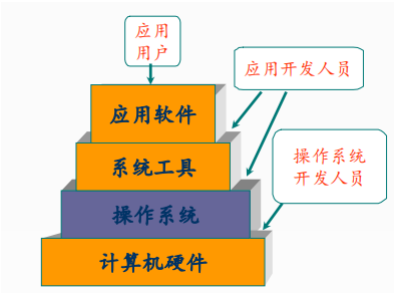
操作系统定义

- 操作系统是直接控制和管理计算机硬件、软件资源，合理地对各类作业进行调度，以方便用户使用的程序集合

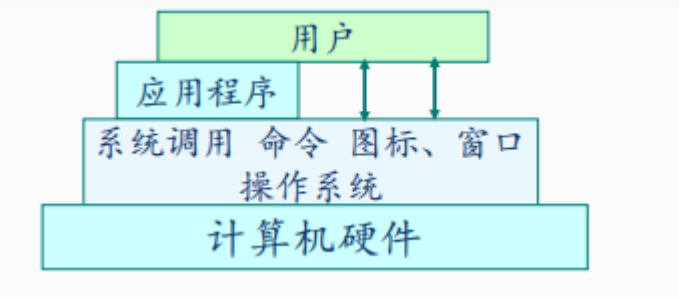


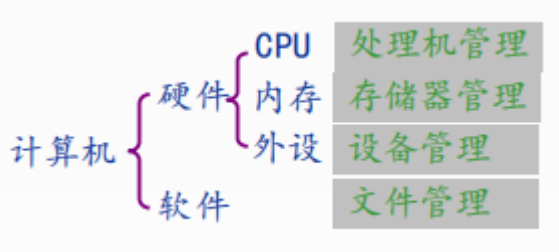
操作系统功能示意图

OS在计算机中的地位



OS的作用

- 作为用户和计算机间的接口
- 作为计算机系统资源的管理者
- 实现了对计算机资源的抽象
- 

- 

操作系统分类

- 批处理操作系统
- 分时操作系统
- 实时操作系统

- 网络操作系统
- 分布式操作系统

操作系统的功能

- 处理机管理功能
- 存储器管理功能
- 设备管理功能
- 文件管理功能
- 用户接口

处理机的管理（进程管理）

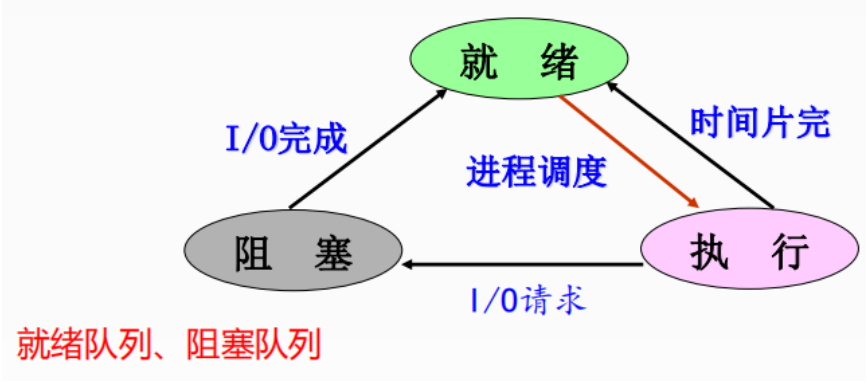
进程的定义

- 进程：程序关于某个数据集合的一次执行过程
- 进程的特征（与程序比较）
  - 结构特征：进程控制块(PCB) + 程序 + 数据 = 进程实体
  - 动态性--最基本特征：
    - 进程：进程实体的一次执行过程，有生命周期
    - 程序：程序是一组有序指令的集合，是静态的概念

进程的三种基本状态

- 就绪状态(Ready)：进程已获得除CPU之外的所有必需的资源，一旦得到CPU控制权，立即可以运行
- 运行状态(Running)：进程已获得运行所必需的资源，它正在处理机上执行
- 阻塞状态(Blocked)：正在执行的进程由于发生某事件而暂时无法执行时，便放弃处理机而处于暂停状态，称该进程处于阻塞状态或等待状态

进程的三种基本状态以及各状态之间的转换：



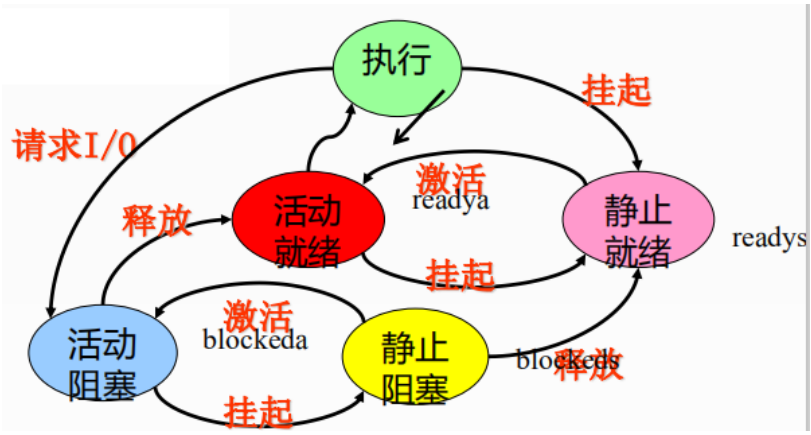
例题：1，0；n-1，0；n，0；

某一时刻单CPU系统中有n个进程，

- 处于运行态的进程最多为（ ），最少为（ ）；
- 处于就绪队列的进程最多为（ ），最少为（ ）；
- 处于阻塞队列的进程最多为（ ），最少为（ ）。

进程的五种状态：

引入挂起状态后，增加了挂起状态(静止状态)到非挂起状态(活动状态)的转换，或者相反



进程互斥与同步

- 进程间两种形式的制约关系
  - 间接相互制约关系 --- 源于资源共享
  - 直接相互制约关系 --- 源于进程合作
- 临界资源（Critical Resource）：把一段时间内只允许一个进程访问的资源称为临界资源或独占资源
- 临界区（Critical Section）：每个进程中访问临界资源的那段代码称为临界区

信号量机制

- 信号量是OS提供的管理公有资源的有效手段
- 信号量是一个整数，**当信号量大于等于零时，代表可供并发进程使用的资源数量，当信号量小于零时，表示处于阻塞态进程的个数**
- Wait 操作（P操作）：
  1. 申请资源，减量操作，S.value=S.value-1
  2. 当S.value<0时，表示资源分配完，进行自我阻塞
- Signal操作（V操作）：
  1. 释放资源，增量操作，S.value=S.value+1
  2. 当S.value<=0，唤醒S.L链表中的等待进程
- 信号量的应用
  - 利用信号量实现进程互斥（模式）为使多个进程互斥的访问某临界资源，须为该资源设置一互斥信号量mutex，并设其初始值为**互斥资源的数量**，然后将各进程访问资源的临界区CS置于wait(mutex)和signal(mutex)之间即可
  - 实现互斥访问的互斥信号量是资源的个数n

    ```
    p1：
       P（n）；
       thing；
       v（n）；
    p2:
       p(n);
       thing;
       v(n);
    ```

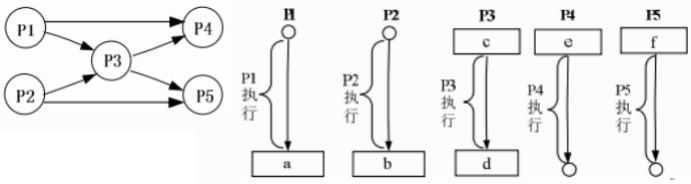
  - 利用信号量实现前驱关系（模式）：设有两个并发执行的进程P1和P2，P1中有语句S1，P2中有语句S2，希望在S1执行后再执行S2；使用进程P1和P2共享一个公共信号量S，并赋予其初值0。**有几个有向边就设置几个初始值为0的信号量**

```
p1: S1;
    v(s);
p2: p(s);
    S2;
```

- 利用记录型信号量实现同步（模式）：生产者消费者模型
  - P1，p2两进程因合作完成一项任务而共用一个变量x
  - 进程p2将处理结果送入x；进程P1将x的结果打印

例题：V(S1)、V(S2)与V(S3)、V(S4)、；P（S1）、P（S2）与V（S3）、V（S4）；P（S2）、P（S5）与P（S4）、P（S6）；

进程P1、P2、P3、P4和P5 的前趋图如下，若用PV操作控制进程P1～P5并发执行的过程，则需要设置 6 个信号量S1、S2、S3、S4、S5和S6，且信号量S1～S6的初值都等于零。下图中 a和 b 处应分别填写（ C ）；c和d处应分别填写（ B ），e和f处应分别填写(C)。



进程调度

- 也称短程调度（Short-Term Scheduling），用来决定就绪队列中的哪个进程应获得处理机，然后再由分派程序把处理机分配给该进程
- 进程调度方式：
  - 非抢占方式（Non-preemptive Mode）：一旦把处理机分配给某进程后，便让该进程一直执行，直至该进程完成或发生某事件而被阻塞时，才把处理机分配给其他进程，决不允许进程抢占已分配出去的处理机
    - 评价：实现简单、系统开销小；适用于大多数的批处理OS，但在要求比较严格的实时系统中，不宜采用这种调度方式
  - 抢占方式（Preemptive Mode）：允许调度程序根据某种原则，去暂停某个正在执行的进程，将处理机重新分配给另一进程
  - 抢占的原则：
    - 时间片原则：各进程按时间片运行，一个时间片用完时，停止该进程执行重新进行调度
    - 短作业（进程）优先原则：短作业（进程）可以抢占长作业（进程）的处理机
    - 优先权原则：优先权高的可以抢占优先权低的进程的处理机

调度算法

- 先来先服务：是一种最简单的调度算法，既可用于作业调度，也可用于进程调度
  - 进程调度采用FCFS算法时，每次调度都从就绪队列中选择一个最先进入该队列的进程，为之分配处理机，使之运行
  - FCFS算法比较有利于长作业（进程），而不利于短作业（进程）
  - 超市买了瓶水，前面有10个人排队，这样就不想排队了
- 短作业（进程）优先调度算法：对长作业不公平的，也没考虑到作业的紧迫性，关键任务有时无法及时执行
  - 短进程优先（SPF）调度算法，是从就绪队列中选出一估计运行时间最短的进程，将处理机分配给它，使它立即执行

    - 短作业优先（SJF）的调度算法：从后备队列中选择一个或若干个估计运行时间最短的作业，将它们调入内存运行
    - SJF调度算法的优缺点:
        - 优点：有效降低作业的平均等待时间，提高系统吞吐量
        - 缺点：对长作业不利；该算法完全未考虑作业的紧迫程度，因而不能保证紧迫性作业（进程）会被及时处理；由于作业（进程）的长短只是根据估计执行时间定的，主观因素较大，不一定能真正做到短作业优先
- 高优先权优先调度算法
    - 优先权类型
        - 静态优先权：在创建进程时确定的，在进程的整个运行期间保持不变。利用某一范围的整数来表示（0～7），又称为优先数。数字越小，优先级越高，数字越大，优先级越小。
        - 动态优先权：在创建进程时所赋予的优先权可以随进程的推进或随其等待时间的增加而改变。
    - 高响应比优先调度算法：数值越大的优先级越高
        - 优先权=(等待时间+要求服务时间)/要求服务时间；由于等待时间不服务时间之和就是系统的响应时间，故上式又表示为：Rp=响应时间/要求服务时间
        - 特点：
            1. 如作业等待时间相同，则要求服务的时间愈短优先权愈高，所以该算法利于短作业
            2. 当要求服务的时间相同，作业优先权的高低决定于其等待时间的长短，所以是先来先服务
            3. 对于长作业，作业的优先级可以随等待时间的增加而提高，当其等待时间足够长也可获得处理机
- 时间片轮转调度算法
    - 每个进程被分配一个时间段，称作它的时间片，即该进程允许运行的时间
    - 如果在时间片结束时进程还没有运行结束，则CPU将被剥夺并分配给另一个进程，该进程到就绪队列尾重新排队
    - 如果进程在时间片内阻塞或结束，则CPU当即进行切换

死锁（只有临界资源才会发生死锁）

- 是指多个进程在运行过程中因争夺资源而造成的一种僵局，当进程处于这种状态时，若无外力作用，它们都将无法再向前推进
- 产生死锁的原因
    - 竞争资源：当系统中供多个进程共享的资源如打印机、公用队列等，其数目不足以满足诸进程的需要时，会引起诸进程对资源的竞争而产生死锁
        - 可剥夺性资源：资源分配给进程后可以被高优先级的进程剥夺。如CPU、主存
        - 不可剥夺性资源：分配给进程后只能在进程用完后才释放的资源。如磁带机、打印机等
    - 进程间推进顺序非法：进程在运行过程中，请求和释放资源的顺序不当，也同样会导致产生死锁
- 产生死锁的必要条件
    - 互斥条件
        - 进程访问的是临界资源，即在一段时间内某资源只由一个进程占用。如果此时还有其他进程请求该资源，则请求者只能等待，直至占有该资源的进程用完释放
    - 请求和保持条件
        - 一进程在请求新的资源的同时，保持对已分配资源的占有
    - 不剥夺条件

        - 指进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放
    - 环路等待条件
        - 指在发生死锁时，必然存在一个进程--资源的环形链。 即进程集合{P0,P1,P2,...,Pn}中的P0正在等待一个P1占用的资源；P1正在等待一个P2占用的资源，……，Pn正在等待一个已被P0占用的资源
- 处理死锁的基本方法
    - 预防死锁
        - 是一种较简单和直观的事先预防方法。该方法是通过设置某些限制条件，去破坏产生死锁的四个必要条件的一个或几个，来预防发生死锁
            - 摒弃“请求和保持”条件

              如果进程有一个资源不能申请到，那么就需要将已经占有的资源也释放掉
            - 摒弃“不剥夺”条件

              可以一个一个的申请，当有一个申请不到的时候，那么就将自己的资源释放
            - 摒弃“环路等待”条件

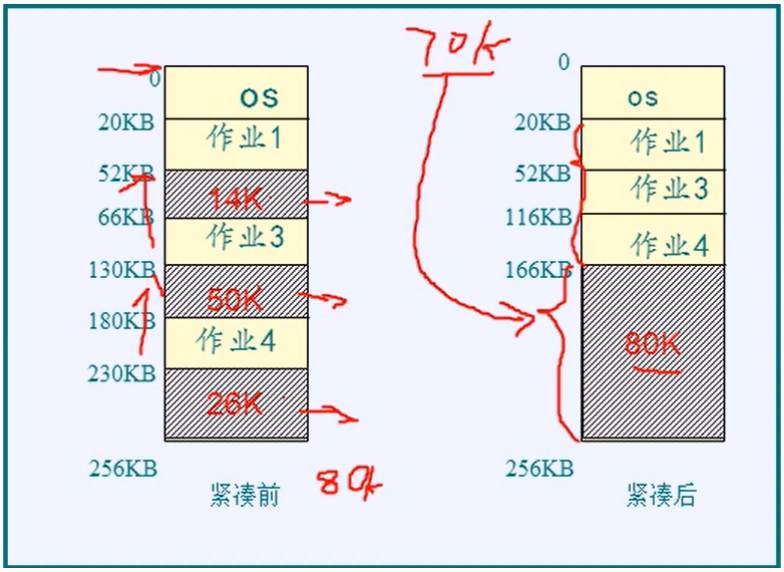
              将资源编号，只能申请资源号大于现在占有的资源的资源号
    - 避免死锁
        - 该方法同样是属于事先预防的策略，这种方法不是预先加上各种限制条件以预防产生死锁的可能性，而是用某种方法去防止系统进入不安全状态，使死锁不致于最终发生
            - 银行家算法避免死锁
    - 检测死锁
        - 这种方法并不须事先采取任何限制性措施，也不必检查系统是否已经进入不安全区
        - 此方法允许系统在运行过程中发生死锁，但可通过系统所设置的检测机构，及时的检测出死锁的发生，并精确的确定不死锁有关的迪程和资源；然后采取适当的措施，从系统中将已发生的死锁清除掉
    - 解除死锁
        - 是与死锁检测相配套的一种措施。当检测到系统中已发生死锁时，须将进程从死锁状态中解脱出来
        - 常用的实施方法是撤销或挂起一些进程，以便回收一些资源，再将这些资源分配给已处于阻塞状态的进程，使之转为就绪状态，以继续运行
        - 死锁的检测不解除措施，有可能使系统获得较好的资源利用率和吞吐量，但在实现上难度也最大

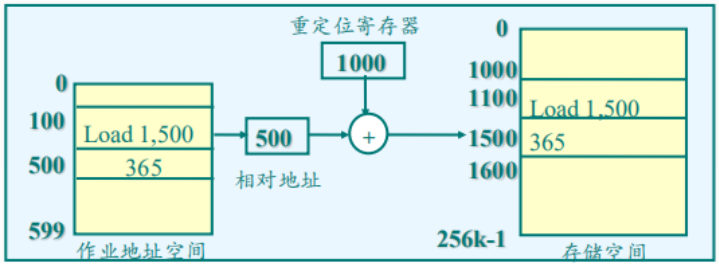
存储管理

- 存储管理主要是指对内存的管理，负责内存分配和回收，内存的保护和扩充。
- 存储管理的目的是尽量提高内存的使用效率

内存的分配方式

- 连续的分配方式：指为一个用户程序分配一个连续的内存空间
    - 类型
        - 单一连续分配：DOS操作系统使用，某一时刻只有一个用户使用的操作系统中使用这种方式
        - 固定分区分配
        - 动态分区分配：为把一个新作业装入内存，需按照一定的分配算法，从空闲分区表或空闲分区链中选出一分区分配给该作业

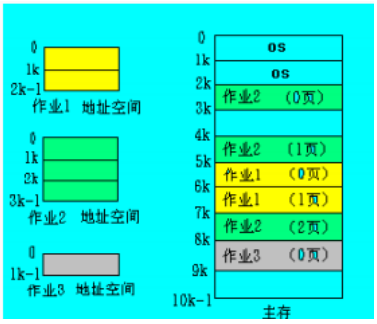
        - 常用的分配算法
            - 首次适应算法
            - 循环首次适应算法
            - 最佳适应算法：外碎片最严重（外碎片：想用用不了；内碎片：分出去不用）需要从大到小排序块大小，效率最高
            - 最坏适应算法：碎片往往最少
    - 可重定位分区分配：如果在系统中只有若干个小分区，即使它们的容量总和大于要装入的程序，但由于这些分区不相邻也无法将程序装入内存
    - 

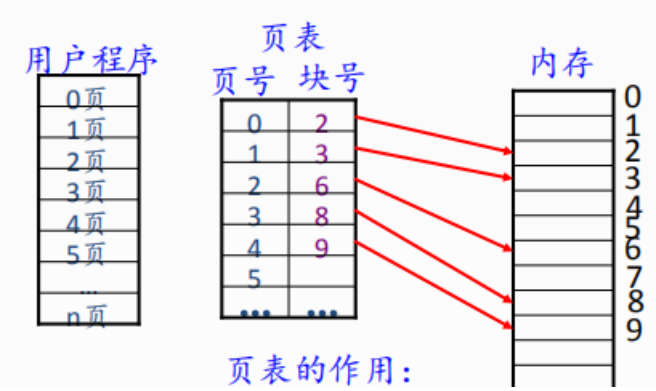
    - 解决方法：将内存中的所有作业进行移动，使它们全部邻接，这样把原来分散的小分区拼接成大分区，这种方法称为“拼接”或“紧凑”
- 动态重定位的实现：
    - 地址转换需要重定位寄存器的支持
    - 在动态运行时装入的方式，将相对地址转换为物理地址的工作在程序指令真正要执行时才进行
    - 程序执行时访问的内存地址是相对地址与重定位寄存器中的地址相加而成
    - 

- 对换与覆盖技术
    - 覆盖技术：一个作业的若干程序段或数据段的某些部分共享内存空间
    - 对换技术：内存中的数据对换到外存（Windows中的虚拟内存）
        - 把内存中暂时不能运行的进程或者暂时不用的程序和数据，调到外存上，以便腾出足够的内存空间，再把已具备运行条件的进程和进程所需要的程序和数据调入内存
        - 对换的分类
            - 整体对换(或进程对换)：以整个进程为单位
            - 页面对换或分段对换：以页或段为单位
                - 连续分配方式会形成“碎片”，虽然可以通过“紧凑”（外碎片）解决，但开销大。如果允许将一个进程直接分散地装入许多不相邻的分区中，则无需“紧凑”，由此产生离散分配方式
- 离散的分配方式：
    - 分页存储管理方式：离散分配的基本单位是页

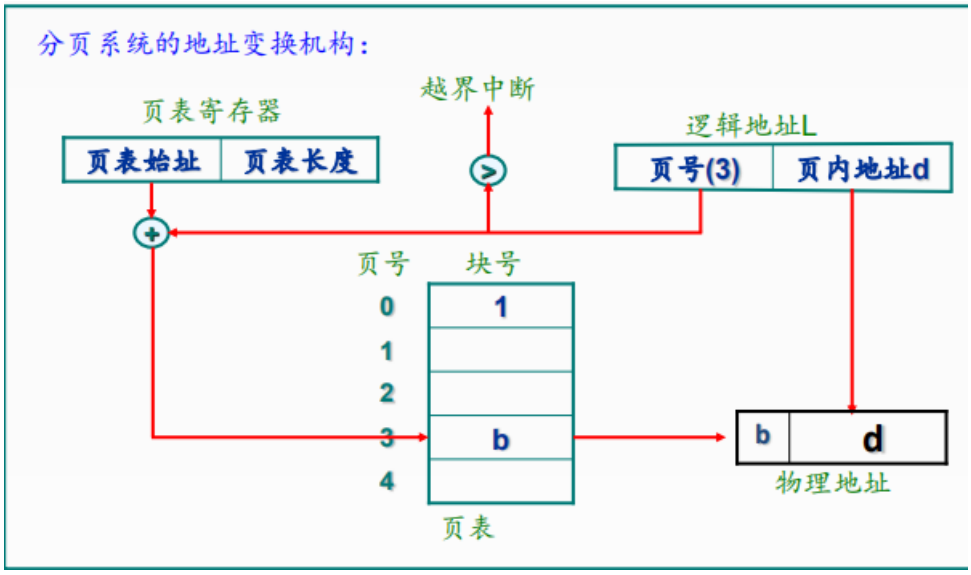
- 分段存储管理方式：离散分配的基本单位是段

基本分页存储管理方式

- 页面与页表
  - 页面
    - 分页式存储管理的原理
      1. 将一个进程的逻辑地址空间分成若干个大小相等的片称为页面或页，并为各页加以编号，从0开始。同时把内存空间分成与页面相同大小的若干个存储块，称为块或页框
      2. 在为进程分配内存时，以块为单位将进程的若干个页分别装入到多个可以不相邻的物理块中
      3. 进程的最后一页经常装不满一块而形成“页内碎片”
  - 

  - 地址变换
    - 若给定一个逻辑地址空间中的地址为A，页面大小为L，则：页号P = INT[A/L]、页内地址d = [A] MOD L
- 基本分页式存储管理的实现
  - 

  - 进程的每一页离散地存储在内存的任一存储块中，为方便查找，系统为每一进程建立一张页面映像表，简称页表
    - 页表实现了从页号到物理块号的地址映射
    - 由于页内地址与物理地址是一 一对应的，因此，地址变换机构的任务是借助于页表，将逻辑地址中的页号转换为内存中的物理块号
  - 由于页表是存放在内存中的，CPU在每存取一个数据时，需要两次访问内存
    - 存储器利用率提高，处理器处理速度降低
      - 第一次：访问页表，找到指定页的物理块号，将块号不页内偏移量拼接形成物理地址
    - 第二次：从第一次所得地址中获得所需数据，或向此地址中写入数据
  -

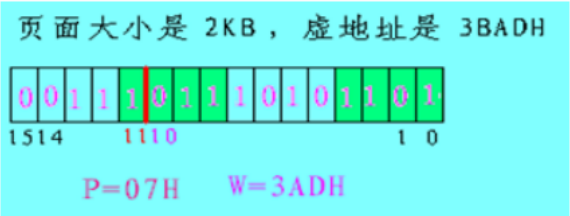


- 具有快表的地址变换机构
  - 解决方法：在地址变换机构中，增设一个具有并行查寻能力的特殊高速缓冲寄存器，称为“联想存储器”或“快表”

- 地址变换机构
  - 快表

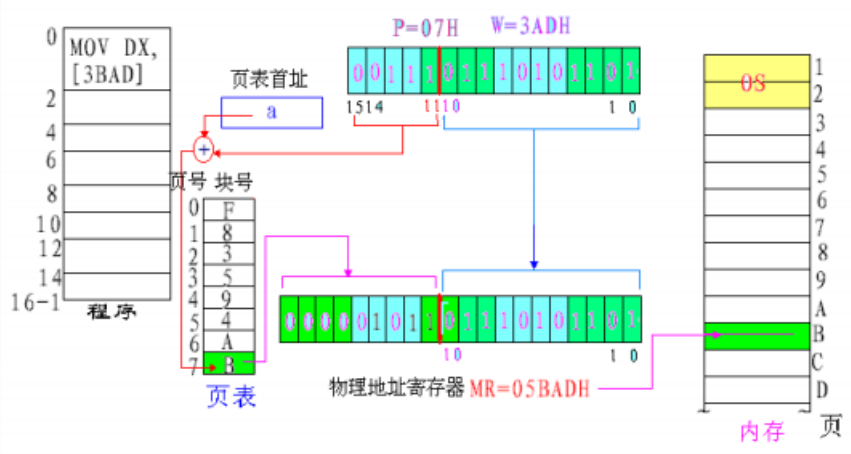
例题1：

例1.设页面大小为2KB，将逻辑地址3BADH划分为页号和页内偏移量两部分。用16进制表示。



例题2：

例2. 设页面大小为2KB，作业的页表如下图。求逻辑地址3BADH的物理地址。用16进制表示。



例题3：

例3：有一系统采用页式存储管理，有一作业大小是8KB，页大小为2KB，依次装入内存的第7、9、A、5块，试将虚地址0AFEH转换成内存地址。

虚地址 0AFEH

0000 1010 1111 1110

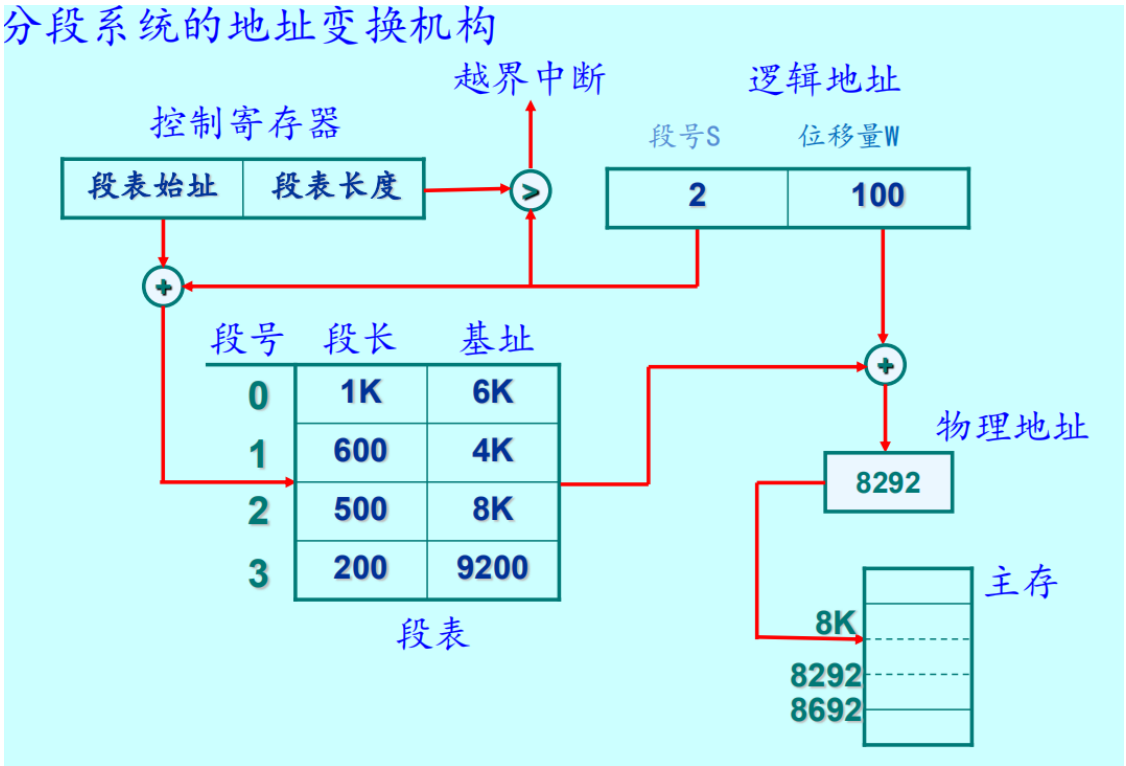
P=1 d=010 1111 1110

MR=0100 1010 1111 1110

=4AFEH

页号	块号
0	7
1	9
2	A
3	5

基本分段式存储管理的实现

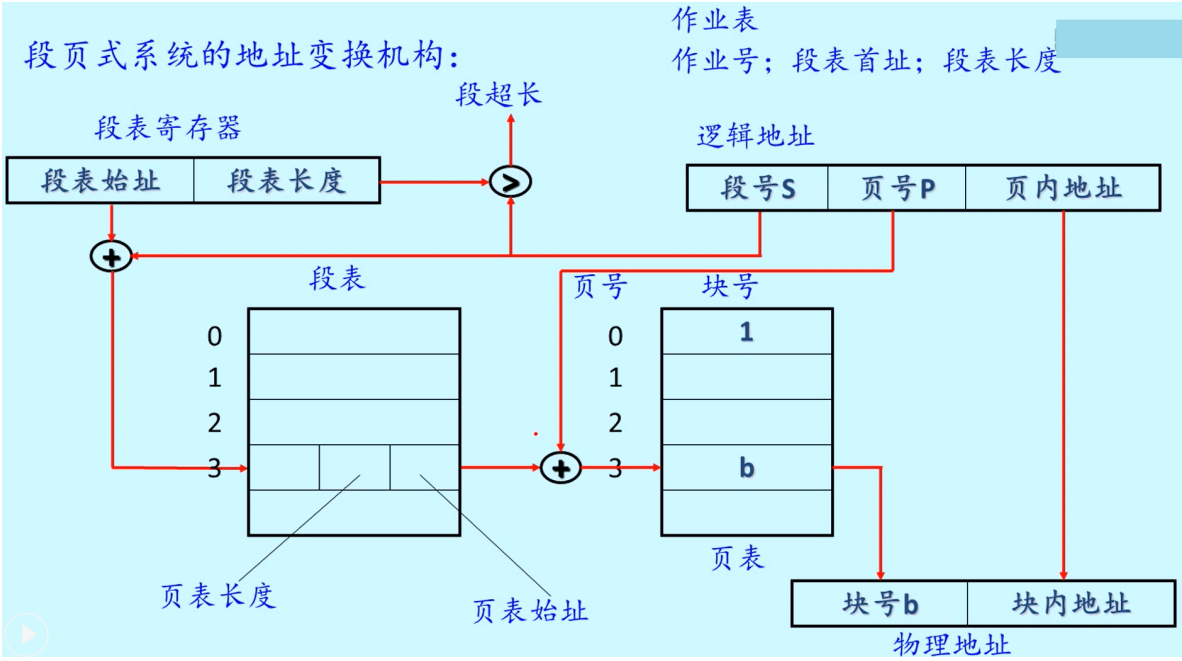
- 段表
  - 为使程序正常运行，须在系统中为**每个进程**建立一张段映射表，简称“段表”。每个段在表中占有一个表项
  - 段表结构
    - 段号；段在内存中的起始地址(基址)；段长
  - 段表用于实现从逻辑段到物理内存区的映射
  - 段表可以存放在寄存器中，但更多的是存放在内存中
- 地址变换机构
  - 在系统中设置段表寄存器，用于存放段表始址和段表长度，以实现从进程的逻辑地址到物理地址的变换
  - 当段表存放在内存中时，每访问一个数据，都需访问两次内存，降低了计算机的速率
  - 解决方法：设置联想寄存器，用于保存最近常用的段表项
- 

分页和分段的主要区别

- 相似点
  - 采用离散分配方式，通过地址映射机构实现地址变换
- 不同点
  - 页是信息的物理单位，分页是为了满足系统的需要
  - 段是信息的逻辑单位，含有意义相对完整的信息，是为了满足用户的需要
- 页的大小固定且由系统确定，由系统把逻辑地址分为页号和页内地址，由机器硬件实现
- 段的长度不固定，取决于用户程序，编译程序对源程序编译时根据信息的性质划分
  - 分页的作业地址空间是一维的
  - 分段的作业地址空间是二维的

段页式存储管理

- 获得一条指令或数据，需访问三次内存
  - 第一次：访问内存中的段表，取得页表始址
  - 第二次：访问内存中的页表，取得该页所在的物理块号，将块号与页内地址形成物理地址
  - 第三次：根据第二次所得的地址，取出指令或数据
- 缺点：访存次数增加两倍
- 解决方法：增设高速缓冲寄存器 快表



页面置换算法

- 最佳置换算法：
  - 最佳置换算法是一种理想化的算法，具有最好的性能，但难于实现；算法无法实现，但可评价其他算法
  - 其所选择的被淘汰页面，将是以后永不再用的，或许是在最长(未来)时间内不再被访问的页面
  - 优点：保证获得最低的缺页率
  - 缺点：无法预知一个进程在内存的若干个页面，哪个在未来最长时间内不再被访问
- 先进先出置换算法：
  - 先进先出置换算法最直观，但可能性能最差，故应用极少
  - 算法总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面予以淘汰
  - 实现：算法实现简单，只需把一个进程已调入内存的页面，按先后次序链接成一个队列，并设置一个指针(替换指针)，使它总是指向最老的页面

    - 缺点：算法与进程的实际运行规律不相适应，因为进程中的某些页面经常被访问，但先进先出置换算法不能保证这些页面不被淘汰
    - Belady现象：如果对一个进程未分配它所要求的全部页面，有时就会出现分配的页面数增多但缺页率反而提高的异常现象。发生在FIFO（先进先出）置换算法
- 最近最久未使用（LRU）算法：
    - 算法根据页面调入内存后的使用情况进行决策
    - 由于无法预测各页面将来的使用情况，只能利用“最近的过去”作为“最近的将来”的近似，因此，LRU置换算法是选择最近最久未使用的页面予以淘汰
    - 实现：该算法赋予每个页面一个访问字段，用来记录一个页面自上次被访问以来所经历的时间t，当需淘汰一个页面时，选择现有页面中其t值最大的，即最近最久未使用的页面予以淘汰

设备管理

I/O系统包括

- 输入、输出设备
- 存储功能的设备
- 设备控制器

设备管理的概念

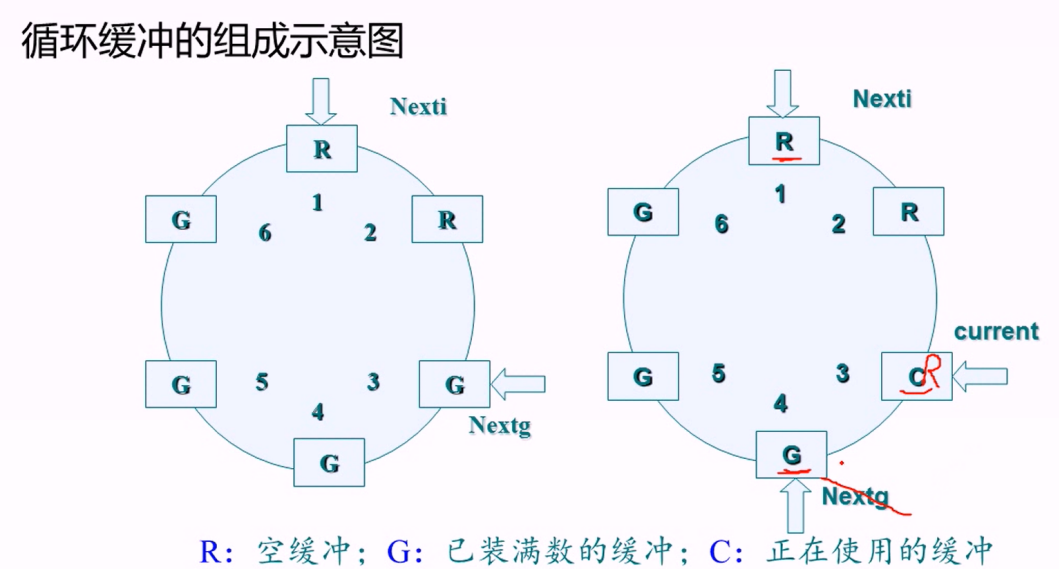
- 提供的功能
    - 提供和进程管理系统的接口
    - 进行设备分配
    - 实现设备和设备之间、设备和CPU之间的并行操作
    - 进行缓冲区管理

I/O控制方式

- 程序I/O方式
- 中断控制I/O方式
- 直接存储器访问(DMA) 方式
- I/O通道控制方式
    - 字节多路通道
    - 选择通道
    - 成组多路通道

缓冲管理

- 引入缓冲区的原因
    - 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
- 单缓冲：单次使用只能读或者写
    - 每当用户进程发出一I/O请求时，OS便在主存中为之分配一缓冲区
- 在字符设备输入时，缓冲区用于暂存用户输入的一行数据，在输入期间，用户进程被挂起以等待数据输入完毕
    - 在输出时，用户进程将一行数据输入到缓冲区后，继续执行处理

    - 当用户进程已有第二行数据输出时，如果第一行数据尚未被提取完毕，则此时用户进程应阻塞
- 多缓冲：在设备输入时，先将数据送入第一缓冲区，装满后便转向第二缓冲区；此时输出设备从第一个缓冲区进行输出
    - 为了加快输入和输出速度，提高设备利用率
    - 当在其他缓冲区输入时，OS可以从第一缓冲区中移出数据，并送入用户进程
- 循环缓冲：类似生产者消费者模型
    - 循环缓冲有多个大小相同的缓冲区
- 类型
    - 用于装输入数据的空缓冲区R
    - 已装满数据的缓冲区G
        - 计算进程正在使用的现行工作缓冲区C
    - 

- 缓冲池（Buffer Pool）：既可输入又可输出的公用缓冲池
    - 类型
        - 空缓冲区
        - 装满输入数据的缓冲区
        - 装满输出数据的缓冲区

设备的分配

- 分配原则：
    - 静态分配：不会发生死锁
    - 动态分配（按需分配）：使用时，请求申请，会发生死锁
- 分配策略：
    - 先请求先分配
    - 优先级高者先分配

磁盘管理

- 磁盘的访问时间
    - 寻道时间Ts：把磁臂从当前位置移到指定磁道上所经历的时间
    - 旋转延迟时间Tr：指定扇区移动到磁头下面所经历的时间
    - 传输时间Tt：数据从磁盘读出或向磁盘写入数据所经历的时间
- 磁盘调度算法

    - 先来先服务：根据进程请求访问磁盘的先后次序进行调度
        - 优点：公平、简单，且每个进程的请求都能依次得到处理，不会出现某一进程的请求长期得不到满足的情况
        - 缺点：未对寻道进行优化，致使平均寻道时间可能较长。仅适用于请求磁盘I/O的进程数目较少的场合
    - 最短寻道时间优先：优先满足访问磁道与当前磁头所在磁道距离最近的进程，以使每次的寻道时间最短
        - 缺点：这种调度算法不能保证平均寻道时间最短
    - 扫描（SCAN）算法（电梯调度算法）
        - 算法既能获得较好的寻道性能，又能防止进程饥饿，被广泛用于大、中、小型机和网络中的磁盘调度
        - 当磁头刚从里向外移动过某一磁道时，恰有一进程请求访问此磁道，这时该进程必须等待，待磁头从里向外，然后再从外向里扫描完所有要访问的磁道后，才处理该进程的请求，致使该进程的请求被严重地推迟
    - 循环扫描CSCAN：为了减少请求进程的延迟，CSCAN算法规定磁头单向移动。若规定只自里向外移动，当磁头移到最外的被访问磁道时，磁头立即返回到最里的欲访磁道，即将最小磁道号紧接着最大磁道号构成循环，进行扫描
- 虚设备与SPOOLing技术：假脱机技术
    - 为缓和CPU的高速性与I/O设备低速性间的矛盾而引入了脱机输入、脱机输出技术
- 该技术是利用专门的外围控制机，将低速设备上的数据传送到高速磁盘上；或者相反
    - 这样就可以在主机的直接控制下实现脱机输入输出。此时外围操作与CPU对数据的处理同时进行，我们把这种在联机情况下实现的同时外围操作称为SPOOLing（Simultaneaus Periphernal Operating On—Line），或称为假脱机操作。
    - 三大部分
        - 输入井和输出井。是磁盘上开辟的两个大存储空间
    - 输入缓冲区和输出缓冲区。在内存中开辟两个缓冲区，输入缓冲区暂存由输入设备送来的数据，后送输入井；输出缓冲区暂存从输出井送来的数据，后送输出设备
        - 输入进程和输出进程。利用两个进程模拟脱机I/O时的外围处理机
    - 系统特点
        - 提高了I/O的速度。利用输入输出井模拟成脱机输入输出，缓和了CPU和I/O设备速度不匹配的矛盾
    - 将独占设备改造为共享设备
        - 实现了虚拟设备功能。多个进程同时使用一台独占设备，虚拟成了多台设备

文件管理

文件和文件系统

- 文件是指具有文件名的若干相关元素的集合
    - 负责管理文件的系统软件
    - 被管理的对象--文件
- 现代OS中通过文件系统来组织和管理计算机中存储的数据
- 文件的逻辑结构
    - 从用户观点出发所观察到的文件组织形式，是用户可以直接处理的数据及其结构，它独立于文件的物理特性，又称为文件组织
    - 分类

        - 有结构文件，是指由一个以上的记录构成的文件，又把它称为记录式文件；根据记录的长度可分为定长记录文件；不定长记录文件
            - 组织方式
                - 顺序文件。由一系列记录按某种顺序排列所形成的文件。通常是定长记录
                - 索引文件。当记录可变长时，通常为之建立一张索引表，并为每个记录设置一个表项以加快对记录检索的速度
                - 索引顺序文件。上述两种方式的结合。为文件建立一张索引表，为每一组记录中的第一个记录设置一个表项
                - 直接文件
        - 无结构文件，这是指由字符流构成的文件，故又称为是流式文件
            - 大量的源程序、可执行文件、库函数等，所采用的就是无结构的文件形式，即流式文件。其长度以字节为单位。对流式文件的访问，则是采用读写指针来指出下一个要访问的字符
            - UNIX 系统中，所有的文件都被看做是流式文件
- 文件的物理结构
    - 文件在外存上的存储组织形式。与存储介质的存储性能和采用的外存分配方式有关；由于磁盘具有可直接访问的特性，故当利用磁盘来存放文件时，具有很大的灵活性
    - 分配方法：在一个系统通常只采用一种方法
        - 连续分配：连续分配要求为每一个文件分配一组相邻的盘块。在采用该方式时，可把逻辑文件中的记录顺序的存储到邻接的各物理块中，这样所形成的文件结构成为顺序文件结构，此时的物理文件称为顺序文件。这种分配方式保证了逻辑文件中的记录顺序与存储器中文件占用盘块的顺序的一致性
            - 随着文件的建立与删除不断进行，将产生很多外存的碎片，利用紧凑方法也可消除碎片
        - 链接分配：采用链接分配方式时，可通过在每个盘块上的链接指针，将同属于一个文件的多个离散的盘块链接成一个链表，把这样形成的文件称为链接文件
        - 索引分配
            - 缺点
                - 不能支持高效的直接存取。要对一个文件进行直接存取，需首先在FAT中顺序的查找许多盘块号
                - FAT需占用较大的内存空间。当磁盘容量较大时，FAT可能要占用数MB以上的内存空间。这是令人难以忍受的
                - 可能要花费较多的外存空间。每当建立一个文件时，便须为之分配一个索引块，将分配给该文件的所有盘块号记录于其中
            - 分配方式
                - 单级索引方式
                - 多级索引方式
                - 混合索引方式

存储空间的管理

- 空闲表法 和 空闲链表法
- 位示图法
- 成组链接法

作业管理

作业状态

- 提交
- 后备
- 执行
- 完成

处理机调度

- 高级调度（High Scheduling）
    - 也称为作业调度，是指在后备队列中选择一个或一给作业，为它们建立进程，分配必要的资源，使它们能够运行
    - 在批处理系统中，因作业进入系统后先驻留在外存，故需要有作业调度
    - 在分时系统中为做到及时响应，命令或数据被直接送入内存，故不需作业调度
    - 在实时系统中，不需作业调度
- 中级调度（Intermediate-Level Scheduling）
    - 是为了提高内存利用率和系统吞吐量
    - 应使那些暂时不能运行的进程不再占用宝贵的内存资源，而将它们调到外存去等待，把此时的进程状态称为就绪驻外存状态或挂起状态
- 低级调度（Low Level Scheduling）
    - 也称为进程调度或短程调度，用来决定就绪队列中的哪个进程应获得处理机
    - 就 绪
    - 阻 塞
    - 执 行

调度算法

- 先来先服务
- 短作业（进程）优先调度算法
- 高优先权优先调度算法
- 高响应比优先调度算法

用户接口

- 操作系统接口
    - 命令接口
    - 程序接口

就绪

执行

阻塞