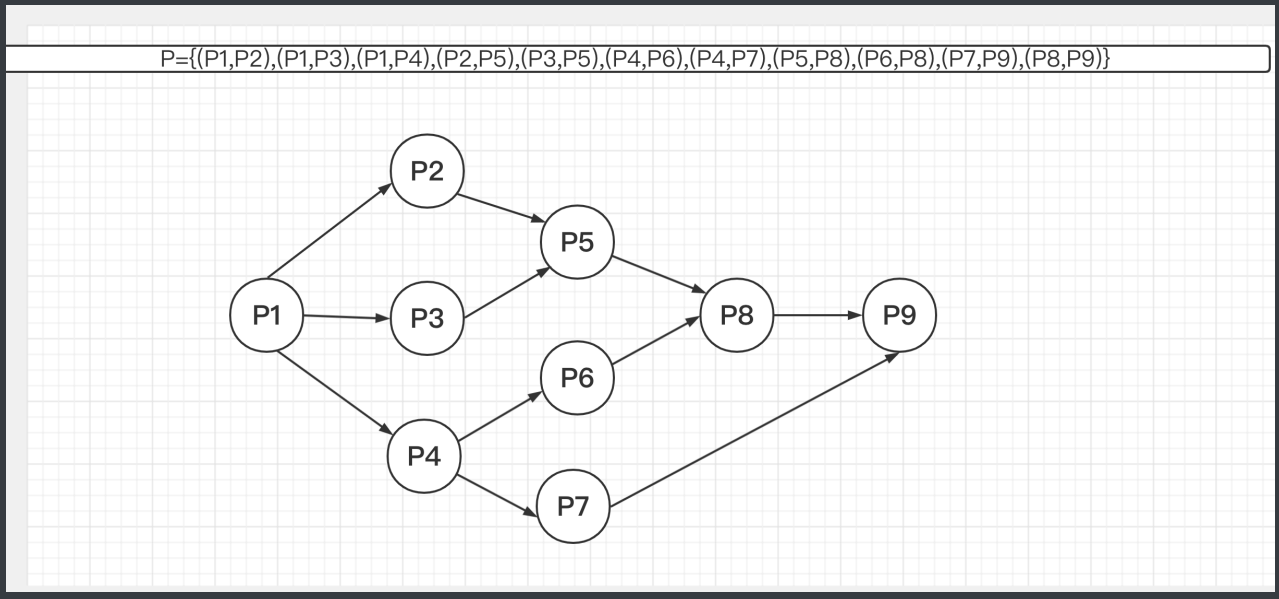
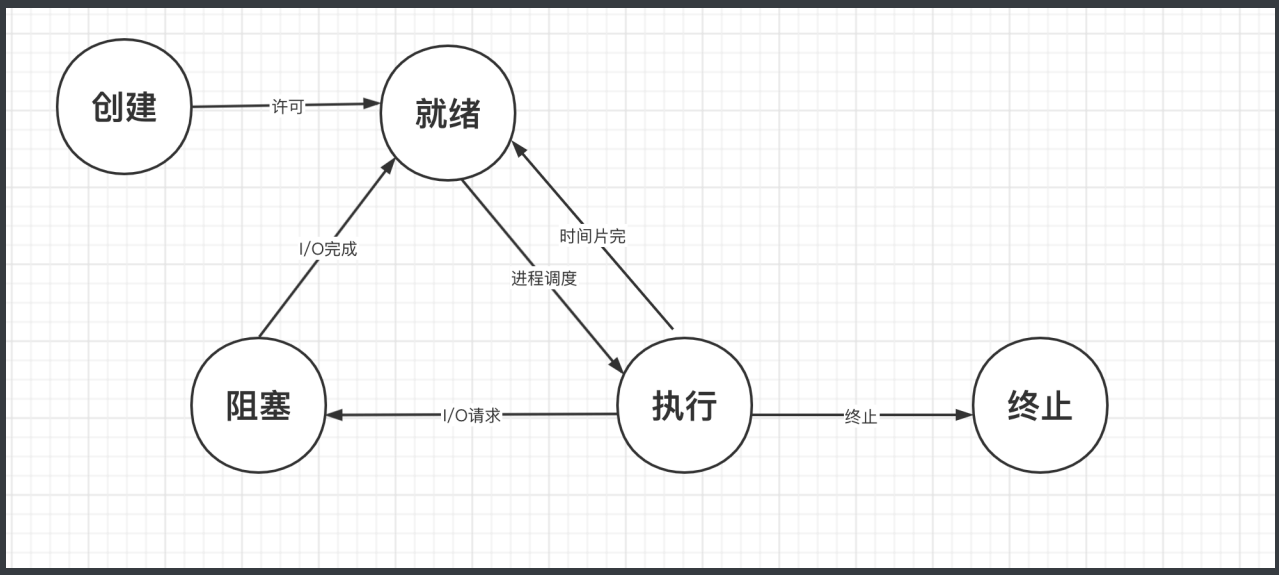
期末考试范围

- 第一章（选择填空）
  - 操作系统的接口（图）
    1. OS作为用户与计算机硬件系统之间的接口
    2. 用户可以通过命令方式、系统调用方式和图标-窗口方式来实现与操作系统的通信
  - 1.1.3推动操作系统发展的动力
    1. 不断提高计算机资源利用率
    2. 方便用户
    3. 器件的不断更新换代
    4. 计算机体系结构的不断发展
    5. 不断提出新的应用需求
  - 现代操作系统有哪几种，历程包含
    1. 未配置操作系统的计算机系统：人工操作方式、脱机操作方式
    2. 单道批处理系统
    3. 多道批处理系统
    4. 分时系统
    5. 实时系统
    6. 微机操作系统
  - 1.2.4分时系统工作原理
    1. 主要动力是：提高资源利用率和系统吞吐量
    2. 定义：在一台主机上连接了多个配有显示器和键盘的终端并由此所组成的系统
    3. 特征：多路性、独立性、及时性、交互性
    4. 原理：分时操作系统的核心原理在于将作业直接放入内存，并引入了时间片的概念，采用轮转运行的方式，规定每个作业每次只能运行一个时间片，然后就暂停该作业并立即调度下一个作业运行。在不长的时间内使所有的作业都执行一个时间片的时间，便可以使每个用户都能及时地与自己的作业进行交互，从而使用户的请求得到及时响应。这样就解决了在分时系统中最重要的及时接收、及时处理问题。
- 第二章（选择填空大题）
  - 大题（会画前驱图）



- 状态转换图（大题）

  进程的三种状态：就绪态、执行态、阻塞态

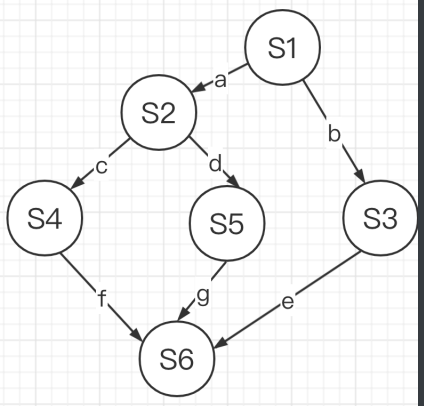


- 利用信号实现前驱关系（重点）

  1. 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。我们希望在S1执行后在执行S2。
  2. 只需使进程P1和P2共享一个公用信号S，并赋予其初值为0，将signal(S)操作放在语句S1后面，而在S2语句前面插入wait(S)操作
  3. 在进程P1中，用S1；signal(S)；
  4. 在进程P2中，用wait(S)；S(2)
  5. 由于S被初始化为0，这样若P2先执行必定阻塞，只有在进程P1执行完S1；signal(S)；操作后使S增为1时，P2进程方能成功执行语句S2
  6. 同样，可以利用信号量按照语句间的前驱关系（如下图所示）实现一个各更为复杂的可并发执行的程序

为了使右图程序正确执行，保证前驱关系应分别设置信号量a,b,c,d,e,f,g，代码如下

```
p1() {S1; signal(a); signal(b); }
p2() {wait(a); S2; signal(c); signal(d); }
p3() {wait(b); S3; signal(e); }
p4() {wait(c); S4; signal(f); }
p5() {wait(d); S5; signal(g); }
p6() {wait(e); wait(f); wait(g); S6; }
main() {
    semaphore a, b, c, d, e, f, g ;
    a.value = b.value = c.value = d.value = e.value = f.value = g.value = 0;
    cobegin
        p1(); p2(); p3(); p4(); p5(); p6();
    coend
}
```



- 进程的定义和特点（了解）选择填空
  1. 定义：
     1. 进程是程序的一次执行
     2. 进程是一个程序及其数据在处理机上顺序执行时所发生的活动
     3. 进程是具有独立功能的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
  2. 特征：
     1. 动态性
     2. 并发性
     3. 独立性
     4. 异步性
  3. PCB中的信息
     1. 进程标识符
     2. 处理机状态
     3. 进程调度信息
     4. 进程控制信息
  4. 进程的创建
     1. 申请空白PCB
     2. 为新进程分配其运行所需要的资源
     3. 初始化PCB
     4. 将新进程插入就绪队列
- 第三章
  - 作业调度算法（重点）
    1. 先来先服务（FCFS）

1. 既可以用于作业调度，也可以用于进程调度
2. 按照作业到达的先后次序

作业	提交时间	运行时间
1	8.0	1.0
2	8.5	0.5
3	9.0	0.2
4	9.1	0.1

1. 先来先服务（FCFS）

就是提交时间，也有的题里叫做到达时间

完成时间 – 提交时间（到达时间）

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	8.0	1.0	8.0	9.0	1.0	1
2	8.5	0.5	9.0	9.5	1.0	2
3	9.0	0.2	9.5	9.7	0.7	3.5
4	9.1	0.1	9.7	9.8	0.7	7

平均周转时间：（1.0 + 1.0 + 0.7 + 0.7 ）/ 4
平均带权周转时间：（1 + 2 + 3.5 + 7 ）/ 4

开始时间 + 运行时间

周转时间 / 运行时间

2. 短作业优先（SJF）

1. 按照作业的长短，作业越短，优先级越高

作业	提交时间	运行时间
1	8.0	1.0
2	8.5	0.5
3	9.0	0.2
4	9.1	0.1

2. 短作业优先调度算法（SJF）

就是提交时间，也有的题里叫做到达时间

完成时间 – 提交时间（到达时间）

作业	提交时间	运行时间	开始时间	完成时间	周转时间	带权周转时间
1	8.0	1.0	8.0	9.0	1.0	1
2	8.5	0.5	9.3	9.8	1.3	2.6
3	9.0	0.2	9.0	9.2	0.2	1
4	9.1	0.1	9.2	9.3	0.2	2

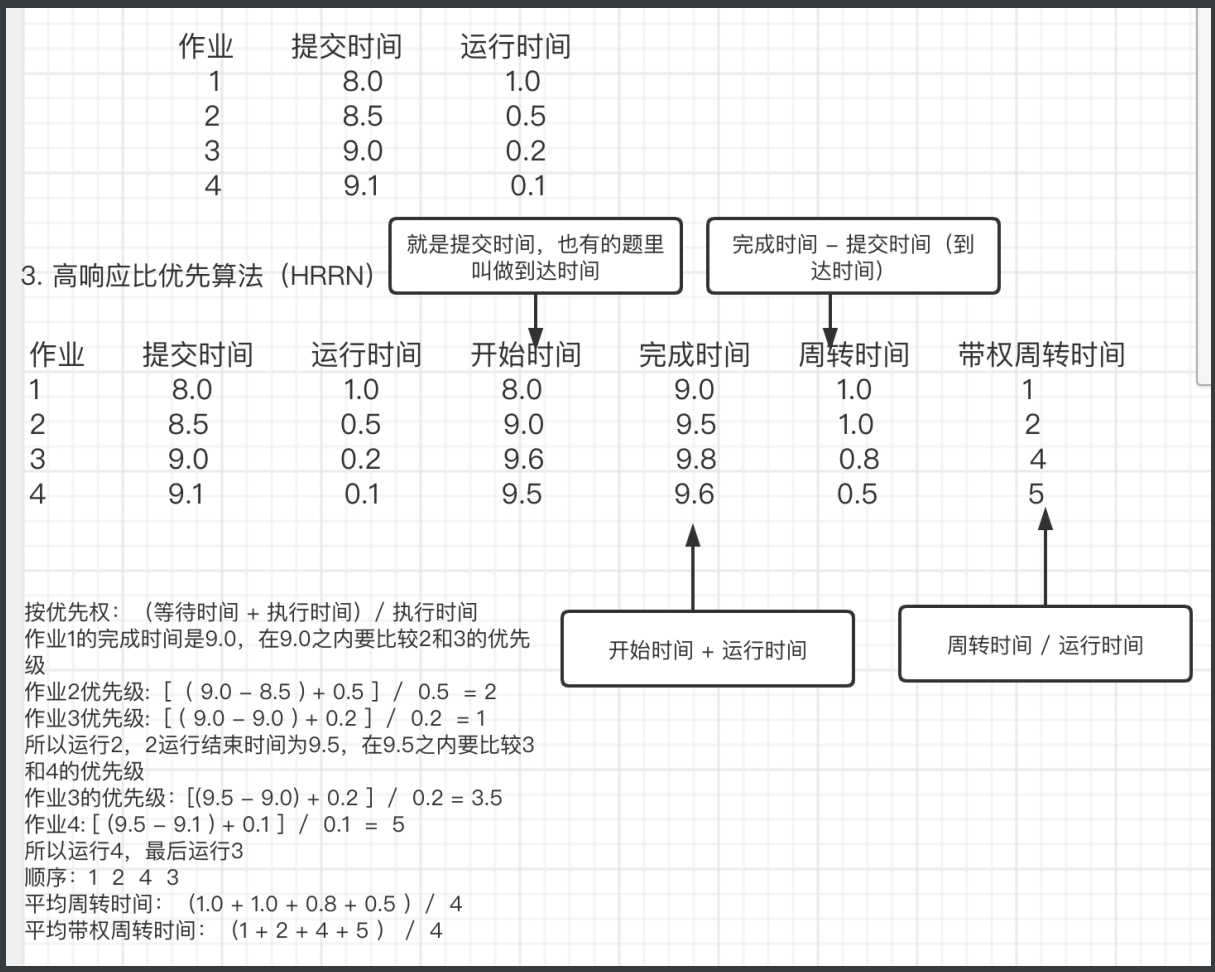
因为作业越短优先级越高，所以当作业1运行完成后是9.0，在9.0时间之内，只有2和3的提交时间是小于9.0的，所以比较2和3的运行时间，3的运行时间短，所以作业3运行，完成后时间为9.2，在9.2时间之内，还有2和4的提交时间是小于9.2的，所以比较2和4的运行时间，4短，所以4运行，运行完成时间为9.3，最后2运行

开始时间 + 运行时间

周转时间 / 运行时间

顺序：1 3 4 2
平均周转时间：（1.0 + 1.3 + 0.2 + 0.2 ）/ 4
平均带权周转时间：（1 + 2.6 + 1 + 2 ）/ 4

3. 优先级调度算法（PSA）
    1. 优先级越高，越紧迫
    2. 既可以用于作业调度，也可以用于进程调度
4. 高响应比调度算法（HRRN）
    1. 优先级 = （等待时间 + 要求服务时间）/ 要求服务时 = 响应时间 / 要求服务时间



- 进程调度算法
    1. 轮转调度算法
    2. 优先级调度算法
    3. 多队列调度算法
    4. 多级反馈队列调度算法
- 3.2.3 先来先服务 短作业优先 会画表 求响应时间，周转时间
    1. 见上图
- 高响应比优先调度算法（了解算法实现原理）
    1. 见上图
- 银行家算法（大题）

1. 死锁定义：如果一组进程中的每一个进程都在等待仅有改组进程中的其他进程才能引发的事件，那么该组进程是死锁的
2. 死锁产生的条件：
    1. 互斥条件：在某段时间内，一个资源只能被一个进程占用
    2. 请求和保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放
    3. 不可抢占条件：进程已获得的资源未使用完之前不能被抢占
    4. 循环等待条件：若干进程之间形成一种头尾相接的循环等待关系
3. 处理死锁的方法：
    1. 预防死锁
    2. 避免死锁（银行家算法）
    3. 检测死锁
    4. 接触死锁
4. 银行家算法

拿书上的题来举例，会了算法就行了

现在有5个进程，3类资源，各个资源数量分别为10、5、7。在T0时刻的资源分配情况如图所示

资源情况 进程	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	1	0	7	4	3	3	3	2
P1	3	2	2	2	0	0	1	2	2			
P2	9	0	2	3	0	2	6	0	0			
P3	2	2	2	2	1	1	0	1	1			
P4	4	3	3	0	0	2	4	3	1			

首先求T0时刻的安全性，求安全性只需要看有没有安全序列

资源情况 进程	Work			Need			Allocation			Work + Allocation			Finish
	A	B	C	A	B	C	A	B	C	A	B	C	
	上一张图给了Available就是可用的资源3 3 2，所以看5个进程中的Need 就是需要资源的数据 看哪一个Need < Available 小于3 3 2 的有P1和P3 按顺序先分配P1 所以P1的Work就是刚才剩下的Available 3 3 2，Need和Allocatin照抄												
P1	3	3	2	1	2	2	2	0	0	5	3	2	true
	P1的5 3 2就相当于是给P1分配完之后回收了，所以也是Available可用的资源，在找Need小于5 3 2 的进程，P3和P4小于，先算P3，P3的Work就是可用的现有的资源为5 3 2												
P3	5	3	2	0	1	1	2	1	1	7	4	3	true
	P3的7 4 3 也是分配给P3之后回收的 最后剩下的可用的资源，找 Need小于 7 4 3 的进程 P4 P2 P0 都小于 随便按顺序P4												
P4	7	4	3	4	3	1	0	0	2	7	4	5	true
	P4的 7 4 5 就是可用的资源，在找Need < 7 4 5 的 P0和P2 写谁都可以，按书写一个P2吧就												
P2	7	4	5	6	0	0	3	0	2	10	4	7	true
	所以最后就是P0 10 4 7												
P0	10	4	7	7	4	3	0	1	0	10	5	7	true
	记得验证最后的10 5 7 和题里面一开始给的各个资源的数量一定得相等												

P1请求资源：P1发出请求向量Request1（1，0，2）
这种的只要记住这三个步骤：
1. 如果Request1（1，0，2） <= Need1（1，2，2），就继续往下走，否则出错，
2. 如果Request1（1，0，2） <= Available1（3，3，2），就继续往下走，否则现有的资源不够，P1等待
3. 可分配给P1，修改数值 Available = Available – Request1
   Allocation1 = Allocation1 + Request1
   Need1 = Need1 – Request1 因为他要请求资源嘛，所以Available和Need 可用的和需要的都要减去请求的，相反，已经分配的就要加上请求的
   所以形成的资源请求情况 如下图

资源情况 进程	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
										Available = Available – Request1		
P0	7	5	3	0	1	0	7	4	3	2	3	0
				Allocation1 = Allocation1 + Request1			Need1 = Need1 – Request1					
P1	3	2	2	3	0	2	0	2	0			
P2	9	0	2	3	0	2	6	0	0			
P3	2	2	2	2	1	1	0	1	1			
P4	4	3	3	0	0	2	4	3	1			

然后根据第一问继续看有没有安全序列

P4请求资源：P4发出请求向量Request4（3，3，0）
这种的只要记住这三个步骤：
1. 如果Request4（3，3，0） <= Need4（4，3，1），就继续往下走，否则出错，
2. 如果Request4（3，3，0） <= Available（2，3，0），就继续往下走，否则现有的资源不够，P4等待
所以P4等待
P0请求资源：P0发出的请求为Request0（0，2，0）
1. 2. 按照上面的步骤都符合，继续往下走
3. 可分配给P0，修改数值 Available = Available – Request0 = 2，3，0 – 0，2，0 = 2，1，0
Allocation0 = Allocation0 + Request0 = 0，1，0 + 0，2，0 = 0，3，0
Need0 = Need0 – Request0 = 7，4，3 – 0，2，0 = 7，2，3
因为他要请求资源嘛，所以Available和Need 可用的和需要的都要减去请求的，相反，已经分配的就要加上请求的
所以形成的资源请求情况 如下图

资源情况 进程	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
										Available = Available – Request1		
P0	7	5	3	0	3	0	7	2	3	2	1	0
				Allocation1 = Allocation1 + Request1			Need1 = Need1 – Request1					
P1	3	2	2	3	0	2	0	2	0			
P2	9	0	2	3	0	2	6	0	0			
P3	2	2	2	2	1	1	0	1	1			
P4	4	3	3	0	0	2	4	3	1			

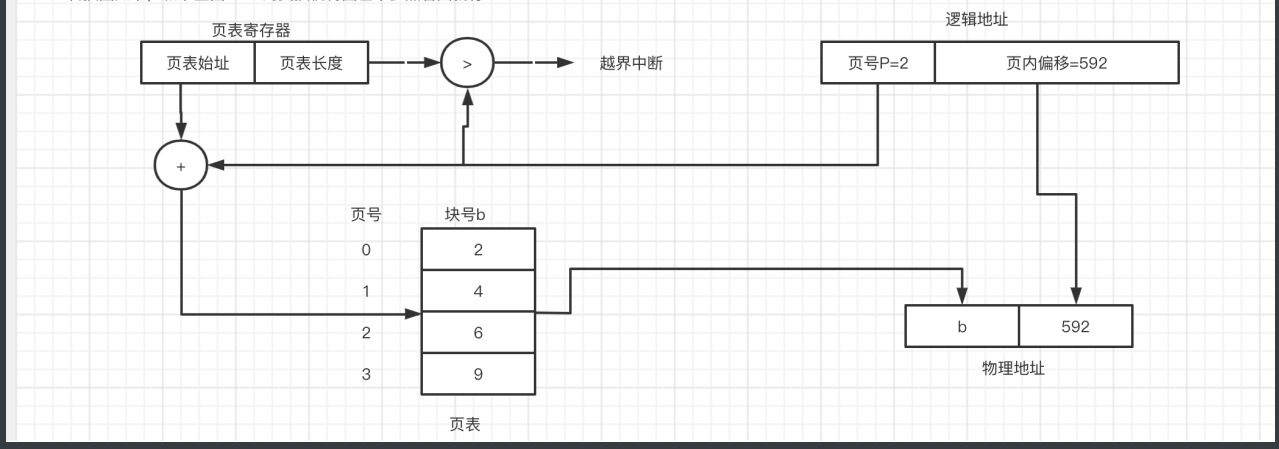
然后根据第一问继续看有没有安全序列，因为可用资源2 1 0 已经没有小于Need的进程了，就是满足不了任何进程的需要了，所以不安全，也就是系统不分配资源

- 第四章
  - 简答题 首次适应 循环适应 算法（了解优缺点）4.3.4
    1. 首次适应算法（FF）
       1. 该算法倾向于优先利用内存中低址部分的空闲分区，从而保留了高址部分的大空闲区。
       2. 缺点是：低址部分不断被划分，会留下许多碎片以及增加查找可用空闲分区时的开销
    2. 循环首次适应算法（NF）
       1. 该算法能使内存中的空闲分区分布的更均匀，从而减少了查找空闲分区时的开销
       2. 缺点：会缺乏大的空闲分区
    3. 最佳适应算法（BF）
       1. 缺点：在存储器中会留下许多难以利用的碎片
    4. 最坏适应算法（WF）
       1. 优点：可使剩下的空闲区不至于太小，产生碎片的可能性最小，查找效率高
  - 分页 简答题 求地址 148页 根据逻辑号寻找物理地址4.5.1

例题：
在采用页式存储管理的系统中，某作业的逻辑地址空间分为4页，（每页2048字节），且已知该作业的页表如下表。试借助地址转换图（即要求画出页式存储管理系统地址转换示意图）求出逻辑地址4688所对应的物理地址

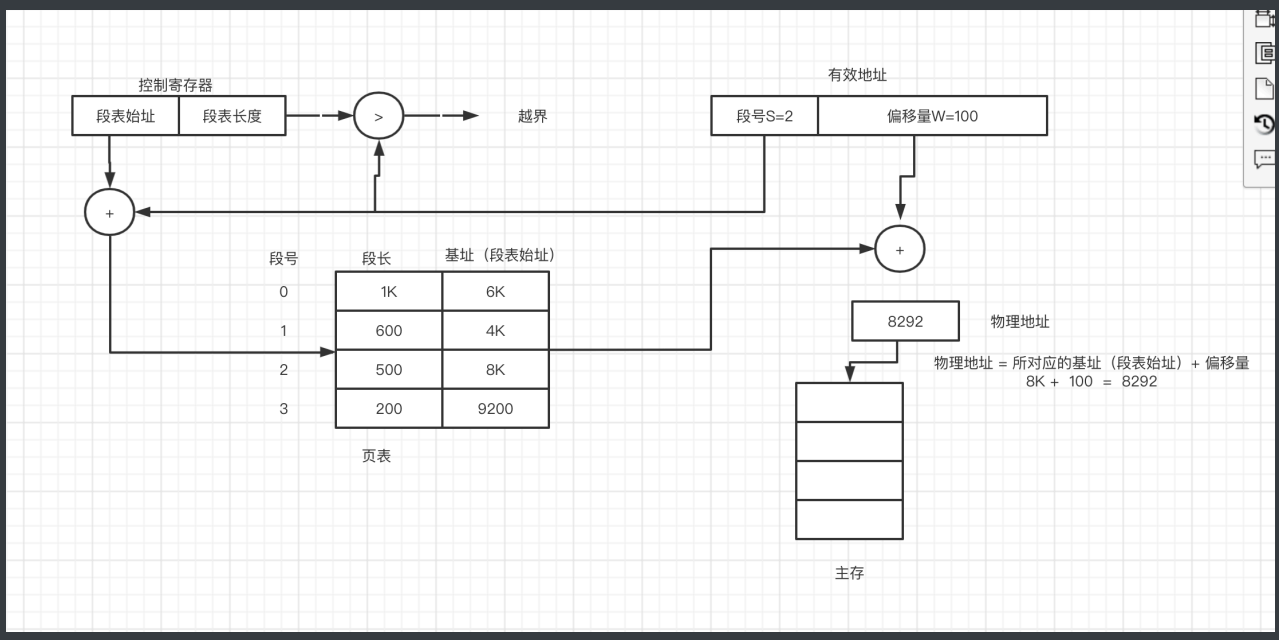
页号	块号
0	2
1	4
2	6
3	9

逻辑地址4688所对应的
首先用逻辑地址空间 / 页大小
4688 / 2048 = 2.....592
其中商也就是2 是页号P
余数592是页内地址，也叫页偏移
物理地址 = 所对应的块号 * 页面大小 + 页偏移 = 6 * 2048 + 592 = 12880
转换图如下，和书里面4.5.2的变换机构图差不多照着画就行



- 地址变换机构（了解）分页访问几次内存 分段访问几次 短页式几次

1. 地址变换任务就是借助于页表来完成的
2. 下面的是段式地址变换机构，上面的是页式



1. 访问内存

   分页：两次；

   一是访问内存中存放的页表，实现地址变换,得到真正的物理地址；

   二是访问真正的物理地址得到相应的指令或数据

分段：两次；

一是访问内存中存放的段表,实现地址变换,得到真正的物理地址；

二是访问真正的物理地址得到相应的指令或数据

段页式：三次；

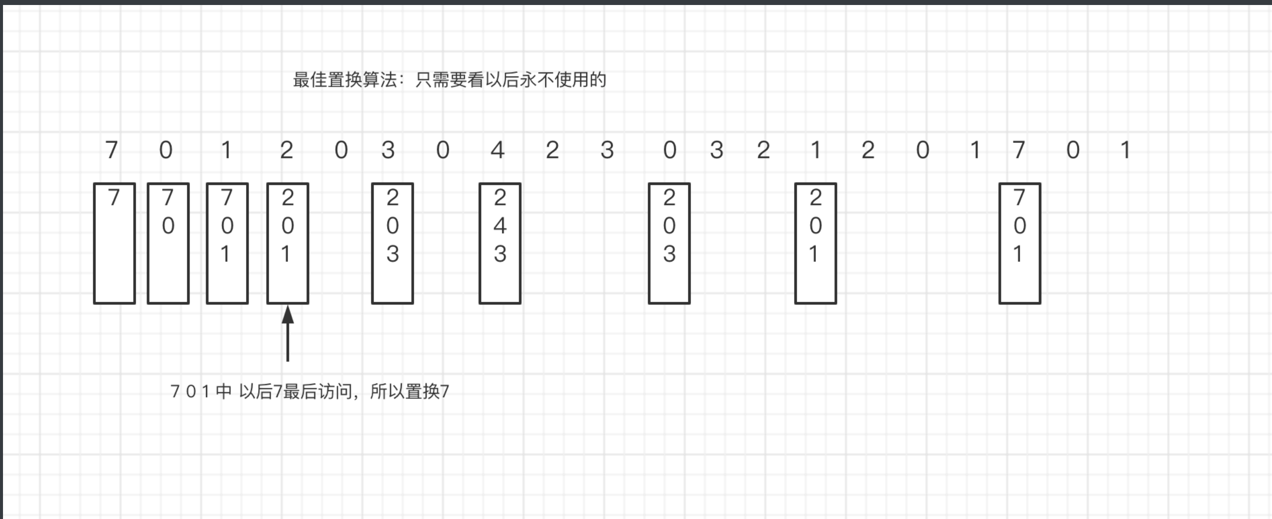
一是访问内存中存放的段表,查找段内页表的起始地址；

二是访问内存中存放的页表，实现逻辑地址到物理地址的变换；

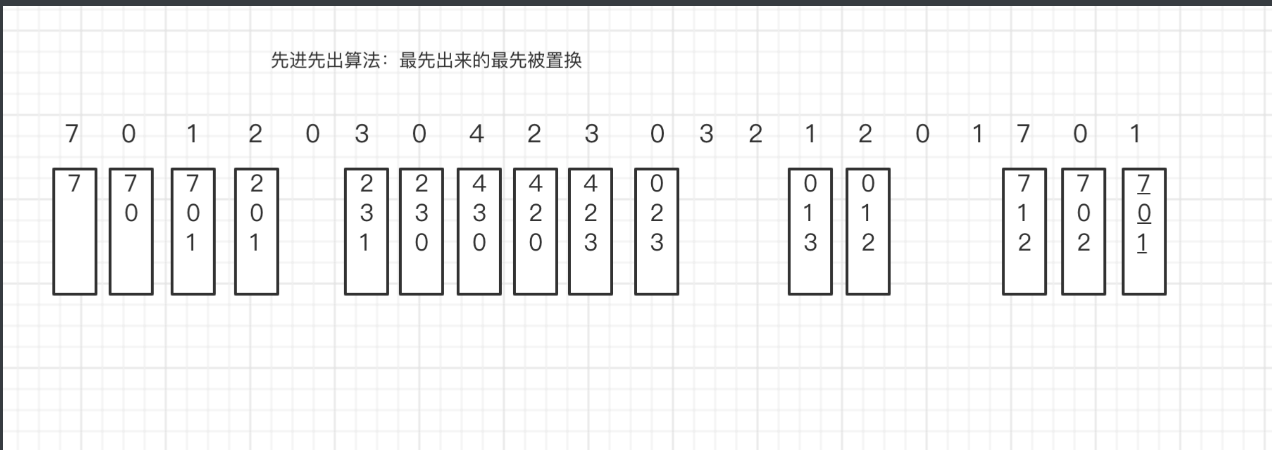
三是访问真正的物理地址得到相应的指令或数据

- 分段为甚么提出来 分页不行吗
  1. 页是信息的物理单位，分页是系统的行为，对用户不可见，分段则可以更好的满足用户的需要
  2. 页的大小固定且由系统决定，而段长度缺不固定
  3. 分页的用户程序地址空间是一维的，用户程序的地址是属于单一的线性地址空间，而分段是用户的行为，用户程序的空间地址是二维的
- 分页 分段 整体

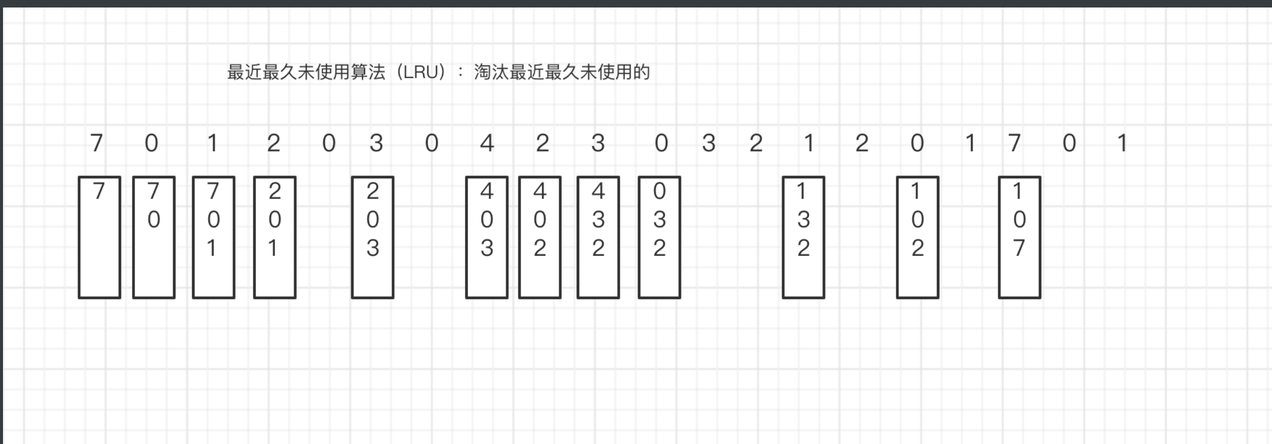
- 第五章
  - 虚拟存储器特征，定义
    - 定义：具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的一种存储器系统
    - 特征：
      - 多次性
      - 对换性
      - 虚拟性
  - 页面置换算法（大题）
    1. 最佳置换算法
    2. 缺页率最低



1. 先进先出置换算法



1. 最近最久未使用算法（LRU）



1. 最少使用算法（LFU）

  - 最佳 先进先出 算法 LRU （重点）

- 第六章
  - 选择填空多
  - 寻找磁盘 早期磁盘调度算法 FCFS（大题）

1. 先来先服务（FCFS）

SSTF写错了，是FCFS

例：假设磁盘有200个磁道，现在随机请求磁盘队列，他们到达的次序分别处于55，58，39，18，90，160，150，38，184，当前磁头在100号磁道上，并正由外向里移动，求按照FCFS、SSTF、SCAN、CSCAN算法进行磁盘调度算法时候的请求次序，并计算相应的平均寻道长度

SSTF就根据请求访问的先后次序就行
（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	100–55 = 45
58	58–55 = 3
39	58 – 39 = 19
18	21
90	72
160	70
150	10
38	112
184	146

平均寻道长度 = （45 + 3 + 19 + 21 + 72+ 70 + 10 + 112 + 146） / 9 = 55.3

1. 最短寻道时间优先（SSTF）

可能出现：饥饿现象

例：假设磁盘有200个磁道，现在随机请求磁盘队列，他们到达的次序分别处于55，58，39，18，90，160，150，38，184，当前磁头在100号磁道上，并正由外向里移动，求按照FCFS、SSTF、SCAN、CSCAN算法进行磁盘调度算法时候的请求次序，并计算相应的平均寻道长度

SSTF最短寻道时间优先，就是按照距离当前磁头最近的进行访问

首先排序一下

18 38 39 55 58 90 150 160 184

⇧

100

（从100号磁道开始）

被访问的下一个磁道号	移动距离（磁道数）
100离90近，所以是90	100–90 = 10
90离58近，所以是58	90–58 = 32
58离55近，所以是55	58 – 55 = 3
55离39近，所以是39	16
38	1
18	20
150	132
160	10
184	24

平均寻道长度 = （10 + 32 + 3 + 16 + 1+ 20 + 132 + 10 + 24） / 9 = 27.5

- Scan Cscan 四种算法（大题）

1. 扫描算法（SCAN）

例：假设磁盘有200个磁道，现在随机请求磁盘队列，他们到达的次序分别处于55，58，39，18，90，160，150，38，184，当前磁头在100号磁道上，并正由外向里移动，求按照FCFS、SSTF、SCAN、CSCAN算法进行磁盘调度算法时候的请求次序，并计算相应的平均寻道长度

SCAN扫描算法 就是模拟做电梯就完了 从100 先网上走 ，走到头再往下走，很简单

首先排序一下

18 38 39 55 58 90 150 160 184

100

（从100号磁道开始）

被访问的下一个磁道号	移动距离（磁道数）
150	50
160	10
184	24
90	94
58	32
55	3
39	16
38	1
18	20

平均寻道长度 = 27.8

1. 循环扫描算法（CSCAN）

例：假设磁盘有200个磁道，现在随机请求磁盘队列，他们到达的次序分别处于55，58，39，18，90，160，150，38，184，当前磁头在100号磁道上，并正由外向里移动，求按照FCFS、SSTF、SCAN、CSCAN算法进行磁盘调度算法时候的请求次序，并计算相应的平均寻道长度

CSCAN扫描算法 就是单方向的模拟做电梯就完了 从100 先网上走 ，走到头再从下往上，很简单

首先排序一下
18 38 39 55 58 90 150 160 184

100 ⇧
（从100号磁道开始）

被访问的下一个磁道号	移动距离（磁道数）
150	50
160	10
184	24
18	166
38	20
39	1
55	16
58	3
90	32

平均寻道长度 = 35.8

- 设备 分配的整个流程（顺序）选择填空
  1. 设别分配的数据结构
     1. 设别控制表DCT
     2. 控制器控制表COCT
     3. 通道控制表CHCT
     4. 系统设备表SDT
  2. 设备分配程序
     1. 分配设备
     2. 分配控制器
     3. 分配通道
- I/O设备的控制方式 209页 6.4.3
  1. 使用轮循的可编程I/O方式
  2. 使用中断的可编程I/O方式
  3. 直接存储器访问方式

4. I/O通道控制方式

- 第七章
  - 选择填空
  - 文件的类型 属性 文件的结构
    1. 文件的类型：
       1. 按用途分类：系统文件、用户文件、库文
       2. 按文件中的数据的形式分类：源文件、目标文件、可执行文件
       3. 按存取控制属性分类：只执行文件、只读文件、读写文件
       4. 按组织形式和处理方式分类：普通文件、目录文件、特殊文件
    2. 层次结构：
       1. 对象及其属性
       2. 对对象操纵和挂你的软件集合
       3. 文件系统接口
  - 逻辑结构和物理结构
    1. 书上有定义
  - 逻辑结构包含什么?
    1. 逻辑结构

       按照是否有结构来分

       1. 有结构文件（记录式文件）
       2. 无结构文件（流式文件）

       按照文件的组织方式来分

       1. 顺序文件
       2. 索引文件
       3. 索引顺序文件
- 算法要记住英文单词（缩写）
- 压缩包 班级学号 姓名