计算机考研系列书课包

玩转数据结构

主讲人 刘财政

第七讲　高阶专题

本讲内容

考点一：经典同步问题专题

考点二：存储器管理专题

考点三: 文件管理专题

考点一：

经典同步问题专题

考点框架

- 生产者-消费者问题
- 哲学家进餐问题
- 读者-写者问题
- 服务员-顾客问题
- 其他问题

考点一：经典同步问题专题

问题类型	进程性质	互斥	合作
生产者-消费者问题	性质相同	有（互斥使用缓冲区）	有（相互通知）
哲学家进餐问题	性质相同	有（互斥使用餐具）	无
读者-写者问题	性质不同	有（互斥使用数据区）	无
服务员-顾客问题	性质不同	无	有（相互通知）

考点一：经典同步问题专题

生产者-消费者问题

考点一：经典同步问题专题

1、三个进程 P1、P2、P3 互斥使用一个包含 N(N>0)个单元的缓冲区。P1 每次用 produce()生成一个正 整数并用 put()送入缓冲区某一空单元中；P2 每次用 getodd()从该缓冲区中取出一个奇数并用 countodd()统 计奇数个数；P3 每次用 geteven()从该缓冲区中取出一个偶数并用 counteven()统计偶数个数。请用信号量机 制实现这三个进程的同步与互斥活动，并说明所定义信号量的含义。 要求用伪代码描述。

考点一：经典同步问题专题

定义信号量 odd 控制 P1 与 P2 之间的同步；even 控制 P1 与 P3 之间的同步；empty 控制生产者与消费者之 间的同步；mutex 控制进程间互斥使用缓冲区。程序如下：

semaphore odd = 0, even = 0, empty = N, mutex = 1;

考点一：经典同步问题专题

```
Process P1( ){
   while（true）{
       x = produce();    //生成一个数
       P(empty);          //判断缓冲区是否有空单元
       P(mutex);          //缓冲区是否被占用
        Put();
       V(mutex);          //释放缓冲区
       if(x % 2 == 0)
            V(even);     //如果是偶数，向 P3 发出信号
       else
            V(odd);     //如果是奇数，向 P2 发出信号
       }
}
```

考点一：经典同步问题专题

```
Process P2( ) {
    while（true）{
        P(odd);          //收到 P1 发来的信号，已产生一个奇数
        P(mutex);        //缓冲区是否被占用
        getodd();
        V(mutex);        //释放缓冲区
        V(empty);        //向 P1 发信号，多出一个空单元
        countodd();
    }
}
```

考点一：经典同步问题专题

```
Process P3( ) {
  while（true）{
        P(even);          //收到 P1 发来的信号，已产生一个偶数
        P(mutex);         //缓冲区是否被占用
        geteven();
        V(mutex);         //释放缓冲区
        V(empty);         //向 P1 发信号，多出一个空单元
        counteven();
        }
}
```

考点一：经典同步问题专题

2、系统中有多个生产者进程和多个消费者进程，共享一个能存放 1000 件产品的环形缓冲区（初始为空）。当缓冲区未满时，生产者进程可以放入其生产的一件产品，否则等待；当缓冲区未空时，消费者进程可以从缓冲区取走一件产品，否则等待。要求一个消费者进程从缓冲区连续取出 10 件产品后，其他消费者进程才可以取产品。请使用信号量 P, V(wait(), signal())操作实现进程间的互斥与同步，要求写出完整的过程，并说明所用信号量的含义和初值。

考点一：经典同步问题专题

这是典型的生产者和消费者问题，只对典型问题加了一个条件，只需在标准模型上新加 一个信号量，即可完成指定要求。 设置四个变量 mutex1、mutex2、empty 和 full，mutex1，用于控制一个消费者进程 一个周期(10 次)内对于缓冲区的控制，初值为 1，mutex2 用于进程单次互斥的访问缓冲区， 初值为 1，empty 代表缓冲区的空位数，初值为 0，full 代表缓冲区的产品数，初值为 1000， 具体进程的描述如下：

semaphore mutex1=1;

semaphore mutex2=1;

semaphore empty=1000;

semaphore full=0;

考点一：经典同步问题专题

```
producer(){
    while(1){
        生产一个产品;
        P(empty);          //判断缓冲区是否有空位
        P(mutex2);         //互斥访问缓冲区
        把产品放入缓冲区;
        V(mutex2);         //互斥访问缓冲区
        V(full);           //产品的数量加 1
    }
}
```

考点一：经典同步问题专题

```
consumer(){
    while(1){
        P(mutex1);          //连续取 10 次
        for(int i = 0; i < 10; ++i){
            P(full);        //判断缓冲区是否有产品
            P(mutex2);      //互斥访问缓冲区
            从缓冲区取出一件产品;
            V(mutex2);      //互斥访问缓冲区
            V(empty);       //腾出一个空位
            消费这件产品;
        }
        V（mutex1）
    }
}
```

考点一：经典同步问题专题

3、有 A、B 两人通过信箱进行辩论，每个人都从自己的信箱中取得对方的问题。将答案和向对方提出的新问题组成一个邮件放入对方的邮箱中。假设 A 的信箱最多放 M 个邮件，B 的信箱最多 放 N 个邮件。初始时 A 的信箱中有 x 个邮件（0<x<M），B 的信箱中有 y 个（0<y<N）。辩论者每取出 一个邮件，邮件数减 1。A 和 B 两人的操作过程描述如下：

CoBegin

A{	B{
while(TRUE){	while(TRUE){
从 A 的信箱中取出一个邮件；	从 B 的信箱中取出一个邮件；
回答问题并提出一个新问题；	回答问题并提出一个新问题；
将新邮件放入 B 的信箱；	将新邮件放入 A 的信箱；
}	}
}	}

CoEnd

考点一：经典同步问题专题

当信箱不为空时，辩论者才能从信箱中取邮件，否则等待。当信箱不满时，辩论者才能将新邮件放入信箱，否则等待。请添加必要的信号量和 P、V（或 wait、signal）操作，以实现上述过程的同步。 要求写出完整过程，并说明信号量的含义和初值。

考点一：经典同步问题专题

```
semaphore Full_A = x;          //Full_A 表示 A 的信箱中的邮件数量
semaphore Empty_A = M-x;       // Empty_A 表示 A 的信箱中还可存放的邮件数量
semaphore Full_B = y;          //Full_B 表示 B 的信箱中的邮件数量
semaphore Empty_B = N-y;       // Empty_B 表示 B 的信箱中还可存放的邮件数量
semaphore mutex_A = 1;         //mutex_A 用于 A 的信箱互斥
semaphore mutex_B = 1;         //mutex_B 用于 B 的信箱互斥
```

考点一：经典同步问题专题

```
Process A{
    while(TRUE){
        P(Full_A);
        P(mutex_A);
        从 A 的信箱中取出一个邮件;
        V(mutex_A);
        V(Empty_A);
        回答问题并提出一个新问题;
        P(Empty_B);
        P(mutex_B);
        将新邮件放入 B 的信箱;
        V(mutex_B);
        V(Full_B);
    }
}
```

```
Process B{
    while(TRUE){
        P(Full_B);
        P(mutex_B);
        从 B 的信箱中取出一个邮件;
        V(mutex_B);
        V(Empty_B);
        回答问题并提出一个新问题;
        P(Empty_A);
        P(mutex_A);
        将新邮件放入 A 的信箱;
        V(mutex_A);
        V(Full_A);
    }
}
```

考点一：经典同步问题专题

4、现有 5 个操作 A、B、C、D 和 E，操作 C 必须在 A 和 B 完成后执行，操作 E 必须在 C 和 D 完成后执行，请使用信号量的 wait(), signal(),操作（P、 V 操作） 描述上述操作之间的同步关系，并说明所用信号量及其初值。

考点一：经典同步问题专题

本题要求实现操作的先后顺序，没有互斥关系，是一个简单的同步问题。本题虽然有5 个操作，但是只有4 个同步关系，因此分别设置信号量SAC 、SBC、SCE和SDE对应4个同步关系。

```
Semaphore SAC = 0;        //控制A和C的执行程序
Semaphore SBC = 0;        //控制B和C的执行程序
Semaphore SCE = 0;         //控制C和E的执行程序
Semaphore SDE = 0;         //控制D和E的执行程序
```

考点一：经典同步问题专题

5个操作可描述如下：

```
Process A(){
 完成动作A；
 V(SAC);          //实现A、C之间的同步关系
}
Process B(){
 完成动作B；
 V(SBC);          //实现B、C之间的同步关系
}
```

```
Process C(){
  //C必须在A、B都完成后才能完成
  P(SAC);
  P(SBC)；
  完成动作C；
 //实现C、E之间的同步关系
  V(SCE);
}
```

考点一：经典同步问题专题

```
Process D(){
        完成动作D;
         V(SDE);
         //实现D、E之间的同步关系
}
Process E(){
        //E必须完成C、D后才能执行
        P(SCE);
        P(SDE);
        完成动作E;
}
```

考点一：经典同步问题专题

5、系统运行有三个进程：输入进程、计算进程和打印进程，它们协同完成工作。输入进程和计算进程之间共用缓冲区buffer1,计算进程和打印进程之间共用缓冲区buffer2。输入进程接收外部数据放入buffer1中;计算进程从buffer1中取出数据进行计算，然后将结果放入buffer2;打印进程从buffer2取出数据打印输出。

用算法描述这三个进程的工作情况，并用wait和signal原语实现其同步操作。

考点一：经典同步问题专题

输入进程→ buf1→计算进程→buf2→打印进程

从键盘输入到打印机输出的数据传送过程，可以看作是由键盘输入进程到计算进程,以及由计算进程到打印输出进程这两个数据传送进程所组成。

其中，对键盘输入进程而言，计算进程是消费者进程;而对打印输出进程而言,计算进程又是生产者进程。据此可将它们之间的同步问题描述如下:

var:mutex1,mutex2,empty1,empty2,full1,full2 = 1,1,1,1,0,0;

考点一：经典同步问题专题

```
Process Produce():begin
 repeat
        P(empty1);//生产者测试是否有空闲
        P(mutex1);//测试是否互斥
          input a data from keyboard;
          Add to buffer1;
        V(mutex1);//离开并修改互斥标志
        V(full1);//生产者生产了资源，修改资源个数
 until false
end
```

考点一：经典同步问题专题

```
Process CalP() :begin
        repeat
         P(full1);              //消费者测试是否有资源
         P(mutex1);     //测试是否互斥
           Take a data from buffer1;
           Add to ch1;
         V(mutex1);      //离开并修改互斥标志
         V(empty1);     //消费者使用资源，再次有空闲
           calculate ch1;
         P(empty2);     //生产者测试是否有空闲
         P(mutex2);    //测试是否互斥
           Take a data from ch1;
           Add to buffer2;
         V(mutex2);    //离开修改互斥标志
         V(full2);        //生产者生产了资源，修改资源个数
        until false
end
```

考点一：经典同步问题专题

```
Process  PutP() :begin
  repeat
        P(full2);//消费者测试是否有资源
        P(mutex2);//并测试修改互斥标志
          Take a data from buffer2;
          Add to printer controler;
        V(mutex2);//修改互斥标志
        V(empty2);//消费者使用资源，再次有空闲
       start printer;
   until false
end
```

考点一：经典同步问题专题

6、桌上有一空盘，允许存放一只水果。爸爸可向盘中放苹果，也可向盘中放桔子，儿子专等吃盘中的桔子，女儿专等吃盘中的苹果。规定当盘空时一次只能放一只水果供吃者取用，请用P、V原语实现爸爸、儿子、女儿三个并发进程的同步。

考点一：经典同步问题专题

在本题中，爸爸、儿子、女儿共用一个盘子，盘中一次只能放一个水果。当盘子为空时，爸爸可将一个水果放入果盘中。若放入果盘中的是桔子，则允许儿子吃，女儿必须等待；若放入果盘中的是苹果，则允许女儿吃，儿子必须等待。本题实际上是生产者-消费者问题的一种变形。这里，生产者放入缓冲区的产品有两类，消费者也有两类，每类消费者只消费其中固定的一类产品。

typedef int semaphore;//设置四个信号量
semaphore empty=1;//盘子是否为空
semaphore mutex=1;//一次只能进行一个操作：取或者放
semaphore orange=0;
semaphore apple=0;

考点一：经典同步问题专题

```
Process Father() {
    while(true) {
        P(empty);
        P(mutex);
        put_in();//放入桔子或者苹果
        V(mutex);
        if(放入苹果)
            V(apple);
        else
            V(orange);
    }
}
```

考点一：经典同步问题专题

```
Process Sun() {
    while(true){
        P(orange);
        P(mutex);
        orange_out();  //儿子拿走桔子
        V(mutex);
        V(empty);
    }
}
```

考点一：经典同步问题专题

```
Process Daughter() {
    while(true) {
        P(apple);
        P(mutex);
        apple_out();  //女儿拿走苹果
        V(mutex);
        V(empty);
    }
}
```

考点一：经典同步问题专题

7、设有一个可以装A、B两种物品的仓库, 其容量无限大, 但要求仓库中A、B两种物品的数量满足下述不等式: -M≤A物品数量 - B物品数量≤N 其中M和N为正整数. 试用信号量和PV操作描述A、B两种物品的入库过程.

考点一：经典同步问题专题

已知条件-M≤A物品数量 - B物品数量≤N 可以拆成两个不等式，即

- A物品数量 - B物品数量≤N
- B物品数量 - A物品数量≤M

这两个不等式的含义是：仓库中A物品可以比B物品多，但不能超过N个； B 物品可以比A物品多，但不能超过M个。

semaphore mutex=1,a,b=m,empty1,empty2=N,full1,full2=0;

考点一：经典同步问题专题

- 若只放入A，而不放入B，则A产品最多可放入N次便被阻塞；
- 若只放入B，而不放入A，则B产品最多可放入M次便被阻塞；
- 每放入一次A,放入产品B的机会也多一次；
- 同理，每放入一次B,放入产品A的机会也多一次.

考点一：经典同步问题专题

```
semaphore mutex=1，sa=N，sb=M;
procedure A(){
  while(TURE) {
     P(sa);
     P(mutex);
     A产品入库;
     V(mutex);
     V(sb);
  }
}
```

```
procedure B(){
   while(TURE){
      P(sb);
      P(mutex);
      B产品入库;
      V(mutex);
      V(sa);
   }
}
```

考点一：经典同步问题专题

8、设有一个可以装A、B两种物品的仓库，其容量有限（分别为N），但要求仓库中A、B两种物品的数量满足下述不等式:

-M≤A物品数量 - B物品数量≤N

其中M和N为正整数。另外，还有一个进程消费A,B，一次取一个A，B组装成C。试用信号量和PV操作描述A、B两种物品的入库过程。

考点一：经典同步问题专题

已知条件-M≤A物品数量 - B物品数量≤N 可以拆成两个不等式，即

- A物品数量 - B物品数量≤N
- B物品数量 - A物品数量≤M

这两个不等式的含义是：仓库中A物品可以比B物品多，但不能超过N个； B 物品可以比A物品多，但不能超过M个。

semaphore mutex=1,a,b=m,empty1,empty2=N,full1,full2=0;

考点一：经典同步问题专题

A物品入库：

```
process A(){
  while(TRUE) {
    p(empty1);
    P(a);
    P(mutex);
    A物品入库;
    V(mutex);
   V (b);
   V(full1);
  }
}
```

B物品入库：

```
process B(){
   while(TRUE) {
    p(empty2);
    P(b);
    p(mutex);
    B物品入库;
    V(mutex);
    V(a);
    V(full2);
  }
}
```

```
process C(){
   while(TRUE) {
     P(full1);
     P(full2);
     P(a);
     P(b);
    组装;
     V(a);
     V(b);
     V(empty1);
     V(empty2);
  }
}
```

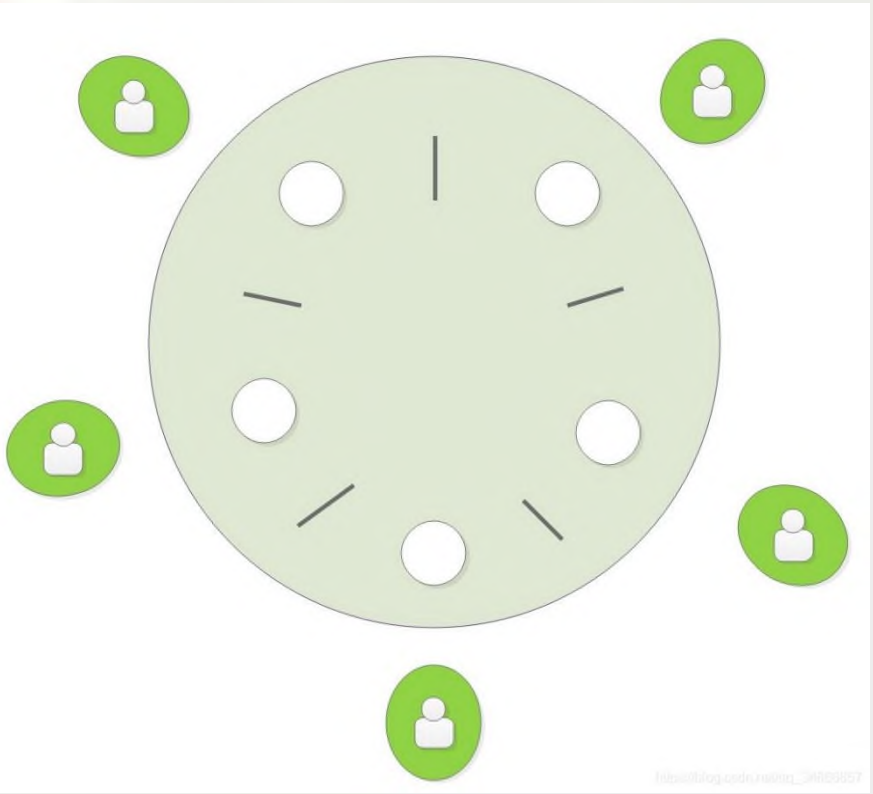
考点一：经典同步问题专题

哲学家进餐问题

考点一：经典同步问题专题

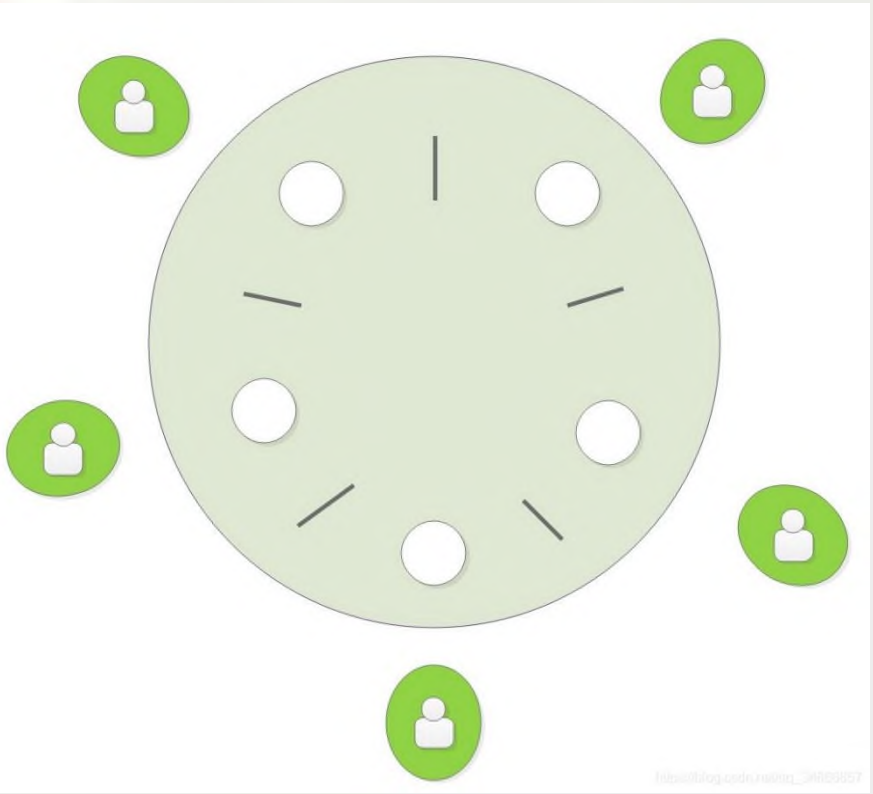
有五个哲学家围在一张圆桌，分别坐在周围的五张椅子上，在圆桌上有五个碗和五支筷子，他们的生活方式是交替的进行思考和进餐。平时，一个哲学家进行思考，饥饿时便试图取用其左右最靠近他的筷子，只有在他拿到两支筷子时才能进餐。进餐完毕后，放下筷子继续思考。

考点一：经典同步问题专题



- 我们可以从上面的题目中得出，筷子是临界资源，同一根筷子同一时刻只能有一个哲学家可以拿到。

考点一：经典同步问题专题



- 由问题描述我们可以知道，一共有五个哲学家，也就是五个进程；五支筷子，也就是五个临界资源；
- 哲学家想要进餐，必须要同时获得左边和右边的筷子，这就是要同时进入两个临界区（使用临界资源），才可以进餐。
- semaphore mutex[5] = {1,1,1,1,1}; //初始化信号量

考点一：经典同步问题专题

```
semaphore mutex[5] = {1,1,1,1,1};                    //初始化信号量
void philosopher(int i){
  while(true){
     //thinking                    //思考
     P(mutex[i]);                  //判断缓冲池中是否仍有空闲的缓冲区
     P(mutex[(i+1)%5]);     //判断是否可以进入临界区（操作缓冲池）
     //eat                         //进餐
     V(mutex[i]);                 //退出临界区，允许别的进程操作缓冲池
     V(mutex[(i+1)%5]);   //缓冲池中非空的缓冲区数量加1，可以唤醒等待的消费者进程
  }
}
```

考点一：经典同步问题专题

可采取以下几种解决方法：

(1) 至多只允许有四位哲学家同时去拿左边的筷子，最终能保证至少有一位哲学家能够进餐，并在用毕时能释放出他用过的两只筷子，从而使更多的哲学家能够进餐。

(2) 仅当哲学家的左、右两只筷子均可用时，才允许他拿起筷子进餐。

考点一：经典同步问题专题

可采取以下几种解决方法：

(3) 规定奇数号哲学家先拿他左边的筷子，然后再去拿右边的筷子；而偶数号哲学家则相反。按此规定，将是1、 2号哲学家竞争1号筷子；3、4号哲学家竞争3号筷子。即五位哲学家都先竞争奇数号筷子，获得后，再去竞争偶数号筷子，最后总会有一位哲学家能获得两只筷子而进餐。

考点一：经典同步问题专题

```
semaphore mutex[5] = {1,1,1,1};                    //初始化信号量
void philosopher(int i){
  while(true){
     //thinking                    //思考
     P(mutex[i]);                  //判断缓冲池中是否仍有空闲的缓冲区
     P(mutex[(i+1)%5]);     //判断是否可以进入临界区（操作缓冲池）
     //eat                         //进餐
     V(mutex[i]);                 //退出临界区，允许别的进程操作缓冲池
     V(mutex[(i+1)%5]);   //缓冲池中非空的缓冲区数量加1，可以唤醒等待的消费者进程
  }
}
```

考点一：经典同步问题专题

```
semaphore mutex[5] = {1,1,1,1};                    //初始化信号量
void philosopher(int i){
  while(true){
     //thinking                    //思考
     P(mutex[i]);                  //判断缓冲池中是否仍有空闲的缓冲区
     P(mutex[(i+1)%5]);     //判断是否可以进入临界区（操作缓冲池）
     //eat                         //进餐
     V(mutex[i]);                 //退出临界区，允许别的进程操作缓冲池
     V(mutex[(i+1)%5]);   //缓冲池中非空的缓冲区数量加1，可以唤醒等待的消费者进程
  }
}
```

用资源限制人数

考点一：经典同步问题专题

2、有 n(n≥3)位哲学家围坐在一张圆桌边，每位哲学家交替地就餐和思考。在圆桌中心有 m(m≥1)个碗，每两位哲学家之间有 1 根筷子。每位哲学家必须取到一个碗和两侧的筷子之后，才能就餐，进餐完毕，将碗和筷子放同原位，并继续思考。为使尽可能多的哲学家同时就餐，且防止出现死锁现象．请使用信号量的 p、v 操作(wait( )、signal( ) 操作)描述上述过程中的互斥与同步，并说明所用信号量及初值的含义。

考点一：经典同步问题专题

- 可以用碗这个限制资源来避免死锁：
- 当碗的数量m小于哲学家的数量n时，可以直接让碗的资源量等于m，确保不会出现所有哲学家都拿一侧筷子而无限等待另一侧筷子进而造成死锁的情况；
- 当碗的数量m大于等于哲学家的数量n时，为了让碗起到同样的限制效果，我们让碗的资源量等于n-1，这样就能保证最多只有n-1个哲学家同时进餐，所以得到碗的资源量为min{n-1，m}。在进行PV 操作时，碗的资源量起限制哲学家取筷子的作用，所以需要先对碗的资源量进行P 操作。

考点一：经典同步问题专题

```
semaphore bowl;                    //用于协调哲学家对碗的使用
semaphore chopsticks[n];           //用于协调哲学家对筷子的使用
for(int i=0;i<n;i++)
   chopsticks[i]=1;                //设置两个哲学家之间筷子的数量
bowl=min(n-1,m);                   //bowl≤n-1,保证不死锁
while(TRUE) {                      //哲学家i的程序
  思考;
  P(bowl);                         //取碗
  P(chopsticks[i]);                //取左边筷子
  P(chopsticks[i+1]%n);            //取右边筷子
  就餐;
  V(chopsticks[i])
   V(chopsticks[i+1]%n);
   V(bowl);
}
```

考点一：经典同步问题专题

服务员-顾客问题

考点一：经典同步问题专题

1、某银行提供 1 个服务窗口和 10 个供顾客等待的座位。顾客到达银行时，若有空座位，则到取号机上领取一个号，等待叫号。取号机每次仅允许一位顾客使用。当营业员空闲时，通过叫号选取一位顾客，并为其服务。顾客和营业员的活动过程描述如下：

```
cobegin {
  process 顾客 i {
    从取号机获取一个号码;
    等待叫号;  获取服务;
  }
  process 营业员 {
    while（TRUE）{
    叫号;
    为客户服务;
    }
  }
}coend
```

请添加必要的信号量和 P、V（或 wait()、signal()）操作，实现上述过程中的互斥与同步。要求写出完整的过程，说明信号量的含义并赋初值。

考点一：经典同步问题专题

```
semaphore seets = 10;        // 有 10 个坐位的资源信号量
semaphore  mutex = 1;            // 取号机互斥信号量
semaphore  haveCustom = 0;       // 顾客与营业员同步，无顾客时营业员休息
semaphore service = 1            // 顾客与营业员同步，表示有一名营业员
process 顾客 {
    P(seets); // 等空位
    P(mutex); // 申请使用
    取号机从取号机上取号;
    V(mutex); // 取号完毕
    V(haveCustom); // 通知营业员有新顾客到来
    等待营业员叫号;
    P（service）
    接受服务；
}
```

考点一：经典同步问题专题

```
process 营业员 {
    while(True) {
        叫号;
        P(haveCustom); // 没有顾客则休息
        为顾客服务
        V（service）
        V(seets); // 离开坐位
    }
}
```

考点一：经典同步问题专题

2、假设一个系统中有三个抽烟者进程，每个抽烟者不断地卷烟并抽烟。抽烟者卷起并抽掉一颗烟需要有三种材料：烟草、纸和胶水。一个抽烟者有烟草，一个有纸，另一个有胶水。系统中还有两个供应者进程，它们无限地供应所有三种材料，但每次仅轮流提供三种材料中的两种。得到缺失的两种材料的抽烟者在卷起并抽掉一颗烟后会发信号通知供应者，让它继续提供另外的两种材料。这一过程重复进行。

考点一：经典同步问题专题

抽烟者问题可描述如下：

```
int  random;
Semaphore  offer1 = 0;  定义烟草和纸的组合信号量
Semaphore offer2 = 0;  定义烟草和胶水的组合信号量
Semaphore  offer3 = 0;  定义胶水和纸的组合信号量
Semaphore finish = 0;  定义抽烟是否完成的信号量
Process P1(){
   while(1){
    random = random();
    random = random % 3;
    if( random == 0 )
       v(offer1);
    else if(random == 1)
        v(offer2);
    else   v(offer3);
    p(finish)
  }
}
```

考点一：经典同步问题专题

抽烟者问题可描述如下：

```
Process P2(){
   while(1){
       p(offer3);
       拿纸和胶水，卷成烟
      v(finish)
    }
}
```

```
Process P3(){
   while(1){
       p(offer2);
       拿烟草和胶水，卷成烟
      v(finish)
    }
}
```

```
Process P4(){
   while(1){
       p(offer1);
       拿纸和烟草，卷成烟
      v(finish)
    }
}
```

考点一：经典同步问题专题

3、面包师有很多面包，由n个销售人员推销。每个顾客进店后取一个号，并且等待叫号，当一个销售人员空闲下来时，就叫下一个号。试设计一个使销售人员和顾客同步的算法。

考点一：经典同步问题专题

顾客进店后按序取号，并等待叫号，销售人员空闲之后也是按序叫号，并销售面包。因此同步算法只要对顾客取号和销售人员叫号进行合理的同步即可。我们使用两个变量i和j分别记录当前的取号值和叫号值，并各自使用一个互斥信号用于对i和j的访问和修改。

```
int i=0, j=0;
semaphore mutex_i=1,mutex_j=1;
Process  Consumer() {
        //进入面包店
       p(mutex_i);
       //取号i
       i++;
      v(mutex_i);
     //等待叫号i并购买面包
}
```

考点一：经典同步问题专题

```
Process Seller() {
    while(1) {
            p(mutex_j);
            if(j<i) {    //号j已有顾客取走并等待
                //叫号j
                 j++;
                v(mutex_j);
               //销售面包
            } else {
                v(mutex_j);
               //休息片刻
            }
     }
}
```

考点一：经典同步问题专题

读者-写者问题

考点一：经典同步问题专题

1、读者-写者问题，如果一直有源源不断地读者，那么写者将处于饥饿状态，如何实现公平的读者-写者问题。

考点一：经典同步问题专题

- 为实现Reader与Writer进程间在读或写时的互斥而设置了一个互斥信号量Wmutex（可写否）

考点一：经典同步问题专题

- 设置一个整型变量Readcount表示正在读的进程数目。由于只要有一个Reader进程在读，便不允许Writer进程去写。因此，仅当Readcount=0，表示尚无Reader进程在读时，Reader进程才需要执行Wait(Wmutex)操作。若wait(Wmutex)操作成功，Reader进程便可去读，相应地，做Readcount+1操作。同理，仅当Reader进程在执行了Readcount减1操作后其值为0时，才须执行signal(Wmutex)操作，以便让Writer进程写。又因为Readcount是一个可被多个Reader进程访问的临界资源，因此，应该为它设置一个互斥信号量rmutex

考点一：经典同步问题专题

- 为了实现读者和写者公平，必须设置一个mutex，当写者早到时，可以限制后续读者进入

考点一：经典同步问题专题

```
int rcount = 1
semaphore rmutex = 1;       //用于读者进程互斥修改rcount;
semaphore wmutex = 1;       //用于读者和写者互斥访问文件
semaphore mutex = 1;        //用于实现公平竞争
```

考点一：经典同步问题专题

```
process reader(){
    P(mutex);
    P(rmutex);
    if(rcount == 0)
        P(wmutex);
    rcount++;
    V(rmutex);
    V(mutex);
    do_reading;
    P(rmutex);
    rcount--
    if(rcount == 0)
        V(wmutex);
    V(rmutex);
}
```

```
process writer(){
    P(mutex);
    P(wmutex);
    do_writing;
    V(wmutex);
    V(mutex);
}
```

考点一：经典同步问题专题

2、一个主修动物行为学、辅修计算机科学的学生参加了一个课题，调查花果山的猴子是否能被教会理解死锁。他找到一处峡谷，横跨峡谷拉了一根绳索（假设为南北方向），这样猴子就可以攀着绳索越过峡谷。只要它们朝着相同的方向，同一时刻可以有多只猴子通过。但是如果在相反的方向上同时有猴子通过则会发生死锁（这些猴子将被卡在绳索中间，假设这些猴子无法在绳索上从另一只猴子身上翻过去）。如果一只猴子想越过峡谷，它必须看当前是否有别的猴子在逆向通过。请使用PV 操作来解决该问题。

考点一：经典同步问题专题

- 由于不允许两个方向的猴子同时跨越绳索，所以对绳索应该互斥使用，但同一个方向可以允许多只猴子通过，所以临界区可允许多个实例访问。
- 本题的难点在于位于南北方向的猴子具有相同的行为，当一方有猴子在绳索上时，同方向的猴子可继续通过，但此时要防止另一方的猴子跨越绳索。
- 类比经典的读者/写者问题，可以发现类似之处，但又不完全相同，因为没有类似的写者。进一步分析可将此题归结为两种读者间的同步与互斥问题。

考点一：经典同步问题专题

```
int southnum=0;
int northnum=0;
semaphore mutex=1;   //绳索互斥访问
semaphore south=1;   //南方猴子互斥信号量
semaphore north=1;   //北方猴子互斥信号量
```

考点一：经典同步问题专题

```
process southi(){
    P(south);
    if(southnum==0)
        P(mutex);
    southnum++;
    V(south);
   //cross越过绳索
    P(south);
    southnum--;
    if(southnum==0)
        V(mutex);
    V(south);
}
```

```
process  northi(){
    P(north);
    if(northnum==0)
        P(mutex);
    northnum++;
    V(north);
   //cross越过绳索
    P(north);
    northnum--;
    if(northnum==0)
        V(mutex);
    V(north);
}
```

考点一：经典同步问题专题

3、假设一个录像厅有0,1，2三种不同的录像片可由观众选择放映，录像厅的放映规则为:

- 任一时刻最多只能放映一种录像片，正在放映的录像片是自动循环放映的，最后一个观众主动离开时结束当前录像片的放映；
- 选择当前正在放映的录像片的观众可立即进入，允许同时有多位选择同一种录像片的观众同时观看，同时观看的观众数量不受限制；
- 等待观看其他录像片的观众按到达顺序排队，当一种新的录像片开始放映时，所有等待观看该录像片的观众可依次序进入录像厅同时观看。用一个进程代表一个观众，
- 要求:用信号量方法PV实现，并给出信号量定义和初始值。

考点一：经典同步问题专题

semaphore mutex =1, mutex0 = 1, mutex1=1, mutex2=2

int count0=0, count1=0, count2=0;

考点一：经典同步问题专题

```
process zero(){
    P(mutex0);
    if(count0==0)
        P(mutex);
   count0++;
    V(mutex0);
   //cross越过绳索
    P(mutex0);
    count0 --;
    if(count0 ==0)
        V(mutex);
    V(count0);
}
```

```
process one(){
    P(mutex1);
    if(count1==0)
        P(mutex);
   count1++;
    V(mutex1);
   //cross越过绳索
    P(mutex1);
    count1 --;
    if(count1 ==0)
        V(mutex);
    V(count1);
}
```

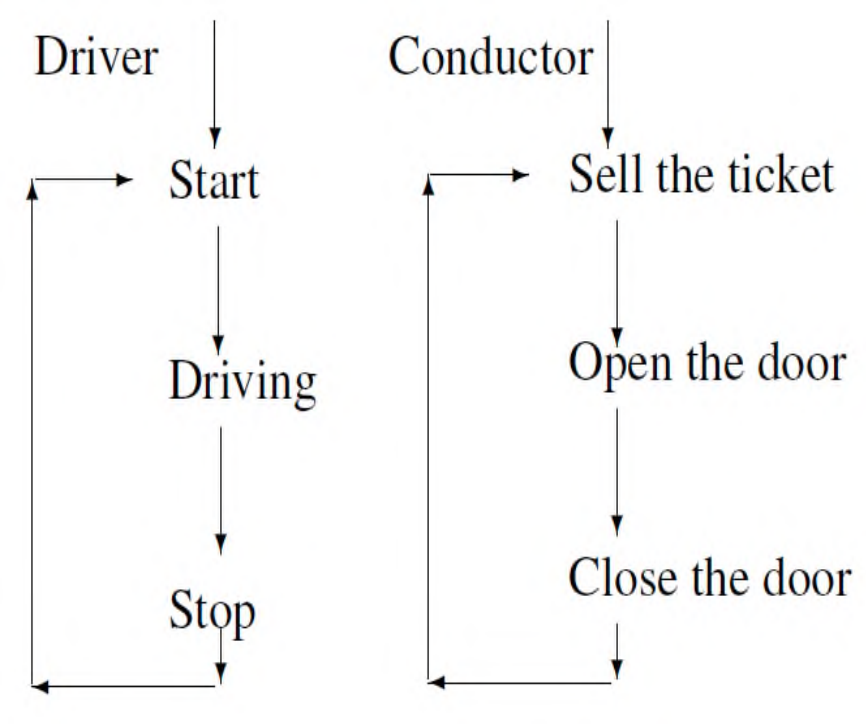
```
process two(){
    P(mutex2);
    if(count2==0)
        P(mutex);
   count2++;
    V(mutex2);
   //cross越过绳索
    P(mutex2);
    count2 --;
    if(count2 ==0)
        V(mutex);
    V(count2);
}
```

考点一：经典同步问题专题

其他问题

考点一：经典同步问题专题

1、设公共汽车上，司机和售票员的活动分别如下：司机的活动：启动车辆：正常行车；到站停车。售票员的活动：关车门；售票；开车门。在汽车不断地到站、停车、行驶过程中，这两个活动有什么同步关系？用信号量和PV 操作实现它们的同步



考点一：经典同步问题专题

在汽车行驶过程中，司机活动与售票员活动之间的同步关系为：售票员关车门后，向司机发开车信号，司机接到开车信号后启动车辆，在汽车正常行驶过程中售票员售票，到站时司机停车，售票员在车停后开门让乘客上下车。因此，司机启动车辆的动作必须与售票员关车门的动作取得同步；售票员开车门的动作也必须与司机停车取得同步。应设置两个信号量：S1 、S2 ；

- S1表示是否允许司机启动汽车（其初值为0 ）
- S2表示是否允许售票员开门（其初值为0 ）

用P 、V 原语描述如下：

考点一：经典同步问题专题

```
var S1,S2 : semaphore ;
S1=0；S2=0；
Procedure driver(){
  while (TRUE) {
      P(S1);
      Start;
      Driving;
      Stop;
      V(S2);
  }
}
```

```
Procedure Conductor(){
  while (TRUE) {
     关车门；
      V (s1);
      售票；
      P(s2);
      开车门；
      上下乘客；
  }
}
```

考点一：经典同步问题专题

2、某寺庙，有小和尚、老和尚若干。庙内有一水缸，由小和尚提水入缸，供老和尚饮用。水缸可容纳10桶水，每次入水、取水仅为1桶，不可同时进行。水取自同一井中，水井径窄，每次只能容纳一个水桶取水。设水桶个数为3个，试用信号灯和PV操作给出老和尚和小和尚的活动。

考点一：经典同步问题专题

- n个小和尚从井里面提水进程之间互斥，用 mutex1 来表示；
- n个小和尚将水倒在缸里、n个老和尚从缸里面取水，这些进程互斥，用mutex2来表示；
- 不管是谁要使用水桶都要确认空闲水桶的数量，用 amount 来表示；
- 用 full 来表示水缸里面有几桶水，用 empty 来表示水缸里面还能放几桶水；

semaphore mutex1=1,mutex2=1;

semaphore amount=3,empty=10,full=0;

考点一：经典同步问题专题

```
void yong_monk i(i=1,2,3,,,n) (){
    while(TRUE){
            P(empty);
            p(amount);
            P(mutex1);
            从井里打水；
            V(mutex1);
            P(mutex2);
            往缸里面倒水；
            V(mutex2);
            V(amount);
            V(full);
    }
}
```

```
void old_monk i(i=1,2,3,,,n) () {
        while(TRUE){
                P(full);
                P(amount);
                P(mutex2);
                从缸里面取水使用；
                V(mutex2);
                V(amount);
                V(empty);
        }
}
```

考点一：经典同步问题专题

3、某博物馆最多可容纳 500 人同时参观，有一个出入口，该出入口一次仅允许一个人通过。参观者的活动描述如下：

cobegin 参观者进程 i：{

…

进门;

…

参观;

…

出门;

…

} coend

请添加必要的信号量和 P、V（或 wait()、signal()）操作，以实现上述过程中的互斥与同步。要求写出完整的过程，说明信号量的含义并赋初值。

考点一：经典同步问题专题

定义两个信号量

```
Semaphore empty = 500;    // 博物馆可以容纳的最多人数
Semaphore mutex = 1;      // 用于出入口资源的控制
参观者进程（）{
    P ( empty );
    P ( mutex );
    进门;
    V( mutex );
    参观;
    P ( mutex );
    出门;
    V( mutex );
    V( empty );
}
```

考点一：经典同步问题专题

4、某进程中有3个并发执行的线程thread1、thread2和thread3，其伪代码如下所示。

```
//复数的结构类型定义
typedef struct
}
    float a;
    float b;
}   cnum;
cnum x，y，z；//全局变量

//计算两个复数之和
cnum add(cnum p，cnum q)
{
    cnum s;
    s.a=p.a+q.a;
    s.b=p.b+q.b;
    return s;
}
```

```
thread1
{
    cnum w;
    w=add(x，y);
    ……
}

thread2
}
    cnum w;
    w=add(y，z);
    ……
}
```

```
thread3
{
    cnum w;
    w.a=1;
    w.b=1;
    z=add(z，w);
    y=add(y，w);
    ……
}
```

请添加必要的信号量和P、V(或wait()、signal())操作，要求确保线程互斥访问临界资源，并且最大程度地并发执行。

考点一：经典同步问题专题

semaphore mutex_y1=1； //mutex_y1用于thread1与thread3对变量y的互斥访问;

semaphore mutex_y2=1； //mutex_y2用于thread2与thread3对变量y的互斥访问;

semaphore mutex_z=1； //mutex_z用于变量z的互斥访问;

互斥代码如下:

```
thread1{
  cnum w;
  wait(mutex_y1);
  w=add(x，y);
  signal(mutex_y1);
}
```

考点一：经典同步问题专题

```
thread2{
  cnum w:
  wait(mutex_y2):
  wait(mutex_z);
  w=add(y，z);
  signal(mutex_z);
  signal(mutex_y2);
}
```

```
thread3{
  cnum w;
  w.a=1;
  w.b=1;
  wait(mutex_z);
  z=add(z，w);
  signal(mutex_z);
  wait(mutex_y1);
  wait(mutex_y2);
  y=add(y, w);
  signal(mutex_y1);
  signal(mutex_y2);
}
```

谢谢大家

考点二：

存储器管理专题

考点框架

- 基本分页方式
- 页面置换算法
- 虚拟存储器

考点二：存储器管理专题

基本分页方式

考点二：存储器管理专题

1、某计算机主存按字节编址，逻辑地址和物理地址都是32位，页表项大小为 4 字节。请回答下列问题。

（1）若使用一级页表的分页存储管理方式，逻辑地址结构为：

页号（20位）	页内偏移量（12位）

则页的大小是多少字节？页表最大占用多少字节？

【解析】（1)因为页内偏移量是12位,所以页大小为4KB=2^{12}(字节),页表项数为2^{32}/4 KB=2^{20},因此该一级页表最大为2^{20}×4B=4 MB=2^{22}(字节)

考点二：存储器管理专题

3、某计算机主存按字节编址，逻辑地址和物理地址都是 32 位，页表项大小为 4 字节。请回答下列问题。

（2）若使用二级页表的分页存储管理方式，逻辑地址结构为：

页目录号（10 位）	页表索引（10 位）	页内偏移量（12 位）

设逻辑地址为 LA，请分别给出其对应的页目录号和页表索引的表达式。

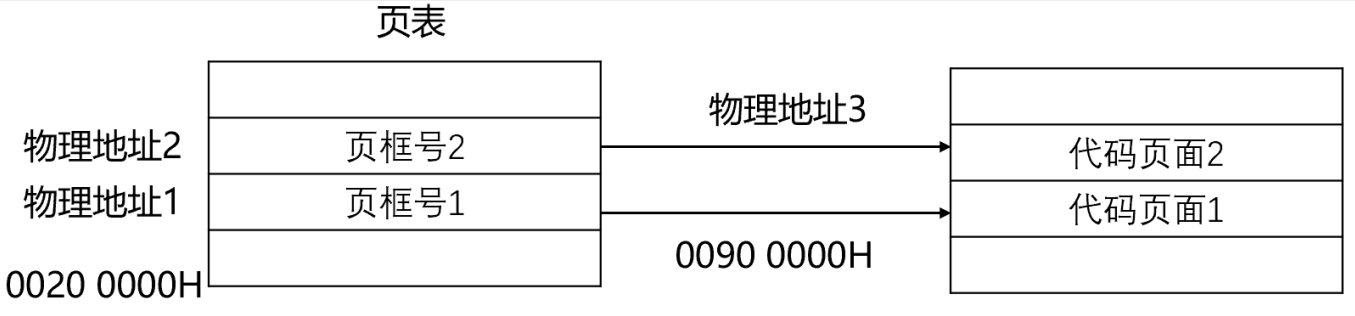
(2)页目录号可表示为LA DIV 2^{22}

页表索引可表示为(LA DIV 2^{12})mod 2^{10}

考点二：存储器管理专题

1、某计算机主存按字节编址，逻辑地址和物理地址都是 32 位，页表项大小为 4 字节。请回答下列问题。

（3）采用（1）中的分页存储管理方式，一个代码段起始逻辑地址为 0000 8000H，其长度 为 8 KB，被装载到从物理地址 0090 0000H 开始的连续主存空间中。页表从主存 0020 0000H 开始的物理地址处连续存放，如下图所示（地址大小自下向上递增）。请计算出该代码段对应的两个页表项的物理地址、这两个页表项中的页框号以及代码页面2的起始物理地址。

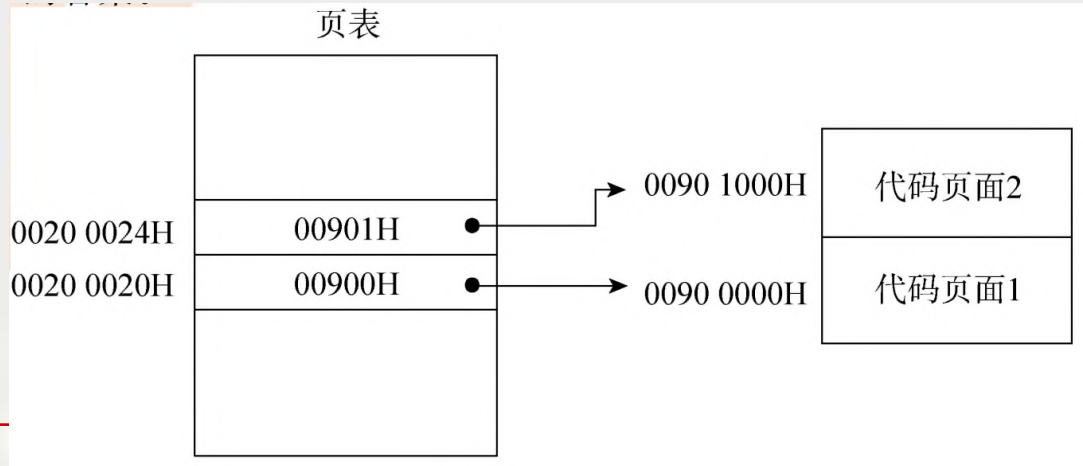


考点二：存储器管理专题

(3)代码页面l的逻辑地址为0000 8000H,表明其位于第8个页处,对应页表中的第8个页表项,所以第8个页表项的物理地址=页表起始地址十8×页表项的字节数=0020 0000H+8×4=0020 0020H,

第9个页表项的物理地址为0020 0000H十9×4=0020 0024H,又由于起始物理地址为0090 0000H,即内存块号为00900,

又代码段占据连续的8 KB,因此内存块号为00900和00901,分别对应物理地址00900000H和0090 1000H,且页表项0020 0020中存放内存块号00900,页表项0020 0024中存放内存块号00901,由此可得如图所示的答案。



考点二：存储器管理专题

2、某计算机系统按字节编址，采用二级页表的分页存储管理方式，虚拟地址格式如下所示：

10 位	10 位	12 位
页目录号	页表索引	页内偏移量

请回答下列问题。

1）页和页框的大小各为多少字节？进程的虚拟地址空间大小为多少页？

1）页和页框大小均为4KB=2^{12}(字节)。

进程的虚拟地址空间大小为$2^{32}/2^{12}=2^{20}$(页)。

考点二：存储器管理专题

2、某计算机系统按字节编址，采用二级页表的分页存储管理方式，虚拟地址格式如下所示：

10位	10位	12位
页目录号	页表索引	页内偏移量

请回答下列问题。

2）假定页目录项和页表项均占 4 个字节，则进程的页目录和页表共占多少页？要求写出计算过程。

（2）因为页目录项和页表项均占4个字节，因此页目录占据的页数为$(2^{12} * 4)/2^{12}$=1页;

页表索引占的页数为$(2^{10}\times 2^{10} * 4)/2^{12}$=1024,

所以页目录和页表共占1025页。

考点二：存储器管理专题

2、某计算机系统按字节编址，采用二级页表的分页存储管理方式，虚拟地址格式如下所示：

10位	10位	12位
页目录号	页表索引	页内偏移量

请回答下列问题。

3）若某指令周期内访问的虚拟地址为 0100 0000H 和 0111 2048H，则进行地址转换时共访问多少个二级页表？要求说明理由。

（3）需要访问一个二级页表。因为虚拟地址0100 0000H=0000 0001 0000 0000 0000 0000 0000B和虚拟地址01112048H=0000 0001 0001 0001 0010 0000 0100 1000B,通过分析地址结构,可知两个虚拟地址的高10位,即页目录号均为4,因此两个地址位于同一个二级页表中。

考点二：存储器管理专题

3、一个进程的大小占5个页面，每页的大小为1 KB，系统为它分配了3个物理块。当前进程的页表如表所示。

页号	块号	存在位 P	访问位 R	修改位 M
0	0x1C	1	1	0
1	0x3F	1	1	1
2	—	0	0	0
3	0x5D	1	0	0
4	—	0	0	0

(1) 有哪些页面不在内存?

(1) 不在内存的是第2和4页(按页号)。

考点二：存储器管理专题

3、一个进程的大小占5个页面，每页的大小为1 KB, 系统为它分配了3个物理块。当前进程的页表如表所示。

页号	块号	存在位 P	访问位 R	修改位 M
0	0x1C	1	1	0
1	0x3F	1	1	1
2	—	0	0	0
3	0x5D	1	0	0
4	—	0	0	0

(2) 请分别计算进程中虚地址为0x3B7 、0x12A5 、0x1432 单元的物理地址（用十六进制表示），并说明理由。

(2) 0x3B7,即2进制的0000 00 11 1011 0111,对应页号为0,该页的存在号为l,表明该块在内存中,把页号0000 00换成1C,即 01 1100,得到二进制的物理地址0111 0011 10110111,物理地址为0x 73 B7。

考点二：存储器管理专题

3、一个进程的大小占5个页面，每页的大小为1 KB, 系统为它分配了3个物理块。当前进程的页表如表所示。

页号	块号	存在位 P	访问位 R	修改位 M
0	0x1C	1	1	0
1	0x3F	1	1	1
2	—	0	0	0
3	0x5D	1	0	0
4	—	0	0	0

(2) 请分别计算进程中虚地址为0x3B7 、0x12A5 、0x1432 单元的物理地址（用十六进制表示），并说明理由。

(2) 0x12 A5对应的二进制数为00010010 1010 0101,页号为4,现在该页不在内存中,产生缺页,换出第三页。

考点二：存储器管理专题

3、一个进程的大小占5个页面，每页的大小为1 KB, 系统为它分配了3个物理块。当前进程的页表如表所示。

页号	块号	存在位 P	访问位 R	修改位 M
0	0x1C	1	1	0
1	0x3F	1	1	1
2	—	0	0	0
3	0x5D	1	0	0
4	—	0	0	0

(2) 请分别计算进程中虚地址为0x3B7 、0x12A5 、0x1432 单元的物理地址（用十六进制表示），并说明理由。

(2) 0x1432对应的二进制为000101 00 0011 0010,该逻辑地址的页号是5,显然地址越界。

考点二：存储器管理专题

5、某段式存储管理系统的段表如表所示。请将逻辑地址[0,137] 、[l,9000] 、[2 ,3600] 、[3,230]转换成物理地址。

段号	段大小/KB	段起址/KB
0	15	40
1	8	80
2	10	100

(1)对于逻辑地址[0,137],段号为0,小于进程的总段数,故段号合法。查段表可得该段的段地址为40 KB。段长为15 KB,段内位移为137,故而段内地址合法。因此可得到物理地址为:40 KB+137 B=40960 B+137 B=41097 B。

考点二：存储器管理专题

5、某段式存储管理系统的段表如表所示。请将逻辑地址[0,137] 、[l,9000] 、[2 ,3600] 、[3,230]转换成物理地址。

段号	段大小/KB	段起址/KB
0	15	40
1	8	80
2	10	100

(2）对于逻辑地址[1,9000],段号为1,小于进程的总段数,故段号合法。查段表的项得到该段的段地址为80 KB。该段段长为8 KB,段内位移9000大于段长8 KB=8192 B。因此,产生越界中断。

考点二：存储器管理专题

5、某段式存储管理系统的段表如表所示。请将逻辑地址[0,137] 、[l,9000] 、[2 ,3600] 、[3,230]转换成物理地址。

段号	段大小/KB	段起址/KB
0	15	40
1	8	80
2	10	100

(3）对于逻辑地址[2,3600],段号为2,小于进程的总段数,故段号合法,查段表的项得到该段的段地址为100 KB。段内位移为3600,段长为10 KB。段内位移3600小于段长10 KB，故段内地址合法。因此,可得到物理地址为:100 KB+3600 B=10240 B+3600 B=10600 B。

考点二：存储器管理专题

5、某段式存储管理系统的段表如表所示。请将逻辑地址[0,137] 、[l,9000] 、[2 ,3600] 、[3,230]转换成物理地址。

段号	段大小/KB	段起址/KB
0	15	40
1	8	80
2	10	100

(4）对于逻辑地址[3,230],段号为3,段内位移为230。由于段号3大于进程的总段数2 ， 故段号不合法,因此产生越界中断。

考点二：存储器管理专题

6、某系统采用分页存储管理方式，设计如下:页面大小为4KB，允许用户虚地址空间最大为16页，允许系统物理内存最多为512个内存块。试问该系统虚地址寄存器和物理地址寄存器的长度各是多少位?

【解析】页面大小L=4KB=2^12字节，即页内偏移量占12位。由于物理块大小等于页面大小所以物理块大小为2^12字节，物理块内地址位数占12位。

允许用户虚地址空间最大为16页=2^4页,即页号占4位。

允许系统物理内存最多为512个内存块=2^9个内存块，即内存块位数占9位。

虚地址寄存器位数=页号位数+页内偏移量位数=4+12=16位。

物理地址寄存器位数=物理块位数+内存块位数=12+9-21位。

考点二：存储器管理专题

页面置换算法

考点二：存储器管理专题

1、设某计算机的逻辑地址空间和物理地址空间均为64KB，按字节编址。若某进程最多需要 6 页(Page) 数据存储空间，页的大小为 1KB，操作系统采用固定分配局部置换策略为此进程分配 4 个页框(Page Frame)。 在时刻260前的该进程访问情况如下表所示（访问位即使用位）。

页号	页框号	装入时刻	访问位
0	7	130	1
1	4	230	1
2	2	200	1
3	9	160	1

当该进程执行到时刻 260 时，要访问逻辑地址为 17CAH 的数据。 请回答下列问题：

(1) 该逻辑地址对应的页号是多少?

考点二：存储器管理专题

【解析】(1) 首先根据题意可知，该计算机的逻辑地址空间和物理地址空间大小均为64KB，即2^16字节，而每个页的大小为1KB，即2^10字节，因此一个页可以存储2^10个字节的数据。由此，可将该进程最多需要的6页的数据存储空间看作一个逻辑地址空间大小为6KB，即2^10*6B的区域，而该进程要访问的逻辑地址为17CAH，即0001 0111 1100 1010B。因为页的大小为1KB，所以逻辑地址中低10位为页内偏移量，而高6位为页号，即000101B，即对应第5页。

考点二：存储器管理专题

1、设某计算机的逻辑地址空间和物理地址空间均为64KB，按字节编址。若某进程最多需要 6 页(Page) 数据存储空间，页的大小为 1KB，操作系统采用固定分配局部置换策略为此进程分配 4 个页框(Page Frame)。 在时刻260前的该进程访问情况如下表所示（访问位即使用位）。

页号	页框号	装入时刻	访问位
0	7	130	1
1	4	230	1
2	2	200	1
3	9	160	1

当该进程执行到时刻 260 时，要访问逻辑地址为 17CAH 的数据。 请回答下列问题：

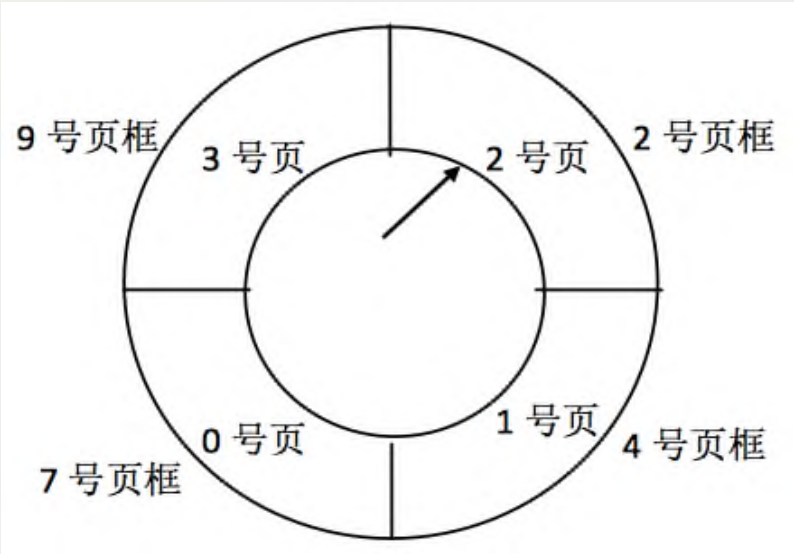
(2) 若采用先进先出（FIFO）置换算法，该逻辑地址对应的物理地址是多少?要求给出计算过程。

考点二：存储器管理专题

(2) 对于FIFO置换算法，需要将最先进入物理内存的页淘汰，因此需要找到最先进入物理内存的页的页框号，即页表中最小的时间戳所对应的页框号。根据题意，此时该进程已经被分配了4个页框，因此需要淘汰其中的一个来腾出空间存放新的页。根据时间戳情况可知，当前最先进入物理内存的页为第0页，时间戳为0，因此需要将第5页装入第7个页框中，此时该页的物理地址即为0001 1111 1100 1010B。

考点二：存储器管理专题

(3) 若采用时钟（CLOCK）置换算法，该逻辑地址对应的物理地址是多少？要求给出计算过程（设搜索下一页的指针沿顺时针方向移动，且当前指向2号页框，示意图如下）。



考点二：存储器管理专题

(3) 对于时钟算法，需要在物理内存中找到一个未被使用过的页框，若所有页框都被使用过，则需要替换一个被使用的页框。时钟算法中使用了一个指针，指向下一个被查找的页框，具体操作过程为：每当要查找一页框时，首先检查指针所指的页框是否被使用过，如果使用过则将其使用位清零，并将指针指向下一个页框；如果未被使用过，则将该页框替换掉并更新页表信息，然后将指针指向下一个页框。根据题意和时钟算法的操作过程，需要先从2号页框开始查找，发现其未被使用过，因此将第5页装入2号页框中并将其使用位设置为1，此时该页的物理地址即为0000 1011 1100 1010B。

考点二：存储器管理专题

2、某请求分页系统的页面置换策略如下：

从0时刻开始扫描，每隔5个时间单位扫描一轮驻留集（扫描时间忽略不计）且在本轮没有被访问过的页框将被系统回收，并放入到空闲页框链尾，其中内容在下一次分配之前不清空。当放发生缺页时，如果该页曾被使用过且还在空闲页链表中，则重新放回进程的驻留集中；否则，从空闲页框链表头部取出一个页框。

忽略其它进程的影响和系统开销。初始时进程驻留集为空。目前系统空闲页的页框号依次为 32、15、21、41。 进程 P 依次访问的<虚拟页号，访问时刻>为<1,1>、<3,2>、<0,4>、<0,6>、<1,11>、<0,13>、<2,14>。请回答下列问题。

考点二：存储器管理专题

【解析】依题意可列出驻留集及进程变化情况表，

虚拟页	<1,1>	<3,2>	<0,4>	5.扫描	<0,6>	10.扫描	<1,11>	<0,13>	<2,14>
进程驻留集	1->32	1->32	1->32	1->32	1->32	0->21	0->21	0->21	0->21
		3->15	3->15	3->15	3->15		**1->32**	1->32	1->32
			0->21	0->21	0->21				**2->41**
缺页	√	√	√						√
页框链表指针	32	15	21	41	41	41	41	41	41
空闲页框链表						1->32	3->15	3->15	3->15
						3->15			

考点二：存储器管理专题

2、某请求分页系统的页面置换策略如下：

从0时刻开始扫描，每隔5个时间单位扫描一轮驻留集（扫描时间忽略不计）且在本轮没有被访问过的页框将被系统回收，并放入到空闲页框链尾，其中内容在下一次分配之前不清空。当放发生缺页时，如果该页曾被使用过且还在空闲页链表中，则重新放回进程的驻留集中；否则，从空闲页框链表头部取出一个页框。

忽略其它进程的影响和系统开销。初始时进程驻留集为空。目前系统空闲页的页框号依次为 32、15、21、41。 进程 P 依次访问的<虚拟页号，访问时刻>为<1,1>、<3,2>、<0,4>、<0,6>、<1,11>、<0,13>、<2,14>。请回答下列问题。

（1）当虚拟页为<0,4>时，对应的页框号是什么?

因此根据上表，

（1）页框号为 21。因为起始驻留集为空，而 0 页对应的页框为空闲链表中的第三个空闲页框（21），其对应的页框号为 21。

考点二：存储器管理专题

2、某请求分页系统的页面置换策略如下：

从0时刻开始扫描，每隔5个时间单位扫描一轮驻留集（扫描时间忽略不计）且在本轮没有被访问过的页框将被系统回收，并放入到空闲页框链尾，其中内容在下一次分配之前不清空。当放发生缺页时，如果该页曾被使用过且还在空闲页链表中，则重新放回进程的驻留集中；否则，从空闲页框链表头部取出一个页框。

忽略其它进程的影响和系统开销。初始时进程驻留集为空。目前系统空闲页的页框号依次为 32、15、21、41。 进程 P 依次访问的<虚拟页号，访问时刻>为<1,1>、<3,2>、<0,4>、<0,6>、<1,11>、<0,13>、<2,14>。请回答下列问题。

（2）当虚拟页为<1,11>时，对应的页框号是什么？说明理由。

（2）页框号为 32。理由：因 11>10 故发生第三轮扫描，页号为 1 的页框在第二轮已处于空闲页框链表中，此刻该页又被重新访问，因此应被重新放回驻留集中，其页框号为 32。

考点二：存储器管理专题

2、某请求分页系统的页面置换策略如下：

从0时刻开始扫描，每隔5个时间单位扫描一轮驻留集（扫描时间忽略不计）且在本轮没有被访问过的页框将被系统回收，并放入到空闲页框链尾，其中内容在下一次分配之前不清空。当放发生缺页时，如果该页曾被使用过且还在空闲页链表中，则重新放回进程的驻留集中；否则，从空闲页框链表头部取出一个页框。

忽略其它进程的影响和系统开销。初始时进程驻留集为空。目前系统空闲页的页框号依次为 32、15、21、41。 进程 P 依次访问的<虚拟页号，访问时刻>为<1,1>、<3,2>、<0,4>、<0,6>、<1,11>、<0,13>、<2,14>。请回答下列问题。

（3）当虚拟页为<2,14>时，对应的页框号是什么？说明理由。

（3）页框号为 41。理由：因为第 2 页从来没有被访问过，它不在驻留集中，因此从空闲页框链表中取出链表头的页框 41，页框号为 41。

考点二：存储器管理专题

2、某请求分页系统的页面置换策略如下：

从0时刻开始扫描，每隔5个时间单位扫描一轮驻留集（扫描时间忽略不计）且在本轮没有被访问过的页框将被系统回收，并放入到空闲页框链尾，其中内容在下一次分配之前不清空。当放发生缺页时，如果该页曾被使用过且还在空闲页链表中，则重新放回进程的驻留集中；否则，从空闲页框链表头部取出一个页框。

忽略其它进程的影响和系统开销。初始时进程驻留集为空。目前系统空闲页的页框号依次为 32、15、21、41。 进程 P 依次访问的<虚拟页号，访问时刻>为<1,1>、<3,2>、<0,4>、<0,6>、<1,11>、<0,13>、<2,14>。请回答下列问题。

（4）这种方法是否适合于时间局部性好的程序？说明理由。

（4）合适。理由：如果程序的时间局部性越好，从空闲页框链表中重新取回的机会越大，该策略的优势越明显。

考点二：存储器管理专题

3、某操作系统采用分页式虚拟存储管理方法,现有一个进程需要访问的地址序列(字节)分别是:115,228,120,88,446,102,321,432,260,167。假设该进程的第0页已经装入内存，并分配给该进程300字节,页的大小为100字节,试回答以下问题:

(1)按LRU调度算法将产生多少次页面置换?依次淘汰的页号是什么?页面置换率为多少?

(1)页的大小是100字节,因此进程要访问的页面序列是1、2、1,0、4、1,3、4、2、1。分配给该进程300字节,故系统中共有3个页面。采用最近最久未使用(LRU)页面置换算法的页面走向如下表所示。

页面	1	2	1	0	4	1	3	4	2	1
内存情况	1	1	1	1	1	1	1	1	2	2
		2	2	2	4	4	4	4	4	4
				0	0	0	3	3	3	1
是否置换页					√		√		√	√

由上表可知,产生4次页面置换,依次淘汰的页号是2、0、1、3。页面置换率=4/10×100% = 40% 。

考点二：存储器管理专题

3、某操作系统采用分页式虚拟存储管理方法,现有一个进程需要访问的地址序列(字节)分别是:115,228,120,88,446,102,321,432,260,167。假设该进程的第0页已经装入内存，并分配给该进程300字节,页的大小为100字节,试回答以下问题:

(2)LRU页面置换算法的基本思想是什么?

(2)利用局部性原理,根据一个作业在执行过程中过去的页面访问历史来推测未来的行为。

考点二：存储器管理专题

4、(1)分页和分段属于离散型的存储管理方式,相对于连续内存管理方法的主要优点是什么?

(1)连续内存管理方法会形成许多碎片,虽然可通过紧凑方法将许多碎片拼接成可用的大块区域,但需要为此付出巨大的开销。离散型的存储管理方式允许将一个进程分散装入许多不相邻的分区中,这样便可以充分地利用内存空间,而不需要再进行紧凑拼接碎片。

考点二：存储器管理专题

4、 (2)某操作系统采用段式存储管理,假设有如下段表(注意:其中数字为十进制表示)。

段号	段的长度(字节)	主存起始地址
0	660	219
1	14	3300
2	100	90
3	580	1237
4	96	1952

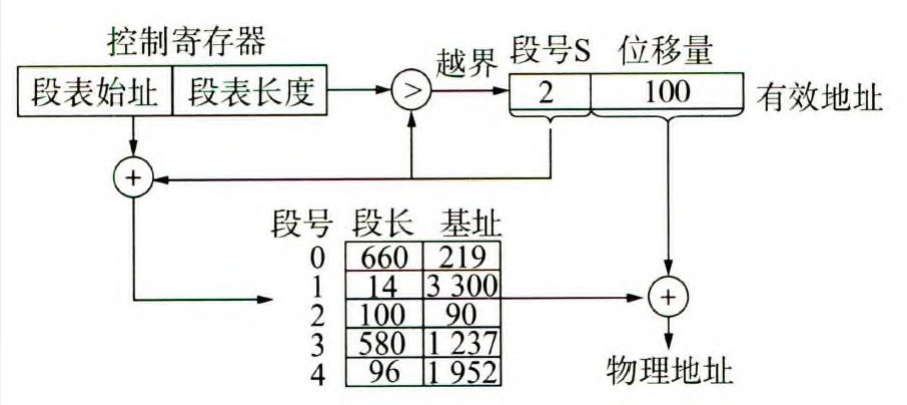
试解决下列问题:

(a)给定段号和段内地址,完成段式管理中的地址变换过程(并用图示)。

(a)地址变换=段的起始地址+段内地址,因此各段在内存中的地址计算如下:

第0段在内存中的地址为219+S。

第1段在内存中的地址为3 300+S。



考点二：存储器管理专题

4、 (2)某操作系统采用段式存储管理,假设有如下段表(注意:其中数字为十进制表示)。

段号	段的长度(字节)	主存起始地址
0	660	219
1	14	3300
2	100	90
3	580	1237
4	96	1952

试解决下列问题:

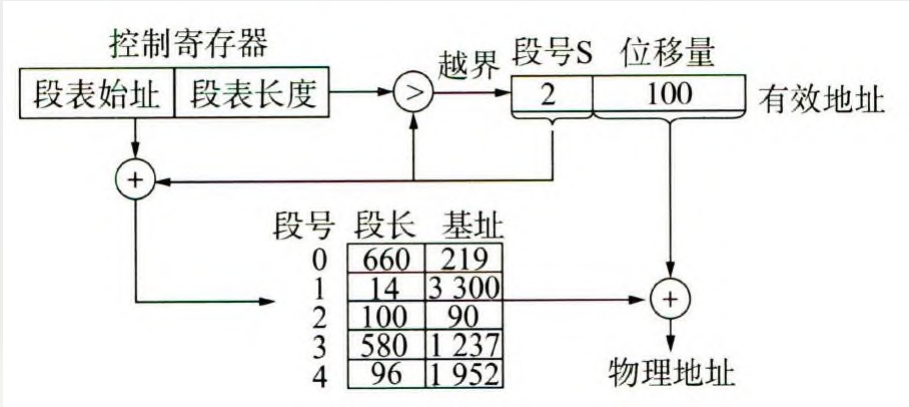
(a)给定段号和段内地址,完成段式管理中的地址变换过程(并用图示)。

(a)地址变换=段的起始地址+段内地址,因此各段在内存中的地址计算如下:

第2 段在内存中的地址为90+S。

第3段在内存中的地址为1237+S。

第4段在内存中的地址为1952+S。



考点二：存储器管理专题

4、 (2)某操作系统采用段式存储管理,假设有如下段表(注意:其中数字为十进制表示)。

段号	段的长度(字节)	主存起始地址
0	660	219
1	14	3300
2	100	90
3	580	1237
4	96	1952

试解决下列问题:

(b)计算[0,430],[1,10],[2,500],[3,400]的内存地址,其中方括号内的第一元素为段号,第二元素为段内地址。

(b)[0,430]的内存地址为219+430=649。

[1,10]的内存地址为3 300+10=3 310。

[2,500]的段内地址过长(500>100) ,访问越界。

[3,400]的内存地址为1237+400= 1 637。

考点二：存储器管理专题

4、 (2)某操作系统采用段式存储管理,假设有如下段表(注意:其中数字为十进制表示)。

段号	段的长度(字节)	主存起始地址
0	660	219
1	14	3300
2	100	90
3	580	1237
4	96	1952

试解决下列问题:

(c)存取内存中的一条指令或数据至少要访问几次内存?如何提高地址转换的效率?

(c)至少访问两次内存。第一次访问段表得到起始地址+段内地址,第二次访问主存物理地址实现存取。

可以用高速缓冲寄存器(快表)来减少对内存的访问次数,提高地址转换效率。

考点二：存储器管理专题

5、在一个请求分页系统中,采用LRU页面置换算法时,假如一个作业的页面走向为4,3,2,1,4,3,5,4,3,2,1,5,开始时页面都不在内存中,当分配给该作业的物理内存块数M分别为3和4时。

(1)页面置换过程,分别计算在访问过程中所发生的缺页次数和缺页率。

(1)物理内存块数为3时,页面置换过程如下表所示。

页面	4	3	2	1	4	3	5	4	3	2	1	5
内存情况	4	4	4	1	1	1	5	5	5	2	2	2
		3	3	3	4	4	4	4	4	4	1	1
			2	2	2	3	3	3	3	3	3	5
是否置换页	√	√	√	√	√	√	√			√	√	√

由上表可知,共缺页10次,缺页率=10/12×100%=83.3% 。

考点二：存储器管理专题

5、在一个请求分页系统中,采用LRU页面置换算法时,假如一个作业的页面走向为4,3,2,1,4,3,5,4,3,2,1,5,开始时页面都不在内存中,当分配给该作业的物理内存块数M分别为3和4时。

(1)页面置换过程,分别计算在访问过程中所发生的缺页次数和缺页率。

(1)物理内存块数为4时,页面置换过程如下表所示。

页面	4	3	2	1	4	3	5	4	3	2	1	5
内存情况	4	4	4	4	4	4	4	4	4	4	4	5
		3	3	3	3	3	3	3	3	3	3	3
			2	2	2	2	5	5	5	5	1	1
				1	1	1	1	1	1	2	2	2
是否置换页	√	√	√	√			√			√	√	√

由上表可知,共缺页8次,缺页率=8/12×100%=66.7% 。

考点二：存储器管理专题

5、在一个请求分页系统中,采用LRU页面置换算法时,假如一个作业的页面走向为4,3,2,1,4,3,5,4,3,2,1,5,开始时页面都不在内存中,当分配给该作业的物理内存块数M分别为3和4时。

(2)根据两种情况下的页面缺页率,能够得到什么结论?

(2)采用LRU页面置换算法时,物理内存块数增加,缺页率降低。

考点二：存储器管理专题

6、请求分页系统中,设某进程共有9个页,分配给该进程的主存块数为5(即工作集为5),进程运行时,实际访问页面的次序是0,1,2,3,4,5,0,2,1,8,5,2,7,6,0,1,2。试求:

(1) FIFO页面调度算法,列出其页面淘汰次序和缺页中断次数,以及最后留驻主存的页号顺序。

(1) 采用FIFO 页面调度算法，页面置换情况如表所示。

访问序列	0	1	2	3	4	5	0	2	1	8	5	2	7	6	0	1	2
内存块 1	0	0	0	0	0	5	5	5	5	5	5	5	7	7	7	7	7
内存块 2		1	1	1	1	1	0	0	0	0	0	0	0	6	6	6	6
内存块 3			2	2	2	2	2	2	1	1	1	1	1	1	0	0	0
内存块 4				3	3	3	3	3	3	8	8	8	8	8	8	1	1
内存块 5					4	4	4	4	4	4	4	2	2	2	2	2	2
淘汰的页						0	1		2	3		4	5	0	1	8	
缺页否?	√	√	√	√	√	√	√		√	√		√	√		√	√	

页面淘汰顺序为0 、1、2 、3 、4 、5 、0 、1 、8，缺页中断次数为14 次。最后留驻主存的页号顺序为7 、6 、0 、1、2 。

考点二：存储器管理专题

6、请求分页系统中,设某进程共有9个页,分配给该进程的主存块数为5(即工作集为5),进程运行时,实际访问页面的次序是0,1,2,3,4,5,0,2,1,8,5,2,7,6,0,1,2。试求:

(2) LRU页面调度算法,列出其页面淘汰次序和缺页中断次数,以及最后留驻主存的页号顺序。

(2) 采用LRU 页面调度算法，页面置换情况如表所示。

访问序列	0	1	2	3	4	5	0	2	1	8	5	2	7	6	0	1	2
内存块 1	0	0	0	0	0	5	5	5	5	5	5	5	5	5	5	1	1
内存块 2		1	1	1	1	1	0	0	0	0	0	0	7	7	7	7	7
内存块 3			2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
内存块 4				3	3	3	3	3	1	1	1	1	1	6	6	6	6
内存块 5					4	4	4	4	4	8	8	8	8	8	0	0	0
淘汰的页						0	1		3	4			0	1	8	5	
缺页否?	√	√	√	√	√		√		√	√			√	√	√	√	

由表可知，页面淘汰顺序为0 、1、3 、4 、0 、1、8 、5，缺页中断次数为13 次。最后留驻主存的页号顺序为1 、7 、2 、6 、0 。

考点二：存储器管理专题

6、请求分页系统中,设某进程共有9个页,分配给该进程的主存块数为5(即工作集为5),进程运行时,实际访问页面的次序是0,1,2,3,4,5,0,2,1,8,5,2,7,6,0,1,2。试求:

(3) CLOCK 页面调度算法,列出其页面淘汰次序和缺页中断次数,以及最后留驻主存的页号顺序。

(3) 采用CLOCK 页面调度算法，（蓝色表示指针位置，* 号表示访问标志为1）。

访问序列	0	1	2	3	4	5	0	2	1	8	5	2	7	6	0	1	2
内存块 1	0 *	0 *	0 *	0 *	0 *	5 *	5 *	5 *	5 *	5 *	5 *	5 *	7 *	7 *	7 *	7 *	7 *
内存块 2		1 *	1 *	1 *	1 *	1	0 *	0 *	0 *	0 *	0 *	0 *	0	6 *	6 *	6 *	6 *
内存块 3			2 *	2 *	2 *	2	2	2 *	2	2	2	2 *	2	2	0 *	0 *	0 *
内存块 4				3 *	3 *	3	3	3	1 *	1 *	1 *	1 *	1	6	6	1 *	1 *
内存块 5					4 *	4	4	4	4	8 *	8 *	8 *	8	8	8	8	2 *
淘汰的页						0	1		3	4			5	0	2	6	8 *
缺页否?	√	√	√	√	√	√	√		√	√			√	√	√	√	√

由表可知，页面淘汰顺序为0 、1 、3 、4 、5、0 、2 、6 、8，缺页中断次数为14 次。最后留驻主存的页号顺序为7 、6 、0 、1、2 。

考点二：存储器管理专题

6、请求分页系统中,设某进程共有9个页,分配给该进程的主存块数为5(即工作集为5),进程运行时,实际访问页面的次序是0,1,2,3,4,5,0,2,1,8,5,2,7,6,0,1,2。试求:

(4) OPT页面调度算法,列出其页面淘汰次序和缺页中断次数，以及最后留驻主存的页号顺序。

(4) 采用OPT 页面调度算法，页面置换情况如表所示。

访问序列	0	1	2	3	4	5	0	2	1	8	5	2	7	6	0	1	2
内存块 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
内存块 2		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
内存块 3			2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
内存块 4				3	3	3	3	3	3	8	8	8	7	7	7	7	7
内存块 5					4	5	5	5	5	5	5	5	5	6	6	6	6
淘汰的页						4				3			8	5			
缺页否?	√	√	√	√	√	√				√			√	√			

由表可知，页面淘汰顺序为4 、3 、8 、5(或4 、3 、5 、8) 缺页中断次数为9 次。最后留驻主存的页号顺序为0 、1、2 、7 、6(或0 、1、2 、6 、7) 。

考点二：存储器管理专题

7、在某个请求分页系统中，一个进程已分配到4个页框(Page Frame)，如表所示（所有数字都为十进制数，且以0开始）。操作系统采用固定分配局部置换策略为此进程分配4个页框。当进程访问第4页时，产生缺页中断，请分别用FIFO、LRU 算法，决定缺页中断服务程序选择换出的页面。

页号	页框号	装入时间	最近访问时间	访问位	修改位
2	0	60	161	0	1
1	1	130	160	0	0
0	2	26	162	1	0
3	3	20	163	1	1

由于采用固定分配局部置换策略，所以该进程只能占用4 个页框。从装入时间和最近访问时间排序可看出页面走向是： 3 、0 、2 、1 、1 、2 、0 、3 、4 。采用FIFO 置换算法时的页面置换情况如表所示，从中看到最后访问页面4 时被置换的是页面3, 由于该页框的修改位为1, 在换出主存后需要回写。

页面走向	3	0	2	1	1	2	0	3	4
物理块 0			2	2					2
物理块 1				1					1
物理块 2		0	0	0					0
物理块 3	3	3	3	3					4
缺页否?	√	√	√	√					√

采用LRU 置换算法时的页面置换情况如表所示，从中看到最后访问页面4 时被置换的是页面1, 由于该页框的修改位为0, 在换出主存后不需要回写（即重新保存）。

页面走向	3	0	2	1	1	2	0	3	4
物理块 0			2	2					2
物理块 1				1					4
物理块 2		0	0	0					0
物理块 3	3	3	3	3					3
缺页否?	√	√	√	√					√