系统运行有 3 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 3 个进程的工作情况，并用 wait 和 signal 原语实现其同步操作。

2.过桥问题

请用信号量解决以下的“过独木桥”问题：同一方向的行人可连续过桥，当某一方向有人过桥时，另一方向的行人必须等待；当某一方向无人过桥时，另一方向的行人可以过桥。

3.生产者-消费者问题

用一个数组表示具有 *n* 个缓冲区的缓冲池；用输入指针 in，指示下一个可投放消息的缓冲区；用输出指针 out，指示下一个可获取消息的缓冲区，利用记录型信号量解决资源使用问题。

4.打水问题

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 30 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 5 个，试用信号量和 PV 操作给出老和尚和小和尚的活动。

5.缓冲区问题

设有一缓冲池 P，P 中含有 20 个可用缓冲区，一个输入进程将外部数据读入 P，另有一个输出进程将 P 中数据取出并输出。若进程每次操作均以一个缓冲区为单位，试用记录型信号量写出两个进程的同步算法，要求写出信号量的初值。

6.阅览室问题

假定一个阅览室最多可容纳 100 人，读者进入和离开阅览室时都必须在阅览室门口的一个登记表上进行登记，而且每次只允许一人进行登记操作，请用记录型信号量机制实现上述问题的同步。

7.吃水果问题

桌子上有一只盘子，盘子只能放入一只水果。爸爸专向盘中放苹果，妈妈专向盘中放桔子，儿子专等吃盘中的桔子，女儿专等吃盘中的苹果。试用 P、V 操作完成上述 4 个进程。

8.3 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 3 个进程的同步与互斥活动，并说明所定义的信号量的含义。要求用伪代码描述。

9.某银行提供 1 个服务窗口和 10 个顾客等待座位。顾客到达银行时，若有空座位，则到取号机领取一个号，等待叫号。取号机每次仅允许一个顾客使用。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
      {
          从取号机获得一个号码;
          等待叫号;
          获得服务;
      }

    process 营业员
      {
        while（true）
          {
            叫号;
            为顾客服务;
          }
      }
  }
coend
```

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

10.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，以实现上述操作过程中的互斥与同步。要求写出完整的过程，说明信号量含义并赋初值。

11.读者一写者问题（Readers—Writers Problem）

问题描述：有一个许多进程共享的数据区，这个数据区的一块空间；有一些只读取这个数据区的进程（Reader）和一程（Writer），此外还需要满足以下条件：

①任意多个读进程可以同时读这个文件；

②一次只有一个写进程可以往文件中写；

③如果一个写进程正在进行操作，禁止任何读进程文件。

用信号量来实现读者写者问题的以下调度算法。

（1）读者优先：一个读者试图进行读操作时，如果这时正有其他读者在进行操作，它可以直接开始读操作，而不需要等待。

（2）写优先：一个读者试图进行读操作时，如果有其他写者在等待或进行写操作，他要等待这些写者完成写操作后才开始读操作。

12.理发师问题（Barber Problem）

问题描述：理发店里有一位理发师、一把理发椅和 n 把供等候理发的顾客坐的椅子。如果没有顾客，理发师便在理发椅上睡觉。一个顾客到来时，它必须叫醒理发师。如果理发师正在理发时又有顾客来到，则如果有空椅子可坐，就坐下来等待，否则就离开。

13.吸烟者问题（Smoker Problem）

3个吸烟者在一间房间内，还有一个香烟供应者。为了制造并抽掉香烟，每个吸烟者需要3样东西：烟草、纸和火柴。供应者有丰富的货物提供。3个吸烟者中，第1个有自己的烟草，第2个有自己的纸，第3个有自己的火柴。供应者将两样东西放在桌子上，允许一个吸烟者进行对健康不利的吸烟。当吸烟者完成吸烟后唤醒供应者，供应者再放两样东西（随机地）在桌面上，然后唤醒另一个吸烟者。试为吸烟者和供应者编写程序解决问题。

14.有4个并发进程：R1、R2、W1和W2，它们共享可以存放一个数的缓冲区。进程R1每次从磁盘读入一个数存放到缓冲区中，供进程W1打印输出；进程R2每次从键盘读一个数存放到缓冲区中，供进程W2打印输出。当缓冲区满时，不允许再向缓冲区中存放数据；当缓冲区空时，不允许再从缓冲区中取出数据打印输出。试用PV操作实现4个进程的协调运行。

15.设公共汽车上，司机的活动顺序是：启动车辆、正常行车、到站停车；售票员的活动顺序是：关车门、售票、开车门。现假设初始状态为：司机和售票员都已经在车上，汽车处于停止状态，车门处于开的状态。在汽车不断地到站、停车、行驶过程中，请用信号量的PV操作实现司机与售票员之间的同步关系。

16.哲学家进餐问题

5位哲学家吃面条，只有五根筷子，每个人必须用一双筷子才能吃面条。请用信号量的PV操作描述哲学家之间的关系。

17.有一个阅览室，共有100个座位。读者进入阅览室时必须在入口处进行登记；离开阅览室时必须进行注销。试用PV操作描述读者进入/离开阅览室的同步与互斥关系。

参考答案

一、选择题部分

1.【解析】解 PV 操作算法题目时，我们总结出了一套方便掌握的“五步曲”，本部分的每一个题目，我们都可用这套方法来解答。

第一步：找进程

①输入进程；②计算进程；③打印进程。

第二步：找动作

（1）输入进程

①从外部接收数据；②把数据放入 buffer1 中。

（2）计算进程

①从 buffer1 中取出数据；②计算；③把结果放入 buffer2 中。

（3）输出进程

①从 buffer2 中取出数据；②打印输出。

第三步：找关系

①输入进程与计算进程访问 buffer1，是互斥关系。

②输入进程把数据放入 buffer1，计算进程才能取数据进行计算，是同步关系。

③打印进程把计算后的数据放入 buffer2，输出进程才能取走数据输出打印，是同步关系。

④输出进程和打印进程访问 buffer2，是互斥关系。

第四步：找初值

①输入进程与计算进程互斥使用 buffer1，设置互斥信号量 mutex1，初值为 1。

②计算进程与输出进程互斥使用 buffer2，设置互斥信号量 mutex2，初值为 1。

③为同步输入进程和计算进程使用 buffer1，设置信号量 empty1 表示 buffer1 为空，初始值为 1。设置 full1，表示 buffer1 为满，初始值为 0。输入进程从外部接收数据之后，把数据放入空的缓冲区，full1=1，等待计算进程读取。同时，计算进程把 buffer1 中的数据取去计算，buffer1 为空，empty=1，输入进程可以继续从外部接收数据放入 buffer1。

④为同步计算进程和输出进程使用 buffer2，设置信号量 empty2 表示 buffer2 为空，初始值为 1。设置 full2，表示 buffer2 为满，初始值为 0。计算进程计算数据完毕之后，把数据放入空的缓冲区 buffer2，full2=1，等待输出进程读取输出打印。同时，输出进程把 buffer2 中的数据取去计算，buffer2 为空，empty2=1，计算进程可以将新的计算后的数据放入 buffer2。

第五步：写 PV 算法。

输入进程、计算进程和打印进程之间的同步问题描述如下。

```
var：mutex1，rautex2，empty1，empty2，full1，full2：=1，1，1，1，0，0；
Input_p：
  begin
     repeat
          wait（empty1）；                    //等待缓冲区 buffer1 空
          wait（mutex1）；                    //互斥访问 buffer1
          从外部接收数据；
          把数据放入 buffer1 中；
          signal（mutex1）；                  //释放互斥信号量
             signal（full1）；               //缓冲区 buffer1 满
     until false
  end
Cal_P：
  begin
       repeat
            wait（full1）；                   //等待缓冲区 buffer1 为满
            wait（mutex1）；                  //与输入进程互斥访问 buffer1
            从 bufferl 取出数据；
            signal（mutex1）；                //释放互斥信号量
            signal（empty1）；                //从 buffer1 取走数据，置 empty1=1
```

```
            用取到的数据进行计算；
            wait（empty2）；              //等待缓冲区 buffed 为空，放入数据
            wait（mutex2）；              //与输出进程互斥访问 buffer2
            把计算后的数据放入 buffer2；
            signal（mutex2）；          //释放互斥信号量
            signal（full2）；             //计算进程向 buffer2 写入数据，full2=1
        until false
    end
  Output_p：
    begin
        repeat
            wait（full2）；             //等待缓冲区 2 为满
            wait（mutex2）；              //与计算进程互斥访问 bnffer2
            从 buffer2 取出数据；
            signal（mutex2）；          //释放互斥信号
            signal（erapty2）；         //数据被输出进程输出打印，empty2=1
            打印输出；
        until false
    end
```

2.【解析】按照 PV 算法“五步曲”，将独木桥的两端分别标记为 A 和 B。可解本题如下。

第一步：找进程。

①方向 A 到 B 的行人；②方向 B 到 A 的行人。

第二步：找动作。

（1）方向 A 到 B 的行人的动作过独木桥。（2）方向 B 到 A 的行人的动作过独木桥。

第三步：找关系。

不同方向上的行人要互斥过桥。

第四步：找初值。

①用整形变量 countA 和 countB 分别表示 A、B 端上等待过独木桥的行人数，初值为 0；

②设置信号量 SA 用来实现对 cmmtA 的互斥访问，初始值为 1；

③设置信号量 SB 用来实现对 countB 的互斥访问，初始值为 1；

④设置信号量 mutex 用来实现两个方向的行人对独木桥的互斥使用，初始值为 1。

第五步：写 PV 算法，具体描述如下。

```
Var SA，SB，mutex：semaphore：=1，1，1；
CountA，countB：integer：=0，0：
begin
    parbegin
    processA_B：
      begin
        wait（SA）；                              //要过桥，等待修改 countA 的信号量
        if（countA=0）then wait（mutex）；//若 A 到 B 无人等待过桥，需等待 mutex 信号
        countA：=countA+1；                  //等待过桥的人数量加 1
        signal（SA）；                          //释放互斥信号量
        过独木桥；
        wait（SA）；                              //过了桥，等待修改 countA 的互斥信号量
        countA：=countA－1；              //等待过桥的人数减 1
        if(countA=0)then signal(mutex); //如果 A 到 B 已无人要过桥，释放 mutex 信号
        signal（SA）；                          //释放修改 countA 信号
      end
  processB_A：
    begin
      wait（SB）；                              //要过桥，等待修改 countB 的信号量
      if（countB=0）then wait（mutex）；//若 B 到 A 无人等待过桥，需等待 mutex 信号
```

```
        countB：=countB+1；                    //等待过桥的人数量加 1
        signal（SB）；                         //释放互斥信号量
        过独木桥；                             //
        wait（SB）；                           //过了桥，等待修改 countB 的互斥信号量
       countB：=countB-1；                     //该方向上过桥人数减 1
       if（countB=0）then signal（mutex）；  //如果 B 到 A 已无人要过桥，释放 mutex 信号
       signal（SB）；                          //释放修改 countB 信号
    end
  parend
end
```

3.【解析】本题可用 PV 操作“五步曲”来解答，解题过程如下。

第一步：找进程

①生产者；②消费者。

第二步：找操作

（1）生产者

①生产一个数据；②将数据放入 in 指针指向的缓冲区。

（2）消费者

①从 out 指针指向的缓冲区读取数据；②消费数据。

第三步：找关系

①只有缓冲区有空闲，生产者才能将生产的数据放入（空闲）缓冲区中；

②只有缓冲区有数据，消费者才能取数据（来消费）。

第四步：找初值

①n 个缓冲区，可用信号量 empty 表示。设初始状态下缓冲区全部为空，即为 n。此时生产者可以有 n 个空闲的缓冲区用来存放数据，消费者初始状态下没有直接消费的数据，需等待。

②设 full 表示当前已经被生产者生产和放入数据的缓冲区数量，初始值为 0。

③缓冲区必须互斥访问，所以，设置互斥信号量 mutex，初值为 1。

④用一个数组 buffer，用来表示缓冲区。

⑤两个指针 in 和 out，in 指针指向生产者可以直接存放生产数据的缓冲区，out 指针指向消费者可以直接消费生产者生产好的数据的缓冲区。

【注意】buffer 数组是用来表示缓冲区的，而 empty 和 full 是用来表示缓冲区的空闲和使用情况的，这二者不一样，请读者仔细体会。

第五步：写算法

PV 算法如下。

```
Varmutex，empty，full：semaphore：=1，n，0；//三个信号量
buffer：array[0，…，n-1]of item；//n 个缓冲区
in，out：integer：=0，0；//生产者和消费者指针
begin
  parbegin
    生产者进程：
      begin
        repeat
          …
          produce an item nextp；      //生产一个产品（数据）放入 nextp
          …
          wait（empty）；              //看看有没有空闲的缓冲区
          wait（mutex）；              //有空闲的缓冲区
          buffer（in）；=nextp；       //把 nextp 中的产品送往 buffer（in）；
          in：=（in+1）modn；          //in 指针指向下一个位置
          signal（mutex）；            //释放互斥信号
          signal（full）；             //满缓冲区加 1
        untilfalse；
```

```
        end
      消费者进程：
        begin
          repeat
            wait（full）；                    //消费者看看有没有满的缓冲区
            wait（mutex）；                   //有满的缓冲区
            nextc：=buffer（out）；           //从 buffer（out）中取出产品放入 nextc
            out：=（out+1）modn；             //out 指针指向下一个位置
            signal（mutex）；                 //释放缓冲区互斥访问信号
            signal（empty）；                 //空闲缓冲区的数量加 1
            consume the item in nextc；//消费 nextc 中的产品
          untilfalse；
        end
    parend
end
```

4.【解析】根据解 PV 算法“五步曲”，可解本题如下。

第一步：找进程

①老和尚（若干）；②小和尚（若干）。

第二步：找动作

（1）老和尚（若干，操作都一样）

①取水桶；②去水缸边排队；③从水缸打水；④喝水；⑤放下水桶。

（2）小和尚（若干，操作都一样）

①取水桶；②去井边排队；③从井里打水；④将水抬到水缸边，排队将水放入水缸；⑤将水倒人水缸；⑥放下水桶。

【注意】并不是所有操作都要写出来，有些操作隐藏在 PV 操作里面。比如小和尚操作序列里面的取水桶可以用 P（buckets），而放下水桶用 V（buckets）。

第三步：找关系

①井一次只能让一个小和尚打水，此时小和尚打水是互斥关系。

②水缸一次只能有一个和尚取水（老和尚）或者倒水（小和尚），是互斥关系。

③小和尚往水缸倒水，水缸有水，老和尚才能取水喝，是同步关系。

④老和尚去水缸取水，水缸不满，小和尚才能去水井打水，是同步关系。

第四步：找初值

①设缸中水的信号 empty，用于表示缸还能装得水的桶数。初始状态下，缸中无水，empty=30。

②设置水缸中水的桶数的信号量为 full，初始状态下，水缸为空，full=0。

③设水桶的个数的信号量为 buckets，五个水桶，信号量初值为 5。

④互斥信号量 mutex_well 用于实现小和尚从井里打水的互斥。

⑤互斥信号量 mutex_bigjar 用于实现和尚之间互斥往水缸倒水或取水。

第五步：写算法

算法如下。

```
semaphore empty=30;          //表示缸中目前还能装多少桶水，初始时能装 30 桶水
semaphore full=0;            //表示缸中有多少桶水，初始时缸中没有水
semaphore buckets=5;         //表示有多少只空桶可用，初始时有 5 只桶可用
semaphore mutex_well=1;      //用于实现对井的互斥操作
semaphore mutex_bigjar=1;    //用于实现对缸的互斥操作
young_monk（）
  {
    while（true）
      {
        P（empty）；             //等待缸中水不满信号（缸中水的桶数小于 30 即为不满）
        P（buckets）；           //缸中还可以装入水，再去看看有没有桶
```

```
            去井边；
            P（mutex_well）；          //去井边互斥打水
            从井里打水；
            V（mutex_well）；          //打到水了，释放互斥信号，让待打水者可打水
            将水提到水缸边；
            P（mutex_bigjar）；        //等待互斥信号量，把水倒入缸中
            将水放入水缸；
            V（mutex_bigjar）；        //释放水缸互斥信号量
            V（buckets）；             //打完水，放下水桶，水桶可利用的数量加 1
            V（full）；                //水缸里面多了一桶水
        }
    }
old_monk（）
  {
    while（true）
      {
        P（full）；                    //看看水缸里面有没有水可以打
        P（buckets）；                 //水缸里面有水，于是再去看看有没有桶
        P（mutex_bigjar）；            //还有桶，于是等待水缸互斥访问信号量
        get water;                  //打水
        V（mutex_bigjar）；            //释放互斥信号量，其他老和尚可打水，小和尚可倒水
        drink water;                //喝水
        V（buckets）；                 //喝完水之后，放下水桶，可用的水桶数量加 1
        V（empty）；                   //缸中的水少了一桶，可装水的桶量多了一桶
      }
  }
```

5.【解析】本题类似于第一题，但仅涉及两个进程一个缓冲池，比第一题简单很多。我们简单地解析一下。

图 2.7 是生产者消费者问题的缓冲池情况，其中 n=20。利用解 PV 操作的“五步曲”，可解本题如下。

第一步：找进程

①输入进程；②输出进程。

第二步：找操作

（1）输入进程

①读取外部数据；②将数据放入缓冲池 P。

（2）输出进程

①从缓冲池 P 读出数据；②将数据输出。

第三步：找关系

①输入进程和输出进程需互斥访问缓冲池。

②输出进程取缓冲池数据输出，输入进程等到有空闲的缓冲区，才能读取数据放入缓冲区中，是同步关系。

③输入进程从外部读入数据放入缓冲池，输出进程等待缓冲池中有数据，才取数据输出，同步关系。

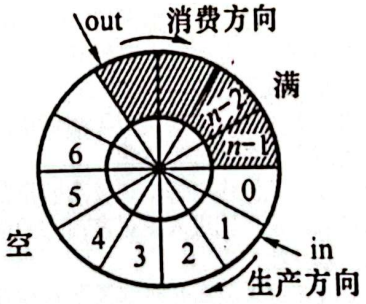

图 2.7

第四步：找初值

①输入进程和输出进程互斥访问缓冲池，设 mutex 互斥信号，初值为 1。

②初始状态下，设所有缓冲区为空，用信号量 empty 表示空闲缓冲区的数量，empty 的初值为 20。

③初始状态下，设满的缓冲区的数量 full 信号为 0。

④用 in、out 数组指针来分别指示当前输入进程和输出进程指向缓冲区的位置。

⑤用数组 P[20]表示缓冲区的数量，是定值，为 20。

第五步：写算法

PV 算法如下。

```
semaphore mutex=1;                    //访问缓冲池的互斥信号量
semaphore empty=20;                   //空闲缓冲区的信号量，初始状态下所有缓冲区为空
semaphore full=0;                     //满缓冲区的信号量，初始状态下没有满的缓冲区
int in，out=0;                        //输入进程和输出进程的数组指针
itemp[20];                            //缓冲区
void Producer（）
  {
    while（ture）
      {
          从外部接收数据放入 nextp 中；
        wait（empty）；               //看看有没有空闲缓冲区
        wait（mutex）；               //有空闲缓冲区
        p[in]：=nextp；               //把 nextp 中的数据放入 in 指针指向的缓冲区中
        in：=（in+1）mod20；          //in 指针指向数组的下一个位置（循环缓冲区）
        signal（mutex）；             //释放互斥信号量
        signal（full）；              //满的缓冲区数量加 1
      }
  }
void Consumer（）
  {
    while（ture）
      {
        wait（full）；                //看看有没有满的缓冲区
        wait（mutex）；               //有满的缓冲区，互斥访问
        nextc：=p[out]；              //从 out 指针指向的缓冲区位置读数据入 nextc 中
        out：=（out+1）mod20；        //out 指针指向数组下一个位置
        signal（mutex）；             //释放访问缓冲区的互斥信号量
        signal（empty）；             //空闲缓冲区数量加 1
      }
  }
```

6.【解析】阅览室问题，是我们常遇到的问题，其实阅览室问题、博物馆参观问题、银行问题、售票厅问题等，这些问题都很相似，就是容量一定，顾客在容纳得下的情况下才能进入。利用解 PV 算法的“五步曲”，可解析本题如下。

第一步：找进程

读者（若干）（其实类似于阅览室问题的一系列问题如博物馆问题、售票厅问题等，通常都只步及一类或者简单的几类进程）

第二步：找操作（每个读者的操作都相似）

①进入登记；②进入阅览室；③阅读；④离开登记；⑤离开阅览室。

第三步：找关系

①读者在进入和离开时都需要互斥登记。

②读者在阅览室人数不满的时候，才能进入阅览室。

③读者离开了以后，给阅览室腾出空间，等待进入阅览室的读者才能进入。

第四步：找初值

①设读者进入和离开登记的互斥信号量 mutex，初始值为 1。

②阅览室最多只能容得下 100 人，设置信号量 empty 表示当前阅览室还能进来的人数。设初始态下阅览室没有人，empty=100。

第五步：写算法

定义信号量 sum 和 mutex，初值分别为 100 和 1，则第 i 个读者的活动描述如下。

```
procedure Pi（i=1，2，3……）
    begin
        wait（empty）；               //进入阅览室之前，先看看阅览室是不是人满了
        wait（mutex）；               //人没有满，访问信号量 mutex，互斥登记
```

```
        进入登记;
        signal（mutex）;        //释放互斥信号量，让待入者可进，待出者可出
        进入阅览室;
      阅读;
        wait（mutex）;          //要离开，互斥登记
        离开登记;
        signal（mutex）;        //释放互斥登记的信号量
        离开阅览室;
        signal（empty）;        //读者离开后，阅览室可进入的人数加 1
end
```

7.【解析】根据解 PV 算法“五步曲”，解析本题如下。

第一步：找进程

①爸爸；②妈妈；③儿子；④女儿。

第二步：找动作

（1）爸爸：往盘中放入苹果。

（2）妈妈：往盘中放入桔子。

（3）儿子：①从盘中取出桔子；②吃桔子。

（4）女儿：①从盘中取出苹果；②吃苹果。

第三步：找关系

①盘子是空的，爸爸和妈妈可以互斥地向盘中放水果，是互斥关系。

②妈妈往盘子放入桔子，儿子可取来吃，是同步关系。

③爸爸往盘子放入苹果，女儿可取来吃，是同步关系。

第四步：找初值

①爸爸和妈妈互斥向盘中放水果。设互斥信号量 $S_$，初值为 1，表示盘为空。盘子只有一个，P（S）成立的人可以抢到盘子。所以，无需再建立互斥访问盘子的信号 mutex。

②用同步信号 S_0 来同步妈妈和儿子的操作。设 S_0 初值为 0，表示妈妈尚未将桔子放入盘中。

③用同步信号 S_1 来同步爸爸和女儿的操作。设 S_1 初值为 0，表示爸爸尚未将苹果放入盘中。

第五步：写算法

算法描述如下。

```
S，So，Sa：semaphore：=1，0，0;
process Father（）
  begin
        repeat
              P（S）;              //等待盘子为空的信号
              将苹果放入盘中;
              V（S1）;             //V1 信号表示盘子里面放的是苹果
        until false;
    end;
process Mother（）
  begin
        repeat
              P（S）;              //等待盘子为空的信号
              将桔子放入盘中;
              V（S0）;             //V2 表示盘子里面放的是桔子
        until false;
    end;
process Son（）
  begin
        repeat
              P（S0）;             //等待妈妈往盘子里面放桔子
              从盘中取出桔子;
              V（S）;              //取走了桔子，盘子空了
```

```
            吃桔子;
        until false;
    end;
process Daughter ()
  begin
      repeat
          P (S1) ;          //等待爸爸往盘子里放苹果
          从盘中取出苹果;
          V (S) ;           //取走了苹果，盘子空了
          吃苹果;
      until false;
    end;
```

8.【解析】本题是生产者—消费者问题的延伸。本题用解 PV 操作“五步曲”来求解，过程如下。

第一步：找进程（题目已告知有 3 个进程）

①进程 P1；②进程 P2；③进程 P3。

第二步：找动作

（1）P1 进程

①produce ()　　（注释：用 produce () 生产一个正整数）

②put ()　　（注释：用 put () 把数据放入缓冲区的某一个单元中）

（2）P2 进程

①getodd ()　　（注释：用 getodd () 从缓冲区取一个奇数）

②countodd ()　　（注释：用 cmmtodd () 来统计奇数个数）

（3）P3 进程

①geteven ()　　（注释：用 getodd () 从缓冲区取一个偶数）

②counteven ()　　（注释：用 counteven () 统计偶数的个数）

第三步：找关系

①进程 P1、P2、P3 必须互斥访问缓冲区。

②P1 生产了奇数，放入缓冲区，P2 才能取出奇数和统计。

③P1 生产了偶数，放入缓冲区，P3 才能取出偶数和统计。

第四步：找初值

①定义信号量 mutex，来控制 P1、P2、P3 互斥访问缓冲区。

②定义信号量 odd 来表示生产的数据为奇数，控制 P1 和 P2 的同步。

③定义信号量 even 来表示生产的数据为偶数，控制 P1 和 P3 的同步。

④定义信号量 empty 表示空闲缓冲区的数量，来控制 P1 与 P2、P3 的同步。

第五步：写 PV 算法

PV 算法实现如下。

```
semaphore empty=N，even=0，odd=0，mutex=1；//初始化信号量
Process P1：
  while (1)
    {
        number=produce () ;        //利用 produce () 生产一个正整数
        wait (empty) ;             //检查是否有空闲的缓冲区
        wait (mutex) ;             //有空闲的缓冲区，互斥访问缓冲区
        put (number) ;             //把 number 放入空闲缓冲区中
        signed (mutex) ;           //放完产品，释放互斥信号
        if (number%2= =0)
           signal (even) ;         //若生产的该数据是偶数，则缓冲区中偶数个数加 1
        else
           signal (odd) ;      //若生产的该数据是奇数，则缓冲区中奇数个数加 1
    }
Process P2：
```

```
while（1）
  {
        wait（odd）；              //看看缓冲区有没有奇数
        wait（mutex）；            //缓冲区有奇数，互斥访问缓冲区
        getodd（）；               //从缓冲区读取一个奇数
        signal（mutex）；          //释放互斥信号量
        signal（empty）；          //取走一个数据，空闲缓冲区加 1
        countodd（）；             //重新计算奇数的个数
  }
Process P3：
  while（1）
  {
        wait（even）；             //看看缓冲区有没有偶数
        wait（mutex）；            //缓冲区有偶数，互斥访问缓冲区
        geteven（）；              //从缓冲区取走一个偶数
        signal（mutex）；          //释放缓冲区互斥信号量
        signal（empty）；          //数据被取走了，空闲缓冲区加 1
        counteven（）；            //重新计算偶数的个数
  }
```

9.【解析】本题的过程，很多同学分析不清楚，借助图 2.8，我们可分析本题。顾客 i 到达银行时，若有空闲座位，则取号，等待营业员叫号。取号机需要互斥访问。顾客取走了号码，营业员空闲时，通过叫号，给顾客提供服务。营业员叫号之后，可供顾客等待的座位有空闲，其他顾客可进入银行等待。

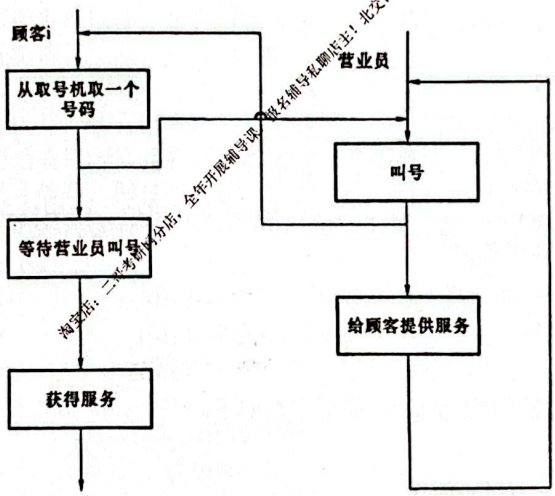

图 2.8

利用解 PV 操作的“五步曲”，可解本题如下。

第一步：找进程

①顾客进程（每个顾客一个进程）；②营业员（1 个服务窗口，设只有一个营业员）

第二步：找动作

（1）顾客进程（由图 2.8 可知顾客进程的动作如下）

①从取号机取一个号码；②坐在座位上等待营业员叫号；③获得服务；④离开。

（2）营业员进程（由图 2.8 可知营业员进程的动作如下）

①叫号；②为顾客服务。

第三步：找关系

①取号机每次仅允许一个顾客使用，顾客间取号是互斥关系。

②顾客从取号机上取走了号码，营业员才能叫号，是同步关系。

③营业员叫号，有顾客离开供顾客等待的座位去服务窗口让营业员为其服务，新顾客看到有空

闲的等待座位，才能进入银行取号，是同步关系。

第四步：找初值

①顾客取号时取号机需要互斥访问，设置 mutex 互斥信号，初始值为 1。

②设初始化状态下，银行里无人等待营业员的服务，设信号 empty 为空座位的数量，初始值 10.

③设 full 信号，用来表示当前等待营业员服务的顾客数量，0<full<10，full 的初始值为 0，表示无人等待营业员的服务。

第五步：写算法

算法如下。

```
semaphore empty=10;              //表示空余座位数量的资源信号量，初值为 10
semaphore mutex=1;               //互斥信号量，初值为 1，用于实现对取号机的互斥访问
semaphore full=0;                //表示顾客数量的资源信号量，初值为 0
cobegin
  {
    process 顾客 i
      {
          P（empty）;            //看看有没有空椅子
          P（mutex）;            //有空椅子，互斥去取号机上取号
          从取号机获得一个号码;  //
          V（mutex）;            //释放取号机互斥取号信号
          V（full）;             //等待营业员服务的顾客数量加 1
          P（call）;             //等待营业员叫号
          获得服务;
      }
    process 营业员
      {
          while（TRUE）
            {
              P（full）;         //有顾客等待服务
              V（call）;         //营业员叫号
              V（empty）;        //相应号码的顾客离开座位，空闲椅子数量加 1
              为顾客服务;
            }
      }
  }
coend
```

10.【解析】利用解 PV 算法的“五步曲”，解析本题如下。

第一步：找进程

参观者进程（若干）

第二步：找操作（每个参观者的进程都可以用如下操作来描述）

①进门；②参观；③出门。

第三步：找关系

出入口一次仅允许一个人通过，参观者在进入口处是互斥出入的关系。

第四步：找初值

①出入口处需互斥通过，故而需要一个互斥信号 mutex，初值为 1。

②博物馆最多可容纳 500 人，设 empty 信号，表示当前还可进馆参观的人数，初始时博物馆里空无一人，empty=500。

第五步：写算法

算法如下。

```
semaphore empty=500;             //博物馆可以容纳的最多人数
semaphore mutex=1;               //用于控制参观者互斥地访问出入口
cobegin
   参观者进程 i:
```

```
        {
            P（empty）；              //先看看博物馆是不是满了
            P（mutex）；              //博物馆没有满可以进去，则在出入口处互斥进入
            进门；
            V（mutex）；              //释放互斥信号量①
            参观；
            P（mutex）；              //参观完毕，出口处互斥出馆②
            出门；
            V（mutex）；              //释放互斥信号量③
            V（empty）；              //离开博物馆了，管内可允许进入人数加 1
        }
    coend
```

【问题】设想一下，取消①和②处的 PV 操作，会产生什么问题？

【解析】这个问题很简单，但是很有代表性，因为现象很明显。一个顾客抢占到了 mutex 信号，进馆参观了，但是拿着这个信号量不放，导致：馆内的人出不来，馆外的人进不去。

在本章的 PV 算法中，有很多 PV 操作稍微改动一下位置，会影响算法的性能。请同学们试着改变一下第 7 题的吃水果问题中的儿子和女儿中的 V 操作。若是他们吃完水果之后再释放空盘信号，会出现什么情况？答：一个孩子拿到了水果，就在那儿慢腾腾地吃，爸爸妈妈削好的水果放不进盘子去，另一个孩子眼巴巴地干等。

11.【解析】本题考查经典同步问题中读者写者问题。

（1）读者优先算法

对于读者优先，应满足下列条件。

如果新读者到：

①无读者、写者，新读者可以读；②有写者等待，但有其他读者正在读，则新读者也可以读；③有写者写，新读者等待。

如果新写者到：

①无读者，新写者可以写；②有读者，新写者等待；③有其他写者，新写者等待。

读者优先的设计思想是读进程只要看到有其他读进程正在读，就可以继续进行读；写进程必须等待所有读进程都不读时才能写，即使写进程可能比一些读进程更早提出申请。

单纯使用信号量是不能解决读者—写者问题的，必须引入计数器 Readcmmt 来对读者进行计数。

读者优先算法描述如下。

```
semaphore wmutex=1;          //用于写者与其他读者/写者互斥的访问共享数据
semaphore rmutex=1;          //用于读者互斥的访问
int readcount=0;             //读者计数器
cobegin
  procedurereader_i          //读者进程
    begin                    //i=l，2，....
        P（rmutex）；        //对共享变量 Readcount 操作的互斥信号量
        Readcount++;         //读者数量加 1
          if（readcount==1） //若是第一个读者，需等待 wmutex 信号，封锁写者写
              P（wmutex）；
        V（rmutex）；        //修改完 Readcount 信号量，释放互斥访问信号
        读数据；
        P（rmutex）；        //读完数据，需要修改 Readcount 信号
        Readcount--;         //读者数量减 1
          if（readcount==0） //如果当前已无读者在读，释放 wmutex，允许写者写
              V（wmutex）；
        V（rmutex）；        //释放修改共享变董 Readcount 的互斥信号量
      end
  procedureWriter_j
    begin                    //j=l，2，....
```

```
            P（wmutex）；              //看看有没有读进程，若没有读进程，则可写
            写数据；
            V（wmutex）；              /*写完数据，释放 wmutex 信号，若有读者要进入
          end                          //临界区，唤醒一个读者进程进入临界区*/
    coend
```

（2）写者优先算法

写者优先算法的设计思想是一个写者到达时，如果有正在工作的读者，该写者只要等待正在工作的读者完成就可以进行写操作，而不必等候其后面到来的读者也完成它们的工作。

该算法当一个写者在等待时，后到达的读者是在写者之后须等待，而不是立即允许进入。关于实现写者优先的算法，其实有很多种，我们可采用下面的算法来实现写者优先。

```
    semaphore wmutex=1;            //写者访问 writecount 的互斥信号量
    semaphore rmutex=1;            //读者访问 readcount 的互斥信号量
    int readcount=0，writecount=0; //读者写者计数器
      semaphore rwrautex=1;        //读者、写者互斥访问文件
    semaphore r=1;                 //读者队列
    semaphore read=1;              //一个读者与一个写者竞争访问文件
      cobegin
    procedure reader_i
      begin                        //i=1，2，....
        repeat
            P（r）；               //保证写者优先，若有写者写第二个之后读者在 r 上排队
            P（read）；            //一个读进程与一个写进程在 read 上竞争
            P（rmutex）；          //等待修改读者数量 readcount 的互斥信号
            readcount++；          //读者数量加 1
            if(readcount==1)       //第一个读进程需要判断是否有写进程在临界区，若有
                                   //读进程等待，若无，阻塞写进程
                  P（wmutex）；
            V（rmutex）；          //释放修改共享变量 readcount 的互斥信号量
            V（read）；            //从 read 队列中唤醒一个进程
            V（r）；
            读数据；
            P（rmutex）；          //等待修改读者数量 readcount 的互斥信号
            readcount--；          //读者数量减 1
            if（readcount==0）     //最后一个离开临界区的读进程需要判断是否有
                                   //写进程需要进入临界区
                V（wmutex）；      //若有，唤醒一个写进程进临界区
            V（rmutex）；          //释放互斥信号量
            until false;
          End

  procedureWriter_j
    begin                          //j=1，2，....
      repeat
          P（wmutex）；            //开始对 writecount 共享变量进行互斥访问
          writecount++；           //写者数量加 1
          if(writecount==1)P(read)；//第一个写进程需要判断是否有读进程在临界区
                                   //若有，写进程阻塞，若无，阻塞新的读进程
          V（wmutex）；            //释放互斥信号量
          P（rwmutex）；           //其他写进程在 rwmutex 上排队
          写数据；
          P（wmutex）；            //写完文件，离开登记
          writecount--；           //写者人数减 1
```

```
        if(writecount==0)V(read); //最后一个离开临界区的写进程需要判断是
                                  //否有读进程需要进入，若有
                                  //唤醒一个读进程进入临界区
        V（wmutex）;              //释放互斥信号量
      until false;
    end
coend
```

12.【解析】利用解 PV 算法“五步曲”，可解本题如下。

第一步：找进程

①理发师；②等待理发的顾客（每个顾客的进程一样）。

第二步：找操作

（1）理发师：①（无顾客）睡觉；②（有顾客）给顾客理发。

（2）顾客：①（理发师在理发，且有空椅子则）等待理发；②（无空椅子）离开。

第三步：找关系

①有顾客等待时，理发师才能理发；

②理发师空闲时，顾客才能理发。

第四步：设初值

①用变量 waiting 用来记录等候理发的顾客数，初值均为 0；

②用信号量 customers 来记录等候理发的顾客数，初值为 0；

③信号量 barbers 用来记录正在等候顾客的理发师数，初值为 1；

④信号量 mutex 用于互斥访问 waiting 变量，初值为 1。

算法如下。

```
int waiting=0;                          //等候理发的顾客数
int chairs=n;                           //为顾客准备的椅子数
semaphore customers=0，barbers=1，mutex=1;
cobegin
  barber（）
    begin
      while（TRUE）{
        P（cutomers）;                  //若无顾客，理发师睡眠
        P（mutex）;                     //进程互斥
        waiting：=waiting-1;            //等候顾客数少一个
        V（mutex）;                     //开放临界区
        理发师给顾客理发;
        V（barbers）;                   //理发师替该顾客理完发后，空闲了
      }
      }
    end                                 //其他等待理发的顾客可以理发了
customer（）
    begin
        P（mutex）;                     //等待互斥访问 waiting 的信号量
        if（waiting<n）{
            waiting；=waiting+1;        //等候顾客数加 1
            V（customers）;             //必要的话唤醒理发师
        V（mutex）;                     //开放临界区
        P（barbers）;                   //无理发师，顾客坐着养神
        get-haircut（）;                //一个顾客坐下理发/
      }
       else V（mutex）;                 //人满了，不等了，直接走了
    end
coend
```

13.【解析】利用解 PV 算法“五步曲”，假设 1 号吸烟者拥有烟草 tobacco，2 号吸烟者拥有纸 paper，

3 号吸烟者拥有火柴 match。解本题如下。

第一步：找进程

①供应者；②吸烟者 1（有烟草）；③吸烟者 2（有纸）；④吸烟者 3（有火柴）。

第二步：找动作

（1）供应者随机供应两样东西。

（2）吸烟者 1：①取原料（纸和火柴）；②做香烟；③吸烟。

（3）吸烟者 2 进程的动作：①取原料（烟草和火柴）；②做香烟；③吸烟。

（4）吸烟者 3 进程的动作：①取原料（烟草和纸）；②做香烟；③吸烟。

第三步：找关系

①供应者同一时刻最多只供应两种吸烟材料，所以三个吸烟者不需要用一个特别的信号量来表不。

②供应者与三个吸烟者之间分别存在同步关系。

第四步：找初值

①分别用信号量 S1、S2、S3 来表示供应者分别供应给吸烟者 1、吸烟者 2、吸烟者 3 的信号量，初值为 0。

②用信号量 S 表示当前无人吸烟的信号，用于吸烟者吸完烟后唤醒供应者进程，初始值为 1。

第五步：写 PV 算法。

PV 算法描述如下。

```
var S，S1，S2，S3：semaphore：=1，0，0，0；
fiagl，flag2，fiag3：Boolean：=true，true，true；
cobegin
process 供应者
    begin
      repeat
         P（S）；
         取两样香烟原料放桌上，由 flagi 标记
                                          flagl、flag2、flag3 代表烟草、纸、火柴
         if flag2&flag3 thenV（S1）；      //供纸和火柴
         else if flagl&fiag3 thenV（S2）   //供烟草和火柴
         elseV（S3）；                        //供烟草和纸
      untilefalse；
    end
process 吸烟者 1
    begin
      repeat
            P（Sl）；                      //等待供应者供应纸和火柴
            取纸和火柴；
            做香烟；
            V（S）；                          //唤醒供应者进程，可以继续随机供应原料
        了
            吸香烟；
       untile false；
   end
process 吸烟者 2
   begin
      repeat
            P（S2）；                      //等待供应者供应烟草和火柴
            取原料；
            做香烟；
            V（S）；                          //唤醒供应者进程，可以继续随机供应原料
       了
           吸香烟；
     untile false；
```

```
    end
  process 吸烟者 3
    begin
      repeat
            P（S3）；                          //等待供应者供应烟草和纸
            取原料；
            做香烟；
            V（S）；                           //唤醒供应者进程，可以继续随机供应原料了
            吸香烟；
      untile false；
    end
  coend
```

14.【解析】很多操作系统算法题中，并没有指明有哪些进程，需要考生自己找。本题中，四个进程当做已知条件告知。

第一步：找进程

①进程 R1；②进程 R2；③进程 W1；④进程 W2。

第二步：找动作

（1）R1 进程：①从磁盘上读入一个数；②将数存放到缓冲区中。

（2）W1 进程：①将 R1 进程放进缓冲区中的数取出；②打印输出。

（3）R2 进程：①从键盘读入一个数；②将数存放到缓冲区中。

（4）W2 进程：①将 R2 进程放进缓冲区中的数取出；②打印输出。

第三步：找（同步和互斥）关系

①当缓存区无数据时，R1 才可以向缓冲区写入数据；

②当缓存区无数据时，R2 才可以向缓冲区写入数据；

③当缓存区中是 R1 写的数据时，W1 才可以将数据从缓冲区中读出；

④当缓存区中是 R2 写的数据时，W2 才可以将数据从缓冲区中读出。

第四步：找初值

①由于 R1、R2 只能缓冲区为空的时候，才能写入。仅需要一个信号量 empty，来表示缓冲区的状态即可。可以 empty=1 表示缓冲区为空，R1、R2 可以向缓冲区写入数据；

②为了识别缓冲区的数据时 R1 写的数据，用一个 full1 信号来表示。fulll=1 表示缓冲区的数据由 R1 写入；

③为了识别缓冲区的数据时 R2 写的数据，用一个 full2 信号来表示。full2=1 表示缓冲区的数据由 R2 写入。

第五步：写 PV 操作

算法描述如下。

```
empty，fulll，full2：semaphore：=1，0，0；
process R1（）
    begin
        repeat
            从磁盘上读入一个数；
            P（empty）；                  //等待缓冲区有空闲
            将数存放到缓冲区中；
            V（full1）；                  //R1 写缓冲区的数量加 1
         until false；
    end；

process R2（）
    begin
        repeat
            从键盘上读入一个数；
            P（empty）；                  //等待缓冲区有空闲
            将数存放到缓冲区中；
```

```
            V（full2）；              //R2 写缓冲区的数量加 1
        until false;
    end;

process W1（）
    begin
        repeat
            P（full1）；              //看看缓冲区是否有 R1 写的数据
            将缓冲区中的数取出；
            V（empty）；              //取走数据后，空闲缓冲区多了一个
            打印输出；
        until false;
    end;

process W2（）
    begin
        repeat
            P（full2）；              //看看缓冲区是否有 R2 写的数据
            将缓冲区中的数取出；
            V（empty）；              //取走数据后，空闲缓冲区多了一个
            打印输出；
        until false;
    end;
```

15.【解析】本题已给出司机和售票员的活动顺序。再看司机和售票员操作关系，如图 2.9 所示。

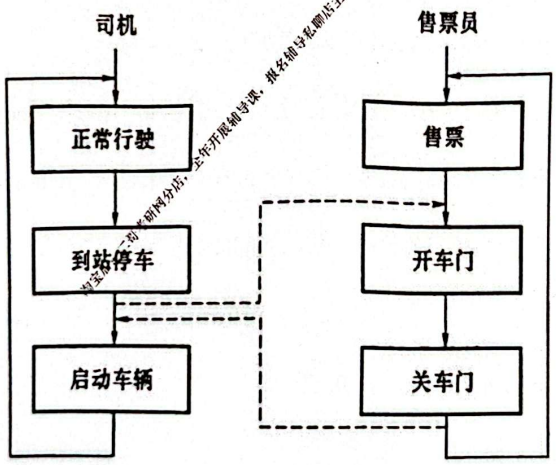

图 2.9

利用解 PV 算法的“五步曲”，可有如下解题过程。

第一步：找进程

①司机；②售票员。

第二步：找动作

（1）司机：①启动车辆；②正常行车；③到站停车。

（2）售票员：①关闭车门；②售票；③开车门。

第三步：找关系

①当售票员将车门关上后司机才可以启动车辆，同步关系；

②司机到站停车后，售票员打开车门，同步关系。

第四步：找初值

①关门信号 open，表示是否允许售票员开门，初值为 0，表示门开着；

②开车信号 start，初值为 0，表示没有停车。

第五步：写 PV 算法

算法描述如下。

```
open，start：：semaphore;
  open：=0;
  start：=0;
processDriver（）
    begin
        repeat
            正常行驶;
            到站停车;
            V（open）;          //可以开车门了
            P（start）;         //等待售票员启动车辆的信号
            启动车辆;
        until false;
    end;
process Busman（）
    begin
        repeat
                售票;
            P（open）;          //等待司机的开车门信号
            开车门;
                关车门;
            V（start）;         //售票员关门了，司机可以开车了
        until false;
    end
```

16.【解析】哲学家进程问题，利用 PV 算法解题“五步曲”可有过程如下。

第一步：找进程：5 个进程，5 个哲学家各一个进程。

第二步：找操作（五个哲学家都是一样的动作）

①思考；②拿起左边筷子；③拿起右边筷子；④吃面条；⑤放下右边筷子；⑥放下左边筷子。

第三步：找关系

筷子是互斥资源，每个人都只能使用他左右的两根筷子。

第四步：找初值

①最多只能有 4 个人同时拿起筷子吃饭，并用 mim 表示允许吃面的人数，初值为 4。

②要是 5 个人同时拿起筷子，每个人都只能拿到一只筷子，等待别人放下筷子。这种情况下，产生了死锁。令 chopstick[5]表示 5 根筷子，初值为 1。

第五步：写 PV 算法

算法描述如下。

```
chopstick[0～4]：semaphore;
num：semaphore;
chopstick[0～4]：=1;
cobegin
    process Pi（i=0，2，…，4）
        begin
            思考;
            P（num）;                  //看看是不是已经抢到拿起筷子的资格了
            P（chopstick[i]）;         //拿起左边的筷子
            P（chopstick[i+1%5]）;     //拿起右边的筷子
            吃面条;
            V（chopstick[i+1%5]）;     //放下右边的筷子
            V（chopstick[i]）;         //放下左边的筷子
            V（num）;                  //吃过了，资格让给别人
```

```
        end
coend;
```

17.【解析】根据题目要求，可利用解 PV 操作的“五步曲”解本题如下。

第一步：找进程

读者（数量若干）。

第二步：找操作（每个读者的操作一样）

①登记；②进入阅览室；③读书；④离开阅览室；⑤注销。

第三步：找关系

①当教室内有空座位时，读者才可以登记进入阅览室，是同步关系。

②同时只能有一个读者在人口处进行登记，同时只能有一个读者在出口处进行注销，互斥关系。

第四步：找初值

①设座位的信号量为 seat，表示还有空余座位的数量，初始状态下设教室无人，seat=100。

②设互斥登记进入的信号量为 Sin，初始值为 1。

③设互斥离开注销的信号量为 Sout，初始值为 1。

第五步：写算法

算法描述如下。

```
Sin，Sout，seat：semaphore：=1，1，100；
cobegin
    processReader-i（i=1，2，…，n）；
        begin
            P（seat）；          //看看是不是有空闲座位
            P（Sin）；           //有座位，则互斥进入登记
            登记
            V（Sin）；           //登记完毕，释放登记进入的互斥信号量
            进入阅览室；
            读书；
            离开阅览室；
            P（Sout）；          //离开了，互斥离开注销
            注销；
            V（Sout）；          //释放离开注销的互斥访问信号量
            V（seat）；          //空闲座位数量加 1
        end
coend;
```


第 3 章 处理机调度和死锁

章 923 操作系统大纲考试内容】

要求理解的内容包括：处理机调度类型与模型，处理机调度实现机理，调度算法与评价准则；

要求掌握的内容包括：处理机主要调度算法设计实现及应用。

利用银行家算法给出避免死锁的资源分配方案，死锁检测算法及应用。

考点 1 调度的概念和基本准则

考点主要内容】

【1】调度的基本概念；

【2】三级调度：[1]作业调度；[2]中级调度；[3]进程调度。

【3】调度方式：剥夺式和非剥夺式。

【4】CPU 调度准则：系统吞吐量、CPU 利用率、周转时间、等待时间等。

考点核心命题】

一、选择题部分

.在一般操作系统中必不可少的调度是（ B ）A

..进程调度　　B.中级调度　　C.高级调度　　=　　D.作业调度

在多进程系统中，进程什么时候占用处理器，取决于（ B ）。

.进程相应的程序段的长度　　B.进程调度策略