操作系统之 PV金典

本版作者：王昭礼

中文版本：版本 1.0，2008年4月

邮箱地址：WZLF11@163.com

Preface

本文仅供学习交流使用，严禁商业交易。题目主要来自于网络以及本人搜集整理的各个高校考研试题(文中业已注明)，答案多数经我个人的检查修改后收录该文，还有一些答案是由我自己写的。目的在于让大家更加系统掌握P,V操作题目的处理技巧。在此声明：若有错误请与我联系，我会尽快更正其中的错误并升级版本。若有更为新颖的题目也可以和我联系继续添加到文章当中。若有版权争议，请与我本人联系！

♡

目　　录

第一章 The P,V Theorem 4
一 Introduction of P,V Theorem 4
(一) Some Conceptions 4
(二) The Most Important Conceptions 4
二 Several Typical Examples 5
(一) 生产者一消费者问题(producer-consumer problem) 5
(二) 读者一写者问题(Readers-Writers Problem) 6
(三) 哲学家进餐问题(The Dining Philosophers Problem) 8
(四) 理发师问题(Barber Problem) 9
(五) 吸烟者问题(Smoker Problem) 10
第二章 Typical Excises 13
一 生产者-消费者问题扩展 13
(一) 扩展一(北大1991) 13
(二) 扩展二(北大1995) 13
(三) 扩展三 15
二 读者-写者问题扩展 16
(一) 扩展一 16
(二) 扩展二(苏州大学2004) 17
三 吸烟者问题扩展 18
(一) 扩展一(北大1999) 18
(二) 扩展二 18
第三章 九阴真经之研究生题辑 20
一 真经之银行排队问题(北京大学2000) 20
二 真经之生产消费问题扩展(浙江大学2001) 21
三 华南理工2000 23
四 真经之生产者消费者扩展(同济1996) 24
五 真经之理发师问题扩展(电子科技大学2000) 24
六 真经之读者写者问题扩展(南航2001) 26
七 真经之南航2002 27
八 真经之管道通信问题(西北工大2000) 28
九 真经之吃水果问题(南京大学2000) 29
十 真经之安全岛问题(南开1997) 30
十一真经之珍珑棋局问题 32

十二真经之公交车问题(哈尔滨工业大学2000)…… 32
十三真经之少林寺问题 …… 33
十四真经之过桥问题…… 34
十五真经之帐户问题(南京大学2000)…… 39
十六真经之机房上机问题(北大1997)…… 40
十七真经之3进程问题 …… 41
十八真经之生产消费问题扩展(北大1994) …… 46
十九真经之流程问题(北大1991) …… 48
二十九阴真经之北航篇 …… 50
(一) 智取考场 …… 50
(二) 读写者问题(2005)…… 50
(三) 吸烟者问题…… 51
(四) 生产者-消费者扩展 …… 52
(五) 阅览室问题…… 52
(六) P,V改错(2001) …… 54
(七) 面包店(2001) …… 54
(八) 公交车问题(2002)…… 54
(九) P,V改错(2002) …… 54
第四章 福尔摩斯探案之网络搜捕 …… 56
一 打印机问题 …… 56
二 批处理系统问题…… 62
三 桔子汁生产线问题 …… 63
四 保管员问题 …… 64
五 招聘问题 …… 65
六 博物馆-公园问题 …… 67
七 生产流水线问题…… 68
八 知错能改 …… 70
第五章 独孤九剑之一剑定乾坤…… 73
一 试题类型总结 …… 73
(一) 名词解释 …… 73
(二) 填空题 …… 73
(三) 判断题 …… 73
(四) P,V题…… 74
(五) 计算题 …… 74
(六) 证明题 …… 74
二 常考概念归纳 …… 74

第一章 The P,V Theorem

在操作系统理论中有一个非常重要的概念叫做P,V原语。在我们研究进程间的互斥的时候经常会引入这个概念，将P,V操作方法与加锁的方法相比较，来解决进程间的互斥问题。实际上，他的应用范围很广，他不但可以解决进程管理当中的互斥问题，而且我们还可以利用此方法解决进程同步与进程通信的问题。

一 Introduction of P,V Theorem

阐述P,V原语的理论不得不提到的一个人便是赫赫有名的荷兰科学家E.W.Dijkstra。如果你对这位科学家没有什么印象的话，提起解决图论中最短路径问题的Dijkstra算法应当是我们再熟悉不过的了。P,V原语的概念以及P,V操作当中需要使用到的信号量的概念都是由他在1965年提出的。

1 Some Conceptions

信号量是最早出现的用来解决进程同步与互斥问题的机制，包括一个称为信号量的变量及对它进行的两个原语操作。信号量为一个整数，我们设这个信号量为：S。很显然，我们规定在S大于等于零的时候代表可供并发进程使用的资源实体数，S小于零的时候，表示正在等待使用临界区的进程的个数。根据这个原则，在给信号量附初值的时候，我们显然就要设初值大于零。

p操作和v操作是不可中断的程序段，称为原语。P,V原语中P是荷兰语的Passeren，相当于英文的pass,V是荷兰语的Verhoog,相当于英文中的incremnet。

P原语操作的动作是:

（1）S减1;

（2）若S减1后仍大于或等于零，则进程继续执行;

（3）若S减1后小于零，则该进程被阻塞后进入与该信号相对应的队列中，然后转进程调度。

V原语操作的动作是:

（1）S加1;

（2）若相加结果大于零，则进程继续执行;

（3）若相加结果小于或等于零，则从该信号的等待队列中唤醒一等待进程，然后再返回原进程继续执行或转进程调度。

需要提醒大家的是：P,V操作首先是一个原语操作，对于每一个进程来说，都只能进行一次。而且必须成对使用。且在P,V愿语执行期间不允许有中断的发生。

对于具体的实现，方法非常多，可以用硬件实现，也可以用软件实现。这里不再赘述。

2 The Most Important Conceptions

临界资源是指每次仅允许一个进程访问的资源。属于临界资源可以是硬件的打印机、磁带机等,软件的有消息缓冲队列、变量、数组、缓冲区等。每个进程中访问临界资

源的那段程序称为临界区（临界资源是一次仅允许一个进程使用的共享资源）。每次只准许一个进程进入临界区，该进程进入后不允许其他进程进入。

进程的同步和互斥互斥：是指某一资源同时只允许一个访问者对其进行访问，具有唯一性和排它性。但互斥无法限制访问者对资源的访问顺序，即访问是无序的。

同步：是指在互斥的基础上（大多数情况），通过其它机制实现访问者对资源的有序访问。在大多数情况下，同步已经实现了互斥，特别是所有写入资源的情况必定是互斥的。少数情况是指可以允许多个访问者同时访问资源。

二 Several Typical Examples

本节我们讨论几个利用信号量来实现进程互斥和同步的经典例子。这里的问题关键是如何选择信号量和如何安排P、V原语的使用顺序。

依据信号量与进程的关系，我们可把进程中使用的信号量分成私有信号量和公用信号量。私有信号量是指只与制约进程和被制约进程有关的信号量；公用信号量是指与一组并发进程有关的信号量。这里请不要和C++、JAVA 等编程语言的公有、私有相混淆。这里指的是相对于共享资源来说的。

1 生产者一消费者问题(producer-consumer problem)

生产者一消费者问题(producer-consumer problem)是指若干进程通过有限的共享缓冲区交换数据时的缓冲区资源使用问题。

问题描述：假设“生产者”进程不断向共享缓冲区写人数据(即生产数据)，而“消费者”进程不断从共享缓冲区读出数据(即消费数据)；共享缓冲区共有n个；任何时刻只能有一个进程可对共享缓冲区进行操作。所有生产者和消费者之间要协调，以完成对共享缓冲区的操作。

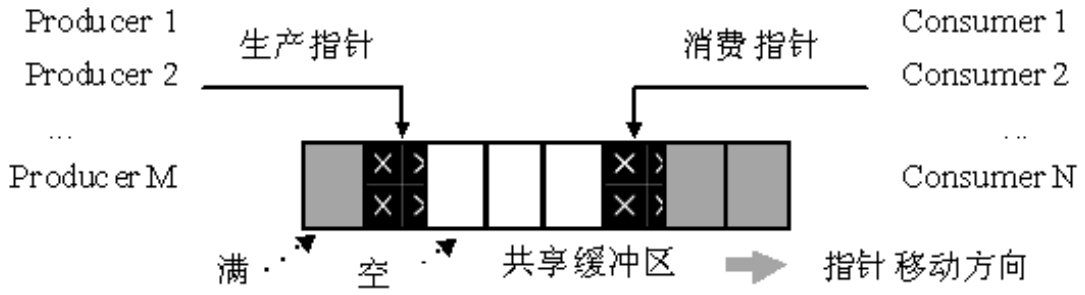


Figure 1.1: producer-consumer problem

我们可把共享缓冲区中的n个缓冲块视为共享资源，生产者写人数据的缓冲块成为消费者可用资源，而消费者读出数据后的缓冲块成为生产者的可用资源。为此，可设置三个信号量：full、empty和mutex。其中：full表示有数据的缓冲块数目，初值是0；empty表示空的缓冲块数初值是n；mutex用于访问缓冲区时的互斥，初值是1。实际上，full和empty间存在如下关系：full+ empty = N

The P,V code Using Pascal

```
buffer: array [0..k-1]of integer;
in,out: 0..k-1;//in记录第一个空缓冲区，out记录第一个不空的缓冲区
empty,full,mutex: semaphore;
                  //empty控制缓冲区不满,full控制缓冲区不空,mutex保护临界区;
```

```
                //初始化empty=k,full=0,mutex=1
cobegin
 procedure producer:      procedure consumer:
   while true then           while true  then
    begin                    begin
    produce(&item);          p(full);
    p(empty);                p(mutex);
    p(mutex);                item:=buffer[out];
    buffer[in]:=item;        out:=(out+1) mod k;
    in:=(in+1) mod k;        v(mutex);
    v(mutex);                v(empty);
    v(full);                 consume(&item);
    end                      end
coend
```

注意: 这里每个进程中各个P操作的次序是重要的。各进程必须先检查自己对应的资源数在确信有可用资源后再申请对整个缓冲区的互斥操作; 否则, 先申请对整个缓冲区的互斥操后申请自己对应的缓冲块资源, 就可能死锁。出现死锁的条件是, 申请到对整个缓冲区的互斥操作后, 才发现自己对应的缓冲块资源, 这时已不可能放弃对整个缓冲区的占用。如果采用AND信号量集, 相应的进入区和退出区都很简单。如生产者的进入区为Swait(empty, mutex), 退出区为Ssignal(full, mutex)。

2 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, 这个数据区可以是一个文件或者主存的一块空间; 有一些只读取这个数据区的进程(Reader)和一些只往数据区写数据的进程(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, 禁止任何读进程度文件。

实验要求用信号量来实现读者写者问题的调度算法。实验提供了Semaphore类, 该类通过P()、V()两个方法实现了P、V原语的功能。实验的任务是修改Reader类的Read方法以及Writer类的Write方法。

我们需要分两种情况实现该问题:

读优先: 要求指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, 他可直接开始读操作, 而不需要等待。

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进行写操作或正在进行写操作, 他要等待该写者完成写操作后才开始读操作。

The P,V code Using Pascal

读者优先算法:

```
rwmutex    用于写者与其他读者/写者互斥的访问共享数据
rmutex     用于读者互斥的访问
```

```
readcount 读者计数器
var rwmutex, rmutex :  semaphore := 1, 1 ;
int readcount = 0;
  cobegin
    procedure  reader_i
     begin      // i=1,2,….
     P(rmutex);
     Readcount + +;
     if (readcount = = 1) P(rwmutex);
     V(rmutex);
     读数据;
     P(rmutex);
     Readcount - -;
     if (readcount = = 0) V(rwmutex);
     V(rmutex);
     end
    procedure Writer_j
     begin    // j = 1,2,….
     P(rwmutex);
     写更新;
     V(rwmutex);
     end
 Coend
```

The P,V code Using Pascal

写者优先:

1)多个读者可以同时进行读

2)写者必须互斥(只允许一个写者写,也不能读者写者同时进行)

3)写者优先于读者(一旦有写者,则后续读者必须等待,唤醒时优先考虑写者)

如果读者数是固定的,我们可采用下面的算法:

rwmutex: 用于写者与其他读者/写者互斥的访问共享数据

rmutex: 该信号量初始值设为10,表示最多允许10个读者进程同时进行读操作

```
  var rwmutex, rmutex :  semaphore := 1, 10 ;
cobegin
 procedure reader_i
   begin   // i=1,2,….
   P(rwmutex); //读者、写者互斥
   P(rmutex);
   V(rwmutex); // 释放读写互斥信号量,允许其它读、写进程访问资源
   读数据;
   V(rmutex);
   end
 procedure Writer_j
   begin   // j = 1,2,….
   P(rwmutex);
```

```
    for (i = 1;i <= 10;i + +) P(rmutex);
       //禁止新读者，并等待已进入的读者退出
    写更新;
    for (i = 1;i <= 10;i + +) V(rmutex);
      // 恢复允许rmutex 值为10
    V(rwmutex);
    end
Coend
```

问题扩展

如果读者写者均是平等的即二者都不优先，如何实现？

3 哲学家进餐问题(The Dining Philosophers Problem)

Figure 1.2: The Dining Philosophers Problem

问题描述:

(由Dijkstra首先提出并解决)5个哲学家围绕一张圆桌而坐，桌子上放着5支筷子，每两个哲学家之间放一支；哲学家的动作包括思考和进餐，进餐时需要同时拿起他左边和右边的两支筷子，思考时则同时将两支筷子放回原处。如何保证哲学家们的动作有序进行？如：不出现相邻者同时要求进餐；不出现有人永远拿不到筷子；

The P,V code Using Pascal

解法一:

```
semaphore Fork[i] := 1(i = 0, 1, 2, 3, 4)
begin
  Thiking;
  Being hangery;
  P(Fork[i mod 5]);
  p(Fork[(i + 1) mod 5]);
  Eating;
  v(Fork[i mod 5]);
  v(Fork[(i + 1) mod 5]);
```

```
  end
  解法二:
  semaphore c[0]~c[4], 初值均为1;
  Integer i = 0, 1,..., 4;
  procedure philosopher_i
   begin
   if i mod 2 = = 0 then
      begin
      p(c[i]);
      p(c[i + 1] mod 5);
      Eating;
      v(c[i]);
      v(c[i + 1] mod 5);
      end
   else
    begin
    p(c[i + 1] mod 5);
    p(c[i]);
    Eating;
    v(c[i + 1] mod 5);
    v(c[i]);
    end
  end
```

---

4 理发师问题(Barber Problem)

问题描述:

理发店理有一位理发师、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。

The P,V code Using Pascal

1)控制变量waiting用来记录等候理发的顾客数，初值均为0;
2)信号量customers用来记录等候理发的顾客数，并用作阻塞理发师进程，初值为0;
3)信号量barbers用来记录正在等候顾客的理发师数，并用作阻塞顾客进程，初值为0;
4)信号量 mutex用于互斥，初值为1

```
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=0,mutex=1;
cobegin
  barber()
  begin
      while(TRUE); //理完一人,还有顾客吗?
```

```
        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(barbers);   //理发师去为一个顾客理发
        V(mutex);     //开放临界区
        cut-hair( ); //正在理发
    end
    customer()
    begin
        P(mutex);  //进程互斥
        if (waiting)
        begin
          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( );       //一个顾客坐下等理/
        end
        else
        V(mutex); //人满了,走吧!
    end
coend
```

---

5 吸烟者问题(Smoker Problem)

问题描述:

三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，每个吸烟者需要三样东西: 烟草、纸和火柴。供应者有丰富的货物提供。三个吸烟者中，第一个有自己的烟草，第二个有自己的纸，第三个有自己的火柴。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。

问题分析:

- 供应者seller随即产生两样东西，提供它们，这里用普通变量来表示
- 吸烟者进程smoker根据其排号不同，拥有不同的一件东西。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。其他号码错误返回。
- 吸烟者的序号代表他们拥有的东西，用他们的序号和供应者产生的两样东西比较，如果都不相等，则说明他拥有的东西和供应者产生的东西匹配，它可以吸烟。如果其中一个相等，则推出，继续排队。
- mutex信号量代表一个只能进入的门，每次只有一个吸烟者可以进入进行比较和吸烟。
- 每个吸烟者在吸烟完毕之后出门之前要叫醒供应者，调用seller进程。

The P,V code Using Pascal

```
vars , S1 ,S2 , S3 ; semaphore ;
S:=1 ; S1:=S2:=S3:=0 ;
fiag1 , flag2 , fiag3 : Boolean ;
fiag1:=flag2:=flag3:=true;
cobegin
 process 供应者
  begin
  repeat
  P(S) ;
  取两样香烟原料放桌上，由flagi标记;
   //nago1 、nage2 、nage3 代表烟草、纸、火柴
  if flag2 & flag3 then V(S1) ;            //供纸和火柴
  else if flag1 & fiag3 then V(S2);        //供烟草和火柴
  else  V(S3) ;                            //供烟草和纸
  untile false ;
  end
 process 吸烟者1
  begin
  repeat
  P(S1) ;
  取原料;
  做香烟;
  V(S) ;
  吸香烟;
  untile false ;
  end
 process 吸烟者2
  begin
  repeat
  p(S2);
  取原料;
  做香烟;
  V(S) ;
  吸香烟;
  untile false ;
  end
 process 吸烟者3
  begin
  repeat
  P(S3);
  取原料;
  做香烟;
  V(S);
  吸香烟;
```

```
  untile false ;
  end
coend
```

第 二 章　Typical Excises

一　生产者-消费者问题扩展

1　扩展一(北大1991)

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.

问题分析:

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.

The P,V code Using Pascal

```
Semaphore mutex=1, sa=N, sb=M;
cobegin
  procedure A:          procedure B:
   while(TURE)            while(TURE)
    begin                 begin
     p(sa);                p(sb);
     p(mutex);             p(mutex);
     A产品入库;            B产品入库;
     V(mutex);             V(mutex);
     V(sb);                V(sa);
    end                   end
coend
```

2　扩展二(北大1995)

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。

问题分析:

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。

The P,V code Using Pascal

```
semaphore mutex=1,a,b=m,empty1,empty2=N,full1,full2=0;
cobegin
  process(A);
  process(B);
  process(C)
coend
 A物品入库
process A
  begin
      while(TRUE)
      begin
      p(empty1);
      P(a);
      p(mutex);
      A物品入库;
      v(mutex);
      V(b);
      v(full1);
      end
  end
  B物品入库:
process B
   begin
      while(TRUE)
      begin
      p(empty2);
      P(b);
      p(mutex);
      B物品入库;
      v(mutex);
      V(a);
      p(full2);
      end
   end
process C
   begin
      while(TRUE)
      begin
      p(full1);
```

```
      p(full2);
      p(a);
      P(b);
      组装;
      V(a);
      v(b);
      v(empty1);
      v(empty2);
      end
   end
```

---

3 扩展三

设P,Q,R共享一个缓冲区，P，Q构成一对生产者-消费者，R既为生产者又为消费者。使用P,V 实现其同步。

问题分析:

略。

The P,V code Using Pascal

```
  var mutex,full,empty: semaphore;
  full:=1;
  empty:=0;
  mutex:=1;
cobegin
 Procedure P        Procedure Q        Procedure R
  begin              begin              begin
  while true         while true         if empty:=1 then
  p(empty);          p(full);             begin
  P(mutex);          P(mutex);            p(empty);
  Product one;      consume one;          P(mutex);
  v(mutex);          v(mutex);            product;
  v(full);           v(empty);            v(mutex);
  end                end                  v(full);
                                          end
                                        if full:=1 then
                                          begin
                                          p(full);
                                          p(mutex);
                                          消费一个产品;
                                          v(mutex);
                                          v(empty);
                                          end
  coend
```

思考:

假使P,Q共享缓冲区容量为N，Q,R共享容量为M，该如何实现？这里不再详述。

二 读者-写者问题扩展

1 扩展一

如果读者写者均是平等的即二者都不优先。

The P,V code Using Pascal

```
var w,s,mutex:semaphore;
RC:integer;
w,s,mutex:=1;
RC:=0;
cobegin
 Procedure Reader          Procedure Writer
  begin                     begin
   while TRUE                while TRUE
    p(w);                     p(w);
    p(mutex);                 p(s);
    if RC==0 then              Writing;
    p(s);                     v(s);
    RC:=RC+1;                 v(w);
    v(mutex);                end
    v(w);
    Reading;
    p(mutex);
    RC:=RC-1;
    if RC==0 then
    v(s);
    v(mutex);
    end
coend
```

对读者-写者问题作一条限制，最多只允许rn个读者同时读。为此，又引入了一个信号量L，赋予其初值为rn，通过执行SP(L,1,1)操作来控制读者的数目，每当一个读者进入时，都要做一次SP(L,1,1)操作，使L的值减1。当有rn个读者进入读后，L便减为0，而第rn+1个读者必然会因执行SP(L,1,1)操作失败而被封锁。利用一般信号量机制解决读者-写者问题的算法描述如下:

The P,V code Using Pascal

```
var  rn:integer;          /*允许同时读的读进程数
     L:semaphore:=rn;     /*控制读进程数信号量,最多rn
     W:semaphore:=1;
begin
```

```
      cobegin
       process reader
         begin
         repeat
         SP(L,1,1 ;W,1,0);
         Read the file;
         SV(L,1);
         until false;
        end
       process writer
        begin
        Repeat
        SP(W,1,1;L,rn,0);
        Write the file;
        SV(W,1);
        until false;
       end
   coend
end.
```

上述算法中，SP(W,1,0) 语句起开关作用，只要没有写者进程进入写，由于这时W=1，读者进程就都可以进入读文件。但一旦有写者进程进入写时，其W=0，则任何读者进程及其他写者进程就无法进入读写。SP(W,1,1;L,rn,o) 语句表示仅当既无写者进程在写(这时W=1) 、又无读者进程在读(这时L=rn) 时，写者进程才能进行临界区写文件。

2 扩展二(苏州大学2004)

在经典的同步问题中有一个读者－写者的问题。它的实现方法一般都是基于读者优先策略的，现在请用PV操 作来实现基于先来先服务策略的读者－写者的问题，具体要求描述如下:

- 存在m个读者和n个一写者，共享同一个缓冲区;
- 当没有读者在读，写者在写时，读者，写者均可进入读或写
- 当有读者在读时:
  (1)写者来了，则写者等待;
  (2)读者来了，分两种情况处理: 无写者等待，则读者可以直接进入读操作，如果有写者等待，则读者必须依次等待;
- 当有写者在写时，写者或读者来了，均需等待;
- 当写者写完后，如果等待队列中第一个是写者，则唤醒该写者; 如果等待队列中的第一个是读者，则唤醒该队列中从读者开始连续的所有读者;
- 当最后一个读者读后，如果有写者在等待，则唤醒第一个等待的写者。

三 吸烟者问题扩展

1 扩展一(北大1999)

在一间酒吧里有三个音乐爱好者队列，第一队的音乐爱好者只有随身听，第二队的只有音乐磁带，第三队只有电池。而要听音乐就必须随身听，音乐磁带和电池这三种物品俱全。酒吧老板依次出售这三种物品中的任意两种。当一名音乐爱好者得到这三种物品并听完一首乐曲后，酒吧老板才能再一次出售这三种物品中的任意两种。于是第二名音乐爱好者得到这三种物品，并开始听乐曲。全部买卖就这样进行下去。试用P，V操作正确解决这一买卖。

问题分析:

该问题与吸烟者解法相同。

2 扩展二

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:

1) 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;

2) 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;

3) 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。(最好也能写出录像厅的进程)

问题分析:

电影院一次只能放映一部影片,希望观看的观众可能有不同的爱好,但每次只能满足部分观众的需求,即希望观看另外两部影片的用户只能等待. 分别为三部影片设置三个信号量s0,s1,s2,初值分别为1,1,1电影院一次只能放一部影片,因此需要互斥使用.由于观看影片的观众有多个,因此必须分别设置三个计数器(初值都是0),用来统计观众个数.当然计数器是个共享变量，需要互斥使用。

The P,V code Using Pascal

```
var s=1,s0=1,s1=1,s2=1:semaphore;
var count0=0,count1=0,count2=0;

cobegin
process videoshow0 //vcd_id = 0
begin
    repeat
        p(s0);
        count0 = count0 +1;
        if(count0=1) p(s);
        v(s0);
        看影片;
```

```
        p(s0);
        count0 = count0 -1;
        if(count0=1) v(s);
        v(s0);
    until false
end

process videoshow1 //vcd_id = 1
begin
    repeat
        p(s1);
        count1 = count1 +1;
        if(count1=1) p(s);
        v(s1);
        看影片;
        p(s1);
        count1 = count1 -1;
        if(count1=1) v(s);
        v(s1);
    until false
end
process videoshow2 //vcd_id =2
begin
    repeat
        p(s2);
        count2 = count2 +1;
        if(count2=1) p(s);
        v(s2);
        看影片;
        p(s2);
        count1 = count1 -1;
        if(count2=1) v(s);
        v(s2);
    until false
end
coend
```

第 三 章　九阴真经之研究生题辑

该部分主要来自各个高校的P,V试题，对于答案我尽量保证正确。经过三次审阅!

一　真经之银行排队问题(北京大学2000)

问题描述:

银行有n个柜员,每个顾客进入银行后先取一个号,并且等着叫号,当一个柜员空闲后,就叫下一个号.

问题分析:

将顾客号码排成一个队列,顾客进入银行领取号码后,将号码由队尾插入;柜员空闲时,从队首取得顾客号码,并且为这个顾客服务,由于队列为若干进程共享，所以需要互斥.柜员空闲时,若有顾客,就叫下一个顾客为之服务.因此,需要设置一个信号量来记录等待服务的顾客数.

The P,V code Using Pascal

```
begin
var mutex=1,customer_count=0:semaphore;
    cobegin
    process customer
        begin
        repeat
        取号码;
        p(mutex);
        进入队列;
        v(mutex);
        v(customer_count);
        end

    process serversi(i=1,...,n)
        begin
        repeat
        p(customer_count);
        p(mutex);
        从队列中取下一个号码;
        v(mutex);
        为该号码持有者服务;
        end
    coend
```

```
end
```

思考:

某车站售票厅，任何时刻最多可容纳20名购票者进入，当售票厅中少于20名购票者时，则厅外的购票者可立即进入，否则需在外面等待。若把一个购票者看作一个进程，请回答下列问题

(1)用PV操作管理这些并发进程时，应怎样定义信号量，写出信号量的初值以及信号量各种取值的含义。

(2)若欲购票者最多为n个人，写出信号量可能的变化范围(最大值和最小值)。

定义信号量S,初值为20，当$s>0$时，它表示可以继续进入购票厅的人数，当$s=0$时表示厅内已有20人正在购票，当$s<0$时$|S|$表示正等待进入的人数。

The P,V code Using Pascal

```
var S:semaphore;
    S=20;
   cobegin
    procedure P_i:
     begin
     p(s);
      .
     Enter and buy ticket;
      .
     v(s)
     end
   coend
```

(2)最大值为20，最小值为20-N。

二 真经之生产消费问题扩展(浙江大学2001)

假设缓冲区buf1和缓冲区buf2无限大，进程p1向buf1写数据，进程p2向buf2写数据，要求buf1数据个数和buf2数据个数的差保持在(m,n)之间(m<n,m,n都是正数).

问题分析:

题中没有给出两个进程执行顺序之间的制约关系，只给出了一个数量上的制约关系，即m≤—buf1数据个数－buf2数据个数≤n. 不需要考虑缓冲区的大小，只需要考虑两个进程的同步和互斥. p2向buf2写数据比p1向buf1写数据的次数最少不超过m次，最多不能超过n次，反之也成立. 所以是一个生产者和消费者问题。将等式展开得: (1)m≤(buf1数据个数－buf2数据个数)≤n; (2)m≤(buf2数据个数－buf1数据个数)≤n;由于m,n都是正数，等式只有一个成立，不妨设(1)成立. 在进程p1和p2都没有运行时，两个缓冲区数据个数之差为0,因此，p1必须先运行，向buf1至少写m+1个数据后再唤醒p2运行. 信号量s1表示p1一次写入的最大量,初值为n，s2表示p2一次写入的最大量,初值为-m.

The P,V code Using Pascal

```
begin
var mutex1=1,mutex2=1,s1=n,s2=-m:semaphore;
```

```
    cobegin
    process p1
        begin
        repeat
        get data;
        p(s1);
        p(mutex1);
        写数据到buf1;
        v(mutex1);
        v(s2);
        end
    process p2
        begin
        repeat;
        get data;
        p(s2);
        p(mutex2);
        写数据到buf2;
        v(mutex2);
        v(s1);
        end
    coend
end
```

思考:

p1和p2每次执行时需要进行一些额外的操作. 对于p2来说, 它首先必须在自己的缓冲区buf2中写入至少m个数据, 此后p1和p2的同步可简单通过两个信号量来控制. 题目的一个变形是要求: -m≤(buf2数据个数 - buf1数据个数)≤n;那么信号量的初值就变成m和n,若只有p1向buf1放入数据而p2不放入数据到buf2中, 则p1最多可放m次. 因此, 设置信号量s1,初值为m,此外, 每当p2放入一个数据到buf2中时, 则使信号量s1增 1 , 即p1增加一次放入数据到buf1的机会. 反之, 若只有p2向buf2放入数据而p1不放入数据到buf1中, 则p2最多可放次. 因此, 设置信号量s2,初值为n,此外, 每当p1放入一个数据到buf1中时, 则使信号量s2增 1 , 即p2增加一次放入数据到buf1的机会.

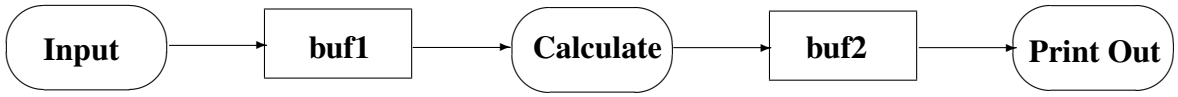
The P,V code Using Pascal

```
begin
var mutex1=1,mutex2=1,s1=m,s2=n:semaphore;
    cobegin
    process p1
        begin
        repeat
        get data;
        p(s1);
        p(mutex1);
        写数据到buf1;
```

```
        v(mutex1);
        v(s2);//p1每放入一个数据到buf1同时使s2增加 1
        end
    process p2
        begin
        repeat;
        get data;
        p(s2);
        p(mutex2);
        写数据到buf2;
        v(mutex2);
        v(s1);//p2每放入一个数据到buf2同时使s1增加 1
        end
    coend
end
```

三 华南理工2000

一个从键盘输入到打印机输出的数据处理流程图如图所示。其中键盘输入进程通过缓冲区buf1把数绝传送给计算进程，计算进程把处理结果通过buf2传送给打印进程。假设上述两个缓冲区的大小分别为n1和n2，试写出键盘输入进程、计算进程及打印进程间的同步算法。



问题分析:

本题解决的试具有多个缓冲区的生产者和消费者之间的多阶段同步问题。由于每个缓冲区中均有多个存储单元，因而要护持使用。所以要为每个缓冲区设置一个互斥信号量。

The P,V code Using Pascal

```
Begin
 var empty1,empty2,full1,full2,mutex1,mutex2:semaphore;
     empty1:=n1;
     empty2:=n2;
     full1:=0;
     full2:=0;
     mutex1:=mutex2:=1;
   cobegin
    procedure  Input  procedure Calculate  procedure  Print_Out
    begin              begin                begin
     Input a data;     p(full1);            p(full2);
     p(empty1);        p(mutex1);           p(mutex2);
     p(mutex1);        Get from buf1;       Get Data from buf2;
```

```
    Put to buf1;        v(mutex1);          v(mutex2);
    v(mutex1);          v(empty1);          v(empty2);
    v(full1);           Calculate it;       Print out the data;
    end                 p(empty2);          end
                        p(mutex2);
                        put result to buf2;
                        v(mutex2);
                        v(full2);
                        end
  coend
```

四 真经之生产者消费者扩展(同济1996)

设有N个计算进程和M个打印进程共享同一个缓冲区，缓冲区长度为8。各计算进程不断地把计算得到的结果送入缓冲区，各打印进程不断的从缓冲区取数并打印。要求:既不漏打，也不重复打印任一个结果。并且，为了高效地工作，计算机进程在使用缓冲区的同时，允许打印进程从缓冲区中取数，反之亦然。请用P、V操作作为同步机制，并用类PASCAL或类C，描述对应于计算进程和打印进程的程序。

五 真经之理发师问题扩展(电子科技大学2000)

有一个理发师,一把理发椅和n把供等候理发的顾客坐的椅子,若没有顾客,则理发师睡觉,当一个顾客到来时,必须唤醒理发师进行理发, 若理发师正在理发,又有顾客到来,则若有空椅子可坐就坐下来等,若没有空椅子就离开.

问题分析:

需要设置一个信号量barber,初值为0,用于控制理发师和顾客之间的同步关系.还需要设置一个信号量customer,初值为0,用于离开顾客与等候顾客之间的同步控制,为了记录等候的顾客数, 应该设置一个计数器count,初值为0.当一个顾客到达时,需要在count上做加1操作,并根据count值的不同分别采取不同的动作，当顾客离开时,要对count上做减1操作,并根据count值的不同分别采取不同的动作;由于count是共享变量,因此要互斥使用，为此设置一个互斥信号量mutex;

The P,V code Using Pascal

```
begin
var barber=0,customer=0,count=0,mutex=1:semaphore;
cobegin
 process barber
  begin
  repeat
  p(customer);
  p(mutex);
  count = count -1;
  v(barber);
  v(mutex);
  理发;
  until false
  end
```

```
 process customer
  begin
  repeat;
  p(mutex);
  if(count<n){
  count = count +1;
  v(customer);
  p(barber);
  理发;
  }
  else{
  v(mutex);
  离开;
  }
  until false
  end
coend
end
```

思考:

有3个理发师,3把理发椅子,n把供等候理发的顾客坐的椅子.由于有3位理发师,所以一次同时可以为三个顾客服务,设置信号量max_capacity, 用于表示空闲椅子的数量,初值为n.信号量barber_chair表示空闲理发师(椅) 的数量,初值为3;信号量cust_ready,finished,leave_b_chair分别表示是否有顾客到来,理发完成,离开理发椅,它们的初值都为0;

The P,V code Using Pascal

```
begin
var max_capacity=n,barber_chair=3,cust_ready=0,finished=0,
    leave_b_chair = 0:semaphore;
cobegin
 process barber
  begin
  repeat
  p(cust_ready);
  理发;
  until false
  end
 process customer
  begin
  repeat;
  p(max_capacity);//是否有空闲椅子;
  进入店里;
  p(barber_chair);//是否有空闲的理发椅;
   坐在理发椅上;
  v(cust_ready);//唤醒理发师;
```

```
  p(finished);//是否完成理发;
  离开理发椅;
  v(leave_b_chair);
  离开店;
  v(max_capacity);
  until false
  end
coend
end
```

六 真经之读者写者问题扩展(南航2001)

问题描述:

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）。如果一只猴子相越过峡谷，它必须看当前是否有别的猴子在逆向通过。请使用P/V 操作来解决该问题。

问题分析:

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，但此时要防止零一方的猴子跨越绳索。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。

The P,V code Using Pascal

```
Begin
  var mutex,Smutex,Nmutex,SmonkeyCount,NmonkeyCount:semaphore;
      SmonkeyCount:=0;//从南向北攀越绳索的猴子数量
      NmonkeyCount:=0;//从北向南攀越绳索的猴子数量
      mutex:=1;//绳索互斥信号量
      Smutex:=1;//南方向猴子间的互斥信号量
      Nmutex:=1;//北方向猴子间的互斥信号量
  cobegin
      procedure South_i(i=1,2,3,...)
        begin
        p(Smutex);
        if SmonkeyCount==0 then
           p(mutex);
        SmonkeyCount:=SmonkeyCount+1;
        v(Smutex);
        Cross the cordage;
```

```
        p(Smutex);
        SmonkeyCount:=SmonkeyCount-1;
        if SmonkeyCount==0 then
           v(mutex);
        v(Nmutex);
        end
       procedure North_j(j=1,2,3,...)
        begin
        p(Nmutex);
        if NmonkeyCount==0 then
           p(mutex);
        NmonkeyCount:=NmonkeyCount+1;
        v(Nmutex);
        Cross the cordage;
        p(Nmutex);
        NmonkeyCount:=NmonkeyCount-1;
        if NmonkeyCount==0 then
           v(mutex);
        v(Nmutex);
        end
   coend
```

七 真经之南航2002

进程P1和P2通过两个缓冲区给进程P11、P12、P21、P22传递信息，进程P11、P12取进程P1的信息，进程P21、P22取进程P2的信息。假定这两个缓冲区一样大小，所要传递的信息也与缓冲区一样大，同一时刻只能由一个进程往缓冲区中送信息或取信息。试用PV操作来实现这6个进程之间的同步与互斥关系。

The P,V code Using Pascal

```
var mutex,S11,S12,S21,S22,empty1,
    empty2,full1,full2:semaphore;
    empty1=empty2=1;
    full1=full2=0;
    sij=0;(i,j=1,2)
    mutex=1;
    cobegin
     Procedure P1:                 procedure P2:
      begin                        begin
      p(empty1);                   p(empty2);
      P(mutex);                    p(mutex);
      put message into buff1;      put message into buff2;
      v(mutex);                    v(mutex);
      v(S11);                      v(s21);
      v(S12);                      v(S22);
      v(fll1);                     v(full2);
```

```
      end                          end
     procedure Sij:(i=1,2,j=1,2)
      begin
      p(fulli);
      p(sij);
      p(mutex);
      Get message from buffi;
      v(mutex);
      v(emptyi)
      end
   coend
```

八 真经之管道通信问题(西北工大2000)

在管道通信机制中，用信号量描述读进程和写进程访问管道文件的过程，假设管道文件大小为10KB.

问题分析:

UNIX系统中，利用一个打开的共享文件来连接两个相互通信的进程，这个共享文件叫管道. 作为管道输入的发送进程，以字符流的形式将信息送入管道，而作为管道输出的接收进程，从管道中获取信息. 管道通信机制要提供三方面的协调能力: (1)互斥. 当一个进程对管道进行读／写操作时，另一个进程必须等待. (2) 同步. 当写进程把数据写入管道后便去睡眠等待，直到输出进程取走数据后唤醒它. 若一次写入的数据超过缓冲区剩余空间的大小，当缓冲区满时，写进程必须阻塞，并唤醒读进程。(3)对方是否存在. 只有确定对方存在时，才能够进行通信. 本题只需要考虑互斥，同步问题。由于只有一对进程访问管道，因此不需要设置互斥信号量，只要设置两个同步信号量empty,full. 分别表示管道可写和可读.

The P,V code Using Pascal

```
begin
pipe:array[09] of kilobytes;
ts=10,length,in=0,out=0:integer;
empty,full:semaphore=1,0;
cobegin
process PipeWriter
 begin
 repeat
 产生数据;
 p(empty);
 length = data length;
 while(length>0 and ts>0)
 begin
 pipe[in] = data of 1KB;
 in = (in+1) mod n;
 ts = ts-1;
 length = length - 1;
```

```
 end
 v(full);
 end
process Consumer
 begin
 repeat;
 p(full);
 从缓冲区取出一件物品;
 out = (out+1) mod n;
 ts = ts +1;
 v(empty);
 end
coend
end
```

九 真经之吃水果问题(南京大学2000)

问题描述:

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。

问题分析:

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。

The P,V code Using Pascal

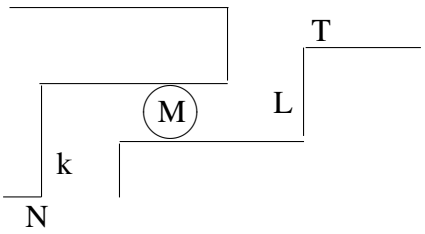
在本题中，应设置三个信号量S、So、Sa，信号量S表示盘子是否为空，其初值为1;
信号量So表示盘中是否有桔子，其初值为0; 信号量Sa表示盘中是否有苹果，其初值为0。
同步描述如下:

```
   S = 1;
   Sa = 0;
   So = 0;
cobegin
 Procedure father;          /*父亲进程*/
 Procedure son;             /*儿子进程*/
 Procedure daughter;        /*女儿进程*/
coend
Procedure father:
  begin
  while(TRUE)
  begin
```

```
  P(S);
  将水果放入盘中;
  if(放入的是桔子)
  V(So);
  else
  V(Sa);
  end
  end
 Procedure son:
  begin
  while(TRUE)
  begin
  P(So);
  从盘中取出桔子;
  V(S);
  吃桔子;
  end
  end
 Procedure daughter:
  begin
  while(TRUE)
  begin
  P(Sa);
  从盘中取出苹果;
  V(S);
  吃苹果;
  end
end
```

十 真经之安全岛问题(南开1997)

在南开大学至天津大学间有一条弯曲的路，每次只允许一辆自行车通过，但中间有小的安全岛M（同时允许两辆车），可供两辆车在已进入两端小车错车，设计算法并使用P，V实现。



问题分析:

由于安全岛M仅仅允许两辆车停留,本应该作为临界资源而要设置信号量，但根据题意,任意时刻进入安全岛的车不会超过两辆(两个方向最多各有一辆)，因此，不需要为M设置信号量,在路口s和路口t都需要设置信号量,以控制来自两个方向的车对路口资

源的争夺.这两个信号量的初值都是1.此外，由于从s到t的一段路只允许一辆车通过,所以还需要设置另外的信号量用于控制,由于M的存在,可以为两端的小路分别设置一个互斥信号量.

The P,V code Using Pascal

```
  var T2N, N2T,L,M,K:semaphore;
  T2N:=1;
  N2T:=1;
  L:=1;
  K:=1;
  M:=2;
cobegin
  Procedure Bike T2N
  begin
    p(T2N);
    p(L);
      go T to L;
    p(M);
      go into  M;
    V(L);
    P(k);
     go K to s;
    V(M);
    V(k);
    V(T2N);
  end
  Procedure Bike N2T
  begin
    P(N2T);
    p(k);
     go v to k;
    p(M);
     go into M;
    V(k);
    P(L);
      go L to T;
    V(M);
    V(L);
    V(N2T);
  end
coend
```

十一 真经之珍珑棋局问题

问题描述:

在一个盒子里，混装了数量相等的黑白围棋子．现在用自动分拣系统把黑子、白子分开，设分拣系统有二个进程P1 和P2，其中P1 拣白子; P2 拣黑子。规定每个进程每次拣一子; 当一个进程在拣时，不允许另一个进程去拣; 当一个进程拣了一子时，必须让另一个进程去拣. 试写出两进程P1 和P2 能并发正确执行的程序。

问题分析:

大家熟悉了生产-消费问题(PC)，这个问题很简单。题目较为新颖，但是本质非常简单即: 生产-消费问题的简化或者说是两个进程的简单同步问题。答案如下:

The P,V code Using Pascal

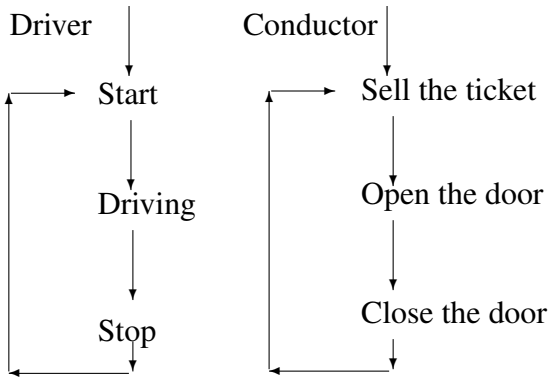
设信号量s1 和s2 分别表示可拣白子和黑子;
不失一般性，若令先拣白子。

```
var S1 , S2 : semaphore;
S1 : = l; S2 : =0;
cobegin
    process P1          process P2
     begin              begin
     repeat             repeat
     P(S1);             p(S2);
     pick The white;    pick the black;
     V(S2);             v(s1);
     until false ;      until false;
     end                end
coend
```

十二 真经之公交车问题(哈尔滨工业大学2000)

问题描述:

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P 、V 操作实现它们的同步。



问题分析:

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：

The P,V code Using Pascal

```
var S1,S2 : semaphore ;
    S1=0; S2=0;
cobegin
Procedure driver                Procedure Conductor
 begin                           begin
 while  TRUE                     while TRUE
 begin                           begin
 P(S1);                          关车门;
 Start;                          v(s1);
 Driving;                        售票;
 Stop;                           p(s2);
 V(S2);                          开车门;
 end                             上下乘客;
end                              end
coend
```

十三　真经之少林寺问题

问题描述：

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。

问题分析：

从井中取水并放入水缸是一个连续的动作可以视为一个进程，从缸中取水为另一个进程。

设水井和水缸为临界资源，引入mutex1,mutex2; 三个水桶无论从井中取水还是放入水缸中都一次一个，应该给他们一个信号量count，抢不到水桶的进程只好为等待，水缸满了时，不可以再放水了。设empty控制入水量，水缸空了时，不可取水设full。

The P,V code Using Pascal

```
var mutex1,mutex2,empty,full,count:semaphore;
mutex1:=mutex2:=1;
empty:=10;
full:=0;
count:=3;

cobegin
  Procedure Fetch_Water                    Procedure Drink_Water
    begin                                   begin
     while true                               while true
      p(empty);                               p(full);
      P(count);                               p(count);
      P(mutex1);                              p(mutex2);
       Get Water;                               Get water and
      v(mutex1);                                Drink water;
      P(mutex2);                              p(mutex2);
      pure water into the jar;                v(empty);
      v(mutex2);                              v(count);
      v(count);                              end
      v(full);
    end
coend
```

十四 真经之过桥问题

问题描述:

一座小桥(最多只能承重两个人)横跨南北两岸，任意时刻同一方向只允许一人过桥，南侧桥段和北侧桥段较窄只能通过一人，桥中央一处宽敞，允许两个人通过或歇息。试用信号灯和PV操作写出南、北两岸过桥的同步算法。

问题分析:

桥上可能没有人，也可能有一人，也可能有两人。

- 两人同时过桥
- 两人都到中间
- 南(北)来者到北(南)段

共需要三个信号量，load用来控制桥上人数，初值为2，表示桥上最多有2人；north用来控制北段桥的使用，初值为1，用于对北段桥互斥；south用来控制南段桥的使用，初值为1，用于对南段桥互斥。

The P,V code Using Pascal

```
var load,north,south:semaphore;
load=2;
north=1;
south=1;
      GO_South()
        P(load);
        P(north);
          过北段桥;
          到桥中间;
        V(north);
        P(south);
          过南段桥;
          到达南岸;
        V(south);
        V(load);
      GO_North()
        P(load);
        P(south);
          过南段桥;
          到桥中间
        V(south);
        P(north);
          过北段桥;
          到达北岸
        V(north);
        V(load);
```

思考:

某条河上有一独木桥可以使行人通过，现在河的两边都有人过河，为了保障安全，依照如下规则过桥:

同一方向的可连续过河，若某方向有人过河，则另一方等待。

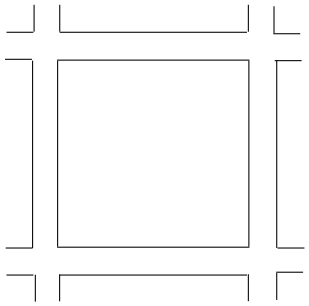
The P,V code Using Pascal

```
var S,S1,S2:semaphore;
    rc1,rc2:integer;
    S,S1,S2:=1;
```

```
rc1=rc2=0;
cobegin
 procedure East2West_i:  procedure West2East_i:
  begin                   begin
  p(S1);                  p(S2);
  rc1:=rc1+1;             rc2:=rc2+1;
  if rc1==1 then P(s);    if rc2==1 then p(s);
  v(S1);                  v(S2);
  过独木桥;               过独木桥;
  p(S1);                  p(S2);
  rc1:=rc1-1;             rc2=rc2-1;
  if rc1==0 then v(S);    if rc2==0 then v(s);
  v(S1);                  v(S2);
  end                     end
coend
```

思考:

一条公路两次横跨运河，两个运河桥相距100 米，均带有闸门，以供船只通过运河桥。运河和公路的交通均是单方向的。运河上的运输由驳船担负。在一驳船接近吊桥A时就拉汽笛警告，若桥上无车辆，吊桥就吊起，直到驳船尾P通过此桥为止。对吊桥B也按同样次序处理。一般典型的驳船长度为200 米，当它在河上航行时是否会产生死锁？若会，说明理由，请提出一个防止死锁的办法，并用信号量来实现驳船的同步。



问题分析:

当汽车或驳船未同时到达桥A 时，以任何次序前进不会产生死锁。但假设汽车驶过了桥A，它在继续前进，并且在驶过桥B 之前，此时有驳船并快速地通过了桥A，驳船头到达桥B，这时会发生死锁。因为若吊起吊桥B 让驳船通过，则汽车无法通过桥B；若不吊起吊桥B让汽车通过，则驳船无法通过桥B。可用两个信号量同步车、船通过两座桥的动作。

The P,V code Using Pascal

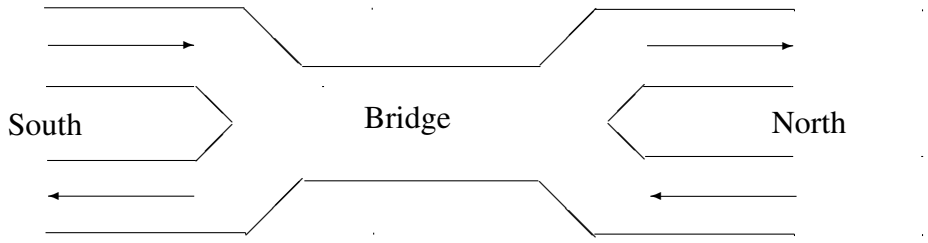
解答:

```
var   Sa,Sb:semaphore;
      Sa:=Sb:=1;
  cobegin
      process 驳船
```

```
    begin
    P(Sa);
    P(Sb);
    船过桥A 、B;
    V(Sa);
    V(Sb);
    end
  process 汽车
    begin
    P(Sa);
    P(Sb);
    车过桥A 、B;
    V(Sa);
    V(Sb);
    end
coend
```

思考(华中科技大学2000):

如图所示:



(1)桥每次只能有一辆车通过。
(2)不允许两车交会,但允许同方向的多辆车依次通过

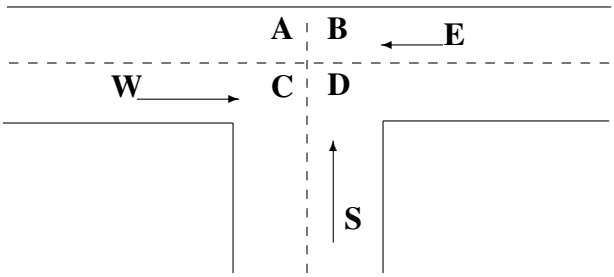
The P,V code Using Pascal

```
begin
mutex:semaphore=1;
 cobegin
  process Scar//南边来的车
    begin
    come;
    p(mutex);
    过桥;
    v(mutex);
    go;
    end
  process Ncar//北边来的车
    begin
    come;
```

```
        p(mutex);
        过桥;
        v(mutex);
        go;
        end
    coend
    end
  begin
    var Smutex=1,Nmutex=1,mutex=1:semaphore;
    SCarCount=0,NCarCount=0:integer;
    cobegin
      process Scari(i=1,2)
        begin
        p(Smutex);
        if(SCarCount=0) then p(mutex);
        SCarCount = SCarCount +1;
        v(Smutex);
        过桥;
        p(Smutex);
        SCarCount = SCarCount -1;
        if(SCarCount=0) then v(mutex);
        v(Smutex);
        end
      process Ncarj(j=1,2)
        begin
        p(Nmutex);
        if(NCarCount=0) then p(mutex);
        NCarCount = NCarCount +1;
        v(Nmutex);
        过桥;
        p(Nmutex);
        NCarCount = NCarCount -1;
        if(NCarCount=0) then v(mutex);
        v(Nmutex);
        end
    coend
    end
```

思考:

设有一个T型路口，其中A、B、C、D处各可容纳一辆车，车行方向如图所示:



试找出死锁并用有序分配法消除之，要求资源编号合理。

解:(1)E方向两辆车分别位于A和B; S方向一辆车位于C; W方向一辆车位于D。(2)S方向两辆车分别位于B和C; E方向一辆车位于A; W方向一辆车位于D。

The P,V code Using Pascal

设位置资源C、B、A、D的编号从低到高依次为1、2、3、4，管理4个位置的信号量分别为S1，S2，S3，S4，信号量的初值均为1。

车辆活动如下:

```
semaphore  S1=1,  S2=1,  S3=1,  s4=l;
cobegin
procedure W:直行  E:左转       S:左转
 begin             begin        begin
 p(S1);            p(S2);       p(S1);
 p(S4);            Enter B;     Enter C;
 Enter D;          p(S3);       p(S2);
 Enter C;          Enter A;     Enter B;
 v(S4);            v(S2);       v(S1);
 Out of C;         p(S4);       p(S3);
 v(S1);            Enter D;     Enter A;
 end               v(S3);       v(S2);
                   out of D;    out of A;
                   v(S4);       V(S3);
                   end          end
coend
```

十五　真经之帐户问题(南京大学2000)

两人公用一个账号，每次限存或取10元;

The P,V code Using Pascal

```
begin
var mutex=1:semaphore;
amount =0:integer;
cobegin
 process save
  m1: integer;
  begin
  repeat
```

```
  p(mutex);
  m1= amount ;
  m1 = m1 +10;
  amout = m1;
  v(mutex);
  end
 process take
  m2: integer;
  begin
  repeat;
  p(mutex);
  m2= amount ;
  m2 = m2 -10;
  amout = m2;
  v(mutex);
  end
coend
```

十六 真经之机房上机问题(北大1997)

某高校计算机系开设网络课并安排上机实习，假设机房共有2m台机器，有2n名学生选课（m，n均大于等于1），规定:

- 每两个学生组成一组，各占一台及其协同完成上机实习;
- 只有一组两个学生到齐，并且此时机房有空闲机器时，该组学生才能进入机房;
- 上机实习由一名教师检查，检查完毕，一组学生同时离开机房

试用P、V实现其过程。

✎注意:

本题目隐含一个进程(Guard)。

The P,V code Using Pascal

```
 var stu,computer,enter,finish,test:semaphore;
 ste:=2N;
 computer:=2M;
 enter:=0;
 finish:=0;
 test:=0;

 cobegin
  Procedure Student     Procedure Teacher    Procedure Guard
    begin                 begin                 begin
    p(computer);          p(finish);             p(stu);
    p(stu);                Test the work;        p(stu);
     Start computer;      v(test);               Enter;
    v(finish);            v(test);              v(enter);
```

```
    v(test);              end                    v(enter);
    v(computer);                                 end
    end
coend
```

十七 真经之3进程问题

问题描述:

设有A、B、C三组进程, 它们互斥地使用某一独占型资源R, 使用前申请, 使用后释放. 资源分配原则如下:

- 当只有一组申请进程时, 该组申请进程依次获得R;
- 当有两组申请进程时, 各组申请进程交替获得R, 组内申请进程交替获得R;
- 当有三组申请进程时, 各组申请进程轮流获得R, 组内申请进程交替获得R.

试用信号灯和PV操作分别给出各组进程的申请活动程序段和释放活动程序段.

The P,V code Using Pascal

```
Int Free=1;//设备状态标志
semaphore mutex=1;
semaphore qa=qb=qc=0;        //各组等待队列
Int counta=countb=countc= 0;//等待队列长度
 A组申请:
P(mutex);
 if Free==1 then
   begin
     Free=0;
     V(mutex);
   end
   else
     begin
       counta++;
       V(mutex);
       P(qa);
     end
    A组释放:
 if countb>0 then
     begin
       countb--;
       V(qb);
     end
   else
     begin
     if countc>0 then
       begin
       countc--;
```

```
        V(qc);
        end
      else
        begin
        if (counta > 0)
          begin
          counta--;
          V(qa);
          end
        else
          begin
          Free=1;
          end
        end
     end
```

A组进程活动可以给出B组和C组进程活动。

思考(南大1997):

今有三个并发进程R，M，P，它们共享了一个可循环使用的缓冲区B，缓冲区B共有N个单元。进程R负责从输入设备读信息，每读一个字符后，把它存放在缓冲区B的一个单元中；进程M负责处理读入的字符，若发现读入的字符中有空格符，则把它改成“,”；进程P负责把处理后的字符取出并打印输出。当缓冲区单元中的字符被进程P取出后，则又可用来存放下一次读入的字符。请用PV操作为同步机制写出它们能正确并发执行的程序。

The P,V code Using Pascal

```
var empty,full1,full2,mutex:semaphore;
char buffer[n];
int in=0,out1=0,out2=0;
    empty=N;
    full1=0;
    full2=0;
    mutex=1;
  cobegin
    procedure R;
    procedure M;
    procedure P;
  coend

  procedure R:          procedure M:        procedure P:
   begin                 begin               begin
    while True then       while True then     while True then
    begin                 begin               begin
    char x;               char x;             char x;
    Read a char to x;     p(full1);           p(full2);
```

```
    p(empty);              p(mutex);            p(mutex);
    p(mutex);              x=buffer[out1];      x=buffer[out2];
    buffer[in]=x;          if x==" "then        out2=(out2+1)%N;
    in=(in+1)%N;             begin              v(mutex);
    v(mutex);                x=",";             v(empty);
    v(full1);                buffer[out1]=x;    输出字符x;
    end                      end                end
   end                     out1=(out1+1)%N;   end
                           v(mutex);
                           v(full2);
                           end
                         end
```

思考(北大1996):

有P1、P2、P3三个进程共享一个表格F，P1对F只读不写，P2对F只写不读，P3对F先读后写。进程可同时读F，但有进程写时，其他进程不能读和写。用(1)信号量和P、V操作。(2)管程编写三进程能正确工作的程序。

The P,V code Using Pascal

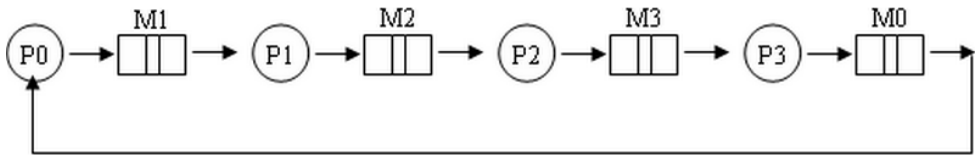
答: (1)信号量和P、V操作。这是读--写者问题的变种。
其中，P3既是读者又是写者。读者与写者之间需要互斥，
写者与写者之间需要互斥，为提高进程运行的并发性，
可让读者尽量优先。

```
var  rmutex,wmutex:semaphore;
rmutex:=wmutex:=1;
count:integer;count:=0;
cobegin
{
  process P1
begin
   repeat
    P(rmutex);
    count:=count+1;
    if count=1 then P(wmutex);
    V(rmutex);
    Read F;
    P(rmutex);
    count:=count-1;
    if count=0 then V(wmutex);
    V(rmutex);
    untile false;
end
  process P2
begin
   repeat
```

```
    P(wmutex);
    Write F;
    V(wmutex);
untile false;
  process P3
begin
   repeat
    P(rmutex);
    count:=count+1;
    if count=1 then P(wmutex);
    V(rmutex);
    Read F;
    P(rmutex);
    count:=count-1;
    if count=0 then V(wmutex);
    V(rmutex);
    P(wmutex);
    Write F;
    V(wmutex);
    untile false;
end
}
coend
```

思考(国防科技大学1996):

如图所示，四个进程Pi（i=0…3）和四个信箱Mj（j=0…3），进程间借助相邻信箱传递消息，即Pi每次从Mi中取一条消息，经加工后送入M(i+1)mod4，其中M0、M1、M2、M3分别可存放3、3、2、2个消息。初始状态下，M0装了三条消息，其余为空。试以PV操作为工具，写出Pi（i=0…3）的同步工作算法。



The P,V code Using Pascal

```
var mutexl , mutexZ , mutex3 , mutex0 :semaphore;
Mutex1 = nutex2:=mutex3:=mutex0:=1;
Empty0,empty1,empty2, empty3; semaphore;
empty:=0 ; empty1:=3 ; empty:=2:=empty3:=2;
full0 , full1 , full2 , full3:semphore ;
full0:=3;full1:=full2:=full3:=0;
in0,in1,in2,in3,out0 ,out2,out3,;intger;
in0:=in1: = in2: = in3:=out0:=out1:=out2:=out3:=0;
cobegin
```

```
process P0
  begin
  repeat
  P(full0);
  P(mutex0);
  从M0[out0]取一条消息;
  out0:=(out0+1)mod 3;
  V(mutex0);
  V(empty0);
  加工消息;
  P(empty1);
  P(mutex1);
  消息已M1[in1];
  In1:=(in1+1) mod 3;
  V(mutex1);
  V(full1);
  untile false;
  end
process P2
 begin
 repeat
 P(full2);
 P(mutex2);
 从M2[out2]取一条消息;
 out2:=(out2+1)mod 2;
 V(mutex2);
 V(empty2);
 加工消息;
 P(empty3);
 P(mutex3);
 消息已M3[in3];
 in3:=(in3+1)mod 2 ;
 V(mutex3);
 V(full3);
 untile false ;
 end
```

```
process P1
 begin
 repeat
 p(full);
 p(mutex1);
Get a message from M[out1];
 out1:=(out1+1)mod 3;
 v(mutex1);
 v(empty1);
  加工消息;
 p(empty2);
 p(mutex2);
  消息已M2[in2];
 In2:=(in2+1)mod 2;
 V(mutex2);
 v(full2);
 until false;
 end
process P3
 begin
 repeat
 p(full3);
 P(mutex3);
从M3[out3] 取一条消息;
 out3:=(out3+1)mod 2;
 v(mutex3);
 v(empty3);
  加工消息;
 p(empty0);
 p(mutex0);
  消息已MO[in0];
 In0:=(in0+1)mod 3;
 v(mutex0);
 v(full0);
 until false;
 end
```

```
coend
```

思考:

四个进程A、B、C、D都要读一个共享文件F，系统允许多个进程同时读文件F。但限制是进程A和进程C不能同时读文件F，进程B和进程D 也不能同时读文件F。为了使这四个进程并发执行时能按系统要求使用文件，现用PV操作进行管理，请回答下面的问题:

(1)如何定义信号量及初值;

(2)在下列的程序中填上适当的P、V操作，以保证它们能正确并发工作:

```
进程A:     进程B:     进程C:     进程D:
begin      begin      begin      begin
...        ...        ...        ...
reade F;   reade F;   reade F;   reade F;
...        ...        ...        ...
end        end        end        end
```

解答:

(1)定义二个信号量S1、S2，初值均为1，即: S1=1，S2=1。其中进程A和C使用信号量S1，进程B和D使用信号量S2。

(2)从[1]到[8]分别为: P(S1) V(S1) P(S2) V(S2) P(S1) V(S1) P(S2) V(S2)

思考(北大1996):

今有k个进程，它们的标号依次为1、2、...、k，如果允许它们同时读文件file，但必须满足条件: 参加同时读文件的进程的标号之和需小于K，请使用: 1)信号量与P、V操作，2)管程，编写出协调多进程读文件的程序。1): 使用信号量与P、V操作

The P,V code Using Pascal

```
var waits,mutex:semaphore;
   numbersum:integer:=0;
   wait:=0;mutex:=1;
cobegin
 process readeri(var number:integer;)
    begin
      P(mutex);
      L: if numbersum+number⩾K then
         begin
         V(mutex);
         P(waits);
         goto L;
         end
      else numbersum:=numbersum+number;
      V(mutex);
      Read file;
      P(mutex);
      numbersub:=numbersum-number;
      V(waits);
      V(mutex);
coend
```

十八 真经之生产消费问题扩展(北大1994)

进程A1、A2，。。。An1通过m个缓冲区向进程B1、B2、···,Bn2不断发送消息。发送和接收工作遵循下列规则:

- 每个发送进程一次发送一个消息，写入一个缓冲区，缓冲区大小等于消息长度
- 对每个消息，B1，B2，Bn2都须各接收一次，读入各自的数据区内
- m个缓冲区都满时，发送进程等待，没有可读消息时，接收进程等待。

试用P、V操作组织正确的发送和接收工作。

问题分析:

每个缓冲区只要写一次但要读n2次，因此，可以看成n2组缓冲区，每个发送者要同时写n2个缓冲区，而每个接收者只要读它自己的缓冲区。

The P,V code Using Pascal

```
Sin[n2]=m Sout[n2]=0;
  cobegin
    procedure Aj:
     while (1)
     begin
     for(i=1;i<=n2;i++)
     P(Sin[i]);
     P(mutex);
     将数据放入缓冲区
     V(mutex);
     for(i=1;i<=n2;i++)
     V(Sout[2]);
     end
    procedure Bi:
     while (1)
     begin
     P(Sout[i]);
     P(mutex);
     从缓冲区取数
     V(mutex);
     V(Sin[i]);
     end
 coend
```

思考(北大1994原题):

判别下列用P、V操作表示同步算法是否正确？如不正确，试说明理由，并修改成正确算法。

```
VAR buffer : ARRAY 0..N-1 OF T;
In, out:0…N-1;
VAR S1,S2:Semaphore;
S1:=0;S2: =N;
In: =out: =0;
    PROCEDURE A:
```

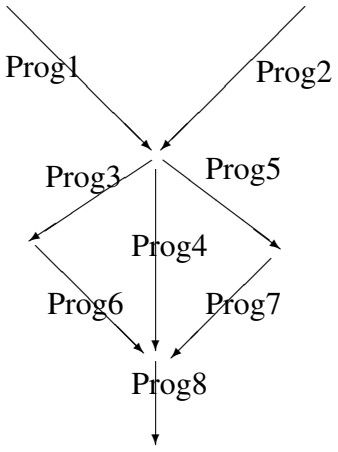
```
 BEGIN
 REPEAT
 生产数据m;
 P(S2);
 Buffer(in):=m;
 In:=(in+1) MODN;
 V(S1);
 Forever
 END
PROCEDURE B:
 BEGIN
 REPEAT
 V(S2);
 m;=buffer(out);
 消费m;
 Out :=(out+1)MODN;
 P(S1);
 forever
 END
```

分析: 本题目是一个标准的生产者 消费者问题。题中所给的算法与标准算法不同，但考生不能因此就说这个算法不正确。考生须仔细分析试题中所给出的算法。在本题中，进程B在使用缓冲区前(读缓冲区)无需进行任何P操作，即进程B不会因任何原因被阻塞。这与题目中控制要求不相符。因此这个算法实现是错误的。此外，对缓冲区的访问也没有用互斥信号量进行控制。

十九 真经之流程问题(北大1991)

问题描述:

设有8个程序prog1,prog2,…prog8。它们在并发系统中执行时有如下图所示的制约关系，使用P、V操作实现这些程序间的同步。



问题分析:

本题目是用来检查考生对使用P、V操作实现进程间同步的掌握情况。一般地，若要求进程B在进程A之后方可执行时，只需在进程P操作，而在进程A执行完成时对同一信

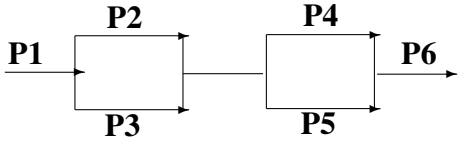
号量进行V操作即可。本题要求列出8个进程（程序）的控制关系，使题目显得较为复杂。但当对进程间的同步理解透彻后，应不难写出对应的程序。解这一类问题还应注意的一点是，要看清图示的制约关系，不要漏掉或多处制约条件。

The P,V code Using Pascal

```
BEGIN
var s13, s14, s15, s23, s24, s25,s36, s48,
    s57, s68, s78:semaphore;
s13 :=0;s14 :=0;s15 :=0;
s23 :=0;s24 :=0;s25 :=0;
s36 :=0;
s48 :=0;
s57 :=0;
s68 :=0;
s78 :=0;
COBEGIN
  prog1:          prog2:          prog3:          prog4:
    BEGIN           BEGIN           BEGIN           BEGIN
    do work;        do work;        V(S13);         P(S14);
    V(s13);         V(s23);         V(S23);         P(S24);
    V(s14);         v(s24);         do work;        do work;
    v(s15);         v(s25);         v(s36);         v(s48);
    END             END             END             END
  prog5:          prog6           prog7           prog8
    BEGIN           BEGIN           BEGIN           BEGIN
    P(s15);         p(s36);         P(s57);         P(s48);
    P(s25);         do work;        Do work;        P(s68);
    Do work;        v(s68);         V(S78);         P(s78);
    V(57);          END             END             do work;
    END                                             END
COEND
END
```

思考(北邮2000):

一组合作进程，执行顺序如图所示，请用P,V操作实现各个进程之间的同步关系。



此类问题相对易于处理，出题的样式多样，有些题目不只是让考生实现其过程有时会让考生更正其中的错误或者判断其是否有死锁的可能并更正之。

二十 九阴真经之北航篇

1 智取考场

把学生和监考老师都看做进程，学生有N个人，教师1人，考场门口每次只能进出一个人，进考场原则是先来先进，当N个学生都进入考场后，教师才能发试卷。学生交卷后可以离开考场，教师要等收上来全部试卷并封装试卷后才能离开考场。

- 问共需设置几个进程？
- 使用P,V操作解决上述问题中的同步和互斥关系.

The P,V code Using Pascal

```
var mutex,Beginready,Testready,Endready:semaphore;
    //mutex用以标示教室门这个临界资源
    //beginready等待考生来全，标示考试开始
    mutex:=1;
    Beginready:=-(N-1);
    Testready:=0;
    Endready:=-(N-1);
 cobegin
   Procedure Student              Procedure Teacher
    P(mutex);                       P(mutex);
     Enter;                          Enter;
    v(mutex);                       v(mutex);
     Waiting;
    P(Beginready);
    ------------------------------------------------
                                    p(Beginready);
                                     Hand Out;
                                    v(Beginready);
    ------------------------------------------------
    p(Testready);
    v(Testready);
     答题;
     交卷;
     离开;
    v(Endready);
    ------------------------------------------------
                                   p(Endready);
                                    封卷离开;
```

2 读写者问题(2005)

我们将只读数据的进程称为“读者”进程，而写或者修改数据的进程称为“写者”进程，允许多个“读者”同时读数据，但不运行写者与其它读者或者写者进程同时访问数据。另

外，要保证：一旦有写者等待，新到达的读者必须等待，直到该写者完成数据访问为止，用P,V 操作实现读者，写者同步。

问题分析:

- 互斥资源：读写者问题，隐含一个互斥资源-读写的问件
- 互斥锁：读文件时不能写，写文件时不能读文件
- 读进程：允许多个文件读，读进程时>0时，锁定文件，读文件进程<1时，解锁；读进程数>0时，说明读进程拥有锁
- 写进程:拥有锁时写文件

The P,V code Using Pascal

解答:

增加一个信号量w: =1，用以在写进程到达时封锁后续进程

```
cobegin
procedure Reader                    procedur Writer
begin                                begin
 p(w);                               p(w);
 p(rmutex);                          p(wmutex);
 if rcount==0 then                     写数据;
   p(wmutex);                        v(wmutex);
 v(rmutex);                          v(w)
 v(w);                               end
 读数据;
 p(rmutex);
   rcount:=rcount-1;
  if rcount==0 then
   v(rmutex);
 v(rmutex);
 end
coend
```

3 吸烟者问题

这个题目很传统，和前面问题一致这里简单阐述一下。这里再给出一个简单解法。

- 三个吸烟者(A,B,C)和一个经销商(D)，三个吸烟者可以吸烟的条件不一样，具体看经销商往桌子上放的原料
- 每个吸烟者需要一个进程，分别和经销商进行同步
- 互斥资源：桌子
- A,B,C,D四个进程，A表示烟草拥有者，B是纸拥有者，C火柴拥有者，D经销商
- S实现互斥，表示桌子上是否放有东西
- Sad,Sbd,Scd分别表示进程AD,BD,CD之间的同步

The P,V code Using Pascal

```
cobegin
经销商              烟草拥有者        纸拥有者          火柴拥有者
begin               begin             begin             begin
p(s);               p(Sad);           p(Sbd);           p(Scd);
放原料;             取纸和火柴;       取烟草和火柴;     取纸和烟草;
if(纸和火柴)        v(s);             v(s);             v(s);
  v(Sad);             吸烟;           吸烟;             吸烟;
else                end               end               end
 if(烟草和火柴)
 v(Sbd);
 else
 v(Scd);
 end
coend
```

4 生产者-消费者扩展

有n+1个进程A1，A2，···,An和B:

- (1)A1,A2,···,An通过同一个缓冲池各自不断地向B发送消息，B不断地取消息，它必须取走发来的每个消息，刚开始时缓冲区为空，使用P,v操作实现之。
- (2)若缓冲区个数增至M个，试用P,V实现正确通讯

这个问题较为简单，请读者自行解决。这里给出参考答案。

The P,V code Using Pascal

```
var full,empty,mutex:semaphore;
    full=0;
    empty=1;
    mutex=1;
  cobegin
    procedure A_i(i=1,...,n)      procedure B:
     begin                         begin
     p(empty);                     P(full);
     p(mutex);                     p(mutex);
     put message to the buffer;    Get the message;
     v(mutex);                     v(mutex);
     v(full);                      v(empty);
     end                           end
```

5 阅览室问题

有一个阅览室，共有100个座位，读者进入时必须先在一张登记表上登记，该表为每一个座位列一表目，包括座号和读者姓名等，读者离开时要消掉登记的信息，试问;

- (1)为描述读者的动作，应编写几个程序，设置几个进程？

- (2)试用PV操作描述各个进程之间的同步互斥关系。

问题分析:

读者动作有两个，一个时填表进入阅览室，这时要考虑阅览室里是否有空位；一是读者阅读完毕，离开阅览室，这时的操作要考虑阅览室里是否有读者。读者在阅览室读书时，由于没有引起资源的变动，不算动作变化。

算法的信号量有三个：seats-表示阅览室时否有座位(初值为100)；readers-表示阅览室里的读者数，初值为0；用于互斥的mutex，初值为1。

The P,V code Using Pascal

```
var seats, raaders,mutex:semaphore;
    seats:=100;
    readers:=0;
    mutex:=1;
cobegin
  procedure Enter
   begin
   while TRUE
    begin
    p(seats);  //没有座位则离开
    p(mutex);  //进入临界区
     填写登记表;
     进入阅览室阅读;
    v(mutex);  //离开临界区
    v(readers);
    end
   end
 procedure Leave
   begin
    while TRUE
     begin
     p(readers);
     p(mutex);
     消掉登记;
     离开阅览室;
     v(mutex);
     v(seats);
     end
    end
 coend
```

思考:

某超市可同时供100人购物，入口处备有篮子，每个购物者可带一个篮子入超市购物，出口时结账并归还篮子(出入口仅容一人)。

6 P,V改错(2001)

设有两个优先级相同的进程P1，P2如下。令信号S1，S2的初值为0，已知z=2，试问P1，P2并发运行结束后x=？y=？z=？

```
进程P1          进程P2
y:=1;             x:=1;
y:=y+2;           x:=x+1;
V(S1);             P(S1);
z:=y+1;           x:=x+y;
P(S2);             V(S2);
y:=z+y;           z:=x+z;
```

受信号量S1和S2的控制，进程P1和P2中P,V操作的顺序应明确。但当进程P2执行v(S2)调用后，可能会产生这种情形，即P2中的语句“z:=x+z”可以在P1中的语句“y:=z+y”前面或后面执行，因而P1和P2并发运行结束后，有两种可能的结果。即:

x=5、y=12、z=9或x=5、y=7、z=9。

7 面包店(2001)

面包师有很多面包，由n个销售人员推销。每人顾客进店后先取一个号，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。

问题分析:

该题目和银行排队 问题一致的。解法一样！！

8 公交车问题(2002)

在一辆公共汽车上，司机和售票员各行其职，司机负责开车和到站停车；售票员负责售票和开、关门，当售票员关好车门后，司机才能继续开车行驶。试用P、V操作实现司机与售票员之间的同步。

9 P,V改错(2002)

下面是两个并发执行的进程。它们能正确运行吗？若不能请举例说明，并改正之:

```
parbegin
        var x:integer;
        process P1                process P2
        var y, z:integer;         var t, u:integer;
        begin                     begin
        x:=1;                     x:=0;
        y:=0;                     t:=0;
        if x⩾1 then y:=y+1;        if x⩽1 then t:=t+2;
        z:=y;                     u:=t;
        end                       end
parend
```

他们不能正确运行。因为题中的进程P1和P2之间有一个共享变量X，由于进程的并发执行，可能会产生与时间有关的错误。比如进程P1先于P2运行，程序执行完成后Z的值是1，u的值为2.但若进程P1 执行到赋值语句“X: =1”后，进程调度P2运行，此时X的值又被重新赋值为0，待进程P2运行结束后，进程P1运行，当两个进程都结束运行时，Z的值0，u的值2.若要使P1和P2能正确执行，必须设置一互斥信号量，以便对P1和P2中的临界区互斥使用。改正后的程序如下:

The P,V code Using Pascal

```
parbegin
var x:integer;
mutex:semaphore;
        process P1                  process P2
        var y, z:integer;           var t, u:integer;
        begin                       begin
        p(mutex);                   p(mutex);
        x:=1;                       x:=0;
        y:=0;                       t:=0;
        if x⩾1 then y:=y+1;          if x⩽1 then t:=t+2;
        v(mutex);                    v(mutex);
        z:=y;                       u:=t;
        end                         end
parend
```

第 四 章　福尔摩斯探案之网络搜捕

一　打印机问题

问题描述:

设系统中有5台类型相同的打印机，依次编号为1～5。又设系统中有n个使用打印机的进程，使用前申请，使用后释放。每个进程有一个进程标识，用于区别不同的进程。每个进程还有一个优先数，不同进程的优先数各异。当有多个进程同时申请时，按照进程优先数由高到低的次序实施分配。试用信号灯和PV操作实现对于打印机资源的管理，即要求编写如下函数和过程:

(1)函数require(pid，pri): 申请一台打印机。参数pid为进程标识，其值为1到n的整数; pri为进程优先数，其值为正整数; 函数返回值为所申请到打印机的编号，其值为1到5的整数;

(2)过程return(prnt): 释放一台打印机。参数prnt为所释放打印机的编号，其值为1到5的整数。

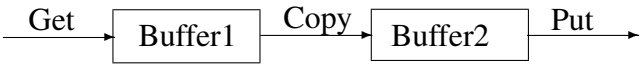
The P,V code Using Pascal

解:
```
#define N 5
 Int flag[N+1];//flag[0]表示可用打印机数,
 //flag表示第i号打印机的状态(1<=i<=N)，0表示占用，1表示空闲
 PCB *queue=NULL;//进程阻塞队列
 semaphore mutex_flag=1;//用于对flag数组的互斥操作
 semaphore mutex_queue=1;//用于对阻塞队列的互斥操作
 int require(int pid,int priority)
   {
   P(mutex_flag);
   if(flag[0]>0)
     {
       flag[0]--;
       for(int i=1;i<N+1;i++)
       if(flag= =1)
       {
       flag=0;
       break;
       }
     V(mutex_flag);
     return i;
     }
   else
```

```
        {
        V(mutex_flag);
        p(mutex_queue);
        将进程pid按其优先数插入到等待队列queue中;
        V(mutex_queue);
        }
      }
       return(int print)
       {
        P(mutex_flag);
        if(queue==NULL)
        {
          flag[0]++;
          flag[print]=1;
          V(mutex_flag);
        }
        else
        {
          V(mutex_flag);
          p(mutex_queue);
          将print分配给queue队首进程;
          queue下移;
          V(mutex_queue);
        }
    }
```

思考1(北大1990，西北工大2000):

有三个进程PA、PB和PC用以解决打印问题：PA将文件记录从磁盘读入主存的缓冲区1，每执行一次读一个记录；PB将缓冲区1的内容复制到缓冲区2，每执行一次复制一个记录；PC将缓冲区2的内容打印出来，每执行一次打印一次记录。缓冲区的大小和一个记录大小一样。请用P、V操作来保证文件的正确打印。



问题简评:

本题的第一部分是检查考生对基本概念的记忆与理解，是解答本题的基础。题目的第二部分时一个典型的生产者　消费者问题，其中的难点在于PB既是生产者优势消费者，处理不好可能造成同步错误或死锁。相对于上面问题简单很多，请大家自行完成！(以下代码仅供参考)

The P,V code Using Pascal

```
var mutex1,mutex2,avail1,full1,avial2,full2:semaphore;
//程序中mutex1和mutex2是两个公用信号量，同于控制进程对
//缓冲区1和缓冲区2这两个临界资源的访问的互斥。
```

```
//Avial1、full1、avial2和full2分别对应于连个缓冲区.
Mutex1:=1;
Mutex2:=1;
Avial1:=1;
Avial2:=1;
Full1:=0;
Full2:=0;
COBEGIN
 procedure PA:          procedure PB:          procedure PC:
 BIGIN                  BIGIN                  BIGIN
  L1:read from disk;    L2:P(full1);           L3: P(full2);
  P(avial1);            P(mutex1);             P(mutex2);
  P(mutex1);            Get from buffer 1;     Get from buffer 2;
  Put to buffer 1;      V(avial1);             V(avial2);
  V(full1);             V(mutex1);             V(mutex2);
  V(mutex1);            P(avial2);             Goto L3;
  Goto L1;              P(mutex2);             END
 END                    Put to buffer 2;
                        V(full2);
                        V(mutex2);
                        Goto L2;
                        END
COEND
```

思考2:

有三个并发进程: R 负责从输入设备读入信息块，M 负责对信息块加工处理; P 负责打印输出信息块。今提供;

l)一个缓冲区，可放置K 个信息块;

2)二个缓冲区，每个可放置K 个信息块;

试用信号量和P 、V 操作写出三个进程正确工作的流程。

The P,V code Using Pascal

```
var A , B :array [0,...,k-l] of item ;
sPut1 : semaphore:=k;
SPut2:  semaPhore:=k;
sget1 : semaPhore : = 0 ;
sget2 : semaphore : = 0 ;
put1 : integer : =O ;
put2: integer : = 0 ;
get1 : integer : =O ;
get2 : integer : = O ;
cobegin
process reader ;                processsn manager;
begin                           begin
Ll : read message into x;       L2 : P(sgetl);
```

```
P(SPut1);                          x : = A[get1];
A[put1]:=x ;                       get1 : (get1+1)mod k;
Put1:=(put1+1) mod k;              V(sput1);
V(sget1);                          print the message into x;
Goto L1;                           P(sput2);
                                   Put2:=(put2+1) mod k;
                                   V(sget2);
                                   Goto L2;
                                   End;
 process Writer;
 begin
 L3:p(sget2);
 x:=b[get2];
 gert2:=(get2+1)mod K;
 v(sput2);
 goto L3;
 end

Coend
```

思考2(北理2002):

有一个文件F,有A,B两组进程共享这个文件，同组的进程可以同时读文件F,但当有A组(或B组)的进程在读文件F时就不允许B 组(或A组) 的进程读,

解:定义两个计数器C1,C2,分别记录A组和B组中读文件的进程数,三个信号量S1,S2,SAB,其中S1用于通知A组进程已经有B组进程在读文件F了,S2用于通知B进程已经有A进程在读文件F了,SAB用于实现对共享变量C1和C2以及临界区的互斥访问.

The P,V code Using Pascal

```
var S1,S2,SAB:semaphore;
    S1=S2=SAB=1;
 C1,C2:integer;
 C1=C2=0;
cobein
 process A-i(i=1,2)
    begin
    repeat
    P(SAB);
    C1 = C1+1;
    if(C1=1) then P(S2);
    V(SAB);
    读文件F;
    P(SAB)
    C1 = C1-1;
    if(C1==0)V(S2)
    V(SAB)
```

```
    until false
    end
 process B-i(i=1,2)
    begin
    repeat
    P(SAB);
    C2 = C2+1;
    if(C2=1) then P(S1);
    V(SAB);
    读文件F;
    P(SAB)
    C2 = C2-1;
    if(C2==0)V(S1)
    V(SAB)
    until false
    end
coend
```

思考3:

get进程读数据到buffer1里，然后进程copy从buffer1里的数据复制到buffer2里，再由put进程取出buffer2里的数据去进行打印.

分析:这是两个阶段的生产-消费问题.第一阶段的生产者和消费者是get和copy,第二阶段的生产者和消费者是copy和put.为简单计,假设buffer1,buffer2 都是单缓冲区,因此只要设4个信号量empty1,full1,empty2,full2,就可以了.

The P,V code Using Pascal

```
buffer:integer
var empty1,empty2,full1,full2,:semaphore;
    empty1=1,empty2=1,full1=0,full2=2;
cobegin
  process Get
    begin
    repeat
    读数据
    p(empty1);
    把数据放到buffer1里
    v(full1);
    until false
    end
  process Copy
    begin
    repeat
    p(full1)
    从buffer1里读出数据
    v(empty1);
```

```
    p(empty2)
    把数据放到buffer2里
    v(full2)
    until false
    end
  process Put
    begin
    repeat
    p(empty2)
    从buffer2里读出数据
    v(full2);
    打印数据
    until false
    end
coend
```

思考4:

输入进程输入数据到缓冲区buffer1中，计算进程从buffer1中读出数据进行计算,并把结果送入buffer2,然后打印进程从buffer2中读出结果进行打印.假设缓冲区大小分别为n1和n2.

The P,V code Using Pascal

```
buffer:integer
empty1,empty2,full1,full2,mutex1,mutex2:semaphore=n1,n2,0,0,1,1;
cobegin
  process Input
    begin
    repeat
    输入数据
    p(empty1);
    p(mutex1);
    把数据放到buffer1里
    v(mutex1);
    v(full1);
    until false
    end
  process Compute
    begin
    repeat
    p(full1)
    p(mutex1);
    从buffer1里读出数据
    v(mutex1);
    v(empty1);
    p(empty2)
```

```
    p(mutex2);
    把数据放到buffer2里
    v(mutex2);
    v(full2);
    until false
    end
  process Print
    begin
    repeat
    p(empty2)
    p(mutex2);
    从buffer2里读出数据
    v(full2);
    v(full2);
    打印数据
    until false
    end
coend
```

二 批处理系统问题

设某个批处理系统中，有三个进程：卡片输入进程、作业调度进程、作业控制进程。他们之间的合作关系是：

- 只要卡片输入机上有作业信息输入，进程把作业逐个输入至输出井并为每个作业建立一个JCB块并把它插入至后备作业队列(JCB链表)中。
- 当内存中无作业运行时，作业调度进程从JCB中选一个作业，把该作业装入内存。
- 作业控制进程负责处理已调入内存的作业。

(1)P,V写出输入和调度进程的同步。
(2)用消息缓冲痛惜，写出调度进程与作业控制进程间的同步算法。

The P,V code Using Pascal

```
procedure 输入:
 begin
 L1:如果有卡片 then L2
    等待卡片;
 L2:把作业输入至输出井并建立JCB块;
 p(s);
 把JCB插入链中;
 v(mutex);
 v(s);
 Goto L1;
 end
procedure 调度:
 begin
```

```
 M:P(s);
 p(mutex);
 查JCB;
 v(s);
 send();//向控制进程发信息
 receive();//接受信息
 Goto M;
 end
procedure 作业控制:
 begin
 N:receive();
  处理;
 send();//向调度发信息
 Goto N;
 end
```

三 桔子汁生产线问题

问题描述:

现有三个生产者P1 、P2 、P3 , 他们都要生产水, 每个生产者都已分别购得两种不同原料, 待购得第三种原料后就可配制成桔子水, 装瓶出售。有一供应商能源源不断地供应糖、水、桔子精, 但每次只拿出一种原料放入容器中供给生产者。当容器中有原料时需要该原料的生产者可取走, 当容器空时供应商又可放入一种原料。假定: 生产者P1已购得糖和水; 生产者P2 已购得水和桔子精; 生产者P3 已购得糖和桔子精; 试用: 信号量与P 、v 操作, 写出供应商和三个生产者之间能正确同步的程序.

问题分析:

该问题与吸烟者问题异曲同工, 解决方法一致。请读者自行完成。

思考:

一个快餐厅有4类职员:(1)领班: 接受顾客点菜;(2)厨师: 准备顾客的饭菜;(3)打包工: 将做好的饭菜打包;(4)出纳员: 收款并提交食品。每个职员可被看作一个进程, 试用一种同步机制写出能让四类职员正确并发运行的程序。

The P,V code Using Pascal

```
var   S1,S2,S3,S4:semaphore;
      S1:=1; S2:=S3:=S4:=0;
cobegin
 process P1           process P2
  begin                begin
  repeat               repeat
   Consumer come;      p(S2);
  P(S1);               准备顾客的饭菜;
   Serve consumer;     V(S3);
  V(S2);              until false;
  untile false;       end
```

```
  end
 process P3            process P4
  begin                begin
  repeat               repeat
  P(S3);               p(S4);
  将做好的饭菜打包;      收款并提交食品;
  V(S4);               v(S1);
  untile false;        until false;
  end                  end
coend
```

四 保管员问题

问题描述:

有一材料保管员，他保管纸和笔若干。有A、B两组学生，A组学生每人都备有纸，B组学生每人都备有笔. 任一学生只要能得到其他一种材料就可以写信。有一个可以放一张纸或一支笔的小盒，当小盒中无物品时，保管员就可任意放一张纸或一支笔供学生取用，每次允许一个学生从中取出自己所需的材料，当学生从盒中取走材料后允许保管员再存放一件材料，请用信号量与P、v操作，

问题分析:

该问题同上，不再赘述。

The P,V code Using Pascal

```
var
  s, Sa.Sb, mutexa, mutexb: semaphore;
  s: = mutexa : =mutexb: = 1;
  sa: = sb: = 0;
  box: (PaPer, Pen);
  cobegin
    process 保管员
      begin
        repeat
        P(S);
        take a material intobox ;
        if (box)=Paper then V(Sa);
        else V(Sb);
        untile false ;
      end

    Process A组学生
      begin
        repeat
        P(Sa);
        P(mutexa);
```

```
        take the pen from box ;
        V(mutexa);
        V(S);
        write a letter;
        untile false ;
      end

    Process B组学生
      begin
        repeat
        P(Sb);
        P(mutexb);
        take the paper from box ;
        V(mutexb);
        V(S);
        wnte a letter ;
        untile false ;
     end
  Coend.
```

五 招聘问题

问题描述:

现有100名毕业生去甲、乙两公司求职，两公司合用一间接待室，其中甲公司招收10人，乙公司准备招收10人，招完为止。两公司各有一位人事主管在接待毕业生，每位人事主管每次只可接待一人，其他毕业生在接待室外排成一个队伍等待。试用信号量和P、v操作实现人员招聘过程。

问题分析:

由于毕业生仅排成一队，故用如图的一个队列数据结构表示。在队列中不含甲、乙公司

A	B	A	A	B	Sm A	B	Sn B	A	···				

都接待过的毕业生和已被录用的毕业生。只含标识为A（被甲接待过）或只含标识为B（被乙接待过）及无标识的毕业生队列。此外，sm 和Sn 分别为队列中甲、乙正在面试的毕业生i (i = 1 , 2 ， …， 100 ）标识、即此刻另一方不得面试该毕业生i 。K1和K2 为甲、乙所录取的毕业生数，C1 、C2 为互斥信号量。注意，如果甲录取了一人，且该生没有被乙面试的话，则乙面试的毕业生将减1 。办法是: 如果甲录取了一人，且该生没有被乙面试可把乙的面试计数器C2加1（相当于乙已面试了他），从而，保证乙面试的人数值为100 。反之对甲亦然。

The P,V code Using Pascal

```
var Sa,Sb,mutex:semaphore;
Sa:=Sb:=mnutex:=1;
```

```
C1,C2,K1,K2: integer;
C1:=C2:=K1:=K2:=0;
cobegin
 process 甲公司
  begin
  L1: P(mutex);
  P(Sa);
  C1:==C1+1 ;
  V(Sa);
  If C1≤100 then
  {
   从标识为B 且不为Sn 或
   无标识的毕业生队列中选
   第i 个学生，将学生i 标
   识为A 和Sm
   }
  V(mutex) ;
  面试;
  P(mutex);
  if 合格then
  {
   K1:=K1+1;
   if 学生i 的标识不含B then
  {
   P (Sb);
   C2:=C2+1;
   V(Sb);
   将学生i 从队列摘除;
  }
  else 将学生i 从队列摘除;
  else if 学生i 的标识含B then
           将学生i 从队列摘除;
       else
           取消学生i 的Sm 标识;
  V(mutex);
  If(K1<10)&(C2<100) then
  goto L1;
 end
 process 乙公司
  begin
  L2:P(mutex);
  P(Sb);
  C2:=C2+1;
  V(Sb);
  if C2≤100 then
```

```
  从标识为A 且不为sm 或无标识的
  毕业生队列中选第i个学生将学生i
  标识为B和Sn
  V(mutex);
  面试;
  P(mutex);
  if 合格then
  {
   K2:=K2+1;
   if 学生i 的标识不含A then
   {
   P(Sa)
   C1:=C1+1;
   V(Sa);
   将学生i 从队列摘除;
   }
  else 将学生i 从队列摘除;
  else if 学生i 的标识含A then
                  将学生i 从队列摘除;
       else
          取消学生i 的Sn 标识;
  V(mutex);
  if(K2<10)&(c1<100)then
  goto L2;
  }
  end
coend .
```

六 博物馆-公园问题

问题描述:

Jurassic公园有一个恐龙博物馆和一个花园，有m 个旅客租卫辆车，每辆车仅能乘一个一旅客。旅客在博物馆逛了一会，然后，排队乘坐旅行车，挡一辆车可用喊飞它载入一个旅客，再绕花园行驶任意长的时间。若n 辆车都已被旅客乘坐游玩，则想坐车的旅客需要等待。如果一辆车已经空闲，但没有游玩的旅客了，那么，车辆要等待。试用信号量和P 、V 操作同步m 个旅客和n 辆车子。

问题分析:

这是一个汇合机制，有两类进程: 顾客进程和车辆进程，需要进行汇合、即顾客要坐进车辆后才能游玩，开始时让车辆进程进入等待状态

The P,V code Using Pascal

解答:

```
var sc1 , sck , sc , Kx,xc , mutex : semaphore ;
sck:=kx:=sc:=xc:=0;
```

```
sc1:=n ; mutex : = 1 ;
sharearea : 一个登记车辆被服务乘客信息的共享区;
cobegin
     process 顾客i ( i = 1 , 2 , … )
     begin
      P (sc1) ;          /*车辆最大数量信号量
      P (mutex) ;        /*封锁共享区，互斥操作
      在共享区sharearea登记被服务的顾客的信息:
                          起始和到达地点，行驶时间
      V (sck) ;          /* 释放一辆车 ,即顾客找到一辆空车
      P(Kx);             /* 待游玩结束之后，顾客等待下车
      V(sc1) ;           /*空车辆数加1
      End
      Process 车辆j(j=1,2,3…)
      Begin
      L:P(sck);          /*车辆等待有顾客来使用
        在共享区sharearea登记一辆车被使用，并与顾客进程汇合;
      V(mutex);          /*这时可开放共享区，让另一顾客雇车
      V(kx);             /*允许顾客用此车辆
      车辆载着顾客开行到目的地;
      V(xc);             /*允许顾客下车
      Goto L;
      End
coend
```

七 生产流水线问题

问题描述:

设自行车生产线上有一只箱子，其中有N 个位置(N ⩾3)，每个位置可存放一个车架或一个车轮; 又设有三个工人，其活动分别为:

```
工人1活动:        工人2活动:            工人3活动:
do{               do{                   do{
  加工一个车架;      加工一个车轮;          箱中取一车架 ;
  车架放入箱中;      车轮放入箱中;          箱中取二车轮 ;
} while (1)       } while (1)           组装为一台车 ;
                                        } while (1)
```

试分别用信号灯与PV 操作实现三个工人的合作，要求解中不含死锁。

问题分析:

用信号灯与PV 操作实现三个工人的合作首先不考虑死锁问题，工人1与工人3、工人2与工人3构成生产者与消费者关系，这两对生产/消费关系通过共同的缓冲区相联系。从资源的角度来看，箱字中的空位置相当于工人1和工人2的资源，而车架和车轮相当于工人3的资源。定义三个信号灯如下:

The P,V code Using Pascal

```
semaphore empty=N;
semaphore wheel=0;
semaphore frame=0;
三位工人的活动分别为:
procedure 工人1:
  do{
  加工一个车架 ;
  P(empty);
  车架放入箱中 ;
  V(frame);
} while (1)

procedure 工人 2:
  do{
  加工一个车轮 ;
  P(empty);
  车轮放入箱中 ;
  V(wheel);
  } while (1)

procedure 工人3:
  do{
  P(frame);
  箱中取一车架 ;
  V(empty);
  P(wheel);
  P(wheel);
  箱中取二车轮;
  V(empty);
  V(empty);
  组装为一台车 ;
  } while (1)
```

分析上述解法易见，当工人1推进速度较快时，箱中空位置可能完全被车架占满或只留有一个存放车轮的位置，而当此时工人3同时取2个车轮时将无法得到，而工人2又无法将新加工的车轮放入箱中；当工人2推进速度较快时，箱中空位置可能完全被车轮占满，而当此时工人3同取车架时将无法得到，而工人1又无法将新加工的车架放入箱中。上述两种情况都意味着死锁。为防止死锁的发生，箱中车架的数量不可超过N-2，车轮的数量不可超过N-1，这些限制可以用两个信号灯来表达。如此，可以给出不含死锁的完整解法如下:

The P,V code Using Pascal

```
semaphore s1=N-2;
semaphore s2=N-1;
  procedure 工人 1:
  do {
    加工一个车架 ;
    P(s1);
    P(empty);
    车架放入箱中 ;
    V(frame);
    } while (1)

  procedure 工人2:
    do{
    加工一个车轮 ;
      p(s2);
      P(empty);
    车轮放入箱中 ;
      V(wheel);
      } while (1)

  procedure 工人 3 :
     do {
    P(frame);
      箱中取一车架 ;
      V(empty);
    V(s1);
      P(wheel);
      P(wheel);
       箱中取二车轮 ;
      V(empty);
      V(empty);
      V(s2);
      V(s2);
      组装为一台车 ;
      } while (1)
```

详细描述还应考虑对箱子单元的描述以及访问互斥问题。建议车架放在箱子的一端，车轮放在箱子的另一端，车架与车轮都采用后进先出的管理方式。

The P,V code Using Pascal

```
Semaphore s1=N-2, s2=N-1, mutex=1;
int in1=0, in2=N-1;
int buf[N];
procedure 工人1:
    do {
    加工一个车架;
    P(s1);
    P(empty);
    P(mutex);
    Buf[in1]=车架;
    in1=in1+1;
    V(mutex);
    V(frame);
    } while (1)

procedure 工人2:
    do{
    加工一个车轮;
    P(s2);
    P(empty);
    P(mutex);
    Buf[in2]=车轮;
    in2=in2-1;
    V(mutex);
    V(wheel);
    } while (1)

procedure 工人 3 :
    do{
    P(frame);
    P(mutex);
    Temp1=Buf[in1-1];
    in1=in1-1;
    V(mutex);
    V(empty);
    V(s1);
    P(wheel);
    P(wheel);
    P(mutex);
    Temp2=Buf[in2+1];
    in2=in2+1;
    Temp3=Buf[in2+1];
    In2=in2+1;
    V(mutex);
    V(empty);
    V(empty);
    V(s2);
    V(s2);
    组装为一台车 ;
    } while(1)
```

八 知错能改

问题描述:

进程p0,p1共享变量flag,turn;他们进入临界区的算法如下:

```
var flag:array[0..1] of boolean;//初值为false
turn:01
process i (0或1)
   while true
   do begin
    flag[i] =true;
    while turn!=i
    do begin
    while flag[j]==false
    do skip;//skip为空语句
```

```
    turn = i
    end
    临界区;
    flag[i] = false;
    出临界区;
    end
```

该算法能否正确地实现互斥?若不能,应该如何修改(假设flag,turn单元内容的修改和访问是互斥的).

问题分析:

不能正确实现互斥.考虑如下情况:process0先执行到flag[0] =true,process1开始执行,进入内循环时,将turn设置为1;此时进程调度转到process0, process0可以进入内循环,由于flag[1]的值为true,所以process0再次将turn的值设置为0,重复上述操作,两个进程谁也不能进入临界区.

The P,V code Using Pascal

```
var flag:array[0..1] of boolean;//初值为false
    turn:0 1
 cobegin
    process 0
     while true
     do begin
     flag[0] =true;
     turn = 1
     while flag[1]==true and turn = 1
     do skip;//skip为空语句
     临界区;
     flag[0] = false;
     出临界区;
     end
    process 1
     while true
     do begin
     flag[1] =true;
     turn = 0
     while flag[0]==true and turn = 0
     do skip;//skip为空语句
     临界区;
     flag[1] = false;
     出临界区;
     end
 coend
```

容易证明这种方法保证了互斥,对于进程0,一旦它设置flag[0]为true,进程1就不能进入其临界段.若进程1已经在其临界段中，那么flag [1]=true并且进程0被阻塞进入临

界段.另一方面,防止了相互阻塞,假设进程0阻塞于while循环,这意味着flag[1]为true,而且turn=1,当flag[1]为false或turn为0时,进程0就可进入自己的临界段了.

第 五 章　独孤九剑之一剑定乾坤

章节要点

- 试题类型总结(来自各高校研究生试题)
- 主要概念归纳(研究生考试中常考概念)
- 北航试题探析(探析北航各科题目)

一　试题类型总结

作为知识学习来说，操作系统内容繁杂，它整合了组成原理、数据结构以及更为复杂的算法。它也是一门实践性很强的课程。另外知识点理解难度比较大，请大家不要仅仅拘泥于考试。

研究生入学考试作为国内最为严谨的考试，计算机专业课很多高校都会在初试中考查。但是操作系统的出题题型比较单一，考查的重点在基本概念和算法。题目难度不大。它的难度和个人的勤奋程度是成反比，不存在任何技巧，只是考查大家对算法的熟知程度和识记程度。所以应对操作系统考试还是比较容易的。下面简述一下操作系统常考到题目类型:

1　名词解释

该题目类型很常见，像华中科技大学、上海交大、清华、北航、北邮、南航等等都有出过题，但是大家留心会发现真正作为考试来说名词解释很少，而且重复率很高。在常考概念里我都有列举，但是难免挂一漏万。我主要依据北航、清华、上海交大的题目把常考到的概念总结了一下。

应对名词解释主要两个方法:

(1)识记概念，很多同学不愿意记忆。到了考场很多概念是只可意会，不能用准确语言描述。这样丢分太可惜了。这个部分没有技巧主要是记忆。

(2)上文讲到该部分重复率很高，所以要自己查找历年试题总结出来重复率高的概念重点记忆。注意: 要自己完全能默写出来，不要眼高手低。

2　填空题

该题型比较少见了，因为也是考查基本概念，但是范围较名词解释广了一些，难度比名词解释小。现在，多数高校已经不再考查该类题型。大家依据自己报考学校来确定复习目标。应对填空题目，需要较为广的知识，甚至很细微的地方都可以考查。这是名词解释考查不到的。

3　判断题

作为判断题目，很多高校试卷上依然较多出现。这类题目分数不多，难度不大。基本上也是基本概念理解的考查，而不是概念考查了。有时候可以依据自己计算机基本知

识猜出来。所以比较好应对。

4 P,V题

有些书也称为算法题，这类题目比较难出题，一般很多年才会出来一些较为新颖的题目，有些创新根本就是换汤不换药。考查大家的理解能力。而且一般难度比较大。可以说是操作系统题目的压轴题或者说是重头戏。有时候会让大家用管程来实现，这类题目不多。一般你可以使用P,V操作实现，管程实现也是很简单。只是需要一点转化。鉴于篇幅，这里不再赘述!

只要本文题目可以自己做出来或者参考答案自己可以理解，这部分要花较多时间来复习，P,V操作题目也可以应对自如。

5 计算题

该题型主要考查基本算法，整体难度不大。主要涉及内存管理、作业调度、进程调度、死锁(银行家算法)等等算法集中的知识点。这部分分数也较多，希望大家自己多加练习。各高校知识考查主要集中在一下三部分: 内存分配、作业调度、银行家算法。

应对策略:

(1)要熟练基本算法。如内存分配算法、银行家算法、几个常见的调度算法、响应比等等，这个部分需要一定量的练习题，在各自大学图书馆的书籍中都有典型题目。大家自己练习。

(2)总结规律。因为是算法题目，所以规律性很强。这个需要自己去认真总结。

6 证明题

这类题目比较少，实际上是算法题目一个衍生品。只要算法题可以搞定，这个题目肯定没有问题。

二 常考概念归纳

临界区
重定位
静态重定位
动态重定位
操作系统
操作系统的基本特征
系统调用
死锁
死锁的必要条件
死锁原因
信号量信号量的物理意义
P,V操作
进程进基本特征
进程控制块组成(PCB)
进程上下文
进程状态
进程与程序的区别联系
线程为什么引入线程

文件文件控制块组成(FCB)
文件系统
作业JCB
Spooling技术
虚拟存储技术通道
目录
中断
RAID
交换技术
原语
工作集