***PV原语***

PV原语是操作系统中非常最重要的一个概念,通过操作信号量来处理进程间的同步与互斥的问题。其核心就是一段不可分割不可中断的程序。

**P原语**：P是荷兰语Proberen（测试）的首字母。为阻塞原语，负责把当前进程由运行状态转换为阻塞状态，直到另外一个进程唤醒它。操作为：申请一个空闲资源（把信号量减1），若成功，则退出；若失败，则该进程被阻塞；

**V原语**：V是荷兰语Verhogen（增加）的首字母。为唤醒原语，负责把一个被阻塞的进程唤醒，它有一个参数表，存放着等待被唤醒的进程信息。操作为：释放一个被占用的资源（把信号量加1），如果发现有被阻塞的进程，则选择一个唤醒之。

**对信号量的操作可以分为三种情况**

　　具体PV原语对信号量的操作可以分为三种情况：

**1）**把信号量视为一个加锁标志位，实现对一个共享变量的互斥访问。

　　实现过程：

　　P(mutex); // mutex的初始值为1 访问该共享数据;

　　V(mutex);

　　非临界区

**2）**把信号量视为是某种类型的共享资源的剩余个数，实现对一类共享资源的访问。

　　实现过程：

　　P(resource); // resource的初始值为该资源的个数N 使用该资源；

　　V(resource); 非临界区

**3）**把信号量作为进程间的同步工具

　　实现过程：

　　临界区C1；

　　P(S);

　　V(S);

临界区C2；

PV原语的题型大致分为以下七类：

1. ***生产者-消费者问题和进程同步问题【***莫静雯】
2. ***哲学家问题和安全岛问题***【代猛】
3. ***排队问题和吸烟者问题***【郭婷婷】
4. ***读者-写者问题***【毛盾】

**生产者—消费者问题­** 莫静雯

**共性：**

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。

简单说来，即满足以下三点：

1. 有进有出；
2. 消费、生产程序并发执行；
3. 二进程是同步的，即不允许一个消费者进程到一个空缓冲区去取产品，也不允许生产者进程向一个已经装满产品且尚未被取走的缓冲区中投放产品！

**解题方法步骤：**

**一**、互斥变量mutex是必不可少的，保证各进程的互斥，亦即保护临界区；

二、根据缓冲区的个数设立信号量empty、full，一般来讲，我们先设立empty值为缓冲区个数或者一个缓冲区内可允许存放的物品总数（根据题意而定），full值为0。我们需要知道的是empty、full信号量的意义是用于在消费者、生产者在进入临界区时作检测，亦即通过P操作来看自己现在是否能进入缓冲区。

**三、**生产者进入临界区时要先P(empty)，再P(mutex)，出临界区时要执行V(mutex)，再V(full)；消费者进入临界区时要先P(full)，再P(mutex)，出临界区时要执行V(mutex)，再V(empty)。**（注意P一定要先对资源信号量empty进行操作，然后再对互斥信号量mutex进行操作，否则可能引起进程死锁）**

四、进入缓冲区后进程做的事变根据不同的题而定了。

生产者—消费者问题的大体答题方向便是如上所述，当然，根据不同的题目的要求，便需要在这个答题框架的基础上加工完善，一下是一些例题。在例题中，符合以上框架的部分都被加粗了，方便观看。

**原始问题：**

假设“生产者”进程不断向共享缓冲区写入数据(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。

**分析：**这个原始问题基本完全符合如上框架，处理的关键点是将若干缓冲池当做环形状态处理。因为有n个缓冲池，所以empty=n。

**解答：**

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

**Var muex, empty, full: semaphore:=1,n,0;**

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

**扩展一：仓库问题**

一、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-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.

**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的机会也多次；同理，每放入一次B,放入产品A的机会也多一次.

此题其实是没有消费者的生产者—消费者问题，只是处理生产者之间的同步问题。其实这里sa相当于A物品的empty信号量，而sb相当于B物品的empty信号量，只是当他们出临界区时需要进行V操作的不是full信号量，而是针对于两种物品的牵扯关系而执行的对方empty加一操作。

**解答：**

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

**二、**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-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两种物品的入库出库过程。

**分析：**此题和上题相比，是出现了消费者，是一个完整的生产者—消费者问题，在满足了如上框架的基础上增加了A、B货物的制约关系。因为有A、B两种物品，两种物品进出仓库的时候都需要检验能否进入临界区，所以需要设立两个empty，两个full，且empty分别是他们所能允许进入仓库的最大值。但是缓冲区是只有一个的，所以互斥信号量mutex只需要一个。

**解答：**

semaphore  **mutex=1,a,empty1=m,b,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);**

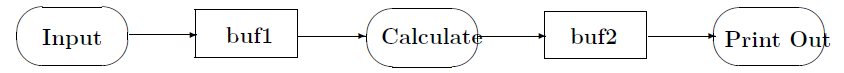
**v(empty2);**

end

end

**扩展二：**

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。



**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。

**解答：**

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

**扩展三：**

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。

**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。

在本题中，应设置三个信号量empty、So、Sa，信号量empty表示盘子是否为空，其初值为l；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。

这一题和框架也是可以说完全符合的，我们可以看作是有两个盘子，一个装orange，一个装apple，那么So可看做full\_orange，而Sa可看做是full\_apple.

**解答：**

**empty＝1;**

Sa＝0;

So＝0;

**mutex=1;**

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(empty);**

**P(mutex);**

将水果放入盘中;

if（放入的是桔子）

**V(mutex);**

V(So);

Else

**V(mutex);**

V(Sa);

end

end

Procedure son:

begin

while(TRUE)

begin

P(So);

**P(mutex);**

从盘中取出桔子;

**V(mutex);**

**V(empty);**

吃桔子;

end

end

Procedure daughter:

begin

while(TRUE)

begin

P(Sa);

**P(mutex);**

从盘中取出苹果;

**V(mutex);**

**V(empty);**

吃苹果;

end

end

**扩展四：**

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。

**分析：**

我们可以发现这个题是完满上诉三个条件的，由此可知它可归为一个消费 者生产者问题，那么，需要的进程便是两个，生产者进程和消费者进程，在此题中便是Enter和Leave进程。如何描述进程，直接套用答题框架，只是empty和full两个资源变量的名字应该改一下。

**解答：**

var seats, readers, mutex : semaphore;

seats:=100; (相当于empty)

readers:=0; (相当于full)

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

**进程同步问题** 莫静雯

**共性：**涉及到进程顺序执行问题，即进程同步问题。

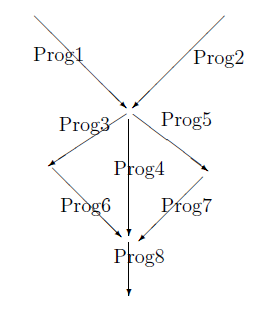
简单说来，便是涉及到你先我后，你干完什么我才能继续完成我的工作的 问题，且一般不涉及对临界资源的访问。

**解题方法：**

1. 分析有几个同步进程，它们分别是什么。
2. 针对每一个同步进程设置一个信号量，且初始值都为0。由于此类问题一般不涉及对临界资源的访问，所以不需要互斥信号量mutex。
3. 对于一个进程来讲，若另外一个进程的哪个步骤执行完才能执行我，则需要P一下对应信号量确保我能执行；若另外一个进程的执行需要我的某个步骤的执行完，则我执行完后需要V一下对应信号量。

**问题一：**

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。



**分析：**这一题的顺序制约关系如上图所示，共有11个同步关系。为每个同步

**解答：**

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; P(S13); P(S14);

V(s13); V(s23); P(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

**问题二：**

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。

**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。

因此，顺序制约问题是1、售票员关了车门车子才能启动；2、车子停了售票员才能开车门。因此，设置两个信号量Start，Open分别表示现在司机能不能开车和售票员能不能开车门。

**解答：**

var Start, Open : semaphore ;

Start=0； Open=0；

cobegin

Procedure driver Procedure Conductor

Begin begin

while TRUE while TRUE

begin begin

P(Start); 关车门；

Start; V(Start);

Driving; 售票；

Stop; P(Open);

V(Open); 开车门；

end 上下乘客；

end end

coend

**问题三：**

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：

1. 每两个学生组成一组，各占一台及其协同完成上机实习；

2. 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；

3．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

**分析：**

这一题的顺序制约关系是1、要有两个人到了才能一起进机房 2、要学生完成作业了老师才能检查。所以我们设置Enter信号量控制什么时候能进机房，设置Finish信号量控制什么时候老师能检查。但是需要注意的是，两个学生都到了才能Enter，那么我们就多用一个进程Guard来控制Enter。

**解答：**

var stu, computer, enter, finish, test: semaphore;

stu:=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(enter); Test the work; p(stu);

Start computer; v(test); Enter;

v(finish); v(test); v(enter);

v(test); end end

v(computer);

end

coend

老师，生产者—消费者主要就是这些内容，其实自己整理整理，写写算算之后发现也没有一开始想的那么难了！果然还是实践出真知啊！另外，以上很多自己写的文字都还没有完善，可能会有些概念性的错误或者是冗余拖沓、不知所云的内容，望老师加以指导修正~

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

***代猛***

# 问题特征

主要特征包括以下几点：

1. 临界资源的数目多于两个。

2. 会出现一个进程对多个临界资源进行访问。

3．各个进程之间没有确定的前驱后继关系。

4. 各个进程操作相同。

# 解题思路

思路1:

通过给进程赋予不同的特征，将其分类，是不同类型的进程的动作不同，从而实现对临界资源的有序访问。

思路2：

利用and型信号量进行解决。即对于进程所需的临界资源，要么全部分配给他，要么一个都不给。

# 典型例题

原始问题

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；

有问题的解题方法：

semaphore Fork[i] := 1(i = 0, 1, 2, 3, 4)

begin

Thinking;

Being hungry;

P(Fork[i mod 5]);

p(Fork[(i + 1) mod 5]);

Eating

V(Fork[i mod 5]);

V(Fork[(i + 1) mod 5]);

end

解题方法一：

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

解题方法二：

semaphore c[0]~c[4], 初值均为1；

Integer i = 0, 1,..., 4;

procedure philosopher\_i

begin

Thinking;

Being hungry;

p(c[i + 1] mod 5, c[i]);

Eating;

v(c[i + 1] mod 5, c[i]);

end

**安全岛问题**🡪**过桥问题 代猛**

**问题特征：**

**一般会有类似于双向前进的字眼，如过桥，过路。涉及两类多个进程。对于临界资源的访问，一般不是一直互斥进行，会因为中间一个某个“安全岛”而将临界资源分为三部分。**

**解题思路：**

**思路一：依照路口顺序来设置信号量，如A安全岛问题。**

**思路二：根据路段来设置信号量，如过桥问题一。**

**A：安全岛问题：**

问题描述: 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n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n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.

原始解答： 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

注释：我认为信号量M不是必须的。因为P(N2T) 和P (T2N)已经保证了同一方向不会同时有两辆车在路中。信号量M的作用重复。

**B：过桥问题一：**

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。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。

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);

**C：过桥问题二：**

问题描述：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。

问题分析：设该桥为东西方向。则由东向西的进程与由西向东的进程互斥访问该独木桥。而同一走向的个进程之间，可以同时的访问独木桥。

解题思路:为由东向西的进程和由西向东的进程设置信号量S用来互斥的访问独木桥。设置整型变量rc1，rc2分别用来表示由东向西和由西向东的在独木桥上的人数。

解题方法：var S,S1,S2:semaphore;

rc1,rc2:integer;

S,S1,S2:=1;

rc1=rc2=0;

cobegin

procedure East2West\_i: begin

p(S1);

rc1:=rc1+1;

if rc1==1 then P(s);

v(S1);

过独木桥;

p(S1);

rc1:=rc1-1;

if rc1==0 then v(S);

v(S1);

end

procedure West 2East\_i: begin

p(S2);

rc2:=rc2+1;

if rc2==1 then P(s);

v(S2);

过独木桥;

p(S2);

rc2:=rc2-1;

if rc2==0 then v(S);

v(S2);

end

coend

**排队等候问题：银行排队**🡪**理发排队 郭婷婷**

**A:银行排队问题（面包店问题）**

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.

解题：

Begin //customer\_count起共享剩余资源统计

var mutex=1,customer\_count=0:semaphore; //其中mutex起加锁作用

cobegin

process customer

begin

repeat

取号码；

p(mutex);

进入队列；

v(mutex);

v(customer\_count); //共享资源++

end

process serversi(i=1,...,n)

begin

repeat

p(customer\_count); //共享资源总数判断并**- -**

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

**B: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。

问题分析：理发师睡觉与否、有没有等候理发的顾客、顾客到来时有没有空椅子等都需要设置相应的信号量。

解答：

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 //起加锁作用的信号量

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( ); //一个顾客坐下等理发

end

else

V(mutex); //人满了,走吧!

end

coend

**C:理发师问题拓展(Barber Problem)**

问题描述：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.

问题分析：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;

解答：

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

排队等候问题总结：

共性：涉及到等候问题、服务者服务问题 。因此过程中涉及到的信号量较多。

解决方法：

1. 银行排队问题较简单，只涉及到两个信号量：

对于顾客： 取号

P(mutex)

排队等候

V(mutex)  
V(customer数量统计变化)

对于银行服务人员：

P(customer数量统计)

P(mutex)

从队列中取号

V(mutex)

服务

过程中要注意mutex和customer数量统计 这两个信号量的先后顺序

1. 还没有整理好，这个问题【先问一下老师。自己的理解是否正确】

**吸烟者问题（Smoker Problem)** 郭婷婷

**A：吸烟者问题：（桔子汁生产线问题）**

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。

问题分析：1、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。

2、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。

3、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。

4、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。

5、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。

解答： 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

**B：吸烟者问题拓展:**

问题描述: 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。

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。

解答： 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); //只有有一个观众时才开始播放电影

v(s0);

看影片；

p(s0);

count0 = count0 -1;

if(count0=1) v(s); //当最后一个观众离开时主动结束电影的播放

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

吸烟者问题总结：

共性：吸烟者(电影院观众)均持有部分资源，要根据供应者提供的不同类型的资源，使不同的吸烟者(观众)进行吸烟(观看电影)活动。

解决方法：

1、吸烟者问题：

对于供应者：P(S)

取烟原料

If

依据烟原料的不同

V(不同的吸烟者)

Elseif

Else

对于吸烟者(以s1为例):

P(S1)

取烟原料

做香烟

V(S) ;

吸香烟

2、与吸烟者问题相比，电影院问题增加了count变量

对于影片播放(以videoshow0为例 )：

P(s0)

Count0=count0+1 //只有有一个观众时才开始播放电影

If(count0=1)

P(S)

V(s0)

看电影

Count0=count0-1;

if(count0=1) //当最后一个观众离开时主动结束电影的播放

V(S);

V(s0)

两者本质上是没有什么差别的。但要注意区分S和s0/s1/s2的位置

**读者—写者问题­**

该问题的两种情况：

1. 读优先（RF）：

有写需要的进程无条件地一律排在有读需要的进程之后进行（即没有进程请求读时才允许写）；

1. 写优先（WF）：

有读需要的进程无条件地一律排在有写需要的进程之后进行（即没有进程请求写时才允许读），但要求有一个同时允许的最大读者数；

无论以上哪种情况，均要保证：

1. 允许多个读进程可以同时读取；
2. 每次只允许一个进程进行写操作；
3. 如果有进程在写，则禁止任何进程读。

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

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：

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。

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

可以认为是一种允许多个进程同时写的，但不允许读写同时存在的扩展。

最后：有两个问题我们打算作为课堂讨论，因为我们几个商量后也觉得理得不是很清楚【来自老师给的PV文件】

十六真经之机房上机问题(北大1997)--【争议】

(二十) 智取考场【争议】

陈老师：1、您觉得我们是做成PPt讲好呢，还是word或者PDF?

2、您什么时候有空呢？郭婷婷的理发师问题还有点不清楚，想找您讨论一下。