1. 设有一个变化后的生产者——消费者问题：若干个进程共享使用N个缓冲区，生产者

进程每次生产一个数据放入一个缓冲区；而消费者进程则又分为两类：一类每次从缓冲区取用一个数据消费，另一类则当缓冲区有两个数据时才同时取用，请用信号量及信号量集机制写出进程的同步算法。（15分）

解 ：

var full, empty;

full = 0,empty = N;

var mutex;

mutex = 1;

begin

Cobegin

P1:

Repeat

P(empty)

放入数据

V(full)

P2:

Repeat

P(mutex)

P(full)

取出数据

V(empty)

V(mutex)

P3:

Repeat

P(mutex)

P(full)

P(full)

取出数据

V(empty)

V(empty)

V(mutex)

Coend

end

2. 设有一缓冲池P，其中含有8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P、V操作表示两进程之间的制约关系。

解：VAR mutex,empty,full:semaphore:=1,8,0;

begin

parbegin

input\_process:begin

repeat

wait(empty);

wait(mutex);

put in a data;

signal(mutex);

signal(full);

until false

end

output\_process:beign

repeat

wait(full);

wait(mutex);

draw out a data;

signal(mutex);

signal(empty);

until false;

end

parend

end

3. 设有三个进程，P1与P2共享10个缓冲区M，P2与P3共享20个缓冲区N。P1每次计算并产生1个数据写入M；P2每次从M取1个数据，处理后将1个结果写入N；P3每次从N取1个数据打印输出。设每个缓冲区存放1个数据，请使用信号量写出这三个进程协同工作的程序。

解：var mutex1,mutex2,empty1,empty2,full1,full2:semaphore:=1,1,10,20

buffer1:array[0…9]of item;

buffer2:array[0…19]of item;

in1,out1,in2,out2:=0,0,0,0;

begin

parbegin

P1:begin

repeat

产生数据;

wait(empty1);

wait(mutex1);

buffer1[in]:=产生的数据；

in1：=(in1+1)mod 10;

signal(mutex1);

signal(full1);

until false;

end

P2:begin

Repeat

Wait(full1);

Wait(mutex1);

从buffer1[out]取数据；

out1：＝(out1+1)mod 10;

signal(mutex1);

signal(empty1);

wait(empty2);

wait(mutex2);

buffer1[in2]:=取出的数据；

in2：＝(in2+1)mod 20;

signal(mutex2);

signal(empty2);

until false

end

p3:begin

Repeat

Wait(full2);

Wait(mutex2);

从buffer2[out2]取数据，打印；

out2：＝（out2＋1）mod 20；

signal(mutex2);

signal(empty2);

until false;

end;

parend;

end

4. 下列并行程序工作时，可能产生死锁，请你分析什么情况下可正确工作，什么情况下会发生死锁。（9分）

VAR s,s1,s2:semaphore;

Begin s:=1; s1:=n-1; s2:=0;

Parbegin

Producer: Repeat Customer: Repeat

P(s); P(s2);

P(s1); P(s);

生产产品； 消费产品；

V(s2); V(s1);

V(s); V(s);

Until false; Until false;

Parend;

End.

答：当S1=0时，即当缓冲区为空后，此时procedure 做P（S）,P(S1)操作后，Customer做P(S2),P(S)操作，发生死锁。

答：程序中Producer先执行的是对互斥信号量的P(s)操作，然后执行的是对资源量的P(s1)操作，这样可能会引起死锁。应该将两者互调位置。另外，Producer中V( )操作应先执行V(s)再执行V(s1)，这样才不会引起死锁。

Customer 中V(s)应先执行，然后才执行V(s1)。这样才会正常工作。

5. 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人。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。请使用信号量写一个避免死锁的程序来解决该问题。

解：Var mutex1,mutex2,mutex3:Semaphore:=1,1,1;

count，count1:integer;

begin

parbegin

从南到北的猴子：begin

repeat

wait(mutex1);

if count=0 then wait(mutex2);

count=count+1;

signal(mutex1);

越过峡谷；

wait(mutex1);

count=count-1;

if count=0 then singal(mutex2);

signal(mutex1);

until false;

end

从北到南的猴子：begin

repeat

wait(mutex3);

if count1=0 then wait(mutex2);

count1=count1+1;

signal(mutex3);

越过峡谷；

wait(mutex3);

count1=count1-1;

if count1=0 then singal(mutex2);

signal(mutex3);

until false;

end

parend

end.

· **信号量初始化**：

* mutex1和mutex3用于分别保护count和count1变量的互斥访问。
* mutex2用于控制南北方向的猴子是否可以进入绳索。

· **从南到北的猴子**：

* 首先等待mutex1，检查是否有猴子在从南到北方向上通过（count = 0）。
* 如果count为0，意味着没有猴子在从南到北通过，此时阻塞从北到南方向的猴子（wait(mutex2)）。
* 进入绳索时，增加count的值。
* 越过峡谷后，减少count的值。如果count为0，表示没有猴子在从南到北通过，释放从北到南方向猴子的阻塞（signal(mutex2)）。

· **从北到南的猴子**：

* 类似于从南到北的逻辑，首先等待mutex3，检查是否有猴子在从北到南方向上通过（count1 = 0）。
* 如果count1为0，意味着没有猴子在从北到南通过，此时阻塞从南到北方向的猴子（wait(mutex2)）。
* 进入绳索时，增加count1的值。
* 越过峡谷后，减少count1的值。如果count1为0，表示没有猴子在从北到南通过，释放从南到北方向猴子的阻塞（signal(mutex2)）。

6. 某一从A到B的单向行车路段AB，为保证行车的安全需设计一个自动管理系统，管理原则如下：

当AB段之间无车行驶时，可让到达A点的一辆车进入AB段行驶；

当AB段有车行驶时，让到达A点的车等待；

当在AB段行驶的车驶出B点后，可让等待在A点的一辆车进入AB段。

请回答下列问题：

（1）把每一辆需经过AB段的车辆看作是一个进程，则这些进程在AB段执行时，它们之间的关系是同步还是互斥？

（2）用PV操作管理AB段时，应怎样定义信号量，给出信号量的初值；说明信号量

（3）若每个进程的程序如下，请在( )中填上适当的语句，以保证行车的安全。

Var ( )

parbegin

Processi(i=1,2,...)

Begin

到达A点：

( )

在AB段行驶；

驶出B点；

( )

end;

parend

答：

1）互斥

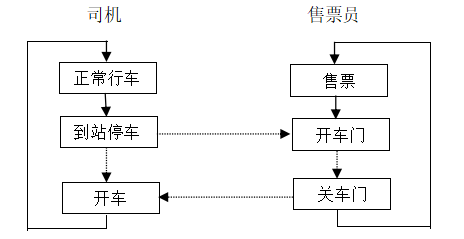
2）mutex＝1；当mutex＝－2时，表示在A点有2两车在等待

3）var mutex:semaphore:=1

p(mutex)

v (mutex)

7. 一辆公共汽车，司机、售票员协同工作：只有得到车门关闭信号后，司机才能开动汽车；只有汽车到站停稳后，售票员才能开门让乘客上、下车。同步操作过程如下图，试用P、V操作表示它们的制约关系



解：

Var mutex1,mutex2：semaphore:=0，0

Begin

Parbegin

司机：begin

repeat

正常开车；

到站停车；

v(mutex2);

p(mutex1);

开车；

until false;

售票员：begin

repeat

售票；

p (mutex2);

开车门；

关车门；

v(mutex1);

until false；

end

parend

end