1. **Illustrate how a binary semaphore can be used to implement mutual exclusion among n processes.（使用二值信号量实现互斥）举例说明**

**The n processes share a semaphore, mutex, initialized to 1. Each process Pi is organized as follows:**

**do {**

**wait(mutex);**

**/\* critical section \*/**

**signal(mutex);**

**/\* remainder section \*/**

**} while (true);**

1. 下面是两个并发执行的进程,它们能正确执行吗?若不能.请举例说明,并修改之.

int x;

Process P1:

void fp1( )

{

int y,z;

x=1;

y= 0;

if (x>=1)

y=y+1;

z=y;

}

Process P2:

void fp2( )

{

int t,u;

x=0;

t= 0;

if(x<=1) t=t+2;

u=t;

}

**不能正确工作，结果具有不确定性，举例说明如下：**

**进程P1 进程P2**

**1 x=1 5 x=0**

**2 y=0 6 t=0**

**3 if x>=1 then y=y+1 7 if x<=1 then t=t+2**

**4 z=y 8 u=t**

**如果P1和P2顺序执行，那么就是从1到4,然后从5到8，则结果就是y=1,z=1,t=2,u=2。**

**如果P1和P2顺序是12563478，则结果就是y=0,z=0,t=2,u=2**

**原因在于使用公共变量x，因此需要P1，P2互斥执行，修改如下：**

**int x;**

**int S;**

**Process P1:**

**void fp1( )**

**{**

**int y,z;**

**P(S);**

**x=1;**

**y= 0;**

**if (x>=1)**

**V(S);**

**y=y+1;**

**z=y;**

**}**

**Process P2:**

**void fp2( )**

**{**

**P(S);**

**int t,u;**

**x=0;**

**t= 0;**

**if(x<=1) t=t+2;**

**V(S);**

**u=t;**

**}**

1. “生产者-消费者”问题演变1（只有同步）：一个缓冲区，一个生产者，一个消费者，生产者不断地生产，消费者不断地消费。只有缓冲区为空时生产者才能放产品，只有缓冲区有数据，消费者才能取产品。用PV操作写出相应的代码段。

**设置两个信号量full和empty。full表示缓冲区是否有产品，由于进程在初始状态时，缓冲区中没有数据，所以初值为0，变换范围：[-1，1]。empty表示缓冲区是否为空，由于初始进程缓冲区为空，初值则为1，变换范围为[-1,1]。**

**Semaphore empty=1,full=0**

**main()**

**{**

**// begin**

**Producer();**

**Consumer();**

**// end**

**}**

**Producer()**

**{**

**while(true)**

**{**

**wait(empty);**

**putdata;**

**signal(full);**

**}**

**}**

**Consumer()**

**{**

**while (true)**

**{**

**wait(full);**

**putdata;**

**signal(empty);**

**}**

**}**

1. “生产者-消费者”问题演变2（既有同步，也有互斥）：一个缓冲区，多个生产者，多个消费者，生产者不断地生产，消费者不断地消费。只有缓冲区为空时生产者才能放产品，只有缓冲区有数据，消费者才能取产品。用PV操作写出相应的代码段。

**缓冲区变成临界资源，不允许多个进程同时操作缓冲区，既不允许多个生产者同时放产品，也不允许多个消费者同时取产品。那么需要增加信号量mutex实现对缓冲区的互斥访问，初值为1.**

**full和empty范围分别变成[-n,1]和[-m,1]，其中n是消费者进程数量，m是生产者进程数量。**

**Semaphore mutex=1,empty=1,full=0;**

**main()**

**{**

**// begin**

**Producer();**

**Consumer();**

**// end**

**}**

**Producer()**

**{**

**While(true)**

**{**

**wait(empty);**

**wait(mutex)**

**putdata;**

**signal (mutex)**

**signal(full);**

**}**

**}**

**Consumer()**

**{**

**While(true)**

**{**

**wait(full);**

**wait(mutex)**

**getdata;**

**signal (mutex)**

**signal(empty);**

**}**

**}**

1. 桌上有一个空盘，最多可以容纳两个水果，每次只能放入或取出一个水果。爸爸专向盘中放苹果，妈妈专向盘中放橘子。儿子专门等吃盘中橘子，女儿专门等吃盘中苹果，请用PV操作实现爸爸、妈妈、儿子、女儿之间的同步与互斥操作。

**可以看作是两个生产者和两个消费者共享一个大小为2的缓冲区，由于只能每次放入或取出一个水果，因此爸爸和妈妈，儿子和女儿之间是互斥关系。爸爸放苹果，女儿吃苹果；妈妈放橘子，儿子吃橘子。因此爸爸和女儿及妈妈和儿子是同步关系。设置4个信号量。Empty表示还可以放几个水果，初值是2，apple对应放入的苹果，orange对应放入的橘子，初值都为0；mutex表示对盘子的互斥访问，初值为1。**

**Semaphore mutex=1,empty=1,full=0;**

**main()**

**father()**

**{**

**while(true)**

**{**

**wait(empty);**

**wait(mutex)**

**放苹果;**

**signal(mutex)**

**signal(apple);**

**}**

**}**

**mother()**

**{**

**while (true)**

**{**

**wait(empty);**

**wait(mutex)**

**放橘子;**

**signal(mutex)**

**signal(orange);**

**}**

**}**

**son()**

**{**

**while (true)**

**{**

**wait(orange);**

**wait(mutex)**

**取橘子吃;**

**signal(mutex)**

**signal(empty);**

**}**

**}**

**daughter()**

**{**

**while (true)**

**{**

**wait(apple);**

**wait(mutex)**

**取苹果吃;**

**signal(mutex)**

**signal(empty);**

**}**

**}**