第2章 进程 作业

1、S1:a=b+c, S2:b=c+d, S3:c=d+e, S4:d=e+1, S5:e=2。用信号量描述前趋关系。

前驱图为：S5---->S4----->S3----->S2----->S1

设信号量分别为a,b,c,d 初始值为0

*semaphore* a=0, b=0, c=0, d=0

*int main*()

{

*cobegin*

{

{S5; V(a); }

{P(a); S4; V(b) }

{P(b); S3; V(c); }

{P(c); S2; V(e); }

{P(e); S1}

}

}

2、包含下述五条语句的程序段：S1：A1=πr12，S2：A2=πr22，S3：A=A1-A2，S4：L=2π(r1+r2)，S5：π=3.14，利用信号量实现前趋关系

S1

前驱图为：S5->S4 S3

S2

*semaphore* a=0, b=0, c=0, d=0

*int main*()

{

*cobegin*

{

{S5; V(a); }

{P(a); S4; V(b); V(c) }

{P(b); S1; V(d); }

{P(c); S2; V(e); }

{P(d);P(e); S3}

}

}

3、利用Test-and-set指令实现互斥。0表示资源空闲，1表示资源被占用。

int TS(int lock)

{

int Temp = \_\_\_\_ lock;

lock = \_\_1\_\_ ;

return \_Temp\_\_\_ ;

}

int main()

{

while(TS(lock)){ };

critical section;

lock = \_\_0\_\_ ;

remainder section;

}

4、利用Swap指令实现进程互斥。0表示资源空闲，1表示资源被占用。

void Swap(int a, int b)

{

int temp;

temp = \_a\_\_\_ ;

a = \_\_\_\_b ;

b = \_\_\_temp\_ ;

}

int main()

{

key = \_\_\_1\_ ;

do{

Swap(lock, key);

}while(key == \_\_1\_\_ );

critical section;

lock = \_\_\_\_0 ;

remainder section;

}

5、不看课件写生产者-消费者问题、读者-写者问题、哲学家问题的伪代码

//生产者消费者问题

Semaphore mutex=1,empty=n,full=0;//empty空缓冲区，full满缓冲区

Int in=0,out=0;

Void producer(){  
 while(true){

Producer an item nextp//nextp用于暂时存放刚刚生产的产品

Wait(empty)

Wait(muetx)

//数组buffter为缓冲池

Buffter[in]=nextp;//临界区

In=(in+1)%n;//临界区

Signal(mutex)

Signal(full)

}

}

Void consumer()

{

While(true)

{  
 wait(full);

Wait(muetx)

Nextc=Buftter[out];//nextc存放每次要消费的产品

Out=(out+1)%n

Signal(mutex)

Signal(empty)

consume the item in nextc;

}

}

*int main*()

{

*cobegin*

{

producer();

consumer();

}

}

//读者写者问题 ----允许多个进程同时读一个共享对象不允许一个writer进程和其他reader进程或writer进程同时访问共享对象

Semaphore rmutex=1,wrutex=1;

Int readcount=0;//正在读的进程数目

Void reader(){

While(true){

Wait(rmutex);

If(readcount==0)

Wait(wrutex);

Readcount++;

Signal(rmutex)

Perform read

Wait(rmutex);

Readcount--;

If(readcount==0)

Signal (wrutex);

Signal(rmutex)

}}

Void write()

{

While(true){

Wait(wrutex);

Perform write

Signal(wrutex);

}

｝

*int main*()

{

*cobegin*

{

read();

write ();

}

}

//哲学家问题

enum estate{thinking,hungry,eating};

estate state[5] = {thinking};

condition self[5];

void pickup(int i) //i: 哲学家编号

{

state[i]=hungry;

test(i);

if (state[i]!=eating) self[i].wait;

}

void putdown(int i);

{

state[i]=thinking;

test(mod(i-1, 5));

test(mod(i+1, 5));

}

void test(int k);

{

if (state[mod(k-1, 5)]!=eating && state[k]==hungry && state[mod(k+1, 5)]!=eating)

{

state[k]=eating;

self[k].signal;

}

}

6、桌上有一空盘，最多允许存放一个水果。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个水果供吃者取用，请用信号量实现爸爸、儿子、女儿三个并发进程的同步。

Semaphore fruit,mutex=1;

Void father(){

While(true){

Wait(mutex);

If(fruit==0)

Fruit++;

Signal(mutex);

}

}

Void son(){

While(true){

Wait(mutex);

If(fruit>0&&fruit==orange)

Fruit--;

Signal(mutex);

}

}

Void daughter(){

While(true){

Wait(mutex);

If(fruit>0&&fruit==apple)

Fruit--;

Signal(mutex);

}

}

*int main*()

{

*cobegin*

{

father ();

son ();

daughter ();

}

}

7、甲、乙、丙三人约定到南校门集合后一起看电影。要求用信号量机制实现进程同步。

Semaphore amutex=1,bmutex=1,cmutex=1;

Int count=0,count2=0,count3=0;//每个人到达南校门次数

Void a(){

While(true){

Wait(amutex);//甲到达

Count++

If(count>=2)

Count==1;

If(count+count2+count3==3){

Wait(bmutex);

Wait(cmutex);

}

Signal(amutex);

人来齐了

Wait(amutex);

//甲离开

If(count+count2+count3==3){

Signal (bmutex);

Signal (cmutex);

}

Count--;

If(count<0)

Count=0;

Signal(amutex);

}

}

Void b(){

While(true){

Wait(bmutex);//乙到达

Count2++

If(count2>=2)

Count2==1;

If(count+count2+count3==3){

Wait(amutex);

Wait(cmutex);

}

Signal(bmutex);

人来齐了

Wait(bmutex);

//乙离开

If(count+count2+count3==3){

Signal (amutex);

Signal (cmutex);

}

Count2--;

If(count2<0)

Count2=0;

Signal(bmutex);

}

}

Void c(){

While(true){

Wait(cmutex);//丙到达

Count3++

If(count3>=2)

Count3==1;

If(count+count2+count3==3){

Wait(amutex);

Wait(bmutex);

}

}

Signal(cmutex);

人来齐了

Wait(cmutex);

//丙离开

If(count+count2+count3==3){

Signal (bmutex);

Signal (cmutex);

}

Count3--;

If(count3<0)

Count3=0;

Signal(cmutex);

}

*int main*()

{

*cobegin*

{

a ();

b ();

c ();

}

}

8、车站售票厅，最多可容纳20名购票者。当售票厅中少于20人时，厅外的购票者可立即进入，否则需在外面等待。设售票厅只有1扇门，同一个时刻只能1个人进出。把每个购票者看作一个进程。1）写进程同步；2）若所有购票者共n人，则信号量可能的变化范围是什么？

Semaphore mutex=1,full =0,empty=20;full=0为满缓冲池，empty为空缓冲池

Void in(){

While(true){

wait (empty)；

wait(mutex);

signal(full);

signal(mutex);

}

}

Void out(){  
while(true){

Wait(full);

wait(mutex);

signal(empty);

signal(mutex);

}

}

*int main*()

{

*cobegin*

{

in();

out();

}

}

变化范围：[0,20]

9、复印室里有一个操作员为顾客复印资料，有5把椅子供顾客休息等待复印。如果没有顾客，则操作员休息。当顾客来到复印室时，如果有空椅子则坐下来，并唤醒操作员；如果没有空椅子则必须离开复印室。

Semaphore mutex=1,mmutex=1,empty=5,full=0;

Int count=0;//顾客数量

Void man(){

Wait(mutex);

If(count==0)

Wait(mmutex);

Signal(mutex);

}

Void in(){

While(true){

Wait(mutex);

If(count==0)

Signal(mmnutex);

Count++;

If(count>5)

Count=5;

Signal(mutex);

}

}

Void out(){

While(true){

Wait(mutex);

Count--;

If(count<0)

Count=0;

Signal(mutex);

}

}

*int main*()

{

*cobegin*

{

man();

in();

out();

}

}

10、用信号量集的方法解决哲学家问题（伪代码）

Semaphore chopstick[5]={1,1,1,1,1},mutex=1;

Void f()

{

Think();

Wait(mutex);

Wait(chopstick[i]);

Wait(chopstick[(i+1)%5]);

Signal(mutex);

Eat();

Signal (chopstick[i]);

Signal (chopstick[(i+1)%5]);

Think();

}