**1.吃水果问题：桌上有个能盛下n个水果的空盘子。父亲不停地向盘中放苹果或桔子，儿子不停地从盘中取出桔子，女儿不停地从盘中取出苹果。规定3人不能同时从盘中取放水果。试用信号量实现3人代表的进程之间的同步。**

1）定性：互斥+同步

2）进程：父亲、儿子、女儿

3）信号量：

互斥信号量：

mutex=1 //临界区互斥信号量,标记盘子是否被占用

同步信号量：

empty=n //可以放水果个数，初始化为 n

orange=0 //盘子内橘子个数，初始化为空 0

apple=0 //盘子内苹果个数，初始化为 0

4）代码:

semaphore empty= n, orange = 0, apple = 0, mutex = 1;

Dad() {

while (1) {

P(empty);//如果先申请临界资源再申请空盘子，则可能出现申请临界资源成功却因为申请空盘子失败而阻塞，结果后面的进程想申请临界资源却无法申请而阻塞，造成连锁反应;

P(mutex);

将水果放入盘中;

V(mutex);

if (放入的是桔子) V(orange); //盘内橘子数加一

else V(apple); //盘内苹果数加一

}

}

Son() {

while (1) {

P(orange); //盘内橘子数减一

P(mutex);

从盘中取一个桔子;

V(mutex);

V(empty); //可放水果数加一

享用桔子;

}

}

Daughter()

while (1) {

P(apple); //盘内苹果数减一

P(mutex);

从盘中取一个苹果;

V(mutex);

V(empty); //可放水果数加一

享用苹果;

}

}

void main(){

cobegin

Dad(); Son();Daughter();

Coend}

**2.** **N 个缓冲区的计算打印问题：计算进程CP负责不断地计算数据并送入缓冲区，打印进程IOP负责不断地从缓冲区取数据打印，有n个缓冲区。**

**Ⅰ.第一种方法：信号量法**

1）定性：互斥+同步

2）进程：计算、打印

3）信号量：

互斥信号量：

mutex=1 //临界区互斥信号量 ，标记缓冲区是否被占用

同步信号量：

empty=n //空闲缓冲区数，初始化为 n

full=0 //缓冲区占用数，初始化为空0

4）代码：

semaphore mutex=1; //临界区互斥信号量 ，标记缓冲区是否被占用 empty=n; //空闲缓冲区数，初始化为 n full=0 ; //缓冲区占用数，初始化为空0

void CP( ) //计算进程

{

while( 1 )

{

得到一个计算结果；

P(empty); //空缓冲区减一

P(mutex);

把数据送到缓冲区；

V(mutex);

V(full); //缓冲区占用数加一

}

}

void IOP( ) //打印进程

{

while( 1 )

{

P(full); //缓冲区占用数减一

P(mutex);

从缓冲区取一个数；

V(mutex);

V(empty); //空缓冲区加一

从打印机上输出;

}

}

void main()

{

cobegin

CP();

IOP();

coend

}

**Ⅱ.第二种方法：新增临界资源法**

semaphore mutex=1,Sa=1;

int num=n;

void CP()

{

while(1)

{

得到一个计算结果;

P(mutex);

P(Sa);

if(num<n)

{

num++;

把数送入缓冲区;

}

V(Sa);

V(mutex);

}

}

void IOP()

{

while(1)

{

P(mutex);

P(Sa);

if(num>0)

{

num--;

从缓冲区取一数;

}

V(Sa);

V(mutex);

从打印机上打印;

}

}

void main()

{

cobegin

CP();

IOP();

coend

}

**3.用管程解决读者写者问题（读者优先）：**

**管程：**

monitor readwriter( )

{

bool write; //判断是否有人正在写

int rcount; //记录有几个读进程

condition rq, wq; //读和写的队列

public:

void startread( )

{

if(write) cwait(rq); //如果有人在写，就阻塞读

rcount++;

csignal(rq); //唤醒下一个等待的读者开始读

}

void endread( )

{

rcount--;

if(rcount==0) csignal(wq);//如果没人读，就唤醒写

}

void startwrite( )

{

if( rcount>0 || write) cwait(wq); //如果有人在读或者在写，就阻塞写

write=true;

}

void endwrite( )

{

write=false;

if(rq!=NULL) csignal(rq); //读进程想读，就唤醒（读者优先）

else csignal(wq); //否则唤醒等待的写者（读者优先）

}

}

{write = false;

rcount = 0; }

}RW;

**读者和写者：**

**读者进程：**

void reader( )

{

while(1)

{

RW.startread( );

开始读

RW.endread;

}

}

**写者进程：**

void writer( )

{

while(1)

{

RW.startwrite( );

开始写

RW.endwrite;

}

}

**主程序：**

void main( )

{

cobegin

reader();

writer();

coend

}