



## 进程同步与互斥：习题2

- 假设有两个生产者进程A、B和一个销售者进程C，他们共享一个无限大的仓库。
  - ◆ 生产者每次循环生产一个产品，然后入库供销售。
  - ◆ 销售者每次循环从仓库中取出一个产品进行销售。
  - ◆ 不允许同时入库，也不允许边入库边出库。
  - ◆ 要求生产A产品和B产品的件数满足以下关系：  
 $-n \leq A \text{的件数} - B \text{的件数} \leq m$ ，其中n、m是正整数
  - ◆ 要求销售A产品和B产品的件数满足以下关系：  
 $-n \leq A \text{的件数} - B \text{的件数} \leq m$ ，其中n、m是正整数
- 试用信号量实现这一同步问题。



Semaphore mutex=1,full=0,fullA=0,fullB=0,SAB=m,SBA=n;

int difference=0;

void ProducerA()

```
{
    do{
        wait(SAB);
        produceA();
        signal(SBA);
        wait(mutex);
        addProductionA();
        signal(mutex);
        signal(fullA);
        signal(full);
    }while(1)
}
```

void ProducerB()

```
{
    do{
        wait(SBA);
        produceB();
        signal(SAB);
        wait(mutex);
        addProductionB();
        signal(mutex);
        signal(fullB);
        signal(full);
    }while(1)
}
```

```

void consumer()
{
    do{
        wait(full);
        if(difference<=n)
        {
            wait(fullA);
            wait(mutex);
            takeAwayProductionA();
            signal(mutex);
            difference++;
        }
        else if(difference>=m)
        {
            wait(fullB);
            wait(mutex);
            takeAwayProductionB();
            signal(mutex);
            difference--;
        }
        else
        {
            wait(mutex);
            int type=takeAwayRandom();
            if(type==A)
            {
                wait(fullA);
                difference++;
            }
            else
            {
                wait(fullB);
                difference--;
            }
        }
        sellProduction();
    }while(1)
}

```



## 进程同步与互斥：习题3

- 考虑三个吸烟者进程和一个经销商进程的系统
  - ◆ 每个吸烟者连续不断地做烟卷并抽他做好的烟卷，做一支烟卷需要烟草、纸和火柴三种原料。
  - ◆ 这三个吸烟者分别掌握有烟草、纸和火柴。
  - ◆ 经销商源源不断地提供上述三种原料，但他只随机的将其中的两种原料放在桌上，具有另一种原料的吸烟者就可以做烟卷并抽烟，且在做完后给经销商发信号，然后经销商再拿出两种原料放在桌上，如此反复。
- 基于信号量设计一个同步算法描述他们的活动



```
Semaphore S[3]={0,0,0};
```

```
Semaphore empty=1;
```

```
void smoker(int i)
```

```
{
    do{
        wait(S[i]);
        makeCigarette();
        signal(empty);
        smoke();
    }while(1)
}
```

```
void dealer()
```

```
{
    do{
        wait(empty);
        int i=randomPut();
        signal(S[i]);
    }while(1)
}
```



## 进程同步与互斥：习题4

■ 现有4个进程R1，R2，W1，W2，它们共享可以存放一个数的缓冲器B。

- ◆ 进程R1每次把从键盘上输入的一个数存放到缓冲器B中，供进程W1打印输出；
- ◆ 进程R2每次从磁盘上读一个数放到缓冲器B中，供进程W2打印输出。
- ◆ 当一个进程把数据存放到缓冲器B后，在该数还没有被打印输出之前不准任何进程再向缓冲器中存数
- ◆ 在缓冲器B中还没有存入一个新的数之前不允许任何进程加快从缓冲区中取出打印。

■ 怎样才能使这四个进程并发执行时协调工作？



Semaphore empty=1,SR1=0,SR2=0;

buffer B;

void R1()

```
{
    do{
        int x=readFromKeyboard();
        wait(empty);
        B=x;
        signal(SR1);
    }while(1)
}
```

void R2()

```
{
    do{
        int x=readFromDisk();
        wait(empty);
        B=x;
        signal(SR2);
    }while(1)
}
```

void W1()

```
{
    do{
        wait(SR1);
        int x=B;
        signal(empty);
        printOut(x);
    }
```

```

    }while(1)
}

```

```

void W2()
{
    do{
        wait(SR2);
        int x=B;
        signal(empty);
        printOut(x);
    }while(1)
}

```

**进程同步与互斥：习题5**

■ a, b两点之间是一段东西向的单行车道，现要设计一个自动管理系统，管理规则如下：

- ◆ 当ab之间有车辆在行驶时，同方向的车可以同时驶入ab段，但另一方向的车必须在ab段外等待；
- ◆ 当ab之间无车辆在行驶时，到达a点(或b点)的车辆可以进入ab段，但不能从a点和b点同时驶入，当某方向在ab段行驶的车辆驶出了ab段且暂无车辆进入ab段时，应让另一方向等待的车辆进入ab段行驶。

■ 请用信号量机制为工具，对ab段实现正确管理以保证行驶安全。

Operating System Concepts 7.20 Southeast University

```


int ab=0, ba=0;
Semaphore Sa=1,Sb=1,mutex=1;
void a()
{
    do{
        wait(Sa);
        if(ab==0)
        {
            wait(mutex);
        }
        ab++;
        signal(Sa);
        wait(Sa);
        ab--;
        if(ab==0)signal(mutex);
        signal(Sa);
    }while(1)
}

```

```


void b()
{
    do{
        wait(Sb);
        if(ba==0)
        {
            wait(mutex);
        }
        ba++;
        signal(Sb);
        wait(Sb);
        ba--;
        if(ab==0)signal(mutex);
        signal(Sb);
    }while(1)
}

```



## 进程同步与互斥：习题6

- 和尚挑水问题：寺庙里有多个小、老和尚，一水缸，水缸容积**10**桶水。
- 小和尚取水放入水缸，老和尚从水缸取水饮用
- 水取自同一水井，水井每次只容一个桶取水。
- 桶总数**3**个，每次倒水入水缸水、或者从水缸中取水都仅为一桶。
- 试用信号量描述和尚取水、饮水的互斥与同步过程。

Operating System Concepts
7.23
Southeast University


```

Semaphore mutex_well=1,mutex_tank=1;
Semaphore empty=10,full=0;
Semaphore buckets=3;
void oldMonk()
{
    do{
        wait(full);
        wait(buckets);
        wait(mutex_tank);
        carryWater();
        signal(mutex_tank);
        signal(buckets);
        signal(empty);
    }
}


```

```

    }while(1)
}

void littleMonk()
{
    do{
        wait(empty);
        wait(buckets);
        wait(mutex_well);
        fetchWater();
        signal(mutex_well);
        wait(mutex_tank);
        pourWater();
        signal(mutex_well);
        signal(full);
        signal(buckets);
    }while(1)
}


```



### 进程同步与互斥：习题7

- 嗜睡的理发师问题：
  - ◆ 一个理发店由一个有N张沙发的等候室和一个放有一张理发椅的理发室组成。没有顾客要理发时，理发师便去睡觉。
  - ◆ 当一个顾客走进理发店时，如果所有的沙发都已经占用，他便离开理发店；否则，如果理发师正在为其他顾客理发，则该顾客就找一张空沙发坐下等待；如果理发师因无顾客正在睡觉，则由新到的顾客唤醒理发师为其理发。
  - ◆ 在理发完成后，顾客必须付费，直到理发师收费后才能离开理发店。
- 试用信号量实现这一同步问题。

Operating System Concepts
7.26
Southeast University



```

int count=0;
Semaphore mutex_count=1;
Semaphore chair_empty=1,chair_full=0;
Semaphore sofa=N;
Semaphore cutFinished=0,payment=0,receipt=0;
void barber()
{
    do{
        wait(chair_full);
        cutHair();
        signal(cutFinished);
        wait(payment);
        acceptPayment();
    }
}

```

```

        signal(receipt);
    }while(1)
}

void client()
{
    do{
        wait(mutex_count);
        if(count>N)
        {
            signal(mutex_count);
            leave();
        }
        else
        {
            count++;
            signal(mutex_count);
            if(count==1)
            {
                wait(chair_empty);
            }
            else
            {
                wait(sofa);
                sitOnSofaForWaiting();
                wait(chair_empty);
                signal(sofa);
            }
            sitOnChairForCutting();
            signal(chair_full);
            wait(cutFinished);
            pay();
            signal(payment);
            wait(receipt);
            signal(empty);
            wait(mutex_count);
            count--;
            signal(mutex_count);
            leave();
        }
    }while(1)
}

```