Operating Systems

Chapter 5 Mutual Exclusion(互斥) and Synchronization(同步)

经典习题

4. 理发师问题

- 一个理发店有 1 个理发师，n 张椅子，1 张理发椅。若没有要理发的顾客，则理发师就去睡觉（什么也不干）；若有顾客走进理发店且所有的椅子都被占用了，则该顾客就离开理发店；若理发师正在为人理发，则该顾客就找一张空椅子坐下等待；若理发师在睡觉，则顾客就唤醒他，请用信号量描述这个过程。

4. 理发师问题

- 理发师：睡觉 / 工作
  - 关键：收到“有客人“信号就工作（同步问题）
  - 工作内容：输出“理发” / 结束：空凳子数 +1 ，有空余理发师（同步）
- 客人：理发 + 等待 / 离开
  - 关键：凳子
    - 有凳子则等待，凳子数 -1 ，通知理发师客人来了（同步），等待理发师（同步）
    - 无凳子则离开
- 互斥数据：凳子数

4．理发师问题

不正确

```
void baber()
{
   while(true) {
      semwait(guest)# 唤醒
      semwait(mutex)
      count--;
            if(count!=0)
             semsignal(guest)// 别睡
            semsignal(mutex)
       }
}
```

```
void customer(){
    while(true) {
            semwait(mutex);
      if (count<n+1) {
            count++
                  if(count==1)
            semsignal(guest)
                  else
            set on chair waiting…
      }
      else{
            leave the baber's            }
       semsignal(mutex)
}
```

1．理发师问题

```
Sem barber =1,guest=0;
void thebaber()
{
   while(true) {
      semwait(guest)# 唤醒
      semwait(mutex)
      count--;
            semsignal(mutex)
           output ： serving…
           semsignal(barber)
      }
}
```

Baber 是一个线程，即便收到多个 guest 消息，也会在本次循环结束后，才执行一次 semwati(guest)

Main 创建 1 个 barber, N 个 customer.

-1 在 baber 更合理，空了一个等待的位置

```
void customer 1(){
    semwait(mutex);
    if (count<n+1) {
          count++
          semsignal(mutex);
          semsignal(guest)
          semwait(barber)//waiting…
          output ： haricutting thenleaving
     }else{
           semsignal(mutex);
           output ： leaving
      }// semsignal(mutex);//here cause deadlock
}
```

2011 考研

45、(8 分) 某银行提供 1 个服务窗口和 10 个供顾客等待的座位。顾客到达银行时，若有空座位，则到取号机上领取一个号，等待叫号。取号机每次仅允许一位顾客使用。当营业员空闲时，通过叫号选取一位顾客，并为其服务。顾客和营业员的活动过程描述如下：

```
cobegin
    {
    process 顾客 i
        {
        从取号机获得一个号码;
        等待叫号;
        获得服务;
        }

process 营业员
    {
    while(TRUE)
        {
        叫号;
        为顾客服务;
        }
    }
}coend
```

请添加必要的信号量和 P、V（或 wait()、signal()）操作，实现上述过程中的互斥与同步。要求写出完整的过程，说明信号量的含义并赋初值。

2011 考研

【答案解析】此题考察的知识点是共享资源的使用与 P、V 操作以防止死锁。

```
Semaphore seets =10;//表示空余座位数量的资源信号量，初值为 10
Semaphore mutex = 1; //管理取号机的互斥信号量，初值为 1，表示取号机空闲
Semaphore custom = 0; //表示顾客数量的资源信号量，初值为 0
Process 顾客
{
    P(seets); //找个空座位
    P(mutex); //在看看取号机是否空闲
        从取号机取号;
    V(mutex) //放开那个取号机
    V(custom); //取到号，告诉营业员有顾客

    等待叫号;
    V(seets) //被叫号，离开座位
    接受服务;
}
Process 营业员
{
    While(true)
    {
        P(custom); //看看有没有等待的顾客
        叫号;
        为顾客服务;
    }
}
```

wait

wait

signal

signal

signal

wait

2011 考研

```
Sem clerk=1,guest=0,chair=n;
void theclerk()
{
   while(true) {
      semwait(guest)
            semsignal(chair);//choice1
      output ： serving…
      semsignal(clerk)
            semsignal(chair);//choice2


       }
}
```

Main 创建 1 个 clerk, N 个 customer

Semsignal(chair) 其中一处即可

```
void customer 1(){
            semwait(chair)
            mutex_lock();
            get a NO.
            mutex_unlock():
         semsignal(guest)
            output ： waiting…
            semwait(clerk)
            semsignal(chair);//choice3
            output ： serving
thenleaving
      }
}
```

5. 超市购物问题

超市可以容纳 500 人同时购物，有 6 扇可供出入的门，既可以进又可以出，每扇门只允许一个人通过，使用信号量解决一下问题：

a)描述购物过程；

b)如果增加一个限制条件：顾客进出必须走同一个门，这个过程又是怎么样的；

5. 超市购物问题

问题定性：一个混合问题（同步和互斥）。

同步：超市里面维持 500 个顾客的规模；

互斥：每道门都是临界资源；

进程的类别：两类代表进程，进入超市，离开超市。

初始化：

同步信号量一个，定义为 s=500,

互斥量 6 个，定义为 s1=s2=s3=s4=s5=s6=1

5. 超市购物问题

```
void enter()
{
    semwait(s);
    choose the door i
    semwait(si);
    cross the door  i;
    semsignal(si)
    buy something;

}
```

```
void leave()
{
    choose door i;
    semwait(si);
    cross the door;
    semsignal(si)
    semsignal(s)


}
```

5. 超市购物问题

```
void customer()
{
    semwait(s);
    choose random door i
    semwait(si);
    cross the door  i;
    semsignal(si)
    buy something;//sleep(random time)
        choose  random door i;
        semwait(si);
        cross the door;
        semsignal(si);
        semsignal(s);
}
```

5. 超市购物问题

如果增加一个限制条件：顾客进出必须走同一个门，这个过程又是怎么样的
；

```
void customer(){
    semwait(s);
    choose random door i
    semwait(si);
    cross the door  i;
    semsignal(si)
    buy something;//sleep(random time)
        semwait(si);
        cross the door;
        semsignal(si);
        semsignal(s);
}
```

6. 数据搬移

- 有三个进程： Read 、 Move 和 Print ，共享两个缓存 B1 和 B2
  - 进程 Read ：读取一条记录，并放在缓存 B1 中
  - 进程 Move ：从缓存 B1 中读取记录，处理后放入缓存 B2 中
  - 进程 Print ：从 B2 中读取数据并打印

  请通过信号量的等待和激发操作填空

6. 数据搬移

```
Initialize
Semaphore S0 = 1;
Semaphore S1 = 0;
Semaphore S2 = _____;(1)
Semaphore S3 = _____;(2)
```

Read Process	Move Process	Print Process

char x;
while (true)
{
Read a record to x;
_____;(3)
B1 = x;
_____;(4)
}
	char x, y;
	while (true)
	{
	_____;(5)
	x = B1;
	_____;(6)
	Process x,
	store the result to y;
	_____;(7)
	B2 = y;
	_____;(8)
	}
		char x;
		while (true)
		{
		_____;(9)
		x = B2;
		_____;(10)
		Print x;
		}

6. 数据搬移

- 知识点：互斥和同步、信号量
- Page 154 ，结合 book5.3 信号量
- 需要注意的地方：系统运行时，必须保证 READ 优先使用 B1 缓存， MOVE 优先使用 B2 缓存（原因：初始状态 B1 和 B2 缓冲中的数据是无效数据）

Problems 2: – 思路

- ✓ 对于 READ 进程：在对 B1 缓冲区读取数据时，首先要判断 MOVE 进程是否将上一次读的数据取走，如果没有取走， READ 等待；否则，读取一段数据放到 B1 ，然后通知 MOVE 进程， B1 缓存可用；
- ✓ 对于 MOVE 进程：首先判断 B1 缓冲区中的数据是否可用，如果没有等待；否则，从 B1 缓存中读取数据，然后对数据进行加工。

生产者消费者

- ✓ 加工完毕后

- ✓ 判断 PRINT 进程是否已将上次 MOVE 进程放到 B2 缓冲区中的数据取走，如果没有取走，等待；否则，将加工后的数据存放到 B2 缓冲区，然后通知 PRINT 进程可以使用 B2 缓冲区；
- ✓ 对于 PRINT 进程：判断 B2 缓冲区的数据是否可用，如果不可用，则等待；如果可用，则打印，然后通知 MOVE 进程， B2 的数据已取走， MOVE 进程可对 B2 缓冲区 进行更新操作。

生产者消费者

6. 数据搬移

通过上面分析，可以知道 ，在这个程序中我们需要使用四个信号量，其中：

- S0 用于表明 READ 是否可以使用 B1 ；（ B1 中是否有空位置 ）
- S1 用于表明 MOVE 是否可以使用 B1 ；（ B1 中是否有数据 ）
- S2 用于表明 MOVE 是否可以使用 B2 ；（ B2 中是否有有空位置 ）
- S3 用于表明 PRINT 是否可以使用 B2 ；（ B2 中是否有数据 ）

6. 数据搬移

- 初始化：因为由于优先级的问题，所以有
  - Semsignal s0=1; （ B1 中有位置 ）1 表示有位置，0 表示没有位置
  - Semsignal s1=0; （ B1 中没有数据 ）1 表示有数据，0 表示没有数据
  - Semsignal s2=1; （ B2 中有位置 ）1 表示有位置，0 表示没有位置
  - Semsignal s3=0; （ B2 中没有 产品 ）1 表示有数据，0 表示没有数据

6. 数据搬移

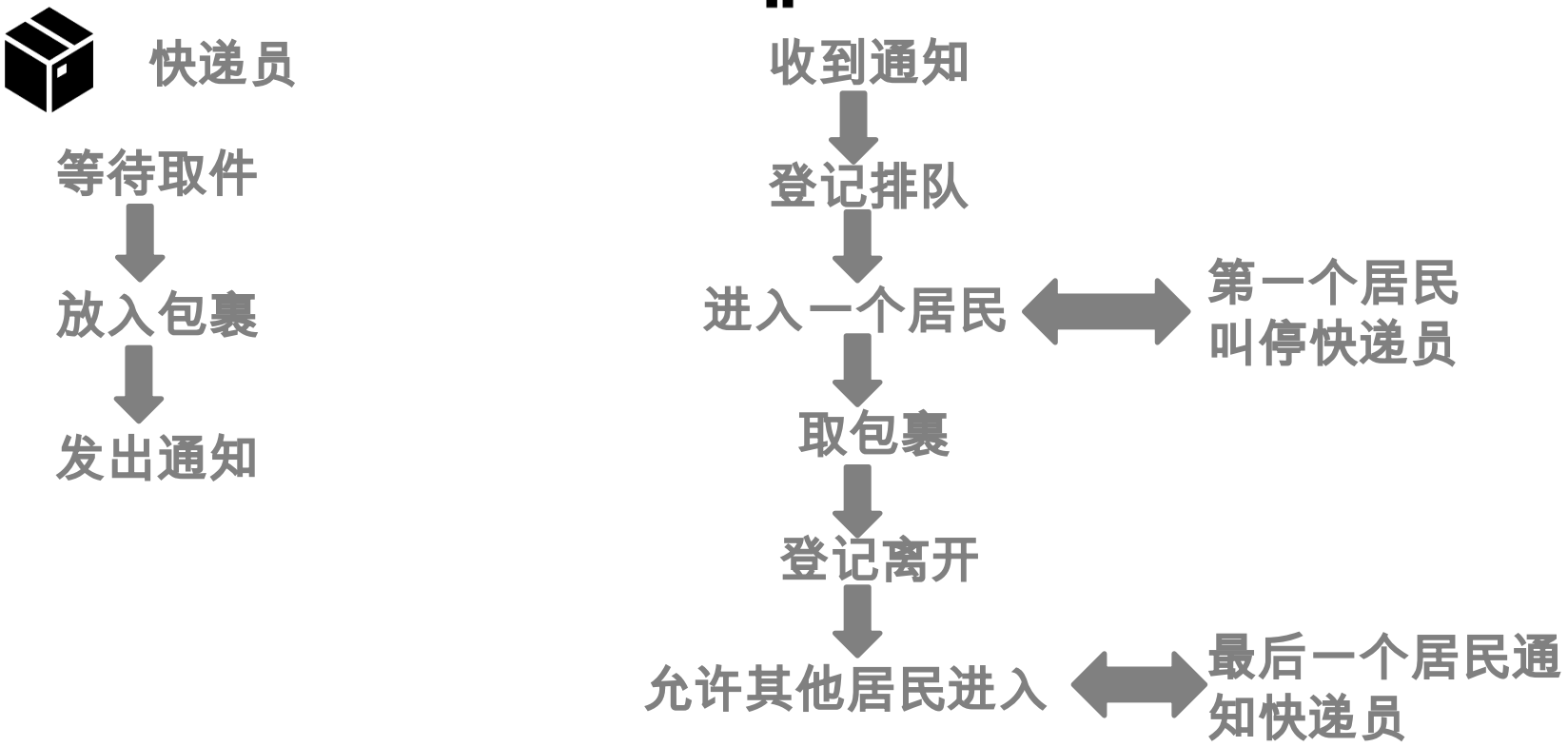
Read Process	Move Process	Print Process

char x;
while (true)
{
Read a record to x;
Semwait(s0);(3)
B1 = x;
Semsignal(s1);(4)
}
	char x, y;
	while (true)
	{
	Semwait(s1);(5)
	x = B1;
	semsignal(s0);(6)
	Process x,
	store the result to y;
	semwait(s2);(7)
	B2 = y;
	Semsignal(s3);(8)
	}
		char x;
		while (true)
		{
		Semwait(s3);(9)
		x = B2;
		semsignal(s2);(10)
		Print x;
		}

7. 快递柜问题

小区里有**1个快递柜**，该柜子有**20个格子**，快递员负责向快递柜放入包裹（**每次只能放入1个**包裹），放好一个包裹发出一个**取件通知**，若快递柜满了，则快递员需要**等待空闲格子**出来；居民凭快递通知，从指定快递柜中取走属于自己的包裹，**每次只能允许一个居民**取包裹；一个快递柜若有新的居民取包裹，快递员需要让**居民先取包裹**。假设初始时快递柜是空的，定义信号量并初始化，使用 P、V 操作模拟快递员和居民进程之间的同步与互斥。

7. 快递柜问题


快递员
等待取件
放入包裹
发出通知
居民
收到通知
登记排队
进入一个居民
第一个居民
叫停快递员
取包裹
登记离开
允许其他居民进入
最后一个居民通
知快递员

sem writemutex=1,readmutex=1**, z=1,mutex1=1,mutex2=1**;

```
void* courier (void *arg){
  while(1){

        semWait(writemutex)
        writedata();
        semSignal(writemutex)

  }}
```

```
void* resident(void *arg){
  while(1){

   semWait(mutex1)
   rcount++
   if(rcount==1)
         semWait(writemutex)
    semSignal(mutex1)

     readdata();

    semWait(mutex1)
    rcount--
     if(rcount == 0)
          semSignal(writemutex)
     semSignal(mutex1)

  }}
```

sem writemutex=1,readmutex=1**, z=1,mutex1=1,mutex2=1**;

sem Number=20

```
void* courier (void *arg){
  while(1){

      semWait(number)
     semWait(writemutex)
      writedata();
      semSignal(writemutex)

}}
```

```
void* resident(void *arg){
  while(1){

  semWait(mutex1)
 rcount++
 if(rcount==1)
       semWait(writemutex)
   semSignal(mutex1)

    readdata();
   semsignal(number)

   semWait(mutex1)
   rcount--
    if(rcount == 0)
         semSignal(writemutex)
    semSignal(mutex1)

}}
```

sem writemutex=1,readmutex=1**, z=1,mutex1=1,mutex2=1**;

Number=20

```
void* courier (void *arg){
  while(1){

      semWait(number)
     semWait(writemutex)
      writedata();
      semSignal(writemutex)

  }}
```

```
void* resident(void *arg){
  while(1){

  semWait(mutex1)
 rcount++
 if(rcount==1)
       semWait(writemutex)
   semSignal(mutex1)
   semWait(mutextakeout)
    readdata();
   semSignal(mutextakeout)
  semsignal(number)
  semWait(mutex1)
  rcount--
   if(rcount == 0)
        semSignal(writemutex)
    semSignal(mutex1)

  }}
```

sem writemutex=1,readmutex=1**, z=1,mutex1=1,mutex2=1**;

Number=20

```
void* courier (void *arg){
  while(1){

      semWait(number)
     semWait(writemutex)
      writedata();
      semSignal(writemutex)

     semSignal(message i)//20 个消息

  }}
```

```
void* resident(void *arg){
  while(1){
 semwaitl(message i)
  semWait(mutex1)
 rcount++
  if(rcount==1)
        semWait(writemutex)
   semSignal(mutex1)
   semWait(mutextakeout)
    readdata();
   semSignal(mutextakeout)
  semsignal(number)
  semWait(mutex1)
  rcount--
   if(rcount == 0)
        semSignal(writemutex)
    semSignal(mutex1)

  }}
```

sem writemutex=1,readmutex=1**, z=1,mutex1=1,mutex2=1**;

Number=20 ， courier_queue=1

Mutextakeout=1

```
void* courier(void *arg){
  while(1){
    semWait(courier_queue)
    semWait(number)// 格子 i
    semWait(writemutex)// 储物柜
    writedata();
    semSignal(writemutex)
    semSignal(courier_queue)
    semSignal(message i)
  }}
```

```
void* resident(void *arg){
  while(1){
  semwaitl(message i)
  semWait(mutex1)
  rcount++
  if(rcount==1)
      semWait(writemutex)
  semSignal(mutex1)
  semWait(mutextakeout) // 储物柜
   readdata();
  semSignal(mutextakeout)
  semsignal(number)
  semWait(mutex1)
  rcount--
   if(rcount == 0)
       semSignal(writemutex)
   semSignal(mutex1)
  }}
```