Operating Systems

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

经典习题

review

- 一个生产者生产 N次

```
void produce () {
        while(true) {
                i++// 无需互斥
                semwait(mayprdc)
                outputs: “producing %d”,i
                semsignal(mayconsume);
}
```

review

- N个生产者竞争生产，每个只生产一次
  - int i=0
  - 如果 mayprdc 只能是 0 和 1（起到互斥作用），那么将 i++ 放入 semwait 后面即可
  - 如果 mayprdc 可以 >1（起不到互斥作用），那么将 i++ 需要额外的互斥锁 mutex

```
void produce 1() {
        i++;// 需要互斥
        semwait(mayprdc)
        outputs: “producing %d”,i
        semsignal(mayconsume);
}
```

```
void produce n() {
        i++; // 需要互斥
        semwait(mayprdc)
        outputs: “producing %d”,i
        semsignal(mayconsume);
}
```

Agenda

- 1. 吃水果问题
- 2. 抽烟者问题
- 3. 过桥问题
- 4. 理发师问题
- 5. 超市问题
- 6. 数据搬移
- 7. 快递柜问题

1. 吃水果

- 吃水果问题：**桌子有一只盘子**，只允许放一个水果，父亲专向盘子放苹果，母亲专向盘子放桔子儿子专等吃盘子的桔子，女儿专等吃盘子的苹果。只要盘子为空，父亲或母亲就可以向盘子放水果， 仅当盘子有自己需要的水果时，儿子和女儿可从盘子取出。

1. 吃水果

- 生产者：父母
  - 等待信号：盘子空
  - 发出信号：某个水果生产完成
- 消费者：儿女
  - 等待信号：某个水果生产完成
  - 发出信号：盘子空
- 互斥数据
  - 盘子

1. 吃水果

apple=0,orange=0,empty=1(对应盘子互斥)

```
void father() {
        while(true) {
                semwait(empty)
                output ： apple ready…
                semsignal(apple)
        } }
```

```
void son(){
    while(true) {
                semwait(orange);
                output ： eat orange …
                semsignal(empty)
}}
```

```
void mother() {
        while(true) {
                semwait(empty)
                output ： orange ready…
                semsignal(orange )
}}
```

```
void daughter(){
    while(true) {
                semwait(apple);
                output ： eat apple…
                semsignal(empty)
}}
```

2. 抽烟者问题

- 抽烟者问题。假设一个系统中有三个抽烟者进程，每个抽烟者不断地卷烟并抽烟。抽烟者卷起并抽掉一颗烟需要有三种材料：烟草、纸和胶水。一个抽烟者有烟草，一个有纸，另一个有胶水。系统中还有两个供应者进程，它们无限地供应所有三种材料，每个供应者每次提供三种材料中的两种。得到缺失的两种材料的抽烟者在卷起并抽掉一颗烟后会发信号通知供应者，让它继续提供。

2. 抽烟者问题

- 抽烟者 tobacco(烟草)paper(纸)glue(胶水)
  - 等待各自信号 semt,semp,semg, 等到则集齐材料 (同步)
  - 发出下一个可以生产信号 mayprdc
- 生产者 produce1, produce2
  - 等待可以生产信号 (互斥) mayprdc
  - 发出 semt,semp,semg 之一 (同步及互斥)

2. 抽烟者问题

semt=0,semp=0,semg=0,mayprdc=1

```
void produce1() {
        while(true) {
                semwait(mayprdc)
                num=random%3;
                if(num==0)
                    semsignal(semt);
                else if(num==1)
                    semsignal(semp);
                else
                    semsignal(semg);
}
```

```
void produce2() {
        while(true) {
                semwait(mayprdc)
                num=random%3;
                if(num==0)
                    semsignal(semt);
                else if(num==1)
                    semsignal(semp);
                else
                    semsignal(semg);
}
```

2. 抽烟者问题

semt=0,semp=0,semg=0,mayprdc=1

```
void tobacco() {
        while(true) {
                semwait(semt)
                 output: t is smoking
                semsignal(mayprdc);
}
```

```
void glue() {
        while(true) {
                semwait(semg)
                 output: g is smoking
                semsignal(mayprdc);
}
```

```
void paper() {
        while(true) {
                semwait(semp)
                 output: p is smoking
                semsignal(mayprdc);
}
```

3. 过桥问题

- 有一座桥，东西走向，汽车可以从东往西走，也可以西往东走，桥上每次只允许朝一个方向走，请用信号量描述下列过程：
  - 如果某一个方向的车占有桥，让这个方向的车优先过桥；

3. 过桥问题

1. 问题 1 的模型和读者写者问题中的读者优先问题类似，在问题中存在两类进程，从东往西走的车，和从西往东走的车，两者存在竞争关系，竞争争夺桥的使用权，这是互斥关系；同时对两类进程的数目进行统计，所以这个问题是一个典型的读者优先问题，且双方均为读者；

1. 初始化：全局变量 ecount 和 wcount 分别统计两边过桥的汽车数目
2. 定义三个信号量： mutex=1(桥的互斥),emutex=1(对 ecount 修改的互斥),wmutex=1(对 wcount 修改的互斥);

3. 过桥问题

```
ecount =0,wcount=0,emutex=1,wmutex=1,mutex=1
```

```
void  e2w(){
    semwait(emutex)
    ecount++
    if (ecount==1)
            semwait(mutex)
    semsignal(emutex)
    cross the bridge from east to west
    semwait(emutex)
    ecount--
    if (ecount==0)
            semsignal(mutex)
    semsignal(emutex)
}
```

```
void  w2e(){
    semwait(wmutex)
    wcount++
    if (wcount==1)
            semwait(mutex)
    semsignal(wmutex)
    cross the bridge from west to east
    semwait(wmutex)
    wcount--
    if (wcount==0)
            semsignal(mutex)
    semsignal(wmutex)
}
```

3. 过桥问题

- 修改方案：让两个方向的车公平的过桥。

- 隐藏涵义：东西方向的车在同一个队列（ 信号量）上排队
- 信号量 queue=1, 线程都需要先 semwait(queue)
  计数信号量

3. 过桥问题

```
void  e2w(){
   semwait(queue);
   semwait(emutex)
    ecount++
    if (ecount==1)
            semwait(mutex)
    semsignal(emutex)
    cross the bridge from east to west
    semwait(emutex)
    ecount--
    if (ecount==0)
            semsignal(mutex)
    semsignal(emutex)
}
```

```
void  w2e(){
     semwait(queue);
     semwait(wmutex)
     wcount++
     if (wcount==1)
            semwait(mutex)
      semsignal(wmutex)
      cross the bridge from west to east
       semwait(wmutex)
       wcount--
     if (wcount==0)
            semsignal(mutex)
     semsignal(wmutex)
}
```

3. 过桥问题

- 同时要满足如果 east 方向的车正在通行，又来一辆 east 方向的车，而第三辆是 west 方向的车，那么第二辆车可以在第一辆通行过程中加入（依然是读者问题）
- Semsignal(queue) 的位置？

- 按照读者问题的解决办法，加在读者排队 semwait(emutex) 和 semwait(wmutex) 后面

3. 过桥问题

```
void  e2w(){
   semwait(queue);
   semwait(emutex)
    ecount++
    if (ecount==1)
            semwait(mutex)
    semsignal(queue)
    semsignal(emutex)
   // semsignal(queue)
    cross the bridge from east to west
    semwait(emutex)
    ecount--
    if (ecount==0)
            semsignal(mutex)
    semsignal(emutex)
}
```

```
void  w2e(){
     semwait(queue);
     semwait(wmutex)
     wcount++
     if (wcount==1)
            semwait(mutex)
      semsignal(queue)
     semsignal(wmutex)
     //semsignal(queue)
     cross the bridge from west to east
      semwait(wmutex)
      wcount--
     if (wcount==0)
            semsignal(mutex)
     semsignal(wmutex)
}
```