作业4

> 主要知识点

```
//经典信号量机制
P(S): while S<=0 do skip
    S:=S-1;
V(S): S:=S+1;

//计数信号量机制
Type semaphore = record
    value : integer;
    L : list of process;
end

Procedure P(S)
    var S : semaphore;
    begin
        S.value := S.value -1;
        if S.value<0 then block(S.L);
    end

procedure V(S)
    var S : semaphore;
    begin
        S.value := S.value + 1;
        if S.value<=0 then wakeup(S.L)
    end
```

1. 读者写者问题（写者优先）：1）共享读; 2）互斥写、读写互斥; 3）写者优先于读者（一旦有写者，则后续读者必须等待，唤醒时优先考虑写者）。

```
int readcount = 0;
int writecount = 0;
semaphore rmutex = 1;   //保护readcount
semaphore wmutex = 1;   //保护writecount
semaphore readTry = 1;  //写者优先
semaphore resource = 1; //写写互斥 读写互斥
// Writer
writer() {
while(true) {
    <ENTRY Section>
        P(wmutex);
        writecount++;
        if (writecount == 1)
            P(readTry)
        V(wmutex);
        P(resource);

    <CRITICAL Section>
        write
        V(resource);
```

```
    <EXIT Section>
        P(wmutex);
        writecount--;
        if (writecount == 0)
            V(readTry);
        V(wmutex);
    }
}

// Reader
reader() {
while(true) {
    <ENTRY Section>
        P(readTry);
        P(rmutex);
        readcount++;
        if (readcount == 1)
            P(resource);
        V(rmutex);
        V(readTry);

    <CRITICAL Section>
        read

    <EXIT Section>
        P(rmutex);
        readcount--;
        if (readcount == 0)
            V(resource);
        V(rmutex);
    }
    P(mutex);
    if readcount=0 then P(wmutex);
        readcount:=readcount+1;
    V(mutex);

    read

    P(mutex)
        readcount:=readcount-1;
    if readcount=0 then V(wmutex);
    V(mutex)
    }
}
```

2. 寿司店问题。假设一个寿司店有 5 个座位，如果你到达的时候有一个空座位，你可以立刻就坐。但是如果你到达的时候 5 个座位都是满的有人已经就坐，这就意味着这些人都是一起来吃饭的，那么你需要等待所有的人一起离开才能就坐。编写同步原语，实现这个场景的约束。

```
int eating = 0;
int waiting = 0;
bool must_wait = false;
semaphore mutex = 1;    //保护eating和waiting
semaphore queue = 0;
```

```
6
7  while(true) {
8      P(mutex);
9      if(must_wait) {
10         waiting++;
11         V(mutex);
12         P(queue);
13     } else {
14         eating++;
15         if (eating == 5)
16             must_wait = true;
17         else must_wait = false;
18         V(mutex);
19     }
20
21     sit_and_eat();
22
23     P(mutex);
24     eating--;
25     if (eating == 0) {
26         int n = min(5,waiting);
27         waiting -= n;
28         eating +=n;
29         if (eating == 5)
30             must_wait = true;
31         else must_wait = false;
32         while(n--)
33             V(queue);
34     }
35     V(mutex);
36 }
```

3. 进门问题。（1）请给出 P、V 操作和信号量的物理意义。（2）一个软件公司有 5 名员工，每人刷卡上班。员工刷卡后需要等待，直到所有员工都刷卡后才能进入公司。为了避免拥挤，公司要求员工一个一个通过大门。所有员工都进入后，while最后进入的员工负责关门。请用 P、V 操作实现员工之间的同步关系。

   (1)

   - P操作：表示申请资源，如果资源可用（信号量的值大于0），则减少资源数量并继续执行；如果资源不可用（信号量的值小于等于0），将信号量的值减一并阻塞该进程，等待资源可用。
   - V操作：表示释放资源，将信号量的值加一，如果有其他进程正在等待该资源，执行V操作会唤醒其中一个进程，使其可以继续执行。
   - 信号量是一个整型变量，用于控制对共享资源的访问。通过P、V操作对信号量进行操作，实现进程间同步和互斥的机制。

   (2)

```
1  int count = 0;
2  semarhore mutex = 1;    //保护count
3  semaphore barrier = 0;
4  semaphore door = 1;
5
6  P(mutex);
7  count++;
```

```
V(mutex);

if(count == 5)
    V(barrier);
P(barrier);
V(barrier);

P(door);
indoor();
    P(mutex);
    count--;
    if(count == 0)
        closedoor();
    V(mutex);
V(door);
```

4. 搜索-插入-删除问题。三个线程对一个单链表进行并发的访问，分别进行搜索、插入和删除。搜索线程仅仅读取链表，因此多个搜索线程可以并发。插入线程把数据项插入到链表最后的位置；多个插入线程必须互斥防止同时执行插入操作。但是，一个插入线程可以和多个搜索线程并发执行。最后，删除线程可以从链表中任何一个位置删除数据。一次只能有一个删除线程执行；删除线程之间，删除线程和搜索线程，删除线程和插入线程都不能同时执行。

   请编写三类线程的同步互斥代码，描述这种三路的分类互斥问题。

```
semaphore insertMutex = 1;
semaphore searchMutex = 1;
semaphore No_search = 1;    //为1时没有搜索进程
semaphore No_insert = 1;    //为1时没有插入进程
int searcher = 0;
int inserter = 0;

Search() {
    P(searchMutex);
    searcher++;
    if (searcher == 1)
        P(No_search);
    V(searchMutex);

    Searching();

    P(searchMutex);
    searcher--;
    if (search == 0)
        V(No_search);
    V(searchMutex);
}

Insert() {
    P(insertMutex);
    inserter++;
    if (inserter == 1)
        P(No_insert);
    V(insertMutex);

    P(insertMutex);
```

```
    Inserting();
    V(insertMutex);

    P(insertMutex);
    inserter--;
    if(inserter == 0)
        V(No_insert);
    V(insertMutex);
}

Delete() {
//删除线程之间，删除线程和搜索线程，删除线程和插入线程不能同时执行
    P(No_search);
        P(No_insert);
            Deleting();
        V(No_insert);
    V(No_search);
}
```