第三章课后作业

191300073 杨斯凡 191300073@smail.nju.edu.cn

1. 不论是抢占式进程（线程）还是在非抢占式中，Peterson 算法总是有效的

因为进程的执行总是遵守原子性的，执行的基本单位是指令，因此不可能在指令执行的过程中被打断，只要保证了指令执行是正确的，那么 Peterson 算法就可以保证临界区不会被多个进程（线程）同时访问，因为 turn 保证了一次只有一个进程访问。

2.循环调度不会出现这样的问题，因为循环调度可以保证 L 迟早被调度执行，而优先级调度就不会，优先级调度 L 执行的唯一原因是比 L 优先级高的进程（线程）全部执行完毕，而 H 又陷入了死等的状态，故 L 不可能进行执行。而在循环调度中，L 迟早会被调度执行，那么 L 和 H 的等待条件中就有一个一定会被满足，则就可以继续执行。

3. 不满足，不妨假设 P0 进程运行结束，只剩下 P1 一个进程，这个时候 turn 被设为 0，那么 P1 就得一直等待下去无法运行，故不满足要求。

课后习题:

3. 在对 S2 进行 P V 操作之前，，此时 z=5，x=10，而有 4 种情况:
    1. V(S2)->z=z+x-> P(S2) -> y=z+y　　z=15　y=19 x=10
    2. V(S2)-> P(S2) -> z=z+x-> y=z+y　　z=15　y=19 x=10
    3. V(S2)-> P(S2) -> y=z+y -> z=z+x　　z=15　y=9　x=10
    4. P(S1)->x=x+y->V(S2)->z=z+x->z=y+1-> P(S2) -> y=z+y z=5　y=9 x=10

22.

```
信号量:
semaphore waits,mutex
waits=0
mutex=1
int sum=0
cobegin
    process read_i(int num){
        P(mutex)
        if(num+sum<M){
            sum+=num
            V(mutex)
            }
        else{
        V(mutex)
        P(wait)
        }
        Read File
        P(mutex)
        sum-=num
        V(waits)
        V(mutex)
    }
coend
```

```
管程：

type read_file=monitor{
    int sum=0,count=0
    code small=0
    InterfaceModule _INTEGRAL_MAX_BITS
    define start_read,end_read
    use enter,leave,wait,signal
}

procedure start_read(int num){
        enter(IM);
        while(num+sum>=M)  {wait(small,count,IM);}
        sum+=number;
        leave(IM);
}
procedure end_read(int num){
        enter(IM);
        sum-=number;
        signal(small,count,IM)
        leave(IM);
}
cobegin
    process read_i(){
        int num=Process number
        read_file.srart_read(num)
        Read File
        read_file.end_read(num)
    }
```

24.

（1）

易知P_0的 Need 分别为 0,0,1,2 ，P_1为 1,7,5,2，P_2为 2,3,5,6，P_3为 0,6,5,2，P_4为 0,6,5,6，则可知存在安全序列{P_0，P_3 ，P_4，P_1，P_2}

则系统此时处于安全状态

（2）不能分配，因为分配之后，系统将不会满足任何进程的需求，将会造成死锁，处于不安全状态

29.

易知，每个缓冲区需要进行一次写操作，但是需要进行n_2次读操作，可以看做n_2个长度为 m 的缓冲区，每个进程 A 同时写n_2次，因此需要设置一个互斥信号量 mutex，full[n_2]，empty[n_2]信号量来表示缓冲区信息以便于读写的等待，

伪代码如下：

```
semaphore mutex=1,empty[n_2]=m,full[n_2]=0//empty的每一个元素是m，full的每一个元素是0

cobegin
procedure send(){
for (int i=0;i<n2;i++){
      p(empty[i]);
      p(mutex);
      Put message to buffer
      V(mutex);
      }
for(int i=0;i<n2;i++) V(full[i]);
}
procedure receive(i){
    p(full[i]);
    p(mutex);
    Take the message
    V(mutex);
    V(emtpy[i]);
}

process A_i(){
    while(1)    send();
}
process B_i(){
    while(1)    receive(i);
}

main(){
    while(1){
        for(int i=1;i<n_1;i++)  A_i();
        for(int i=0;i<n_2;i++)  B_i();
    }
}
coend
```