OS 第4章作业

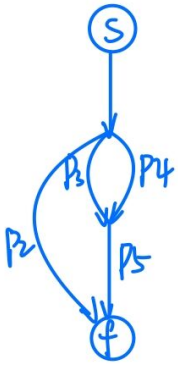
徐锦慧_CS2011_U202011675

1.

1）进程执行次序是混合式的，既有顺序的，也有并行的。P1进程执行结束后，P2、P3、P4才可以执行；P3、P4执行结束后，P5才可以执行；P2、P5结束后任务终止。

2）程序描述如下：

```
main()
{ int  S2=0, S3=0, S4=0;   /* 分别表示进程P2、P3、P4能否开始执行 */
  int  S35=0, S45=0;   /* 分别表示P3、P4进程是否执行完 */
  cobegin
     P1(); P2(); P3(); P4(); P5();
  coend
}
```



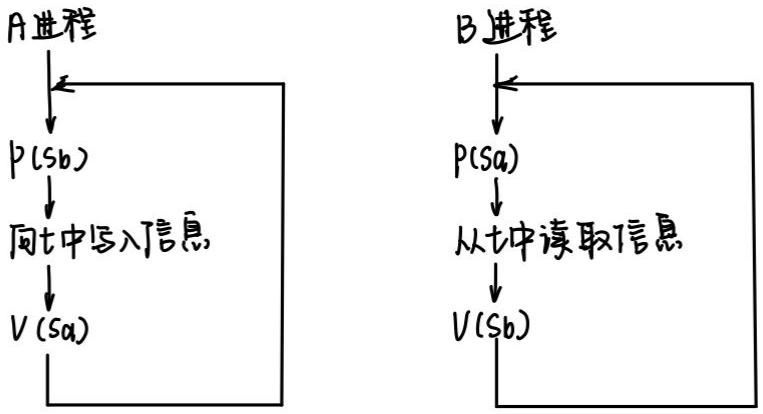

```
P1()          P2()          P3()          P4()          P5()
{             {             {             {             {
  ...           P(S2);        P(S3);        P(S4);        P(S35);
  V(S2);        ...           ...           ...           P(S45);
  V(S3);                      V(S35);       V(S45);       ...
  V(S4);                    }             }             }
}
```

2.

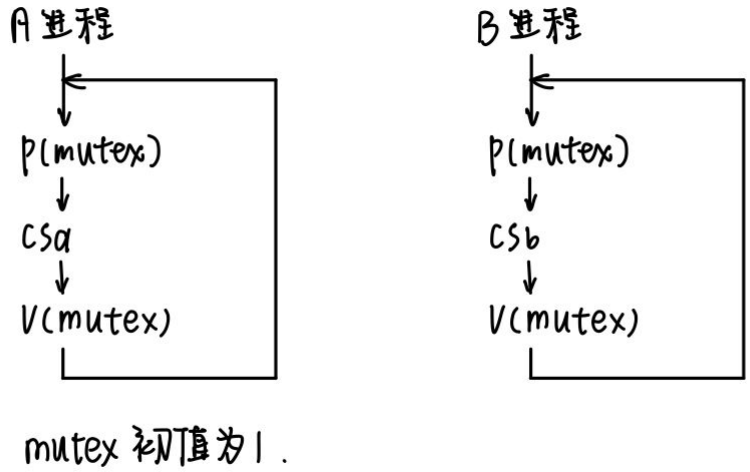
1）错误。算法不能保证只有P1、P2进程结束后，P3进程才能执行。改正的算法描述如下：

```
main()
{
   int S1=0,S2=0;  /* S1、S2分别表示P1、P2进程是否执行完*/
   cobegin
      P1(); P2(); P3();
   coend
}
P1()              P2()              P3()
{                 {                 {
   ...               ...               P(S1);
   V(S1);            V(S2);            P(S2);
}                 }                    ...
                                    }
```

2）错误。A进程写入前应判断t中是否有空位，B进程读取前应判断t中是否有信息，故应采用2个信号量：Sa表示t中有无信息，Sb表示t中有无空位，算法描述如下：



3）错误。a、b进程共享同一临界资源，应使用同一信号灯，算法描述如下：



3.

1)

- 变迁3：进程在执行过程中，需要等待某事件的发生才能继续执行
- 变迁2：运行进程在分得的时间片内未完成， 时间片到时
- 变迁4：等待进程等待的事件发生

2)

① 2→5： 可能发生，当高优先就绪队列非空时可能。

② 2→1： 可能发生， 当运行进程的时间片到时发生变迁 2， 若此时高优先就绪队列为空，则会发生变迁1。

③ 4→5： 不可能发生， 因为采用的是非剥夺式调度。

④ 4→2 ： 不可能发生。

⑤ 3→5： 可能发生， 当高优先就绪队列非空时可能。

3)

① 调度策略： 首先调度高就绪队列中的进程投入运行， 为其分配的时间片大小为100ms。 只有当高就绪队列中的所有进程全部运行完或处于阻塞状态，才调度低优先就绪队列中的进程，为其分配的时间片大小为500ms。 若一个运行进程时间片到时未完成就进入低优先就绪队列。 若某进程在运行期间因等待某事件发生而进入阻塞队列，则当等待事件完成后，进入高优先就绪队列。
② 调度效果： 优先照顾了 I/O 量大的进程，但通过给计算型进程分配更长的时间片也照顾了计算型进程。

4.

1）sleep(1);
2）退出次序为：p2 → main → p1
3）a in p2 = 102
a in main = 101
a in p1 = 102

5.填写程序如下：

```
P南向北（）  /* 自南向北过桥行人*/
{
    P(mutex1)
    count1++;
    if (count1==1)
        P(mutex)
    V(mutex1);
     过桥；
    P(mutex1)
    count1--;
    if  (count1==0)
        V(mutex)
    V(mutex1);
}
```

```
P北向南（）  /* 自北向南过桥行人*/
{
    P(mutex2)
    count2++;
    if (count2==1)
        P(mutex)
    V(mutex2);
     过桥；
    P(mutex2)
    count2--;
    if  (count2==0)
        V(mutex)
    V(mutex2);
}
```

6.程序描述如下：

```
main()
{
    int apple=0, orange=0 ; /* 分别表示盘子中的苹果、桔子个数 */
    int free=2;  /* 表示盘中空闲的可置数 */
    int mutex=1;  /* 互斥信号灯 */
    cobegin
       father(); mother(); son(); daughter();
    coend
}

father()
{
   while(true){
     P(free);
     P(mutex);
     放一个苹果;
     V(mutex);
     V(apple);
   }
}

mother()
{
   while(true){
     P(free);
     P(mutex);
     放一个桔子;
     V(mutex);
     V(orange);
   }
}

son()
{
   while(true){
     P(orange);
     P(mutex);
     吃一个桔子;
     V(mutex);
     V(free);
   }
}

daughter()
{
   while(true){
     P(apple);
     P(mutex);
     吃一个苹果;
     V(mutex);
     V(free);
   }
}
```