第 4、6 章

4-9、

（1）

发生变迁 2 的原因：时间片到；或者当正在运行的一个低优先级进程时，就绪状态有一个高优先级进程，则将低优先级从运行状态转为就绪状态，发生变迁 2。

发生变迁 3 的原因：需要请求某些服务时，如需要执行 I/O 操作等。

发生变迁 4 的原因：请求的服务完成时，发生变迁 4。

（2）当出现变迁 3 时，CPU 空闲，此时系统调度程序必定会选取一个处于就绪状态的进程投入运行，故发生变迁 3。所需条件为发生变迁 1 的时刻，系统中应该有其它处于就绪的进程，即就绪队列非空。

（3）

a.2->1 一定会发生，当出现变迁 2 时，CPU 空闲，此时系统调度程序必定会选取一个处于就绪状态的进程投入运行，而变迁 2 完成后，就绪队列必不为空，一定会发生变迁 1。

b.3->2 不可能发生，当发生变迁 3 后，CPU 空闲，没有进程可调度到就绪状态，也就不可能发生变迁 2。

c.4->1 可能发生，在优先权原则下高优先级进程会先运行。则发生该变迁的条件是，当发生变迁 4 后，进入就绪状态的进程优先级最高，于是 CPU 暂停当前运行进程，发生变迁 1，运行该最高优先级进程。

4-12、

程序描述：

```
main()
{
    int mutex = 1;  /*公共变量Q可被使用*/
    cobegin
       p1();p2();P3();...;Pn();
    coend
}

pi()  /*i从1到n*/
{
    ...
    p(mutex);
    使用公共变量Q;
    v(mutex);
    ...
}
```

信号灯范围为[-(n-1)，1]

mutex = 1：没有进程进入临界区执行，公共变量 Q 可被使用；

mutex = 0：有一个进程进入临界区执行，公共变量 Q 正在被使用；

mutex = -1：有一个进程进入临界区执行，公共变量 Q 正在被使用，且有另一个进程正在等待使用公共变量 Q；

…

mutex = -(n–1)：有一个进程进入临界区执行，公共变量 Q 正在被使用，且有另外 n–1 个进程正在等待使用公共变量 Q；

4-13、

a.

程序描述：

```
main()
{
    int s2=0; /*表示p2能否开始执行*/
    int s3=0; /*表示p3能否开始执行*/
    int s4=0; /*表示p4能否开始执行*/
    cobegin
      p1();
      p2();
      p3();
      p4();
    coend
}
```

```
p1()
{
    ...
    v(s2);
    v(s3);
    v(s4);
}
```

```
p2()
{
    p(s2);
    ...
}
```

```
p3()
{
    p(s3);
    ...
}
```

```
p4()
{
    p(s4);
    ...
}
```

b.

程序描述：

```
main()
{
    int s1=0; /*表示p1是否完成*/
    int s2=0; /*表示p2是否完成*/
    cobegin
      p1();
      p2();
      p3();
    coend
}
```

```
p1()
{
    ...
    v(s1);
}
```

```
p2()
{
    ...
    v(s2);
}
```

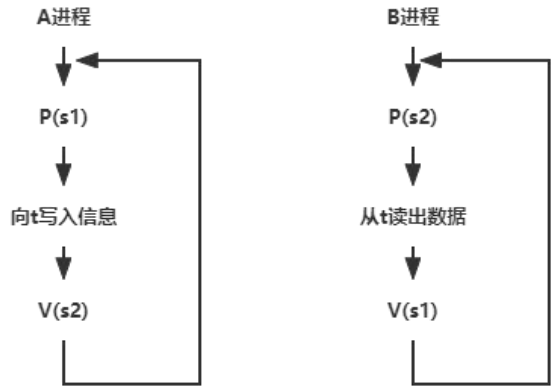
```
p3()
{
    p(s1);
    p(s2);
    ...
}
```

4-18、

（1）错误，信号灯初值不能为负数。
改正为：

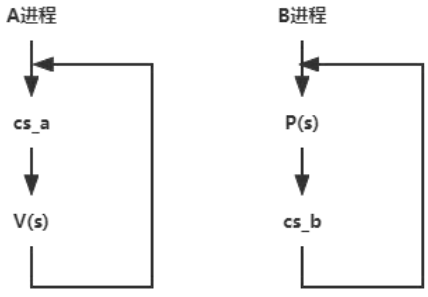
```
main()
{
    int s1=0; /*表示p1是否完成*/
    int s2=0; /*表示p2是否完成*/
    cobegin
      p1();
      p2();
      p3();
    coend
}
p1()
{
    ...
    v(s1);
}
p2()
{
    ...
    v(s2);
}
p3()
{
    p(s1);
    p(s2);
    ...
}
```

（2）错误，公用同一个缓冲区，不仅要互斥，还应满足同步原则。
改正为：



注：s1的初值为1；s2的初值为0

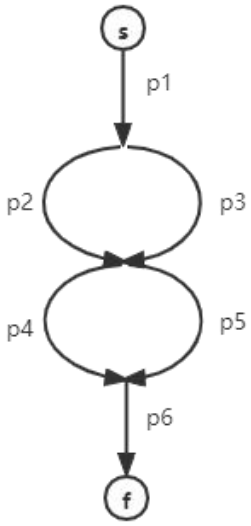
（3）错误，当共享一临界资源时，只需要一个互斥锁即可。
改正为：



注：信号灯s初值为0

4-31、

先画出 6 个进程合作的进程流程：



在根据进程的合作流程计算对应过程值：
p1 后，x=y=1，z=0； p2 和 p3 合作后，x=z=3，y=2；p4 和 p5 合作后，x=5，y=2，z=5；p6 后，z=12。
故 z 的最后结果为 12.

4-32、

算法描述如下：

```
main()
{
    int sb1=1; /*表示缓冲区B1是否为空*/
    int sb2=1; /*表示缓冲区B2是否为空*/
    int tb1=0; /*表示缓冲区B1是否有数据*/
    int tb2=0; /*表示缓冲区B2是否有数据*/
    cobegin
      p1();
      p2();
      p3();
    coend
}
```

```
p1()
{
    while(整个过程未结束){
        从输入设备读取数据;
        if(数据>0){
            p(sb2);
            将数据放入缓冲区B2;
            v(tb2);
        }
        else {
            p(sb1);
            将数据放入缓冲区B1;
            v(tb1);
        }
    }
}
```

```
p2()
{
    while(整个过程未结束){
        p(tb1);
        从B1取出数据并加工;
        v(sb1);
        p(sb2);
        将加工后的数据放入B2;
        v(tb2);
    }
}
```

```
p3()
{
    while(整个过程未结束){
        p(tb2);
        从B2取出数据;
        v(sb2);
        输出数据;
    }
}
```

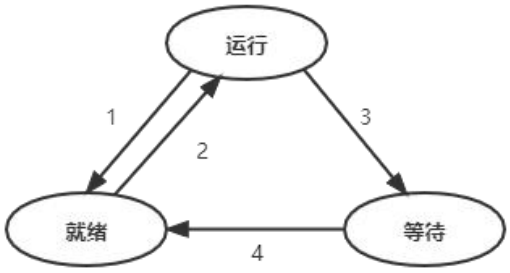
6-3、

（1）
发生变迁 3 的原因：当在运行状态下的进程，需要进行 I/O 操作时，发生变迁 3；
发生变迁 2 的原因：在运行时时间片到时未完成任务，则发生变迁 2；
发生变迁 4 的原因：当 I/O 操作完成后，发生变迁 4。
（2）
①2->5：可能发生，当运行状态下的进程时间片到，且高优先就绪队列不为空。
②2->1：可能发生，当运行状态下的进程时间片到，且高优先就绪队列为空时，会调度低优先就绪队列中的队首进程。
③4->5：不会发生，因为采用的是非剥夺方式调度。
④4->2：不会发生，两者没有因果性。
⑤3->5：可能发生，当运行中的进程需要 I/O 操作时，且高优先就绪队列不为空。
（3）
调度策略：在运行过程中有多种时间片和多种优先级，所以采用的是可变时间片和可变优先级相结合的调度策略。首先优先选取高优先就绪队列的进程投入运行，分配时间片为 100ms，且一旦有 I/O 操作时，将会请求 I/O，I/O 完成后又被赋予高优先数，故对有 I/O 操作的进程有较为优先的调度。当高优先队列为空后，调度低优先就绪队列进程，同时设置更长的时间片 500ms，为其提供更长的运行时间。若有 I/O 操作，同样按开始所述进行调度。

调度效果：I/O 次数多的进程会被赋予高的优先级，计算量大的进程赋予低的优先级，所以优先照顾 I/O 量大的进程。低优先级的进程会分配较大的时间片，高优先级的进程分配较小的时间片，所以适当平衡照顾了计算量大的进程。

6-11、

（1）
进程状态变迁图如下图：



上图为考虑时间片机制下的状态变迁图，若不考虑时间片，则无变迁 1.
变迁 1 的原因：时间片到；
变迁 2 的原因：当运行状态为空，且就绪队列中有进程。就绪队列中按优先数排序，优先数高的排在队列前面，优先得到调度；
变迁 3 的原因：目前运行的进程需要进行 I/O 等操作；
变迁 4 的原因：完成 I/O 等操作后。

（2）

调度次序	A	B	A	C	B	C
占用时间(ms)	30	30	10	20	10	20

6-12、

（1）

“先来先服务调度算法”：P1→P2→P3→P4→P5

“非抢占式的优先数调度算法”：P1→P4→P3→P5→P1

（2）

表 6.10

进程	先来先服务调度算法等待时间	非抢占式的优先数调度算法等待时间
P1	0	0
P2	10	18
P3	11	11
P4	13	10
P5	14	13
先来先服务平均等待时间= 9.6		
非抢占式的优先数平均等待时间= 10.4		