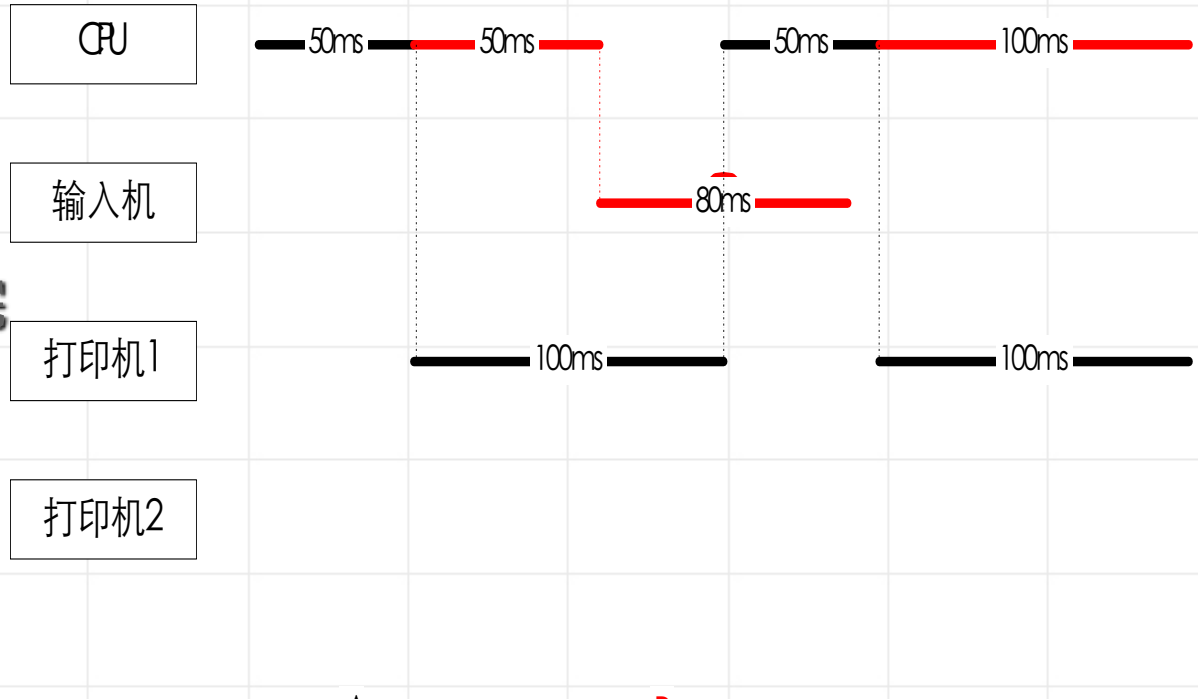
1-9

答：（1）见右图。



（2）有，B50ms计算完后，A 100ms打印仍在进行，中间CPU空闲50ms。

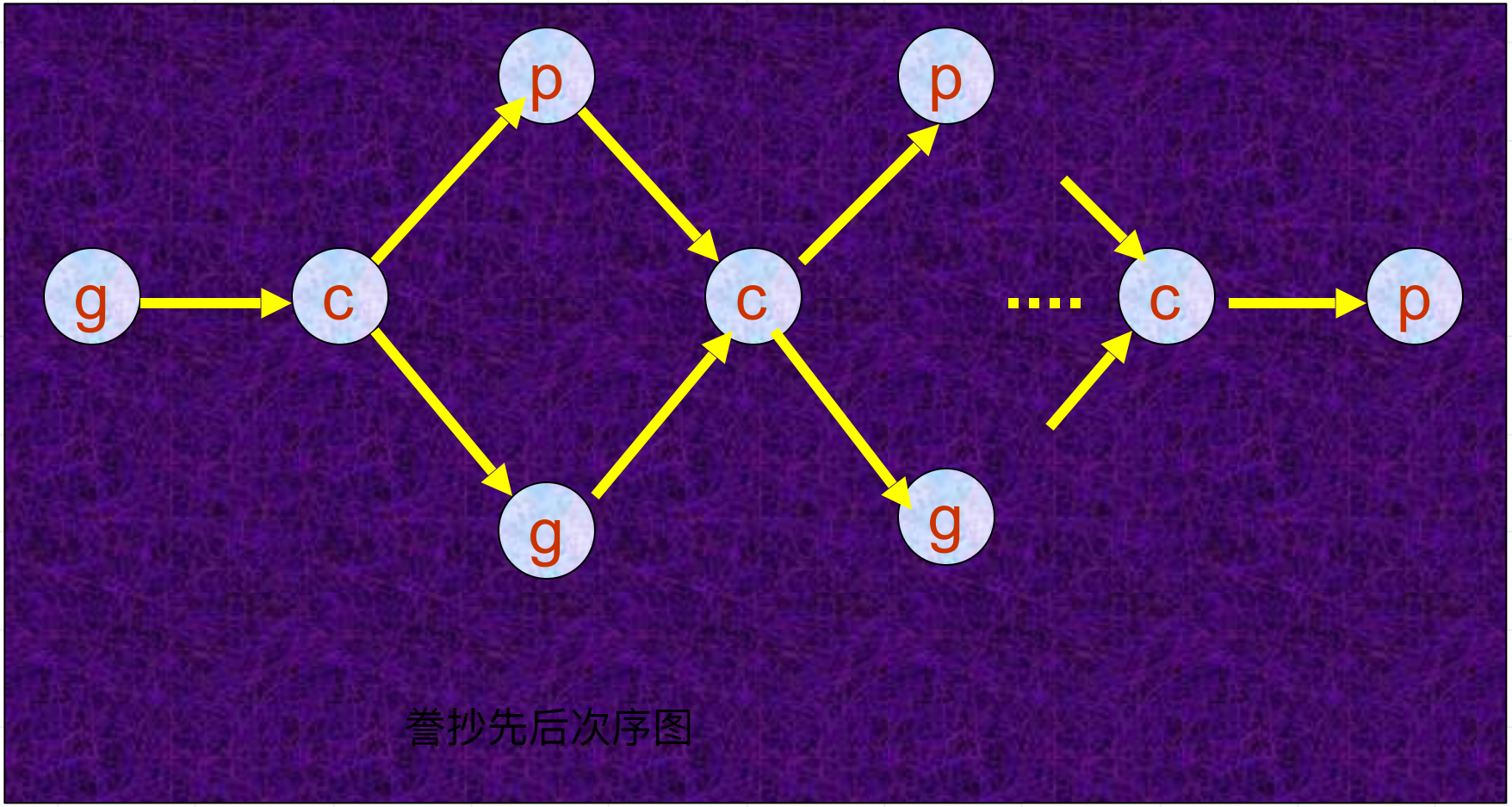
（3）有，B一开始需要等待50ms；B 80ms输入后，等待20ms后才能进行计算。

4-4

(1) I_n、C_n和P_n之间有先后顺序要求，这是由于程序本身的逻辑要求；使用同一设备的不同作业的程序段，如$C_1 \ldots C_n$，$I_1 \ldots I_n$，$P_1 \ldots P_n$，之间有先后顺序要求，这是由于设备某一时刻只能为一个作业服务。

(2) 不同作业中使用不同设备的程序段，占用不同设备，无逻辑关系，可以并发执行，如I_2和C_1，I_3、C_2和P_1。

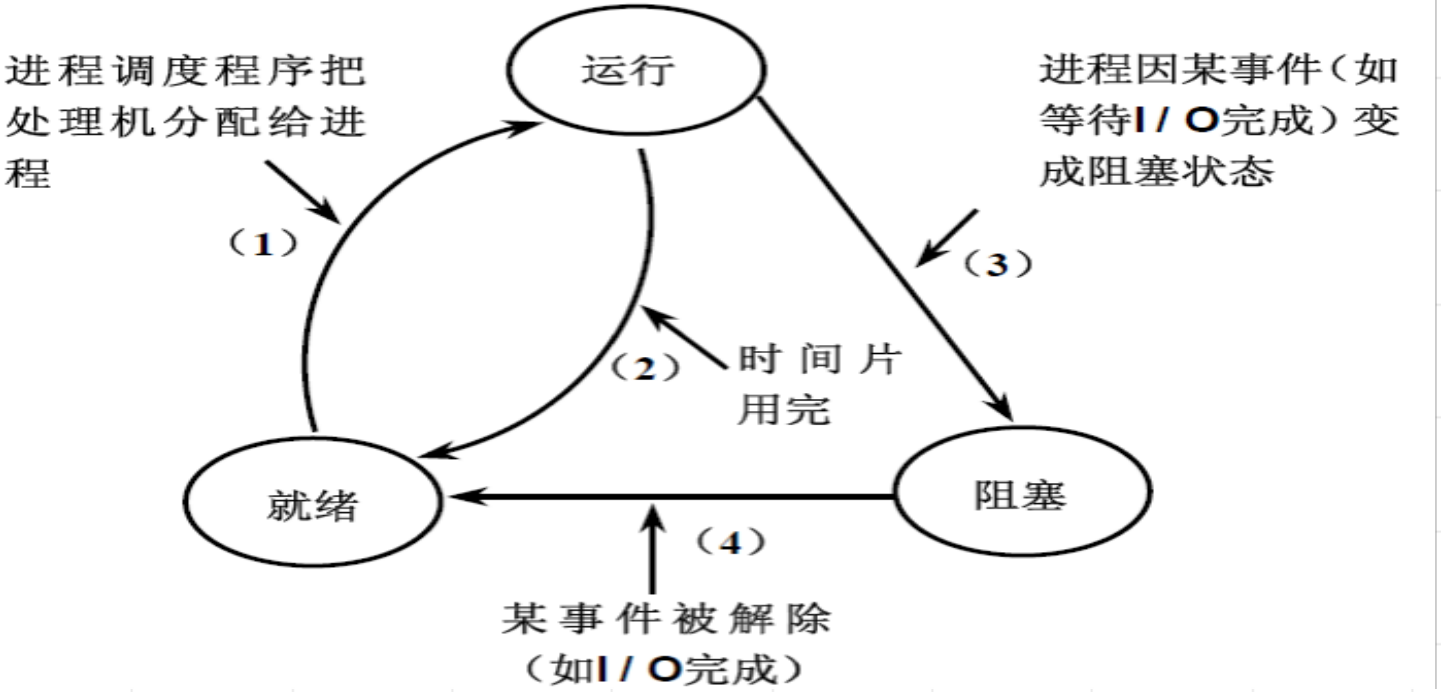
4-5



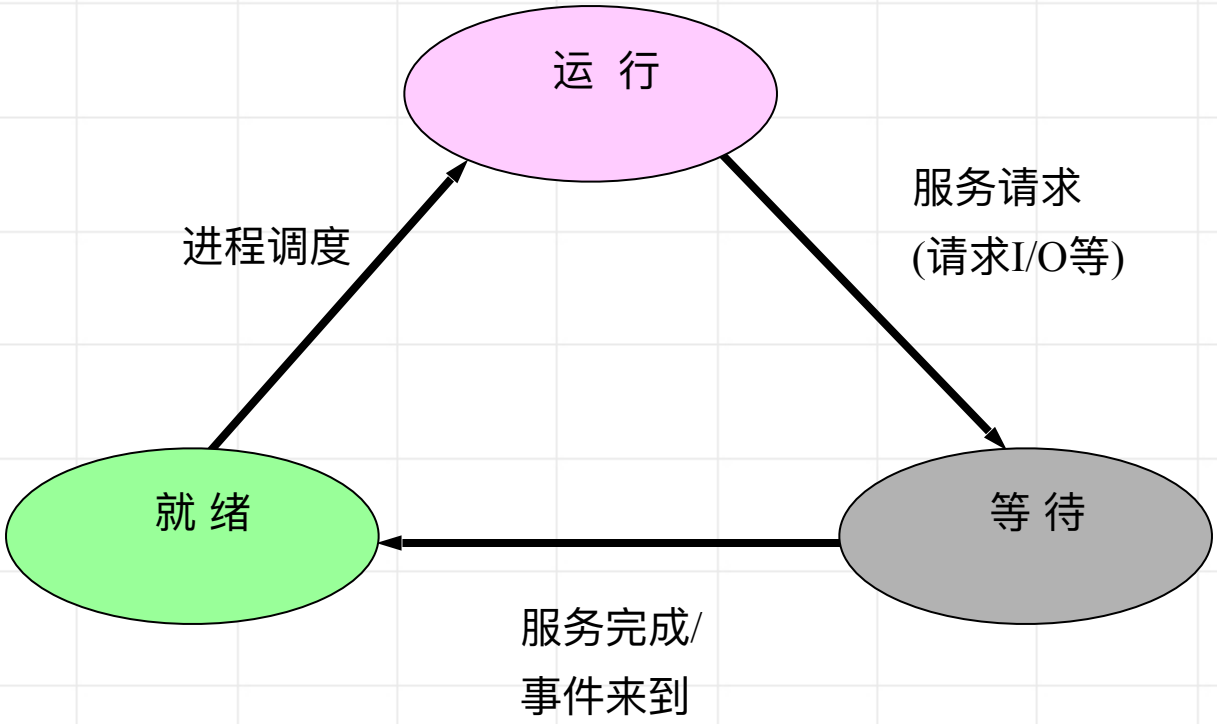
誊抄先后次序图

4-6

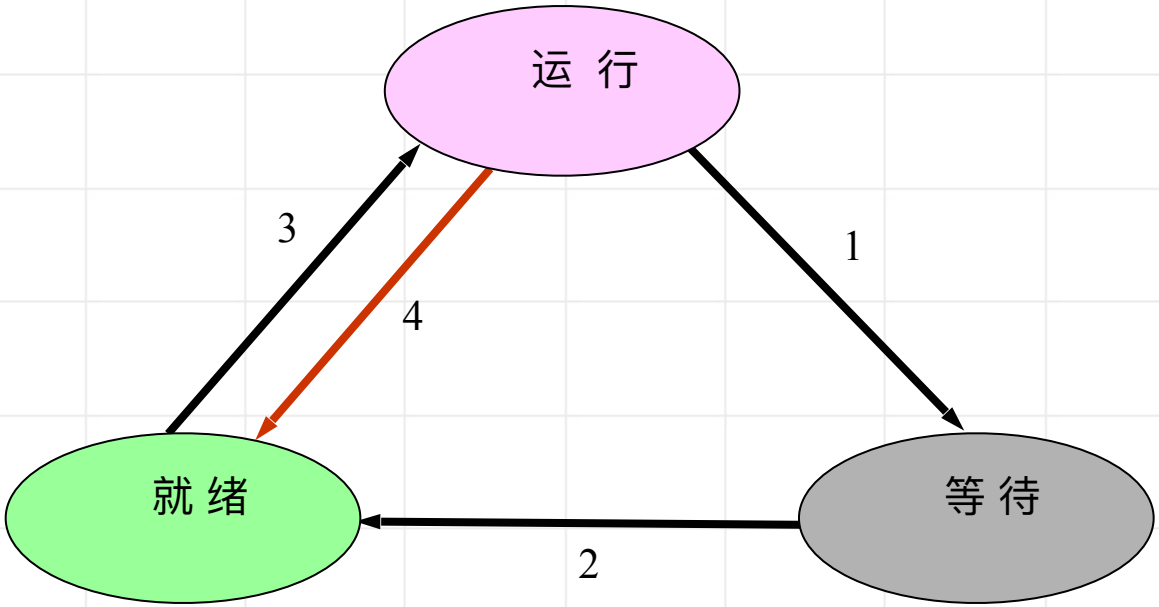
进程有三个基本状态：运行状态、就绪状态和等待状态（又称阻塞、挂起或睡眠）。



4-7



4-8



4-9

非剥夺调度方式时：

1、当运行进程在分得的时间片内未完成，时间片到将发生变迁2；当运行进程在执行过程中，需要等待某事件的发生才能继续向下执行时会发生变迁3；当等待进程等待的事件发生，会发生变迁4。

2、在就绪队列非空时

3、a. 2->1 ；会，无条件发生

b. 3->2; 不可能；

c. 4->1;会，CPU空闲时发生；度执行的变迁1。

可剥夺调度方式?

4-12

```
main ( ){
int mutex=1; //Q的互斥信号灯
cobegin
        P1();
        P2();
        …
        Pn();
coend
}
```

```
Pi(){
        …
        P(mutex);
        使用Q;
        Q(mutex);
        …
}
```

mutex取值范围[–(n–1),1]。

4-13

(a)解：

```
main(){
int s12=0,s13=0,s14=0; //分别表示
        P2、P3、P4进程能否开始执行。
cobegin
P1();
P2(;)
P3();
P4();
coend
}
P1(){
p1 execute;V(s12);V(s13);
V(s14);
}
```

```
P2(){
P(s12);
p2 execute;
}

P3(){
P(s13);
p3 execute;
}

P4(){
P(s14);
p4 execute;
}
```

4-13

(b)解：

```
main(){
int s1=0,s2=0; //分别表示P1、P2是否完成
cobegin
P1();
P2();
P3();
coend
}

P5(){
P1  execute;
V(s1);
}

P2(){
P2 execute;
V(s2);
}

P3(){
P(s1);
P(s2);
P3 execute;
}
```

4-14

```
Main()
{ int s2=s3=s4=S5=0;//分别表示P2、P3、P4、P5能否开始
        执行
 cobegin
    p1();p2();p3();p4();p5();
Coend
}
P1() {      p2() {      p3() {    p4() {    p5()  {
…..           P(S2)    P(S3)      P(S4)     P(S5)
V(S2)          …..       ……        ……       P(S5)
V(S3)      }           V(S5)     V(S5)     ……
V(S4)}                }           }          }
```

4-15

sb:缓冲区s中是否有空，初值为1；

tb: 缓冲区t中是否有空，初值为1；

sa:缓冲区s中是否有数据，初值为0；

ta: 缓冲区t中是否有数据，初值为0；

```
main()
{ int  sa, ta, sb, tb;
  sa=ta=0;
  sb=tb=1;
  cobegin
    get();
    copy();
    put();
  coend
  }
```

```
get()
{while(f不为空)
   { 从f中读取一个数据;
     p(sb);
     送数据到s中;
     v(sa);
     }
  }
```

```
copy()
{
  while(未完成)
   {  p(sa);
      p(tb);
      t=s;
      v(ta);
      v(sb);
      }
```

```
put()
   {while(写操作未完成)
        {p(ta);
          从t中取出数据;
          v(tb);
          打印数据;
          }
      }
```

4-30

```
生产线A() {
    while (1) {
        生产一个产品;
        P(Empty_F1);
          P (mutex_F1);
    零件送F1;
          V (mutex_F1);
         V(Full_F1);
        }
}
生产线B () {
    。。。。。。
}
```

```
组装( )
{
   while (1) {
      P(Full_F1);
      P (mutex_F1) ;
      取一个零件A;
      V (mutex_F1);
      V(Empty_F1);
      P(Full_F2);
      P (mutex_F2) ;
      取一个零件B;
      V (mutex_F2);
      V(Empty_F2);
     组装并编号;
      P (Empty_F3);
      P(mutex_F3);
      放入F3;
      V (mutex_F3);
      if (编号为奇数)
        V (Full_C1);
       else V(Full_C2);
   }
```

```
传输1 ()
{ while (1) {
      P(Full-C1);
      P(mutex_F3);
      取偶数号产品;
      V (mutex_F3);
      V(Empty_F3);
      传输;
}}
传输2 ()
{ while (1) {
      P(Full-C2);
      P(mutex_F3);
      取奇数号产品;
      V (mutex_F3);
      V(Empty_F3);
      传输;
}}
```

```
Main()
{
 int  Empty_F1=N1;
int  Full_F1=0;
int mutex_F1=1;
  。。。。。
  cobegin
     。。。。
  coend
}
```

4-32

```
P1()  {
      while  (1)  {
            tmp=读数据;
            if   (tmp > 0)  {
                  P(Empty_B2);
                  tmp送B2;
                  V(Full_B2);
            }  else  {
                  P(Empty_B1);
                  tmp送B1;
                  V(Full_B1);
            }
      }
}
```

```
P2( )
{
    while (1)  {
        P(Full_B1);
        tmp=从B1取数据;
        V(Empty_B1);
       加工数据，结果送tmp;
        P(Empty_B2);
        tmp送B2;
        V(Full_B2);
    }
}

P3  ()
{
    while (1)  {
        P(Full_B2);
        取数据;
        V（Empty_B2);
        输出;
}
}
```

```
Main()
{
   int   Empty_B2=1;
   int   Full_B2=0;
   int   Empty_B1=1;
  int    Full_B1=0;
   cobegin
        P1();
        P2();
        P3();
   coend
}
```

4-34

```
Enter()  {
        P(num);
        P(mutex);
        通过通道;
        V(mutex);
        找车位停车;
}
```

```
Exit ( )
{
        P  (mutex);
        通过通道;
        V(mutex);
        V(num);
}
```

```
Main()
{
   int   num=N;    //…
   int   mutex=1;   //…….

   cobegin
      Enter();
      Exit ();
   coend
}
```

5-5

(1) FCFS

下磁道	移道数
86	143-86=57
147	61
91	56
177	86
94	83
150	56
102	48
175	73
130	45
总道数	555
平均	61.7

5-5

(2) SSTF

下磁道	移道数
147	4
150	3
130	20
102	28
94	8
91	3
86	5
175	89
177	2
总道数	162
平均	18

5-5

(3) SCAN

下磁道	移道数
147	4
150	3
175	25
177	2
130	47
102	28
94	8
91	3
86	5
总道数	125
平均	13.9

5-5

(4) CSCAN

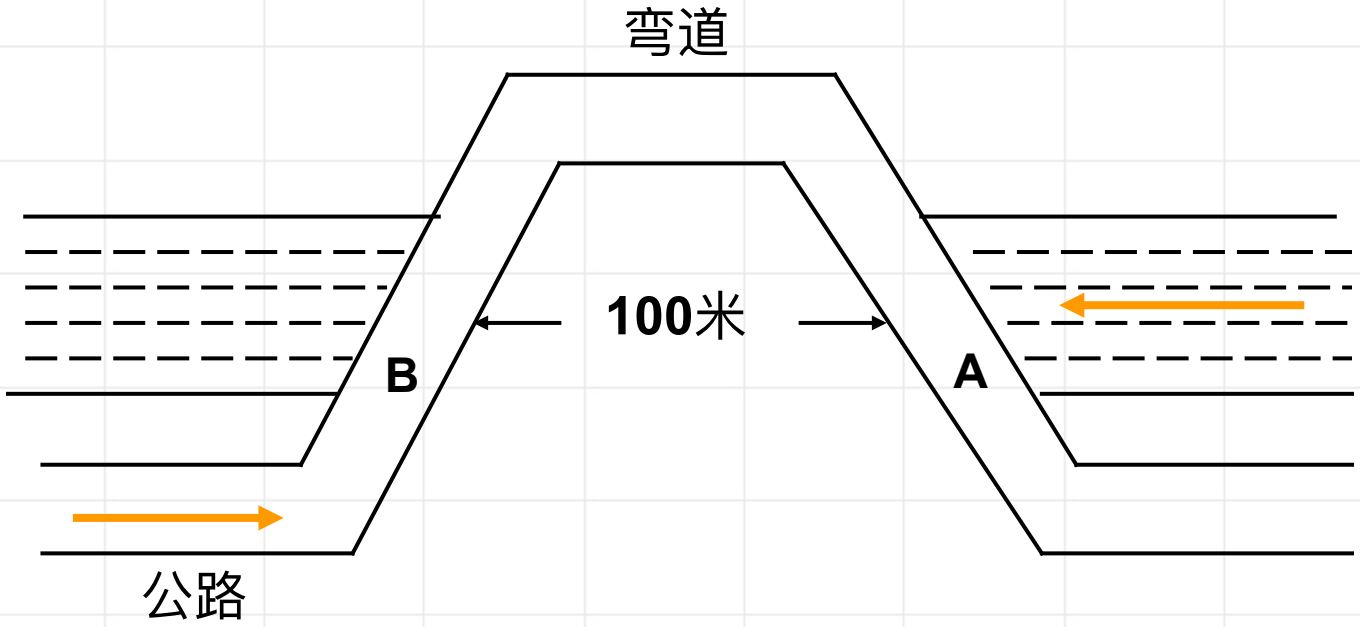
下磁道	移道数
147	4
150	3
175	25
177	2
86	91
91	5
94	3
102	8
130	8
总道数	149
平均	16.6

5-10

不失一般性，假设k(k<=p)个进程发生了死锁，则K个进程占用了m个资源，且每个进程至少还需要1个资源，没有参与死锁的P-K个进程至少需要一个资源，则P个进程对资源的最大需要数目之和大于等于m+p-k+k=m+p,矛盾。

4. 死锁问题

- 5-11



```
Ship()
{
    .......;
    P(sa);
    过吊桥A;
    P(sb);
    过吊桥B;
    V(sa);
    V(sb);
    .......;
}
```

```
Car()
{
    .......;
    P(sb);
    过吊桥B;  ....过得去吗?
    V(sb);
    P(sa);
    过吊桥A;
    V(sa);
    .......;
}
```

这是个错误的答案

```
Ship()
{
    .......;
    P(sb);
  P(sa);
    过吊桥A;
    过吊桥B;
    V(sa);
    V(sb);
    .......;
}
```

```
Car()
{
    .......;
    P(sb);
    过吊桥B;
    V(sb);
    P(sa);
    过吊桥A;
    V(sa);
    .......;
}
```

正确

方案1：令两个吊桥只能同时吊起或放下

```
Ship()
{
    ……;
    P(s);
    过吊桥A;
    过吊桥B;
    V(s);
    ……;
}
```

```
Car()
{
    ……;
    P(s);
    过吊桥B;
    过吊桥A;
    V(s);
    ……;
}
```

正确，但并发度低

```
Ship()
{
    .......;
    P(sb);
  P(sa);
    过吊桥A;
    过吊桥B;
    V(sa);
    V(sb);
    .......;
}
```

```
Car()
{
    .......;
    P(sb);
  P(sa);
    过吊桥B;
    过吊桥A;
    V(sb);
    V(sa);
    .......;
}
```

比上面的答案还差

方案2

- 为提高并发度，同类进程之间不必互斥。如：驳船2不必等待正在通过的驳船1做完V操作后，才有可能获得通过的资格。而是：只要吊桥是吊起的（即当前有一艘驳船获得了通过资格），则后来的驳船不必做P操作也的那艘船负责释放资源；汽车亦能通过，由最后通过然。
- 因此对正在通过的汽车和驳船分别设置一个计数器和相应的互斥信号灯。

```
Main()
{
    int s=1;          /* 吊桥是否能通过 */
    int ship_n=1;         /*正在在过桥的驳船数*/
    int mutex1=1;         /*ship_n的互斥信号灯*/
    int car_n=0; /*正在在过桥的汽车数*/
    int mutex2=1;         /*car_n的互斥信号灯*/
    cobegin
       ship();
       car();
    coend;
}
```

```
Ship()
{
    ……;
    P(mutex1);
    if ship_n=0 then
    P(s);
    ship_n++;
    V(mutex1);
    过吊桥A 和B;
    P(mutex1);
    ship_n--;
    if ship_n=0 then
    V(s);
    V(mutex1);
    ……;
}
```

```
Car()
{
    ……;
    P(mutex2);
    if car_n=0 then
    P(s);
    car_n++;
    V(mutex2);
    过吊桥B 和A;
    P(mutex2);
    car_n--;
    if car_n=0 then
    V(s);
    V(mutex2);
    ……;
}
```

并发度任然低

```
Ship()
{
    ……;
    P(mutex1);
    if ship_n=0 then
    { P(sb); P(sa);}
    ship_n++;
    V(mutex1);
    过吊桥A 和B;
    P(mutex1);
    ship_n--;
    if ship_n=0 then
    { V(sb); V(sa);}
    V(mutex1);
    ……;
}
```

```
Car()
{
    ……;
    P(mutex2);
    if car_b=0 then
    P(sb);
    car_b++;
    V(mutex2);
    过吊桥B ;
    P(mutex2);
    car_b--;
    if car_b=0 then
    V(sb);
    V(mutex2);
}
```

```
    ……;
    P(mutex3);
    if car_a=0 then
    P(sa);
    car_a++;
    V(mutex3);
    过吊桥A ;
    P(mutex3);
    car_a--;
    if car_a=0 then
    V(sa);
    V(mutex3);
    ……;
```

好答案

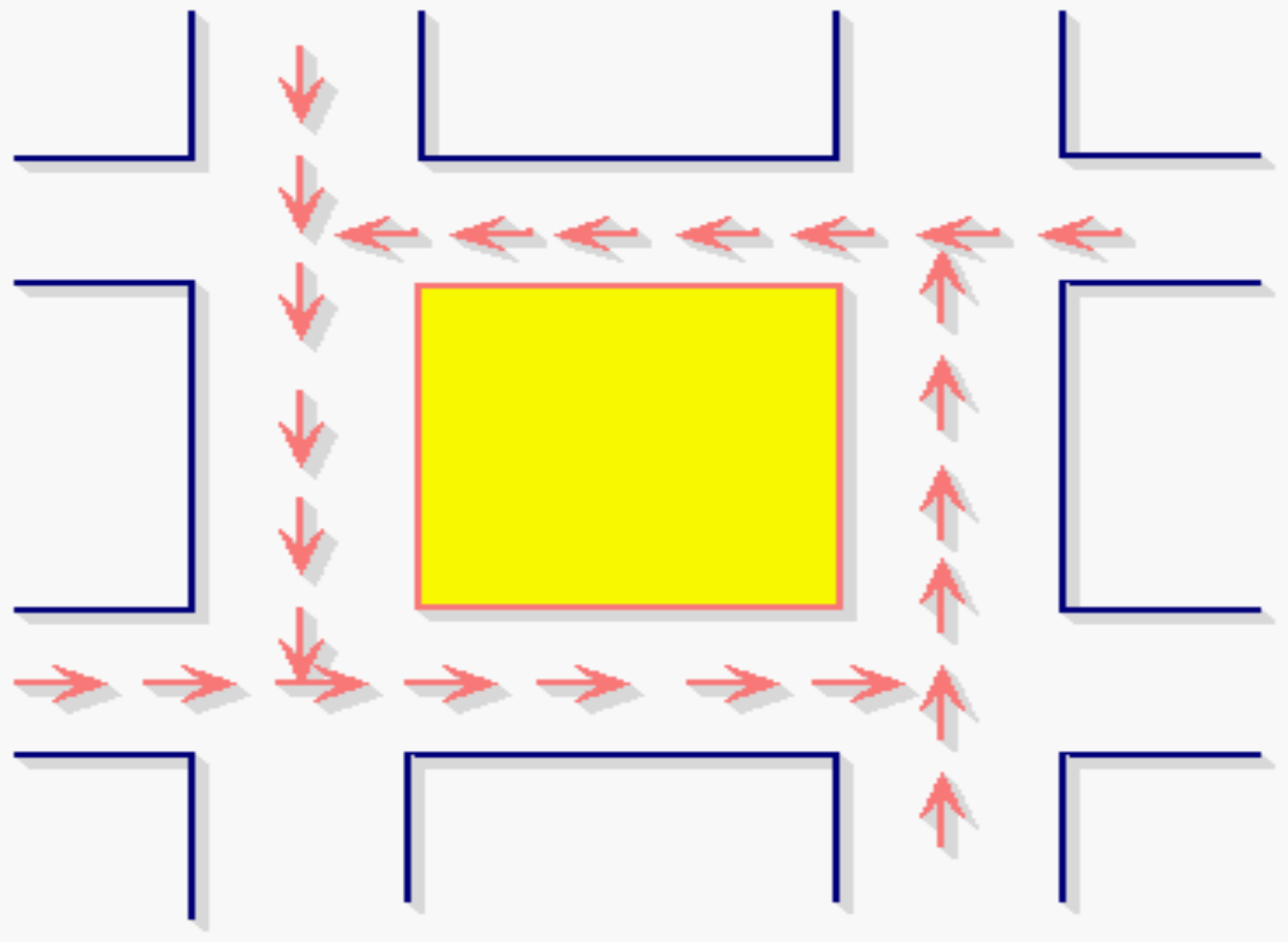
方案3

- 为提高并发度，考虑充分利用吊桥间弯道的空间。即当驳船在过桥A时，允许汽车通过桥B，当驳船需通过桥B时，这些汽车可暂时停在弯道上，等驳船通过后再过桥A。
- 由于弯道的空间有限，因此汽车在过桥B时要先考察弯道是否有空位让其停车。

```
Main()
{
    int sa=1;          /* 吊桥A是否能通过 */
    int sb=1;          /* 吊桥B是否能通过 */
    int park=n;        /* 弯道上的停车空间 */
    cobegin
        ship();
        car();
    coend;
}
```

```
Ship()
{
    .......;
    P(sa);
    过吊桥A;
   P(sb);
    过吊桥B;
    V(sa);
    V(sb);
    .......;
}
```

```
Car()
{
    .......;
   P(park);
    P(sb);
    过吊桥B;....过得
    去?
    V(sb);
    P(sa);
    过吊桥A;
    V(sa);
   V(park);
    .......;
}
```



```
int A=B=C=D=1;/*路口是否为空*/
main( )
{
        cobegin
                AC(); CD();DB();BA();
                coend;
    }
```

P118 5-7

设左上方为桥A，右上方为桥B，左下为桥C，右下为桥D,分别设立互斥信号灯A，B， C，D，初值都为1

```
AC( )          CD( )          DB()           BA()
{              {              {              {
  P(A)           P(C)           P(D)           P(B)
  过A            过C            过D            过B   ……过得去吗?
 V(A)           V(C)           V(D)           V(B)
  P(C)           P(D)           P(B)           P(A)
  过C            过D            过B            过A   ……过得去吗?
  V(C)           V(D)           V(B)           V(A)
}              }              }              }
```

P118 5-7

设左上方为桥A，右上方为桥B，左下为桥C，右下为桥D,分别设立互斥信号灯A，B， C，D，初值都为1,采用有控分配方法(A<C<D<B)

```
AC( )          CD( )          DB()           BA()
{              {              {              {
  P(A)           P(C)           P(D)           P(A)
  过A            过C            过D            P(B)
 V(A)           V(C)           V(D)           过B
  P(C)           P(D)           P(B)           过A
  过C            过D            过B            V(B)
  V(C)           V(D)           V(B)           V(A)
}              }              }              }
```

P118 5-7

设左上方为桥A，右上方为桥B，左下为桥C，右下为桥D

以AC方向的车辆为例：

设信号灯Sac：表示A、C两桥中能停小车的辆数，初值为5

设立A、C两桥的互斥信号灯MutexA，MutextC，初值都为1。

```
Void  AC(boolean small)
{       P（Sac）；
        if（！small） /*大车*/
                P（Sac）；
        P（MutexA）；
        过A桥；
                                        V（MutexA）；
                                ……
                                        P（MutexC）；
                                        过C桥；
                                        V（MutexC）；
                                        V(Sac);
                                        if（！small） /*大车*/
                                                V（Sac）；
}
```

最多允许三个路口有车:

```
Int  count=3;mutexA=mutexB=mutexC=mutexD=1;
Void Driving_AC( ) {
            P(count);
            P(mutexA);          过A路口;       V(mutexA);
          …….
            P(mutexC); 过C路口;  V(mutexC);
            V(count);
}
Main()  { cobegin   Driving_AC();Driving_AB();……
Coend;}
```

5-13

（1）有可能发生死锁，譬如A申请3个资源，即发生了死锁

（2）不会，存在安全序列C-〉A-〉B

5-14

（1）是，存在安全序列P4-〉P3-〉P5-〉P1-〉P2

（2）P4允许，因为存在安全序列

P5拒绝，因为不存在安全序列

6-3

（1）系统服务请求、时间片到、服务完成

（2） 2-〉5：会，高优先就绪不为空

2-〉1：会，高优先就绪为空

4-〉5：会，CPU空闲

4-〉2：不是

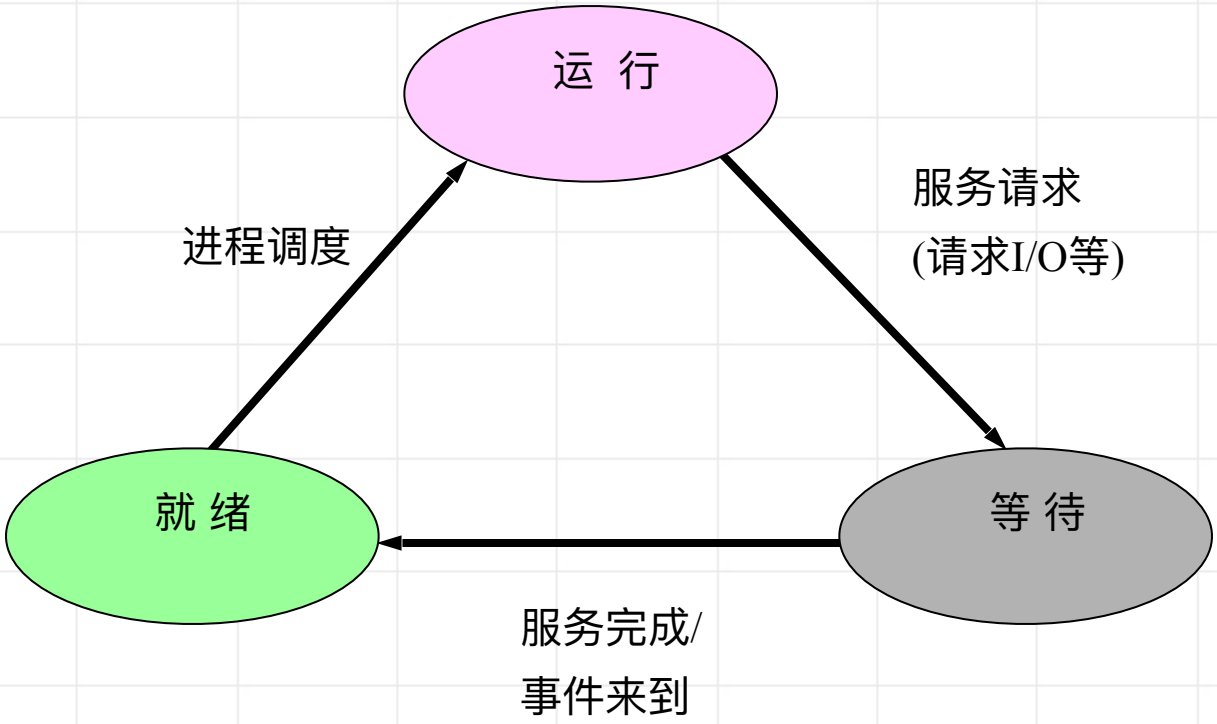
3-〉5：会，高优先就绪不为空

（3）调度策略： 优先调度与时间片调度相结合的调度算法，当CPU空闲时，若高优先就绪队列非空，则从高优先就绪队列中选择一个进程运行，分配时间片为100ms；当CPU空闲时，若高优先就绪队列为空，则从低优先就 列中选择一个进程运行，分配时间片为500ms。

调度效果：优先照顾I/O量大的进程；适当照顾计算量大的进程。

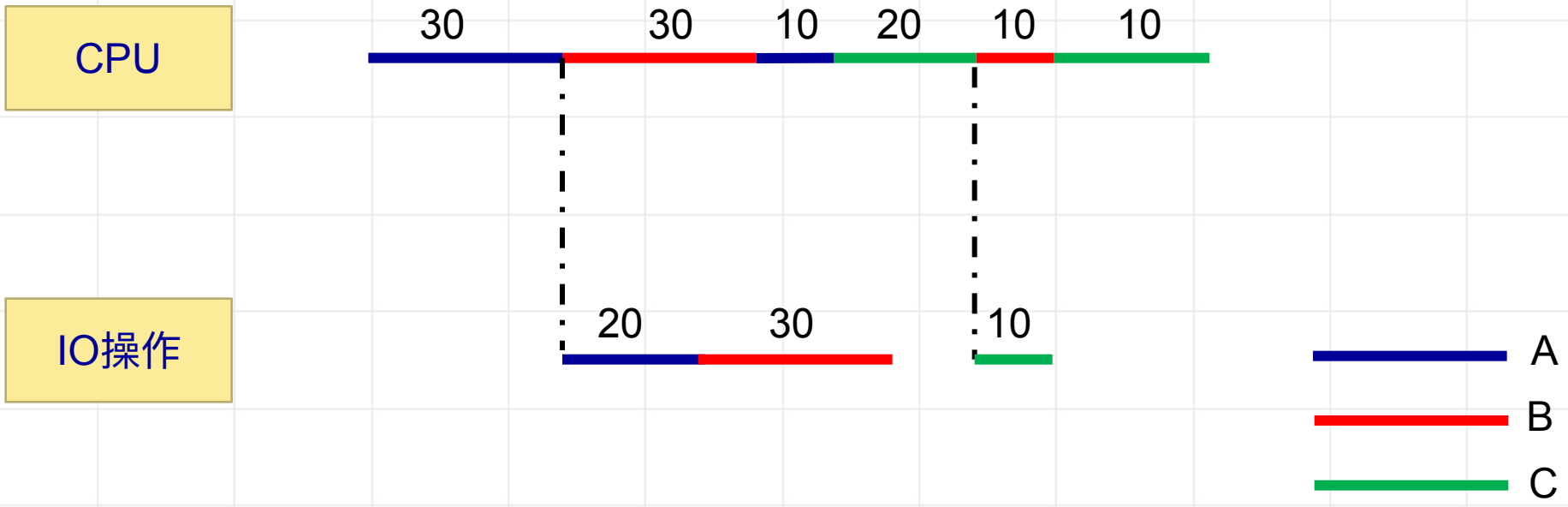
6-11

(1)



(2) A --〉 B --〉 A --〉 C --〉 B --〉 C

30ms、30ms、10ms、20ms、10ms、20ms



6-12

（1） FIFO：P1-〉P2-〉P3-〉P4-〉P5

优先数：P1-〉P4-〉P3-〉P5-〉P2

（2）略，平均等待时间分别为9.6、10.4