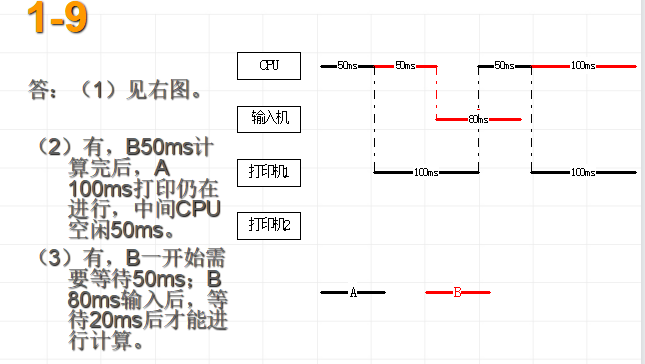
1-9 设一计算机系统有输入机一台、打印机两台，现有A、B两道程序同时投入运行，且程序A先运行，程序B后运行。程序A的运行轨迹为：计算50ms，打印信息100ms，再计算50ms，打印信息100ms，结束。程序B运行的轨迹为：计算50ms，输入数据80ms，再计算100ms，结束。回答如下问题。

（1）用图画出这两道程序并发执行时的工作情况。

（2）说明在两道程序运行时，CPU 有无空闲等待?若有，在哪段时间内等?为什么会空闲等待?

（3）程序 A、B 运行时有无等待现象?在什么时候会发生等待现象?



4-9 某系统进程调度状态变迁图如图4.31所示，请说明：

运 行

就 绪

等 待

4

1

2

3

(1) 什么原因会导致发生变迁2、变迁3、变迁4 ？

答：发生变迁2的原因：时间片到

发生变迁3的原因：请求I/O或其他系统调用

发生变迁4的原因：I/O完成或其他系统调用完成

(2) 在什么情况下，一个进程的变迁3 能立即引起另一个进程发生变迁1 ？

答：一个进程的变迁3 能立即引起另一个进程发生变迁的条件是，就绪队列非空。

(3) 下列因果变迁是否可能发生？若可能，需要什么条件？

a. 2→1； b. 3→2； c. 4→1

答：a. 2→1 不需要条件，一定会发生。

b. 3→2 不可能发生。

c. 4→1 可能发生，条件：就绪队列为空，或在可剥夺调度方式下，转变为就绪状态的进程优先级最高。

4-29某公园有一个长凳，其上最多可以坐5个人。公园里的游客遵循以下规则使用长凳：

(1) 如果长凳还有空间可以坐，就坐到长凳上休息，直到休息结束，离开长凳。

(2) 如果长凳上没有空间，就转身离开。

试用信号灯的P、V操作描述这一场景。

答：

main(){

int count=5; // 长凳上可坐的人数

int mutex=0; // 访问count的互斥信号灯

cobegin

guesti; // i =1,2,3,4……

coend

}

guesti {

P(mutex);

if (count == 0) {

V(mutex);

return;

}

count--;

V(mutex);

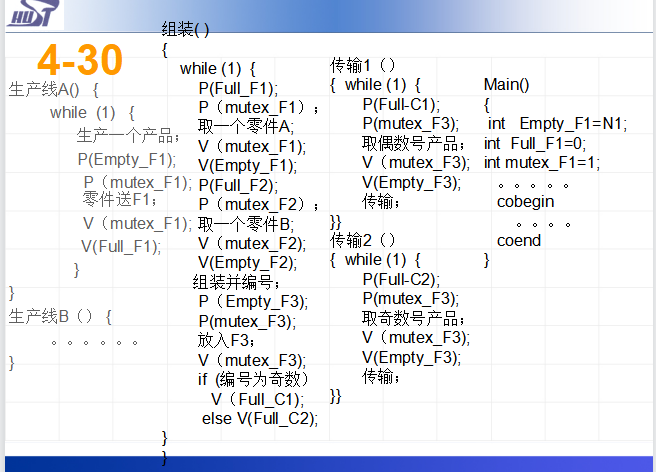
休息；

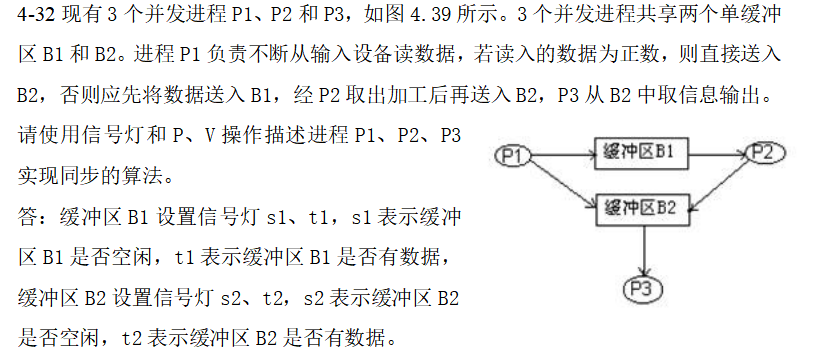
P(mutex);

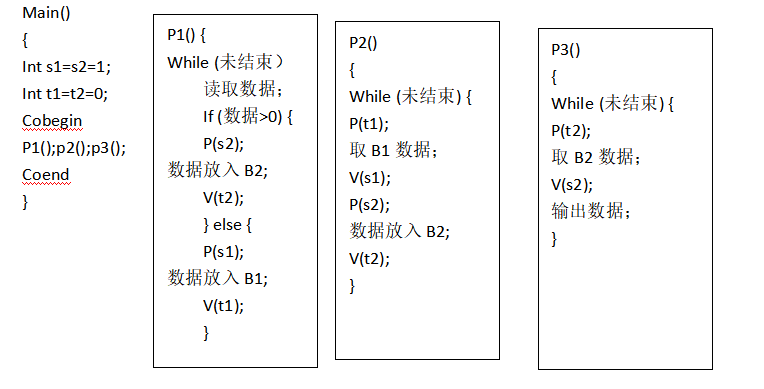
Count++;

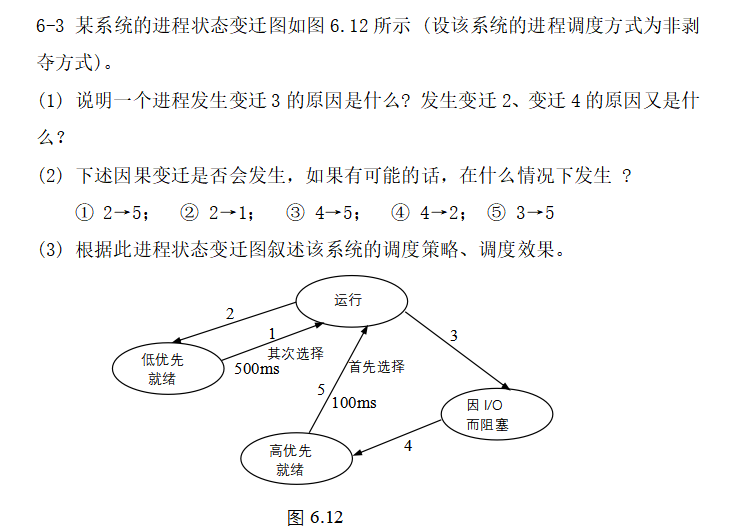
V(mutex);

}









答：(1) 发生变迁 3 的原因：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。

发生变迁 2 的原因：运行进程在分得的时间片 100ms 或 500ms 内未完成，当其时间片到时会发生变迁 2。

发生变迁 4 的原因：当等待进程等待的事件发生了，将会发生变迁 4。

(2)

① 2→5 可能发生。当高优先就绪队列非空，就会发生。

② 2→1 可能发生，当运行进程的时间片到时发生的变迁 2，若此时高优先就绪队列为空，必然引起低优先就绪队列中的一个就绪进程被调度执行而发生变迁 1。

③ 4→5 不可能发生，因为采用的是非剥调度夺式。

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候会发生。

(3) **调度策略**：首先调度高就绪队列中的进程 (一般是 I/O 型进程) 投入运行，给高优先就绪队列中的进程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因等待某事件发生处于阻塞状态，高就绪队列中没有进程可运行时，才调度低优先就绪队列中的进程 (一般是计算型进程) ，给低优先就绪队列中的进程分配的时间片大小为 500ms。若一个运行进程时间片 100ms 或500ms 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，则当所等待事件完成后，它将进入高优先就绪队列。

**调度效果：**这种算法优先照顾了 I/O 量大的进程 (高优先级) ，但通过给计算型进程分配更长的时间片也适当照顾了计算型进程。

5-11图5.9 表示一带闸门的运河，其上有两架吊桥，吊桥坐落在一条公路上，为使该公路避开一块沼泽地而其横跨运河两次。运河和公路的交通都是单方向的，运河的基本运输由驳船担负。在一艘驳船接近吊桥A 时就拉汽笛警告，若桥上无车辆，吊桥就吊起，直到驳船尾部通过该桥为止，对吊桥B按同样次序处理

(1) 一艘典型驳船的长度为200 米，当它在河道航行时是否会产生死锁？若会，其理由是什么？

(2) 如何能克服一个可能的死锁？请想出一个防止死锁的办法。

(3) 如何利用信号灯的P、V 操作实现车辆和驳船的同步？



图5.9

(1) 答：驳船长200 米，当驳船通过了A 桥，其船头到达B 桥，请求B 桥吊起，而此时它的尾部占据A 桥，若这个时候B 桥及B桥到A 桥之间的公路都被汽车占据，而汽车又要求通过A 桥。这样驳船和汽车都无法前进，形成死锁的局面。

(2) 答：方案之一。可规定资源按序申请和分配，从而破坏了死锁的循环等待条件，防止死锁的发生。规定如B 桥的序号小于A 桥的序号，驳船和汽车都必须先申请序号小的资源B 桥，申请得到满足后，再申请序号大的资源A 桥。

(3) 答：将每台车的行驶看作是进程，则有Auto1，Auto2，Autoi i个汽车进程。将每条驳船的航行看作是进程，则有Ship1，Ship2，Shipj个驳船进程。桥A和桥B对车和船为互斥资源。

方案1：

main{

int SA=1；//A桥的互斥信号量//

int SB=1；//B桥的互斥信号量//

cobegin

Auto1;Auto2;Autoi;

Ship1; Ship2; Shipj;

coend

}

Autoi(){

车在公路上行驶；

P（SB）；

过B桥；

V（SB）；

过弯道；

P（SA）；

过A桥；

V（SA）；

车在公路上行驶；

}

Shipj(){

运河航行；

P（SB）；

P（SA）；

吊起过A桥；

运河航行；

吊起过B桥；

V（SA）；

V（SB）；

运河航行；

}

方案2：方案1的缺点是船在过A桥前需要先占用B桥，使桥的通过率降低，增加交通阻塞的可能。因此方案2将弯道作为有界缓中区，基本思想是只要弯道有空，车就可以通过B桥，进入弯道。而船则不用先占用B桥，使桥的通过率降低。

main{

int SA=1；//A桥的互斥信号量

int SB=1；//B桥的互斥信号量

int Sn=n；//弯道可容纳汽车数n

cobegin

Auto1;Auto2;Autoi;

Ship1; Ship2; Shipj;

coend

}

Autoi(){

车在公路上行驶；

P（Sn）；

P（SB）；

过B桥；

V（SB)；

过弯道；

P（SA）；

上A桥；

V（Sn）；

过A桥；

V（SA）；

车在公路上行驶；

}

Shipj(){

运河航行；

P（SA）；

船头行驶至B桥；

P（SB）；

运河航行；

船尾过A桥；

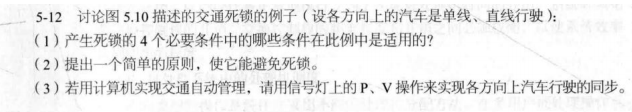
V（SA）；

船尾过B桥；

V（SB）；

运河航行；

}



解：（1） 路口是共享资源。

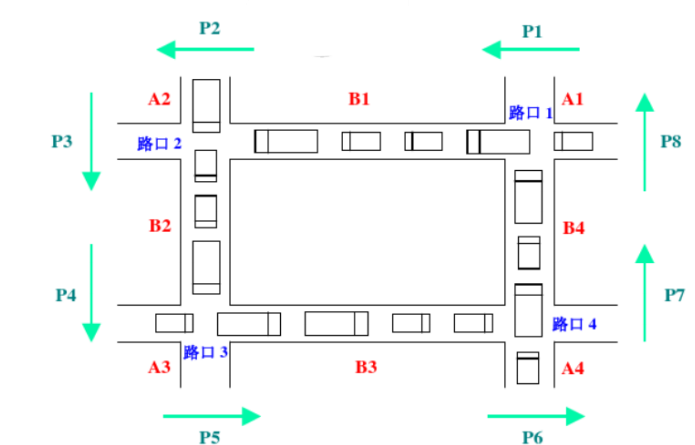
1. 互斥条件：路口必须互斥使用。因为汽车对它需要的路口是排他性控制的。

2. 不剥夺条件：汽车一旦占用路口，除非自己让开，没法剥夺。

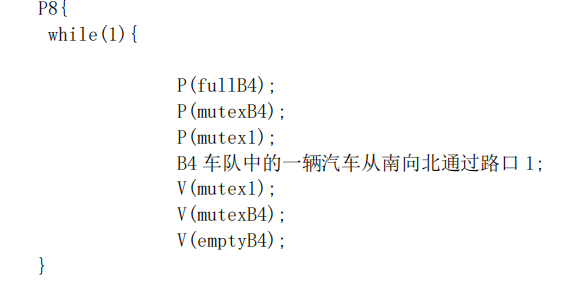
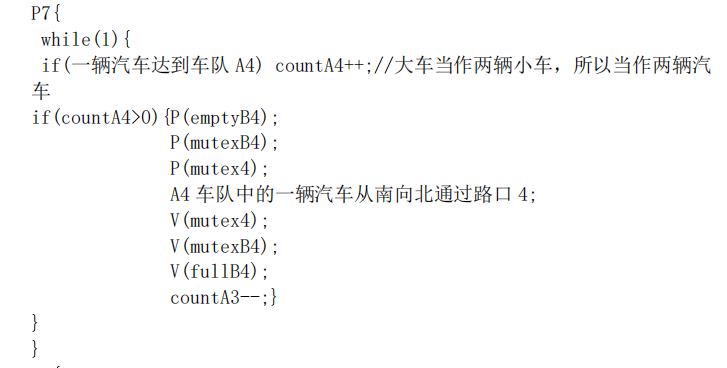
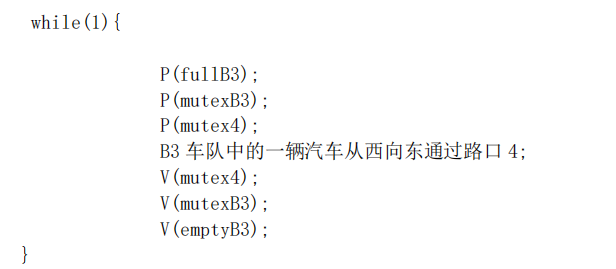
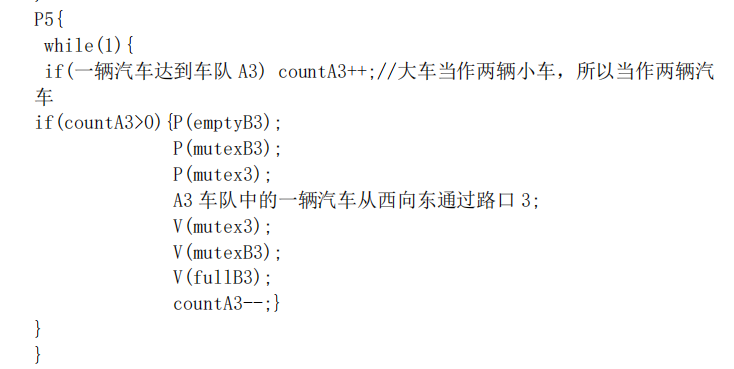
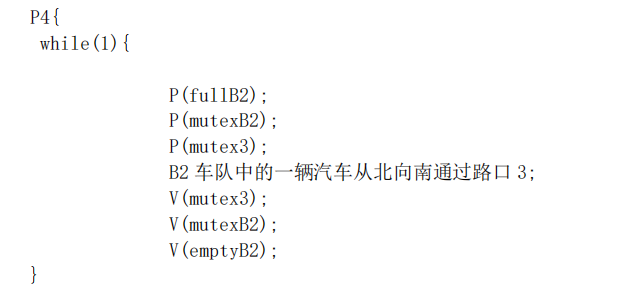
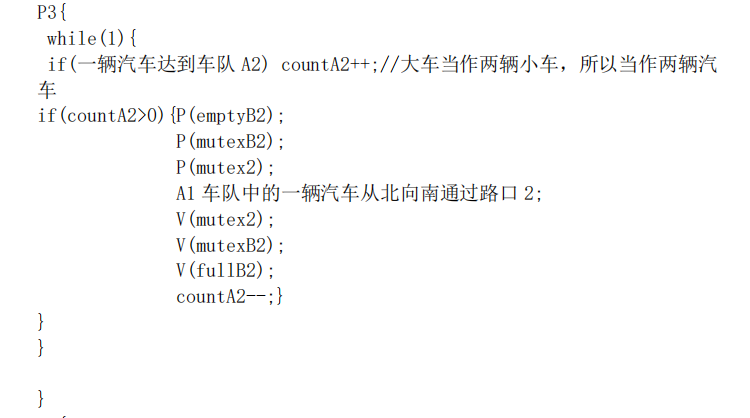
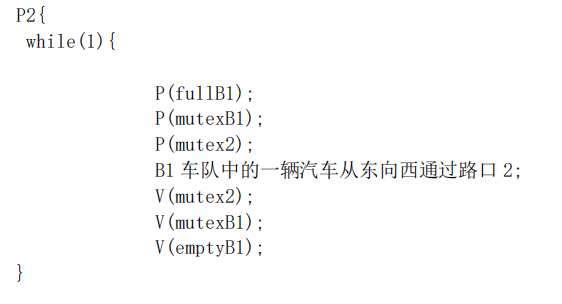
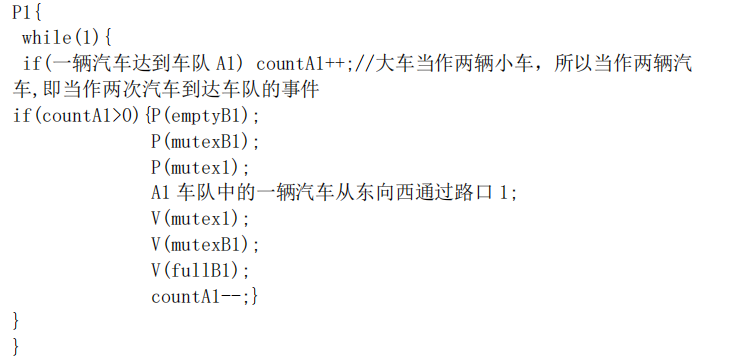
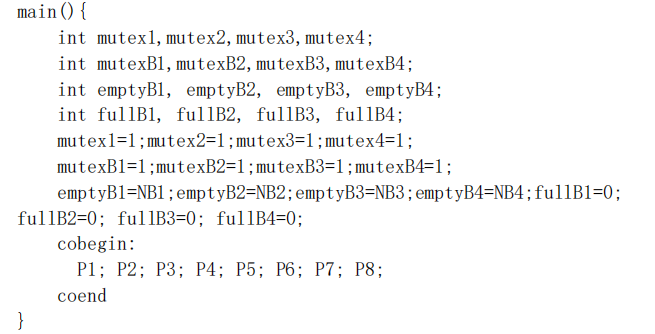
3. 部分分配条件：每个方向的车队都占有一个路口，同时为申请新路口而等待 排队中。

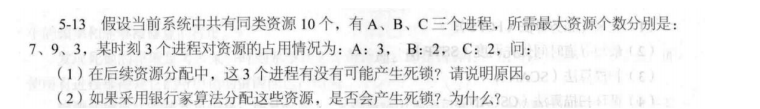
4. 环路等待条件：占领路口的车都在等待其他车占有的路口，循环等待中。

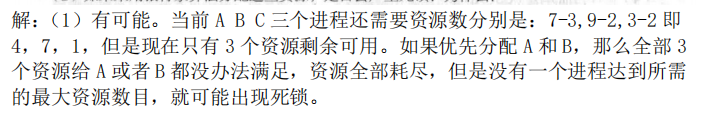
（2）可以在每个路口设置红绿灯。当绿灯亮时候，南北方向车队可以通行，东西方向禁止。红灯亮时候，南北方向车队禁止通行，东西方向允许通行。

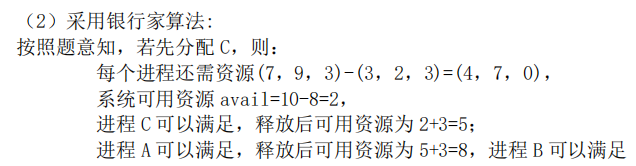
（3）

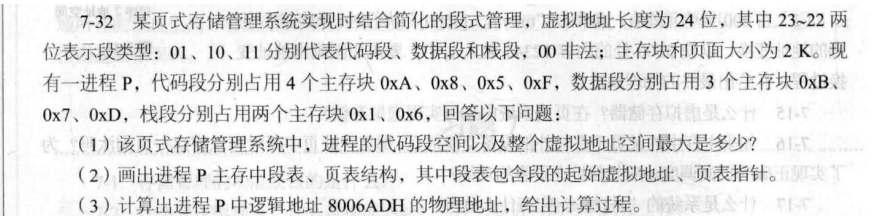
图中，大车长度是小车的两倍。我们就把大车当成两辆小车来处理，即两辆汽车经过路口。其中:A1 表示从东向西预通过路口 1 的车队;B1 表示位于路口 1 和路口 2 之间的车队，即从东向西预通过路口 2 的车队;A2 表示从北向南预通过路口 2 的车队;B2 表示位于路口 2 和路口 3 之间的车队，即从北向南预通过路口 3 的车队;A3 表示从西向东预通过路口 3 的车队;B3 表示位于路口 3 和路口 4 之间的车队，即从西向东预通过路口 4 的车队;A4 表示从南向北预通过路口 4 的车队;B4 表示位于路口 4 和路口 1 之间的车队，即从南向北预通过路口 1 的 车队。P1 进程控制车队 A1 中的汽车从东向西依次通过路口 1;P2 进程控制车队 B1 中的汽车从东向西依次通过路口 2;P3 进程控制车队 A2 中的汽车从北向南依次通过路口 2;P4 进程控制车队 B2 中的汽车从北向南依次通过路口 3;P5 进程控 制车队 A3 中的汽车从西向东依次通过路口 3;P6 进程控制车队 B3 中的汽车从西向东依次通过路口 4;P7 进程控制车队 A4 中的汽车从南向北依次通过路口 4;P8 进程控制车队 B4 中的汽车从南向北依次通过路口 1。其中，P1、P3、P5、P7 相当于 4 个生产者，P2、P4、P6、P8 相当于 4 个消费者，4 个路口之间的 4 段公路相当于 4 个有限缓冲区。因此需要设置 4 对同步信号量: emptyB1 和fullB1，emptyB2 和 fullB2,emptyB3 和 fullB3，emptyB4 和 fullB4。其中，emptyBi (i=1,2,3,4）表示车队 Bi 所在的公路段还能容纳多少辆汽车（假设车队 Bi 所在的公路段最多可以容纳 NBi 辆汽车，i=1，2，3，4;fullBi(i=1,2,3,4）表示车队 Bi 所在的公路段上现在已有的汽车数。显然，emptyBi(i=1,2,3,4）的初始值等于 NBi (i=1,2,3,4)，fullBi (i=1,2,3,4）的初始值等于 0。除此之外，还设置 4 个互斥信号灯 mutexBi(i=1、2、3、4)，用来实现对 i 车队的互斥使用，初始值均为 1。还需要设置 4 个互斥信号灯mutexi (i=1、2、3、4)，用来实现汽车对路口 i 的互斥使用，其初始值均为1。另外，还设置了 4 个计数器 countA1，countA2，countA3，countA4 用来分别记录当前 A1 车队、A2 车队、A3 车队和 A4 车队中的汽车数，初始值均为 0。











（1）答： 代码段空间：2^22 ，整个虚拟地址空间 2^24

（2）

