# 第 5 章作业

## 5-11、

（1）会产生死锁。由于驳船长度为 200m，而两桥中间为 100m，可知，当驳船经过 A 桥后，A 桥被船占用，且当船到达 B 桥时依然占用。如果此时 B 桥和弯道全被车占用，那么船将无法占用 B 桥。此时车也无法通过 A 桥通行，产生死锁。
（2）资源分配前，查看下一状态是否可能是死锁状态，若是则阻止进程请求发生。具体实现是，对车进行约束，当通过 B 桥时，查看弯道是否有空位，若有，则通过 B 桥；若没有，则停下等待。
（3）实现如下：

```
main(){
    int sA=1;  /*表示A能否被占用*/
    int sB=1;  /*表示B能否被占用*/
    int sn=n;  /*表示弯道空余车位*/
    cobegin
      car1();car2();...;cari();...
      ship1();ship2();...;shipi();...
    coend
}
```

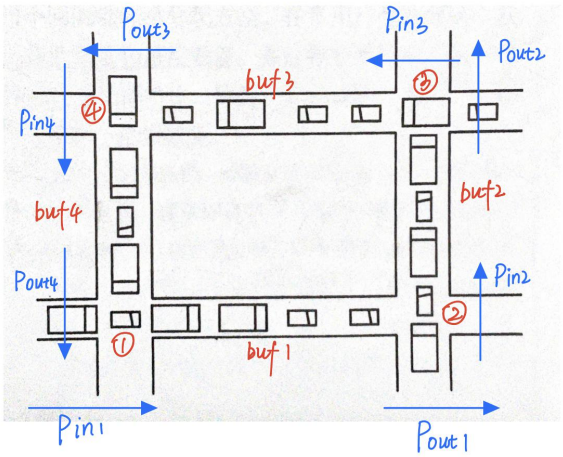
```
cari(){
    在公路行驶;
    p(sn);
    p(sB);
    占用并通过桥B;
    v(sB);
    通过弯道;
    p(sA);
    v(sn);
    占用并通过桥A;
    v(sA);
    在公路上行驶;
}
```

```
shipi(){
    在河道上航行;
    p(sA);
    继续航行到桥B;
    p(sB);
    继续航行到船尾过桥A;
    v(sA);
    继续航行到船尾过桥B;
    v(sB);
    在河道上航行;
}
```

## 5-12、

（1）
①互斥条件：路口是互斥的，一次只能被一辆车使用；
②不剥夺条件：路口在被使用时，其他车辆无法使用，不能剥夺路口的使用权；
③部分分配：占用路口的车在占用路口的同时，申请下一个路口的使用；
④环路条件：占用路口的车在等待其他有车占用的路口。
（2）
联系现实不难发现，交通信号灯可以解决这种情况，绿灯亮时，南北通行；红灯亮时，东西通行。若不采用额外的装置，则对不同路线上的路口可以分成入路口和出路口，出路口的优先级大于入路口即可，即优先让出路口的车离开这个路口系统。

（3）




如图所示，进程划分为 8 个，即 4 个进路口进程和 4 个出路口进程，分别控制路口车辆的运行。为每个路口设置互斥量 Rmutexi。路口中间路段设为车缓冲区 Cbufi，由于车长不一，为方便考虑，设其初值为最多容纳大车的数量 ni。对 Cbufi 的进出有 Cfulli 与 Cemptyi 指示，访问缓冲区时设置互斥 Cmutexi。

在进路口进程中，获取路口使用权前，还需要获取车辆缓冲区空间，只有在 buf 区不满时，进入路口。而出路口进程，在获取路口使用权前，只需要得知 buf 不为空即可。由此可知，进入口与出入口之间为生产者-消费者关系。在获取路口权限前，受到生产者-消费者关系的制约。消除了部分分配的死锁必要条件，故不会产生死锁。

```
main(){
    int Rmutex1=Rmutex2=Rmutex3=Rmutex4=1;  /*指示4个路口的占用情况*/
    int Cempty1=n1;Cempty2=n2;Cempty3=n3;Cempty4=n4; /*指示4个车辆缓冲区buf的空车位*/
    int Cfull1=Cfull2=Cfull3=Cfull4=0;   /*指示4个车辆缓冲区已有的车数*/
    int Cmutex1=Cmutex2=Cmutex3=Cmutex4=1; /*对对应的车辆缓冲区操作进行互斥*/
    cobegin
      Pin1();Pin2();Pin3();Pin4();
      Pout1();Pout2();Pout3();Pout4();
    coend
}
```

```
Pin1(){
while(1){
    车辆到来;
    p(Cempty1);
    p(Rmutex1);
    车辆通过路口1;
    v(Rmutex1);
    p(Cmutex1);
    车辆进入车辆缓冲区1;
    v(Cmutex1);
    v(Cfull1);
}
}
```

```
Pin2(){
while(1){
    车辆到来;
    p(Cempty2);
    p(Rmutex2);
    车辆通过路口2;
    v(Rmutex2);
    p(Cmutex2);
    车辆进入车辆缓冲区2;
    v(Cmutex2);
    v(Cfull2);
}
}
```

```
Pin3(){
while(1){
    车辆到来;
    p(Cempty3);
    p(Rmutex3);
    车辆通过路口3;
    v(Rmutex3);
    p(Cmutex3);
    车辆进入车辆缓冲区3;
    v(Cmutex3);
    v(Cfull3);
}
}
```

```
Pin4(){
while(1){
    车辆到来;
    p(Cempty4);
    p(Rmutex4);
    车辆通过路口4;
    v(Rmutex4);
    p(Cmutex4);
    车辆进入车辆缓冲区4;
    v(Cmutex4);
    v(Cfull4);
}
}
```

```
Pout1(){
while(1){
    p(Cfull1);
    p(Rmutex2);
    p(Cmutex1);
    车辆离开车辆缓冲区1;
    v(Cmutex1);
    车辆通过路口2;
    v(Rmutex2);
    v(Cempty1);
}
}
```

```
Pout2(){
while(1){
    p(Cfull2);
    p(Rmutex3);
    p(Cmutex2);
    车辆离开车辆缓冲区2;
    v(Cmutex2);
    车辆通过路口3;
    v(Rmutex3);
    v(Cempty2);
}
}
```

```
Pout3(){
while(1){
    p(Cfull3);
    p(Rmutex4);
    p(Cmutex3);
    车辆离开车辆缓冲区3;
    v(Cmutex3);
    车辆通过路口4;
    v(Rmutex4);
    v(Cempty3);
}
}
```

```
Pout4(){
while(1){
    p(Cfull4);
    p(Rmutex1);
    p(Cmutex4);
    车辆离开车辆缓冲区4;
    v(Cmutex4);
    车辆通过路口1;
    v(Rmutex1);
    v(Cempty4);
}
}
```

5-13、

（1）可能产生死锁。例如，若首先满足 A 的最大需求，由于此时可分配资源只有 3 个，而 A 还需要 4 个资源，所剩余的资源无法满足资源请求，产生死锁。
（2）如果采用银行家算法分配资源，则不会产生死锁。当前剩余可分配资源 3 个，C 还需 1 个资源，先满足 C 的请求。进程 C 结束后释放资源，可分配资源变为 3+2=5 个。再满足 A 的需求，A 还需 4 个资源。进程 A 结束后释放资源，可分配资源变为 5+3=8 个。最后满足 B 的需求，B 还需 7 个资源。进程 B 结束后释放资源。故分配次序为 C-A-B。

5-14、

（1）系统处于安全状态，安全序列为 P4-P3-P5-P1-P2.
alloc = (15, 2, 17)； avail = (17, 5, 20) - alloc = (2, 3, 3)
max(4,*) - alloc(4,*) = (2, 2, 1) < (2, 3, 3)，满足 P4 进程请求，进程结束释放后，
avail = (2, 3, 3) + alloc(4,*) = (2, 3, 3)+(2, 0, 4)=(4, 3, 7)，
max(3,*) - alloc(3,*) = (0, 0, 6) < (4, 3, 7)，满足 P3 进程请求，进程结束释放后，
avail = (4, 3, 7) + alloc(3,*) = (4, 3, 7)+(4, 0, 5)=(8, 3, 12)，
max(5,*) - alloc(5,*) = (5, 1, 0) < (8, 3, 12)，满足 P5 进程请求，进程结束释放后，
avail = (8, 3, 12) + alloc(5,*) = (8, 3, 12)+(3, 1, 4)=(11, 4, 16)，
max(1,*) - alloc(1,*) = (3, 4, 7) < (11, 4, 16)，满足 P1 进程请求，进程结束释放后，
avail = (11, 4, 16) + alloc(1,*) = (11, 4, 16)+(2, 1, 2)=(13, 5, 18)，
max(2,*) - alloc(2,*) = (1, 4, 4) < (13, 5, 18)，满足 P2 进程请求，进程结束释放后，
avail = (13, 5, 18) + alloc(2,*) = (13, 5, 18)+(4, 0, 2)=(17, 5, 20).
（2）不能满足它们的请求。
P4 和 P1 提出请求后，资源剩余可分配数为(2,3,3)-(2,2,1)=(0,1,2)，此时 A 类资源为 0，只能等待其他进程释放 A 资源才能避免其他进程死锁。而不需要 A 资源的进程为 P4(0,2,0)，B 资源不满足，故 P4 无法释放；P3(0,0,6)，C 资源不满足，故 P3 也无法释放，故产生死锁。