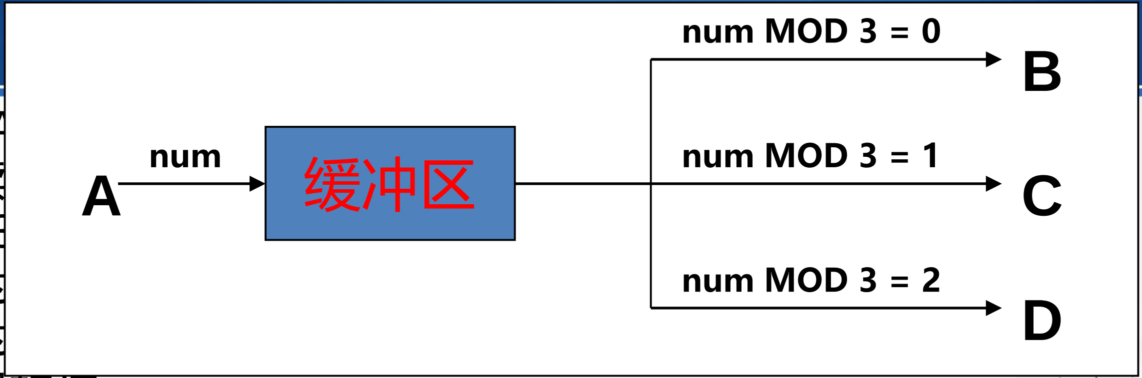
操作系统

Operating system

胡燕

大连理工大学

- 设有4个进程A
环地从文件读
环读入MOD 3
读入MOD 3为
读入MOD 3为
够正确执行的程序。



所需同步信号量：Sempty :=1 、SB、SC、SD := 0

```
Process PA
Begin
    P(Sempty);
    <读入num至缓冲区>
    if (num MOD 3 = 0)
        V(SB);
    if (num MOD 3 = 1)
        V(SC);
    else
        V(SD);
end
```

```
Process PB
Begin
    P(SB);
    <从缓冲区读入num并累计求和>
    V(Sempty);
end
```

```
Process PC
Begin
    P(SC);
    <从缓冲区读入num并累计
求和>
    V(Sempty);
end
```

```
Process PD
Begin
    P(SD);
    <从缓冲区读入num并累计
求和>
    V(Sempty);
end
```

练习题答案讨论：

某寺庙，有小，老和尚若干，由小和尚提水倒入缸供老和尚饮用。水缸可容10桶水，水取自同一井中。水井窄，每次只能容纳1个桶取水。水桶总数为3个。每次从缸中取水，向缸中倒入水仅为1桶，且不可同时进行。试给出有关取井水、入缸水，取缸水的算法。

semaphore empty=10; // 表示缸中目前还能装多少桶水，初始时能装10桶水
semaphore full=0; // 表示缸中有多少桶水，初始时缸中没有水
semaphore buckets=3; // 表示有多少只空桶可用，初始时有3只桶可用
semaphore mutex_well=1; // 用于实现对井的互斥操作
semaphore mutex_bigjar=1; // 用于实现对缸的互斥操作

```
young_monk(){
  while(1){
    P(empty);
    P(buckets);
    go to the well;
    P(mutex_well);
    get water;
    V(mutex_well);
    go to the temple;
    P(mutex_bigjar);
    pure the water into the big jar;
    V(mutex_bigjar);
    V(buckets);
    V(full);
  }
}
```

```
old_monk(){
  while(){
    P(full);
    P(buckets);
    P(mutex_bigjar);
    get water;
    V(mutex_bigjar);
    drink water;
    V(buckets);
    V(empty);
  }
}
```

内容纲要　7.1 死锁基本概念

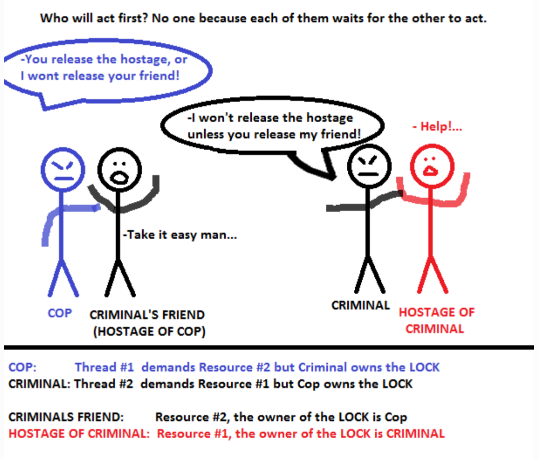
一、死锁概念

二、死锁示例

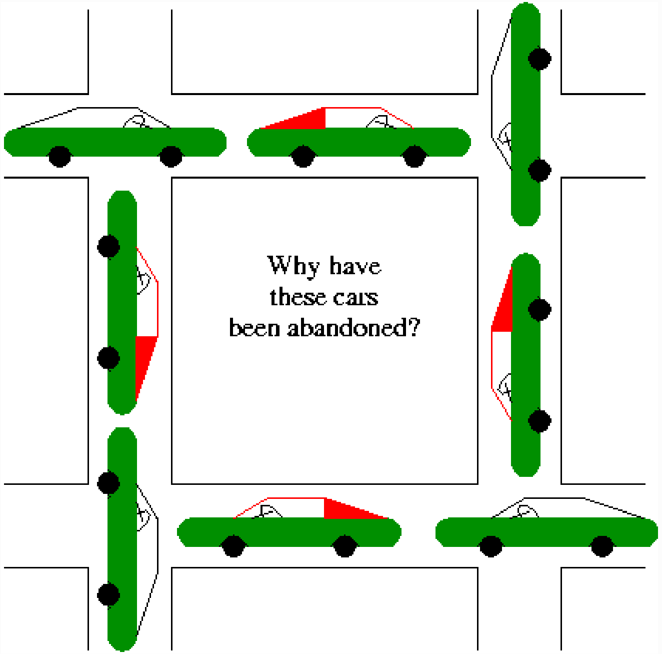
三、死锁成因

一、死锁概念

- 什么是死锁



Who will act first? No one because each of them waits for the other to act.
-You release the hostage, or I wont release your friend!
-I won't release the hostage unless you release my friend!
- Help!...
-Take it easy man...
COP
CRIMINAL'S FRIEND (HOSTAGE OF COP)
CRIMINAL
HOSTAGE OF CRIMINAL
COP: Thread #1 demands Resource #2 but Criminal owns the LOCK
CRIMINAL: Thread #2 demands Resource #1 but Cop owns the LOCK
CRIMINALS FRIEND: Resource #2, the owner of the LOCK is Cop
HOSTAGE OF CRIMINAL: Resource #1, the owner of the LOCK is CRIMINAL



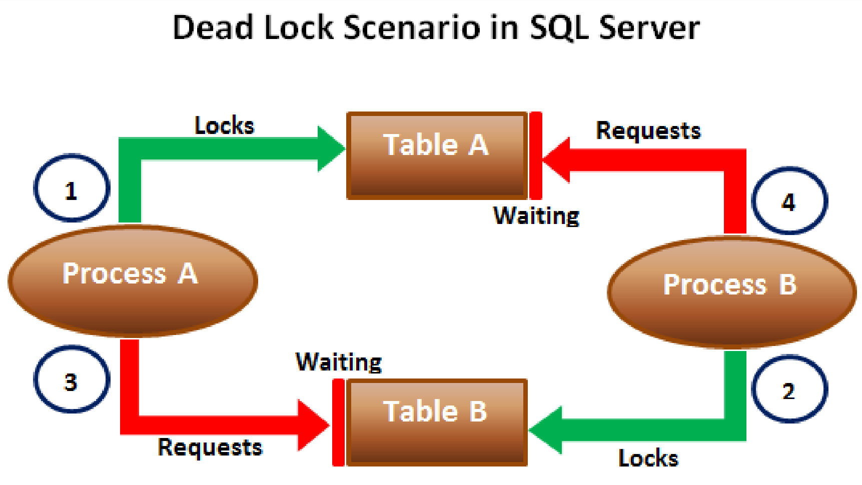
Why have these cars been abandoned?

一、死锁概念

- **什么是死锁**
  - A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set

    一个由阻塞进程构成的进程集中，每个进程持有部分资源，同时又在等待被另一个进程所占有的一个或多个资源

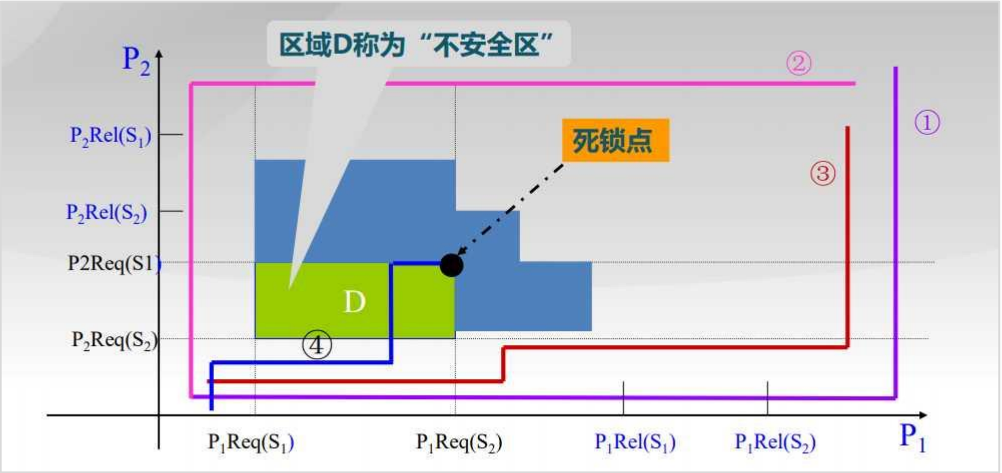
二、死锁示例



- 因进程A、B同时操作两个table的需求，造成的互锁

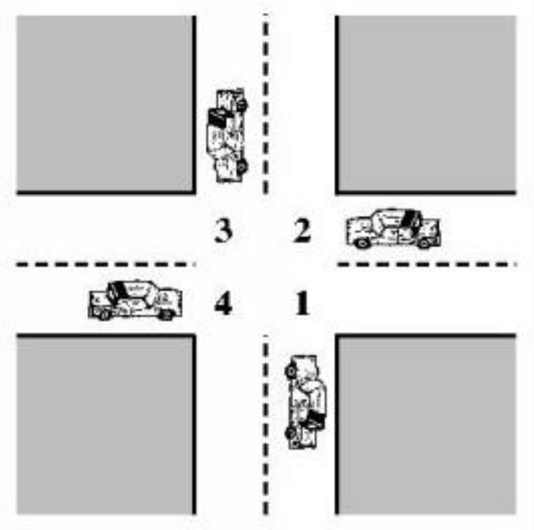
三、死锁成因

- **导致死锁的两大原因**
  - 系统资源不足
  - 进程推进顺序不当

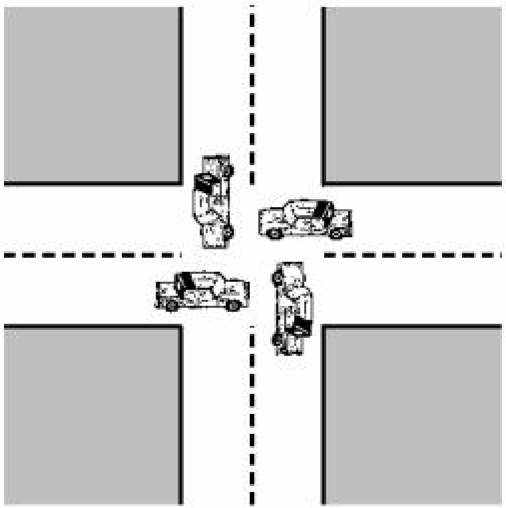


三、死锁成因

- **4 cars deadlock scenario**



(a)Deadlock possible



(b)Deadlocked

十字路口，空位资源稀缺

4车前进方式不当，导致死锁

本讲小结

- 死锁概念
- 死锁示例
- 死锁成因

内容纲要　7.2 死锁必要条件

一、死锁必要条件

二、持有并等待

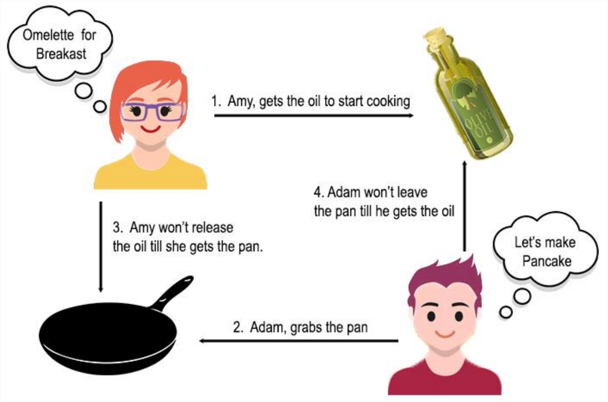
三、循环等待

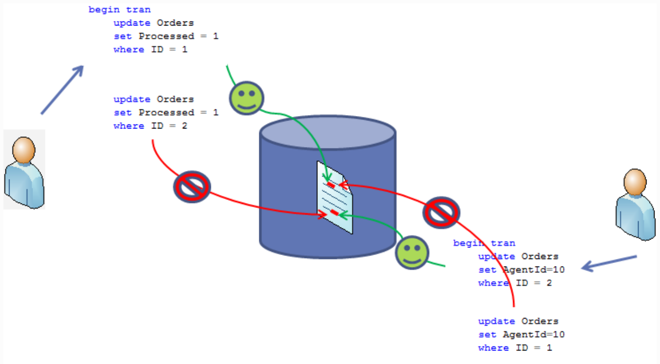
一、死锁必要条件

- **形成死锁的四大必要条件**
  - 资源以互斥方式使用（Mutual exclusion）
  - 持有并等待（Hold and wait）
  - 已持有资源不可被剥夺（No preemption）
  - 循环等待（Circular wait）

二、持有并等待

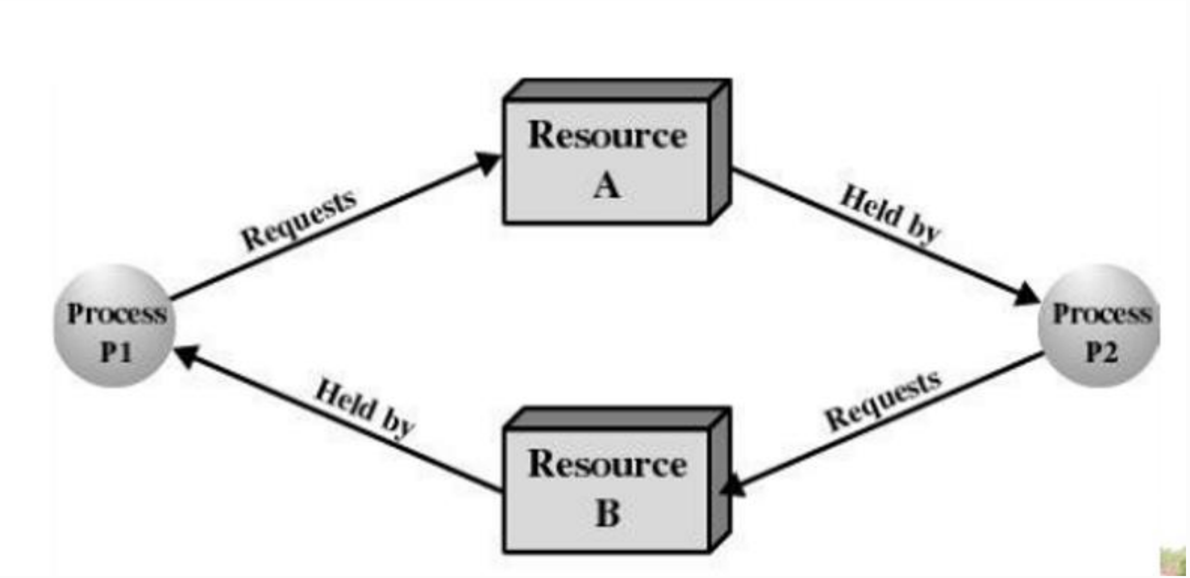
- **Hold-and-Wait**
  - 进程已占有一部分资源，并请求更多资源





三、循环等待

- **死锁必要条件4：循环等待**
  - 形成一个资源等待环路，且每个进程至少占有环路中下一进程需要的1个资源



本讲小结

- 死锁必要条件

内容纲要　　7.3 资源分配图

一、资源分配图

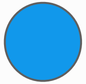
二、基于资源分配图的死锁分析

一、资源分配图

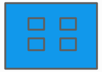
- 死锁的现象，本质上都可以归结到资源分配不当问题
- 资源分配图：可以用来为死锁进行建模

一、资源分配图

- 节点:
  - 分为2类：进程节点与资源节点

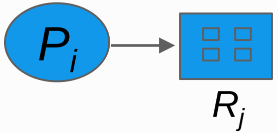


Process

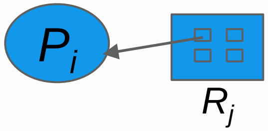


Resource Type
(with 4 instances)

- 边:
  - 资源请求边
  - 资源分配边



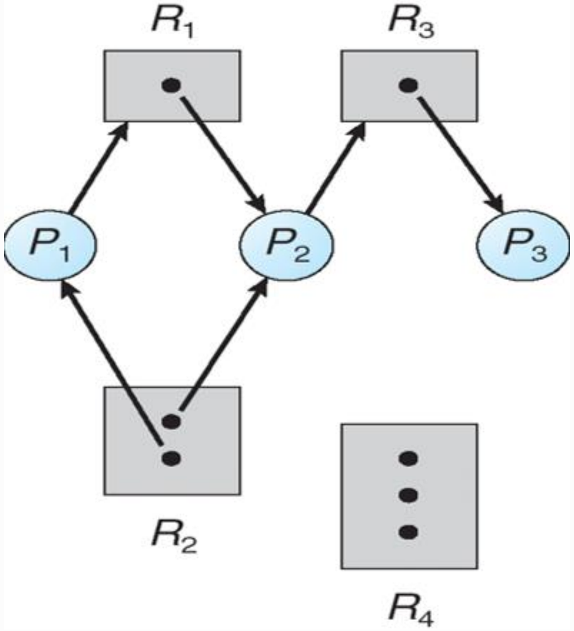
Request edge



Assignment edge

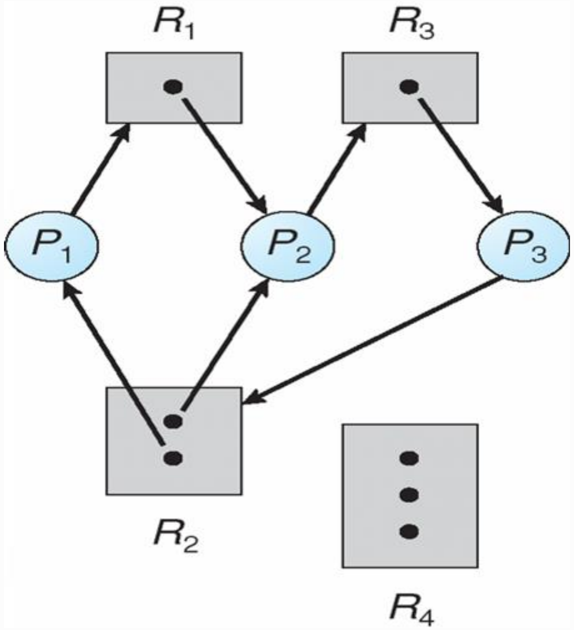
二、基于资源分配图的死锁分析

- 资源分配图示意图1：无环



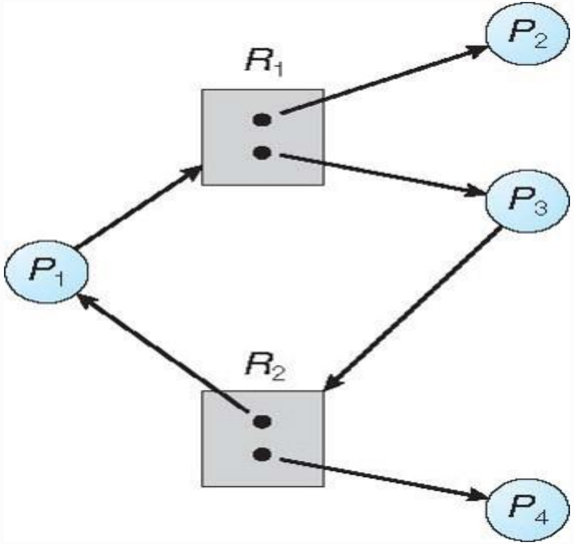
二、基于资源分配图的死锁分析

- 资源分配图示意图2：有环



二、基于资源分配图的死锁分析

- 资源分配图示意图3：有环

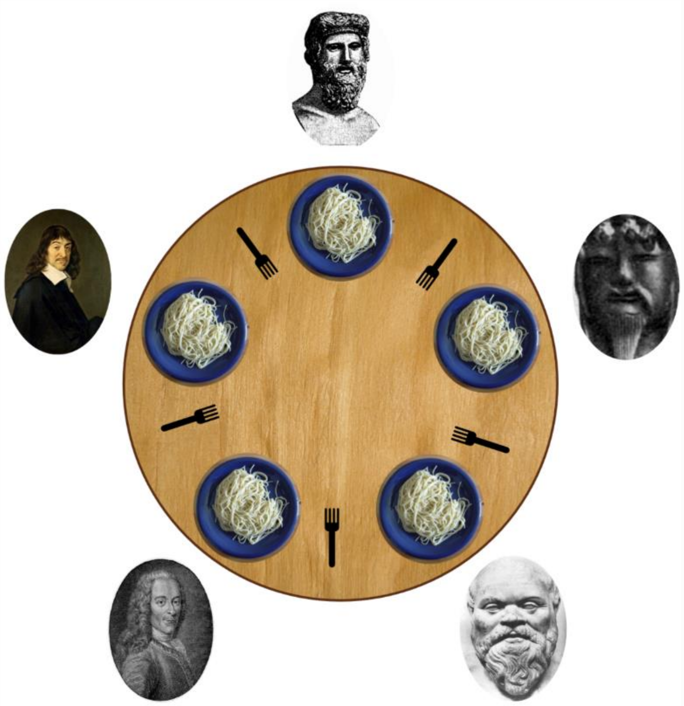


本讲小结

- 基于资源分配图的死锁建模与分析

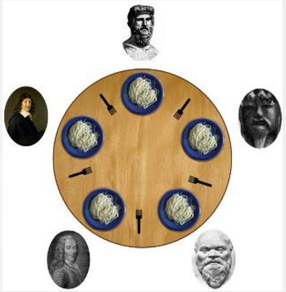
E、哲学家就餐问题

- 五位哲学家就餐

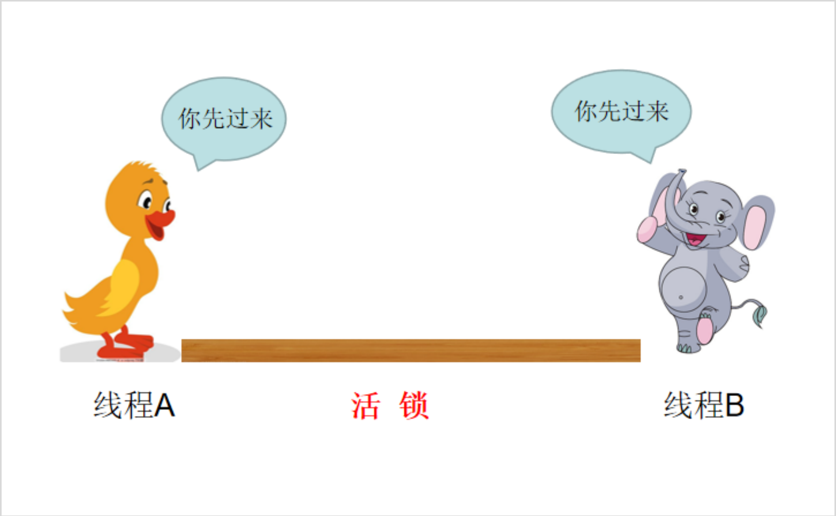


E、哲学家就餐问题-管程实现

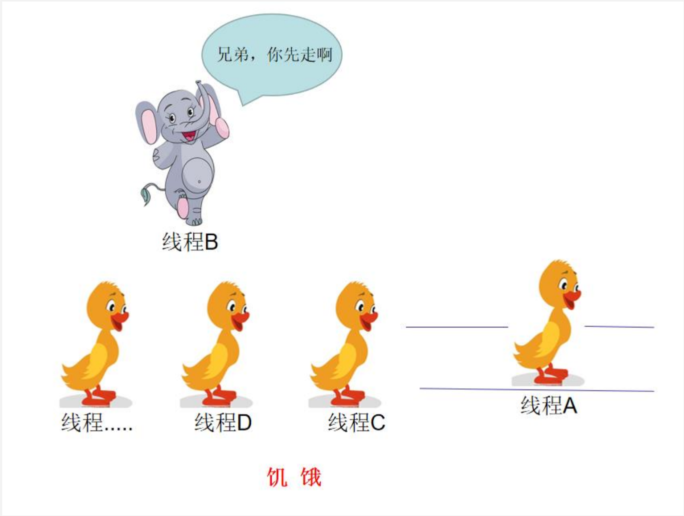
```
monitor DiningPhilosophers
  {
   enum { THINKING; HUNGRY, EATING) state [5] ;
   condition self [5];

   void pickup (int i) {
          state[i] = HUNGRY;
          test(i);
          if (state[i] != EATING) self [i].wait;
   }

      void putdown (int i) {
          state[i] = THINKING;
                  // test left and right neighbors
           test((i + 4) % 5);
           test((i + 1) % 5);
       }

      void test (int i) {
              if ( (state[(i + 4) % 5] != EATING) &&
              (state[i] == HUNGRY) &&
              (state[(i + 1) % 5] != EATING) ) {
                   state[i] = EATING ;
                        self[i].signal () ;
              }
       }

        initialization_code() {
            for (int i = 0; i < 5; i++)
            state[i] = THINKING;
       }
}
```



活锁与饥饿



Livelock Demo



Starvation Demo