華中科技大學

操 作 系 统

实 验 报 告

实验名称	哲学家就餐问题
班　级	提高 2001 班
学　号	U202013830
姓　名	陈艺文
任课老师	刘澍

电子信息与通信学院

一、 问题描述

有五个哲学家围坐在一圆桌旁，桌中央有一盘通心粉，每人面前有一只空盘子，每两人之间放一只筷子。每个哲学家的行为是思考，感到饥饿，然后吃通心粉。为了吃通心粉，每个哲学家必须拿到两只筷子，并且每个人只能直接从自己的左边或右边去取筷子。

二、 分配方式

方式一（不会进入死锁）

仅当一个哲学家左右两边的筷子都可用时，才允许他拿筷子。这样要么一次占有两只筷子（所有线程需要的资源）进行下一步的吃通心粉，然后释放所有的资源；要么不占用资源，这样就不可能产生死锁了。

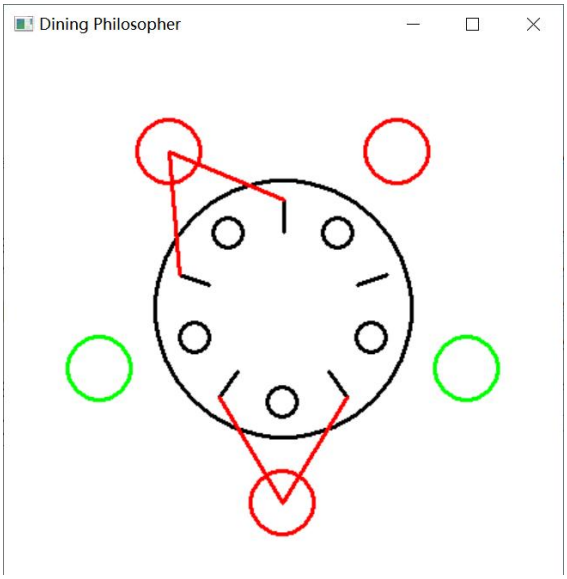

图 2-1 不产生死锁

方式二（会进入死锁）

当筷子（资源）可用时，先分配左边的筷子，等待一会后再分配右边的筷子，由于这个过程中，左边的筷子一直没有释放，就有可能产生死锁了。

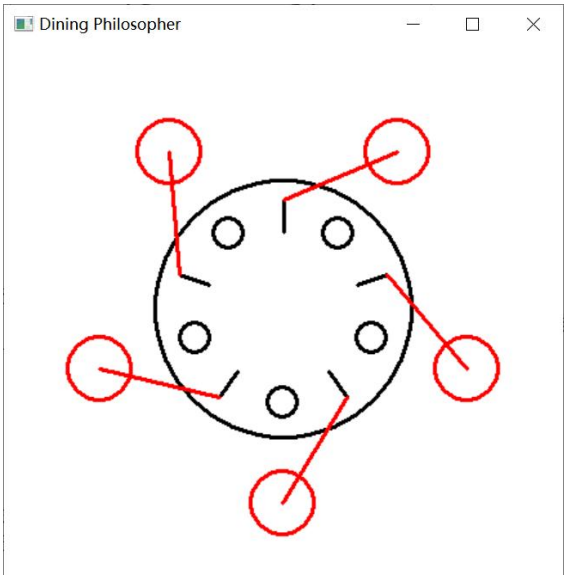

图 2-2 进入死锁

三、 伪代码（信号量和 PV 操作）

对于每一个哲学家来说，他们需要抢占的临界资源就是左右的两支筷子。
如下写出两种分配资源方式的算法：

1. **不发生死锁的方式（要么一下占用两支筷子，要么不占用）**

```
var mutexleftchopstick,mutexrightchopstick;
beging:
    resting;//休息状态
    waiting;//等待状态
    p(mutexleftchopstick);//判断左筷子状态
    p(mutexrightchopstick);//判断右筷子状态
    GetResource(leftchopstick,rightchopstick);
    eating;//左右筷子都为空则占用，进入 eating 状态
    v(mutexleftchopstick);//释放左筷子
    v(mutexrightchopstick);//释放右筷子
end
```

2. **发生死锁的方式（一旦可以占用筷子，就马上占用）**

```
var mutexleftchopstick,mutexrightchopstick;
beging:
    resting;//休息状态
    waiting;//等待状态
    p(mutexleftchopstick);//判断左筷子状态
    GetResource(leftchopstick);//立即占用左筷子
    p(mutexrightchopstick);//判断右筷子状态
    GetResource(rightchopstick);//立即占用右筷子
    eating;//进入 eating 状态
    v(mutexleftchopstick);//释放左筷子
    v(mutexrightchopstick);//释放右筷子
end
```

四、 思考题（其他解决死锁的办法）

方案一：最多允许有四位哲学家同时去拿左边的筷子，然后再拿右边的筷子，最终保证至少有一位哲学家能够进餐，并在就餐完毕时同时释放他用过的两只筷子，从而使更多的哲学家能够进餐。

```
semaphore mutex[5] = {1,1,1,1,1};    //初始化信号量
semaphore count = 4; //控制最多允许四位哲学家同时进餐

void philosopher(int i){
  do {
    resting;  //休息状态
    waiting;  //等待状态
    P(count);  //判断是否超过四人准备进餐
    P(mutex[i]); //判断哲学家左边的筷子是否可用
    P(mutex[(i+1)%5]);// 判断哲学家右边的筷子是否可用
    eat;   //进入 eating 状态
    V(mutex[i]);//释放左边筷子
    V(mutex[(i+1)%5]);//释放右边筷子
    V(count);//用餐完毕，别的哲学家可以开始进餐
  }while(true);
}
```

方案二：规定奇数号的哲学家先拿起他左边的筷子，然后再去拿他右边的筷子；而偶数号的哲学家则先拿起他右边的筷子，然后再去拿他左边的筷子，此时需要在代码中添加个判断，来决定获取左、右筷子的顺序。

```
semaphore mutex[5] = {1,1,1,1,1};    //初始化信号量

void philosopher(int i){
  do {
    resting;
     waiting;
    if(i%2 == 1){
      P(mutex[i]);//判断哲学家左边的筷子是否可用
      P(mutex[(i+1)%5]);//判断哲学家右边的筷子是否可用
    }else{
      P(mutex[(i+1)%5]);//判断哲学家右边的筷子是否可用
      P(mutex[i]);//判断哲学家左边的筷子是否可用
    }
    eat;
    V(mutex[i]);//释放左筷子
    V(mutex[(i+1)%5]);//释放右筷子
  }while(true);
}
```