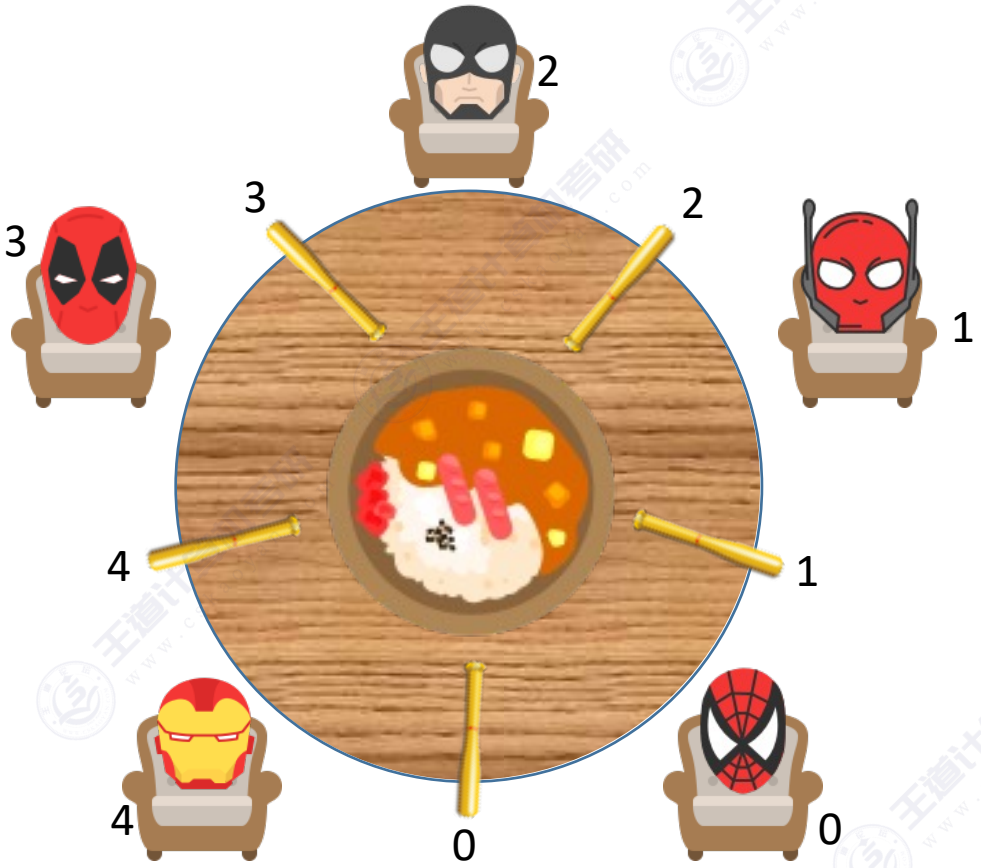
本节内容

哲学家进餐问题

问题描述

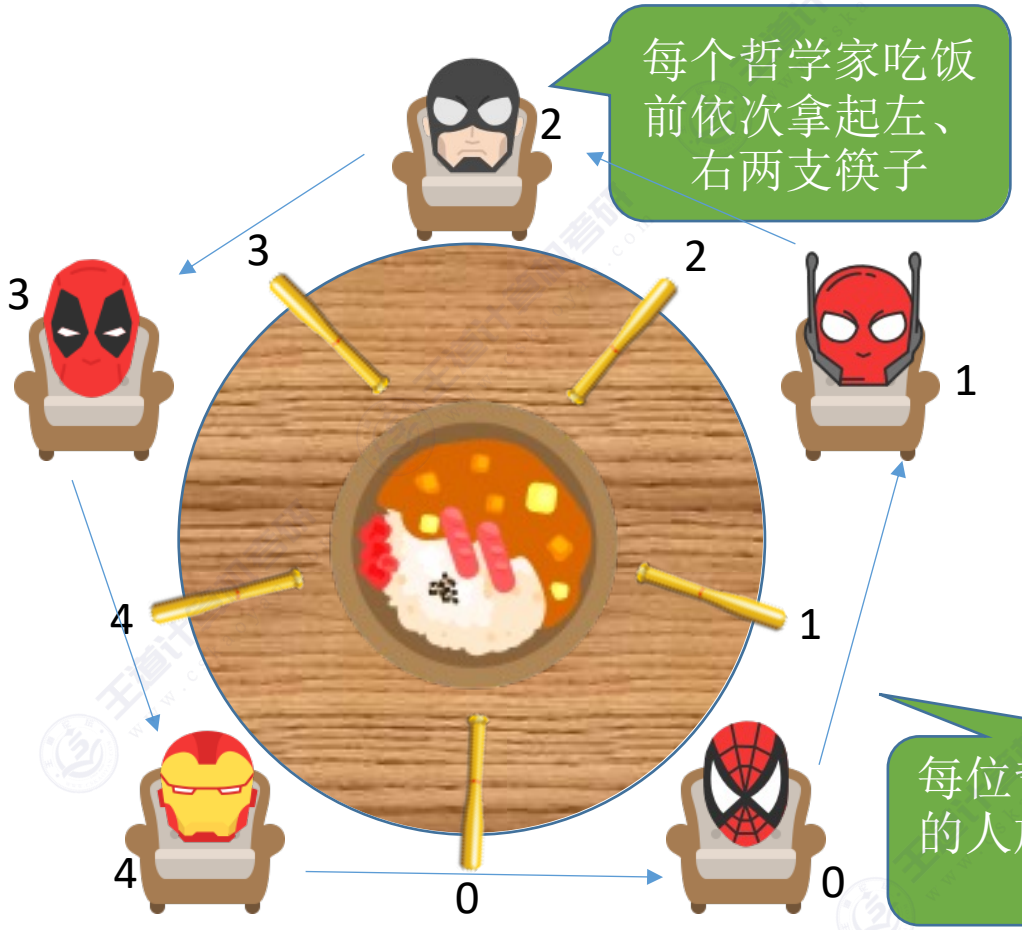
一张圆桌上坐着5名哲学家，每两个哲学家之间的桌上摆一根筷子，桌子的中间是一碗米饭。哲学家们倾注毕生的精力用于思考和进餐，哲学家在思考时，并不影响他人。只有当哲学家饥饿时，才试图拿起左、右两根筷子（一根一根地拿起）。如果筷子已在他人手上，则需等待。饥饿的哲学家只有同时拿起两根筷子才可以开始进餐，当进餐完毕后，放下筷子继续思考。



1. 关系分析。系统中有5个哲学家进程，5位哲学家与左右邻居对其中间筷子的访问是互斥关系。
2. 整理思路。这个问题中只有互斥关系，但与之前遇到的问题不同的事，每个哲学家进程需要同时持有两个临界资源才能开始吃饭。如何避免临界资源分配不当造成的死锁现象，是哲学家问题的精髓。
3. 信号量设置。定义互斥信号量数组chopstick[5]={1,1,1,1,1}用于实现对5个筷子的互斥访问。并对哲学家按0~4编号，哲学家 i 左边的筷子编号为 i，右边的筷子编号为 (i+1)%5。

问题分析

一张圆桌上坐着5名哲学家，每两个哲学家之间的桌上摆一根筷子，桌子的中间是一碗米饭。哲学家们倾注毕生的精力用于思考和进餐，哲学家在思考时，并不影响他人。只有当哲学家饥饿时，才试图拿起左、右两根筷子（一根一根地拿起）。如果筷子已在他人手上，则需等待。饥饿的哲学家只有同时拿起两根筷子才可以开始进餐，当进餐完毕后，放下筷子继续思考。



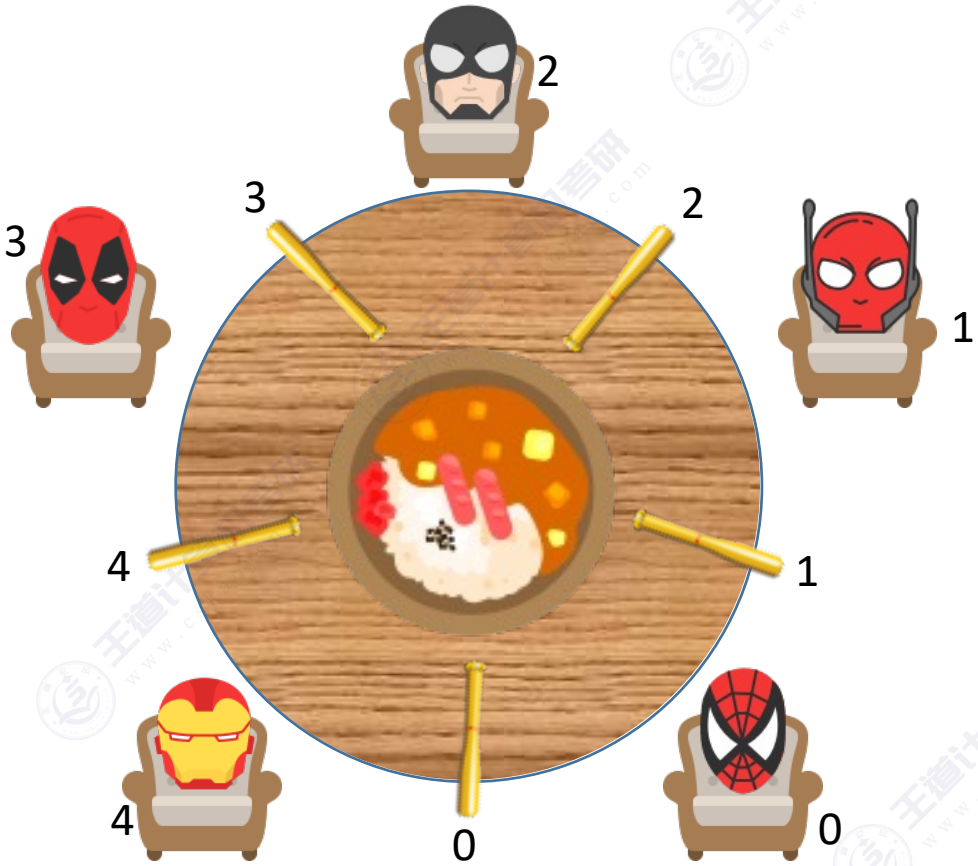

```
semaphore chopstick[5]={1,1,1,1,1};
Pi (){          //i号哲学家的进程
    while(1){
        P(chopstick[i]);              //拿左
        P(chopstick[(i+1)%5]);        //拿右
        吃饭…
        V(chopstick[i]);              //放左
        V(chopstick[(i+1)%5]);        //放右
        思考…
    }
}
```

每位哲学家循环等待右边的人放下筷子（阻塞）。发生“死锁”

如果5个哲学家并发地拿起了自己左手边的筷子…

如何实现

一张圆桌上坐着5名哲学家，每两个哲学家之间的桌上摆一根筷子，桌子的中间是一碗米饭。哲学家们倾注毕生的精力用于思考和进餐，哲学家在思考时，并不影响他人。只有当哲学家饥饿时，才试图拿起左、右两根筷子（一根一根地拿起）。如果筷子已在他人手上，则需等待。饥饿的哲学家只有同时拿起两根筷子才可以开始进餐，当进餐完毕后，放下筷子继续思考。



如何防止死锁的发生呢？

①可以对哲学家进程施加一些限制条件，比如最多允许四个哲学家同时进餐。这样可以保证至少有一个哲学家是可以拿到左右两只筷子的

如何实现

一张圆桌上坐着5名哲学家，每两个哲学家之间的桌上摆一根筷子，桌子的中间是一碗米饭。哲学家们倾注毕生的精力用于思考和进餐，哲学家在思考时，并不影响他人。只有当哲学家饥饿时，才试图拿起左、右两根筷子（一根一根地拿起）。如果筷子已在他人手上，则需等待。饥饿的哲学家只有同时拿起两根筷子才可以开始进餐，当进餐完毕后，放下筷子继续思考。

思考并尝试用代码实现这两种方案...

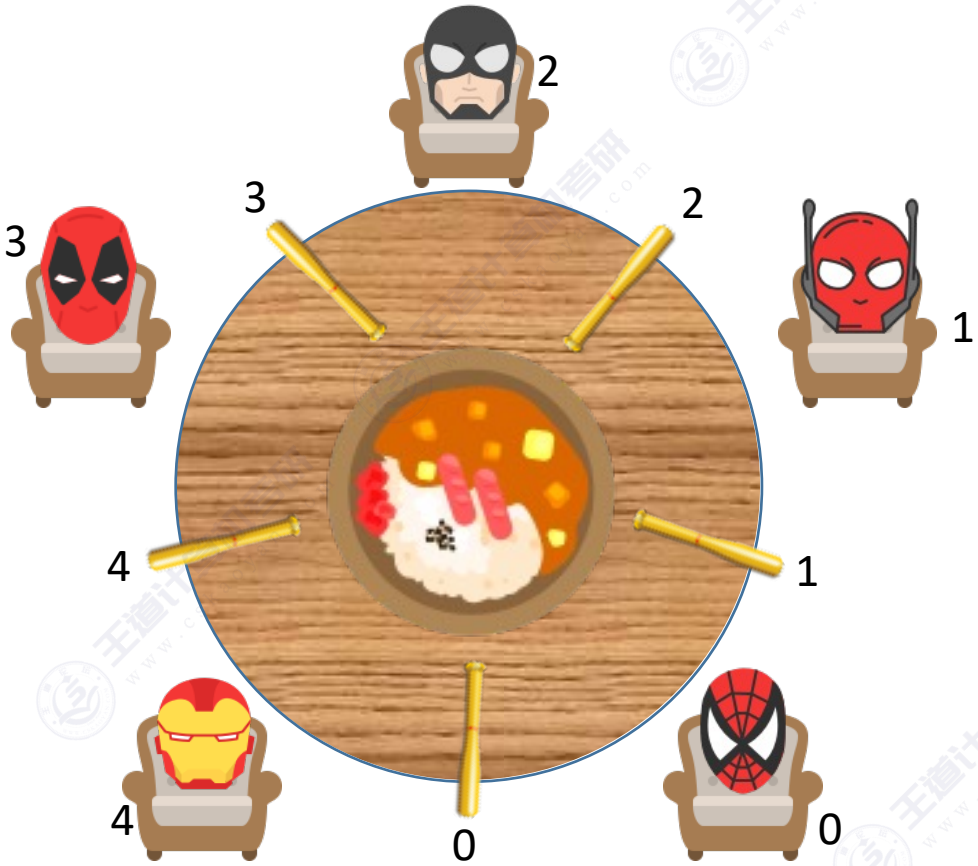
如何防止死锁的发生呢？

①可以对哲学家进程施加一些限制条件，比如最多允许四个哲学家同时进餐。这样可以保证至少有一个哲学家是可以拿到左右两只筷子的

②要求奇数号哲学家先拿左边的筷子，然后再拿右边的筷子，而偶数号哲学家刚好相反。用这种方法可以保证如果相邻的两个奇偶号哲学家都想吃饭，那么只会有其中一个可以拿起第一只筷子，另一个会直接阻塞。这就避免了占有一支后再等待另一只的情况。

如何实现

一张圆桌上坐着5名哲学家，每两个哲学家之间的桌上摆一根筷子，桌子的中间是一碗米饭。哲学家们倾注毕生的精力用于思考和进餐，哲学家在思考时，并不影响他人。只有当哲学家饥饿时，才试图拿起左、右两根筷子（一根一根地拿起）。如果筷子已在他人手上，则需等待。饥饿的哲学家只有同时拿起两根筷子才可以开始进餐，当进餐完毕后，放下筷子继续思考。

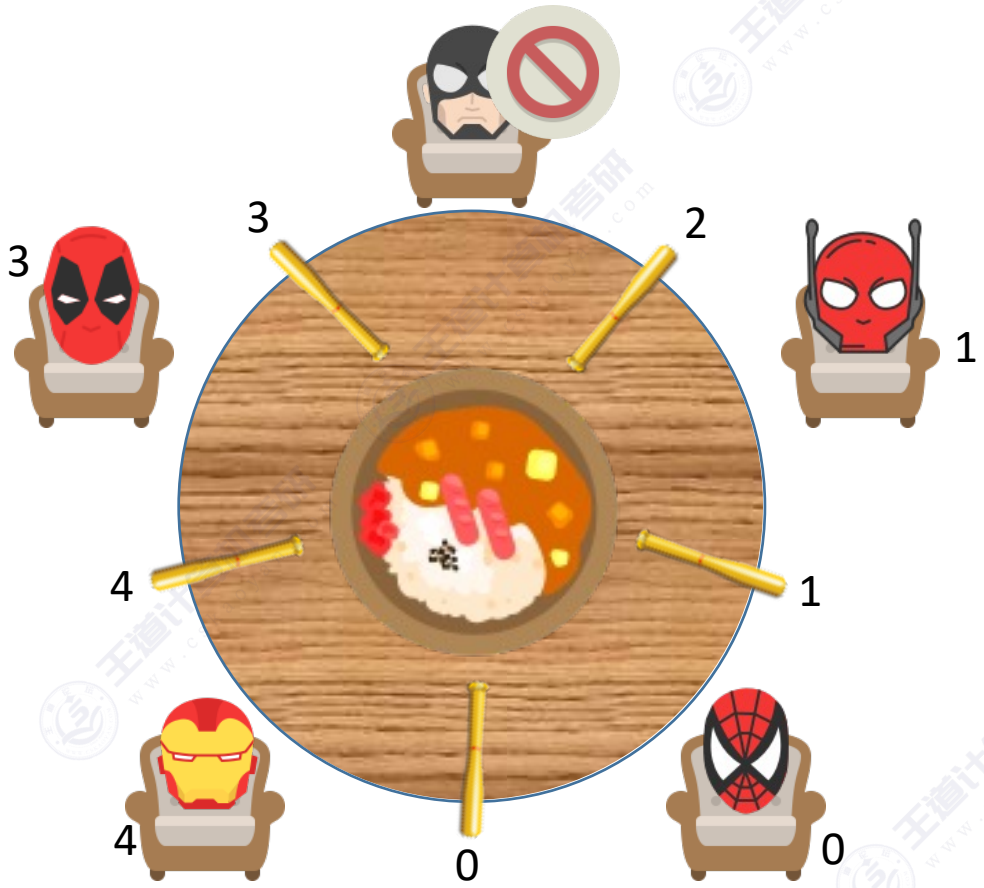


如何防止死锁的发生呢？

③仅当一个哲学家左右两支筷子都可用时才允许他抓起筷子。

如何实现

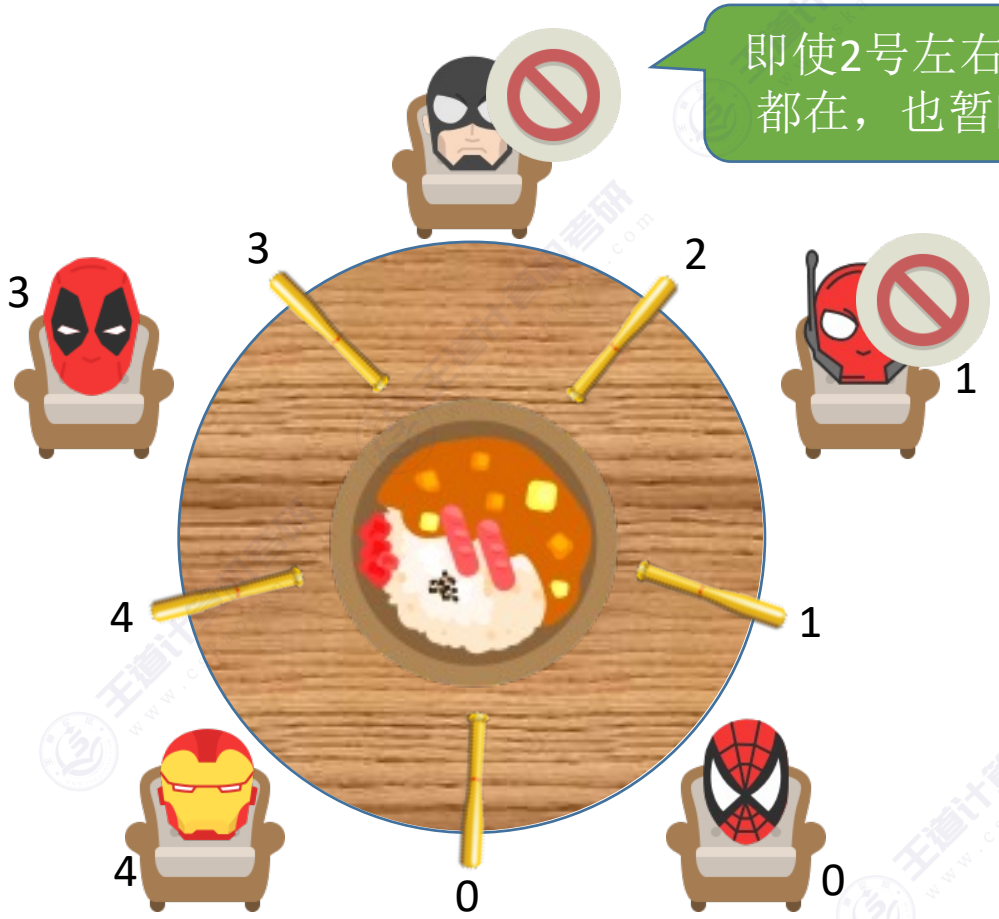
一张圆桌上坐着5名哲学家，每两个哲学家之间的桌上摆一根筷子，桌子的中间是一碗米饭。哲学家们倾注毕生的精力用于思考和进餐，哲学家在思考时，并不影响他人。只有当哲学家饥饿时，才试图拿起左、右两根筷子（一根一根地拿起）。如果筷子已在他人手上，则需等待。饥饿的哲学家只有同时拿起两根筷子才可以开始进餐，当进餐完毕后，放下筷子继续思考。




```
semaphore chopstick[5]={1,1,1,1,1};
semaphore mutex = 1;    //互斥地取筷子
Pi (){                  //i号哲学家的进程
    while(1){
        P(mutex);
        P(chopstick[i]);            //拿左
        P(chopstick[(i+1)%5]);      //拿右
        V(mutex);
        吃饭…
        V(chopstick[i]);            //放左
        V(chopstick[(i+1)%5]);      //放右
        思考…
    }
}
```

如何实现

一张圆桌上坐着5名哲学家，每两个哲学家之间的桌上摆一根筷子，桌子的中间是一碗米饭。哲学家们倾注毕生的精力用于思考和进餐，哲学家在思考时，并不影响他人。只有当哲学家饥饿时，才试图拿起左、右两根筷子（一根一根地拿起）。如果筷子已在他人手上，则需等待。饥饿的哲学家只有同时拿起两根筷子才可以开始进餐，当进餐完毕后，放下筷子继续思考。




```
ore chopstick[5]={1,1,1,1,1};
ore mutex = 1;      //互斥地取筷子
Pi (){                          //i号哲学家的进程
    while(1){
        P(mutex);
        P(chopstick[i]);                //拿左
        P(chopstick[(i+1)%5]);          //拿右
        V(mutex);
        吃饭…
        V(chopstick[i]);                //放左
        V(chopstick[(i+1)%5]);          //放右
        思考…
    }
}
```

如何实现

张圆桌上坐着5名哲学家，每两个哲学家之间的桌上摆一
用于思考和进餐，哲学家在思考时，
根筷子（一根一根地拿起）。如果筷
根筷子才可以开始进餐，当进餐完毕后

因此这种方法并不能保证只有两边的筷子都可用时，才允许哲学家拿起筷子。

更准确的说法应该是：各哲学家拿筷子这件事必须互斥的执行。这就保证了即使一个哲学家在拿筷子拿到一半时被阻塞，也不会有别的哲学家会继续尝试拿筷子。这样的话，当前正在吃饭的哲学家放下筷子后，被阻塞的哲学家就可以获得等待的筷子了。

此时4号右边的筷子不可用，但4号仍然会先拿起左边的筷子

```
semaphore chopstick[5]={1,1,1,1,1};
semaphore mutex = 1;     //互斥地取筷子
Pi (){                   //i号哲学家的进程
    while(1){
        P(mutex);
        P(chopstick[i]);              //拿左
        P(chopstick[(i+1)%5]);        //拿右
        V(mutex);
        吃饭…
        V(chopstick[i]);              //放左
        V(chopstick[(i+1)%5]);        //放右
        思考…
    }
}
```

知识回顾与重要考点

哲学家进餐问题的关键在于解决进程死锁。

这些进程之间只存在互斥关系，但是与之前接触到的互斥关系不同的是，每个进程都需要同时持有两个临界资源，因此就有“死锁”问题的隐患。

如果在考试中遇到了一个进程需要同时持有多个临界资源的情况，应该参考哲学家问题的思想，分析题中给出的进程之间是否会发生循环等待，是否会发生死锁。

可以参考哲学家就餐问题解决死锁的三种思路。

公众号：王道在线

b站：王道计算机教育

抖音：王道计算机考研