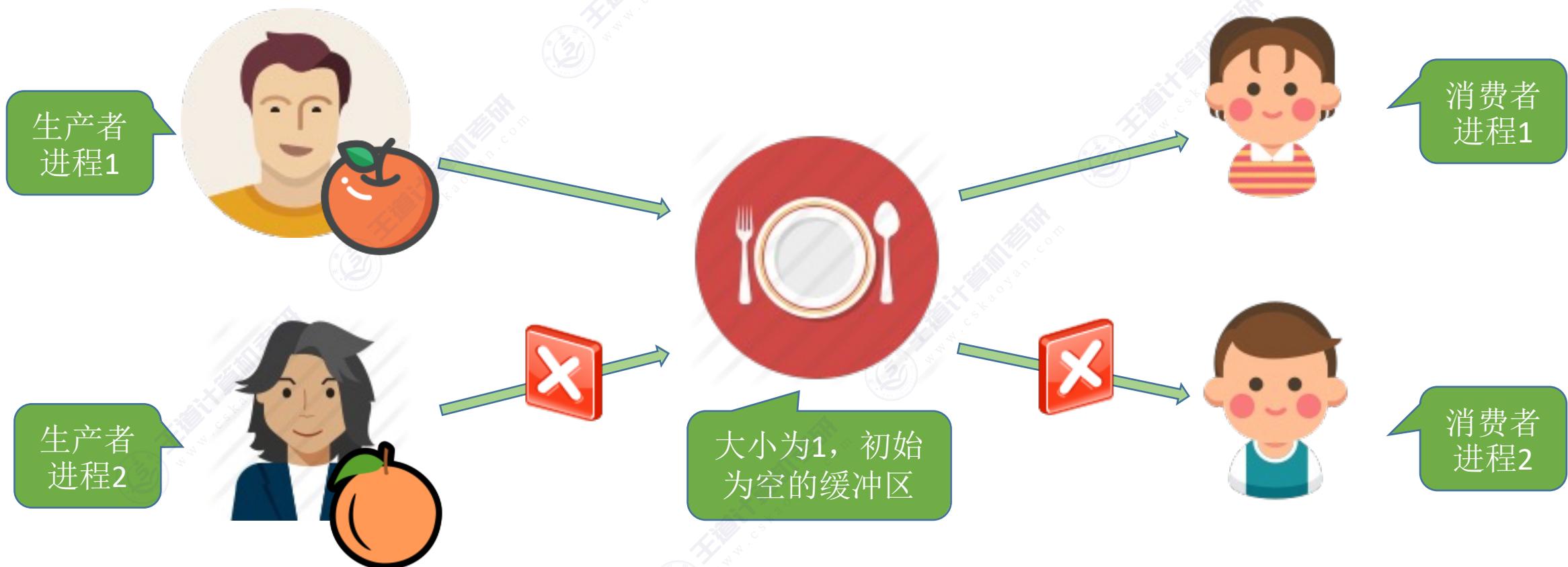


本节内容

多生产者-多
消费者

问题描述

桌子上有一只盘子，每次只能向其中放入一个水果。爸爸专向盘子中放苹果，妈妈专向盘子中放橘子，儿子专等着吃盘子中的橘子，女儿专等着吃盘子中的苹果。只有盘子空时，爸爸或妈妈才可向盘子中放一个水果。仅当盘子中有自己需要的水果时，儿子或女儿可以从盘子中取出水果。用PV操作实现上述过程。

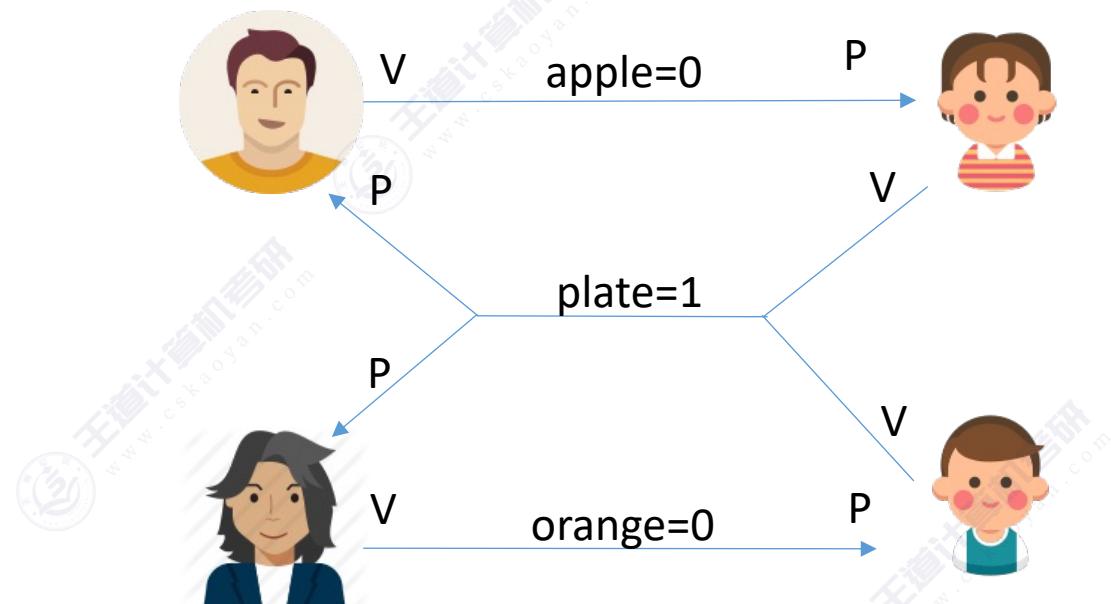


问题分析

桌子上有一只盘子，每次只能向其中放入一个水果。爸爸专向盘子中放苹果，妈妈专向盘子中放橘子，儿子专等着吃盘子中的橘子，女儿专等着吃盘子中的苹果。只有盘子空时，爸爸或妈妈才可向盘子中放一个水果。仅当盘子中有自己需要的水果时，儿子或女儿可以从盘子中取出水果。

1. 关系分析。找出题目中描述的各个进程，分析它们之间的同步、互斥。
2. 整理思路。根据各进程的操作流程确定P、V操作的大致顺序。
3. 设置信号量。设置需要的信号量，并根据题目条件确定信号量初值。（互斥信号量初值一般为1，同步信号量的初始值要看对应资源的初始值是多少）

互斥：在临界区前后分别PV
同步：前V后P

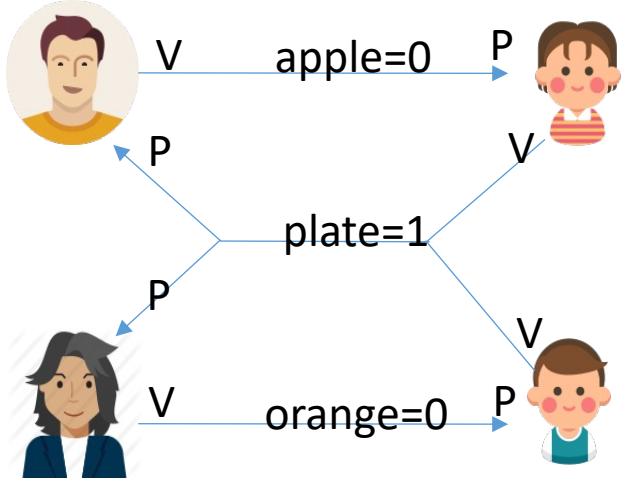


互斥关系： $(\text{mutex} = 1)$
对缓冲区（盘子）的访问要互斥地进行

同步关系（一前一后）：

1. 父亲将苹果放入盘子后，女儿才能取苹果
2. 母亲将橘子放入盘子后，儿子才能取橘子
3. 只有盘子为空时，父亲或母亲才能放入水果

“盘子为空”这个事件可以由儿子或女儿触发，事件发生后才允许父亲或母亲放水果



如何实现

问题：可不可以不用互斥信号量？

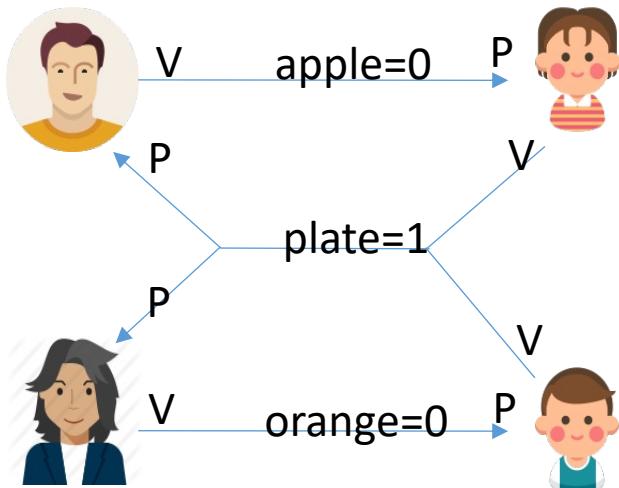
```
semaphore mutex = 1;           // 实现互斥访问盘子（缓冲区）
semaphore apple = 0;           // 盘子中有几个苹果
semaphore orange = 0;          // 盘子中有几个橘子
semaphore plate = 1;           // 盘子中还可以放多少个水果
```

```
dad () {
    while(1) {
        准备一个苹果;
        P(plate);
        P(mutex);
        把苹果放入盘子;
        V(mutex);
        V(apple);
    }
}
```

```
mom () {
    while(1) {
        准备一个橘子;
        P(plate);
        P(mutex);
        把橘子放入盘子;
        V(mutex);
        V(orange);
    }
}
```

```
daughter () {
    while(1) {
        P(apple);
        P(mutex);
        从盘中取出苹果;
        V(mutex);
        V(plate);
        吃掉苹果;
    }
}
```

```
son () {
    while(1) {
        P(orange);
        P(mutex);
        从盘中取出橘子;
        V(mutex);
        V(plate);
        吃掉橘子;
    }
}
```



如何实现

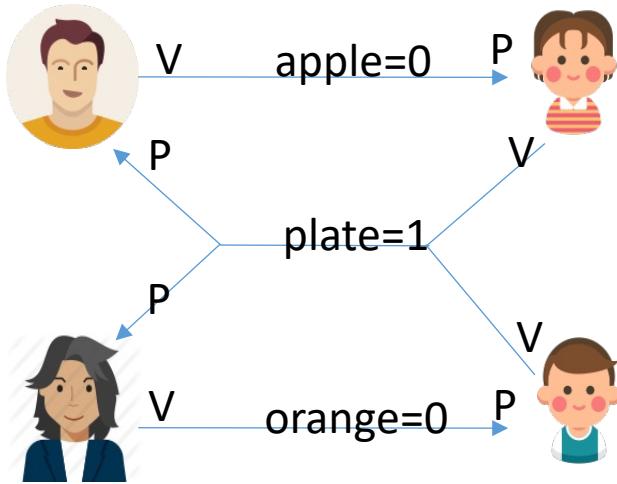
结论：即使不设置专门的互斥变量mutex，也不会出现多个进程同时访问盘子的现象

```
semaphore mutex = 1; // 实现互斥访问盘子（缓冲区）
semaphore apple = 0; // 盘子中有几个苹果
semaphore orange = 0; // 盘子中有几个橘子
semaphore plate = 1; // 盘子中还可以放多少个水果
```

纳尼？

原因在于：本题中的缓冲区大小为1，在任何时刻，apple、orange、plate 三个同步信号量中最多只有一个1。因此在任何时刻，最多只有一个进程的P操作不会被阻塞，并顺利地进入临界区...

如果盘子容量为2的话...会被阻塞。如果刚开始是父亲进程先上处理器运行，则：
① 父亲 P(mutex), 父亲 P(plate) → 父亲放入苹果 V(apple), 女儿进程被唤醒，其他进程不可能访问临界资源（盘子）→女儿 P(apple), 访问盘子, V(plate), 等待盘子的母亲进程（其他进程暂时都无法进入临界区）→.....



```

dad () {
    while(1) {
        准备一个苹果;
        P(plate);
        把苹果放入盘子;
        V(apple);
    }
}

```

```

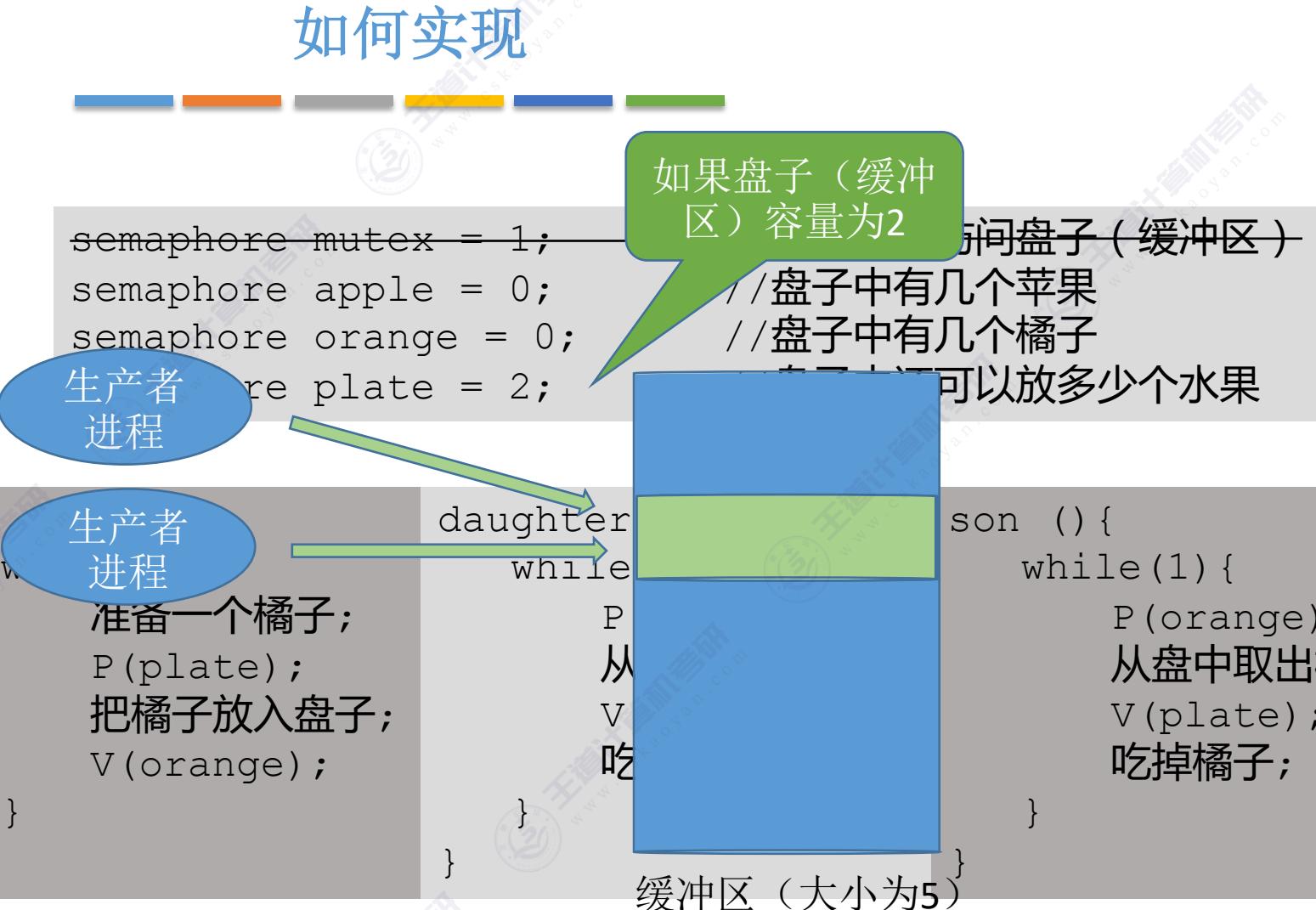
mom () {
    while(1) {
        准备一个橘子;
        P(plate);
        把橘子放入盘子;
        V(orange);
    }
}

```

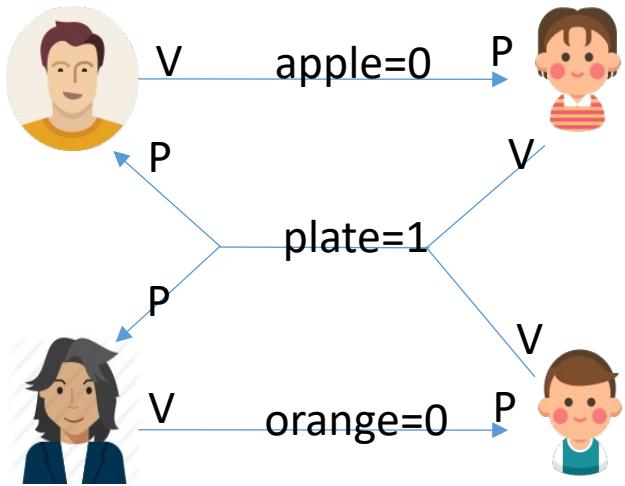
```

son () {
    while(1) {
        P(orange);
        从盘中取出橘子;
        V(plate);
        吃掉橘子;
    }
}

```



父亲 $P(\text{plate})$, 可以访问盘子 → 母亲 $P(\text{plate})$, 可以访问盘子 → 父亲在往盘子里放苹果, 同时母亲也可以往盘子里放橘子。于是就出现了两个进程同时访问缓冲区的情况, 有可能导致两个进程写入缓冲区的数据相互覆盖的情况。



```

dad () {
    while(1) {
        准备一个苹果;
        P(plate);
        把苹果放入盘子;
        V(apple);
    }
}

```

```

mom () {
    while(1) {
        准备一个橘子;
        P(plate);
        把橘子放入盘子;
        V(orange);
    }
}

```

```

daughter () {
    while(1) {
        P(apple);
        从盘中取出苹果;
        V(plate);
        吃掉苹果;
    }
}

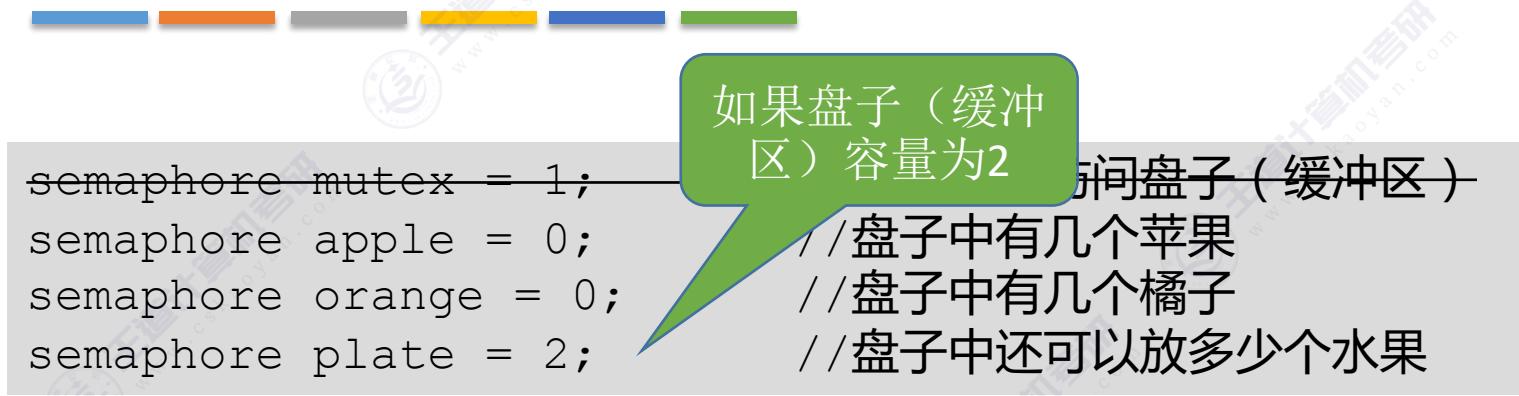
```

```

son () {
    while(1) {
        P(orange);
        从盘中取出橘子;
        V(plate);
        吃掉橘子;
    }
}

```

如何实现



父亲 $P(\text{plate})$, 可以访问盘子 → 母亲 $P(\text{plate})$, 可以访问盘子 → 父亲在往盘子里放苹果, 同时母亲也可以往盘子里放橘子。于是就出现了两个进程同时访问缓冲区的情况, 有可能导致两个进程写入缓冲区的数据相互覆盖的情况。因此, 如果缓冲区大小大于1, 就必须专门设置一个互斥信号量 mutex 来保证互斥访问缓冲区。

知识回顾与重要考点



总结：在生产者-消费者问题中，如果缓冲区大小为1，那么有可能不需要设置互斥信号量就可以实现互斥访问缓冲区的功能。当然，这不是绝对的，要具体问题具体分析。

建议：在考试中如果来不及仔细分析，可以加上互斥信号量，保证各进程一定会互斥地访问缓冲区。但需要注意的是，实现互斥的P操作一定要在实现同步的P操作之后，否则可能引起“死锁”。

PV 操作题目的解题思路：

1. 关系分析。找出题目中描述的各个进程，分析它们之间的同步、互斥关系。
2. 整理思路。根据各进程的操作流程确定P、V操作的大致顺序。
3. 设置信号量。设置需要的信号量，并根据题目条件确定信号量初值。（互斥信号量初值一般为1，同步信号量的初始值要看对应资源的初始值是多少）

知识回顾与重要考点

解决“多生产者-多消费者问题”的关键在于理清复杂的同步关系。

在分析同步问题（一前一后问题）的时候不能从单个进程行为的角度来分析，要把“一前一后”发生的事看做是两种“事件”的前后关系。

比如，如果从单个进程行为的角度来考虑的话，我们会有以下结论：

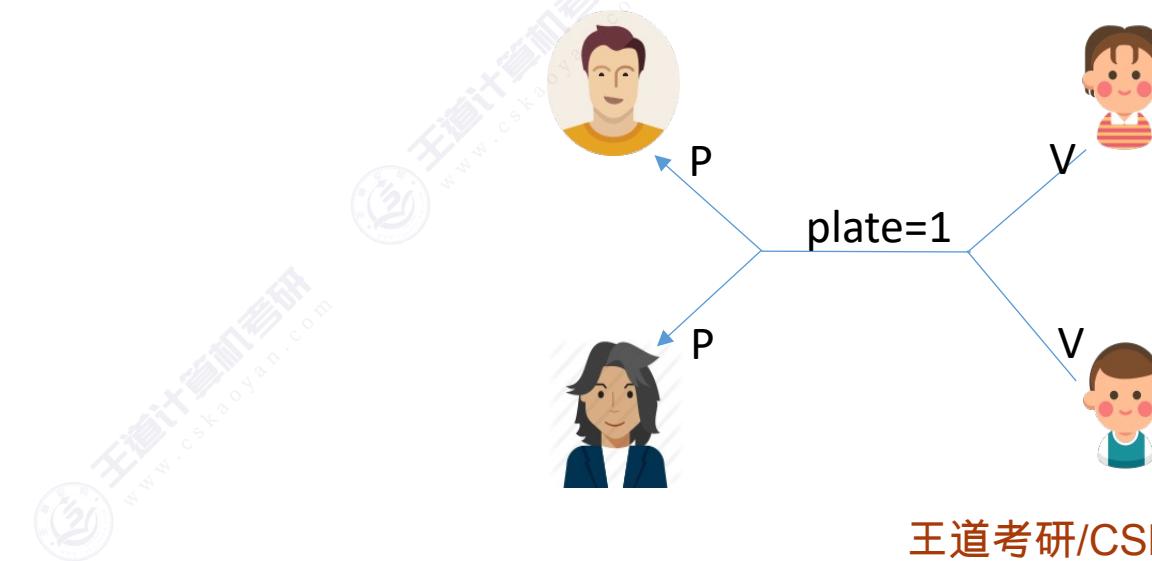
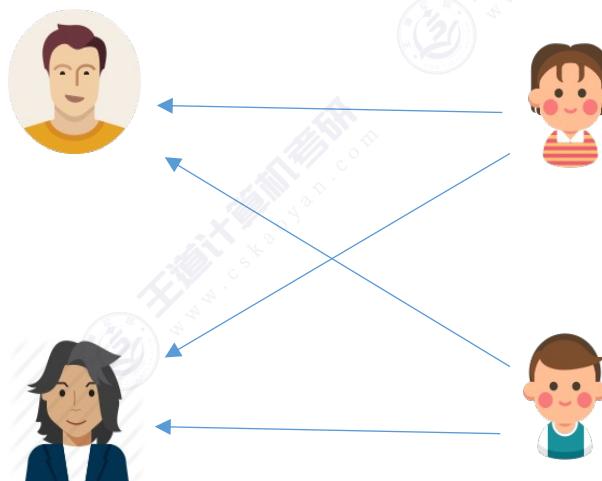
如果盘子里装有苹果，那么一定要女儿取走苹果后父亲或母亲才能再放入水果

如果盘子里装有橘子，那么一定要儿子取走橘子后父亲或母亲才能再放入水果

这么看是否就意味着要设置四个同步信号量分别实现这四个“一前一后”的关系了？

正确的分析方法应该从“事件”的角度来考虑，我们可以把上述四对“进程行为的前后关系”抽象为一对“事件的前后关系”

盘子变空事件 \rightarrow 放入水果事件。“盘子变空事件”既可由儿子引发，也可由女儿引发；“放水果事件”既可能是父亲执行，也可能是母亲执行。这样的话，就可以用一个同步信号量解决问题了





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