

本节内容

# 生产者消费者 者问题

## 问题描述

系统中有一组生产者进程和一组消费者进程，生产者进程每次生产一个产品放入缓冲区，消费者进程每次从缓冲区中取出一个产品并使用。（注：这里的“产品”理解为某种数据）

生产者、消费者共享一个初始为空、大小为n的缓冲区。

只有缓冲区没满时，生产者才能把产品放入缓冲区，否则必须等待。

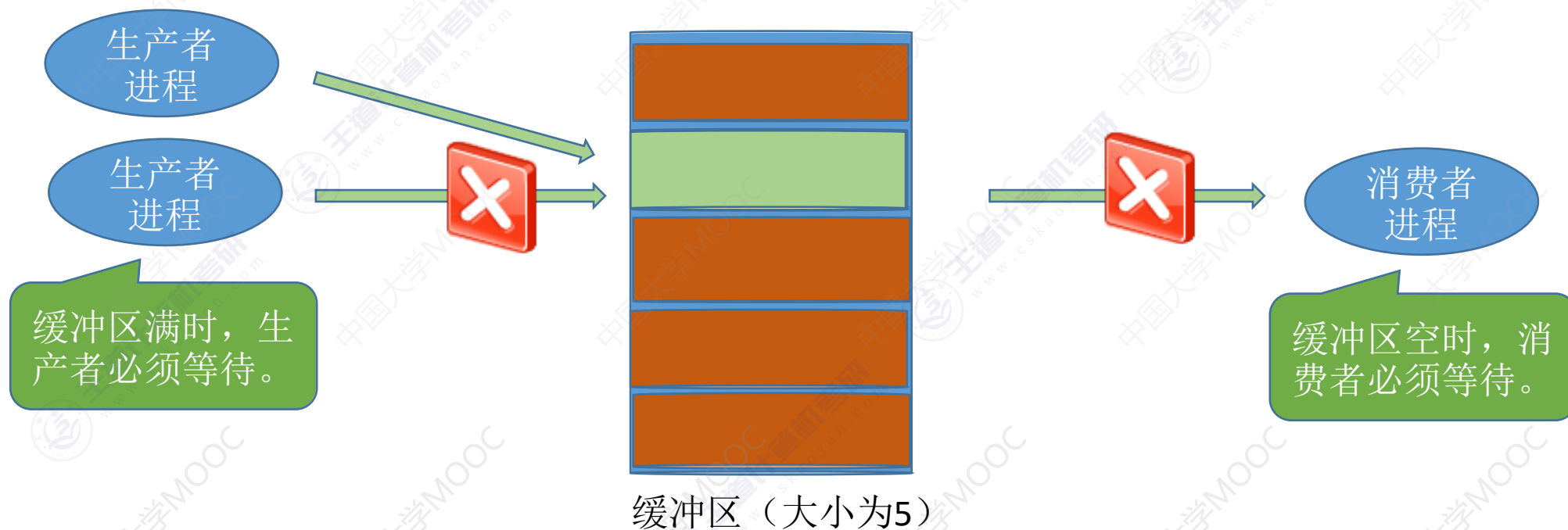
缓冲区没满→生产者生产

只有缓冲区不空时，消费者才能从中取出产品，否则必须等待。

缓冲区没空→消费者消费

缓冲区是临界资源，各进程必须互斥地访问。

互斥关系



## 问题分析

系统中有一组生产者进程和一组消费者进程，生产者进程每次生产一个产品放入缓冲区，消费者进程每次从缓冲区中取出一个产品并使用。（注：这里的“产品”理解为某种数据）

生产者、消费者共享一个**初始为空、大小为n的缓冲区**。

只有**缓冲区没满**时，生产者才能把产品放入缓冲区，否则必须等待。

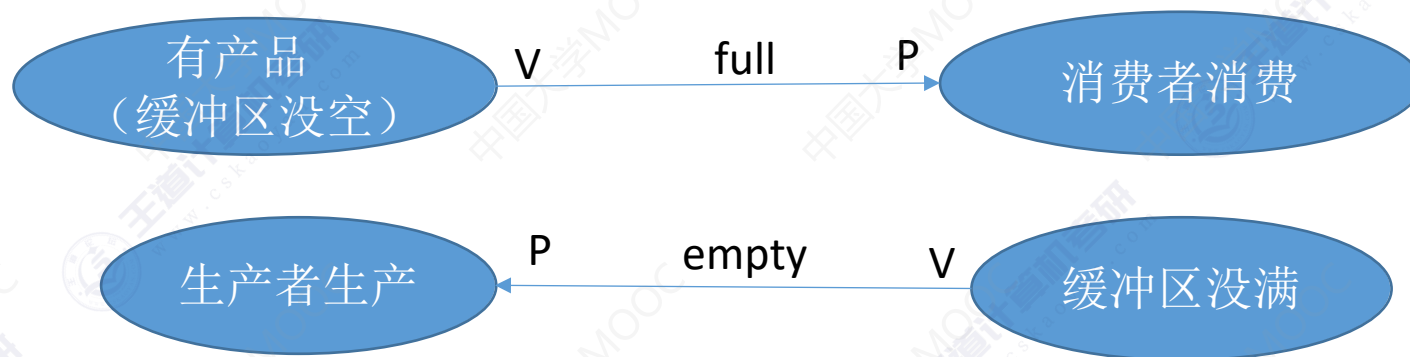
缓冲区没满→生产者生产

只有**缓冲区不空**时，消费者才能从中取出产品，否则必须等待。

缓冲区没空→消费者消费

缓冲区是临界资源，各进程必须**互斥地访问**。

互斥关系



PV操作题目分析步骤：

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。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，同步信号量的初值要看对应资源的初始值是多少）

## 问题分析

系统中有一组生产者进程和一组消费者进程，生产者进程每次生产一个产品放入缓冲区，消费者进程每次从缓冲区中取出一个产品并使用。（注：这里的“产品”理解为某种数据）

生产者、消费者共享一个**初始为空、大小为n的缓冲区**。

只有**缓冲区没满**时，生产者才能把产品放入缓冲区，否则必须等待。

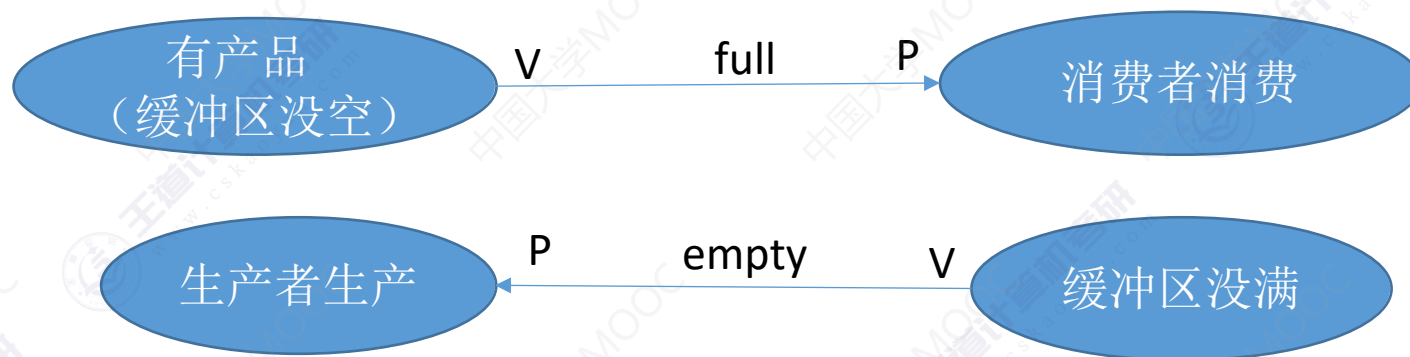
缓冲区没满 → 生产者生产

只有**缓冲区不空**时，消费者才能从中取出产品，否则必须等待。

缓冲区没空 → 消费者消费

缓冲区是临界资源，各进程必须**互斥地访问**。

互斥关系



```
semaphore mutex = 1;  
semaphore empty = n;  
semaphore full = 0;
```

// 互斥信号量，实现对缓冲区的互斥访问

// 同步信号量，表示空闲缓冲区的数量

// 同步信号量，表示产品的数量，也即非空缓冲区的数量

如何

有产品  
(缓冲区没空)

V

full

P

消费者消费

生产者、消费者共享一个初始为空、大小为n的缓冲区。

只有缓冲区没满时，生产者才能把产品放入缓冲区。

只有缓冲区不空时，消费者才能从中取出产品，

缓冲区是临界资源，各进程必须互斥地访问。

生产者生产

P

empty

V

缓冲区没满

```
semaphore mutex = 1;
```

```
semaphore empty = n;
```

```
semaphore full = 0;
```

//互斥信号量，实现对缓冲区的互斥访问

//同步信号量，表示空闲缓冲区的数量

//同步信号量，表示产品的数量，也即非空缓冲区的数量

```
producer () {
```

```
while(1) {
```

```
    生产一个产品;
```

```
    P(empty);
```

消耗一个空闲缓冲区

```
    P(mutex);
```

```
    把产品放入缓冲区;
```

```
    V(mutex);
```

```
    V(full);
```

增加一个产品

```
}
```

```
}
```

实现互斥是在同一进程中进行一对PV操作

```
consumer () {
```

```
while(1) {
```

```
    P(full);
```

消耗一个产品（非空缓冲区）

```
    P(mutex);
```

```
    从缓冲区取出一个产品;
```

```
    V(mutex);
```

```
    V(empty);
```

```
    使用产品;
```

增加一个空闲缓冲区

```
}
```

实现两进程的同步关系，是在其中一个进程中执行P，另一进程中执行V



## 思考：能否改变相邻P、V操作的顺序？

```
producer () {  
    while(1) {  
        ① P(mutex);  
        ② P(empty);  
        把产品放入缓冲区;  
        V(mutex);  
        V(full);  
    }  
}
```

mutex 的P操作在前

生产一个产品;

```
consumer () {  
    while(1) {  
        ③ P(mutex);  
        ④ P(full);  
        从缓冲区取出一个产品;  
        V(mutex);  
        V(empty);  
        使用产品;  
    }  
}
```

能否放到PV操作之间？

使用产品;

若此时缓冲区内已经放满产品，则  $empty=0$ ， $full=n$ 。

则生产者进程执行①使  $mutex$  变为0，再执行②，由于已没有空闲缓冲区，因此生产者被阻塞。由于生产者阻塞，因此切换回消费者进程。消费者进程执行③，由于  $mutex$  为0，即生产者还没释放对临界资源的“锁”，因此消费者也被阻塞。

这就造成了生产者等待消费者释放空闲缓冲区，而消费者又等待生产者释放临界区的情况，生产者和消费者循环等待被对方唤醒，出现“死锁”。

同样的，若缓冲区中没有产品，即  $full=0$ ， $empty=n$ 。按③④①的顺序执行就会发生死锁。

因此，实现互斥的P操作一定要在实现同步的P操作之后。

V操作不会导致进程阻塞，因此两个V操作顺序可以交换。

## 知识回顾与重要考点

PV 操作题目的解题思路：

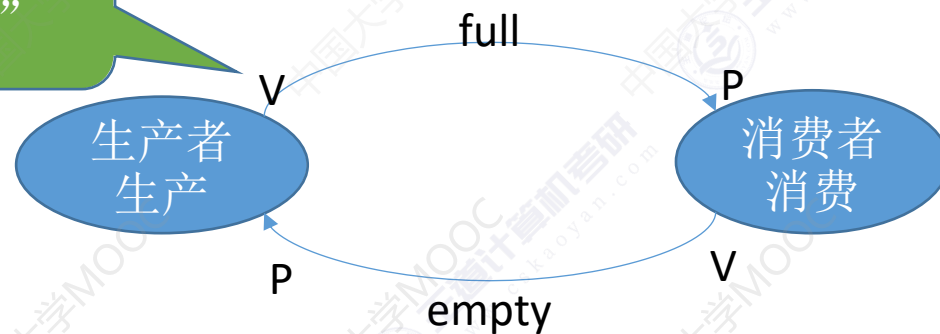
1. 关系分析。找出题目中描述的各个进程，分析它们之间的同步、互斥关系。
2. 整理思路。根据各进程的操作流程确定P、V操作的大致顺序。
3. 设置信号量。设置需要的信号量，并根据题目条件确定信号量初值。（互斥信号量初值一般为1，同步信号量的初始值要看对应资源的初始值是多少）

生产者消费者问题是一个互斥、同步的综合问题。

对于初学者来说最难的是发现题目中隐含的两对同步关系。

有时候是消费者需要等待生产者生产，有时候是生产者要等待消费者消费，这是两个不同的“一前一后问题”，因此也需要设置两个同步信号量。

实现“一前一后”  
需要“前V后P”



**易错点：**实现互斥和实现同步的两个P操作的先后顺序（死锁问题）

## 问题分析

系统中有一组生产者进程和一组消费者进程，生产者进程每次生产一个产品放入缓冲区，消费者进程每次从缓冲区中取出一个产品并使用。（注：这里的“产品”理解为某种数据）  
生产者、消费者共享一个初始为空、大小为 $n$ 的缓冲区。  
只有缓冲区没满时，生产者才能把产品放入缓冲区，否则必须等待。  
只有缓冲区不空时，消费者才能从中取出产品，否则必须等待。  
缓冲区是临界资源，各进程必须互斥地访问。

刚开始空闲缓冲区的数量为 $n$ ，

生产者每次生产一个产品放入缓冲区，消费者每次从缓冲区中取出一个产品并使用。

同步关系。缓冲区满时，生产者要等待消费者取走产品

同步关系。缓冲区空时（即没有产品时），消费者要等待生产者放入产品

互斥



如何用信号量机制（P、V操作）实现生产者、消费者进程的这些功能呢？  
信号量机制可实现互斥、同步、对一类系统资源的申请和释放。

设置初值为1的互斥信号量

设置初值为0的同步信号量（实现“一前一后”）

设置一个信号量，初始值即为资源的数量（本质上属于“同步问题”，若无空闲资源，则申请资源的进程需要等待别的进程释放资源后才能继续往下执行）

PV操作题目分析步骤：

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的初值（1，同步信号量的初始值要看对应资源的初始值是多少）

生产者每次要消耗（P）一个空闲缓冲区，并生产（V）一个产品。  
消费者每次要消耗（P）一个产品，并释放一个空闲缓冲区（V）。  
往缓冲区放入/取走产品需要互斥。



## 问题分析

系统中有一组生产者进程和一组消费者进程，生产者进程每次从缓冲区中取出一个产品并使用。（注：这里的生产者、消费者共享一个**初始为空、大小为n的缓冲区**。

只有**缓冲区没满**时，生产者才能把产品放入缓冲区，否则必须等待。

只有**缓冲区不空**时，消费者才能从中取出产品，否则必须等待。

缓冲区是临界资源，各进程必须**互斥地访问**。

刚开始空闲缓冲区的数量为n，  
非空闲缓冲区（产品）的数量为0

同步关系。缓冲区满时，生产者要等待消费者取走产品

同步关系。缓冲区空时（即没有产品时），消费者要等待生产者放入产品

互斥



如何用信号量机制（P、V操作）实现生产者、消费者进程的这些功能呢？  
信号量机制可实现互斥、同步、对一类系统资源的申请和释放。

设置初值为1的互斥信号量

设置初值为0的同步信号量（实现“一前一后”）

设置一个信号量，初始值即为资源的数量（本质上属于“同步问题”，若无空闲资源，则申请资源的进程需要等待别的进程释放资源后才能继续往下执行）

```
semaphore mutex = 1;  
semaphore empty = n;  
semaphore full = 0;
```

//互斥信号量，实现对缓冲区的互斥访问

//同步信号量，表示空闲缓冲区的数量

//同步信号量，表示产品的数量，也即非空缓冲区的数量



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