目录

[一、实验内容： 2](#_Toc167532443)

[二、实验目的： 2](#_Toc167532444)

[三、实验要求： 2](#_Toc167532445)

[四、实现方法： 2](#_Toc167532446)

[4.1 基本思想 2](#_Toc167532447)

[4.2 整体程序运行逻辑 3](#_Toc167532448)

[1.初始化 3](#_Toc167532449)

[2. 输入进程信息 3](#_Toc167532450)

[3. 模拟进程调度和执行 3](#_Toc167532451)

[4. 结束 4](#_Toc167532452)

[4.3 相关数据结构 4](#_Toc167532453)

[1-PCB结构体： 4](#_Toc167532454)

[2-QNode队列结点 4](#_Toc167532455)

[3-LinkQueue (队列指针节点) 5](#_Toc167532456)

[4-定义就绪队列、生产者等待队列、消费者等待队列、完成队列的头尾指针。定义缓冲区队列，产品个数、产品队列的队头队尾 5](#_Toc167532457)

[5-定义相关资源量：mutex临界资源，互斥信号量，【仓库个数】 ：一次只有一个进程在运行，empty\_，空缓存区【篮子个数】，full，满缓冲区【产品数】 5](#_Toc167532458)

[4.4 相关详细功能代码 5](#_Toc167532459)

[4.4.1 初始化：（定义一些队列的相关操作以及相关变量的赋初值） 5](#_Toc167532460)

[4.4.2主函数： 8](#_Toc167532461)

[4.4.3P操作：（-1） 11](#_Toc167532462)

[4.4.4V操作：（+1） 11](#_Toc167532463)

[4.4.5唤醒消费者： 11](#_Toc167532464)

[4.4.6唤醒生产者： 12](#_Toc167532465)

[4.4.7特别说明在展示函数中的指针：（head，tail） 12](#_Toc167532466)

[五、实验结果： 12](#_Toc167532467)

[5.1 定义缓存区的大小为2，然后依次输入0,1，创建3个进程。 12](#_Toc167532468)

[5.2 第一轮模拟，进程依次执行直到就绪队列为空 12](#_Toc167532469)

[5.3 第二轮模拟，继续输入进程，依次执行直到就绪队列为空 14](#_Toc167532470)

[六、总结： 16](#_Toc167532471)

# 一、实验内容：

1. 由用户指定要产生[的](http://www.tiandige.com/" \t "_blank)进程及其类别[，](http://www.tiandige.com/)存入进入就绪队列[。](http://www.tiandige.com/)

1. 调度程序从就绪队列中提取一个就绪进程运行[。](http://www.tiandige.com/" \t "_blank)如果申请[的](http://www.tiandige.com/)资源被阻塞则进入相应[的](http://www.tiandige.com/)等待队列[，](http://www.tiandige.com/)调度程序调度就绪队列中[的](http://www.tiandige.com/)下一个进程[。](http://www.tiandige.com/)进程运行结束时[，](http://www.tiandige.com/)会检查对应[的](http://www.tiandige.com/)等待队列[，](http://www.tiandige.com/)激活队列中[的](http://www.tiandige.com/)进程进入就绪队列[。](http://www.tiandige.com/)运行结束[的](http://www.tiandige.com/)进程进入over链表[。](http://www.tiandige.com/" \t "_blank)重复这一过程直至就绪队列为空[。](http://www.tiandige.com/)

1. 程序询问是否要继续？如果要转至①开始执行[，](http://www.tiandige.com/)否则退出程序[。](http://www.tiandige.com/)

# 二、实验目的：

通过实验模拟生产者与消费者之间的关系，了解并掌握他们之间的关系及其原理。由此增加对进程同步的问题的了解。

# 三、实验要求：

每个进程有一个进程控制块（PCB）表示[。](http://www.tiandige.com/" \t "_blank)进程控制块可以包含如下信息：进程类型标号、进程系统号、进程状态、进程产品（字符）、进程链指针等等[。](http://www.tiandige.com/" \t "_blank)

系统开辟了一个缓冲区[，](http://www.tiandige.com/" \t "_blank)大小由buffersize指定[。](http://www.tiandige.com/" \t "_blank)

程序中有三个链队列[，](http://www.tiandige.com/" \t "_blank)一个链表[。](http://www.tiandige.com/)一个就绪队列（ready）[，](http://www.tiandige.com/)两个等待队列：生产者等待队列（producer）；消费者等待队列（consumer）[。](http://www.tiandige.com/)一个链表（over）[，](http://www.tiandige.com/)用于收集已经运行结束[的](http://www.tiandige.com/)进程

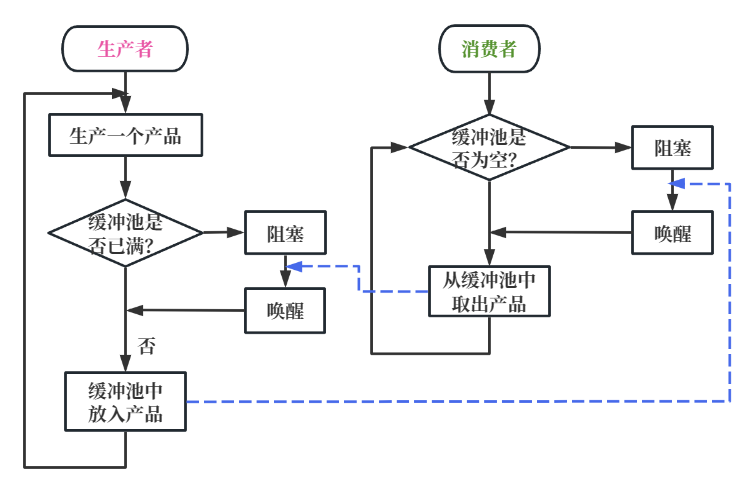
本程序通过函数模拟信号量[的](http://www.tiandige.com/" \t "_blank)原子操作[。](http://www.tiandige.com/)

# 四、实现方法：

## 4.1 基本思想

算法原理：

1. 生产者不断的向缓冲区中写入数据，消费者不断的从缓冲区中读取数据；
2. 生产者和消费者对缓冲区的操作是互斥的，即当前只能由一个进程对这个缓冲区进行操作；
3. 生产者生产时要判断当前缓冲区是否已满，当缓冲区满时，不能再写入，消费者消费时要判断当前缓冲区是否已空，当缓冲区空时，不能再读取；



## 4.2 整体程序运行逻辑

### 1.初始化

* **输入参数**：
  + 用户输入要加入的进程数量 **count** 和缓冲区容量 **BUFFERSIZE**。
* **初始化数据结构**：
  + 初始化就绪队列 **ready**。
  + 初始化生产者等待队列 **producer** 和消费者等待队列 **customer**。
  + 初始化已完成进程链表 **over**。

### 2. 输入进程信息

* 用户依次输入每个进程的信息，包括进程类型（生产者或消费者）、进程号和产品。
* 所有输入的进程都被加入到就绪队列 **ready** 中。

### 3. 模拟进程调度和执行

* 进入主调度循环 **Process**，模拟系统的运行。
* 在每一轮中，从就绪队列 **ready** 中取出一个进程进行处理。
  + **处理生产者进程**：
    - 检查缓冲区是否有足够空间放入新产品。
      * 如果有空间：
        + 将产品放入缓冲区。
        + 检查消费者等待队列 **customer**，如果有等待的消费者进程，将其激活并加入就绪队列 **ready** 的队头。
        + 将当前生产者进程加入已完成进程链表 **over**。
      * 如果没有空间：
        + 将生产者进程加入生产者等待队列 **producer**。
  + **处理消费者进程**：
    - 检查缓冲区是否有产品可供消费。
      * 如果有产品：
        + 从缓冲区取出产品。
        + 检查生产者等待队列 **producer**，如果有等待的生产者进程，将其激活并加入就绪队列 **ready** 的队头。
        + 将当前消费者进程加入已完成进程链表 **over**。
      * 如果没有产品：
        + 将消费者进程加入消费者等待队列 **customer**。
* 在每一轮结束后，调用 **Display** 函数打印当前系统状态，包括就绪队列、生产者等待队列、消费者等待队列、已完成进程链表、缓冲区数据量和缓冲区剩余存储量。

### 4. 结束

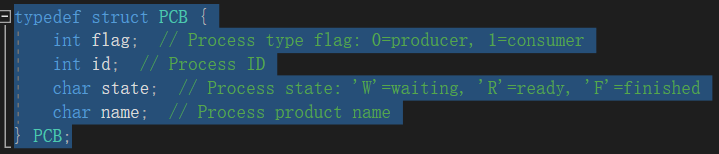
* 主调度循环 **Process** 结束，当就绪队列 **ready** 为空时，所有进程均已处理完毕，程序结束。

## 4.3 相关数据结构

### 1-PCB结构体：

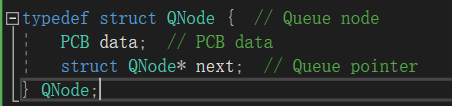
创建 PCB 结构体，每个进程用一个进程控制块 PCB 来代表，其中包含：

**进程类型标号:** flag=0生产者，flag=1消费者；**进程系统号**: 整个系统中该进程的编号，**进程状态：** W等待 R就绪 F完成；**进程产品名:** 用于标记产品名称，不同生产者会生产不同产品，消费者也会消费不同产品，但是在该实验中消耗和使用的产品都是一种产品 。



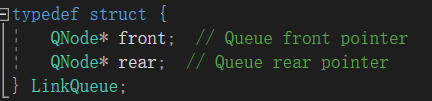
### 2-QNode队列结点

的定义与实验1相同：队列是一种线性数据结构，具有先进先出的特性，它可以用来存储需要按照顺序执行的任务，如操作系统中的进程队列，我们使用队列结点类型**QNode，其中包含了一个PCB数据和一个指向下一个队列结点的指针。**

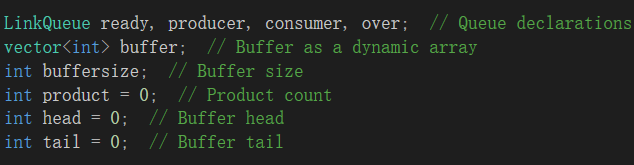


### 3-LinkQueue (队列指针节点)

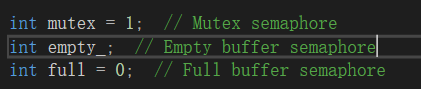
定义Queue (队列节点)，用于实现队列的数据结构，包含进程数据和指向下一个队列节点的指针。



### 4-定义就绪队列、生产者等待队列、消费者等待队列、完成队列的头尾指针。定义缓冲区队列，产品个数、产品队列的队头队尾



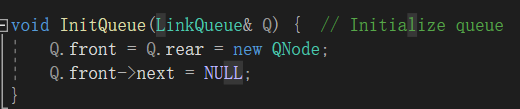
### 5-定义相关资源量：mutex临界资源，互斥信号量，【仓库个数】 ：一次只有一个进程在运行，empty\_，空缓存区【篮子个数】，full，满缓冲区【产品数】



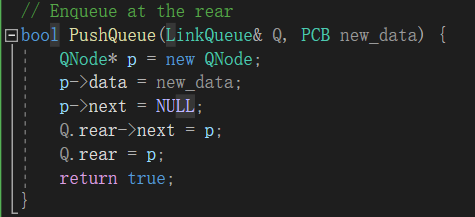
## 4.4 相关详细功能代码

### 4.4.1 初始化：（定义一些队列的相关操作以及相关变量的赋初值）

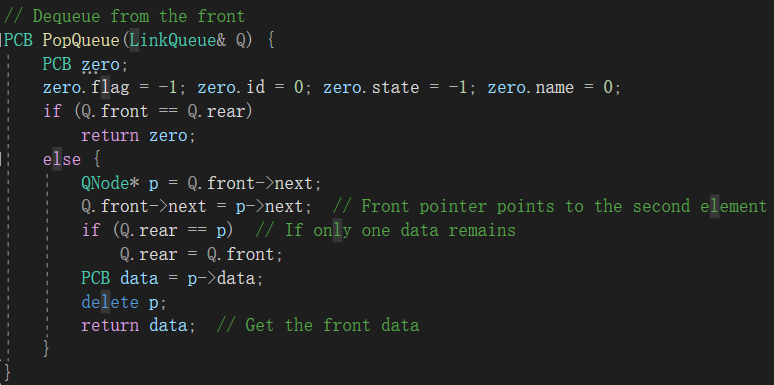
**初始化队列：**



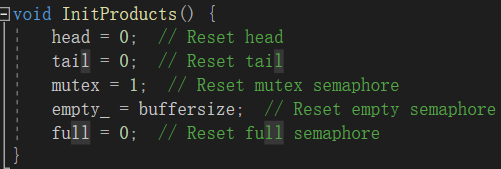
**尾插法入队：**



**队头出队：**



**初始化相关变量：**



定义展示的相关内容：

**DisplayBuffer**

这个函数用来展示缓冲区的当前状态，包括缓冲区内每个位置的内容（是否有产品）、产品总数、缓冲区的输入指针（tail）和输出指针（head）。

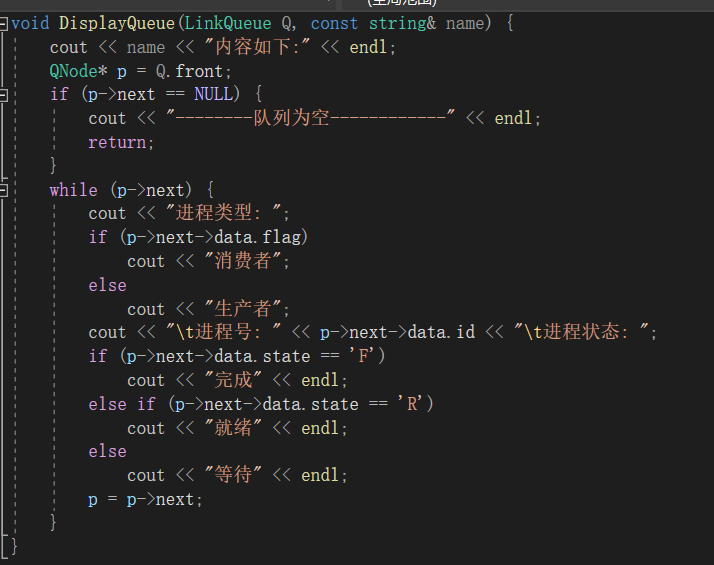
* **缓冲区显示**：遍历缓冲区数组，显示每个位置的内容。如果位置为空（值为-1），显示“空”；否则显示该位置的产品编号。
* **状态信息显示**：
  + 显示当前缓冲区中产品的总数。
  + 显示放入产品的位置（缓冲区的尾部指针**tail**）。
  + 显示取出产品的位置（缓冲区的头部指针**head**）。



**DisplayQueue：**

这个函数用来显示一个队列的内容，它接受一个队列和该队列的名称作为参数。函数详细展示队列中的每个进程的类型、进程号和状态。

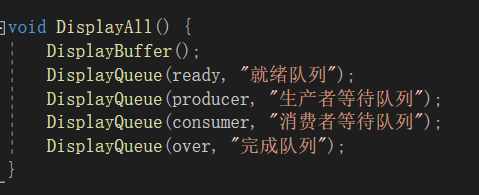
* **队列名称显示：**打印传入的队列名称，以标识当前展示的队列。
* **队列内容显示：**
  + 遍历队列中的每个节点。
  + 对于每个进程，显示其类型（生产者或消费者）、进程号以及当前状态（完成、就绪或等待）。



**DisplayAll：**

这个函数是一个综合展示函数，调用上述的 **DisplayBuffer** 和三次 **DisplayQueue** 来展示所有相关的状态信息。

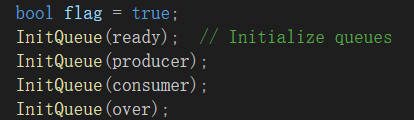
* 调用 DisplayBuffer：显示缓冲区的详细状态。
* 调用 DisplayQueue：
  + 展示就绪队列的状态。
  + 展示生产者等待队列的状态。
  + 展示消费者等待队列的状态。
  + 展示已完成的进程队列的状态。



### 4.4.2主函数：

**1-初始化程序中的三个链队列**：

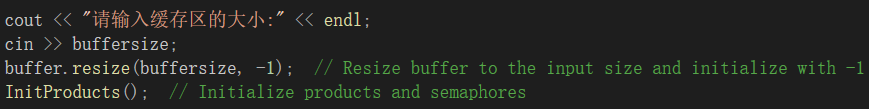
一个链表:一个就绪队列（**ready**）[，](http://www.tiandige.com/)两个等待队列：生产者等待队列（**producer**）；消费者等待队列（**consumer**）[。](http://www.tiandige.com/)一个链表（**over**）[，](http://www.tiandige.com/)用于收集已经运行结束[的](http://www.tiandige.com/)进程



**2-输入缓冲区容量和进程数量：**

输入缓冲区大小；输入缓冲区的大小 **buffersize**，并调整 **buffer** 的大小，并初始化为 **-1**（表示空）。

调用 **InitProducts()** 初始化产品数量和信号量。



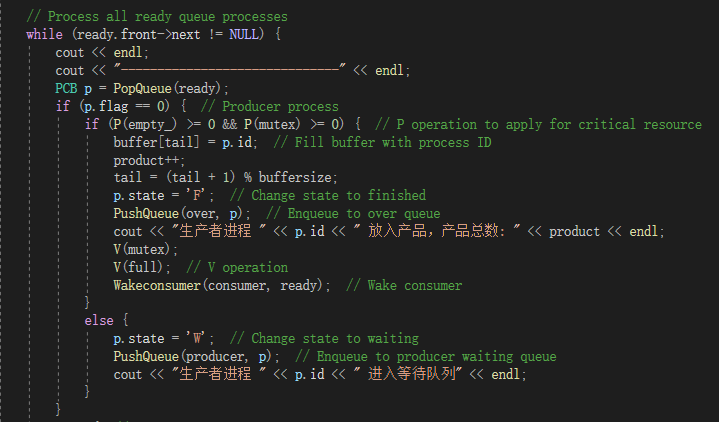
**3-创建进程：**

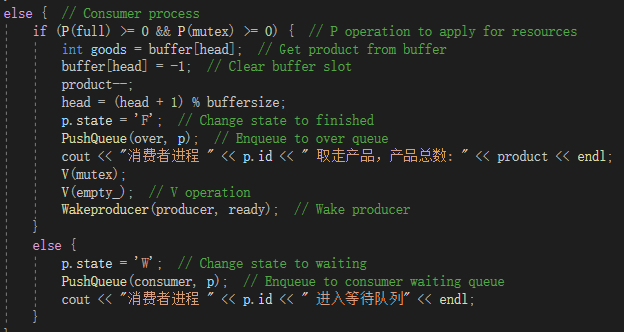
* 输入需要创建的进程数量 **num**。
* 对每个进程，创建 **PCB**，设置进程 ID 和状态。
* 根据用户输入设置进程类型（生产者或消费者），并将进程加入就绪队列 **ready**。



**4-处理就绪队列中的进程**

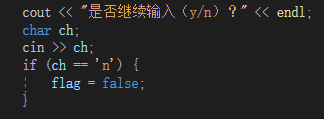
* 从就绪队列 **ready** 中取出进程 **p**。
* **如果是生产者进程**：
  + 尝试通过 **P(empty\_)** 和 **P(mutex)** 操作申请资源。
  + 如果成功，放入产品，更新缓冲区，调整指针，设置状态为完成，并加入完成队列 **over**，唤醒消费者。
  + 如果申请资源失败，将进程状态设为等待，并加入生产者等待队列 **producer**。
* **如果是消费者进程**：
  + 尝试通过 **P(full)** 和 **P(mutex)** 操作申请资源。
  + 如果成功，取出产品，更新缓冲区，调整指针，设置状态为完成，并加入完成队列 **over**，唤醒生产者。
  + 如果申请资源失败，将进程状态设为等待，并加入消费者等待队列 **consumer**。
* 每处理完一个进程后，调用 **DisplayAll** 函数显示当前系统状态。





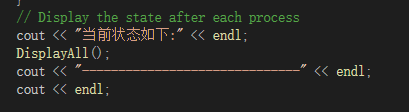
**5. 循环控制**

* 询问用户是否继续输入新进程。
* 如果用户输入 'n'，设置 **flag** 为 **false** 以终止主循环。



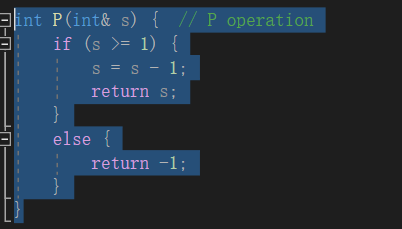
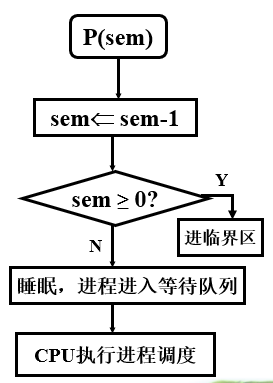
**6. 显示最终状态**

在主循环结束后，调用 DisplayAll 函数显示最终的系统状态。



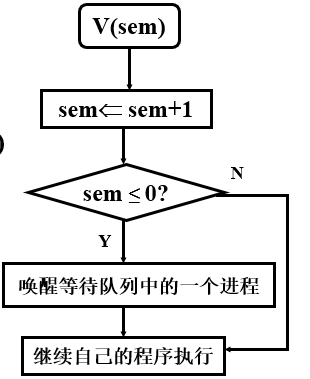
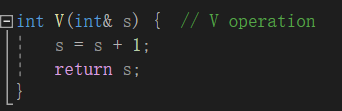
### 4.4.3P操作：（-1）

S=S-1; 若S≥0，则调用P(S)的进程继续运行；若S<0，则调用P(S)的进程被阻塞，并把它插入到等待信号量S的阻塞队列中。



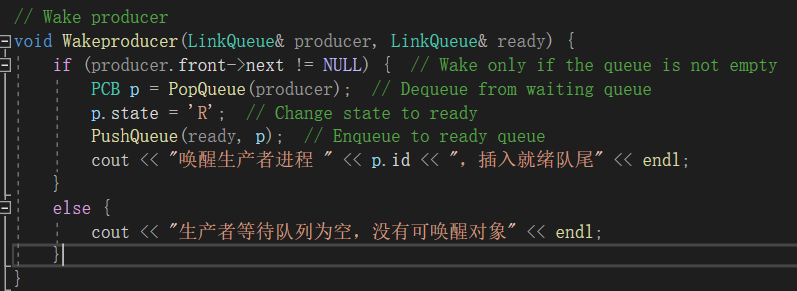
### 4.4.4V操作：（+1）

S=S+1; 若S>0，则调用V(S)的进程继续运行；若S≤0，从等待信号量S的阻塞队列中唤醒头一个进程， 然后调用V(S)的进程继续运行

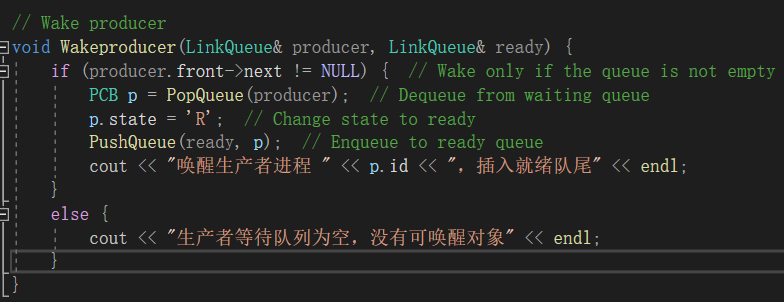
### 4.4.5唤醒消费者：

其实就是**将进程从消费者等待队列改为就绪队列**：如果消费者等待队列不为空，将消费者等待队列的队头出队列，改状态为就绪态，插入就绪队列的队尾。否则没有可以唤醒的对象。因此传入的是consumer等待队列，ready就绪队列。



### 4.4.6唤醒生产者：

与上述相同，**将进程从生产者等待队列改为就绪队列**：如果生产者等待队列不为空，将生产者等待队列的队头出队列，改状态为就绪态，插入就绪队列的队尾。否则没有可以唤醒的对象。因此传入的是producer等待队列，ready就绪队列。



### 4.4.7特别说明在展示函数中的指针：（head，tail）

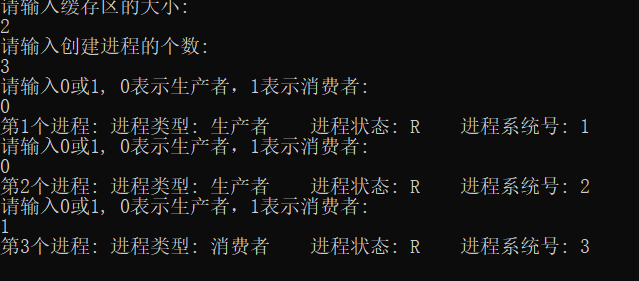
在生产者-消费者模型中，缓冲区通常被实现为循环队列。生产者将产品放入缓冲区，而消费者从缓冲区中取出产品。为了管理这个过程，使用两个指针：

* 尾部指针 tail：指向下一个可以放入产品的位置。
* 头部指针 head：指向下一个可以取出产品的位置。

# 五、实验结果：

## 5.1 定义缓存区的大小为2，然后依次输入0,1，创建3个进程。

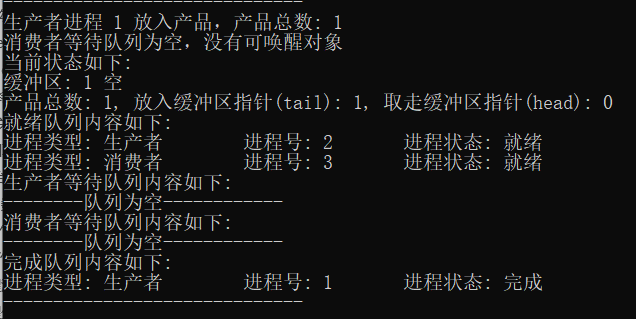
第一次创建了3个进程，然后依次输入 0（生产者）、0（生产者）、1（消费者）



## 5.2 第一轮模拟，进程依次执行直到就绪队列为空

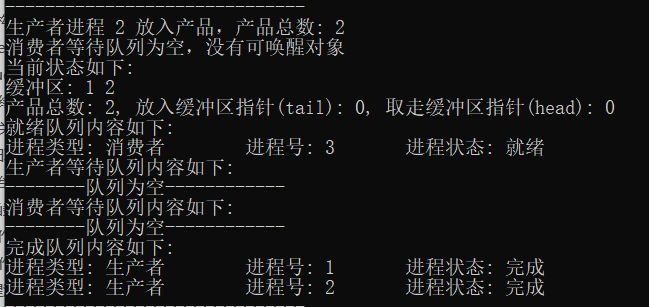
**第一次执行**，将就绪队列队头出队，为生产者进程1，此时缓冲区为空，生产者进程 1 将产品放入，产品总数: 1；

更新tail和head指针，以及相关内容。

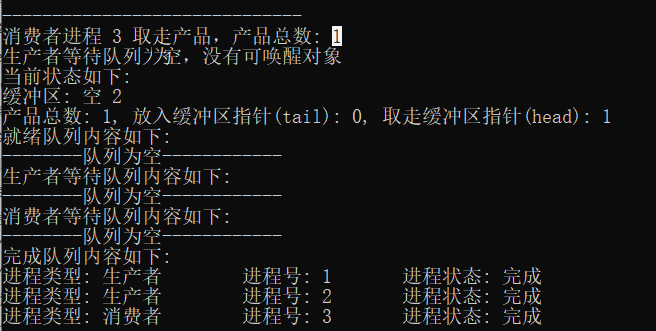


**第二次执行**，就绪队列队头出队，此时为生产者进程2，此时缓存区有空位，放入产品，产品总数: 2；消费者等待队列为空，没有可唤醒对象。

更新指针和其他内容如下图所示。



**第三次执行，**就绪队列队头出队，为消费者进程3，此时缓冲区中有产品，取走产品，此时产品总数: 1。根据head指针指向的位置，将生产者进程1中生产的产品取走，同时更新指针head和相关内容。

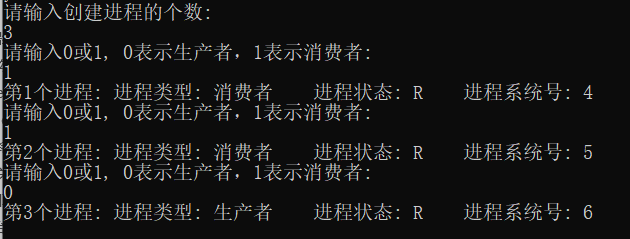


就绪队列为空后，本次模拟结束，选择是否继续下一次：

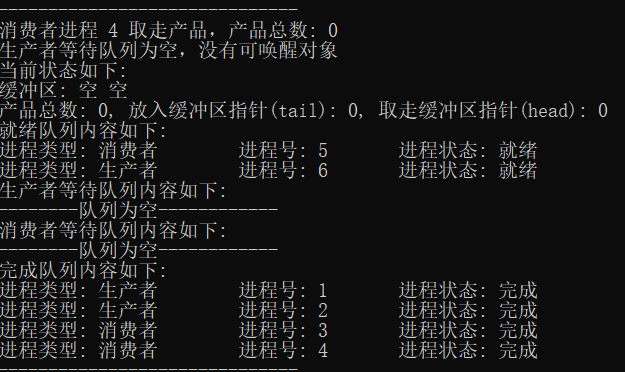


## 5.3 第二轮模拟，继续输入进程，依次执行直到就绪队列为空

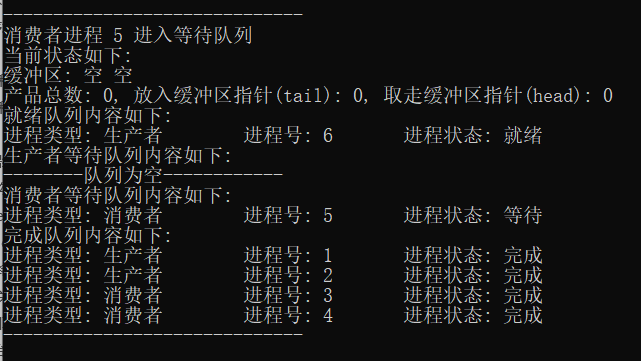
第二次创建了3个进程，然后依次输入 1（消费者），1（消费者），0（生产者）



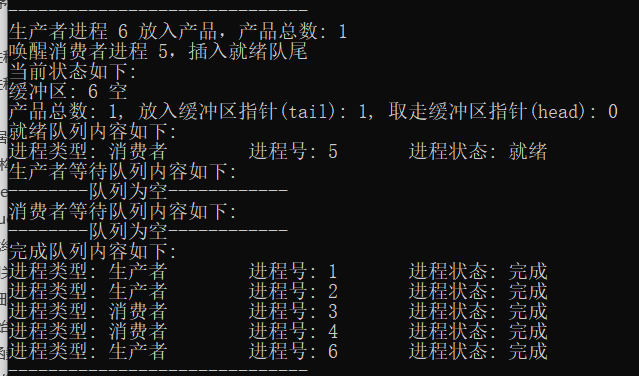
第四次执行，就绪队列队头出队，为消费者进程4，此时缓存区中有产品剩余，可以执行，取走产品，此时产品总数: 0。更新head指针和其他队列等相关信息如下：



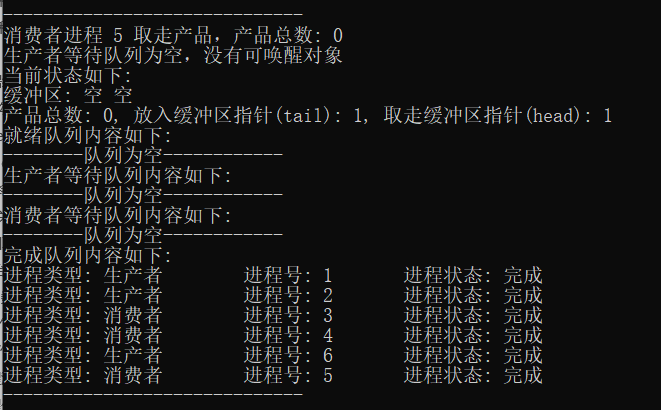
第五次执行，就绪队列队头出队，为消费者进程5，此时缓存区产品数量为0，无法执行，因此该进程进入等待队列。更新相关信息如下：



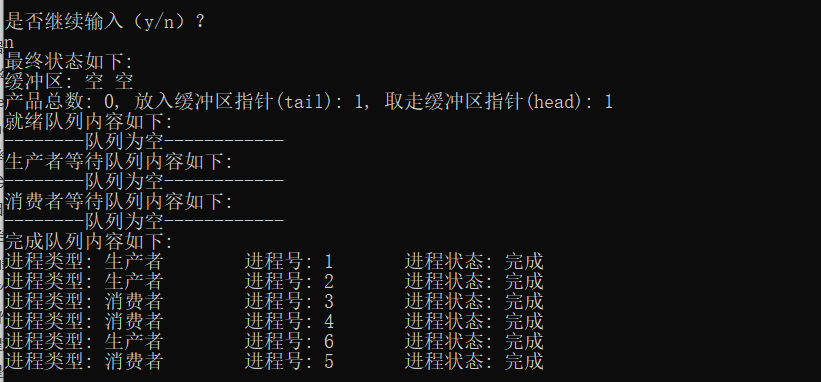
第六次执行，就绪队列队头出队，为生产者进程6，判断缓冲区中可以放入产品。可以执行，放入产品，产品总数: 1。同时**唤醒消费者进程 5**，插入就绪队尾。更新指针（tail），以及相关队列信息。



第七轮，就绪队列队头出队，为消费者进程5，检测缓冲区有产品，取走产品，此时产品总数: 0。更新缓冲区指针（head）以及相关的队列信息。



就绪队列为空后，本次模拟结束，选择是否继续下一次，输入n之后，程序输出最终的状态，结束：



六、总结：

在这次的实验中，我通过模拟生产者消费者问题，深化了对进程同步问题的理解。生产者消费者问题的核心在于确保生产动作与消费动作的有序进行——即只有当产品得到生产之后，消费者才能执行消费操作；当缓冲区达到饱和时，又必须等待消费者消费后，生产者才能再次生产。这样的过程体现了在系统中，尽管各个进程可能异步执行，但它们之间需要通过同步机制来协同完成任务。

进程间通信是多进程系统中不可或缺的组成部分，它允许信息在独立的进程之间传递，以实现这些进程之间的协同工作。同步机制和互斥机制作为进程通信的一种方式，它借助于操作系统提供的原语，如信号量的P（等待）操作和V（信号）操作，来协调进程活动。

在此次实验中，我采用了互斥信号量mutex（代表仓库容量）来确保在任一时刻，仅有生产者或消费者之一可以访问共享资源。此外，还使用了信号量empty来表示空缓冲区的数量，进而判断是否还能继续放置产品；以及信号量full来记录缓冲区中的产品数量，确保不会超过缓冲区容量。

由于此次实验所涉及的数据结构与前次实验相似，我的对进程队列的操作技能得到了进一步加强。综上所述，这一系列的实验活动不仅加深了我对进程同步机制和进程间通信方式的理解，也提升了对相关数据结构的应用熟练度，为后续更高级的并发编程打下了坚实的基础。