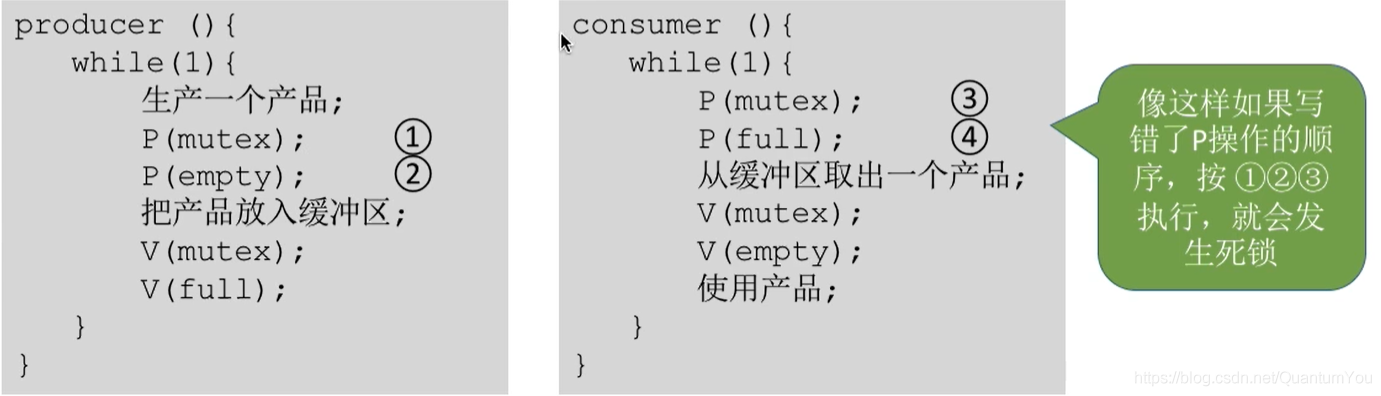
静心 尽力 沉淀

@[toc]

# 管程

## 出现原因

* 信号量机制存在的问题：编写程序困难、易岀错
* 能不能设计一种机制，让程序员写程序时不需要再关注复杂的PV操作，让写代码更轻松呢
* 1973年， Brinch Hansen首次在程序设计语言（ Pascal中引入了“管程”成分一一一种高级同步机制



## 管程定义

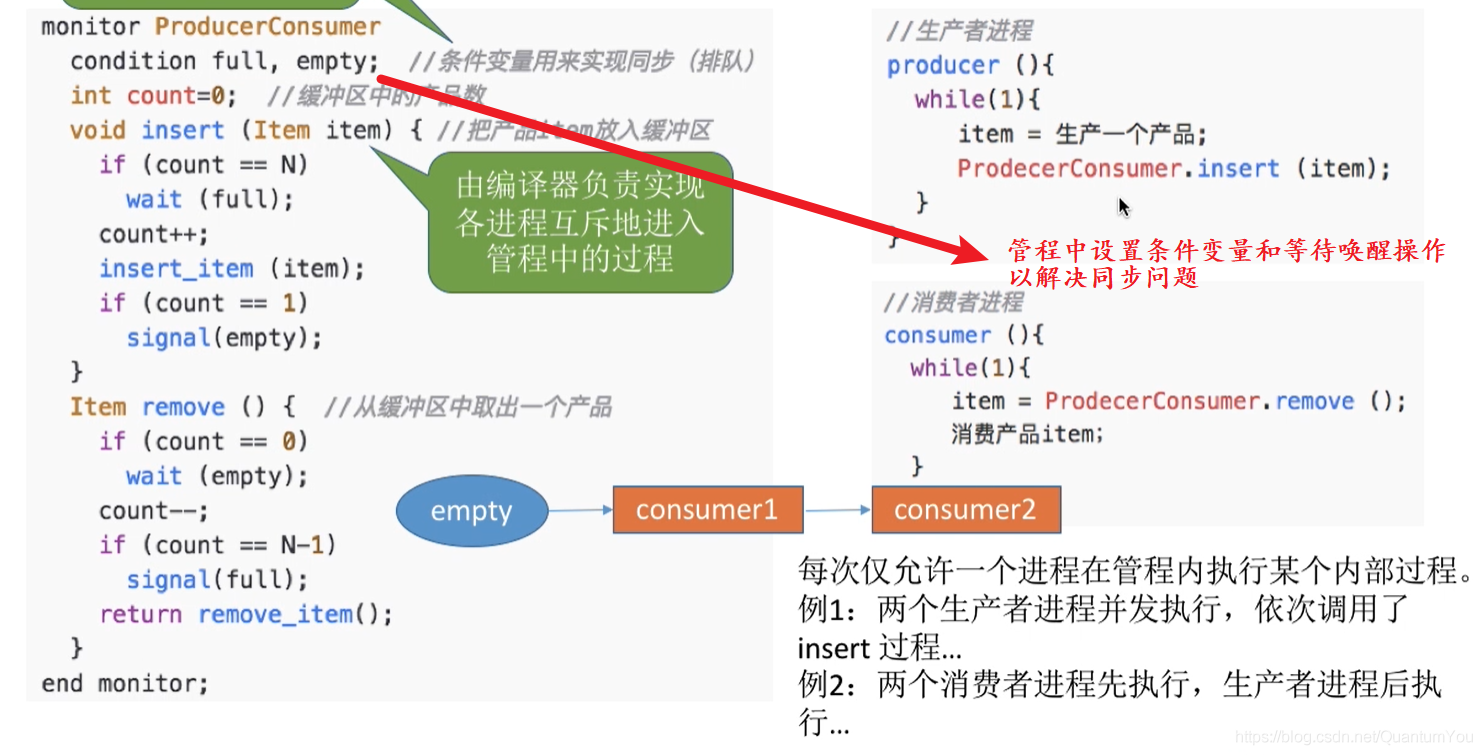
管程是一种特殊的软件模块，有这些部分组成：

* 1、局部于管程的共享数据构说明
* 2、对该数据结构进行操作的一组过程（函数）；
* 3、对局部于管程的共享数据设置初始值的语句；
* 4、管程有一个名字。
* 管程的定义有点类似于类的定义

## 管程特征

* 1、局部于管程的数据只能被局部于管程的过程所访问
* 2、一个进程只有通过调用管程内的过程才能进入管程访问共享数据；
* 3、每次仅允许一个进程在管程内执行某个内部过程。

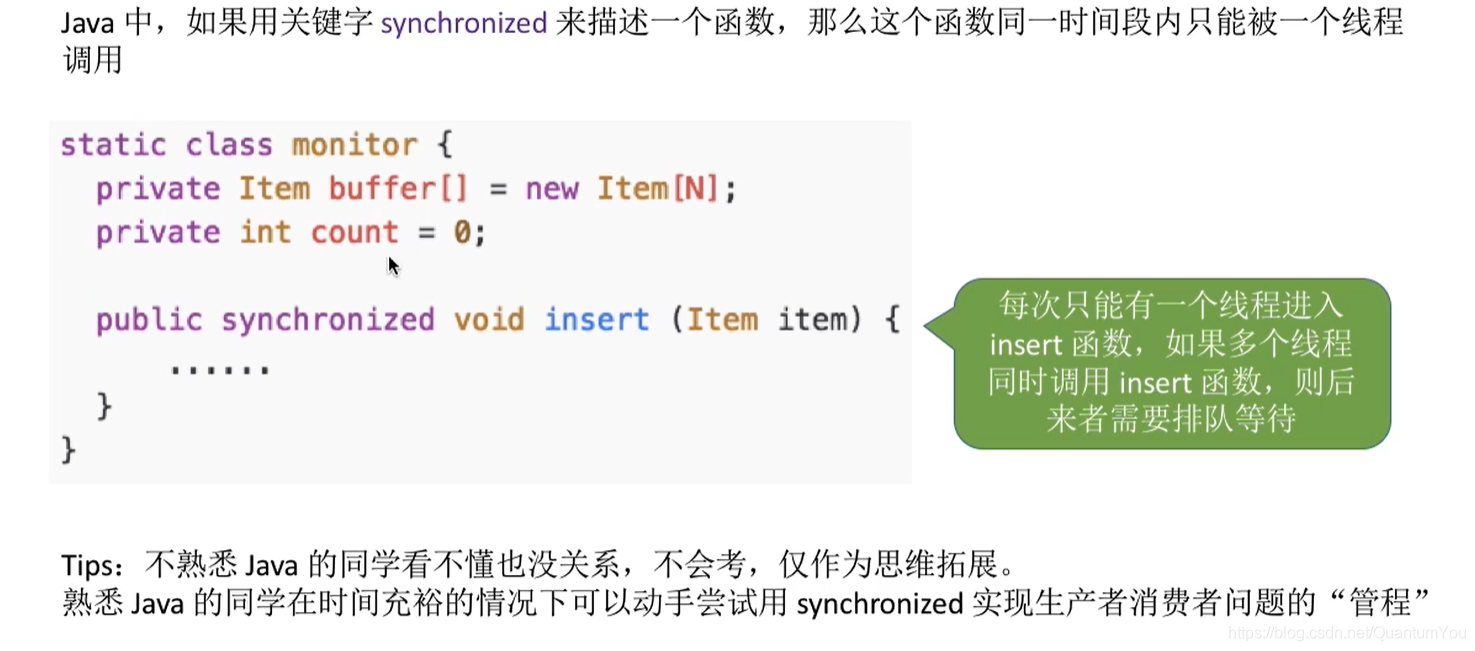
## 用管程解决生产者消费者问题



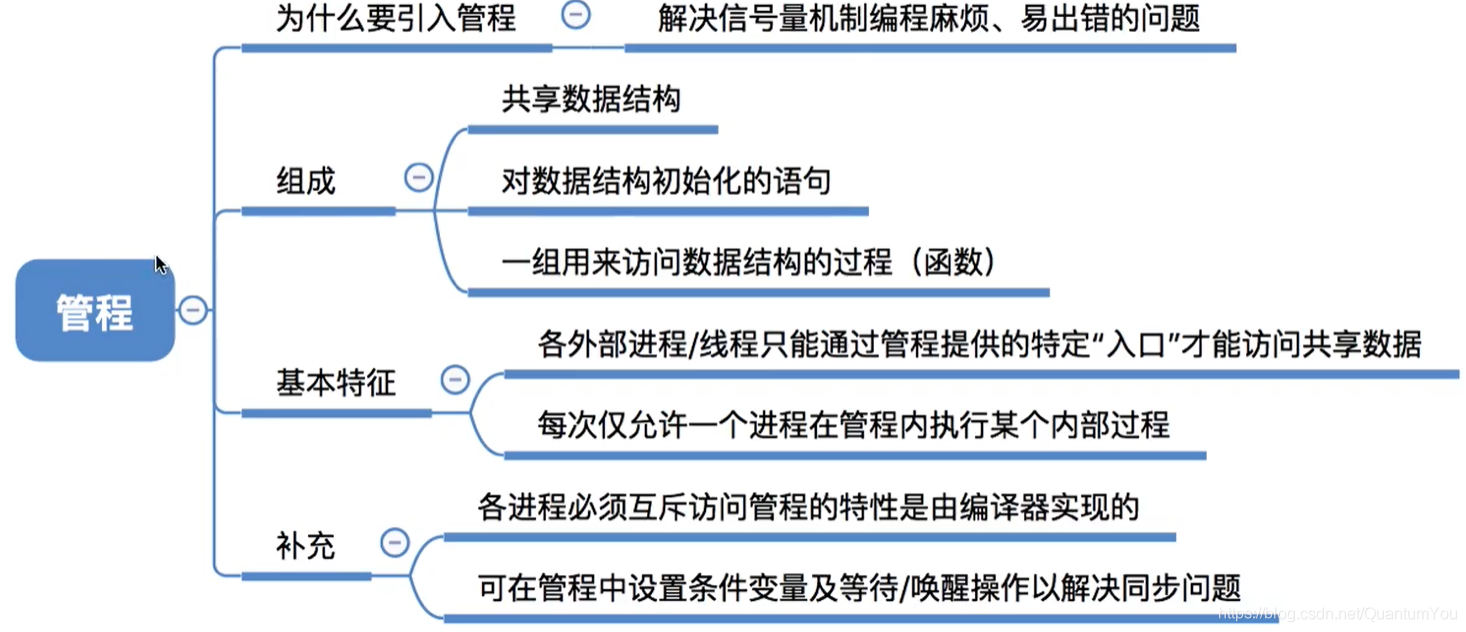
引入管程的目的无非就是要更方便地实现进程互斥和同步。

* 1、需要在管程中定义共享数据（如生产者消费者问题的缓冲区)
* 2、需要在管程中定义用于访问这些共享数据的“入口”一一其实就是一些函数（如生产者消费者问题中，可以定义一个函数用于将产品放入缓冲区，再定义一个函数用于从缓冲区取出产品）
* 3、只有通过这些特定的“入口”才能访问共享数据
* 4、管程中有很多“入口”，但是每次只能开放其中一个“入口”，并且只能让一个进程或线程进入（如生产者消费者问题中，各进程需要互斥地访问共享缓冲区。管程的这种特性即可保证一个时间段内最多只会有一个进程在访问缓冲区。==注意：这种互斥特性是由编译器负责实现的  
  程序员不用关心）
* 5、可在管程中设置条件变量及等待/唤醒操作以解决同步问题。可以让一个进程或线程在条件变量上等待（此时，该进程应先释放管程的使用权，也就是让出“入口”）；可以通过唤醒操作将等待在条件变量上的进程或线程唤醒。
* 程序员可以用某种特殊的语法定义一个管程（比如： monitor ProducerConsumer∴….end monitor；）之后其他程序员就可以使用这个管程提供的特定“入口”很方便地使用实现进程同步/互斥了

## Java 中类似于管程的机制



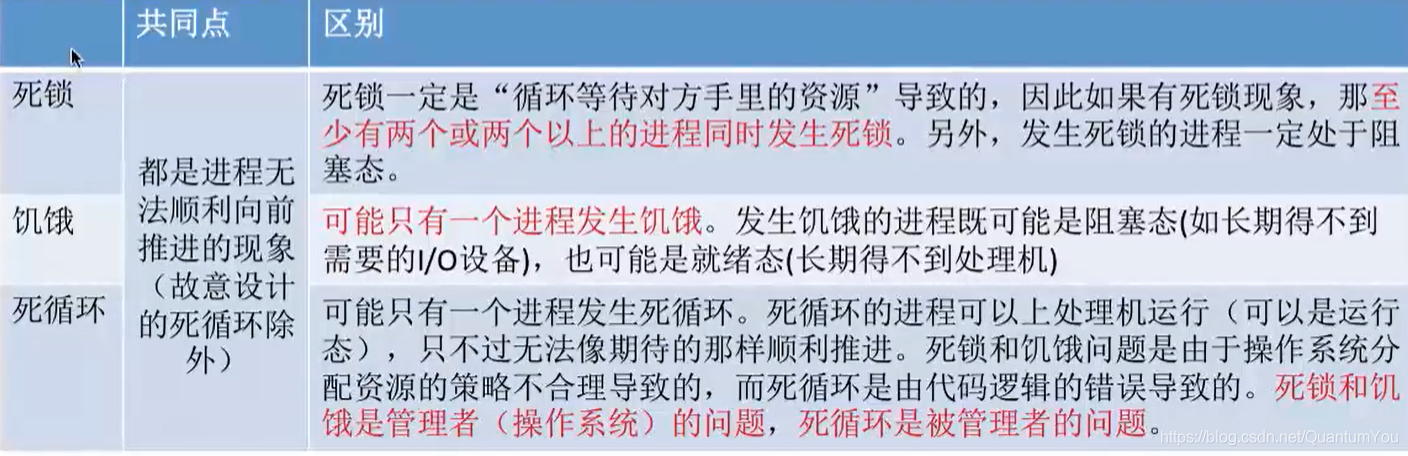
## 小结

  
注意：管程的基本特征

# 死锁

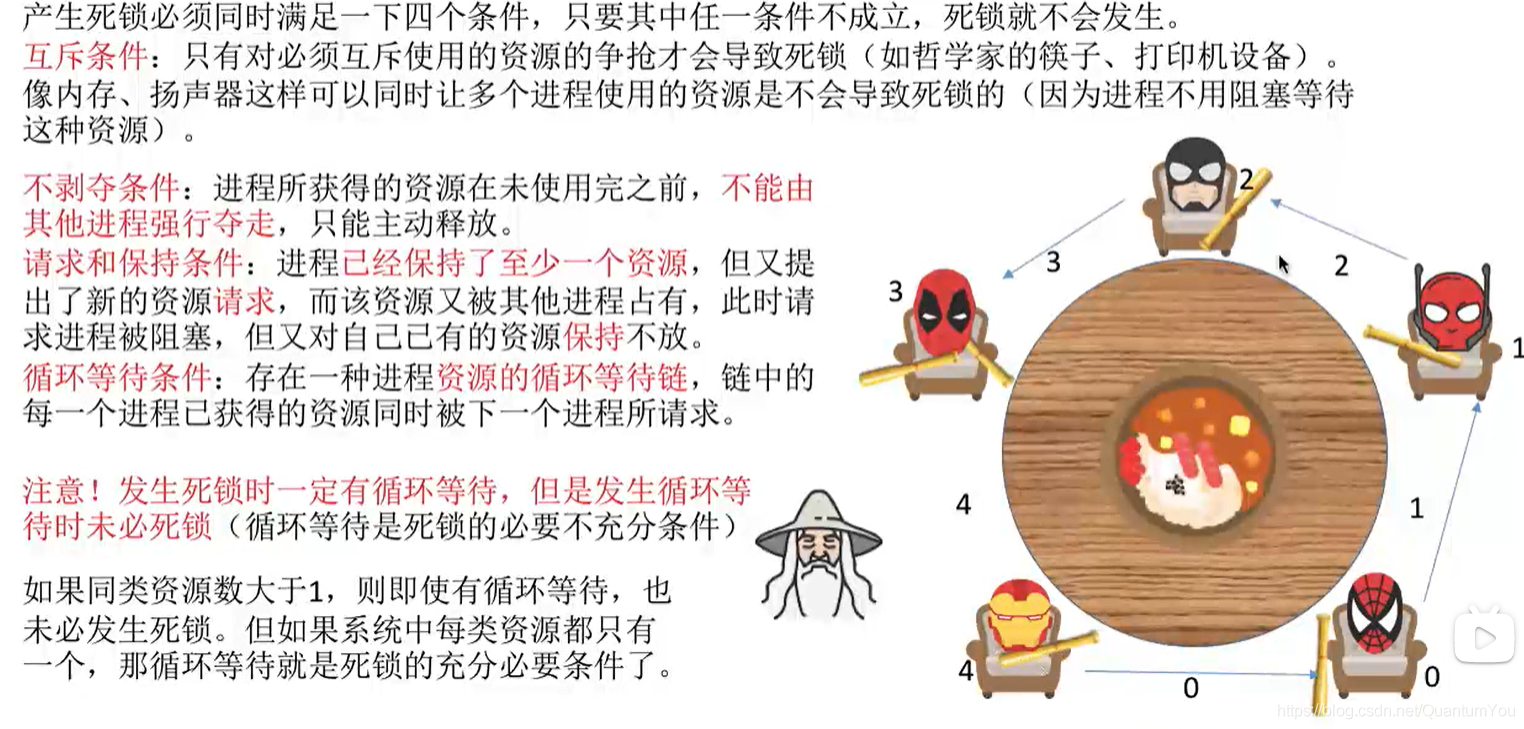
* 在并发环境下，各进程因竞争资源而造成的一种互相等待对方手里的资源，导致各进程都阻塞，都无法向前推进的现象，就是“死锁”。发生死锁后若无外力干涉，这些进程都将无法向前推进
* **死锁**：各进程互相等待对方手里的资源，导致各进程都阻塞，无法向前推进的现象。
* **饥饿**：由于长期得不到想要的资源，某进程无法向前推进的现象。比如：在短进程优先（SPF）算法中，若有源源不断的短进程到来，则长进程将一直得不到处理机，从而发生长进程“饥饿”.
* **死循环**：某进程执行过程中一直跳不出某个循环的现象。有时是因为程序逻辑bug导致的，有时是程序员故意设计的。

## 死锁、饥饿、死循环区别

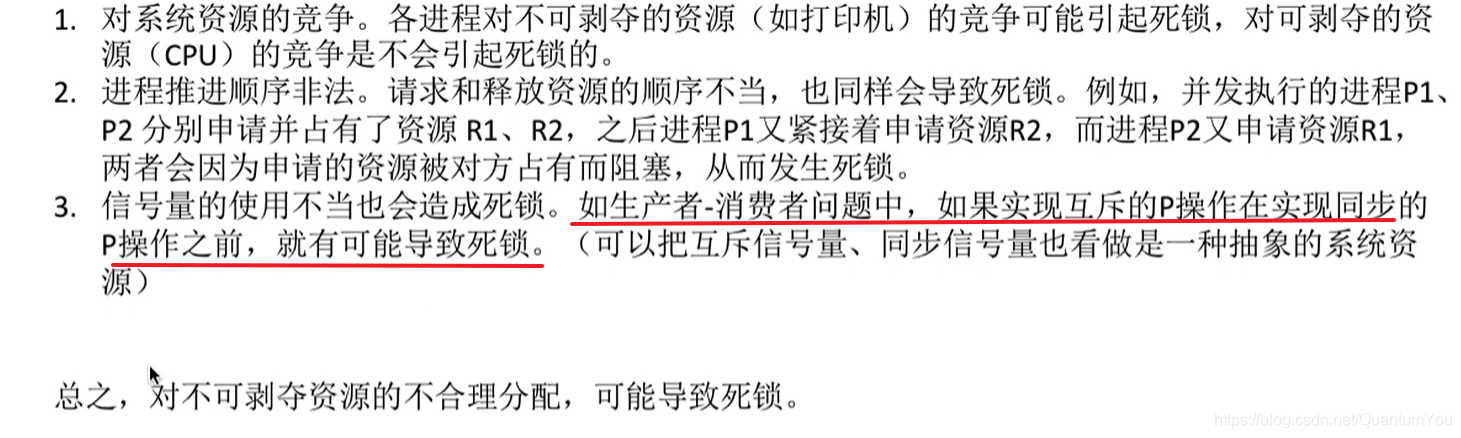


## 死锁产生的必要条件

* 互斥条件
* 不剥夺条件
* 请求和保持条件
* 循环等待条件



## 什么时候发生死锁



## 死锁的处理策略

* 1、预防弧锁。破坏死锁产生的四个必要条件中的一个或几个。
* 2、避免死锁。用某种方法防止系统进入不安全状态，从而避免死锁（银行家算法）
* 3、死锁的检测和解除。允许死锁的发生，不过操作系统会负责检测岀死锁的发生，然后采取某种措施解除死锁。

## 小结

