signal语句只可能作为一个管程过程的最后一条语句。我们将采纳Brinch Hansen 的建议，因为它在概念上更简单，并且更容易实现。如果在一个条件变量上有若干进程正在等待，则在对该条件变量执行signal操作后，系统调度程序只能在其中选择一个使其恢复运行。

顺便提一下，还有一个Hoare和Brinch Hansen都没有提及的第三种方法，该方法让发信号者继续运行，并且只有在发信号者退出管程之后，才允许等待的进程开始运行。

条件变量不是计数器，条件变量也不能像信号量那样积累信号以便以后使用。所以，如果向一个条件变量发送信号，但是在该条件变量上并没有等待进程，则该信号会永远丢失。换句话说，wait操作必须在signal之前。这条规则使得实现简单了许多。实际上这不是一个问题，因为在需要时，用变量很容易跟踪每个进程的状态。一个原本要执行signal的进程，只要检查这些变量便可以知道该操作是否有必要。

在图2-34中给出了用类Pascal语言，通过管程实现的生产者-消费者问题的解法框架。使用类Pascal语言的优点在于清晰、简单，并且严格符合Hoare/Brinch Hansen模型。

读者可能会觉得wait和signal操作看起来像前面提到的sleep和wakeup，而我们已经看到后者存在严重的竞争条件。是的，它们确实很像，但是有个很关键的区别：sleep和wakeup之所以失败是因为当一个进程想睡眠时另一个进程试图去唤醒它。使用管程则不会发生这种情况。对管程过程的自动互斥保证了这一点：如果管程过程中的生产者发现缓冲区满，它将能够完成wait操作而不用担心调度程序可能会在wait完成之前切换到消费者。甚至，在wait执行完成而且把生产者标志为不可运行之前，根本不会允许消费者进入管程。

尽管类Pascal是一种想象的语言，但还是有一些真正的编程语言支持管程，不过它们不一定是Hoare和Brinch Hansen所设计的模型。其中一种语言是Java。Java是一种面向对象的语言，它支持用户级线程，还允许将方法（过程）划分为类。只要将关键词synchronized加入到方法声明中，Java 保证一旦某个线程执行该方法，就不允许其他线程执行该对象中的任何synchronized方法。

使用Java管程解决生产者-消费者问题的解法如图2-35所示。该解法中有4个类。外部类（outer class）ProducerConsumer创建并启动两个线程，p和c。第二个类和第三个类producer和consumer分别包含生产者和消费者的代码。最后，类our_monitor是管程，它有两个同步线程，用于在共享缓冲区中插入和取出数据项。与前面的例子不同，我们在这里给出了insert和remove的全部代码。

在前面所有的例子中，生产者和消费者线程在功能上与它们的等同部分是相同的。生产者有一个无限循环，该无限循环产生数据并将数据放入公共缓冲区中；消费者也有一个等价的无限循环，该无限循环从公共缓冲区取出数据并完成一些有趣的工作。

该程序中比较意思的部分是类our_monitor，它包含缓冲区、管理变量以及两个同步方法。当生产者在insert内活动时，它确信消费者不能在remove中活动，从而保证更新变量和缓冲区的安全，且不用担心竞争条件。变量count记录在缓冲区中数据项的数量。它的取值可以取从0到$N-1$之间任何值。变量lo是缓冲区槽的序号，指出将要取出的下一个数据项。类似地，hi是缓冲区中下一个将要放入的数据项序号。允许$lo = hi$，其含义是在缓冲区中有0个或N个数据项。count的值说明了究竟是哪一种情形。

Java中的同步方法与其他经典管程有本质差别：Java没有内嵌的条件变量。反之，Java提供了两个过程wait和notify，分别与sleep和wakeup等价，不过，当它们在同步方法中使用时，它们不受竞争条件约束。理论上，方法wait可以被中断，它本身就是与中断有关的代码。Java需要显式表示异常处理。在本文的要求中，只要认为go_to_sleep就是去睡眠即可。

通过临界区互斥的自动化，管程比信号量更容易保证并行编程的正确性。但管程也有缺点。我们之所以使用类Pascal和Java，而不像在本书中其他例子那样使用C语言，并不是没有原因的。正如我们前面提到过的，管程是一个编程语言概念，编译器必须要识别管程并用某种方式对其互斥做出安排。C、Pascal以及多数其他语言都没有管程，所以指望这些编译器遵守互斥规则是不合理的。实际中，如何能让编译器知道哪些过程属于管程，哪些不属于管程呢？

在上述语言中同样也没有信号量，但增加信号量是很容易的：读者需要做的就是向库里加入两段短小的汇编程序代码，以执行up和down系统调用。编译器甚至用不着知道它们的存在。当然，操作系统必须知道信号量的存在，或至少有一个基于信号量的操作系统，读者仍旧可以使用C或C++（甚至是汇编语言，如果读者乐意的话）来编写用户程序，但是如果使用管程，读者就需要一种带有管程的语言。