2.3.8 消息传递

上面提到的其他的方法就是消息传递（message passing）。这种进程间通信的方法使用两条原语send和receive，它们像信号量而不像管程，是系统调用而不是语言成分。因此，可以很容易地将它们加入到库例程中去。例如：

```
send(destination, &message);
```

和

```
receive(source, &message);
```

前一个调用向一个给定的目标发送一条消息，后一个调用从一个给定的源（或者是任意源，如果接收者不介意的话）接收一条消息。如果没有消息可用，则接收者可能被阻塞，直到一条消息到达，或者，带着一个错误码立即返回。

1. 消息传递系统的设计要点

消息传递系统面临着许多信号量和管程所未涉及的问题和设计难点，特别是位于网络中不同机器上的通信进程的情况。例如，消息有可能被网络丢失。为了防止消息丢失，发送方和接收方可以达成如下一致：一旦接收到信息，接收方马上回送一条特殊的确认（acknowledgement）消息。如果发送方在一段时间间隔内未收到确认，则重发消息。

现在考虑消息本身被正确接收，而返回给发送者的确认信息丢失的情况。发送者将重发信息，这样接收者将接收到两次相同的消息。对于接收者来说，如何区分新的消息和一条重发的老消息是非常重要的。通常采用在每条原始消息中嵌入一个连续的序号来解决此问题。如果接收者收到一条消息，它具有与前面某一条消息一样的序号，就知道这条消息是重复的，可以忽略。不可靠消息传递中的成功通信问题是计算机网络的主要研究内容。更多的信息可以参考相关文献（Tanenbaum，1996）。

消息系统还需要解决进程命名的问题，在send和receive调用中所指定的进程必须是没有二义性的。身份认证（authentication）也是一个问题。比如，客户机怎么知道它是在与一个真正的文件服务器通信，而不是与一个冒充者通信？

对于发送者和接收者在同一台机器上的情况，也存在若干设计问题。其中一个设计问题就是性能问题。将消息从一个进程复制到另一个进程通常比信号量操作和进入管程要慢。为了使消息传递变得高效，人们已经做了许多工作。例如，Cheriton（1984）建议限制信息的大小，使其能装入机器的寄存器中，然后便可以使用寄存器进行消息传递。

2. 用消息传递解决生产者-消费者问题

现在我们来考察如何用消息传递而不是共享内存来解决生产者-消费者问题。在图2-36中，我们给出了一种解法。假设所有的消息都有同样的大小，并且在尚未接收到发出的消息时，由操作系统自动进行缓冲。在该解决方案中共使用N条消息，这就类似于一块共享内存缓冲区中的N个槽。消费者首先将N条空消息发送给生产者。当生产者向消费者传递一个数据项时，它取走一条空消息并送回一条填充了内容的消息。通过这种方式，系统中总的消息数保持不变，所以消息都可以存放在事先确定数量的内存中。

如果生产者的速度比消费者快，则所有的消息最终都将被填满，等待消费者，生产者将被阻塞，等待返回一条空消息。如果消费者速度快，则情况正好相反：所有的消息均为空，等待生产者来填充它们，消费者被阻塞，以等待一条填充过的消息。

消息传递方式可以有许多变体。我们首先介绍如何对消息进行编址。一种方法是为每个进程分配一个惟一的地址，让消息按进程的地址编址。另一种方法是引入一种新的数据结构，称作信箱（mailbox）。信箱是一个用来对一定数量的消息进行缓冲的地方，信箱中消息数量的设置方法也有多种，典型的方法是在信箱创建时确定消息的数量。当使用信箱时，在send和receive调用中的地址参数就是信箱的地址，而不是进程的地址。当一个进程试图向一个满的信箱发消息时，它将被挂起，直到信箱内有消息被取走，从而为新消息腾出空间。

对于生产者-消费者问题，生产者和消费者均应创建足够容纳N条消息的信箱。生产者向消费者信箱发送包含实际数据的消息，消费者则向生产者信箱发送空的消息。当使用信箱时，缓冲机制的作用是很清楚的：目标信箱容纳那些已被发送但尚未被目标进程接收的消息。