和移出操作是原子性的，所以，如果两个进程同时执行in操作，只有其中一个会成功，除非存在两个或更多的匹配元组。在元组空间中甚至可以有同一个元组的多个副本存在。

in采用的匹配算法是很直接的。in原语的域，称为模板（template），（在概念上）它与元组空间中的每个元组的同一个域相比较，如果下面的三个条件都符合，那么产生出一个匹配：

1) 模板和元组有相同数量的域。

2) 对应域的类型一样。

3) 模板中的每个常数或者变量均与该元组域相匹配。

形式参数，由问号标识后面跟随一个变量名或类型所给定，并不参与匹配（除了类型检查例外），尽管在成功匹配之后，那些含有一个变量名称的形式参数会被赋值。

如果没有匹配的元组存在，调用进程便被挂起，直到另一个进程插入了所需要的元组为止，此时该调用进程自动复活并获得新的元组。进程阻塞和自动解除阻塞意味着，如果一个进程与输出一个元组有关而另一个进程与输入一个元组有关，那么谁在先是无关紧要的。惟一的差别是，如果in在out之前被调用了，那么会有少许的延时存在，直到得到元组为止。

在某个进程需要一个不存在的元组时，阻塞该进程的方式可以有许多用途。例如，该方式可以用于信号量的实现。为了要建立信号量S或在信号量S上执行一个up操作，进程可以执行如下操作

```
out("semaphore S");
```

要执行一个down操作，可以进行

```
in("semaphore S");
```

在元组空间中（"semaphore S"）元组的数量决定了信号量S的状态。如果信号量不存在，任何要获得信号量的企图都会被阻塞，直到某些其他的进程提供一个为止。

除了out和in操作，Linda还提供了原语read，它和in是一样的，不过它不把元组移出元组空间。还有一个原语eval，它的作用是同时对元组的参数进行计算，计算后的元组会被放进元组空间中去。可以利用这个机制完成一个任意的运算。以上内容说明了怎样在Linda中创建并行的进程。

2. 发布/订阅（Pubilsh/Subscribe）

由于受到Linda的启发，出现了基于协作的模型的一个例子，称作pubilsh/subscribe（Oki等人，1993）。它由大量通过广播网网络互联的进程组成。每个进程可以是一个信息生产者、信息消费者或两者都是。

当一个信息生产者有了一条新的信息（例如，一个新的股票价格）后，它就把该信息作为一个元组在网络上广播。这种行为称为发布（publishing）。在每个元组中有一个分层的主题行，其中有多个用圆点（英文句号）分隔的域。对特定信息感兴趣的进程可以订阅（subscribe）特定的专题，这包括在主题行中使用通配符。在同一台机器上，只要通知一个元组守护进程就可以完成订阅工作，该守护进程监测已出版的元组并查找所需要的专题。

发布/订阅的实现过程如图8-41所示。当一个进程需要发布一个元组时，它在本地局域网上广播。在每台机器上的元组守护进程则把所有的已广播的元组复制进入其RAM。然后检查主题行看看哪些进程对它感兴趣，并给每个感兴趣的进程发送一个该元组的副本。元组也可以在广域网上或Internet上进行广播，这种做法可以通过将每个局域网中的一台机器变作信息路由器，用来收集所有已发布的元组，然后转送到其他的局域网上再次广播的方法来实现。这种转送方法也可以进行得更为聪明，即只把元组转送给至少有一个需要该元组的订阅者的远程局域网。不过要做到这一点，需要使用信息路由器交换有关订阅者的信息。

这里可以实现各种语义，包括可靠发送以及保证发送，即使出现崩溃也没有关系。在后一种情形下，有必要存储原有的元组供以后需要时使用。一种存储的方法是将一个数据库系统和该系统挂钩，并让该数据库订阅所有的元组。这可以通过把数据库封装在一个适配器中实现，从而允许一个已有的数据库以发布/订阅模型工作。当元组们经过时，适配器就一一抓取它们并把它们放进数据库中。

发布/订阅模型完全把生产者和消费者分隔开来，如同在Linda中一样。但是，有的时候还是有必要知道，另外还有谁对某种信息感兴趣。这种信息可以用如下的方法来收集：发布一个元组，它只询问："谁对信息x有兴趣？"。以元组形式的响应会是："我对x有兴趣。"