第6章 基于锁的并发数据结构设计

本章主要内容

- 并发数据结构设计的意义
- 指导如何设计
- 实现为并发设计的数据结构

在上一章中，我们对底层原子操作和内存模型有了详尽的了解。在本章中，我们将先将底层的东西放在一边(将会在第7章再次提及)，来对数据结构做一些讨论。

数据结构的选择，对于程序来说，是其解决方案的重要组成部分，当然，并行程序也不例外。如果一种数据结构可以被多个线程所访问，其要不就是绝对不变的(其值不会发生变化，并且不需同步)，要不程序就要对数据结构进行正确的设计，以确保其能在多线程环境下能够(正确的)同步。一种选择是使用独立的互斥量，其可以锁住需要保护的数据(这种方法已经在第3和第4章中提到)，另一种选择是设计一种能够并发访问的数据结构。

在设计并发数据结构时，你可以使用基本多线程应用中的构建块(之前章节中有提及)，比如，互斥量和条件变量。当然，你也已经在之前的章节的例子中看到，怎样联合不同的构建块，对数据结构进行写入，并且保证这些构建块都是在并发环境下是线程安全的。

在本章，我们将了解一些并发数据结构设计的基本准则。然后，我们将再次重温锁和条件变量的基本构建块。最后，会去了解更为复杂的数据结构。在第7章，我们将了解，如何正确的“返璞归真”，并使用第5章提到的原子操作，去构建无锁的数据结构。

好吧！多说无益，让我们来看一下并发数据结构的设计，都需要些*什么*。