测试您对练习3.20的解决方案的多线程程序。您将创建多个线程（例如100个），每个线程将请求一个pid，休眠一段随机时间，然后释放该pid。（睡眠一段随机的时间近似于典型的pid用法，在该用法中，将pid分配给新进程，该进程执行，然后终止，然后在进程的终止中释放该pid。）在UNIX和Linux系统上，睡眠是通过sleep（）函数完成，该函数传递了一个表示睡眠秒数的整数值。该问题将在第7章中进行修改。

4.29第3章中的练习3.25涉及使用Java线程API设计回显服务器。该服务器是单线程的，这意味着在当前客户端退出之前，该服务器无法响应并发的回显客户端。修改练习3.25的解决方案，使回显服务器在单独的请求中为每个客户端提供服务。

程式设计专案

项目1 — Sudoku解决方案验证器

数独难题使用9×9网格，其中每个列和行以及9个3×3子网格中的每个必须包含所有数字1…9。
4.26 给出了一个有效的数独难题的例子。该项目包括
设计一个多线程应用程序，该应用程序确定Sudoku拼图的解决方案是否有效。

有多种不同的方法对该应用程序进行多线程处理。一种建议的策略是创建检查以下条件的线程：

- 检查每列是否包含数字1到9的线程
- 检查每行是否包含数字1到9的线程

6	2	4	5	3	9	1	8	7
5	1	9	7	2	8	6	3	4
8	3	7	6	1	4	2	9	5
1	4	3	8	6	5	7	2	9
9	5	8	2	4	7	3	6	1
7	6	2	3	9	1	4	5	8
3	7	1	9	5	6	8	4	2
4	9	6	1	8	2	5	7	3
2	8	5	4	7	3	9	1	6

图4.26 9×9数独难题的解决方案。

- 九个线程来检查3×3子网格中的每一个是否包含数字1到9

这将导致总共十一个独立的线程用于验证Sudoku拼图。但是，欢迎您为该项目创建更多线程。例如，您可以创建九个单独的线程并让每个线程检查一列，而不是创建一个检查所有九列的线程。

I. 将参数传递给每个线程

父线程将创建工作线程，并向每个工作线程传递必须在Sudoku网格中检查的位置。此步骤将需要将几个参数传递给每个线程。最简单的方法是使用结构创建数据结构。例如，传递必须在其中开始验证线程的行和列的结构，如下所示：

```
/ *用于将数据传递到线程的结构* /
typedef结构
{
 int行;int
  列;
}参数;
```

Pthread和Windows程序都将使用类似于以下所示的策略创建工作线程：

```
参数* data = (parameters *) malloc (sizeof
 (parameters) ) ;数据->行= 1;
data->column = 1;
/ *现在创建将数据作为参数传递给它的线程* /
```

数据指针将传递给pthread create ()（Pthreads）函数或CreateThread ()（Windows）函数，后者再将其作为参数传递给作为单独线程运行的函数。

II. 返回结果到父线程

每个工作线程都被分配了确定数独难题特定区域有效性的任务。工人执行此检查后，必须将其结果传递回父母。处理此问题的一种好方法是创建一个整数值数组，该数组对于每个线程都是可见的。该数组中的i⊟索引对应于i⊟worker线程。如果工人将其对应的值设置为1，则表示其数独难题的区域有效。值为0表示相反。当所有辅助线程都完成后，父线程将检查结果数组中的每个条目，以确定Sudoku拼图是否有效。

项目2 —多线程排序应用程序

编写一个多线程排序程序，该程序的工作方式如下：整数列表分为两个大小相等的较小列表。两个单独的线程（我们

术语排序线程）使用您选择的排序算法对每个子列表进行排序。然后，两个子列表由第三个线程合并-合并线程
—将两个子列表合并为一个排序列表。

因为全局数据在所有线程之间共享，所以设置数据的最简单方法可能是创建全局数组。每个排序线程将在该数组的一半上工作。还将建立与未排序整数数组大小相同的第二个全局数组。然后，合并线程将两个子列表合并到此第二个数组中。在图形上，该程序的结构如图4.27所示。

该编程项目将需要将参数传递给每个排序线程。特别是，有必要标识每个线程将从其开始排序的起始索引。有关将参数传递给线程的详细信息，请参阅项目1中的说明。

一旦所有排序线程都退出，父线程将输出排序后的数组。

项目3 -叉联接分类应用程序

使用Java的fork-join并行性API实现前面的项目（多线程排序应用程序）。该项目将以两种不同的版本进行开发。每个版本将实现不同的分而治之排序算法：

1. 快速排序
2. 合并排序

Quicksort实施将使用Quicksort算法将要排序的元素列表基于以下内容分为左半部分和右半部分：

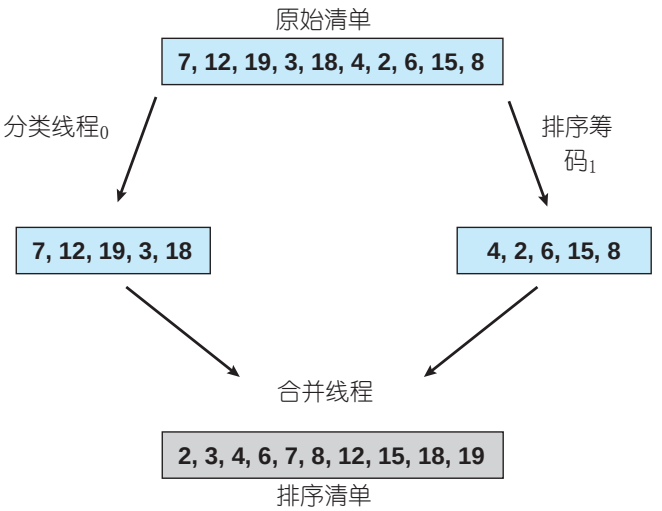

图4.27多线程排序。

枢轴值的位置。Mergesort算法会将列表分为两个大小相等的两半。对于Quicksort和Mergesort算法，当要排序的列表都在某个阈值之内时（例如，列表的大小为100或更小），请直接应用简单的算法，例如Selection或Insertion排序。大多数数据结构文本描述了这两种众所周知的分而治之排序算法。

4.5.2.1节中显示的SumTask类扩展了RecursiveTask，它是一个具有结果的ForkJoinTask。由于此分配将涉及对传递给任务的数组进行排序，但不返回任何值，因此您将创建一个扩展RecursiveAction的类，该类是不带结果的ForkJoinTask（请参见图4.19）。

传递给每种排序算法的对象是实现所需的Java的Comparable接口，这需要在每种排序算法的类定义中得到反映。本文的源代码下载包括Java代码，这些Java代码为开始该项目提供了基础。