实验 2：多线程编程

截止日期

请参考实验室作业要求。

目标

本项目的目标是：（1）很好地理解多线程；（2）练习创建线程并协调线程的运行。

第一部分 - 数独解法验证器

数独谜题使用一个 9×9 的网格，其中每一列、每一行以及九个 3×3 的子网格中的每一个都必须包含数字 1 到 9 。下图展示了一个有效的数独谜题的示例。本项目包括设计一个多线程应用程序，用于确定数独谜题的解决方案是否有效。

对于这个应用程序的多线程处理，有几种不同的方式。一种建议的策略是创建线程来检查以下标准：

一个线程用于检查每一列是否包含数字 1 到 9 。
一个线程用于检查每一行是否包含数字 1 到 9 。
☒ 检查九条线索，以确定每个 3×3 的子网格都包含数字 1 至 9 。

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

这将导致验证一个数独谜题的总共有 11 个独立线程。不过，欢迎您为这个项目创建更多的线程。例如，与其创建一个线程来检查所有九列，您可以创建九个独立的线程，让每个线程检查一列。

向每个线程传递参数

父线程会创建工作线程，并向每个工作线程传递其在数独网格中必须检查的位置。此步骤需要向每个线程传递几个参数。最简单的方法是使用结构体创建一个数据结构。例如，传递线程必须开始验证的行和列的结构体如下所示：

```
/* structure for passing data to threads */
typedef struct
{
int row;
int column;
} parameters;
```

Pthreads 和 Windows 程序都会使用类似于如下所示的策略来创建工作线程：

```
parameters *data = (parameters *) malloc(sizeof(parameters));
data->row = 1;
data->column = 1;
/* Now create the thread passing it data as a parameter */
```

The data pointer will be passed to either the `pthread_create()` (Pthreads) function or

CreateThread（）（Windows）函数，它又会将其作为参数传递给要作为单独线程运行的那个函数。

将结果返回给父线程

每个工作线程都被分配了确定数独谜题特定区域有效性的任务。一旦一个工作线程完成了这个检查，它必须将结果传递给父线程。处理这个问题的一个好方法是创建一个对每个线程都可见的整数数组。这个数组的第i个索引对应于第i个工作线程。如果一个工作线程将其对应的值设置为1，表示其数独谜题的区域是有效的。值为0则表示无效。当所有工作线程完成后，父线程会检查结果数组中的每个条目，以确定数独谜题是否有效。

第二部分 - 多线程排序应用程序

编写一个多线程排序程序，其工作方式如下：将一个整数列表分成两个大小相等的子列表。两个独立的线程（我们称之为排序线程）分别对每个子列表使用您选择的排序算法进行排序。然后，第三个线程——合并线程——将这两个子列表合并为一个已排序的列表。

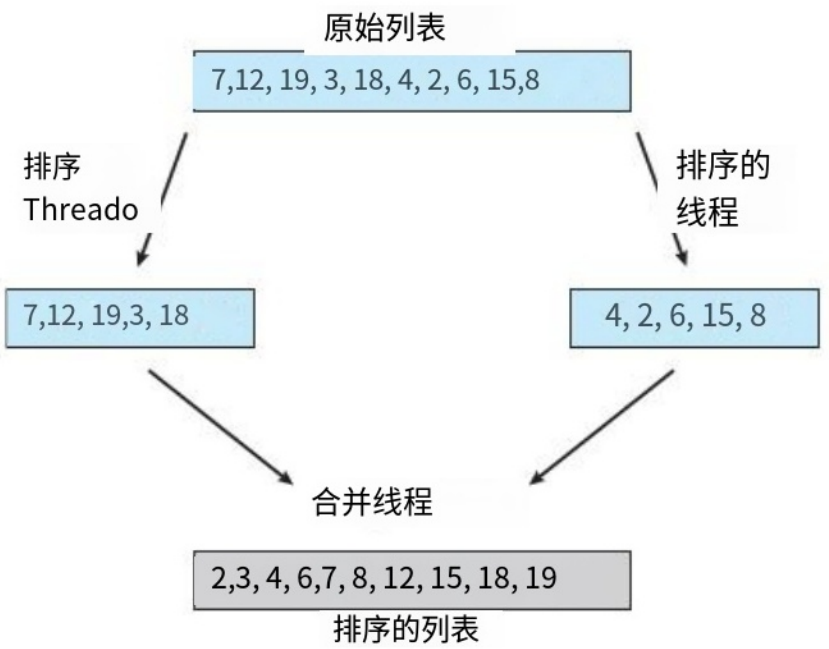

图 2. 多线程排序

由于全局数据在所有线程之间共享，也许设置数据的最简单方法是在创建全局数组。每个排序线程将处理这个数组的一半。还将创建一个与未排序整数数组大小相同的第二个全局数组。

然后，合并线程会将这两个子列表合并到这个第二个数组中。从图形上看，这个程序的结构如图2所示。这个编程项目需要向每个排序线程传递参数。特别是，需要确定每个线程开始排序的起始索引。关于向线程传递参数的详细信息，请参考项目1中的说明。
一旦所有排序线程都退出，父线程将输出已排序的数组。

提交

您的提交内容应包括代码、一份简要描述您的设计的自述文件、如何编译/使用您的代码，以及一份报告，报告应包括以下部分（但不限于）：

总结 Linux（或 Windows，如适用）提供的线程控制方法，并描述其使用方法。

☒ 您对该程序的设计
实验结果（统计数据）的截图及分析
☒ 遇到的问题及您的解决方案
☒ 参考资料
您的建议和意见

环境

Linux（推荐使用，2.6 之后的任何内核版本均可）以及 C/C++ 。

参考文献

N/A