Project 3: Multithreaded Sorting Application

& Fork-Join Sorting Application

Project 3-1: Multithreaded Sorting Application

实现细节：

一、准备

1、全局变量

两个全局变量，一个存原数组，一个存排序后的数组

2、index 类型

声明将排序的开始位置和结束位置传递给线程而生命的 index 类型

3、quicksort()

依照快速排序算法写函数 quicksort()

4、sort()

写快速排序函数的包裹函数 sort()，作为快速排序线程的“run”，即写成一个函数指针

5、merge()

依照归并排序算法写函数 merge()，直接作为归并线程的“run”,也是一个函数指针

二、main()

1、输入数组大小

2、希望数组大小至少为 2，否则提示非法并退出

3、循环输入数组

4、实例化 3 个 index 对象，存在一个 index 类型数组中，为三个排序线程提供排序的开始位置和结束位置。

5、创建与初始化线程

6、依次运行线程

7、输出排序结果

8、释放空间

9、结束返回

运行结果：

```
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch4$ gcc -o MSApp
 MSApp.c -lpthread
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch4$ ./MSApp
What is the size of your array? Please input(must bigger than 1):
10
Please print your array:
7 12 19 3 18 4 2 6 15 8
The sorted array is:
2 3 4 6 7 8 12 15 18 19
```

Project 3-2: Fork-Join Sorting Application

（一）QuickSort 实现细节：

一、配制环境

按照网上找到的资料配制 java 环境：https://www.linuxidc.com/Linux/2018-10/155020.htm

二、类与成员函数

1、代码框架

依照课本 Fork Join in Java 内容搭建代码框架 ForkJoinTask_Quick 类，是 RecursiveAction 的派生类。

2、设置“small”的标准 THRESHOLD。

3、全局变量

设置全局变量：数组大小、数组、起始位置、终止位置。

4、类的构造函数

类的构造函数用于初始化参数。

5、compute()

重载的 compute()函数，相当于线程的 run 函数。当问题规模小于 THRESHOLD 时，直接调用选择排序算法；当问题规模大于 THRESHOLD 时，使用快排算法算出 pivot 的位置，利用 pivot 将问题分为 leftTask 与 rightTask，将其实例化为新的 ForkJoinTask_Quick 类对象，为其创建新线程，并执行。最后将左任务与右任务 merge 起来，返回。

6、quicksort()

依照快速排序算法写 quicksort()函数，但是不需要在最后递归，而是返回 pivot 的值。

7、selectionsort()

依照选择排序算法写 selectionsort()函数，用于问题规模划分至小于 THRESHOLD 的时候。

8、merge()

依照归并排序算法写函数 merge()，用于归并已完成的左任务和右任务。

三、主函数

1、实例化 ForkJoinPool 类线程池对象。

2、利用随机数生成初始数组。

3、实例化 ForkJoinTask_Quick 对象 task，是用于排序的任务。

4、将其加入线程池开始运行。

5、最后将排好序的数组输出。

（二）MergeSort 实现细节：

基本与 QuickSort 相同，只有划分左任务和右任务时，划分的标准为 mergesort()函数返回，实际上就是对半划分。

（三）运行结果：

```
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch4$ java FJSApp-
Quick.java
The original array is:
6 74 65 84 7 64 82 47 95 31 55 73 18 29 59 28 14 56 87 27 19 46 17 68 69 27 84 4
7 5 90 39 40 69 28 48 0 89 46 59 29 34 17 75 83 27 87 13 86 75 46 1 57 80 10 31
86 97 82 99 64 42 75 88 22 65 51 46 10 0 77 94 34 48 26 13 60 45 88 2 97 32 72 2
0 24 2 2 60 20 74 85 28 87 22 77 18 43 26 84 39 50
The sorted array is:
0 0 1 2 2 2 5 6 7 10 10 13 13 14 17 17 18 18 19 20 20 22 22 24 26 26 27 27 27 28
 28 28 29 29 31 31 32 34 34 39 39 40 42 43 45 46 46 46 46 47 47 48 48 50 51 55 5
6 57 59 59 60 60 64 64 65 65 68 69 69 72 73 74 74 75 75 75 77 77 80 82 82 83 84
84 84 85 86 86 87 87 87 88 88 89 90 94 95 97 97 99
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch4$
```

```
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch4$ java FJSApp-
Merge.java
The original array is:
31 57 21 28 8 28 73 6 81 82 7 54 40 85 96 61 21 32 22 33 52 45 72 80 67 29 10 72
 78 32 92 36 60 23 21 97 59 77 86 36 71 13 67 15 20 94 62 83 23 85 22 92 8 99 39
 49 57 79 37 92 61 35 7 90 36 88 47 4 92 53 28 89 83 59 11 64 48 59 74 94 42 93
42 97 16 6 61 76 67 50 6 10 47 67 45 15 44 42 43 90
The sorted array is:
4 6 6 6 7 7 8 8 10 10 11 13 15 15 16 20 21 21 21 22 22 23 23 28 28 28 29 31 32 3
2 33 35 36 36 36 37 39 40 42 42 42 43 44 45 45 47 47 48 49 50 52 53 54 57 57 59
59 59 60 61 61 61 62 64 67 67 67 67 71 72 72 73 74 76 77 78 79 80 81 82 83 83 85
 85 86 88 89 90 90 92 92 92 92 93 94 94 96 97 97 99
thousanrance@thousanrance-VirtualBox:~/Desktop/Code/OS/project/ch4$
```