Programming Assignment two

COMP3033 OS 1001-1003 2st semester 2021-2022

This assignment will involve writing a **multithreaded sorting** program. Once you are finished, write a report, copy the code of your program to the end of the report, Put your name and student ID number on your report. Also, add some pictures of your program running in the report. Write a paragraph to describe how you solve this problem. Did you encounter any problems? how do you fix it? In addition, upload the C file on iSpace.

Late homework assignments will not be accepted unless you have a valid written excuse (medical, etc.). You must do this assignment alone. No teamwork or "talking with your friends" will be accepted. No copying from the Internet. Cheating means zero.

You can do this assignment on a computer running on a Linux system.

Instruction:

The sorting program works as follows: A list of integers is divided into two smaller lists of equal size. Two separate threads (which we will term ***sorting threads***) sort each sublist using a sorting algorithm of your choice. The two sublists are then merged by a third thread—a ***merging thread*** — which merges the two sublists into a single sorted list.

Because global data are shared across all threads, perhaps the easiest way to set up the data is to create a global array. We ask you randomly to create 10000 positive integers (value <10000) in this array for sorting. Here is the sample code segment:

```
/* create 10000 random integers (value <10000) saved in a list */
    SIZE = 10000;
    for(i = 0; i < SIZE; i++)
        list[i] = rand() % 10000;
```

Each sorting thread will work on one half of this array. A second global array of the same size as the unsorted integer array will also be established. The merging thread will then merge the two sublists into this second array. Graphically, this program is structured as in Figure 1.

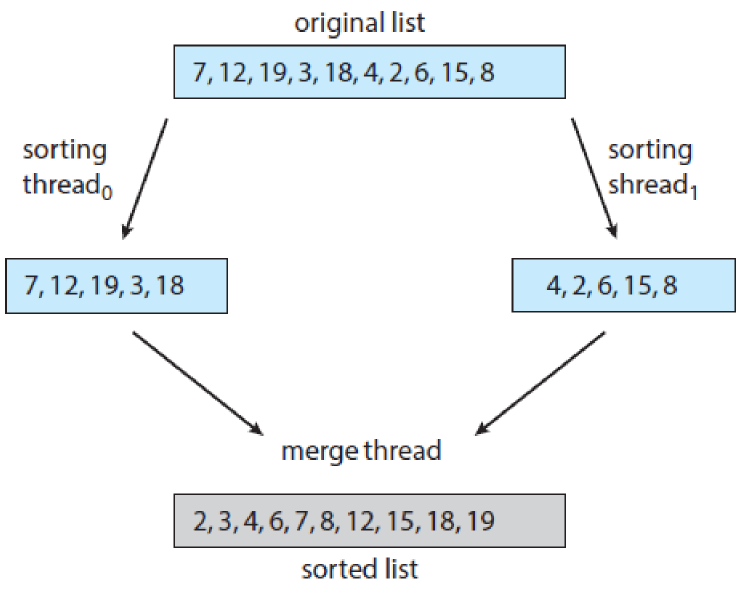

Figure 1 Multithreaded sorting.

This program will require passing parameters to each of the sorting threads. In particular, it will be necessary to identify the starting index from which each sorting thread is to begin sorting.

I. Passing Parameters to Each Sorting Thread

The parent thread will create two worker sorting threads, passing each worker thread the start and end index/location that it uses for sorting. This step will require passing several parameters to each sorting thread. The easiest approach is to create a data structure using a struct. For example, a structure to pass the start and end index where a thread must begin sorting would appear as follows:

```
/* structure for passing two parameters to sorting threads */
typedef struct
{
    int startIndex;
    int stopIndex;
} parameters;
```

Pthreads program will create worker threads using a strategy similar to that shown below:

```
// for sorting thread working on the first half data
parameters *data1 = (parameters *) malloc(sizeof(parameters));
data1->startIndex = 0;
data1->stopIndex = N/2-1;
```

```
    // for sorting thread working on the second half data
    parameters *data2 = (parameters *) malloc(sizeof(parameters));
    data2->startIndex = N/2;
    data2->stopIndex = N-1;

    /* Now create the thread passing it data as a parameter */
```

The data pointer will be passed to the pthread_create()function, which in turn will pass it as a parameter to the function that is to run as a separate thread.

Similarly, you can pass parameters to merge thread using the same struct above.

```
    parameters *data3 = (parameters *) malloc(sizeof(parameters));
    data3->startIndex = 0;
    data3->stopIndex = N/2;
```

The parent thread will output the sorted array to an output file on your disk once two sorting threads and merge thread have exited.

Command to compile your code:
Suppose your code file name is sort.c, use the following command to build an executable file sort:

```
>gcc -o sort sort.c  - lpthread
```

Run it:

```
>./sort
```

Good Luck!