

Concurrent Programming

Lab 1 Report

Author: Sarayu Managoli

Program Overview:

The program is intended to implement fork/join method of merge sort or quick sort and locked bucket sort. The execution of these sorting algorithms is dependent upon the inputs provided by the user. This program is expected to function like "sort -n" Linux command and output the sorted values onto the required file.

Brief Introduction to Bucket Sort:

Bucket Sort functions by distributing the elements of an array into various number of buckets. Each of the buckets is then sorted individually, either by using an inbuilt function, STL Map, different sorting algorithm, or by recursively applying the bucket sorting algorithm. Bucket sort can be exceptionally fast because of the way elements are assigned to buckets, typically using an array where the index is the value. The average time complexity for Bucket Sort is $O(n + k)$. The worst time complexity is $O(n^2)$. The space complexity for Bucket Sort is $O(n+k)$. Bucket sort is mainly useful when input is uniformly distributed over a range. For this Lab, I have used maps to sort the bucket

Brief Introduction of Merge Sort

It is said to be one of the most efficient sorting algorithms. Similar to quick sort, this too works on the principle of divide-and-conquer. It breaks down an array into smaller arrays until each of the arrays consists of only one element. Until the arrays are down to one element, merge sort is called. These single elements are later merged into forming a single sorted array. Here, the array is divided exactly from the middle element. Here additional storage is used as compared to quick sort. Merge sort does not work on the right array until the left array is sorted.

Brief Introduction to Pthreads

Pthreads are defined as a set of C language programming types and procedure calls, implemented with a pthread.h header/include file and a thread library - though this library may be part of another library, such as libc, in some implementations. A single process can contain multiple threads, all of which are executing the same program. These threads share the same global memory (data and heap segments), but each thread has its own stack (automatic variables). Most pthreads functions return 0 on success, and an error number of failure.

Deliverables

As a part of the deliverables, the following artefacts are present in the ZIP folder:

1. Lab write-up
2. Source Files (in folder Code)
3. Makefile (in folder Code)

Source File

This file contains the source code for the implementation of Lab 0. The following functions are included as a part of the source file:

1. `read_from_file(char *read_file_name,vector<int> &readarray)` - Reads the integer values from the file specified in the command line argument.
2. `write_to_file(char *write_file_name,vector<int> &writearray)` - Writes the integer values from the arrays and stores it in a file as mentioned.
3. `main(int argc, char **argv)` - Initialises the variables and calls other functions.

Merge sort functions:

4. `mergefunction(vector<int>& mergearray, int left, int mid, int right)` - Merges the left and the right subarrays onto the main array.
5. `sort_merge(vector<int>& mergearray, int left, int right)` - Implements merge sort by arranging the elements in an ascending order.

Bucket sort functions:

6. `bucket_store(void* arguments)` – Thread handler for bucket sort. Uses a map to sort the array.
7. `getMax(vector<int>& arr, int size)` – Returns the maximum value out of all elements array.

Makefile

Makefile consists of two targets, `all` and `clean`. The steps to execute are also printed in the file.

Compilation steps:

Open command prompt in the folder `Lab1/Code`, type **"make"**. This will create the executable **"mysort"**. To delete the executable, type **"make clean"**.

Execution steps:

To print the author's name, type **"./mysort --name"**

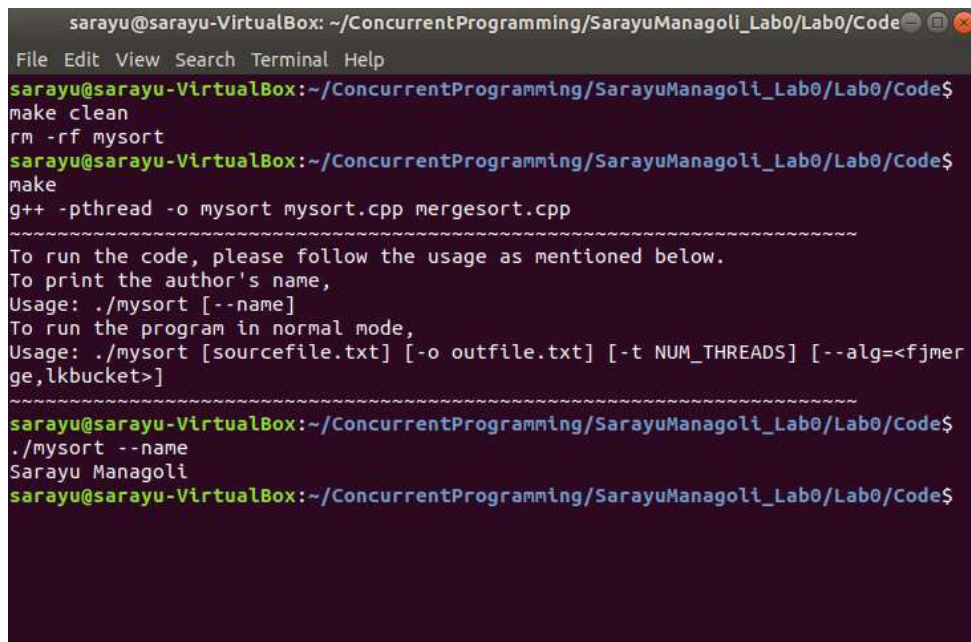
To run the program in normal mode, type **"./mysort [sourcefile.txt] [-o outfile.txt] [-t NUM_THREADS] [--alg=<fjmerge,lkbucket>]"**

Error handling and extant bug:

Error handling has been done if incremental inputs are missing.

Extant bugs: The program does not handle the errors related to the inputs from the user being out of order. This can be considered as future scope of work.

Output:



```
sarayu@sarayu-VirtualBox: ~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code
File Edit View Search Terminal Help
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$
make clean
rm -rf mysort
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$
make
g++ -pthread -o mysort mysort.cpp mergesort.cpp
~~~~~
To run the code, please follow the usage as mentioned below.
To print the author's name,
Usage: ./mysort [--name]
To run the program in normal mode,
Usage: ./mysort [sourcefile.txt] [-o outfile.txt] [-t NUM_THREADS] [--alg=<fjmer
ge,lkbucket>]
~~~~~
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$
./mysort --name
Sarayu Managoli
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$
```

```

sarayu@sarayu-VirtualBox: ~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code
File Edit View Search Terminal Help

sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$
./mysort case1.txt -o your1.txt -t 5 --alg=lkbucket
Thread count is 5
Num count = 5

Unsorted array is:
5
3
10
7
9

Implementing Bucket Sort
Creating thread 0
Creating thread 1
Creating thread 2
Creating thread 3
Creating thread 4
Joined thread 1
Joined thread 2
Joined thread 3
Joined thread 4
Joined thread 5
array_num: 1
Sorted array is:
3
5
7
9
10

Sorted output successfully written to the specified file!
Time taken in ns: 908162
Time taken in s: 0.000908
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$

```

```

sarayu@sarayu-VirtualBox: ~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code
File Edit View Search Terminal Help

sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$
./mysort case1.txt -o your1.txt -t 5 --alg=fjmerge
Thread count is 5
Num count = 5

Unsorted array is:
5
3
10
7
9

Implementing Merge Sort
Creating thread 0
Creating thread 1
Creating thread 2
Creating thread 3
Creating thread 4
Joining thread 1
Joining thread 2
Joining thread 3
Joining thread 4
Joining thread 5
array_num: 1
Sorted array is:
3
5
7
9
10

Sorted output successfully written to the specified file!
Time taken in ns: 526528
Time taken in s: 0.000527
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$

```

```
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$  
./mysort case1.txt -o your1.txt -t 5  
Thread count is 5  
Num count = 5  
  
Unsorted array is:  
5  
3  
10  
7  
9  
  
*****No sorting algorithm specified!*****  
Please mention a sorting algorithm  
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$
```

```
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$  
./mysort case1.txt -o your1.txt -t 151 --alg=fjmerge  
Thread count is 151  
Num count = 382  
  
Thread count is too high. Please provide a value lesser than or equal to 150.  
sarayu@sarayu-VirtualBox:~/ConcurrentProgramming/SarayuManagoli_Lab0/Lab0/Code$
```

References

<https://stackoverflow.com/questions/25833541/reading-numbers-from-file-c>

<https://www.geeksforgeeks.org/merge-sort/>

https://codeyarns.com/2015/01/30/how-to-parse-program-options-in-c-using-getopt_long/

<https://www.geeksforgeeks.org/bucket-sort-2/>

<https://stackoverflow.com/questions/52767944/merge-sort-with-pthreads-in-c>

<https://stackoverflow.com/questions/48965540/how-do-i-insert-arrays-as-values-into-map-in-c>

<https://dyclassroom.com/sorting-algorithm/bucket-sort>