Assignment_03

Open Date: 08-10-2024 Submission date: 14-10-2024

Guidelines

- This is a handwritten assignment, and any single guideline violation will lead to a zero mark on your assignment.
- You have to submit this assignment in both hard and soft form.
- You will have maximum marks if you have done the entire task.
- You are encouraged to get help from the Internet and books.
- Deadlines should be kept in mind **no extension** in assignment dates.
- This is an individual assignment. **PLAGARISM IS NOT ACCEPTABLE!**
- Follow the instructions as it is, otherwise, your assignment would not be accepted at all

Question no 1:
PTHREAD

Implement a code that take two matrix name MATA of size n*m and second Matrix MATB of size m*k and generate a resultant matrix MATC of size n*k by performing multiplication using threads. The number of threads created in system will be equal to n. each thread perform multiplication of each row in MATA with MATB to produced the resultant value for particular row in MATC .

Question no 2:

Implement an "alarm clock" class. Threads call **"Alarm:GoToSleepFor(int howLong)"** to go to sleep for a period of time. The alarm clock can be implemented using the hardware Timer device (i.e. timer.h). When the timer interrupt goes off, the Timer interrupt handler checks to see if any thread that had been asleep needs to wake up now. There is no requirement that threads start running immediately after waking up; just put them on the ready queue after they have waited for the approximately the right amount of time. The person will set an alarm and it will display a message after the set amount of time. Provide the options **to snooz alarm for 10 sec or stop the** alarm as well.

Question no 3:

Implement a queue using a singly linked list. Declare a global variable to hold the value of an integer. Now create two threads. One will open a file and read integers from the file one by one in the declared variable. Second thread will be reading the variable and insert the data into the linked list. After both threads complete their task the main program will output the contents of the queue

to the screen. Your task is to find out if the queue has same numbers that were there in the file or are there some missing values or some numbers in access.

Question no 4:

Write a C++/C code to implement the given sorting Algorithms. The main program thread first generates 1,000,000 Random Integers (or array range can be inserted dynamically) at runtime and stores these integers in four files (Unsorted_00.txt, Unsorted_01.txt, Unsorted_02.txt, Unsorted_03.txt) each containing 250,000 integers.

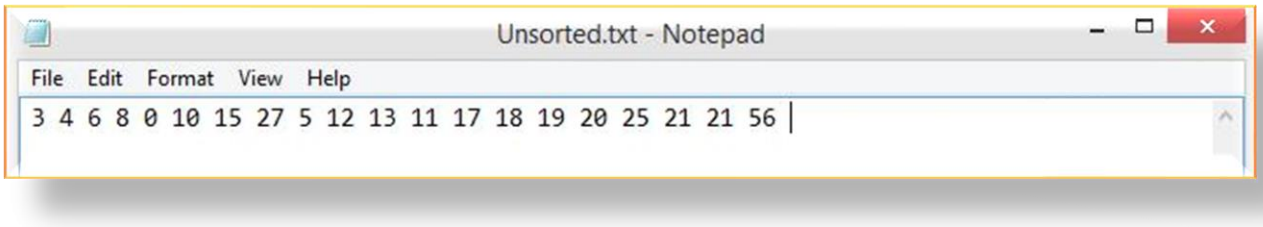
- The main thread will then create different threads each one implementing one of the below given sorting algorithm and parent will wait for these threads to complete their execution.
- Each chunk is to be treated as an individual sorting task and implemented by a separate thread running one of the algorithms given below
- Finally when all the threads complete their tasks, they will exit by saving the results in their respective files, merge.txt, selection.txt, insertion.txt and quick.txt.
- When a thread starts it will print its algorithm name, start time and thread id
- When the thread ends it will print its exit time
- The main program then uses these results further and merges the four sorted lists together to make one big sorted list
- Final sorted list should be stored in a text file named as (Sorted.txt).

The sorting algorithms that you are supposed to implement are:

I. **Merge Sort**
II. **Selection Sort**
III. **Insertion Sort**
IV. **Quick Sort**

Input, output sample:

Sample Input



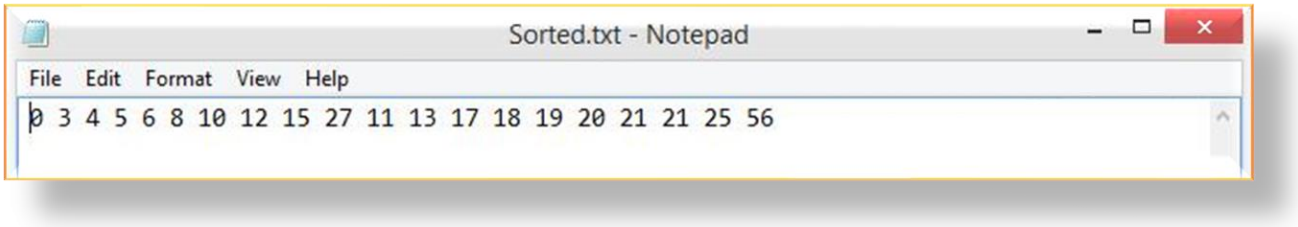
- After Merge sort on **First** chunk:

0	3	4	5	6	8	10	12	15	27

- After Selection Sort on **Second** chunk:

11	13	17	18	19	20	21	21	25	56

Sample Output



Question no 4:
Fork-Join Parallelism

Take an array of N size sort the data using thread. The apply fork and joint framework to divide the array into left and right array then create two processes using fork system call that is left process and right process that find the maximum number in each sub array. Finally combine the result to find the maximum value among both resultant values.