Jordan University of Science and Technology
CS375 Operating systems - Assignment 3
Fall 2021-2022

Objectives:

Student should be able to:

- ✓ Write a simple Multithreading problem that requires *Task Parallelism*.
- ✓ Write a simple Multithreading problem that requires *Data Parallelism*.

What to submit: Due: December 11, 2021 (there will be NO QUIZZES next week)

- ✓ You have to work on your assignment by yourself.
- ✓ You have to submit the report and codes on time.
- ✓ The file name of your report has to be formatted as CS375_Ass#3_X_Y.doc (or .docx or .Pdf) where **X** is your full name; **Y** is your ID number.
- ✓ You are required to submit:
    - All assignment's parts as **fully documented source codes** (feel free to use any programming language).
    - A simple report contains **print screen for the final output of each part** and a **short discussion on the results you got**.
    - Add whatever necessary details you need.

After submitting your work, you should schedule an appointment. Check the discussion date(s) on the e-learning to discuss your work **on your personal computer.** Your grade will be given based on your both submission and discussion. You are expected to demonstrate any related question(s) **without refereeing to any supporting material during the discussion.**

Jordan University of Science and Technology
CS375 Operating systems - Assignment 3
Fall 2021-2022

Project Description

Refer to the textbook for Chapter 4 Programming Project: Multithreading programming problems on pages 195- 197, and 199 of the 9th edition.

- Problems: 4.21, 4.24 and 4.26
- Project 2—Multithreaded Sorting Application (page 199)

Part I - Multithreaded / Task parallelism

a. Write a multithreaded program that calculates various statistical values for a list of numbers (try small/large/very large numbers). This program has to create **three** separate worker threads. One thread will determine the average of the numbers, the second will determine the maximum value, and the third will determine the minimum value. For example, suppose your list contains the Integers `90 81 78 95 79 72` and `85`, The program will report the following:

```
    The average value is 82
    The minimum value is 72
    The maximum value is 95
```

The variables representing the average, minimum, and maximum values will be stored globally. The worker threads will set these values, and the **parent** thread will output the values once the workers have exited.

b. Write a multithreaded program that outputs prime numbers. This program should work as follows: The user will run the program and will enter a number as an input. The program will then create a **separate thread** that outputs all the prime numbers less than or equal to the number entered by the user.

c. Solve the previous problems **without using threads**. Use the **time function** with each part (a, b and c) and show, for each one, you have to answer these questions **in your report:**

1. What is the exact execution time?
2. Is there any difference? Why?

HINT1: To ensure answering questions correctly, you have to try your code on different list sizes, i.e. try for 100-element list, then 1000, then 10000, 100000… 10^9 elements and so on.

HINT2: you can compute running time for certain code segment in many different ways, for example, in C++ programming language you can follow this method: https://www.geeksforgeeks.org/measure-execution-time-function-cpp/

Jordan University of Science and Technology
CS375 Operating systems - Assignment 3
Fall 2021-2022

Part II – Multithreaded / Data parallelism

a. Write a multithreaded sorting program that works as follows: A list of integers is divided into **Four** smaller lists of equal size. **Four** separate threads (which we will term **sorting threads**) sort each sub-list using a **sorting algorithm of your choice**. (You may use any built in sorting function). A **Fifth thread** then merges the Four sub-lists — a **merging thread** — which merges the Four sub-lists into a single sorted list.
Because global data are shared cross all threads, perhaps the easiest way to set up the data is to create a global array. Each sorting thread will work on one 1/4 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 four sub-lists into this second array. Graphically, this program is structured **similarly** according to Figure 4.20 if we intend to partition the array into two parts.
This programming project will require passing parameters to each of the sorting threads. In particular, it will be necessary to identify the starting index from which each thread is to begin sorting. Refer to the instructions in Project 1 for details on passing parameters to a thread. The **parent** thread will output the sorted array once all sorting threads have exited.

b. Solve the previous problem **without using threads**. Use the **time function** with each part (a and b) and show, for each one, you have to answer these questions **in your report:**
1. What is the exact execution time?
2. Is there any difference? Why?

HINT1: To ensure answering questions correctly, you have to try your code on different list sizes, i.e. try for 100-element list, then 1000, then 10000, 100000… 10^9 elements and so on.

HINT2: you can compute running time for certain code segment in many different ways, for example, in C++ programming language you can follow this method:

https://www.geeksforgeeks.org/measure-execution-time-function-cpp/

Edited: Nov 27, 2021