Jordan University of Science and Technology
CS375 Operating systems - Assignment 4
Spring 2019-2020

Objectives:

Student should be able to learn how to write a multithreaded application that utilizes different software synchronization tools to prevent possible race condition scenario.

What to submit: Due: April 19, 2020 [there will be NO postponing]

- ✓ You have to work your assignment as a team with at MOST TWO members.
- ✓ At least one of the group members has to submit the report and codes on time.
- ✓ The file name of your report has to be formatted as CS375_Ass#4_X_Y.doc (or **.**docx or **.**Pdf) where **X** is the first student ID number and **Y** is the second student 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 final results you got**.
  - Add whatever necessary details you need **[optional]**

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 4
Spring 2019-2020

Description

Write a multi-threaded application in which there are **three threads**. All threads open the same file. The content of this file is assumed to be lines that each consist of two numbers (you can start by a file that contain the line "0 0"). The first number in each line is the id of the thread that wrote the second number. Each thread performs the **following steps:**

- Open the file.
- Reads the last line in the file
- Close the file.
- Increment the second number in the last line by 1.
- Perform some computation such as finding the sum of the numbers between 1 and 1000000.
- Open the file.
- Write the thread id and the incremented number to the end of the file.
- Close the file.

Note that: at the end, all numbers (second number in each line) that are written to the file by all threads should be in sequence. Also, you should figure out what the critical region is, and to prevent the **race condition**, write your own code using:

a. Mutex lock synchronization tool
b. Semaphore synchronization tool