Lab 2

Multi-Threading, Real Time Tasks, and Simple Synchronization

WEEK-1

Objective

The goals are to:

- Learn how to create multiple threads within a process.
- Appreciate the computational advantage and potential disadvantages of multi-threading.

Prelab

Before coming to the lab, you should list and describe the functions to: *create*, *exit* and *join* a pthread. Include information like the function name, arguments and return values.

Remember to include a simple flow chart or pseudo code for the program you need to implement.

Lab Procedure: Threaded Program

1. This first week of lab 2 you will implement a searching algorithm. You will do that using different number of **pthreads**. Given a text file with a matrix, you will need to search for a specific number in that matrix. The number to search should be an input parameter of the program. You can find a few example files named "NxM.txt" on Canvas, under *Modules > Additional Lab Files.*

   Each of those example files is formatted as follows: the first line contains the number of rows and number of columns of the matrix. The next lines contain the actual matrix data, where each number is separated by a space. Consider the following example

   ```
   3 10     // these are the number of rows and columns
   1  2  3  4  5  6  7  8  9 10     // matrix data
   11 12 13 14 15 16 17 18 19 20
   21 22 23 24 25 26 27 28 29 30
   ```

   Be sure to check for errors when opening and reading the files.

2. After the search has been completed, your program should display the results of the search (i.e., how many times did your algorithm find the specific number in the matrix). It should also display the search execution time. Perform the matrix

search with different number of threads. You should try the cases shown in the table-1 (the search results should be the same, but the execution times may not be the same). For each case, you should run your program multiple times and calculate average search times. Fill in the observation table-1.

Case	**Avg. Search Time (2x100.txt)**	**Avg. Search Time (15x15.txt)**	**Avg. Search Time (20x10.txt)**
One thread to search the entire matrix			
One thread for searching each row of the matrix			
One thread for searching each column of the matrix			
One thread for each element of the matrix			

Table 1: Observation Table

In addition to the cases above, think of other number of threads you can try. What is the best number of pthreads for doing the search, according to your results? Why do you think that was the case? Answer the questions in your report.

3. Since you are using **pthreads**, you need to add the parameter *pthread* to your linker options when you build your project. To do this in Eclipse, go to the *Properties*, and under where it says *"GCC C Linker"*, click on *Libraries*. On the right, this will bring up places to add libraries to link to, and their paths (if needed). Simply click on the add button under the *libraries* (denoted *–l*) and type in *pthread*. The pthread library is in a known location so we do not need to add a path of where to find the library. Also make sure in your program you include the header: *pthread.h*.

WEEK-2

Objective

The goals are to:

- Learn how to initialize threads as Real Time tasks.
- Notice the advantages/disadvantages of synchronizing task/threads relying on time events alone.

Prelab

Before coming to lab on the second week of lab 2, check the file *"RT_tasks_functions.c"* (on *Canvas*). Investigate the different functions and structures shown in the file. You do not need to submit a description of the functions or their arguments and return types, but make sure to read about them. You will need them for this assignment, so you should have an idea of what they are doing. What you do need to submit, as always, is a simple flow chart or pseudo code for the program you need to implement this week.

Lab Procedure: Real Time Tasks in User Space

1. In this part of the lab there are two files that contain strings that need to be collated together into one file. An original text was divided into two parts. The odd lines of the text were saved into the file named **first.txt**, and the even lined were saved into the file **second.txt**. Your task is to reconstruct and display the original text. To achieve this you will develop a program that creates three pthreads, and a common buffer (a character array).

2. Two of the pthreads will be used as file readers that read in one line of their file at a time (the first pthread opens and reads first.txt, the second opens and reads second.txt), and save the read line to the common buffer. The third pthread will be used to take information from the common buffer, and store it in a string array that can hold up to 20 strings. Finally, the reconstructed text will be printed in the main thread.

3. After the files have been opened, all pthreads should initialize themselves as real time tasks, and schedule themselves at alternating times so that data can be read in from the files and stored in the array in the correct sequence. You will need to synchronize the tasks by adjusting their periods and starting times. Make sure to try different combinations. You may want to start with relatively long periods, to

ensure that the reading and saving into the buffers is completed on time. Then, try reducing the periods. **Can you find the minimum period(s) that still allows you to reconstruct the original file properly?**

4. Remember to you include the appropriate headers in your program (check the *"RT_tasks_functions.c"* file). Since you are creating pthreads, the corresponding header and library need to be included.

5. Make sure to report the experiments that you ran, the times that you used, etc