

## Lab2

# **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 multithreading.

## 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.

## Lab Procedure: Threaded Program

1. Since you are using **pthread**s, first, you have to include the header file [pthread.h](#).
  - If you are cross-compiling using Eclipse on the Workstation, please copy the configurations that we have done in Lab0 to your Lab2's Eclipse project.
  - If you are compiling on the Pi, you will need to add the parameter *pthread* to your linker options when you build your project. On the terminal, you could run:

```
gcc Lab2_Week1.c -o Lab2_Week1 -lpthread
```

2. This first week of lab 2, you will implement a 1D-convolution procedure. You will do that using a different number of **pthread**s. Given a text file with a matrix and a 1D filter of size 1x3, you will need to convolve the filter with the matrix. You can find a few example files named "NxM.txt" on Canvas, under *Modules > Lab2 and a template code "Lab2-Week1-Basic.c"*

Each of those example files is formatted as follows: the first line contains the number of rows and columns of the matrix. The next few lines (equal to row count) contain the actual matrix data, where each number in a row is separated by a space. After that, a new line with rows and column numbers holds the filter information. The last line contains the filter data. Consider the following example.

```
2 5 // number of rows and columns of the matrix
1 2 3 4 5 // First Row of the Matrix
11 12 13 14 15 // Second Row of the Matrix
1 3 // number of rows and columns of filter
1 2 1
```

Be sure to **check for errors** when accessing the files, such as opening, closing and reading. You could find a short video tutorial on how to convolve nx1 matrix with a 3x1 filter on this [youtube](#) link. For the above example, you should get the following results.

---

```
4 8 12 16 14
34 48 52 56 44
```

3. After the convolution has been completed over the entire matrix, your program should display:
- The results of the convolution
  - Execution time
  - Total number of convolution operations performed

Perform the convolution with a different number of threads. You should try the cases shown in the table-1 (the convolution results and the total number of convolutions operations should be the same, but the execution times may not be the same). For each case, you should run your program multiple times (at least 10) and calculate average convolution times. Fill in the observation table-1.

Case	Avg. Conv. Time (2x10.txt)	Avg. Conv Time (20x10.txt)
One thread to convolve the filter with the <b>entire matrix</b>		
One thread for convolving the filter with <b>each row</b> of the matrix		
One thread for convolving <b>each element</b> of the matrix		

Table 1: Observation Table

According to your results, what is the best number of pthreads for doing the convolution? Why do you think that was the case? Answer the questions in your report.

# 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 "[Lab2-Week2-Basic.c](#)". Investigate the different functions and structures shown in the file. Also add a brief description about following functions along with their input arguments and return values:

- sched\_setscheduler() or (pthread\_attr\_init(), pthread\_attr\_setschedpolicy() and pthread\_attr\_setschedparam)
- clock\_gettime()
- clock\_nanosleep()

## 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 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 you use `sudo` while running your executable.
6. Add `time.h` to access all timer related functions and structures.
  - If you are cross-compiling using Eclipse on the Workstation, please copy the configurations that we have done in Lab0 to your Lab2’s Eclipse project.
  - If you are going to compile on the Pi, you need to link your executable to `rt` library. So, while compiling you need run:

```
gcc Lab2_Week2.c -o Lab2_Week2 -lpthread -lrt
```
7. Make sure to report the experiments that you ran, the times that you used, etc.

## References

- [Multithreaded Programming](#)
- [HOWTO build a simple RT application](#)
- [HOWTO build a basic cyclic application](#)