Lab 2 Assignment
CS 244- Spring 2019

The purpose of this assignment is to get you familiar with multi-thread programming on embedded systems such as Raspberry Pi. You will need to practice with two state-of-the art multi-thread programming methods: (1) OpenMP and (2) pthreads.

Step 1 – Learning the basics of multi-thread programming with Open MP

1. First, check the resources below to understand programming with Open MP.
   http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf
   http://www.openmp.org/wp-content/uploads/openmp-examples-4.5.0.pdf
2. Extra readings will also help, you should explore yourself.
3. You should be able to explain the scope of OpenMP and possible advantages of using it.

Step 2 – Understanding the modifications on canny_local.c for OpenMP

1. Download the modified canny_local.c code named canny_local_omp.c from the course website under Assignment 2. You will use the test.pgm from last week as the image file.
   https://sites.google.com/view/ecps204-winter2018/project
2. Compare the differences between the original canny_local.c file. Notice that the modifications are made for Blur in the x – direction inside gaussian_smooth fun.ction.
3. Make sure you comment on these differences on the report.

Step 3 – Compiling and running the modified code

1. To compile the canny_local_omp.c code, you need to add –fopenmp option while you compile the code with gcc. For example:
   - gcc canny_local_omp.c –lm –fopenmp –o ./canny_omp
2. Compare the performance difference between canny_local.c and canny_local_omp.c. Use the time command to acquire the time information. For example:
   - time ./ canny_omp test.pgm 1.0 1.0 0.8
3. The real execution time of the code will be displayed as real x s.
4. Comment on the performance improvement using the Open MP.
5. Put the snapshots of the results that display the time of the run in your reports.

Step 4 – Modifying the Blur in the y - direction

1. After you studied and understood how Blur in the x – direction is modified for multi-thread using OpenMP, your task is to modify the Blur in the y – direction with similar style as the sample code.
2. Compile and test the performance change after you have done Blur in the y – direction in terms of execution time.
3. Explain your methodology and show the performance improvement after modification.
4. Put the snapshots of the results that display the time of the run in your reports.

Step 5 – Learning the basics of multi-thread programming with pthread

1. You will also need to read the online resources to study pthread. The following link will be useful for this project in terms of multi thread programming with pthread. http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
2. You should be able to explain the scope of `pthread` and possible advantages of using it.

Step 6 – Understanding the modifications on canny_local.c for pthread

1. Download the modified canny_local.c code named canny_local_pthread.c from the course website.
2. Compare the difference between the original canny_local.c file. Notice that the changes are made in the code for Blur in the x – direction inside gaussian_smooth function.
3. Make sure you comment on these differences on the report.

Step 7 – Compiling and running the modified code with pthreads

1. To compile the canny_local_pthread.c code, you need to add -lpthread option while you compile the code with gcc. For example:
    - gcc canny_local_ pthread.c –lm -lpthread –o ./canny_pthr
2. Compare the performance difference between canny_local.c and canny_local_pthread.c by exploiting the time command.
3. Comment on the performance improvement using the pthreads.
4. Put the snapshots of the results that display the time of the run in your reports.

Step 8 – Modifying the Blur in the y - direction with pthreads

1. After you studied and understood how Blur in the x – direction is modified for multi-thread using pthreads, your task is to modify the Blur in the y – direction with similar style as the sample code.
2. Compile and test the performance change after you have done Blur in the y – direction in terms of execution time.
3. Explain your methodology and show the performance improvement after modification.
4. Put the snapshots of the results that display the time of the run in your reports.

Step 9 – Compare OpenMP and pthreads

5. Compare the performance of the code without multi-threading, with implementation of OpenMP and with implementation of pthreads.