Operating System Homework 3 Report

Student ID: 0310120
Name: 陳家煒

Detailed description of the implementation:

(Number of threads, the purpose of those threads, how do you use mutex lock and semaphore…etc.)

HW3-1:

Number of thread: 30

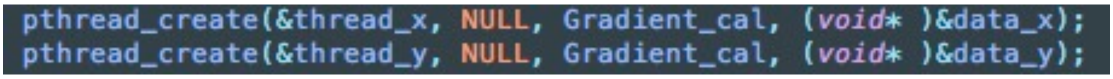
First of all I create **5** main threads for each input image, and in each main thread I use **1** thread for converting RBG image to Grey and **5** threads for convolving. Therefore, there are (**5 x (1 + 5) = 30**) threads for HW3-1.

As to synchronization, I use **semaphore** to ensure converting is performed before convolving.

HW3-2:

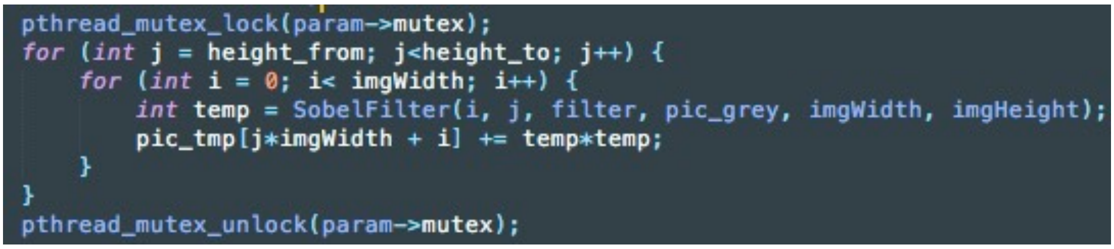
Number of thread: 55

First of all I create **5** main threads for each input image, and in each main thread I use **1** thread for converting RBG image to Grey and **5** threads for convolving. Within each thread for convolving, I create two more thread for calculating Gx and Gy. Therefore, there are
(**5 x (1 + 5 x (2)) = 55**) threads for HW3-1.



```
pthread_create(&thread_x, NULL, Gradient_cal, (void* )&data_x);
pthread_create(&thread_y, NULL, Gradient_cal, (void* )&data_y);
```

As to synchronization, I use **semaphore** to ensure converting is performed before convolving. Plus, I let Gx and Gy's result store in **same array**, so I use **mutex** to guarantee that array will serve as a critical section.

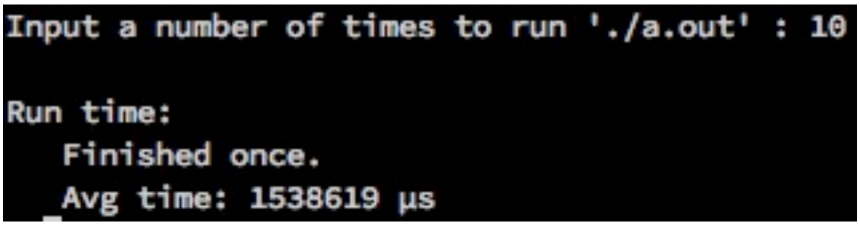


```
pthread_mutex_lock(param->mutex);
for (int j = height_from; j<height_to; j++) {
    for (int i = 0; i< imgWidth; i++) {
        int temp = SobelFilter(i, j, filter, pic_grey, imgWidth, imgHeight);
        pic_tmp[j*imgWidth + i] += temp*temp;
    }
}
pthread_mutex_unlock(param->mutex);
```

Your speed:

HW3-1 baseline	HW3-1 speed	HW3-2 baseline	HW3-2 speed
1538619 us	833131 us	1427514 us	766081 us
Speedup: 1.8467		Speedup: 1.8633	

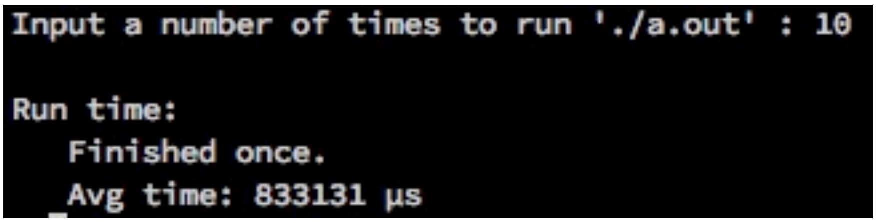
Hw3-1 baseline



```
Input a number of times to run './a.out' : 10

Run time:
    Finished once.
    Avg time: 1538619 µs
```

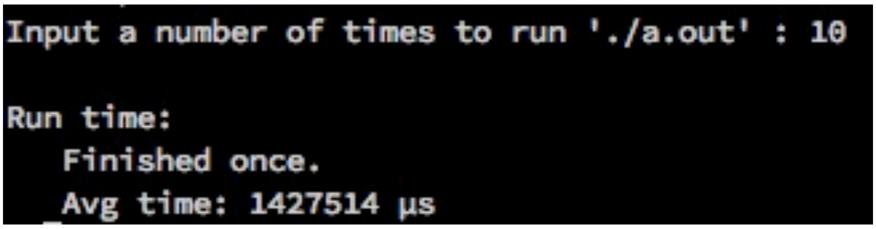
Hw3-1 thread + synchronization (semaphore)



```
Input a number of times to run './a.out' : 10

Run time:
    Finished once.
    Avg time: 833131 µs
```

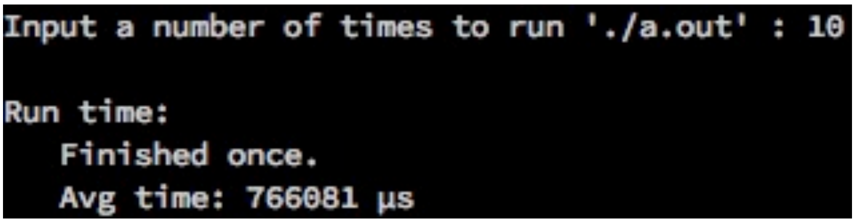
Hw3-2 baseline



```
Input a number of times to run './a.out' : 10

Run time:
    Finished once.
    Avg time: 1427514 µs
```

Hw3-2 thread + synchronization (mutex + semaphore)



```
Input a number of times to run './a.out' : 10

Run time:
    Finished once.
    Avg time: 766081 µs
```

Problems encountered and solutions:

1. Decision of thread number:
   It's not true that more threads guarantee more speedup, so I have to figure out the decent number of threads for each task by doing experiment.
2. Parallel programming is sometime hard to debug:
   Think more carefully before coding since thread might have some race issue if we don't consider synchronization. Also, divide the problem into different independent problem will help you think intuitively when doing parallel programming.