**Operating System Homework 4 Report**

Student ID: 0516215

Name: 林亮穎

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

Number of threads: 2

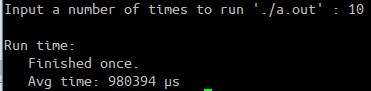
Purpose of those threads : 一條thread用來做Mean filter，另一條用來做Sobel filter

如何實作：

將整張圖分成k塊(split by row)，並在main thread開一個計數用的clock。當t==0時，第一條thread去做第一塊圖的mean filter，做完後將thread\_cnt++，第二條thread這時先什麼都不做就將thread\_cnt++，當main thread 發現thread==2時，就代表兩條thread這時都做完各自在這回合內該做的工作了，於是便將global\_clock++，開始進入下一回合。下個回合(t==1)，第一條thread會做第二塊圖的mean filter，第二條thread會去做第一塊圖的sobel filter，以此類推，只要在k+1個回合就能將整張圖該做的兩種filter通通做完。

所以同步的重點就在如何保護thread\_cnt這個變數，我用pthread的semaphore來實作。當一條thread完成自己在這回合內的工作時，會將thread\_cnt++，並且接著做wait，直到另一條thread作完也將thread\_cnt++以後(這條thread也會做wait)，當main thread發現到thread\_cnt == 2時，會發出broadcast通知其他thread不用再做wait了，並將global\_clock++，開始進行下個回合的工作。由於pthread的semaphore做wait需要跟一個mutex搭配，所以code中在thread\_cnt前會需要先lock(mutex)。

**Your speed:**



1867302 / 980394 = 1.9046

約加速1.9倍

**Problems encountered and solutions:**

由於單純靠multithread加速的結果不盡理想，所以在合乎spec規定的條件之下，另外再做了一些輔助加速的手法。