1.What have you done

這次作業要使用 pthread 來執行 image smoothing，我使用 busy waiting and a mutex 的方式，一開始我宣告:

(1)thread_num : number of thread

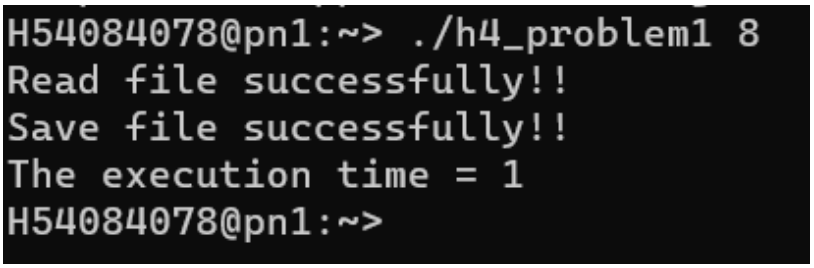
(2)local_height : 每個 thread 被分配到要處理多少部分的 image

(3)兩個 counter(counter1 跟 counter2) : 兩個 counter 功能一樣，都是做 busy waiting : 我用迴圈 create 出多個 threads，每一個 thread 被 create 後會進入我定義的 run function 裡面，thread 進到 run function 後 counter 就+1。counter+1 這個指令是 critical section，所以我用 pthread_mutex_lock 和 pthread_mutex_unlock 保護起來，避免 race condition。而之後我用 while 迴圈等到 counter 等於 number of thread 時才會繼續動作。當 counter 等於 number of thread 時，我讓每個 thread 進行平滑運算，而 thread 0 執行 swap function。執行完 run function 後 thread 就會跳回 main function，在 main function 裡我用迴圈把每個 terminated thread 做 pthread_join，結束 multithread image smoothing。

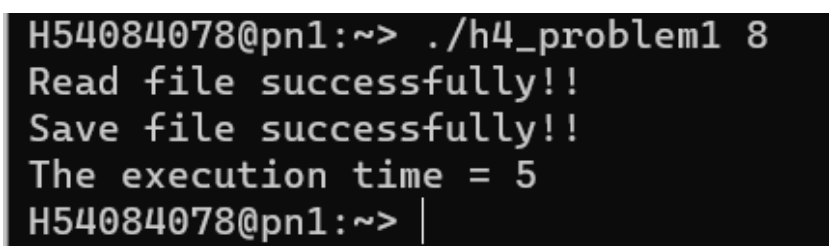
2.Analysis on your result

(1)平滑次數(8 threads) : 10 vs 100 vs 1,000 vs 10,000

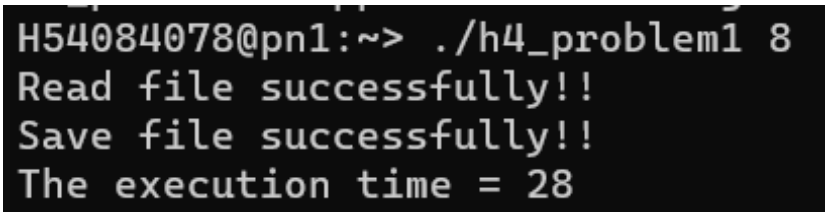
10:



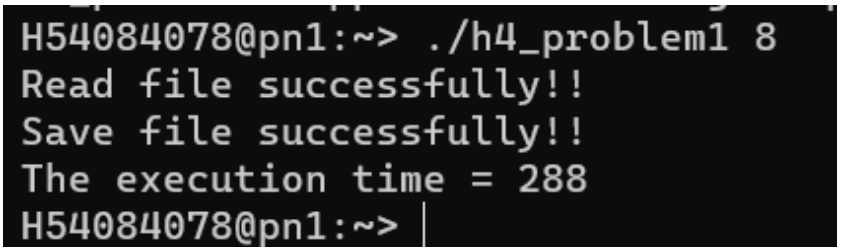
100:



1,000:



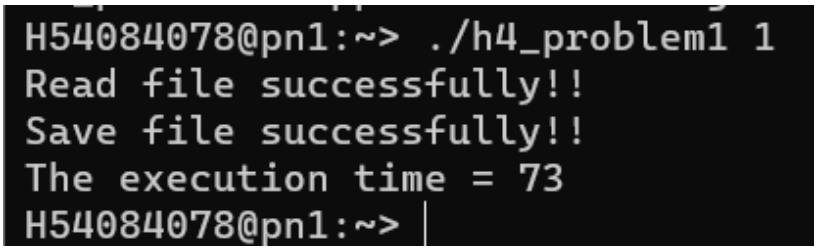
10,000:



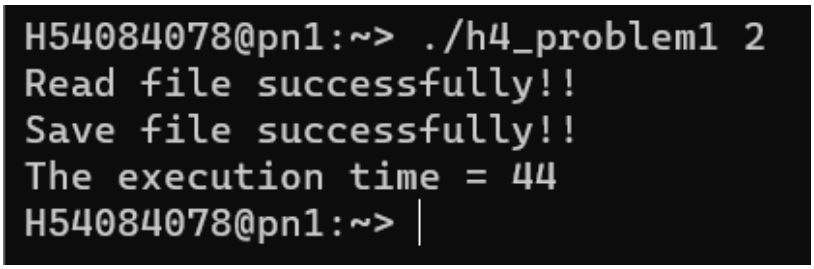
平滑次數每增加 10 倍，執行時間就以非線性增加。因此得知，平滑越多次，每條 thread 要執行的迴圈次數越多，所花時間也就越多。

(2)thread 數量(smooth 1,000 次) : 1 vs 2 vs 4 vs 8 vs 16

1:



2:



4:

```
H54084078@pn1:~> ./h4_problem1 4
Read file successfully!!
Save file successfully!!
The execution time = 24
H54084078@pn1:~>
```

8:

```
H54084078@pn1:~> ./h4_problem1 8
Read file successfully!!
Save file successfully!!
The execution time = 28
H54084078@pn1:~>
```

16:

```
H54084078@pn1:~> ./h4_problem1 16
Read file successfully!!
Save file successfully!!
The execution time = 49
H54084078@pn1:~>
```

4 thread 所花時間<8 thread 所花時間<2 thread 所花時間<16 thread 所花時間<1 thread 所花時間。

越多條 thread 執行時間越短，所以在 4 條 thread 以下時，thread=4 的執行時間< thread=2 的執行時間<thread=1 的執行時間。然而當 thread=8 的時候，執行時間卻比 thread=4 的時候多，thread=16 的執行時間又比 thread=8 的執行時間越多，推測在 create thread 和 terminate thread 所產生的 overhead 大於平行處理所節省的時間。

3.Any difficulties?

pthread 相對於 mpi 簡單許多，只要正確使用 pthread_create、pthread_mutex_lock、pthread_mutex_unlock、pthread_join 等這些 pthread 提供的 API，然後保護好 critical section 就能做出平行處理。

4.(optional) Feedback to Tas

助教辛苦了，要改這麼多考卷還要看我們的作業>＜