NCTU OS HW2 report 2018

Name: 林亮穎

Student ID: 0516215

Question	Answer
Q1. Briefly describe about your design for the problem "Sum Checker" and total num of threads you used in your code.	
Q2. Show your thread info screenshots while "Sum Checker" code running.	
Q3. Compare the time between Single-thread and Multi-thread.	
Q4. What you learned from doing OS hw2 or some improvements you want to say to TAs.	

p.s You can reference to homework info pdf and show your answer as the format for Q2 and Q3.

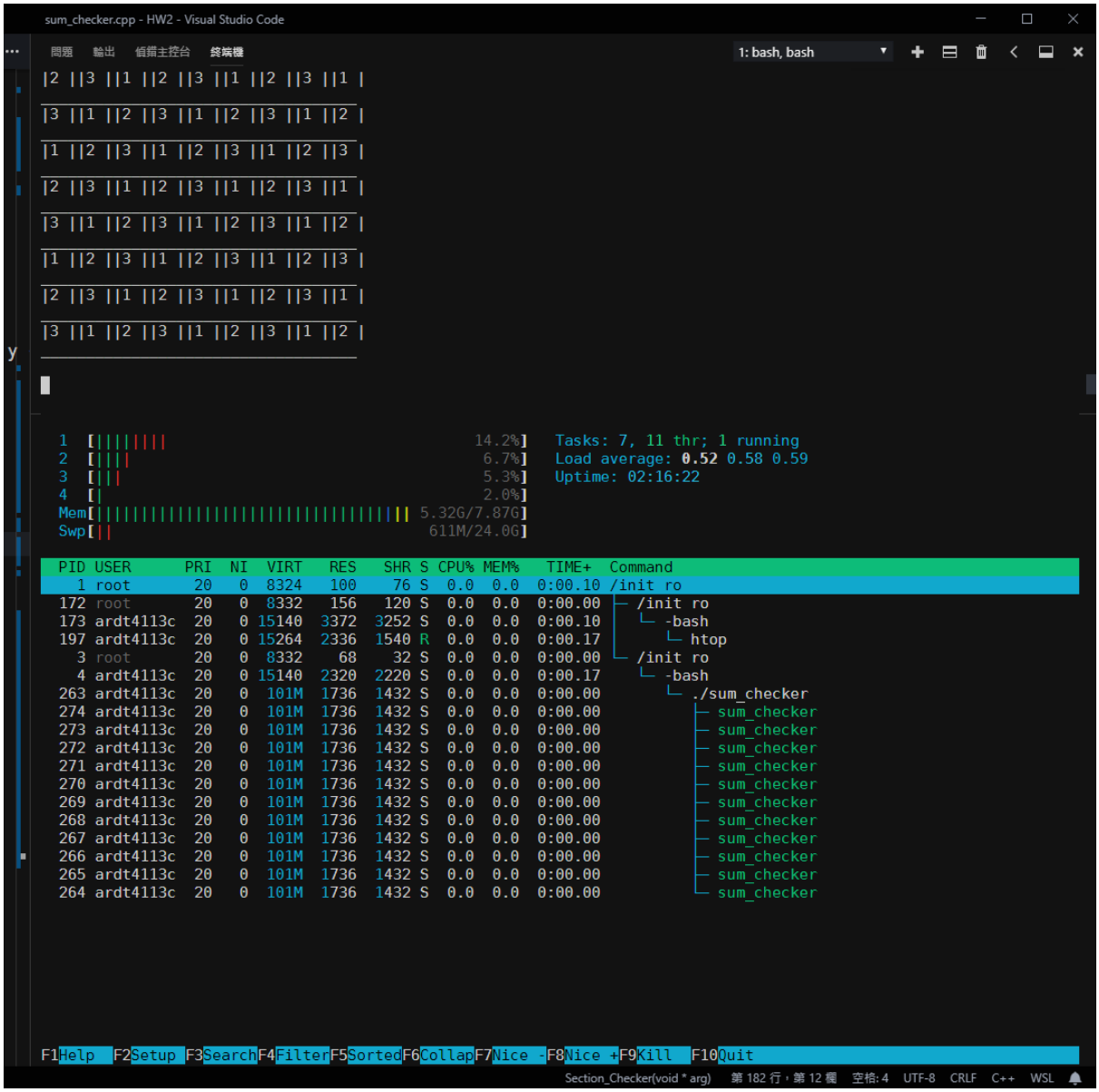
一、第一題答案

我用了 11 條 thread 來完成 Sum Checker，其中一條是用來掃過整個 puzzle 的九個 row，確認每個 row 的數字總和是不是都相同；另一條是用來確認每個 column 的數字和是否相同；剩下的九條 thread 當中的每一條 thread 各自負責檢查其中一個 3*3 方陣的數字和，如此一來 puzzle 中的九個 3*3 方陣都會被檢查到。

一開始執行程式時，我以第一條 row 的數字和做為每個區域內的數字和基準。而在每一條 thread 在檢查各自負責區域內的數字和時，若是數字和與第一條 row 的數字和相同，該條 thread 就會回傳 0，表示該區域符合 puzzle 的規則；反之若是不相同，thread 便會立即中止並回傳 1，表示該區域不符合規則。

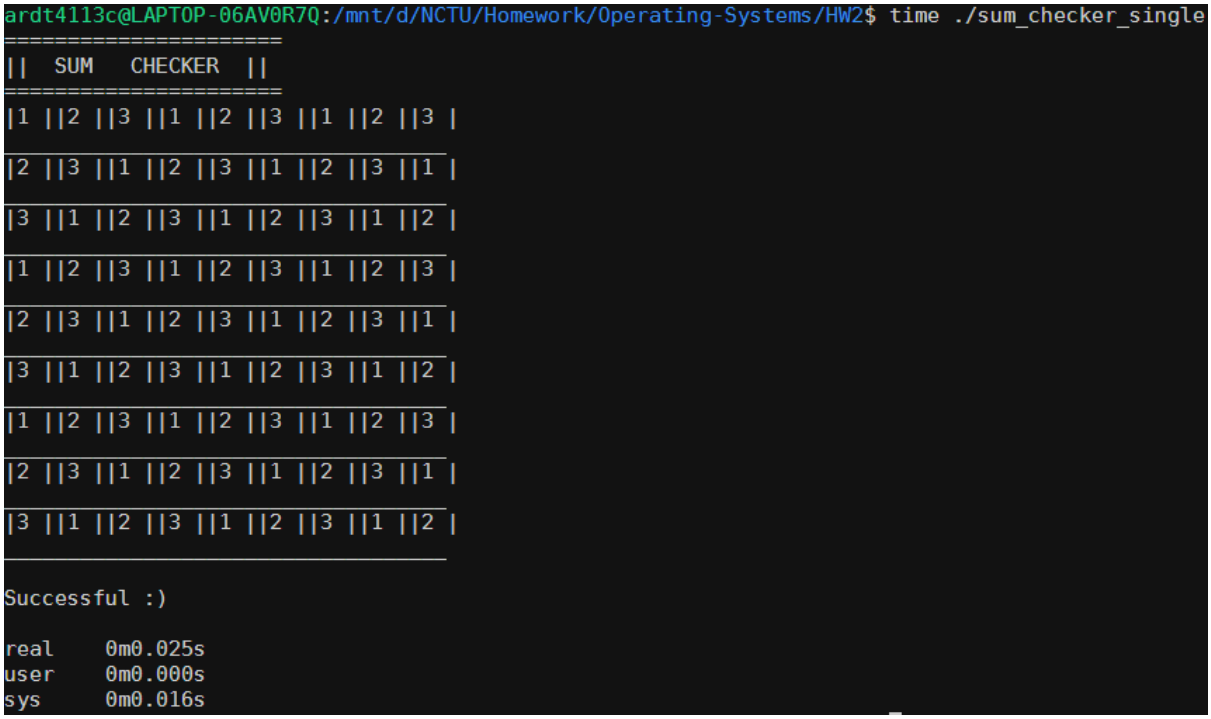
所以當 main thread 在執行 pthread_join()要合併各條 thread 的結果時，若是有任何一條 thread 回傳的結果是 1，我就會將 rv 設為 0，代表這個 puzzle 有部分區域的數字和與其他區域不同，程式就會輸出" Must check again " 的錯誤訊息。而若是每條 thread 皆回傳 0，代表整個 puzzle 皆符合規則，程式便會輸出" Successful" 的成功訊息。

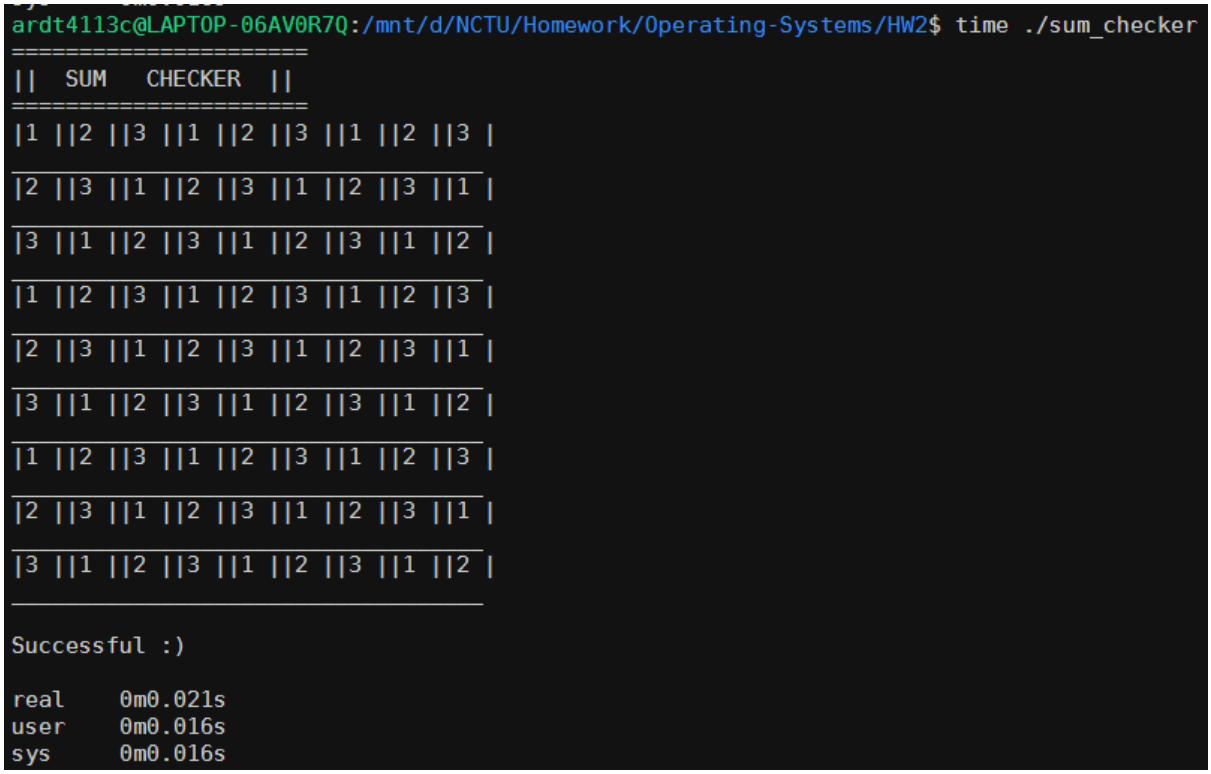
二、第二題答案



由截圖可看出 sum_checker 執行時真的創了 11 條 thread。然而由於我寫的 sum_checker 程式只要約 0.02 秒就可以跑完，因為跑得太快了，執行的時候在 htop 上並不能看到這些創出來的 thread。所以為了產生上圖，我在每條 thread 結束前另外加了 sleep(5)，如此一來 htop 才來得及顯示這些創出來的 thread。另外，在下一題以及正式繳交的程式上我並沒有加上 sleep(5)。

三、第三題答案





上圖是 single thread 版的執行結果，下圖是 multithread 版的執行結果，可以看出 multithread 的較 single thread 的快了 0.004 秒，Speed up= Ts/Tm = 0.025/0.021 =1.1904，multithread 版比 single thread 版的約快上 1.19 倍。

四、第四題答案

在這次作業中我學到了如何利用 pthread 的函式庫，創建新的 thread 以及合併多條 thread 的執行結果，整個作業並不會太難，也能學到新東西。

另外，給 TA 的建議是：我覺得作業的 puzzle 尺寸給得太小了，無論是 single thread 還是 multithread，程式基本上花不到 0.05 秒就能跑完，這樣子實在是很難看出 multithread 的所帶來的 speedup 好處。也因此在寫作業的 Q3 時，時常會出現 single thread 比 multithread 快上 0.001 秒之類的結果，很容易被這類的小誤差影響實際計算 speedup 的結果。