CS305 作業系統概論 Prog. #2 Multithreading. 說明報告.docx

作者 ：1043335 賴詩雨

版本	修改日期	說明	修改者
1.0.0.0	2017-05-08	初稿	賴詩雨
1.0.0.1	2017-05-011	修改	賴詩雨

目錄

第1章　程式完成部分:

a. 基本功能

b. 進階功能 - advmerge

c. 進階功能 - advquick

第2章　設計理念

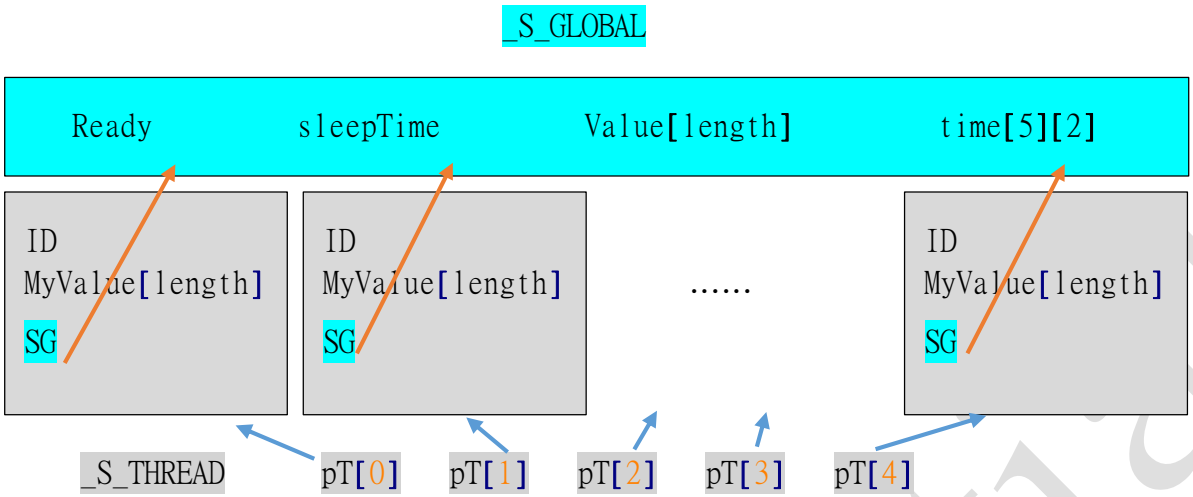
利用 pthread 的共用變數讓所有 work_thread 等待，直到 main_thread 讀好資料後，啟動變數更改成 1(Ready)，所有的 work_thread 才開始一起排序。

2.1 程式中特殊介紹

1. 有三個結構，分別為 _S_GLOBAL 、 _S_THREAD 和 _S_THREAD
2.
```
typedef struct _S_GLOBAL
{
    int       Ready;
    int       sleepTime;
    int       Value[length];
    int       time[5][2];
} S_GLOBAL, *P_GLOBAL;
```
3.
```
typedef struct _S_THREAD
{
    int        ID;
    int        MyValue[length];
    P_GLOBAL   SG;
} S_THREAD, *P_THREAD;
```
4.
```
typedef struct _S_ADV
{
    int        PID;
    int        CID;
    int        HalfData[length];
    P_THREAD   ST;
} S_ADV, *P_ADV;
```
5. 整個程式中有用到的結構宣告為

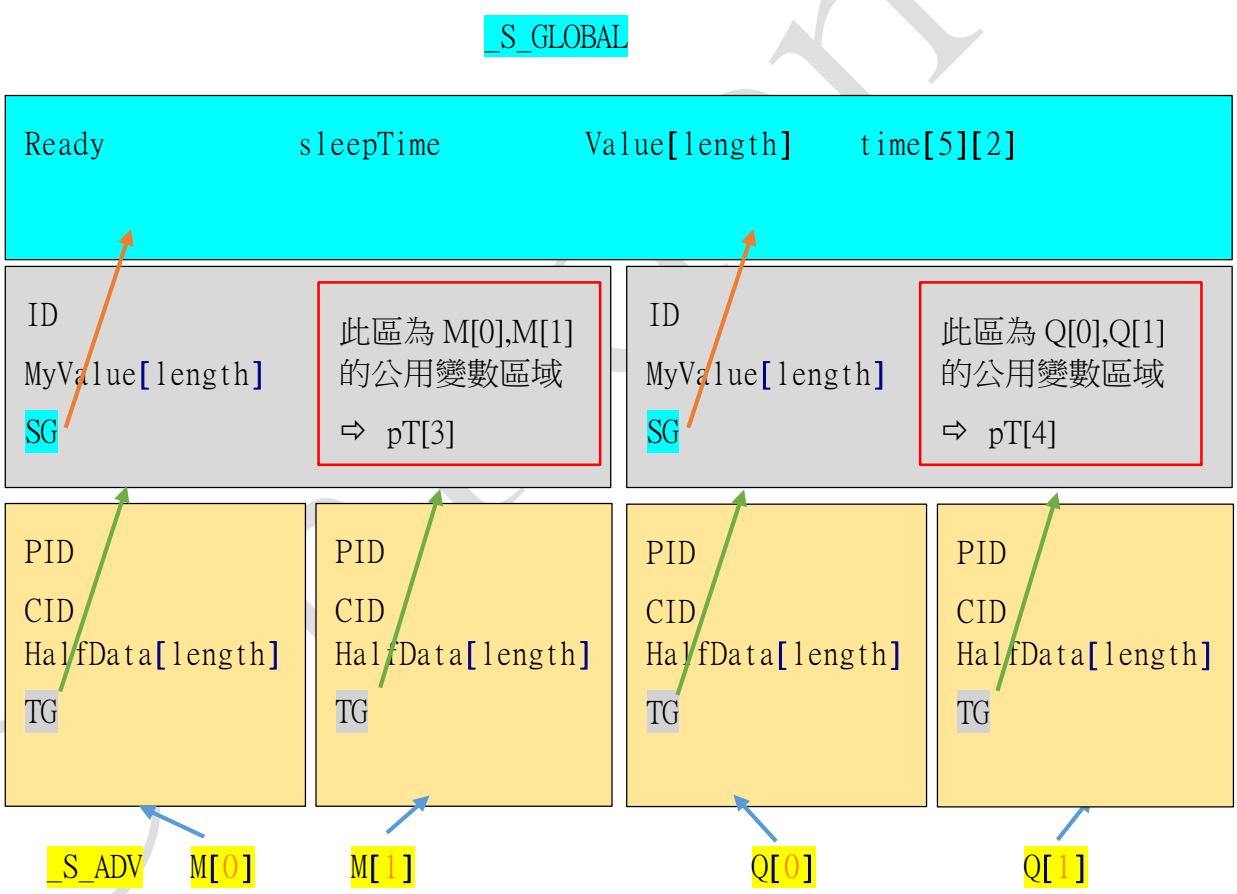
```
P_THREAD pT[5]; // 用於基本功能中
P_ADV M[2];     // 用於進階功能 - advmerge
P_ADV Q[2];     // 用於進階功能 - advquick
```

6.



7. 第六點的結構為 pT[0]~pT[4]的結構，淺藍色為共用變數，灰色為區域變數

8.

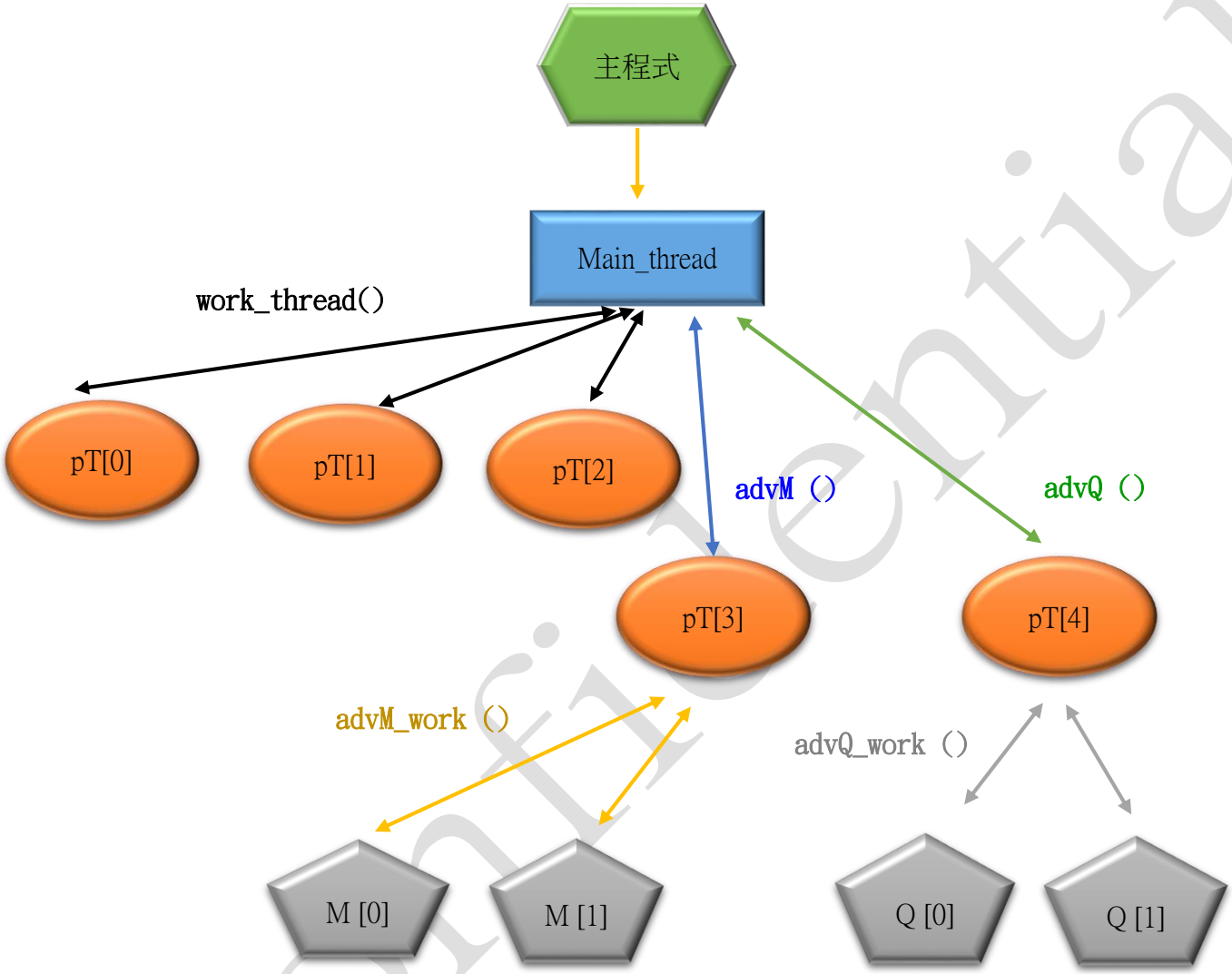


9. 第八點的結構為 M[0]、M[1]、Q[0]、Q[1] 的結構，黃色區域為進階功能中再次創造的 pThread 的區域變數，灰色區域為 M[0]、M[1]的共用變數區和 Q[0]、Q[1]的共用變數區，淺藍色為大家的共用變數區

2.2 Thread 產生示意圖

EX :

```
pthread_create (&KidThread1, NULL, (void *) &work_thread, pT[0]);
```



第3章　程式流程簡介

1. 在 main_thread 設定好結構的基本需求後，會開始 crate 五個 work_thread (pT[0]~pT[4]) ，並會在讀入資料後等待他們結束。
2. 在所有 work_thread 建立好之後，他們才會收到訊號一起往下執行。
3. pT[0]~p[2]會如 p.4 的圖一樣,呼叫 work_thread(),然後會去檢查 main_thread 讀入資料沒，若還沒，則會根據在命令列輸入的秒數睡覺及印出現在時間。
4. 再來等到 main_thread 讀好資料並改變啟動變數後(Ready),pT[0]~pT[2]會一起繼續往下執行並根據自己的 ID 去做他們要做的排序，並且輸出排序的檔案和將 (1) 回傳時間 及 (2)排序時間 放到共用變數中。
5. pT[3]是要執行 advMerge 的，他會如 p.4 的圖一樣，呼叫 advM()，並在 advM()中設定好結構的基本需求後，create 兩個 thread，並呼叫 advM_work()，advM()則會等待 advM_work()結束。
6. 在 advM_work()中，會有 M[0]和 M[1]，他們一樣會等到 main_thread 讀好資料後才一起繼續執行，然後根據自己的 CID 去決定他們的分工排序，排序完之後再回到 advM()去做最後排序並輸出檔案，以及將 (1) 回傳時間 及 (2)排序時間 放到共用變數中。
7. pT[4]是要執行 advQuick 的，他會如 p.4 的圖一樣，呼叫 advQ()，並在 advQ()中設定好結構的基本需求後，create 兩個 thread，並呼叫 advQ_work()，advQ()則會等待 advQ_work()結束。
8. 在 advQ_work()中，會有 Q[0]和 Q[1]，他們一樣會等到 main_thread 讀好資料後才一起繼續執行，然後根據自己的 CID 去決定他們的分工排序，排序完之後再回到 advQ()去輸出檔案，以及將 (1) 回傳時間 及 (2)排序時間 放到共用變數中。
9. 五個 work_thread 都執行結束後，main_thread 才可以繼續執行，他會根據 thread 們放入共用變數區的時間，去做排序，再把時間印出。

   Note: 如果時間相同，則按照回傳時間的順序印出。如果回傳時間相同，以 bubble sort, merge sort 以及 quick sort 的順序印出。

第4章　程式輸出簡介

黑字 => 會印出的訊息

藍字 => 程式的動作

紅字 => 平行執行的部分

```
Main thread : work thread for bubblesort
Main thread : work thread for mergesort
Main thread : work thread for quicksort
Main thread : work thread for advMerge
Main thread : work thread for advQuick
advMerge : work thread for child_advmerge_1
advMerge : work thread for child_advmerge_2
advQuick : work thread for child_advquick_1
advQuick : work thread for child_advquick_2
以創建的 work_thread 在等待 main_thread create 完畢
Bubblesort : [ 現在時間 ] sleep 3 second
mergesort  : [現在時間] sleep 3 second
quicksort  : [現在時間] sleep 3 second
advMerge   : [現在時間] sleep 3 second
advQuick   : [現在時間] sleep 3 second
child_advmerge_1 : [ 現在時間 ] sleep n second
child_advmerge_2 : [ 現在時間 ] sleep n second
child_advquick_1 : [ 現在時間 ] sleep n second
child_advquick_2 : [ 現在時間 ] sleep n second
所有的 work_thread 在等待 main_thread 讀入資料且根據輸入數字去睡覺
Main thread : read data
Main thread : write data to the shared area
```

Main thread : tell work thread start to work

所有 wotk_thread 開始執行，main_thread 等待他們執行結束

>> pT[0] ~ pT[2]

Work_thread : sort begin

Work_thread : sort end

Work_thread : sort 秒數

Work_thread : sort 寫入 txt

Work_thread : 將時間放到與 main_thread 的共用區域

>> pT[3] avdmerge

pT[3] : create M[0]

pT[3] : create M[1]

create 出的兩個 thread 等待 main_thread 讀入資料並更改啟動變數

M_thread : sort begin

M_thread : sort end

M_thread : 將結果放到與 pt[3]的共用區域

pT[3] : advmerge begin

pT[3] : advmerge end

pT[3] : advmerge 秒數

pT[3] : advmerge 寫入 txt

pT[3] : 將時間放到與 main_thread 的共用區域

>> pT[4] avdquick

pT[4] : create Q[0]

pT[4] : create Q[1]

create 出的兩個 thread 等待 main_thread 讀入資料並更改啟動變數

Q_thread : sort begin

Q_thread : sort end

Q_thread : 將結果放到與 pt[4]的共用區域

pT[4] : advquick 秒數

pT[4] : advquick 寫入 txt

pT[4] : 將時間放到與 main_thread 的共用區域

Main thread : 將時間排序並由小到大印出 (共有五個)

第5章 程式如何編譯

1. 有附上 Makefile 檔提供操作

   OR

2. 在命令列輸入 : gcc 1043335_02.c -lpthread -o 1043335_02
3. 編譯成功後，輸入 : ./1043335_02 number.txt 3

NOTE : number.txt 可改成要讀取的檔案名稱 && 3 可改成 1~5 任意數

第6章 如何操作

1. 將 Makefile 中 run 的 number.txt 改成要讀的檔案名稱將 number.txt 改成要讀的檔案名稱，且數字 3 改成想檢查的秒數區間
2. make main => 編譯
3. make run => 執行
4. make start => 編譯 & 執行
5. make clean => 清除編譯、執行後產生的檔案
6. make all => 清除檔案後編譯執行
7. 或是可以直接在命令列輸入 **第五章** 提供的指令