CS305 作業系統概論

Prog. #2 Multithreading說明報告

汪文豪(學號：1071710)

* **如何編譯與測試操作程式：**

1. 編譯：g++ 檔名.cpp –o 檔名 –lrt -pthread

2. 執行：./檔名 欲讀入資料檔案之檔名

補充：資料檔案內之infix必須無空格，否則會報錯

只能有一行運算式，程式只會執行第一行指令

3. 輸出結果

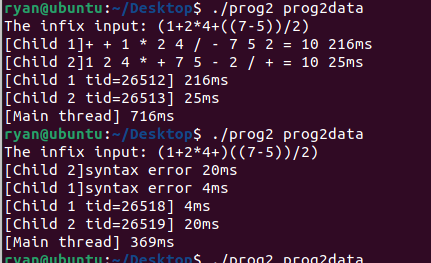
範例(資料檔案檔名為prog2data)：

一. g++ hw2.cpp –o prog2 –lrt –pthread

二. ./prog2 prog2data

三. 輸出結果

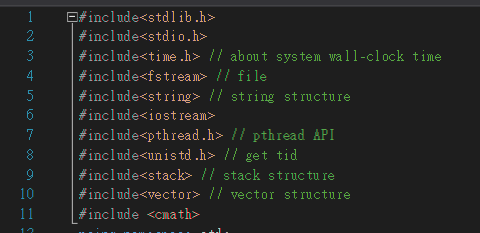




若要測試程式可直接從資料檔案內改資料，儲存後直接執行程式即可

* **設計理念：**

**本次使用到的特殊函式庫**



1.與2.是C/C++語言標準函式庫

3.time.h是計算wall-clock time

4.是為了做讀檔，會用到ifstream

7.是因應本次作業使用的pthread API

8.為了取得其thread之tid，使用的函式庫

11.是為了做四捨五入所用的

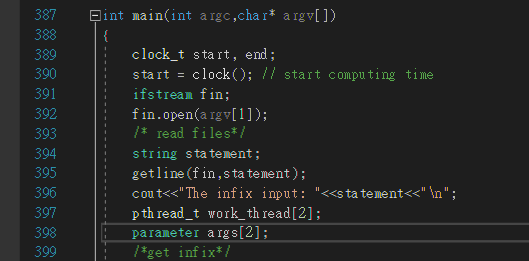
其餘皆是其作業所需的基本資料結構所用到的

**本次程式說明：**

* 完成部分：

**基礎1：主執行緒從命令列讀入檔名，並完成內容輸出。本項滿分10分**

391行至396行宣告接收資料之變數並讀入資料內容，存入statement之string結構中，將infix內容做輸出。



**基礎2：主執行緒能正確產生子執行緒。本項滿分10分。**

透過陣列產生兩個子執行緒，倘若創建不成功，406行會判斷到且將錯誤印出



**基礎3：在不考慮大數運算下，子執行緒能夠正確完成正確的 prefix 與 postfix 的轉換與運算，並印出結果。本項滿分40分。**

336行之work\_computing function即是處理pthread之運算，由於兩個thread之先後順序無法控制，有可能導致重疊輸出，故一開始先用mutex\_lock，之後做時間的計算，先取得開始執行的時間(339行)，要離開前再計算其結束時間(377-378行)並做兩者間的時間差並印出與儲存至struct，之後再解鎖並結束thread之執行。

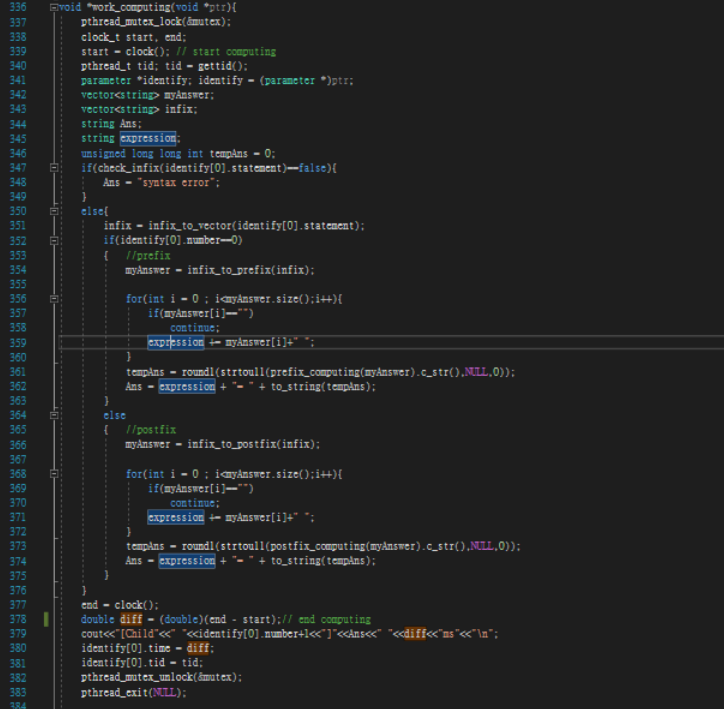
341行則是取得傳入進來之多參數struct資料

347判斷infix是否合法，若不合法就跳過運算式顯示syntax error

350行開始運算之block，首先之351行則是將string轉成vector，將operand、operator、parentness等切成一塊一塊的block

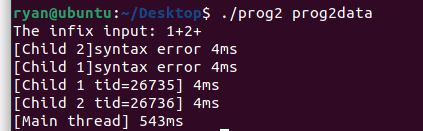
354行則是先將infix轉成prefix，之後將其表達式儲存到expression中，由於operand 和 operator要分隔，因此後面會再加一個空格分隔

Ans則是將expression與等號和使用prefix\_computing計算其結果答案，全數存至Ans中，而post與prefix一樣，只是換另一個function。另外tempAns則是因為需要四捨五入取整數，故在計算的時候是使用long double，之後使用roundl將long double四捨五入取整數，放置unsigned long long int之tempAns變數中。此外判斷是prefix or postfix全看當初放入筆者所設計之parameter strcut之number來做區分，辨別哪一個是work\_thread1或2，並進入自己的計算區塊中。最後計算出的答案統一於379行印出，並將其時間與tid儲存到其傳進來的struct之identify中，最後統一將其時間與tid傳回main thread印出。



**基礎4：在不考慮大數運算下，子執行緒能夠正確處理infix運算式中的各種錯誤。例如 “1+2+”，應輸出錯誤訊息。本項滿分20分。**

如上一題所述，在347行會做判斷，若有問題即會將syntax error放入ans並顯示：



而檢查infix之錯誤寫在第260行之check\_infix function中，由於程式碼太長故不貼上。最主要的核心概念是計算必須按照operand + operator + operand之順序排列，因此最後的數量會是operand\_number == operator\_number + 1，若否則是錯誤；過程中也會判斷是否按照operand + operator + operand之順序，否則也會是錯誤。而左括號 – 右括號數量必須衡大於0，否則代表右括號比左括號先出現，直接return false

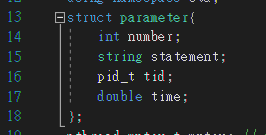
另外還有一些括號的額外判斷也都有做，但較為複雜故不在此贅述

**基礎5：主執行緒與子執行緒不透過任何IPC機制交換資料，並且正確印出子執行緒的tid與執行時間。本項滿分20分。**

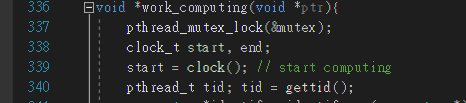
本次程式透過struct傳遞與操作其多參數，儲存time、tid和infix表達式與判斷是為work\_thread1或work\_thread2之number共四個參數

時間也都分別在子執行緒開始執行將時間存入start，即將結束時存入end，並計算之間的差後印出並將資料存入其struct paramete指標所指向的位址，之後在main thread印出

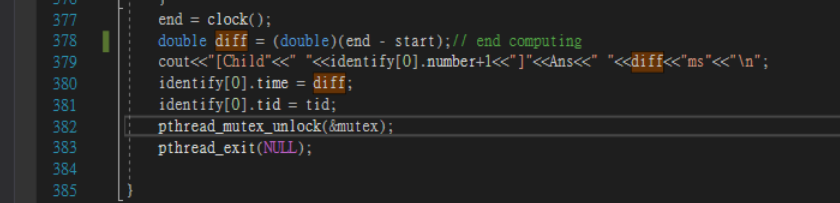
tid則是透過gettid()取得，資料交換方式相同



取得tid



取得開始時間和tid



存入parameter指標指向之位址

**其他Function用途解說：**

1. 20行之order\_list，postfix & prefix需要用到，判斷其operator優先序
2. 29行之check\_isOperator，判斷字元是否是operator
3. 34行之check\_stringOperator，因為之後會將表達式做切割，故不會是字元而是字串，為方便操作則多寫一個判斷字串是否是operator之function
4. 39行之check\_isNumber，判斷其字元是否是0 – 9範圍的數字字元
5. 45行之infix\_to\_vector，將infix切成一塊一塊的block，方便做計算
6. 80行之infix\_to\_postfix，將infix轉成postfix表達式
7. 122行之reverse\_infix，由於prefix需要將infix倒置，故寫此function
8. 130行之infix\_to\_prefix，將infix轉成prefix表達式
9. 187行之compute，因應其prefix & postfix計算，會需要用到這個function，先轉成long double計算，以利之後完整結果做四捨五入，將兩個operand計算後將結果轉成string回傳
10. 209和234行之postfix computing & prefix computing則是做其表達式的運算