作業系統

HW1

實作Multi Threads和multi Processes

班級：資訊三甲

學號：10627116

姓名：許逸翔

老師：鍾斌賢

1. 開發平台
   1. 作業系統：Windows10教育版 64bit
   2. CPU：i5-7200U 2.5GHz 四核心
   3. RAM：8GB DDR4-2134 雙通道
2. 使用開發環境
   1. 虛擬環境
      1. 系統：CentOs7 1804版 64bit
      2. CPU：雙核心
      3. RAM：4GB
   2. IDE：Visual Studio
   3. Language：C++
   4. GDB：x86\_64-redhat-linux-gn (-std=c++11)
3. 完成的功能
   1. 能完成任意數量資料的排序，資料必須是int的表示的數值
   2. 有多重Threads和Processes選項，可以指定啟動的數量
   3. 排序方式
      1. Bubble Sort with a Process
      2. Bubble Sort with multi Threads & Merge Sort with multi Threads
      3. Bubble Sort with multi Processes & Merge Sort with multi Processes
      4. Bubble Sort split k part with a Process & Merge Sort with a Process
   4. 可以自動檢查數值排列是否正確
   5. 能輸出程式執行的CPU Time和Turnaround Time
4. 流程
   1. Main
      1. 使用者將給定的資料準備
      2. 程式依據文件敘述的選擇排序模式
      3. 讀取資料、進行排序
   2. Multi Threads
      1. 將function作為參數傳入，建立新的Thread
      2. 搭配ref(data)使得Threads之間共用相同的資料
      3. 運用Thread.join()來等待其他Thread的完成
   3. Multi Processes
      1. 使用fork將程式複製，根據pid來判斷子程式與父程式
      2. 搭配shmget取得share memory，再用shmat將指標指向真實的存放空間，即可讀寫共用的資料
      3. 運用wait()來等待子Process的結束
5. 使用到的函式庫
   1. 資料結構：使用<vector>存資料，因為不確定大小
   2. 使用<time.h>來計時
   3. 使用<fstream>來完成I/O
   4. 使用<random>來產生亂資料
   5. 使用<thread>來控制執行序
   6. 使用<unistd.h>來控制程序
   7. 使用<sys/shm.h>來完成IPC(share memory)
6. 排序方式的分析與比較(詳細的數據資料請參考附件A)
   1. Bubble Sort with a Process(藍線) 速度是最慢的，時間複雜度大約是，相較於Merge Sort的 慢的許多
   2. 沒有想到將Bubble Sort切塊再與Merge Sort並用能有這麼好的效果(黃線)，其實Multi Threads(橘線) 或Multi Processes(灰線)相差不大，可見Threads和Processes做平行化的效果並不顯著
   3. Multi Processes(灰線) 的速度是快於Multi Threads(橘線)，因為兩者沒有浪費資料複製時間，都是的對記憶體作操作，前者是用Share Memory，後者是ref傳參考，所以在資料傳輸上的出發點相同，差別只有在CPU的使用，Multi Processes能佔有CPU的時間比Multi Threads(One Processes)多，速度上大概快了一倍

* 1. 當使用資料量不切分時，速度大約界在0.5~1倍的Bubble Sort之間
  2. Multi Threads(藍線) 雖然是light-weighted process，不需要context switch但過使用過多的Threads，會使得每個Thread能用的時間減少，所以使用太多的Threads會造成反效果
  3. Multi Processes(橘線) 也不適合使用太大的數量，即使每個Process都能使用一個time slice，所以效果較Multi Threads好，但仍需要context switch的時間，所以使用太多的Processes會造成反效果
  4. 灰線之所以略輸橘線是因為採用k-way Merge Sort，所以在比較的時候需要歷遍較多的資料，因此在速度上小輸一節
  5. 可以選擇資料切分數量大小的重要性，如果太小會因為Bubble Sort導致速度大幅降低，如果太大又會造成Merge Sort效果不佳，因此大約使用「資料量開根號後的數值」作為「多執行序/多程序的數量」是最佳的選擇