由於標頭檔的關係無法用Windows編譯執行!!!

我會自行攜帶環境去機側!!!感激不盡

然後因為ubantu與window文字上轉換時可能產生亂碼，希望助教允許更改，非常感謝

●**開發環境:**

我使用的是C++語言，因為考量到某些標頭檔在Windows中無法使用，像是#include <sys/syscall.h>，因此我改成利用Linux執行，所以下載了Ubantu，並且安裝了g++，還有Code Block編譯器，但在上機執行時我會使用終端機來執行。

●**實作方法和流程:**

**→ 建立process和thread的方法:**

建立process的過程中，我宣告一個vector來存k個子process，接著用vfork()這個函式來生出子process，然後每個子process會進行泡沫排序，做完以後，該子process結束，並且存進vector，然後父process會等待所有子process完成，用的是wait()函式。

建立thread的過程中，要使用 C++ 的標準庫 std::thread 建立多個thread。我宣告一個vector來存k個子threads。接下來使用 for 迴圈建立 k 個threads緒，再用 for 迴圈讓每個執行緒依序調用 join() 函數，等待所有執行緒完成排序。最後再將k-1個threads合併排序

**→ 流程說明:**

我用一個public vector aList存入所有讀檔的資料，用到字串atoi成int，

方法1:純粹的bubble sort，start = 0，end = aList.size()

方法2:切成k份分別bubble sort再一起merge sort，這一題我的bubble寫的超長，因為寫這題是沒想到可以呼叫之前寫過的bubble，反正就是有一個Ksize他會是每個k份的最後一個數字的陣列位置，假設輸入100筆資料，切成5份，我會變成1-20、21-40、41-60、61-80、80-100，依序進入else進行bubble sort，然後如果有餘數，最後一組我會讓他進if，那裏也是做bubble sort，只是條件是Ksize= aList.size()+1。然後再利用Merge sort，Merge sort是一個遞迴形式，會呼叫自己跟Merge，而Merge裡面會有芬左邊陣列跟右邊陣列，比較左右兩部分中的元素大小，來決定哪個元素先放入合併後的結果中，然後有多用兩個while處理剩餘元素，避免出現輸出都是零

方法3:

我的子程序是用vfork()，一開始用fork()還要寫一堆東西處理共享記憶體，結果用vfork()後，基本上泡沫跟合併排序就和第二題差不多，總之就是宣告一個vector來存k個子process，接著用vfork()這個函式來生出子process，然後每個子process會進行泡沫排序，做完以後，該子process結束，並且存進vector，然後父process會等待所有子process完成，用的是wait()函式。然後這些是每組自己做排序，接下來進入k-1個process合併排序，兩兩合併排序一次，i在2的次方時pid 是父進程的 pid，因此 pid 的值會是非零的，會進入 else 區塊，其他時候進入if，做合併排序。

方法4:

建立thread的過程中，要使用 C++ 的標準庫 std::thread 建立多個thread。我宣告一個vector來存k個子threads。接下來使用 for 迴圈建立 k 個threads緒，再用 for 迴圈讓每個執行緒依序調用 join() 函數，等待所有threads完成排序。inplace\_merge 函數將排序後的向量合併，用兩個 for 迴圈，分別從 i=1 和 i=2 的位置開始，每次合併 i\*step 個元素。std::inplace\_merge 這個函數會合併已排序的兩組，然後再將k-1個threads合併排序，至於泡沫跟合併排序的做法都跟上面的做法差不多。

●**探討結果與原因:**

相同K值，不同資料筆數(N)的執行時間，如表1

相同資料筆數(N = 1W)，不同K值的執行時間，如表2

相同資料筆數(N = 10W)，不同K值的執行時間，如表3

表1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K = {5} | N=1w | N=10w | N=50W | N=100w |
| 方法一 | 3.25963 | 384.105 | 14832.428 | 189923.9981 |
| 方法二 | 0.599317 | 58.7313 | 1682.9 | 13928.32 |
| 方法三 | 0.01882 | 0.05735 | 0.36038 | 0.61053 |
| 方法四 | 0.313894 | 26.6114 | 898.892 | 5234.1512 |

由圖表中可發現，泡沫排序法最耗時間，他的時間複雜度相當大，而方法三CPU速度最快，因為他是多個process並行排序，可節省很多時間

表2:

|  |  |  |
| --- | --- | --- |
| N={1W} | K = 5 | K = 7 |
| 方法一 | 3.25963 | 3.38786 |
| 方法二 | 0.599317 | 0.491677 |
| 方法三 | 0.01882 | 0.02577 |
| 方法四 | 0.313894 | 0.214543 |

可發現除了做法1，其他三者皆為k越大執行CPU越小，因為做法1不需要用到切K份，所以基本上差距只是誤差值，而其他三者則是因為若切的份數越多，則一份要做的泡沫排序法的時間越短，主要變成是合併排序，而泡沫排序若減少筆數便會大幅減少執行時間，因為泡沫排序法的時間複雜度相當大

表3:

|  |  |  |
| --- | --- | --- |
| N={10W} | K = 5 | K = 7 |
| 方法一 | 384.105 | 400.913 |
| 方法二 | 58.7313 | 45.4191 |
| 方法三 | 0.05735 | 0.08639 |
| 方法四 | 26.6114 | 19.5201 |

結果的解釋與1萬筆時大致相同，只是執行時間都變長