1093304 作業系統_hw2 說明報告

設計理念：

```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <sys/types.h>
#include <pthread.h>
#include <cmath>
#include <vector>
#include <string>
#include <map>
using namespace std;

vector<string> id, document;    //文件的ID、存放各個文件句子
map<string, vector<int>> word; //存每個句子中的單字，key存單字、val存key在每個句子中所出現的次數
vector<double> avg;            //每組詞頻向量的平均

string only_word(string &sentence) //刪除句子中的標點符號
{
    if (sentence.back() == '\r') //若該句結尾有\r則將其刪除
    {
        sentence.pop_back();
    }

    for (size_t i = 0; i < sentence.size(); i++) //若句子中有標點符號則將其改為空格
    {
        if (!isalnum(sentence[i]))
        {
            sentence[i] = ' ';
        }
    }

    return sentence;
}

bool only_letter(const string &word) //回傳當前的單字是否只用英文字母組成
{
    for (size_t i = 0; i < word.size(); i++)
    {
        if (!isalpha(word[i])) //若當前單字中的字元不是英文字母
        {
            return false; //則回傳false
        }
    }

    return true; //單字確定只由英文字母組成，回傳true
}
```

第 13～15 行：建立存文件 id 和各文件句子的 vector，map 存每個單字在每個文件中出現的次數、avg 存每個文件和其他所有文件所算出的 Cos 平均。

第 17～33 行：對文件做初步處理，主要是將標點符號和\r 去掉。

第 35～46 行：檢查當前擷取的單字是否只用英文字母組成。

```
48  void *avg_cosine(void *input)
49  {
50      size_t cur = (size_t)input;                             //將輸入進來的參數轉成size_t，current == 當前的詞頻向量
51      pair<timespec, timespec> CPUtime;                       //計算CPU時間，pair中的first和second分別代表start和end
52      clock_gettime(CLOCK_THREAD_CPUTIME_ID, &CPUtime.first); //紀錄start的CPU時間
53      unsigned long int tid = pthread_self();                 //紀錄當前的tid
54      cout << "[TID=" << tid << "] DocID:" << id[cur] << " [";
55      for (auto it = word.begin(); it != word.end(); it++) //印出詞頻向量
56      {
57          cout << (it == word.begin() ? "" : ",") << it->second[cur];
58      }
59
60      cout << "]\n";
61      for (size_t tar = 0; tar < document.size(); tar++) // target == 對象的詞頻向量
62      {
63          if (cur != tar) //若2詞頻向量不相同(不能自己和自己做運算)
64          {
65              double curSum = 0, tarSum = 0, cur_x_tar_Sum = 0, result; // 2向量的平方總和、2向量的相乘總和、2向量的最終Cos結果
66              for (auto it = word.begin(); it != word.end(); it++)      //計算所有的Cos值
67              {
68                  curSum += it->second[cur] * it->second[cur];
69                  tarSum += it->second[tar] * it->second[tar];
70                  cur_x_tar_Sum += it->second[cur] * it->second[tar];
71              }
72
73              avg[cur] += (result = cur_x_tar_Sum / (sqrt(curSum) * sqrt(tarSum)));                                   //所有Cos的值加總
74              cout << fixed << setprecision(4) << "[TID=" << tid << "] cosine(" << id[cur] << "," << id[tar] << ")=" << result << "\n"; //印出當前的Cos值
75          }
76      }
77
78      cout << "[TID=" << tid << "] Avg_cosine: " << (avg[cur] /= document.size() - 1) << "\n"; //計算並印出平均餘弦相似係數
79      clock_gettime(CLOCK_THREAD_CPUTIME_ID, &CPUtime.second);                                 //紀錄end的CPU時間
80      //印出的時間差即CPU時間，由於tv_nsec == 奈米秒，故除以CLOCKS_PER_SEC(在linux環境中等於1000000)則等於毫秒(ms)
81      cout << "[TID=" << tid << "] CPU time: " << double(CPUtime.second.tv_nsec - CPUtime.first.tv_nsec) / CLOCKS_PER_SEC << "ms\n";
82      return NULL;
83  }
```

第 48～83 行：計算平均 cosine 的函式，其中第 52 行和 79 行是算出當下的 CPU 時間，最後在第 81 行印出 2 者的時間差就是花費的 CPU 時間。

```
85  int main(int argc, char *argv[])
86  {
87      pair<timespec, timespec> main_CPUtime;                    // main thread的CPU時間
88      clock_gettime(CLOCK_THREAD_CPUTIME_ID, &main_CPUtime.first); //紀錄start的CPU時間
89      if (argc != 2) //從命令列讀入檔名參數，若輸入的參數數量不符則印出錯誤訊息並結束
90      {
91          cout << "Input error\n";
92          pthread_exit(0);
93      }
94
95      ifstream inFile(argv[1]); //讀檔
96      if (!inFile)              //若沒讀到檔
97      {
98          cout << "File could not be opened\n";
99          pthread_exit(0);
100     }
101
102     for (string input; !inFile.eof();) //若還未到檔案結尾
103     {
104         getline(inFile, input);         //一行一行地從inFile讀入input
105         id.push_back(only_word(input)); //將讀取到的ID記錄到id中
106         getline(inFile, input);
107         document.push_back(only_word(input)); //將讀取到的句子記錄到document中
108     }
109
110     inFile.close(); //讀檔關閉
111     for (size_t i = 0; i < document.size(); i++)
112     {
113         stringstream getWord(document[i]);              //擷取句子中的單字
114         for (string input; getline(getWord, input, ' ');) //若能擷取到單字
115         {
116             if (only_letter(input) && input != "") //若當前單字僅由字母組成，且該單字 != 空字串
117             {
118                 auto it = word.insert({input, vector<int>(document.size())}); //則將該單字加入map中，vector空間 == 句子數，存該單字在對應的句子中出現的次數
119                 it.second ? it.first->second[i] = 1 : word[input][i]++; //若it.second == true，代表此單字是新增進來的，則在對應的句子index中放入1；否則代表該單字之前已存在過，則在對應的句子index中 + 1
120             }
121         }
122     }
```

第 88 行：紀錄 main thread 當下的 CPU 時間。

第 89～100 行：若輸入或讀檔錯誤，則印出錯誤訊息並結束程式。

第 102～108 行：將讀進來的 ID 和文件存到對應的 vector 中。

第 111～122 行：擷取每份文件中的單字，若單字只由字母組成則放進 map 中，其中在 118 行，insert 回傳的值是一個 pair<iterator, bool>，bool 表示當前的 insert 是否成功。若 bool 值等於 true，則代表原本的 map 沒有這個單字，所以 insert 成功，map 內的元素會 +1，並且 iterator 指向該元素；反之則代表欲 insert 的 key 已存在 map 中，所以 insert 失敗，其 iterator 指向指定的 key 的元素。

```
124     vector<pthread_t> multithread(document.size()); //有幾份文件句子就創建幾個thread
125     avg.assign(document.size(), double());          //每個句子都會和其他句子算出Cos值並最後做平均，故存放平均數的空間 == 文件數
126     for (size_t i = 0; i < multithread.size(); i++) //建立多執行緒
127     {
128         pthread_create(&multithread[i], NULL, avg_cosine, (void *)i);
129         cout << "[Main thread]: create TID: " << multithread[i] << ", DocID: " << id[i] << "\n";
130     }
131
132     for (size_t i = 0; i < multithread.size(); i++)
133     {
134         pthread_join(multithread[i], NULL); //等待執行緒完成工作
135     }
136
137     double avg_max = 0, index;              //所有平均中的最大值就是關鍵文件、index == 關鍵文件所在的位置
138     for (size_t i = 0; i < avg.size(); i++) //找出最大值並記錄該值所在的位置
139     {
140         avg_max = max(avg_max, avg[index = i]);
141     }
142
143     cout << "[Main thread] KeyDocID: " << id[index] << " Highest Average Cosine: " << avg_max << "\n"; //印出關鍵文件的ID及其值
144     clock_gettime(CLOCK_THREAD_CPUTIME_ID, &main_CPUtime.second);                                      //紀錄end的CPU時間
145     cout << "[Main thread] CPU time: " << double(main_CPUtime.second.tv_nsec - main_CPUtime.first.tv_nsec) / CLOCKS_PER_SEC << "ms\n"; //印出main thread的CPU時間
146     pthread_exit(0); //執行緒終止
147     return 0;
148 }
```

第 124 行：根據文件數創建 thread。

第 126～130 行：建立多執行緒。

第 132～135 行：等待執行緒完成工作。

第 137～141 行：找出所有平均中的最大值。

第 144 行：紀錄 main thread 當下的 CPU 時間。

第 145 行：2 時間相減即為 main thread 的 CPU 時間。

執行畫面：

```
peko@peko-VirtualBox:~$ g++ -o 1093304 1093304.cpp -lpthread
peko@peko-VirtualBox:~$ ./1093304 data.txt
[Main thread]: create TID: 140058892826176, DocID: 0001
[Main thread]: create TID: 140058884433472, DocID: 0002
[Main thread]: create TID: 140058876040768, DocID: 0003
[Main thread]: create TID: 140058867648064, DocID: 0004
[TID=140058884433472] DocID:0002 [0,2,0,0,1,0,1,0,0,2,0,2,0,1,0,0,0,1,1,1,1]
[TID=140058884433472] cosine(0002,0001)=0.5869
[TID=140058884433472] cosine(0002,0003)=0.5735
[TID=140058884433472] cosine(0002,0004)=0.6489
[TID=140058884433472] Avg_cosine: 0.6031
[TID=140058884433472] CPU time: 0.0454ms
[TID=140058892826176] DocID:0001 [1,0,0,0,2,1,0,1,0,1,0,1,0,0,0,1,0,3,1,1,1]
[TID=140058892826176] cosine(0001,0002)=0.5869
[TID=140058892826176] cosine(0001,0003)=0.4264
[TID=140058892826176] cosine(0001,0004)=0.4523
[TID=140058892826176] Avg_cosine: 0.4885
[TID=140058892826176] CPU time: 0.0312ms
[TID=140058867648064] DocID:0004 [0,1,0,1,1,0,1,0,0,1,0,1,2,1,2,0,0,1,0,1,1]
[TID=140058867648064] cosine(0004,0001)=0.4523
[TID=140058867648064] cosine(0004,0002)=0.6489
[TID=140058867648064] cosine(0004,0003)=0.5893
[TID=140058867648064] Avg_cosine: 0.5635
[TID=140058867648064] CPU time: 0.0634ms
[TID=140058876040768] DocID:0003 [0,1,2,0,1,0,1,0,1,0,1,1,0,1,1,0,1,1,0,1,1]
[TID=140058876040768] cosine(0003,0001)=0.4264
[TID=140058876040768] cosine(0003,0002)=0.5735
[TID=140058876040768] cosine(0003,0004)=0.5893
[TID=140058876040768] Avg_cosine: 0.5297
[TID=140058876040768] CPU time: 0.0388ms
[Main thread] KeyDocID: 0004 Highest Average Cosine: 0.6031
[Main thread] CPU time: 0.2844ms
```

開啟終端機，輸入 g++ -o 1093304 1093304.cpp -lpthread，再輸入./1093304 +讀檔文件名稱(如圖中的 data.txt)後便可執行程式。