HW1 Pthread Programming

以下有些重要事項需要同學注意

1. 關於input 參數 請讓你的程式能夠接受兩個命令行參數 第一個是 環境cpu 數目 第二個是 執行的次數 eg: ./pi 4 10000
2. 若你測試scalability 為了讓你的程式可以在指定的Cpu數目下執行 可以使用taskset -c [a numerical list of processors] command eg: taskset -c 1,2 ./pi 2 100000 綁定程式在第1,2 core上執行

"Your program should be scalable."，意思是我們測試的時候會讓同學們的程式分別跑在1 core, 2 cores, 4 cores的不同環境下，不要求一定要達到2倍或4倍的speedup，但在多核心的環境下仍需有較好的 performance，此項目也會列入評分。

1. 關於output結果 請大家直接輸出 pi\_estimate 結果, eg: 3.1415926

TIPS: pi\_estimate 在執行1000萬次後大概會準確到 3.14XXX。

1. 關於ToolChain 我們會用以下指令 compile 同學們的程式碼:

C++: g++ -pthread -std=c++11 -O2 -s pi.cpp -o pi

或

C: gcc -pthread -std=gnu99 -O2 -s pi.c -o pi

gcc 版本爲 5.4.0

**※以上為作業說明，接下來為助教提醒，請仔細閱讀※  
※不接受任何事後補救辦法，請務必檢查清楚※**

**上傳作業前，請務必下載附加的Makefile檔案，並且放於有pi.c的資料夾下**

* **執行make編譯檔案**
* **執行make test確認執行情況**

**如果無法通過make或是make test表示你的檔案或是input格式有問題直接以0分計算**

**並且檢查作業格式**

* **本次作業繳交時，你可以自己準備Makefile，但是輸出的檔案一定要是pi**
* **且壓縮為一個zip檔案**

**不符合作業繳交格式，也直接以0分計算**

**評分的標準的細項會儘快公佈，在公佈標準前繳交的作業皆會先評分一次，提供參考成績**

T.A.

**作業一評分標準**

**2019年 03月 26日(Tue) 07:22**

正確性 70分  
在1億次的計算之後，四捨五入的數值為3.14(3.145~3.135)  
同時必須在3秒之內完成計算。

以1,2,4核心數，各跑7次共21組，每組3分，完成19組以上額外加10分，超過70以70分計算  
（允許運氣不好兩次，但是實作正確失敗的可能性很低）  
失敗8組以上，零分計算。

Scalability 15分  
同樣計算1億次，並循環5次後取平均

#1 2核心比單核心快1.4倍以上  
#2 4核心比單核心快2倍以上  
#3 4核心比2核心快1.4倍以上  
#4  2核心的單位處理器執行指令數為單核心的 0.6倍  
#5  4核心的單位處理器執行指令數為單核心的 0.4倍  
  
#4，#5是倚賴perf紀錄每個核心的執行指令數，你必須將工作正確的分工給不同的核心  
上面的測資一個3分共15分  
  
在37106的機器上，且已經編譯好名為pi執行檔的資料夾下，可以直接下score  
他會執行評分腳本，並顯示測試的紀錄。  
  
或是你可以執行 P3 <your command>來測試你的程式是否正確，他會提供一些額外的訊息  
其中，CPI的訊息值得參考，一般來說核心數量不會影響CPI，然而，如果你的CPI隨著核心數有劇烈變化，那就是存在Bug。

範例：

Input:  
P3 ./pi 4 1000000

Output：

['/usr/local/bin/evaluate', './pi', '4', '100000000']

Avg: 0.238002022 sec

Core 0 :

Cycles: 550088765

Instructions: 476065882

Branch-miss rate: 0.0002126145077459915

CPI 1.1554887375861143

Core 1 :

Cycles: 550103786

Instructions: 475504684

Branch-miss rate: 0.0002069800938936609

CPI 1.1568840529024105

Core 2 :

Cycles: 550425096

Instructions: 476589570

Branch-miss rate: 0.0007507930985338224

CPI 1.1549247626212213

Core 3 :

Cycles: 550086960

Instructions: 475379612

Branch-miss rate: 0.00022442225428026522

CPI 1.1571530333109867

剩下的15分，是依照最後的排名，這裡的範例提供助教的實作參考  
這並不是最快的參考，而是一個標準的參考速度，如果你比這個速度慢太多，你可能需要想辦法再優化一下。  
此外，已經繳交作業的三位同學，我先替各位測試了一次成績，如果你覺得不滿意，可以重新上傳  
會再測一次。  
  
成績列表在這裡：  
<https://docs.google.com/spreadsheets/d/1I3Uh-nj-3dCMNtjBjrnlHz4LasbB1Yo-q6KX38PyPAA/edit?usp=sharing>

**0656818**

請問我們需要改Makefile 去使用gcc-6 還是server會統一用gcc-6

"/usr/bin/gcc" 現在是連到gcc-5

**ycsu**

不用，助教這邊會優先用gcc-6編譯，除非失敗才會用你提供的Makefile

**yuchen**

@ycsu 助教,請問 Scalability 的單位處理器執行指令數 是怎麼算的?

#4 2核心的單位處理器執行指令數為單核心的 0.6倍

#5 4核心的單位處理器執行指令數為單核心的 0.4倍

2核心的單位處理器執行指令數 = (Core 0 指令數 + Core 1 指令數)/2??? 4核心的單位處理器執行指令數 = (Core 0 指令數 + Core 1 指令數 + Core 2 指令數 + Core 3 指令數)/4???

(被編輯過)

**ycsu**

恩....這個目的是希望你有確實將工作分配在不同的核心上 我之前比較懶，所以score的腳本都是以Core 0的指令數來計算。 因為按常理設計，指令數應該平均分配到每個核心上。 感謝你提醒我，所以我會調整成測試，所有使用核心都要符合標準 如果標準太嚴，可能還會再調整

---------------------

Q: drand48

A: https://www.twblogs.net/a/5bca37f02b7177735197319b/

參考： <http://www.man7.org/linux/man-pages/man3/drand48.3.html>. drand48 返回服從均勻分佈的·[0.0, 1.0) 之間的 double 型隨機數。These functions generate pseudo-random numbers using the linear congruential algorithm and 48-bit integer arithmetic.

The drand48() and erand48() functions return nonnegative double-precision floating-point values uniformly distributed over the interval [0.0, 1.0). The **srand48**(), **seed48**() and **lcong48**() functions are initialization functions, one of which should be called before using **drand48**(), **lrand48**() or **mrand48**().

注意,對隨機數取模是個不好的設計,因為取模得到序列的可能不夠均勻。

一、rand()

rand()函数用来产生随机数，但是，rand()的内部实现是用线性同余法实现的，是伪随机数，由于周期较长，因此在一定范围内可以看成是随机的。

#include<stdlib.h>

main()

{

int i,j;

for(i=0;i<10;i++)

{

j=1+(int)(10.0\*rand()/(RAND\_MAX+1.0));

printf("%d ",j);

}

}

   执行：

9 4 8 8 10 2 4 8 3 6

9 4 8 8 10 2 4 8 3 6 //再次执行仍然产生相同的随机数

srand（设置随机数种子）

表头文件：#include<stdlib.h>

定义函数：void srand (unsigned int seed);

函数说明：

srand()用来设置rand()产生随机数时的随机数种子。参数seed必须是个整数，通常可以利用geypid()或time(0)的返回值来当做seed。如果每次seed都设相同值，rand()所产生的随机数值每次就会一样。

范例

/\* 产生介于1 到10 间的随机数值，此范例与执行结果可与rand（）参照\*/

#include<time.h>

#include<stdlib.h>

main()

{

int i,j;

srand((int)time(0));

for(i=0;i<10;i++)

{

j=1+(int)(10.0\*rand()/(RAND\_MAX+1.0));

printf(" %d ",j);

}

}

     执行：与rand范例比较

5 8 8 8 10 2 10 8 9 9

2 9 7 4 10 3 2 10 8 7

又或：

用"int x = rand() % 100;"来生成 0 到 100 之间的随机数这种方法是不或取的，比较好的做法是： j=(int)(ｎ\*rand()/(RAND\_MAX+1.0))产生一个0到ｎ之间的随机数

int main(void)

{

   int i;

   time\_t t;

   srand((unsigned) time(&t));

   printf("Ten random numbers from 0 to 99/n/n");

   for(i=0; i<10; i++)

       printf("%d/n", rand() % 100);

   return 0;

}

除以上所说的之外，补充一点就是srand这个函数一定要放在循环外面或者是循环调用的外面，否则的话得到的是相同的数字。

MSDN中的例子。

// crt\_rand.c

// This program seeds the random-number generator

// with the time, then displays 10 random integers.

//

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

int main( void )

{

   int i;

   // Seed the random-number generator with current time so that

   // the numbers will be different every time we run.

   //

   srand( (unsigned)time( NULL ) );

   // Display 10 numbers.

   for( i = 0;   i < 10;i++ )

      printf( "  %6d/n", rand() );

  printf("/n");

  // Usually, you will want to generate a number in a specific range,

  // such as 0 to 100, like this:

  {

     int RANGE\_MIN = 0;

     int RANGE\_MAX = 100;

     for (i = 0;    i < 10; i++ )

      {

         int rand100 = (((double) rand() /

                        (double) RAND\_MAX) \* RANGE\_MAX + RANGE\_MIN);

         printf( "  %6d/n", rand100);

      }

  }

总结：

我们知道rand()函数可以用来产生随机数，但是这不是真真意义上的随机数，是一个伪随机数，是根据一个数，我们可以称它为种了，为基准以某个递推公式推算出来的一系数，当这系列数很大的时候，就符合正态公布，从而相当于产生了随机数，但这不是真正的随机数，当计算机正常开机后，这个种子的值是定了的，除非你破坏了系统，为了改变这个种子的值，C提供了 srand()函数，它的原形是void srand( int a) 功能是

初始化随机产生器既rand()函数的初始值，即使把种子的值改成a; 从这你可以看到通过sand()函数，我们是可以产生可以预见的随机序列，

那我们如何才能产生不可预见的随机序列呢？我们可能常常需要这样的随机序列，是吧。利用srand((unsign)(time(NULL))是一种方法，因为每一次运行程序的时间是不同的，对了，你知道time() 函数的功能是返回从1970/01/01到现在的秒数的吧，可能这个起始时间不正确，你查一下对不对吧，C还提供了另一个更方便的函数，randomize()

原形是void randomize()，功能是用来始初rand() 的种子的初始值，而且该值是不确定的，它相当于srand((unsign)(time(NULL)) 不过应注意的是randomize()的功能要通过time来实现所以在调用它时头文件要包含time.h罢了

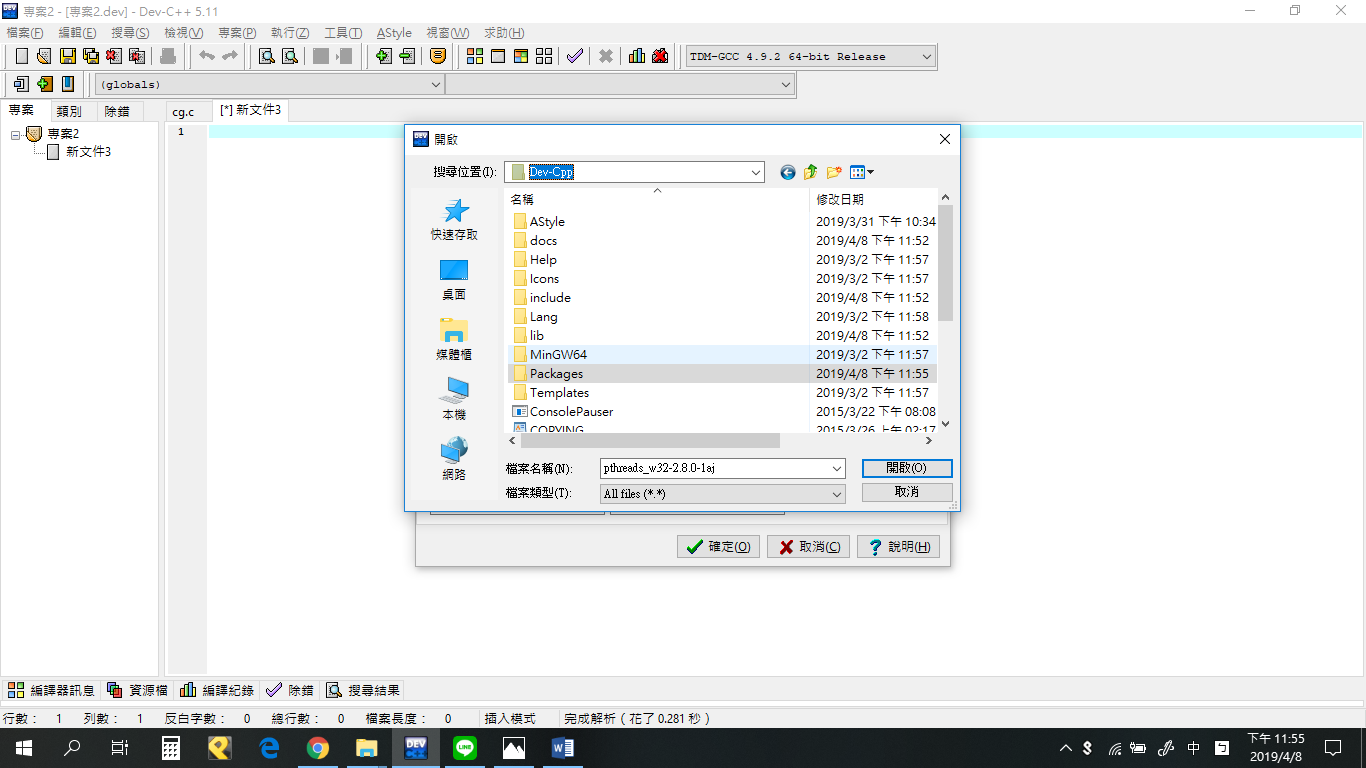
---------------------

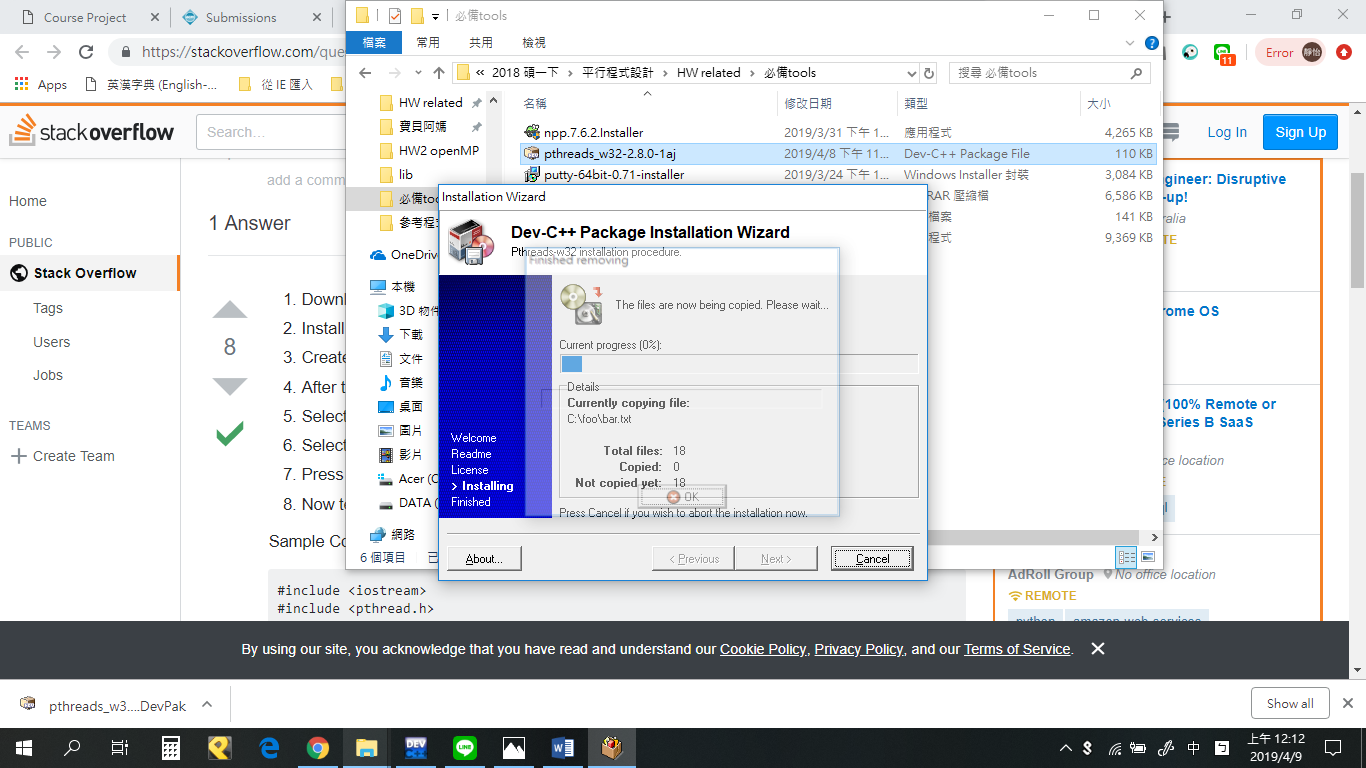
作者：xiaoxiongli

来源：CSDN

原文：https://blog.csdn.net/xiaoxiongli/article/details/1667215

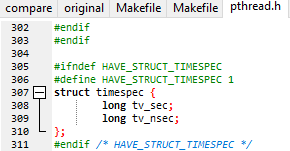
版权声明：本文为博主原创文章，转载请附上博文链接！





<https://sourceware.org/pthreads-win32/>

<https://blog.csdn.net/whu_piece/article/details/6787233>



#define HAVE\_STRUCT\_TIMESPEC //https://bbs.csdn.net/topics/391910387

