# 2017 Fall, CSCI 3150 – Assignment 2

The Chinese University of Hong Kong

Prepared by: Pengfei Zhang

Validated by: Xuxuan Zhou

Graded by: Xuxuan Zhou

Recommended Deadline: 13 Nov 2017

## Contents

# 1 Introduction

In this assignment, you have to implement a part of a "permissible" scheduler that returns a set of "permissible" jobs from a large batch of jobs submitted to an operating system. A permissible scheduler aims to execute only jobs that are meaningful to the system (because of limited resources). You **must** use PThread/OpenMP to leverage multiple cores to return the answer as quick as possible.

## 1.1 Permissible Point

A task/job/process is modeled as a data point. Given a multi-dimension dataset, a set of points are regarded as the permissible points if they are **not prevailed** by any other points. Under the assumption that smaller values are better, a point $p$ **prevails** another point $q$ if (i) all dimensions of $p$ are smaller than or equal to their corresponding dimensions of $q$ and (ii) there exists at least one dimension of $p$ that is strictly smaller than its corresponding dimension of $q$. Thus, **the set of permissible points consists of those points that are no worse than any other points when all dimensions are considered together**. For example, given the following 3-dimension (excluding `ID`) dataset:

| ID | dimension 1 | dimension 2 | dimension 3 |
|---|---|---|---|
| 1 | 0.1536 | 0.5631 | 0.7853 |
| 2 | 0.2534 | 0.5315 | 0.4735 |
| 3 | 0.1235 | 0.3436 | 0.9813 |
| 4 | 0.5343 | 0.5674 | 0.8753 |
| 5 | 0.6437 | 0.7832 | 0.1235 |
| 6 | 0.4536 | 0.3134 | 0.3342 |
| 7 | 0.6582 | 0.3253 | 0.9231 |
| 8 | 0.4536 | 0.3134 | 0.4353 |
| 9 | 0.1536 | 0.5631 | 0.7853 |

- Point *1* prevails point *4* because all dimensions of point *1* are smaller than their corresponding dimensions of point *4*.

- Point *6* prevails point *8* as `dimension 3` of point *6* is smaller than corresponding dimension of point *8*, although their other two dimensions have the same values.
- Point *1* and point *2* cannot prevail each other, because `dimension 1` of point *1* is smaller than corresponding dimension of point *2*, but `dimension 2` and `dimension 3` of point *1* are greater than corresponding dimensions of point *2* and vice versa. Point *1* and point *9* cannot prevail each other either, as all dimensions of them are equaivalent.

From the above dataset, we can retrieve points *1*, *2*, *3*, *5*, *6*, *9* as its permissible points. In real life, a job/task/process might have dimensions like *estimated CPU time required*, *estimated I/O required*, etc.

## 2 Grading

### 2.1 Correctness (100%)

- There is a toy test case for you, the number of points is very small, you could use that to do simple testing and debugging.
- There are 10 test cases to check whether your program have correctly computed the answers.
- Different test cases will generate data different in sizes, dimensions, and data distribution, and will invoke your program with the generated dataset.
- Each test case worths 10 marks, and the toy test case worths 0 marks.

### 2.2 Bonus (25%)

There are 2 extra bonus test cases for those having good performance. If your program's real-time[1] in maximum speed (i.e., using all 4 cores) finishes faster than:

- (5%) our demo using 3 threads (cores), you get 5 marks extra bonus. [bonus test case 1]

[1]For multi-threading program, we shall not use user-time + sys-time because the threads are overlapping.

- (20%) our demo using 4 threads (cores), you get another 20 marks extra bonus. [bonus test case 2]

To eliminate any rounding error, if your program's running time is within our demo running time ±5%, we will run both programs 10 times and compare the average. The TA will not launch any other unnecessary processes while grading.

### 2.3 Note

- The TA will grade the latest version in your repo as of 4 Dec 2017, 11:00AM. That is the latest date. No late submission after that.
- For this assignment, you shall and we will set our course VM to have: (1) Base memory sets to 1G; (2) The number of processors sets to 4.

  Especially, our test has a data generator that is hardware dependent. As the expected outputs in our tester are hard-coded based on the given course VM, you may not pass the test (even your code is correct) if you are running on another platform. **So, run on our VM**.
- You **must** distribute the workload to all processors/cores.

## 3 Your assignment

You are given the following files:

| **Name** | Description |
|---|---|
| `/asgn2-pthread.c` | Code skeleton for you (**Work on it**). |
| `/Makefile` | Makefile (Don't touch) |
| `/pthreaddemo` | Executable demo. Beating it can get the bonus. |
| `/testcase` | (Don't touch) `testsuite.txt` lists the test cases and the other files contain the expected output of each test case. |
| `/runtest.c` | (Don't touch) Our grader. It will use CUnit to run the test. It will invoke the `asgn2_pthread` function in your `asgn2-pthread.c`. |
| `/util.h` | (Don't touch) Contains some utility data structure. |
| `/util.c` | (Don't touch) Contains some utility functions including the point dataset generator. |

Note: The above assumes the course's 32-bit (4 virtual CPU enabled) VM is used.

## 3.1 The assignment package

Follow the steps in section 2.1 of assignment 0 to report your Github account (if you have not done so yet), and go here `https://classroom.github.com/a/r9z4gDt7` to clone the starter package to your Github account. After this, in your terminal use the command clone "`git clone {Your Repo URL}`" and your username/password to get a local copy of the repo to your desired directory in the same way as described in assignment 0. The new repo should contain the starter package for this assignment.

**Warning: DON'T change the file names, otherwise you get 0 marks**

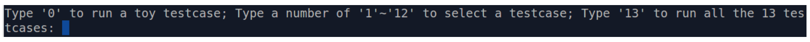
## 3.2 To begin

### 3.2.1 Run our grader

```
make
```

```
./runtest
```

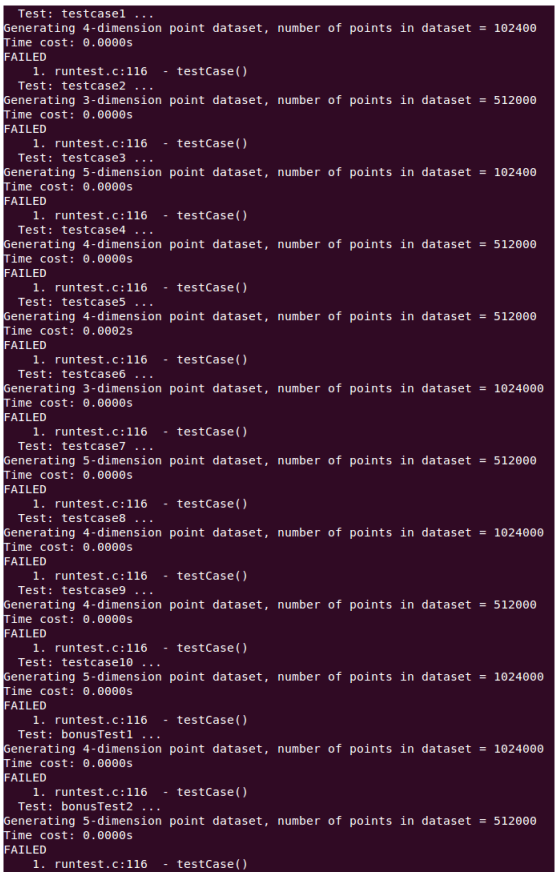
You will see:



```
Type '0' to run a toy testcase; Type a number of '1'~'12' to select a testcase; Type '13' to run all the 13 tes
tcases: 
```

- Type '0' to run the toy test case.
- Type '1' - '12' to select a test case to run.
- Type '13' to run all 13 test cases.

Before you work on your homework and type '13' to run all 13 test cases, you shall see something like below, because no test cases can pass yet.



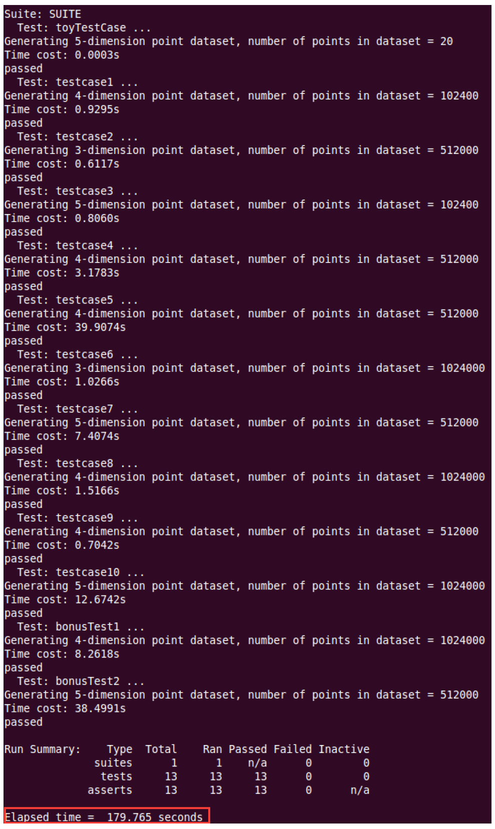
```
  Test: testcase1 ...
Generating 4-dimension point dataset, number of points in dataset = 102400
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase2 ...
Generating 3-dimension point dataset, number of points in dataset = 512000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase3 ...
Generating 5-dimension point dataset, number of points in dataset = 102400
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase4 ...
Generating 4-dimension point dataset, number of points in dataset = 512000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase5 ...
Generating 4-dimension point dataset, number of points in dataset = 512000
Time cost: 0.0002s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase6 ...
Generating 3-dimension point dataset, number of points in dataset = 1024000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase7 ...
Generating 5-dimension point dataset, number of points in dataset = 512000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase8 ...
Generating 4-dimension point dataset, number of points in dataset = 1024000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase9 ...
Generating 4-dimension point dataset, number of points in dataset = 512000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: testcase10 ...
Generating 5-dimension point dataset, number of points in dataset = 1024000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: bonusTest1 ...
Generating 4-dimension point dataset, number of points in dataset = 1024000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
  Test: bonusTest2 ...
Generating 5-dimension point dataset, number of points in dataset = 512000
Time cost: 0.0000s
FAILED
    1. runtest.c:116  - testCase()
```

After doing the assignment, if your solution is correct and you type these two commands one by one:

```
make
```

```
./runtest
```

you shall see something like this:



```
Suite: SUITE
  Test: toyTestCase ...
Generating 5-dimension point dataset, number of points in dataset = 20
Time cost: 0.0003s
passed
  Test: testcase1 ...
Generating 4-dimension point dataset, number of points in dataset = 102400
Time cost: 0.9295s
passed
  Test: testcase2 ...
Generating 3-dimension point dataset, number of points in dataset = 512000
Time cost: 0.6117s
passed
  Test: testcase3 ...
Generating 5-dimension point dataset, number of points in dataset = 102400
Time cost: 0.8060s
passed
  Test: testcase4 ...
Generating 4-dimension point dataset, number of points in dataset = 512000
Time cost: 3.1783s
passed
  Test: testcase5 ...
Generating 4-dimension point dataset, number of points in dataset = 512000
Time cost: 39.9074s
passed
  Test: testcase6 ...
Generating 3-dimension point dataset, number of points in dataset = 1024000
Time cost: 1.0266s
passed
  Test: testcase7 ...
Generating 5-dimension point dataset, number of points in dataset = 512000
Time cost: 7.4074s
passed
  Test: testcase8 ...
Generating 4-dimension point dataset, number of points in dataset = 1024000
Time cost: 1.5166s
passed
  Test: testcase9 ...
Generating 4-dimension point dataset, number of points in dataset = 512000
Time cost: 0.7042s
passed
  Test: testcase10 ...
Generating 5-dimension point dataset, number of points in dataset = 1024000
Time cost: 12.6742s
passed
  Test: bonusTest1 ...
Generating 4-dimension point dataset, number of points in dataset = 1024000
Time cost: 8.2618s
passed
  Test: bonusTest2 ...
Generating 5-dimension point dataset, number of points in dataset = 512000
Time cost: 38.4991s
passed

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests     13     13     13      0        0
             asserts     13     13     13      0      n/a

Elapsed time =  179.765 seconds
```

**Warning: The elapsed time within the red rectangle in above picture is not correct (CUnit has not supported threading well yet), so just ignore it. The time cost per individual test case is accurate.**

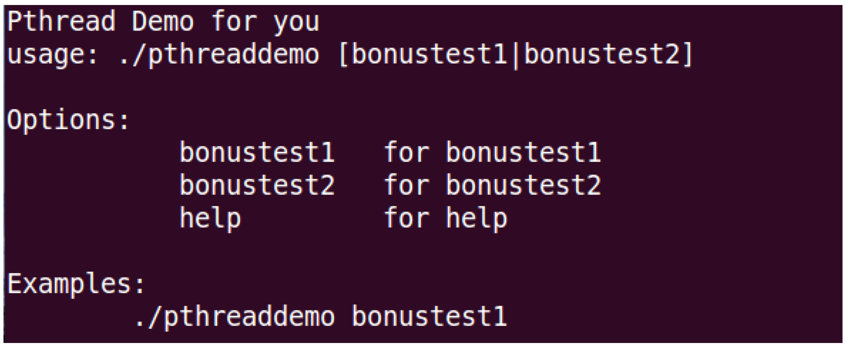
### 3.2.2 Run our demo (in case you want to try the bonus)

We give you our implementation of the assignment, `pthreaddemo`. You can run it in a standalone mode. If you type these two commands one by one:

```
chmod a+x ./pthreaddemo
```

```
./pthreaddemo help
```

You will see its usage.

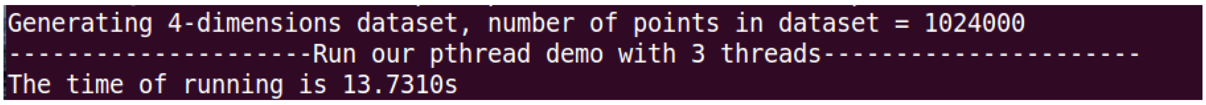


```
Pthread Demo for you
usage: ./pthreaddemo [bonustest1|bonustest2]

Options:
           bonustest1   for bonustest1
           bonustest2   for bonustest2
           help         for help

Examples:
        ./pthreaddemo bonustest1
```

It will either compute the results for bonus test case 1 or bonus test case 2 and it tells you our demo's running time on your machine. You may refer to that running time as your target to get the bonus score.



```
Generating 4-dimensions dataset, number of points in dataset = 1024000
--------------------Run our pthread demo with 3 threads---------------------
The time of running is 13.7310s
```

## 3.3 Your job

Your job is to start from the given `asgn2-pthread.c` and pass as many test cases as possible. Don't change the function signature `asgn2_pthread`. We explain some items in that file:

- Point is a C structure including an integer, ID, and a float pointer, values, which is defined in `util.h`. That is the data structure we use to represent a (generated) data point.
- thread_number: control how many threads used to run your code. The main thread doesn't count. So, you shall use PThread to create `thread_number` extra threads to speed up your job.
- Your program should store all the permissible points in a Point array[2], and set the variable permissiblePoints the pointer to that array. The Point array should be in

[2]The order of permissible points does not matter. The CUnit test program will sort the permissible points based on ID when grading. The sorting won't count towards the running time.

heap. Your code should also set the variable permissiblePointNum as the number of permissible points found.

- In the comment, there is a for-loop. You may uncomment it to print the content of first 20 points. That may get you a quick feeling about how the point dataset is stored in the memory.

[Minor] For your information, `/testcase/testsuite.txt` contains 13 lines. Each line specifies the parameter values of one test case.

- The 1st value is the number of data points to be generated
- The 2nd value is the number of dimensions to be generated.
- The 3rd value is the data characteristic for the data to be generated.
- The 4th value is the random seed that we use to generate random data.
- The 5th value is the number of threads (excluding the main thread) to use.
- The last value is the path to the expected result of the generated data.

[Minor] For your information, `/testcase/*_result.txt` contains the expected result of each test case. It lists the number of permissible points in the first line. Each subsequent line lists the ID of the permissible points in ascending order. The CUnit test program will read this file to check the output produced by your code at runtime.

## 4 Change Log

| | |
|---|---|
| 1.0 | this document |

## 5 Questions

If you have doubts about the assignment, you are encouraged to ask questions on Piazza **using the corresponding tag**. Please focus on knowledge. Unhealthy questions/comments that focus on scores and grades are not encouraged.

**If you find any (possible) bugs, send private questions on Piazza to us instead** — otherwise that may cause unnecessary panic among the class if that is not a real bug.

## 6 Academic Honesty

We follow the University guide on academic honesty against any plagiarism.

## 7 General Notes

- This specification and our grading platform are both based our given course VM. You should compile, debug and run the assignment program on that VM. So, if you insist to develop on another platform other than our VM and got any question, **test it on our VM before you ask**.
- The TA reserves the right to adjust your scores for any request that requires their extra manual effort.
- Unless specified, you should work and fill your code in designated areas. There are cases that the modification outside the designated area leads to misbehaviour of the grader. Proceed with caution and look back the changes if your output is different from what is shown in the grader.
- While we have already tried our best to prepare this assignment, we reserve all the rights to update the specification and the grading scheme. If there are any mistakes/bugs which are on ours, the TA will step in and do manual grading to ensure you get what you deserve. Please respect each other. Any irresponsible, unfair, biased sentiment would regard as a disciplinary case.
- If this is a programming assignment, only C is allowed, not even C++. If this is a scripting assignment, only bash shell script is allowed, not even Python. Furthermore, for C programming assignments, use the "exit" function **parsimoniously** because it might influence the grader as well. Therefore, use "return" instead of "exit" whenever possible.

- Although this is not an algorithm class, you still shouldn't implement your assignment with very poor complexity (e.g., $O(n^3)$). The TA reserves the right to terminate a test case and regard that as a fail test case when a program takes unreasonably long time (e.g., with deadlock) to finish.

- **(Frequently Asked)** [Output format] If the assignment package includes a demo, then our grader defines test cases based on the given demo. In that case, your output shall **exactly** follow that demo. For example, hypothetically, if our demo outputs a message like:

  ```
  command not␣␣found
  ```

  with two spaces between "`not`" and "`found`". Your output shall also match that in order to pass the test. The good news is that, if our given demo has not implemented something (e.g., missed certain error checking), you also don't need to do so. **No test cases would be defined based on something that our demo has not implemented**.

- **(Frequently Asked)** [Scope of error handling] The scope of **error checking and handling** shall refer to **both our given demo (if given) and our given test cases**. First, the corresponding output message shall **exactly** follow our demo. Second, you are informed that our demo may have implemented more error checking that what our grader will test. In that case, it is fine that you implement only the error checking that is tested by our grader. So, one top tip is:

  **CHECK THE (SOURCE OF)**

  **TEST CASES BEFORE YOU ASK**

- **(Frequently Asked)** [No hidden test case] We are not intended to run any secret/extra test cases that deliberately break your assignment. That is, WYSIWYG — your final score shall be generally indicated by what the grader reports when it runs on the course VM. However, we do reserve the right to run some additional test cases to avoid any mis-conduct (e.g., hard-coding the results), and/or invite you to explain the source code to the teaching assistants and adjust the scores accordingly.

- We welcome discussions among classmates. But don't share your assignment with the others in any means. For example, don't put your source code in any public venue (e.g, public repo, your homepage, Facebook). We handle plagiarism strictly. On submitting this assignment, you are agreed that your code's copyright belongs to the Chinese University of Hong Kong. Unless with our written approval, you **must not** release your source code and this specification now and **forever**. If you share your code with anyone without our written consent, that would regard as a disciplinary case as long as you are still a CUHK student (i.e., even after this course). If you share your code with anyone without our written consent after your graduation, that would regard as a breach of copyright and we reserve all the rights to take the corresponding legal actions.

- Google is your friend. We encourage you use Google for help to do the assignment. However, if you happen to find any source codes related to this assignment, you still cannot copy it but use your own way to implement it. You need to put down your list of source code references as comments in the top of your source code.