CSE130 Spring 2021 : Assignment 2

In this assignment you will implement a multi-threaded merge sort using POSIX Threads.

This lab is worth 5% of your final grade.

Submissions are due NO LATER than 23:59 Monday April 19 2021 (1 week)

Setup

SSH in to one of the two CSE130 teaching servers using your CruzID Blue password:

```
      $ ssh <cruzid>@noggin.soe.ucsc.edu   ( use Putty http://www.putty.org/ if on Windows )
or    $ ssh <cruzid>@nogbad.soe.ucsc.edu
or    $ ssh <cruzid>@olaf.soe.ucsc.edu
or    $ ssh <cruzid>@thor.soe.ucsc.edu
```

Authenticate with Kerberos: ***(do this every time you log in)***

```
      $ kinit      ( you will be prompted for your Blue CruzID password )
```

Authenticate with AFS: ***(do this every time you log in)***

```
      $ aklog
```

Create a suitable place to work: ***(only do this the first time you log in)***

```
      $ mkdir -p CSE130/Assignment2
      $ cd CSE130/Assignment2
```

Install the lab environment: ***(only do this once)***

```
      $ tar xvf /var/classes/CSE130/Assignment2.tar.gz
```

Build the starter code:

```
      $ cd ~/CSE130/Assignment2      ( always work in this directory )
      $ make
```

Then try:

```
      $ make grade      ( runs the required functional and non-functional tests - see below )
                        ( also tells you what grade you will get - see below )
```

Run the provided single process merge sort:

```
      $ ./sort -s 32
```

Run the skeleton multi process merge sort:

```
      $ ./sort -m 32    ( this will fail to sort the randomly generated array )
```


Additional Information

In Assignment 1 you merge sorted an array in multiple processes; in this assignment you merge sort in multiple threads. The POSIX Thread functions you will need are:

`pthread_create`	http://man7.org/linux/man-pages/man3/pthread_create.3.html
`pthread_join`	http://man7.org/linux/man-pages/man3/pthread_join.3.html
`pthread_exit`	http://man7.org/linux/man-pages/man3/pthread_exit.3.html

To achieve the required speedup, you will need to create multiple threads.

Note that shared memory is ***not*** required to complete this assignment and should not be used.

Requirements

Basic:

- You have implemented a multi-threaded merge sort that correctly sorts random arrays of integers when using the supplied `merge()` function

Advanced:

- Your implementation is at least 2.625 times faster than the supplied single threaded merge sort when using the supplied `merge()` function

What steps should I take to tackle this?

Consider how many threads you will need and what portions of the supplied array they will be sorting and/or merging. Then consult the lecture handouts on POSX threads for implementation details.

As always, take small steps towards a complete implementation, don't try to do the whole thing in one go.

A first step might be to create a single thread to sort the entire array and have the main process wait for it to exit. This will pass the functional test but not the non-functional (speed up) tests. Then perhaps have two threads sort one half of the array each.

Remember, do not use shared memory for this assignment. Threads have access to the same local memory so all threads you create can see the given array of integers.

How much code will I need to write?

A model solution that satisfies all requirements adds approximately 25 lines of executable code.

Grading scheme

The following aspects will be assessed:

1. (100%) **Does it work?**

    a. Functional tests pass (45%)
    b. Non-Functional (performance) tests pass (45%)
    c. Your implementation is free of compiler warnings and memory errors (10%)

2. (-100%) **Did you give credit where credit is due?**

    a. Your submission is found to contain code segments copied from on-line resources and you failed to give clear and unambiguous credit to the original author(s) in your source code (-100%). You will also be subject to the university academic misconduct procedure as stated in the class academic integrity policy.

    b. Your submission is determined to be a copy of a past or present student's submission (-100%)

    c. Your submission is found to contain code segments copied from on-line resources that you did give a clear an unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:

        - < 25% copied code — No deduction
        - 25% to 50% copied code — (-50%)
        - > 50% copied code — (-100%)

What to submit

In a command prompt:

```
$ cd ~/CSE130/Assignment2
$ make submit
```

This creates a gzipped tar archive named `CSE130-Assignment2.tar.gz` in your home directory.

****** UPLOAD THIS FILE TO THE APPROPRIATE CANVAS ASSIGNMENT ******

