

CSE130 Winter 2020 : 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, Sunday February 2 2020 (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

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/Winter20/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 at least four threads.

Note that shared memory is not required to complete this assignment.

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 2.5 times faster than the supplied single process merge sort when using the supplied `merge()` function

What steps should I take to tackle this?

Come to sections and ask.

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 (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 and unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:
 - < 33% copied code No deduction
 - 33% to 66% copied code (-50%)
 - > 66% 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 ******

§