CSE130 Spring 2021 : Assignment 1

In this assignment you will implement a multi process merge sort using the `fork()` and `wait()` system calls and both UNIX and POSIX shared memory system calls.

This lab is worth 5% of your final grade.

Submissions are due NO LATER than 23:59, Monday April 12 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/Assignment1
      $ cd CSE130/Assignment1
```

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

```
      $ tar xvf /var/classes/CSE130/Assignment1.tar.gz
```

Build the starter code:

```
      $ cd ~/CSE130/Assignment1      ( 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 UNIX Shared Memory and POSIX Shared Memory single process merge sorts:

```
      $ ./usort -s 32
      $ ./psort -s 32
```

Run the skeleton UNIX Shared Memory and POSIX Shared Memory multi process merge sorts:

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

Sorting fewer than 32 integers will display the sorted array after execution which may help with debugging.

Additional Information

We will be covering UNIX and POSIX shared memory in class on Thursday April 8 but we have already covered process creation and control using the `fork()` and `wait()` system calls so you can make a start on this assignment now and add shared memory either through your own research or after we cover it in class.

Requirements

Basic:

- You have implemented a UNIX and POSIX Shared Memory multi process merge sorts that correctly sorts random arrays of integers when using the supplied `merge()` function

Advanced:

- Your implementations are at least 1.7 times faster than the supplied single process merge sort when using the supplied `merge()` function

What steps should I take to tackle this?

Review the assignment introduction from class then if you still have questions come to office hours and ask.

How much code will I need to write?

A model solution that satisfies all requirements adds approximately 60 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 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/Assignment1
$ make submit
```

This creates a gzipped tar archive named `CSE130-Assignment1.tar.gz` in your home directory.

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

