SWE 3243 / 5243 — Project 4
Operating Systems — *Fall 2013*

PROJECT 4
Process Synchronization

This project is comprised of three phases to familiarize you with forks and threads. You are free to code your solutions using any tools or operating system you choose, but be aware that your project will be graded using Ubuntu 12.04. You are responsible for ensuring that your code compiles and runs correctly on this operating system using standard APIs.

PART ONE

(75 points)

Producers and Consumers

Solve Programming Project 6.40 using pThreads and mutex locks.

Name your file(s) for this project **project4a.cpp**

Note: If you know a classmate who has a copy of the 9th edition of the textbook, it has a slightly better write up of this problem that you might wish to read. Refer to Chapter 5, Programming Project 3, in the 9th Edition)

BONUS

The first project team that shows – in person, during scheduled office hours – a correct, working solution to this problem with attractive and clear output will receive 15 bonus points. The second team to do so will receive 7 bonus points.

PART TWO

(125 points)

Instructions

Using the 10,000 unsorted numbers list attached to the project description, create a program that sorts the list in ascending order by dividing the list among four separate child processes. The parent and four child processes must communicate and synchronize according to the following rules:

1) The array of 10,000 unsorted numbers and 10,000 sorted numbers must be shared between all processes using POSIX shared memory.
2) Each of the four child processes will be responsible for ONLY 2,500 of the unsorted numbers (i.e. 0 – 2499 for child 1, 2500-4999 for child 2, etc.).
3) Each of the child processes will use the SELECTION SORT algorithm to collaboratively build the sorted array in shared memory.
4) Each child will add to the sorted array 10 NUMBERS AT A TIME ONLY.
5) The parent process will be responsible ensuring the 10 elements added by the child processes are in the correct position relative to the rest of sorted array.
6) No process may write to the shared sorted array when either another child process is writing to it or the parent process is sorting it. Use semaphores to accomplish this.

If the sorting was done on a 30-element array with three child processes, the sorting might be visualized like the following:

Unsorted Array:

93	24	13	7	84	56	76	33	31	99	87	82	26	42	12	9	5	49	44	54	67	86	72	11	53	51	60	92	3	15

First child's elements	Second child's elements	Third child's elements

Note that the unsorted array is never changed, just read from.

Step 1: **First** child sorts its 10 elements using selection sort:
Sorted Array:

7	13	24	31	33	56	76	84	93	99																				

Step 2: **Third** child swaps in and add its 10 elements using selection sort:
Sorted Array:

7	13	24	31	33	56	76	84	93	99	3	11	15	51	53	60	67	72	86	92										

Note first that the third child just appends to the sorted array. This implies that there is also some shared variable to keep track of the current index. Note also that while both 0-9 and 10-19 are sorted amongst themselves, they are not sorted relative to each other.

Step 3: **Parent** process swaps in and sorts the sorted array relative to itself:
Sorted Array:

3	7	11	13	15	24	31	33	51	53	56	60	67	72	76	84	86	92	93	99										

Step 4: **Second** child swaps in and sorts its 10 elements, adding them to the end of the sorted array:
Sorted Array:

3	7	11	13	15	24	31	33	51	53	56	60	67	72	76	84	86	92	93	99	5	9	12	26	42	44	49	54	82	87

Step 5: **Parent** process swaps in again and sorts the sorted array relative to itself:
Sorted Array:

3	57	9	7	11	12	13	15	24	26	31	33	42	44	49	51	53	64	56	60	67	72	76	82	84	86	87	92	93	99

Name your file(s) for this project **project4b.cpp**

PROJECT SUBMISSION

1. ALL SOURCE CODE YOU TURN IN MUST CONTAIN THE FOLLOWING AT THE TOP:

```
// CS3242 Operating Systems
// Fall 2013
// Project 4: Process Synchronization, Part 1
// John S. Doe and Bob A. Smith
// Date: 9/23/2013
// File: partone.c
```

2. Zip ALL source code files for your project into a single ZIP file named "DOE_SMITH.ZIP" (where Doe and Smith are the surnames of the two students) and upload that as your submission in Dropbox on D2L by the posted deadline.