FLORIDAPOLY

Programming Assignment (Threads)

Topic: Pthread API
Resources: Textbook Three easy pieces, chapter 27 (Thread API)
Other resources: http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html#BASICS

Submission deadline: April 15th 11:59pm

Goal: Make familiar students with the POSIX thread (pthread) libraries

Deliverables: A c program. No report required for this assignment.

NOTE: Test your program in the ember system (Unix system of Floridapoly)

NOTE: roles

a. parent received the input through the command line using: int main(int argc, char *argv[]), don't forget o use (atoi) to transform the char 7 into the integer.

b. child computes the sequence and stored the result in an array.

c. parent prints the sequence

The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8, ... Formally, it can be expressed as:

$fib_0 = 0$
$fib_1 = 1$
$fib_n = fib_{n-1} + fib_{n-2}$

Write a multithreaded program that generates the Fibonacci series using Pthreads thread library. This program should work as follows: The user will enter on the command line the number of Fibonacci numbers that the program is to generate (* in this case 7 *). The program will then create a separate thread that will generate the Fibonacci sequence, placing the sequence in data-code (section) that is shared by the threads (an array is probably the most convenient data structure). When the thread finishes execution, the parent thread will output the sequence generated by the child thread. Because the parent thread cannot begin outputting the Fibonacci sequence until the child thread finishes, this will require having the parent thread wait for the child thread.

Example of expected outcome: If the name of the executable program is **luis** then you may run the program as follows:

./luis 7

Output:

0 1 1 2 3 5 8

NOTE: Just for the condition of this semester, use the fix value of 7 (so, the program doesn't have to work for any input, namely just for the number 7), in that way you can use an array of size 7 to store the Fibonacci series (child thread has to generate this sequence), so you don't need to use dynamic memory allocation

Advise:

1. EASY, just declare the array where parent and child can access it. (code-data section, i.e., global array, check slides examples)
2. Don't forget to lock the critical section or share variable, namely the array
3. For thread synchronization use **LOCKS**