CSI 3131 Assignment 2 Summer 2024

Due: June 30, 2024

Threads synchronization and cooperative execution

Goal

Practice threads synchronization and cooperatively execute some tasks, while solving some related issues like deadlocks and races.
Reference: the textbook chapter 4 to 6.

Description

Please read the complete assignment document carefully before starting.

Write a program (C Pthread is suggested but you can also use java or windows if you prefer) which will have to create **4 threads**. These 4 threads share some resources and cooperatively execute some tasks.

The program should have 3 integer variables shared among all threads, int: **a**, **b** and **turn**. These variables should be the only global variables. Any other needed variables should be declared as a local variable to each function/thread. The main() thread should ask the user to enter a zero or positive values: for variables **a**, **b**, and **turn** (use scanf). **turn** can take only values from 0 to 3. Then the main() thread should create the 4 theads (**t0**, **t1**, **t2** and **t3**). Each thread will run on a different function with the prototype: **void *thr*n*()**, where ***n*** is the thread number (0 to 3).
Once started, the thread scheduling should work as follows:

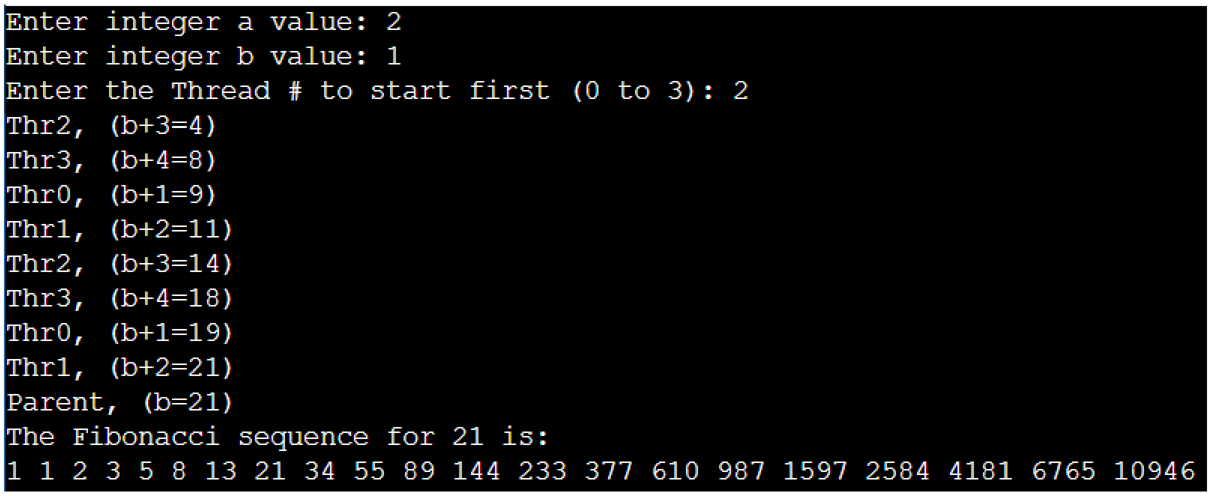
a. The threads will run in round robin (RR)
b. The value of the variable **turn** define which thread will run first e.g., if **turn=2** then Th2 →Th3→Th0→Th1→Th2…
c. The value of the variable **a** define the number of threads RR repetition before terminating e.g., if **a=2** and **turn=0** then Th0→Th1→Th2→Th3→Th0→Th1→Th2→Th3.
d. Only 1 thread is allowed to execute its critical section (CS) at a time. E.g., when Th0 is executing, all other threads should be in a busy wait (BW).
e. When an executing thread exit its CS it must set the value of variable **turn** to the next Thread to execute.
f. Then main() thread (or the parent thread) should wait (join) after all 4 children threads to terminate before continuing the execution of its remaining section to produce some useful data then terminates.

Each thread task is as follows:

a. Th0: calculates **b** = **b** + 1, print the Thead # and **b** value for an **a** repetition
b. Th1: calculates **b** = **b** + 2, print the Thead # and **b** value for an **a** repetition
c. Th2: calculates **b** = **b** + 3, print the Thead # and **b** value for an **a** repetition
d. Th3: calculates **b** = **b** + 4, print the Thead # and **b** value for an **a** repetition

e. Th-main(), (after joining all thteads) will print “Parent” and the final value of **b**. Then it will calculate the Fibonacci sequence for b, and print them before terminating.

To test your program here is some running results output:



```
Enter integer a value: 2
Enter integer b value: 1
Enter the Thread # to start first (0 to 3): 2
Thr2, (b+3=4)
Thr3, (b+4=8)
Thr0, (b+1=9)
Thr1, (b+2=11)
Thr2, (b+3=14)
Thr3, (b+4=18)
Thr0, (b+1=19)
Thr1, (b+2=21)
Parent, (b=21)
The Fibonacci sequence for 21 is:
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946
```

Background information:

Use this youtube link:
https://youtu.be/d9s_d28yJq0?list=PLfqABt5AS4FmuQf70psXrsMLEDQXNkLq2 for threads tutorial, see the below appendix about Fibonacci sequence, and refer to the textbook and lecture slides chapter 4 to 6.

To complete the assignment:

1. You need to submit your prog.c file and enter your name and student ID as a header comment at the top of your file.
2. Submit you prog.c (**do not compress**) file in the dedicated submission link on BS by the due date (displayed on the submission link).
3. Optional but it is highly recommended that you also submit a “proof” file in PDF format. This file should contain your same code (a copy/paste from prog.c) and a screenshot of its execution output.

Good luck!

Appendix

The following sequence is called the Fibonacci Sequence:

1, 1, 2, 3, 5, 8, 13, 21, 34, . . .

It is defined by:

$F_1 = 1$

$F_2 = 1$

$F_i = F_{i-1} + F_{i-2}$, when $i > 2$

The numbers in the sequence are called Fibonacci numbers (“F-numbers”).

Pseudo code:

```
3 variables v1=1, v2=1, and v3
If b<=v1, print v1  else print v1, v2
and for i=2, while i<b, i++{
  v3=v1+v2
  v2=v1
  v1=v3
  print v3
}
```