Programming Homework Assignment #2

J. H. Wang

Apr. 10, 2023

Programming Homework #2

- Programming exercises:
  - Programming problems: 4.17*, (4.21**,) 6.33*
    - Note: Each student must complete all programming problems on your own
  - Programming projects for Chap. 4* (*2) & Chap. 6* (*3)
    - Team-based: At least one selected programming project from each chapter
- Due: two weeks (Apr. 24, 2023)

Programming Problems

- 4.17*: An interesting way of calculating pi is to use a technique known as *Monte Carlo*, which involves randomization. This technique works as follows:
  - Suppose you have a circle inscribed within a square, (Assume that the radius of this circle is 1.)
  - First, generate a series of random points as simple (x,y) coordinates
  - These points must fall within the Cartesian coordinates that bound the square
  - Of the total number of random points that are generated, some will occur within the circle
  - (… to be continued)

- Next, estimate pi by performing the following calculation:
  - Pi=4*(number of points in circle) / (total number of points)

– Write a multithreaded version of this algorithm that creates a separate thread to generate a number of random points.
  - The thread will count the number of points that occur within the circle and store that result in a global variable.
  - When this thread has exited, the parent thread will calculate and output the estimated value of pi.

- [optional] (4.21**): 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 sequence using either the Java, Pthread, or Win32 thread library.
  (… to be continued)

- This program should work as follows:
  - On the command line, the user will enter the number of Fibonacci numbers that the program is to generate
  - The program will then create a separate thread that will generate the Fibonacci numbers, placing the sequence in data that can be 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 until the child finishes, the parent will have to wait for the child thread to finish

- 6.33*: Modify the program in Exercise 4.17 so that you create several threads, each of which generates random points and determines if the points fall within the circle.
  - Each thread will have to update the global count of all points that fall within the circle.
  - Protect against race conditions on updates to the shared global variable by using mutex locks.
  - (Note: You can use mutex lock or semaphores in Pthread, or Windows API if you want.)

End-of-Chapter Programming Projects

- Programming Projects for Chap. 4: (Choose one)
  - Project 1. Sudoku solution validator
    - To design a multithreaded application that determines whether the solution to a Sudoku puzzle is valid.
    - Passing parameters to each thread
    - Returning results to the parent thread
  - Project 2. Multithreaded sorting application
    - Write a multithreaded sorting program that works as follows: A list of integers is divided into two smaller lists of equal size. Two separate threads sort each sublist using a sorting algorithm of your choice. The two sublists are then merged by a third thread.

End-of-Chapter Programming Projects

- Programming Projects for Chap. 6: (Choose one)
  - Project 1: The Sleeping Teaching Assistant
    - Room: 1 desk with a chair and computer
    - Hallway: 3 chairs
    - POSIX threads, mutex locks, and semaphores
  - Project 2: The Dining Philosophers Problem
    - Pthread mutex locks and condition variables
  - Project 3: Producer-Consumer Problem
    - Use standard counting semaphores for empty and full and a mutex lock, rather than a binary semaphore, to represent mutex.
    - Producer and consumer threads
    - Pthreads mutex locks/semaphores
    - Windows mutex locks/semaphores

Homework Submission

- For programming exercises, please upload your program to the iSchool+ as follows:
  - Program uploading: a compressed file (in .zip format) including source codes, execution snapshot, and documentation
  - The documentation should clearly identify:
    - Team members and responsibility
    - Compilation or configuration instructions if it needs special environment to compile or run
  - Please contact with the TA if you are unable to upload your homework

Any Question or Comments?