EE442 PROGRAMMING ASSIGNMENT 1

Finding Prime Numbers

Due: March 31, 2024, 23:55

***For your questions use forum or send an email to Kamil Sert,** ksert@metu.edu.tr.

Submission

- Send your homework compressed in an archive file with name "**eXXXXXXX_ee442_pa1.tar.gz**", where X's are your **7-digit student ID number**. You will **not** get full credit if you fail to submit your work as required.
- Your work will be graded on its correctness, efficiency, clarity and readability as a whole.
- Comments will be graded. You should insert comments in your source code at appropriate places without including any unnecessary detail.
- Late submissions are welcome, but are penalized according to the following policy:
  - 1 day late submission : HW will be evaluated out of **70**.
  - 2 days late submission : HW will be evaluated out of **40**.
  - Later submissions : HW will NOT be evaluated.
- The homework must be written in **C** (**not** in C++ or any other language).
- You should **not** call any external programs in your code.
- **Check** what you upload. Do not send corrupted, wrong files or unnecessary files.
- The homework is to be prepared **individually**. Group work is **not** allowed. Your code will be **checked** for cheating.
- The design should be your original work. However, if you partially make use of a code from the Web, give proper **reference** to the related website in your comments. Uncited use is unacceptable.
- **METU honor code is essential**. Do **not** share your code. Any kind of involvement in cheating will result in a **zero** grade, for **both** providers and receivers.

Introduction

In this homework, you are asked to implement a safe queue structure and associated functions to be used by threads. The threads will be used to find prime numbers in a range of random positive numbers.

You are expected to use appropriate *IPC primitives* (mutexes, semaphores, locks, condition variables, sleep-wait etc.) for critical section and/or synchronization.

Command-line options should be used for changing the parameters of the program.

Queue Structure

Queue is an abstract data type in which the elements are inserted to the rear and are removed from the front (first in, first out). In this homework, it will be implemented as a dynamically allocated array size of which is selected from the command-line. The array will be used circularly. The actual C struct which holds the queue should include the following fields:

- The integer pointer holding the memory location of the array
- The maximum size of the queue
- The current size of the queue
- The index of the front

Any additional variables directly related to the queue operation should also be a field of the struct. If the queue is full, the threads wanting to insert should wait until it is possible. Similarly, if the queue is empty, the threads wanting to remove should wait.

For the operation of the queue, four functions should be implemented, namely:

- `QueueInitialize`: Initialize the necessary fields of the queue.
- `QueueInsert`: Insert an integer to the queue.
- `QueueRemove`: Remove an integer from the queue.
- `QueueDestroy`: Destroy the necessary fields of the queue.

The declarations and implementations of the struct and the related functions should be in `queue.h` and `queue.c` files.

Generator Thread

The program will create one generator thread. This thread will generate random positive integers and insert them to the queue. The quantity of the numbers and their possible range will be decided by the command-line arguments.

After each generation, the number generated will be inserted to the rear of queue. If the queue is full, then the number will be discarded. Then, the generator thread will sleep and generate as usual.

Number Generation Rate

Between each number generation, the generator thread should sleep for a random amount of time with an exponential distribution (the rate is selected with -g option standing for generation time).

From a continuous uniform distribution x between 0 and 1, the exponential distribution can be generated as follows:

$$f(y, \lambda) = -\frac{1}{\lambda}\ln(1 - x)$$

where λ is the rate.

Worker Threads

These threads will remove a number from the queue and find whether the number is prime or not. If the number is a prime number, then the following message is going to be printed.

"Thread ID: 3059108672, Number: 79 is a prime number."

If the number is not a prime number, then the first 10 divisors of the number is going to be printed. A result should be as follows:

"Thread ID: 3059108672, Number: 96 is not a prime number. Divisors: 1 2 3 4 6 8 12 24 32 48"

Then, worker threads will start over until the generator thread finishes working.

Command-line arguments

The program should use four optional arguments to change parameters:

- `-t`: Number of worker threads (default 3)
- `-q`: The maximum size of the queue, which is shared by the threads (default 5)
- `-r`: The amount of the random numbers (default 10)
- `-m`: The lower bound of the range of the random numbers (default 1)
- `-n`: The upper bound of the range of the random numbers (default 100, maximum 2000)
- `-g`: The rate of generation time (default 100)

Instead of parsing the command-line arguments directly, you may want to use `getopt()`.

Hints

- If you have main.c, queue.c and queue.h as source files, you can compile your code with this command:
  gcc -o prime main.c queue.c –pthread
- If you have only main.c as a source file, you can compile your code with this command:
  gcc -o prime main.c –pthread
- Some libraries need to be linked explicitly. One example is math.h library where gcc needs -lm option.

Remarks

- There should be **no deadlock or starvation.**
- Synchronization variables should be used **only for critical sections.**
- There should be **no memory leak.**
- You can find information about headers, functions and types here.
- Send only the source code (`queue.h, queue.c, main.c`). Do not send any executable, since your code will be recompiled.

POSIX thread (pthread) library

You may find useful information about pthread library (thread create, thread synchronization etc.) and examples in the following web pages.

- https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html#SYNCHRONIZATION
- https://www.educative.io/answers/how-to-create-a-simple-thread-in-c
- https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-pthread-create-create-thread
- https://pubs.opengroup.org/onlinepubs/9699919799/

Pthread Mutex

An example code to show how pthread mutexes are used for thread synchronization is given below.

```
int count = 0;                          /* shared count variable */
pthread_mutex_t mutex;                  /* pthread mutex */

int main()
{
        ……
        ……    /* main code */
        ……
}

/* Each thread executes this function. */
void * countFunction(void *arg)
{
        int i;
        for (i = 0; i < 5; i++)
        {
                /* Enter critical section. Acquire mutex lock. */
                Pthread_mutex_lock(&mutex);
                count++;
                /* Exit critical section. Release mutex lock. */
                Pthread_mutex_unlock(&mutex);
        }
        return NULL;
}
```

Examples

```
ee442@ee442-VirtualBox: ~/Desktop/HW1$ ./prime
GENERATION_RATE: 100.000000

Thread ID: 3042589504, Number: 53 is a prime number.
Thread ID: 3059108672, Number: 76 is not a prime number. Divisors: 1 2 4 19 38 76
Thread ID: 3067767616, Number: 48 is not a prime number. Divisors: 1 2 3 4 6 8 12 16 24 48
Thread ID: 3042589504, Number: 89 is a prime number.
Thread ID: 3067767616, Number: 43 is a prime number.
Thread ID: 3059108672, Number: 20 is not a prime number. Divisors: 1 2 4 5 10 20
Thread ID: 3042589504, Number: 28 is not a prime number. Divisors: 1 2 4 7 14 28
Thread ID: 3059108672, Number: 11 is a prime number.
Thread ID: 3067767616, Number: 7 is a prime number.
Thread ID: 3042589504, Number: 63 is not a prime number. Divisors: 1 3 7 9 21 63
```

```
ee442@ee442-VirtualBox: ~/Desktop/HW1$ ./prime -t 5 -r 5 -n 1000
GENERATION_RATE: 100.000000

Thread ID: 3075828544, Number: 743 is a prime number.
Thread ID: 3067435840, Number: 840 is not a prime number. Divisors: 1 2 3 4 5 6 7 8 10 12
Thread ID: 3042257728, Number: 373 is a prime number.
Thread ID: 3050650432, Number: 385 is not a prime number. Divisors: 1 5 7 11 35 55 77 385
Thread ID: 3059043136, Number: 487 is a prime number.
```