Faculty of natural & Applied Sciences
Department of Computer Science

Spring 2020
Wednesday, April 15, 2020

CSC 414 Applied Operating Systems

Project 2 – Concurrent Programming

Due date: Monday, April, 27, 2020, 10 PM

1. Overview

In this individual project you will have to use the Master/Slave model to compute prime numbers using two methods:

- Forking children
- Threads.

2. Requirements

Calculating Prime Numbers

Prime numbers are computed up to a maximum `max_prime` using the function below:

```
unsigned long calculate_primes(int slave_id, int num_slaves, unsigned long max_prime)
{
    unsigned long  num, i, primes = 0, flag;
    num = 3 + 2*slave_id;

    while (num < max_prime)
    {
        flag = 0;
        for (i = 2; i <= num/2; i++)
        {
            if (num%i == 0)
            {
                flag = 1;
                break;
            }
        }
        if (flag == 0 && (num > 1))
        {
            ++primes;
        }
        num += 2*num_slaves;
      }
      return primes;
}
```

Creating Slaves

A master process creates (fork/create thread) n slaves (slave_id = 0..n-1) and waits on them while they compute the prime numbers. Each slave will compute the primes starting at 3 + 2*slave_id up to max_prime stepping by 2*n_slaves, where max_prime is the maximum integer used to search for a prime and n_slaves is the number of slaves.

In both cases, children and threads, the slaves must display the number of primes found and the running time. In the case of the threads, however, the prime numbers must be stored in a global array.

You should use POSIX threads as given in the example (test_thread_with_struct_arg) on the Blackboard. Note that the function `calculate_primes` must be adapted to the threads case by changing its signature and adding the calculated prime number to the global array.

Displaying Running Times

The program should display the time (in milliseconds) spent by each child/thread in addition to the total time spent by the main program. The following is an example of the output in the case of fork:

```
$ ./doPrimes 100000 4 process
Process 1 computed 2399 prime numbers in 577 milliseconds
Process 0 computed 2409 prime numbers in 592 milliseconds
Process 2 computed 2399 prime numbers in 561 milliseconds
Process 3 computed 2384 prime numbers in 577 milliseconds
This machine calculated all prime numbers under 100000 using 4 slaves in 639
milliseconds
```

Similar output is obtained with the threads method, just replace Process with Thread.

Using Command Line Arguments

The program should be called using the following arguments in order:

1. max_prime: positive integer representing the maximum prime
2. n_slaves: positive integer representing the number of slaves
3. "process" /"thread": string determining if the slaves are processes or threads.

For example:

`doPrimes 20 4 process` uses the fork method with 4 children and computes the primes up to 20

`doPrimes 20 4 thread` uses the thread method with 4 threads and computes the primes up to 20

3. Hints and Tips

Time Measurements

Measuring children time:

clock()

`clock_t clock(void)` returns the number of clock ticks elapsed since the program was launched. To get the number of seconds used by the CPU, you will need to divide by `CLOCKS_PER_SEC`.

```
#include <time.h>
#include <stdio.h>
int main () {
    clock_t start_t, end_t, total_t;

    start_t = clock();
    /* do stuff */
    end_t = clock();

    total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
    printf("Total time taken by CPU: %f\n", total_t  );
    return(0);
}
```

Measuring Parent Time

times - get process and waited-for child process times

Upon successful completion, `times()` shall return the elapsed real time, in clock ticks, since an arbitrary point in the past

```
#include <sys/times.h>
#include <stdio.h>
int main () {
    static clock_t st_time;
    static clock_t en_time;
    static struct tms st_cpu;
    static struct tms en_cpu;

    st_time = times(&st_cpu);
    /* do stuff: fork children and wait on them. */
    en_time = times(&en_cpu);
    printf("Real Time: %jd,(intmax_t)(en_time - st_time
}
```

4. Report

Grading scores shall be provided for required elements stated above using the point values listed below.

1. Project Code submission: **80** points total as follows:
    a. Proper functionality: **[70 points]**
    b. Comments: up to 10 points awarded based on completeness and quality of comments provided as follows **[10 points]**:
        i. program header containing all components: **4 pts**
        ii. description for each code block: **3 pts**
        iii. one comment for non-obvious, interesting, or important parts of your code.: **3 pts**
2. Project Report submission: **20** points total as follows:
    a. Test Plan: Test your program for the following configurations: **10 pts**
        i. Non parallel: process only.
        ii. parallel: Children and threads using:
        max_prime = 1000000;
        n_slaves: 2, 4, 8, 16, 32, 64.
    b. Test Results: provide 2 screen shots of 2 code executions: 2 children, 2 threads: **5 pts**
    c. Plot using Excel 2 speedup curves showing the *number of children/threads* in the x coordinate and the ratio: *cpu time/non parallel cpu time* in the y coordinate: **5 pts**

Submit the project files in a compressed zip archive to kkhaldi@ndu.edu.lb.