ISE 302–Operating Systems 18.10.2018

2018–2019, Homework 1

Introduction

You will code a multi-processing and multi-threading program that finds the prime numbers. The number interval for search (**interval_min** and **interval_max**), the number of processes (**np**) and the number of threads (**nt**) will be given as command line arguments. The number interval will be used as a search space to find prime numbers. You need to divide the number interval among **np** process, and each process should also divide them among **nt** threads. Threads will be responsible for finding the prime numbers in the assigned range. Upon completion of threads, processes should print the prime numbers that are calculated in ascending order.

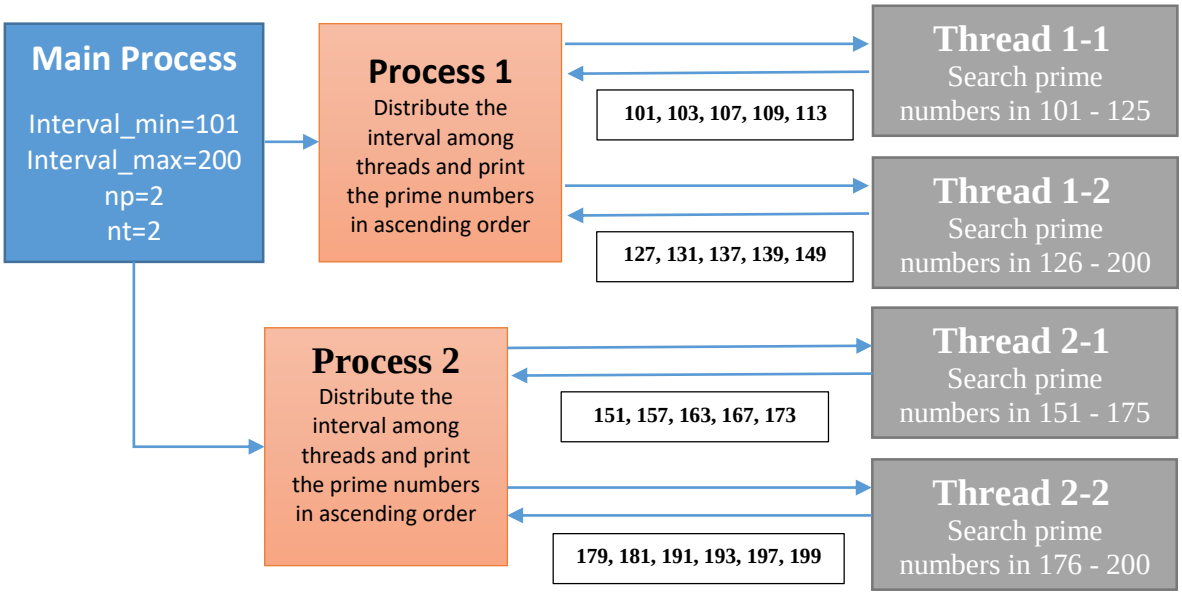
Program Input

Program inputs are the number interval (interval_min, interval_max), the number of processes(**np**) and the number of threads(**nt**). These inputs should be received as a command line argument. To evaluate your programs, multiple options will be used.

./yourprogram interval_min interval_max np nt

Example command: **./yourprogram 101 200 2 2**

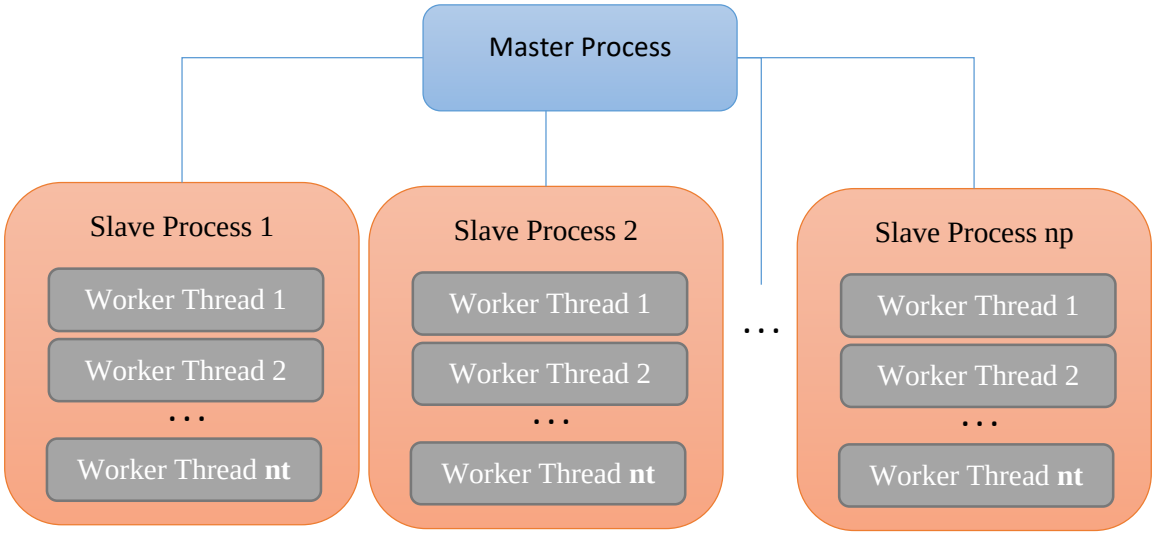
It will create **2** processes and each process will create **2** threads.
Process **1** will be responsible for the numbers between **101-150**
Thread 1-**1** will check for primality in the range of **101-125**
Thread 1-**2** will check for primality in the range of **125-150**
Process **2** will be responsible for the numbers between **151-200**
Thread 2-**1** will check for primality in the range of **151-175**
Thread 2-**2** will check for primality in the range of **176-200**



Processes and Threads

Your program should have the following functionality for the processes and threads.

- **Master Process:** Master process should receive arguments as a command line argument Then, it should create **np** slave processes. It will also divide the number interval into the number of processes, and distribute them among the slave processes. (E.g. If the initial range is 1-10 and the number of slave processes is 2, the processes will be assigned to the ranges of 1-5 and 6-10)
- **Slave Processes:** Each slave process should create **nt** worker threads and assign a number range to each thread. Upon completion of its threads, the slave process must print the prime numbers **that are calculated by the worker threads** in ascending order.
- **Worker Threads:** Each worker thread is responsible for checking primality of numbers in the defined range.



Program Output

A sample program output is given below. Your program must print similar information to the screen. Please note that, the order of lines may be different at each run.

```
./program 101 200 2 2

Master: Started.
Slave 2: Started. Interval 151-200
Thread 2.2: searching in 176-200
Slave 1: Started. Interval 101-150
Thread 1.2: searching in 126-150
Thread 1.1: searching in 101-125
Thread 2.1: searching in 151-175
Slave 2: Done. Prime numbers are:  151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199,
Slave 1: Done. Prime numbers are:  101, 103, 107, 109, 113, 127, 131, 137, 139, 149,
Master: Done.
```

Include development environment information, compilation and running commands as a comment in your code.

EXAMPLE:

```
// Development environment: Lubuntu 16.04 or ITU SSH servers
// To compile: g++ -c Homework1.cpp –pthread
// To run: ./a.out interval_min interval_max np nt
// Example: ./a.out 101 200 2 2
```

Appendix

You may make use of the following code examples for reading command-line arguments and converting strings to numbers in C and C++.

- https://rosettacode.org/wiki/Command-line_arguments
- https://www.geeksforgeeks.org/converting-strings-numbers-cc/

Academic dishonesty including but not limited to cheating, plagiarism, collaboration is unacceptable and subject to disciplinary actions.