CS/SE 4348: Operating Systems Concepts
Section 004
Programming Project 1

Instructor: Neeraj Mittal

Assigned on: Wednesday, June 1, 2022
Due date: Monday, June 20, 2022

This is an individual assignment. Each student is expected to work independently and submit only their own work. Copying or using work not your own will result in disciplinary action and the suspected incident will be referred to the Office of Community Standards and Conduct for investigation!

1 Project Description

Given an array of numbers A and a number v, we use $count(A, v)$ to denote the number of occurrences of v in A. We refer to this problem as *occurrence-counting problem*. In the project, you will use concurrency using processes (not threads) to solve the occurrence-counting problem by having each process scan only a portion of the array A.

Your program will accept three *command-line* arguments: the array size s, a file name f, and the degree of concurrency d. The file f will contain array A of size s as a sequence of integers. The degree of concurrency d specifies how many processes will be used to concurrently scan the array. Your program has to be handle any errors in the command line arguments *robustly*. For example, if file f contains fewer than s entries or non-numeric values, then the program should print a reasonable error message and terminate.

When the program is run, it shall read the array A from the file f and then spawn d worker processes using fork( ) system call. The main process will then repeatedly execute the following sequence of steps until it terminates:

(a) Prompt the user to enter a number and read its value, say v.

(b) Use the worker processes to concurrently count the number of occurrences of the number v in the array A.

(c) Aggregate the output generated by all worker processes and print the result on the screen.

To terminate the program, the user will need to type "quit" (instead of a number) at the prompt. Upon reading "quit", the main process will instruct all worker processes to terminate, collect the return status of each worker process using wait( ) system call, print the total number of queries handled on the screen, and then terminate.

You must use *shared memory* to share the input array A among all processes (the main process as well as worker processes). You will also need a synchronization mechanism that allows:

(i) The main process to instruct the worker processes to either initiate the scan of their respective portions (of the array) or terminate.

(ii) A worker process to report its result back to the main process.

You can implement this mechanism using any IPC available on the department machines including, but not limited to, shared memory, pipe, message-passing or signal.

You should write your program in C/C++. Name your program as `my-count`. Also, *ensure that your program runs on one of the department machines cs1.utdallas.edu or cs2.utdallas.edu; otherwise you will not get any credit.* Any deviation from the description would result in significant penalty.

2 Grading Criteria

As such, projects will be graded with these criteria in mind:

- Solutions must adequately address the problem at hand. Specifically:
  - The solution represents a good-faith attempt to actually address the requirements for the assignment.
  - The program complies and executes.
  - The program runs correctly.
- The solution constitutes a high quality product expected of a professional. Specifically:
  - The program is easy to read and to understand, that is, it is well commented. In addition, method and variable names are meaningful, all potentially confusing/complex code is well documented.
  - The general design of the program is clear and reasonable.
  - All procedure and function headers include comments explaining what the method is supposed to do (not how it does it) and the purpose of each formal parameter. Be as precise and careful as you can be.
  - The program is robust and handles important errors and exceptions properly.

3 Submission Information

You have to submit your project through eLearning. Along with all the source files, submit the following:

(i) A Makefile to compile the program,

(ii) A README file that contains the instructions for running the compiled program, and

(iii) A PDF document `method.pdf` that describes the synchronization mechanism used to coordinate the actions of the main process with worker processes, and vice versa.

Points will be deducted if you fail to submit any of the above-mentioned files.