Operating Systems Lab (CS 470):

Lab 3: Write a program in C/C++ handling a basic menu –in text mode, with 3 items *(Problem 1*, *Problem 2*, and *Exit* item. When the user selects *Problem1* (typing 1 or 2 or 3 from the keyboard) the software should start running the problem 1. If the *Problem2* menu item is selected, the problem 2 should run. The software should exist only if the *Exit* menu is selected. The problem 1 and problem 2 solutions should run as many times the user wishes.

Problem 1: Given a sequence of numbers $x_0, x_1, \ldots, x_n^2 \in \{0,1\}$ in a binary file, where x_i represents a square matrix element in a linear fashion (concatenate the lines of the matrix for example), write a simulation process in C/C++ as follows: a) start M threads, b) each thread generates a column and a row randomly, c) considering the element location in the matrix given by the generated column and row change the value of the matrix element to 0 if its 4/8 neighbors are predominantly 0, and 1 otherwise.

Problem 2: Given a sequence of numbers, $x_0, x_1, \ldots, x_n^2 \in R$ in a text file- separated by a '\t' character, write a simulation process in C/C++ as follows: a) start M threads, b) each thread is generating an (i,j) pair randomly such as $0<=i<=j<n$, and each thread is sorting the values from x_i to x_j in the text file and write the values in the file once sorted. The threads are finished when the list is completely sorted. Each thread each time after generating the (i,j) pair should randomly select which sorting algorithm should consider. The possible algorithms should be a) quick sort, b) insert sort and c) bubble sort.

Overview

Reading and modifying (increment, decrement, assign) a shared variable/file between different threads needs extra attention due to inconsistency which might occur if the processes run in parallel. In order to avoid this, different mechanism are implemented to solve the critical section problem.

Instructions

The numbers in the text/binary file should be generated randomly. After each update the values of the matrix/elements should be written in the binary file/text file. No storage in the memory is allowed for the matrix/list elements. The simulations end when all elements in the matrix turn 0 or 1/the list of elements is completely sorted.
To lock the files consider the `fcntl()` function. To protect the critical section consider `pthread_mutex_lock()` and `pthread_mutex_unlock()` functions, respectively. The stopping criteria should run in a separate process for both problems.

Notes

- The number of elements (n) should be provided as command line argument, while the number of threads (M) should be read from the console at each time.
- Each step in the simulation process should be traceable. Printing should be provided.

Rubric

Task	**Points**
Error handling	2
Simulation running Problem1	4
Simulation running Problem2	4