Домашнее задание 5
Рахматуллин Карим БПИ207
Вариант 20

Условие задания:

20. ***Задача о программистах.*** В отделе работают три программиста. Каждый программист пишет свою программу и отдает ее на проверку другому программисту. Программист проверяет чужую программу, когда его собственная уже написана. По завершении проверки, программист дает ответ: программа написана правильно или написана неправильно. Программист спит, если не пишет свою программу и не проверяет чужую программу. Программист просыпается, когда получает заключение от другого программиста. Если программа признана правильной, программист пишет другую программу, если программа признана неправильной, программист исправляет ее и отправляет на проверку тому же программисту, который ее проверял. Создать многопоточное приложение, моделирующее работу программистов.

Требуется разработать консольное приложение с использованием библиотеки POSIX Threads языка программирования C или стандартной библиотеки языка программирования C++.

Выбранная модель - Взаимодействующие равные

Описание модели из источника: https://pro-prof.com/forums/topic/parallel-programming-paradigms

Взаимодействующие равные – модель, в которой исключен не занимающийся непосредственными вычислениями управляющий поток. Распределение работ в таком приложении либо фиксировано заранее, либо динамически определяется во время выполнения. Одним из распространенных способов динамического распределения работ является **«портфель задач»**. Портфель задач, как правило, реализуется с помощью разделяемой переменной, доступ к которой в один момент времени имеет только один процесс.
Вычислительная задача делится на конечное число подзадач. Как правило, каждая подзадача должна выполнить однотипные действия над разными данными. Подзадачи нумеруются, и каждому номеру определяется функция, которая однозначно отражает номер задачи на соответствующий ему набор данных. Создается переменная, которую следует выполнять следующей. Каждый поток сначала обращается к портфелю задач для выяснения текущего номера задачи, после этого увеличивает его, потом берет соответствующие данные и выполняет задачу, затем обращается к портфелю задач для выяснения следующего номера задачи.
Естественно должен быть предусмотрен механизм остановки процессов при исчерпывании всего множества задач, как в «производителях и потребителях».
То есть поток получает задачу из портфеля и пока задача остается не выполненной, поток ее решает, а затем снова получает задачу из портфеля.
Таким образом, процессы работают независимо, каждый со своей скоростью, синхронизация происходит с помощью портфеля задач.

Причина выбора этой модели:

В данной задаче программисты-потоки являются равноправными по отношению друг к другу, есть портфель задач, который им нужно выполнить, нет надобности в управляющем потоке.

Реализация:

first_proger - поток первого программиста, пишет программу, отправляет ее случайному программисту на проверку, проверяет программы других программистов
second_proger - поток второго программиста, пишет программу, отправляет ее случайному программисту на проверку, проверяет программы других программистов

third_proger - поток третьего программиста, пишет программу, отправляет ее случайному программисту на проверку, проверяет программы других программистов
total_tasks_cnt - портфель задач, переменная для определения кол-ва невыполненных программ
bag_mutex - мютекс для корректного обращения к портфелю задач.

После запуска программы пользователю предоставляется возможность ввести кол-во программ для каждого программиста.