Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

# ЛАБОРАТОРНА РОБОТА № 7

з дисципліни «Операційні системи»

Тема: «Потоки. Синхронізація I»

Виконала:

студентка 3 курсу

групи КС-32

Дібцева Анна Миколаївна

Перевірив: Споров О.Є.

Харків – 2020

Цель: изучить проблемы, которые могут возникнуть в следствии совместной работы нескольких потоков с одним «объектом» и рассмотреть некоторые пути их решения: мьютексы и блокировки чтения-записи.

# ХОД РАБОТЫ

**Задание 1**

Требуется написать программу, которая создаёт заданное количество пар потоков-потомков, один из которых заданное количество раз увеличивает значение переменной на 1, а другой – уменьшает.

Количество потоков и операций можно передавать при помощи опций, их обработкой занимается функция options().

Потоки реализованы в потоковых функциях addThreadFunc и negThreadFunc. В главной функции программы они создаются в цикле по очереди при помощи функции pthread\_create(). После этого, главный поток так же их циклически ожидает при помощи функции pthread\_join(), далее выводится финальное значение переменной

Рисунок 1 - Последствия условий гонок

**Задание 2**

Требуется посчитать численное значение определённого интеграла на указанном промежутке с указанной точностью, распределив интегральную область между потоками.

При запуске программа из командной строки получает количество потоков, диапазон интегрирования и точность для расчётов. Далее инициализируется мьютекс функцией pthread\_mutex\_init() (динамическая реализация мьютекса) и создается массив структур, которым будет осуществляться передача аргументов потоку. После этого диапазон интергирования разбивается между потоками, каждый из которых высчитывает свою часть и прибавляет ее к общему результату, блокируя для этого мьютекс

Для вычисления интеграла каждый поток вызывает функцию integral(), которая считает его по составному методу средних прямоугольников, разбивая данную область на еще меньшие интервалы. Количество интервалов постоянно удваивается, пока не будет достигнута указанная точность.

Пример выполнения программы для функции, заданной в условиях задания:

Рисунок 2 - вычисление определённого интеграла на промежутке

**Задание 3**

В задании 3 рассматривается другой механизм синхронизации потоков – блокировки чтения/записи.

Блокировки чтения/записи используются так же, как и мьютексы, за исключением того, что у них есть три режима работы – разблокированный, заблокированный на чтение и заблокированный на запись, что позволяет более практично распределять ресурсы.

Реализация: в программе существует массив заданного размера, и несколько потоков, которые его обслуживают: потоки-писатели, которые генерируют псевдослучайное число в указанном диапазоне и записывают его в случайную ячейку массива (установив блокировку на запись), потоки-читатели, которые считывают информацию из массива, устанавливая блокировку на чтение, и один отсоединенный поток-информатор, который выводит состояние массива (для этого он так же устанавливает блокировку на чтение). Основной поток создаёт одного за другим все эти потоки, после чего ожидает из завершения (кроме отсоединённого, который завершается при помощи отмены потока тогда, когда основной поток завершит все остальные).

Переменная блокирововк чтения-записи инициализируется аналогично мьютексам: функцией pthread\_rwlock\_init(), а уничтожается - функцией pthread\_rwlock\_destroy()

Читателям и писателям была дана случайная задержка перед выполнением их работы, а информатору –задержка в 1 секунду после каждого информационного сообщения.

Рисунок 3 - пример работы с блокировками чтения/записи

# ВЫВОДЫ

Таким образом, в ходе выполнения данной работы были рассмотрены возможности синхронизации потоков при помощи мьютексов и блокировок чтений/записи. Соответствующие исходные файлы, исполняемые файлы и скрипты сборки помещены в архив.