**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет "Высшая школа экономики"**

МИКРОПРОЕКТ №2

**Пояснительная записка**

По курсу «Архитектура вычислительных систем»

|  |  |  |
| --- | --- | --- |
| ФИО | Группа | Дата |
| Устюхин Андрей Олегович | БПИ192 | 12.12.2020 |

**Москва – 2020 г.**

**Вариант 3**

**1. Текст задания**

3. Задача о читателях и писателях. Базу данных разделяют два типа процессов – читатели и писатели. Читатели выполняют транзакции, которые просматривают записи базы данных, транзакции писателей и просматривают и изменяют записи. Предполагается, что в начале БД находится в непротиворечивом состоянии (т.е. отношения между данными имеют смысл). Каждая отдельная транзакция переводит БД из одного непротиворечивого состояния в другое. Для предотвращения взаимного влияния транзакций процесс-писатель должен иметь исключительный доступ к БД. Если к БД не обращается ни один из процессов-писателей, то выполнять транзакции могут одновременно сколько угодно читателей. Создать многопоточное приложение с потоками-писателями и потоками-читателями. Реализовать решение, используя семафоры.

**2. Применяемые расчетные методы**

Для решения данной задачи применялась библиотека POSIX Thread. Также для решения была выбрана модель многопоточных приложений “Производители и потребители”. При такой модели одни из потоков «производят» данные, другие их «потребляют», что и требуется для данной задачи, так как Писатели производят большое целое число и записывают его в Базу данных, а читатели как раз работают со “строчками” (элементами) в этой Базе данных. Поэтому в программе представлены 2 главные функции, одна для писателей, а другая для читателей. В каждой из функций происходит работа с соответствующими потоками (потоками писателей или читателей) и обрабатываются условия работы тех или иных процессов, в зависимости от количества возможных элементов в Базе данных – по умолчанию это число равно 15 (это придумано больше для добавления логики к процессам чтения и записи). Элемент для Базы данных формируется случайным образом, но как большое целое число (можно было помещать туда и строки, но я решил выбрать как всегда числа). Наконец, важной частью реализации стало использование семафоров и двоичных семафоров. Это сделано для того, чтобы блокировать выполнение определенных потоков в момент, когда это требуется. В особенности, в задаче встречаются критические секции, и чтобы операции записи в общие данные, которые могут быть изменены или открыты для чтения несколькими потоками, проходили успешно, нужно было также использовать семафоры, мьютексы (двоичный семафор) и условные переменные . Вывод результата, возможно, сделан не сверх подробно, но я выводил все действия, которые производились писателями или читателями, с указанием значения строки в Базе данных и ее координаты в этой Базе данных.

Примечание: так как процесс получается бесконечный, то я использовал обычный пересчет кол-ва выводимых строк, и когда это кол-во доходит до заданного, программа завершает работу. Можно было это также сделать с помощью функции sleep() из unistd.h, но на ОС Windows с этим возникают проблемы, поэтому я сделал через число итераций.

**3. Список используемых источников**

**-** [**http://www.softcraft.ru**](http://www.softcraft.ru)

**-** [**https://ru.wikipedia.org/wiki/POSIX\_Threads**](https://ru.wikipedia.org/wiki/POSIX_Threads)