Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

Департамент программной инженерии

Пояснительная записка к микропроекту №2

По дисциплине

«Архитектура вычислительных систем»

3 вариант

Работу выполнил:

Студент группы БПИ-194, Аникеев Егор Васильевич

Преподаватель:

Легалов Александр Иванович

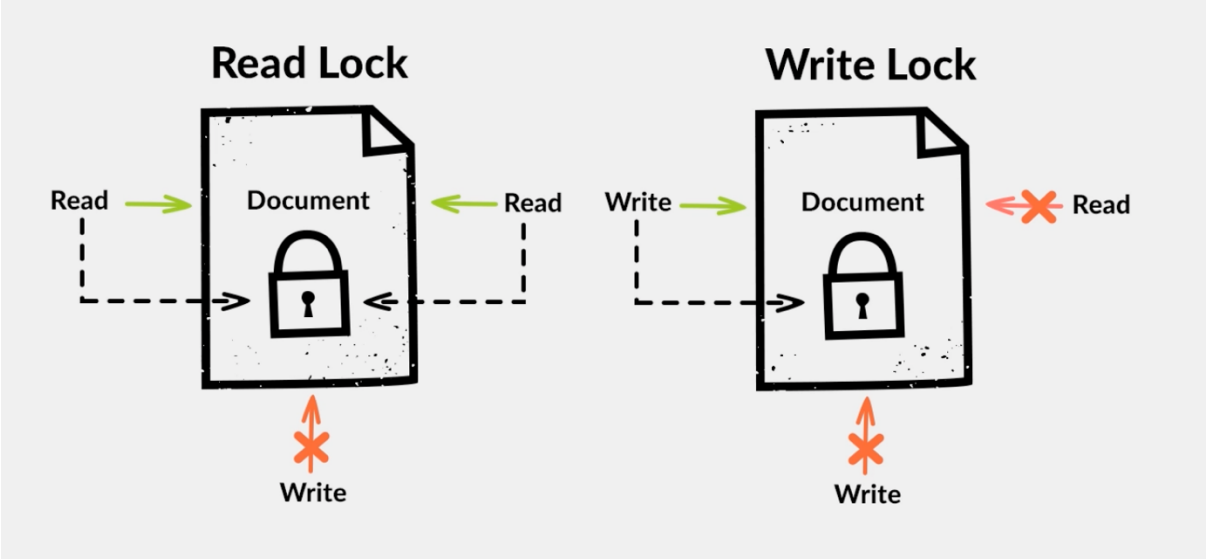
Москва 2020

Задание

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.

Решение

Сперва определим проблему задачи о читателях и писателях: есть данные, которые либо изменяются писателями, либо читатели используют эти данные без их изменения. Примитивным решением этого может быть просто ограничивать доступ к данным любого потока, но это не оптимально, потому что потоки-читатели могут одновременно обратиться к данным без возникновения конфликтов, а проблемы при множественном доступе могут возникнуть только если с данными работает поток-писатель. Решение такой задачи имеет описывает следующая картинка:



Для ограничения работы потоков с данными используем семафор со следующими правилами:

- Если с данными хочет работать поток-читатель и с данными работают только другие потоки-читатели и не работают потоки-писатели, то разрешаем пришедшему потоку получить данные.
- Если с данными хочет работать поток-читатель и с данными работает поток-писатель, то ставим пришедший поток в режим ожидания, пока поток-писатель не освободит данные.

- Если с данными хочет работать поток-писатель и с данными работает другой поток-писатель или хотя бы один поток-читатель, то ожидаем пока данные освободятся от других потоков. Иначе разрешаем пришедшему потоку доступ к данным.

В качестве базы данных выберем информацию о студентах, которые поступили в вуз: Имя и год рождения. Потоки-читатели будут выводить информацию о случайных студентах, а потоки-писатели будут записывать в базу данных поступивших студентов.

Для решения задачи воспользуемся библиотекой Posix thread и ее механизмами синхронизации потоков для реализации семафора: pthread_mutex_t - для ограничения доступа потоков к общим данным (количество активных поток читателей и писателей) и pthread_cond_t - для определения состояния, в котором находится поток. Также воспользуемся стандартной библиотекой потоков для запуска потоков и синхронизации вывода данных в файлы.

Для имитации продолжительности работы потоков будем отправлять поток-писатель в сон на 1000мс, а поток-читатель на 200мс, чтобы увидеть, что читатели в это время могут обратиться к данным.

Формат входных данных

Входными данными программы являются следующие аргументы командной строки:

1. Количество потоков-читателей;
2. Количество о дополнительных потоков-писателей(изначально в программе работает 2 потока);
3. Расположение файла, в который будет записываться информация о работе потоков-писателей;
4. Расположение файла, в который будет записываться информация о работе потоков-читателей.

Формат выходных данных

В файл, который был указан третьим аргументом командной строки, выводится информация о работе потоков-писателей в формате <время с начала работы программы>: <информация о начале или конце записи данных>.

В файл, который был указан четвертым аргументом командной строки, выводится информация о работе потоков-читателей в формате <время с начала работы программы>: <имя студента> - <год рождения студента>.

В консоль выводится итоговая база данных после всех преобразований.

Изображения тестов

1. Если входные данные некорректны, то программа выведет следующее сообщение.

```
You should write 4 command line arguments: firtst - count of readers, second - count of writers,
    third - file for writer info and fourth - file for reader info
```

2. При 10 потоках-читателей и 5 потоках-писателей результат работы будет иметь следующий вид:

```
1.001:Egor-2000
1.001:Egor-2000
1.001:dj-1975
1.002:Egor-2000
1.002:Artem-1999
1.002:dj-1975
1.002:Maxim-2001
1.002:Artem-1999
1.002:dj-1975
1.002:Egor-2000
```

Информация о потоках-читателях

```
0:STARTING UPDATE INFO ABOUT STUDENT dj
1:ENDING UPDATE INFO ABOUT STUDENT dj
1.205:STARTING UPDATE INFO ABOUT STUDENT cpdjk
2.206:ENDING UPDATE INFO ABOUT STUDENT cpdjk
2.207:STARTING UPDATE INFO ABOUT STUDENT mkee
3.213:ENDING UPDATE INFO ABOUT STUDENT mkee
3.213:STARTING UPDATE INFO ABOUT STUDENT luqh
4.215:ENDING UPDATE INFO ABOUT STUDENT luqh
4.216:STARTING UPDATE INFO ABOUT STUDENT kpnap
5.223:ENDING UPDATE INFO ABOUT STUDENT kpnap
5.223:STARTING UPDATE INFO ABOUT STUDENT wybcd
6.226:ENDING UPDATE INFO ABOUT STUDENT wybcd
6.226:STARTING UPDATE INFO ABOUT STUDENT yi
7.232:ENDING UPDATE INFO ABOUT STUDENT yi
```

Информация о потоках-писателях

```
List of students
Artem: 1999
Egor: 2000
Maxim: 2001
cpdjk: 1979
dj: 1975
kpnap: 1979
luqh: 1993
mkee: 1977
wybcd: 1998
yi: 1972
```

Вывод в консоль

3. При 1 потоке-писателе и 2 потоках-читателях:

```
1.005:Maxim-2001
1.005:Artem-1999
```

Информация о потоках-читателях

```
0:STARTING UPDATE INFO ABOUT STUDENT dj
1.004:ENDING UPDATE INFO ABOUT STUDENT dj
1.209:STARTING UPDATE INFO ABOUT STUDENT yi
2.211:ENDING UPDATE INFO ABOUT STUDENT yi
2.211:STARTING UPDATE INFO ABOUT STUDENT cpdjk
3.213:ENDING UPDATE INFO ABOUT STUDENT cpdjk
```

Информация о потоках-писателях

```
List of students
Artem: 1999
Egor: 2000
Maxim: 2001
cpdjk: 1979
dj: 1975
yi: 1972
```

Вывод в консоль

Список используемых источников

1. Курс на https://stepik.org/ "Многопоточность в Swift" глава 1.7 "Read write lock". (Использовался для понимания проблемы читателей-писателей).
2. "Введение в POSIX threads" http://ccfit.nsu.ru/arom/data/PP_ICaG/03_pthreads_ru.pdf