



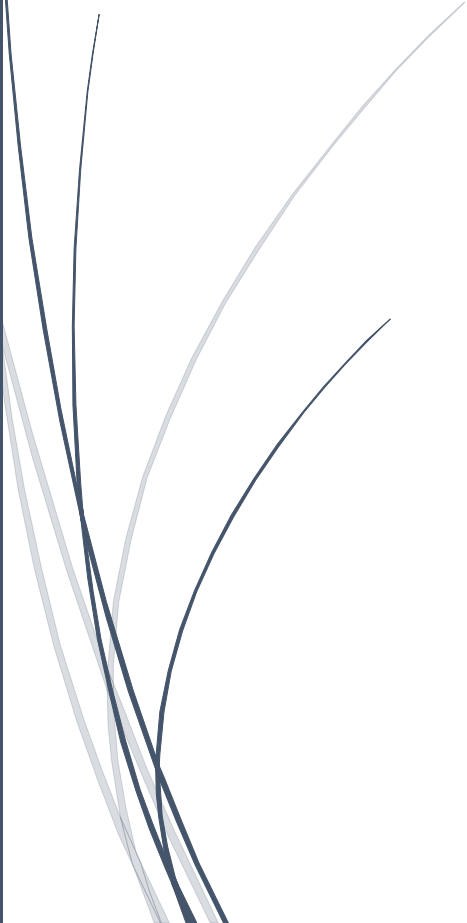
12.12.2020

Пояснительная записка

Микропроект 2 по ABC

НИУ ВШЭ

Профессор департамента
программной инженерии
факультета компьютерных наук
Легалов Александр Иванович



Бакытбек уулу Нуржигит -
БПИ197, ВАРИАНТ 3

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 4 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.

Описание проекта

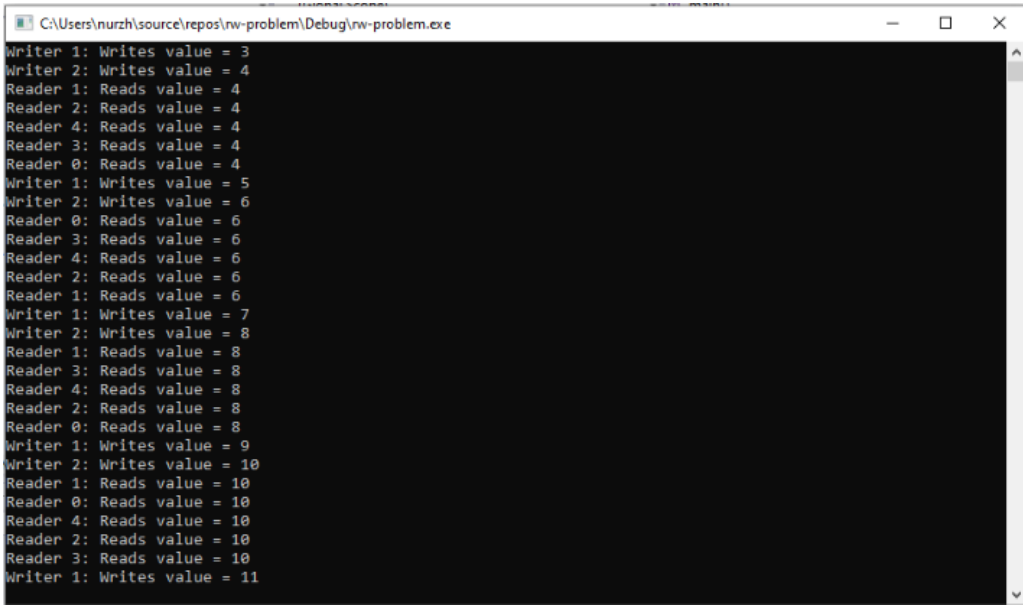
Проект разработан на языке C++. Имитируется процесс работы сервера с читателями и писателями.

Основные алгоритмы

Нужно было решить проблему конкуренции доступа к данным между читателями и писателями. Решение - блокировка новых потоков-читателей, появившихся после блокировки потока-писателя.

Альтернативный вариант – отдать предпочтение потокам-читателям (т. е. новые потоки читатели могут начинать свою работу, не обращая внимания на заблокированный процесс писатель), однако в этом случае блокировка потока-писателя может продолжаться бесконечно долго.

Пример выполнения программы



```
C:\Users\nurzh\source\repos\nw-problem\Debug\nw-problem.exe
Writer 1: Writes value = 3
Writer 2: Writes value = 4
Reader 1: Reads value = 4
Reader 2: Reads value = 4
Reader 4: Reads value = 4
Reader 3: Reads value = 4
Reader 0: Reads value = 4
Writer 1: Writes value = 5
Writer 2: Writes value = 6
Reader 0: Reads value = 6
Reader 3: Reads value = 6
Reader 4: Reads value = 6
Reader 2: Reads value = 6
Reader 1: Reads value = 6
Writer 1: Writes value = 7
Writer 2: Writes value = 8
Reader 1: Reads value = 8
Reader 3: Reads value = 8
Reader 4: Reads value = 8
Reader 2: Reads value = 8
Reader 0: Reads value = 8
Writer 1: Writes value = 9
Writer 2: Writes value = 10
Reader 1: Reads value = 10
Reader 0: Reads value = 10
Reader 4: Reads value = 10
Reader 2: Reads value = 10
Reader 3: Reads value = 10
Writer 1: Writes value = 11
```

Текст программы

```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>
#include <thread>
#include <chrono>

// Бакытбек уулу Нуржигит, БПИ197, вариант 2

sem_t mutex_sema; // Для блокировки изменения счетчика количества активных потоков-читателей
sem_t write_sema; // Для блокировки для изменения данных
int data = 0;
int readers_count = 0; // - переменная-счетчик количества активных потоков-читателей

//стартовая функция потоков - читателей
void* Reader(void* param)
{
    int pNum;
    pNum = *((int*)param);

    while (true) {
        std::this_thread::sleep_for(std::chrono::milliseconds(300));
        // Блокировка доступа к переменной readers_count
        sem_wait(&mutex_sema);
        // Изменение счетчика активных читателей
        ++readers_count;
        if (readers_count == 1) {
            // Блокировка доступа к хранилищу (если поток-читатель первый)
            sem_wait(&write_sema);
        }
        // Снятие блокировки доступа к readers_count
        sem_post(&mutex_sema);
        // Выполнение операции чтения
        printf("Reader %d: Reads value = %d\n", pNum, data);

        // Блокировка доступа к переменной readers_count
        sem_wait(&mutex_sema);
        // Изменение счетчика активных читателей
        --readers_count;
```

```

        // Снятие блокировка доступа к хранилищу (если завершается последний поток-читатель)
        if (readers_count == 0) {
            sem_post(&write_sema);
        }
        // Снятие блокировки доступа к readers_count
        sem_post(&mutex_sema);
    }

    return nullptr;
}

//стартовая функция потоков - писателей
void* Writer(void* param)
{
    int pNum;

    pNum = *((int*)param);

    while (true) {
        // Блокировка доступа к хранилищу
        sem_wait(&write_sema);

        // Изменение данных
        std::this_thread::sleep_for(std::chrono::seconds(3));

        data++;

        printf("Writer %d: Writes value = %d\n", pNum, data);

        // Снятие блокировки доступа к хранилищу
        sem_post(&write_sema);
    }

    return nullptr;
}

int main() {

    int i;

    //инициализация семафоров
    sem_init(&mutex_sema, 0, 1);
    sem_init(&write_sema, 0, 1);
    //запуск писателей и читателей
    pthread_t threadW[2];
    int writers[2];
    pthread_t threadR[4];

```

```

int readers[4];

for (i = 0; i < 4; i++) {

    if (i < 2) {
        writers[i] = i + 1;
        pthread_create(&threadW[i], nullptr, Writer, (void*)(writers + i));
    }
    readers[i] = i + 1;
    pthread_create(&threadR[i], nullptr, Reader, (void*)(readers + i));
}

//пусть главный поток тоже будет читателем
int mNum = 0;
Reader((void*)&mNum);
return 0;
}

```

Список использованной литературы:

- 1) <http://softcraft.ru/edu/comparch/practice/thread/02-sync/readwriters01/main.cpp>
- 2) <https://habr.com/ru/post/261273/>
- 3) http://hpc-education.ru/files/lectures/2011/gergel/gergel_2011_lecture02.pdf