Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Навчально-науковий комплекс «Інститут прикладного системного аналізу»
Кафедра системного проектування

Лабораторна робота № 1
з курсу «Операційні системи»

Виконала студентка
Групи ДА-02
Лесечко О. Р

Київ 2022

Завдання:

Розробити вирішення класичної задачі синхронізації про «Читачі-письменники» за допомогою взаємо виключення (WIN32).

Опис завдання:

Необхідно забезпечити монопольний доступ до бази даних процесу, що виконує запис даних, та множинний доступ до процесів, що виконують читання даних. Завдання може бути сформульована з наданням пріоритету як процесу-письменнику, і процесам-читачам.

Код програми:

```
#define _CRT_SECURE_NO_WARNINGS // для работы localtime

#include <iostream>
#include <thread>
#include <mutex>
#include <condition_variable>
#include <random>
#include <ctime>



/*  это примитив синхронизации, который можно использовать
    для блокировки потока или нескольких потоков одновременно,
    пока другой поток не изменит общую переменную(условие) и не
    уведомит об этом condition_variable.    */
std::condition_variable cv;

/*  общая переменная для потоков  */
bool ready = false;
/*  общая переменная для потоков  */
bool processed = false;

/*  этот мьютекс используется для условной переменной cv    */
std::mutex m;

/*  класс Runnable, служит общим представлением читателя и писателя */
class Runnable {
public:
    /*  это циклический процесс, и каждый раз,
        проходя свой цикл, он производит определенную порцию информации,
        которую должен обработать читатель  */
    void read() {
        while (true) {
            std::this_thread::sleep_for(std::chrono::milliseconds(5000));
            {
                // ждать, пока write() отправит данные
                std::unique_lock<std::mutex> lk(m);
                cv.wait(lk, [] {return ready; });
                lk.unlock();
```

```
                // количество читателей увеличилось на 1
                readcnt_++;

                tm* time = nowTime();
                std::cout << "> id thread = " << std::this_thread::get_id() << "\n";
                std::cout << "  Time: " << time->tm_hour << ":" << time->tm_min << ":" <<
time->tm_sec << "\n";
                std::cout << "  The reader begins to read\n\n";

                // время работы читателя
                int duration = generationRandomSleep(5000, 10000);
                std::this_thread::sleep_for(std::chrono::milliseconds(duration));

                time = nowTime();
                std::cout << "> id thread = " << std::this_thread::get_id() << "\n";
                std::cout << "  Time: " << time->tm_hour << ":" << time->tm_min << ":" <<
time->tm_sec << "\n";
                std::cout << "  The reader ends to read\n\n";

                // количество читателей уменьшить на 1
                readcnt_--;
            }
            // если читателей 0, разрешить работу писателю
            if (readcnt_ == 0) {
                processed = true;
                cv.notify_all();
            }
        }
    }
    /*  это циклический процесс, и каждый раз,
        проходя свой цикл, он производит определенную порцию информации,
        которую должен обработать писатель  */
    void write() {
        while (true) {
            {
                // ждать, пока read() отправит данные
                std::unique_lock<std::mutex> lk(m);
                cv.wait(lk, [] {return processed; });
                processed = false;

                std::cout << "> id thread = " << std::this_thread::get_id() << "\n";
                std::cout << "  Time: " << nowTime()->tm_hour << ":" << nowTime()->tm_min
<< ":" << nowTime()->tm_sec << "\n";
                std::cout << "  The writer begins to write\n\n";

                // время работы писателя
                int duration = generationRandomSleep(6000, 10000);
                std::this_thread::sleep_for(std::chrono::milliseconds(duration));

                std::cout << "> id thread = " << std::this_thread::get_id() << "\n";
                std::cout << "  Time: " << nowTime()->tm_hour << ":" << nowTime()->tm_min
<< ":" << nowTime()->tm_sec << "\n";
                std::cout << "  The writer ends to write\n\n";
            }
            // разрешить работу читателям
            ready = true;
            cv.notify_all();
        }
```

```
    }
private:
    /*  счетчик читателей   */
    int readcnt_ = 0;
    /*  генерация случайного числа задержки

        @param min минимальное число с какого рандомится число
        @param max максимальное число до какого рандомится число

        @returns случайное число задержки */
    int generationRandomSleep(int min, int max) {
        static std::mt19937 rnd(std::time(nullptr));
        return std::uniform_int_distribution<>(min, max)(rnd);
    }
    /*  фунція для знаходження точного часу */
    tm* nowTime()
    {
        time_t now = time(NULL);
        struct tm* tm_struct = localtime(&now);
        return tm_struct;
    }
};

int main() {
    Runnable runnable;

    std::thread t1(&Runnable::read, &runnable);
    std::thread t2(&Runnable::read, &runnable);
    std::thread t3(&Runnable::write, &runnable);

    processed = true;
    cv.notify_one();

    t1.join();
    t2.join();
    t3.join();

    return 0;
}
```

Результат роботи:

На скріншоті ми бачимо роботу програми.

- Спочатку в 17:20:17 з функції write() з’являється напис, що письменник почав роботу.
- В 17:20:24 ми бачимо, що письменник завершив роботу и 2 потоки читача в той же час почали роботу.
- В 17:20:30 читати завершив потік з id 15196
- В 17:20:33 читати завершив потік з id 11180, і так як письменник вже готовий працювати, а всі читачі завершили свою роботу, письменник починає роботу

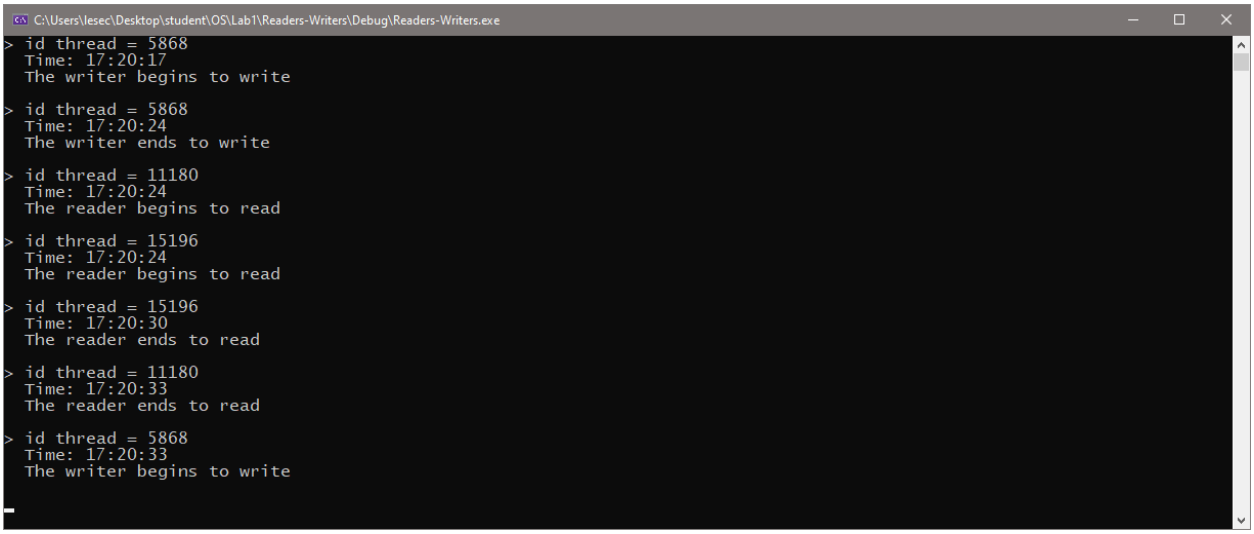

Рис. 1 Результат роботи

Висновки:

У ході виконання даної лабораторної роботи я реалізовувала вирішення проблемної задачі про читачів та письменників, що ставила перед нами завдання про синхронізацію потоків, тобто запобігання одночасному виконанню деякого критичного набору операцій. Головним було використання у контексті задачі «події», що дозволяли б доступ до так званої бібліотеки або тільки читачам, або тільки одному письменнику. Результат роботи описаний вручну, а до коду додані коментарі.