

Правительство Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования «Национальный

исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

Департамент программной инженерии

Отчет к домашнему заданию По дисциплине

«Архитектура вычислительных систем»

Работу выполнил:

Студент группы БПИ-191 Рычков К.П.

Москва 2020

Задание

Преподаватель проводит экзамен у группы студентов. Каждый студент заранее знает свой билет и готовит по нему ответ. Подготовив ответ, он передает его преподавателю. Преподаватель просматривает ответ и сообщает студенту оценку. Требуется создать многопоточное приложение, моделирующее действия преподавателя и студентов. При решении использовать парадигму «клиент-сервер».

Модель

Клиенты и серверы – способ взаимодействия неравноправных потоков. Клиентский поток запрашивает сервер и ждет ответа. Серверный поток ожидает запроса от клиента, затем действует в соответствии с поступившим запросом.

Решение

При разработке были использованы события для синхронизации действий между потоками сервера и клиентом.

Программой была смитирована следующая ситуация: студенты входят в аудиторию друг за другом каждый 1-1,5 секунды и берут какой-то билет после чего начинают свою подготовку к экзамену (2 – 5 секунд) после чего ждут пока преподаватель будет готов их принять и сдают ему свой ответ. Преподаватель принимает ответ студента на протяжении 1-3 секунд и ставит ему оценку от 0 до 10, после чего передает оценку студенту и студент выводит информацию о своей оценке на экран.

Код программы

```
#include <iostream>
#include <windows.h>
#include <vector>
#include <ctime>
#include <omp.h>

bool endOfExam = false;    // Флаг остановки экзамена

/**
 * Функция реализующая поток студента
 * @param param параметр с данными о студенте
 * @return код завершения потока
 */
void Student(int param) {
    HANDLE teacherReady, dataReady, serverAnswer;

    //Открываем события до тех пор, пока все из них не будут проинициализированы
    while (teacherReady == nullptr || dataReady == nullptr || serverAnswer == nullptr)
    {
        teacherReady = OpenEvent(EVENT_ALL_ACCESS, FALSE, LPCSTR("teacherReady"));
    }
    //Готовность учителя принимать ответ
    dataReady = OpenEvent(EVENT_ALL_ACCESS, FALSE, LPCSTR("dataReady")); //Переданы
    //ли данные студентом в память
    serverAnswer = OpenEvent(EVENT_ALL_ACCESS, FALSE, LPCSTR("serverAnswer"));
    //Учитель дал ответ студенту
```

```

        Sleep(10);
    }

    //Создаем общую память для общения студентов и преподавателя
    HANDLE mapFile = OpenFileMapping(FILE_MAP_ALL_ACCESS, FALSE, LPCSTR("MyShared"));
    while (mapFile == nullptr) {
        Sleep(10);
        mapFile = OpenFileMapping(FILE_MAP_ALL_ACCESS, FALSE, LPCSTR("MyShared"));
    }
    int *data = (int*)MapViewOfFile(mapFile, FILE_MAP_READ | FILE_MAP_WRITE, 0, 0, 0);

    int studNumber = (DWORD) param; //получаем параметр переданный в поток
    srand(time(0));

#pragma omp critical
    std::cout << "Student " << studNumber << ": take a ticket and prepare his answer" <<
    std::endl;
    Sleep(rand() % 3000 + 2000); //Студент готовится к ответу

    WaitForSingleObject(teacherReady, INFINITE); //Студент ждет готовности преподавателя
    принимать его ответ
#pragma omp critical
    std::cout << "Student " << studNumber << " start to answer." << std::endl;

    data[0] = studNumber; //Студент передает свой номер преподавателю
    SetEvent(dataReady); //Дает понять преподавателю, что готов отвечать

    WaitForSingleObject(serverAnswer, INFINITE); //Ожидает ответа от преподавателя
#pragma omp critical
    std::cout << "Student " << studNumber << ": have a mark " << data[1] << std::endl;
}

/**
 * Реализует преподавателя
 * @param param
 * @return
 */
void Teacher(int countStudents) {
    //События преподавателя для синхронизации
    HANDLE teacherReady = CreateEvent(nullptr, FALSE, FALSE, LPCSTR("teacherReady"));
    //Готовность учителя принимать ответ
    HANDLE dataReady = CreateEvent(nullptr, FALSE, FALSE, LPCSTR("dataReady"));
    //Переданы ли данные студентом в память
    HANDLE serverAnswer = CreateEvent(nullptr, FALSE, FALSE, LPCSTR("serverAnswer"));
    //Учитель дал ответ студенту

    //Создаем общую память для общения студентов и преподавателей
    HANDLE mapFile = CreateFileMapping(INVALID_HANDLE_VALUE, nullptr, PAGE_READWRITE, 0,
    sizeof(int) * 2, LPCSTR("MyShared"));
    int *data = (int*) MapViewOfFile(mapFile, FILE_MAP_READ | FILE_MAP_WRITE, 0, 0, 0);

    int checkedStudents = 0;
    //Пока экзамен не окончен учитель принимает ответы студентов
    while (checkedStudents < countStudents) {
        SetEvent(teacherReady); //Устанавливает готовность учителя принимать ответы
        WaitForSingleObject(dataReady, INFINITE); //Ожидает пока студент предаст свои
        данные в память
#pragma omp critical
        std::cout << "Teacher: start to receive answer from student " << data[0] <<
        std::endl;
        Sleep(rand() % 2000 + 1000); //Принимается ответ студента

        data[1] = rand() % 11; //Определяется оценка
#pragma omp critical

```

```

        std::cout << "Teacher: set a mark " << data[1] << " to student " << data[0] <<
std::endl;

        SetEvent(serverAnswer); //Преподаватель дает понять студенту, что закончил
проверку
        checkedStudents++;
    }
}

/**
 * Считывает число
 * @param minVal максимальное значение вводимого числа
 * @param maxVal минимальное значение вводимого числа
 * @param str название вводимых данных
 * @return считанное число
 */
int ReadNumber(int minVal, int maxVal, std::string str) {
    int number;
    std::cout << "Input count of students (" << minVal << ";" << maxVal << "):";
    std::cin >> number;
    while (number < minVal || number > maxVal) {
        std::cout << "Incorrect input..." << std::endl;
        std::cout << "Input " << str << " again:";
        std::cin >> number;
    }
    return number;
}

int main() {
    srand(time(0));

    int studentsCount = ReadNumber(1, 100, "count of students"); //Считываем количество
студентов
    auto *students = new HANDLE[studentsCount];
#pragma omp parallel num_threads(studentsCount + 1)
    {
        if (omp_get_thread_num() > 0) {
            int i = omp_get_thread_num();
            Student(i);
        } else
            Teacher(studentsCount);
    }
    WaitForMultipleObjects(studentsCount, &students[0], TRUE, INFINITE);
    studentsCount = 0;
    endOfExam = true; //Заканчиваем экзамен
    delete[] students;
    return 0;
}

```

Тестирование

```
Input count of students (1;100):10
Student 1: take a ticket and prepare his answer
Student 2: take a ticket and prepare his answer
Student 3: take a ticket and prepare his answer
Student 4: take a ticket and prepare his answer
Student 5: take a ticket and prepare his answer
Student 6: take a ticket and prepare his answer
```

Рисунок 2 – Создание студентов.

```
Student 1 start to answer.
Teacher: start to receive answer from student 1
Student 7: take a ticket and prepare his answer
Student 8: take a ticket and prepare his answer
Teacher: set a mark 9 to student 1
Student 1: have a mark 9
Student 2 start to answer.
Teacher: start to receive answer from student 2
Student 9: take a ticket and prepare his answer
Teacher: set a mark 1 to student 2
Student 3 start to answer.
Student 2: have a mark 1
Teacher: start to receive answer from student 3
Student 10: take a ticket and prepare his answer
Teacher: set a mark 5 to student 3
Student 3: have a mark 5
Student 4 start to answer.
Teacher: start to receive answer from student 4
Teacher: set a mark 10 to student 4
Student 4: have a mark 10
Student 5 start to answer.
Teacher: start to receive answer from student 5
Teacher: set a mark 0 to student 5
```

Рисунок 3 – Преподаватель принимает экзамен

```
Student 5 start to answer.
Teacher: start to receive answer from student 5
Teacher: set a mark 0 to student 5
Student 5: have a mark 0
Student 6 start to answer.
Teacher: start to receive answer from student 6
Teacher: set a mark 7 to student 6
Student 6: have a mark 7
Student 7 start to answer.
Teacher: start to receive answer from student 7
Teacher: set a mark 8 to student 7
Student 7: have a mark 8
Student 8 start to answer.
Teacher: start to receive answer from student 8
Teacher: set a mark 7 to student 8
Student 9 start to answer.
Student 8: have a mark 7
Teacher: start to receive answer from student 9
Teacher: set a mark 7 to student 9
Student 9: have a mark 7
Student 10 start to answer.
Teacher: start to receive answer from student 10
Teacher: set a mark 2 to student 10
Student 10: have a mark 2

Process finished with exit code 0
```

Рисунок 4 – Завершение экзамена

```
Input count of students (1;100):101
Incorrect input...
Input count of students again:-10
Incorrect input...
Input count of students again:1
Student 1: take a ticket and prepare his answer
Student 1 start to answer.
Teacher: start to receive answer from student 1
Teacher: set a mark 9 to student 1
Student 1: have a mark 9

Process finished with exit code 0
```

Рисунок 5 – Некорректные данные

Список используемых источников

1. Википедия (2020) «Клиент-сервер» (https://ru.wikipedia.org/wiki/Клиент_сервер).
2. Habr (2020) «Клиент-сервер шаг — за — шагом, от однопоточного до многопоточного (Client-Server step by step)» (<https://habr.com/ru/post/330676/>).
3. SoftCraft «Многопоточное программирование. OpenMP» (<http://www.softcraft.ru/edu/comparch/practice/thread/03-openmp/>)
4. Coursera «Технология OpenMP, особенность и ее компоненты» (<https://ru.coursera.org/lecture/parallelnoye-programmirovaniye/2-2-tiekhnologhiia-openmp-osobiennosti-i-ieie-komponenty-vhScx>)
5. Coursera «Задание параллельной области и опции, влияющие на ее выполнение» (<https://ru.coursera.org/lecture/parallelnoye-programmirovaniye/2-3-zadaniie-paralliel-noi-oblasti-i-optsii-vliiaiushchiie-na-ieie-vypolnieniie-Ywk5H>)
6. Coursera «Модель памяти. Классы переменных в OpenMP» (<https://ru.coursera.org/lecture/parallelnoye-programmirovaniye/2-4-modiel-pamiati-klassy-pieriemiennykh-v-openmp-OYrCJ>)
7. Microsoft «OpenMP Directives» (<https://docs.microsoft.com/ru-ru/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160#parallel>)