

4Правительство Российской Федерации

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

Факультет компьютерных наук

Пояснительная записка к домашнему заданию 4

Вариант 19

По дисциплине

“Архитектура вычислительных систем”

Работу выполнил

Студент группы БПИ-194

_____ И.С. Попов

подпись, дата

Работу проверил

_____ А.И. Легалов

подпись, дата

Москва 2020

Содержание

Постановка задачи	3
Описание алгоритма	3
Пример работы программы	3
Приложение	5
Список использованной литературы.....	7

Постановка задачи

Вариант 19:

У одной очень привлекательной студентки есть N поклонников. Традиционно в день св. Валентина очень привлекательная студентка проводит романтический вечер с одним из поклонников. Счастливый избранник заранее не известен. С утра очень привлекательная студентка получает N «валентинок» с различными вариантами романтического вечера. Выбрав наиболее заманчивое предложение, студентка извещает счастливчика о своем согласии, а остальных – об отказе. Требуется создать многопоточное приложение, моделирующее поведение студентки. При решении использовать парадигму «клиент-сервер» с активным ожиданием.

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.

Описание алгоритма

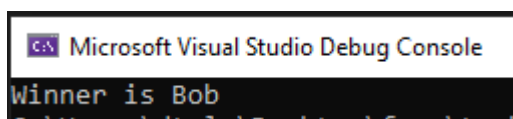
Было реализовано 2 класса, имитирующие пользователя (ClientFan) и сервер (ServerStudentka). В методе main() генерируется N поклонников, а так же их письма. Создается массив (вектор) из N потоков, каждый из потоков отвечает за письмо каждого поклонника (запросы клиентов в модели Клиент-сервер). Далее вызывается серверный поток который начинает обрабатывать все письма. Серверный поток вызывается параллельно потокам клиентов (не получилось реализовать по другому используя open mp). В классе есть поле, которое хранит число необработанных писем. Вероятность того, что письмо понравится студентке – $1/\text{count}$, count – число необработанных писем. Если письмо понравилось то выбирается счастливчик, изменяется логическая переменная ответственная за продолжение работы потоков и все потоки завершаются.

Пример работы программы

В силу простоты алгоритма и одинакового принципа работы на разном количестве входных данных было произведено несколько тестов на разных размерах (на одном из которых было сделано несколько тестов, чтобы убедиться, что студентка выбирает поклонника случайно).

1 Поклонник:

```
const size_t N = 1;
vector<ClientFan> fans =
{
    ClientFan("Bob", "I love you!"),
};
ServerStudentka* server = new ServerStu
```



Microsoft Visual Studio Debug Console

Winner is Bob

3 Поклонника:

```
const size_t N = 3;
vector<ClientFan> fans =
{
    ClientFan("Bob", "I love you!"),
    ClientFan("Mike", "I <3 you"),
    ClientFan("Jihmbo", "I'm totally into you"),
};
```

Microsoft Visual Studio Debug Console

Winner is Jihmbo

5 Поклонников:

```
const size_t N = 5;
vector<ClientFan> fans =
{
    ClientFan("Bob", "I love you!"),
    ClientFan("Mike", "I <3 you"),
    ClientFan("Jihmbo", "I'm totally into you"),
    ClientFan("Alex", "You mean so much to me"),
    ClientFan("Mark", "You're my ideal woman"),
};
```

1 запуск:

Microsoft Visual Studio Debug Console

Winner is Bob

2 запуск:

Microsoft Visual Studio Debug Console

Winner is Mike

3 запуск:

Microsoft Visual Studio Debug Console

Winner is Bob

Приложение

Код программы:

```
#include <omp.h>
#include <iostream>
#include <vector>
#include <thread>
#include <mutex>
#include <deque>
#include <cstdlib>
#include <algorithm>
#include <random>

using namespace std;

mt19937 gen((int)time(0));
uniform_int_distribution<int> rnd(0, 1000);

class ClientFan
{
public:
    ClientFan() {}
    ClientFan(string name, string msg) {
        this->name = name;
        message = msg;
    }
    string message;
    string name;
};

class ServerStudentka
{
public:
    ServerStudentka() {
        fans = {};
    }

    void StartCheck()
    {
        mx.lock();
        for (size_t i = count - 1; i >= 0 ; i--){
            if (((rnd(gen) & i + 1) == 0) || (i == 0)) {
                winner = fans[i];
                cout << "Winner is " << winner.name;
                stop = true;
                break;
            }

            fans.pop_back();
        }
        mx.unlock();
    }

    void SendMessage(ClientFan fan) {
        mx.lock();
        fans.push_back(fan);
        count++;
        mx.unlock();
        while (!stop) {
        }
    }
};
```

```

    }

protected:

    size_t count = 0;
    mutex mx;

    bool stop = false;
    vector<ClientFan> fans;
    ClientFan winner;
};

int main()
{
    const size_t n = 5;
    vector<ClientFan> fans =
    {
        ClientFan("Bob", "I love you!"),
        ClientFan("Mike", "I <3 you!"),
        ClientFan("Jihmbo", "I'm totally into you!"),
        ClientFan("Alex", "You mean so much to me"),
        ClientFan("Mark", "You are my ideal woman"),
    };
    ServerStudentka* server = new ServerStudentka();
#pragma omp parallel num_threads(n + 1)
    {
#pragma omp for
        for (int i = 0; i < n + 1; ++i)
        {
            if (i == n) {
                server->StartCheck();
            }
            else {
                server->SendMessage(fans[i]);
            }
        }
    }
}

```

Список использованной литературы

1. Habr (2020) «Клиент-сервер шаг — за — шагом, от однопоточного до многопоточного (Client-Server step by step)» (<https://habr.com/ru/post/330676/>).
2. Metanit (2020) «Многопоточное клиент-серверное приложение TCP» (<https://metanit.com/sharp/net/4.3.php>).
3. Cyberforum (2020) «Простой клиент-сервер с многопоточностью» (<https://www.cyberforum.ru/java-networks/thread1557122.html>).
4. Docs Microsoft (2020) «Creating Threads» (<https://docs.microsoft.com/en-us/windows/win32/procthread/creating-threads>).
5. Shalaton Grost17 «Потоки (threads) в WinAPI» (<http://shatalov.ghost17.ru/winapi/threads.html>).
6. Легалов А.И.(2020) «Многопоточность. Простая многопоточная программа. Основные функции» (<http://softcraft.ru/edu/comparch/practice/thread/01-simple/>).
7. Легалов А.И.(2020) «Многопоточность. Синхронизация потоков. Методы синхронизации» (<http://softcraft.ru/edu/comparch/practice/thread/02-sync/>).
8. Параллельное программирование на OpenMP (<http://ccfit.nsu.ru/arom/data/openmp.pdf>)