

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

Лабораторная работа №7 по курсу
«Технологии программирования»

Выполнил студент группы ИВТ-32 _____ /Рзаев А. Э./
Проверил доцент кафедры ЭВМ _____ /Долженкова М. Л./

Киров 2017

1 Задание

Написать программу, решающую задачу о парикмахере, используя несколько потоков.

2 Результат работы

Экранные формы приведены в приложении А.

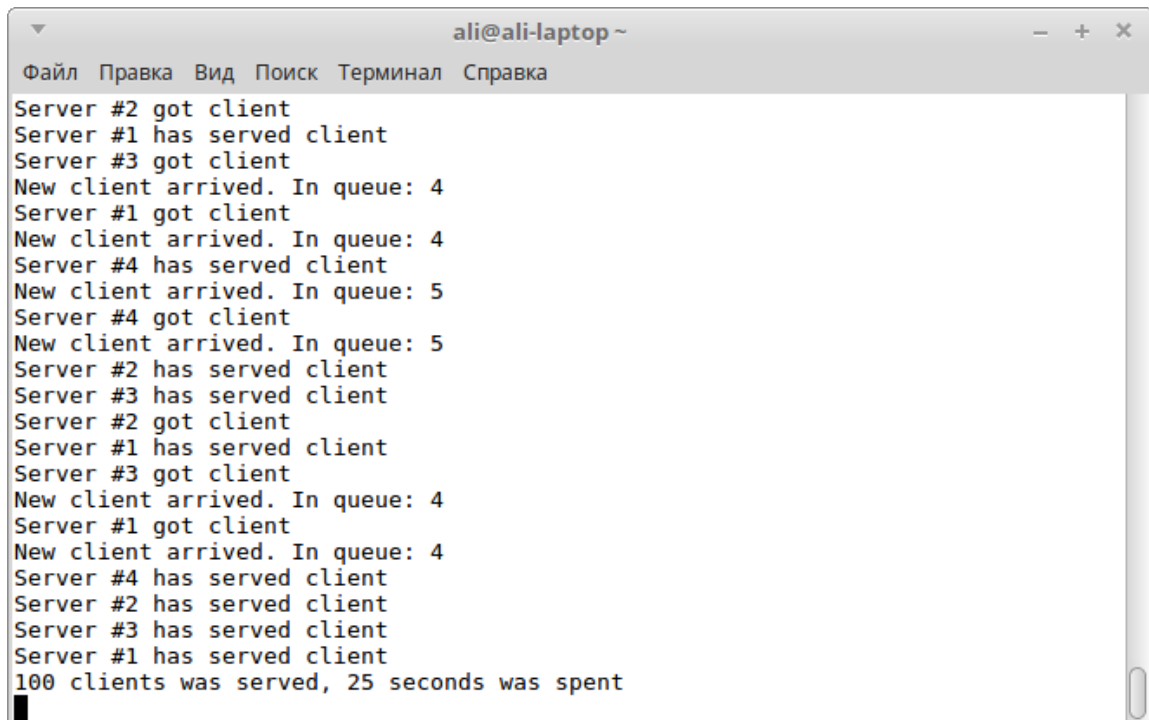
3 Листинг программы

Листинг программы приведен в приложении Б.

4 Вывод

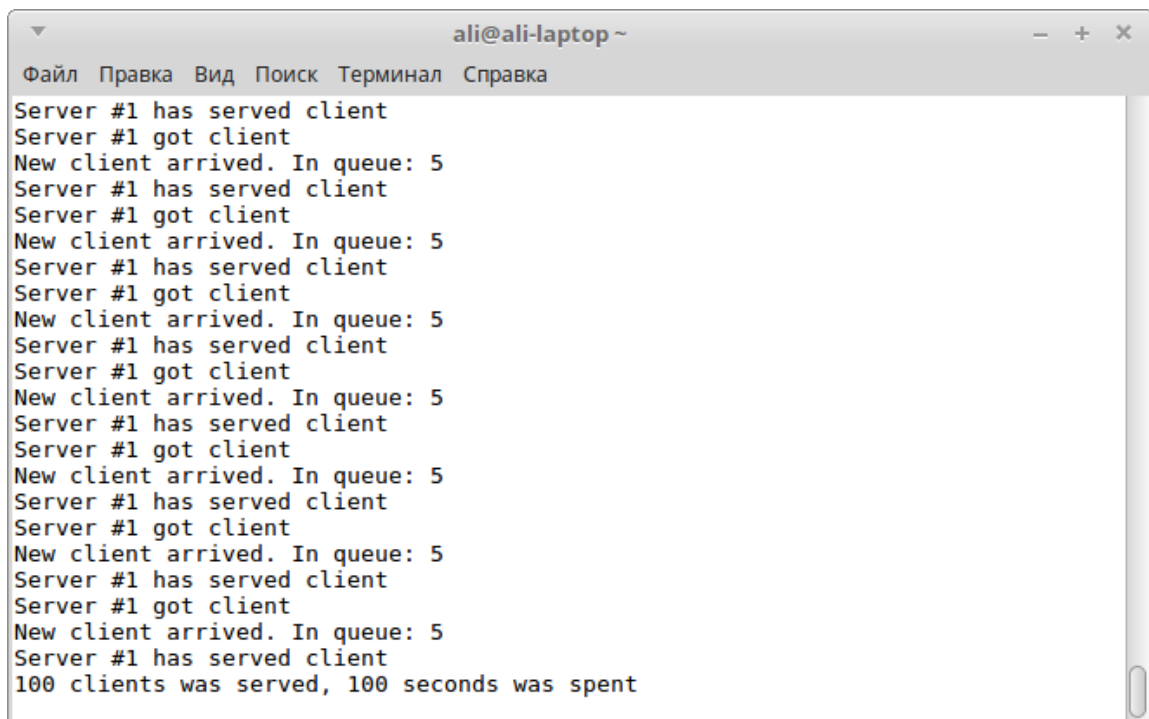
В ходе выполнения лабораторной работы была написана многопоточная программа, решающая задачу о парикмахере. Изучены такие средства синхронизации потоков, как мьютексы. Также было подсчитано время работы программы в однопоточном и многопоточном исполнении. На компьютере с 2 физическими и 2 виртуальными ядрами прирост производительности был в 4 раза.

Приложение А
(обязательное)
Экранные формы



```
ali@ali-laptop ~
Файл Правка Вид Поиск Терминал Справка
Server #2 got client
Server #1 has served client
Server #3 got client
New client arrived. In queue: 4
Server #1 got client
New client arrived. In queue: 4
Server #4 has served client
New client arrived. In queue: 5
Server #4 got client
New client arrived. In queue: 5
Server #2 has served client
Server #3 has served client
Server #2 got client
Server #1 has served client
Server #3 got client
New client arrived. In queue: 4
Server #1 got client
New client arrived. In queue: 4
Server #4 has served client
Server #2 has served client
Server #3 has served client
Server #1 has served client
100 clients was served, 25 seconds was spent
```

Рисунок 1 – Результат работы программы с использованием потоков



```
ali@ali-laptop ~
Файл Правка Вид Поиск Терминал Справка
Server #1 has served client
Server #1 got client
New client arrived. In queue: 5
Server #1 has served client
Server #1 got client
New client arrived. In queue: 5
Server #1 has served client
Server #1 got client
New client arrived. In queue: 5
Server #1 has served client
Server #1 got client
New client arrived. In queue: 5
Server #1 has served client
Server #1 got client
New client arrived. In queue: 5
Server #1 has served client
Server #1 got client
New client arrived. In queue: 5
Server #1 has served client
Server #1 got client
100 clients was served, 100 seconds was spent
```

Рисунок 2 – Результат работы программы без использования потоков

Приложение Б
(обязательное)
Листинг программы

main.cpp

```
#include <iostream>
#include <thread>
#include <mutex>
#include <vector>
#include <atomic>
#include <chrono>

const unsigned int MAX_QUEUE_SIZE = 5,
                  MAX_CLIENTS = 100;
unsigned int q_size = 0, count = 0;
std::mutex q_mutex, io_mutex, c_mutex;

bool test_count() {
    std::lock_guard<std::mutex> c_lock(c_mutex);
    return count < MAX_CLIENTS;
}

void work(int num) {
    while (test_count()) {
        bool ok = false;
        {
            std::lock_guard<std::mutex> q_lock(q_mutex);
            if (q_size > 0) {
                std::lock_guard<std::mutex> c_lock(c_mutex);
                count++;
                q_size--;
                ok = true;
            }
        }
        if (ok) {
            {
                std::lock_guard<std::mutex> io_lock(io_mutex);
                std::cout << "Server #" << num << " got client" <<
                    std::endl;
            }

            std::this_thread::sleep_for(std::chrono::milliseconds
                (800));

            {
                std::lock_guard<std::mutex> io_lock(io_mutex);
                std::cout << "Server #" << num << " has served client"
                    << std::endl;
            }
        }
    }
}
```

```

        std::this_thread::sleep_for(std::chrono::milliseconds
            (200));
    }
}

void add_client() {
    while (test_count())
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(200));
        std::lock_guard<std::mutex> q_lock(q_mutex);

        if (q_size < MAX_QUEUE_SIZE) {
            q_size++;

            std::lock_guard<std::mutex> io_lock(io_mutex);
            std::cout << "New client arrived. " << "In queue: " <<
                q_size << std::endl;
        }
    }
}

int main() {
    auto start = std::chrono::system_clock::now();
    auto w1 = std::thread(work, 1);
    auto w2 = std::thread(work, 2);
    auto w3 = std::thread(work, 3);
    auto w4 = std::thread(work, 4);
    auto a = std::thread(add_client);
    w1.join();
    w2.join();
    w3.join();
    w4.join();
    a.join();
    auto stop = std::chrono::system_clock::now();
    std::cout
        << MAX_CLIENTS << " clients was served, "
        << std::chrono::duration_cast<std::chrono::seconds>(stop -
            start).count() << " seconds was spent"
        << std::endl;
    std::cin.get();
    return 0;
}

```