

|  |  |
| --- | --- |
| https://lh7-us.googleusercontent.com/QuBaagabzZYLr7U3ZbYOtMusd5FbWWxIvMUTN8jrLHzSLMg534z9gXRTIG1Us4i_lOwmWlaBxKedNt-SQ26dm4WmyqwjGDmEO6z8GE3QrZosqvHM88J2EFeVf1u0GzyCZQlhWmp1Zeo85tKo4LJVXQ | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)» |

Институт № 3 «Системы управления, информатика и электроэнергетика»

Кафедра 304 «Вычислительные машины, системы и сети»

Отчёт по лабораторной работе №3  
по учебной дисциплине «Web-технологии»

на тему «Сокеты»

Выполнил  
студент группы М3О-119БВ-24

Нарзиев А.Т.

Принял

Ассистент каф.304 Борисов А.И.

Москва  
2024

# Содержание

[Задание 3](#_Toc194422155)

[Решение 4](#_Toc194422156)

[1. Разработка клиента 4](#_Toc194422157)

[2. Разработка сервера 7](#_Toc194422158)

[3. Разработка многопоточности 9](#_Toc194422159)

[4. Тесты 10](#_Toc194422160)

[Вывод 11](#_Toc194422161)

# Задание

1. Разработать программу клиент, работающий на сокетах на выбранном Вами языке программирования. Данный клиент должен хранить лог файл соединений. В лог файле хранится время подключения и адрес сервера, время отправки сообщения и само сообщение, время получения сообщения и само сообщение. При подключении к серверу клиент через определенный промежуток времени (задается разработчиком клиента) отправляет сообщение серверу: ФИО и группа студента, а также ждет от сервера. Данные о сервере необходимо вводить через файл конфигурации. В отчете привести код клиента и один тестовый лог файл.
2. Разработать программу сервер, которая ожидает строку от клиента. Через промежуток времени (моделирование работы сервера) он отправляет в ответ эту же строку, только зеркально отраженную и добавляет к сообщению ФИО студента, написавшего сервер. (пример ответа: Чиволвап Йирю Вотит. Сервер написан Титовым Ю.П. М3О-1ХХБВ-ХХ).

Сервер должен эмулировать работу путем временной задержки. Отключение клиентов от сервера совершать через определенное время работы клиента. Сервер и клиент должны быть написаны на различных языках программирования.

Сервер ведёт лог-файл, в котором содержится:

* Время запуска сервера,
* Время подсоединения клиента,
* Адрес клиента,
* Время получения сообщения и само сообщение,
* Время отправки сообщения и само сообщение, Время отключения клиента от сервера.

Лог-файл не очищать.

1. Доработать программу сервер для выполнения асинхронных запросов от клиентов. Чётные варианты реализуют многопоточный асинхронный сервер, нечетные – многопроцессный. Номер варианта определяется номером по списку.

При доработке сервера необходимо создать новый проект, т.е. в результате должно получиться две программы сервера.

1. Запустить 5 клиентов в сети кафедры и 1 сервер (5 студентов). В отчете указать настройки для подключения и лог файлы клиентов и серверов. Проводить тестирование одновременного подключения, т.е. второй, третий и т.д. пользователь подключается к серверу пока первый пользователь еще не получил ответ. Желательно провести тестирование как на однопоточном, так и на асинхронном сервере, описать отличия.

# Решение

## Разработка клиента

***Клиент – client-socket.cpp:***

1. #include <iostream>
2. #include <fstream>
3. #include <cstring>
4. #include <sstream>
5. #include <ctime>
6. #ifdef \_\_GNUC\_\_
7. #define \_WIN32\_WINNT 0x0501
8. #endif
9. #include <winsock2.h>
10. #include <ws2tcpip.h>
11. using namespace std;
12. #pragma comment(lib, "ws2\_32.lib") // Подключение библиотеки Winsock
13. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
14. \* Функция для записи события в лог-файл.
15. \*
16. \* @param message Сообщение, которое будет записано в лог.
17. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
18. void logEvent(const string& message) {
19. ofstream logFile("client\_log.txt", ios\_base::app);
20. if (logFile.is\_open()) {
21. time\_t now = time(0);
22. char\* dt = ctime(&now); // Строка для хранения времени
23. dt[strlen(dt) - 1] = '\0'; // Удаляем символ новой строки
24. logFile << dt << " - " << message << endl;
25. logFile.close();
26. }
27. }
28. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
29. \* Основная функция клиента.
30. \*
31. \* Эта функция отвечает за инициализацию Winsock, создание сокета,
32. \* подключение к серверу, отправку сообщения и получение ответа.
33. \*
34. \* @return Код завершения программы.
35. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
36. int main() {
37. // Инициализация Winsock
38. WSADATA wsaData;                                 // Структура для хранения версии Winsock и используемых протоколах
39. if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) { // Используем версию 2.2 Winsock
40. logEvent("Ошибка инициализации Winsock");
41. return 1;
42. }
43. // Загрузка конфигурации
44. string serverAddress = "127.0.0.1"; // IP-адрес умолчанию
45. int serverPort = 2371;              // Порт по умолчанию
46. // Чтение конфигурации из файла
47. ifstream configFile("config.txt");
48. if (configFile.is\_open()) {
49. string line;
50. while (getline(configFile, line)) {
51. istringstream configStream(line);
52. string key;
53. if (configStream >> key) {
54. if (key == "ip\_address") {
55. configStream >> serverAddress; // Чтение IP-адреса
56. } else if (key == "port") {
57. configStream >> serverPort; // Чтение порта
58. }
59. }
60. }
61. configFile.close();
62. } else {
63. logEvent("Не удалось открыть файл конфигурации. Используются настройки по умолчанию.");
64. }
65. // Создание сокета
66. SOCKET sock = socket(AF\_INET, SOCK\_STREAM, 0);
67. if (sock == INVALID\_SOCKET) {
68. logEvent("Ошибка создания сокета: " +
69. to\_string(WSAGetLastError()));
70. WSACleanup();
71. return 1;
72. }
73. sockaddr\_in server;                                          // Структура, содержит информацию о сервере, к которому присоединяемся
74. server.sin\_family = AF\_INET;                                 // Указываем, что используем IPv4
75. server.sin\_port = htons(serverPort);                         // Преобразуем порт в сетевой порядок байтов
76. server.sin\_addr.s\_addr = inet\_addr(serverAddress.c\_str());   // Преобразуем строку IP-адреса в формат, используемый в sockaddr\_in
77. //inet\_pton(AF\_INET, serverAddress.c\_str(), &server.sin\_addr);
78. if (server.sin\_addr.s\_addr == INADDR\_NONE) {
79. logEvent("Некорректный IP-адрес: " + serverAddress);
80. closesocket(sock);
81. WSACleanup();
82. return 1;
83. }
84. /\*\*
85. \* Подключение к серверу.
86. \*
87. \* Программа пытается установить соединение с сервером с помощью функции connect(). Если подключение не удалось,
88. \* записывается сообщение об ошибке, сокет закрывается, и программа завершает выполнение.
89. \*/
90. if (connect(sock, (struct sockaddr\*)&server, sizeof(server)) == SOCKET\_ERROR) {
91. logEvent("Ошибка подключения к серверу " + serverAddress + ":" +
92. to\_string(serverPort) + " - " + to\_string(WSAGetLastError()));
93. closesocket(sock);
94. WSACleanup();
95. return 1;
96. }
97. logEvent("Подключение к серверу " + serverAddress + ":" + to\_string(serverPort));
98. // Формирование сообщения
99. string fio = "Нарзиев Артемий Тимурович";
100. string group = "М3О-119БВ-24";
101. string message = fio + ", " + group;
102. // Задержка перед отправкой
103. Sleep(2000);
104. // Отправка сообщения
105. logEvent("Отправка сообщения: " + message);
106. int bytesSent = send(sock, message.c\_str(), message.length(), 0);
107. if (bytesSent == SOCKET\_ERROR) {
108. logEvent("Ошибка отправки сообщения: " + to\_string(WSAGetLastError()));
109. closesocket(sock);
110. WSACleanup();
111. return 1;
112. }
113. /\*\*
114. \* Ожидание ответа от сервера и обработка полученных данных.
115. \*
116. \* Если данные получены, они записываются в лог. Если произошла ошибка,
117. \* записывается сообщение об ошибке. Если сервер закрыл соединение,
118. \* также записывается соответствующее сообщение.
119. \*/
120. char buffer[1024] = {0};
121. int bytesReceived = recv(sock, buffer, sizeof(buffer) - 1, 0);
123. if (bytesReceived > 0) {
124. buffer[bytesReceived] = '\0';
125. logEvent("Получено сообщение от сервера: " + string(buffer));
126. } else if (bytesReceived == SOCKET\_ERROR) {
127. logEvent("Ошибка получения сообщения от сервера: " + to\_string(WSAGetLastError()));
128. } else {
129. logEvent("Соединение закрыто сервером");
130. }
131. // Закрытие соединения
132. closesocket(sock);
133. //cout << "Vse OK" << endl; // Проверка компиляции
134. WSACleanup(); // Освобождение ресурсов Winsock
135. logEvent("Соединение закрыто");
136. return 0;
137. }

***Тестовый лог-файл клиента - client\_log.txt:***

Wed Mar 19 19:36:24 2025 - Подключение к серверу 127.0.0.1:2371

Wed Mar 19 19:36:26 2025 - Отправка сообщения: Нарзиев Артемий Тимурович, М3О-119БВ-24

Wed Mar 19 19:36:28 2025 - Получено сообщение от сервера: 42-ВБ911-О3М ,чиворумиТ йиметрА веизраН. Сервер написан Нарзиевым А.Т. М3О-119БВ-24

Wed Mar 19 19:36:28 2025 - Соединение закрыто

***Файл конфигурации клиента – config.txt:***

ip\_address 192.168.107.166

port 9090

## Разработка сервера

***Сервер – server-socket.py:***

import socket

import time

import logging

from datetime import datetime

# Настройка логирования, чтобы записывать события в файл

#logging.basicConfig(filename='server\_log.txt', level=logging.INFO, format='%(asctime)s - %(message)s')

logging.basicConfig(filename='server\_log.txt',

                    level=logging.INFO,

                    format='%(asctime)s - %(message)s',

                    datefmt='%Y-%m-%d %H:%M:%S',

                    encoding='utf-8'

                    )

def log\_event(message):

    """

    Функция для записи события в лог.

    :param message: Сообщение, которое будет записано в лог.

    """

    logging.info(message)

def start\_server(host='127.0.0.1', port=2371):

    """

    Функция для запуска TCP-сервера.

    :param host: Хост, на котором будет запущен сервер (по умолчанию '127.0.0.1').

    :param port: Порт, на котором будет слушать сервер (по умолчанию 2371).

    """

    # Создаем сокет для сервера

    with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server\_socket:

        server\_socket.bind((host, port))    # Привязываем сокет к хосту и порту

        server\_socket.listen()              # Начинаем слушать входящие соединения

        log\_event("Сервер запущен на {}:{}".format(host, port))

        while True:

            client\_socket, addr = server\_socket.accept()  # Принимаем подключение клиента

            with client\_socket:

                log\_event("Подключен клиент: {}".format(addr))

                # Получаем сообщение от клиента

                data = client\_socket.recv(1024) # Получаем данные размером до 1024 байт

                if not data:

                    break

                message = data.decode('utf-8') # Декодируем байты в строку

                log\_event("Получено сообщение: {}".format(message))

                # Эмулируем работу сервера с задержкой

                time.sleep(2) # Задержка в 2 секунды

                # Отправляем обратно зеркальное сообщение

                reversed\_message = message[::-1] # Зеркалим строку

                response\_message = "{}. Сервер написан Нарзиевым А.Т. М3О-119БВ-24".format(reversed\_message)

                client\_socket.sendall(response\_message.encode('utf-8')) # Отправляем ответ клиенту

                log\_event("Отправлено сообщение: {}".format(response\_message))

if \_\_name\_\_ == "\_\_main\_\_":

    start\_server()  # Запускаем сервер

***Тестовый лог-файл сервера - server\_log.txt:***

2025-03-20 23:15:45 - Сервер запущен на 127.0.0.1: 2371

2025-03-20 23:15:52 - Подключен клиент: ('127.0.0.1', 52507)

2025-03-20 23:15:54 - Получено сообщение от ('127.0.0.1', 52507): Нарзиев Артемий Тимурович, М3О-119БВ-24

2025-03-20 23:15:56 - Отправлено сообщение клиенту: 42-ВБ911-О3М ,чиворумиТ йиметрА веизраН. Сервер написан Нарзиевым А.Т. М3О-119БВ-24

## Разработка многопоточности

***Сервер с поддержкой многопоточности – server-socket-multi-thread.py:***

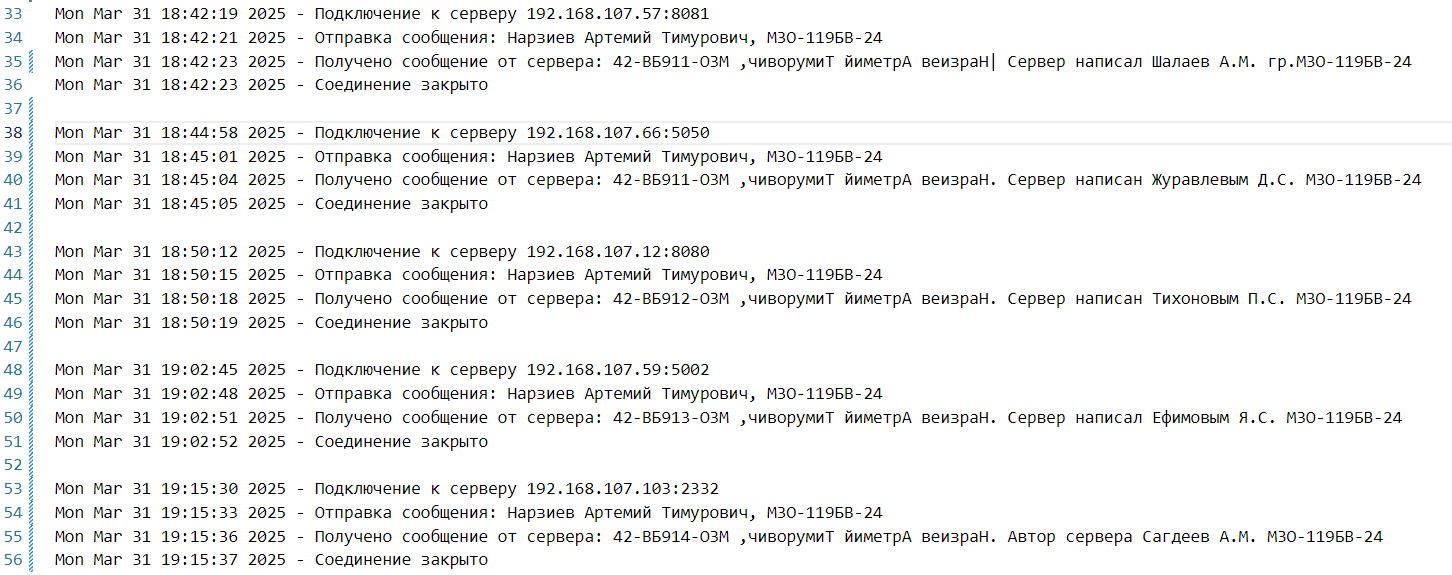
1. import socket
2. import time
3. import logging
4. import threading
5. import os
6. from datetime import datetime
7. # Настройка логирования, чтобы записывать события в файл
8. #logging.basicConfig(filename='server\_log.txt', level=logging.INFO, format='%(asctime)s - %(message)s')
9. logging.basicConfig(filename='server\_log.txt',
10. level=logging.INFO,
11. format='%(asctime)s - %(message)s',
12. datefmt='%Y-%m-%d %H:%M:%S',
13. encoding='utf-8'
14. )
15. def log\_event(message):
16. logging.info(message)
17. def client\_processing(client\_socket, addr):
18. with client\_socket:
19. log\_event("Подключен клиент: {}".format(addr))
20. # Получаем сообщение от клиента
21. data = client\_socket.recv(1024)  # Получаем данные размером до 1024 байт
22. if not data:
23. return
24. message = data.decode("utf-8")  # Декодируем байты в строку в утф8
25. log\_event("Получено сообщение от {}: {}".format(addr, message))
26. time.sleep(2)  # Задержка в 2 секунды
27. reversed\_message = message[::-1]  # Зеркалим полученную строку
28. response\_message = "{}. Сервер написан Нарзиевым А.Т. М3О-119БВ-24".format(reversed\_message)
29. client\_socket.sendall(response\_message.encode('utf-8'))
30. log\_event("Отправлено сообщение клиенту: {}".format(response\_message))
31. def start\_server(host='127.0.0.1', port=2371):
32. # Создаем сокет для сервера
33. with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server\_socket:
34. server\_socket.bind(('127.0.0.1', 2371))  # Привязываем сокет к хосту и порту
35. server\_socket.listen()  # Начинаем слушать входящие соединения
36. log\_event("Сервер запущен на {}: {}".format(host, port))
38. while True:
39. client\_socket, addr = server\_socket.accept()  # Принимаем подключение клиента
41. # Создаем новый поток для обработки клиента
42. client\_thread = threading.Thread(target=client\_processing, args=(client\_socket, addr))
43. client\_thread.start()  # Запускаем поток
45. if \_\_name\_\_ == "\_\_main\_\_":
46. start\_server()  # Запускаем сервер

## Тесты

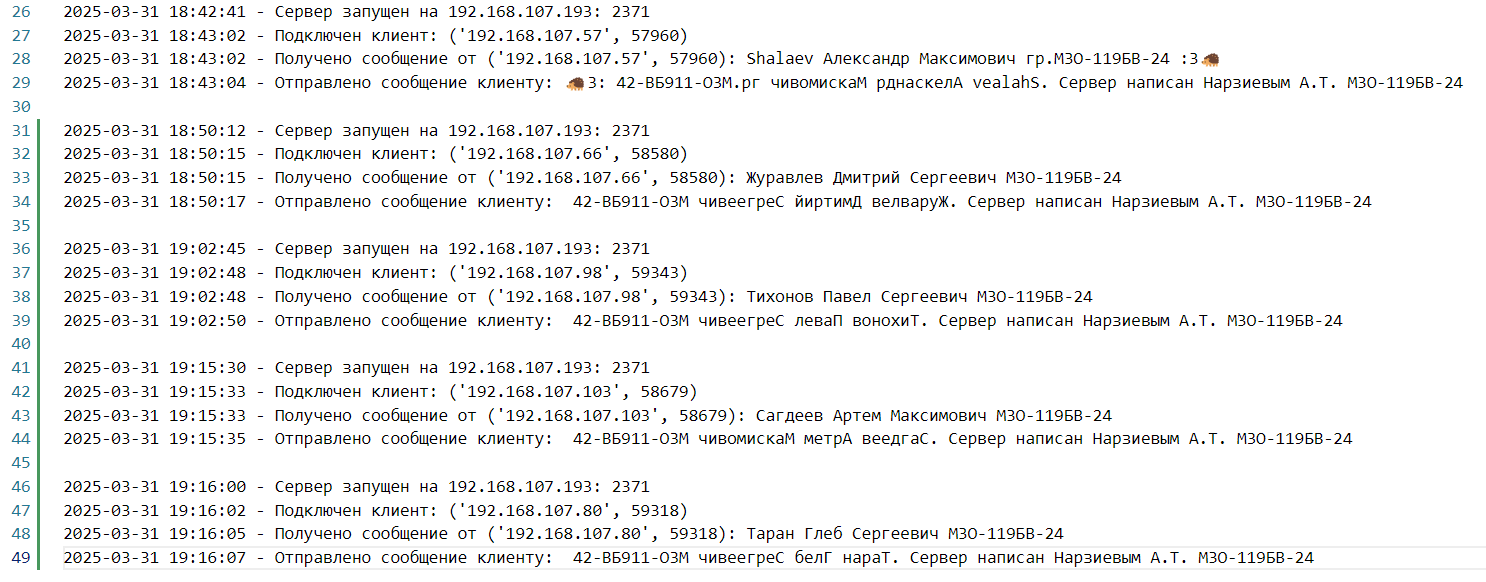
Тесты проводились на многопоточном варианте сервера с пятью людьми в одной сети.

Результаты тестирования приведены ниже.

**Тестирование клиента:**

****

**Тестирование сервера:**



# Вывод

Разработка программы завершена на том основании, что:

1. Полученные результаты совпали с ожидаемыми
2. Считаю набор тестов полным