**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»

Web-технологии

Отчет по лабораторной работе №2

Сокеты

Выполнил студент группы М3О-409Б-20:

Мокшин И. Н.

Проверил, Титов Ю.П.

Москва 2023 г.

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.

**Код клиента (javascript)**

**client.js**

const net = require('net');

const Logger = require('./logger');

const fs = require('fs');

const config = require('./config.json')

var logger = new Logger("client.log");

var client = new net.Socket();

client.on('data', function(data) {

    logger.log('Received: ' + data);

    client.destroy();

});

client.on('close', function() {

    logger.log('Connection closed');

});

setTimeout(main, 1066);

function main() {

    client.connect(config.port, config.address, function() {

        logger.log('Client connected');

        setTimeout(() => {

            client.write('Mokshin Ivan Nikolayevich\0');

            logger.log('Sent message');

        }, 500);

    });

}

**logger.js**

const fs = require("fs");

class Logger {

constructor(fileName) {

this.file = fs.createWriteStream(fileName, { flags: "a" });

}

log(message) {

console.log(message);

this.file.write(`[${new Date().toISOString()}] ${message}\n`);

}

close() {

this.file.end();

}

}

module.exports = Logger;

**config.json**

{

    "address": "127.0.0.1",

    "port": 8888

}

1. 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 Титовы Ю.П. М3О-3ХХБ-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 ведет лог-файл, в котором содержатся Время запуска сервера, Время подсоединения клиента, Время получения сообщения и само сообщение, Время отправки сообщения и само сообщение, Время отключения клиента от сервера.

**Program.cs**

internal class Program

{

private static async Task Main(string[] args)

{

// V1

//ServerV1.Run();

// V2

await ServerV2.Run();

}

}

**Logger.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection.Emit;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace WebTecLr2\_Server

{

public class Logger

{

StreamWriter file { get; set; }

public Logger(string fileName)

{

file = new StreamWriter(fileName, true);

}

public void Log(string message)

{

Console.WriteLine(message);

file.WriteLine($"[{DateTime.Now}] {message}");

file.Flush();

}

~Logger()

{

file.Dispose();

}

}

}

**ServerV1.cs**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

using WebTecLr2\_Server;

class ServerV1

{

public static void Run()

{

var logger = new Logger("server.log");

IPEndPoint ipPoint = new IPEndPoint(IPAddress.Any, 8888);

using Socket socket = new Socket(

AddressFamily.InterNetwork,

SocketType.Stream,

ProtocolType.Tcp

);

socket.Bind(ipPoint);

socket.Listen();

logger.Log($"Сервер запущен по адресу {ipPoint}");

while (true)

{

logger.Log("Ожидание подключения");

using Socket client = socket.Accept();

logger.Log($"Подключён клиент: {client.RemoteEndPoint}");

while (true)

{

Thread.Sleep(1000);

byte[] b = new byte[1024];

var amount = client.Receive(b);

if (amount > 0)

{

var str = Encoding.ASCII.GetString(b.AsSpan().TrimEnd((byte)0));

logger.Log($"Получен запрос от клиента {client.RemoteEndPoint}. {str}");

var response = Encoding.ASCII.GetBytes(str.Reverse().ToArray());

Thread.Sleep(1500);

client.Send(response);

logger.Log($"Обработан запрос от клиента {client.RemoteEndPoint}");

} else

{

logger.Log($"Соединение закрыто {client.RemoteEndPoint}");

break;

}

}

}

}

}

1. Доработать программу сервер для выполнения асинхронных запросов от клиентов. Четные 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.

**ServerV1.cs**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using WebTecLr2\_Server;

using static System.Runtime.InteropServices.JavaScript.JSType;

public static class ServerV2

{

private static Logger logger;

private static Socket socket;

private static Thread listenThread;

public static async Task Run()

{

logger = new Logger("server.log");

IPEndPoint ipPoint = new IPEndPoint(IPAddress.Any, 8888);

socket = new Socket(

AddressFamily.InterNetwork,

SocketType.Stream,

ProtocolType.Tcp

);

socket.Bind(ipPoint);

socket.Listen();

logger.Log($"Сервер запущен по адресу {ipPoint}");

while (true)

{

var client = await socket.AcceptAsync();

var thread = new Thread(() => HandleClientAsync(client));

thread.Start();

}

}

private static async Task HandleClientAsync(Socket client)

{

while (true)

{

Thread.Sleep(1000);

byte[] data = new byte[1024];

var amount = await client.ReceiveAsync(data);

if (amount > 0)

{

var thread = new Thread(() => HandleRequestAsync(client, data));

thread.Start();

}

else

{

logger.Log($"Соединение закрыто {client.RemoteEndPoint}");

break;

}

}

}

private static async Task HandleRequestAsync(Socket client, byte[] data)

{

var str = Encoding.ASCII.GetString(data.AsSpan().TrimEnd((byte)0));

logger.Log($"Получен запрос от клиента {client.RemoteEndPoint}. {str}");

var response = Encoding.ASCII.GetBytes(str.Reverse().ToArray());

Thread.Sleep(1500);

client.Send(response);

logger.Log($"Обработан запрос от клиента {client.RemoteEndPoint}");

}

}

1. Запустить 5 клиентов в сети кафедры и 1 сервер (5 студентов). В отчете указать настройки для подключения и лог файлы клиентов и серверов.