

Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н. Тихонова

ОТЧЕТ
по Лабораторной работе №9
по дисциплине «Системное программирование»
по теме «Сокеты ОС»

Студент гр. БИБ201
Морин Д.А.
«4» июня 2023 г.

Руководитель
Преподаватель
Д.В. Смирнов
«___» _____ 2023 г.

Москва 2023

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	2
1 ВВЕДЕНИЕ.....	2
2 ОСНОВНАЯ ЧАСТЬ.....	3
2.1 Клиент-серверная часть.....	3
2.2 Дополнительное задание на 5 баллов.....	8
3 ЗАКЛЮЧЕНИЕ.....	10
ПРИЛОЖЕНИЕ А.....	11

1 ВВЕДЕНИЕ

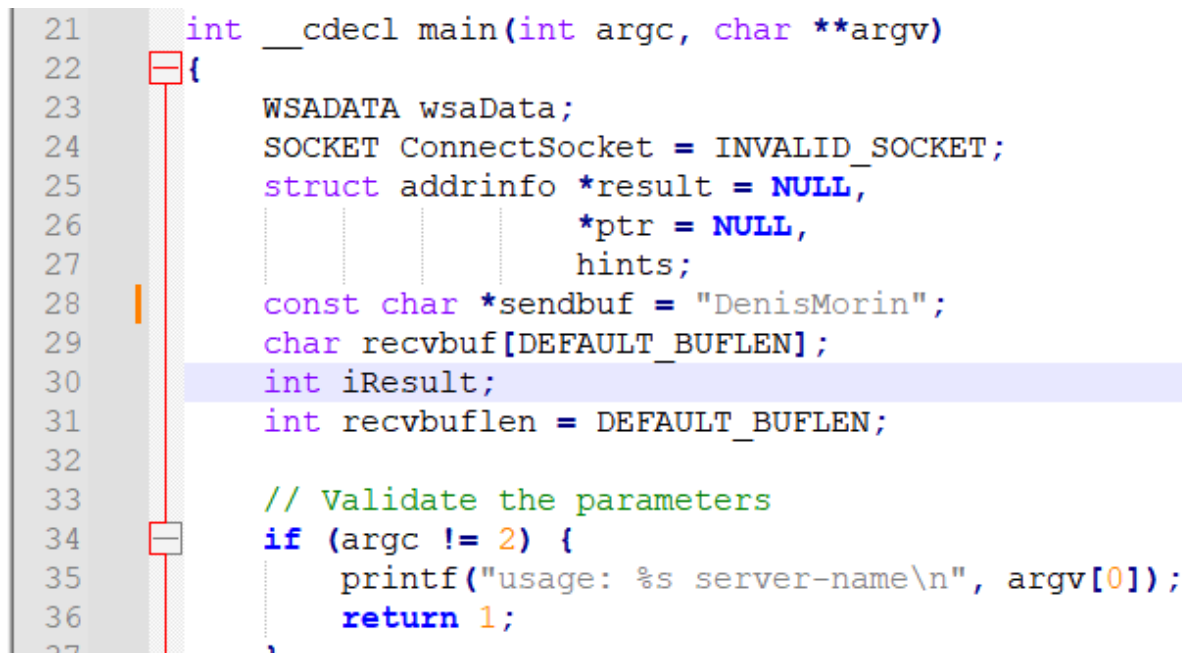
Цель данной практической работы заключается в изучении и дальнейшем практическом освоении клиент-серверной модели сетевого взаимодействия. Для достижения поставленной цели необходимо изучить и модифицировать программы "server.c" и "client.c". Данное программное обеспечение выполняет серверную и клиентскую задачи соответственно. Серверная часть программы "server.c" принимает соединения от клиентов и обрабатывает их запросы. Клиентская часть программы "client.c" устанавливает соединение с сервером и отправляет запросы на его обработку.

Для выполнения этой работы необходимо будет произвести компиляцию программ, настроить среду для мониторинга активности сети, а также запустить сервер и клиент для проверки их функциональности.

2 ОСНОВНАЯ ЧАСТЬ

2.1 Клиент-серверная часть

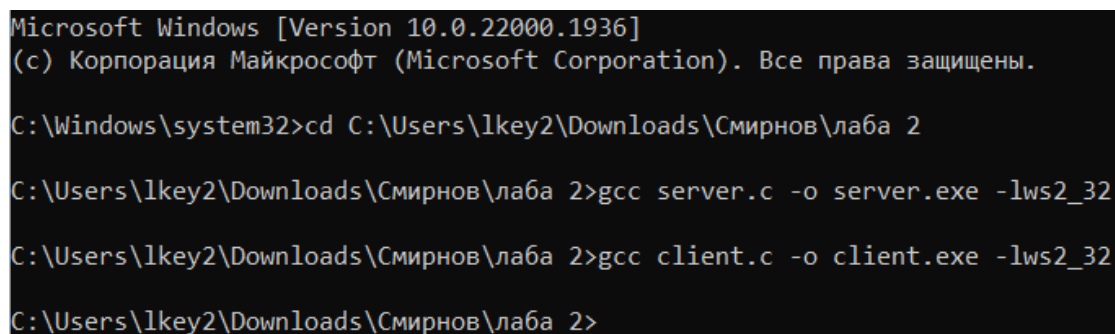
Согласно заданию была изменена строка “This is a test” в файле client.c на “DenisMorin” (Рисунок 1).



```
21 int __cdecl main(int argc, char **argv)
22 {
23     WSADATA wsaData;
24     SOCKET ConnectSocket = INVALID_SOCKET;
25     struct addrinfo *result = NULL,
26         *ptr = NULL,
27         hints;
28     const char *sendbuf = "DenisMorin";
29     char recvbuf[DEFAULT_BUFLEN];
30     int iResult;
31     int recvbuflen = DEFAULT_BUFLEN;
32
33     // Validate the parameters
34     if (argc != 2) {
35         printf("usage: %s server-name\n", argv[0]);
36         return 1;
37     }
```

Рисунок 1 – Измененный файл client.c

Следующим шагом были скомпилированы файлы server.c и client.c (Рисунок 2).



```
Microsoft Windows [Version 10.0.22000.1936]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Windows\system32>cd C:\Users\lkey2\Downloads\Смирнов\лаба 2
C:\Users\lkey2\Downloads\Смирнов\лаба 2>gcc server.c -o server.exe -lws2_32
C:\Users\lkey2\Downloads\Смирнов\лаба 2>gcc client.c -o client.exe -lws2_32
C:\Users\lkey2\Downloads\Смирнов\лаба 2>
```

Рисунок 2 – Компиляция файлов

В программе Process Monitor настроил фильтры по имени процесса для файлов “client.exe” и “server.exe”, и чтобы отображались только сетевые события “Show Network Activity” (Рисунок 3).

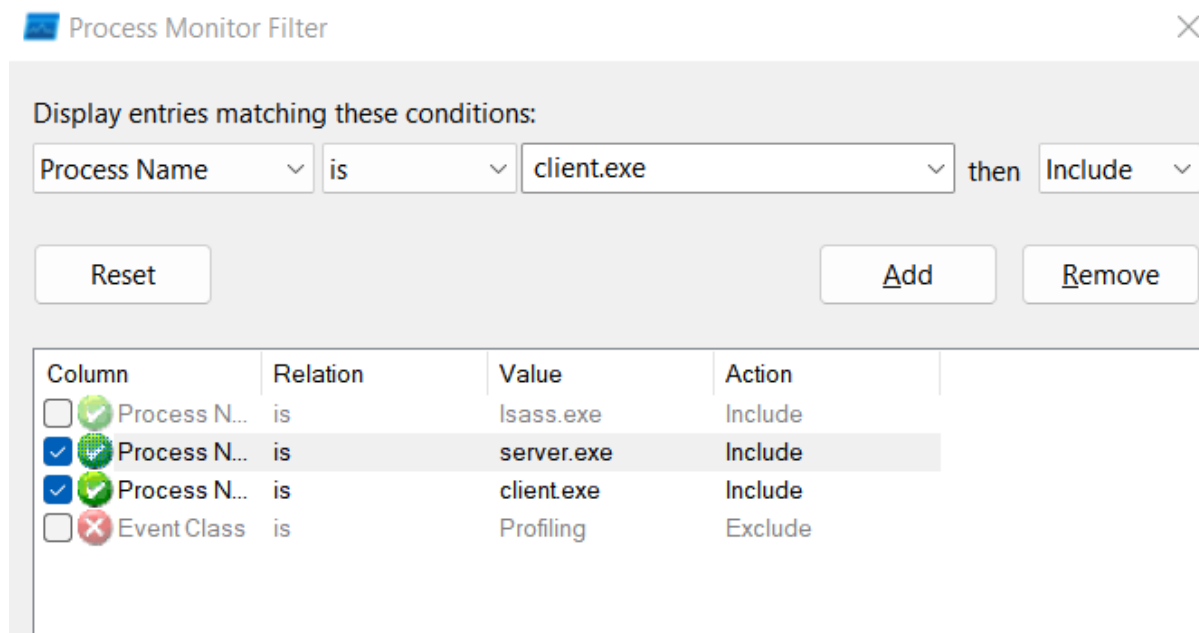


Рисунок 3 – Настройка фильтров в Process Monitor

Также был настроен фильтр по протоколу tcp в программе Wireshark (Рисунок 4) и отображение трафика по loopback (Рисунок 5).

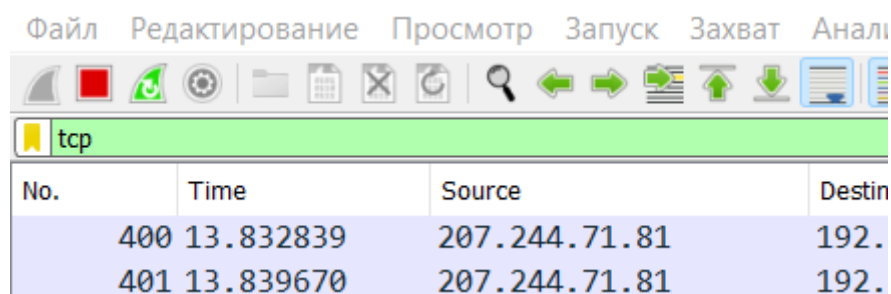


Рисунок 4 – Настройка фильтра в Wireshark

Подключение по локальной сети* 9	—	Ethernet	✓
Подключение по локальной сети* 8	—	Ethernet	✓
Подключение по локальной сети* 7	—	Ethernet	✓
> Беспроводная сеть	—	Ethernet	✓
> Подключение по локальной сети* 10	—	Ethernet	✓
> Подключение по локальной сети* 1	—	Ethernet	✓
> VirtualBox Host-Only Network	—	Ethernet	✓
> Adapter for loopback traffic capture	—	BSD loopback	✓
> Подключение по локальной сети	—	Ethernet	✓

Рисунок 5 – Настройка на захват трафика по loopback

Запускаю исполняемый файл сервера “server.exe” и команду отслеживания всех (флаг “-a”) TCP-соединений и TCP и UDP порты, на которых компьютер прослушивается (Рисунок 6).

```

Администратор: Командная строка - server.exe
Microsoft Windows [Version 10.0.22000.1936]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Windows\system32>cd C:\Users\lkey2\Downloads\Смирнов\лаба 2
C:\Users\lkey2\Downloads\Смирнов\лаба 2>server.exe

C:\Windows\system32>netstat -a
Активные подключения
Имя    Локальный адрес    Внешний адрес    Состояние
TCP    0.0.0.0:135        LaptopDenis:0    LISTENING
TCP    0.0.0.0:445        LaptopDenis:0    LISTENING
TCP    0.0.0.0:5040       LaptopDenis:0    LISTENING
TCP    0.0.0.0:7070       LaptopDenis:0    LISTENING
TCP    0.0.0.0:27015      LaptopDenis:0    LISTENING
TCP    0.0.0.0:49664      LaptopDenis:0    LISTENING
TCP    0.0.0.0:49665      LaptopDenis:0    LISTENING
TCP    0.0.0.0:49666      LaptopDenis:0    LISTENING
TCP    0.0.0.0:49667      LaptopDenis:0    LISTENING
TCP    0.0.0.0:49668      LaptopDenis:0    LISTENING
TCP    0.0.0.0:49669      LaptopDenis:0    LISTENING
TCP    127.0.0.1:12025     LaptopDenis:0    LISTENING
TCP    127.0.0.1:12110     LaptopDenis:0    LISTENING
TCP    127.0.0.1:12119     LaptopDenis:0    LISTENING
TCP    127.0.0.1:12143     LaptopDenis:0    LISTENING
TCP    127.0.0.1:12465     LaptopDenis:0    LISTENING
TCP    127.0.0.1:12563     LaptopDenis:0    LISTENING
TCP    127.0.0.1:12993     LaptopDenis:0    LISTENING
TCP    127.0.0.1:12995     LaptopDenis:0    LISTENING
TCP    127.0.0.1:27275     LaptopDenis:0    LISTENING
TCP    169.254.114.52:139 LaptopDenis:0    LISTENING
TCP    192.168.31.34:139   LaptopDenis:0    LISTENING
TCP    192.168.31.34:49650 relay-daa16496:https ESTABLISHED
TCP    192.168.31.34:49871 20.54.37.73:https ESTABLISHED
TCP    192.168.31.34:49891 247:https        ESTABLISHED

```

Рисунок 6 – Запуск “server.exe” и “netstat -a”

При запуске клиента “client.exe” с аргументом “loopback” начинают передаваться данные (“DenisMorin”), и в консоль выводится число принятых и переданных символов (длина данных). После этого и клиент, и сервер прекращают соединение (Рисунок 7).

```

Microsoft Windows [Version 10.0.22000.1936]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Windows\system32>cd C:\Users\lkey2\Downloads\Смирнов\лаба 2
C:\Users\lkey2\Downloads\Смирнов\лаба 2>server.exe
Bytes received: 10
Bytes sent: 10
Connection closing...
C:\Users\lkey2\Downloads\Смирнов\лаба 2>server.exe
Bytes received: 10
Bytes sent: 10
Connection closing...
C:\Users\lkey2\Downloads\Смирнов\лаба 2>

Microsoft Windows [Version 10.0.22000.1936]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Windows\system32>cd C:\Users\lkey2\Downloads\Смирнов\лаба 2
C:\Users\lkey2\Downloads\Смирнов\лаба 2>client.exe loopback
Unable to connect to server!
C:\Users\lkey2\Downloads\Смирнов\лаба 2>client.exe loopback
Unable to connect to server!
C:\Users\lkey2\Downloads\Смирнов\лаба 2>client.exe loopback
Bytes Sent: 10
Bytes received: 10
Connection closed
C:\Users\lkey2\Downloads\Смирнов\лаба 2>

```

Рисунок 7 – Работа “client.exe” и “server.exe”

На рисунке 8 TCP-порт 27015 отображается как "LISTENING". Это означает, что серверный сокет слушает на этом порту и ожидает входящие данные.

```

C:\Windows\system32>netstat -a

Активные подключения

Имя      Локальный адрес      Внешний адрес      Состояние
TCP      0.0.0.0:135           LaptopDenis:0      LISTENING
TCP      0.0.0.0:445           LaptopDenis:0      LISTENING
TCP      0.0.0.0:5040          LaptopDenis:0      LISTENING
TCP      0.0.0.0:7070          LaptopDenis:0      LISTENING
TCP      0.0.0.0:27015         LaptopDenis:0      LISTENING
TCP      0.0.0.0:49664         LaptopDenis:0      LISTENING
TCP      0.0.0.0:49665         LaptopDenis:0      LISTENING
TCP      0.0.0.0:49666         LaptopDenis:0      LISTENING
TCP      0.0.0.0:49667         LaptopDenis:0      LISTENING
TCP      0.0.0.0:49668         LaptopDenis:0      LISTENING
TCP      0.0.0.0:49669         LaptopDenis:0      LISTENING
TCP      169.254.114.52:139    LaptopDenis:0      LISTENING
TCP      192.168.31.34:139     LaptopDenis:0      LISTENING
TCP      192.168.31.34:49650   relay-daa16496:https ESTABLISHED
TCP      192.168.31.34:49871   20.54.37.73:https   ESTABLISHED
TCP      192.168.31.34:52135   149.154.167.41:https ESTABLISHED
TCP      192.168.31.34:52335   a96-16-49-206:https CLOSE_WAIT
TCP      192.168.31.34:52339   a96-16-49-206:https CLOSE_WAIT

```

Рисунок 8 – Список активных соединений

В программе Process Monitor показывается, что по протоколу tcp сервером была получена информация длиной 10 байт, эта же информация отправлена клиентом, получено клиентом и затем отправлено сервером (Рисунок 9).

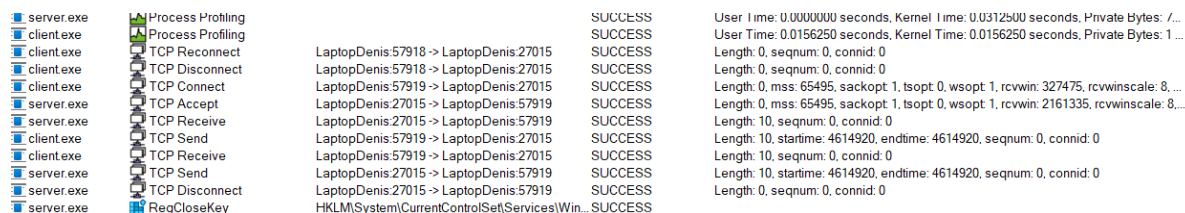


Рисунок 9 – Результат работы Process Monitor

С помощью Wireshark можно узнать, какая информация длиной 10 байт была передана (Рисунок 10).

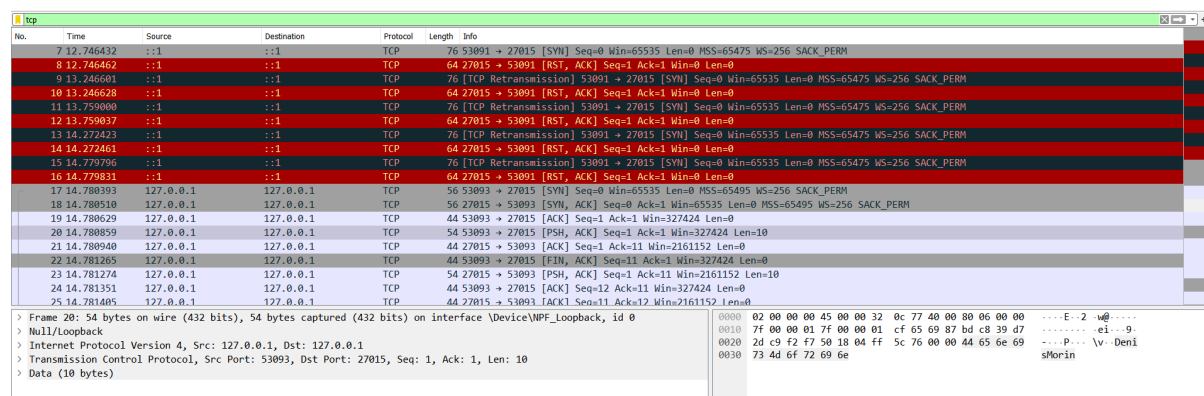
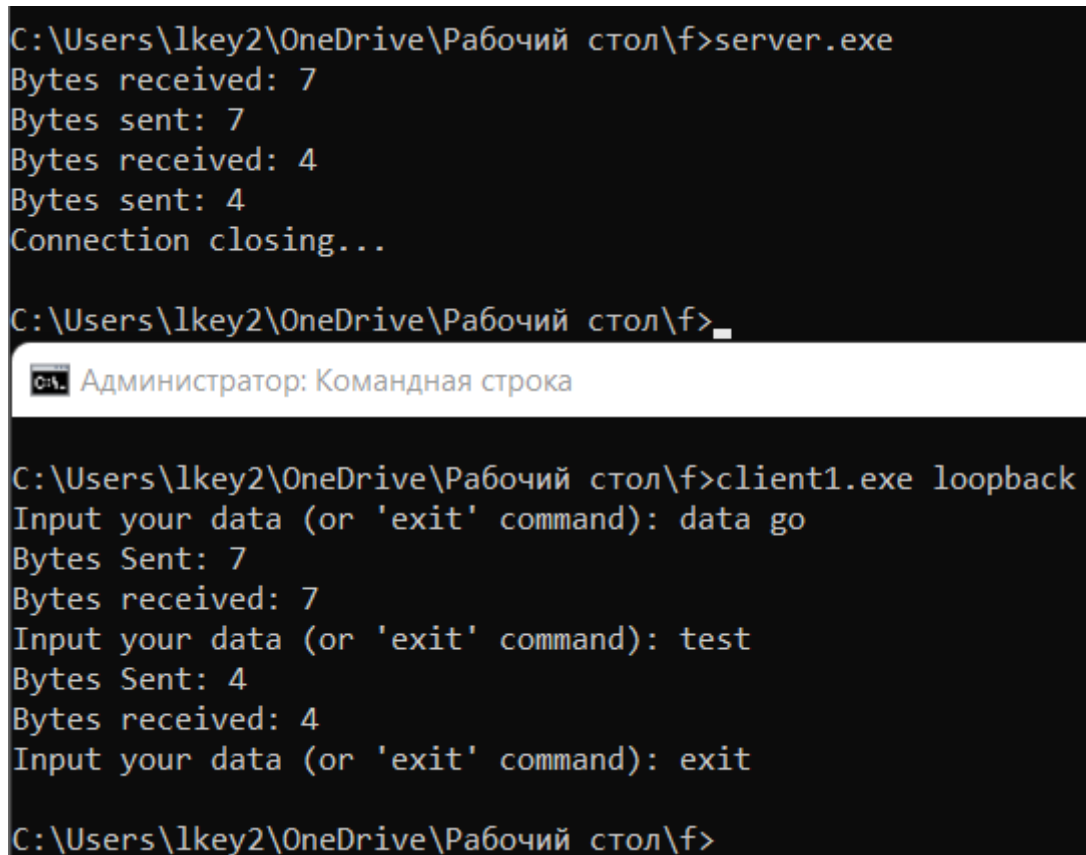


Рисунок 10 – Найдено сообщение “DeniMorin” в одном из пакетов при работе “server.exe” и “client.exe”

2.2 Дополнительное задание на 5 баллов

В качестве дополнительного задания был модифицирован код программы “client.c” (“client1.c”) таким образом, чтобы пользователь в консоли клиентского приложения мог вносить с клавиатуры те данные, которые нужно отправить (об этом будет выводиться подсказка в консоли). Для закрытия соединения пользователю нужно отправить команду (строку)

“exit”. На рисунке 11 представлена работа с измененной программой “client1.exe”.



```
C:\Users\lkey2\OneDrive\Рабочий стол\>server.exe
Bytes received: 7
Bytes sent: 7
Bytes received: 4
Bytes sent: 4
Connection closing...

C:\Users\lkey2\OneDrive\Рабочий стол\>_
Администратор: Командная строка

C:\Users\lkey2\OneDrive\Рабочий стол\>client1.exe loopback
Input your data (or 'exit' command): data go
Bytes Sent: 7
Bytes received: 7
Input your data (or 'exit' command): test
Bytes Sent: 4
Bytes received: 4
Input your data (or 'exit' command): exit

C:\Users\lkey2\OneDrive\Рабочий стол\>
```

Рисунок 11 – Отправка и получение данных, которые вводит пользователь, а также прекращение соединения по команде пользователя

Была изменена логика клиента: запрос на ввод данных или команды выхода (exit), выделение памяти для ввода строки с консоли, сравнение ввода с командой выхода и если пользователь ввел не exit, то его сообщение отправлялось серверу и получалось обратно от него (Листинг 1). Полный код программы представлен в приложении А.

```
do {
    memset(sendbuf, 0, sizeof(sendbuf));

    printf("Input your data (or 'exit' command): ");
    fgets(sendbuf, sizeof(sendbuf), stdin);
```

```

sendbuf[strcspn(sendbuf, "\n")] = '\0';

if (strcmp(sendbuf, "exit") == 0)
    break;

iResult = send(ConnectSocket, sendbuf, (int)strlen(sendbuf), 0);
if (iResult == SOCKET_ERROR) {
    printf("send failed with error: %d\n", WSAGetLastError());
    closesocket(ConnectSocket);
    WSACleanup();
    return 1;
}

printf("Bytes Sent: %ld\n", iResult);

memset(recvbuf, 0, sizeof(recvbuf));

iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);
if (iResult > 0) {
    printf("Bytes received: %d\n", iResult);
}
if (iResult == 0) {
    printf("Connection closed by server.\n");
}

}
while (strcmp(sendbuf, LOGOUT) != 0);

```

Листинг 1 – Измененная часть кода для “client1.c”

3 ЗАКЛЮЧЕНИЕ

В рамках данной практической работы были получены знания об основах работы с сетью в ОС Windows, а также навыки разработки программ.

ПРИЛОЖЕНИЕ А

Код файла “client1.c”

```
#define WIN32_LEAN_AND_MEAN

#ifdef _WIN32
#define OS_WIN32
#endif

/* ws2_32.dll has getaddrinfo and freeaddrinfo on Windows XP and later.
 * minwg32 headers check WINVER before allowing the use of these */
#ifndef WINVER
#define WINVER 0x0501
#endif

#include <windows.h>
#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define DEFAULT_BUFLen 512
#define DEFAULT_PORT "27015"
#define LOGOUT "exit"

int __cdecl main(int argc, char **argv)
{
    WSADATA wsaData;
    SOCKET ConnectSocket = INVALID_SOCKET;
    struct addrinfo *result = NULL,
        *ptr = NULL,
        hints;
    char sendbuf[DEFAULT_BUFLen];
    char recvbuf[DEFAULT_BUFLen];
    int iResult;
    int recvbuflen = DEFAULT_BUFLen;

    // Validate the parameters
    if (argc != 2) {
        printf("usage: %s server-name\n", argv[0]);
        return 1;
    }
}
```

```

}

// Initialize Winsock
iResult = WSASStartup(MAKEWORD(2,2), &wsaData);
if (iResult != 0) {
    printf("WSASStartup failed with error: %d\n", iResult);
    return 1;
}

ZeroMemory(&hints, sizeof(hints));
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
hints.ai_protocol = IPPROTO_TCP;

// Resolve the server address and port
iResult = getaddrinfo(argv[1], DEFAULT_PORT, &hints, &result);
if (iResult != 0) {
    printf("getaddrinfo failed with error: %d\n", iResult);
    WSACleanup();
    return 1;
}

// Attempt to connect to an address until one succeeds
for (ptr=result; ptr != NULL; ptr=ptr->ai_next) {

    // Create a SOCKET for connecting to server
    ConnectSocket = socket(ptr->ai_family, ptr->ai_socktype,
                           ptr->ai_protocol);
    if (ConnectSocket == INVALID_SOCKET) {
        printf("socket failed with error: %ld\n", WSAGetLastError());
        WSACleanup();
        return 1;
    }

    // Connect to server.
    iResult = connect(ConnectSocket, ptr->ai_addr,
(int)ptr->ai_addrlen);
    if (iResult == SOCKET_ERROR) {
        closesocket(ConnectSocket);
        ConnectSocket = INVALID_SOCKET;
        continue;
    }
}

```

```

    }
    break;
}

freeaddrinfo(result);

if (ConnectSocket == INVALID_SOCKET) {
    printf("Unable to connect to server!\n");
    WSACleanup();
    return 1;
}

do {
    memset(sendbuf, 0, sizeof(sendbuf));

    printf("Input your data (or 'exit' command): ");
    fgets(sendbuf, sizeof(sendbuf), stdin);

    sendbuf[strcspn(sendbuf, "\n")] = '\0';

    if (strcmp(sendbuf, "exit") == 0)
        break;

    iResult = send(ConnectSocket, sendbuf, (int)strlen(sendbuf), 0);
    if (iResult == SOCKET_ERROR) {
        printf("send failed with error: %d\n", WSAGetLastError());
        closesocket(ConnectSocket);
        WSACleanup();
        return 1;
    }

    printf("Bytes Sent: %ld\n", iResult);

    memset(recvbuf, 0, sizeof(recvbuf));

    iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);
    if (iResult > 0) {
        printf("Bytes received: %d\n", iResult);
    }
    if (iResult == 0) {
        printf("Connection closed by server.\n");
    }
}

```

```

    }

}

while (strcmp(sendbuf, LOGOUT) != 0);

// shutdown the connection since no more data will be sent
iResult = shutdown(ConnectSocket, SD_SEND);
if (iResult == SOCKET_ERROR) {
    printf("shutdown failed with error: %d\n", WSAGetLastError());
    closesocket(ConnectSocket);
    WSACleanup();
    return 1;
}

// cleanup
closesocket(ConnectSocket);
WSACleanup();

return 0;
}

```