

ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

Ассистент

должность, уч. степень, звание

подпись, дата

Д.А. Кочин

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

МЕЖСЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ МЕЖДУ ПРОЦЕССАМИ

по курсу: ОПЕРАЦИОННЫЕ СИСТЕМЫ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

4232

подпись, дата

М.Д. Порохняк

инициалы, фамилия

Санкт-Петербург 2024

1. Цель работы

Организация межсетевого взаимодействия средствами WinAPI и POSIX.

2. Задание

Требуется организовать взаимодействие типа клиент-сервер средствами WinAPI и POSIX в соответствии с индивидуальным заданием.

Номер варианта сохранить в файле TASKID.txt в корне репозитория.

В разделе issues создать не менее трех задач. Решить задачи, разметив написанный код в репозитории. Каждый коммит должен быть привязан к конкретной задаче.

Код приложения клиента разместить в файле client.cpp. Код приложения сервера в server.cpp.

Осуществить рефакторинг кода в соответствии с Google C++ Style Guide.

Исходный вариант задания указан в таблице 2.1.

Таблица 2.1 Исходный вариант

Вариант	Инд. Зад.	Протокол	Сервер	Клиент
19	12	TCP	Windows	Linux

3. Ход работы

В разделе issues были созданы задачи со следующими заголовками: “Рефакторинг разработанных приложений”, “Разработка приложения клиента”, “Разработка приложения сервера”. Сами issues и их содержимое приведено на рисунках 3.1, 3.2, 3.3, 3.4.

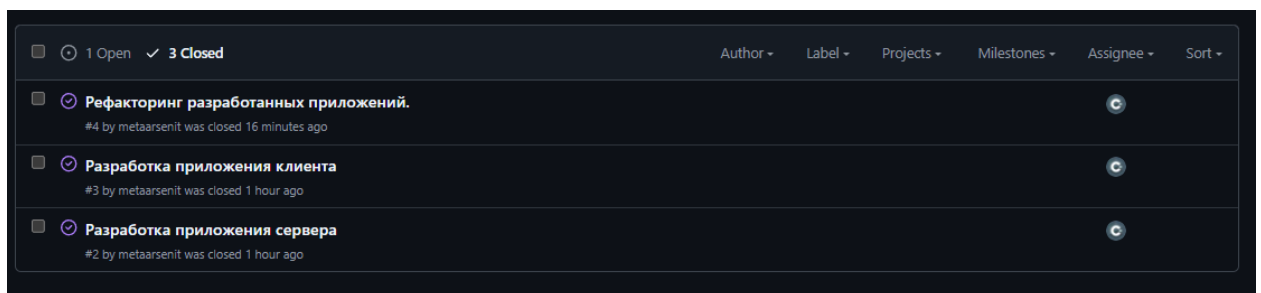
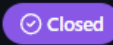



Рисунок 3.1 Закрытые issues


Рефакторинг разработанных приложений. #4


 metaarsenit opened this issue 1 hour ago · 0 comments




metaarsenit commented 1 hour ago

Выполняется после [#2](#) и [#3](#). Требуется осуществить рефакторинг кода `client.cpp` и `server.cpp` в соответствии с [Google C++ Style Guide](#)






metaarsenit self-assigned this 1 hour ago




metaarsenit added a commit that referenced this issue 33 minutes ago




Attempt to refactor `server.cpp` and `client.cpp` for issue [#4](#)

5190dce




metaarsenit added a commit that referenced this issue 33 minutes ago




Attempt to refactor `server.cpp` and `client.cpp` for issue [#4](#)

✖ 043dffe




metaarsenit added a commit that referenced this issue 26 minutes ago




Second attempt to refactor `client.cpp` and `server.cpp` for issue [#4](#)

✖ df2a795




metaarsenit added a commit that referenced this issue 23 minutes ago




Third attempt, refactored `server.cpp` for issues [#4](#)

✖ 4bc4cf6




metaarsenit added a commit that referenced this issue 19 minutes ago




Added `include<cstdio>` in examples to pass linter test for issue [#4](#)

✖ 200e8a6




metaarsenit added a commit that referenced this issue 17 minutes ago



Fixed typo in `examples/linux/receiver.c` to pass linter [#4](#)

✖ d0c50ea



metaarsenit closed this as completed 16 minutes ago

Рисунок 3.2 Рефакторинг разработанных приложений

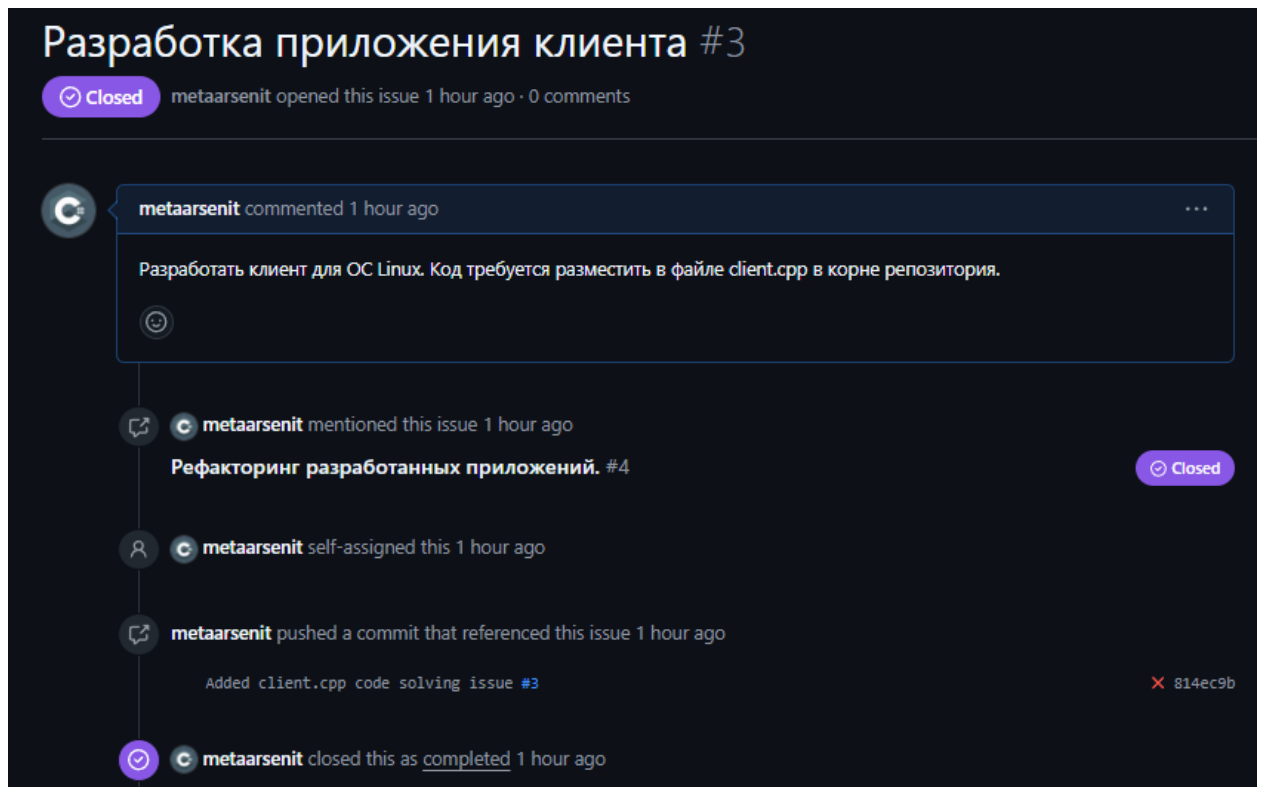


Рисунок 3.3 Разработка приложения клиента

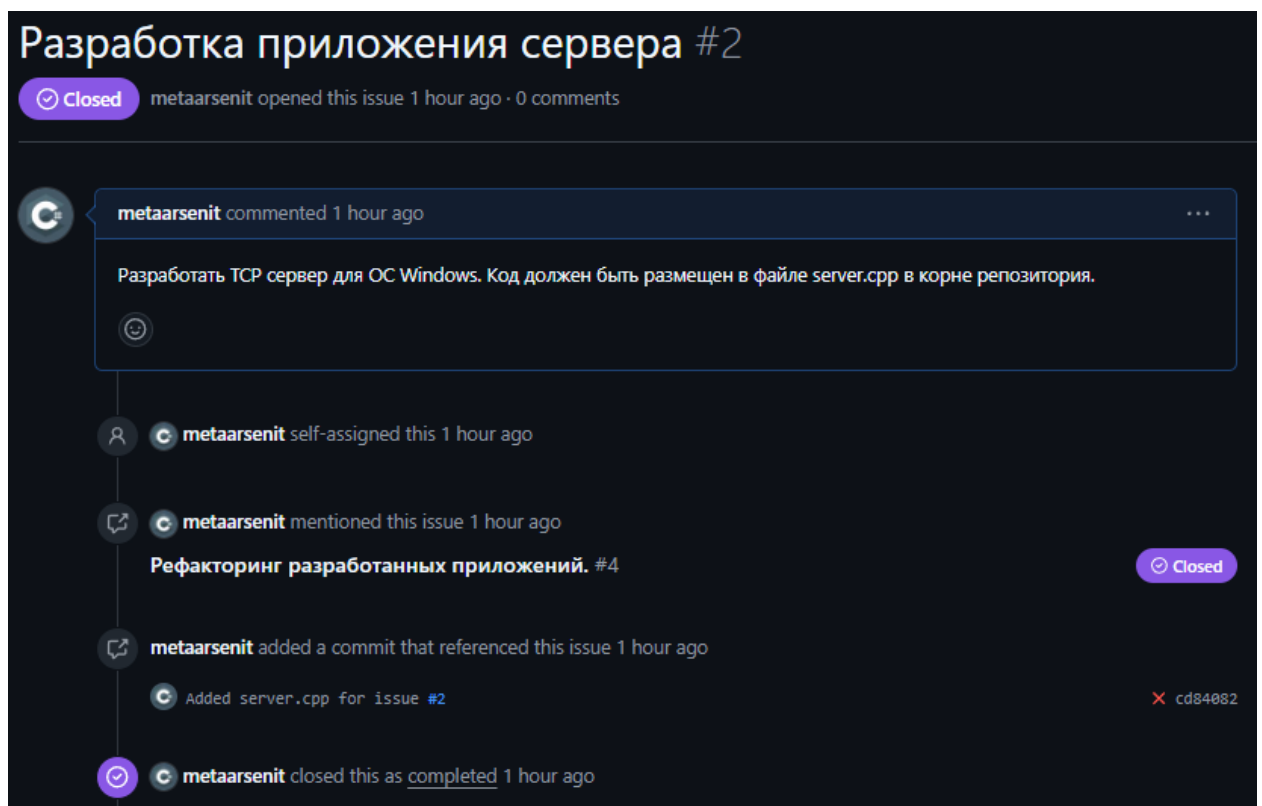
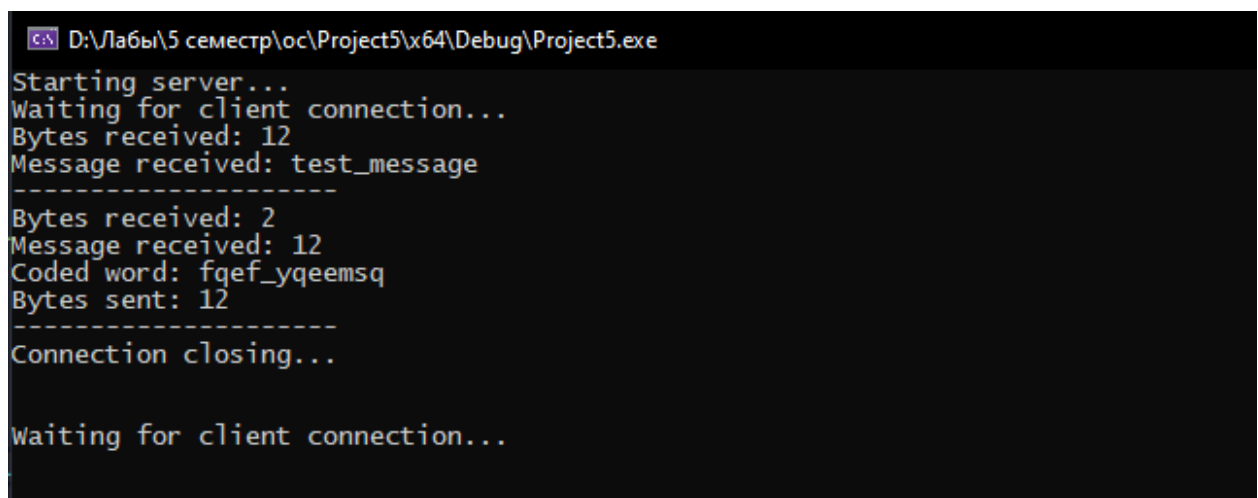


Рисунок 3.4 Разработка приложения сервера

В ходе выполнения лабораторной работы были написаны файлы client.cpp (TCP Linux клиент) и server.cpp (TCP Windows сервер). Их полный код приведен в приложении А.

Проверим работоспособность. Для этого запустим сервер на хост машине с Windows, а клиент внутри Oracle VM VirtualBox с ОС Ubuntu. Для создания локальной сети воспользуемся Logmein Hamachi.

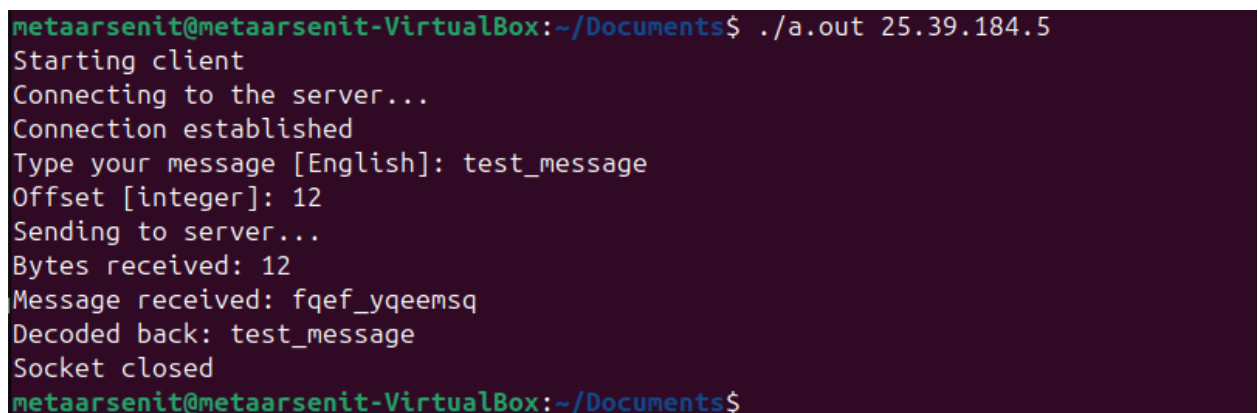
Примеры работы сервера и клиента продемонстрированы на рисунках 3.5 и 3.6.



```
C:\> D:\Лаб\5 семестр\ос\Project5\x64\Debug\Project5.exe
Starting server...
Waiting for client connection...
Bytes received: 12
Message received: test_message
-----
Bytes received: 2
Message received: 12
Coded word: fqef_yqeemsg
Bytes sent: 12
-----
Connection closing...

Waiting for client connection...
```

Рисунок 3.5 Вывод консоли сервера



```
metaarsenit@metaarsenit-VirtualBox:~/Documents$ ./a.out 25.39.184.5
Starting client
Connecting to the server...
Connection established
Type your message [English]: test_message
Offset [integer]: 12
Sending to server...
Bytes received: 12
Message received: fqef_yqeemsg
Decoded back: test_message
Socket closed
metaarsenit@metaarsenit-VirtualBox:~/Documents$
```

Рисунок 3.6 Вывод консоли клиента

4. Выводы

В результате выполнения данной лабораторной работы были получены навыки и умения создания клиент-серверных приложений с использованием средств WinAPI и POSIX. Был написан клиент для ос LINUX и сервер для ос WINDOWS с использованием протокола TCP.

ПРИЛОЖЕНИЕ А

client.cpp

```
// Copyright 2025 metaarsenit
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<cstring>
#include<string>
#include<iostream>

char* decode(const char* word, int N, int length) {
    if (word == nullptr)
        return nullptr;

    char* answer = new char[length + 1];
    for (int i = 0; i < length; i++) {
        int value = static_cast<int>(word[i]);
        // 65-90 A-Z
        if (65 <= value && value <= 90) {
            answer[i] = (abs((value - 65 + N * 25) % 26)) + 65;
        } else if (97 <= value && value <= 122) {
            // 97-122 a-z
            answer[i] = (abs((value - 97 + N * 25) % 26)) + 97;
        } else {
            // special characters are unchanged
            answer[i] = static_cast<char>(value);
        }
    }
}
```

```
    answer[length] = '\0';
    return answer;
}

using std::cout, std::cin;
using std::string, std::to_string;

int main(int argc, char* argv[]) {
    cout << "Starting client\n";

    int clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    sockaddr_in serverAddress;
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(8080);
    serverAddress.sin_addr.s_addr = inet_addr(argv[1]);

    cout << "Connecting to the server...\n";
    connect(clientSocket,
            (struct sockaddr*)&serverAddress,
            sizeof(serverAddress));
    cout << "Connection established\n";

    cout << "Type your message [English]: ";
    string _message;
    cin >> _message;
    cout << "Offset [integer]: ";
    unsigned int _offset;
    cin >> _offset;
```



```

const char* offset = to_string(_offset).c_str();
cout << "Sending to server...\n";
const char* message = _message.c_str();
send(clientSocket, message, strlen(message), 0);
send(clientSocket, offset, strlen(offset), 0);

int buflen = 1024;
char recvbuf[1024];

int iResult = recv(clientSocket, recvbuf, buflen, 0);
if (iResult > 0) {
    cout << "Bytes received: " << iResult << "\n";
    recvbuf[iResult] = '\0';
    cout << "Message received: " << recvbuf << "\n";
    cout << "Decoded back: ";
    cout << decode(recvbuf, _offset, strlen(recvbuf)) << "\n";
}
close(clientSocket);
cout << "Socket closed\n";

return 0;
}

```

server.cpp

```

// Copyright 2025 metaarsenit
#undef UNICODE

#define WIN32_LEAN_AND_MEAN

#include <windows.h>

```

```

#include <winsock2.h>
#include <ws2tcpip.h>
#include <stdlib.h>
#include <stdio.h>
#include<iostream>

#pragma comment(lib, "Ws2_32.lib")

#define DEFAULT_BUFLen 1024
#define DEFAULT_PORT "8080"

// Cesar algorithm
char* code(const char* word, int N, int length) {
    if (word == nullptr)
        return nullptr;

    char* answer = new char[length+1];
    for (int i = 0; i < length; i++) {
        int value = static_cast<int>(word[i]);
        // 65-90 A-Z
        if (65 <= value && value <= 90) {
            answer[i] = ((value - 65 + N) % 26) + 65;
        } else if (97 <= value && value <= 122) {
            // 97-122 a-z
            answer[i] = ((value - 97 + N) % 26) + 97;
        } else {
            // special characters are unchanged
            answer[i] = static_cast<char>(value);
        }
    }
}

```

```
    answer[length] = '\0';
    return answer;
}

// https://learn.microsoft.com/ru-ru/windows/win32/winsock/finished-server-and-client-code
int main() {
    std::cout << "Starting server...\n";

    WSADATA wsaData;
    int iResult;

    SOCKET ListenSocket = INVALID_SOCKET;
    SOCKET ClientSocket = INVALID_SOCKET;

    struct addrinfo* result = NULL;
    struct addrinfo hints;

    int iSendResult;
    char recvbuf[DEFAULT_BUFLen];
    int recvbuflen = DEFAULT_BUFLen;

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

    ZeroMemory(&hints, sizeof(hints));
```

```
hints.ai_family = AF_INET;

hints.ai_socktype = SOCK_STREAM;

hints.ai_protocol = IPPROTO_TCP;

hints.ai_flags = AI_PASSIVE;


// Resolve the server address and port

iResult = getaddrinfo(NULL, DEFAULT_PORT, &hints, &result);
if (iResult != 0) {
    std::cout << "getaddrinfo failed with error: " << iResult << "\n";
    WSACleanup();
    return 1;
}


// Create a SOCKET for the server to listen for client connections.
ListenSocket = socket(result->ai_family,
                      result->ai_socktype,
                      result->ai_protocol);

if (ListenSocket == INVALID_SOCKET) {
    std::cout << "socket failed with error: ";
    std::cout << WSAGetLastError() << "\n";
    freeaddrinfo(result);
    WSACleanup();
    return 1;
}


// Setup the TCP listening socket
iResult = bind(ListenSocket,
              result->ai_addr,
              static_cast<int>(result->ai_addrlen));

if (iResult == SOCKET_ERROR) {
```

```
std::cout << "bind failed with error: ";
std::cout << WSAGetLastError() << "\n";
freeaddrinfo(result);
closesocket(ListenSocket);
WSACleanup();
return 1;
}
```

```
freeaddrinfo(result);
```

```
iResult = listen(ListenSocket, SOMAXCONN);
if (iResult == SOCKET_ERROR) {
    std::cout << "listen failed with error: ";
    std::cout << WSAGetLastError() << "\n";
    closesocket(ListenSocket);
    WSACleanup();
    return 1;
}
```

```
while (true) {
    std::cout << "Waiting for client connection...\n";

    // Accept a client socket
    ClientSocket = accept(ListenSocket, NULL, NULL);
    if (ClientSocket == INVALID_SOCKET) {
        std::cout << "accept failed with error: ";
        std::cout << WSAGetLastError() << "\n";
        closesocket(ListenSocket);
```

```

WSACleanup();

return 1;

}

// Receive until the peer shuts down the connection
int N = 0;
char* word = nullptr;
int c = 0;

do {
    iResult = recv(ClientSocket, recvbuf, recvbuflen, 0);
    if (iResult > 0) {
        std::cout << "Bytes received: " << iResult << "\n";
        recvbuf[iResult] = '\0';
        std::cout << "Message received: " << recvbuf << "\n";
        // Echo the buffer back to the sender
        if (c == 0) {
            word = new char[iResult+1];
            for (int i = 0; i < iResult; i++) {
                word[i] = recvbuf[i];
            }
            word[iResult] = '\0';
        }

        if (c == 1) {
            char* number = new char[iResult];
            for (int i = 0; i < iResult; i++) {
                number[i] = recvbuf[i];
            }
            int N = atoi(number);

```

```

        delete[] number;

        char* codedWord = code(word, N, strlen(word));
        if (codedWord == nullptr)
            break;

        std::cout << "Coded word: " << codedWord << "\n";

        iSendResult = send(ClientSocket,
                           codedWord,
                           strlen(codedWord), 0);
        if (iSendResult == SOCKET_ERROR) {
            std::cout << "send failed with error: ";
            std::cout << WSAGetLastError() << "\n";
            closesocket(ClientSocket);
            WSACleanup();
            return 1;
        }

        std::cout << "Bytes sent: ";
        std::cout << iSendResult << "\n";
    }

    c++;
    std::cout << "-----\n";
} else if (iResult == 0) {
    std::cout << "Connection closing...\n";
} else {
    std::cout << "recv failed with error: ";
    std::cout << WSAGetLastError() << "\n";
}

```

```
        closesocket(ClientSocket);
        WSACleanup();
        return 1;
    }
} while (iResult > 0);

delete[] word;
std::cout << "\n\n";
// shutdown the connection
iResult = shutdown(ClientSocket, SD_SEND);
if (iResult == SOCKET_ERROR) {
    std::cout << "shutdown failed with error: ";
    std::cout << WSAGetLastError() << "\n";
    closesocket(ClientSocket);
    WSACleanup();
    return 1;
}
}

closesocket(ListenSocket);
closesocket(ClientSocket);
WSACleanup();

return 0;
}
```