Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

09.03.01 Информатика и вычислительная техника

Профиль: Программное обеспечение средств вычислительной техники и автоматизированных систем

**Отчёт по курсовой работе**

**по дисциплине «Операционные системы»**

**по теме**

**«Многопользовательский сетевой чат»**

Выполнил: студент 3 курса гр. ИП-814

Краснов И.В.

Проверил: преподаватель кафедры ПМиК

Нужнов А.В.

**Оглавление**

[**1.** **Постановка задачи** 3](#_Toc59463436)

[**1.1** **Цель создания** 3](#_Toc59463437)

[**1.2** **Перечень решаемых задач** 3](#_Toc59463438)

[**2.** **Используемое программное обеспечение** 4](#_Toc59463439)

[**3.** **Результаты работы программы** 5](#_Toc59463440)

[**4.** **Описание работы алгоритма** 9](#_Toc59463441)

[**4.1** **Клиентская часть** 9](#_Toc59463442)

[**4.2** **Серверная часть** 9](#_Toc59463443)

[**5.** **Листинг** 11](#_Toc59463444)

[**5.1** **Серверная часть** 11](#_Toc59463445)

[**5.2** **Клиентская часть** 13](#_Toc59463446)

[**6.** **Ссылки** 19](#_Toc59463447)

# **Постановка задачи**

## **Цель создания**

Главная цель курсовой работы по теме «Сетевой многопользовательский чат» - создание многопользовательского сетевого чата для обмена текстовыми сообщениями между пользователями.

## **Перечень решаемых задач**

Функционал данного чата должен предусматривать:

* обмен текстовыми сообщениями между пользователями чата
* подключение двух и более пользователей к чату

# **Используемое программное обеспечение**

В процессе реализации данного курсового проекта будут использованы:

**Developer Command Prompt for VS 2019 –** данная утилита будет использована для компиляции и сборки серверной части проекта. Она предоставляет удобный оптимизирующий компилятор.

**Dev-C++ 5.11** – данная IDE будет использована для компиляции и сборки клиентской части курсового проекта, а также для редактирования исходного кода серверной части. IDE предоставляет аппаратные средства для редактирования, компиляции, сборки проектов на языках C и С++. Дополнительно Dev-C++ предоставляет возможность создавать проекты с использованием нескольких файлов с исходным кодом, которые будут компилироваться и собираться совместно и для которых будет написан Makefile. Dev-C++ поставляется с предустановленным компилятором TDM-GCC 4.9.2 для 64-разрядных и 32-разрядных операционных систем.

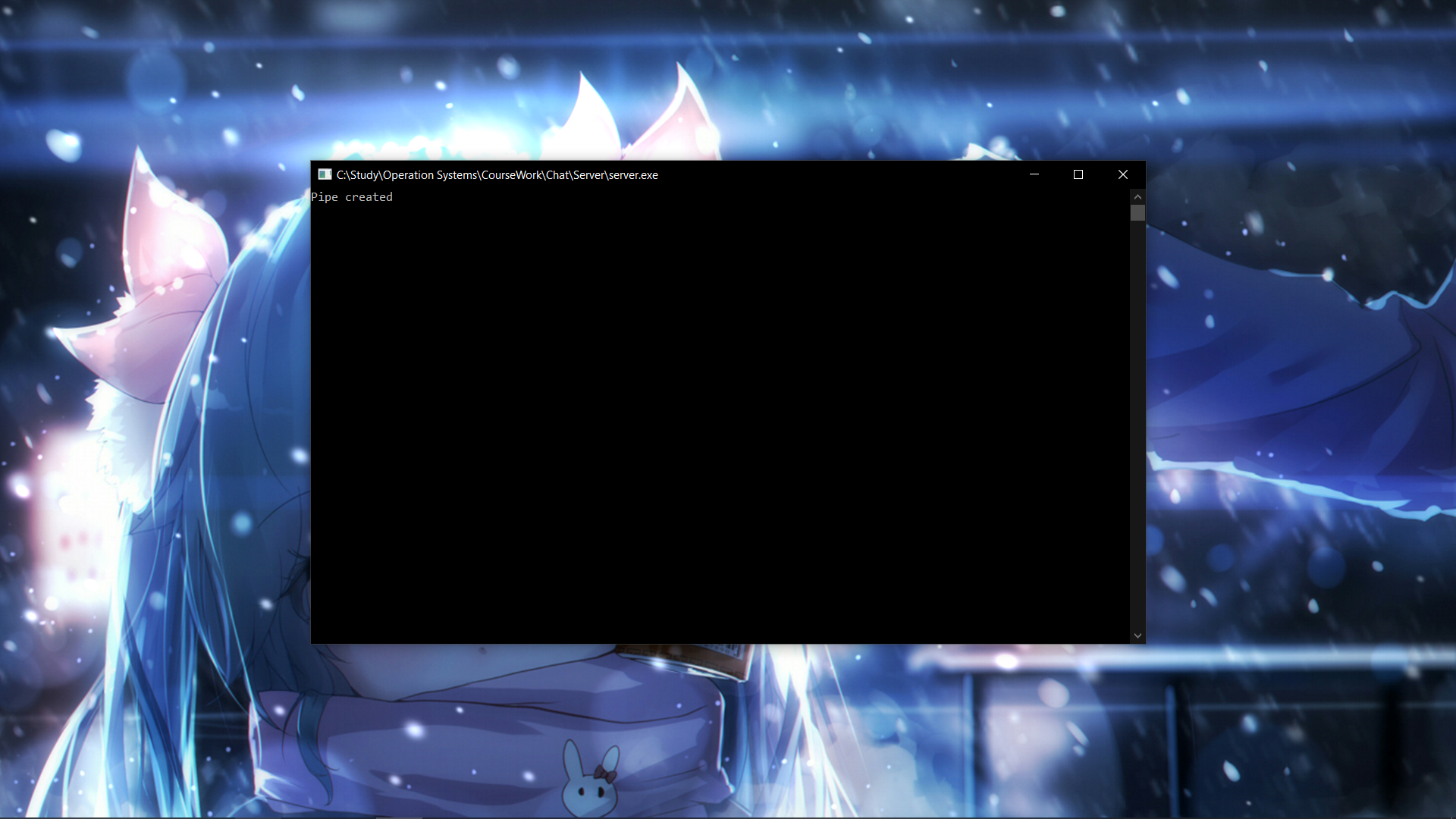
**Windows API** – данный набор функций интерфейсов предоставляет возможность создания приложений с интерфейсом на операционных системах семейства Microsoft Windows. В курсовом проекте данный набор будет использован для создания интерфейса клиентской части для удобства пользования.

Курсовой проект реализован на операционной системе Windows 10. Интерфейс содержат только клиентские приложения. Серверная часть не имеет интерфейса и работает в невидимом для пользователя режиме.

# **Результаты работы программы**

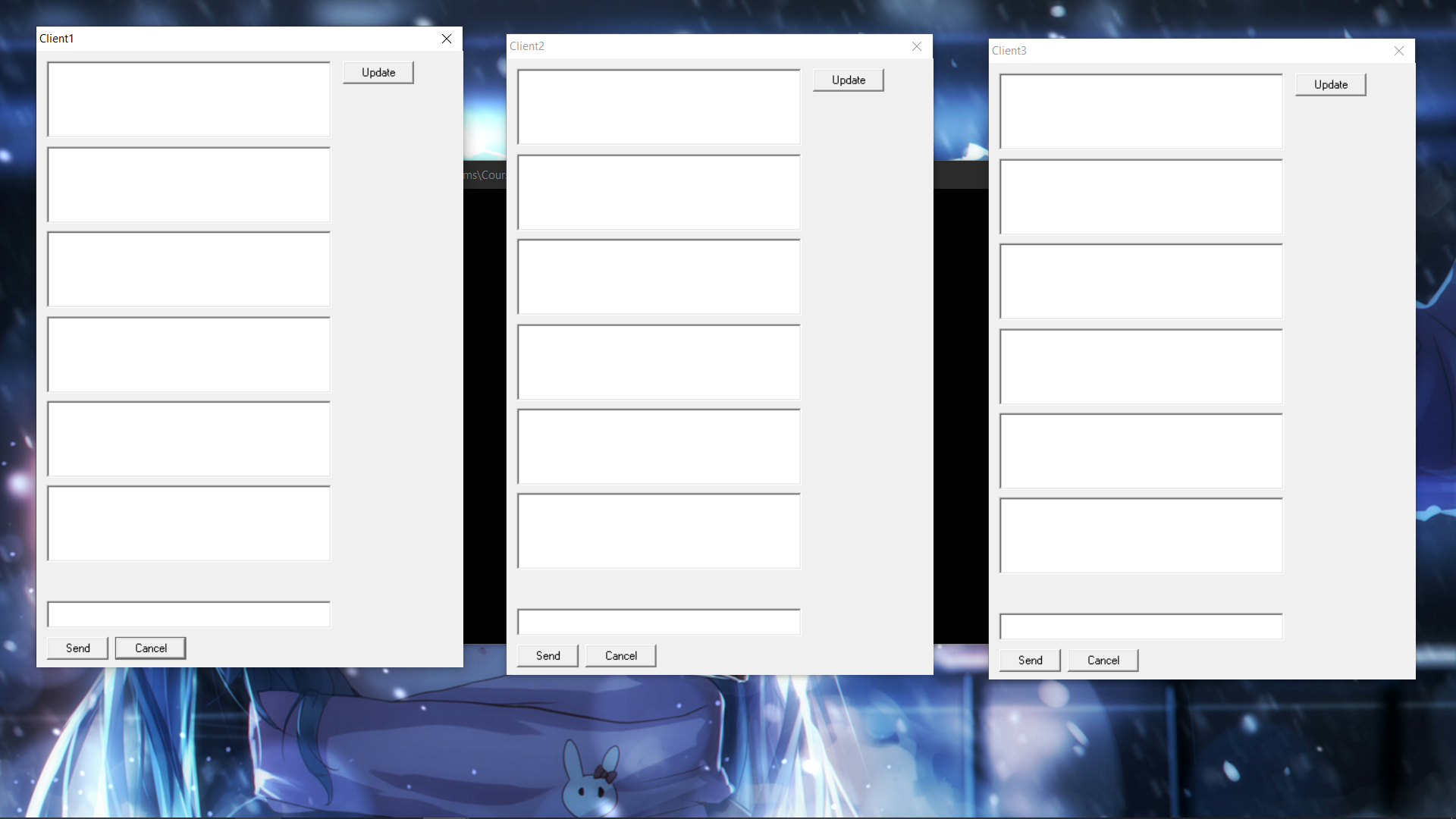
Ниже будут представлены процесс запуска проекта и его работа.

Запуск сервера. Сервер сообщит, что создал канал для подключения клиента.



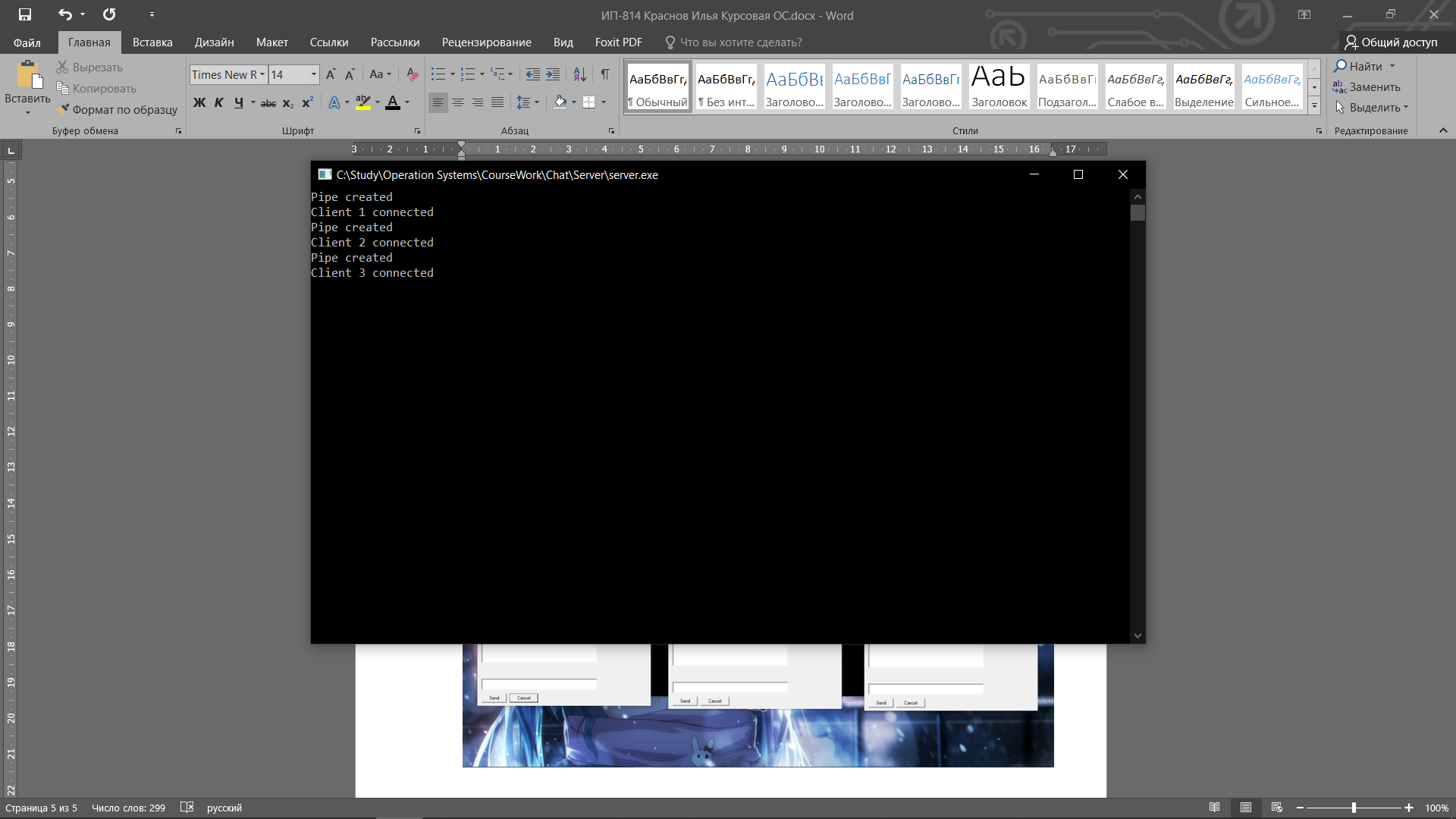
(рис. 1) Сервер после запуска

Запускаем клиенты. Клиенты в своих консолях отобразят, что успешно подключились к серверу или наоборот не смогли этого сделать



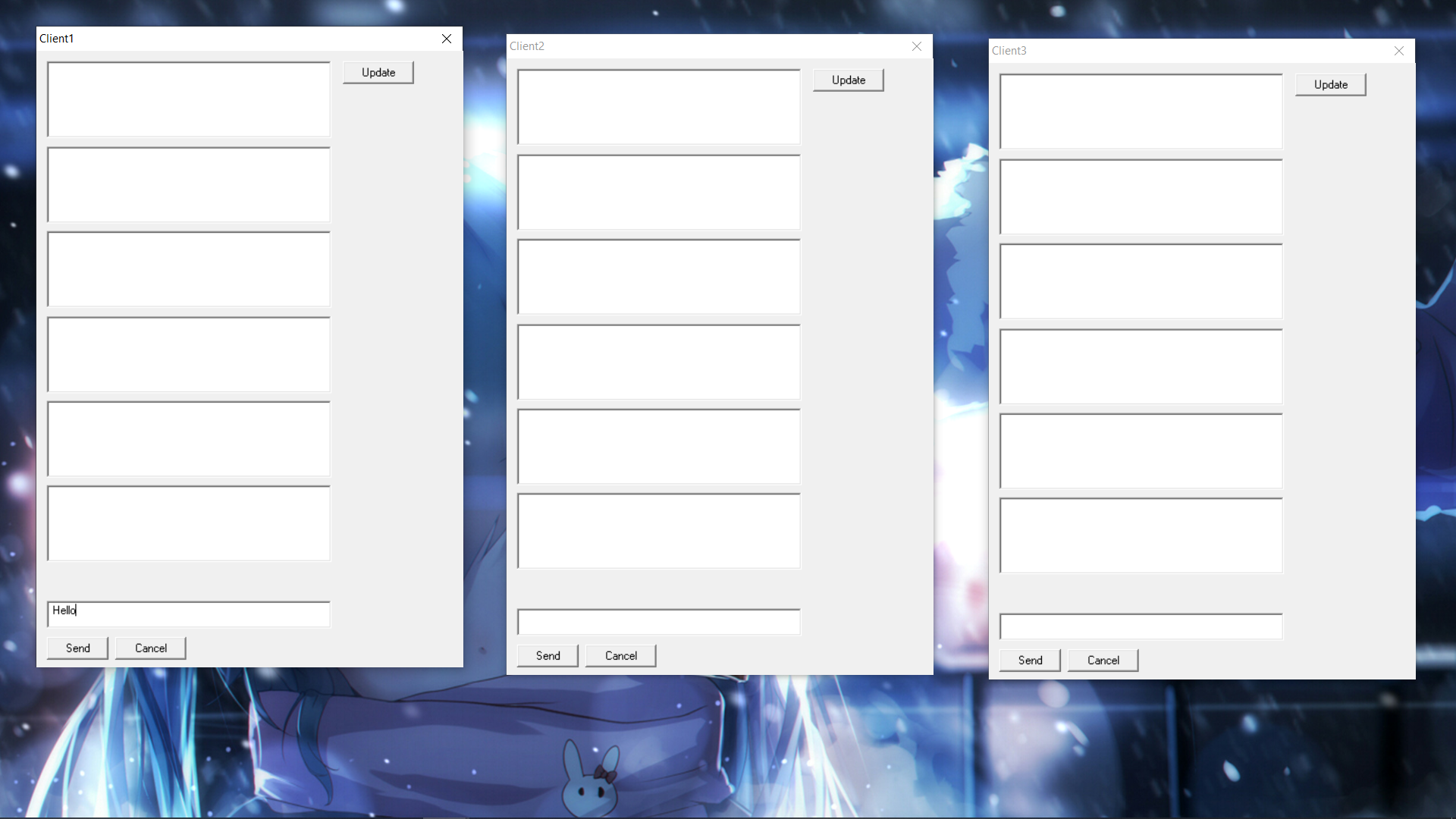
(рис. 2) Клиенты после запуска

Сервер должен выводить уведомление об успешном или неудавшемся подключении клиента.



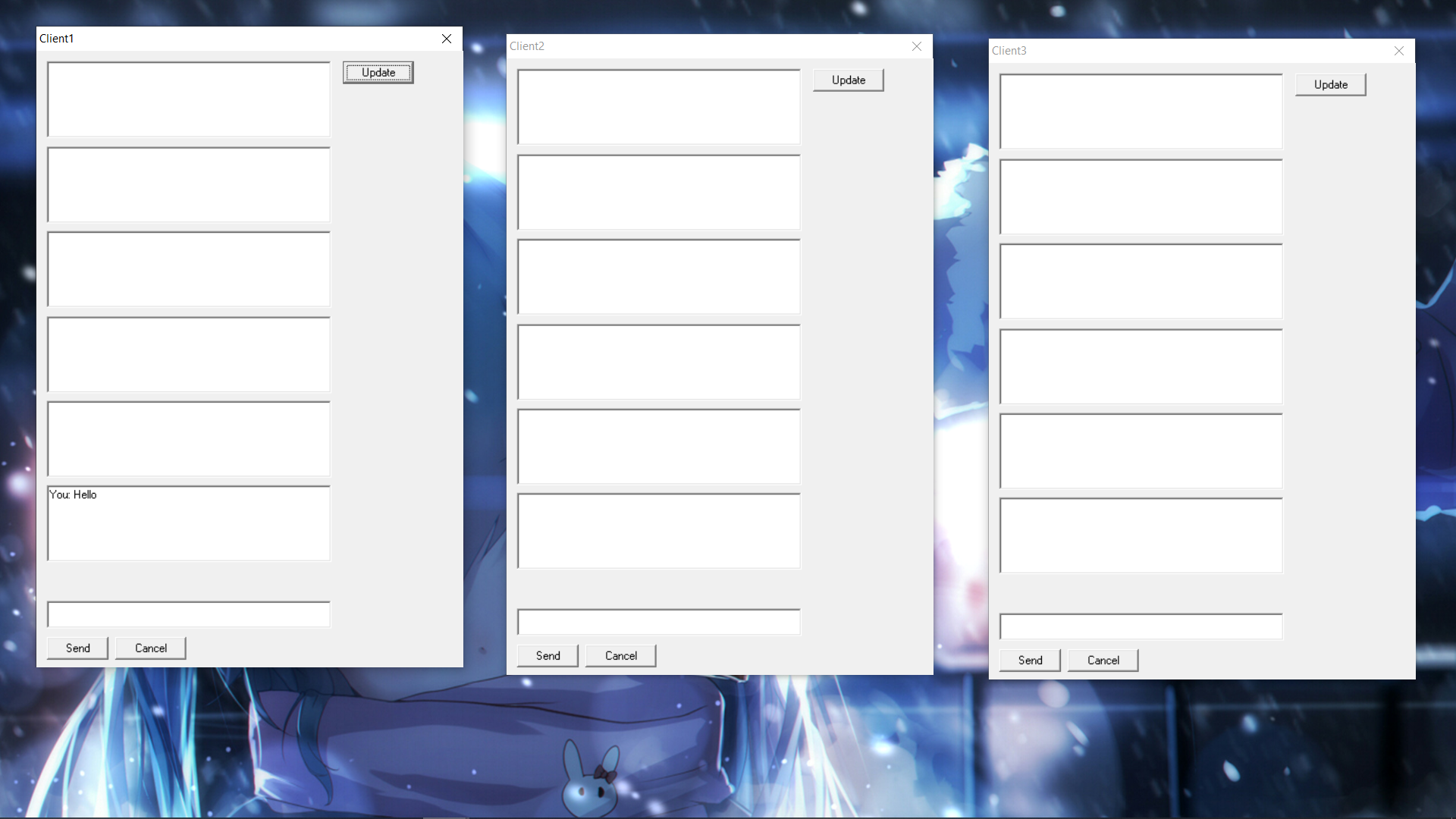
(рис. 3) Сервер после подключения клиентов

Всё готово для общения между клиентами. Для отправки сообщения нужно ввести текст сообщения в нижнее поле и нажать кнопку «Send».



(рис. 4) Отправка сообщения

Для получения сообщения необходимо нажать кнопку «Update»



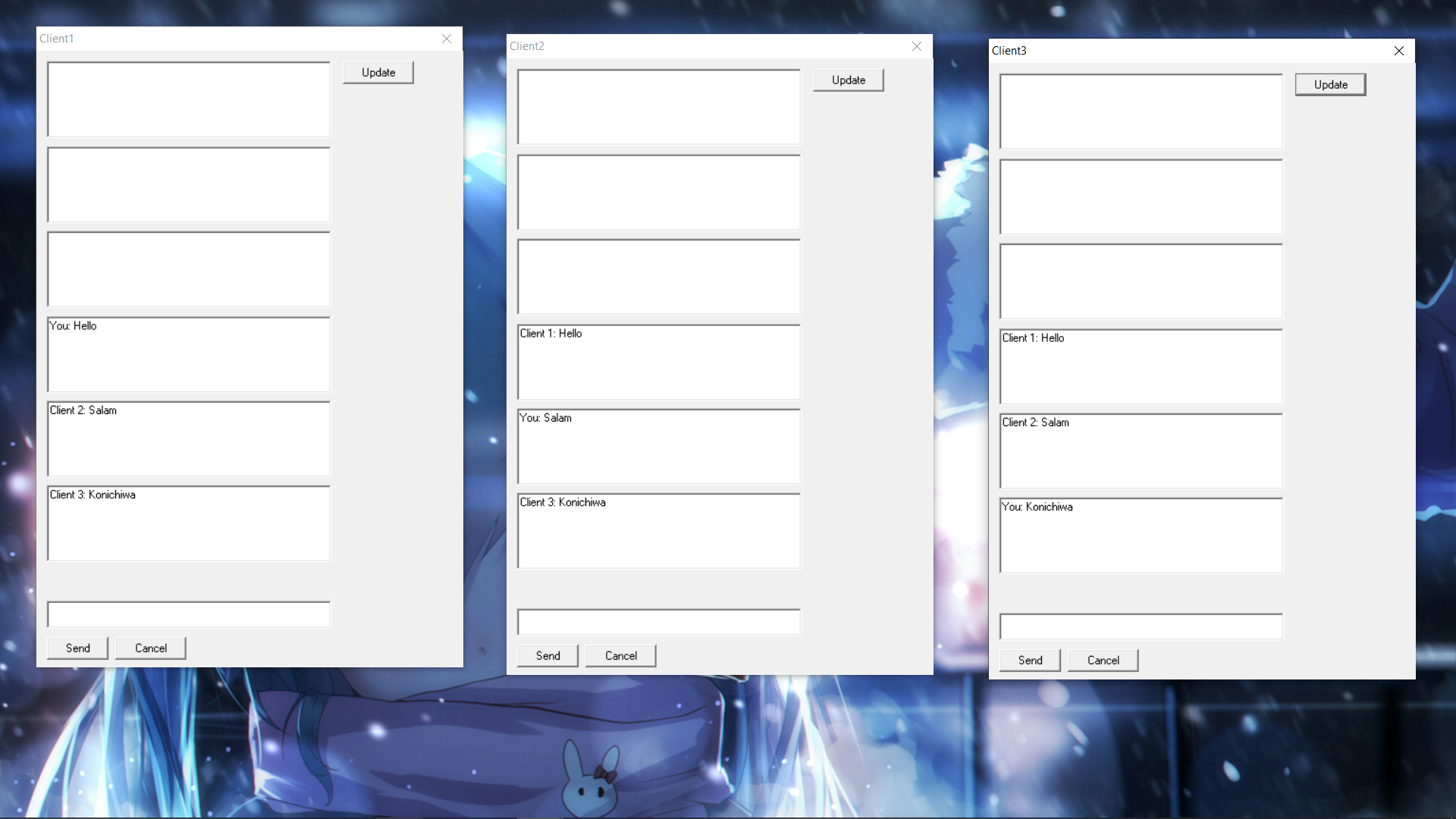
(рис. 5) Получение сообщения клиентом – отправителем.

Остальные клиенты также получат сообщения, когда нажмут на кнопку обновления.



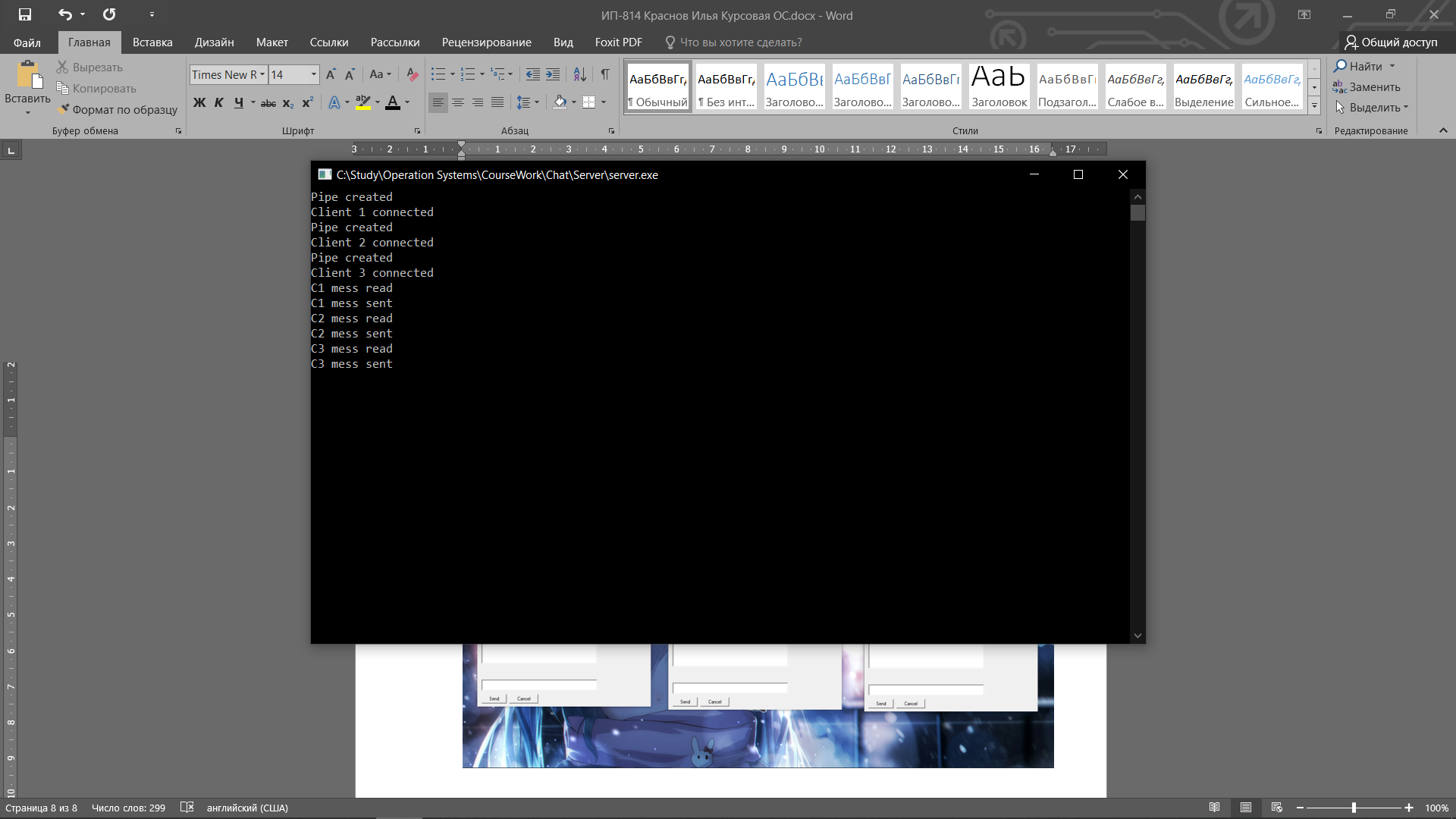
(рис. 6) Получение сообщения клиентами

Аналогично остальные клиенты могут писать сообщения в общий чат.



(рис. 7) Переписка между клиентами

Сервер будет выводить информацию о том, от кого он получает сообщение и удалось ли ему это соообщение разослать.



(рис. 8) Вывод сервера во время общения клиентов

# **Описание работы алгоритма**

## **Клиентская часть**

При запуске клиент начинает поиск доступного канала для связи с сервером. Имя канала предоставлено заранее в виде переменной типа LPSTR. Независимо от успешности подключения, клиент выводит графический интерфейс. Возможность общения доступна только при успешном подключении. Интерфейс клиента состоит из 7 текстовых полей и 3 кнопок. Нижнее текстовое поле служит для ввода сообщения, которое будет отправлено. Верхние 6 полей работают как история сообщений. При поступлении сообщения, новое сообщение выводится в самое нижнее поле из этих 6. Сообщение, которое ранее находилось в нижнем поле, поднимется на 1 поле вверх. Так произойдёт с каждым сообщением в поле. Сообщение из верхнего поля будет потеряно. Программа способна хранить не более 6 последних сообщений. Для получения сообщения необходимо нажать кнопку «Update». При её нажатии программа заходит в канал, через который осуществляется общение с сервером. При наличии там сообщения она выводит его в нижнее поле и производит сдвиг ранее полученных сообщений, сохранённых в этих полях. Получение сообщения невозможно, если через канал не было передано сообщение. Для отправки сообщения необходимо нажать кнопку «Send». При её нажатии программа извлекает текст из поля ввода сообщения и отправляет его серверу через канал. Текстовое поле ввода сообщения отчищается. Отправка сообщений идёт поочерёдно между клиентами. Клиент не моет отправить сообщение вне очереди. Длинна сообщения ограничивается 255 символами. При нажатии на кнопку «Cancel» клиентская программа завершит свою работу.

## **Серверная часть**

Сервер не имеет графического интерфейса, не имеет возможности к взаимодействию с собой со стороны сторонних лиц. Сервер выводит информацию о создании каналов, подключении клиентов, получении сообщения через канал и его отправке пользователям. Количество ожидаемых клиентов и имена каналов для них предопределены заранее в виде переменных типа int и LPSTR соответственно. Поочерёдно создаёт канал для подключения клиентов и ожидает подключения. Сервер выводит информацию о создании канала и подключении клиента. После подключения указанного количества клиентов, сервер начинает передачу сообщений. Он поочерёдно заходит в каждый канал и ожидает сообщения от клиента. В один момент времени сервер может просматривать только один канал, потому общение происходит по очереди, начиная с первого клиента. Номер клиента определяется номером переменной с именем канала для клиента, которая хранится в массиве имён каналов типа LPSTR. При получении сообщения от клиента, сервер просматривает из какого канала было получено сообщение и запоминает это. Далее сервер начинает рассылку сообщений через все каналы. Если сообщение отправляется не по каналу, через который сообщение пришло, то в начале сообщения дополнительно будет приписано имя клиента-отправителя. Если сообщение отправляется по тому же каналу, по которому оно пришло, сервер укажет, что данный клиент является автором этого сообщения. Тогда сообщение будет начинаться с “You : ”. Затем сервер начинает просмотр следующего канала и ждёт получения сообщения от него и так до бесконечности.

# **Листинг**

## **Серверная часть**

Состоит из одного файла на языке C++.

**server.cpp**

#include <windows.h>

#include <stdio.h>

#include <string.h>

#define CLIENT\_NUMBER 3

int main(void)

{

HANDLE\* hPipe = new HANDLE[CLIENT\_NUMBER];

LPSTR\* lpPipeName = new LPSTR[CLIENT\_NUMBER];

char server\_output[255 + 10] = " ";

char client1\_input[255] = " ";

char client2\_input[255] = " ";

char client3\_input[255] = " ";

DWORD iBytesToRead = 255;

lpPipeName[0] = TEXT("\\\\.\\pipe\\MyPipe0");

lpPipeName[1] = TEXT("\\\\.\\pipe\\MyPipe1");

lpPipeName[2] = TEXT("\\\\.\\pipe\\MyPipe2");

for(int i = 0; i < CLIENT\_NUMBER; i++){

hPipe[i] = CreateNamedPipe(lpPipeName[i],

PIPE\_ACCESS\_DUPLEX,

PIPE\_TYPE\_MESSAGE |

PIPE\_READMODE\_MESSAGE |

PIPE\_WAIT,

PIPE\_UNLIMITED\_INSTANCES,

4096,

4096,

NMPWAIT\_USE\_DEFAULT\_WAIT,

NULL);

if(hPipe[i] == INVALID\_HANDLE\_VALUE){

printf("Pipe hasn't been created\n");

return GetLastError();

} else

printf("Pipe created\n");

if((ConnectNamedPipe(hPipe[i], NULL)) == 0) {

printf("Client %d could not connect\n", i + 1);

return 0;

} else

printf("Client %d connected\n", i + 1);

while(true)

{

memset(client1\_input, 0, 255);

ReadFile(hPipe[0], client1\_input, sizeof(client1\_input), NULL, NULL);

printf("C1 mess read\n");

for (int i = 0; i < CLIENT\_NUMBER; i++){

memset(server\_output, 0, 265);

if (i == 0)

strcpy(server\_output, "You: ");

else

strcpy(server\_output, "Client 1: ");

strcat(server\_output, client1\_input);

WriteFile(hPipe[i], server\_output, strlen(server\_output), NULL, NULL);

}

printf("C1 mess sent\n");

memset(client1\_input, 0, 255);

ReadFile(hPipe[1], client2\_input, sizeof(client2\_input), NULL, NULL);

printf("C2 mess read\n");

for (int i = 0; i < CLIENT\_NUMBER; i++){

memset(server\_output, 0, 265);

if (i == 1)

strcpy(server\_output, "You: ");

else

strcpy(server\_output, "Client 2: ");

strcat(server\_output, client2\_input);

WriteFile(hPipe[i], server\_output, strlen(server\_output), NULL, NULL);

}

printf("C2 mess sent\n");

memset(client3\_input, 0, 255);

ReadFile(hPipe[2], client3\_input, sizeof(client3\_input), NULL, NULL);

printf("C3 mess read\n");

for (int i = 0; i < CLIENT\_NUMBER; i++){

memset(server\_output, 0, 265);

if (i == 2)

strcpy(server\_output, "You: ");

else

strcpy(server\_output, "Client 3: ");

strcat(server\_output, client3\_input);

WriteFile(hPipe[i], server\_output, strlen(server\_output), NULL, NULL);

}

printf("C3 mess sent\n");

}

for (int i = 0; i < CLIENT\_NUMBER; i++)

CloseHandle(hPipe[i]);

return 0;

}

## **Клиентская часть**

Состоит из файла с исходным кодом на языке C++, ресурсного файла для интерфейса WinAPI и заголовочного файла. Файлы разных клиентов идентичны и отличаются только именем канала. Для листинга будут использованы файлы первого клиента.

**main.cpp**

**#include <windows.h>**

**#include "client1\_lib.h"**

**#include <stdio.h>**

**#include <string.h>**

**HANDLE hPipe;**

**LPSTR lpPipeName = TEXT("\\\\.\\pipe\\MyPipe0");**

**char input\_buff[255 + 10];**

**char output\_buff[255] = " ";**

**char message1[255 + 10];**

**char message2[255 + 10];**

**char message3[255 + 10];**

**char message4[255 + 10];**

**char message5[255 + 10];**

**char message6[255 + 10];**

**DWORD iBytesToWrite = 255;**

**DWORD iBytesToRead;**

**LRESULT CALLBACK**

**DlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);**

**int WINAPI WinMain(HINSTANCE hInstance,**

**HINSTANCE hPrevInstance,**

**LPSTR lpCmdLine,**

**int nCmdShow) {**

**MSG msg;**

**DialogBox(hInstance,(LPCTSTR)IDD\_HOSTTEST,NULL,**

**(DLGPROC)DlgProc);**

**while (GetMessage(&msg, NULL, 0, 0)) {**

**TranslateMessage(&msg);**

**DispatchMessage(&msg);**

**}**

**return msg.wParam;**

**}**

**LRESULT CALLBACK**

**DlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) {**

**switch (message) {**

**case WM\_INITDIALOG:**

**hPipe = CreateFile(lpPipeName,**

**GENERIC\_READ |**

**GENERIC\_WRITE,**

**0,**

**NULL,**

**OPEN\_EXISTING,**

**0,**

**NULL);**

**if(hPipe == INVALID\_HANDLE\_VALUE){**

**printf("Client couldn't connected\n");**

**return GetLastError();}**

**else**

**printf("Client connected\n");**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT1, " ");**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT2, " ");**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT3, " ");**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT4, " ");**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT5, " ");**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT6, " ");**

**return TRUE;**

**case WM\_COMMAND:**

**switch (LOWORD(wParam) ) {**

**case IDCANCEL:**

**PostQuitMessage(0);**

**return TRUE;**

**case IDC\_BTN1:**

**memset(output\_buff, 0, 255);**

**GetDlgItemText(hDlg,IDC\_EDIT\_INPUT,output\_buff, 255);**

**WriteFile(hPipe, output\_buff, strlen(output\_buff), NULL, NULL);**

**SetDlgItemText(hDlg,IDC\_EDIT\_INPUT, " ");**

**break;**

**case IDC\_BTN2: //Update**

**memset(input\_buff, 0, 265);**

**memset(message1, 0, 265);**

**memset(message2, 0, 265);**

**memset(message3, 0, 265);**

**memset(message4, 0, 265);**

**memset(message5, 0, 265);**

**memset(message6, 0, 265);**

**ReadFile(hPipe, input\_buff, sizeof(input\_buff), NULL, NULL);**

**for (int i = 0; i < 255; i++)**

**printf("%c",input\_buff[i]);**

**GetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT6,message6, 265);**

**if (strcmp(message6, " ") != 0){**

**GetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT5,message5, 265);**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT5,message6);**

**if (strcmp(message5, " ") != 0){**

**GetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT4,message4, 265);**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT4,message5);**

**if (strcmp(message4, " ") != 0){**

**GetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT3,message3, 265);**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT3,message4);**

**if (strcmp(message3, " ") != 0){**

**GetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT2,message2, 265);**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT2,message3);**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT1,message2);**

**} else**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT2,message3);**

**} else**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT3,message4);**

**} else**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT4,message5);**

**} else**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT5,message6);**

**SetDlgItemText(hDlg,IDC\_EDIT\_OUTPUT6,input\_buff);**

**break;**

**}**

**break;**

**default:**

**return FALSE;**

**}**

**}**

**client1\_resource.rc**

**#include <windows.h>**

**#include "client1\_lib.h"**

**IDD\_HOSTTEST DIALOG DISCARDABLE 0, 0, 300, 400**

**STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CAPTION |**

**WS\_SYSMENU**

**CAPTION "Client1"**

**FONT 8, "MS Sans Serif"**

**BEGIN**

**PUSHBUTTON "Cancel",IDCANCEL,55,380,50,15**

**PUSHBUTTON "Send",IDC\_BTN1,7,380,43,15**

**PUSHBUTTON "Update",IDC\_BTN2,215,7,50,15**

**EDITTEXT IDC\_EDIT\_INPUT,7,357,200,18,ES\_AUTOHSCROLL**

**EDITTEXT IDC\_EDIT\_OUTPUT1,7,7,200,50,ES\_AUTOHSCROLL**

**EDITTEXT IDC\_EDIT\_OUTPUT2,7,62,200,50,ES\_AUTOHSCROLL**

**EDITTEXT IDC\_EDIT\_OUTPUT3,7,117,200,50,ES\_AUTOHSCROLL**

**EDITTEXT IDC\_EDIT\_OUTPUT4,7,172,200,50,ES\_AUTOHSCROLL**

**EDITTEXT IDC\_EDIT\_OUTPUT5,7,227,200,50,ES\_AUTOHSCROLL**

**EDITTEXT IDC\_EDIT\_OUTPUT6,7,282,200,50,ES\_AUTOHSCROLL**

**END**

**client1\_lib.h**

**#define IDD\_HOSTTEST 101**

**#define IDC\_EDIT\_INPUT 1000**

**#define IDC\_EDIT\_OUTPUT1 1001**

**#define IDC\_EDIT\_OUTPUT2 1002**

**#define IDC\_EDIT\_OUTPUT3 1003**

**#define IDC\_EDIT\_OUTPUT4 1004**

**#define IDC\_EDIT\_OUTPUT5 1005**

**#define IDC\_EDIT\_OUTPUT6 1006**

**#define IDC\_BTN1 1007**

**#define IDC\_BTN2 1008**

**#define CLIENT\_NUMBER 3**

# **Ссылки**

1. Репозиторий с учебными материалами, использованными при подготовке к реализации курсового проекта: [mlkv52git/sibsutis\_os: курс Операционные системы (github.com)](https://github.com/mlkv52git/sibsutis_os)
2. Репозиторий курсового проекта, где можно отследить ход его разработки и текущее состояние проекта: [Study/Operation Systems/CourseWork/Chat at master · Krasnoffsky/Study (github.com)](https://github.com/Krasnoffsky/Study/tree/master/Operation%20Systems/CourseWork/Chat)