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

Якунин А.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** 1](#_Toc59463446)4

# **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 2017 –** данная утилита будет использована для компиляции и сборки серверной части проекта. Она предоставляет удобный оптимизирующий компилятор.

**QT Creator 5.15.0** – данная IDE будет использована для компиляции, создания интерфейса 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 QT Creatorпредоставляет возможность создавать проекты с использованием нескольких файлов с исходным кодом, которые будут компилироваться и собираться совместно и для которых будет написан Makefile. QT Creatorпоставляется с предустановленным компилятором MinGW 7.3.0 для 64-разрядных операционных систем.

**windows.h** – В курсовом проекте данный набор будет использован для создания именованных каналов

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

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

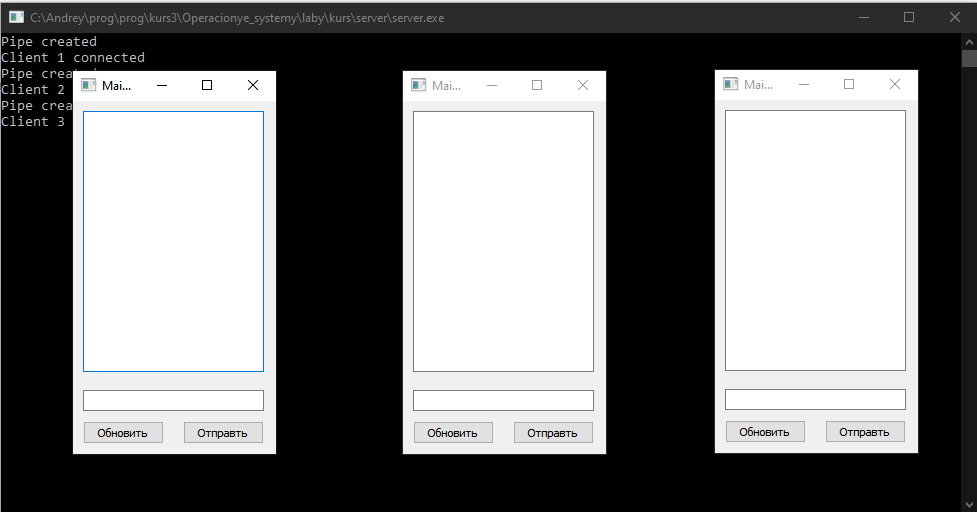


Рис. 2 Клиенты после запуска

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 «Отправить».

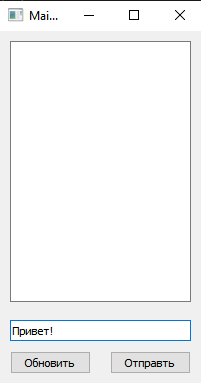


Рис. 4 Отправка сообщения

Для получения сообщения необходимо нажать кнопку «Обновить»

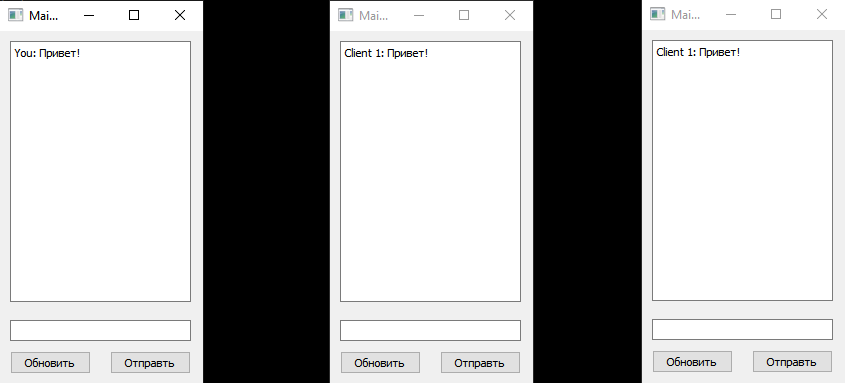


Рис. 5 Получение сообщения клиентом – отправителем.

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

Аналогично остальные клиенты могут писать сообщения в общий чат.

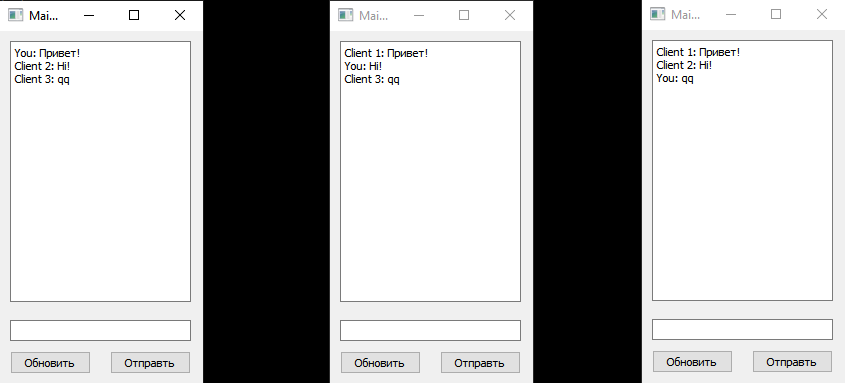


Рис. 7 Переписка между клиентами

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

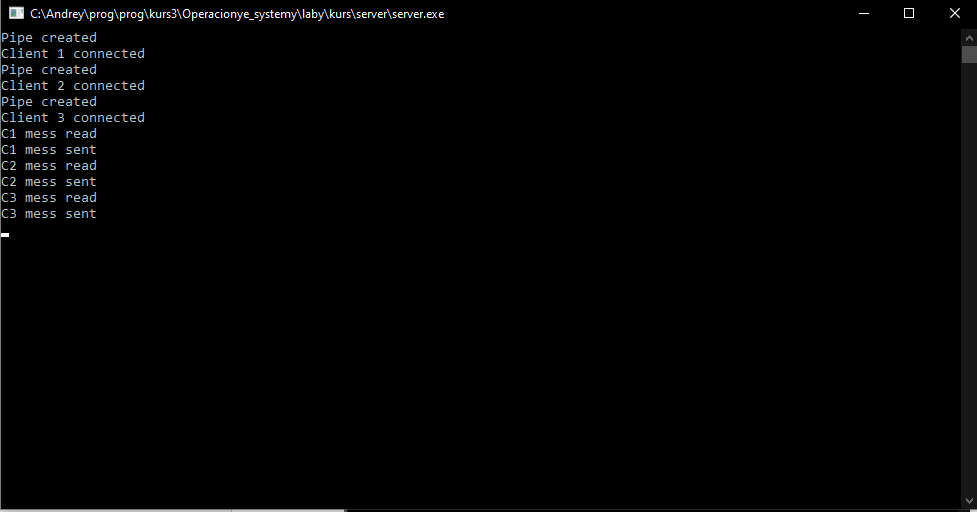


Рис. 8 Вывод сервера во время общения клиентов

# **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 2 текстовых полей и 2 кнопок. Нижнее текстовое поле служит для ввода сообщения, которое будет отправлено. Верхние поле работает как история сообщений. Для получения сообщения необходимо нажать кнопку «Обновить». При её нажатии программа заходит в канал, через который осуществляется общение с сервером. При наличии там сообщения она выводит его. Получение сообщения невозможно, если через канал не было передано сообщение. Для отправки сообщения необходимо нажать кнопку «Отправить». 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 может отправить сообщение вне очереди. Длинна сообщения ограничивается 255 символами.

## **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++){

//Creating pipes

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");

//Connecting clients

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)

{

//catch message from client1

memset(client1\_input, 0, 255);

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

printf("C1 mess read\n");

//send message from client1 to all clients

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");

//catch message from client2

memset(client1\_input, 0, 255);

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

printf("C2 mess read\n");

//send message from client2 to all clients

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");

//catch message from client3

memset(client3\_input, 0, 255);

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

printf("C3 mess read\n");

//send message from client3 to all clients

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**

Файлы разных клиентов идентичны и отличаются только именем канала. Для листинга будут использованы файлы первого клиента.

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <windows.h>

#include <stdio.h>

#include <string.h>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

HANDLE hPipe;

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

char input\_buff[255 + 10];

char output\_buff[255] = " ";

private slots:

void on\_pushButton\_send\_clicked();

void on\_pushButton\_update\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

hPipe = CreateFileA(lpPipeName,

GENERIC\_READ |

GENERIC\_WRITE,

0,

NULL,

OPEN\_EXISTING,

0,

NULL);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_send\_clicked()

{

memset(output\_buff, 0, 255);

strcpy(output\_buff, ui->lineEdit\_send->text().toStdString().c\_str());

ui->lineEdit\_send->clear();

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

}

void MainWindow::on\_pushButton\_update\_clicked()

{

memset(input\_buff, 0, 265);

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

ui->textEdit\_main->append(input\_buff);

}