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»**

по направлению 09.03.01 «Информатика и вычислительная техника»,

направленность (профиль) – «Программное обеспечение средств вычислительной техники и автоматизированных систем», квалификация – бакалавр,

программа академического бакалавриата,

форма обучения – очная, год начала подготовки (по учебному плану) – 2016

Выполнил: студент 3 курса гр. ИП-814 / Краснов И.В. /

Проверил: преподаватель кафедры ПМиК / Нужнов. А. В /

.

**Оглавление**

[Введение 3](#_Toc60066471)

[1. Постановка задачи 4](#_Toc60066472)

[2. Анализ предметной области 5](#_Toc60066473)

[2.1 Цель создания 5](#_Toc60066474)

[2.2 Перечень решаемых задач 5](#_Toc60066475)

[2.3 Специальное и общесистемное программное обеспечение 5](#_Toc60066476)

[3. Программное проектирование 7](#_Toc60066477)

[3.1 Функциональное обеспечение 7](#_Toc60066478)

[3.2 Алгоритмическое обеспечение 7](#_Toc60066479)

[3.3 Архитектурное обеспечение 7](#_Toc60066480)

[3.4 Информационное обеспечение 7](#_Toc60066481)

[4. Программная рецализация 8](#_Toc60066482)

[4.1 Комментарии (server.cpp) 8](#_Toc60066483)

[4.2 Комментарии (client1.cpp) 8](#_Toc60066484)

[4.3 Серверная часть 8](#_Toc60066485)

[4.4 Клиентская часть 11](#_Toc60066486)

[5. Сопроводительная документация программного обеспечения 17](#_Toc60066487)

[5.1 Разработка описания приложения 17](#_Toc60066488)

[5.2 Разработка руководства пользователя 17](#_Toc60066489)

[5.3 Анализ программного обеспечения 17](#_Toc60066490)

[6. Заключение 19](#_Toc60066491)

[7. Ссылки 20](#_Toc60066492)

# **Введение**

При написании многопроцессных программ возникает необходимость обмена данными между этими процессами. Одним из вариантов такого обмане выступают каналы: именованные и анонимные.

**Именованные каналы** обеспечивают межпроцессное взаимодействие между сервером канала и одним или несколькими клиентами канала. Они предоставляют больше функциональных возможностей, чем анонимные каналы, которые обеспечивают межпроцессное взаимодействие на локальном компьютере. Именованные каналы поддерживают полную дуплексную связь по сети и несколько экземпляров сервера, связь на основе сообщений и олицетворение клиента, позволяющее подключающимся процессам использовать собственные наборы разрешений на удаленных серверах.

# **Постановка задачи**

Я выбрал в качестве курсовой работы – создание сетевого чата на основе сокетов и именованных каналов, включающая в себя следующие пункты, которые мне следовало реализовать в приложении:

Разработка сетевого приложения, реализующего:

а) сетевую игру/чат на основе именованных каналов.

# **Анализ предметной области**

## **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

## **Специальное и общесистемное 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 будет реализован на операционной системе Windows 10. Интерфейс содержат только клиентские приложения. Серверная часть не имеет интерфейса и работает в невидимом для пользователя режиме.

# **Программное проектирование**

## **Функциональное обеспечение**

Прежде, чем начинать разработку необходимо заранее определить требуемый функционал нашего программного обеспечения.

В данном проекте в качестве актеров выступают пользователь (клиентское приложение) и сервер. Пользователь взаимодействует непосредственно с серверным приложением и должен иметь возможность выполнить следующие действия:

* Присоединиться к серверу;
* Отослать сообщение клиенту;
* Принимать сообщения от клиента.

## **Алгоритмическое обеспечение**

Алгоритм обработки команд сервера на клиенте.

Клиент может получать текстовую информацию: как только клиент получает текстовое сообщение, оно отображается в поле вывода информации.

## **Архитектурное обеспечение**

Для взаимодействия клиента с сервером используются именнованные каналы Windows. Необходимым условием корректной работы сервера и клиента является наличие на машине библиотеки windows.h.

## **Информационное обеспечение**

Сервер с клиентом обменивается одним типом данных. Если серверу пришло сообщение строкового типа, то сервер обрабатывает сообщение и рассылает их всем остальным клиентам, в том числе и отправителю.

# **Программная рецализация**

## **Комментарии (server.cpp)**

* Создание именованного канала:

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

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)

* Подключаем клиентов:

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

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

return 0;}

* Запускаем бесконечный цикл, на каждой итерации которого происходит обработка сообщений от клиентов и их рассылка.

## **Комментарии (client1.cpp)**

* LRESULT CALLBACK DlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) – функция-обработчик возможных событий в приложении(создание окна - case WM\_CREATE, обработка нажатий клавиш - case WM\_COMMAND, подключение клиента до демонстрации интерфейса пользователя – case WM\_INIT)

## **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

# **Сопроводительная документация программного обеспечения**

## **Разработка описания приложения**

Общее описание:

«Данный программный продукт предназначен для обмена информацией по локальной или беспроводной сети. Сервер запускается как сетевая служба и работает на компьютере в фоновом режиме. Клиентское приложение не требует никаких настроек».

## **Разработка руководства пользователя**

1. Для компиляции (если приложения не скомпилированы) необходимо открыть среду разработки Dev-C++ и скомпилировать проект с клиентом.

2. Собрать и запустить программу-сервер с помощью консоли, набрав команду cl server.cpp.

Для запуска серверного и клиентского приложения необходимым и достаточным условием являются следующие требования:

- Операционная система: Windows 7 или выше;

- Объём оперативной памяти: 1,00 ГБ;

- Свободное место на диске: 1,00 МБ;

- Наличие сетевой карты.

## **Анализ программного обеспечения**

Анализ исходного кода проводился во время компиляции с помощью оптимизирующего компилятора, встроенного в утилиту Developer Command Prompt for VS 2019.

Результат анализа исходного кода:

* Предупреждение о неиспользуемых переменных – исправлено;
* Критических ошибок в коде не обнаружено.

# **Заключение**

На основе полученных знаний по курсу «Операционные системы» мне в результате удалось выполнить курсовую работу по теме «Многопользовательский сетевой чат на основе именованных каналов», было разработано приложение-клиент, которое имело графический интерфейс, написанный с помощью Windows API.

Данное приложение позволяет удалённо обмениваться информацией и данными по локальной или беспроводной сети.

Считаю, что мне удалось выполнить цель, ставившуюся перед началом создания программного продукта.

# **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)