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-715 / Эпов И.В /

Проверил: профессор кафедры ПМиК / Малков Е.А. /

Новосибирск, 2019

Содержание

[Введение 3](#_Toc533293352)

[1. Постановка задачи 4](#_Toc533293353)

[2. Анализ предметной области 5](#_Toc533293354)

[2.1. Цель создания 5](#_Toc533293355)

[2.2. Перечень решаемых задач 5](#_Toc533293356)

[2.3. Специальное и общесистемное программное обеспечение 5](#_Toc533293357)

[3. Программное проектирование 6](#_Toc533293358)

[3.1. Пользовательский интерфейс 6](#_Toc533293359)

[3.2. Функциональное обеспечение 7](#_Toc533293360)

[3.3. Алгоритмическое обеспечение 7](#_Toc533293361)

[3.4. Архитектурное обеспечение 8](#_Toc533293362)

[3.5. Информационное обеспечение 8](#_Toc533293363)

[4. Программная реализация 9](#_Toc533293364)

[5. Сопроводительная документация программного обеспечения 16](#_Toc533293367)

[5.1. Разработка описания программы 16](#_Toc533293368)

[5.2. Разработка руководства пользователя 16](#_Toc533293369)

[5.3. Анализ программного обеспечения 16](#_Toc533293370)

[6. Заключение 17](#_Toc533293372)

[7. Список использованных источников 18](#_Toc533293373)

# Введение

Работая в сети Ethernet, мы очень часто встречаемся с разного рода многопользовательскими программами. Ими могут быть почтовые клиенты, чаты, форумы, FTP клиенты и т.п. Все эти приложения используют для своей работы разного рода протоколы: FTP, POP, SMTP, HTTP, и т.д. Но базовым для них является единый протокол - TCP/IP. Типичное же приложение TCP/IP построено на клиент-серверной архитектуре. Примером приложения построенного на данной архитектуре, является чат реального времени. При работе с протоколом TCP/IP используют интерфейс сокетов. Но прежде чем перейти к разработке чата, разберем, что же такое сокеты.

**Сокеты** – это интерфейс прикладного программирования для сетевых приложений TCP/IP. Интерфейс сокетов был создан в восьмидесятых годах для операционной системы UNIX. Позднее интерфейс сокетов был перенесен в Microsoft Windows. Сокеты до сих пор используются в приложениях для сетей TCP/IP. В переводе с английского "sockets" – гнезда, т.е. сетевые приложения используют сокеты, как виртуальные разъемы для обмена данными между собой. Сокеты бывают трех видов: клиентские, слушающие и серверные. Клиентские сокеты устанавливают связь с сервером и обмениваются с ним данными. Серверный сокет обменивается данными с клиентом по уже установленному (слушающим сокетом) соединению. Для того чтобы клиент мог установить соединение с сервером, ему необходимо указать его адрес (IP) и номер порта, через который будет происходить обмен данными.

В данной курсовой работе предполагается реализовать программное обеспечение c графическим интерфейсом для обмена сообщениями посредством локальной или беспроводной сети.

Основное предназначение реализуемого программного продукта - передача символьной информации, а так же файлов. Язык, выбранный для реализации курсового проекта – С++ - мощный язык программирования, подходящий для решения любых задач.

Данный программный продукт может использоваться пользователями в офисе, а так же дома.

# 1. Постановка задачи

Наша бригада выбрала в качестве курсовой работы – создание сетевой службы, реализующей функциональность сетевого чата на основе сокетов, включающая в себя следующие пункты, которые нам следовало реализовать в нашем приложении:

* Реализовать программу-сервер на основе сетевой службы;
* Реализовать программу-клиент;
* Разработать протокол взаимодействия клиентской и серверной частей приложения;
* Разработать качественный графический интерфейс для программы-клиент.

# 2. Анализ предметной области

Предметная область проекта - передача информации по сети. Чат, позволяющий организовать общение между неограниченным количеством пользователей.

Требуется разработать приложение, которое включает в себя разработку следующих функциональных частей: разработка серверного приложения и разработка клиентского приложения.

## 2.1. Цель создания

Главная цель курсовой работы по теме «Многопользовательский сетевой чат» – создание многопользовательского сетевого чата с возможностью обмена текстовыми сообщениями между пользователями, подключенными к одному серверу.

## 2.2. Перечень решаемых задач

Функционал данного сетевого чата предусматривает выполнение следующих задач:

* Подключение двух и более пользователей к серверу для возможности общения друг с другом;
* Обмен сообщений между клиентами;
* Работа в чате через удалённый сервер, работающий на основе сетевой службы;
* Выхода клиентов из чата и повторный вход.

## 2.3. Специальное и общесистемное программное обеспечение

При написании данного курсового проекта планируется использовать следующее стороннее программное обеспечение и инструментарий:

**Microsoft Visual Studio 2019** — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.  
 **Windows API** - общее наименование набора базовых функций интерфейсов программирования приложений операционных систем семейств Microsoft Windows корпорации «Майкрософт». Предоставляет прямой способ взаимодействия приложений пользователя с операционной системой Windows. Для создания программ, использующих Windows API, корпорация «Майкрософт» выпускает комплект разработчика программного обеспечения, который называется *Platform SDK*, и содержит документацию, набор библиотек, утилит и других инструментальных средств для разработки.

Разработка будет осуществляться в операционной системе Windows 10. Серверное приложение будет реализовано с отсутствием интерфейса, так как будет работать в скрытом (невидимом для пользователя) режиме, что позволит сфокусироваться на его функциональности.

# 3. Программное проектирование

## 3.1. Пользовательский интерфейс

При загрузке приложений, в первую очередь пользователь видит интерфейс продукта, поэтому особое внимание требуется уделить на разработку интерфейса. Основные критерии создания пользовательского интерфейса:

* Программа должна содержать в себе следующие элементы: строка ввода, поле отображения сообщений, кнопки управления.

Таким образом, программа становится простой даже для неопытного пользователя.

Основное пространство отображения информации на форме будет распределено следующим образом:

* Снизу располагается строка ввода;
* Справа от области ввода строк сообщения располагаются кнопки «Очистить» , «Отправить» и «Х»;
* Основная область окна приходится на вывод диалог между клиентами.

После составления вышеописанных требований был составлен прототип интерфейса главного окна (Рисунок 1).

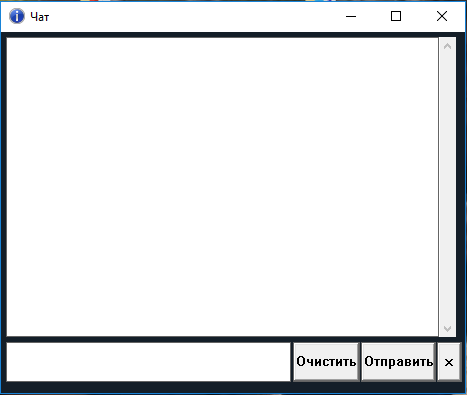


Рисунок 1. Прототип интерфейса главного окна.

При запуске приложения выводится окно с данными о статусе подключения и инициализации:

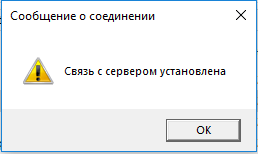


Рисунок 2. Сообщение о статусе подключения.

Серверное приложение работает в скрытом от пользователя режиме и не требует никаких действий. Поэтому пользовательский интерфейс у сервера отсутствует.

## 3.2. 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.

## 3.3. 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.

## 

## 3.4. Архитектурное обеспечение

Для взаимодействия клиента с сервером используется протокол передачи данных TCP. Необходимым условием корректной работы сервера и клиента является наличие на машине библиотеки WinSock2.

**Windows Sockets API (WSA)** - это техническая спецификация, которая определяет, как сетевое программное обеспечение Windows будет получать доступ к сетевым сервисам, в том числе, TCP/IP. Он определяет стандартный интерфейс между клиентским приложением и внешним стеком протоколов TCP/IP. Он основывается на API модели сокетов Беркли, использующейся в BSD для установки соединения между программами.

## 3.5. Информационное обеспечение

Перед реализацией программы было решено обмениваться информацией по протоколу передачи данных TCP. Такой выбор был сделан с учетом специфики задачи. Протокол TCP гарантирует доставку пакетов данных в неизменном виде, последовательности и без потерь.

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.

# 4. Программная реализация

Так как работа над данным продуктом осуществлялась бригадой, то наши обязанности были равномерно распределены.

Моя ответственность легла на разработку программы-сервера (это включает создание сетевой службы, которая содержала бы в себе весь функционал сервера для нашего сетевого чата).

**Описание программы-клиента**

Средства, облегчающие компиляцию программы (дополнительные библиотеки):

#pragma comment (lib,"Ws2\_32.lib")

#pragma comment (lib,"user32.lib")

#pragma comment (lib,"gdi32.lib")

Основные встроенные библиотеки:

*#include <WinSock2.h>*

*#include <stdio.h>*

*#include <conio.h>*

*#include <iostream>*

*#include <string>*

Идентификаторы для номера порта и имени сервера:

*#define PORT 1334*

*#define SERVICENAME "Server\_of\_Chat"*

Объявление глобальных переменных (для простоты реализации) и функций:

*using namespace std;*

*SERVICE\_STATUS wserv\_testStatus;*

*SERVICE\_STATUS\_HANDLE wserv\_testStatusHandle;*

*SOCKET servSock, usersList[2];*

*int usersCount = 0;*

Затем происходит непосредственно работа с сокетами. Для начала необходимо инициализировать библиотеку Winsock:

*if (WSAStartup(0x202, (WSADATA \*)&buff[0]))*

*{*

*printf("Unable to create winsock\n");*

*return 1;*

*}*

Функция socket() создаёт новый сокет. Требуется 3 аргумента, AF\_INET: адресный домен сокета, SOCK\_STREAM: тип сокета, сокет потока в какие символы читаются в непрерывном потоке (TCP), третий аргумент протокола: наиболее подходящий протокол TCP. Если вызов сокета не удаётся, возвращается *SOCKET\_ERROR*.

*servSock = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);*

*if (servSock == INVALID\_SOCKET)*

*{*

*printf("Unable to create socket\n");*

*WSACleanup();*

*system("pause");*

*return SOCKET\_ERROR;*

*}*

sin\_family содержит код для семейства адресов. Он всегда должен быть установлен в AF\_INET. htons () преобразует номер порта из порядка байтов хоста на номер порта в сетевом порядке байтов.

*sin.sin\_family = AF\_INET;*

*sin.sin\_port = htons(PORT);*

*sin.sin\_addr.s\_addr = 0;*

Системный вызов bind () привязывает сокет к адресу, в этом случае адрес текущего хоста и номер порта на котором будет работать сервер.

*retVal = bind(servSock, (LPSOCKADDR)&sin, sizeof(sin));*

*if (retVal == SOCKET\_ERROR)*

*{*

*printf("Unable to bind\n");*

*return SOCKET\_ERROR;*

*}*

*printf("Server started \n");*

Системный вызов listen() позволяет процессу прослушивать сокет для подключения. Программа будет бездействовать здесь, если нет входящего соединения. Первый аргумент - дескриптор файла сокета, а второй размер для количества клиентов, то есть количество соединений, которые может сервер обрабатывать в то время, как процесс обрабатывает определенное подключение.

*retVal = listen(servSock, 10);*

*if (retVal == SOCKET\_ERROR)*

*{*

*printf("Unable to listen\n");*

*WSACleanup();*

*return SOCKET\_ERROR;*

*}*

Системный вызов accept() блокирует процесс до тех пор, пока клиент не подключится к серверу. Таким образом, он пробуждает процесс, когда соединение с клиентом было успешно установлено. Он возвращает новый файловый дескриптор, и вся связь по этому соединению должна осуществляться с использованием нового файлового дескриптора. Второй аргумент - это ссылочный указатель на адрес клиента на другом конце соединения, а третий аргумент - это размер этой структуры.

*int fromlen = sizeof(from);*

*clientSock = accept(servSock, (struct sockaddr\*)&from, &fromlen);*

*if (clientSock == INVALID\_SOCKET)*

*{*

*printf("Unable to accept\n");*

*WSACleanup();*

*return SOCKET\_ERROR;*

*}*

*printf("New connection accepted from %s, port %d\n", inet\_ntoa(from.sin\_addr), htons(from.sin\_port));*

*closesocket(servSock) –* закрываем сокет.

Функция *CtrlHandler* - обработка управления службы:

*VOID \_\_stdcall CtrlHandler (DWORD Opcode)*

*{*

*DWORD status;*

*switch (Opcode)*

*{*

*case SERVICE\_CONTROL\_PAUSE:*

*wserv\_testStatus.dwCurrentState = SERVICE\_PAUSED;*

*break;*

*case SERVICE\_CONTROL\_CONTINUE:*

*wserv\_testStatus.dwCurrentState = SERVICE\_RUNNING;*

*break;*

*case SERVICE\_CONTROL\_STOP:*

*wserv\_testStatus.dwWin32ExitCode = 0;*

*wserv\_testStatus.dwCurrentState = SERVICE\_STOPPED;*

*wserv\_testStatus.dwCheckPoint = 0;*

*wserv\_testStatus.dwWaitHint = 0;*

*if (!SetServiceStatus (wserv\_testStatusHandle, &wserv\_testStatus))*

*status = GetLastError();*

*return;*

*default:*

*break;*

*}*

*if (!SetServiceStatus (wserv\_testStatusHandle, &wserv\_testStatus))*

*status = GetLastError();*

*return;*

*}*

Точка входа в службу: инициализация полей структуры SERVICE\_STATUS, регистрация обработчика управления:

*void \_\_stdcall wserv\_testStart (DWORD argc, LPTSTR \*argv)*

*{*

*DWORD status;*

*wserv\_testStatus.dwServiceType = SERVICE\_WIN32;*

*wserv\_testStatus.dwCurrentState = SERVICE\_START\_PENDING;*

*wserv\_testStatus.dwControlsAccepted = SERVICE\_ACCEPT\_STOP | SERVICE\_ACCEPT\_PAUSE\_CONTINUE;*

*wserv\_testStatus.dwWin32ExitCode = 0;*

*wserv\_testStatus.dwServiceSpecificExitCode = 0;*

*wserv\_testStatus.dwCheckPoint = 0;*

*wserv\_testStatus.dwWaitHint = 0;*

*wserv\_testStatusHandle = RegisterServiceCtrlHandler(TEXT(SERVICENAME), CtrlHandler);*

*if (wserv\_testStatusHandle == (SERVICE\_STATUS\_HANDLE)0)*

*return;*

Точка входа в службу: обновление полей структуры SERVICE\_STATUS, посылка сообщения SCM, выполнение функции *serv()* службы:

*wserv\_testStatus.dwCurrentState = SERVICE\_RUNNING;*

*wserv\_testStatus.dwCheckPoint = 0;*

*wserv\_testStatus.dwWaitHint = 0;*

*if (!SetServiceStatus (wserv\_testStatusHandle, &wserv\_testStatus))*

*status = GetLastError();*

*FILE\* fp;*

*SYSTEMTIME stSystemTime;*

*while (wserv\_testStatus.dwCurrentState!=SERVICE\_STOPPED)*

*{*

*if (wserv\_testStatus.dwCurrentState!=SERVICE\_PAUSED)*

*{*

*serv();*

*}*

*Sleep(5000);*

*}*

*return;*

*}*

Задание точки входа в службу и запуск бесконечного цикла в потоке службы:

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

*{*

*SERVICE\_TABLE\_ENTRY DispatchTable[] = {{ TEXT(SERVICENAME), wserv\_testStart },{ NULL, NULL}};*

*if (!StartServiceCtrlDispatcher( DispatchTable))*

*{*

*}*

*}*

Сначала у нас происходит программное создание службы:

*if (argc > 1 && !stricmp(argv[1], "setup"))*

*{*

*char pname[1024];*

*pname[0] = '"';*

*GetModuleFileName(NULL, pname + 1, 1023);*

*strcat(pname, "\"");*

*SC\_HANDLE scm = OpenSCManager(NULL, NULL, SC\_MANAGER\_CREATE\_SERVICE),svc;*

*if (!scm)*

*{*

*cout << "Can't open SCM\n";*

*exit(1);*

*}*

*if (!(svc = CreateService(scm, SERVICENAME, SERVICENAME, SERVICE\_ALL\_ACCESS, SERVICE\_WIN32\_OWN\_PROCESS, SERVICE\_DEMAND\_START, SERVICE\_ERROR\_NORMAL, pname, NULL, NULL, NULL, NULL, NULL)))*

*{*

*cout << "Registration error!\n";*

*exit(2);*

*}*

*cout << "Successfully registered " << pname << "\n";*

*CloseServiceHandle(svc);*

*CloseServiceHandle(scm);*

*exit(0);*

*}*

Программное удаление службы:

*if (argc > 1 && !stricmp(argv[1], "delete"))*

*{*

*SC\_HANDLE scm = OpenSCManager(NULL, NULL, SC\_MANAGER\_CREATE\_SERVICE);*

*if (!scm)*

*{*

*cout << "Can't open SCM\n";*

*exit(1);*

*}*

*SC\_HANDLE svc = OpenService(scm, SERVICENAME, DELETE);*

*if (!svc)*

*{*

*cout << "Can't open service\n";*

*exit(2);*

*}*

*if (!DeleteService(svc))*

*{*

*cout << "Can't delete service\n";*

*exit(3);*

*}*

*cout << "Service deleted\n";*

*CloseServiceHandle(svc);*

*CloseServiceHandle(scm);*

*exit(0);*

*}*

Рисунок 3. Отображение сетевой службы

# 

# 5. Сопроводительная документация программного обеспечения

## 5.1. Разработка описания программы

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».

## 5.2. Разработка руководства пользователя

1. Для компиляции (если приложения не скомпилированы) необходимо открыть консоль разработчика VS 2015 от имени администраторы (очень важный аспект!!!), перейти в консоли в директорию с исходниками и там набрать следующие команды:

* cl /EHsc Server.cpp – для компиляции программы-сервера;
* cl /EHsc Client.cpp – для компиляции программы-клиента.

2. Для запуска сервера сначала наберите в этой же консоли команду – Server.exe setup – которая установит службу, реализующую данный сервер. Затем зайдите от имени администратора в «Службы» (компонент операционной системы) и запустите там службу Server\_of\_chat.

3. Для запуска программы-клиента просто откройте соответствующий exe-файл.

Запустите Server.exe и несколько Client.exe на соответствующих сторонах.

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.

## 5.3. Анализ программного обеспечения

Анализ исходного кода проводился с помощью встроенной утилиты в среду разработки Microsoft Visual Studio 2019.

Результат анализа исходного кода:

* Предупреждения о переменных, содержащих излишнюю инициализацию – исправлено;
* Предупреждение о неиспользуемых переменных – исправлено;
* Критических ошибок в коде не обнаружено.

# 6. Заключение

На основе полученных знаний по курсу «Операционные системы» нам в результате удалось выполнить курсовую работу по теме «Многопользовательский сетевой чат на основе сокетов». Была разработана сетевая служба, которая играла роль нашего сервера, а также 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.

Интерфейс на клиенте выполнен в наиболее удобном и простом варианте, что позволяет пользователю комфортно работать с приложениями.

В заключении хотелось бы отметить, что нам удалось выполнить цель, ставившуюся перед началом создания программного продукта.

# 7. Список использованных источников

1. Курс лекций Малкова Е.А. «Операционные системы».

2. Организации взаимодействия между Клиентом и Сервером с помощью протоколов TCP, UDP <http://www.intuit.ru/department/os/osintropractice/10/1.html>

3. Происхождение чата, http://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82