|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Лабораторная работа № 3 | | |
| по дисциплине «Операционные системы, среды и оболочки» | | |
| **РАЗРАБОТКА ПРИЛОЖЕНИЯ ИНТЕРАКТИВНОЙ ПЕРЕПИСКИ** | | |
|  | | |
|  | Бригада 4 | Лойченко данила |
| Группа ПМ-12 | ушатов сергей |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватели | кобылянский валерий григорьевич |
|  | Сивак Мария Алекссевна |
| Новосибирск, 2023 | | |

1. **Цель работы**

С помощью API-интерфейса реализовать простой чат**.** Каждая бригада должна написать chat-сервер и chat-клиента. Сервер должен поддерживать соединение сразу от нескольких клиентов. Обмен между клиентами осуществляется через сервер. При получении сообщения от какого-либо клиента, сервер дублирует его на своем экране и оповещает всех подсоединенных клиентов, отправляя каждому из них данное сообщение. При подсоединении нового клиента к chat-серверу, сервер оповещает каждого клиента о новом пользователе, посылая им его IP-адрес, номер порта и имя.

Тестирование сервера провести в режиме удаленного доступа при одновременном подключении не менее двух клиентов, загруженных на разных компьютерах. Также необходимо проверить корректность работы клиента, осуществляющего ввод данных, при одновременной отправке данных другим клиентом.

1. **Ход работы**

**Код сервера:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <locale.h>

#include <iostream>

#include <string>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

#include <vector>

#include <thread>

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

using namespace std;

int ServerThread(SOCKET Client, SOCKADDR\_IN ClientAddr, vector<SOCKET>& allClients)

{

allClients.push\_back(Client);

int ErrorStatus, MaxLength = 1024;

char UserName[20];

char\* RecvBuffer = new char[MaxLength]; // Буфер для приема данных

char\* SendBuffer = new char[MaxLength]; // Буфер для отправки данных

char ip[INET\_ADDRSTRLEN];

bool flag = true;

inet\_ntop(AF\_INET, &(ClientAddr.sin\_addr), ip, INET\_ADDRSTRLEN); // Запись IP

unsigned short port = ClientAddr.sin\_port; //Запись порта

recv(Client, UserName, 20, 0);

\_snprintf\_s(SendBuffer, MaxLength, MaxLength, "К чату подключился %s с ip: %s:%d\n", UserName, ip, port); //Вывод информации всем подключенным пользователям

printf\_s("%s", SendBuffer); //Вывод информации на сервер

for (int i = 0; i < allClients.size(); i++)

if (allClients[i] != Client)

send(allClients[i], SendBuffer, strlen(SendBuffer), 0);

memset(SendBuffer, 0, MaxLength);

while (flag)

{

memset(RecvBuffer, 0, MaxLength);

memset(SendBuffer, 0, MaxLength);

ErrorStatus = recv(Client, RecvBuffer, MaxLength, 0);

if (ErrorStatus > 0)

{

RecvBuffer[ErrorStatus] = 0;

if (!strcmp(RecvBuffer, "/exit")) //Проверка на попытку отключения

{

printf("Пользователь %s с ip %s отключился\n", UserName, ip);

\_snprintf\_s(SendBuffer, MaxLength, MaxLength, "Пользователь %s отключился\n", UserName);

for (int i = 0; i < allClients.size(); i++)

{

if (allClients.at(i) != Client)

send(allClients[i], SendBuffer, strlen(SendBuffer), 0);

else

allClients.erase(allClients.begin() + i);

}

break;

}

printf("Сообщение от %s: %s\n", UserName, RecvBuffer);

\_snprintf\_s(SendBuffer, MaxLength, MaxLength, "%s: %s\n", UserName, RecvBuffer);

for (int i = 0; i < allClients.size(); i++)

send(allClients[i], SendBuffer, strlen(SendBuffer), 0);

}

else

{

printf("Ошибка получения данных от %s. Номер ошибки: %d\n", ip, WSAGetLastError());

for (int i = 0; i < allClients.size(); i++)

if (allClients[i] == Client)

allClients.erase(allClients.begin() + i);

closesocket(Client);

delete[] SendBuffer;

delete[] RecvBuffer;

return 1;

}

}

}

int \_\_cdecl main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

WSADATA WSAData; //Создание структуры, в которую загружаются данные о версии сокетов, используе-мых ОС.

SOCKET ListenSocket, ClientSocket;

sockaddr\_in ServerAddr;

int ErrorStatus, MaxLength = 1024;

unsigned short port = 2004;

WSAStartup(MAKEWORD(2, 2), &WSAData); //Запуск сокетов с указанием версии реализации

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

// Установка адресной информации для сокета

ServerAddr.sin\_addr.s\_addr = INADDR\_ANY; //Регистрирование сервера на всех адресах машины, на которой он запущен

ServerAddr.sin\_family = AF\_INET; //Семейство

ServerAddr.sin\_port = htons(port); //Порт

ErrorStatus = bind(ListenSocket, (sockaddr\*)&ServerAddr, sizeof(ServerAddr));

if (ErrorStatus == SOCKET\_ERROR) {

printf("Ошибка bind. Номер ошибки: %d\n", WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

return 1;

}

ErrorStatus = listen(ListenSocket, 50); //Формирование очереди для входящего подключения и перевод сокета в режим прослушки

if (ErrorStatus == SOCKET\_ERROR) {

printf("Ошибка listen. Номер ошибки: %d\n", WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

return 1;

}

printf("Сервер запущен. Порт %d.\n", port);

vector<SOCKET> clients;

while (true) {

SOCKADDR\_IN ClientAddr;

int CliendAddrSize = sizeof(ClientAddr);

ClientSocket = accept(ListenSocket, (struct sockaddr\*)&ClientAddr, &CliendAddrSize);

std::thread t(ServerThread, ClientSocket, ClientAddr, std::ref(clients));

t.detach(); // Отсоединяет связанный поток от объекта thread

}

closesocket(ListenSocket);

WSACleanup();

return 0;

}

Код клиента:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <locale.h>

#include <iostream>

#include <thread>

#include <stdio.h>

#include <stdlib.h>

#include <winsock2.h>

#include <ws2tcpip.h>

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

using namespace std;

void ClientThread(SOCKET ConnectSocket, char\* recvbuffer, int size) {

int ErrorStatus = 0;

while (true)

{

ErrorStatus = recv(ConnectSocket, recvbuffer, size - 1, 0);

if (ErrorStatus > 0)

{

recvbuffer[ErrorStatus] = 0;

printf("%s", recvbuffer);

}

else

break;

}

}

int \_\_cdecl main(void)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

WSADATA WSAData;

SOCKET ConnectSocket;

sockaddr\_in ServerAddr;

int ErrorStatus = 0, MaxLength = 1024;

char\* RecvBuffer = new char[MaxLength]; // Буфер для приема данных

char\* SendBuffer = new char[MaxLength]; // Буфер для отправки данных

char ip[23];

char\* UserName = new char[20];

unsigned short port;

WSAStartup(MAKEWORD(2, 2), &WSAData); //Запуск сокетов с указанием версии реализации

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

printf("Для подключения к серверу введите IP: ");

cin >> ip;

printf("Для подключения к серверу введите порт: ");

cin >> port;

ServerAddr.sin\_family = AF\_INET; //Семейство

ServerAddr.sin\_port = htons(port); //Порт

if (inet\_pton(AF\_INET, ip, &ServerAddr.sin\_addr) <= 0)

{

printf\_s("Не удалось считать IP адресс сервера.");

return 1;

}

ErrorStatus = connect(ConnectSocket, (sockaddr\*)&ServerAddr, sizeof(ServerAddr));

if (ErrorStatus == SOCKET\_ERROR) {

printf("Ошибка подключения. Номер ошибки: %d\n", WSAGetLastError());

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

printf\_s("Пожалуйста, введите Ваше имя для чата, которое не должно превышать 20 символов и не должно содержать пробелы: ");

scanf("%s", UserName);

getchar();

send(ConnectSocket, UserName, 20, 0);

std::thread t(ClientThread, ConnectSocket, RecvBuffer, MaxLength);

t.detach();

for (bool flag = true; flag;) {

gets\_s(SendBuffer, MaxLength);

if (!strcmp(SendBuffer, "/exit"))

flag = false;

send(ConnectSocket, SendBuffer, strlen(SendBuffer), 0);

}

delete[] UserName;

delete[] SendBuffer;

delete[] RecvBuffer;

closesocket(ConnectSocket);

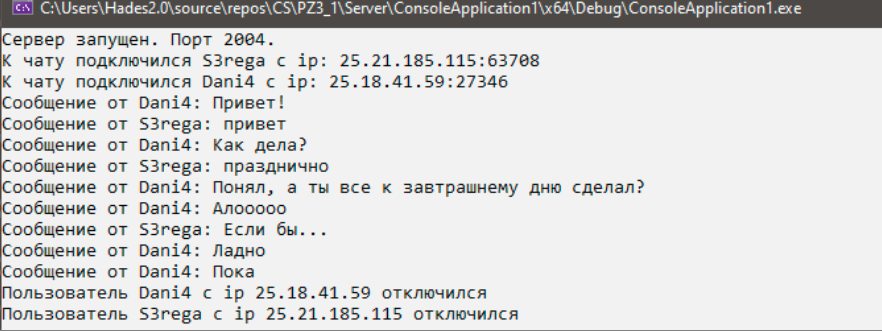
WSACleanup();

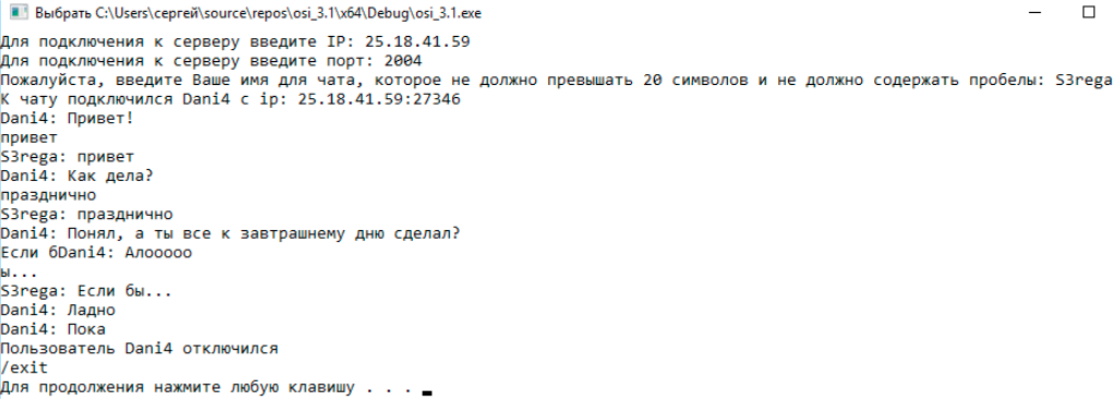
system("pause");

return 1;

}

Серверная часть:

  
Клиентская часть 1:



Клиентская часть 2:

