|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Лабораторная работа № 3 | | |
| по дисциплине «Операционные системы и компьютерные сети » | | |
| **Технология клиент-сервер: эхо-повтор** | | |
|  | | |
|  |  |  |
| Группа ПМ-21 | порсин данил |
| Место для ввода текста. |  |
|  |  |
|  |  |
|  |  |
| Преподаватели | сивак м.а. |
|  |  |
| Новосибирск, 2025 | | |

1. **Цель работы**

Изучить основные принципы разработки клиент-серверных приложений на примере простейшей однопользовательской программы.

1. **Задание**

Клиент пересылает серверу два числа. Сервер возвращает сумму полученных чисел.

1. **Анализ задачи**

#### **Инициализация сервера (функция**server()**)**

* Создание сокета:
  + Инициализируется Winsock (WSAStartup).
  + Создается TCP-сокет (AF\_INET, SOCK\_STREAM).
* Привязка сокета:
  + Указывается IP-адрес (192.168.0.100) и порт (2011).
  + Сокет привязывается к адресу (bind).
* Ожидание подключений:
  + Сервер переходит в режим прослушивания (listen).
  + При подключении клиента создается новый сокет (accept).

1. Работа сервера с клиентом

* Прием данных от клиента:
  + Сервер читает строку с числами (recv).
  + Если получена команда "s", сервер завершает работу.
* Вычисление суммы:
  + Числа парсятся, вычисляется их сумма (sum).
* Отправка результата:
  + Результат отправляется клиенту (send).
  + Соединение с клиентом закрывается (closesocket).

1. Инициализация клиента (функция make\_client())

* Создание сокета:
  + Инициализируется Winsock (WSAStartup).
  + Создается TCP-сокет (AF\_INET, SOCK\_STREAM).
* Настройка адреса сервера:
  + Указывается IP и порт сервера.

1. Подключение клиента к серверу (connect\_server())

* Клиент пытается подключиться (connect).
* Если подключение не удалось, выводится ошибка (WSAGetLastError).

1. Взаимодействие клиента с сервером (sum())

* Отправка чисел:
  + Клиент отправляет строку с числами (send).
  + Если введено "stop", клиент завершает работу.
* Получение результата:
  + Клиент ожидает ответ от сервера (recv).
  + Полученная сумма выводится в консоль.

1. Основной цикл клиента (menu())

* Пользователь вводит IP и порт сервера.
* Если введено "stop", клиент завершает работу.
* В цикле запрашиваются числа, отправляются на сервер, выводится результат.

1. **Используемые функции**

|  |  |
| --- | --- |
| Функция | Назначение |
| WSAStartup | Инициализирует Winsock (библиотеку для работы с сокетами в Windows). |
| WSACleanup | Освобождает ресурсы Winsock. |
| socket | Создает сокет для сетевого взаимодействия. |
| bind | Привязывает сокет к конкретному IP-адресу и порту. |
| listen | Переводит сокет в режим прослушивания входящих подключений. |
| accept | Принимает входящее подключение и создает новый сокет для общения с клиентом. |
| connect | Подключается к серверу по указанному IP и порту. |
| send | Отправляет данные через сокет. |
| recv | Принимает данные из сокета. |
| closesocket | Закрывает сокет и освобождает ресурсы. |
| inet\_pton | Преобразует строку с IP-адресом в бинарный формат (для sockaddr\_in). |
| htons | Конвертирует 16-битное число из хостового порядка байт в сетевой (Big Endian). |

1. **Тестирование**

|  |  |  |
| --- | --- | --- |
| **Назначение** | **Вывод в консоль** | |
| **Сервер** | **Клиент** |
| **Запуск с корректными IP адресом и номером порта со стороны клиента, вычисление сумм вещественных чисел, завершение работы** |  |  |
| **Запуск с некорректным IP адресом с стороны клиента** |  |  |
| **Запуск с некорректным номером порта со стороны клиента** |  |  |

1. **Тест программы**

Серверная часть

#include <ws2tcpip.h>

#include <iostream>

#include <sstream>

#pragma comment(lib, "ws2\_32.lib")

std::string sum(std::string summands) {

std::istringstream sstream(summands);

double a, b;

sstream >> a >> b;

return std::to\_string(a + b);

}

int server() {

PCSTR server\_ip = "192.168.0.100";

int port = 2011;

int clients\_count = 2;

    int recv\_size = 2;

    int send\_size = 1;

int ret\_val;

WSAData wsa\_data;

if (WSAStartup(MAKEWORD(2, 2, ), &wsa\_data) != NO\_ERROR) {

std::cout << "WSAStartup failed" << std::endl;

return 0;

}

SOCKADDR\_IN server\_addr;

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(port);

inet\_pton(AF\_INET, server\_ip, &server\_addr.sin\_addr);

std::cout << "Creating the server socket" << std::endl;

SOCKET server\_socket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_IP);

if (server\_socket == INVALID\_SOCKET) {

std::cout << "Socket creation failed with error: " << WSAGetLastError();

WSACleanup();

return 0;

}

std::cout << "Binding the server socket" << std::endl;

ret\_val = bind(server\_socket, (sockaddr\*)&server\_addr, sizeof(server\_addr));

if (ret\_val == SOCKET\_ERROR) {

std::cout << "Binding server socket failed with error: " << WSAGetLastError();

closesocket(server\_socket);

WSACleanup();

return 0;

}

SOCKADDR\_IN client\_addr;

int client\_addr\_size = sizeof(client\_addr);

while (true) {

std::cout << "Listening to the server socket" << std::endl;

ret\_val = listen(server\_socket, clients\_count);

if (ret\_val == SOCKET\_ERROR) {

std::cout << "Listening to the server socket failed with error: " << WSAGetLastError();

closesocket(server\_socket);

WSACleanup();

return 0;

}

std::cout << "Accepting the client socket" << std::endl;

SOCKET client\_socket = accept(server\_socket, (sockaddr\*)&client\_addr, &client\_addr\_size);

if (client\_socket == INVALID\_SOCKET) {

std::cout << "Accepting failed with error: " << WSAGetLastError();

closesocket(server\_socket);

WSACleanup();

return 0;

}

std::cout << "Receiving client data" << std::endl;

std::string summands;

summands.resize(1024);

ret\_val = recv(client\_socket, (char\*)summands.data(), summands.size(), 0);

if (ret\_val == SOCKET\_ERROR) {

std::cout << "Receiving failed with error: " << WSAGetLastError();

closesocket(client\_socket);

closesocket(server\_socket);

WSACleanup();

return 0;

}

if (summands.data()[0]=='s') {

std::cout << "Shutting down the server" << std::endl;

closesocket(client\_socket);

closesocket(server\_socket);

WSACleanup();

return 1;

}

std::cout << "Sending results" << std::endl;

std::string res;

res = sum(summands);

ret\_val = send(client\_socket, res.data(), res.size(), 0);

if (ret\_val == SOCKET\_ERROR) {

std::cout << "Sending failed with error: " << WSAGetLastError();

closesocket(client\_socket);

closesocket(server\_socket);

WSACleanup();

return 0;

}

closesocket(client\_socket);

}

}

int main() {

    server();

}

Клиентская часть

#include <winsock2.h>

#include <ws2tcpip.h>

#include <locale.h>

#include <string>

#include <sstream>

#include <iostream>

#pragma comment(lib, "ws2\_32.lib")

const int SERVER\_ERROR = 0;

const int SUCCESS = 1;

const int EXIT = 2;

struct CLIENT {

SOCKET socket;

SOCKADDR\_IN socket\_addr;

};

CLIENT client;

std::string input() {

std::string summands;

std::getline(std::cin, summands);

return summands;

}

int make\_client(std::string server\_ip, int port) {

int ret\_val;

WSAData wsa\_data;

if (WSAStartup(MAKEWORD(2, 2), &wsa\_data) != NO\_ERROR) {

std::cout << "WSAStartup failed" << std::endl;

return SERVER\_ERROR;

}

client.socket = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_IP);

if (client.socket == INVALID\_SOCKET) {

std::cout << "Socket creation failed with error: " << WSAGetLastError() << "\n\n";

WSACleanup();

return SERVER\_ERROR;

}

client.socket\_addr.sin\_family = AF\_INET;

client.socket\_addr.sin\_port = htons(port);

inet\_pton(AF\_INET, (PCSTR)server\_ip.data(), &client.socket\_addr.sin\_addr);

return SUCCESS;

}

int connect\_server(std::string server\_ip, int port) {

int ret\_val;

ret\_val = connect(client.socket, (sockaddr\*)&client.socket\_addr, sizeof(client.socket\_addr));

if (ret\_val == SOCKET\_ERROR) {

std::cout << "Connecting failed with error: " << WSAGetLastError() << "\n\n";

closesocket(client.socket);

WSACleanup();

return SERVER\_ERROR;

}

return SUCCESS;

}

int sum(std::string summands, std::string\* res) {

int ret\_val;

ret\_val = send(client.socket, summands.data(), summands.size(), 0);

if (ret\_val == SOCKET\_ERROR) {

std::cout << "Sending failed with error: " << WSAGetLastError() << "\n\n";

closesocket(client.socket);

WSACleanup();

return SERVER\_ERROR;

}

if (summands.compare("stop") == 0) {

closesocket(client.socket);

WSACleanup();

return EXIT;

}

res->resize(1024);

ret\_val = recv(client.socket, (char\*)res->data(), 1024, 0);

if (ret\_val == SOCKET\_ERROR) {

std::cout << "Receiving failed with error: " << WSAGetLastError() << "\n\n";

closesocket(client.socket);

WSACleanup();

return SERVER\_ERROR;

}

closesocket(client.socket);

return SUCCESS;

}

int menu() {

int ret\_val = -1;

while (true) {

std::cout << "Input server IP and port space-separated or \"stop\" for exit" << std::endl;

std::string server\_ip;

int port;

std::cin >> server\_ip;

if (server\_ip.compare("stop") == 0)

return 0;

std::cin >> port;

std::cin.ignore();

ret\_val = make\_client(server\_ip, port);

if (ret\_val == SERVER\_ERROR)

continue;

ret\_val = connect\_server(server\_ip, port);

if (ret\_val == SERVER\_ERROR)

continue;

bool cont\_sum = true;

while (cont\_sum) {

std::cout << "Input 2 summands space separated or \"stop\" for exit" << std::endl;

std::string res;

int ret\_val = sum(input(), &res);

switch (ret\_val) {

case(SUCCESS):

std::cout << "Сумма: " << res << "\n\n";

make\_client(server\_ip, port);

connect\_server(server\_ip, port);

break;

case(EXIT):

return 0;

case(SERVER\_ERROR):

cont\_sum = false;

break;

}

}

}

}

int main() {

setlocale(LC\_ALL, "rus");

menu();

}