Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Факультет непрерывного и дистанционного обучения

**ИНДИВИДУАЛЬНАЯ ПРАКТИЧЕСКАЯ РАБОТА №1**

По дисциплине

**КОМПЬЮТЕРНЫЕ СЕТИ**

По теме

**«Создание последовательного сервера с установлением логического соединения TCP»**

**Вариант № 7**

Выполнил: Галкин И.В.

Группа № 894351, 3 курс

№ зачетки 6535021

Минск 2020

**Постановка задачи**

Разработать приложение, реализующее архитектуру «клиент-сервер». Необходимо реализовать последовательный сервер с установлением логического соединения (TCP). Логику взаимодействия клиента и сервера реализовать следующим образом:

7. Сервер генерирует прогноз погоды на неделю. Клиент посылает день недели и получает соответствующий прогноз.

**Краткое описание алгоритма**

При инициализации проектов, подключается библиотека WinSock2 посредством вызова WSAStartup и, в случае неудачного подключения, выводится сообщение об ошибке инициализации и проект завершает работу.

Далее, создается дескриптор сокета с заданным параметром SOCK\_STREAM указывающий, что это TCP сокет.

Для объекта sockaddr указываем ip адрес сервера (sin\_addr), занимаемый порт (sin\_port) и то, что сокет работает по internetwork (sin\_family).

* 1. **Серверная часть**

Первым делом, для обработки запросов нужно запустить сервер (проект ServerSide).

После начальной инициализации, необходимо привязать объект sockaddr к слушающему сокету при помощи вызова bind.

Далее, сервер начинает слушать все поступающие запросы на подключение (вызов listen с указанием сокета) и при первом же клиентском запросе функция accept выделит отдельный дескриптор сокета для нового подключения. В случае ошибки создания нового дескриптора выведется ошибка в консоль, но приложение продолжит свою работу.

Если подключение прошло успешно, сервер ожидает сообщения от подключенного клиента при помощи функции recv. В зависимости от передаваемого клиентом параметра сервер будет выдавать различные прогнозы погоды в ответ.

Ответ клиенту отправляется вызовом функции send.

* 1. **Клиентская часть**

Первым делом идет инициализация TCP сокета. Если инициализация прошла успешно, клиент пытается подключится к серверу через функцию connect с указанием параметров подключения и сокета. В случае ошибки подключения, выводится сообщение в консоль.

Если подключение прошло успешно, клиенту предлагается на ввод номер дня недели (от 1 до 7), либо выйти из программы.

После корректного ввода отправляется запрос на сервер с введенными данными при помощи функции send с указанием сокета. Если произошла ошибка отправки, выводится сообщение в консоль.

После этого, клиент ожидает ответ от сервера (функция recv) и выводит полученный результат в консоль.

**Листинги с пояснениями**

**Серверная часть:**

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <winsock2.h>

#include <iostream>

#include <string>

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

using namespace std;

void main()

{

WSADATA wsaData;

// NOTE: Try to initialize WinSock library

int initializeOutput = WSAStartup(MAKEWORD(2, 2), &wsaData);

// NOTE: Intialize errors handling

if (initializeOutput != 0) {

printf("WSAStartup failed with error: %d\n", initializeOutput);

return;

}

// NOTE: Define socket with TCP protocol

SOCKET connectSocket = socket(AF\_INET, SOCK\_STREAM, 0);

if (connectSocket == INVALID\_SOCKET) {

printf("socket failed with error: %ld\n", WSAGetLastError());

WSACleanup();

return;

}

// NOTE: Socket params

struct sockaddr\_in ad;

ad.sin\_family = AF\_INET;

ad.sin\_port = htons(8000);

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

int adSize = sizeof(ad);

// NOTE: Bind params to socket

bind(connectSocket, (struct sockaddr\*)&ad, sizeof(ad));

// NOTE: Start to listen requests

listen(connectSocket, 5);

while (1) {

sockaddr\_in remoteAd;

int remoteAdSize = sizeof(remoteAd);

printf("Waiting for clients\n");

// NOTE: Receive connections

SOCKET personalSocket = accept(connectSocket, (struct sockaddr\*)&remoteAd, &remoteAdSize);

if (personalSocket == INVALID\_SOCKET) {

printf("socket failed with error: %ld\n", WSAGetLastError());

continue;

}

printf("\nCreated connection with client\n");

char dataBuffer[1024];

while (recv(personalSocket, dataBuffer, sizeof(dataBuffer), 0) != 0) {

printf("Received packet from %s:%d\n", inet\_ntoa(remoteAd.sin\_addr), ntohs(remoteAd.sin\_port));

printf("Data: %s\n", dataBuffer);

char response[1024];

int choice = atoi(dataBuffer);

switch (choice) {

case 49:

strcpy(response, "Monday:\nDay: Sunshine and clouds mixed. High 46F. Winds WNW at 10 to 15 mph.\nNight: Mostly clear. Low 32F. Winds WNW at 10 to 15 mph.");

break;

case 50:

strcpy(response, "Tuesday:\nDay: Sunny along with a few clouds. High near 40F. Winds light and variable.\nNight: Mostly cloudy skies. Low 38F. Winds S at 5 to 10 mph.");

break;

case 51:

strcpy(response, "Wednesday:\nDay: Cloudy in the morning, then off and on rain showers during the afternoon hours. High 58F. Winds SSE at 15 to 25 mph. Chance of rain 40%. Higher wind gusts possible.\nNight: Windy with rain likely. Potential for heavy rainfall. Low 49F. Winds SSE at 20 to 30 mph. Chance of rain 80%. 1 to 2 inches of rain expected.");

break;

case 52:

strcpy(response, "Thursday:\nDay: Rain showers early with some sunshine later in the day. Morning high of 55F with temps falling sharply to near 35. Winds SW at 10 to 20 mph. Chance of rain 50%.\nNight: Mainly clear skies. Low near 25F. Winds W at 10 to 20 mph.");

break;

case 53:

strcpy(response, "Friday:\nDay: Sunny skies. High 33F. Winds W at 10 to 15 mph.\nNight: Clear. Low 27F. Winds W at 5 to 10 mph.");

break;

case 54:

strcpy(response, "Saturday:\nDay: Except for a few afternoon clouds, mainly sunny. High 38F. Winds light and variable.\nNight: Partly to mostly cloudy. Low 32F. Winds light and variable.");

break;

case 55:

strcpy(response, "Sunday:\nDay: Cloudy skies with periods of rain later in the day. High 49F. Winds SE at 5 to 10 mph. Chance of rain 60%.\nNight: A steady rain in the evening. Showers continuing late. Low 36F. Winds W at 5 to 10 mph. Chance of rain 70%. Rainfall near a half an inch.");

break;

default:

strcpy(response, "Received incorrect data");

break;

}

send(personalSocket, response, strlen(response), 0);

}

closesocket(personalSocket);

}

// NOTE: Close socket

closesocket(connectSocket);

// NOTE: Dispose WinSock lib

WSACleanup();

}

**Клиентская часть:**

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#include <winsock2.h>

#include <iostream>

#include <conio.h>

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

using namespace std;

void main()

{

WSADATA wsaData;

// NOTE: Try to initialize WinSock library

int initializeOutput = WSAStartup(MAKEWORD(2, 2), &wsaData);

// NOTE: Intialize errors handling

if (initializeOutput != 0) {

printf("WSAStartup failed with error: %d\n", initializeOutput);

return;

}

// NOTE: Define socket with TCP protocol

SOCKET connectSocket = socket(AF\_INET, SOCK\_STREAM, 0);

if (connectSocket == INVALID\_SOCKET) {

printf("socket failed with error: %ld\n", WSAGetLastError());

WSACleanup();

return;

}

// NOTE: Socket params

struct sockaddr\_in ad;

ad.sin\_family = AF\_INET;

ad.sin\_port = htons(8000);

ad.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

int adSize = sizeof(ad);

// NOTE: Try to connect to the server

int connectionResult = connect(connectSocket, (struct sockaddr\*)&ad, adSize);

if (connectionResult == SOCKET\_ERROR) {

closesocket(connectSocket);

cout << "Error while connecting socket\n";

return;

}

cout << "You was successfully connected to the server" << endl << endl;

cout << "This program will generate weather forecast for selected day of week" << endl;

int input = 0;

while (input != 8) {

fflush(stdout);

// NOTE: UI

cout << endl << "Please, enter the day of week (from 1 to 7) or 8 to exit" << endl;

input = \_getche();

// NOTE: From 1 to 8

if (input < 49 || input > 56) {

cout << endl << "\*\* Please, input correct number \*\*" << endl;

continue;

}

// NOTE: Exit

if (input == 56) {

break;

}

// NOTE: Prepare request buffer

char inputBuffer[1024];

sprintf\_s(inputBuffer, "%d", input);

// NOTE: Send request

send(connectSocket, inputBuffer, sizeof(inputBuffer), 0);

// NOTE: Receive response

char responseBuffer[1024];

int response = recv(connectSocket, responseBuffer, sizeof(responseBuffer), 0);

if (response == SOCKET\_ERROR) {

int iError = WSAGetLastError();

if (iError == WSAEWOULDBLOCK)

printf("\nrecv failed with error : WSAEWOULDBLOCK\n");

else

printf("\nrecv failed with error code: %ld\n", iError);

continue;

}

responseBuffer[response] = '\0';

cout << endl << "Server response:\n" << responseBuffer << endl;

}

// NOTE: Close socket

closesocket(connectSocket);

// NOTE: Dispose WinSock lib

WSACleanup();

}

**Результаты тестирования**

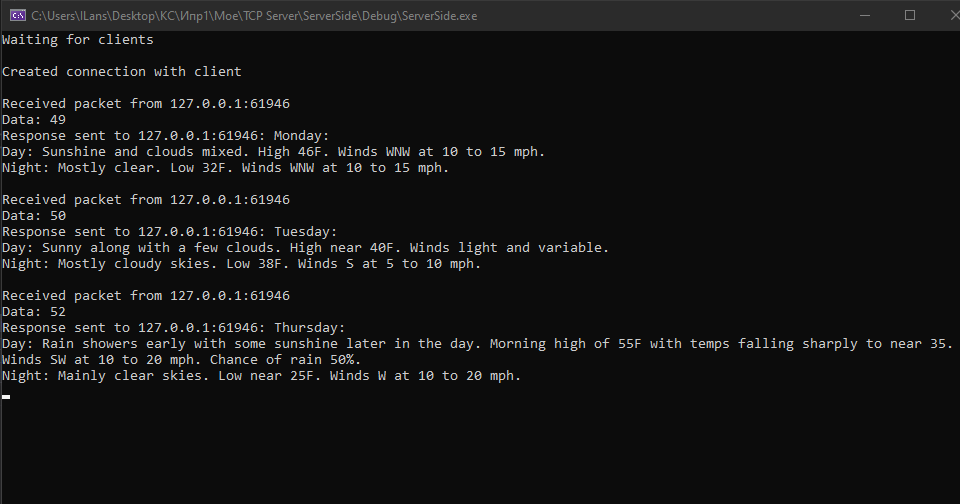


Рисунок 1 – Пример работы сервера

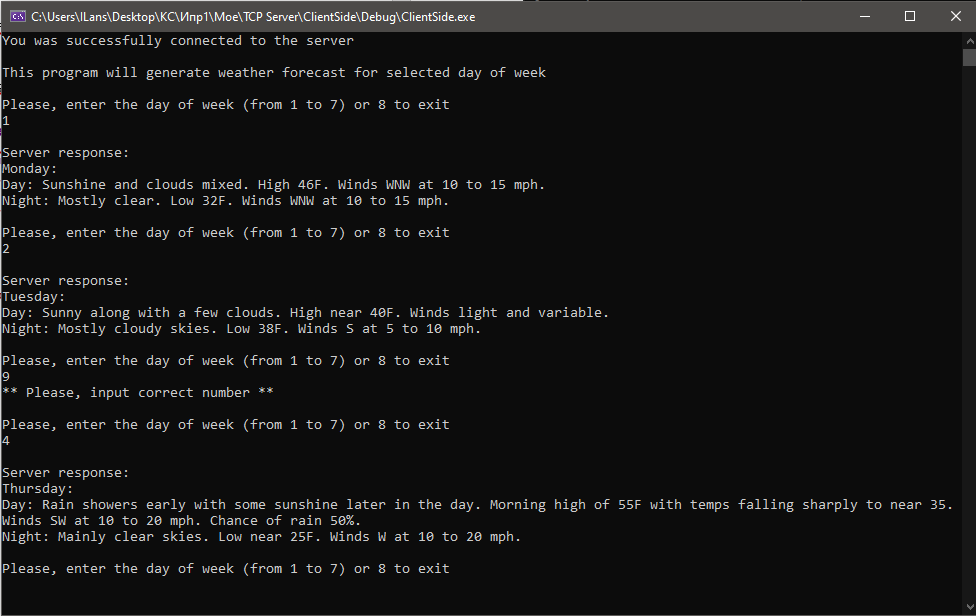


Рисунок 2 – Пример работы клиента

**Контрольные вопросы**

1. **Какая технология называется межсетевым обменом (internetworking)?**

Межсетевое взаимодействие — это способ соединения компьютерной сети с другими сетями с помощью шлюзов, которые обеспечивают общепринятый порядок маршрутизации пакетов информации между сетями. Полученная система взаимосвязанных сетей называется составной сетью, или просто интерсетью.

Наиболее ярким примером межсетевого взаимодействия является Интернет, сеть сетей, основанная на многих базовых технологиях оборудования, но объединённая стандартным набором протоколов межсетевого взаимодействия, известного как TCP/IP.

Самым простым примером составной сети являются две локальные сети, соединённые с помощью маршрутизатора. Использование коммутатора или концентратора для соединения локальных сетей не предполагает межсетевого взаимодействия, а лишь расширяет первоначальную сеть.

1. **Объясните понятие «протоколы» в контексте технологий обмена данными. Что они включают? Примеры.**

Для обеспечения согласованной работы в сетях передачи данных используются различные коммуникационные протоколы передачи данных - наборы правил, которых должны придерживаться передающая и принимающая стороны для согласованного обмена данными. Протоколы — это наборы правил и процедур, регулирующих порядок осуществления некоторой связи.

 Протоколы различны в зависимости от структуры сети. Для организации передачи информации в сети, разработали специальные правила - стандарты, затем эти правила объединили в протоколы

Протокол определяет способ доступа компьютера к кабелю - передающей среде, и также способ передачи информации от одного компьютера к другому.

Существует множество протоколов. И хотя все они участвуют в реализации связи, каждый протокол имеет различные цели, выполняет различные задачи, обладает своими преимуществами и ограничениями.

Протоколы работают на разных уровнях модели взаимодействия открытых систем OSI/ISO. Функции протоколов определяются уровнем, на котором он работает. Несколько протоколов могут работать совместно. Это так называемый стек, или набор, протоколов.

Наиболее популярные сетевые протоколы:

* TCP/IP (Transmission Control Protocol/Internet Protocol)
* AppleTalk
* Набор протоколов OSI.
* DLC (Data Link Control)
* PPTP (Point-to-Point Tunneling Protocol)
* NetBIOS (Network Basic Input/Output System)

Для примера, рассмотрим набор протоколов OSI. Модель OSI – 7-уровневая логическая модель работы сети. Реализуется группой протоколов и правил связи, организованных в несколько уровней:

* Физический уровень – определяет физические характеристики линий связи.
* Канальный уровень – определяет правила использования физического уровня узлами сети.
* Сетевой уровень – отвечает за адресацию и доставку сообщений
* Транспортный уровень – контролирует очередность прохождения компонентов сообщения
* Сеансовый уровень – координирует связь между двумя прикладными программами, работающими на разных рабочих станциях
* Уровень представления – отвечает за преобразования данных из внутреннего формата компьютера в формат передачи
* Прикладной уровень – пограничных уровень между прикладной программой и другими уровнями, обеспечивая удобный интерфейс связи для сетевых программ пользователя.

1. **Назовите отличия TCP/IP от других средств передачи данных.**

TCP/IP - единый промышленный стандарт, разработанный для глобальных сетей (Wide Area Networks, WAN).

Основной плюс - TCP/IP оправданно считается наиболее совершенным и распространенным протоколом из всех доступных на сегодняшний день.

Все современные операционные системы поддерживают протокол TCP/IP, и почти все крупные сети используют его для приема/передачи данных. Кроме того, протокол TCP/IP является стандартным для Интернет. Поэтому если в сети используется доступ в Интернет, что сейчас встречается на каждом шагу, то тут только один выход - непременно устанавливать протокол TCP/IP. TCP/IP очень хорош и для расширения сети. С его использованием локальные сети "легко превращаются ..." в глобальные.

Другое преимущество TCP/IP - возможность объединения неоднородных систем. Сегодня существует множество утилит доступа и передачи данных, позволяющих взаимодействовать самым различным системам. Некоторые из них, например FTP (File Transfer Protocol) и Telnet, поставляются с Windows NT.

1. **Дайте определение понятию «сокет»**

Для обеспечения сетевых коммуникаций используются сокеты. Сокет — это конечная точка сетевых коммуникаций. Каждый использующийся сокет имеет тип и ассоциированный с ним процесс. Сокеты существуют внутри коммуникационных доменов. Домены — это абстракции, которые подразумевают конкретную структуру адресации и множество протоколов, которое определяет различные типы сокетов внутри домена. Примерами коммуникационных доменов могут быть: UNIX домен, Internet домен, и т.д.

В Internet домене сокет — это комбинация IP адреса и номера порта, которая однозначно определяет отдельный сетевой процесс во всей глобальной сети Internet. Два сокета, один для хоста-получателя, другой для хоста-отправителя, определяют соединение для протоколов, ориентированных на установление связи, таких, как TCP.

1. **Опишите функцию, которая используется для приема данных через сокет потока (протокол TCP).**

Для приема данных через сокет потока в случае TCP используется функция:

int recv(int s, char \*buf, int len, int flags).

Первый аргумент – дескриптор сокета, из которого нужно получить данные.

Второй аргумент – буфер для хранения полученных данных.

Третий аргумент – размер буфера для хранения полученных данных

Четвертый аргумент – комбинация флагов, указывающих режим чтения файлов. По умолчанию (значение 0) полученные данные удаляются из сокета.

Функция возвращает количество считанных байтов. В случае ошибки, возвращается -1.

1. **Назовите функцию, используемую для создания сокета. Опишите ее параметры.**

Для создания сокета используется функция socket.

int socket (int domain, int type, int protocol);

Первый аргумент – указывает семейство протоколов создаваемого сокета. Задает правила использования именования и формат адреса. Может принимать такие значения как PF\_INET (IPv4), PF\_INET6 (IPv6), PF\_UNIX (локальные сокеты)

Второй аргумент – указывает на тип сокета. Существует несколько типов:

* SOCK\_STREAM – надёжная потокоориентированная служба (TCP) или потоковый сокет.
* SOCK\_DGRAM служба датаграмм (UDP) или датаграммный сокет.
* SOCK\_SEQPACKET – надежная служба последовательных пакетов
* SOCK\_RAW – сырок сокет – сырой протокол поверх сетевого уровня

Третий аргумент – определяет используемый транспортный протокол. Значение 0 может быть использовано для выбора протокола по умолчанию из указанного семейства (domain) и типа (type). Самые распространённые – это IPPROTO\_TCP, IPPROTO\_SCTP, IPPROTO\_UDP, IPPROTO\_DCCP.

Функция возвращает дескриптор в случае успешного создания сокета или возвращает -1 в случае ошибки.

1. **Опишите функцию, которая используется для пересылки данных через сокет потока (протокол TCP).**

Для пересылки данных через сокет потока в случае TCP используется функция send.

int send (int s, char\* buf, int len, int flags)

Первый аргумент – дескриптор сокета

Второй аргумент – буфер для передачи данных

Третий аргумент – длина буфера для передачи данных

Четвертый аргумент – комбинации флагов для отправки. По умолчанию значение равно нулю, что значит, что запись в сокет происходит в порядке поступления байтов.

Функция возвращает количество записанных в сокет байтов или -1, если произошла ошибка.

1. **Что возвращает функция accept(), в том случае, если соединение открывается успешно?**

Функция возвращает новый дескриптор сокета, через который и происходит общение клиента с сервером.

1. **Назовите функцию, которая используется в приложении-клиенте для посылки запроса на открытие соединения. Опишите ее параметры.**

Для посылки запроса на открытия соединения используется функция connect.

int connect (int s, char \* name, int namelen);

Первый аргумент – дескриптор сокета клиента.

Второй аргумент – объект-структура адреса сервера (объект-структура sockaddr).

Третий аргумент - длина структуры адреса.

Функция возвращает -1 в случае ошибки, в случае успешного выполнения - 0.