МИНЕСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧЕРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«Брестский государственный технический университет»

Кафедра «Интеллектуальные информационные технологии»

Лабораторная работа №1

По дисциплине «Аппаратное и программное обеспечение сетей»

Тема: «Организация TCP – сервера и TCP – клиента»

Выполнил:

студент 3 курса

группы ИИ-21(2)

Пучинский А.А.

Проверил:

Степанчук В.И.

Брест 2023

Цель работы: 1) изучить основы программирования сетевых приложений Windows на базе библиотеки WINSOCK2.H; 2) приобрести навыки по практическому использованию библиотеки для реализации сетевых приложений в среде С++ на базе протоколов TCP.

Вариант 14.

|  |  |  |
| --- | --- | --- |
| Вариант задания для бригады студентов | Номер задания для реализации TCP (UDP) сервера (см. табл. 2). | Номер задания для реализации TCP (UDP) клиента (см. табл. 3). |
| 14 | 6 | 7 |

|  |  |
| --- | --- |
| Номер задания | Функция, выполняемая TCP-сервером |
| 6 | После приема каждой группы из 48 символов сервер отсылает подтверждение в виде контрольной суммы ASCII-кодов символов группы и количество символов, принятых сервером от начала сеанса. Если в потоке принятых символов встречается последовательность «~#~», то сервер отсылает сообщение об окончании сеанса и разрывает соединение. |

Таблица 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | Ввод символов с отсылкой введенной строки по нажатию на клавишу | Ведение файла протокола событий, включающих:  1) время начала и окончания соединения;  2) передаваемую серверу строку и время передачи строки; | Возможность разрыва соединения при помощи команды:  **disconnect <адрес> <порт>** | 2) Автоматическое подключение к серверу с заданным по умолчанию адресом при запуске клиента |
| 7 | PgDn | 1), 2) | + | 2) |

Таблица 3

Реализация TCP-сервера:

import socket

import time

# Создаем TCP-сервер

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# Устанавливаем хост и порт для сервера

host = 'localhost'

port = 12345

# Привязываем сервер к заданному хосту и порту

server\_socket.bind((host, port))

# Ожидаем не более 1 клиента одновременно

server\_socket.listen(1)

print(f"Сервер слушает на {host}:{port}")

# Функция для вычисления контрольной суммы ASCII-кодов символов

def calculate\_checksum(data):

    if isinstance(data, int):

        data = str(data)

    checksum = 0

    for char in data:

        checksum += ord(char)

    return checksum

# Открываем файл протокола

log\_file\_path = "server\_log.txt"

while True:

    # Ожидаем подключение клиента

    client\_socket, client\_address = server\_socket.accept()

    print(f"Подключение клиента с адресом {client\_address}")

    data = b""  # Инициализируем пустые байты для хранения данных от клиента

    # Записываем время начала соединения

    start\_time = time.time()

    with open(log\_file\_path, "a") as log\_file:

        log\_file.write(f"Начало соединения: {time.ctime(start\_time)}\n")

    while True:

        # Принимаем данные от клиента (не более 48 символов за раз)

        chunk = client\_socket.recv(48)

        if not chunk:

            break  # Если клиент закрыл соединение, выходим из цикла

        data += chunk  # Добавляем принятые данные к общим данным

        # Проверяем наличие последовательности "~#~"

        if b"~#~" in data:

            # Если последовательность обнаружена, отправляем сообщение об окончании сеанса и разрываем соединение

            response = f"Сеанс завершен. Принято символов: {len(data)}\n"

            client\_socket.send(response.encode())

            client\_socket.close()

            print(f"Сообщение об окончании сеанса отправлено. Закрыто соединение с клиентом.")

            with open(log\_file\_path, "a") as log\_file:

                # Записываем время окончания соединения

                end\_time = time.time()

                log\_file.write(f"Окончание соединения: {time.ctime(end\_time)}\n")

                log\_file.write(f"Продолжительность соединения: {end\_time - start\_time:.2f} сек\n")

            break

        else:

            # Если последовательность "~#~" не найдена, отправляем подтверждение

            checksum = calculate\_checksum(data.decode())  # Преобразовать data в строку перед вычислением контрольной суммы

            response = f"Подтверждение: {checksum}, Принято символов: {len(data)}\n"

            client\_socket.send(response.encode())

            # Записываем передаваемую клиентом строку и время передачи строки

            with open(log\_file\_path, "a") as log\_file:

                log\_file.write(f"Передано: {data.decode()}, Время передачи: {time.ctime()}\n")

            # Очищаем data для приема следующей группы символов

            data = b""

server\_socket.close()

Реализация TCP-клиента:

import socket

import keyboard

import time

server\_address = ('localhost', 12345)

# Функция для отправки данных на сервер

def send\_data(sock, data):

    try:

        sock.send(data.encode())

    except:

        print("Ошибка отправки данных")

# Создаем сокет и подключаемся к серверу

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(server\_address)

print("Клиент подключен к серверу")

try:

    while True:

        if keyboard.is\_pressed("Page Down"):  # Если нажата клавиша PgDn

            user\_input = input("Введите текст для отправки на сервер (или 'disconnect' для разрыва соединения): ")

            if user\_input.lower() == "disconnect":

                send\_data(client\_socket, "~#~")  # Отправляем последовательность "~#~" для разрыва соединения

                break

            send\_data(client\_socket, user\_input)

            response = client\_socket.recv(1024)  # Получение подтверждения от сервера

            print("Подтверждение от сервера:", response.decode())

except KeyboardInterrupt:

    pass

# Закрываем соединение

client\_socket.close()

**Вывод:** 1) изучил основы программирования сетевых приложений Windows на базе библиотеки WINSOCK2.H; 2) приобрёл навыки по практическому использованию библиотеки для реализации сетевых приложений в среде С++ на базе протоколов TCP.