МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«Брестский государственный технический университет»

Кафедра «Интеллектуальные информационные технологии»

Лабораторная работа №2

По дисциплине «Аппаратное и программное обеспечение сетей»

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

Выполнил:

студент 3 курса

группы ИИ-21(2)

Литвинюк Т. В.

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 UDP.

Вариант 7.

|  |  |  |
| --- | --- | --- |
| 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). |
| 7 | 6 | 2 |

|  |  |
| --- | --- |
| 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;  3) принимаемую от сервера строку и время приема строки. | Возможность разрыва соединения при помощи команды:  **disconnect <адрес> <порт>** | 1) Задание в программе клиента специальной команды и параметров: подключения к серверу: **сonnect**  **<адрес> <порт>**  2) Автоматическое подключение к серверу с заданным по умолчанию адресом при запуске клиента |
| 2 | PgUp | 1), 3) | - | 1) |

Реализация UDP-сервера:

import socket

server = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server.bind(('localhost', 8080))

print("Сервер запущен и ожидает сообщения...")

all\_messages = ""

client\_address = None

while True:

    data, client\_address = server.recvfrom(1024)

    message = data.decode()

    if '~#~' in message:

        server.sendto("!close\_connection".encode(), client\_address)

        server.close()

        break

    all\_messages += message

    if len(all\_messages) > 48:

        control\_sum = sum(map(lambda char: ord(char), all\_messages[:-48]))

        control\_len = len(all\_messages)

        response = f"!control\_sum: {control\_sum}"

        server.sendto(response.encode(), client\_address)

        response = f" !control\_len: {control\_len}"

        server.sendto(response.encode(), client\_address)

    print(message)

    server.sendto(data, client\_address)

Реализация UDP-клиента:

from datetime import datetime

import socket

import threading

import keyboard

import pyautogui

file = open("log.txt", 'w')

client = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  # Создаем UDP сокет

server\_address = ('localhost', 8080)

while True:

    command = input(">>> ")

    if command.startswith("connect"):

        ip\_address, host = command.split()[1:]

        break

file.write(str(datetime.now()) + " --- connection to server\n")

FLAG = True

def press():

    while True:

        keyboard.wait('home')

        pyautogui.press('Enter')

def input\_handler():

    global FLAG

    all\_messages = ""

    while FLAG:

        command = input()

        all\_messages += command

        client.sendto(command.encode(), server\_address)  # Отправляем данные на сервер

        data, server = client.recvfrom(1024)  # Получаем данные от сервера

        message = data.decode()

        if "!close\_connection" in message:

            FLAG = False

            file.write(str(datetime.now()) + " --- disconnect from server\n")

            file.close()

            client.close()

            break

        elif "!control\_sum" in message:

            control\_sum = int(message.split()[1])

            data, server = client.recvfrom(1024)

            message = data.decode()

            print("Сумма: ", control\_sum)

            print("Количество всех символов: ", len(all\_messages))

            if control\_sum == sum(map(lambda char: ord(char), all\_messages[:-48])) and len(all\_messages) == int(message.split()[1]):

                print("Сумма верная.")

            else:

                print("Сумма не верная.")

            data, server = client.recvfrom(1024)

            message = data.decode()

        file.write(message + f" --- {datetime.now()}\n")

        print(message)

exit\_thread = threading.Thread(target=press)

input\_thread = threading.Thread(target=input\_handler)

exit\_thread.start()

input\_thread.start()

exit\_thread.join()

input\_thread.join()

client.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 UDP.