МИНЕСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧЕРЕЖДЕНИЕ 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)

Кирилович 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 4.

|  |  |  |
| --- | --- | --- |
| 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). |
| 4 | 5 | 5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер задания | | Функция, выполняемая TCP-сервером | | | |
| 5 | | После приема каждой цепочки символов, заканчивающейся символом «.», сервер отсылает предложение, в котором слова (цепочки символов, отделенные пробелом) расположены в обратном порядке. Если встречается последовательно два символа «..», то сервер отсылает сообщение об окончании сеанса и разрывает соединение. | | | |
| 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 |
| 5 | Home | | 1), 3) | + | 1) |

Реализация UDP-сервера:

import socket

HOST = '127.0.0.1'

PORT = 12345

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

server\_socket.bind((HOST, PORT))

print(f"Server listening on {HOST}:{PORT}")

while True:

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

if not data:

continue

received\_data = data.decode()

print(f"Received: {received\_data}")

if ".." in received\_data:

response = "The session has ended."

server\_socket.close()

break

else:

words = received\_data.strip(".").split()

reversed\_sentence = " ".join(reversed(words))

response = reversed\_sentence

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

Реализация UDP-клиента:

import socket

import threading

import datetime

import readchar

def receive\_messages(client\_socket):

while True:

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

if not data:

with open("log.txt", "a") as log:

log.write(

f"Disconnecting from the current server..."

f"\tDisconnected at: {datetime.datetime.now()}\n")

print("DISCONNECTED")

break

message = data.decode('utf-8')

print(f"Received from server: {message}")

log\_message = f"Received from server: {message}" \

f"\tReceived at: {datetime.datetime.now()}"

with open("log.txt", "a") as log:

log.write(log\_message + "\n")

def connect\_to\_server(server\_host, server\_port):

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

client\_socket.connect((server\_host, server\_port))

log\_message = f"Connected to server at {server\_host}:{server\_port}" \

f"\tConnected at: {datetime.datetime.now()}"

print("CONNECTED")

with open("log.txt", "a") as log:

log.write(log\_message + "\n")

return client\_socket

def custom\_input():

input\_text = ""

while True:

char = readchar.readchar()

if char == readchar.key.ENTER: # Enter key

break

elif char == "@": # @ key

break

elif char == readchar.key.CTRL\_C: # Handle Ctrl+C as an interruption

raise KeyboardInterrupt

elif char == readchar.key.CTRL\_D: # Handle Ctrl+D as an EOF

raise EOFError

else:

input\_text += char

print(char, end='', flush=True)

print("")

return input\_text

def main():

client\_socket = None

server\_host, server\_port = None, None

while True:

user\_input = custom\_input()

if user\_input.startswith("connect"):

\_, server\_host, server\_port = user\_input.split()

server\_port = int(server\_port)

if client\_socket:

with open("log.txt", "a") as log:

log.write(

f"Disconnecting from the current server..."

f"\tDisconnected at: {datetime.datetime.now()}\n")

print("DISCONNECTED")

client\_socket.close()

client\_socket = connect\_to\_server(server\_host, server\_port)

receive\_thread = threading.Thread(target=receive\_messages, args=(client\_socket,))

receive\_thread.start()

elif user\_input.startswith("disconnect"):

if client\_socket:

with open("log.txt", "a") as log:

log.write(

f"Disconnecting from the current server..."

f"\tDisconnected at: {datetime.datetime.now()}\n")

print("DISCONNECTED")

client\_socket.close()

client\_socket = None

else:

print("Not currently connected to a server.")

elif client\_socket:

server\_address = (server\_host, server\_port)

client\_socket.sendto(user\_input.encode('utf-8'), server\_address)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**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.