МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ 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(1)

Романко Н.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 Python на базе протоколов TCP.

Вариант 15

|  |  |  |
| --- | --- | --- |
| 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). |
| 15 | 7 | 6 |

|  |  |
| --- | --- |
| 7 | В случае приема сервером в потоке символов команды find <строка> <имя файла> сервер выполняет поиск строки в текстовом файле с указанным именем и отсылает в качестве ответа количество найденных строк. Если в файле строка не найдена, 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 |
| 6 | PgUp | 2), 3) | - | 1) |

Таблица 3

Реализация TCP-сервера:

from socket import \*

import os

server = socket(

    AF\_INET, SOCK\_STREAM  #параметры подключения

)

server.bind(

    ('172.22.240.1', 7000)  #адресс и сокет сервера

)

server.listen(5)  #количество подключений

print("SERVER WORKING")

user, addr = server.accept()  #после подключения, выполнять далее

print(f"CONNECTED:\n\t{user}\n\t{addr}\n")

#user.send('Connected successful!'.encode('utf-8'))  #отправить сообщение

def send():

    message = input("Введите текст сообещния: ")

    user.send(message.encode('utf-8'))

def recive():

    print("Ожидание сообщения")

    msg = user.recv( 1024 ).decode('utf-8')

    print(f'SERVER MESSAGE:\n\t{msg}')

    msg = msg.split(" ")

    for word in msg:

        if word == "find" and len(msg) - msg.index(word) >= 3:

            find(msg[msg.index(word) + 1], msg[msg.index(word) + 2])

end = False

def find(string, name):

    global end

    end = False

    found = False

    number = 0

    try:

        with open(name, 'r') as file:

            words = file.read().split()

    except FileNotFoundError:

        print("File not find")

    for word in words:

        if string.lower() in word.lower():

            number += 1

            print("Word found: ", word)

            found = True

    print("Number of words found: ", number)

    if found == False:

        end = True

        print("Word not found, end of connection")

while True:

    if end == True:

        task = "a"

    else:

        task = input("Выберете действие: \n\t1-отправить сообщение\n\t2-получить сообщение\n")

    match task:

        case "1":

            send()

        case "2":

            recive()

        case "e":

            server.close()

            os.system("cls")

            break

        case "a":

            server.close()

            break

        case \_:

            print("Нет такого лействия")

Реализация TCP-клиента:

from socket import \*

import os, msvcrt, datetime

client = socket(

    AF\_INET, SOCK\_STREAM #Параметры подключения

)

ip = "172.22.240.1"

addr = ""

port = 0

connect = False

while connect == False:

    cmd = input()

    cmd = cmd.split()

    for word in cmd:

            if word == "connect" and cmd[cmd.index(word) + 1] == ip and int(cmd[cmd.index(word) + 2]) == 7000 and cmd.index(word) == 0:

                addr = cmd[cmd.index(word) + 1]

                port = int(cmd[cmd.index(word) + 2])

                server\_addr = (addr, port)

                connect = True

                client.connect(

                    (addr, port)  #Адрес и сокет куда подключаться

                )

def send():

    message = ""

    while True:

        if msvcrt.kbhit():

            key = msvcrt.getch()

            try:

                    print(key.decode('utf-8'),end = '', flush=True)

            except UnicodeDecodeError:

                    pass

            if key == b'\xe0' or key == b'\r':

                print(" ")

                client.send(message.encode('utf-8'))

                time = datetime.datetime.now()

                with open("C:\\work\\AIPOSOZI\\Lab1 - TCP\\log.txt", 'a') as file:

                     file.write(f"[{time}] Строка \"{message}\" отправленна на сервер\n")

                break

            else:

                try:

                    if key == b'I' or key == b'\r':

                         pass

                    else:

                        message += key.decode('utf-8')

                except UnicodeDecodeError:

                    pass

def recive():

    print("Ожидание сообщения")

    msg = client.recv( 1024 ).decode('utf-8')

    time = datetime.datetime.now()

    with open("C:\\work\\AIPOSOZI\\Lab1 - TCP\\log.txt", 'a') as file:

        file.write(f"[{time}] Строка \"{msg}\" полученна от сервера\n")

    print(f'SERVER MESSAGE:\n\t{msg}')

while True:

    task = input("Выберете действие:\n\t 1-отправить сообщение\n\t 2-получить сообщение\n")

    match task:

        case "1":

            send()

        case "2":

            recive()

        case "e":

            client.close()

            os.system("cls")

            break

        case \_:

            print("Нет такого действия")

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 Python на базе протоколов UDP.