МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ПИ

Факультет КНТ

Лабораторная работа №4

Тема: «Разработка простейших клиент-серверных программ с использованием TCP сокетов на базе Posix»

Курс: Протоколы компьютерных сетей

Выполнил

ст. гр. ПИ-18А

Данильчук К.М.

Проверил

Чернышова А.В.

Донецк – 2021

ЛИСТИНГ КОДА

Single server

import socket

print("Server start")

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

server\_socket.bind(("localhost", 51000))

server\_socket.listen(1)

message = str()

while True:

# stop server?

if message == "stop server":

server\_socket.close()

print("Server was closed")

break

print("Server is waiting client...")

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

client\_socket.settimeout(30)

print("Client have connected with address {}".format(client\_address))

# process for client

while True:

try:

message = client\_socket.recv(4096)

except Exception:

print("Client isn't available\n")

break

message = message.decode()

if message == "quit" or message == "stop server":

client\_socket.close()

break

print("Receive from client ({}): {}".format(client\_address, message))

message = message.upper()

print("Server send: {}".format(message))

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

Multi server

import socket

import threading

def client\_handler(client\_socket, client\_address):

print("Handler start for {}".format(client\_address))

while True:

try:

message = client\_socket.recv(4096)

except Exception:

print("Client isn't available\n")

break

message = message.decode()

if message == "quit":

client\_socket.close()

print("Client {} was closed".format(client\_address))

break

print("Receive from client ({}): {}".format(client\_address, message))

message = message.upper()

print("Server {} send: {}".format(threading.get\_ident(), message))

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

print("Start server")

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

server\_socket.bind(("localhost", 51000))

server\_socket.listen(10)

while True:

socket, address = server\_socket.accept()

socket.settimeout(60)

thread = threading.Thread(target=client\_handler, args=(socket, address))

thread.start()

Client

import socket

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

client\_socket.connect(("localhost", 51000))

client\_socket.settimeout(2)

while True:

message = input("-> ")

if message == "quit" or message == "stop server":

try:

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

except Exception:

pass

client\_socket.close()

print("Exit")

break

message = message.encode()

try:

client\_socket.send(message)

message = client\_socket.recv(4096)

except Exception:

print("Server isn't available")

print("Exit")

client\_socket.close()

break;

message = message.decode()

print("Server answer: {}".format(message))

СКРИНШОТЫ

Однопользовательский сервер

Пользователь 1 подключается к серверу и пишет разные письма, в то время второй пользователь попробовал подключиться и получил ошибку сервера, после чего пользователь 1 превысил таймаут и сервер его отключил. Сервер перешёл в режим ожидания нового пользователя. (см. рис. 1)

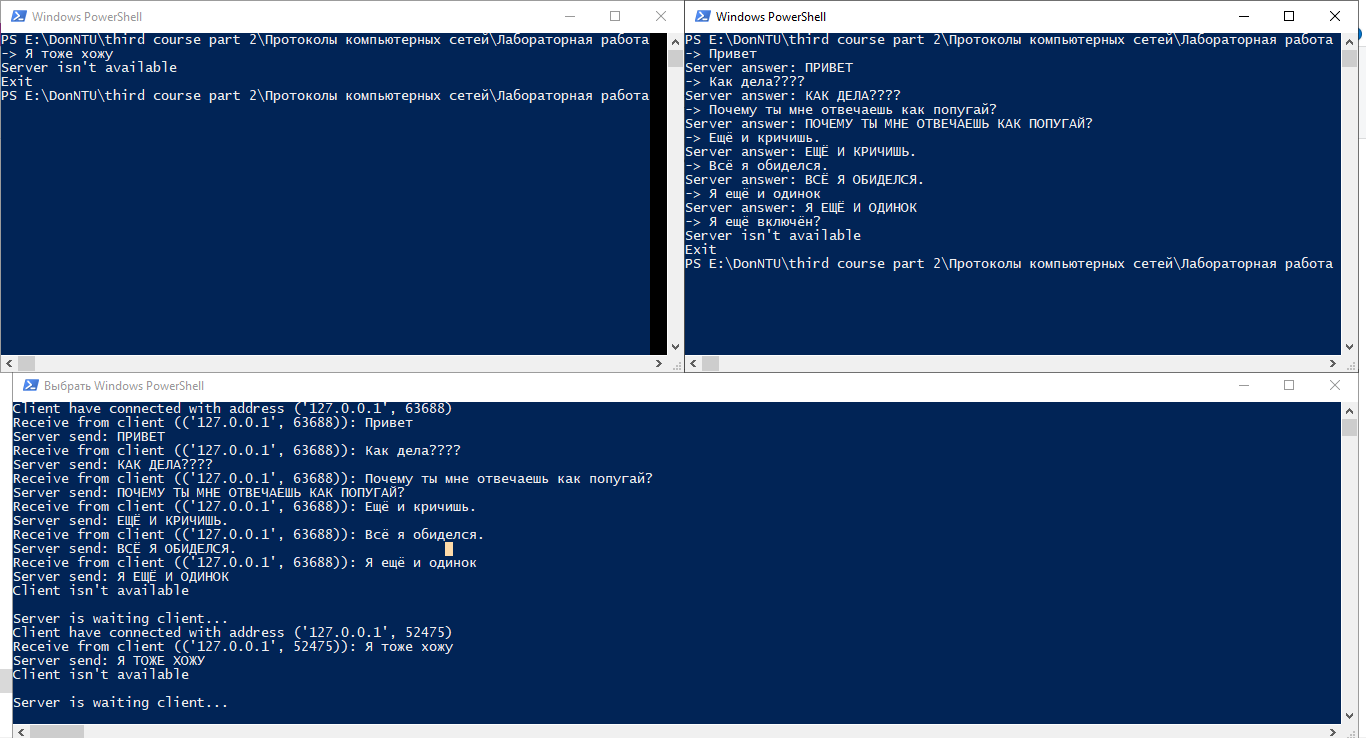


Рисунок 1 – Простое взаимодействие

Следующий сценарий: пользователь 1 подключается и отключается, после чего сервер ожидает пользователя 2, который подключается и отключает сервер.

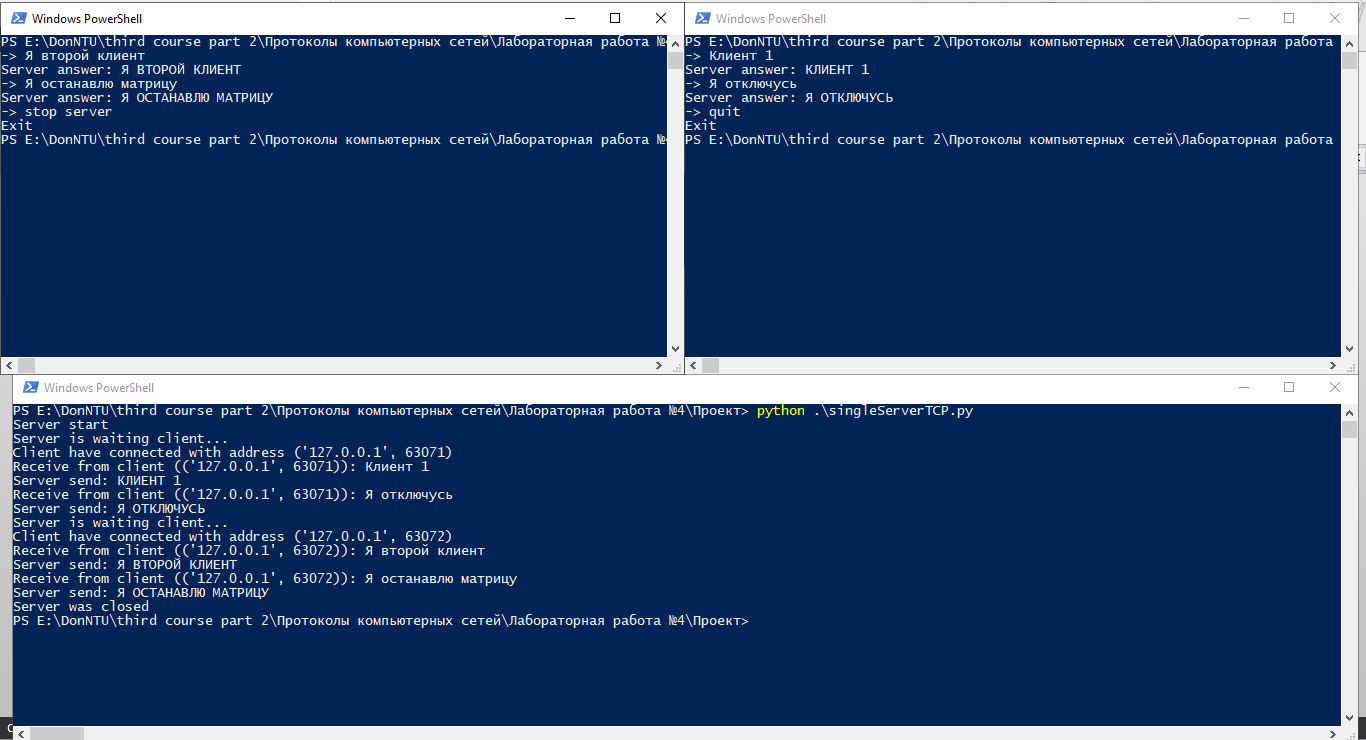


Рисунок 2 – Многопользовательский последовательный сервер

Многопользовательский сервер

Несколько пользователей обращаются к одному серверу, который обрабатывает каждое подключение в отдельном потоке.

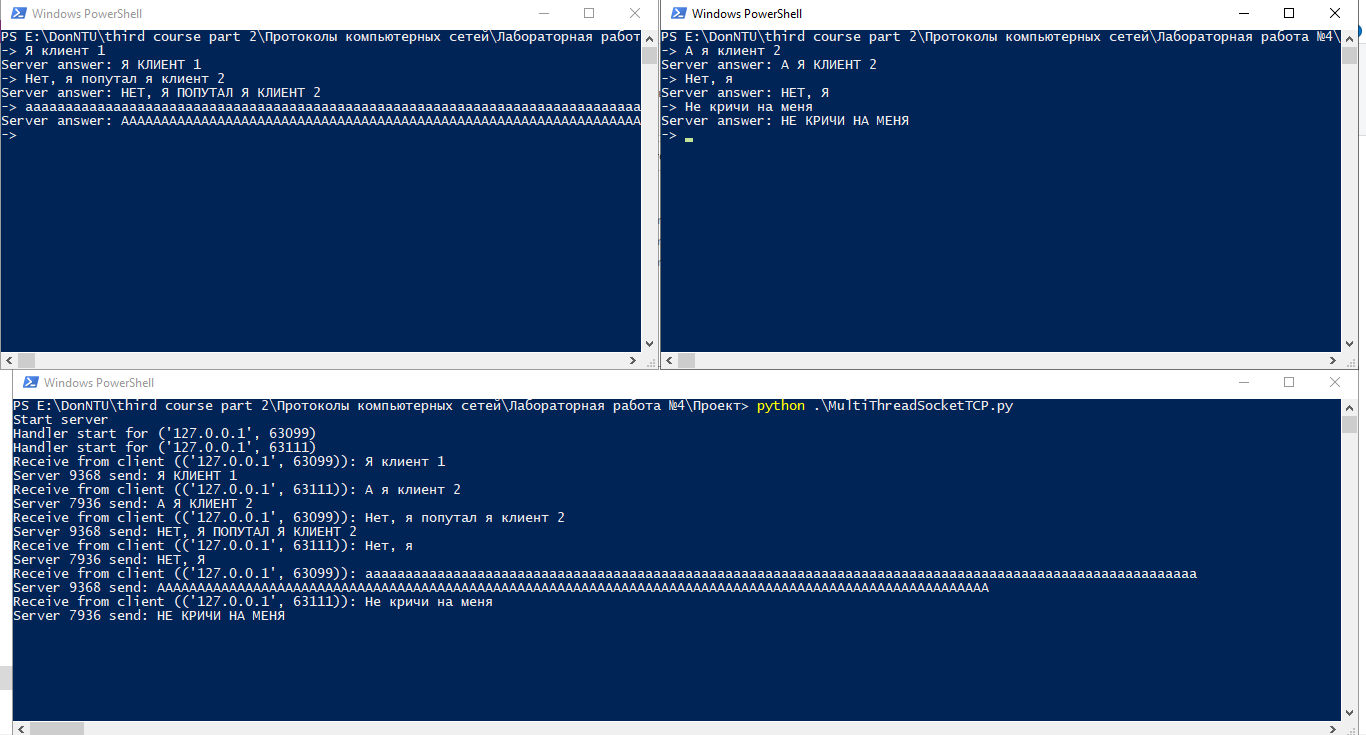


Рисунок 3 – Многопользовательская обработка клиентов