

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО»**

Отчет

По лабораторной работе №1

Работа с сокетами

Автор: Чан Дык Минь

Факультет: ИКТ

Группа: К33392

Преподаватель: Говоров Антон Игоревич

Санкт-Петербург, 2023

Задача №1

Реализовать клиентскую и серверную часть приложения. Клиент отправляет серверу сообщение "Hello, server". Сообщение должно отразиться на сервере. Сервер в ответ отправляет клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента. Обязательно использовать библиотеку socket. Реализовать с использованием протокола UDP.

Решение

1. Сервер

```
import socket

port = 2002
data_recv = 2048
msg_to_client = b"Hello client!"
host = "127.0.0.1"

def main():
    #Create a UDP socket and associate it with the specified IP address and port.
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.bind((host, port))

    try:
        while True:
            #Receive the data and Client address.
            msg, address = s.recvfrom(data_recv)
            print("Server reply to:", address, msg.decode("utf-8"))

            #Reply to Client
            s.sendto(msg_to_client, address)
    except KeyboardInterrupt:
        s.close()

if __name__ == "__main__":
    main()
```

2. Клиент

```
import socket

port = 2002
data_recv = 2048
msg_to_sv = b"Hello server!"
host = "127.0.0.1"

def main():
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

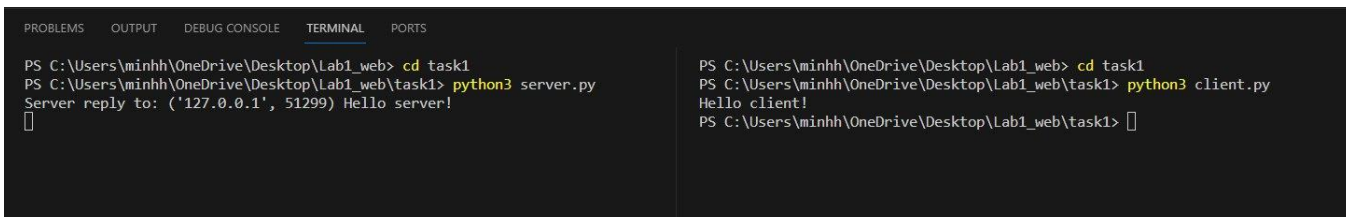
    #associate with the specified IP address and port.
    s.connect((host, port))

    try:
        #Send message to the server
        s.send(msg_to_sv)

        #Receive n bytes of data from the server
        msg, address = s.recvfrom(data_recv)
        print(msg.decode("utf-8"))
    except KeyboardInterrupt:
        s.close()

if __name__ == "__main__":
    main()
```

Демонстрация работы



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\minhh\OneDrive\Desktop\Lab1_web> cd task1
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web\task1> python3 server.py
Server reply to: ('127.0.0.1', 51299) Hello server!
[]

PS C:\Users\minhh\OneDrive\Desktop\Lab1_web> cd task1
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web\task1> python3 client.py
Hello client!
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web\task1> []
```

Задача №2

Реализовать клиентскую и серверную часть приложения. Клиент запрашивает у сервера выполнение математической операции, параметры которой вводятся с клавиатуры. Сервер обрабатывает полученные данные и возвращает результат клиенту.

Мой вариант, согласно списку группы, – площадь параллелограмма.

Решение

1. Сервер

```
import socket

port = 2002
data_recv = 2048
max_pending_conn = 8
host = "127.0.0.1"

#function to calculate the parallelogram area
def calc_parallelogram_area(pair):
    h, b = map(int, pair.split(" "))
    return h*b

def main():
    #Create a TCP socket and associate it with the specified IP address and port.
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((host, port))
    s.listen(max_pending_conn)

    try:
        while True:
            #Accept an incoming connection from the client.
            client, address = s.accept()

            #Receive n bytes of data from the server and decode it
            data = client.recv(port).decode("utf-8")
            print(f>Data recieved: ", data)
            result = calc_parallelogram_area(data)

            #Send the encoded result to client
            client.send(str(result).encode("utf-8"))

    except KeyboardInterrupt:
        client.close()

if __name__ == "__main__":
    main()
```

2. Клиент

```
import socket
port = 2002
data_recv = 2048
host = "127.0.0.1"
print("Parallelogram area calculation")

def main():
    while True:
        #input the height and the base of Parallelogram
        h_str = input("Enter height: ")
        b_str = input("Enter base: ")
        #check if the input are positive integer
        try:
            h = int(h_str)
            b = int(b_str)
            if h > 0 and b > 0:
                break
            else:
                print("Height and/or base is not positive integers.")
        except ValueError:
            print("Invalid input. Please enter valid integer values for height and base.")
    msg = (h_str+" "+b_str).encode("utf-8")
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))

    try:
        #Send input to server
        s.send(msg)
        #Get the answer from server
        msg, address = s.recvfrom(data_recv)
        print("Area of parallelogram is: ", msg.decode("utf-8"))

    except KeyboardInterrupt:
        s.close()

if __name__ == "__main__":
    main()
```

Демонстрация работы

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web> cd task2
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web\task2> python3 server2.py
Data recieved: 3 4
█
```

```
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web> cd task2
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web\task2> python3 client2.py
Parallelogram area calculation
Enter height: 3
Enter base: 4
Area of parallelogram is: 12
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web\task2> █
```

Задача №3

Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

Решение

1. Сервер

```
import socket

port = 2002
data_recv = 2048
max_pending_conn = 8
host = "127.0.0.1"

def main():
    #Create a TCP socket and associate it with the specified IP address and port.
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((host, port))
    s.listen(max_pending_conn)

    try:
        while True:
            #Accept an incoming connection from the client.
            client, address = s.accept()

            #Receive n bytes of data from the server and decode it
            data = client.recv(data_recv).decode("utf-8")
            print(f>Data recieved: ", data)

            # Create an HTTP response with a 200 OK code and insert the content of "index.html"
            into the response.
            with open("index.html") as index:
                response_type = "HTTP/1.1 200 OK \n"
                headers = "Content-Type: text/html \n\n"
                message_body = index.read()
                response = (response_type + headers + message_body).encode("utf-8")

            client.send(response)

    except KeyboardInterrupt:
        client.close()

if __name__ == "__main__":
    main()
```

Демонстрация работы

The image shows a Visual Studio Code (VS Code) editor window with a dark theme. The Explorer sidebar on the left shows a project named 'LAB1_WEB' with files including 'settings.json', 'docs', 'task1', 'client.py', 'server.py', 'task2', 'client2.py', 'server2.py', 'task3', 'index.html', 'server3.py', 'task4', 'client4.py', 'server4.py', 'task5', and 'server5.py'. The 'server3.py' file is selected and open in the editor. The code in 'server3.py' is a Python script that creates a TCP socket, binds it to 'localhost:2002', and listens for incoming connections. It accepts a connection, receives data, decodes it, and prints it. Then, it creates an HTTP response with a 200 OK status and the content of 'index.html', and sends it back to the client. The script includes a try-except block for KeyboardInterrupt and a main function guard.

```
9 #Create a TCP socket and associate it with the specified IP address and port.
10 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
11 s.bind((host, port))
12 s.listen(max_pending_conn)
13
14 try:
15     while True:
16         #Accept an incoming connection from the client.
17         client, address = s.accept()
18
19         #Receive n bytes of data from the server and decode it
20         data = client.recv(data_recv).decode("utf-8")
21         print(f"Data recieved: ", data)
22
23         # Create an HTTP response with a 200 OK code and insert the content of "index.html" as index:
24         with open("index.html") as index:
25             response_type = "HTTP/1.1 200 OK \n"
26             headers = "Content-Type: text/html \n\n"
27             message_body = index.read()
28             response = (response_type + headers + message_body).encode("utf-8")
29
30         client.send(response)
31
32 except KeyboardInterrupt:
33     client.close()
34
35 if __name__ == "__main__":
36     main()
```

Below the editor, the TERMINAL panel shows the command prompt output:

```
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web> cd task3
PS C:\Users\minhh\OneDrive\Desktop\Lab1_web\task3> python3 server3.py
Data recieved: GET / HTTP/1.1
Host: localhost:2002
Connection: keep-alive
sec-ch-ua: "Chromium";v="116", "Not)A;Brand";v="24", "Google Chrome";v="116"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
```

On the right side of the VS Code window, a web browser (Chromium) is open at 'localhost:2002'. The browser displays the text 'Hello world!' in a large, bold, black font on a white background.

Задача №4

Реализовать двухпользовательский или многопользовательский чат. Реализация многопользовательского чата позволяет получить максимальное количество баллов.

Решение

1. Сервер

```
import socket
import threading

port = 2002
data_recv = 2048
host = "127.0.0.1"
max_pending_conn = 8
lock = threading.Lock() #Synchronizing Threads
clients_list = []

def announce(msg):
    global clients_list
    for c in clients_list:
        c.send(msg.encode("utf-8"))

#Function for handling sending messages between clients.
def send_msg_handle(client, address):
    global clients_list
    while True:
        try:
            #Receive data from the client.
            msg = client.recv(data_recv).decode("utf-8")

            #Check if message is null
            if not msg:
                raise Exception

            #Send a message from the client to all other clients.
            with lock:
                for c in clients_list:
                    if c != client:
                        c.send(f"{address[1]}: {msg}".encode("utf-8"))
        except:
            tmp = clients_list
            clients_list = [i for i in tmp if i != client]
            print(f"Client {address} has left the chat")
```



```

        announce(f"Client {address[1]} has left the chat")
        break

def main():
    global clients_list

    #Create a TCP socket and associate it with the specified IP address and port.
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((host, port))
    s.listen(max_pending_conn)

    while True:
        try:
            #Accept an incoming connection from the client.
            client, address = s.accept()
            print(f"Client {address} has joined the chat")
            announce(f"Client {address[1]} has joined the chat")

            #Add the client socket to the list of clients.
            with lock:
                clients_list.append(client)

            #Start a separate thread to handle sending messages between clients.
            threading.Thread(target=send_msg_handle, args=(
                client, address)).start()
        except KeyboardInterrupt:
            s.close()
            break

if __name__ == "__main__":
    main()

```

2. Клиент

```
import socket
import threading

port = 2002
data_recv = 2048
host = "127.0.0.1"
lock = threading.Lock()

#Function for handling incoming messages from the server
def recv_msg_handle(client):
    while True:
        #Receive a message from the server and decode it from a byte string
        msg = client.recv(data_recv).decode("utf-8")
        print(msg)

def main():
    #Create a TCP socket and establish a connection to the server.
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))

    #Create a separate thread to receive messages from the server
    receive_thread = threading.Thread(target=recv_msg_handle, args=(s,))
    receive_thread.start()

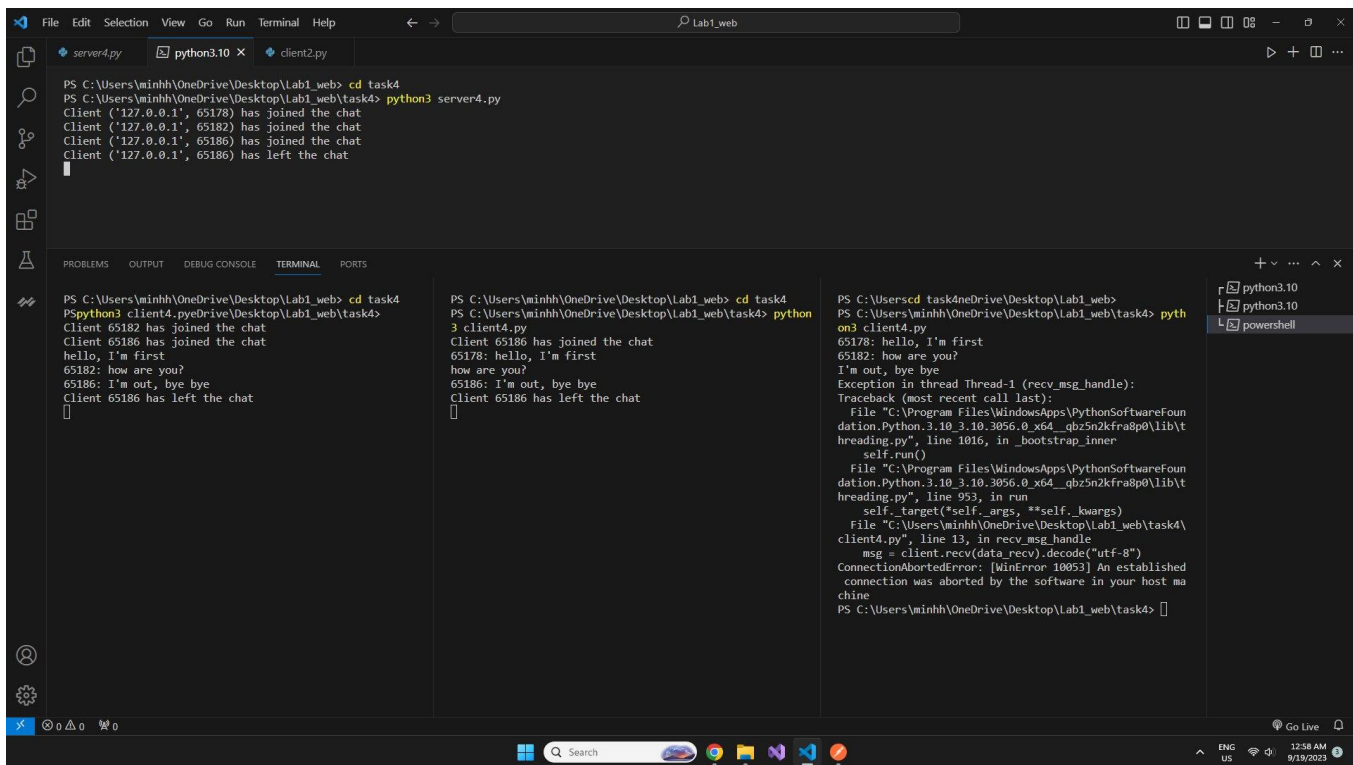
    while True:
        try:
            #Read the message from the keyboard
            message = input()

            #Encoding it into bytes and send the message to the server
            s.send(message.encode("utf-8"))

        except KeyboardInterrupt:
            s.close()
            break

if __name__ == "__main__":
    main()
```

Демонстрация работы



Задача №5

Необходимо написать простой web-сервер для обработки GET и POST http запросов средствами Python и библиотеки socket.

Решение

1. http-server

```
import socket

class MyHTTPServer:
    #The constructor of the MyHTTPServer class. It creates a socket object and sets options for it, then binds the socket to the given address and port, and listens on that port. Additionally, it initializes a dictionary (self.grades) to store the scores.
    def __init__(self, host, port):
        self.conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.conn.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.conn.bind((host, port))
        self.conn.listen(1)
        self.grades = {}

    #This function is the main loop of the server. It listens for and accepts connections from clients and calls the serve_client function to serve the client's request.
    def serve_forever(self):
        while True:
            client, address = self.conn.accept()
            self.serve_client(client)

    #This function reads data sent from the client, then calls the parse_request function to parse the HTTP request.
    def serve_client(self, client):
        data = client.recv(1024).decode("utf-8")
        self.parse_request(client, data)

    #This function parses the HTTP request to extract the method, URL, and HTTP version. It also tries to extract parameters from the URL if any.
    def parse_request(self, client, data):
        lines = data.split("\n")
        method, url, version = lines[0].split()
        params = (
            {p.split("=")[0]: p.split("=")[1] for p in url.split("?")[1].split("&")}
            if "?" in url
            else None
        )
        self.handle_request(client, method, params)
```

```

    #This function processes the HTTP request based on the method (GET or POST) and extracted
    parameters. If the method is GET, it sends the score list as HTML. If the method is POST, it stores
    the grade into the self.grades dictionary and sends a confirmation message.
    def handle_request(self, client, method, params):
        if method == "GET":
            self.send_response(client, 200, "OK", self.grades_to_html())
        elif method == "POST":
            discipline = params.get("discipline")
            grade = params.get("grade")
            self.grades[discipline] = grade
            self.send_response(client, 200, "OK", "Grade successfully saved!")
        else:
            self.send_response(client, 404, "Not Found", "Incorrect method, try a different
method.")

    #This function sends an HTTP response to the client with a status code, reason, and specific
    content. The response is sent as an HTTP string.
    def send_response(self, client, code, reason, body):
        response = f"HTTP/1.1 {code} {reason}\nContent-Type: text/html\n\n{body}"
        client.send(response.encode("utf-8"))
        client.close()

    #This function creates an HTML page containing a list of grades from the self.grades
    dictionary.
    def grades_to_html(self):
        page = (
            f"<html><body><ul>"
            f"{''.join([f'<li>{discipline}: {grade}' for discipline, grade in
self.grades.items()])}"
            f"</ul></body></html>"
        )
        return page

    def main():
        host = "127.0.0.1"
        port = 2002
        server = MyHTTPServer(host, port)
        try:
            server.serve_forever()
        except KeyboardInterrupt:
            server.conn.close()

if __name__ == "__main__":
    main()

```

Демонстрация работы

HTTP

http://localhost:2002/?discipline=Math&grade=3

Save

POST

http://localhost:2002/?discipline=Math&grade=3

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	discipline	Math			
<input checked="" type="checkbox"/>	grade	3			
	Key	Value	Description		

Body

Cookies

Headers (1)

Test Results

200 OK

7 ms

69 B

Save as Example

Pretty

Raw

Preview

Visualize

HTML

1

Grade successfully saved!

HTTP

http://localhost:2002/?discipline=English&grade=4

Save

POST

http://localhost:2002/?discipline=English&grade=4

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	discipline	English			
<input checked="" type="checkbox"/>	grade	4			
	Key	Value	Description		

Body

Cookies

Headers (1)

Test Results

200 OK

8 ms

69 B

Save as Example

Pretty

Raw

Preview

Visualize

HTML

1

Grade successfully saved!



http://localhost:2002/

Save



GET



http://localhost:2002/

Send



Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (1)

Test Results



200 OK

4 ms

104 B



Save as Example



Pretty

Raw

Preview

Visualize

- English: 4
- Math: 3

Вывод: За время работы в лаборатории я изучил сокеты на Python, знал, как инициализировать и соединить сервер с клиентом, смог построить простой многопоточный чат-канал. Кроме того, я также научился Узнать больше о HTTP-запросах, таких как POST и GET.