САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Web-программирование

Отчет

Лабораторная работа №1

Работа с сокетами

Выполнил:
Новиков Г. В.

Группа:
K33402

Проверил:
Говоров А. И.

Санкт-Петербург

2022 г.

Цель работы

Овладеть практическими навыками и умениями реализации web-серверов и использования сокетов.

Выполнение работы

Задание 1

Реализовать клиентскую и серверную часть приложения. Клиент отсылает серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отсылает клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.

```python
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(('127.0.0.1', 5000))
conn.listen(10)

client_socket, address = conn.accept()
data = client_socket.recv(16384)
data = data.decode('utf-8')
print(data)

client_socket.send(b'Hello client! \n')
conn.close()
```

server.py

```python
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.connect(("127.0.0.1", 5000))
conn.send(b'Hello, server! \n')
data = conn.recv(16384)
data = data.decode('utf-8')
print(data)
conn.close()
```

client.py

Задание 2

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

```python
# Теорема Пифагора
import socket
import math

server = socket.socket()
host = '127.0.0.1'
port = 5000
server.bind((host, port))
server.listen(3)


conn, address = server.accept()
first_side_bin = conn.recv(200)
second_side_bin = conn.recv(200)
first_side = first_side_bin.decode('utf-8')
second_side = second_side_bin.decode('utf-8')
hypotenuse = math.sqrt(int(first_side)**2 + int(second_side)**2)
conn.send(str(hypotenuse).encode())
conn.close()
```

server.py

```python
# Теорема Пифагора
import socket

conn = socket.socket()

conn.connect(("127.0.0.1", 5000))
first_side = input("Enter the length of the first side: ")
second_side = input("Enter the length of the second side: ")
conn.send(first_side.encode())
conn.send(second_side.encode())
hypotenuse_bin = conn.recv(200)
hypotenuse = hypotenuse_bin.decode('utf-8')
print('The hypotenuse of a triangle is:', hypotenuse)
conn.close()
```

client.py

Задание 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.

```python
import socket


class MyHTTPServer:

    def __init__(self, host, port):
        self.host = host
        self.port = port

    def serve_forever(self):
        # Starting a server on a socket, handling connections
        conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        conn.bind((self.host, self.port))
        conn.listen(10)
        while True:
            clientsocket, address = conn.accept()
            self.serve_client(clientsocket)

    def serve_client(self, clientsocket):
        # Client connection handling
        data = clientsocket.recv(16384)
        data = data.decode('utf-8')
        url, method, headers, body = self.parse_request(data)
        resp = self.handle_request(url, method, body)
        if resp:
            self.send_response(clientsocket, resp)

    def parse_request(self, data):
        #processing the http + request header
        data = data.replace('\r', '')
        lines = data.split('\n')
        method, url, protocol = lines[0].split()
        i = lines.index('')
        headers = lines[1:i]
        body = lines[-1]
        return url, method, headers, body

```

```python
    def handle_request(self, url, method, body):
        # url processing
        if url == "/":
            if method == "GET":
                resp = "HTTP/1.1 200 OK\n\n"
                with open('index.html', 'r') as f:
                    resp += f.read()
                return resp
            if method == "POST":

                newbody = body.split('&')
                for a in newbody:
                    if a.split('=')[0] == 'subject':
                        subjects.append(a.split('=')[1])
                    if a.split('=')[0] == 'mark':
                        marks.append(a.split('=')[1])

                resp = "HTTP/1.1 200 OK\n\n"
                resp += "<html><head><title>Journal</title></head><body><table border=1>"
                for s, m in zip(subjects, marks):
                    resp += f"<tr><td>{s}</td><td>{m}</td></tr>"
                resp += "</table></body></html>"
                return resp

    def send_response(self, clientsocket, resp):
        clientsocket.send(resp.encode('utf-8'))


if __name__ == '__main__':
    host = '127.0.0.1'
    port = 5000
    serv = MyHTTPServer(host, port)
    subjects = []
    marks = []
    try:
        serv.serve_forever()
    except KeyboardInterrupt:
        pass
```

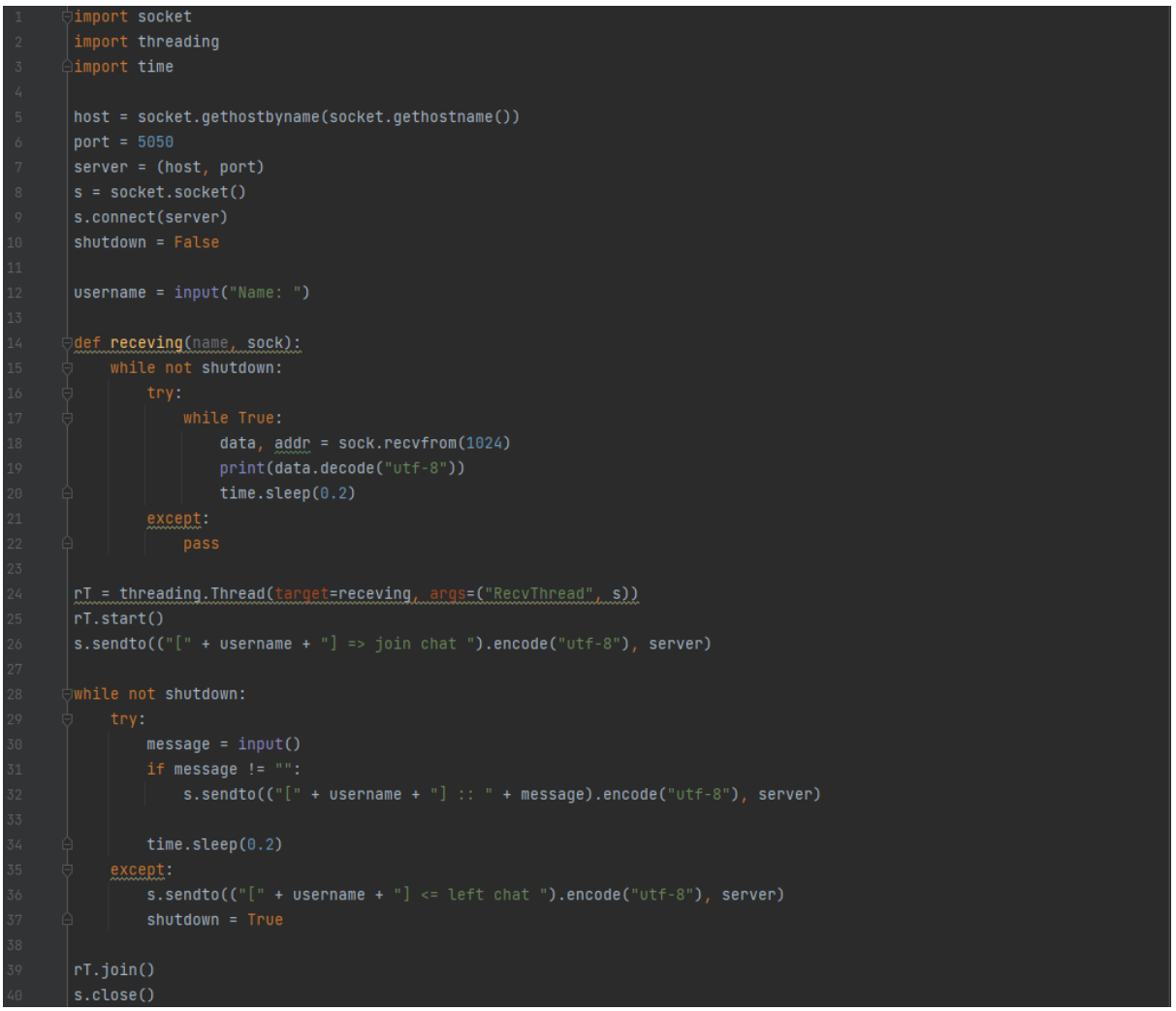
server.py

Задание 4

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

```python
import socket
import threading

host = socket.gethostbyname(socket.gethostname())
port = 5050

server = (host, port)
s = socket.socket()
s.bind(server)
s.listen()
print("[ Server Started ]")

clients = []


def trd(conn):
    while True:
        msg = conn.recv(1024)
        for client, addr in clients:
            client.send(msg)


while True:
    conn, addr = s.accept()
    if addr not in clients:
        clients.append((conn, addr))

    msg = conn.recv(1024)
    print(msg.decode("utf-8"))

    for client, addr in clients:
        client.send(msg)

    threading.Thread(target=trd, args=(conn,)).start()

```

server.py



```python
import socket
import threading
import time

host = socket.gethostbyname(socket.gethostname())
port = 5050
server = (host, port)
s = socket.socket()
s.connect(server)
shutdown = False

username = input("Name: ")

def receving(name, sock):
    while not shutdown:
        try:
            while True:
                data, addr = sock.recvfrom(1024)
                print(data.decode("utf-8"))
                time.sleep(0.2)
        except:
            pass

rT = threading.Thread(target=receving, args=("RecvThread", s))
rT.start()
s.sendto(("[" + username + "] => join chat ").encode("utf-8"), server)

while not shutdown:
    try:
        message = input()
        if message != "":
            s.sendto(("[" + username + "] :: " + message).encode("utf-8"), server)

        time.sleep(0.2)
    except:
        s.sendto(("[" + username + "] <= left chat ").encode("utf-8"), server)
        shutdown = True

rT.join()
s.close()
```

client.py

Вывод

В результате выполненной работы были изучены основы клиент-серверного взаимодействия, работа с сокетами, протокол HTTP, потоки в Python.