Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №1

РАБОТА С СОКЕТАМИ

Выполнил:

Литвак И. Г., гр. К33401

Проверил:

Говоров А. И.

Санкт-Петербург, 2021

Цель работы: овладеть практическими навыками и умениями реализации web-серверов и использования сокетов.

Оборудование: компьютерный класс

Программное обеспечение: Python 3.9, библиотеки Python: sys, socket

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.

Код серверной части:

```python
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(("127.0.0.1", 14900))
conn.listen(10)

sock, address = conn.accept()
data = sock.recv(16384)
data = data.decode("utf-8")
print(data)
msg = "Hello, client!"
sock.send(msg.encode("utf-8"))
conn.close()
```

Код клиентской части:

```python
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.connect(("127.0.0.1", 14900))
msg = "Hello, server!"
conn.send(msg.encode("utf-8"))
data = conn.recv(16384)
print(data.decode("utf-8"))
conn.close()
```

Результат со стороны сервера:

```
Hello, server!

Process finished with exit code 0
```

Результат со стороны клиента:

```
Hello, client!

Process finished with exit code 0
```

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.

Вариант d – площадь параллелограмма

Код серверной части:

```python
# Variant d. - area of parallelogram

import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(("127.0.0.1", 14900))
conn.listen(10)

sock, address = conn.accept()
sock.send("Input length of base of parallelogram:".encode())
b = int(sock.recv(16384).decode())  # Base length
sock.send("Input height of parallelogram:".encode())
h = int(sock.recv(16384).decode())  # Height
area = b * h
sock.send(f"Area of parallelogram is:\n{b} x {h} = {area}".encode())
conn.close()
```

Код клиентской части:

```python
# Variant d. - area of parallelogram

import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.connect(("127.0.0.1", 14900))

# Loop over twice because there are two inputs
for _ in range(2):
    print(conn.recv(16384).decode())  # Message from server
    inp = input(">> ")  # Get base length from user
    conn.send(inp.encode())  # Send it back

print(conn.recv(16384).decode())
conn.close()
```

Результат со стороны клиента:

```
Input length of base of parallelogram:
>> 10
Input height of parallelogram:
>> 5
Area of parallelogram is:
10 x 5 = 50

Process finished with exit code 0
```

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.

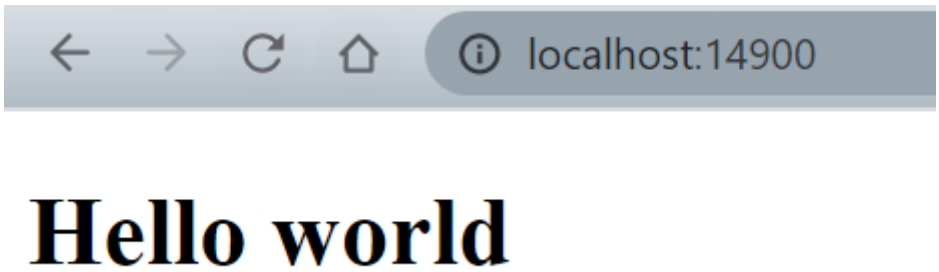
Код серверной части:

```python
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(('127.0.0.1', 14900))
conn.listen(10)


def main():
    while True:
        try:
            client, _ = conn.accept()
            client.recv(4096)
            # Set appropriate headers
            response_type = "HTTP/1.0 200 OK\n"
            headers = "Content-Type: text/html\n\n"
            # Read HTML from file
            with open("index.html", "r") as f:
                body = f.read()
            resp = response_type + headers + body
            client.send(resp.encode())
            client.close()
        except KeyboardInterrupt:
            conn.close()
            break


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

Разметка HTML-страницы:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello</title>
</head>
<body>
    <h1>Hello world</h1>
</body>
</html>
```

Результат:



Задание 4

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

Выбрана реализация **многопользовательского** чата.

Код серверной части:

```python
import socket
import sys
from threading import Thread


class ChatServer:

    def __init__(self, host: str, port: int):
        self.clients = []
        self.host = host
        self.port = port
        self.conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    def _shutdown(self):
        for sock in self.clients:
            sock.close()
        self.conn.close()
        sys.exit(0)

```

```
    def _client_broadcast(self, message: bytes, sender: socket.socket) -> None:
        # Broadcast a message from a client to all the other clients
        for sock in self.clients.copy():
            if sock != sender:
                try:
                    sock.send(message)
                except OSError:
                    # Client disconnected
                    print("Someone disconnected")
                    self.clients.remove(sock)

    def _client_listen(self, sock: socket.socket) -> None:
        # Listen for messages from client
        sock.settimeout(30)
        while True:
            try:
                message = sock.recv(1024)
                print(message.decode())
                self._client_broadcast(message, sock)
            except OSError:
                sock.close()
                break
```

```
    def _main(self) -> None:
        # Run server
        self.conn.bind((self.host, self.port))
        self.conn.listen(10)
        while True:
            try:
                # Accept connection
                sock, address = self.conn.accept()
                print(f"Connection at {address}")
                # Create thread for client
                self.clients.append(sock)
                Thread(target=self._client_listen, args=(sock,)).start()
            except KeyboardInterrupt:
                self._shutdown()

    def run(self) -> None:
        # Wrapper to start thread for _main()
        Thread(target=self._main).start()
```

```

if __name__ == '__main__':
    print("Starting server...")
    server = ChatServer('127.0.0.1', 14900)
    server.run()
    print("Server started")
```

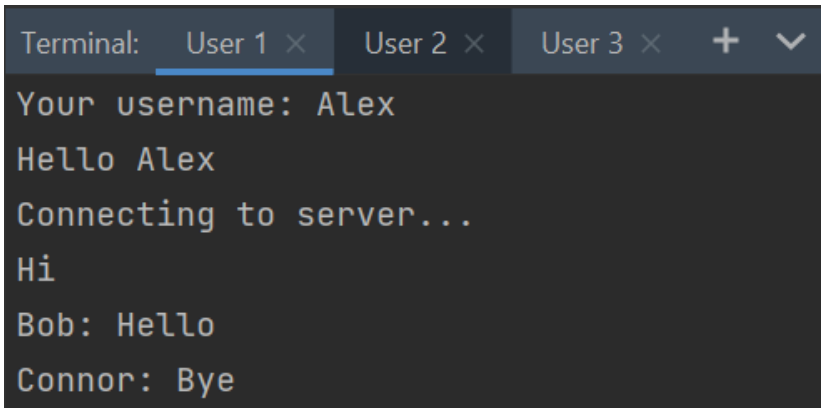
Код клиентской части:

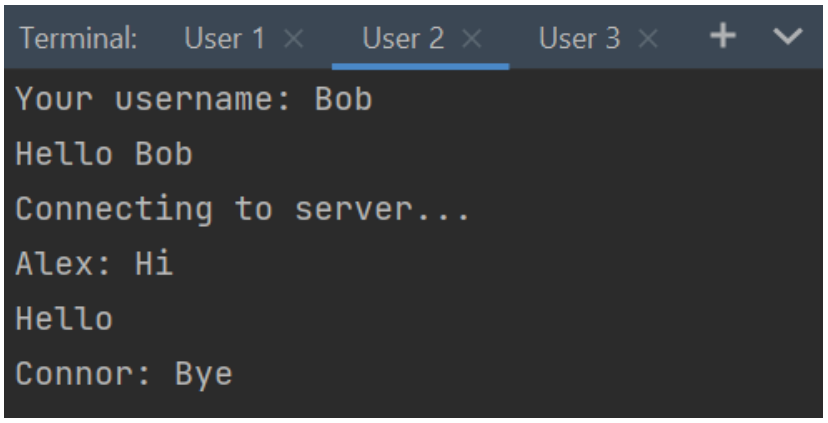
```python
import socket
import sys
from threading import Thread


class ChatClient:
    def __init__(self, host, port, username):
        self.host = host
        self.port = port
        self.username = username
        self.conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    def _send(self):
        # Threaded function for sending messages
        while True:
            try:
                msg = input()
                self.conn.send(f"{self.username}: {msg}".encode())
            except (KeyboardInterrupt, EOFError):
                self.conn.close()
                sys.exit(0)

```

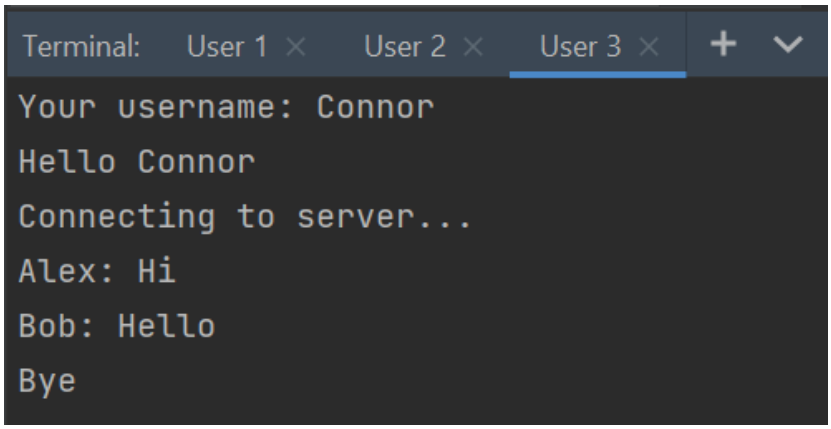
```python
    def _recieve(self):
        # Threaded function for recieving messages
        while True:
            try:
                print(self.conn.recv(1024).decode())
            except KeyboardInterrupt:
                self.conn.close()
                sys.exit(0)
            except ConnectionError:
                # Unexpected connection error
                print("Connection error")
                self.conn.close()
                sys.exit(1)

    def run(self):
        # Connect
        self.conn.connect((self.host, self.port))
        # Run threaded functions
        Thread(target=self._send).start()
        Thread(target=self._recieve).start()

```

```
44
45  if __name__ == '__main__':
46      u = input("Your username: ")
47      print(f"Hello {u}")
48      print("Connecting to server...")
49      client = ChatClient('127.0.0.1', 14900, u)
50      client.run()
51
```

Результат:







Вывод: на примере библиотеки socket языка Python я изучил устройство web-серверов и на практике реализовал их простые примеры.