

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ  
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

Факультет «Инфокоммуникационных технологий»  
Направление подготовки «45.03.04 Интеллектуальные системы в гуманитарной  
сфере»

**О Т Ч Е Т**

**Web-программирование**

**Лабораторная работа №1: Работа с сокетами**

Выполнил:

Студент Грицай А. И. K33421  
Фамилия И.О. номер группы

Проверил:

Преподаватель Говоров А. И.  
Фамилия И.О.

Санкт-Петербург  
2020

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

## Ход работы

### Задание 1.

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

```
import socket
import time

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(("127.0.0.1", 14900))
conn.listen(1)
conn.setblocking(False)

while True:
    try:
        client_s, address = conn.accept()
        print(f"Connectrd to: {address}")
        conn.setblocking(True)
        data = client_s.recv(16384)
        data = data.decode("utf-8")
        print(("Data recieved: " + data))
        server_message = "Hello, client!"
        client_s.send(bytes(server_message, "utf-8"))
        conn.close()
    except socket.error:
        time.sleep(5)
    except KeyboardInterrupt:
        conn.close()
        break
```

Task\_1\_server

```

import socket

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

client_message = "Hello, server!"
conn.send(bytes(client_message, "utf-8"))
data_recv = conn.recv(1024)
data_recv = data_recv.decode('utf-8')
print(("Data recieved: " + data_recv))

conn.close()

```

Task\_1\_client

Выполнение:

```

Connectrd to: ('127.0.0.1', 55404)
Data recieved: Hello, server!

```

```

Data recieved: 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 1: Теорема Пифагора.

```

import socket
import math

socket_serv = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket_serv.bind(("127.0.0.1", 14900))
socket_serv.listen(1)
connection, address = socket_serv.accept()
print(f"Connected to: {address}")

while True:
    data = connection.recv(16384)
    data = str(data.decode())
    print(data)
    if not data == "Pythagoras: find hypotenuse":
        connection.send("Smtg wrong".encode("utf-8"))
        break
    else:
        connection.send("Input cathetus of triangle: a b".encode("utf-8"))
        data = connection.recv(16384)
        data = str(data.decode())

        try:
            a, b = map(int, data.split(' '))
        except:
            connection.send("Input error".encode("utf-8"))

        hypotenuse = math.sqrt(a ** 2 + b ** 2)
        hypotenuse = "Hypotenuse length = " + str(hypotenuse)
        connection.send(hypotenuse.encode())
        socket_serv.close()
        break

```

Task\_2\_server

```

import socket

connection = socket.socket()
connection.connect(("127.0.0.1", 14900))

task = "Pythagoras: find hypotenuse"
connection.send(task.encode("utf-8"))
data = connection.recv(16384)
print(data.decode())
cathetus = input()
connection.send(cathetus.encode())
data = connection.recv(16384)
connection.close()
print(data.decode())

```

Task\_2\_client

Выполнение:

```

Connected to: ('127.0.0.1', 56071)
Pythagoras: find hypotenuse

```

```
Input cathetus of triangle: a b
5 12
Hypotenuse length = 13.0
```

### 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.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hi, there!</title>
</head>
<body>
  <H1>WebWebWeb</H1>
</body>
</html>
```

Task 3 index

```
import socket

connection = socket.socket()
host = "127.0.0.1"
port = 14900
connection.bind((host, port))
print("Starting server on: ", host, port)
connection.listen(10)

exit = False

while not exit:
    try:
        client, (client_host, client_port) = connection.accept()
        print("Connected to", client_host, client_port)
        response_type = "HTTP/1.0 200 OK\n"
        headers = 'Connect-Type: text/html\n\n'
        page = open('index.html', 'r')
        body = ''.join(page)
        response = response_type + headers + body
        client.send(response.encode('utf-8'))
        client.close()
    except KeyboardInterrupt:
        exit = True
        print("Stop")

connection.close()
```

Task\_3\_server

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(("127.0.0.1", 14900))
msg = sock.recv(16384)
print(msg.decode("utf-8"))
sock.close()
```

Task\_3\_client

Выполнение:

```
Starting server on: 127.0.0.1 14900
Connected to 127.0.0.1 56296
```

```
HTTP/1.0 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hi, there!</title>
</head>
<body>
<H1>WebWebWeb</H1>
</body>
</html>

Process finished with exit code 0
```

```
Starting server on: 127.0.0.1 14900
Connected to 127.0.0.1 56312
Stop

Process finished with exit code 0
```

#### Задание 4.

Реализовать двухпользовательский или многопользовательский чат.

#### Реализация многопользовательского чата

```

import threading
import socket

host = '127.0.0.1'
port = 14902

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((host, port))
server.listen(10)
print("Starting server on: ", host, port)

clients = []
nicknames = []

# посылает сообщение всем подключенным участникам чата
def broadcast(message):
    for client in clients:
        client.send(message)

# получает сообщения клиентов
def handle(client):
    while True:
        try:
            message = client.recv(16384)
            broadcast(message)
        except:
            # если возникают проблемы с получением сообщения
            # отключаем и удаляем клиента из чата
            index = clients.index(client)
            clients.remove(client)
            client.close()
            nickname = nicknames[index]
            broadcast('{} left!'.format(nickname).encode('utf-8'))
            nicknames.remove(nickname)
            break

```

```

# получаем сообщения от участников чата
def receive():
    while True:
        client, address = server.accept()
        print("Connected with {}".format(str(address)))
        # запрашиваем и записываем ники участников
        client.send('NICK'.encode('utf-8'))
        nickname = client.recv(16384).decode('utf-8')
        nicknames.append(nickname)
        clients.append(client)
        # транслируем новое подключение и ник участника
        print("Nickname is {}".format(nickname))
        broadcast("{} joined!".format(nickname).encode('utf-8'))
        client.send('Connected to server!'.encode('utf-8'))
        # начинаем обработку потока для участника чата
        thread = threading.Thread(target=handle, args=(client,))
        thread.start()

print("Starting server...")
receive()

```

Task\_4\_server

```

import socket
import threading

nickname = input("Choose your nickname: ")

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 14902))

# подключаемся к серверу и посылаем ник
def receive():
    while True:
        try:
            message = client.recv(16384).decode('utf-8')
            if message == 'NICK':
                client.send(nickname.encode('utf-8'))
            else:
                print(message)
        except:
            print("Smth wrong!")
            client.close()
            break

# отправка сообщений на сервер
def write():
    while True:
        message = '{}: {}'.format(nickname, input(''))
        client.send(message.encode('utf-8'))

# запуск потоков для получения сообщений
receive_thread = threading.Thread(target=receive)
receive_thread.start()

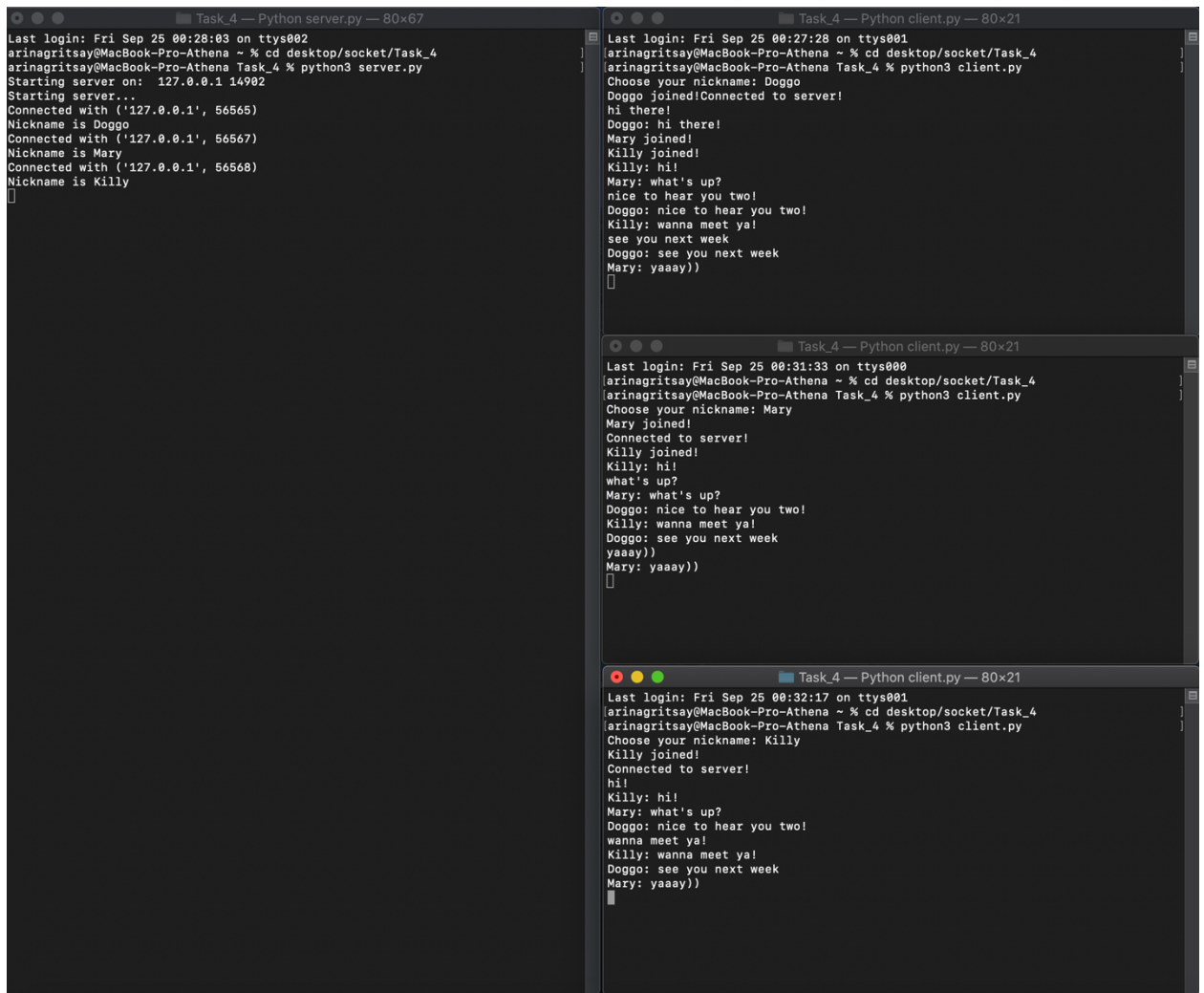
# запуск потоков для отправки сообщений
write_thread = threading.Thread(target=write)
write_thread.start()

```

Task\_4\_client



Выполнение: слева - запуск сервера, справа - три клиента



```
Task_4 — Python server.py — 80x67
Last login: Fri Sep 25 00:28:03 on ttys002
arinagritsay@MacBook-Pro-Athena ~ % cd desktop/socket/Task_4
arinagritsay@MacBook-Pro-Athena Task_4 % python3 server.py
Starting server...
Connected with ('127.0.0.1', 56565)
Nickname is Doggo
Connected with ('127.0.0.1', 56567)
Nickname is Mary
Connected with ('127.0.0.1', 56568)
Nickname is Killy
█

Task_4 — Python client.py — 80x21
Last login: Fri Sep 25 00:27:28 on ttys001
arinagritsay@MacBook-Pro-Athena ~ % cd desktop/socket/Task_4
arinagritsay@MacBook-Pro-Athena Task_4 % python3 client.py
Choose your nickname: Doggo
Doggo joined!Connected to server!
hi there!
Doggo: hi there!
Mary joined!
Killy joined!
Killy: hi!
Mary: what's up?
nice to hear you two!
Doggo: nice to hear you two!
Killy: wanna meet ya!
see you next week
Doggo: see you next week
Mary: yaaay))
█

Task_4 — Python client.py — 80x21
Last login: Fri Sep 25 00:31:33 on ttys000
arinagritsay@MacBook-Pro-Athena ~ % cd desktop/socket/Task_4
arinagritsay@MacBook-Pro-Athena Task_4 % python3 client.py
Choose your nickname: Mary
Mary joined!
Connected to server!
Killy joined!
Killy: hi!
what's up?
Mary: what's up?
Doggo: nice to hear you two!
Killy: wanna meet ya!
Doggo: see you next week
yaaay))
Mary: yaaay))
█

Task_4 — Python client.py — 80x21
Last login: Fri Sep 25 00:32:17 on ttys001
arinagritsay@MacBook-Pro-Athena ~ % cd desktop/socket/Task_4
arinagritsay@MacBook-Pro-Athena Task_4 % python3 client.py
Choose your nickname: Killy
Killy joined!
Connected to server!
hi!
Killy: hi!
Mary: what's up?
Doggo: nice to hear you two!
wanna meet ya!
Killy: wanna meet ya!
Doggo: see you next week
Mary: yaaay))
█
```

**Вывод:** в результате выполнения лабораторной работы получены практические навыки работы с web-серверами, использования сокетов и потоков.