

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1 по
теме: Работа с сокетами
по дисциплине: Web-программирование

Специальность:
09.03.03 Мобильные и сетевые технологии

Выполнил:
студент группы К33402
Борисов М.Е

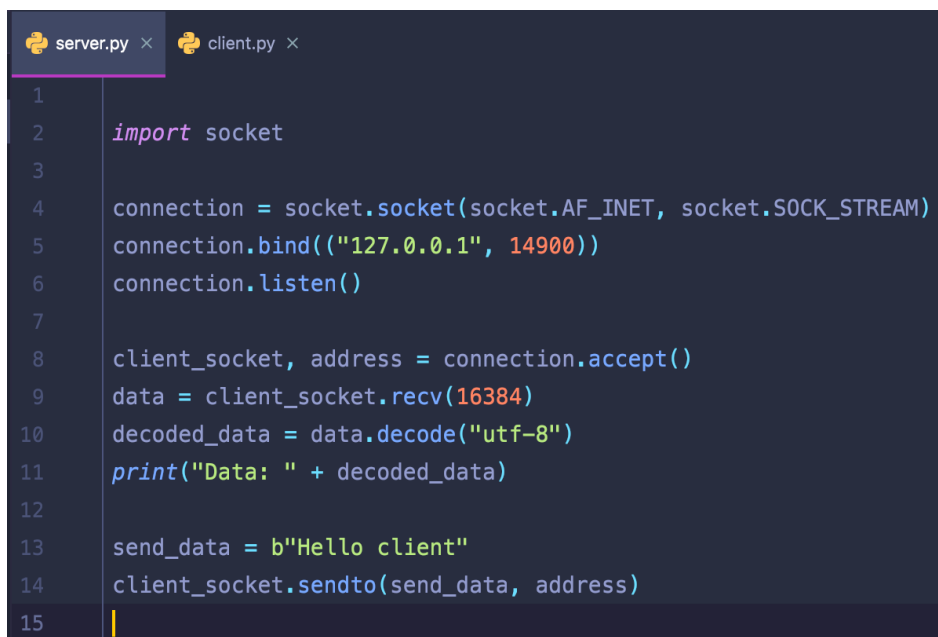
Санкт-Петербург 2021/2022

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

Практическое задание

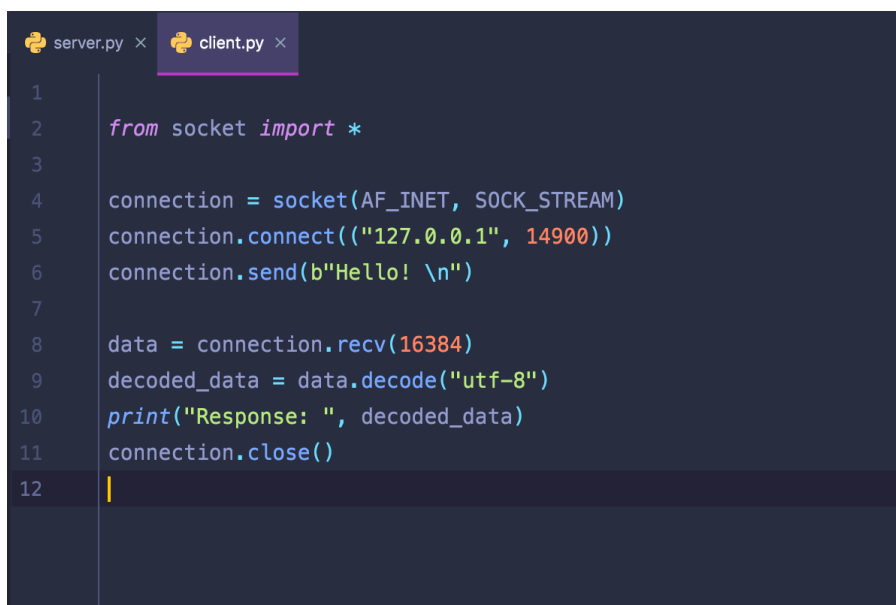
1. Реализовать клиентскую и серверную часть приложения. Клиент отправляет серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отправляет клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.

Код сервера:



```
server.py x client.py x
1
2 import socket
3
4 connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 connection.bind(("127.0.0.1", 14900))
6 connection.listen()
7
8 client_socket, address = connection.accept()
9 data = client_socket.recv(16384)
10 decoded_data = data.decode("utf-8")
11 print("Data: " + decoded_data)
12
13 send_data = b"Hello client"
14 client_socket.sendto(send_data, address)
15
```

Код клиента:



```
server.py x client.py x
1
2 from socket import *
3
4 connection = socket(AF_INET, SOCK_STREAM)
5 connection.connect(("127.0.0.1", 14900))
6 connection.send(b"Hello! \n")
7
8 data = connection.recv(16384)
9 decoded_data = data.decode("utf-8")
10 print("Response: ", decoded_data)
11 connection.close()
12
```

Код выполнения на сервере:

```
(PyEnv) 0300nbc00106439:pythonProject myborisov$ python3 server.py
^C(PyEnv) 0300nbc00106439:pythonProject myborisov$ python3 server.py
Data: Hello!

(PyEnv) 0300nbc00106439:pythonProject myborisov$
```

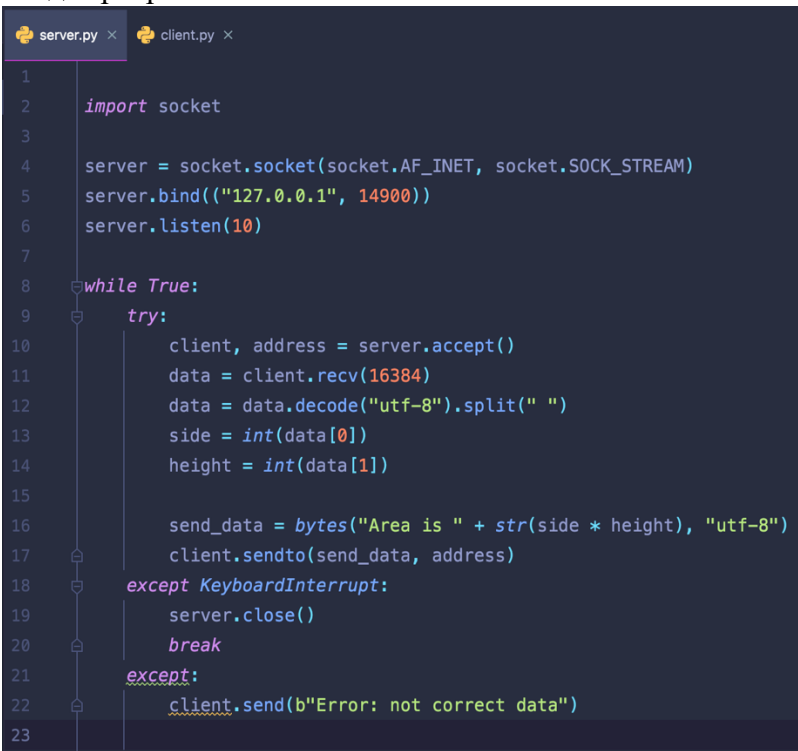
Код выполнения на клиенте:

```
(PyEnv) 0300nbc00106439:pythonProject myborisov$ python3 client.py
Response: Hello client

(PyEnv) 0300nbc00106439:pythonProject myborisov$
```

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. Поиск площади параллелограмма.

Код сервера:



```
server.py x client.py x
1
2 import socket
3
4 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 server.bind(("127.0.0.1", 14900))
6 server.listen(10)
7
8 while True:
9     try:
10         client, address = server.accept()
11         data = client.recv(16384)
12         data = data.decode("utf-8").split(" ")
13         side = int(data[0])
14         height = int(data[1])
15
16         send_data = bytes("Area is " + str(side * height), "utf-8")
17         client.sendto(send_data, address)
18     except KeyboardInterrupt:
19         server.close()
20         break
21     except:
22         client.send(b"Error: not correct data")
23
```

Код клиента:

```
server.py x client.py x
1  from socket import *
2
3  while True:
4      try:
5          side = int(input("Enter parallelogram side: "))
6          height = int(input("Enter parallelogram height: "))
7          break
8      except:
9          print("Enter correct data\n")
10
11  connection = socket(AF_INET, SOCK_STREAM)
12  connection.connect(("127.0.0.1", 14900))
13  data = str(side) + " " + str(height)
14  connection.send(bytes(data, "utf-8"))
15
16  data = connection.recv(16384)
17  decoded_data = data.decode("utf-8")
18  print("Response: ", decoded_data)
19  connection.close()
20
```

Выполнение:

```
(PyEnv) 0300nbc00106439:pythonProject myborisov$ python3 client.py
Enter parallelogram side: 3
Enter parallelogram height: 10
Response: Area is 30
```

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 сервера:

```
server.py x
1  import json
2      from socket import *
3      from email.parser import Parser
4      from functools import lru_cache
5      from urllib.parse import parse_qs, urlparse
6
7      MAX_LINE = 64 * 1024
8      MAX_HEADERS = 100
9
10
11  class MyHTTPServer:
12      def __init__(self, host, port, server_name):
13          self._host = host
14          self._port = port
15          self._server_name = server_name
16          self._lessons = {}
17
18      def serve_forever(self):
19          server = socket(AF_INET, SOCK_STREAM)
20
21          try:
22              server.bind((self._host, self._port))
23              server.listen()
24
25              while True:
26                  connection, _ = server.accept()
27                  try:
28                      self.serve_client(connection)
29                  except Exception as e:
30                      print("Client serving failed", e)
31          finally:
32              server.close()
33
34      def serve_client(self, connection):
35          try:
36              request = self.parse_request(connection)
37              response = self.handle_request(request)
38              self.send_response(connection, response)
```

```
server.py x
39         except ConnectionResetError:
40             connection = None
41         except Exception as e:
42             self.send_error(connection, e)
43
44     def parse_request(self, connection):
45         rfile = connection.makefile('rb')
46         method, target, version = self.parse_request_line(rfile)
47         headers = self.parse_headers(rfile)
48         host = headers.get('Host')
49         if not host:
50             raise HTTPError(400, 'Bad request', 'Host header is missing')
51
52         if host not in (self._server_name, f'{self._server_name}:{self._port}'):
53             raise HTTPError(404, 'Not found')
54
55         return Request(method, target, version, headers, rfile)
56
57     def parse_request_line(self, rfile):
58         raw = rfile.readline(MAX_LINE + 1)
59         if len(raw) > MAX_LINE:
60             raise HTTPError(400, 'Bad request', 'Request line is too long')
61
62         request_line = str(raw, 'iso-8859-1')
63         words = request_line.split()
64         if len(words) != 3:
65             raise HTTPError(400, 'Bad request', 'Malformed request line')
66
67         method, target, ver = words
68
69         if ver != 'HTTP/1.1':
70             raise HTTPError(505, 'HTTP Version Not Supported')
71         return method, target, ver
72
73     def parse_headers(self, rfile):
74         headers = []
75         while True:
76             line = rfile.readline(MAX_LINE + 1)
```

```
server.py x
77         if len(line) > MAX_LINE:
78             raise HTTPError(494, 'Request header too large')
79
80         if line in (b'\r\n', b'\n', b''):
81             break
82
83         headers.append(line)
84         if len(headers) > MAX_HEADERS:
85             raise HTTPError(494, 'Too many headers')
86
87         decoded_headers = b''.join(headers).decode('iso-8859-1')
88         return Parser().parsestr(decoded_headers)
89
90     def handle_request(self, request):
91         if request.path == '/lessons' and request.method == 'POST':
92             return self.handle_post_lessons(request)
93
94         if request.path == '/lessons' and request.method == 'GET':
95             return self.handle_get_lessons(request)
96
97         raise HTTPError(404, 'Not found')
98
99     def handle_post_lessons(self, request):
100         self._lessons[request.query['name'][0]] = request.query['mark'][0]
101
102         return Response(204, 'Updated')
103
104     def handle_get_lessons(self, request):
105         accept = request.headers.get('Accept')
106         if 'text/html' in accept:
107             contentType = 'text/html; charset=utf-8'
108             body = '<html><head></head><body>'
109             body += f'<div>Предметы ({len(self._lessons)})</div>'
110             body += '<ul>'
111             for lessons_name in self._lessons.keys():
112                 body += f'<li>{lessons_name}: {self._lessons[lessons_name]}</li>'
113             body += '</ul>'
114             body += '</body></html>'
```

```
server.py x
115
116     elif 'application/json' in accept:
117         contentType = 'application/json; charset=utf-8'
118         body = json.dumps(self._lessons)
119
120     else:
121         # https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/406
122         return Response(406, 'Not Acceptable')
123
124     body = body.encode('utf-8')
125     headers = [('Content-Type', contentType), ('Content-Length', len(body))]
126
127     return Response(200, 'OK', headers, body)
128
129     def send_response(self, connection, response):
130         wfile = connection.makefile('wb')
131         status_line = f'HTTP/1.1 {response.status} {response.reason}\r\n'
132         wfile.write(status_line.encode('iso-8859-1'))
133
134         if response.headers:
135             for (key, value) in response.headers:
136                 header_line = f'{key}: {value}\r\n'
137                 wfile.write(header_line.encode('iso-8859-1'))
138
139         wfile.write(b'\r\n')
140
141         if response.body:
142             wfile.write(response.body)
143
144         wfile.flush()
145         wfile.close()
146
147     def send_error(self, connection, error):
148         try:
149             status = error.status
150             reason = error.reason
151             body = (error.body or error.reason).encode('utf-8')
152         except:
```



```

server.py x
153         status = 500
154         reason = b'Internal Server Error'
155         body = b'Internal Server Error'
156         resp = Response(status, reason, [('Content-Length', len(body))], body)
157         self.send_response(connection, resp)
158
159
160     class Request:
161     def __init__(self, method, target, version, headers, rfile):
162         self.method = method
163         self.target = target
164         self.version = version
165         self.headers = headers
166         self.rfile = rfile
167
168         @property
169         def path(self):
170         return self.url.path
171
172         @property
173         @lru_cache(maxsize=None)
174         def query(self):
175         return parse_qs(self.url.query)
176
177         @property
178         @lru_cache(maxsize=None)
179         def url(self):
180         return urlparse(self.target)
181
182         def body(self):
183             size = self.headers.get('Content-Length')
184             if not size:
185                 return None
186             return self.rfile.read(size)
187
188
189     class Response:
190     def __init__(self, status, reason, headers=None, body=None):

```

```

server.py x
182 def body(self):
183     size = self.headers.get('Content-Length')
184     if not size:
185         return None
186     return self.rfile.read(size)
187
188
189 class Response:
190     def __init__(self, status, reason, headers=None, body=None):
191         self.status = status
192         self.reason = reason
193         self.headers = headers
194         self.body = body
195
196
197 class HTTPError(Exception):
198     def __init__(self, status, reason, body=None):
199         super()
200         self.status = status
201         self.reason = reason
202         self.body = body
203
204
205 if __name__ == '__main__':
206     host = "127.0.0.1"
207     port = 14900
208     name = "server.local"
209     serv = MyHTTPServer(host, port, name)
210     try:
211         serv.serve_forever()
212     except KeyboardInterrupt:
213         pass
214

```

POST запрос:

POST

http://localhost:14900/lessons?name=math&mark=50

Send

Params

Authorization

Headers (11)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> name	math			
<input checked="" type="checkbox"/> mark	50			
Key	Value	Description		

Body

Cookies

Headers

Test Results

Status: 204 Updated

Time: 17 ms

Size: 24 B

Save Response

GET запрос:

GET <http://localhost:14900/lessons?Content-Type=application/json ...> Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

Headers 6 hidden

KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/> Host	server.local				
<input checked="" type="checkbox"/> Accept	text/html				
<input checked="" type="checkbox"/> User-Agent	Mozilla/5.0				
<input checked="" type="checkbox"/> Content-Type	application/json				
Key	Value	Description			

Body Cookies Headers (2) Test Results Status: 200 OK Time: 5 ms Size: 175 B Save Response

Pretty Raw Preview Visualize

Предметы (1)

- math: 50

4. Реализовать двухпользовательский или многопользовательский чат.

Код сервера:

```
server.py x client.py x
1 from socket import *
2
3
4 def run():
5     clients = []
6     while True:
7         try:
8             data, client_address = connection.recvfrom(16384)
9             if client_address not in clients:
10                 clients.append(client_address)
11             for client in clients:
12                 if client == client_address:
13                     continue
14                 connection.sendto(data, client)
15             except KeyboardInterrupt:
16                 connection.close()
17                 break
18             except:
19                 continue
20
21
22 connection = socket(AF_INET, SOCK_DGRAM)
23 connection.bind(("127.0.0.1", 14900))
24 run()
25
```

Код клиента:

```
server.py x client.py x
1  from socket import *
2  import threading
3
4
5  def read():
6      while True:
7          data = connection.recv(16384)
8          print(data.decode("utf-8"))
9
10
11 def chat():
12     name = input("Enter user name: ")
13     connection.sendto((name + " connected").encode("utf-8"), server_address)
14     while True:
15         connection.sendto((name + ": " + input()).encode("utf-8"), server_address)
16
17
18 server_address = "127.0.0.1", 14900
19 connection = socket(AF_INET, SOCK_DGRAM)
20 thread1, thread2 = threading.Thread(target=read), threading.Thread(target=chat)
21 thread1.start(), thread2.start()
22
```

Запускаем сервер:

```
^C(PyEnv) 0300nbc00106439:pythonProject myborisov$ python3 server.py
```

Добавляем первого пользователя:

```
(PyEnv) 0300nbc00106439:pythonProject myborisov$ python3 client.py
Enter user name: Foo
Bar connected
Hello, Bar!
Bar: Hello, guys :)
Bar: How are you?
Fine, I've done the first web lab
Bar: Cool
Tom connected
Tom: Guys, have you seen Jerry?
Bar: No
no
```

Добавляем второго пользователя:

```
(PyEnv) 0300nbc00106439:pythonProject myborisov$ python3 client.py
Enter user name: Bar
Foo: Hello, Bar!
Hello, guys :)
How are you?
Foo: Fine, I've done the first web lab
Cool
Tom connected
Tom: Guys, have you seen Jerry?
No
Foo: no
```

Добавляем третьего пользователя:

```
(PyEnv) 0300nbc00106439:pythonProject myborisov$ python3 client.py
Enter user name: Tom
Guys, have you seen Jerry?
Bar: No
Foo: no
█
```