Лекция 8

Сетевое программирование

1 Сокеты

***Сокет* (socket)** — название программного интерфейса для обеспечения обмена данными между процессами

TCP (Transmission Control Protocol, протокол управления передачей) – транспортный протокол передачи данных в сетях TCP/IP, предварительно устанавливающий соединение с сетью.

UDP (User Datagram Protocol — протокол пользовательских датаграмм) – транспортный протокол, передающий данные без установки соединения в IP-сети.

Создание сокетов:

```
import socket
s = socket.socket (socket_family, socket_type, …)
```

socket_family – семейство адресов с которыми может взаимодействовать сокет; значения AF_UNIX, AF_INET (по умолчанию)

socket_type – тип связи между двумя конечными точками; значения SOCK_STREAM (для протоколов, ориентированных на соединение), SOCK_DGRAM (для протоколов без установления соединения)

Способы подключения сервера

№	Способ и описание
1	**s.bind()** метод связывает адрес (имя хоста, порт) с сокетом **Пример:** s = socket.socket() s.bind(('localhost', 3333))
2	**s.listen()** метод устанавливает и запускает прослушиватель **Пример:** s.listen(5)
3	**s.accept()** пассивно принимает клиентское соединение, ожидая, пока не придет соединение (блокировка) **Пример:** client, addr = s.accept()

Для клиента:

s.connect() - метод активно инициирует подключение к серверу TCP.

Общие методы сокета

№	Способ и описание
1	**s.recv()** метод получает сообщение (TCP)
2	**s.send()** метод передает сообщение (TCP)
3	**s.recvfrom()** метод получает сообщение (UDP)
4	**s.sendto()** метод передает сообщение (UDP)
5	**s.close()** метод закрывает сокет
6	**s.gethostname()** возвращает имя хоста

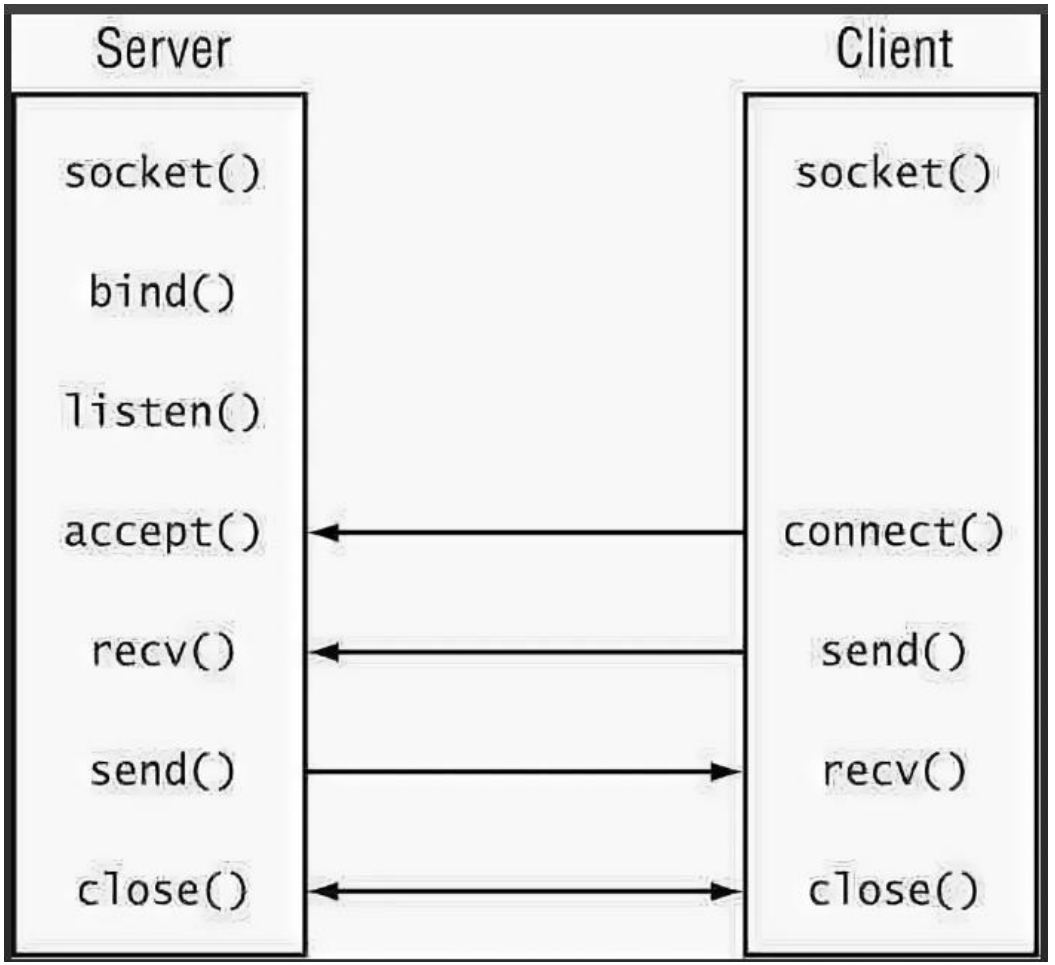
Алгоритм работы сервера и клиента

На стороне сервера:

1. Подключить библиотеку.
2. Создать сокет (socket).
3. Связать сокет (bind).
4. Перевести его в слушающий режим (listen).
5. Принять запрос на соединение (accept).
6. Получить (recv) или отправить (send) данные.
7. Закрыть соединение (close).

На стороне клиента:

1. Подключить библиотеку.
2. Создать сокет (socket).
3. Послать запрос на соединение (connect).
4. Отправить (send) или получить (recv) данные.
5. Закрыть сокет (close).



Клиент посылает серверу строку, сервер получает строку, преобразует буквы строки к верхнему регистру и посылает строку клиенту

СЕРВЕР

```
import socket

sock = socket.socket()
sock.bind(('', 9090))
sock.listen(1)
conn, addr = sock.accept()

print('connected:', addr)

while True:
    data = conn.recv(1024)
    if not data:
        break
    conn.send(data.upper())

conn.close()
```

```
connected: ('127.0.0.1', 5723)
>>>
```

КЛИЕНТ

```
import socket

sock = socket.socket()
sock.connect(('localhost', 9090))
sock.send('hello, world!'.encode('utf-8'))

data = sock.recv(1024)
sock.close()

print(data)
```

```
b'HELLO, WORLD!'
>>>
```

СЕРВЕР

```
import socket

s = socket.socket()

s.bind(('localhost', 3333))
s.listen(1)

while True:
    connect, addr = s.accept()
    print("Connection Address:" + str(addr))

    str_return = "What's your name?"
    connect.send(bytes(str_return, 'utf-8'))

    str_recv = connect.recv(1024)
    print('Client: '+str(str_recv))

    str_return = "Hello, " + str(str_recv)
    connect.send(bytes(str_return, 'utf-8'))

    connect.close()
```

```
Connection Address:('127.0.0.1', 4333)
b'Client: Alex'
Connection Address:('127.0.0.1', 4336)
b'Client: Bob'
Connection Address:('127.0.0.1', 4339)
b'Client: Anna'
```

КЛИЕНТ

```
import socket

s = socket.socket()

s.connect(('localhost', 3333))

str_recv = s.recv(1024)
print(str(str_recv))

str_send = input()
s.send(bytes(str_send, 'utf-8'))
str_recv = s.recv(1024)
print(str(str_recv))
s.close()
```

```
b"What's your name?"
Alex
b"Hello, b'Alex'"
>>>
```

```
b"What's your name?"
Bob
b"Hello, b'Bob'"
```

```
b"What's your name?"
Anna
b"Hello, b'Anna'"
```

СЕРВЕР

```python
import socket

s = socket.socket()

s.bind(('localhost', 3333))
s.listen(5)

# количество сообщений
num = 1

connect, addr = s.accept()
print("Connection Address:" + str(addr))

while True:

    str_return = "Message "+str(num)+":"
    connect.send(bytes(str_return, 'utf-8'))

    str_recv, temp = connect.recv(1024)
    print(b'Message received: '+str_recv)

    if (str_recv == b'exit'): break
    num+=1

    str_return = "Your messaget: " + str(str_recv).upper()
    connect.send(bytes(str_return, 'utf-8'))

connect.close()
```

```
Connection Address:('127.0.0.1', 1768)
b'Message received: text1 TEXT2'
b'Message received: Python PyThOn'
b'Message received: Text3 Text4'
b'Message received: exit'
>>>
```

КЛИЕНТ

```python
import socket

s = socket.socket()
s.connect(("localhost", 3333))

while True:
    str_recv = s.recv(1024)
    print(str(str_recv))

    str_send = input()
    s.send(bytes(str_send, 'utf-8'))

    if str_send == 'exit': break

    str_recv = s.recv(1024)
    print(str(str_recv))

s.close()
```

```
b'Message 1:'
text1 TEXT2
b"Your messaget: B'TEXT1 TEXT2'"
b'Message 2:'
Python PyThOn
b"Your messaget: B'PYTHON PYTHON'"
b'Message 3:'
Text3 Text4
b"Your messaget: B'TEXT3 TEXT4'"
b'Message 4:'
exit
>>>
```

СЕРВЕР

```
import select # для работы с несколькими клиентами
import socket

server = socket.socket()

server.bind(('localhost', 3333))
server.listen(5)

# список клиентов
cl = []

while True:
            #добавляем клиента
            client, addr = server.accept()
            cl.append(client)
            print("New client: " + str(addr))

            str_return = "CONNECT"
            client.send(bytes(str_return, 'utf-8'))

            # получаем сообщение от нового клиента
            data = client.recv(1024)
            print(b'Message received: ' + data)

            #ищем клиентов, ожидающих сообщение
            inp ,outp, exc = select.select([],cl,[])
            print('Num of clients (outp) : '+ str(len(outp)))

            str_return = "Message from client "+str(addr)+": "\
                         + str(data).upper()

            # отправляем клиентам сообщение
            for o in outp:
                o.send(bytes(str_return, 'utf-8'))
```

КЛИЕНТЫ

```
import socket

s = socket.socket()
s.connect(("localhost", 3333))

str_recv = s.recv(1024)
print(str(str_recv))

print('Сообщение для отправки клиентам: ')
str_send = input()
s.send(bytes(str_send, 'utf-8'))

while True:
    str_recv = s.recv(1024)
    print(str(str_recv))

s.close()
```

```
New client: ('127.0.0.1', 5647)
b'Message received: client 1'
Num of clients (outp) : 1
New client: ('127.0.0.1', 5652)
b'Message received: client 2'
Num of clients (outp) : 2
New client: ('127.0.0.1', 5657)
b'Message received: client 3'
Num of clients (outp) : 3
```

```
b'CONNECT'
Сообщение для отправки клиентам:
client 1
b"Message from client ('127.0.0.1', 5647): B'CLIENT 1'"
b"Message from client ('127.0.0.1', 5652): B'CLIENT 2'"
b"Message from client ('127.0.0.1', 5657): B'CLIENT 3'"
```

```
b'CONNECT'
Сообщение для отправки клиентам:
client 2
b"Message from client ('127.0.0.1', 5652): B'CLIENT 2'"
b"Message from client ('127.0.0.1', 5657): B'CLIENT 3'"
```

```
b'CONNECT'
Сообщение для отправки клиентам:
client 3
b"Message from client ('127.0.0.1', 5657): B'CLIENT 3'"
```

Модули для сетевого программирования

соглашение	Назначение	Модули Python
HTTP	веб-доступ (передача гипертекстовых документов со ссылками перехода к другим документам)	HTTPLIB, URLLIB, XMLRPC LIB
NNTP	распространение, запрашивание, размещение новостей	nntplib
FTP	передача файлов	ftplib, URLLIB
SMTP	отправка по электронной почте	smtplib
POP3	получение сообщений эл. почты	poplib
IMAP4	работа с эл. почтой	imaplib