ШКОЛА СП «Вьетсовпетро»

**ПРИКЛАДНОЙ ПРОЕКТ**

По направлению «Информатика»

«Возможности работы с сетью на языке программирования Python»

Выполнил: ученик 11 класса

Воробьев Максим

Руководитель: преподаватель информатики и физики

Перепелицын С.А.

Вунгтау   
2021

**Оглавление**

**Введение…………………………………………………………………….…….3**

**Основная часть………………………………………………….……………….5**

**Заключение……………………………………………...………………………16**

**Список литературы……………………………………...……………………..17**

**Приложение……………………………………...……………………..……….17**

**Введение**

Компьютерная сеть - одно из самых важных и востребованных на данный момент изобретений человечества. Каждый из нас использует ее ежедневно, заходя в интернет. По своей сути, любая компьютерная сеть представляет собой систему, обеспечивающая связь и обмен данными между компьютерами, серверами, маршрутизаторами и др.

Главным достоинством таких сетей является возможность передачи данных на расстоянии с одного устройства на другое. Данное изобретение предоставило человечеству широкие возможности для дальнейшего развития. Вследствие этого появился интернет, который является всемирной системой объединенных компьютерных сетей.

Python - один из наиболее популярных языков программирования. Он используется во многих сферах, от написания веб-приложений, до создания сложных математических моделей для анализа больших данных. Его стандартная библиотека включает множество модулей, один из которых предоставляет возможность работы с сетями на прикладном уровне.

**Актуальность:** тема работы с компьютерными сетями является актуальной, так как в наши дни каждый человек так или иначе ежедневно сталкивается с ними

**Цель проекта:** создать программу на языке программирования Python, предоставляющую возможность обмена сообщениями между пользователями.

**Задачи:**

- собрать материалы о компьютерных сетях;

- изучить принцип работы компьютерных сетей;

- создать программу на основе полученной информации.

**Методы исследования:** теоретический анализ литературы, материалов из сети Интернет, создание схемы работы.

**Практическая значимость:** мой проект может использоваться для демонстрации работы компьютерных сетей на основе обычной передачи текстовых сообщений. Такой уровень подачи обеспечит доступность материала для каждого человека.

**Компьютерные сети. История создания**

Первым применением компьютерной (вычислительной) сети считается использование в советском комплексе ПРО «Система А». Он был построен в 1956-1960 годах в Казахстане. Однако такая система была предложена и ранее. Например, в 1949 году для SAGE (Semi Automatic Ground Environment - американская система полуавтоматической координации действий перехватчиков). В 1958 году руководство MIT учредило организацию Mitre для управления системой SAGE и ее последующим развитием.В 1968 году в ходе программы SAGE создали сеть ARPANET (Advanced Research Projects Agency Network) — предшественник современного Интернета. Также в рамках СССР, в сеть были объединены разработанные Институтом точной механики и вычислительной техники АН СССР компьютеры «Диана I» и «Диана II» (создатели С. А. Лебедев, В. С. Бурцев).

**Принцип работы компьютерных сетей**

В рамках данного проекта мы будем поверхностно рассматривать работу компьютерных сетей ввиду их архитектурной сложности. Одной из основных моделей передачи данных является модель TCP/IP (Transmission Control Protocol / Internet Protocol). Данная модель имеет несколько уровней:

* Прикладной (Application layer) - HTTP, RTSP, FTP, DNS
* Транспортный (Transport layer) - TCP, UDP, SCTP, DCCP
* Сетевой (Network layer) - IP
* Канальный (Link layer) - Ethernet, IEEE 802.11 WLAN, SLIP, Token Ring, ATM и MPLS, T1, E1

Мы рассмотрим только транспортный уровень, так как в массе своей протоколы прикладного уровня работают поверх транспортного, используя разные порты для подключения:

* HTTP на TCP-порт 80 или 8080
* FTP на TCP-порт 20 (для передачи данных) и 21 (для управляющих команд)
* SSH на TCP-порт 22
* запросы DNS на порт UDP (реже TCP) 53
* обновление маршрутов по протоколу RIP на UDP-порт 520

В основе всех протоколов лежит отправка того или иного текстового запроса посредством этого протокола на определенный адрес и получение ответа.

Главные особенностями протоколов транспортного уровня (TCP, UDP):

* TCP — «гарантированный» транспортный механизм с предварительным установлением соединения, предоставляющий приложению надёжный поток данных, дающий уверенность в безошибочности получаемых данных, перезапрашивающий данные в случае потери и устраняющий дублирование данных. TCP позволяет регулировать нагрузку на сеть, а также уменьшать время ожидания данных при передаче на большие расстояния. Более того, TCP гарантирует, что полученные данные были отправлены точно в такой же последовательности. В этом его главное отличие от UDP.
* UDP — протокол передачи датаграмм без установления соединения. Также его называют протоколом «ненадёжной» передачи, в смысле невозможности удостовериться в доставке сообщения адресату, а также возможного перемешивания пакетов. В приложениях, требующих гарантированной передачи данных, используется протокол TCP.

**Использование протоколов транспортного уровня**

UDP обычно используется в таких приложениях, как потоковое видео и компьютерные игры, где допускается потеря пакетов, а повторный запрос затруднён или не оправдан, либо в приложениях вида запрос-ответ (например, запросы к DNS), где создание соединения занимает больше ресурсов, чем повторная отправка.

И TCP, и UDP используют для определения протокола верхнего уровня число, называемое портом.

**Сокеты и программное взаимодействие с TCP/IP**

Для работы компьютерных программ с TCP/IP протоколом были разработаны сокеты. Сокеты - это название программного интерфейса, созданного для обеспечения обмена данными между процессами. Для взаимодействия между машинами с помощью стека протоколов TCP/IP используются адреса и порты. Адрес представляет собой 32-битную структуру для протокола IPv4, 128-битную для IPv6. Номер порта — целое число в диапазоне от 0 до 65535 (для протокола TCP). Эта пара определяет сокет («гнездо», соответствующее адресу и порту). В данном случае сокеты также можно назвать «розетками», запросы - «вилками», а тип протокола - «типом вилки».

**Создание текстового чата на основе полученных знаний**

При написании программы будут использованы библиотеки sys, socket, threading, environs. Модуль sys - используется для получения аргументов, заданных при запуске программы из терминала, socket - для работы с сокетами, threading - для обеспечения многопоточности, environs - предоставляет возможность работы с файлами типа .env.

**1.1 Создания конфигурационных файлов**

При создании данного приложения будет использован язык программирования Python. Для удобства изменения параметров подключения (адрес и порт) создаем конфигурационный «.env» файл. В нем записываем адрес (используем локальный) и порт (любой порт в диапазоне 49152-65535):

HOST="127.0.0.1"

PORT=53535

Также необходимо создать папку data, где будут располагаться файлы \_\_init\_\_.py, ascii\_arts.py и config.py, которые используются для получения данных из конфигурационного файла.

Код \_\_init\_\_.py:

from . import ascii\_arts, config

Код config.py:

from environs import Env

env = Env()

env.read\_env()

HOST = env.str("HOST")

PORT = env.int("PORT")

Код ascii\_arts.py:

BYE = """

██████╗░██╗░░░██╗███████╗██╗██╗██╗

██╔══██╗╚██╗░██╔╝██╔════╝██║██║██║

██████╦╝░╚████╔╝░█████╗░░██║██║██║

██╔══██╗░░╚██╔╝░░██╔══╝░░╚═╝╚═╝╚═╝

██████╦╝░░░██║░░░███████╗██╗██╗██╗

╚═════╝░░░░╚═╝░░░╚══════╝╚═╝╚═╝╚═╝

"""

WELCOME = """

░██╗░░░░░░░██╗███████╗██╗░░░░░░█████╗░░█████╗░███╗░░░███╗███████╗  ██╗██████╗░

░██║░░██╗░░██║██╔════╝██║░░░░░██╔══██╗██╔══██╗████╗░████║██╔════╝  ╚═╝╚════██╗

░╚██╗████╗██╔╝█████╗░░██║░░░░░██║░░╚═╝██║░░██║██╔████╔██║█████╗░░  ░░░░█████╔╝

░░████╔═████║░██╔══╝░░██║░░░░░██║░░██╗██║░░██║██║╚██╔╝██║██╔══╝░░  ░░░░╚═══██╗

░░╚██╔╝░╚██╔╝░███████╗███████╗╚█████╔╝╚█████╔╝██║░╚═╝░██║███████╗  ██╗██████╔╝

░░░╚═╝░░░╚═╝░░╚══════╝╚══════╝░╚════╝░░╚════╝░╚═╝░░░░░╚═╝╚══════╝  ╚═╝╚═════╝░

"""

**1.2 Создание сервера**

Далее создаем файл server.py, который будет использоваться как «хост» для нашего чата. На него будут приходить запросы для входа в чат и сообщения пользователей, которые при получении будут рассылаться всем участникам.

Код server.py:

import sys, socket, threading

from data.config import HOST, PORT

if len(sys.argv) == 3:

HOST = str(sys.argv[1])

PORT = int(sys.argv[2])

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind((HOST, PORT))

server.listen()

clients\_data = dict()

def broadcast(message:str):

for client in clients\_data.values():

client.send(message.encode('utf-8'))

def handle\_client(client):

while True:

try:

message = client.recv(1024).decode('utf-8')

if "[EXIT\_MESSAGE]" in message:

raise Exception('[DISCONNECT]')

broadcast(message)

except:

nickname = str()

for k, v in clients\_data.items():

if v == client:

nickname = k

client.close()

del clients\_data[nickname]

broadcast(f"{nickname} покинул чат!")

print(f"[INFO][DISCONNECT]: {nickname}")

break

def handle\_connections():

while True:

client, address = server.accept()

print(f"[INFO][CONNECT]: {str(address)}")

client.send("NICKNAME".encode('utf-8'))

nickname = client.recv(1024).decode('utf-8')

clients\_data[nickname] = client

broadcast(f"{nickname} подключился к чату! ")

client.send("Успешное подключение к серверу!".encode('utf-8'))

thread = threading.Thread(target=handle\_client, args=(client,))

thread.start()

if \_\_name\_\_ == "\_\_main\_\_":

print("[INFO] Запуск сервера...")

try:

main\_handling\_thread = threading.Thread(target=handle\_connections)

main\_handling\_thread.start()

print("[INFO] Сервер запущен!")

except Exception as err:

print(f"[EXCEPTION] Ошибка исполнения: {str(err)}")

**1.3 Создание клиента**

Клиентом в данном приложении мы считаем программу, запускаемую пользователем для подключения к чату и используемую для отправки сообщений. Для выхода из чата нужно написать «q» (quit - покинуть от англ.). Создаем файл client.py.

Код client.py:

import sys, socket, threading

from data.config import HOST, PORT

from data.ascii\_arts import WELCOME, BYE

if len(sys.argv) == 3:

HOST = str(sys.argv[1])

PORT = int(sys.argv[2])

print(WELCOME)

nickname = input("[:3] Введите ваше имя: ")

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect((HOST, PORT))

def leave\_room():

client.send("[EXIT\_MESSAGE]".encode('utf-8'))

client.close()

exit()

def handle\_receiving():

while True:

try:

message = client.recv(1024).decode('utf-8')

if message == "NICKNAME":

client.send(nickname.encode('utf-8'))

else:

print(message)

except:

print(BYE)

client.close()

break

def handle\_sending():

while True:

message = input("").strip()

if message == "q":

leave\_room()

if message != "":

client.send(f"{nickname}: {message}".encode('utf-8'))

if \_\_name\_\_ == "\_\_main\_\_":

try:

receiving\_handling\_thread = threading.Thread(target=handle\_receiving)

receiving\_handling\_thread.start()

sending\_handling\_thread = threading.Thread(target=handle\_sending)

sending\_handling\_thread.start()

except:

pass

**1.4 Запуск программы**

Для использования чата необходимо в первую очередь чтобы компьютер «хост» запустил сервер (файл server.py), который будет принимать подключения и осуществлять менеджмент сообщений пользователей. Далее пользователи запускают клиент (файл client.py) и подключаются к серверу. При запуске клиента необходимо ввести свое имя. После всех проделанных действий приложение готово к использованию. Для выхода из программы нужно ввести «q», после чего клиент будет отключен от сервера, а все участники получат уведомление о выходе пользователя.

**Заключение**

В настоящее время такие чаты не имеют широкого применения ввиду замены их на более простые и удобные. Однако все современные чаты основаны на протоколах и технологиях, примененных в данном проекте.

Созданный чат можно использовать как учебный пример работы компьютерных сетей в школах, а также для описания базовых принципов их работы.

**Список литературы**

**Книги, печатная периодика**

1. Программирование на Python. Том 1, 2. Марк Лутц.

2. Black Hat Python, 2nd Edition: Python Programming for Hackers and Pentesters 2nd Edition. Justin Seitz.

3. Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers 1st Edition. TJ O'Connor.

**Электронные ресурсы**

1. Интернет // WikipediaURL: [https://ru.wikipedia.org/wiki/Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) (Дата обращения: 09.02.2022)

2. TCP/IP // WikipediaURL: <https://ru.wikipedia.org/wiki/TCP/IP> (Дата обращения: 09.02.2022)

3. Сокет // WikipediaURL: [https://ru.wikipedia.org/wiki/Сокет\_(программный\_интерфейс)](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D0%B5%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81)) (Дата обращения: 09.02.2022)

2. Компьютерная (вычислительная) сеть // WikipediaURL: [https://ru.wikipedia.org/wiki/Вычислительная\_сеть](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) (Дата обращения: 09.02.2022)

**Приложение**

1. Полный код: https://github.com/pythonuntermensh/school\_project