Маслов А.Н. ИД23-1  
Сетевые Системы и Приложения

**Практическая работа - Простейшие TCP-клиент и эхо-сервер**

**Основные задания 1-4:**

server.py

import socket

print("СЕРВЕР: Запуск сервера...")

sock = socket.socket()

sock.bind(('', 9090))

print("СЕРВЕР: Сокет привязан к порту 9090.")

sock.listen(0)

print("СЕРВЕР: Начало прослушивания входящих подключений.")

conn, addr = sock.accept()

print(f"СЕРВЕР: Подключился клиент с адресом {addr}.")

msg = ''

while True:

data = conn.recv(1024)

if not data:

print("СЕРВЕР: Клиент прекратил отправку данных или отключился.")

break

print(f"СЕРВЕР: Получена порция данных от клиента: {data.decode()}")

msg += data.decode()

conn.send(data)

print("СЕРВЕР: Порция данных отправлена обратно клиенту.")

print("СЕРВЕР: Полное сообщение, полученное от клиента:")

print(msg)

conn.close()

print("СЕРВЕР: Клиент отключился.")

sock.close()

print("СЕРВЕР: Остановка сервера. Работа завершена.")

client.py

import socket

sock = socket.socket()

sock.setblocking(1)

print("КЛИЕНТ: Попытка соединения с сервером на адресе localhost:9090...")

sock.connect(('localhost', 9090))

print("КЛИЕНТ: Соединение с сервером установлено.")

msg = input("Введите строку для отправки серверу: ")

print("КЛИЕНТ: Отправка данных серверу...")

sock.send(msg.encode())

print("КЛИЕНТ: Данные успешно отправлены серверу.")

print("КЛИЕНТ: Ожидание ответа от сервера...")

data = sock.recv(1024)

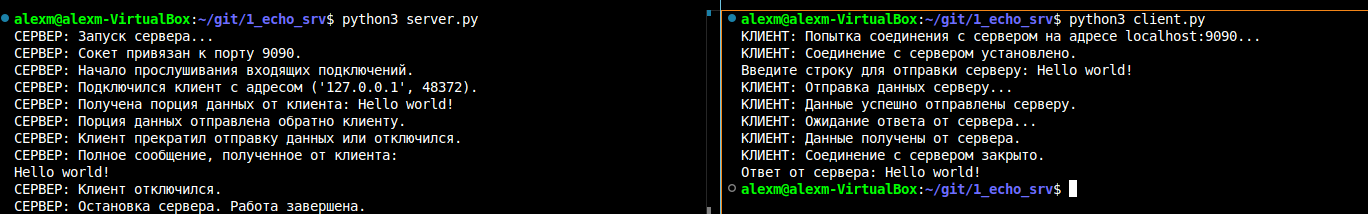
print("КЛИЕНТ: Данные получены от сервера.")

sock.close()

print("КЛИЕНТ: Соединение с сервером закрыто.")

print("Ответ от сервера:", data.decode())

Результат:

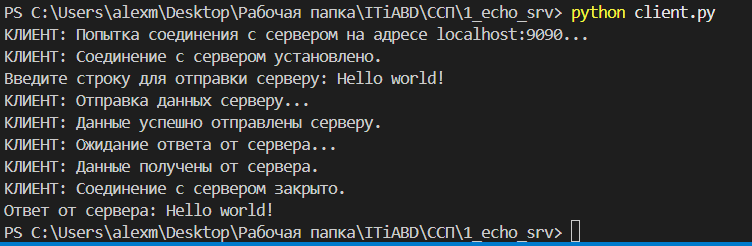


**Задания для самостоятельного выполнения:**

Задание 1:

Подключение с локальной машины.

Локальная машина:



Виртуальная машина:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Задание 2:

client.py

import socket

sock = socket.socket()

sock.setblocking(1)

print("КЛИЕНТ: Попытка соединения с сервером на адресе localhost:9090...")

sock.connect(('localhost', 9090))

print("КЛИЕНТ: Соединение с сервером установлено.")

while True:

msg = input("Введите строку для отправки серверу: ")

if msg == "exit":

sock.close()

print("КЛИЕНТ: Соединение с сервером закрыто.")

break

print("КЛИЕНТ: Отправка данных серверу...")

sock.send(msg.encode())

print("КЛИЕНТ: Данные успешно отправлены серверу.")

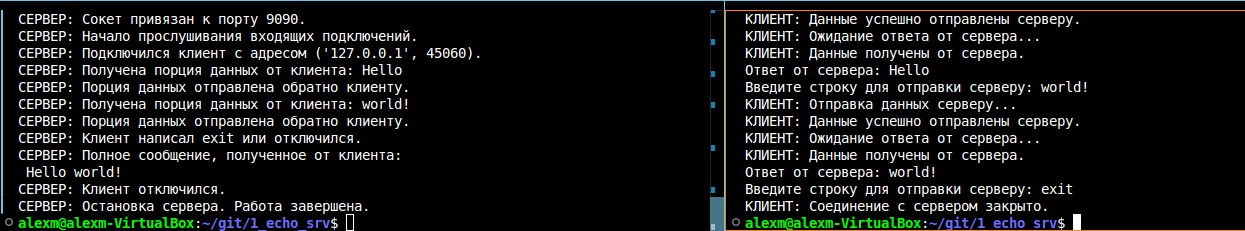
print("КЛИЕНТ: Ожидание ответа от сервера...")

data = sock.recv(1024)

print("КЛИЕНТ: Данные получены от сервера.")

print("Ответ от сервера:", data.decode())

Результат:



Задание 3:

server.py

import socket

print("СЕРВЕР: Запуск сервера...")

sock = socket.socket()

sock.bind(('', 9090))

print("СЕРВЕР: Сокет привязан к порту 9090.")

sock.listen(0)

print("СЕРВЕР: Начало прослушивания входящих подключений.")

shutdown\_requested = False

while not shutdown\_requested:

conn, addr = sock.accept()

print(f"СЕРВЕР: Подключился клиент с адресом {addr}.")

msg = ''

try:

while True:

data = conn.recv(1024)

if not data:

print("СЕРВЕР: Клиент прекратил отправку данных или отключился.")

break

decoded\_data = data.decode().strip().lower()

print(f"СЕРВЕР: Получена порция данных: {decoded\_data}")

if decoded\_data == "shutdown":

conn.send(b"Server shutting down...")

shutdown\_requested = True

break

msg += “ “ + decoded\_data

conn.send(data)

print(f"СЕРВЕР: Полное сообщение от клиента: {msg}")

except ConnectionResetError:

print(f"СЕРВЕР: Клиент {addr} отключился неожиданно.")

finally:

conn.close()

print(f"СЕРВЕР: Соединение с {addr} закрыто.")

sock.close()

print("СЕРВЕР: Остановка сервера. Работа завершена.")

client.py

import socket

sock = socket.socket()

sock.setblocking(1)

print("КЛИЕНТ: Попытка соединения с сервером на адресе localhost:9090...")

sock.connect(('localhost', 9090))

print("КЛИЕНТ: Соединение с сервером установлено.")

while True:

msg = input("Введите строку для отправки серверу: ")

if msg == "exit":

sock.close()

print("КЛИЕНТ: Соединение с сервером закрыто.")

break

print("КЛИЕНТ: Отправка данных серверу...")

sock.send(msg.encode())

print("КЛИЕНТ: Данные успешно отправлены серверу.")

print("КЛИЕНТ: Ожидание ответа от сервера...")

data = sock.recv(1024)

print("КЛИЕНТ: Данные получены от сервера.")

print("Ответ от сервера:", data.decode())

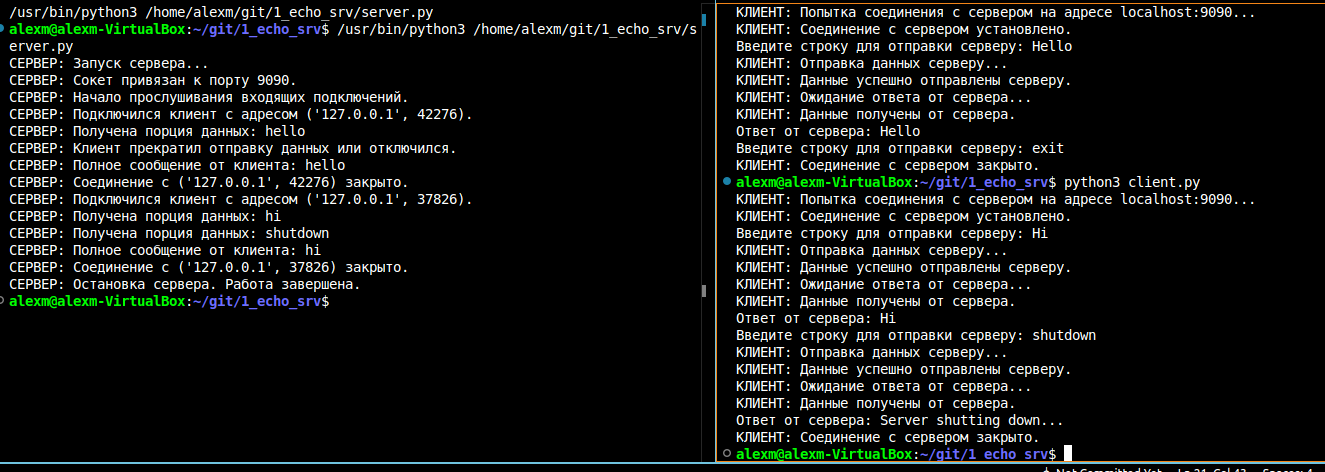
if msg == "shutdown":

sock.close()

print("КЛИЕНТ: Соединение с сервером закрыто.")

break

Результат:



Задание 4:

client.py

import socket

def get\_host(default\_host):

host = input(f"Введите имя хоста (по умолчанию '{default\_host}'): ").strip()

return host or default\_host

def get\_port(default\_port):

while True:

port\_str = input(f"Введите номер порта (по умолчанию {default\_port}): ").strip()

if not port\_str:

return default\_port

try:

port = int(port\_str)

if 0 <= port <= 65535:

return port

else:

print(f"Ошибка: Порт должен быть от 0 до 65535. Используется порт по умолчанию {default\_port}.")

return default\_port

except ValueError:

print("Ошибка: Введите корректное число для порта.")

sock = socket.socket()

sock.setblocking(1)

default\_host = 'localhost'

default\_port = 9090

host = get\_host(default\_host)

port = get\_port(default\_port)

print(f"КЛИЕНТ: Попытка соединения с сервером на адресе {host}:{port}...")

try:

sock.connect((host, port))

print("КЛИЕНТ: Соединение с сервером установлено.")

except Exception as e:

print(f"КЛИЕНТ: Ошибка подключения: {e}")

exit()

while True:

msg = input("Введите строку для отправки серверу: ")

if msg == "exit":

sock.close()

print("КЛИЕНТ: Соединение с сервером закрыто.")

break

print("КЛИЕНТ: Отправка данных серверу...")

sock.send(msg.encode())

print("КЛИЕНТ: Данные успешно отправлены серверу.")

print("КЛИЕНТ: Ожидание ответа от сервера...")

data = sock.recv(1024)

print("КЛИЕНТ: Данные получены от сервера.")

print("Ответ от сервера:", data.decode())

if msg == "shutdown":

sock.close()

print("КЛИЕНТ: Соединение с сервером закрыто.")

break

server.py

import socket

def get\_port(default\_port):

while True:

port\_str = input(f"Введите номер порта для сервера (по умолчанию {default\_port}): ").strip()

if not port\_str:

return default\_port

try:

port = int(port\_str)

if 0 <= port <= 65535:

return port

else:

print(f"Ошибка: Порт должен быть от 0 до 65535. Используется порт по умолчанию {default\_port}.")

return default\_port

except ValueError:

print("Ошибка: Введите корректное число для порта.")

print("СЕРВЕР: Запуск сервера...")

sock = socket.socket()

default\_port = 9090

port = get\_port(default\_port)

try:

sock.bind(('', port))

print(f"СЕРВЕР: Сокет привязан к порту {port}.")

except Exception as e:

print(f"СЕРВЕР: Ошибка привязки к порту {port}: {e}")

exit()

sock.listen(0)

print("СЕРВЕР: Начало прослушивания входящих подключений.")

shutdown\_requested = False

while not shutdown\_requested:

conn, addr = sock.accept()

print(f"СЕРВЕР: Подключился клиент с адресом {addr}.")

msg = ''

try:

while True:

data = conn.recv(1024)

if not data:

print("СЕРВЕР: Клиент прекратил отправку данных или отключился.")

break

decoded\_data = data.decode().strip().lower()

print(f"СЕРВЕР: Получена порция данных: {decoded\_data}")

if decoded\_data == "shutdown":

conn.send(b"Server shutting down...")

shutdown\_requested = True

break

msg += “ “ + decoded\_data

conn.send(data)

print(f"СЕРВЕР: Полное сообщение от клиента: {msg}")

except ConnectionResetError:

print(f"СЕРВЕР: Клиент {addr} отключился неожиданно.")

finally:

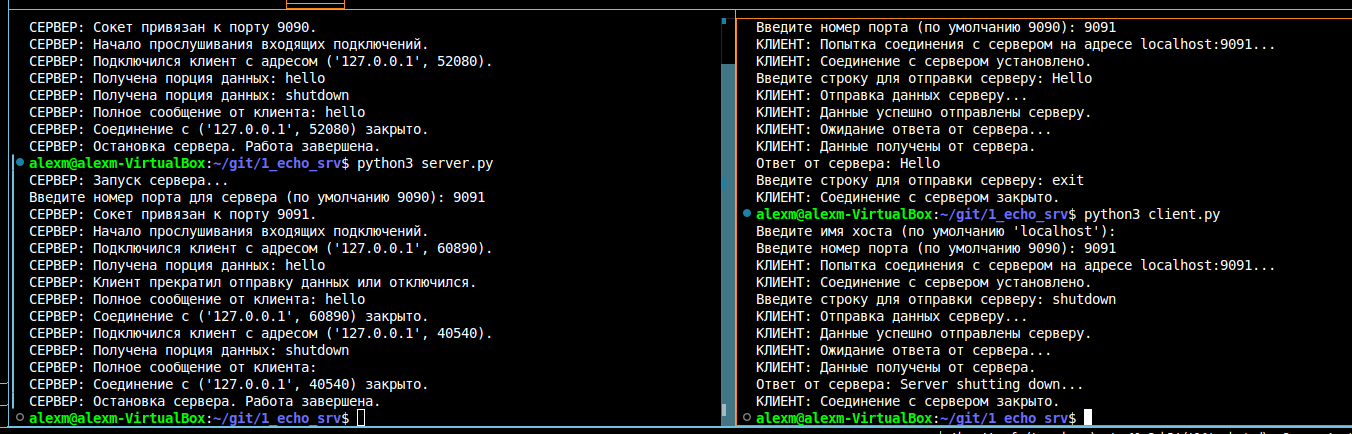
conn.close()

print(f"СЕРВЕР: Соединение с {addr} закрыто.")

sock.close()

print("СЕРВЕР: Остановка сервера. Работа завершена.")

Результат:



Задание 5:

server.py

import socket

import datetime

def get\_port(default\_port):

while True:

port\_str = input(f"Введите номер порта для сервера (по умолчанию {default\_port}): ").strip()

if not port\_str:

return default\_port

try:

port = int(port\_str)

if 0 <= port <= 65535:

return port

else:

print(f"Ошибка: Порт должен быть от 0 до 65535. Используется порт по умолчанию {default\_port}.")

return default\_port

except ValueError:

print("Ошибка: Введите корректное число для порта.")

def log(message, log\_file):

timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

log\_entry = f"[{timestamp}] СЕРВЕР: {message}\n"

try:

with open(log\_file, "a", encoding="utf-8") as f:

f.write(log\_entry)

except Exception as e:

print(f"Ошибка записи в лог-файл: {e}")

log\_file = "server.log"

default\_port = 9090

port = get\_port(default\_port)

log("Запуск сервера...", log\_file)

sock = socket.socket()

try:

sock.bind(('', port))

log(f"Сокет привязан к порту {port}.", log\_file)

except Exception as e:

log(f"Ошибка привязки к порту {port}: {e}", log\_file)

exit()

sock.listen(0)

log("Начало прослушивания входящих подключений.", log\_file)

shutdown\_requested = False

while not shutdown\_requested:

conn, addr = sock.accept()

log(f"Подключился клиент с адресом {addr}.", log\_file)

msg = ''

try:

while True:

data = conn.recv(1024)

if not data:

log("Клиент прекратил отправку данных или отключился.", log\_file)

break

decoded\_data = data.decode().strip().lower()

log(f"Получена порция данных: {decoded\_data}", log\_file)

if decoded\_data == "shutdown":

conn.send(b"Server shutting down...")

shutdown\_requested = True

break

msg += " " + decoded\_data

conn.send(data)

log(f"Полное сообщение от клиента: {msg}", log\_file)

except ConnectionResetError:

log(f"Клиент {addr} отключился неожиданно.", log\_file)

finally:

conn.close()

log(f"Соединение с {addr} закрыто.", log\_file)

sock.close()

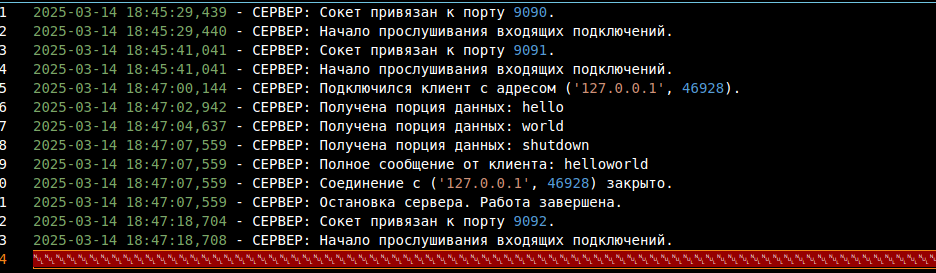
log("Остановка сервера. Работа завершена.", log\_file)

Результат:

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

server.log



Задание 6:

server.py

import socket

import datetime

def get\_port(default\_port):

while True:

port\_str = input(f"Введите номер порта для сервера (по умолчанию {default\_port}): ").strip()

if not port\_str:

return default\_port

try:

port = int(port\_str)

if 0 <= port <= 65535:

return port

else:

print(f"Ошибка: Порт должен быть от 0 до 65535. Используется порт по умолчанию {default\_port}.")

return default\_port

except ValueError:

print("Ошибка: Введите корректное число для порта.")

def log(message, log\_file):

timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

log\_entry = f"[{timestamp}] СЕРВЕР: {message}\n"

try:

with open(log\_file, "a", encoding="utf-8") as f:

f.write(log\_entry)

except Exception as e:

print(f"Ошибка записи в лог-файл: {e}")

log\_file = "server.log"

default\_port = 9090

port = get\_port(default\_port)

current\_port = port

sock = socket.socket()

while current\_port <= 65535:

try:

sock.bind(('', current\_port))

print(f"СЕРВЕР: Используется порт {current\_port}") # Консольный вывод

log(f"Сокет привязан к порту {current\_port}.", log\_file)

break

except OSError:

log(f"Порт {current\_port} занят, пробуем следующий...", log\_file)

current\_port += 1

else:

log("Нет свободных портов в диапазоне 9090-65535", log\_file)

print("СЕРВЕР: Нет свободных портов в диапазоне 9090-65535")

exit()

sock.listen(0)

log("Начало прослушивания входящих подключений.", log\_file)

shutdown\_requested = False

while not shutdown\_requested:

conn, addr = sock.accept()

log(f"Подключился клиент с адресом {addr}.", log\_file)

msg = ''

try:

while True:

data = conn.recv(1024)

if not data:

log("Клиент прекратил отправку данных или отключился.", log\_file)

break

decoded\_data = data.decode().strip().lower()

log(f"Получена порция данных: {decoded\_data}", log\_file)

if decoded\_data == "shutdown":

conn.send(b"Server shutting down...")

shutdown\_requested = True

break

msg += " " + decoded\_data

conn.send(data)

log(f"Полное сообщение от клиента: {msg}", log\_file)

except ConnectionResetError:

log(f"Клиент {addr} отключился неожиданно.", log\_file)

finally:

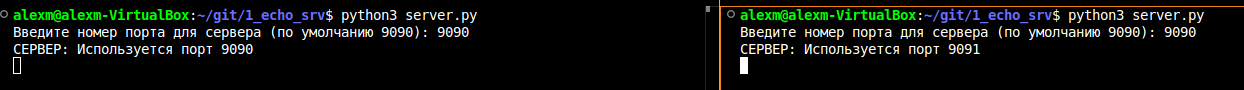
conn.close()

log(f"Соединение с {addr} закрыто.", log\_file)

sock.close()

log("Остановка сервера. Работа завершена.", log\_file)

Результат:



server.log

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.