Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет   
имени первого Президента России Б.Н. Ельцина»

Институт новых материалов и технологий

Кафедра «Теплофизика и информатика в металлургии»

ОТЧЕТ

по практической работе №3

по дисциплине «Операционные системы»

Направление 09.03.02 «Информационные системы и технологии» (уровень бакалавриата)

Образовательная программа  
09.03.02/33.02 «Информационные системы и технологии» (СУОС)

Студент

группы НМТ-323901 О.А. Суворов

Преподаватель:

профессор, д.т.н. А.С.Истомин

Екатеринбург

2024

Практическая работа №3.

1. Цели работы:
2. Изучение особенностей работы многопоточных приложений
3. Изучение механизмов сетевого взаимодействия

**Задача:** Реализовать приложение (чат), которое позволяет обмениваться сообщениями между своими копиями, то есть является одновременно и клиентом, и сервером.

1. Результаты выполнения
2. Описание программы:

Программа представляет собой чат-приложение, реализованное на языке Python, с использованием стандартных библиотек для работы с сетью и многопоточностью. Основные возможности:

* Обмен текстовыми сообщениями между пользователями через протоколы TCP и UDP.
* Настройка адреса и порта через конфигурационный файл config.json или автоматическая инициализация при отсутствии файла.
* Каждое сообщение сопровождается отметкой времени с точностью до секунды.
* Логирование событий и ошибок в файлы chat.log и server.log.

1. Ключевые файлы репозитория:

* client.py: реализация клиентской части приложения.
* server.py: реализация серверной части приложения.
* config.py: модуль для загрузки и обработки конфигурационных данных.

1. Сетевые особенности:

* TCP: используется для установления надежного соединения между клиентом и сервером, обеспечивая гарантированную доставку сообщений.
* UDP: применяется для отправки широковещательных сообщений, позволяя обнаруживать другие экземпляры приложения в сети без установления постоянного соединения.

1. Многопоточность

В серверной части (server.py) для обработки каждого входящего соединения создается отдельный поток, что позволяет обслуживать несколько клиентов одновременно.

В клиентской части (client.py) используется поток для прослушивания входящих сообщений от сервера, обеспечивая асинхронный прием данных.

1. Защита данных и устойчивость

Реализована обработка исключений при сетевых операциях, что предотвращает аварийное завершение программы при возникновении ошибок.

Логирование событий и ошибок позволяет отслеживать работу приложения и диагностировать возможные проблемы.

Корректное завершение работы приложения обеспечивается закрытием всех сокетов и завершением потоков.

1. Принципы работы программы
2. **Инициализация**

При запуске программа загружает параметры подключения из файла config.json с помощью модуля config.py. Если файл отсутствует, используются значения по умолчанию.

1. **Отправка и прием сообщений**

Клиент устанавливает соединение с сервером через TCP и отправляет сообщение. Клиент может отправлять сообщение всем пользователям, конкретному пользователю (@user), может отправлять сообщение через UDP (!broadcast), узнавать какие пользователи в сети (/users) и выходить из чата (/exit).

Сервер принимает соединение, получает сообщение и отправляет другим клиентам.

1. **Логирование**

Все события, включая отправку и получение сообщений, а также ошибки, записываются в лог-файлы с указанием времени события.

1. **Завершение работы**

При завершении работы приложения закрываются все активные соединения и завершаются потоки, что предотвращает зависание и утечки ресурсов.

1. **Работа приложения**

Результат работы при подключенном файле (см. рисунок 1) и при отсутствии файла (см. рисунок 2).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 1 – Результат работы чата

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

Автоматически созданное описание

Рисунок 2 – При отсутствии файла

Файл server.log с ошибками, приходящими на сервер (см. рисунок 3).

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

Автоматически созданное описание

Рисунок 3 – server.log

Файл chat.log с зашифрованными сообщениями между клиентами на сервере (см. рисунок 4).

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 4 – chat.log

Ссылка на репозиторий: <https://github.com/SuvorovOleg19/ChatApp_Suv>

**Заключение**

Программа успешно реализует функционал чат-приложения с использованием многопоточности и сетевого взаимодействия через протоколы TCP и UDP. Все поставленные задачи выполнены, включая:

* Корректную работу с несколькими потоками.
* Надежный обмен сообщениями между клиентом и сервером.
* Настраиваемые параметры подключения.
* Логирование событий и ошибок.

Корректное завершение работы без зависаний.

Данный проект демонстрирует понимание принципов разработки многопоточных сетевых приложений и может служить основой для дальнейшего развития и добавления новых функций.