Федеральное государственное автономное образовательное учреждение высшего образования
Университет ИТМО

Кафедра Вычислительной Техники

Дисциплина: Системное программное обеспечение

Лабораторная работа №3

Вариант 4

Выполнил: **Доморацкий Эридан Алексеевич**

Группа: **P33113**

Преподаватель: **Кореньков Юрий Дмитриевич**

2021 г.

Задание

Разработать клиент-серверное приложение. Для организации взаимодействия по сети, поддержки множества соединения использовать программные интерфейсы (API) операционной системы.

Сервер и клиенты взаимодействуют по протоколу, реализованному на базе сокетов (использовать TCP, если в варианте задания не указано обратное). Сервер должен поддерживать условно неограниченное количество клиентов. На всех этапах взаимодействия клиента и сервера должна быть предусмотрена обработка данных независимо от их размера.

Порядок выполнения:

1. Выполнить анализ предметной области, которая задается вариантом к лабораторной работе. Результатом анализа должен быть набор сущностей, которые будут использоваться на уровне типов данных (структур, операций) при реализации программы.
2. Составить диаграмму, на которой схематически будут показаны результаты анализа: сущности, их атрибуты и взаимосвязи.
3. Составить план постепенного выполнения задания: какие части функциональности, в каком порядке предполагается реализовывать, в каком порядке и как проверять их работоспособность.
4. Загрузить все полученные артефакты в отдельную директорию «docs» репозитория, в корневую директорию положить readme.md с номером варианта и кратким описанием.
5. Продемонстрировать составленные диаграмму и план преподавателю. Для этого достаточно просто отправить преподавателю ссылку на репозиторий.
6. После проверки и получения рекомендаций приступить к реализации программы, создавая на каждый этап выполнения отдельную ветку в репозитории.
7. По завершении каждого этапа создать pull-request, включив преподавателя в число reviewer-ов.
8. Если pull-request отклоняется преподавателем, выполнить необходимые правки и обновить его, запросив повторное ревью.
9. Когда pull-request одобрен, «слить» его с основной веткой кода, после чего создать новую ветку для работы над следующим этапом.

Вариант 4. Веб-сервер

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки.

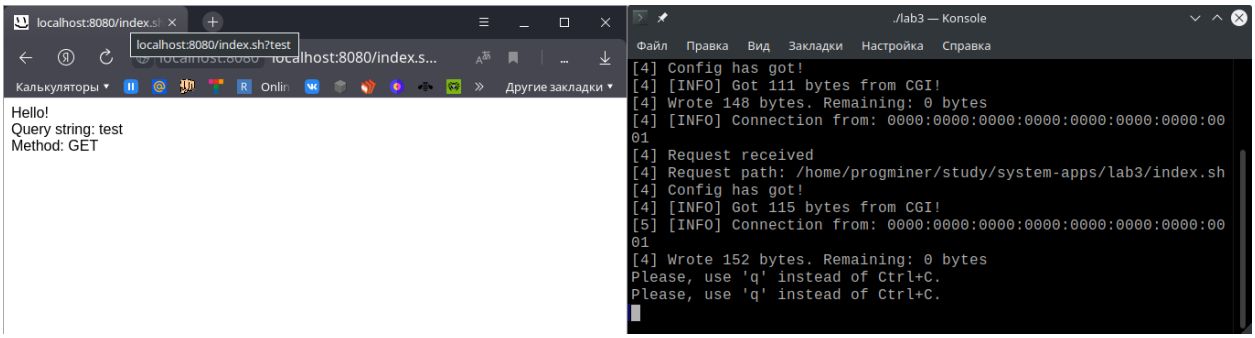
При запуске в режиме сервера аргументом командной строки указывается путь к директории для работы (обслуживания). Завершение программы-сервера осуществляется по нажатию ключевой клавиши (например, Q). В любой директории, начиная с рабочей, может быть расположен файл, содержащий управляющие инструкции. По умолчанию сервер возвращает запрашиваемые по HTTP файлы относительно рабочей директории. При этом управляющими инструкциями задаётся следующая дополнительная информация:

- отображение расширения файла на имя MIME-типа, который отдаётся в заголовках;
- CGI-обработчик или интерпретатор для файлов (скриптов) по расширению в соответствии;
- запрет отдавать файлы с фильтром по регулярному выражению (System.Text.Regex).

В соответствии с HTTP/1.1 через одно соединение может быть передан ряд запросов. Помимо GET, приём тела при обработке PUT/POST запросов на скрипт. В консоль при этом

Выполнение

Ссылка на исходный код: https://gitlab.se.ifmo.ru/ProgMiner/lab3-system-apps/.



Вывод

В ходе выполнения лабораторной работы был реализован HTTP сервер, поддерживающий Connection: keep-alive и вызов CGI (было протестировано с помощью PHP). Для разработки был выбран асинхронный подход, что означает большую скорость, чем синхронный – создание нового потока на каждое соединение, за счёт ограничения максимального числа потоков, в следствие чего ОС тратит меньше времени на переключение контекста и планирование задач. Для реализации был использован системный вызов POSIX poll, предназначенный для одновременного ожидания событий от нескольких файловых дескрипторов (открытых файлов, сокетов и т. д.) с целью перехвата этих событий и их последующей обработки. Однако, такой подход в языке Си сильно усложняет процесс написания кода по причине отсутствия в этом языке таких высокоуровневых механизмов как async/await, замыкания и автоматическое управление памятью. В связи с этим процесс написания кода заметно усложняется необходимостью вручную выделять и инициализировать контекст для каждой callback-функции и не забывать очищать память когда контекст больше не нужен. Кроме того с помощью механизма poll отлично решается задача создания интерактивного консольного интерфейса, т. к. механизм работы с консолью в случае создания оного наиболее эффективен в асинхронном варианте.