Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
Факультет программной инженерии и компьютерной техники

Системное программное обеспечение
Лабораторная №3
Вариант 2

Выполнил:
Беляков Дмитрий
Группа:
P33122
Преподаватель:
Кореньков Ю. Д.

Санкт-Петербург

2021

Цели:

Изучение способов взаимодействия между сетевыми службами низкого уровня в асинхронном режиме.

Описание работы

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, еслив варианте заданияне 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 в качестве элементов данных (типов, структур, операций) и/или составных частей программной архитектуры (компонентов, модулей) 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 вотдельную 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 преподавателяв число reviewer-ов.

8.Если pull-requestотклоняется преподавателем, выполнить необходимые правки и обновить его, запросив повторное ревью.9.Когда pull-requestодобрен,«слить» его с основной веткой кода, после чего создать новую ветку для работы над следующим этапом.

Задачи:

1. Изучить способы организации клиент-серверной архитектуры на языке C
2. Изучить способы асинхронного взаимодействия в языке C
3. Продумать клиентскую архитектуру, продумать взаимодействие состояние клиента и графики
4. Реализовать это всё

Задание:

2. Доска обсуждений (Discussion-board)

Программа может выполняться в двух режимах: сервер или клиент. Режим определяется аргументом командной строки. В режиме сервера линейно отображается журнал всех входящих и исходящих сообщений, завершение программы-сервера выполняется по нажатию

ключевой клавиши (например, Q).При запуске в режиме клиента программе в качестве аргументов командной строки также передается имя пользователя и адрес сервера.Дискуссионная доска представляет собой дерево сообщений, узлы которого можно разворачивать или сворачивать для просмотра ответов на интересующее сообщение. Для отправки своего сообщения в дереве выбирается существующее, под которым появится отправляемый ответ. Предусмотреть прокрутку дерева, если оно не помещается на экране, а такжеиндикацию появления новых ответов, в том числе в неразвёрнутых ветках. Собственное сообщение вводится в отдельной строке в нижней части окна,не блокируя работу дерева.

Аспекты реализации:

Пересылаемые данные:

От клиента — маленькие сообщения содержащие имя отправителя, сообщения, а так же номер родителя, то есть ветку, куда присоеденить. От сервера — полноценные ноды. Так как доска обсуждений представляет собой дерево, то каждая нода содержала id на родителя, ребёнка, а так же «sibling» (брата или сестру) и остальные необходимые атрибуты (имя, дата, сообщение).

Сервер:

Сервер был реализован однопоточно с использованием механизма «poll», что позволило не заниматься синхронизацией потоков и отказаться от примитивов синхронизации. Единственное, на что стоит обратить внимание, чтобы получить всё дерево клиенту необходимо отправить сообщение с parent id равным -1, так как такой id может быть только у корневого сообщения и это его parent id (так как никакого родителя у него нет), то это можно спокойно использовать без разногласий. На добавление нового сообщения сервер отправляет всем клиентам по 2 сообщения: сообщение, где обновился child/sibling id в ходе добавления нового сообщения и само новое сообщение. При отключении клиента возникает event, poll «просыпается» и на выходе получает отрицательное значение считанных байтов, что сигнализирует об отключении, в ходе чего сервер переупаковывает массивы файловых дескрипторов сокетов.

Клиент:

Клиент также однопоточный, ждёт ввода от пользователя на poll, а также новых сообщений от сервера. Состоит из 2 частей — логики приложения и графики (что было самым сложным). Логика представляет собой тривиальное приложение, приходит сообщение — обновляем данные, отправляем сообщение — ждём, нажимаем кнопку — обновляем, что надо обновить. При этом отключен ECHO у пользователя и всё выводится в эмулятор терминала самим приложением.

Вместе с графикой добавилась хранимая информация, кроме дерева хранится массив visible_status (по числу элементов), где хранится для каждой ноды буленовские переменные — новое сообщение и надо ли содержимое этого элемента отображать (скрыта ли ветка), есть также массив draw_order (по числу возможных отображаемых элементов), который по сути является порядком отрисовки элементов в данный момент.

Массив draw_order строится рекурсивно (так как это дерево), учитывая массив visible. Также приложение имеет состояние, с какого элемента надо в данный момент отрисовать и с какой колонки, для того чтобы реализовать возможно листать во все стороны).

Ссылка на код: https://gitlab.se.ifmo.ru/kevinche75/spo_lab3_done_itmo_spring_2021

Вывод: в данной лабораторной работе я освоил основные механизмы асинхронности, разобрался с библиотекой termios, реализовал графику в терминале (что оказалось самым сложным).