Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5-7 по курсу**

**«Операционные системы»**

Студент: Муханов Андрей Ильич

Группа: М8О-209Б-23

Вариант: 12

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Выполненная работа
4. Вывод

**Репозиторий**

https://github.com/andreymuch/OC\_MAI

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

* Управлении серверами сообщений (№5)
* Применение отложенных вычислений (№6)
* Интеграция программных систем друг с другом (№7)

**Задание**

Реализовать распределенную систему по асинхронной обработке запросов. В данной распределенной системе должно существовать 2 вида узлов: «управляющий» и «вычислительный». Необходимо объединить данные узлы в соответствии с той топологией, которая определена вариантом. Связь между узлами необходимо осуществить при помощи технологии очередей сообщений. Также в данной системе необходимо предусмотреть проверку доступности узлов в соответствии с вариантом. При убийстве («kill -9») любого вычислительного узла система должна пытаться максимально сохранять свою работоспособность, а именно все дочерние узлы убитого узла могут стать недоступными, но родительские узлы должны сохранить свою работоспособность.

Управляющий узел отвечает за ввод команд от пользователя и отправку этих команд на вычислительные узлы.  
  
  
  
Отчет по реализации распределенной системы асинхронной обработки запросов

**Выполненная работа**

1. Введение

Система состоит из управляющего узла (`master`) и вычислительных узлов (`worker`), объединенных в бинарное дерево. Узлы взаимодействуют через ZeroMQ, обеспечивая асинхронную обработку команд. Система устойчива к сбоям: при потере узла удаляется его поддерево, но родительские узлы остаются работоспособными.

2. Структура системы

- Управляющий узел (master):

- Управляет созданием/удалением узлов.

- Принимает команды от пользователя (`create`, `exec`, `pingall`,`print`).

- Отслеживает доступность узлов.

- Использует бинарное дерево (`BinaryTree`) для хранения топологии. Изображение выглядит как диаграмма, текст, План, Технический чертеж

Контент, сгенерированный ИИ, может содержать ошибки.

- Вычислительные узлы (worker):

- Обрабатывают команды `exec` (сохранение/загрузка данных в локальный словарь).

- Пересылают сообщения между узлами.

- Отвечают на проверки доступности (`pingall`).

Изображение выглядит как диаграмма, текст, зарисовка, линия

Контент, сгенерированный ИИ, может содержать ошибки.

3. Реализация топологии (бинарное дерево)

Класс `BinaryTree` (binary\_tree.h):

- Узлы: Каждый узел содержит:

- `id` — уникальный идентификатор.

- `ipAdressAndPort` — адреса для связи с родителем и потомками.

- Указатели на левого/правого потомка.

- Методы:

- `addNode()` — добавляет узлы в порядке заполнения уровней дерева.

- `deleteNode()` — удаляет узел и его поддерево.

- `print()` — визуализирует дерево в консоли.

- Механизм проверки уникальности: Используется `unordered\_set<int> ids`.

4. Коммуникация между узлами

- Протокол: ZeroMQ (сокеты `ROUTER`, `DEALER`, `PUSH/PULL`).

- Сценарии:

1. Создание узла:

- Master запускает `worker.exe` с параметрами (адрес родителя, порт для связи, ID).

- В дерево добавляется запись с адресами узла.

2. Пересылка команд:

- Команды `exec` и `pingall` рассылаются через цепочку сокетов.

5. Поддерживаемые команды

1. `create id` — создает узел с указанным ID.

2. `exec id name [value]` — сохраняет/загружает данные:

- `exec 10 key 42` — сохранение.

- `exec 10 key` — загрузка.

3. `pingall` — проверяет доступность всех узлов.

4. `print` — выводит структуру дерева.

6. Проверка доступности узлов

- Механизм `pingall`:

1. Master рассылает команду `pingall`.
2. Worker-ы отвечают `pong id`.
3. Master сравнивает ответы с IDs дерева.
4. Неответившие узлы удаляются вместе с поддеревьями.

7. Обработка сбоев

- При убийстве узла (`kill -9`):

- Master обнаруживает отсутствие `pong` ответа.

- Узел и его потомки удаляются из дерева.

- Пример лога:

Узел 15 не отвечает и будет удален из дерева

**Вывод**

Система удовлетворяет требованиям:

- Топология бинарного дерева реализована через класс `BinaryTree`.

- Асинхронная обработка обеспечена ZeroMQ и многопоточностью.

- Устойчивость к сбоям: родительские узлы остаются работоспособными.

- Команды `create`, `exec`, `pingall` поддерживаются.

: