Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №6-8 по курсу**

**«Операционные системы»**

Управление серверами сообщений

Применение отложенных вычислений

Интеграция программных систем друг с другом

Студент: Трофимов Максим Андреевиич

Группа: М80 – 201Б-18

Вариант: 28

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: **\_\_\_\_\_\_\_\_\_\_\_**

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2019

**Содержание**

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Демонстрация работы программы
5. Вывод

**1. Постановка задачи**

Реализовать распределённую систему по асинхронной обработке запросов. В данной распределённой системе должно существовать 2 вида узлов: «управляющий» и «вычислительный». Объединить узлы топологией «Звезда». Вычислительные узлы должны выполнять запроса на суммирование n целых чисел, а также должна быть предусмотрена проверка доступности всех существующих узлов.

**2. Общие сведения о программе.**

Существует две программы: под управляющий и вычислительный узел. Пользователь запускает программу с параметром — цифрой, являющейся идентификатором этого управляющего узла. Запущенная программа принимает на вход команды согласно варианту:

* create id [parent] — создание вычислительного узла id и присоединяется к узлу [parent], но в моём случае он присоединяется к текущему управляющему узлу.
* remove id — удалить вычислительный узел id
* union id — объединиться с управляющем узлом id, т. е. после объединения вычислительные узлы становятся общими, т. е. с любого управляющего узла можно взаимодействовать с любыми вычислительными узлами объединённых управляющих узлов.
* exec id n k1 … kn отправить на вычислительный узел id запрос на суммирование n целых чисел чисел.
* pingall — проверка на недоступность всех вычислительных узлов. Если все доступны, то выводится ответ -1.
* exit — выход из программы.

**3.Общий метод и алгоритм решения**

При запуске создаётся дочерний процесс, который по факту является созданным управляющим узлом, а родительский процесс отвечает за общение между пользователем и управляющем узлом. Управляющий узел имеет дублирующие функции, т. к. запрос может выть принят как от пользователя, так и от другого управляющего узла. При создании вычислительного узла (далее ВУ) происходит проверка наличия его на текущем управляющем узле (далее УУ) и объединённых с ним УУ. Если не на одном узле он не был найден, то он создаётся на текущем вычислительном узле, иначе сообщение об ошибке. Удаление и Выполнение происходит примерно одинаково. Происходит наличие искомого ВУ на данном УУ, если его нет, то отправляются запросы на все УУ, объединенные с текущим УУ. Если и на них не найден искомый ВУ, то оповещается сообщение об ошибке, иначе результат. При объединении с другим УУ текущий УУ пытается сначала занять порт, который должен быть занят другим УУ, если успешно, то это означает, что такого УУ не существует, если неудачно, то на данный узел посылается запрос, если ответ не получается, то это тоже не наш УУ или он недоступен, иначе ответ принимается и два УУ объединяются. Пинг всех проиходит последовательно, сначала на текущем УУ проверяются ВУ, затем посылаются запросы на объединённые УУ, и выводится результат проверки.

4. Демонстрация работы программы.

териминал 1:

schizo@VivoBook:~/labs/2nd\_kurs/OSI/6lab/src$ ./controller 8090

[8090]>>create 1 1 //создание ВУ 1 (1)

[8090]>>Ok: 3506

[8090]>>remove 1 // удаление ВУ 1 (2)

[8090]>>OK: 1 was removed

[8090]>>create 1 1 // создание ВУ 1 (3)

[8090]>>Ok: 3511

[8090]>>exec 1 3 1 -2 3 // Сложение 1+(-2)+3 на ВУ 1 (4)

[8090]>>OK:1:2

[8090]>>UNION WITH 8080

[8090]>>exec 1 1 1 //запрос на сложение на ВУ 1 (7)

[8090]>>ERROR:1: NOT FOUND

[8090]>>create 1 1 //создание ВУ 1 (9)

[8090]>>ERROR: 1 ALREADY EXISTS // он создан на УУ 8080

[8090]>>exec 1 1 1 // запрос на 1 (10)

[8090]>>OK:1:1

[8090]>>create 11 1 //создание узла для проверки (11)

[8090]>>Ok: 3535

[8090]>>create 111 1 //создание узла для проверки (12)

[8090]>>Ok: 3540

[8090]>>^Z

[1]+ Остановлен ./controller 8090

schizo@VivoBook:~/labs/2nd\_kurs/OSI/6lab/src$ kill 3535

// убийство ВУ 11 (15)

schizo@VivoBook:~/labs/2nd\_kurs/OSI/6lab/src$ kill 3550

// убийство ВУ 22 (16)

schizo@VivoBook:~/labs/2nd\_kurs/OSI/6lab/src$ fg 1

./controller 8090

pingall // проверка доступности узлов (17)

[8090]>>OK: 11;22

[8090]>>exit //выход (18)

терминал 2:

schizo@VivoBook:~/labs/2nd\_kurs/OSI/6lab/src$ ./controller 8080

[8080]>>union 8090 //объединение с УУ 8090 (5)

[8080]>>OK: UNION 8090

[8080]>>remove 1 // удаление ВУ 1 на УУ 8090 (6)

[8080]>>OK: 1 was removed

[8080]>>create 1 1 //создание ВУ 1 (8)

[8080]>>Ok: 3526

[8080]>>create 2 2 //создание узла для проверки (13)

[8080]>>Ok: 3545

[8080]>>create 22 2 //создание узла для проверки (14)

[8080]>>Ok: 3550

[8080]>>exit //выход (19)

**5. Вывод**

Научился работать с очередями сообщений и асинхронной обработкой запросов.