Министерство образования Республики Молдова

Технический университет Молдовы

Факультет Вычислительной техники, Информатики и Микроэлектроники

Программирование в сети

Отчет по лабораторной работе №2

Тема: «Параллелизм, многопоточность, асинхронность»

Выполнил Коваль Роман

ст. гр. TI-196:

Проверила: Ангилоглу Анна

Кишинев 2022

**Теория**

Есть два термина, которые необходимо знать:

**Поток:** Поток является независимой или основной единицей процесса.

**Процесс:** Программа, которая выполняется, называется процессом, в процессе существует несколько потоков.

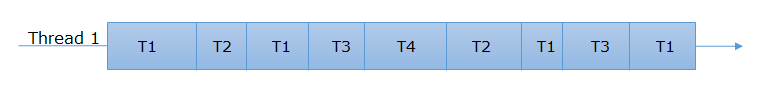
В зависимости от построения программы, различают разные принципы её выполнения:

**Однопоточность –** принцип построения программы, при котором различные функциональные блоки в одном потоке выполняются последовательно, каждый ожидая своей очереди.

Рисунок 1 – сценарий однопоточного синхронного выполнения процесса



Рисунок 2 – сценарий однопоточного асинхронного выполнения процесса



**Многопоточность** – принцип построения программы, при котором различные функциональные блоки в потоках выполняются одновременно, независимо друг от друга.

Выполнение в многопоточности является синхронным и асинхронным.

Рисунок 3 – сценарий многопоточного синхронного выполнения процесса

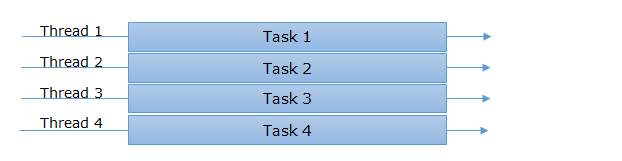
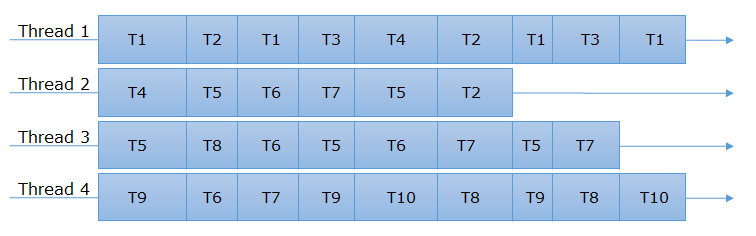


Рисунок 4 – сценарий многопоточного асинхронного выполнения процесса



**Параллелизм –** сценарий, при котором задача разбивается на несколько независимых подзадач, распределяется по потокам и объединяет их результат на выходе.

**Ход работы:**

**Постановка задачи**

Реализовать многопоточность в Node JS. Показать работу нескольких потоков(синхронно или асинхронно). Представить схему потоков, код и результат выполнения программы.

Чтобы реализовать многопоточность в однопоточном Node JS, будет использована библиотека, написанная на TypeScript – worker\_threads. Данная библиотека позволяет разработчику добавить в проект написанный на JS, более одного потока, если это необходимо.

Рисунок 5 – схема потоков в приложении

Рисунок 6 – реализация потоков в Node JS, используя компонент worker\_threads

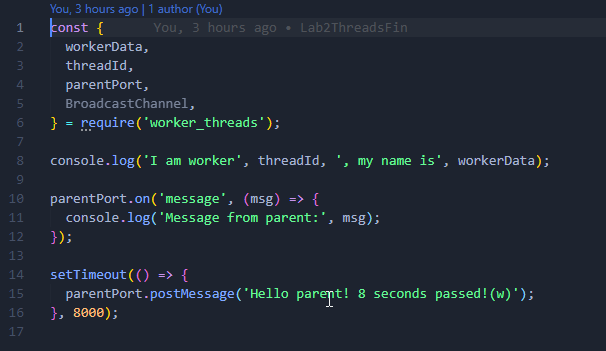
На изображении выше видно, что каждый поток инициализируется с своим идентификатором и некоторыми данными, которые могут быть данными любого типа.

Рисунок 7 – отправка сообщений второстепенным потокам из основного



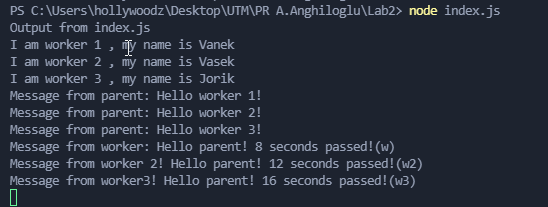
Здесь происходит отправка сообщений из основного потока, а в логика каждого из них есть обработчик, который их перехватывает и отображает в консоль. Красным выделены как раз таки сообщения, а всё остальное – это обработчик сообщений от самих второстепенных потоков родителю.

Рисунок 8 – логика одного из потоков



Тут можно заметить, что изначально, поток представляет себя, затем обрабатывает сообщение от родителя и посылает в ответ своё, с задержкой в 8 секунд. Это сообщение, в свою очередь уже обрабатывается в основном потоке, о чём выше я уже упомянул.

Рисунок 9 – выполнение программы и результат



Первые четыре строчки здесь печатаются практически одновременно, что говорит нам о одновременном асинхронном запуске всех четырех потоков. Далее в ход идут таймеры и последовательно выводят три сообщения из каждого второстепенного потока. Завершается вывод тремя сообщениями из основного потока, которые так же отражают связь главного потока и второстепенных.

**Вывод:** при проектировании современных программ все больше используется асинхронное программирование, чтобы программа имела возможность выполнять несколько операций в конкретный момент времени. Многие разработчики считают, что асинхронный сценарий выполнения продукта – это будущее и я с этим согласен.