**Многопоточное программирование. Потоки. Асинхронное программирование.**

Многопоточное программирование относится к технике создания и запуска нескольких потоков в рамках одной программы. Эти потоки могут работать одновременно, выполняя несколько задач одновременно, и могут помочь повысить производительность и эффективность программы.

В одно процессной однопоточной среде все запросы, например, к интерфейсу ввода-вывода, блокируют любое дальнейшее продвижение процесса. Любая комбинация многопроцессорной или многопоточной реализации программы позволяет пользователю приложения запрашивать несколько файлов, веб-страниц, медиаресурсов одновременно.

В качестве примера рассмотрим взаимодействие пользователя с веб-сайтом. Поскольку ожидается, что клиент World-Wide Web будет использовать большую часть времени выполнения для операций ввода-вывода, таких как «подключение» и «чтение», можно получить высокую степень оптимизации, если несколько потоков могут выполняться одновременно. Параллельное исполнение — это строгое подмножество конкурентного исполнения. Это значит, что на компьютере с одним процессором параллельное программирование – невозможно [1].

Поток — это облегченная единица выполнения, которая может выполняться одновременно с другими потоками в программе. Потоки используются для одновременного выполнения нескольких задач и совместно используют одно и то же пространство памяти, что упрощает обмен данными между ними. Как правило, потоки управляются операционной системой, а их выполнение запланировано планировщиком.

Изображение выглядит как диаграмма, Прямоугольник, текст, линия

Автоматически созданное описание

Синхронизация потоков важна в многопоточном программировании, чтобы избежать состояния гонки (race condition) и обеспечить безопасный и согласованный доступ потоков к общим данным. Такие методы, как блокировка, мониторинг, семафор и мьютекс, могут использоваться для реализации синхронизации потоков и предотвращения одновременного доступа потоков к общим ресурсам, что приводит к повреждению данных, несоответствиям и другим проблемам.

**В качестве примера использования приведу использование семафора при разработке своего проекта. Я использовал семафор при реализации метода, работающего с общими ресурсами. Метод асинхронно закрывает открытый в данный момент виджет, и если во время закрытия виджета другой поток снова вызовет этот метод, то возникнет ошибка.**

**Реализация на языке C# с использованием ключевого слова lock\*Реализация на языке C# с использованием ключевого слова lock:**

**private static readonly object BalanceLock = new();**

**public async Tak AsyncHide()**

**{**

**Widget currentWidget = widgetQueue.Deque();**

**lock (BalanceLock)**

**{**

**await InternalAsyncHide(currentWidget);**

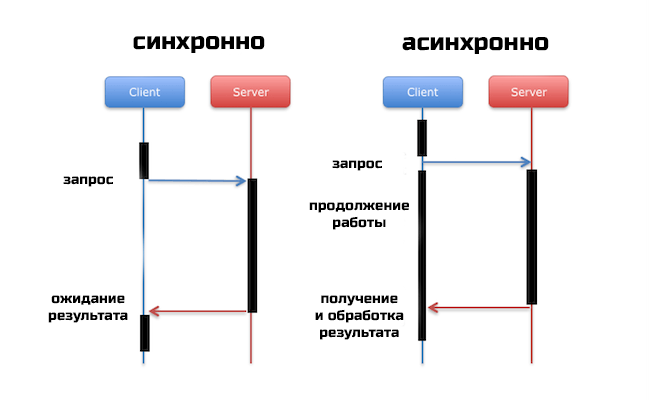
**}**

**}**

Пул потоков:

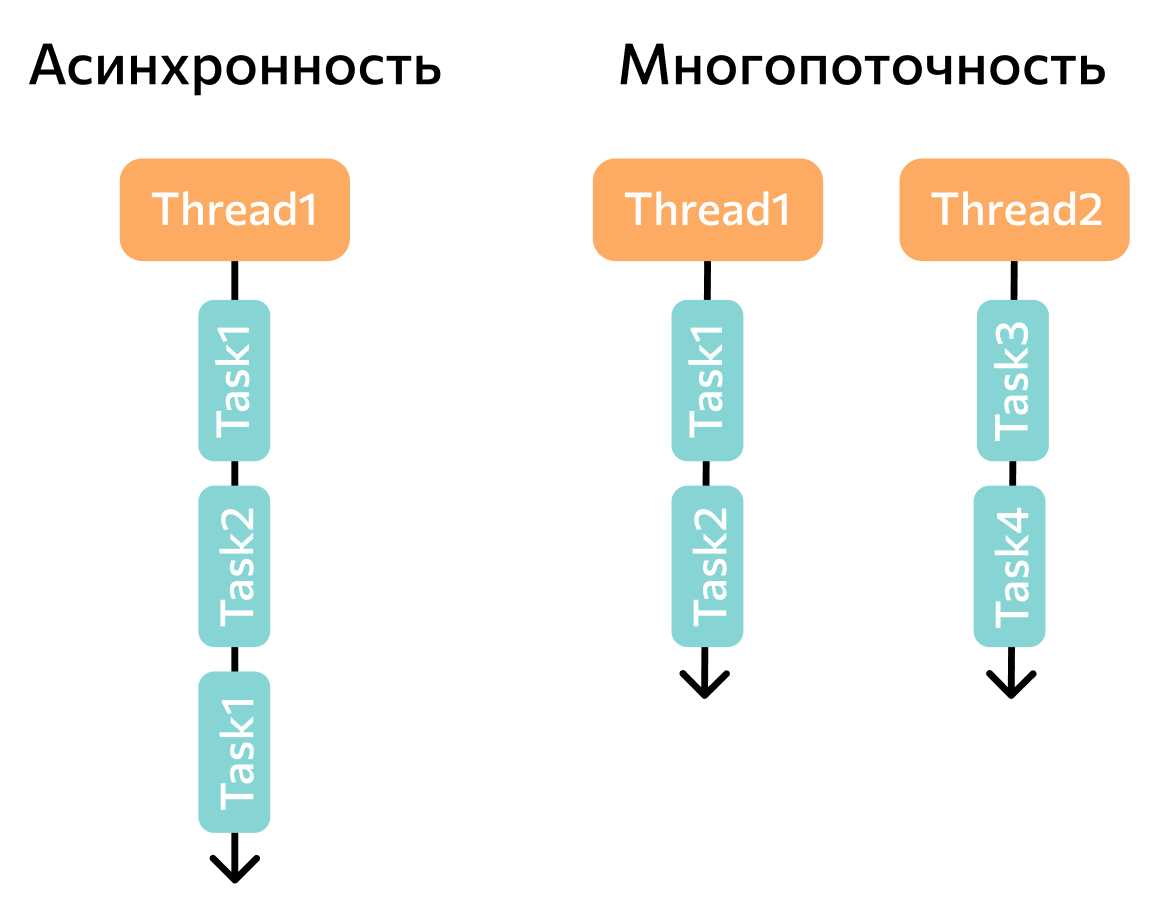
Пул потоков — это набор потоков, которые можно использовать для одновременного выполнения задач в программе. Пул потоков управляет фиксированным количеством потоков и может выделять их задачам по мере необходимости. Пулы потоков полезны для повышения производительности и масштабируемости программы, поскольку они могут повторно использовать существующие потоки вместо создания новых для каждой задачи.

Асинхронное программирование — это тип программирования, который позволяет программе эффективно выполнять задачи неблокирующим образом. Это означает, что вместо того, чтобы ждать завершения задачи перед переходом к следующей, программа может продолжать выполнять другие задачи, пока первая выполняется в фоновом режиме. Этот метод полезен для задач, выполнение которых занимает много времени, таких как чтение или запись данных из файла или выполнение сложных вычислений.



Асинхронное программирование может быть реализовано с использованием различных методов, таких как обратные вызовы, промисы и асинхронность/ожидание. Эти методы позволяют программе планировать задачи и обрабатывать их завершение асинхронно, что делает ее более быстрой и эффективной.

При этом асинхронная операция не предполагает использование многопоточности, ведь асинхронная операция может быть выполнена на удалённом устройстве: веб-узлом, сервером, другим устройством за пределами текущего вычислительного устройства. Отличие асинхронной операции от многопоточности изображено далее:



**При разработке своего проекта VR-симуляции я реализовал асинхронную операцию с продолжением работы программы. Программа асинхронно ожидает ввод пользователя на клавиатуре и реагирует созданием и открытием виджета на экране пользователя, при этом основной поток программы не блокируется и продолжает свою работу.**

**Реализация на языке C#:**

**public Task GetUserInput()**

**{**

**CancellationTokenSource token = new();**

**wigetButton.onClick.addListener(token.TrySetResult);**

**return token.Task;**

**}**

В заключение можно сказать, что многопоточное программирование — это мощная техника, которая позволяет программам выполнять несколько задач одновременно и повышать их производительность и эффективность. Потоки, асинхронное программирование, синхронизация потоков и пулы потоков — это некоторые из важнейших концепций многопоточного программирования, которые помогают обеспечить безопасное и эффективное выполнение параллельных задач. Эти концепции необходимы для разработки высокопроизводительных и быстро реагирующих приложений в современных вычислительных условиях.

Источники:

1. Henrik Frystyk, Multi Threaded Clients, w3.org - https://www.w3.org/People/Frystyk/thesis/multithread.html

2. Microsoft, Рекомендации по работе с потоками - <https://learn.microsoft.com/ru-ru/dotnet/standard/threading/managed-threading-best-practices>

3. Информатика и программирование. Основы информатики - Н. И. Парфилова.