курс development 12 недель

iOS: разработка приложений с 0

r_d

iOS программа курса 20 занятий

1
Swift: начало

2
ООП: основы

3
Создание iOS-приложения в Xcode

4
Создание интерфейса iOS-приложения

5
Динамические интерфейсы, часть 1

6
Динамические интерфейсы, часть 2

iOS программа курса 20 занятий

7

Динамические интерфейсы, часть 3

8

Навигация в приложении, часть 1

9

Навигация в приложении, часть 2

10

Анимации в iOS

11

Работа с памятью в iOS

12

Многозадачность в iOS, часть 1

iOS программа курса 20 занятий

13

Многозадачность в iOS, часть 2

14

Дебаг iOS-приложения

15

Тестирование

16

Хранение данных в приложении

17

Работа с сетью в приложении

18

Сборка приложения

iOS программа курса 20 занятий

19

Современные архитектуры для iOS приложений

20

Защита курсовых проектов

iOS | план занятия | занятие 12

Многозадачность в iOS. Часть 1

- Что такое параллелизм
- thread как первая абстракция
- queue как вторая абстракция (GCD)
- Синхронные и асинхронные операции
- Demo 

r_d

Что такое параллелизм

Более 150 лет назад ученые в Computer Science стали искать решения, как заставить аналитическую машину выдавать несколько результатов одновременно.

С момента создания компьютера и роста вычислительных мощностей было найдено множество способов того, как заставить компьютер “одновременно” считать результаты операций.

Но давайте определимся с терминологией.

Что такое параллелизм

“Параллельные вычисления — способ организации компьютерных вычислений, при котором программы разрабатываются как набор взаимодействующих вычислительных процессов, работающих параллельно (одновременно).” (с)

Что такое параллелизм

Термин определяет сразу множество способов реализации таких вычислений, на разных уровнях:

- Многопроцессорные системы — одна большая задача разбивается на набор задач поменьше и много CPUs одновременно решают эти подзадачи.
- Многоядерные системы — одна большая задача разбивается на такой же набор задач (см. пред. пункт), но в этот раз все они вычисляются на ядрах одного CPU.
- Многопоточные системы — еще раз все тоже самое, только теперь мы решаем ту же задачу на одном ядре одного CPU.

Что такое параллелизм

На курсе мы с вами будет говорить только про последний тип параллелизма — многопоточность. Что же это такое?

Многопоточность — способность центрального процессора (CPU) или одного ядра в многоядерной процессоре одновременно выполнять несколько процессов или потоков, соответствующим образом поддерживаемых операционной системой.

А что же такое процессы и потоки?

thread как первая абстракция

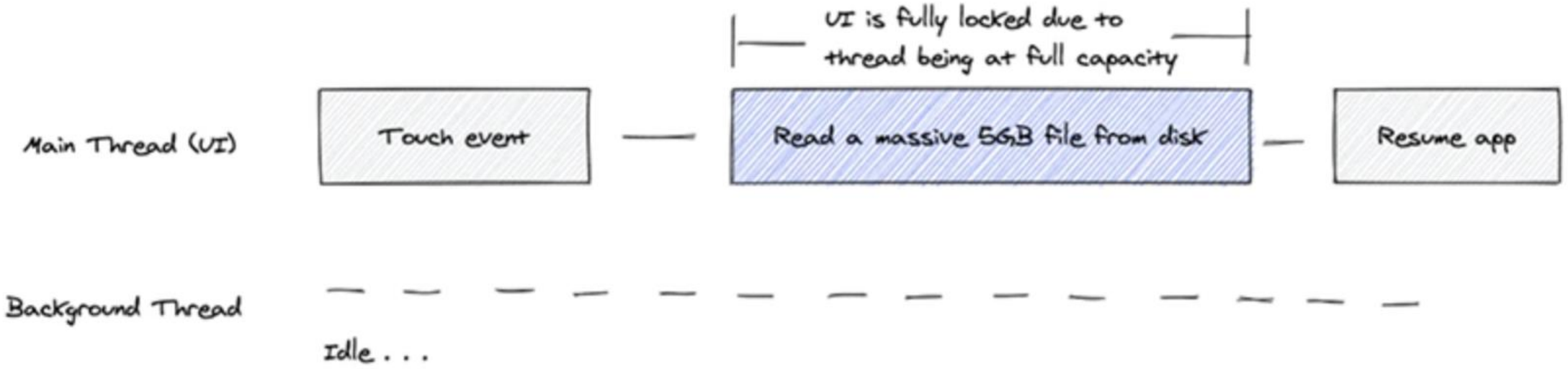
- Процесс — это конкретная программа, которая либо выполняется в данный момент, либо (в рамках современных ОС) находится в ожидании. Каждое приложение, запущенное вами на вашем iPhone — это отдельный процесс.
- Важно понимать, что всё управление по работе с процессами берёт на себя операционная система!
- Каждый процесс — независимая единица, которая тратит ресурсы CPU и под которую выделяется определенное количество памяти. Но всем этим управляет ОС, а не вы :)

thread как первая абстракция

- Поток (thread) — объект внутри процесса, который использует все доступные процессу ресурсы (CPU, память).
- Главное отличие потока от процесса — при создании процесса создаётся один поток (часто зовут основным), но во время работы процесса могут создаваться новые потоки. И все потоки внутри одного процесса будут пользоваться его общими ресурсами.
- Поток — это абстракция, которая помогает создавать “видимость” одновременного выполнения множества подзадач одной большой задачи.

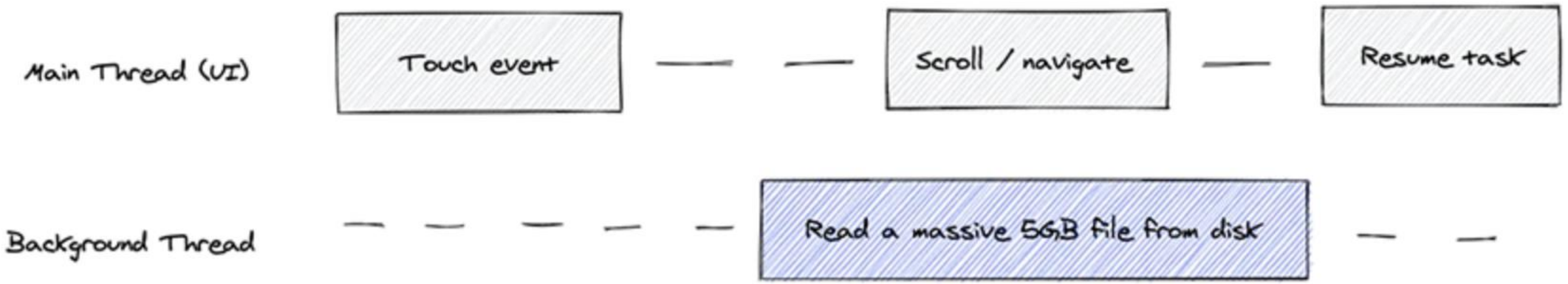
thread как первая абстракция

Теперь визуализируем простой пример — приложение Apple Music, где процесс — само приложение.



thread как первая абстракция

- Один поток даёт возможность обрабатывать события от пользователя и реагировать на них.
- Другой — скачивает музыку вам на iPhone.



queue как вторая абстракция (GCD)

Начиная с версии iOS 2.0, Apple предоставляет специальный класс NSThread, который позволяет разработчику реализовать подобный функционал, как мы видели на предыдущей картинке, но есть одно “но”.

The Move Away from Threads

Although threads have been around for many years and continue to have their uses, they do not solve the general problem of executing multiple tasks in a
of you, the developer. You have to decide how many threads to create and adjust that number dynamically as system conditions change. Another problem i
it uses.

Instead of relying on threads, OS X and iOS take an *asynchronous design approach* to solving the concurrency problem. Asynchronous functions have been
time, such as reading data from the disk. When called, an asynchronous function does some work behind the scenes to start a task running but returns bef
starting the desired task on that thread, and then sending a notification to the caller (usually through a callback function) when the task is done. In the pas
asynchronous function and create your own threads. But now, OS X and iOS provide technologies to allow you to perform any task asynchronously without

One of the technologies for starting tasks asynchronously is *Grand Central Dispatch (GCD)*. This technology takes the thread management code you would

queue как вторая абстракция (GCD)

- Apple говорит о том, что хотя потоки и могут быть использованы для написания многопоточного кода и для решения сложных задач, вся ответственность по управлению потоками в рамках ОС лежит на плечах разработчиков. Порог входа для начинающих относительно высокий.
- Вместо этого Apple предлагает использовать другой, “асинхронный” подход к решению многопоточных задач.

queue как вторая абстракция (GCD)

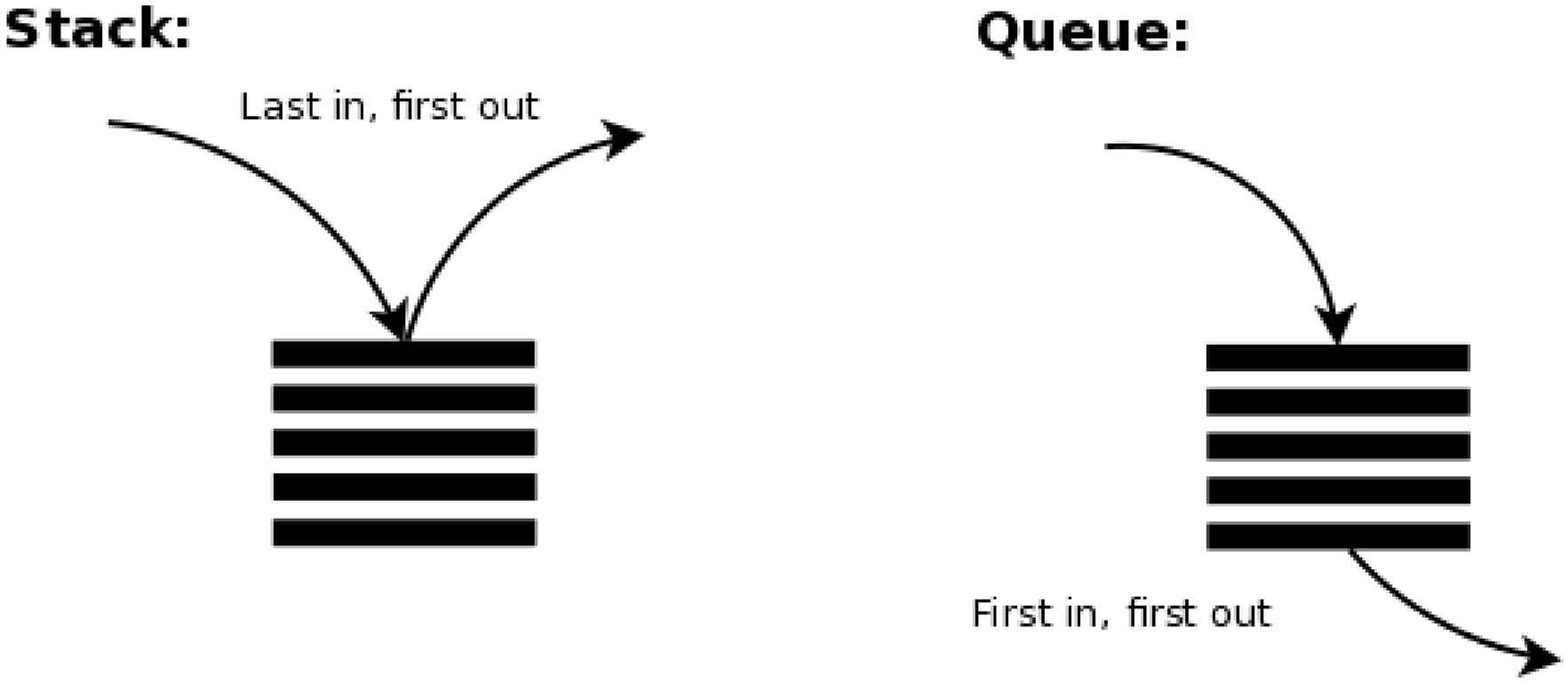
- Apple разработала Grand Central Dispatch — инструмент (библиотека), функциональность которой строится на идее “очереди” из задач, которые создаёт и которыми управляет разработчик во время создания программы.
- Эта библиотека забирает у нас необходимость создавать свои собственные объекты NSThread и оперировать такой низкоуровневой абстракцией, как потоки.
- Вместо NSThread в библиотеке лежит набор классов, которые позволяют в “более человечной форме” оперировать задачами, которые мы хотим выполнять параллельно.

queue как вторая абстракция (GCD)

- Центральный класс (и экземпляры этого класс), который и представляет собой новую абстракцию над обычными потоками называется DispatchQueue.
- DispatchQueue — объект, который занимается управлением поступающих ему задач (последовательно либо параллельно) на одном из заданных потоков.
- Существует заданный по-умолчанию набор потоков, которые мы можем использовать в коде (main thread, background thread, global thread).

queue как вторая абстракция (GCD)

Объект класса DispatchQueue хранит задачи как одноименная структура данных “очередь”.



Синхронные и асинхронные операции

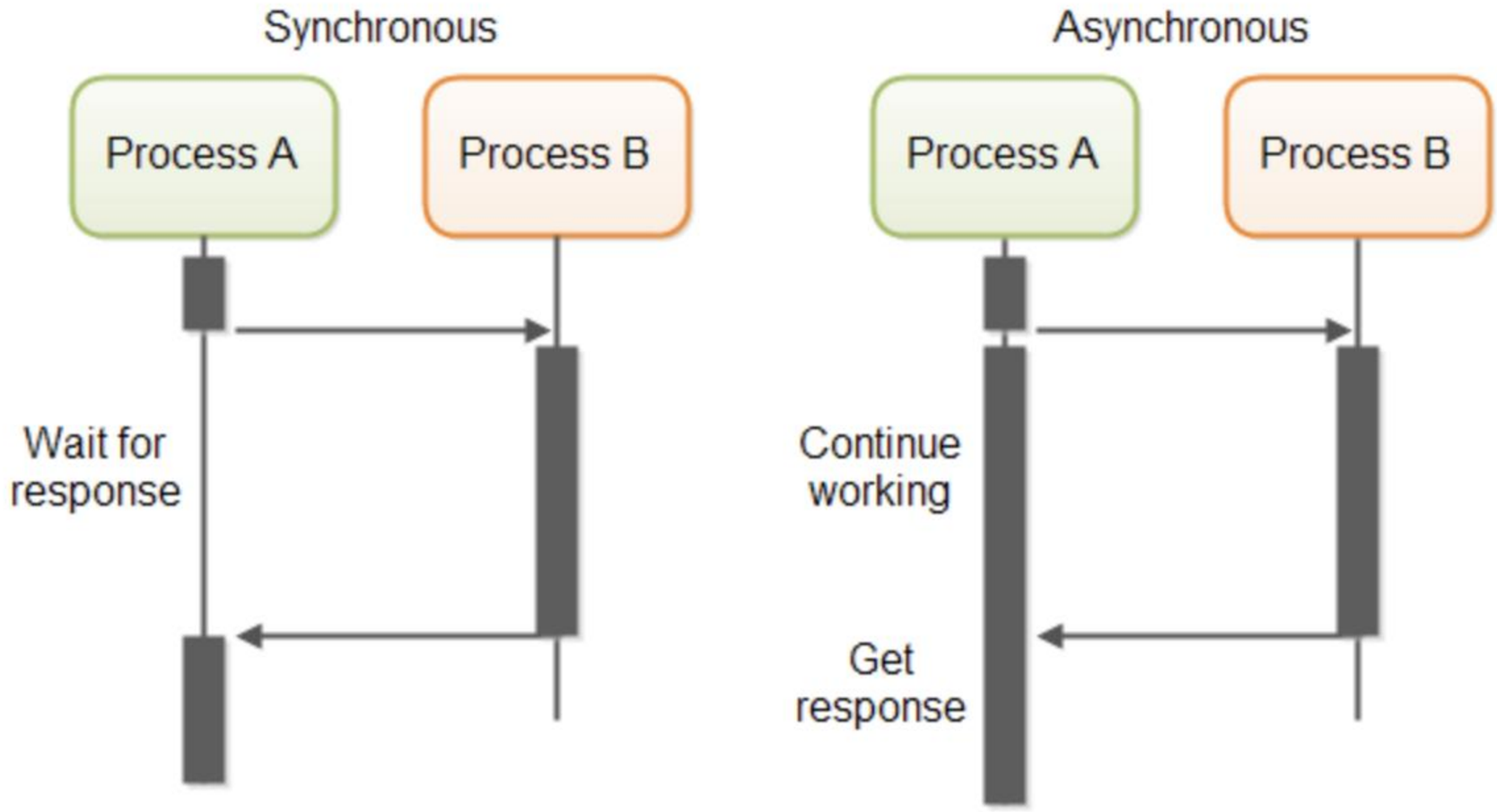
Еще одна очень важная тема в разговоре об очередях (в рамках темы параллелизма в iOS) — понимание синхронных и асинхронных операций. Если простыми словами:

- Когда вы выполняете что-то синхронно, вы ждете его завершения, прежде чем перейти к другой задаче.
- Когда вы выполняете что-то асинхронно, вы можете перейти к другой задаче до ее завершения.

Синхронные и асинхронные операции

- Две синхронные задачи должны быть осведомлены друг о друге, и одна задача должна выполняться каким-то образом, зависящем от другой. Например, задача №1 и задача №2. Когда задача №1 начала выполняться, задача №2 вкурсе про первую, и ждёт, когда задача №1 выполниться.
- Асинхронность означает, что задачи полностью независимы друг от друга. Ни одна из них не должна каким-либо образом учитывать другого — ни при инициализации, ни при исполнении.

Синхронные и асинхронные операции



Синхронные и асинхронные операции

- Важно: все операции по работе с UI пользователя выполняются на главном потоке (main thread)!
- Будьте внимательны, когда выполняете какие-либо операции не на главном потоке, если результат этих операций ваш нужно отобразить на экране.

Demo Time 

спасибо

задавайте вопросы