Работа с асинхронными задачами в iOS

Андрей Анисимов, iOS Разработчик

Давным-давно экран почти любого персонального компьютера выглядел примерно так...

```
Copyright 1988-1991 Microsoft Corp.

Installed A20 handler number 2.
64K High Memory Area is available.

C:\>
C:\>
C:\>command
+--------------------- WARNING! ----------------------+
|                                                     |
|                                                     |
|  The license for this pre-release version of MS-DOS |
|  5.0 has expired.  Please replace it with an updated|
|  version of MS-DOS 5.0 immediately.                 |
|                                                     |
|                                                     |
|          <Press any key to continue>                |
|                                                     |
+-----------------------------------------------------+


Microsoft(R) MS-DOS(R) Version 5.00.490
            (C)Copyright Microsoft Corp 1981-1991.

C:\>_
```

А программист мог сказать. «Многопоточность? Нет, не слышал.»

Основные понятия

- процесс (задача)
- поток
- многопоточность
- многозадачность

Процесс (задача). Это совокупность кода и данных, разделяющих общее виртуальное адресное пространство. Как правило, одна программа это один процесс.

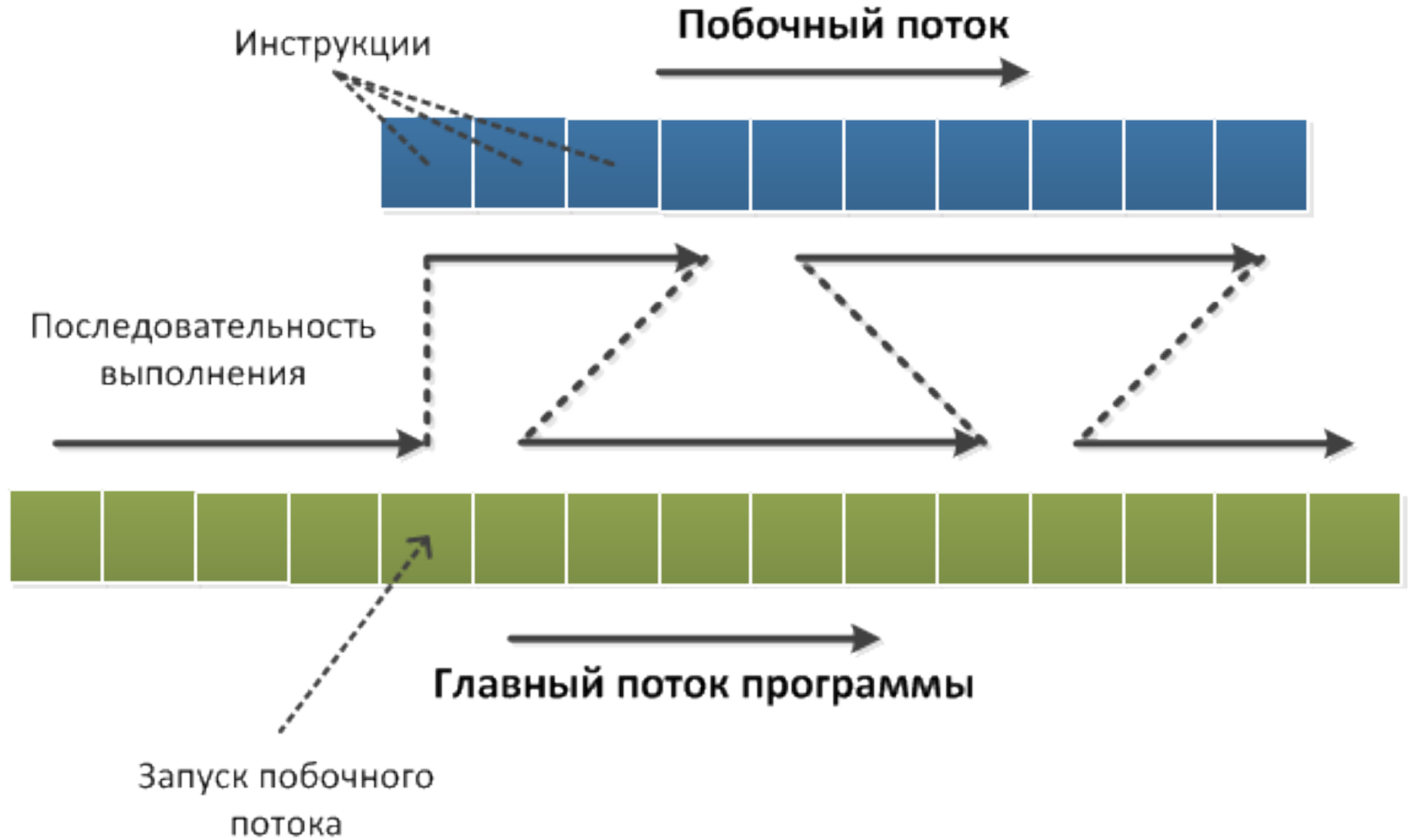
Поток - это одна единица исполнения кода процесса или программы

Любой процесс содержит как минимум один поток, называемый основным или главным.

Многопоточность - свойство операционной системы поддерживать выполнение нескольких потоков у процесса.

Все потоки одного и того же процесса используют общее адресное пространство что дает возможность их взаимодействия, но с другой стороны порождает и трудности.

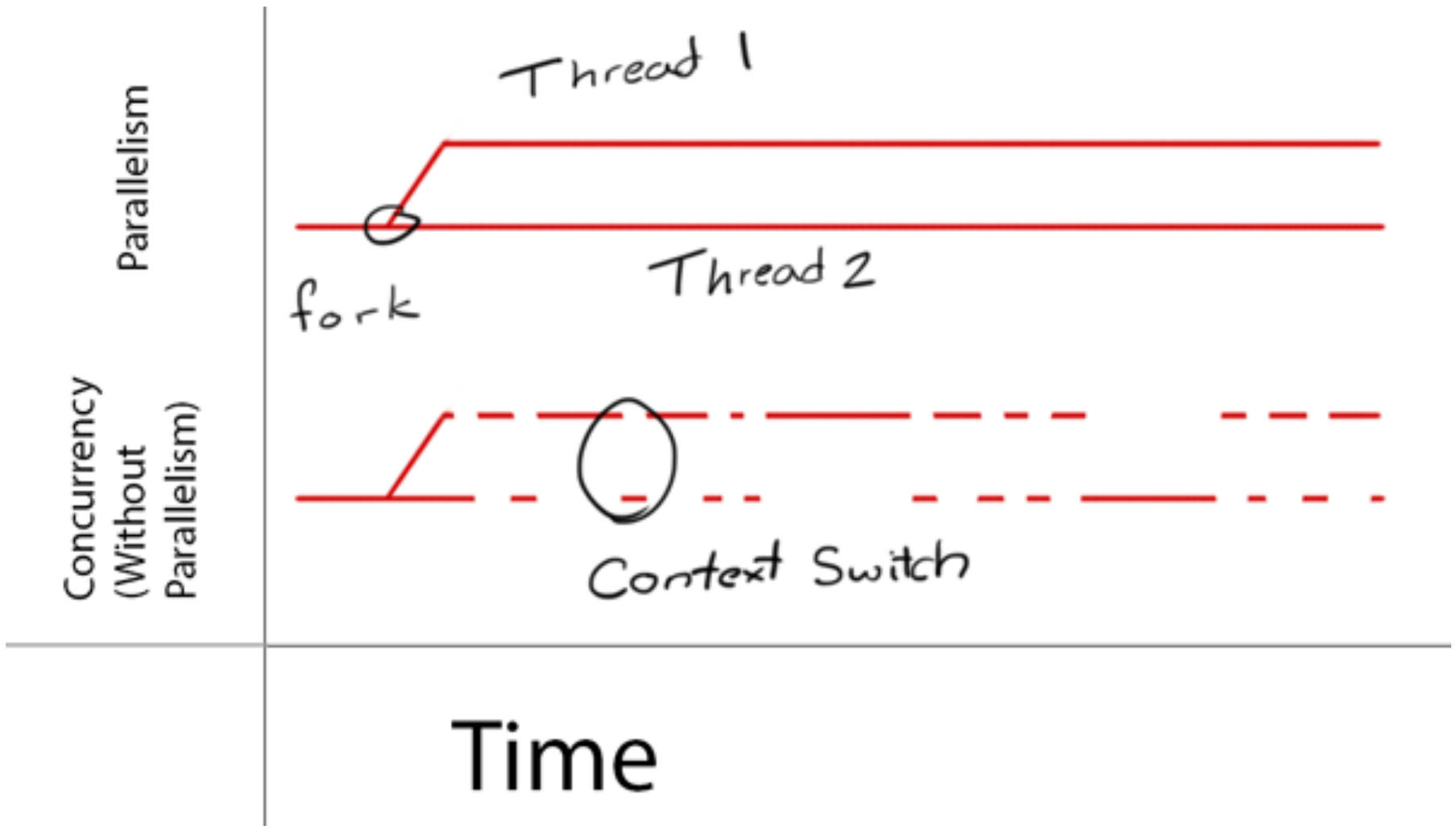
Потоки


Инструкции
Побочный поток
Последовательность выполнения
Главный поток программы
Запуск побочного потока

Многозадачность - это свойство ОС выполнять несколько программ (процессов) параллельно ну или квазипараллельно.

Многопоточность и многозадачность не связаны с количеством процессоров или ядер.

Варианты параллельного выполнения потоков в iOS



Для чего вообще разработчику iOS приложений может понадобиться использовать несколько потоков?

Инструменты в iOS для реализации многопоточности

- Thread
- Grand Central Dispatch
- Operation

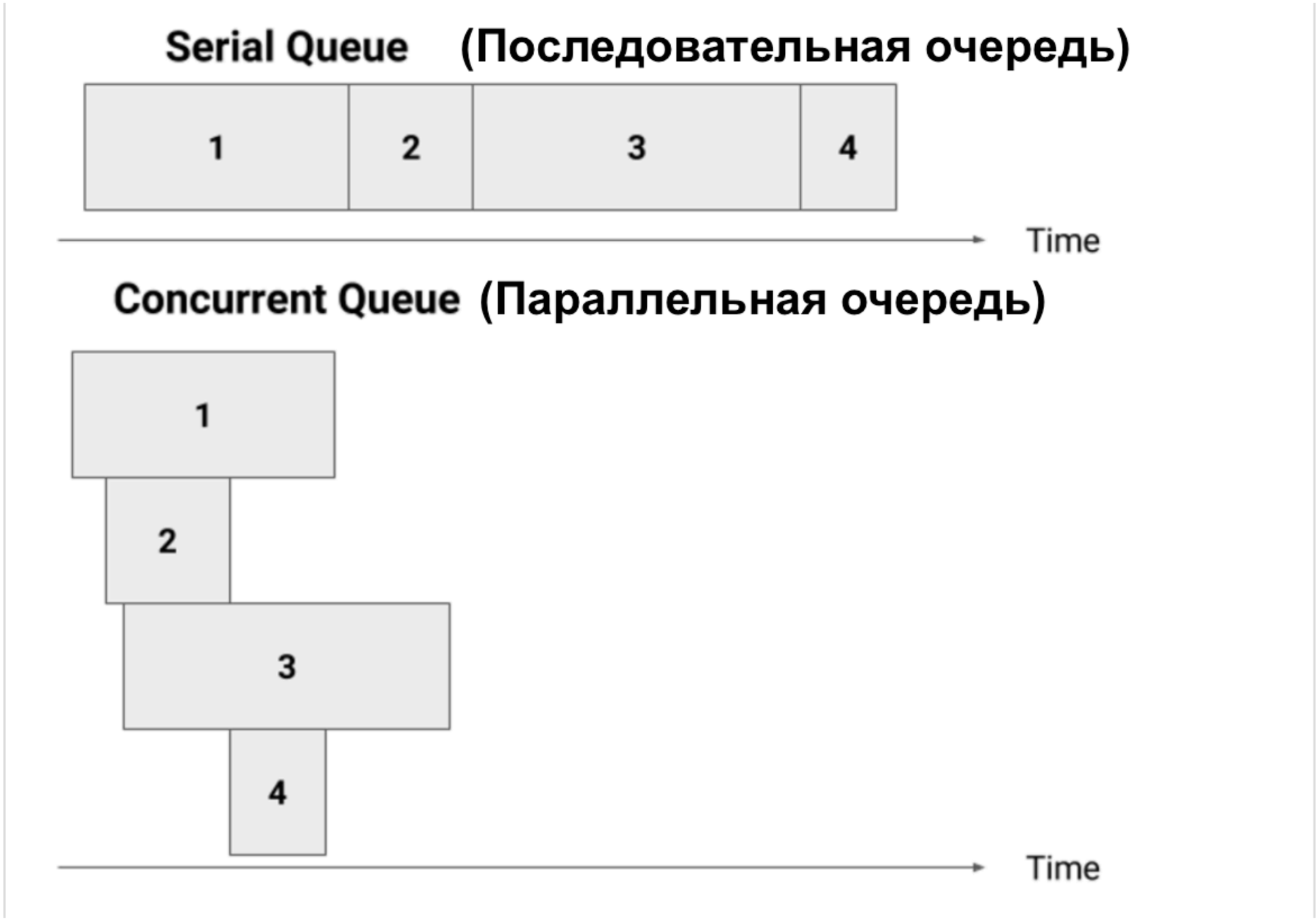
Абстрактные понятия для работы с GCD и Operation

- Очередь (queue)

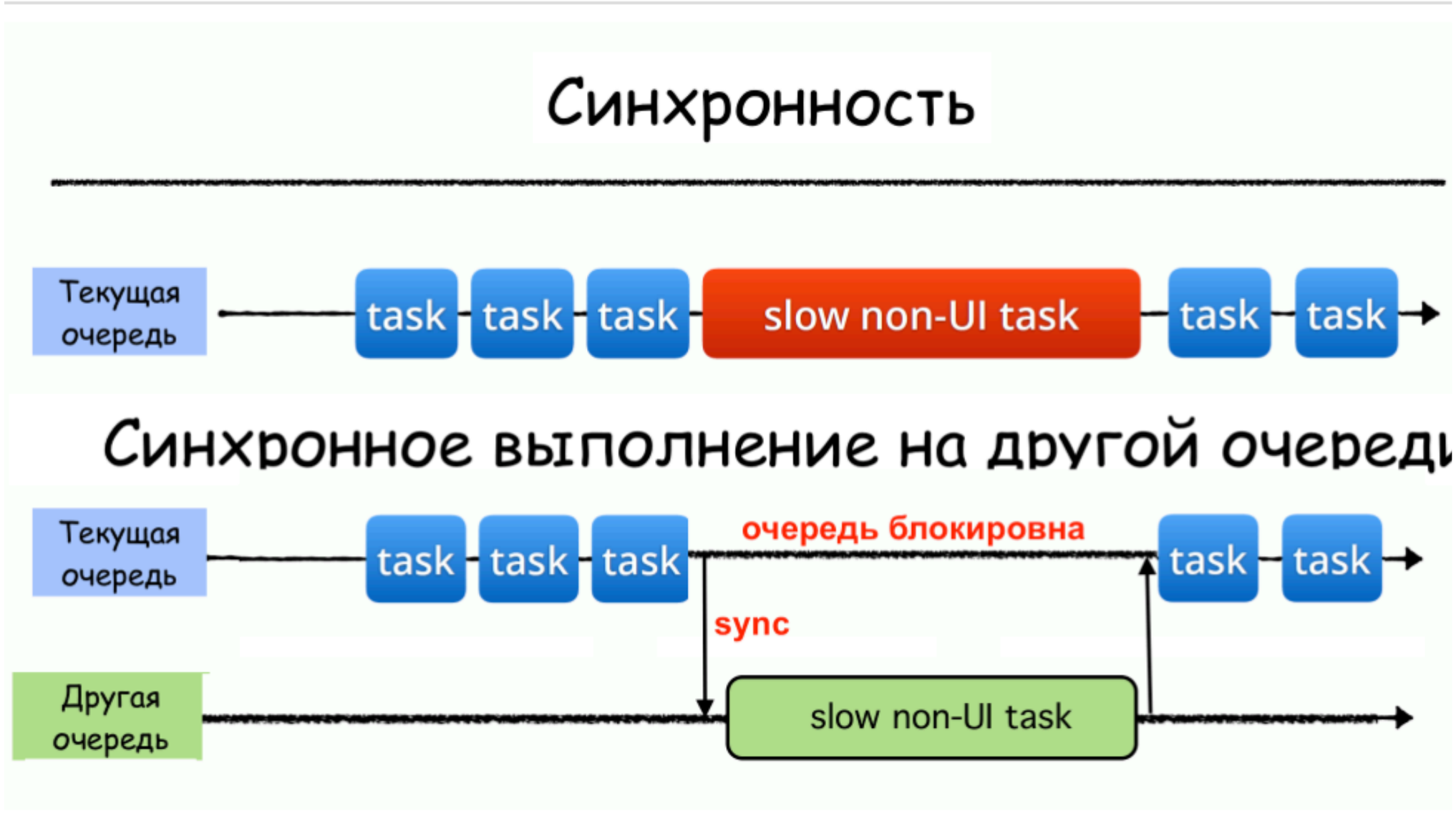
- Задача (task) (анонимный блок кода). Не путать с понятием задачи о котором говорили выше.

Очереди следуют FIFO паттерну (First In, First Out), это означает, что тот, кто первым был поставлен в очередь, будет первым направлен на выполнение. У вас может быть множество очередей и система “выдергивает” замыкания по одному из каждой очереди и запускает их на выполнение в их собственных потоках. Таким образом, вы получаете многопоточность.

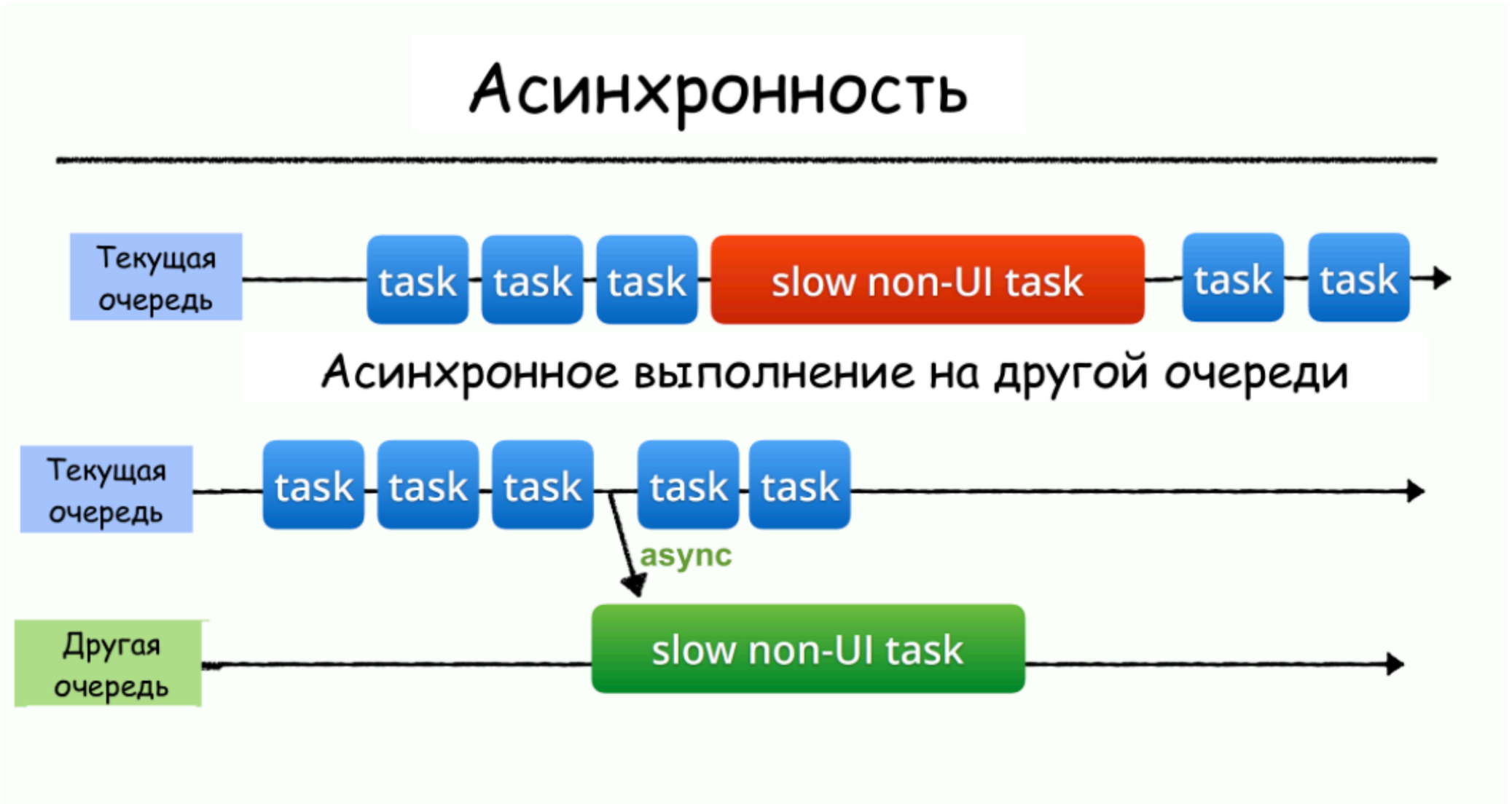
Последовательные и параллельные очереди



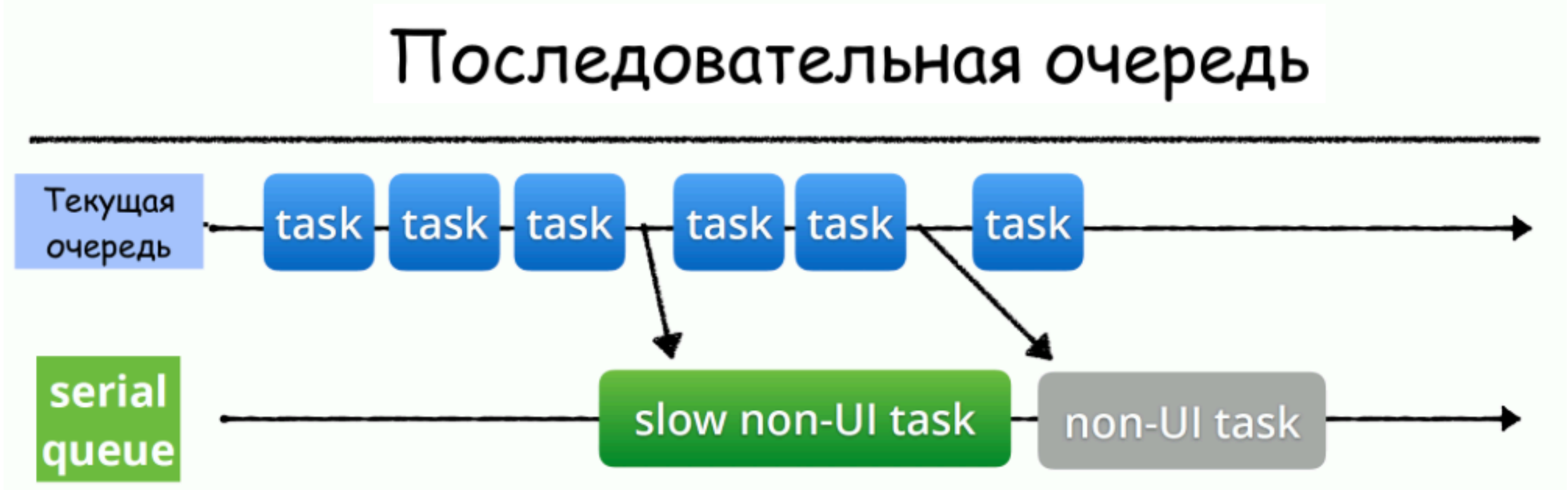
Синхронное выполнение



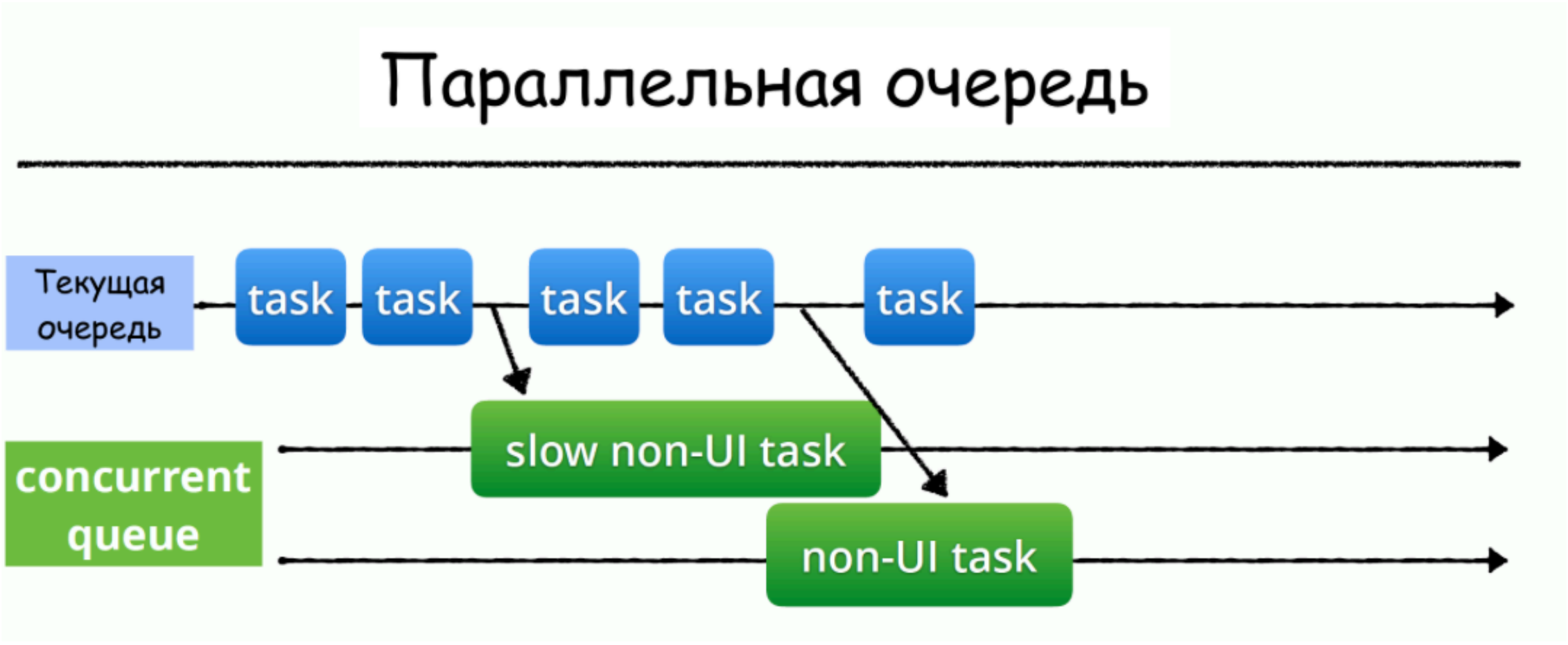
Асинхронное выполнение



«Другая очередь» - последовательная



«Другая очередь» - параллельная

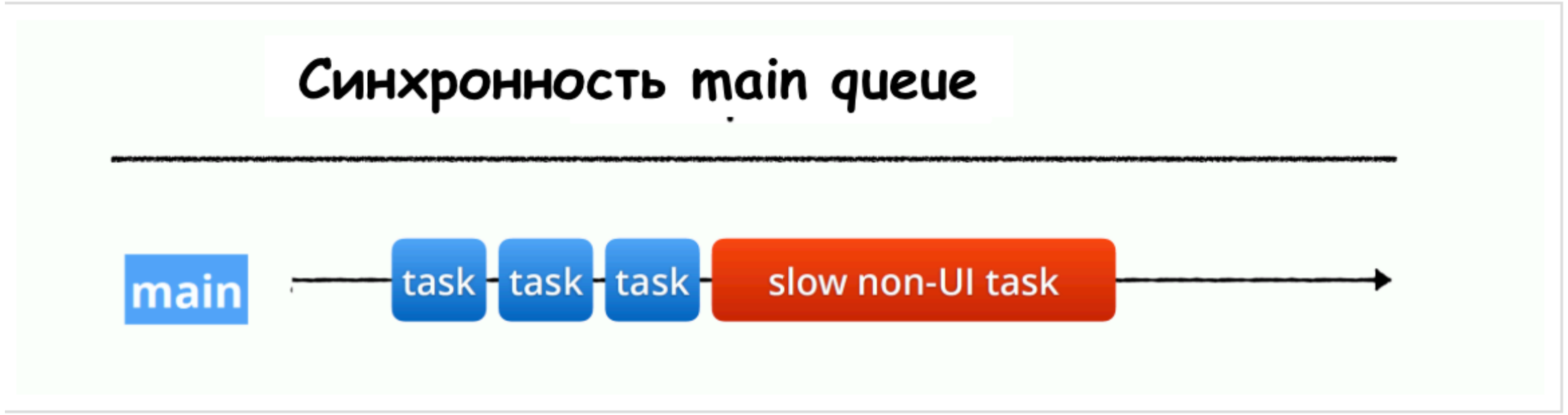


То есть при программировании многопоточности, наша задача как разработчика, сводится к созданию очередей и добавлению на них блоков кода (задач) Далее работает операционная система.

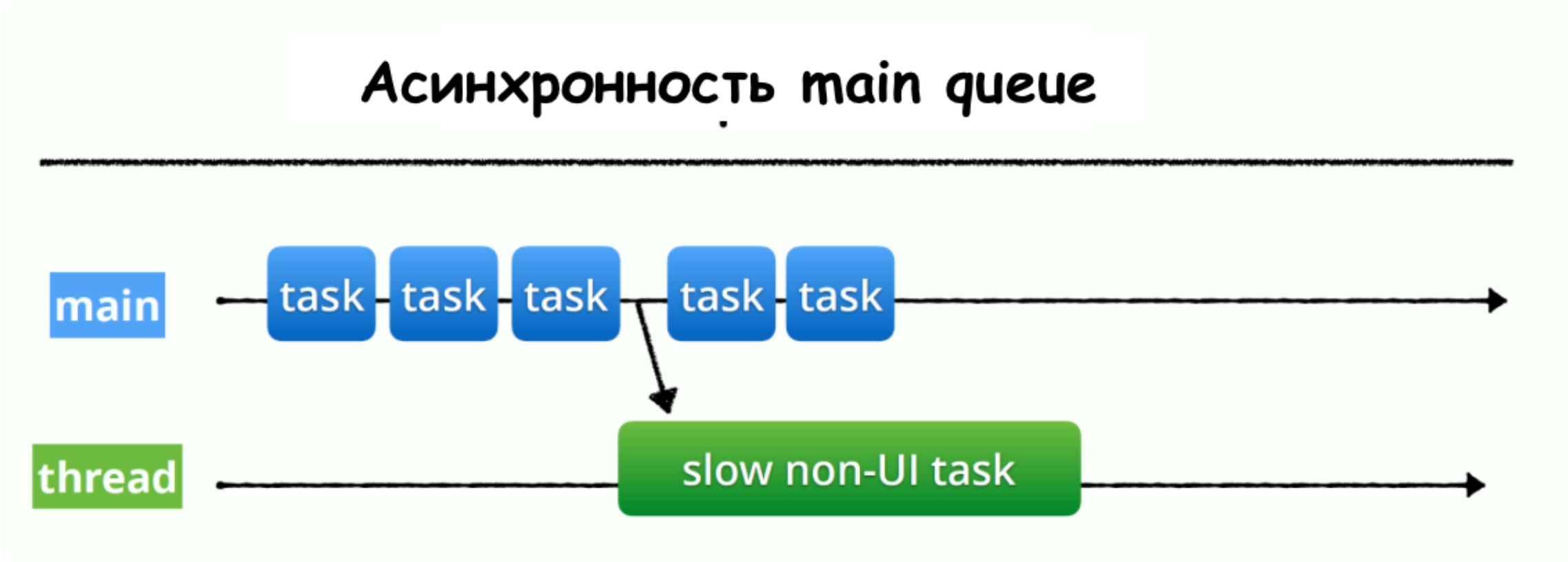
Глобальные очереди

```
// Глобальная последовательная (serial) main queue
let mainQueue = DispatchQueue.main

// Глобальные  concurrent dispatch queues
let userInteractiveQueue = DispatchQueue.global(qos: .userInteractive)
let userQueue = DispatchQueue.global(qos: .userInitiated)
let utilityQueue = DispatchQueue.global(qos:  .utility)
let backgroundQueue = DispatchQueue.global(qos:  .background)

// .default concurrency
let defaultQueue = DispatchQueue.global()
```

Глобальная последовательная очередь для пользовательского интерфейса — Main queue



Нужно уводить затратные по времени операции на другой поток



Очень важно!

Изменять наши UI компоненты нужно только на этом самом основном потоке.

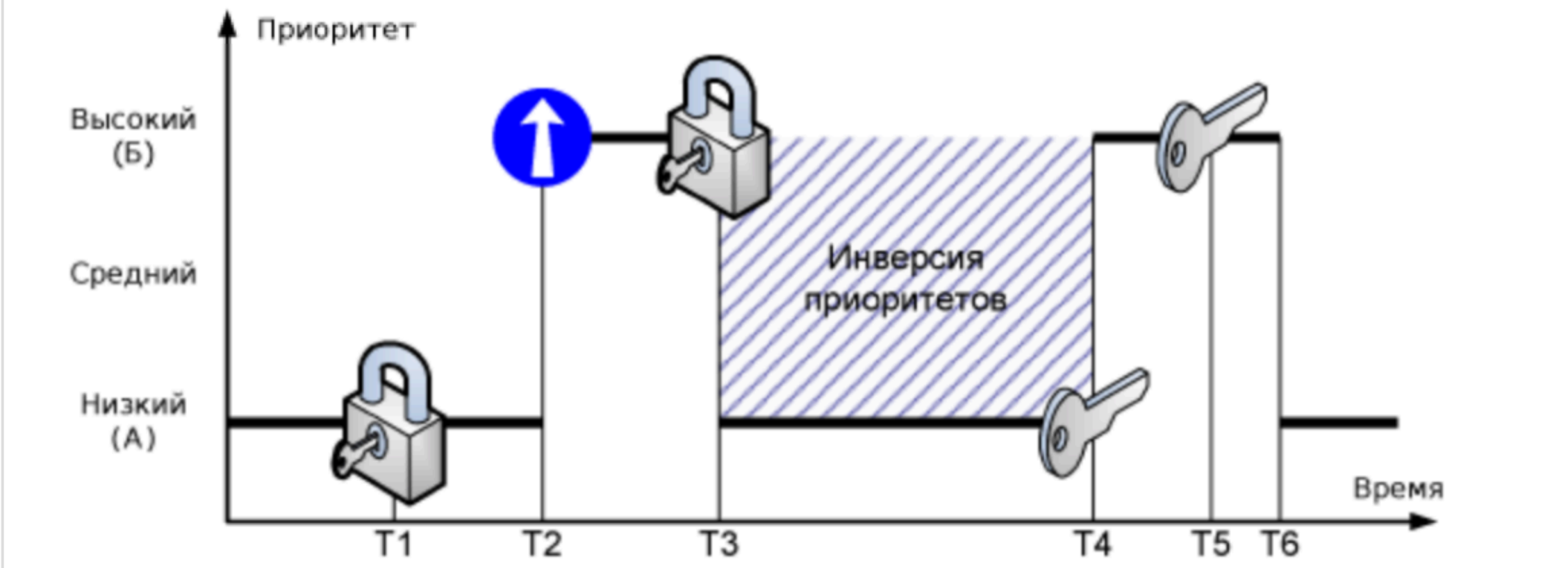
Проблемы многопоточности

- состояние гонки (race condition)
- инверсия приоритетов (priority inversion)
- взаимная блокировка (deadloc)

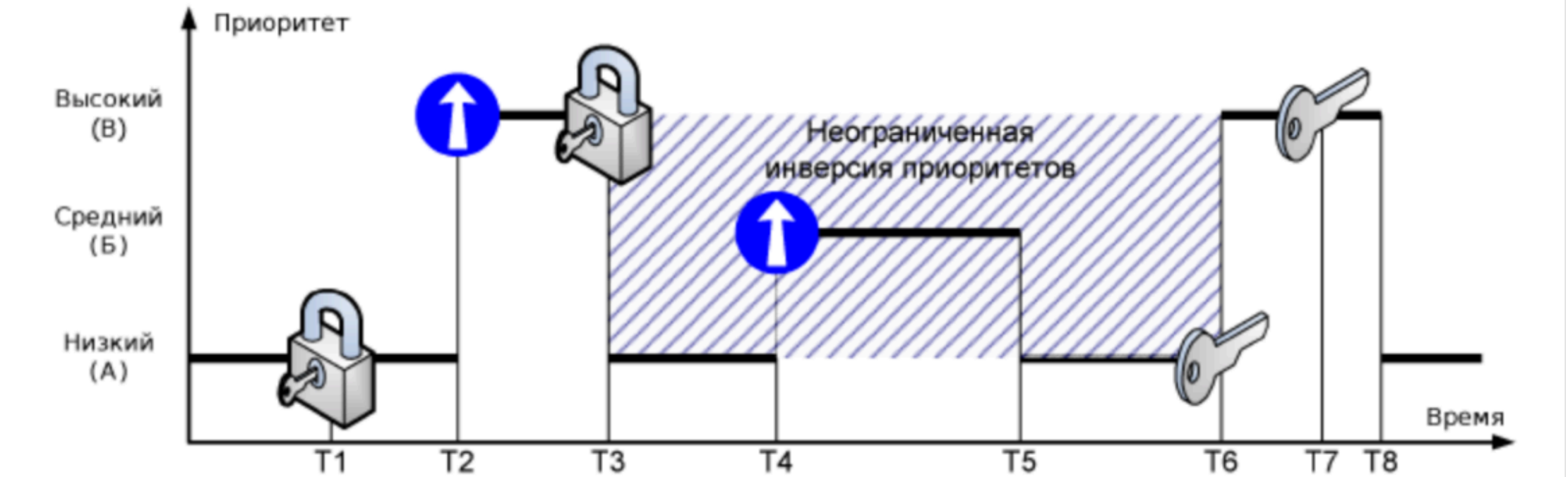
Состояние гонки

```
var value = "😇"

func changeValue() {
    sleep(1)
    value = value + "🐔";
}

mySerialQueue.async {
    changeValue()
}
value
```

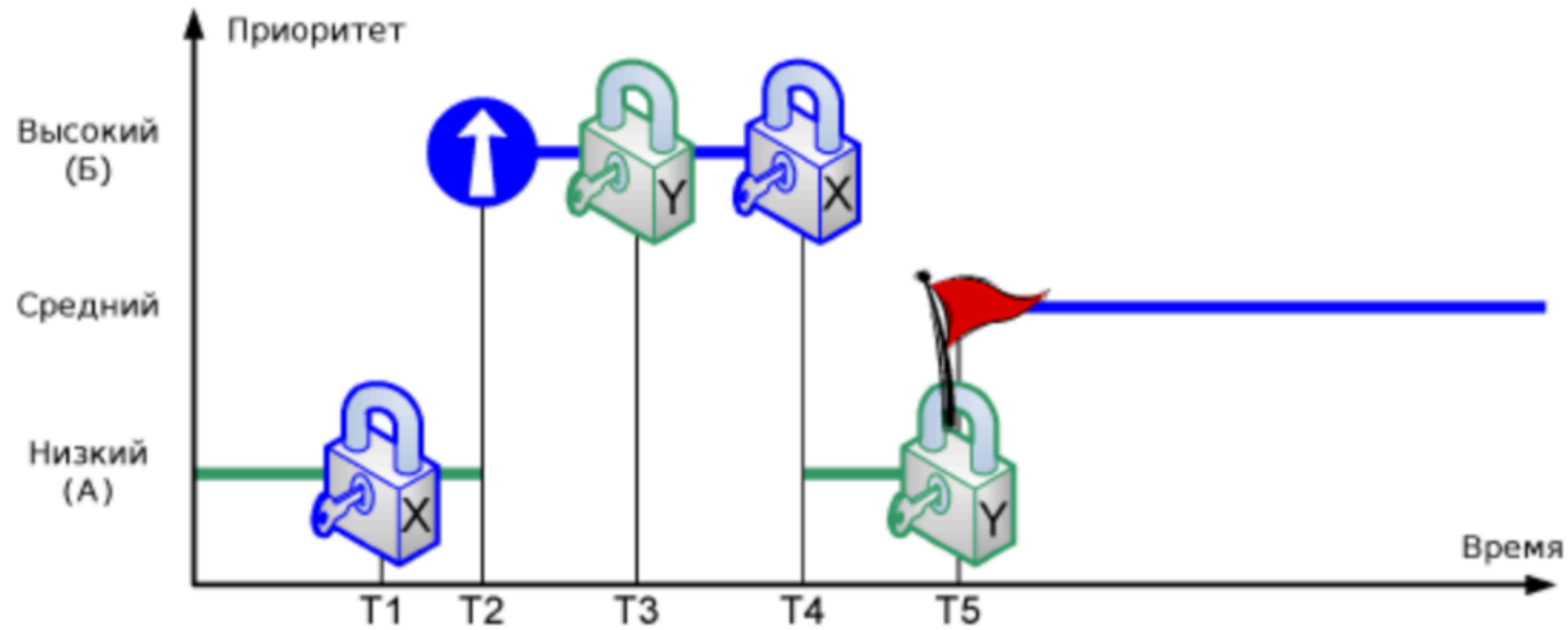
Ограниченная инверсия приоритетов



Неограниченная инверсия приоритетов



Взаимная блокировка

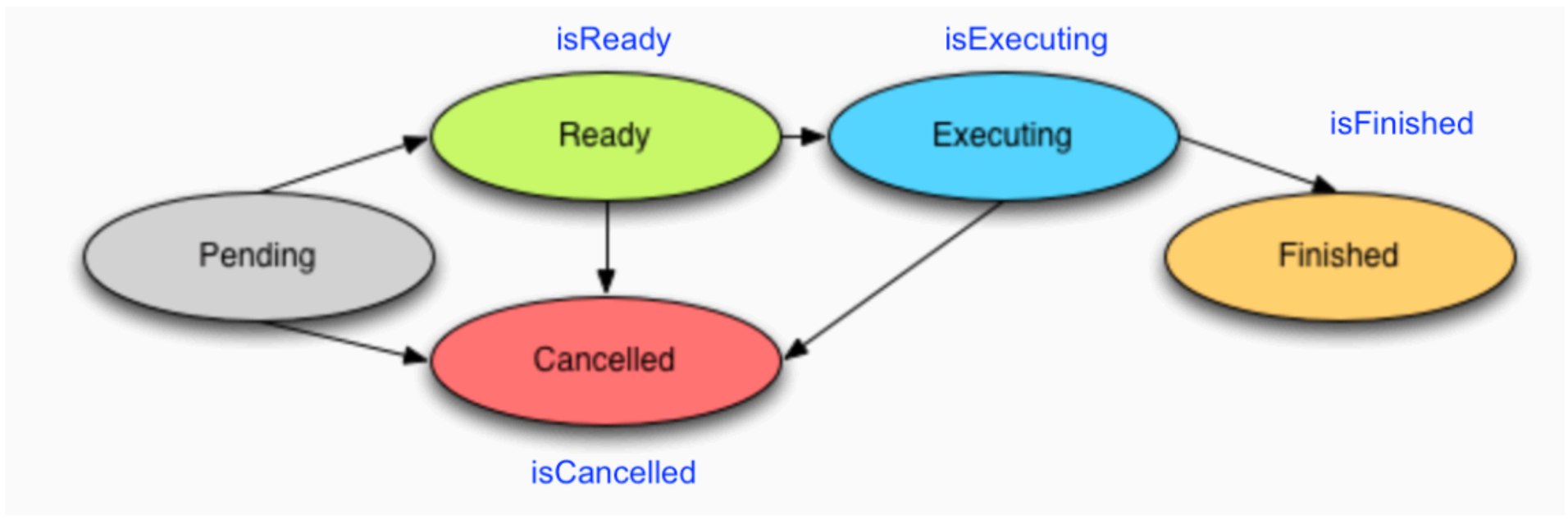


Operation - более высокоуровневый, более абстрактный механизм работы с потоками в iOS

Operation представляет собой законченную задачу и является абстрактным классом, который предоставляет вам потоко-безопасную структуру для моделирования состояния операции, ее приоритета, зависимостей (dependencies) от других Operations и управления этой операцией.

Класс Operation позволяет вам создать некоторую задачу, которую вы в будущем можете запустить на очереди операций OperationQueue, а пока она может ожидать выполнения других Operations

Машина состояний



Возможные состояния

- отложенная (pending)
- готова к выполнению (ready)
- выполняется (executing)
- закончена (finished)
- уничтожена (cancelled)

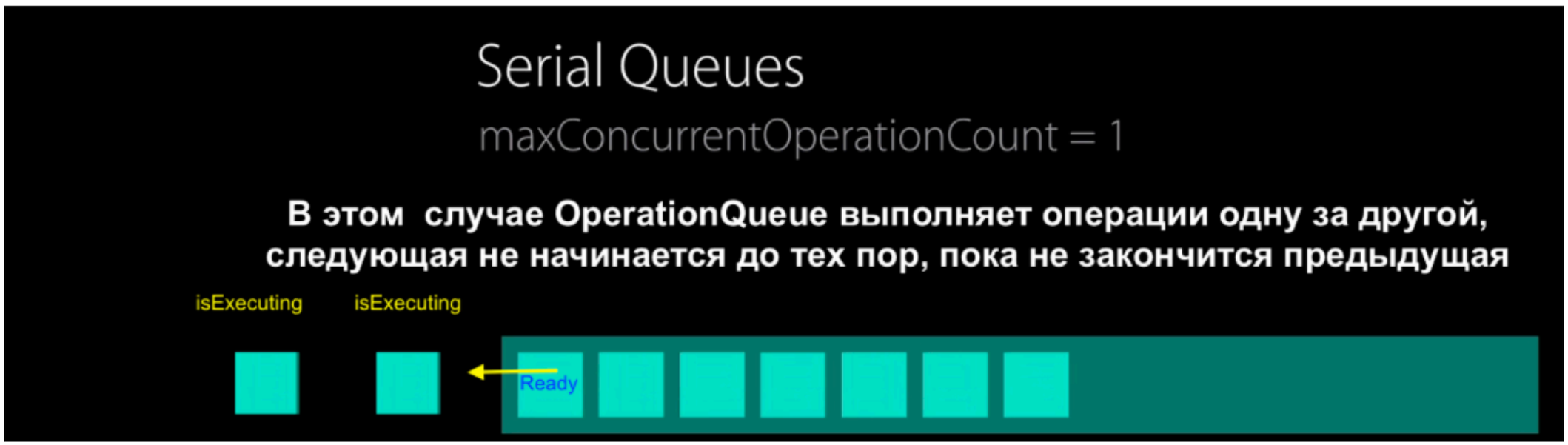
OperationQueue

```swift
let operationQ = OperationQueue()
```

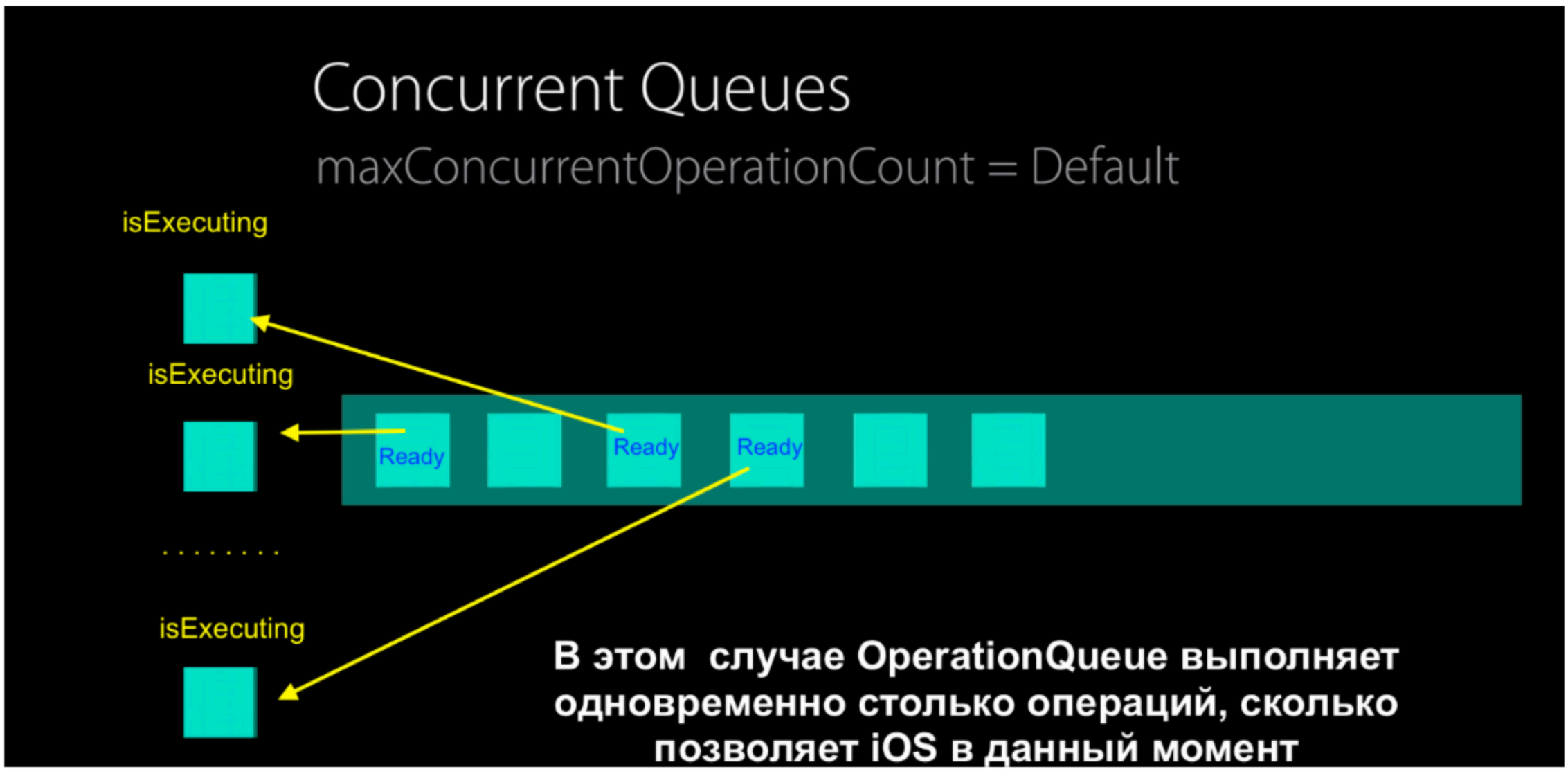
Инициализатор

```swift
open class OperationQueue : NSObject {

  open class var current: OperationQueue? { get }
  open class var main: OperationQueue { get }
  public class let defaultMaxConcurrentOperationCount: Int
  open var maxConcurrentOperationCount: Int

  open func addOperation(_ op: Operation)
  open func addOperation(_ block: @escaping () -> Swift.Void)
  open func addOperations(_ ops: [Operation], waitUntilFinished wait: Bool)

  open var operations: [Operation] { get }
  open var operationCount: Int { get }
  ...
}
```

Последовательная очередь



Параллельная очередь



Материалы

- Материалы по iOS школе
https://github.com/surfstudio/iOSSummerSchool2018
- WWDC 2015.Advanced NSOperations (session 226).
- NSOperation and NSOperationQueue Tutorial in Swift
- CONCURRENCY IN IOS
- Concurrency in Swift: One possible approach
- https://developer.apple.com/videos/play/wwdc2016/213/
- https://www.appcoda.com/grand-central-dispatch/
- http://khanlou.com/2016/04/the-GCD-handbook/
- http://mrdekk.ru/2016/05/19/gcd-handbook/
- https://github.com/apple/swift-evolution/blob/master/proposals/0088-libdispatch-for-swift3.md
- http://dduraz.com/2016/10/26/gcd/
- https://www.uraimo.com/2017/05/07/all-about-concurrency-in-swift-1-the-present/