А так ли нужна Swift Modern Concurrency?

Василий Усов

iOS-разработчик в VK Карты
Автор книг по разработке на Swift
Увлекаюсь программированием более 20 лет

VK Карты

Swift Modern Concurrency

Task, async/await и actor

Concurrency

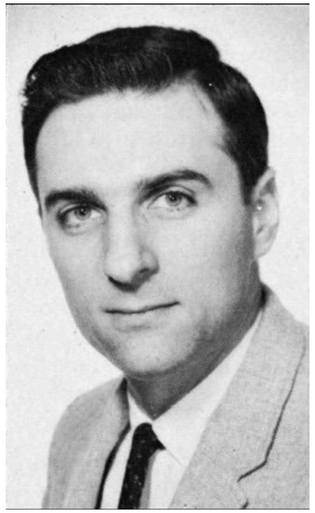
Способность системы «условно» одновременно выполнять несколько задач

Корутины

Haskell C++ JS C# Kotlin Swift Go Lua Python Lisp

Закон Конвея

«Организации проектируют системы,
которые копируют структуру коммуникаций в этой организации»



Мелвин Конвей

Ученый, программист, исследователь и просто хороший человек

Создал ассемблер для компьютера Берроуз
Участвовал в выпуске Mac Pascal

Мелвин Конвей

«Design of a Separable Transition-diagram Compiler», 1963

Компилятор для COBOL, концепция Корутин

«Design of a Separable Transition-diagram Compiler», 1963

Многопоточность

Routine

В переводе - текущий, обычный
Непосредственно сама программа

Subroutine

Подпрограмма - процедура, функция

Coroutine

Сопрограмма

Корутины

Это сабрутины на одном уровне, каждая из которых действует так, как если бы она была главной программой, хотя на самом деле главной программы нет. Это определение не ограничивает количество входов и выходов, которые может иметь корутина.

Мелвин Конвей

«Design of a Separable Transition-diagram Compiler»

Это понятие, обобщающее сабрутины, но имеющие несколько точек входа и выхода

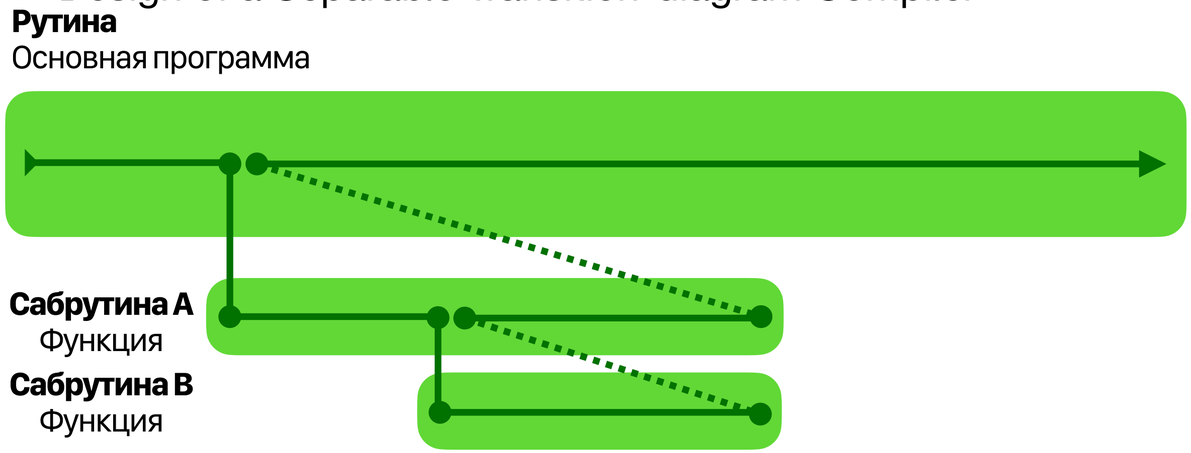
Дональд Кнут

«Искусство программирования»

Это сабрутины на одном уровне, каждая из которых действует так, как если бы она была главной программой, хотя на самом деле главной программы нет. Это определение не ограничивает количество входов и выходов, которые может иметь корутина.

Мелвин Конвей

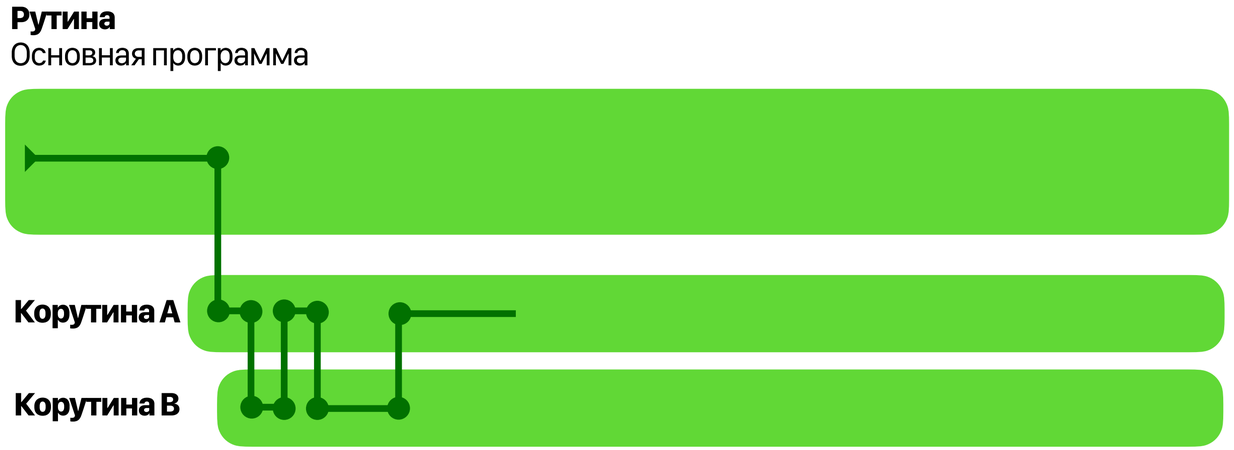
«Design of a Separable Transition-diagram Compiler»



Это понятие, обобщающее сабрутины, но имеющие несколько точек входа и выхода

Дональд Кнут

«Искусство программирования»



Корутина A

Корутина B

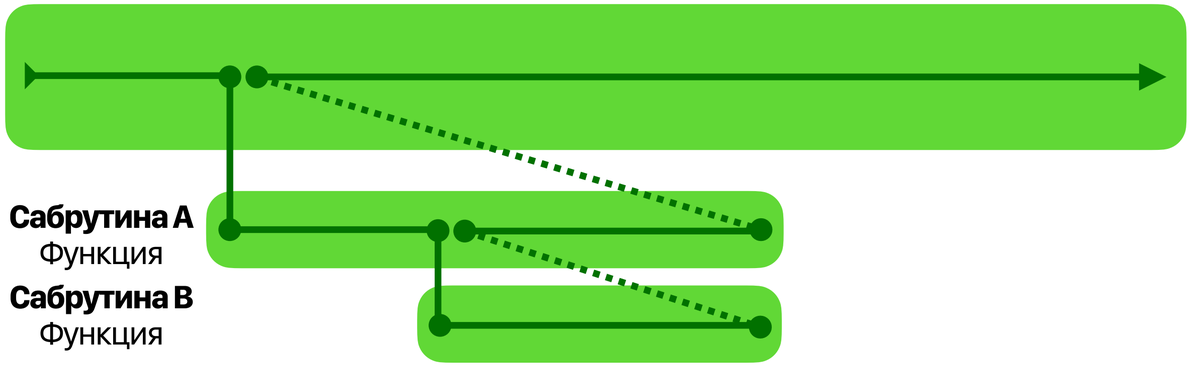
Это сабрутины на одном уровне, каждая из которых действует так, как если бы она была главной программой, хотя на самом деле главной программы нет. Это определение не ограничивает количество входов и выходов, которые может иметь корутина.

Мелвин Конвей
«Design of a Separable Transition-diagram Compiler»

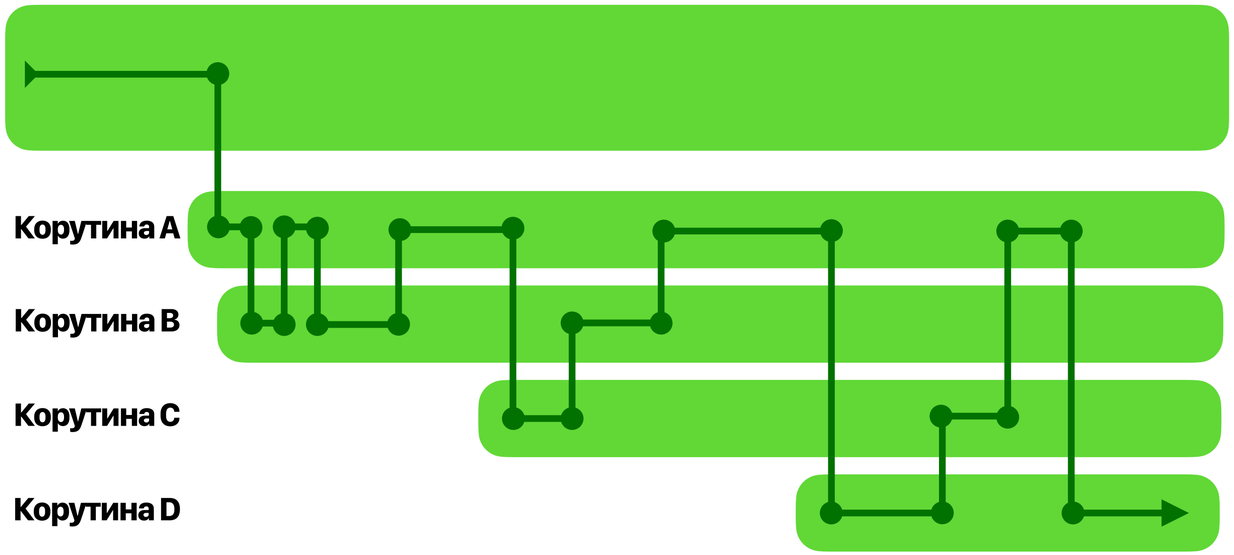
Это понятие, обобщающее сабрутины, но имеющие несколько точек входа и выхода

Дональд Кнут
«Искусство программирования»

Рутина
Основная программа



Рутина
Основная программа



Это легковесные блоки кода, которые могут приостанавливаться с сохранением своего состояния и позже заново запускаться

А в чем плюсы корутин?

Как работают потоки

Как работают потоки

от 20 нс до 20 мкс и более

Как работают корутины

Пул корутин

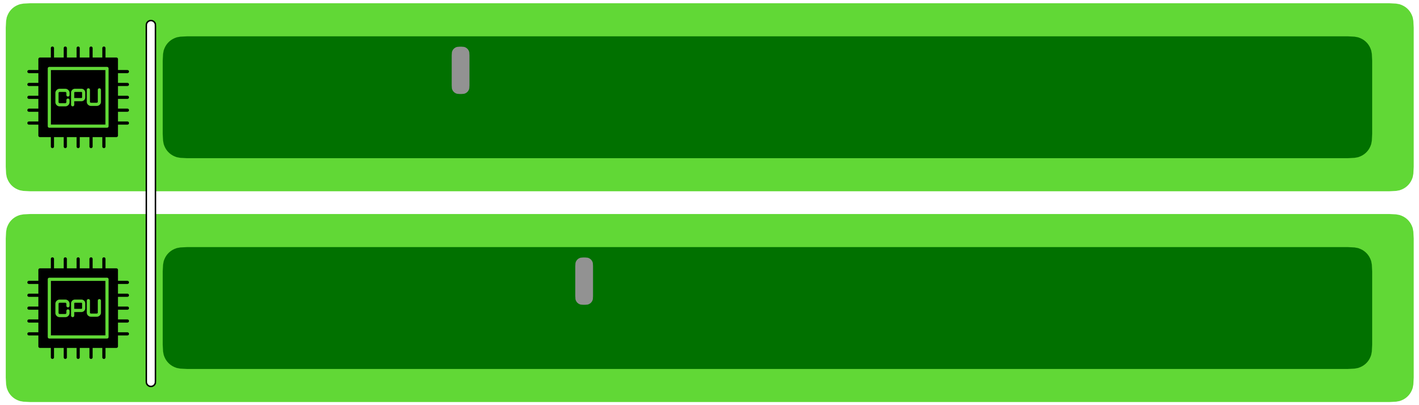
1

2

3

4

Пул потоков



Как работают корутины

от 3 нс до 20 нс и более

Примеры корутин

Lua

```
co = coroutine.create(function ()
       for i=1,10 do
         print("co", i)
         coroutine.yield()
       end
     end)

coroutine.resume(co)
```

Kotlin

```
fun main() = runBlocking {
    launch { doWorld() }
    println("Hello")
}

suspend fun doWorld() {
    delay(1000L)
    println("World!")
}
```

Go

```
func f(from string) {
    for i := 0; i < 3; i++ {
        fmt.Println(from, ":", i)
    }
}

func main() {
    f("direct")
    go f("goroutine")
}
```

Swift

Grand Central Dispatch

C-библиотека libdispatch

Крис Латтнер

Основатель и главный архитектор LLVM
В прошлом руководитель разработки Swift

Крис Латтнер

«Swift Concurrency Manifesto», 2017

Закладывает основы конкурентности

«Swift Concurrency Manifesto», 2017

«Мы фокусируемся на абстракциях конкурентности на основе задач, которые сильно зависят от событий (например, реагирование на события пользовательского интерфейса или запросы от клиентов)»

«Мы должны стремиться сократить время программиста, необходимое для перехода от идеи к рабочей и эффективной реализации»

«Если говорить более абстрактно, нам следует обратиться к моделям параллелизма, предоставляемым другими языками и фреймворками, и собрать воедино лучшие идеи, откуда бы мы их ни взяли, стремясь в целом быть лучше любого конкурента»

«Swift Concurrency Manifesto», 2017

Swift 5.5

«Swift Concurrency Manifesto», 2017

Swift 5.5

Swift Modern Concurrency

«Swift Concurrency Manifesto», 2017

Swift 5.5

Swift Modern Concurrency

Task, async/await, actor

Swift Modern Concurrency - это корутины?

ДА

есть нюансы

Как создавать корутины

```
// Создание задачи с наследованием контекста
Task {}

// Создание задачи с установкой приоритета
Task(priority: .high) {}

// Создание задачи без наследования контекста
Task.detached {}
```

Task и точка останова

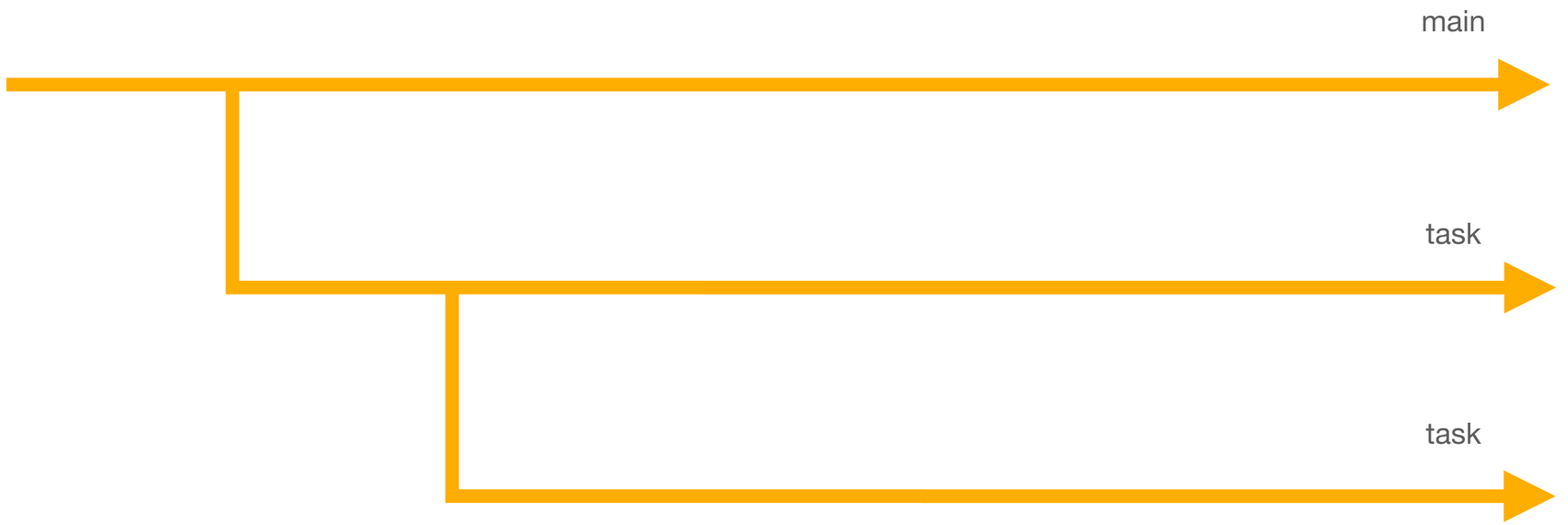

```
Task {
    log("Запуск задачи")
    let configuration = DataLoadingConfiguration(scheme: .https, server: .base)
    let data = await loadData(withConfiguration: configuration)

    log("Данные были загружены")
    let convertedData = Converters.convertServerDataToDBScheme(data: data)
    await saveDataToDB(convertedData)

    // ... другие операции

    Task.yield()

    // ...
}
```

Как создавать таски

```
Task {
    // ...
    Task {
        // ...
    }
    // ...
}
```

Unstructured Concurrency

```
Task {
    // ...
    Task {
        // ...
    }
    // ...
}
```



Structured Concurrency

Structured Concurrency

```
Task {
    // дочерние задачи с помощью async let
    async let image1 = await loadImage(id: 1)
    async let image2 = await loadImage(id: 2)
    async let image3 = await loadImage(id: 3)
    async let image4 = await loadImage(id: 4)

    // ожидаем результатов и передаем их в базу данных
    await saveToDB([image1, image2, image3, image4])
}
```

Structured Concurrency

```
Task {
    // группа дочерних задач с помощью TaskGroup
    let images = await withTaskGroup(of: Data.self) { group -> [Data] in
        for i in 1...4 {
            group.addTask {
                await loadImage(id: i)
            }
        }

        var result = [Data]()
        for await value in group {
            result.append(value)
        }
        return result
    }

    saveToDB(images)
}
```

Structured Concurrency

```
Task {
    // Запускаем 4 независимые задачи
    let image1 = Task { await loadImage(id: 1) }
    let image2 = Task { await loadImage(id: 2) }
    let image3 = Task { await loadImage(id: 3) }
    let image4 = Task { await loadImage(id: 4) }

    // Ожидаем результатов задач
    await (image1.value, image2.value, image3.value, image4.value)
    // и записываем их в базу данных
    // ...
}
```

А в чем плюсы Swift Modern Concurrency?

Быстрее

Unstructured concurrency

Structured concurrency

Удобный синтаксис

Часть языка

Открытый исходный код

Но есть и минусы

Опять что-то учить

iOS ничего не знает про новую асинхронность

А так ли нужна Swift Modern Concurreny?

ДА

есть нюансы

Несколько полезных рецептов

```swift
// Замена GCD-метода asyncAfter
Task {
    try? await Task.sleep(nanoseconds: 5 * NSEC_PER_SEC)
    // ...
}
```

```swift
// Ожидание ресурса
Task {
    while di == nil {
        Task.yield()
    }
    // ...
}
```

```swift
// Фейковая пауза
Task {
    async let sleep: Void = await Task.sleep(nanoseconds: 2 * NSEC_PER_SEC)
    async let data = await loadData()

    try? await (sleep, data)
    // ...
}
```

Что мы не обсудили

Статья
Design of a Separable
Transition-diagram Compiler

Доклад
Романа Елизарова
про корутины в Kotlin

Сессия WWDC22
Swift concurrency
behind the scene

Swift Concurrency
Manifesto

Исходный код функции
переключения контекста
для ARM в macOS

Замеры времени
переключения контекста
от TSUNA

Книга
Is Parallel Programming
Hard, And, If So, What Can
You Do About It?

Я в Telegram