DESENVOLVIMENTO PARA IOS – APPLE 02

IOS SDK 9

Professor: Pedro Henrique
prof.pedrohenrique.iossdk@gmail.com

AGENDA

- Blocos
  - De onde vem?
  - Para que servem?
  - Grand Central Dispatch (GCD)
    - Multithread
- Animação usando blocos;
- Prática

OBJETIVOS DO DIA

- Compreender o conceito por trás dos blocos;
- Aprender como e quando usar um bloco;
- Entender os princípios de animação básica do iOS.

O QUE É BLOCO?

- Um bloco é um trecho de código **anônimo** e **auto-contido**;
- A existência do bloco sempre é ligada a algum outro escopo dentro do programa;
- A sintaxe, similar à uma expressão *Lambda*, serve para criar algo como um *Closure*;
- O bloco pode "interagir" com o que está fora dele, mas o que está de fora não pode "interagir" com o que está dentro do bloco.

DE ONDE VEM OS BLOCOS?

- O conceito de blocos foi adicionado às linguagens C/C++ e Objetive-C pela Apple para tornar mais fácil a implementação da arquitetura Multithread GCD (Grand Central Dispatch);
- Existem em outras linguagens de programação;
  - JavaScript (Closures);
  - Java 8;
  - Etc.

BLOCOS

- Tal qual uma função ou método, um bloco pode receber argumentos e retornar um valor;
- Pode declarar variáveis internas;
- E, diferente de uma função, um bloco pode capturar e modificar o estado **do contexto em que foi declarado**;
- Um bloco pode ser armazenado em uma variável e, portanto:
  - Pode ser retornado ou pode ser passado como argumento.

Um bloco pode ser chamado imediatamente ou em qualquer momento futuro.

LINKS DO CORAÇÃO ❤️

- Closure
  - http://en.wikipedia.org/wiki/Closure_(computer_programming)
- Função Anônima
  - http://en.wikipedia.org/wiki/Anonymous_function
- Escopo Léxico
  - http://en.wikipedia.org/wiki/Scope_(computer_science)#Lexical_scoping
  - http://pt.stackoverflow.com/questions/13034/o-que-são-escopo-léxico-e-escopo-dinâmico-e-quais-são-suas-principais-diferenças
  - http://www.inf.puc-rio.br/~inf1621/escopo.pdf

LINKS DO CORAÇÃO ❤️

- Documentação da Apple sobre Blocos
  - https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Blocks/Articles/00_Introduction.html#//apple_ref/doc/uid/TP40007502-CH1-SW1
- Tutorial de Blocos em Objective-C
  - http://www.appcoda.com/objective-c-blocks-tutorial/
  - http://rypress.com/tutorials/objective-c/blocks

LINKS DO CORAÇÃO ❤️

- GCD – Grand Central Dispatch
  - http://www.raywenderlich.com/60749/grand-central-dispatch-in-depth-part-1
  - https://developer.apple.com/library/ios/documentation/Performance/Reference/GCD_libdispatch_Ref/
  - https://vimeo.com/49718712

PARA QUE SERVEM OS BLOCOS?

- Na aula de hoje, vamos usar para animação, mas eles também têm outros usos:
- Enumeração;
- Ordenação;
- Notificações;
- Tratamento de Erros;
- Multithread (GCD – Grand Central Dispatch)

BLOCO – EXEMPLOS - ANIMAÇÃO

```
//Bloco declarado inline para animação
[UIView animateWithDuration:5 animations:^{
    CGRect frame = self.label.frame;
    frame.origin = CGPointMake(30, 30);

    [self.label setFrame:frame];
}];
```

BLOCO – EXEMPLOS - ANIMAÇÃO

Esse código tem um problema!!!
Veremos mais a diante...

BLOCOS – EXEMPLOS – ORDENAÇÃO

```
NSArray *nomes = @[@"Pedro", @"Ana", @"Marcos", @"Kelly", @"Walter", @"Luke"];

NSComparisonResult (^blocoQueOrdena)(id itemA, id itemB) = ^(id a, id b) {
    NSString *n1 = (NSString *) a;
    NSString *n2 = (NSString *) b;
    return [n1 compare:n2];
};

NSArray *ordenado = [nomes sortedArrayUsingComparator:blocoQueOrdena];
NSLog(@"Array ordenado: %@", ordenado);
```

BLOCOS – EXEMPLOS - ENUMERAÇÃO

```
NSDictionary *coisas = @{@"c1":@1, @"c2":@2, @"c3":@3};

void (^blocoQueEnumera)(id key, id obj, BOOL *parar) = ^(id key, id obj, BOOL *parar) {
    if (!*parar) {
        NSLog(@"Chave %@ : Valor: %@", key, obj);
    }
};
[coisas enumerateKeysAndObjectsUsingBlock:blocoQueEnumera];
```

BLOCO – EXEMPLO – TRATAMENTO DE ERROS

```
/* Requisição HTTP asíncrona com bloco de callback que inclui um
 objeto NSError - para tratamento do erro

 *** ESTE TRECHO DE CÓDIGO NÃO É UM EXEMPLO PERFEITO PARA REQUISIÇÕES HTTP

 */
NSURL *url = [NSURL URLWithString:@"http://www.7-zip.org/a/7z938-src.7z"];
[NSURLConnection sendAsynchronousRequest:[NSURLRequest requestWithURL:url]
                                   queue:[NSOperationQueue new]
                       completionHandler:^(NSURLResponse *response, NSData *data, NSError *connectionError) {
                           if (!connectionError) {
                               NSLog(@"Response: %@", response);
                           } else {
                               NSLog(@"Erro de conexão: %@", connectionError);
                           }
                       }];
```

BLOCO COMO PROPRIEDADE

```
#import <UIKit/UIKit.h>

typedef int(^meuBloco)(int a, int b);

@interface ViewController : UIViewController

@property (copy) int (^meuBloco)(int a, int b);

@end
```

BLOCO COMO PROPRIEDADE

```
int (^somarDoisInteiros)(int a, int b) = ^(int a, int b) {
    return a+b;
};
```

BLOCO COMO PROPRIEDADE

```
int retorno = self.meuBloco(10,22);
NSLog(@"Resultado: %d", retorno);
```

GCD – GRAND CENTRAL DISPATCH

- É uma lib, em C, da Apple que oferece suporte à execução concorrente de código nos ambientes do iOS e do OSX;
- Oferece um modelo simples para implementação de concorrência, evitando a geração de bugs;

GCD – CONCORRENTE VS SERIAL

- Código Serial:
  - Quando uma operação é executada após a outra;
  - A próxima linha de código só executa quando a atual terminar.
- Código Concorrente:
  - As operações podem ser executadas ao mesmo tempo;

GCD – SÍNCRONO VS ASSÍNCRONO

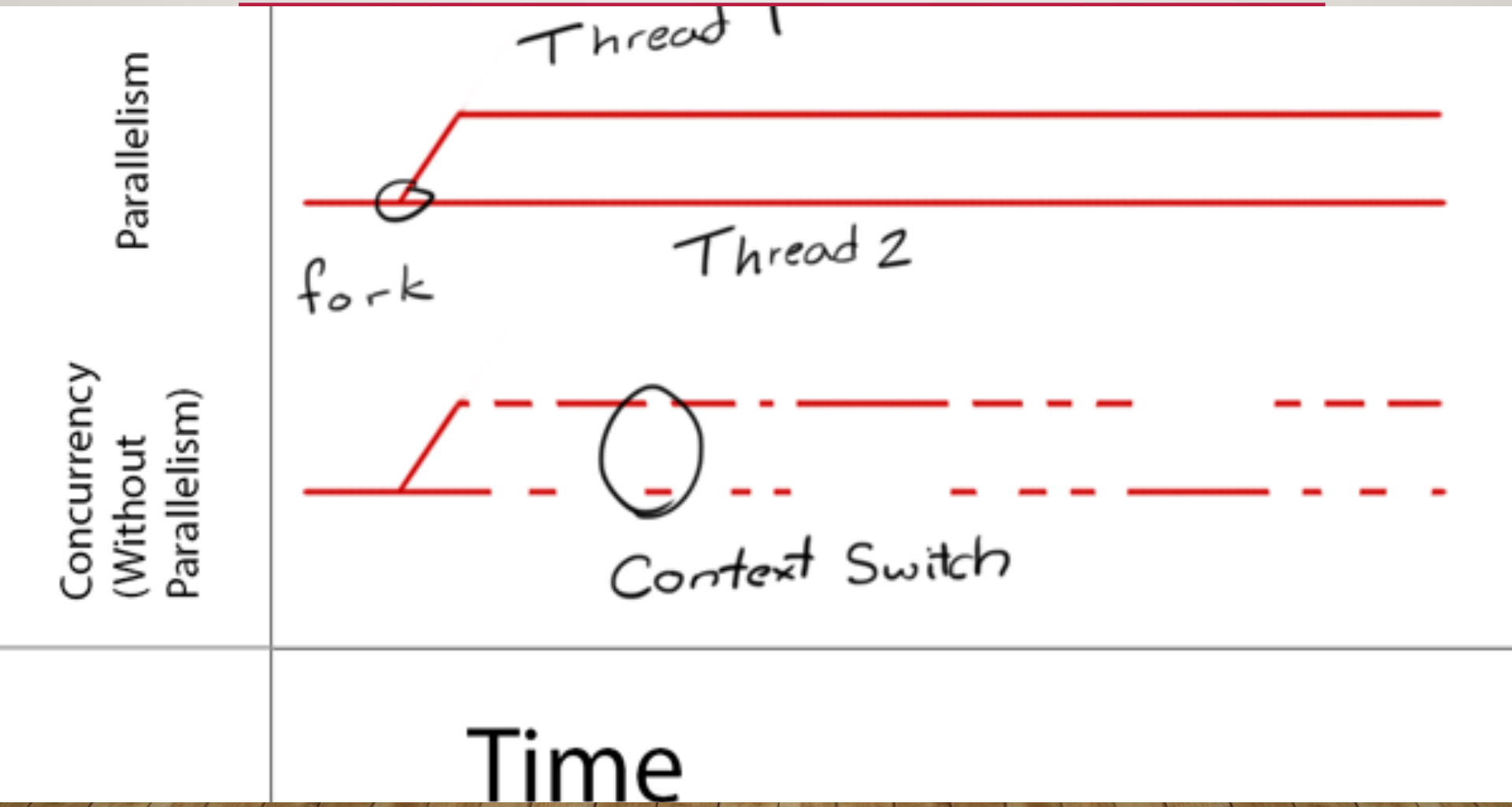
- Uma função é síncrona quando ocorre o bloqueio na thread onde ela foi chamada até que sua execução termine;
- Uma função é assíncrona quando ela retorna de imediato ao fluxo de execução, sem bloquear a thread que a chamou.

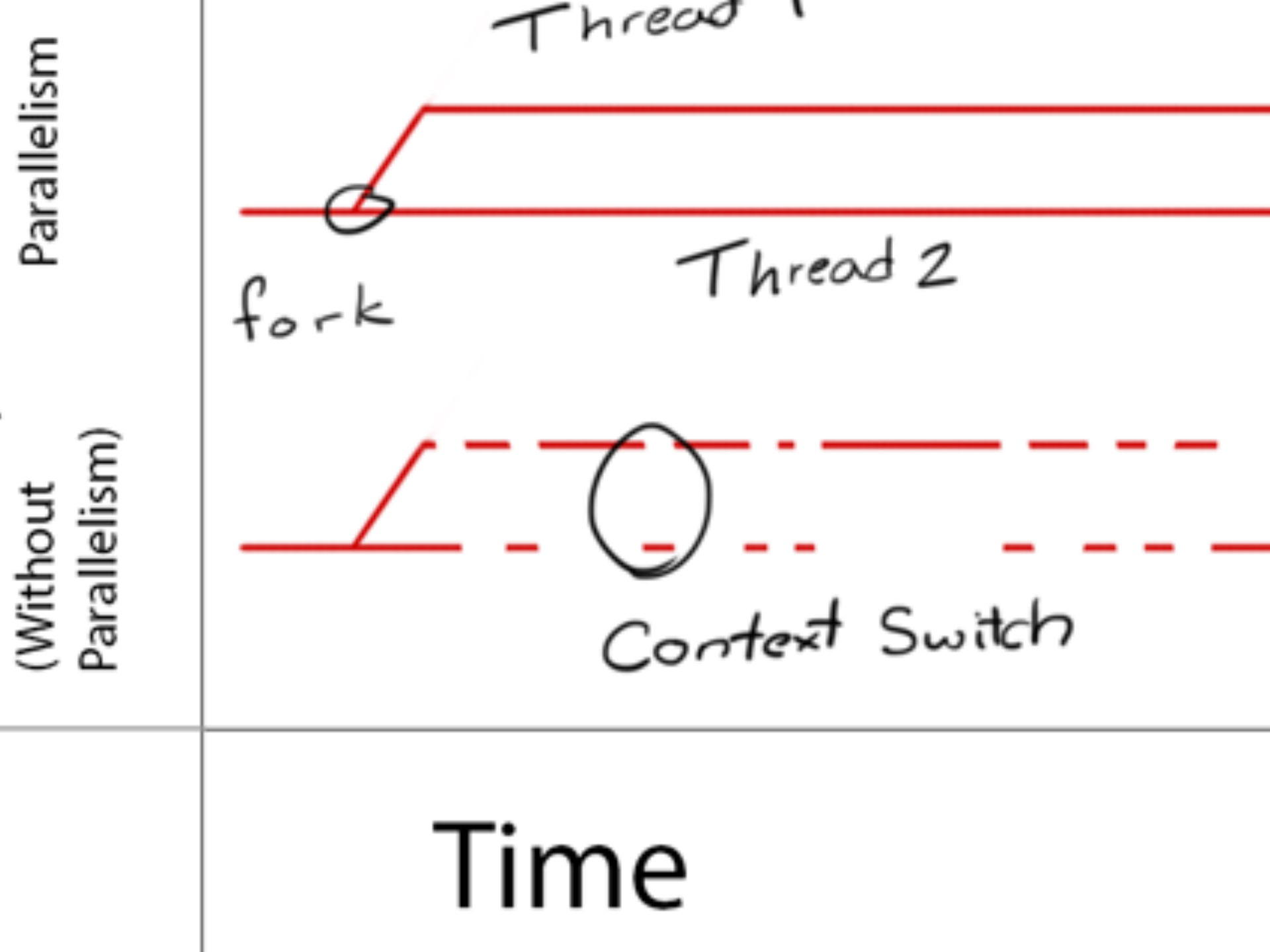
GCD – CONCORRÊNCIA VS PARALELISMO

- Dispositivos com múltiplos núcleos de processamento executam múltiplas threads ao mesmo tempo através de paralelismo;

- Dispositivos single-core, no entanto, só conseguem rodar uma thread por vez.

GCD – CONCORRÊNCIA VS PARALELISMO




Parallelism
(Without Parallelism)
Thread 1
Thread 2
fork
Context Switch
Time

GCD – CONCORRÊNCIA VS PARALELISMO

- Paralelismo **requer** concorrência;
- Concorrência **não garante** paralelismo.

- Com o GCD, você vai estruturar o seu código para dizer o que **pode rodar em paralelo** e o que não pode rodar em paralelo!

GCD - FILAS

- Existem dispositivos iOS tanto single-core quanto multi-core;
- Por este motivo, você não precisa se preocupar em decidir entre paralelismo ou concorrência – o GCD faz isso por você através do mecanismo de filas.

GCD - FILAS

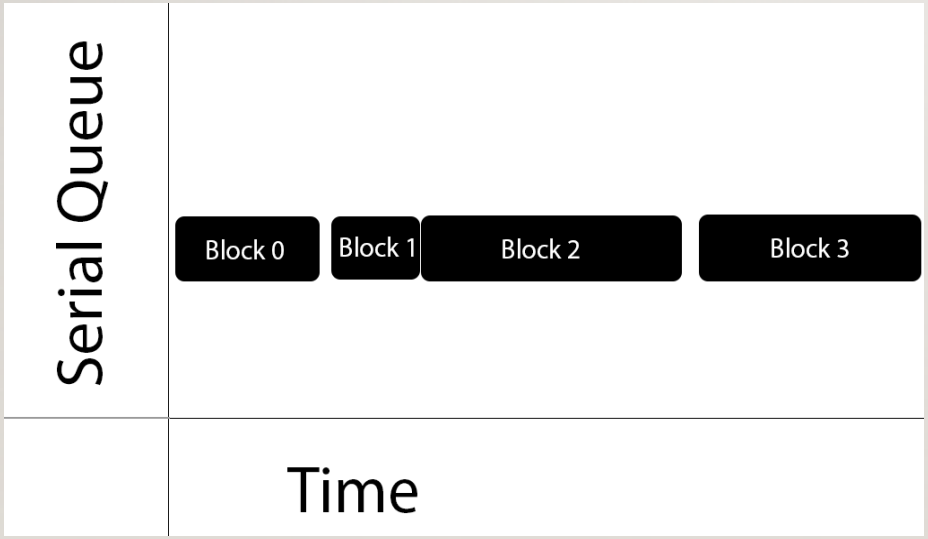
- O GCD oferece o mecanismo de filas de expedição (dispatch queues), que controlam as tarefas – que são passadas através de blocos;
- A ordem de execução das tarefas é FIFO
  - First In, First Out
- O GCD decide se executa as filas através de concorrência ou paralelismo.

GCD - FILAS

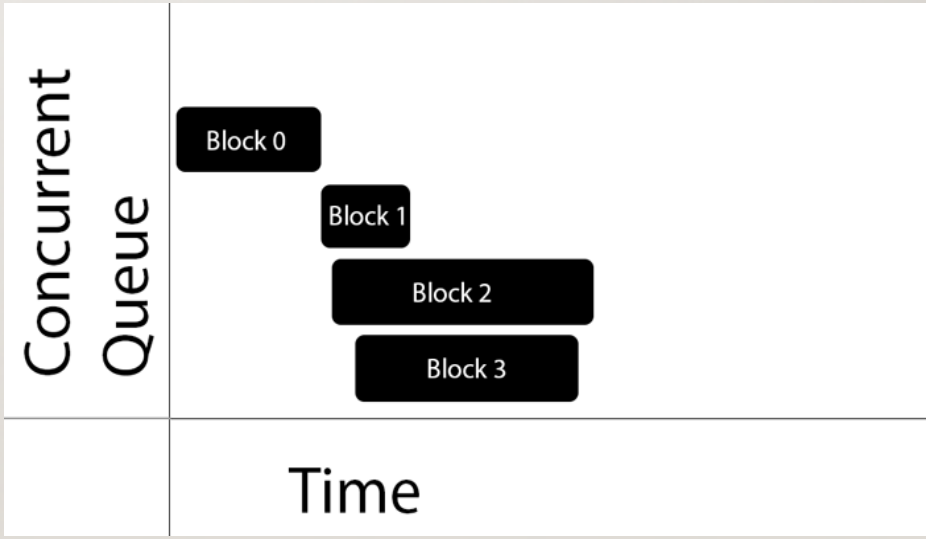
- No ambiente do iOS, temos ***n* filas concorrentes** e, em casos normais, apenas uma fila serial;
- As tarefas em filas concorrentes têm a garantia do sistema de que vão ser executadas na mesma ordem que foram adicionadas à uma dada fila.

GCD - FILAS

FILA SERIAL (MAIN THREAD)



FILA CONCORRENTE (OUTRAS THREADS)



GCD - FILAS

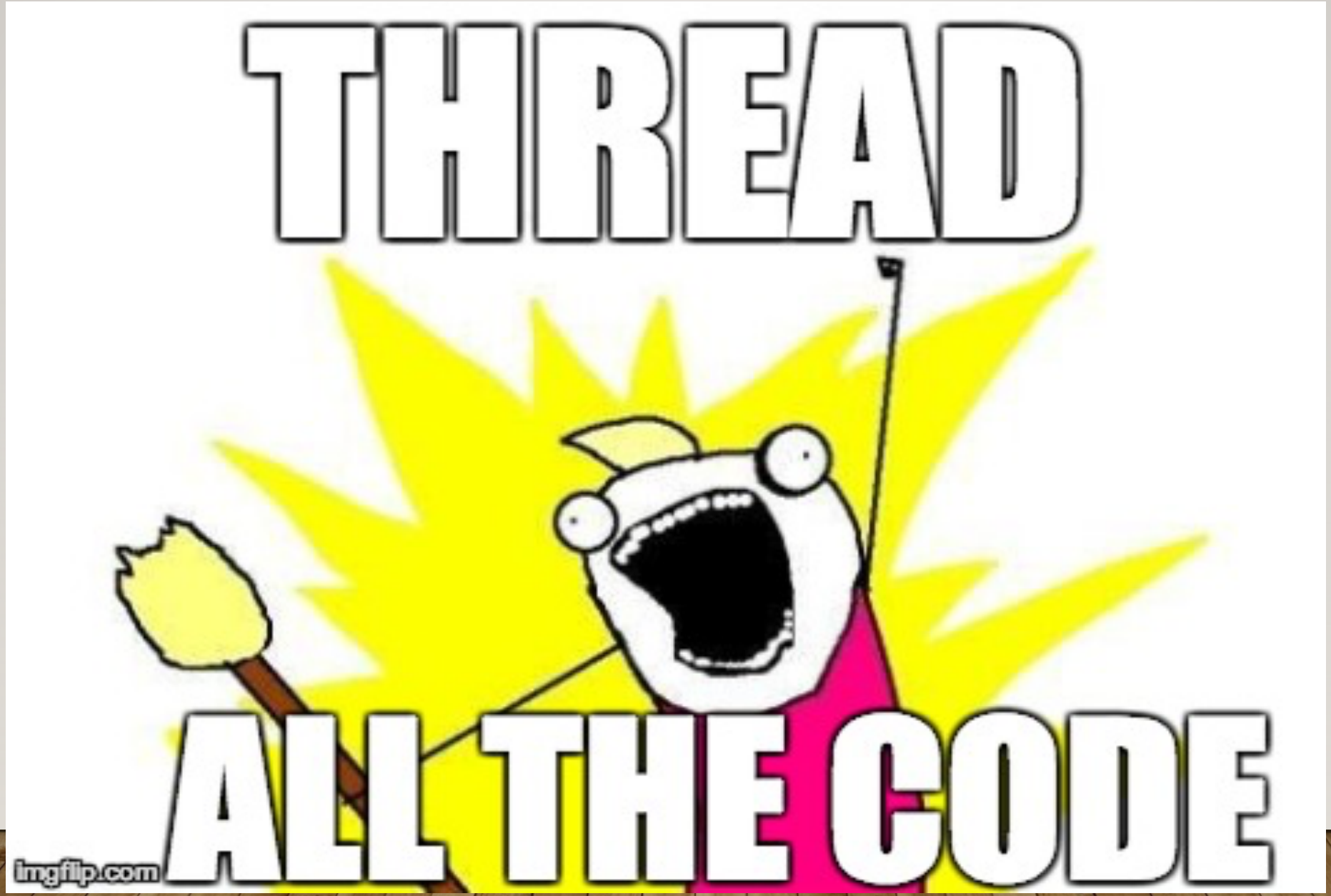
- O iOS já provê 4 filas padrão, **sem contar** a fila principal:
- Main Thread (**dispatch_get_main_queue**());
  - Esta fila é **serial**!
- Background (**DISPATCH_QUEUE_PRIORITY_BACKGROUND**);
- Baixa Prioridade(**DISPATCH_QUEUE_PRIORITY_LOW**);
- Prioridade Padrão (**DISPATCH_QUEUE_PRIORITY_DEFAULT**);
- Alta Prioridade (**DISPATCH_QUEUE_PRIORITY_HIGH**).

GCD - FILAS

- Atenção!
- As filas padrão não são usadas apenas por você! Várias APIs internas do sistema e do aplicativo estão rodando nessas filas!
- Por isso, o iOS te dá a chance de criar suas próprias filas.

GCD - FILAS

- A grande coisa em relação as filas é saber como delegar adequadamente suas chamadas para as filas corretas, a fim de obter a melhor relação Custo vs Performance.
- Como saber isso, então?
- Só exercitando!


THREAD
ALL THE CODE
imgflip.com

ANIMAÇÃO BÁSICA

- Frame;
- Cor / Alpha;
- Transformação e;
- Transição


(+)
COMO FAS/

ANIMAÇÃO BÁSICA

- Se faz usando métodos estáticos da classe UIView e blocos!

```
[UIView animateWithDuration:5
                      delay:1
                    options:UIViewAnimationOptionCurveEaseInOut
                 animations:^{

                     CGRect frame = self.label.frame;
                     frame.origin = CGPointMake(30, 30);
                     [self.label setFrame:frame];

                 } completion:^(BOOL finished) {

                     if (finished) {
                         NSLog(@"A animação terminou!");
                     }

                 }];
```

ANIMAÇÃO BÁSICA - OPÇÕES

UIViewAnimationOption	Descrição
BeginFromCurrentState	Interrompe outras animações em curso
AllowUserInteraction	Processa gestos durante a animação
LayoutSubviews	Anima o ajuste de layout das subviews junto com a animação
Repeat	Repete a animação indefinidamente
Autoreverse	A animação "vai e volta", depois para
OverrideInheritedDuration	Se não informar uma duração, usa a duração da animação em execução
OverrideInheritedCurve	Se não informar uma curva, usa a curva da animação em execução
AllowAnimatedContent	Permite a interpolação de animações (uma animação dentro da outra)
CurveEaseInEaseOut (padrão)	Começa a animação devagar, acelera e desacelera no final
CurveEaseIn	Começa devagar e depois acelera
CurveLinear	Usa a mesma velocidade no decorrer da animação

ANIMAÇÃO BÁSICA

- Para animar mudanças na hierarquia de views, o jeito é esse:

```
UILabel *labelIESB = [[UILabel alloc] initWithFrame:self.label.frame];
[labelIESB setText:@"IESB"];
[labelIESB setBackgroundColor:[UIColor redColor]];
[labelIESB setTextColor:[UIColor whiteColor]];

[UIView transitionFromView:self.label
                    toView:labelIESB
                  duration:10
                   options:UIViewAnimationOptionTransitionFlipFromLeft
                completion:^(BOOL finished) {
                    if (finished) {
                        NSLog(@"Animação Terminou!");
                    }
                }];
```

ANIMAÇÃO BÁSICA - TRANSIÇÕES

- UIViewAnimationOptionTransitionNone
- UIViewAnimationOptionTransitionFlipFromLeft
- UIViewAnimationOptionTransitionFlipFromRight
- UIViewAnimationOptionTransitionCurlUp
- UIViewAnimationOptionTransitionCurlDown
- UIViewAnimationOptionTransitionCrossDissolve
- UIViewAnimationOptionTransitionFlipFromTop
- UIViewAnimationOptionTransitionFlipFromBottom

HORA DE BRINCAR!

- Blocos;
- Animação.