Block闭包中的变量在执行期间被释放会发生什么

原文地址：

<http://stackoverflow.com/questions/12272783/what-happens-when-a-block-is-set-to-nil-during-its-execution>

翻译如下：

Q：

假定我的block中有一个强引用的对象，有时候在block执行期间，强引用的变量被释放，指向nil，那么block会继续完成执行呢还是导致崩溃？我看了exc-bad-access错误，但是不能必现这个问题，所以我不能准确的知道他们为什么会发生。

For example:

- (void)method

{

Self.blcok = ^{

//code

Self.block = nil

//more code – crash here?

}

}

- (void)otherMothod

{

block()

}

A1:

我觉得对于这个问题你没有必要担心，blocks是objective-c对象，一个block不仅retain你使用的这个变量的引用，而且block自身也由ARC管理。如果你调用，一个引用对于他自身来说是存在的。

A2:

根据问题的描述，似乎无法保证block在执行期间被retain，相反，GCD调用，例如dispatch\_async能保证block被retain，从这里可以看出，你无法假定一个普通的调用在block中，可以retain block。所以在你的代码中需要添加这个：

- (void)otherMothod

{

Dispatch\_block\_t localBlock = Block\_copy(block);

localBlock()

Block\_release(localBlock);

}

如果是支持ARC的呢，是不是在正在执行的block中创建了本地的强引用呢？

A3:

我相信对于这个问题我最终有了一个完美的答案，所有这些都是在ARC环境下执行。

一个block在执行期间可以变成一个已经被释放的block。这个block会照常执行，不会报错，但是它捕获的指针会具有不确定（有潜在危险）的值。

假定ObjectA 有一个copy 属性的block，名字叫completion：

@property (nonatomic, copy) void (^complete)();

赋值如下：

\_\_weak ObjectA \*weakSelf = self;

Self.completion = ^{

weakSelf.completion = nil;

[weakSelf doSometingElse];

}

如果这样调用block

- (void)method

{

\_completion(); //直接用ObjectA中block的实例变量调用

}

…然后，假定这个block的实例变量在没有其他引用，它已经被释放，指向它的弱引用变量变成nil，doSometingElse从来没有被调用，要解决这个问题，最好的办好就是通过其访问方法进行调用，这样做是因为会在栈上重新分配一个新的copy。原来的block被释放，但是新的block以及所有指向它的变量将生存在当前的上下文中。

- (void)method

{

self.completion(); //用带有copy属性的访问方法调用block

}

A4:

当执行一个block的方法没有检测是否仍然是一个引用指向这个block，就会发生崩溃，你可能在这个方法中导致崩溃如果你没有这样的检测的话。

- (void)methodWithBlock:( void (^)(void))block

{

If(block)

{

Block();

}

}

非常感谢，虽然这并不是我要表达的意思，在我视角中，block已经在执行中，当block变成nil，在你的视角中，在开始执行block之前将block设置为nil。