

主页 > 业界动态

iOS应用崩溃日志分析

发布于：2013-07-25 15:54 阅读数：78144

“ 为确保你的应用正确无误，在将其提交到应用商店之前，你必定进行了大量的测试工作。它在你的设备上也运行得很好，但是，上了应用商店后，还是有用户抱怨会闪退！如果你跟我一样是个

”

阅读器

转自[raywenderlich](#)

作为一名应用开发者，你是否有过如下经历？

为确保你的应用正确无误，在将其提交到应用商店之前，你必定进行了大量的测试工作。它在你的设备上也运行得很好，但是，上了应用商店后，还是有用户抱怨会闪退！

如果你跟我一样是个完美主义者，你肯定想将应用做到尽善尽美。于是你打开代码准备修复闪退的问题……但是，从何处着手呢？

这时iOS崩溃日志派上用场了。在大多数情况下，你能从中了解到关于闪退的详尽、有用的信息。

通过本教程，你将学习到一些常见的崩溃日志案例，以及如何从开发设备和iTunes Connect上获取崩溃日志文件。你还将学习到符号化（symbolication），从日志追踪到代码。你还将学习调试一个在特定情况下会闪退的应用。

让我们开始动手吧！

什么是崩溃日志，从哪里能得它？

iOS设备上的应用闪退时，操作系统会生成一个崩溃报告，也叫崩溃日志，保存在设备上。

崩溃日志上有很多有用的信息，包括应用是什么情况下闪退的。通常，上面有每个正在执行线程的完整堆栈跟踪信息，所以你能从中了解到闪退发生时各线程都在做什么，并分辨出闪退发生在哪个线程上。

有几种方法可以从设备上获取崩溃日志。

设备与电脑上的iTunes Store同步后，会将崩溃日志保存在电脑上。根据电脑操作系统的不同，崩溃日志将保存在以下位置：

Mac OS X: `~/Library/Logs/CrashReporter/MobileDevice/`

Windows XP: `C:\Documents and Settings<USERNAME>\Application Data\Apple Computer\Logs\CrashReporter\MobileDevice<DEVICE_NAME>`

Windows Vista or 7: `C:\Users<USERNAME>\AppData\Roaming\Apple Computer\Logs\CrashReporter\MobileDevice<DEVICE_NAME>`

开发者通道

排行榜

代码库

图书库

网站库

发码区

工具库

招聘区

外包区

问答区

关注CocoaChina

关注微信

移动版

最近更新

1

苹果要求开发者使用最新的iOS 8和C

2014-08-26

2

论坛源码推荐（8.26）:iOS 8 Today

2014-08-26

3

OCaml 发布 iOS 7 版编译器（OCar

2014-08-26

4

iOS 8自动调整UITableView和UICol

2014-08-25

5

论坛源码推荐（8.25）:Chisel辅助iC

2014-08-25

6

UI 测试

2014-08-22

7

论坛源码推荐（8月21日）：浏览ZIF

2014-08-21

8

对访问控制与protected的理解

2014-08-20

9

IAP最佳实践

2014-08-18

10

论坛源码推荐（8月18日）：VimR 2

2014-08-18

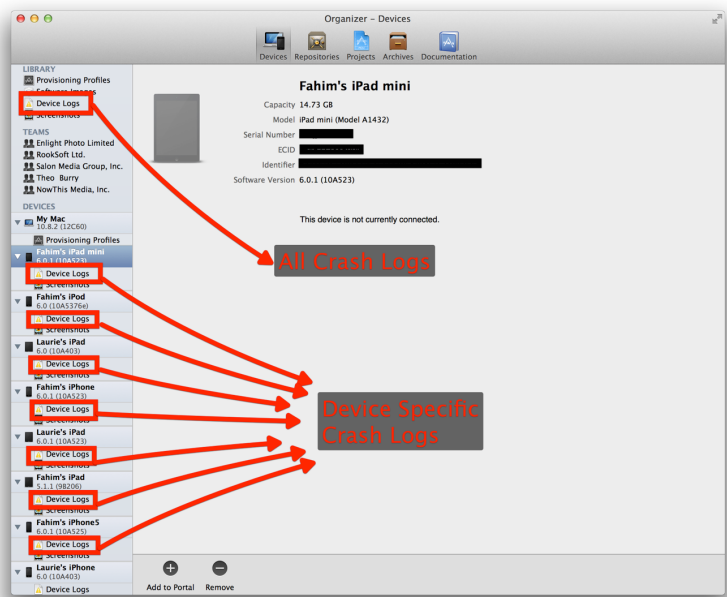
推荐内容

热点内容

当用户抱怨闪退时，你可以要求他让设备与iTunes同步，并根据操作系统的不同，到上述位置把崩溃日志下载下来，然后通过电子邮件发送给你。

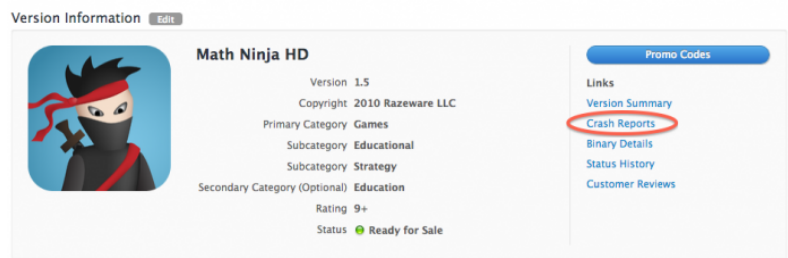
你必需尽量获取用户设备生成的所有崩溃日志。因为崩溃日志越多，就越容易诊断问题所在!

另外，如果你装了Xcode，也能很容易通过Xcode从你的设备上获得崩溃日志。将iOS设备连接到电脑上，然后打开Xcode。从菜单栏上选择 Window 菜单，然后选择 Organizer (快捷方式是 Shift-CMD-2)。在 Organizer 窗口上, 选中 Devices 标签栏。在左侧的导航面板上，选中 Device Logs, 如下图所示:




看看上图，左侧有好几个 Device Logs 菜单项。LIBRARY 下面的Device Logs是你所有设备（曾经连接到Xcode的）的日志。每个设备下面的 Device Logs 是对应设备的日志。

应用提交到App Store后，你也能从 iTunes Connect 获取到用户的崩溃日志。登录到 iTunes Connect 上, 选择 Manage Your Applications, 点击相应的应用, 点击应用图标下面的 View Details 按钮, 然后点击右栏Links部分的 Crash Reports 。





如果没有崩溃日志，试试点击Refresh 按钮刷新一下。如果你的应用还卖得不多，或者刚上架不久，iTunes Connect账号上也可能还没有任何崩溃日志。


如果iTunes Connect上有崩溃日志，你将看到如下图:

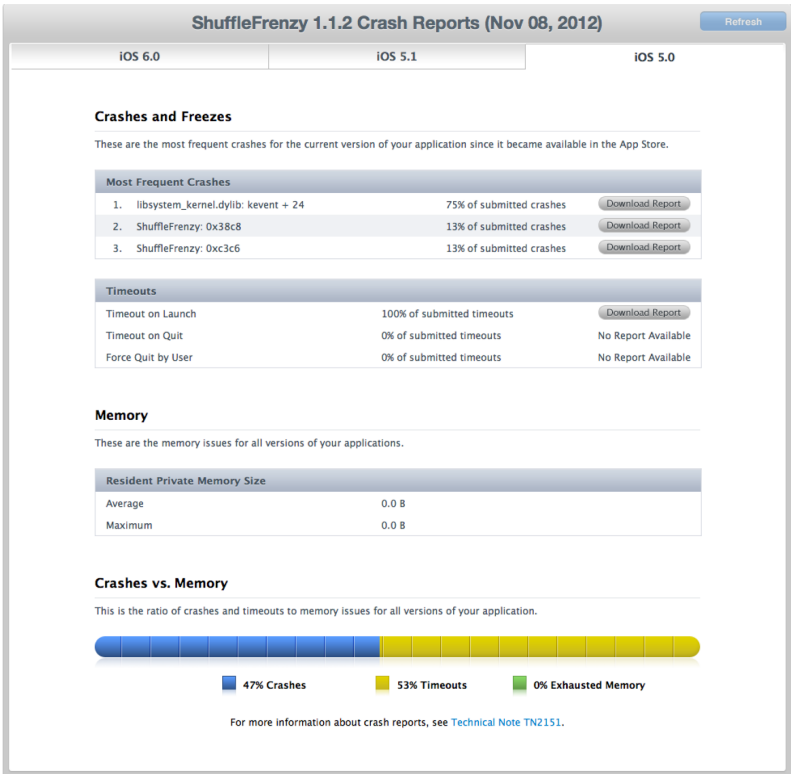
 Testing with Xcode文档(中文版)

 SpriteKit 在iOS8和 OSX10.10 中的新特性

 功能强大的Xcode辅助工具
Faux Pas：帮你找到各种隐形

 Flipboard开源应用内调试工具
FLEX

 iTerm2 2.0版本已发布,添加大量新功能,更易于使用



有时，尽管有用户报告闪退，你仍然看不到崩溃报告。这时，最好让用户直接把崩溃报告发送给你。

什么情况下会产生崩溃日志？

两种主要情况会产生崩溃日志：

- 1.应用违反操作系统规则。
- 2.应用中有Bug。

违反iOS规则包括在启动、恢复、挂起、退出时watchdog超时、用户强制退出和低内存终止。让我们详细了解一下吧。

Watchdog 超时机制

从iOS 4.x开始，退出应用时，应用不会立即终止，而是退到后台。但是，如果你的应用响应不够快，操作系统有可能会终止你的应用，并产生一个崩溃日志。这些事件与下列UIApplicationDelegate方法相对应：

- application:didFinishLaunchingWithOptions:
- applicationWillResignActive:
- applicationDidEnterBackground:
- applicationWillEnterForeground:
- applicationDidBecomeActive:
- applicationWillTerminate:

上面所有这些方法，应用只有有限的时间去完成处理。如果花费时间太长，操作系统将终止应用。

注意: 如果你没有把需要花费时间比较长的操作(如网络访问)放在后台线程上就很容易发生这种情况。关于如果避免这种情况的信息，请参考我们的另外两篇教程：Grand Central Dispatch 和 NSOperations。

用户强制退出

iOS 4.x开始支持多任务。如果应用阻塞界面并停止响应，用户可以通过在主屏幕上双击Home按钮来终止应用。此时，操作应用将生成一个崩溃日志。

注意: 双击Home按钮后，你将看到运行过的所有应用。那些应用不一定是正在运行，也不一定是被挂起。

通常，用户点击Home按钮时，应用将在后台保留约10分钟，然后操作系统自动将其终止。所以双击Home按钮显示的应用列表只是表明那是一系列过去打开过的应用。删除那些应用的图标不会产生任何崩溃日志。

低内存终止

子类化UIViewController时,你或许已经注意到didReceiveMemoryWarning方法。

在前台运行的应用拥有访问和使用内存的最高优化级。然而,这并不意味着该应用能使用设备的所有可用内存——每个应用只能使用一部分可用内存。

当内存使用达到一定程度时,操作系统将发出一个 UIApplicationDidReceiveMemoryWarningNotification 通知。同时,调用 didReceiveMemoryWarning 方法。

此时,为了让应用继续正常运行,操作系统开始终止在后台的其他应用以释放一些内存。所有后台应用被终止后,如果你的应用还需要更多内存,操作系统会将你的应用也终止掉,并产生一个崩溃日志。而在这种情况下被终止的后台应用,不会产生崩溃日志。

注意: 根据 苹果文档, Xcode不会自动添加低内存日志。你必需手动获取日志。然而,根据我的个人经验,使用 Xcode 4.5.2, 低内存日志也会自动导入,只是"Process"和"Type"属性都被标为Unknown(未知)。

另外,值得一提的是在极短时间内分配一大块内存将给系统内存带来巨大负担。这样,也会产生内存警告的通知。

应用中有Bug

如你所想,大多数闪退都是由于应用中有Bug,因此大多数崩溃日志的产生都是因为应用中的Bug。Bug的种类的有很多。

在本教程的后半部分,你将通过调试一个会产生崩溃日志的含有Bug的应用,学习如何找到问题所在并进行修复!

崩溃日志的实例

让我们看看一个崩溃日志的实例,以使你在处理一些实际问题之前心里有谱。

事不宜迟,见见你的新朋友吧:

```
// 1: 进程信息
Incident Identifier: 30E46451-53FD-4965-896A-457FC11AD05F
CrashReporter Key:   5a56599d836c4f867f6eec76a7ee451bf9ae5f31
Hardware Model:      iPhone4,1
Process:             Rage Masters [4155]
Path:                /var/mobile/Applications/A5635B22-F5EF-4CEB-94B6-FE158D885014/Rage Masters.app/Rage Masters
Identifier:          Rage Masters
Version:             ??? (???)
Code Type:           ARM (Native)
Parent Process:      launchd [1]
// 2: 基本信息
Date/Time:           2012-10-17 21:39:06.967 -0400
OS Version:          iOS 6.0 (10A403)
Report Version:      104
// 3: 异常
Exception Type:      00000020
Exception Codes:     0x0000000008badf00d
Highlighted Thread:  0
// 4: 线程回溯
Thread 0 name:       Dispatch queue: com.apple.main-thread
Thread 0:
0  libsystem_kernel.dylib             0x327f2eb4 mach_msg_trap + 20
1  libsystem_kernel.dylib             0x327f3048 mach_msg + 36
2  CoreFoundation                    0x36bd4040 __CFLRunLoopServiceMachPort + 124
3  CoreFoundation                    0x36bd2d9e __CFLRunLoopRun + 878
4  CoreFoundation                    0x36b45eb8 CFLRunLoopRunSpecific + 352
5  CoreFoundation                    0x36b45d44 CFLRunLoopRunInMode + 100
6  CFNetwork                          0x32ac343e CFURLConnectionSendSynchronousRequest + 330
7  Foundation                         0x346e69ba +[NSURLConnection sendSynchronousRequest:returningResponse:error:] + 242
8  Rage Masters                      0x000d4046 0xd2000 + 8262
Thread 1:
0  libsystem_kernel.dylib             0x32803d98 __workq_kernreturn + 8
1  libsystem_c.dylib                  0x3a987cf6 _pthread_workq_return + 14
2  libsystem_c.dylib                  0x3a987a12 _pthread_wqthread + 362
3  libsystem_c.dylib                  0x3a9878a0 start_wqthread + 4
// 5: 线程状态
Thread 0 crashed with ARM Thread State (32-bit):
   r0: 0x00000000   r1: 0x00000000   r2: 0x00000001   r3: 0x39529fc8
   r4: 0xffffffff   r5: 0x2fd7d301   r6: 0x2fd7d300   r7: 0x2fd7d9d0
   r8: 0x2fd7d330   r9: 0x3adb78a8   r10: 0x2fd7d308  r11: 0x00000032
   ip: 0x00000025   sp: 0x2fd7d2ec   lr: 0x001bdb25   pc: 0x30301838
cpsr: 0x00000010
// 6: 二进制映像
Binary Images:
0xd2000 - 0xd7fff +Rage Masters armv7 /var/mobile/Applications/A5635B22-F5EF-4CEB-94B6-FE158D885014/Rage Masters.app/Rage Masters
0x2fe41000 - 0x2fe61fff dyld armv7 /usr/lib/dyld
0x327f2000 - 0x32808fff libsystem_kernel.dylib armv7 /usr/lib/system/libsystem_kernel.dylib
0x328a8000 - 0x328bdfbf libresolv.9.dylib armv7 /usr/lib/libresolv.9.dylib
0x32a70000 - 0x32b3ffff CFNetwork armv7 /System/Library/Frameworks/CFNetwork.framework/CFNetwork
0x32b7a000 - 0x32cc3fff libicucore.A.dylib armv7 /usr/lib/libicucore.A.dylib
0x32cc4000 - 0x32cc5fff CoreSurface armv7 /System/Library/PrivateFrameworks/CoreSurface.framework/CoreSurface
0x32f65000 - 0x32f8afff OpenCL armv7 /System/Library/PrivateFrameworks/OpenCL.framework/OpenCL
```

这报告看起来像天书。:) 我们分几部分来解读吧:

(1) 进程信息

第一部分是闪退进程的相关信息。

Incident Identifier是崩溃报告的唯一标识符。

CrashReporter Key 是与设备标识相对应的唯一键值。虽然它不是真正的设备标识符，但也是一个非常有用的情报:如果你看到100个崩溃日志的CrashReporter Key值都是相同的，或者只有少数几个不同的CrashReporter值，说明这不是一个普遍的问题，只发生在一个或少数几个设备上。

Hardware Model 标识设备类型。 如果很多崩溃日志都是来自相同的设备类型，说明应用只在某特定类型的设备上有问题。上面的日志里，崩溃日志产生的设备是iPhone 4s。

Process 是应用名称。中括号里面的数字是闪退时应用的进程ID。

接下来几行不言自明，无需赘述。

(2) 基本信息

这部分给出了一些基本信息，包括闪退发生的日期和时间，设备的iOS版本。如果有很多崩溃日志都来自iOS 6.0，说明问题只发生在iOS 6.0上。

(3) 异常

在这部分，你可以看到闪退发生时抛出的异常类型。还能看到异常编码和抛出异常的线程。根据崩溃报告类

型的不同，在这部分你还能看到一些另外的信息。

(4) 线程回溯

这部分提供应用中所有线程的回溯日志。回溯是闪退发生时所有活动帧清单。它包含闪退发生时调用函数的清单。看下面这行日志：

```
2 XYZLib 0x34648e88 0x83000 + 8740
```

它包括四列：

帧编号——此处是2。

二进制库的名称——此处是 XYZLib。

调用方法的地址——此处是 0x34648e88。

第四列分为两个子列，一个基本地址和一个偏移量。此处是0x83000 + 8740，第一个数字指向文件，第二个数字指向文件中的代码行。

(5) 线程状态

这部分是闪退时寄存器中的值。一般不需要这部分的信息，因为回溯部分的信息已经足够让你找出问题所在。

(6) 二进制映像

这部分列出了闪退时已经加载的二进制文件。

符号化Symbolication

第一次看到崩溃日志上的回溯时，你或许会觉得它没什么意义。我们习惯使用方法名和行数，而非像这样的神秘位置：

```
6 Rage Masters 0x0001625c 0x2a000 + 3003
```

将这些十六进制地址转化成方法名称和行数的过程称之为符号化。

从Xcode的Organizer窗口获取崩溃日志后过几秒钟，崩溃日志将被自动符号化。上面那行被符号化后的版本如下：

```
6 Rage Masters 0x0001625c -[RAppDelegate application:didFinishLaunchingWithOptions:] (RAppDelegate.m:35)
```

Xcode符号化崩溃日志时，需要访问与App Store上对应的应用二进制文件以及生成二进制文件时产生的 .dSYM 文件。必需完全匹配才行。否则，日志将无法被完全符号化。

所以，保留每个分发给用户的编译版本非常重要。提交应用前进行归档时，Xcode将保存应用的二进制文件。可以在Xcode Organizer的Archives标签栏下找到所有已归档的应用文件。

在发现崩溃日志时，如果有相匹配的.dSYM文件和应用二进制文件，Xcode会自动对崩溃日志进行符号化。如果你换到别的电脑或创建新的账户，务必将所有二进制文件移动到正确的位置，使Xcode能找到它们。

(**注意**: 你必需同时保留应用二进制文件和.dSYM文件才能将崩溃日志完整符号化。每次提交到iTunes Connect的构建都必需归档。

.dSYM文件和二进制文件是特定绑定于每一次构建和后续构建的，即使来自相同的源代码文件，每一次构建也与其他构建不同，不能相互替换。

如果你使用Build 和 Archive 命令,这些文件会自动放在适当位置。如果不是使用Build 和 Archive命令，放在Spotlight能够搜索到的位置（比如Home目录）即可。）

低内存闪退

因为低内存崩溃日志与普通崩溃日志略有不同，所以本教程特别分开说明一下。

iOS设备检测到低内存时，虚拟内存系统发出通知请求应用释放内存。这些通知发送到所有正在运行的应用和进程，试图收回一些内存。

如果内存使用依然居高不下，系统将会终止后台线程以缓解内存压力。如果可用内存足够，应用将能够继续运行而不会产生崩溃报告。否则，应用将被iOS终止，并产生低内存崩溃报告。

低内存崩溃日志上没有应用线程的堆栈回溯。相反，上面显示的是以内存页数为单位各进程的内存使用量。(在撰写本文的时候，一个内存页的大小是4KB。)

被iOS因释放内存页终止的进程名称后面你会看到jettisoned 字样。如果看到它出现在你的应用名称后面，说明你的应用因使用太多内存而被终止了。

低内存崩溃日志看起来像这样：

```
Incident Identifier: 30E46451-53FD-4965-896A-457FC11AD05F
CrashReporter Key:   5a56599d836c4f867f6eec76afee451bf9ae5f31
OS Version:          iPhone OS 3.1.3 (7E18)
Date/Time:            2012-10-17 21:39:06.967 -0400
Free pages:          96
Wired pages:          10558
Purgeable pages:      0
Largest process:      Rage Masters
Processes
  Name                      UUID                      Count resident pages
  Rage Masters              9320 (jettisoned) (active)
  mediaserverd              255
  dataaccessd              505
  syslogd                   71
  apsd                      171
  securitlyd                243
  noLifYd                   2027
  CommCenter               189
  SpringBoard              2158 (active)
  accessoryd                91
  configd                   371
  fairplayd                 93
  mDNSResponder             292
  lockdown                 1204
  launchd                   72
```

当应用发生低内存闪退时，你必需看看应用中内存使用的方式，以及是如何处理低内存警告的。你可以使用 Instruments 工具中使用 Allocations 和 Leaks 来发现内存分配问题和内存泄漏问题。如果你不知道如何利用 Instruments 检查内存问题，可以看看这个教程。

还有，别忘记虚拟内存！ Instruments 工具的 Leaks 和 Allocations 不能跟踪显存使用情况。必需使用 VM Tracker 才能查看显存使用情况。

VM Tracker 默认是关闭的。打开 Instrument, 手动 选中 Automatic Snapshotting 标志或者按下 Snapshot Now 按钮。

本教程后面将会学习如何研究低内存崩溃日志。

异常编码

在研究真实闪退场景之前，还有一点需要重点介绍一下：就是那些有趣的异常编码。

你可以在报告的异常部分——前面代码的第3部分找到异常编码。有些编码比较常见。

通常，异常编码以一些文字开头，紧接着是一个或多个十六进制值，此数值正是说明闪退根本性质的所在。从这些编码中，可以区分出闪退是因为程序错误、非法内存访问或者是其他原因。下面是一些常见的异常编码：

0x8badf00d: 读做“ate bad food”! (把数字换成字母，是不是很像 :p) 该编码表示应用是因为发生 watchdog 超时而被 iOS 终止的。通常是应用花费太多时间而无法启动、终止或响应系统事件。



I 8 BADFOOD

0xbad22222: 该编码表示 VoIP 应用因为过于频繁重启而被终止。

0xdead10cc: 读做 “dead lock”!该代码表明应用因为在后台运行时占用系统资源，如通讯录数据库不释放而被终止。

0xdeadfa11: 读做 “dead fall”! 该代码表示应用是被用户强制退出的。根据苹果文档, 强制退出发生在用户长按开关按钮直到出现 “滑动来关机”, 然后长按 Home按钮。强制退出将产生 包含0xdeadfa11 异常编码的崩溃日志, 因为大多数是强制退出是因为应用阻塞了界面。

(**注意:** 在后台任务列表中关闭已挂起的应用不会产生崩溃日志。一旦应用被挂起, 它何时被终止都是合理的。所以不会产生崩溃日志。)

大展身手的时候到了!好了! 你已经学习了所有分析崩溃日志和修复错误的基础知识!

假设你刚进入Rage-O-Rage有限公司工作。该公司有一个在App Store上热销的应用, 叫 **Rage Masters**。

你的老板安迪要你帮忙解决几个用户经常抱怨闪退问题。你的任务就是研究这些闪退, 符号化用户提供的崩溃日志, 查找问题所在, 并修复之。

你可以从 [这里](#)下载应用的源代码。

(**注意:** 如果你想自己重新生成崩溃报告, 请遵照以下指引:

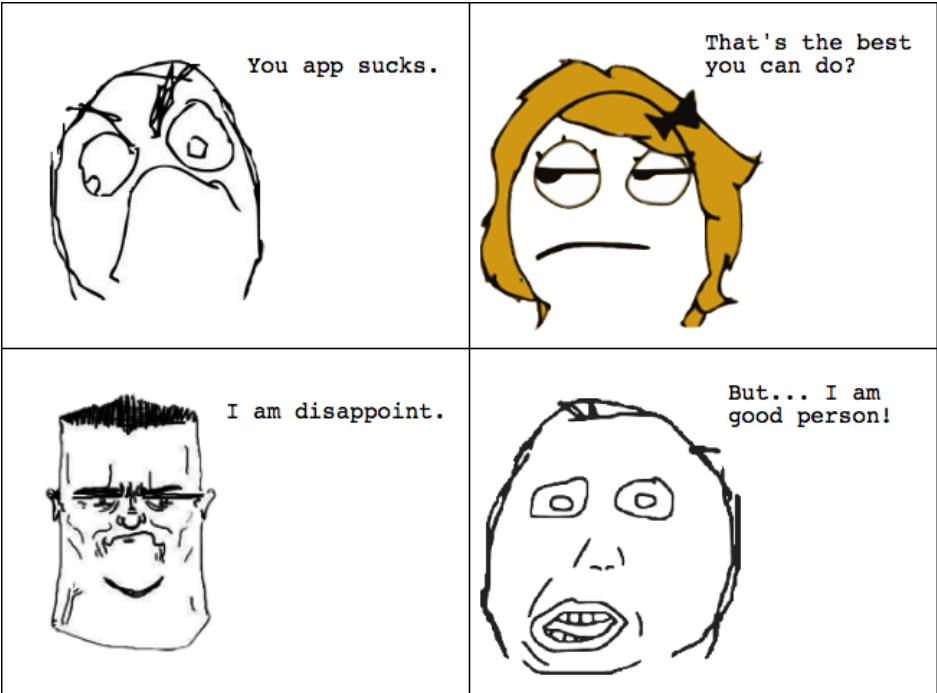
- 1.下载源码然后在Xcode中打开工程文件。
- 2.使用正确的provisioning profile连接到iOS设备。
- 3.从Xcode工具栏上选择iOS设备——不是模拟器作为target, 然后构建应用。
- 4.当你在设备上到默认页面(应用的全屏图片)时, 立即在Xcode上点击停止按钮。
- 5.关闭 Xcode。
- 6.在设备上直接打开应用。
- 7.测试场景, 完成后连接设备到电脑上, 通过Xcode获取崩溃日志。)

场景 1: 糟糕的代码

一封来自用户的邮件: “大哥, 你的应用就是一坨屎! 我将其下载到我自己的iPod Touch和iPhone上, 还下载到儿子的iPod Touch上。在所有的设备上, 都是还没打开就闪退了……”

别一封来自用户的邮件说, “我下载了你们的应用, 一打开就闪退。真悲催...”

另一封邮件说得更明确: “你们的应用不能运行。我把它下载到我和妻子的设备上。所有设备都是一打开就闪退了...”



好吧，别灰心! 这些意见藏着什么玄机呢？让我们看看崩溃日志吧:

```
Incident Identifier: 85833DBA-3DF7-43EE-AF80-4E5C51091F42
CrashReporter Key:   5a56599d836c4f867f6eec76afee451bf9ae5f31
Hardware Model:      iPhone4,1
Process:             Rage Masters [20067]
Path:                /var/mobile/Applications/B2121A89-3D1F-4E61-BB18-5511E1DC150F/Rage Masters.app/Rage Masters
Identifier:          Rage Masters
Version:             ??? (???)
Code Type:           ARM (Native)
Parent Process:      launchd [1]
Date/Time:           2012-11-03 13:37:31.148 -0400
OS Version:          iOS 6.0 (10A403)
Report Version:      104
Exception Type:      00000020
Exception Codes:     0x0000000008badf00d
Highlighted Thread:  0
Application Specific Information:
Soheil-Azarpour.Rage-Masters failed to launch in Time
Elapsed total CPU Time (seconds): 8.030 (user 8.030, system 0.000), 20% CPU
Elapsed application CPU Time (seconds): 3.840, 10% CPU
Thread 0 name: Dispatch queue: com.apple.main-thread
Thread 0:
0  libsystem_kernel.dylib      0x327f2eb4 mach_msg_trap + 20
1  libsystem_kernel.dylib      0x327f3048 mach_msg + 36
2  CoreFoundation              0x36bd4040 __CFRunLoopServiceMachPort + 124
3  CoreFoundation              0x36bd2d9e __CFRunLoopRun + 878
4  CoreFoundation              0x36b45eb8 CFRunLoopRunSpecific + 352
5  CoreFoundation              0x36b45d44 CFRunLoopRunInMode + 100
6  CFNetwork                   0x32ac343e CFURLConnectionSendSynchronousRequest + 330
7  Foundation                  0x346e69ba +[NSURLConnection sendSynchronousRequest:returningResponse:error:] + 242
8  Rage Masters                0x000ea1c4 -[RMAppDelegate application:didFinishLaunchingWithOptions:]
(RMAppDelegate.m:36)
9  UIKit                       0x37f30ad4 -[UIApplication
_handleDelegateCallbacksWithOptions:isSuspended:restoreState:] + 248
10 UIKit                      0x37f3065e -[UIApplication _callInitializationDelegatesForURL:payload:suspended:] + 1186
11 UIKit                      0x37f28846 -[UIApplication
_runWithURL:payload:launchOrientation:statusBarStyle:statusBarHidden:] + 694
12 UIKit                      0x37ed0c3c -[UIApplication handleEvent:withNewEvent:] + 1000
13 UIKit                      0x37ed06d0 -[UIApplication sendEvent:] + 68
14 UIKit                      0x37ed011e _UIApplicationHandleEvent + 6150
15 GraphicsServices           0x370835a0 _PurpleEventCallback + 588
16 GraphicsServices           0x370831ce PurpleEventCallback + 30
17 CoreFoundation             0x36bd4170 __CFRUNLOOP_IS_CALLING_OUT_TO_A_SOURCE1_PERFORM_FUNCTION__ + 32
18 CoreFoundation             0x36bd4112 __CFRunLoopDoSource1 + 134
19 CoreFoundation             0x36bd2f94 __CFRunLoopRun + 1380
20 CoreFoundation             0x36b45eb8 CFRunLoopRunSpecific + 352
21 CoreFoundation             0x36b45d44 CFRunLoopRunInMode + 100
22 UIKit                      0x37f27480 -[UIApplication _run] + 664
23 UIKit                      0x37f242fc UIApplicationMain + 1116
24 Rage Masters                0x000ea004 main (main.m:16)
25 libdyld.dylib              0x3b630b1c start + 0
```

发现问题了吗？异常编码是0x000000008badf00d,还有后面的报告:

```
Application Specific Information:
Soheil-Azarpour.Rage-Masters failed to launch in Time
Elapsed total CPU Time (seconds): 8.030 (user 8.030, system 0.000), 20% CPU
Elapsed application CPU Time (seconds): 3.840, 10% CPU
```

这说明应用在启动时就闪退了，iOS的watchdog机制终止了应用。帅！找到问题了，但是为什么会发生这样的事呢？

接着往下看日志。从下向上读回溯日志。最底下的帧 (frame 25: libdyld.dylib)是最先调用的，然后是帧24, Rage Masters, main (main.m:16)，依此类推。

跟应用源代码相关的帧是最重要的。忽略掉系统库和框架。下一个与代码相关的帧是：

```
8  Rage Masters  0x0009f244 -[RAppDelegate application:didFinishLaunchingWithOptions:] (RAppDelegate.m:35)
```

应用在执行RAppDelegate (RAppDelegate.m:35)类application:didFinishLaunchingWithOptions: 方法第35行代码时闪退。打开Xcode看看那行代码：

```
NSData *directoryData = [NSURLConnection sendSynchronousRequest:request returningResponse:nil error:nil];
```

就是它了！同步调用web服务?! 在主线程上?! 在 application:didFinishLaunchingWithOptions: 方法上?! 谁写的代码呀?!



Network calls on the main thread makes kittens sad.

不管如何，问题得你来修复了。这个调用必需异步进行，甚至更理想的情况是，在application:didFinishLaunchingWithOptions:返回YES之后的其他部分再执行Web服务。

在其他地方调用可能需要比较多的修改。当下，我们只要使应用不闪退就行。可以在日后再实现更好的设计。将上面那行讨厌的代码（及其下面的三行代码）换成下面这个异步的版本吧：

```
[NSURLConnection sendAsynchronousRequest:request
                    queue:[NSOperationQueue mainQueue]
                    completionHandler:^(NSURLResponse *response, NSData *data, NSError *error)
{
    NSURL *cacheDirectory = [[[NSFileManager defaultManager] URLsForDirectory:NSUserDirectory
    inDomains:NSCachesDirectory] lastObject];
    NSURL *filePath = [NSURL URLWithString:kDirectoryFile relativeToURL:cacheDirectory];
    [data writeToFile:[filePath absoluteString] atomically:YES];
}];
```

场景 2: 无法响应事件的按钮

一名用户说：“我不能将某个rage master添加到书签里面。我想添加的时候应用就闪退...”

用一名用户说：“书签不能用... 在详细页面上，点击书签按钮,应用就闪退了!”

上面的抱怨说得不是很清楚，引起问题的原因肯定有多样。看看崩溃日志：

```

Incident Identifier: 3AA63CC-3088-41CC-84D9-82FE03F9F354
CrashReporter Key:   5a56599d836c4f867f6e6c76af6e451bf9ae5f31
Hardware Model:      iPhone4,1
Process:             Rage Masters [20090]
Path:                /var/mobile/Applications/B2121A89-3D1F-4E61-BB18-5511E1DC150F/Rage Masters.app/Rage Masters
Identifier:          Rage Masters
Version:             ??? (???)
Code Type:           ARM (Native)
Parent Process:      launchd [1]
Date/Time:           2012-11-03 13:39:00.081 -0400
OS Version:          iOS 6.0 (10A403)
Report Version:      104
Exception Type:      EXC_CRASH (SIGABRT)
Exception Codes:     0x0000000000000000, 0x0000000000000000
Crashed Thread:      0
Last Exception Backtrace:
0  CoreFoundation      0x36b1f29e __exceptionPreprocess + 158
1  libobjc.A.dylib     0x34f0f97a objc_exception_throw + 26
2  CoreFoundation      0x36c02e02 -[NSObject(NSObject) doesNotRecognizeSelector:] + 166
3  CoreFoundation      0x36c0152c ___forwarding___ + 388
4  CoreFoundation      0x36b58f64 _CF_forwarding_prep_0 + 20
5  UIKit               0x37fbb0a8 -[UIApplication sendAction:to:from:forEvent:] + 68
6  UIKit               0x37fbb05a -[UIApplication sendAction:toTarget:fromSender:forEvent:] + 26
7  UIKit               0x37fbb038 -[UIControl sendAction:to:forEvent:] + 40
8  UIKit               0x37fba8ee -[UIControl(Internal) _sendActionsForEvents:withEvent:] + 498
9  UIKit               0x37fbade4 -[UIControl touchesEnded:withEvent:] + 484
10 UIKit               0x37ee35f4 -[UIWindow _sendTouchesForEvent:] + 520
11 UIKit               0x37ed0804 -[UIApplication sendEvent:] + 376
12 UIKit               0x37ed011e _UIApplicationHandleEvent + 6150
13 GraphicsServices   0x3708359e _PurpleEventCallback + 586
14 GraphicsServices   0x370831ce PurpleEventCallback + 30
15 CoreFoundation     0x36bd416e __CFRUNLOOP_IS_CALLING_OUT_TO_A_SOURCE1_PERFORM_FUNCTION__ + 30
16 CoreFoundation     0x36bd4112 __CFRunLoopDoSource1 + 134
17 CoreFoundation     0x36bd2f94 __CFRunLoopRun + 1380
18 CoreFoundation     0x36b45eb8 CFRunLoopRunSpecific + 352
19 CoreFoundation     0x36b45d44 CFRunLoopRunInMode + 100
20 GraphicsServices   0x370822e6 GSEventRunModal + 70
21 UIKit               0x37f242fc UIApplicationMain + 1116
22 Rage Masters       0x000ca004 main (main.m:16)
23 libdyld.dylib      0x36b30b1c start + 0

```

异常代码是SIGABRT。通常，SIGABRT 异常是由于某个对象接收到未实现的消息引起的。或者，用简单的话说，在某个对象上调用了不存在的方法。

这种情况一般不会发生，因为A对象调用了B方法，如果B方法不存在，编译器会报错。但是，如果你是使用selector间接调用方法的，编译器则无法检测对象是否存在该方法了。

回到崩溃日志。它指出闪退发生在编号为0的线程上。这意味着很可能是在主线程上调用了某个对象没有实现的方法。

如果你接着阅读回溯日志，会发现跟你的代码相关的只有帧22，main.m:16。这没有多大帮助。

继续向上查看框架调用，出现这个：

```

2  CoreFoundation      0x36c02e02 -[NSObject(NSObject) doesNotRecognizeSelector:] + 166

```

这不是你自己写的代码。但至少它确认了是对象调用了没有实现的方法。

回到RMDetailViewController.m文件，因为那是书签按钮实现动作的地方。找到书签功能代码：

```

-(IBAction)bookmarkButtonPressed {

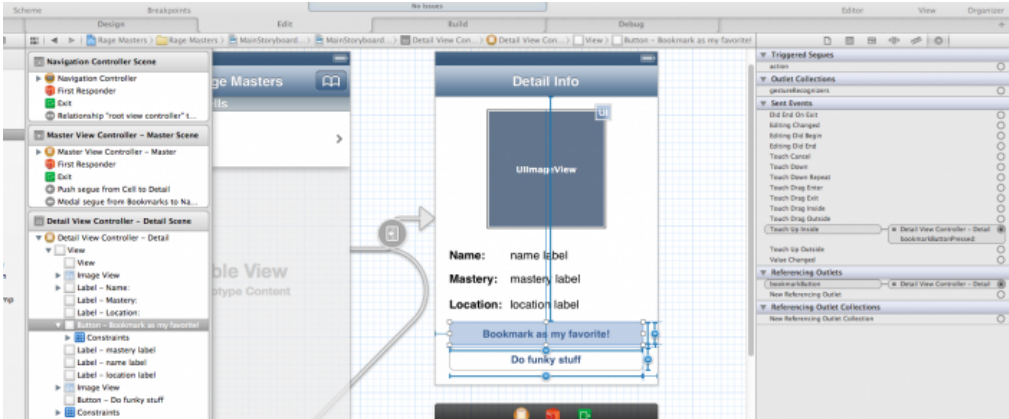
    self.master.isBookmarked = !self.master.isBookmarked;

    // Update shared bookmarks
    if (self.master.isBookmarked)
        [[RMBBookmarks sharedBookmarks] bookmarkMaster:self.master];
    else
        [[RMBBookmarks sharedBookmarks] unbookmarkMaster:self.master];

    // Update UI
    [self updateBookmarkImage];
}

```

看起来没什么问题，再检查一下storyboard (XIB文件)，确认按钮连接的正确性。



就是它了! 在 MainStoryboard.storyboard,按钮连接的是 bookmarkButtonPressed: 而不是bookmarkButtonPressed (注意后面的分号说明方法有一个参数)。 只要将上面的方法签名修改成这样就能修复问题了:

```
-(IBAction)bookmarkButtonPressed:(id)sender {  
    // Remain unchanged..  
}
```

当然,你也可以简单地在XIB文件上删除错误的连接,然后重新连接方法,使XIB文件连接到正确的方法上。两者方法都行。

又处理了一个闪退问题,好样的。

场景 3: 表格上的Bug

另一用户抱怨道,“在书签视图上无法删除书签...” 还有另一用户抱怨同样的问题,“当我试图删除书签时,应用闪退...”

这些邮件没什么作用,还是看看崩溃日志!

```
Incident Identifier: 5B62D681-D8FE-41FE-8D52-AB7E6D6B2AC7
CrashReporter Key: 5a56599d836c4f867f6eec76afee451bf9ae5f31
Hardware Model: iPhone4,1
Process: Rage Masters [20088]
Path: /var/mobile/Applications/B2121A89-3D1F-4E61-BB18-551E1DC150F/Rage Masters.app/Rage Masters
Identifier: Rage Masters
Version: ??? (???)
Code Type: ARM (Native)
Parent Process: launchd [1]
Date/Time: 2012-11-03 13:38:45.762 -0400
OS Version: iOS 6.0 (10A403)
Report Version: 104
Exception Type: EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000
Crashed Thread: 0
Last Exception Backtrace:
0 CoreFoundation 0x36bf729e __exceptionPreprocess + 158
1 libobjc.A.dylib 0x34f0f97a objc_exception_throw + 26
2 CoreFoundation 0x36bf7f18 +[NSException raise:format:arguments:] + 96
3 Foundation 0x346812aa -[NSAssertionHandler
handleFailureInMethod:object:file:lineNumber:description:] + 86
4 UIKit 0x37f04b7e -[UITableView(_UITableViewPrivate) _endCellAnimationsWithContext:] + 7690
5 UIKit 0x3803a4a2 -[UITableView deleteRowsAtIndexPaths:withRowAnimation:] + 22
6 Rage Masters 0x000fd9ca -[RMBBookmarksViewController tableView:commitEditingStyle:forRowAtIndexPath:]
(RMBBookmarksViewController.m:68)
7 UIKit 0x3809a5d4 -[UITableView(UITableViewInternal) animateDeletionOfRowWithCell:] + 80
8 UIKit 0x37fbb0a8 -[UIApplication sendAction:to:from:forEvent:] + 68
9 UIKit 0x37fbb05a -[UIApplication sendAction:toTarget:fromSender:forEvent:] + 26
10 UIKit 0x37fbb038 -[UIControl sendAction:to:forEvent:] + 40
11 UIKit 0x37fba8ee -[UIControl(Internal) _sendActionsForEvents:withEvent:] + 498
12 UIKit 0x37fbb0a8 -[UIApplication sendAction:to:from:forEvent:] + 68
13 UIKit 0x37fbb05a -[UIApplication sendAction:toTarget:fromSender:forEvent:] + 26
14 UIKit 0x37fbb038 -[UIControl sendAction:to:forEvent:] + 40
15 UIKit 0x37fba8ee -[UIControl(Internal) _sendActionsForEvents:withEvent:] + 498
16 UIKit 0x37fbade4 -[UIControl touchesEnded:withEvent:] + 484
17 UIKit 0x37ee35f4 -[UIWindow _sendTouchesForEvent:] + 520
18 UIKit 0x37ed0804 -[UIApplication sendEvent:] + 376
19 UIKit 0x37ed011e _UIApplicationHandleEvent + 6150
20 GraphicsServices 0x3708359e _PurpleEventCallback + 586
21 GraphicsServices 0x370831ce PurpleEventCallback + 30
22 CoreFoundation 0x36bd416e __CFRunLoop_IS_CALLING_OUT_TO_A_SOURCE1_PERFORM_FUNCTION__ + 30
23 CoreFoundation 0x36bd4112 __CFRunLoopDoSource1 + 134
24 CoreFoundation 0x36bd2f94 __CFRunLoopRun + 1380
25 CoreFoundation 0x36b45eb8 CFRunLoopRunSpecific + 352
26 CoreFoundation 0x36b45d44 CFRunLoopRunInMode + 100
27 GraphicsServices 0x370822e6 GSEventRunModal + 70
28 UIKit 0x37f242fc UIApplicationMain + 1116
29 Rage Masters 0x000fb004 main (main.m:16)
30 libdyld.dylib 0x36b630b1c start + 0
```

这看起来跟前面那个崩溃日志很像。是另一个SIGABRT 异常。 你可能想知道是否是相同的问题：发送信息到一个没有实现相应方法的对象？

让我们从回溯日志看看哪些方法被调用了。从底部开始，你的源代码最后被调用的是帧 6：

```
6 Rage Masters 0x00088c66 -[RMBBookmarksViewController tableView:commitEditingStyle:forRowAtIndexPath:]
(RMBBookmarksViewController.m:68)
```

这是UITableViewDataSource 的一个方法。呵呵?! 毫无疑问苹果已经实现了该方法 —— 你可以重载它，但不像是还没有实现。而且,这是个可选的委派方法。 所以问题不是调用了 一个没有实现的方法。

再看看上面的几个帧：

```
3 Foundation 0x346812aa -[NSAssertionHandler handleFailureInMethod:object:file:lineNumber:description:] + 86
4 UIKit 0x37f04b7e -[UITableView(_UITableViewPrivate) _endCellAnimationsWithContext:] + 7690
5 UIKit 0x3803a4a2 -[UITableView deleteRowsAtIndexPaths:withRowAnimation:] + 22
```

帧 5, UITableView调用了它自己的另一个方法 deleteRowsAtIndexPaths:withRowAnimation: 然后是看起来像苹果内部方法的_endCellAnimationsWithContext: 被调用。然后Foundation framework发生异常handleFailureInMethod:object:file:lineNumber:description:.

这些分析结合用户的抱怨，看起来是你在处理UITableView删除行过程中有Bug。回到Xcode。你知道看哪里吗？能从崩溃日志中判断出来？就是RMBBookmarksViewController.m文件的第68行：

```
- (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:
(NSIndexPath *)indexPath {

    [self.tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
withRowAnimation:UITableViewRowAnimationAutomatic];
}
```

发现问题了吗？给你点时间，仔细看一下。

找到了吧! 数据源呢? 代码在表格视图上删除了一行, 但并没有修改背后的数据源。把上面的代码替换成下面的就能修复问题了:

```
-(void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:
(NSIndexPath *)indexPath {

    RMMaster *masterToDelete = [bookmarks objectAtIndex:indexPath.row];
    [bookmarks removeObject:masterToDelete];
    [[RMBookmarks sharedBookmarks] unbookmarkMaster:masterToDelete];

    [self.tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath]
    withRowAnimation:UITableViewRowAnimationAutomatic];
}
```

搞定了! 走起, 讨厌的 bug!!

场景 4: 吃棒棒糖时闪退!

用户邮件说, “当rage master吃棒棒糖时应用就闪退...” 另一用户说, “我让rage master 吃棒棒糖, 没几次应用就闪退了!”

崩溃日志如下:

```
Incident Identifier: 081E58F5-95A8-404D-947B-5E104B6BC1B1
CrashReporter Key:   5a56599d836c4f867f6eec76afee451bf9ae5f31
Hardware Model:       iPhone4,1
OS Version:           iPhone OS 6.0 (10A403)
Kernel Version:       Darwin Kernel Version 13.0.0: Sun Aug 19 00:28:05 PDT 2012; root:xnu-
2107.2.33~4/RELEASE_ARM_S5L8940X
Date:                  2012-11-03 13:39:59 -0400
Time since snapshot:  4353 ms
Free pages:           968
Active pages:         7778
Inactive pages:       4005
Throttled pages:      92319
Purgeable pages:      0
Wired pages:          23347
Largest process:      Rage Masters
```


Processes Name (state)	<UID>	rpages	recent_max	[reason]
lsd <6a9f5b5f36b23fc78f87b6d8f1f49a9d> (daemon) (idle)		331	331	[vm]
aftcd <b0aff2e7952e34a9882fec81a8dcdbb2> (daemon) (idle)		141	141	[vm]
itunesstored <4e0cd9f873de3435b4119c48b2d6d13d> (daemon) (idle)		1761	1761	[vm]
softwareupdates <2bc4b5ae016431c98d3b34f81027d0ae> (daemon) (idle)		311	311	[vm]
Amazon <4600481f07ec3e59a925319b7f67ba14> (suspended)		2951	2951	[vm]
accountsd <ac0fce15c1a2350d951efc498d521ac7> (daemon) (idle)		519	519	[vm]
coresymbolicatio <edba67001f76313b992056c712153b4b> (daemon) (idle)		126	126	[vm]
Skype <504cf2fe60cb3cdea8273e74df09836b> (background)		3187	3187	[vm]
MobileMail <bff817c61ce33c85a43ea9a6c98c29f5> (continuous)		14927	14927	[vm]
MobileSMS <46778de076363d67aeea207464cf581> (background)		2134	2134	[vm]
MobilePhone <3fca241f2a193d0fb8264218d296ea41> (continuous)		2689	2689	[vm]
librariand <c9a9be81aa9632f0a913ce79b911f27e> (daemon)		317	317	[vm]
kbd <3e7136ddcefc3d77a01499db593466cd> (daemon)		616	616	[vm]
lccd <eb5ddcf533663f8d987d67cae6a4c4ea> (daemon)		224	224	[vm]
Rage Masters <90b45d6281e934209c5b06cf7dc4d492> (frontmost) (resume)		28591	28591	[vm]
ptpd <04a56fce67053c57a7979aeea8e5a7ea> (daemon)		879	879	
iaptransportd <f784f30dc09d32078d87b450e8113ef6> (daemon)		230	230	
locationd <892cd1c9ffa43c99a82dba197be5f09e> (daemon)		1641	1641	
syslogd <cbeff142fa0a839f0885afb693fb169c3> (daemon)		237	237	
mediaserverd <80657170daca32c9b8f3a6b1faac43a2> (daemon)		4869	4869	
dataaccessd <2a3f6a518f3f3646bf35eddd36f25005> (daemon)		1786	1786	
aosnotifyd <d4d14f2914c3343796e447cfef3e6542> (daemon)		549	549	
wifid <9472b090746237998cddb9b34f090d0c> (daemon)		455	455	
SpringBoard <27372aae101f3bbc87804edc10314af3> backboardd <5037235f295b33eda98eb5c72c098858>		18749 5801	18749 5801	

(daemon)		
UserEventAgent <1;6edfd8d8ba23187b05772dcdfc94f90>	601	601
(daemon)		
mediaremoted <1;4ff39c50c684302492e396ace813cb25>	293	293
(daemon)		
pasteboardd <1;8a4279b78e4a321f84a076a711dc1c51>	176	176
(daemon)		
springboardservi <1;ff6f64b3a21a39c9a1793321eeffa5304>	0	0
(daemon)		
syslog_relay <1;45e9844605d737a08368b5215bb54426>	0	0
(daemon)		
DTMobileIS <1;23303ca402aa3705870b01a9047854ea>	0	0
(daemon)		
notification_pro <1;845b7beebc8538ca9cee731031983b7>	169	169
(daemon)		
syslog_relay <1;45e9844605d737a08368b5215bb54426>	0	0
(daemon)		
ubd <1;74dc476d1785300e9fcd555fcb8d774>	976	976
(daemon)		
twillerd <1;4b4946378a9c397d8250965d17055b8e>	730	730
(daemon)		
configd <1;4245d73a9e96360399452cf6b8671844>	809	809
(daemon)		
absinthed.N94 <1;7f4164c844fa340caa940b863c901aa9>	99	99
(daemon)		
filecoordination <1;fbab576f37a63b56a1039153fc1aa7d8>	226	226
(daemon)		
distnoted <1;a89af76ec8633ac2bbe99bc2b7964bb0>	137	137
(daemon)		
apsd <1;94d8051dd5f5362f82d775bc279ae608>	373	373
(daemon)		
networkd <1;0032f46009f53a6c80973fe153d1a588>	219	219
(daemon)		
aggregated <1;8c3c991dc4153bc38aee1e841864d088>	112	112
(daemon)		
BTServer <1;c92fbd7488e63be99ec9dbd05824f5e5>	522	522
(daemon)		
fairplayd.N94 <1;7bd896bd00783a4890609d05cf1c86a>	210	210
(daemon)		
fseventsd <1;996cc4ca03793184aea8d781b55bce08>	384	384
(daemon)		
imagent <1;1e68080947be352590ce96b7a1d07b2f>	586	586
(daemon)		
mDNSResponder <1;3e557693f3073697a58da6d27a827d97>	295	295
(daemon)		
lockdownd <1;ba1358c7a8003f1b91af7d5f58dd5bbe>	389	389
(daemon)		
powerd <1;2d2ffed5e69638aeba1b92ef124ed861>	174	174
(daemon)		
CommCenter <1;1f425e1e897d32e8864fdd8eaa803a8>	2212	2212
(daemon)		
notifyd <1;51c0e03da8a93ac8a595442fcaac531f>	211	211
(daemon)		
ReportCrash <1;8c32f231b2ed360bb151b2563bcaa363>	337	337
(daemon)		

这日志跟我们前面见到的相差很多。

这个一个来自iOS 6的低内存崩溃日志。正如我们前面所说的，低内存崩溃日志与其他类型的崩溃日志很不一样，它们不指向特定的文件和代码行。相反，它们画出了闪退时设备上的内存使用情况的图表。

至少，头部还是跟其他崩溃日志很像的：提供了 Incident Identifier, CrashReporter Key, Hardware Model, OS Version等信息。

接下来部分是低内存崩溃日志特有的：

Free pages 指可用内存页数。每页大小约是4KB, 上面的日志中，可用内存约为3,872 KB (或者说 3.9 MB)。

Purgeable pages 是那部分可被清除或重用的内存。在上面的日志中，是0KB。

Largest process是闪退时使用大部分内存的应用名称，在上面的日志中，正是你的应用!

Processes显示了闪退时各进程列表，还包含内存使用量。包含进程名 (第一列), 进程唯一标识符(第二名), 进程使用的内存页数(第三列)。最后一列是每个应用的状态。通常，发生闪退的应用的状态是 frontmost。 这里是 Rage Masters, 使用28591 页 (or 114.364 MB) 内存——这内存太多了!

通过，最大进程和frontmost状态的应用是相同的，而且也是引起低内存闪退的应用进程。但是也可能看到最大进程和 frontmost状态应用不同的例子。比如，如果最大进程是SpringBoard, 忽略它，因为 SpringBoard 进程是显示主屏幕的应用，出现在你双击home按钮等情况，而且它是一直活动的。

低内存发生时，iOS向活动的应用发出低内存警告并终止后台应用。如果前台应用仍然继续增长内存，iOS将终止它。

为了查找低内存问题的原因，你必需使用Instruments剖析应用。如果你不知道怎么做，可以看一下我们 一篇

关于这个方面的教程。:] 另外,你也可以走捷径,响应低内存警告通知,以解决部分闪退问题。

回到Xcode查看RMLollipopLicker.m文件。这是实现吃棒棒糖的视图控制器。看看源代码:

```
#import "RMLollipopLicker.h"

#define COUNT 20

@interface RMLollipopLicker ()
@property (weak, nonatomic) IBOutlet UIProgressView *progressView;
@property (weak, nonatomic) IBOutlet UILabel *label;
@property (weak, nonatomic) IBOutlet UILabel *lickedTimeLabel;
@end

@implementation RMLollipopLicker {
    NSOperationQueue *queue;
    NSMutableArray *lollipops;
}

#pragma mark - Life cycle

- (void)viewDidLoad {
    [super viewDidLoad];

    self.progressView.progress = 0.0;
    self.label.text = [NSString stringWithFormat:@"Tap on run and I'll lick a lollipop %d times!", COUNT];
    self.lickedTimeLabel.text = @"";

    lollipops = [[NSMutableArray alloc] init];
    queue = [[NSOperationQueue alloc] init];
}

- (void)lickLollipop {
    NSURL *fileURL = [[NSBundle mainBundle] URLForResource:@"Lollipop" withExtension:@"plist"];
    NSDictionary *dictionary = [NSDictionary dictionaryWithContentsOfURL:fileURL];
    NSString *lollipop = [dictionary objectForKey:@"Lollipop"];
    [lollipops addObject:lollipop];
}

#pragma mark - IBActions

- (IBAction)doneButtonPressed:(id)sender {
    [self dismissViewControllerAnimated:YES completion:nil];
}

- (IBAction)runButtonPressed:(id)sender {
    [sender setEnabled:NO];
    [queue addOperationWithBlock:^(
        for (NSInteger i = 0 ; i = COUNT) {
            self.label.text = [NSString stringWithFormat:@"Tap on run and I'll lick a lollipop %d times!", COUNT];
            self.progressView.progress = 0.0;
            [sender setEnabled:YES];
        }
    )];
}

@end
```

当用户点击运行按钮,应用开始一个背景线程,调用 lickLollipop 方法若干次,然后更新界面反映吃棒棒糖的数量。lickLollipop 方法从属性列表文件(PLIST)文件读取一个长字符串,然后添加到数组上。这些数据并不重要,能在不影响用户体验的前提下重新创建。

利用每种能够清除和重建数据而不影响用户体验的情况是好习惯。这样能够方便地释放内存,减少低内存警告。

那么,如何提高代码质量呢? 实现 didReceiveMemoryWarning 方法,像下面这样处理数据:

```
-(void)didReceiveMemoryWarning {
    [lollipops removeAllObjects];
    [super didReceiveMemoryWarning];
}
```

下一步?

万岁,你研究了4个闪退案例! 你的应用更完善了,并且学到了一些重要的调试技巧。

你可以到这里下载改进后的项目代码。

CocoaChina是全球最大的苹果开发中文社区，官方微信每日定时推送各种精彩的研发教程资源和工具，介绍app推广营销经验，最新企业招聘和外包信息，以及Cocos2d引擎、Cocos Studio开发工具包的最新动态及培训信息。关注微信可以第一时间了解最新产品和服务动态，微信在手，天下我有！

请搜索微信号“CocoaChina”关注我们！



(163)

被顶起来的评论



星空

Date/Time: 2016-03-17 07:53:35.35 +0800
Launch Time: 2016-03-17 07:53:33.33 +0800
OS Version: iOS 9.2.1 (13D15)
Report Version: 105

Exception Type: EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000
Exception Note: EXC_CORPSE_NOTIFY
Triggered by Thread: 0

Filtered syslog:
None found

Last Exception Backtrace:

(0x180b41900 0x1801aff80 0x180a2abcc 0x180a36674 0x100112ba8 0x100126448 0x10013aeb8 0x10013ac58 0x10013abb8 0x1001085800x100108318 0x1814f644c 0x180af95f4 0x180af9298 0x180af69ac 0x180a25680 0x181f34088 0x18589cd90 0x100134940 0x1805c68b8)

请问楼主 我这是什么问题啊 好像只有6plus / 6splus上才会出现 但是偶尔也不闪退

3月17日 回复 顶(2) 转发

12条评论

4条新浪微博

最新 最早 最热



redye

我查看崩溃日志时显示到的全是 类似于下面这种

5 xiaomu 0x000f376a 0xc5000 + 190314

6 xiaomu 0x00166dc2 0xc5000 + 662978

可以转换成下面这种吗？怎么转换？

8 Rage Masters 0x000ea1c4 -[RAppDelegate application:didFinishLaunchingWithOptions:] (RAppDelegate.m:36)

2015年8月13日 回复 顶 转发



海加尔的风

作者文章提到：从Xcode的Organizer窗口获取崩溃日志后过几秒钟，崩溃日志将被自动符号化。

2015年8月28日 回复 顶 转发



马爱林AIRIN

转

2015年9月9日 回复 顶 转发



创意应用

学习了~

2015年9月28日 回复 顶 转发



敢不敢再扯一点

学习啦，赞

2015年10月10日 回复 顶 转发

乐乐



Incident Identifier: 9E217001-D0BB-42E8-B9A7-1AEA4431804D
CrashReporter Key: a39505dc2699038665a91a1c02b448c38366e670
Hardware Model: iPod5,1
Process: Eye4 [5697]
Path: /private/var/mobile/Containers/Bundle/Application/597E9AA6-09E5-40C8-9B62-5FA2110591F4/Eye4.app/Eye4
Identifier: com.Eye4.ios
Version: 3.0.10 (3.0.6)
Code Type: ARM (Native)
Parent Process: launchd

Date/Time: 2016-01-12 16:12:14.14 +0800
Launch Time: 2016-01-12 16:10:56.56 +0800
OS Version: iOS 9.2 (13C75)
Report Version: 104

Exception Type: EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000
Exception Note: EXC_CORPSE_NOTIFY
Triggered by Thread: 0

Filtered syslog:
None found

Last Exception Backtrace:

```
0 CoreFoundation 0x261ec106 __exceptionPreprocess + 122
1 libobjc.A.dylib 0x25992e12 objc_exception_throw + 34
2 CoreFoundation 0x261ec04c +[NSException raise:format:] + 108
3 Foundation 0x26992d94 NSKVODeallocate + 404
4 libobjc.A.dylib 0x259ad3a4 (anonymous namespace)::AutoreleasePoolPage::pop(void*) + 384
5 CoreFoundation 0x26100954 _CFAutoreleasePoolPop + 12
6 UIKit 0x2a5dbab8 _prepareForCAFlush + 308
7 UIKit 0x2a31be94 _afterCACCommitHandler + 144
8 CoreFoundation 0x261aef54 __CFRunLoop_IS_CALLING_OUT_TO_AN_OBSERVER_CALLBACK_FUNCTION__ + 16
9 CoreFoundation 0x261ad258 __CFRunLoopDoObservers + 276
10 CoreFoundation 0x261ad68a __CFRunLoopRun + 954
11 CoreFoundation 0x26100bf4 CFRunLoopRunSpecific + 516
12 CoreFoundation 0x261009e0 CFRunLoopRunInMode + 104
13 GraphicsServices 0x2734cac4 GSEventRunModal + 156
14 UIKit 0x2a390b9c UIApplicationMain + 140
15 FFFF 0x000c154e main (main.mm:14)
16 libdyld.dylib 0x25daf86e tlv_get_addr + 42
```

大神，这要怎么弄啊，我还是没搞懂啊

1月12日 回复 顶 转发



年轻哥

1

1月14日 回复 顶 转发



我叫宋黎明

回复 redye: 可以。如果你還未找到辦法，可以回復我，我告訴你辦法。

1月16日 回复 顶 转发



我叫宋黎明

回复 乐乐: EXC_CORPSE_NOTIFY
從這個詞來看，似乎是註冊通知後，未及時removeObserver

1月16日 回复 顶 转发



星空

Date/Time: 2016-03-17 07:53:35.35 +0800
Launch Time: 2016-03-17 07:53:33.33 +0800
OS Version: iOS 9.2.1 (13D15)
Report Version: 105

Exception Type: EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000
Exception Note: EXC_CORPSE_NOTIFY
Triggered by Thread: 0

Filtered syslog:
None found

Last Exception Backtrace:

(0x180b41900 0x1801aff80 0x180a2abcc 0x180a36674 0x100112ba8 0x100126448 0x10013aeb8 0x10013ac58 0x10013abb8 0x1001085800x100108318 0x1814f644c 0x180af95f4 0x180af9298 0x180af69ac 0x180a25680 0x181f34088 0x18589cd90 0x100134940 0x1805c68b8)

请问楼主 我这是什么问题啊 好像只有6plus / 6splus上才会出现 但是偶尔也不闪退

3月17日 回复 顶(2) 转发



Letterman

这是我通过第三方库抓取的崩溃日志，楼主帮帮忙教下我怎么看

KSCrash CoreFoundation 2e924f83 868227

KSCrash libobjc.A.dylib 390d5ccf objc_exception_throw 15567

KSCrash CoreFoundation 2e928917 882967

KSCrash CoreFoundation 2e927203 877059

KSCrash CoreFoundation 2e876768 _CF_forwarding_prep_0 153448

KSCrash MapGooAssistant 1b2811 _mh_execute_header 919569

KSCrash MapGooAssistant 1af8a5 _mh_execute_header 907429

KSCrash MapGooAssistant 18cdc7 _mh_execute_header 765383

KSCrash MapGooAssistant 18cac5 _mh_execute_header 764613

KSCrash libdispatch.dylib 395bd833 6195

KSCrash libdispatch.dylib 395bd81f 6175

KSCrash libdispatch.dylib 395bd777 6007

KSCrash CoreFoundation 2e8ef8a1 649377

KSCrash CoreFoundation 2e8ee175 643445

KSCrash CoreFoundation 2e858ebf CFRunLoopRunSpecific 32447

KSCrash CoreFoundation 2e858ca3 CFRunLoopRunInMode 31907

KSCrash GraphicsServices 3375e663 GSEventRunModal 38499

KSCrash UIKit 311a514d UIApplicationMain 450893

KSCrash MapGooAssistant 166eed _mh_execute_header 610029

KSCrash libdyld.dylib 395e2ab7 6839

4月13日 回复 顶 转发



高宸锐

现在有crash日志,有dSYM文件,有.App文件...怎么把crash里的内存地址符号化成我们能看懂的字符串

5月3日 回复 顶 转发

社交帐号登录: 微信 微博 QQ 人人 更多»



说点什么吧...

发布

www.cocoachina.com正在使用多说



苹果开发中文站

网站地图

关于我们

联系我们

合作云平台: 又拍云

京公网安备 11010502011183

京ICP备 11006519号 京ICP证 100954号

Copyright © 2008-2013 CocoaChina.com

资讯频道

游戏开发

App Store研究

iOS开发

游戏开发

Cocos引擎

业界动态

产品设计

开发者论坛

论坛

技术问答

开发者中心

代码库

工具库

开发者平台

开发者平台

关注微信 每日推荐



关注我们

扫一扫 浏览移动版



程序人生 友情链接

Cocos引擎中文官网 cocos2d-x 摩点众筹 美图秀秀 iPhone 版 苹果开发者中心 手游那点事 雷锋网 工程师爸爸 iPad网址导航 麦芽地 Nooidea.com | 装傻充愣 啃苹果论坛 苹果fans 苹果发烧友 9RIA天地会 苹果发烧友 泰然网 eoe开发者社区 维以不永伤 远景苹果主题 游戏邦 爱应用 人人都是产品经理 9秒社团 源码天堂 游戏陀螺 推酷网 SegmentFault