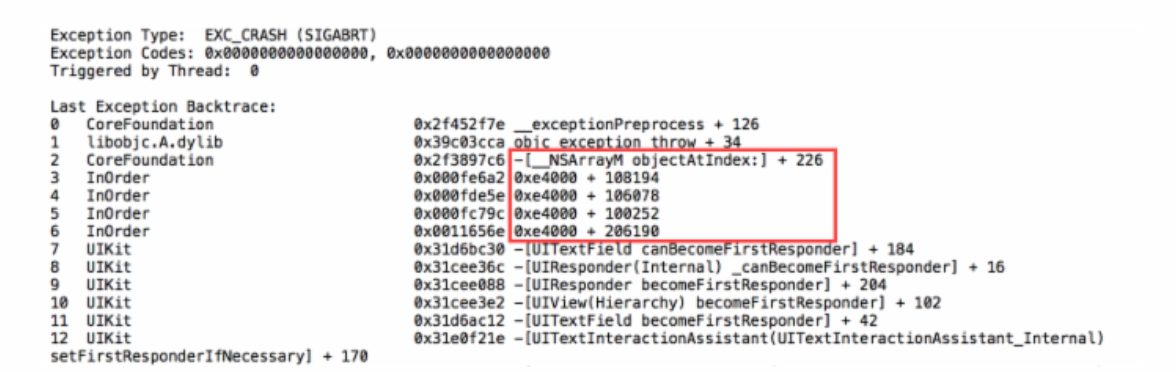
解析线上app崩溃报告

1.简述：想来每个iOS攻城狮，都免不了要接触.crash文件，那么什么是.crash文件?

iOS app的所有崩溃记录都会记录在设备上，所以对于和我一样没有集成让用户发送崩溃报告功能的iOS开发者来说，要获得crash文件就必须先连上崩溃过的机器,然后从崩溃过的机器上导出.crash文件。

2.如何解析.crash文件

我们先看一眼导出来的.crash文件，重点看下崩溃部分的记录，如下图：

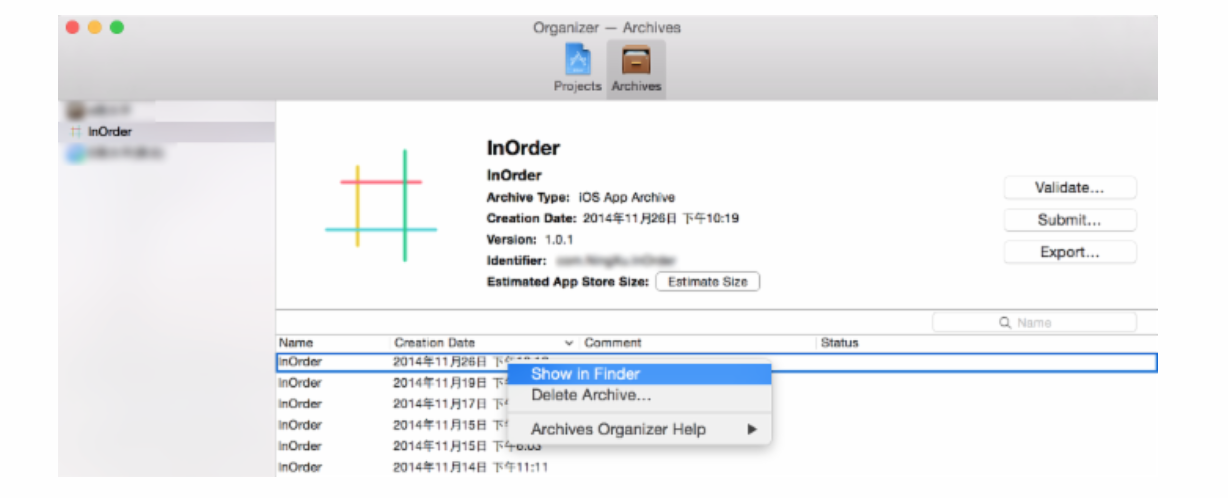


显然从这图里，我们没法定位出具体的错误，和崩溃的具体行数。

我们得到是个二进制的报告，这时候我们需要对它进行反编译。

对.crash文件进行反编译我们需要用到三个文件，缺一不可

2.1找到发布app时的.ipa文件(就是你打包app上传到商店里到那个文件)

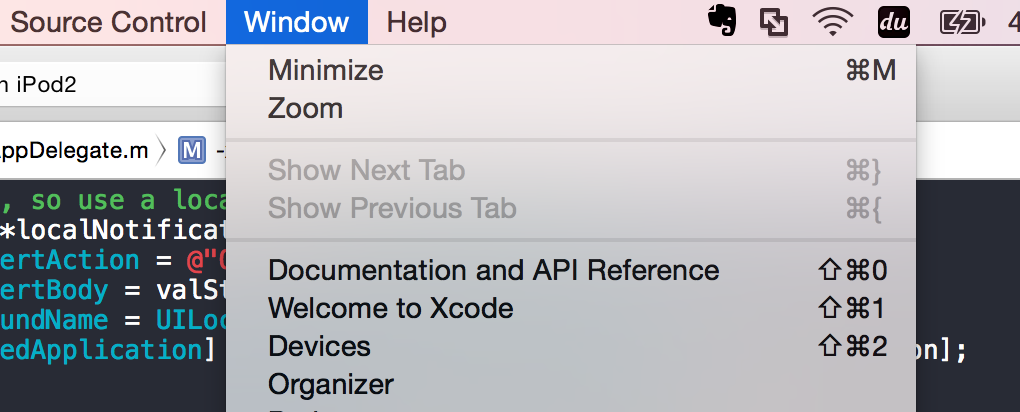
2.2找到Archive时生成的.dSYM文件

Show in Finder后对文件夹中的这个.xcarchive文件右键，显示包内容，就可以看到一个名为dSYMs的文件夹，把里面的.dSYM文件拷出来

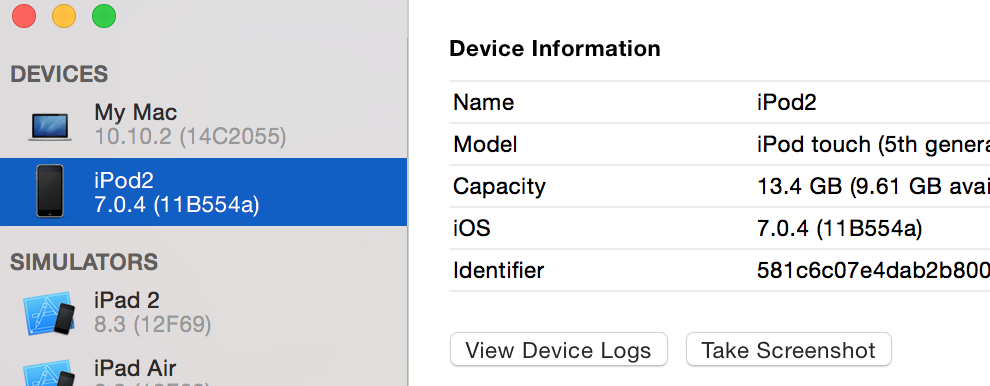
2.3将.ipa、.crash、.dSYMs文件放在同一个文件夹(例如：创建一个bugcrash文件夹)

2.4打开你的xcode，然后打开你发布时的代码(记住必须是发布时的代码，如果不是的话，会出现指向的崩溃行数有误，或者无法解析出来的问题)

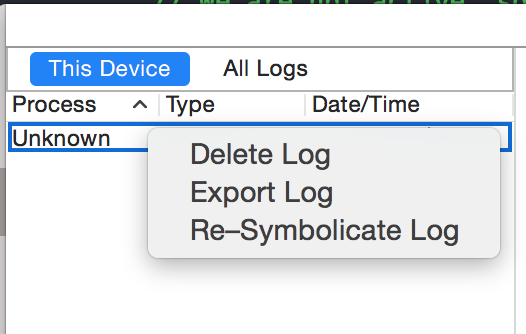
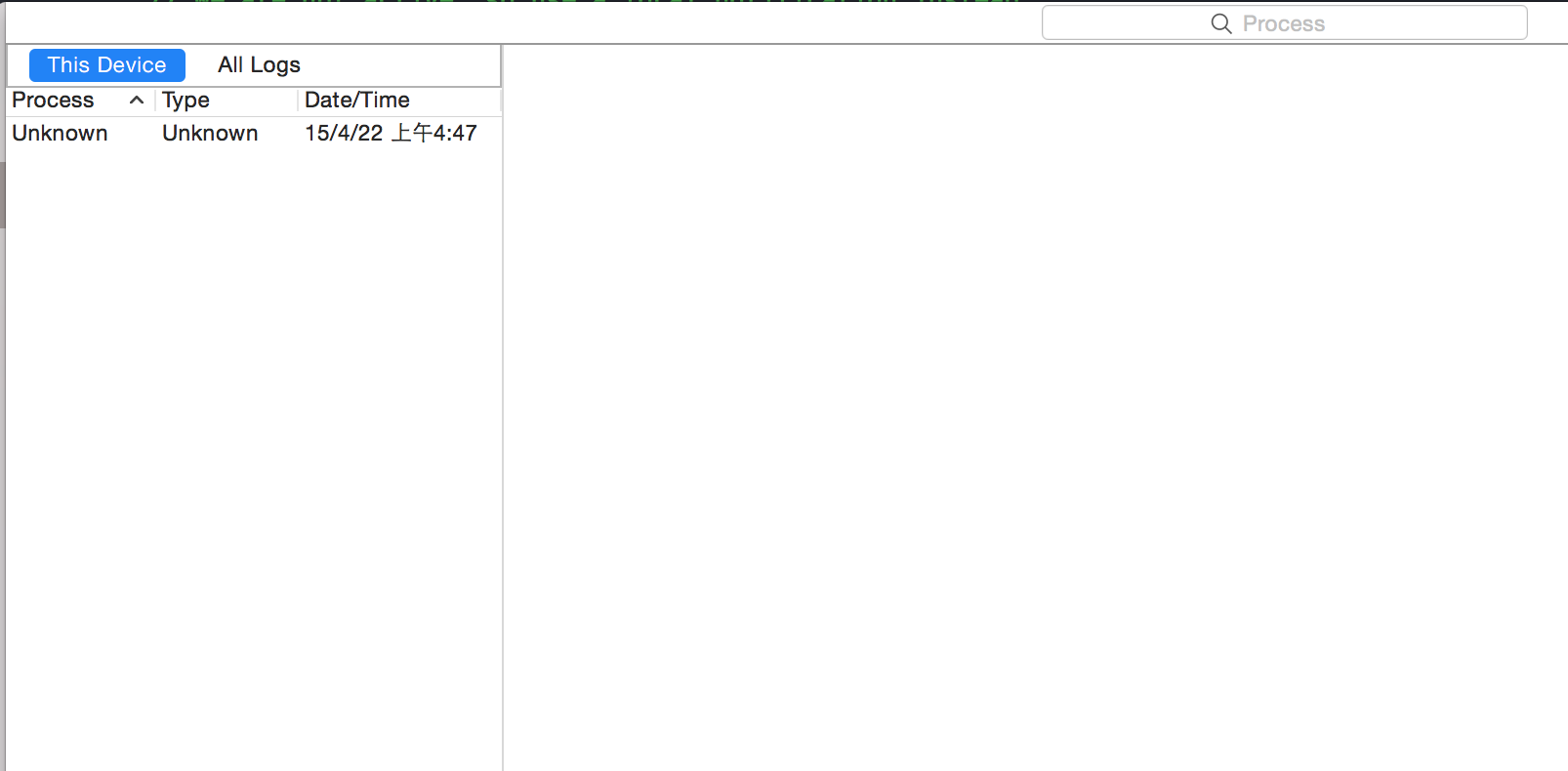
3.利用xcode自带re-symbolicate解析崩溃报告



点Devices进去

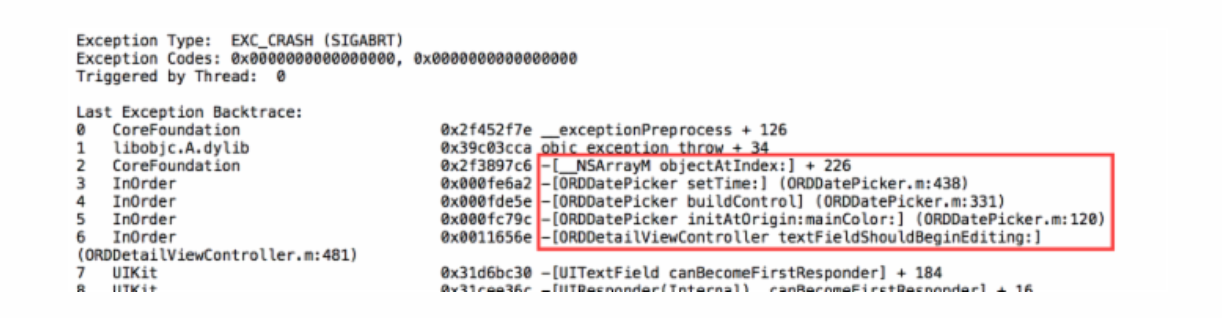


然后点击View Device Logs 进去

现在打开bugcrash文件夹，然后把.crash文件拖入右框中然后选中右键re-symbolicate log进行解析

大约1-10秒中之后，我们就能发现之前的二进制数变成了我们想要看到的具体行数和具体的.crash原因.



4.如果xocde自带的Re-symbolicate仍无法解析，那么我们需要利用symbolicatecrash借助命令行手动解析

symbolicatecrash是一个隐藏工具，它在我的Mac中的具体路径如下(Xcode6.1.app请换成你的Xcode名称)

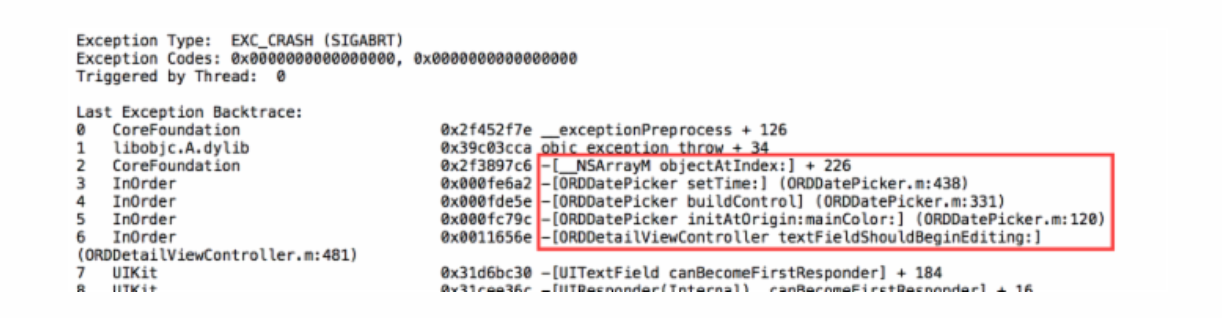
/Applications/Xcode6.1.app/Contents/SharedFrameworks/DTDeviceKitBase.framework/Versions/A/Resources/symbolicatecrash

把这个路径拷贝一下，然后粘到Finder的“前往文件夹”下，前往，就可以看到symbolicatecrash工具了，现在把它也拷到桌面的crash文件夹里。至此，crash文件夹里现在有4个文件了，分别是.app, .crash, .dSYM, symbolicatecrash。接下来就是用终端敲命令，生成更易分析的crash。

./symbolicatecrash /Users/xxxx/Desktop/bugcrash/InOrder.crash /Users/xxxx/Desktop/bugcrash/InOrder.app.dSYM > Control\_symbol.crash

上述命令中，"xxxx"和"InOrder"请自行替换成对应的名称。运行，这时候终端可能会报错Error: "DEVELOPER\_DIR" is not defined at /usr/local/bin/symbolicatecrash line 53. 这时候在终端中再输入如下（Xcode6.1.app依然是要替换成实际名称）

然后再跑一下刚刚的那个命令，这时候看一下桌面的crash文件夹下就会多出一个名为“Control\_symbol.crash”的文件，我们打开看一下。和刚刚一样。



5.介绍一下如何使用友盟分析工具解析友盟平台生成的csv崩溃日志文件

(1)下载umcrashtool文件和友盟错误日志.csv文件

(2)确保对应版本的dSYM文件位于以下路径

/Users/ios31/Library/Developer/Xcode/GuDaShi.app.dSYM

(3)将(1)中文件放到同一个文件夹

(4)打开命令行，进入(3)中目录，执行以下命令

./umcrashtool /Users/ios31/Desktop/error/err4.csv

(5)使用系统number软件打开(4)中生成的err4-symbol.csv文件，可以看到具体崩溃文件和行数