符号表工具iOS版-使用指南

Bugly文档资料

符号表工具iOS版-使用指南
1. 介绍
1.1 环境要求
1.2 符号表提取要求
1.3 配置文件
1.4 上传功能
2. 提取符号表文件的方法
2.1 工具使用方法
2.2 支持的选项
2.3 举一个例子
3. dSYM文件
3.1 什么是dSYM文件？
3.2 如何定位dSYM文件？
3.3 如何找回已发布到App Store的App对应的dSYM文件？
3.3.1 通过Xcode的Archive找回
3.3.2 通过mdfind工具找回
3.4 如果查看dSYM文件的UUID？
3.4.1 通过命令查看
3.4.1 通过符号表工具查看
3.5 XCode Debug编译后没有生成dSYM文件？

1. 介绍

为了能快速并准确地定位用户App发生**Crash的代码位置**，Bugly使用**符号表文件**对App发生Crash的程序**堆栈**进行**解析**和**还原**。

举一个实例：

还原前堆栈	还原后堆栈
0 Test 0x00b122e4 0x00004000 + 11592420	0 Test 0x00b122e4 -[FXLabel initWithFrame:] (FXLabel.m:71)
1 Test 0x0062f37c 0x00004000 + 6468476	1 Test 0x0062f37c -[BTBannerView resetUI] (BTBannerView.m:312)
2 Test 0x0062fe40 0x00004000 + 6471232	2 Test 0x0062fe40 -[BTNavView init] (BTNavView.m:76)

而符号表工具，正是Bugly提供给开发者提取符号表文件（.symbol）的工具。另外，Bugly提供了自动上传符号表文件的方法，请参考《Bugly iOS 符号表配置》，建议使用自动上传的方式。

1.1 环境要求

符号表工具的运行需要Java运行环境（Java SE Runtime Environment），JRE或JDK版本需要 >= 1.6。

1.2 符号表提取要求

提取符号表需要符号表工具和**dSYM文件**（具有调试信息的目标文件，可参考下文的第三部分：**3. dSYM文件**）。

1.3 配置文件

Bugly iOS符号表工具2.3.0及以上版本增加了配置文件的解析功能，工具包中提供了一个与工具JAR同目录的默认配置文件（settings.txt）。

可以通过配置文件设置以下信息：

- Debug：调试模式开关（打印更多Log）
- Upload：上传开关
- ID：Bugly平台的App ID
- Key：Bugly平台的App key

1.4 上传功能

Bugly iOS符号表工具支持上传功能，使用上传功能时，必须要指定以下信息：

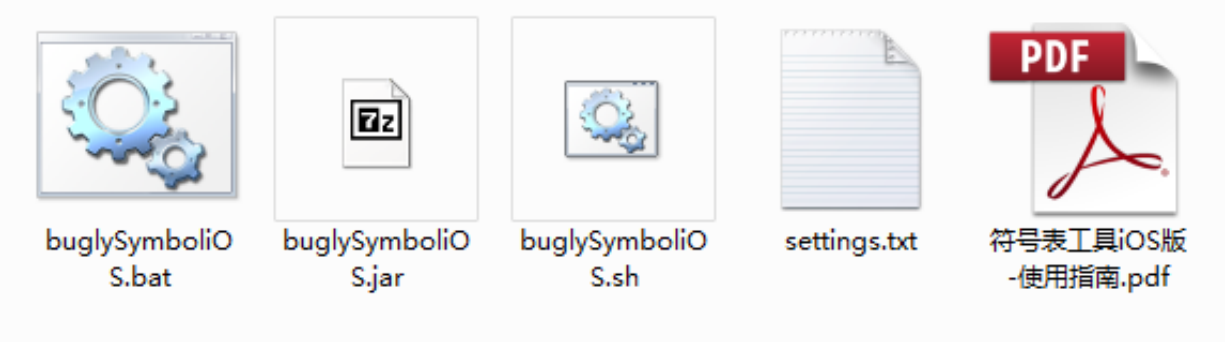
- App ID（可通过配置文件指定）
- App key（可通过配置文件指定）
- App版本
- App包名

目前脚本不支持以上信息的指定，因此需要通过直接执行JAR包来使用上传功能。

2. 提取符号表文件的方法

iOS版符号表工具支持Windows、Linux和Mac三个平台，同时提供了JAR包、各平台脚本和符号表配置文件：

- 符号表工具JAR包（buglySymboliOS.jar）
- Bat脚本（buglySymboliOS.bat）
- Shell脚本（buglySymboliOS.sh）
- 默认符号表配置文件（settings.txt）



使用脚本时，请保证脚本和jar包在同个目录下！

2.1 工具使用方法

```
java -jar <JAR包> <选项>

<Bat脚本> <选项>

<Shell脚本> <选项>
```

2.2 支持的选项

选项	说明
-i	指定文件路径，可指定目录（必选）
-o	输出的符号表zip文件的路径，必须是zip文件
-d	调试模式开关（默认关闭）
-s	指定配置文件（默认读取JAR目录下的“settings.txt”文件）
-u	上传开关
-id	App ID
-key	App key
-package	App包名
-version	App版本

2.3 举一个例子

注意事项

不要直接复制例子中的命令运行，需要根据自己的具体情况更改下命令。

环境和用户信息

- 系统：**Mac OS**
- 用户目录：**/home/batman**
- 符号表工具（已解压）所在目录：**/Users/batman/Downloads/buglySymboliOS**
- dSYM所在目录：**/Users/batman/Desktop/test.app.dSYM**
- APP ID：**900012345**
- APP key：**abcdefghijk**
- APP包名：**com.batman.demo**
- APP版本：**2.3.1**

生成符号表文件

使用符号表工具的JAR包生成符号表文件的命令如下：

```
cd /Users/batman/Downloads/buglySymboliOS

java -jar buglySymboliOS.jar -i /Users/batman/Desktop/test.app.dSYM
```

生成的符号表文与库文件（test.app.dSYM）位于同级目录：/Users/batman/Desktop/

生成符号表文件并自动上传

使用符号表工具的JAR包生成符号表文件，并自动上传的命令如下：

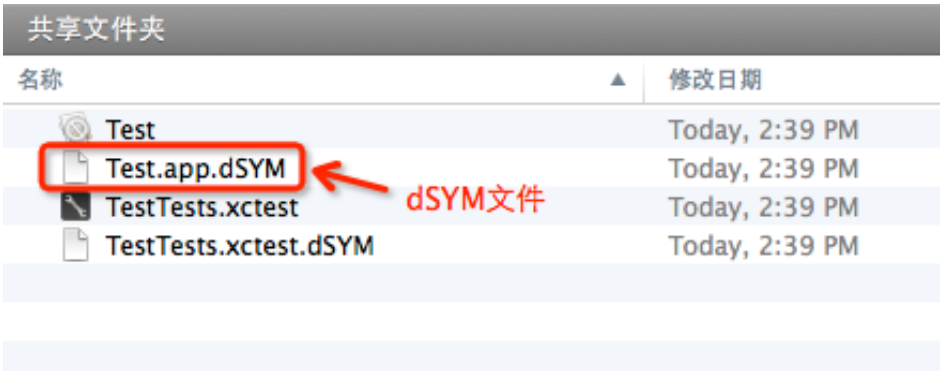
```
cd /Users/batman/Downloads/buglySymboliOS

java -jar buglySymboliOS.jar -i /Users/batman/Desktop/test.app.dSYM
-u -id 900012345 -key abcdefghijk -package com.batman.demo -version
2.3.1
```

3. dSYM文件

3.1 什么是dSYM文件？

iOS平台中，dSYM文件是指具有调试信息的目标文件，文件名通常为：**xxx.app.dSYM**。如下图所示：

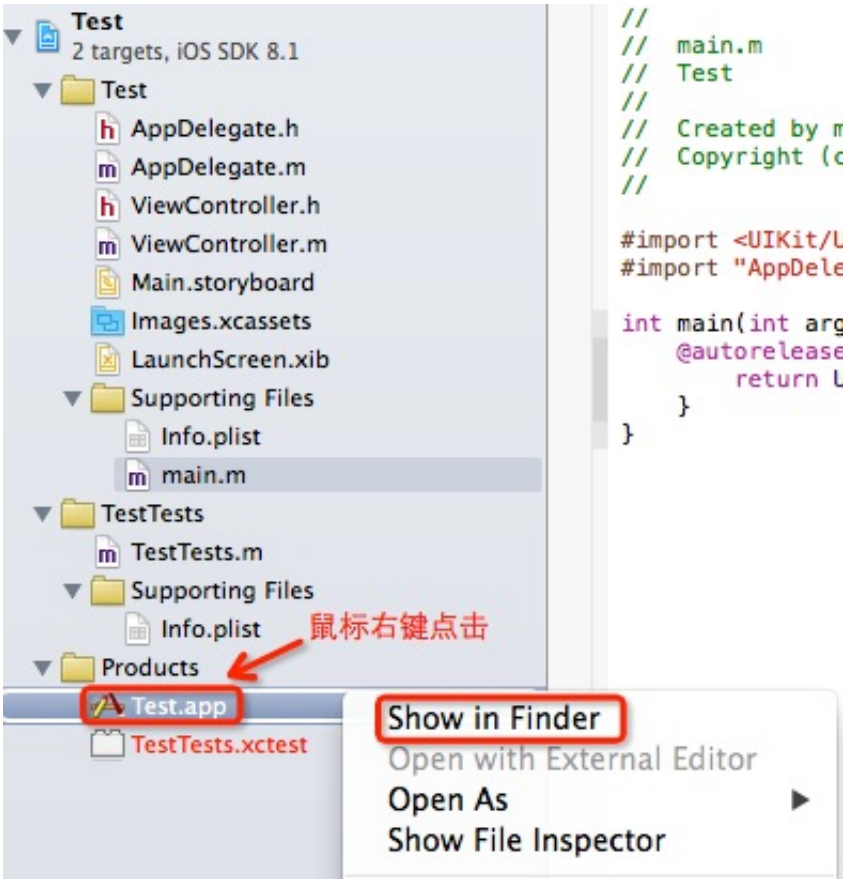


3.2 如何定位dSYM文件？

一般情况下，项目编译完dSYM文件跟app文件在同一个目录下，下面以XCode作为IDE详细说明定位dSYM文件。

```
-> 进入XCode；
-> 打开工程（已编译过）；
-> 在左栏找到“Product”项；
-> 鼠标右键点击编译生成的“xxx.app”；
-> 点击“Show in Finder”；
```

如下图所示：

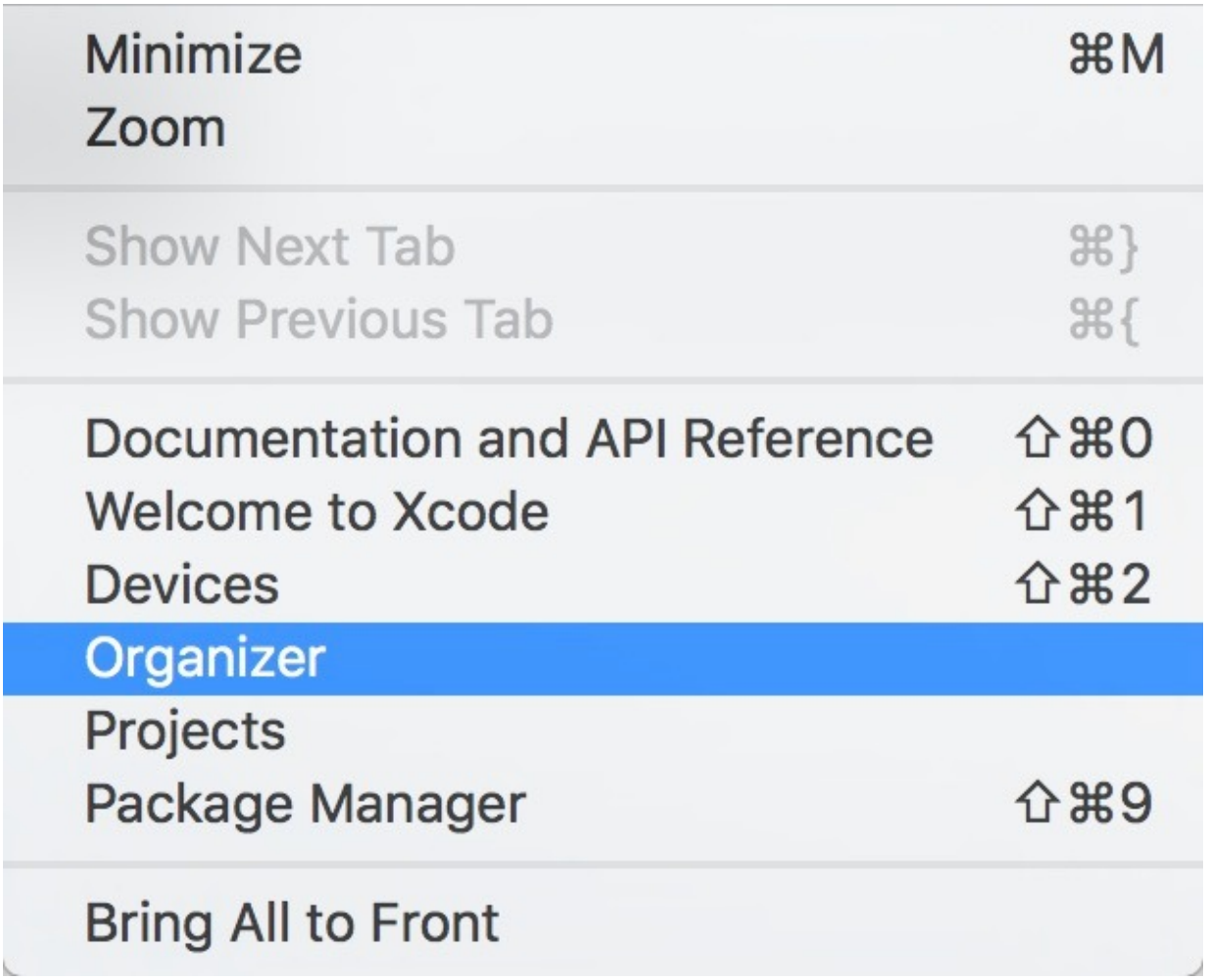


如果有多个**dSYM**文件，可以在使用工具时指定输入为**dSYM**文件所在的目录或者工程目录。

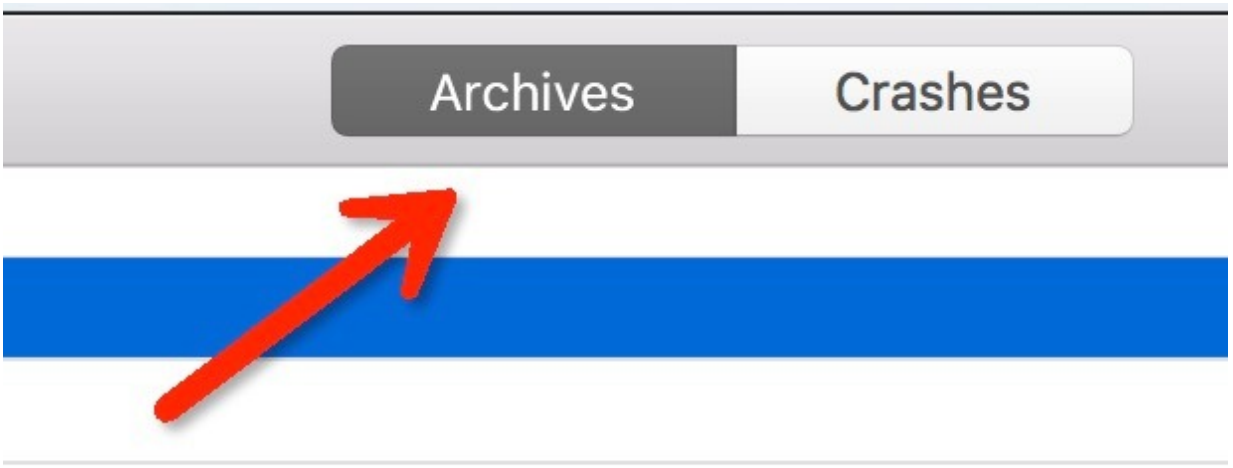
3.3 如何找回已发布到App Store的App对应的dSYM文件？

3.3.1 通过Xcode的Archive找回

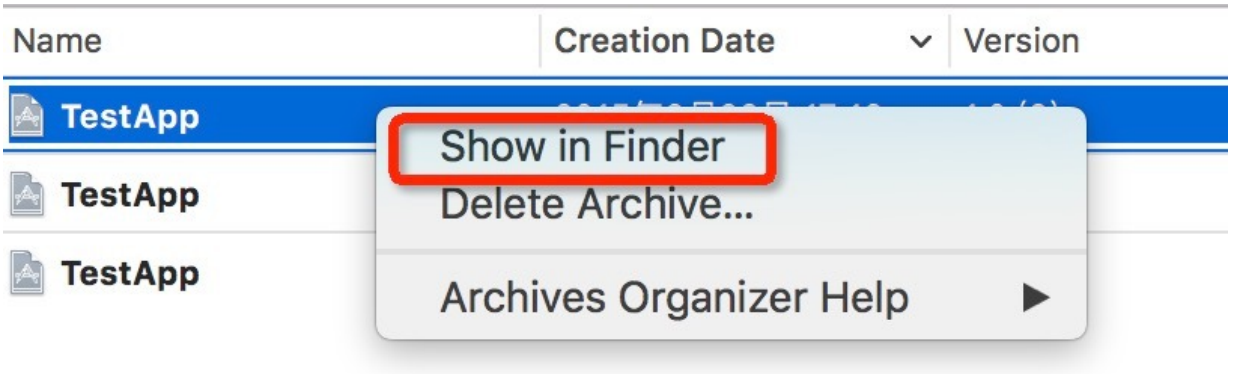
打开 Xcode 顶部菜单栏 -> Window -> Organizer 窗口：



打开 Xcode 顶部菜单栏，选择 Archive 标签：

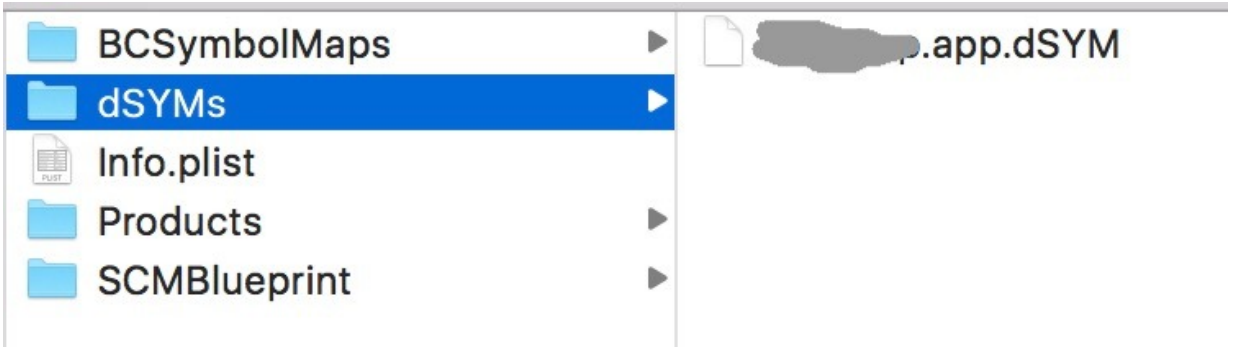


找到发布的归档包，右键点击对应归档包，选择Show in Finder操作：



右键选择定位到的归档文件，选择显示包内容操作：

选择dSYMs目录，目录内即为下载到的 dSYM 文件：



3.3.2 通过mdfind工具找回

在Bugly的issue页面查询到crash对应的UUID：

然后在Mac的Shell中，用mdfind命令定位dSYM文件：

```
mdfind "com_apple_xcode_dsym_uuids == <UUID>"
```

注意，使用mdfind时，UUID需要格式转换（增加“-”），举一个例子：

要定位的dSYM的UUID为：E30FC309DF7B3C9F8AC57F0F6047D65F
则定位dSYM文件的命令如下：

```
mdfind "com_apple_xcode_dsym_uuids == E30FC309-DF7B-3C9F-8AC5-7F0F6
047D65F"
                                    |12345678-1234-1234-1234-xxxxx
xxxxxxx|
```

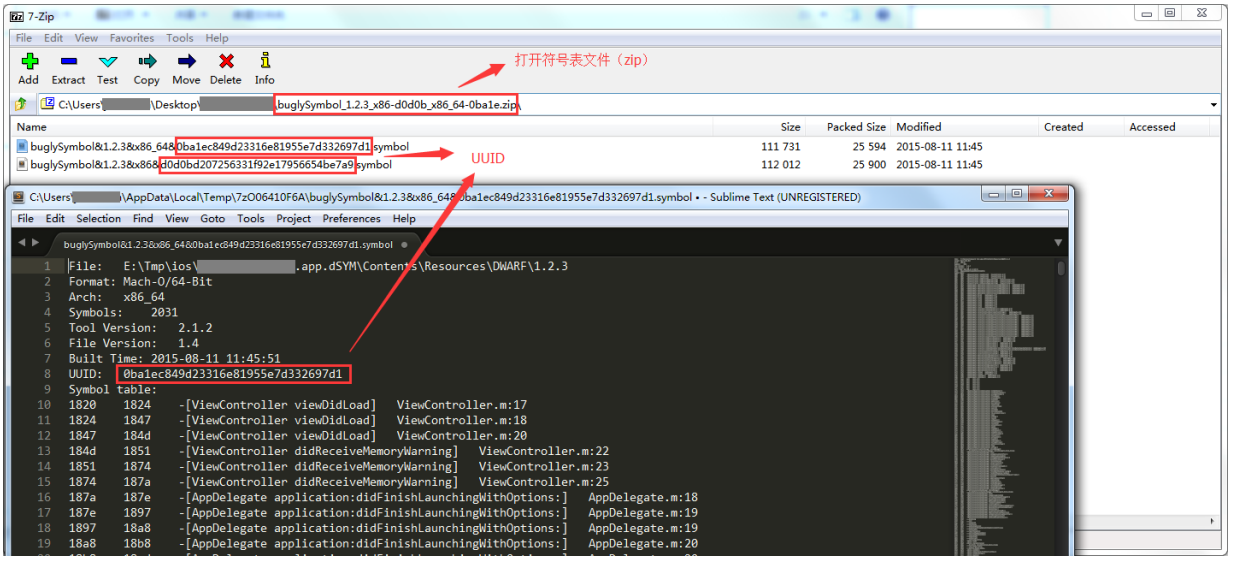
3.4 如果查看dSYM文件的UUID？

3.4.1 通过命令查看

```
xcrun dwarfdump --uuid <dSYM文件>
```

3.4.1 通过符号表工具查看

`-> 使用符号表工具生成dSYM文件对应的符号表文件`
`-> 解压符号表文件（*.zip）`
`-> 打开符号表文件（*.symbol）`
符号表文件中的“UUID”字段，就是dSYM文件的UUID。



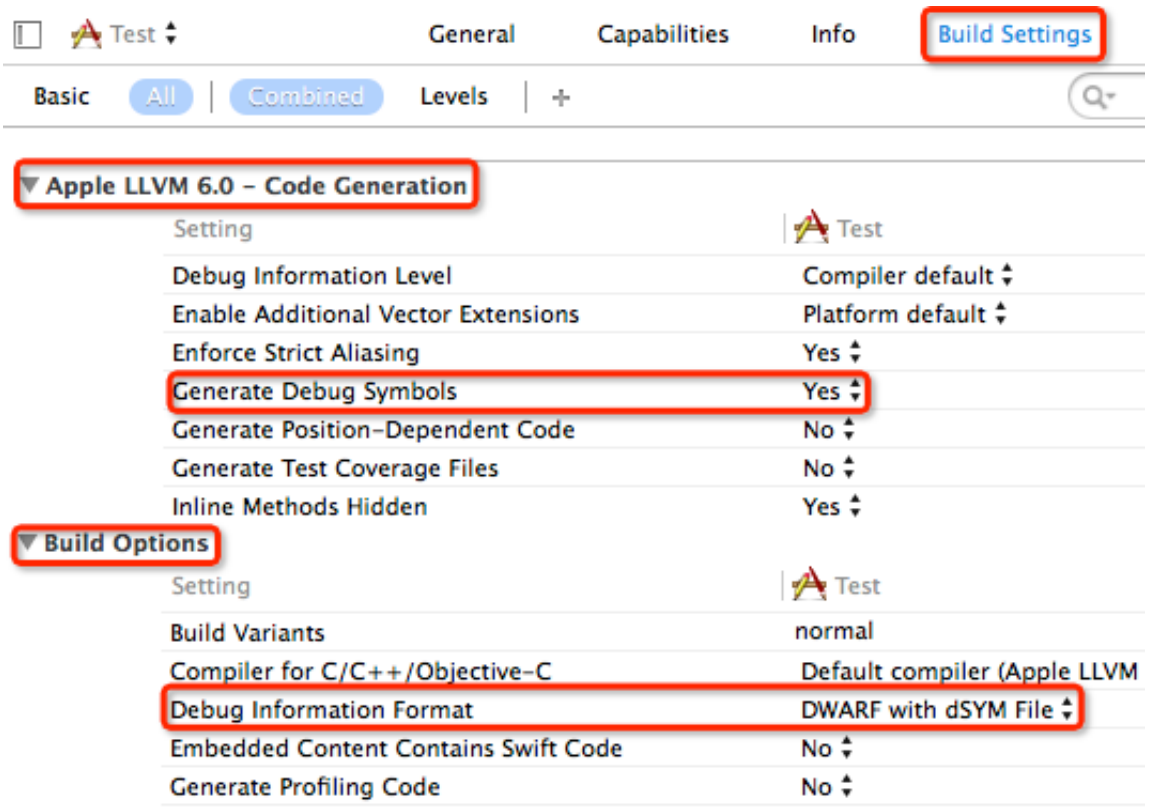
3.5 XCode Debug编译后没有生成dSYM文件？

XCode新建的工程默认会生成dSYM文件，生成dSYM文件的设置方法为：

`XCode -> Build Settings -> Code Generation -> Generate Debug Symbols -> Yes`

`XCode -> Build Settings -> Build Option -> Debug Information Format -> DWARF with dSYM File`

如下图所示：


Test
General
Capabilities
Info
Build Settings
Basic
All
Combined
Levels
Apple LLVM 6.0 – Code Generation
Setting
Test
Debug Information Level
Compiler default
Enable Additional Vector Extensions
Platform default
Enforce Strict Aliasing
Yes
Generate Debug Symbols
Yes
Generate Position-Dependent Code
No
Generate Test Coverage Files
No
Inline Methods Hidden
Yes
Build Options
Setting
Test
Build Variants
normal
Compiler for C/C++/Objective-C
Default compiler (Apple LLVM
Debug Information Format
DWARF with dSYM File
Embedded Content Contains Swift Code
No
Generate Profiling Code
No