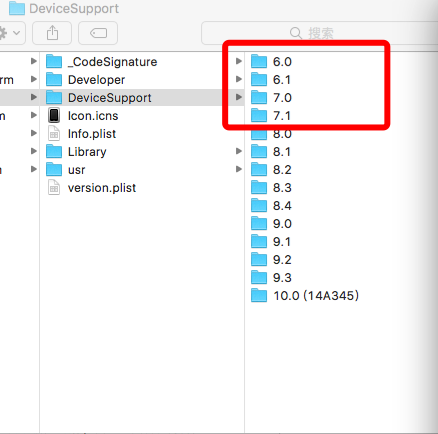
**[iOS开发]Xcode8兼容iOS7以及低版本Xcode调试高版本iOS系统**

原文：http://www.jianshu.com/p/d22c19812d43

我们使用Xcode8新建的工程，默认支持的最低系统是iOS8，我们可以手动更改版本到7.0，但是不支持真机调试。

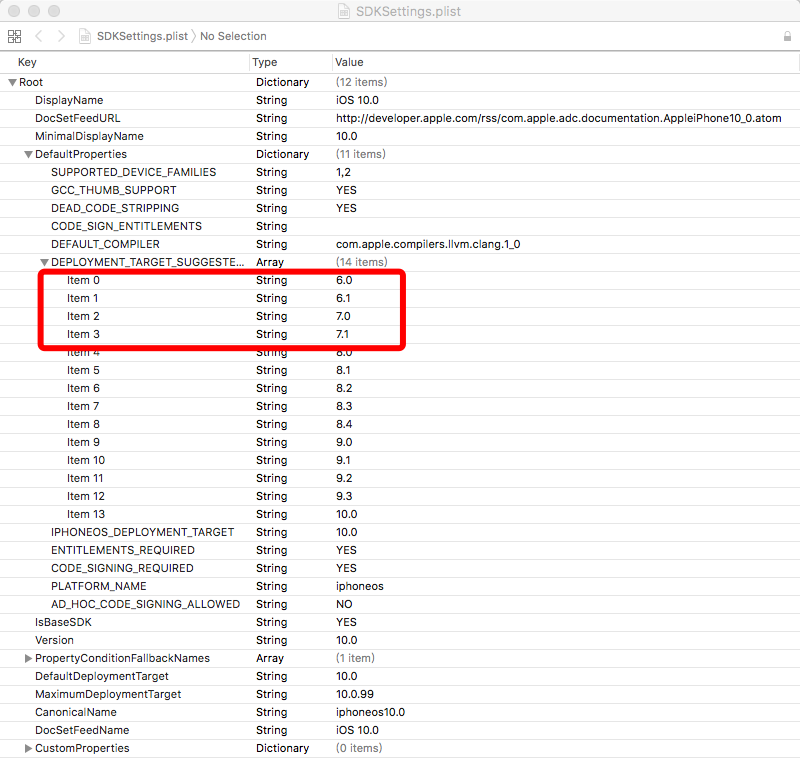
现在的项目一般都要兼容iOS7系统，同时也要兼容iOS10，在Xcode8上面，默认情况下无法调试iOS7，因为缺乏调试iOS7需要的配置文件。同时在低版本的Xcode上面（8以下），也无法调试iOS10的真机。解决办法如下：

我们在升级Xcode8之前，可以先将调试需要的配置文件拷贝出来，方法finder中前往文件夹/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/DeviceSupport进入，将里面如下图所示的文件夹拷贝出来，如果已经升级了Xcode8，里面就没有这些配置，我这里有拷贝好的，[点击下载](https://github.com/duoduoxiao/iOS7SDK.git" \t "_blank)。



504E327F-3013-400F-B20C-9D2C0F924887.png

升级到Xcode8之后，将之前拷贝出来的文件放入之前的文件夹中，在finder中前往文件夹到/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/DeviceSupport，之后我们需要配置一下Xcode，同样的在finder中前往文件夹，打开以下路径/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS.sdk ，用Xcode打开SDKSettings.plist这个文件，加入如下图所示的配置，保存之后重启Xcode8，之后在工程的Deployment Target里面就可以选择7.0了。



D93BBB36-5041-43A6-BA3A-7F5144111DE0.png

如果是Xcode8以下的版本调试适配iOS10，方法是一样的，只不过需要在高版本的Xcode里面把配置文件拷贝出来，上面的下载地址里面也有iOS10的配置包，目前可以调试iOS10.0.2的手机系统。

如果SDKSettings.plist这个文件提示无法修改的话，可以先讲这个文件拷贝一份到桌面，修改后再覆盖进去即可。

**iOS开发～制作同时支持armv7,armv7s,arm64,i386,x86\_64的静态库.a**

* 原文：http://blog.csdn.net/lizhongfu2013/article/details/42387311

一、概要

平时项目开发中，可能使用第三方提供的静态库.a，如果.a提供方技术不成熟，使用的时候就会出现问题，例如：

在真机上编译报错：No architectures to compile for (ONLY\_ACTIVE\_ARCH=YES, active arch=x86\_64, VALID\_ARCHS=i386).

在模拟器上编译报错：No architectures to compile for (ONLY\_ACTIVE\_ARCH=YES, active arch=armv7s, VALID\_ARCHS=armv7 armv6).

要解决以上问题，就要了解一下Apple移动设备处理器指令集相关的一些细节知识。

二、几个重要概念

1、ARM

ARM处理器，特点是体积小、低功耗、低成本、高性能，所以几乎所有手机处理器都基于ARM，在嵌入式系统中应用广泛。

2、ARM处理器指令集

armv6｜armv7｜armv7s｜arm64都是ARM处理器的指令集，这些指令集都是向下兼容的，例如armv7指令集兼容armv6，只是使用armv6的时候无法发挥出其性能，无法使用armv7的新特性，从而会导致程序执行效率没那么高。

还有两个我们也很熟悉的指令集：i386｜x86\_64 是Mac处理器的指令集，i386是针对intel通用微处理器32架构的。x86\_64是针对x86架构的64位处理器。所以当使用iOS模拟器的时候会遇到i386｜x86\_64，iOS模拟器没有arm指令集。

3、目前iOS移动设备指令集

arm64：iPhone5S｜ iPad Air｜ iPad mini2(iPad mini with Retina Display)

armv7s：iPhone5｜iPhone5C｜iPad4(iPad with Retina Display)

armv7：iPhone3GS｜iPhone4｜iPhone4S｜iPad｜iPad2｜iPad3(The New iPad)｜iPad mini｜iPod Touch 3G｜iPod Touch4

armv6 设备： iPhone, iPhone2, iPhone3G, 第一代、第二代 iPod Touch（一般不需要去支持）

4、Xcode中指令集相关选项（Build Setting中）

（1）Architectures

Space-separated list of identifiers. Specifies the architectures (ABIs, processor models) to which the binary is targeted. When this build setting specifies more than one architecture, the generated binary may contain object code for each of the specified architectures.

指定工程被编译成可支持哪些指令集类型，而支持的指令集越多，就会编译出包含多个指令集代码的数据包，对应生成二进制包就越大，也就是ipa包会变大。

（2）Valid Architectures

Space-separated list of identifiers. Specifies the architectures for which the binary may be built. During the build, this list is intersected with the value of ARCHS build setting; the resulting list specifies the architectures the binary can run on. If the resulting architecture list is empty, the target generates no binary.

限制可能被支持的指令集的范围，也就是Xcode编译出来的二进制包类型最终从这些类型产生，而编译出哪种指令集的包，将由Architectures与Valid Architectures（因此这个不能为空）的交集来确定，例如：比如，你的Valid Architectures设置的支持arm指令集版本有：armv7/armv7s/arm64，对应的Architectures设置的支持arm指令集版本有：armv7s，这时Xcode只会生成一个armv7s指令集的二进制包。

再比如：将Architectures支持arm指令集设置为：armv7,armv7s，对应的Valid Architectures的支持的指令集设置为：armv7s,arm64，那么此时，XCode生成二进制包所支持的指令集只有armv7s

在Xcode6.1.1里的 Valid Architectures  设置里， 默认为 Standard architectures(armv7,arm64),如果你想改的话，自己在other中更改。原因解释如下： 使用 standard architectures (including 64-bit)(armv7,arm64) 参数，则打的包里面有32位、64位两份代码，在iPhone5s（ iPhone5s的cpu是64位的 ）下，会首选运行64位代码包， 其余的iPhone（ 其余iPhone都是32位的,iPhone5c也是32位 ），只能运行32位包，但是包含两种架构的代码包，只有运行在ios6，ios7系统上。 这也就是说，这种打包方式，对手机几乎没要求，但是对系统有要求，即ios6以上。 而使用 standard architectures (armv7,armv7s) 参数， 则打的包里只有32位代码， iPhone5s的cpu是64位，但是可以兼容32位代码，即可以运行32位代码。但是这会降低iPhone5s的性能。 其余的iPhone对32位代码包更没问题， 而32位代码包，对系统也几乎也没什么限制。 所以总结如下：  要发挥iPhone5s的64位性能，就要包含64位包，那么系统最低要求为ios6。 如果要兼容ios5以及更低的系统，只能打32位的包，系统都能通用，但是会丧失iPhone5s的性能。

（3）Build Active Architecture Only

指定是否只对当前连接设备所支持的指令集编译

当其值设置为YES，这个属性设置为yes，是为了debug的时候编译速度更快，它只编译当前的architecture版本，而设置为no时，会编译所有的版本。 编译出的版本是向下兼容的，连接的设备的指令集匹配是由高到低（arm64 > armv7s > armv7）依次匹配的。比如你设置此值为yes，用iphone4编译出来的是armv7版本的，iphone5也可以运行，但是armv6的设备就不能运行。  所以，一般debug的时候可以选择设置为yes，release的时候要改为no，以适应不同设备。

1）

Architectures:  armv7, armv7s, arm64ValidArchitectures:  armv6, armv7s, arm64生成二进制包支持的指令集： arm64

2）

Architectures: armv6, armv7, armv7sValid Architectures:  armv6, armv7s, arm64生成二进制包支持的指令集： armv7s

3）

Architectures: armv7, armv7s, arm64Valid Architectures: armv7，armv7s

这种情况是报错的，因为允许使用指令集中没有arm64。

注：如果你对ipa安装包大小有要求，可以减少安装包的指令集的数量，这样就可以尽可能的减少包的大小。当然这样做会使部分设备出现性能损失，当然在普通应用中这点体现几乎感觉不到，至少不会威胁到用户体检。

三、制作静态库.a是指令集选择

现在回归到正题，如何制作一个“没有问题”的.a静态库，通过以上信息了解到，当我们做App的时候，为了追求高效率，并且减小包的大小，Build Active Architecture Only设置成YES，Architectures按Xcode默认配置就可以，因为arm64向前兼容。但制作.a静态库就不同了，因为要保证兼容性，包括不同iOS设备以及模拟器运行不出错，所以结合当前行业情况，要做到最大的兼容性。

ValidArchitectures设置为：armv7｜armv7s｜arm64｜i386｜x86\_64

Architectures设置不变（或根据你需要）:  armv7｜arm64

然后分别选择iOS设备和模拟器进行编译，最后找到相关的.a进行合包，使用lipo -create 真机库.a的路径 模拟器库.a的的路径 －output 合成库的名字.a（详情可以参考http://blog.csdn.net/lizhongfu2013/article/details/12648633）

这样就制作了一个通用的静态库.a