[iOS静态库的编译](http://blog.csdn.net/duxinfeng2010/article/details/8809932)

### [IOS 入门开发之使用Xcode4制作静态库详解（五）](http://blog.csdn.net/xys289187120/article/details/7283763)

分类： [【IOS 入门开发系列锦集】](http://blog.csdn.net/xys289187120/article/category/897790)2012-02-22 17:13 9015人阅读 [评论](http://blog.csdn.net/xys289187120/article/details/7283763#comments)(12) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/xys289187120/article/details/7283763#report)

[xcode4](http://blog.csdn.net/tag/details.html?tag=xcode4" \t "_blank)[ios](http://blog.csdn.net/tag/details.html?tag=ios" \t "_blank)[uiview](http://blog.csdn.net/tag/details.html?tag=uiview" \t "_blank)[ios5](http://blog.csdn.net/tag/details.html?tag=ios5" \t "_blank)[image](http://blog.csdn.net/tag/details.html?tag=image" \t "_blank)[iphone](http://blog.csdn.net/tag/details.html?tag=iphone" \t "_blank)

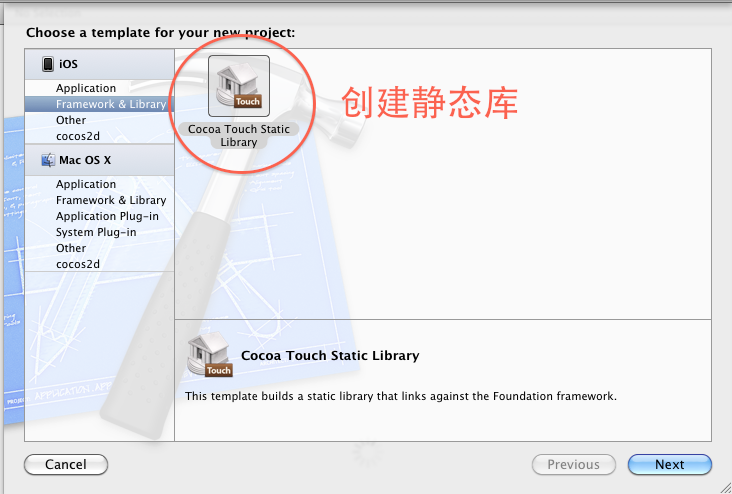
**IOS 入门开发之使用XCODE4制作静态库详解**

**雨松MOMO原创文章如转载，请注明：转载至我的独立域名博客**[**雨松MOMO程序研究院**](http://www.xuanyusong.com/)**，原文地址:**[**http://www.xuanyusong.com/archives/606**](http://www.xuanyusong.com/archives/606)

**注意：首先确保你的XCODE4使用IOS5，否则请下载IOS5在继续阅读。**

**IOS程序是由Objective-c语言构成，而是Objective-c语言中每一个类又分成 .h .m 文件。静态库可以将这些程序的类封装成一个.a文件，第三方应用程序只需要拿到这个.a文件和代码对应的.h文件即可使用静态库中封装的方法。总的来说IOS静态库适合去制作第三方提供的SDK，废话不多说了我们直接进正题。**

**首先打开Xcode4创建一个静态库工程，如下图选择Framework&Library ->Cocoa Touch Static Library （创建静态库），点击右下角按钮“next”继续完成创建。**

****

**工程创建完毕后，我们开始编写代码。我们创建一个UIView类用来显示一张图片，并且点击该视图后打开一个网址。**

**创建视图类的.h文件**

**[cpp]** [view plaincopy](http://blog.csdn.net/xys289187120/article/details/7283763)

1. #import <UIKit/UIKit.h>
3. @interface MyView : UIView
5. @end

创建视图类的.m文件

**显示的图片资源为通过URL下载后谷歌的LOGO，点击该LOGO视图后程序将打开百度的主页。**

**[cpp]** [view plaincopy](http://blog.csdn.net/xys289187120/article/details/7283763)

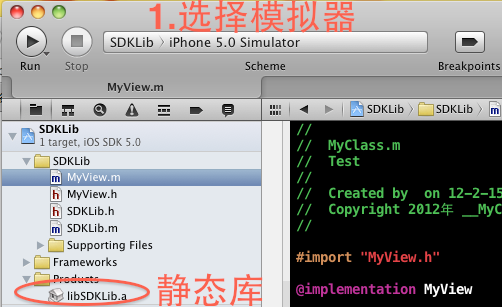
1. #import "MyView.h"
3. @implementation MyView
5. - (id)initWithFrame:(CGRect)frame
6. {
7. //初始化视图位置
8. self = [super initWithFrame:frame];
9. **if** (self) {
11. }
12. **return** self;
13. }

16. - (**void**)drawRect:(CGRect)rect
17. {
18. //这里创建一个图片视图
19. UIImage \*image=[[UIImage alloc]initWithData:[NSData dataWithContentsOfURL:[NSURL URLWithString:@"http://www.google.com.hk/intl/zh-CN/images/logo\_cn.png"]]];
20. [image drawInRect:CGRectMake(0, 0,self.frame.size.width , self.frame.size.height)];
21. [image release];
22. }

25. -(**void**)touchesEnded:(NSSet \*)touches withEvent:(UIEvent \*)event
26. {
27. //点击视图后打开网页
28. [[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"http://www.baidu.com"]];
29. }
31. @end

**到这一步静态库中的代码我们已经写完，现在我们开始制作静态库。静态库的制作方法可分为两种：第一种为在真机上使用的静态库，第二种为在模拟器中使用的静态库。这两种方法制作起来有点小区别，请大家一定看好了，别眨眼睛喔，**我们开始从模拟器入手。

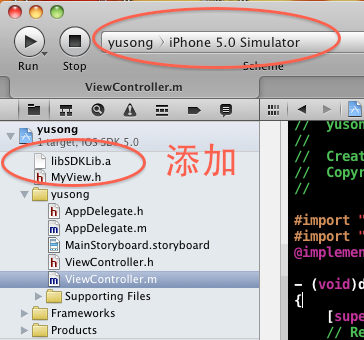
如图，先选择模拟器iPhone 5.0 Simulator，如果你的XCODE4不是IOS5版本那么你将没有这个选项，然后Build工程，构建工程完毕后libSDKLib.a静态库文件遍生成了出来， 选择该文件鼠标右键找到它。



**找到文件以后，默认为Debug-iphoneos / libsdklib.a ，但是这个文件是不能在模拟器中使用的，并且它也不能在真机中使用。你需要选择下面Debug-iphoneosimulator / libsdklib.a这个文件，将libSDKLib.a与对应MyView.h头文件拖拽入使用它的工程当中。**

http://hi.csdn.net/attachment/201202/22/0_1329900029J9jm.gif

**接下来创建一个普通的IOS工程，先选择模拟器运行 iPhone5.0 Simulator ，然后将libSDKLib.a 与MyView.h拖拽添加至程序当中即可。**

****

**运行工程即可看到效果，本例中通过URL加载了一张谷歌的LOGO，触摸点击该视图后打开百度的首页。**

****

**调用静态库的方法如下：**

**[cpp]** [view plaincopy](http://blog.csdn.net/xys289187120/article/details/7283763)

1. #import "ViewController.h"
2. #import "MyView.h"
3. @implementation ViewController
5. - (**void**)didReceiveMemoryWarning
6. {
7. [super didReceiveMemoryWarning];
8. // Release any cached data, images, etc that aren't in use.
9. }
11. #pragma mark - View lifecycle
13. - (**void**)viewDidLoad
14. {
15. [super viewDidLoad];
16. //创建静态库视图
17. MyView \*myView = [[MyView alloc] initWithFrame: CGRectMake(0, 0, 120, 100)];
18. //将静态库视图添加至窗口当中
19. [self.view addSubview:myView];
20. [myView release];
21. }
23. - (**void**)viewDidUnload
24. {
25. [super viewDidUnload];
26. // Release any retained subviews of the main view.
27. // e.g. self.myOutlet = nil;
28. }

31. @end

**OK ，到这一步我们已经将静态库顺利的应用在模拟器当中，下面我们学习如何将静态库应用在真机当中。我们再次回到制作静态库的工程当中，如图打开工程后编译环境选择IOS Device，然后构建构成即可完成。**

****

**使用鼠标右键找到libSDKlib.a，这时候Debug-iphoneos / libSDKlib.a 就是可以在真机上编辑的静态库。**

**http://hi.csdn.net/attachment/201202/22/0_1329900029J9jm.gif**

**最后将libSDKLib.a 与MyView.h拖拽添加至程序当中，连接上真机运行工程即可在真机上看到效果了。值得注意的是模拟器和真机用的libSDKLib.a必需要严格的区分开，否则程序会报错。**

**最后欢迎各位盆友可以和MOMO一起讨论IOS软件开发，从今天开始继续维护博客，你懂的，哇咔咔～～如果你觉得看得不清楚，MOMO附带上本章的源码下载，希望大家可以一起学习 哈哈～。哇咔咔～ MOMO愿和 大家好好学习，大家一起进步哈～！！！**

**下载地址：**[**http://www.xuanyusong.com/archives/606**](http://www.xuanyusong.com/archives/606)

### iOS静态库的编译和lipo命令的使用

2013-01-07 20:37:08|  分类： [iphone\_dev\_note](http://liwpk.blog.163.com/blog/#m=0&t=1&c=fks_084065082094083067081081085095087080082065082087081067)|字号 订阅

静态库的编译：  
选iOS Device,编译出来的.a静态库文件是基于arm架构上的。即可在真机上运行。  
选iPhone Simulator,编译出来的静态库文件是基于i386架构的。可在模拟器中运行。  
  
我们可以通过lipo命令将这两个文件打包成一个通用的.a文件。

==========================================================================================================================================

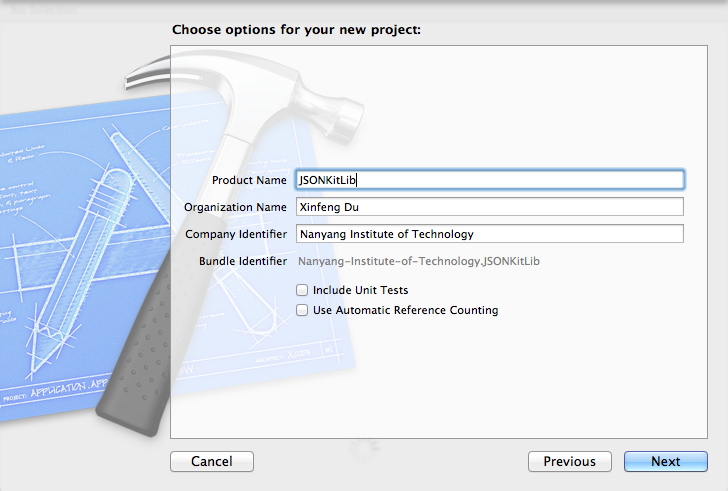
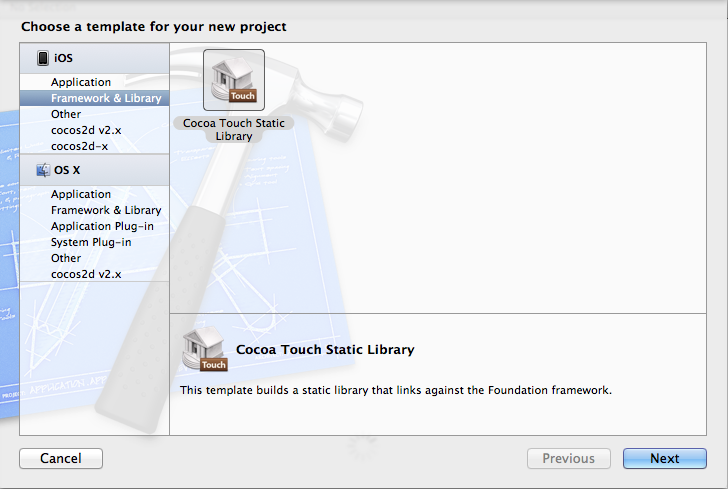
分类： [菜鸟学iOS的笔记](http://blog.csdn.net/duxinfeng2010/article/category/1155790)2013-04-16 18:28 361人阅读 [评论](http://blog.csdn.net/duxinfeng2010/article/details/8809932#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/duxinfeng2010/article/details/8809932#report)

[JSONKit](http://blog.csdn.net/tag/details.html?tag=JSONKit" \t "_blank)[静态库](http://blog.csdn.net/tag/details.html?tag=%e9%9d%99%e6%80%81%e5%ba%93" \t "_blank)

在做第三方的SDK的时候常会把自己的类编译成静态库，在提供给开发者使用的是好就能隐藏自己的私密信息，如私有API、APIKey等；

此处将开源的JSONKit类库编译成静态库，然后再使用我们编译好的静态库来来解析一个json文件

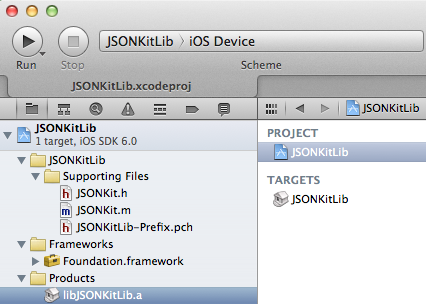
1.新建静态库工程，选择Framework & Library --》next ,命名JSONKitLib



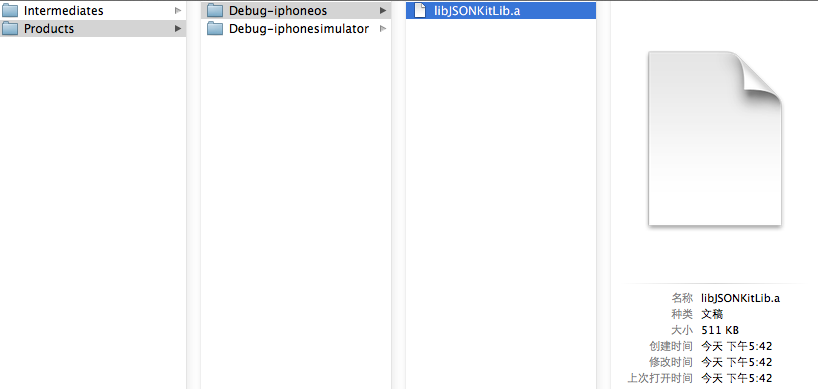
2.删除自动生成的JSONKitLib.h和JSONKitLib.h(因为是打包别人的类，不需要这两个文件，如果是自己制作就没必要删除了)，在<https://github.com/johnezang/JSONKit>下载JSONKit，将JSONKit.h和JSONKit.m拷贝到这个工程之中；

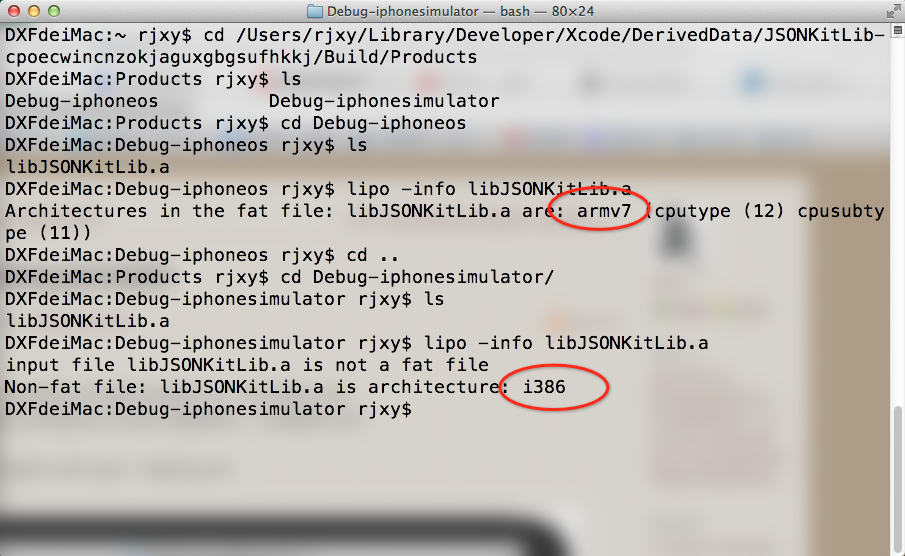
3.此处要特别注意的，因为iOS模拟器是基于Mac OS平台X86架构，iOS设备都是给予ARM架构，所以给他们提供的静态库也必须对于他们的架构，幸运的是Xcode可以同时编译这两个架构的库；

编译的时候调到iOS Device ,  Product  --> Build;  如果调到其他则不能编译出来静态库



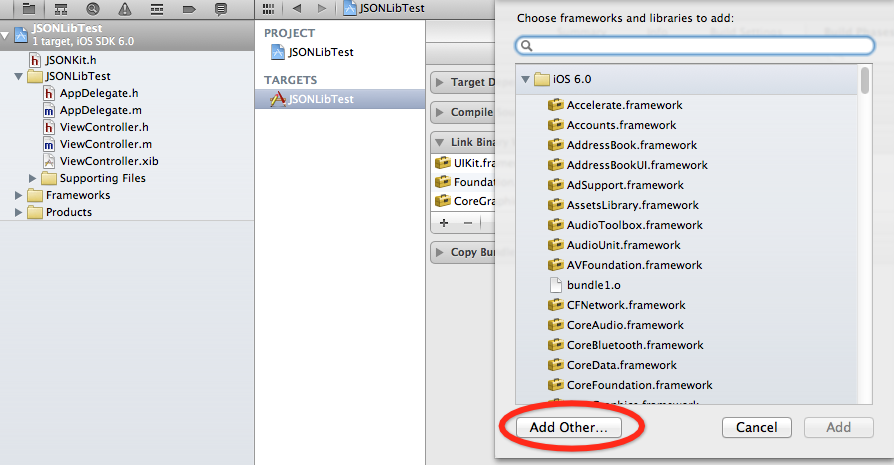
4.编译成功后就会出来libJSONKitLib.a,右键选中，show  in finder  ;会有Debug-iphoneos和Debug-iphonesimulator两个文件夹下的静态库，先让我们看看用lipo查看一下文件类型





一个是ARM架构的  一个是x86平台的i386架构的；如果我们要在模拟器上运行测试需要拷贝Debug-iphonesimulator下的静态库，如果需要在运行真机或是真机测试就必须是Debug-iphoneos下的静态库；

5.现在来测试刚刚编译的静态库，最好将这两库都拷贝出来，方便添加使用；新建一个xcode工程，把JSONKit.h文件添加到新建工程中，然后在把Debug-iphonesimulator下编译的静态库添加进去

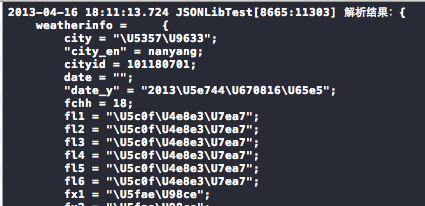


6.此处测试代码测试使用的是国家气象局api,返回的是json文件

**[cpp]** [view plaincopy](http://blog.csdn.net/duxinfeng2010/article/details/8809932)

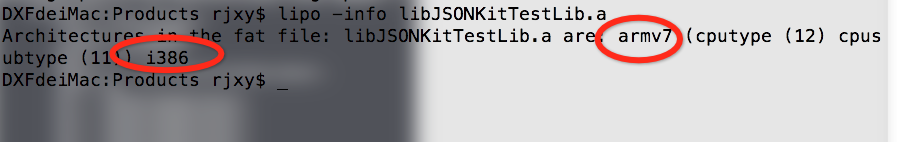
1. - (IBAction)TapJsonPaser:(id)sender {
2. NSError \*error;
3. NSURLRequest \*request = [NSURLRequest requestWithURL:[NSURL URLWithString:@"http://m.weather.com.cn/data/101180701.html"]];
4. NSData \*repsponse = [NSURLConnection sendSynchronousRequest:request returningResponse:nil error:&error];
5. JSONDecoder \*jsonPaser = [[JSONDecoder alloc] init];
7. NSDictionary \*weatherDic = [jsonPaser objectWithData:repsponse error:&error];
8. NSLog(@"解析结果：%@",weatherDic);
9. [jsonPaser release];
10. }

结果



但是有人会觉得这个太麻烦，果断的将这两个静态库打包成一个静态库使用命令

lipo -create Debug-iphoneos/libJSONKitLib.a Debug-iphonesimulator/libJSONKitLib.a -output libJSONKitTestLib.a



方法好虽好，但会牺牲空间，文件会变大，JSONKit源文件12KB + 177KB，编译成静态库后arm架构的511KB  i386架构的263KB,把它们打包成一个静态库是774KB;

# iOS静态库中打包图片资源

# 

# 问题描述:

最近在做一个项目，需要做一个含有图片资源得静态库.a，但是苹果提供得静态库工程无法将图片资源导入，只能通过外加.bundle文件的方式，这样在调试阶段也需要使用bundle文件，而每次更新bundle文件的内容时都需要重新编译bundle工程，生成新的bundle文件。

# 解决方案：

在图片资源不多的时候，其实可以将图片转换成base64的string，通过static方式存储。需要用的时候再通过base64解析成data，最后完成显示。

**[plain]** view plaincopyprint?

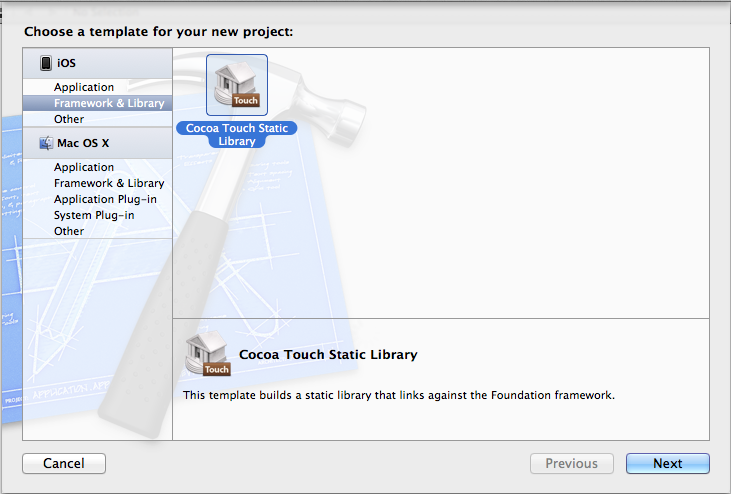
1. NSString \*stringInBase64 = @"";
2. NSData \*imageData = [Base64 decode: stringInBase64];
3. imageView.image = [UIImageimageWithData:imageData];

## ios 静态库制作过程

最近在做Apple的IOS开发，有开发静态库的需求，本身IOS的开发，只允许静态库或者Framework。在Xcode上没有找到允许编译，如同Android上的\*.so和Win32上的dll这样的说法。不过Framework这样的框架，估计也是类似动态库的实现，不过没有具体研究过，后续继续深入研究。

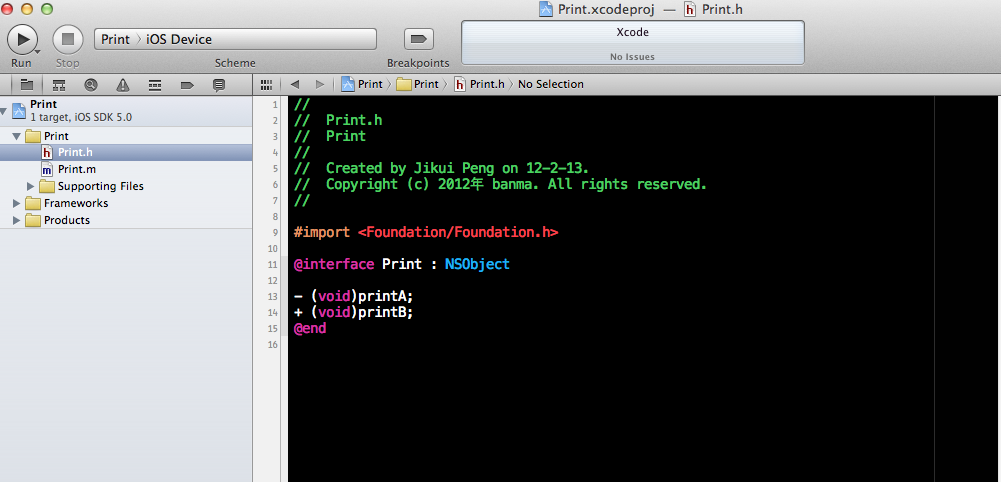
我这个文档的静态库的开发是基于Xcode4.2和iOS SDK5.0编写的。Xcode4跟之前的Xcode3还是有不少的差别的。  
  
下面就简单写一个静态库和一个调用静态库的例子。  
  
静态库的编写：  
  
1.静态库工程的建立：

Xcode New一个新的project，选择IOS下面的Framework&Library，下面有一个Cocoa Touch Static Library。直接next去建立一个Print这样的工程。



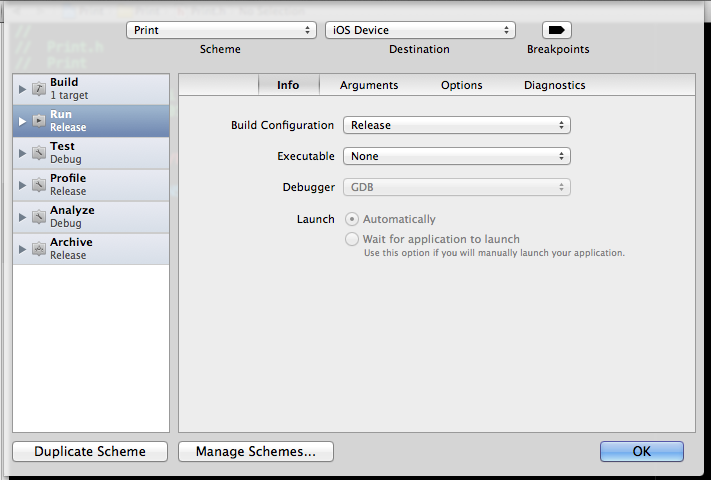
2.工程建立

我们添加很简单的"- (void)printA;"和"+ (void)printB;"的方法，这个方法的实现也是很简单的，在\*.m文件中的实现就是一个"NSLog（@“”);"这样我们编译出来的libstaticlib就可以被其他的IOS工程进行调用了。这块注意一下，目前我们的Print->iOS Device



3.编译前准备

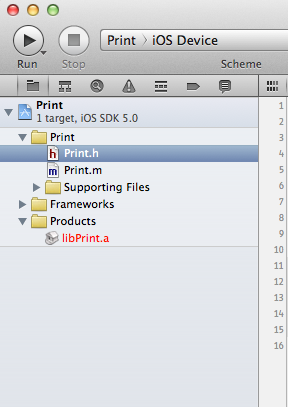
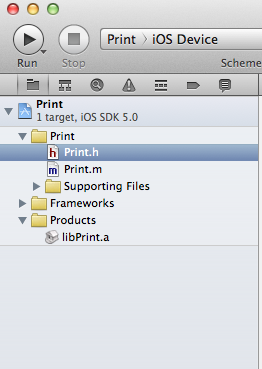
这里分release与debug版本，需要设置一下



4.OK，编译

我们可以在Products下面找到我们编译生成的libstaticlib.a这个文件。

注意未编译前为红色：左                             编译后为黑色： 右

5.ok

在右边 这个文件上右键Open In Finder。这时候我们打开了一个文件夹，里面有一个libPrint.a。后面我们继续编写一个调用这个libPrint.a的App。但是这个App在编译时候会出错，通过这个错误的解决，说明Xcode的一些编译相关的东西。这里，我们在command下面根据路径找到这个文件，用这个非常重要的命令查看一下这个lib的信息！

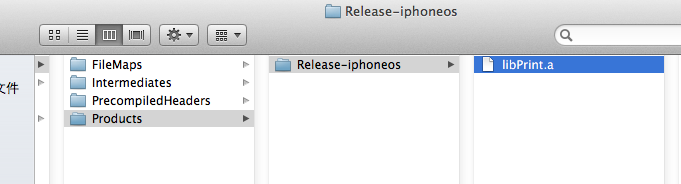
打开终端写入如下命令：这是什么？这就是所在的路径

以下均是在自己的mac上的路径。你需要将/Users/pjk1129/Library/Developer/Xcode/DerivedData/Print-dgfkluumuexoxhcapzidtsmdgqcj/Build/Products/ 替换成你的路径。

cd /Users/pjk1129/Library/Developer/Xcode/DerivedData/Print-dgfkluumuexoxhcapzidtsmdgqcj/Build/Products/

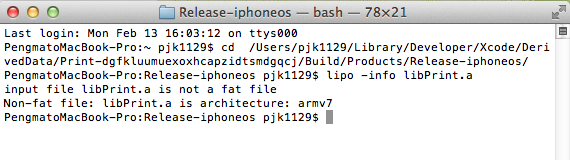
我们手动打开这个路径：可以看到如下信息：Release-iphoneos

这个是给真机用的，那我们要给模拟器和真机封装的静态库应该都能用才好。这个下面介绍！！！！！



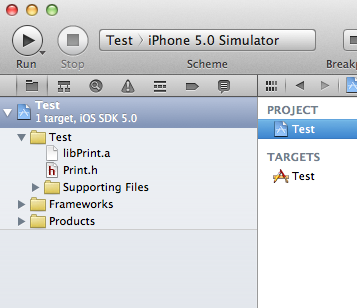
然后再在终端 打开这个文件：cd /Users/user/Library/Developer/Xcode/DerivedData/UItab-dbvoszbzpavyglaboxgflvrnzfce/Build/Products/Debug-iphoneos

命令是：lipo -info libPrint.a，可以看到显示的结果是：Arch是  Arm7。下面我会具体介绍一下。



6.建立一个调用这个libPrint.a的应用

应用就很简单了，我们添加刚才libPrint.a的\*.h文件。 然后在这个工程的Build Phases的Link Binary With Library里面添加我们刚才拷贝出来的那个libPrint.a。



7.插入了h和a文件后

我们就可以在control里面调用这个libPrint.a 函数了。

#import "AstroDiskAppDelegate.h"

#import "Print.h"

@implementation AstroDiskAppDelegate

@synthesize window = \_window;

- (void)dealloc

{

    [\_window release];

    [\_viewController release];

    [super dealloc];

}

- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

{

    //调用libPrint.a

    [Print printB];

    Print  \*print = [[[Print alloc] init] autorelease];

    [print printA];

    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]] autorelease];

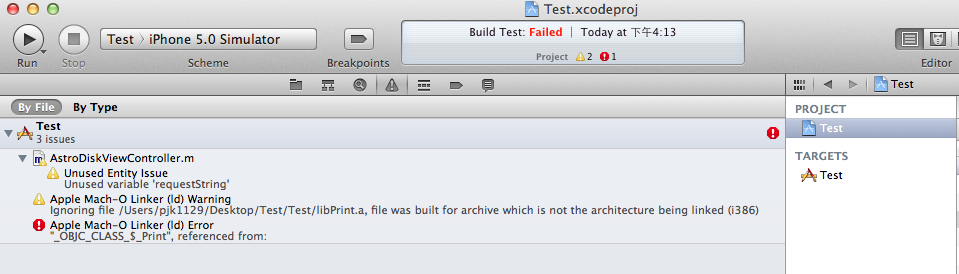
    [self.window makeKeyAndVisible];

    return YES;

}

8.OK，编译运行这个应用程序。

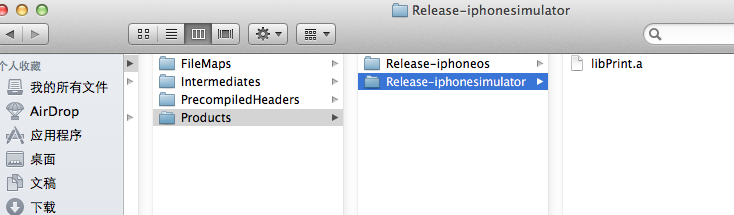
编译出现错误了。这个就是这篇文档的关键了。主要就是为了介绍编译静态库相关的东西。



9.可以看到undefine symbols for architecture i386这样的错误。

其实这个错误原因很简单，就是因为，我们用错了编译出来的libUITab.a lib，在模拟器里面，我们需要的是基于i386构架编译的static lib，但是这个a文件，大家还记得前面说的arm6 arm7构架的么。这个a其实是在iphone这个arm构架上运行的代码。其实show in finder指向的是一个ios device的库，并不是i386模拟器下的库。那如何编译i386的库呢？看见第二张图片的Print>IOS Device了么？我们将这个iOS Device修改成iPhone5.0 Simulator。在进行编译，这样就可以编译出i386下面的库。这个时候我们show in finder打开的文件夹还是ios device下的库。我们最好自己去那个目录下看一下。这里我们用command去查看了一下目录情况，如下：可以看到一个iphoneos的目录和一个simulator的目录。

然后我们再看看这个目录与第4条所示的目录有什么不同！ 同样也有一个libPrint.a文件。



10.错误原因分析

Release-iphoneos里面的是基于arm6 arm7编译出来的库文件。Release-iphonesimulator文件夹下面的是基于i386编译出来的文件。

11.合并.a文件，制作通用静态库

这二个库一个是用于真机运行的一个是用于模拟器运行的.其实我们可以利用lipo将这二个文件打包成一个通用的a文件。命令如下：

将/Users/user/Library/Developer/Xcode/DerivedData/UITab-dgfkluumuexoxhcapzidtsmdgqcj/Build/Products替换成你的路径。

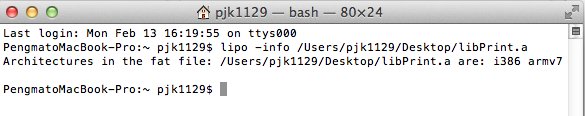
将/Users/user/Desktop/libPrint.a替换成合并后的路径。

lipo -create /Users/pjk1129/Library/Developer/Xcode/DerivedData/Print-dgfkluumuexoxhcapzidtsmdgqcj/Build/Products/Release-iphonesimulator/libPrint.a  /Users/pjk1129/Library/Developer/Xcode/DerivedData/Print-dgfkluumuexoxhcapzidtsmdgqcj/Build/Products/Release-iphoneos/libPrint.a  -output /Users/pjk1129/Desktop/libPrint.a

12.打包以后的库文件，我们查看信息

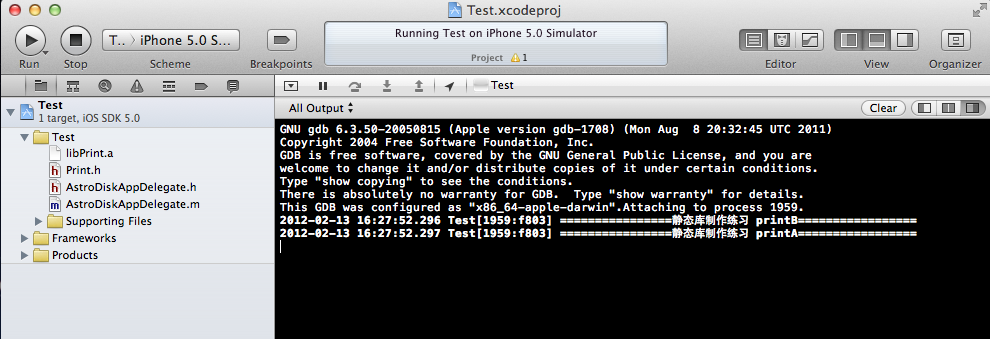
然后打入如下命令：lipo -info /Users/pjk1129/Desktop/libPrint.a

可以看到如下信息：

  
如上，你已经看到了，它具备了i386和arm7的条件。 现在我们这个静态库，支持的构架已经是armv7 i386了。

13.ok，经过上述步骤后

这个通用库生成后，我们将原来，调用这个库进行替换一下，在进行编译，成功编译，运行结果如下所示：



========================================================================静态库的使用