iOS打包静态库（完整篇）

1、 什么是库?

所谓库就是程序代码的集合，是共享程序代码的一种方式。

2、 库的分类

根据程序代码的开源情况，库可以分为两类

- 开源库
  源代码是公开的，你可以看到具体实现。比如GitHub上比较出名的第三方框架AFNetworking、SDWebImage。
- 闭源库
  不公开源代码，只公开调用的接口，看不到具体的实现，是一个编译后的二进制文件。这种常见于一些公司的SDK包，比如高德地图SDK、环信即时通讯SDK等等。而闭源库又分为两类：静态库和动态库。本篇重点要讲的便是其中的静态库。

3、静态库和动态库的存在形式和使用区别

存在形式：

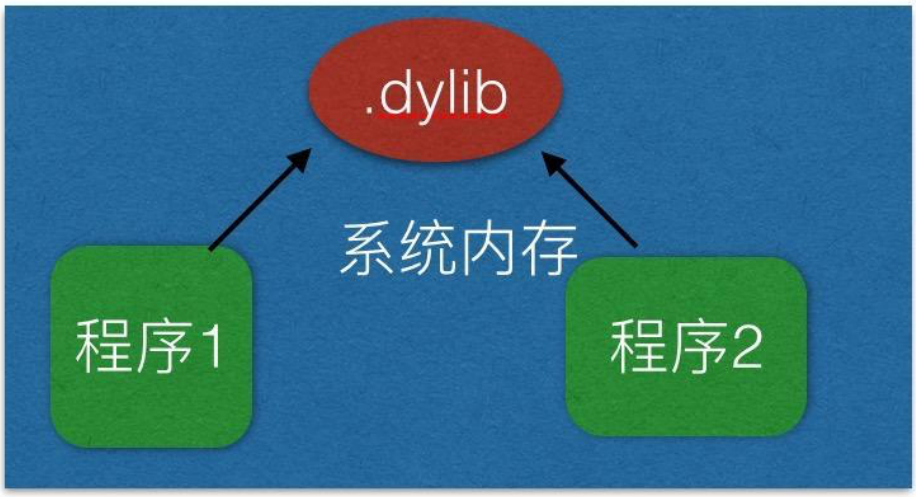
- 静态库
  以".a"或者“.framework”为文件后缀名
- 动态库
  以".dylib"或者“.framework”为文件后缀名（Xcode7 之后 .tbd 代替了 .dylib）

使用区别：

- 静态库链接时会被完整的复制到可执行文件中，被多次使用就有多份拷贝。

静态库被程序使用时

- 动态库链接时不复制，程序运行时由系统动态加载到内存，供程序调用。而且系统只加载一次，多个程序共用，节省内存。



动态库被程序使用时

4、iOS 设备的CPU架构

模拟器:
4s-5: i386
5s-iPhone X（包活iPhone SE）: x86_64

真机(iOS设备):
armv6: iPhone、iPhone 2、iPhone 3G、iPod Touch(第一代)、iPod Touch(第二代)
armv7: iPhone 3Gs、iPhone 4、iPhone 4s、iPad、iPad 2
armv7s: iPhone 5、iPhone 5c (静态库只要支持了armv7,就可以在armv7s的架构上运行)
arm64: iPhone 5s、iPhone 6、iPhone 6 Plus、iPhone 6s、iPhone 6s Plus、iPad Air、iPad Air2、iPad mini2、iPad mini3

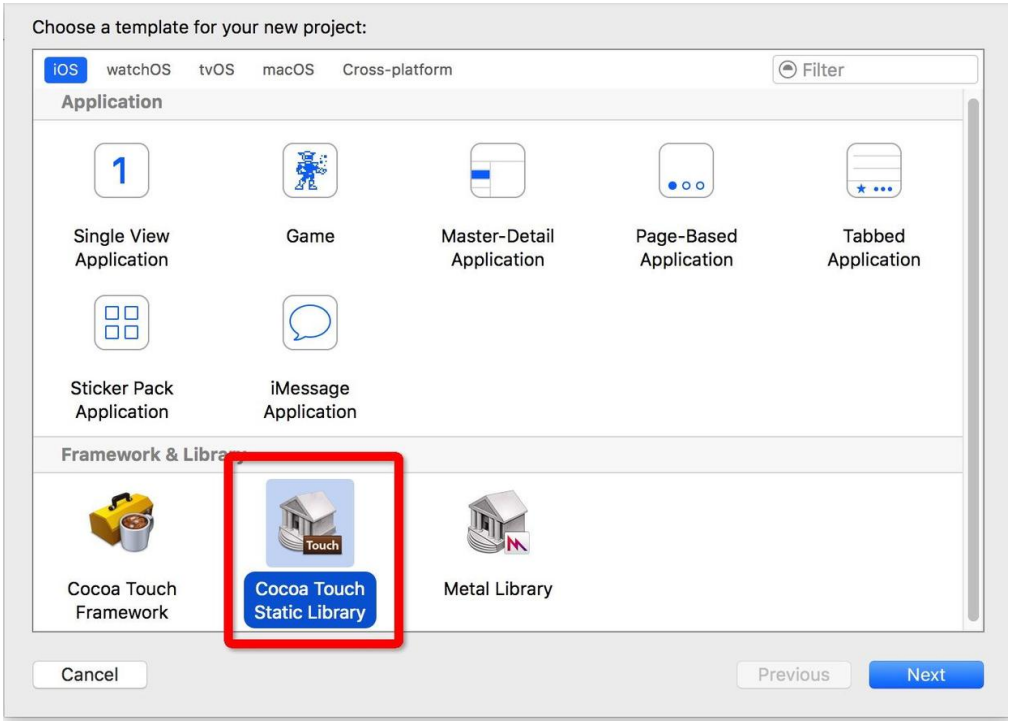
注：iPhone 7、iPhone 7 Plus、iPhone 8、iPhone 8 Plus、iPhone X真机到底是什么架构暂时不得而知（太穷，买不起～ ～），但是模拟器是x86_64。

> 三、打包静态库

因为静态库存在两种形式，我们先看.a静态库的打包

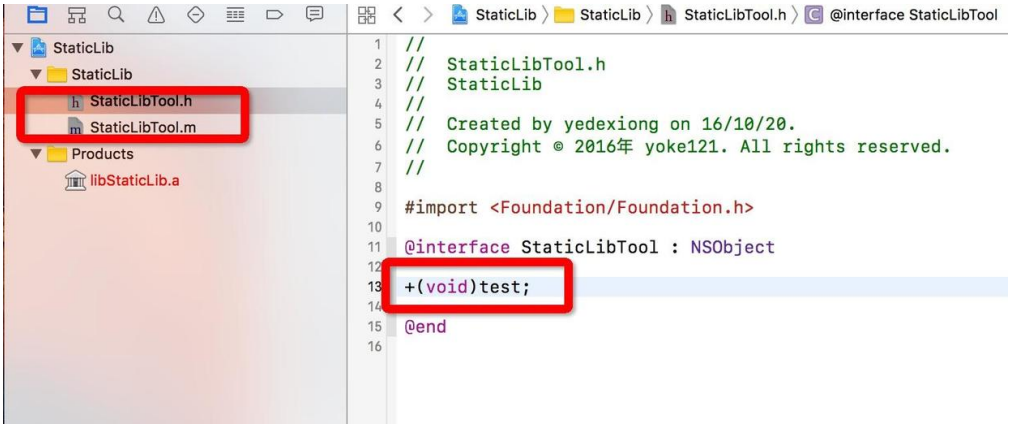
- .a文件静态库打包

1、打开Xcode创建一个新的工程，这里以Xcode8为例，选择工程如下：



创建一个新的工程

2、创建工程完毕后，再创建一个工具类StaticLibTool，添加一个方法用于测试



创建一个工具类，添加测试方法

StaticLibTool.m文件实现如下

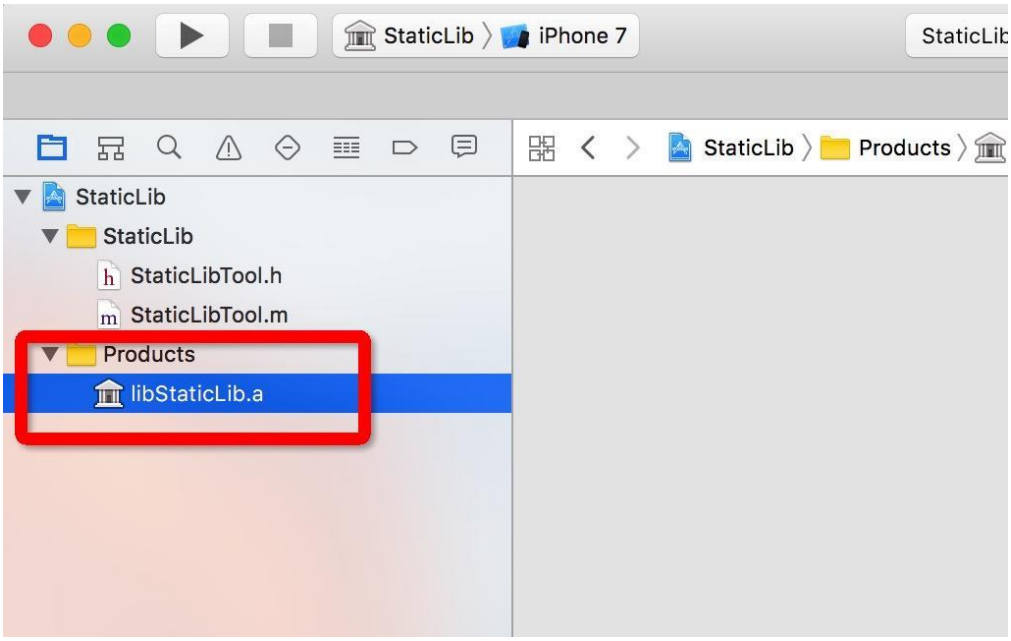
```
#import "StaticLibTool.h"

@implementation StaticLibTool

+(void)test
{
    NSLog(@"测试");
}
```

taticLibTool.m文件实现

3、运行工程进行打包

运行工程打包

运行完毕后，我们会看到工程中Products文件夹下的libStaticLib.a文件由红色变成了黑色。右键show in finder可以在其目录下找到它。这就是我们打包好的.a静态文件了。



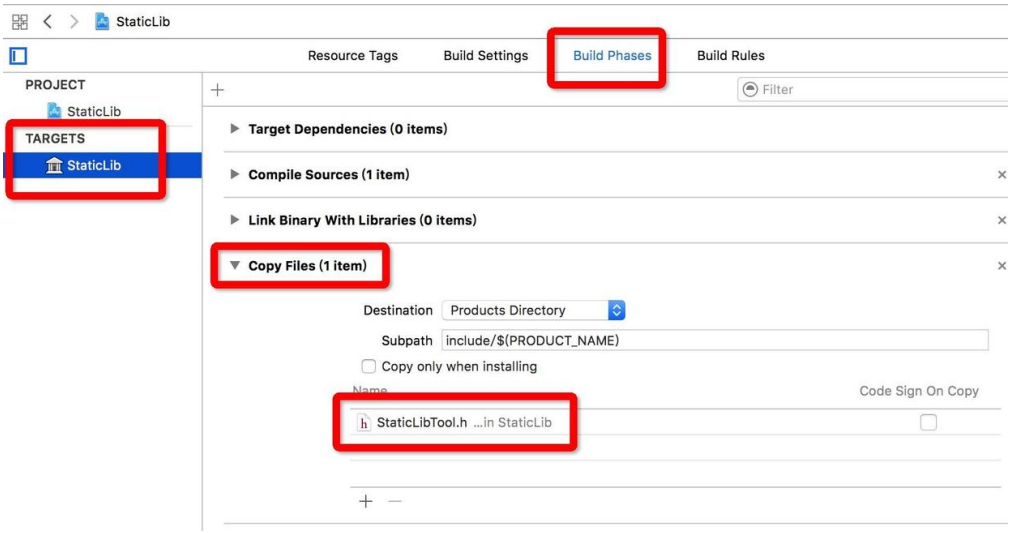
打包好的.a静态文件

但是这样就完了吗？当然没有，我们知道静态库存在的最大意义是隐藏代码的具体实现，但是这也隐藏的太彻底了，总要公开些接口或者头文件供人调用吧。

4、公开接口头文件

targets->Build Phases->Copy Files->"+"你需要公开的头文件

这里我们把新建的测试类StaticLibTool.h公开

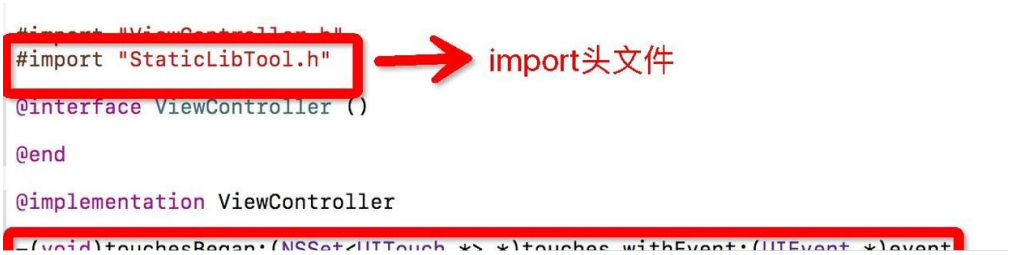


公开接口头文件

公开头文件后，我们再按上述1、2、3流程重新运行打包，我们会得到一个头文件和一个.a静态库（如下图），而这正是我们所需要的。

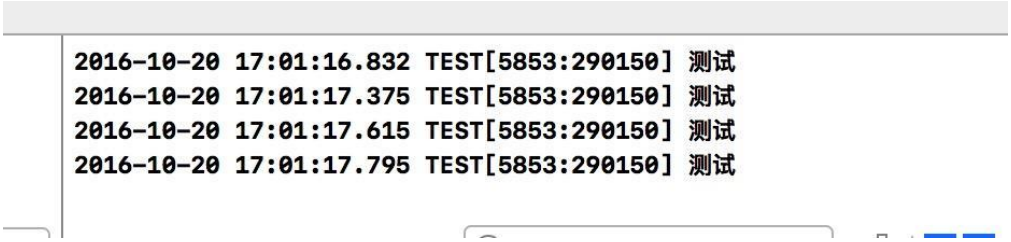
重新运行打包

5、新建一个可运行的工程，把这两个打包好的文件拖入项目测试



测试

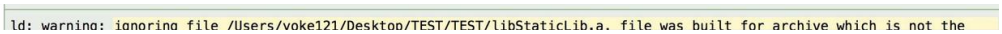
选择Iphone7模拟器运行，程序正常运行，点击模拟器屏幕，打印日志如下：



日志输出

我们可以看到输出没有问题，打包.a静态库大功告成。

但是，别高兴的太早。当我把模拟器切换成Iphone5运行时，编译直接不通过，报错如下：



iPhone 5模拟器运行时的编译错误

上图“Undefined symbols for architecture i386”是什么意思呢？意思是我们的libStaticLib.a静态库不支持i386架构。那i386又是什么鬼？不清楚的可以拉上去看“iOS 设备的CPU架构”，这里就不多做解释了。

iPhone 5模拟器正好是i386架构，而我们打包的静态库不支持。但是iPhone 7模拟器运行却没有问题，这说明我们打包的静态库正好支持iPhone 7模拟器 的cpu架构 x86_64。如何查看静态库所支持的架构，请看下一步。

6、终端查看静态库所支持的架构

终端–>cd进入库文件路径–>lipo –info 库名

终端查看静态库所支持的架构

上图可以看到，我们的静态库仅支持x86_64架构，也就是说此静态库只可运行在iphone5s–iphone7plus之间的模拟器设备。所以刚才我们运行iphone5模拟器时，编译会报错。

到这里就可以进一步解释下，打包静态库时，你用什么模拟器运行，打包出来的静态库就支持什么模拟器的架构，而刚才我打包时是用iPhone7运行，所以仅支持架构x86_64。那么这就太麻烦了，可以打包一个静态库支持多种架构的模拟器吗？答案是肯定的，请看下一步。

7、设置适配所有模拟器架构

project –> buildSeting –> Build Active Architecture Only 设为NO

设置适配所有模拟器架构

设置完成后，我们重新运行打包静态库文件（这时你可随便选一个模拟器），按照上述第6步终端查看其支持的架构，我们可以看到终端输出的结果是同时支持 i386和x86_64，这也就意味着同时支持所有模拟器。

到这里打包.a静态库已经告一段落，但是按上述流程打包的只能在模拟器上跑，真机是不能运行的，因为ios真机设备跟模拟器的架构又不一样（怎么不一样自己拉上去看），所以还没完（我也不想啊），请看下一步

8、打包支持真机架构的静态库

所有流程都跟上面的一样，只是我们运行打包时要选择真机运行，如下图你可以选择自己插上去的真机，也可以选择Generic ios Devices。当然不要忘记了设置支持所有真机机型架构： Build Active Architecture Only 设为NO。

打包支持真机架构的静态库

我们可以看下打包出来的终端查看结果如下：

```
Last login: Fri Oct 21 10:04:36 on ttys000
[yoke121deMacBook-Pro:~ yedexiong$ cd /Users/yoke121/Desktop/未命名文件夹
[yoke121deMacBook-Pro:未命名文件夹 yedexiong$ lipo -info libStaticLib.a
Architectures in the fat file: libStaticLib.a are: armv7 arm64
yoke121deMacBook-Pro:未命名文件夹 yedexiong$
```

终端输出结果

上图可以看到同时支持armv7和arm64，也就是支持所有ios设备。好了到此打包.a静态库算是告一段落。

如果要同时支持模拟器和真机，请使用命令合成.a静态库：lipo -create name1.a所在路径 name2.a所在路径 -output newname.a

- **.frameworke文件静态库打包**

1、依然Xcode创建一个新的工程FrameworkeLib，选择工程如下：

创建一个新的工程

创建完成后我们可以看到，工程本身自带一个FrameworkeLib.h文件，这是类似一个主头文件一样的东西

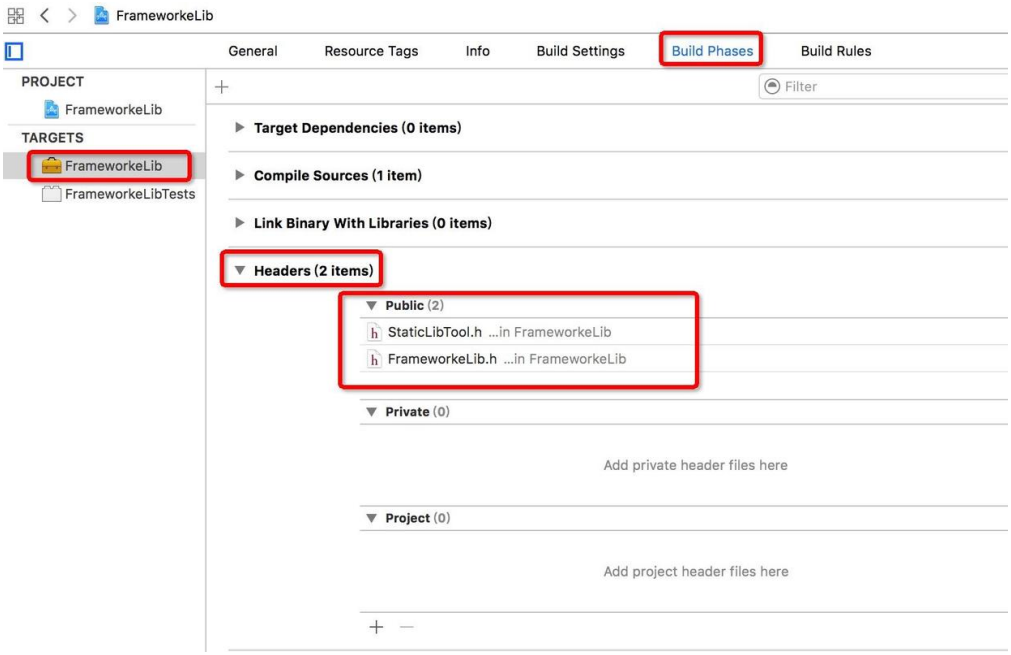
FrameworkeLib.h文件

2、创建需要测试的类，为了方便我把上述打包.a的测试类StaticLibTool直接拖来使用。

3、设置支持所有模拟器架构或真机架构（和打包.a第7步骤一样）

4、公开头文件

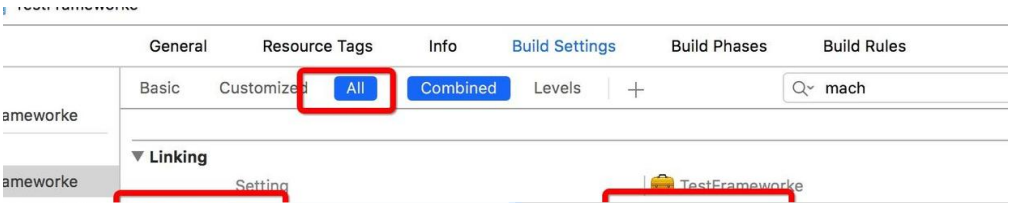
target–Build Phases – Headers –把需要公开的头文件从project拖入Public



暴露头文件

5、设置打包的是静态库。因为动态库也可以是以framework形式存在，所以需要设置，否则默认打出来的是动态库(注意：如果要上线AppSotre，一定要改成静态库，否则审核通不过)

target–>BuildSetting –>搜索关键字mach–>Mach–o Type 设为Static Library（这个默认选项是动态的）



设置打包的是静态库

6、选中真机或模拟器运行设备打包（与打包.a一样），完成后Products文件夹下的FrameworkeLib.framework文件由红色变成了黑色，右键show in finder 显示如下：

打包结果

FrameworkeLib.framework拖入项目便可直接使用，这里就不再进行测试了。此外还要补充的一点是，打包静态库的时候还需注意打包的是测试版（Debug）还是发布版（Release），这个根据你自己的需求决定，而如何进行设置请下一步骤。

7、设置打包静态库的测试版和发布版（.a和.frameworke）

product -> scheme -> Edit scheme -> Run->选择Debug或Release

如果要同时支持模拟器和真机，和.a类似，请使用命令合成framework库：`lipo -create <name1>.framework/<name1> <name2>.framework/<name2> -output newname`
