目录

iOS逆向开发：iOSOpenDev开发插件

- 最新版本：`v1.8.2`
- 更新时间：`20250115`

简介

介绍iOS逆向中如何用iOSOpenDev开发越狱插件tweak。先是对iOSOpenDev概览；然后介绍如何安装iOSOpenDev，以及安装后确认安装成功；然后是普通的插件的开发流程，包括新建iOSOpenDev的Xcode项目、初始化项目的配置，其中包括ssh免密登录、写hook插件tweak代码，包括如何新增文件、编译代码调试代码等；以及带UI界面的插件开发的流程，包括前提和目标、基本流程、使用效果、常见问题等；以及常见问题和一些经验心得；常见问题包括初始化环境方面的问题和编译调试方面的问题。

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- crifan/ios_re_iosopendev_tweak: iOS逆向开发：iOSOpenDev开发插件

如何使用此HonKit源码去生成发布为电子书

详见：crifan/honkit_template: demo how to use crifan honkit template and demo

在线浏览

- iOS逆向开发：iOSOpenDev开发插件 book.crifan.org
- iOS逆向开发：iOSOpenDev开发插件 crifan.github.io

离线下载阅读

- iOS逆向开发：iOSOpenDev开发插件 PDF
- iOS逆向开发：iOSOpenDev开发插件 ePub
- iOS逆向开发：iOSOpenDev开发插件 Mobi

版权和用途说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

其他

作者的其他电子书

本人 `crifan` 还写了其他 `150+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

关于作者

关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

iOSOpenDev概览

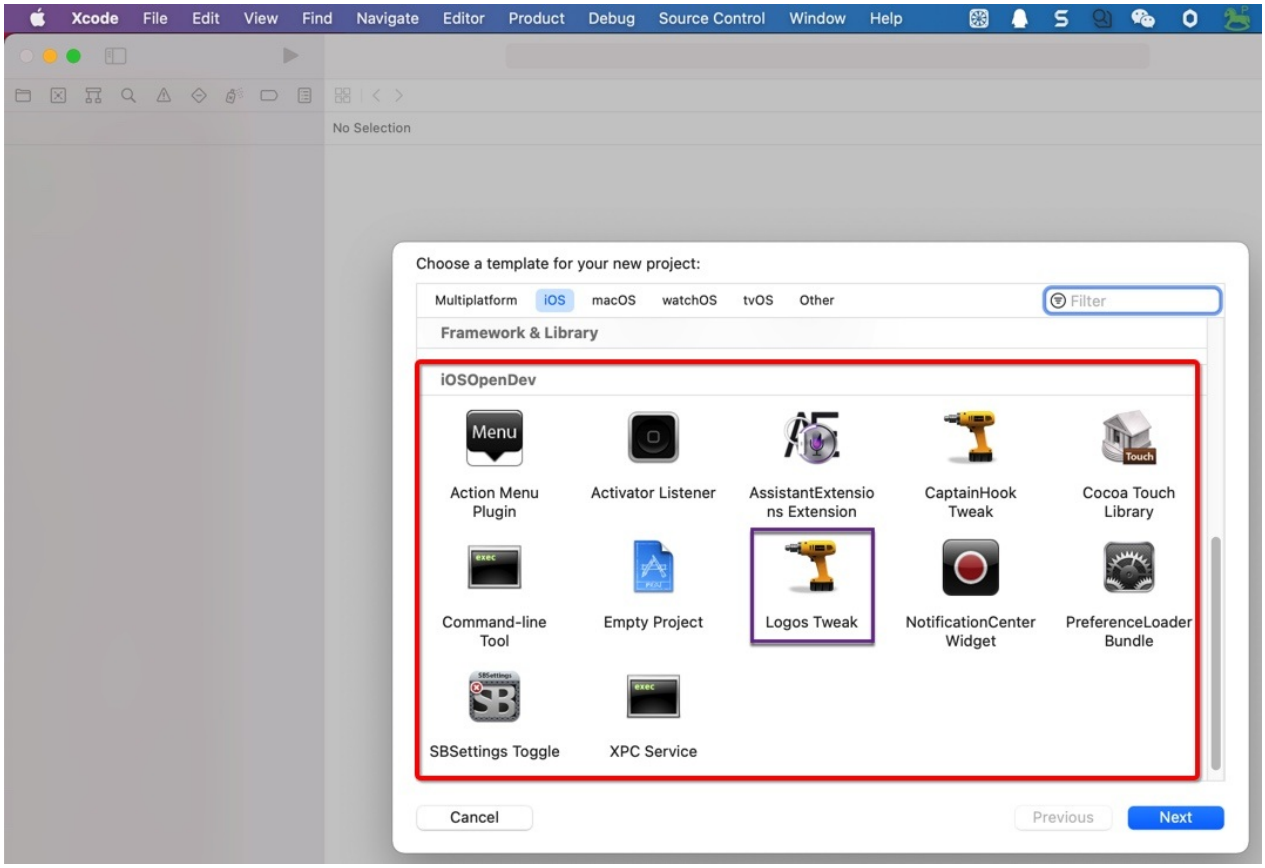
在iOS逆向期间，往往涉及到去开发越狱插件tweak，其中常见的工具=框架之一就是：`iOSOpenDev`。

- `iOSOpenDev`
  - 概述：iOS越狱插件tweak开发框架之一
    - 其他竞品
      - Theos/Logos
      - MonkeyDev
  - 常用核心功能：支持基于XCode（的模板）去创建Logos的tweak越狱插件
  - 官网
    - http://iosopendev.com/
    - 下载
      - http://iosopendev.com/download/
  - Github
    - https://github.com/kokoabim/iOSOpenDev
    - Wiki
      - https://github.com/kokoabim/iOSOpenDev/wiki
      - 对于Logos（=Theos？）的支持
        - https://github.com/kokoabim/iOSOpenDev/wiki/Logos-(Theos)-Support
      - 项目转换
        - https://github.com/kokoabim/iOSOpenDev/wiki/Convert-to-iOSOpenDev-Project

安装iOSOpenDev

从官网iOSOpenDev—Download下载到：iOSOpenDev-1.6-2.pkg，双击去安装：

成功安装后，去 `Xcode` 中新建 `iOS` 项目，即可看到 `iOSOpenDev` 的的选项：



确认安装成功

环境变量

再去确认，是否把iOSOpenDev的相关环境变量，加到启动脚本（此处是`zsh`，所以是`./zshrc`）中了：

```
➜  ~ cat ~/.zshrc | grep iOSOpenDev
export iOSOpenDevPath=/opt/iOSOpenDev
export iOSOpenDevDevice=
export PATH=/opt/iOSOpenDev/bin:$PATH
```

如果没有：

```
✘ crifan@licrifandeMacBook-Pro ~ /opt/iOSOpenDevSetup/bin ~ cat ~/.zshrc | grep iOSOpenDev
```

则自己手动去加上：

```
crifan@licrifandeMacBook-Pro ~ /opt/iOSOpenDevSetup/bin ~ vi ~/.zshrc
crifan@licrifandeMacBook-Pro ~ /opt/iOSOpenDevSetup/bin ~ cat ~/.zshrc | grep iOSOpenDev
export iOSOpenDevPath=/opt/iOSOpenDev
export iOSOpenDevDevice=
export PATH=/opt/iOSOpenDev/bin:$PATH
crifan@licrifandeMacBook-Pro ~ /opt/iOSOpenDevSetup/bin ~ source ~/.zshrc
```

Xcode中的iOSOpenDev的模板

确认是否有多出的template模板：

```
➜  ~ ll ~/Library/Developer/Xcode/
total 0
drwxr-xr-x  8 crifan  staff   256B 10 14 11:13 DerivedData
srwxr-xr-x  1 crifan  staff     0B 10 27 08:54 GPUToolsAgent.sock
drwxr-xr-x  3 crifan  staff    96B 10 27 08:49 Templates
drwxr-xr-x  6 crifan  staff   192B 10 26 22:37 UserData
drwxr-xr-x  5 crifan  staff   160B  9 30 22:11 iOS Device Logs
drwxr-xr-x  4 crifan  staff   128B 10 13 13:54 iOS DeviceSupport
➜  ~ ll ~/Library/Developer/Xcode/Templates
total 0
lrwxr-xr-x  1 root  staff    25B 10 27 08:49 iOSOpenDev -> /opt/iOSOpenDev/templates
```

此处是有的：

- 多出了软链接：
  - `~/Library/Developer/Xcode/Templates/iOSOpenDev`
    - 指向的是：
      - `/opt/iOSOpenDev/templates`

以及接着去看看，具体有哪些模板：

```
➜  ~ ll /opt/iOSOpenDev/templates
total 48
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 Action Menu Plugin.xctemplate
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 Activator Listener.xctemplate
drwxr-xr-x  12 root  wheel   384B 10 27 08:32 AssistantExtensions Extension.xctemplate
drwxr-xr-x   4 root  wheel   128B 10 27 08:32 Base.xctemplate
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 CaptainHook Tweak.xctemplate
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 Cocoa Touch Library.xctemplate
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 Command-line Tool.xctemplate
drwxr-xr-x   4 root  wheel   128B 10 27 08:32 Debian Package.xctemplate
```

```
drwxr-xr-x   4 root  wheel   128B 10 27 08:32 Empty Project.xctemplate
-rw-r--r--   1 root  wheel    18K 10 27 08:49 LICENSE
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 Logos Tweak.xctemplate
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 ManPage.xctemplate
drwxr-xr-x  11 root  wheel   352B 10 27 08:32 NotificationCenter Widget.xctemplate
drwxr-xr-x  12 root  wheel   384B 10 27 08:32 PreferenceLoader Bundle.xctemplate
drwxr-xr-x   7 root  wheel   224B 10 27 08:32 PreferenceLoader.xctemplate
-rw-r--r--   1 root  wheel   352B 10 27 08:49 README.md
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 SBSettings Toggle.xctemplate
drwxr-xr-x   4 root  wheel   128B 10 27 08:32 Unit Tests.xctemplate
drwxr-xr-x   7 root  wheel   224B 10 27 08:32 XPC Service.xctemplate
```

很明显，部分模板，应该就是对应着界面中看到的各个模板：

比如：

- `Logos Tweak.xctemplate` -> `Logos Tweak`
- `Command-line Tool.xctemplate` -> `Command-line Tool`
- `PreferenceLoader Bundle.xctemplate` -> `PreferenceLoader Bundle`

iOSOpenDev中的内容

顺带再去看看，当前iOSOpenDev目录中的内容：

```
➜  /opt ll
total 0
drwxr-xr-x  9 root  wheel   288B 10 27 08:32 iOSOpenDev
drwxr-xr-x  3 root  wheel    96B 10 27 08:32 iOSOpenDevSetup
drwxr-xr-x  3 root  wheel    96B 10 27 08:32 iOSOpenDevUninstall
➜  /opt cd iOSOpenDev
➜  iOSOpenDev pwd
/opt/iOSOpenDev
➜  iOSOpenDev ll
total 48
-rw-r--r--   1 root  wheel    18K 10 27 08:49 LICENSE
-rw-r--r--   1 root  wheel   352B 10 27 08:49 README.md
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 bin
drwxr-xr-x   2 root  wheel    64B 10 27 08:32 frameworks
drwxr-xr-x   8 root  wheel   256B 10 27 08:32 include
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 lib
drwxr-xr-x  21 root  wheel   672B 10 27 08:32 templates
➜  iOSOpenDev ll bin
total 3000
-rwxr-xr-x  1 root  wheel   428K 10 27 08:49 class-dump
-rwxr-xr-x  1 root  wheel   628K 10 27 08:49 class-dump-z
-rwxr-xr-x  1 root  wheel    59K 10 27 08:49 iosod
-rwxr-xr-x  1 root  wheel   383K 10 27 08:49 ldid
➜  iOSOpenDev ll frameworks
➜  iOSOpenDev ll include
total 48
drwxr-xr-x   3 root  wheel    96B 10 27 08:32 ActionMenu
drwxr-xr-x   3 root  wheel    96B 10 27 08:32 AssistantExtensions
drwxr-xr-x   3 root  wheel    96B 10 27 08:32 CaptainHook
drwxr-xr-x  10 root  wheel   320B 10 27 08:32 libactivator
drwxr-xr-x   3 root  wheel    96B 10 27 08:32 logos
-rw-r--r--   1 root  wheel    21K 10 27 08:49 substrate.h
➜  iOSOpenDev ll lib
total 1216
-rwxr-xr-x  1 root  wheel    77K 10 27 08:49 libactionmenu.dylib
-rwxr-xr-x  1 root  wheel   422K 10 27 08:49 libactivator.dylib
-rwxr-xr-x  1 root  wheel   101K 10 27 08:49 libsubstrate.dylib
➜  iOSOpenDev ll templates
total 48
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 Action Menu Plugin.xctemplate
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 Activator Listener.xctemplate
drwxr-xr-x  12 root  wheel   384B 10 27 08:32 AssistantExtensions Extension.xctemplate
drwxr-xr-x   4 root  wheel   128B 10 27 08:32 Base.xctemplate
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 CaptainHook Tweak.xctemplate
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 Cocoa Touch Library.xctemplate
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 Command-line Tool.xctemplate
drwxr-xr-x   4 root  wheel   128B 10 27 08:32 Debian Package.xctemplate
drwxr-xr-x   4 root  wheel   128B 10 27 08:32 Empty Project.xctemplate
```

```
-rw-r--r--   1 root  wheel    18K 10 27 08:49 LICENSE
drwxr-xr-x   6 root  wheel   192B 10 27 08:32 Logos Tweak.xctemplate
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 ManPage.xctemplate
drwxr-xr-x  11 root  wheel   352B 10 27 08:32 NotificationCenter Widget.xctemplate
drwxr-xr-x  12 root  wheel   384B 10 27 08:32 PreferenceLoader Bundle.xctemplate
drwxr-xr-x   7 root  wheel   224B 10 27 08:32 PreferenceLoader.xctemplate
-rw-r--r--   1 root  wheel   352B 10 27 08:49 README.md
drwxr-xr-x   5 root  wheel   160B 10 27 08:32 SBSettings Toggle.xctemplate
drwxr-xr-x   4 root  wheel   128B 10 27 08:32 Unit Tests.xctemplate
drwxr-xr-x   7 root  wheel   224B 10 27 08:32 XPC Service.xctemplate
```

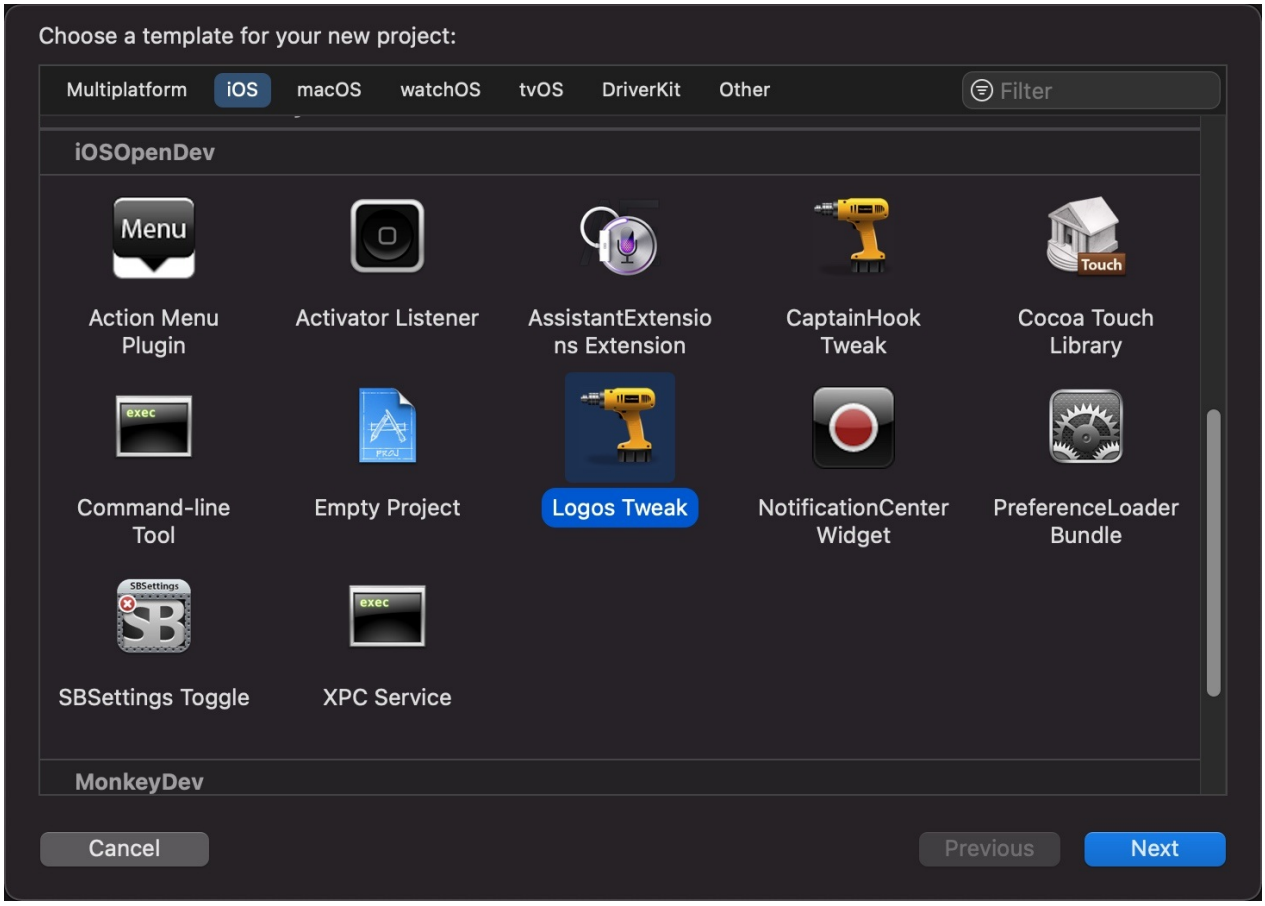
普通的插件开发流程

TODO：

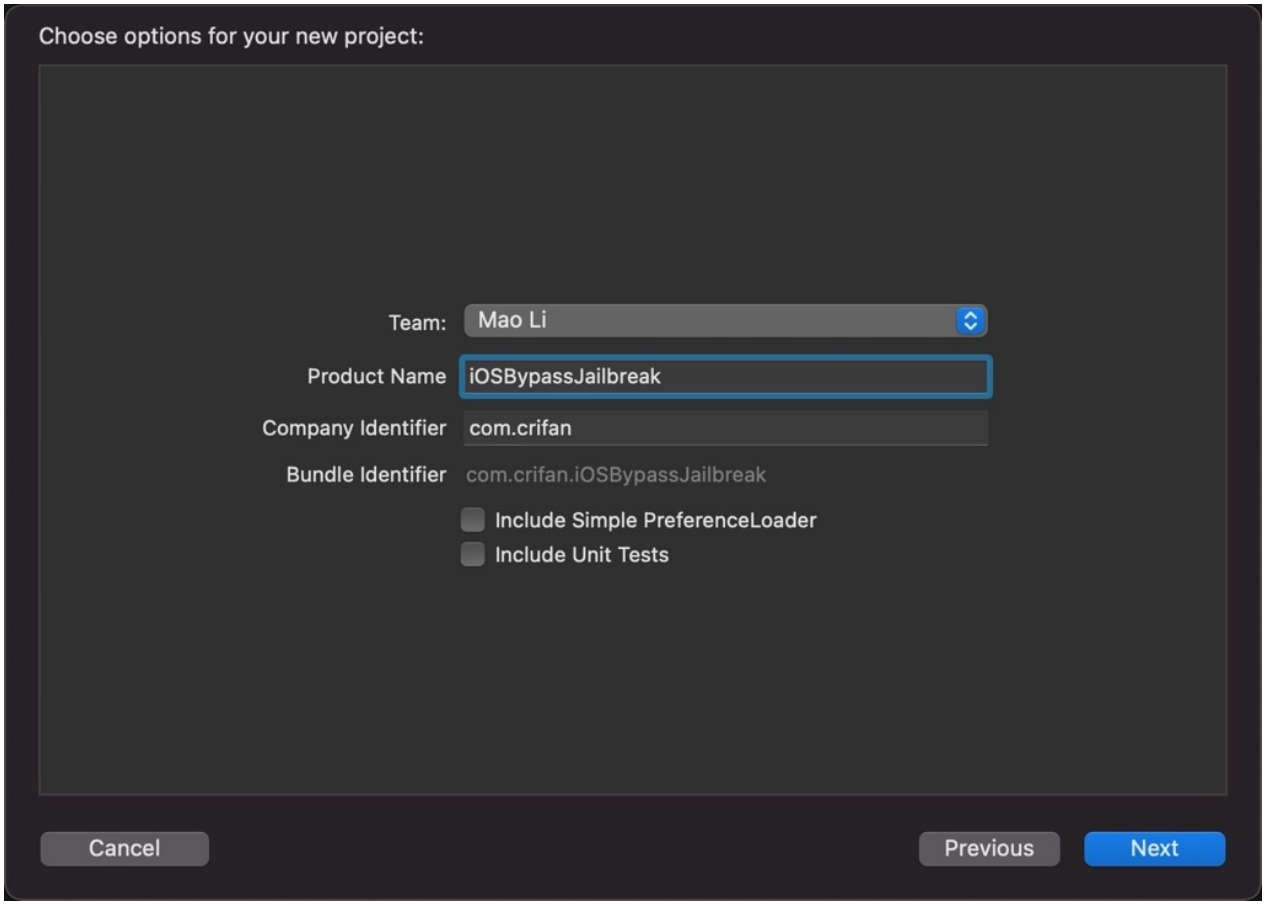
- 【已解决】Mac中用iOSOpenDev开发iOS的theos的Logos的tweak插件
- 【已解决】给iOSOpenDev的Logos的tweak的XCode项目去做基本配置

新建iOSOpenDev的Xcode项目

`Xcode` 中新建 `iOS` 项目，选择： `iOSOpenDev` -> `Logos Tweak` ：



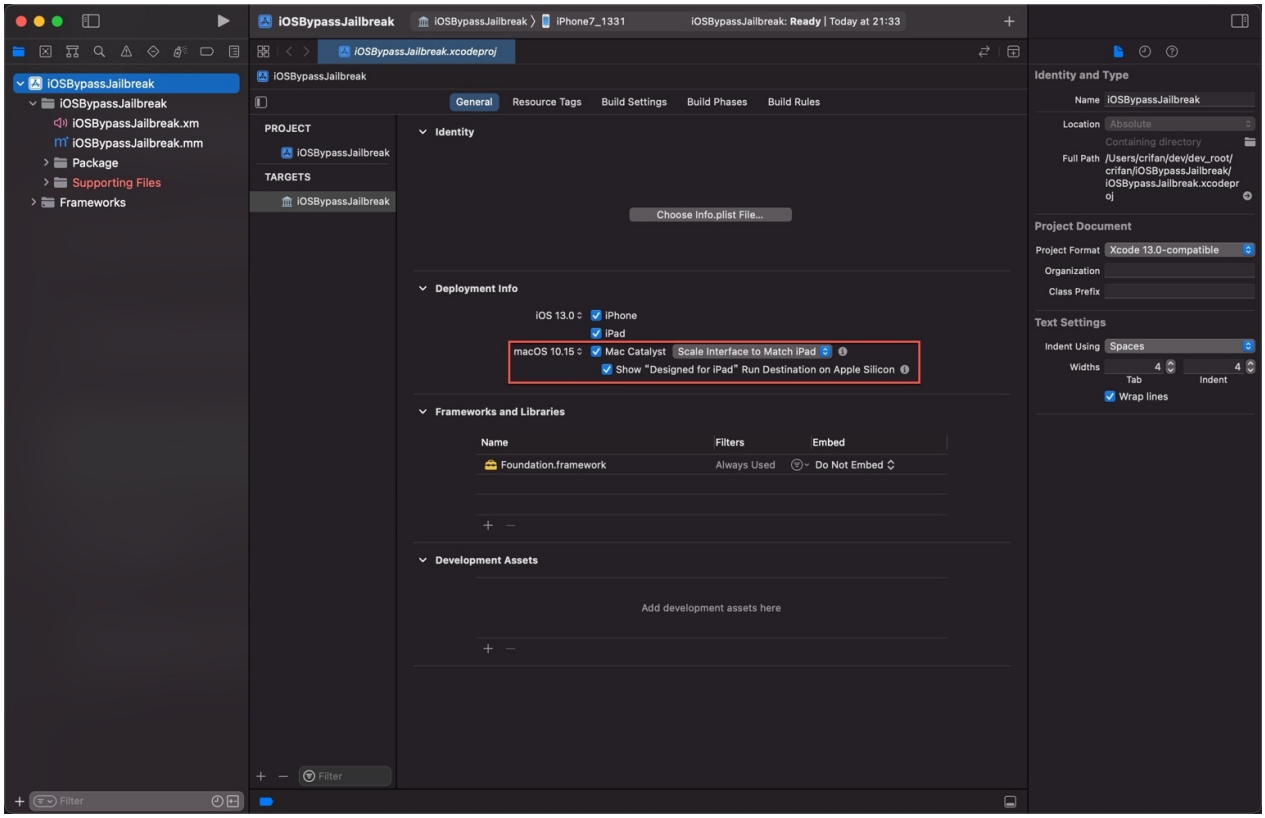
然后填写项目信息：



比如iOSBypassJailbreak的：

- `Product Name`：`iOSBypassJailbreak`
- `Company Bundle`：`com.crifan`
- `Bundle Identifier`：自动生成出 `com.crifan.iOSBypassJailbreak`

点击 `Next` 继续，即可新建出，看起来和普通 `Xcode` 没多大区别的项目：



初始化配置iOSOpenDev的Xcode项目

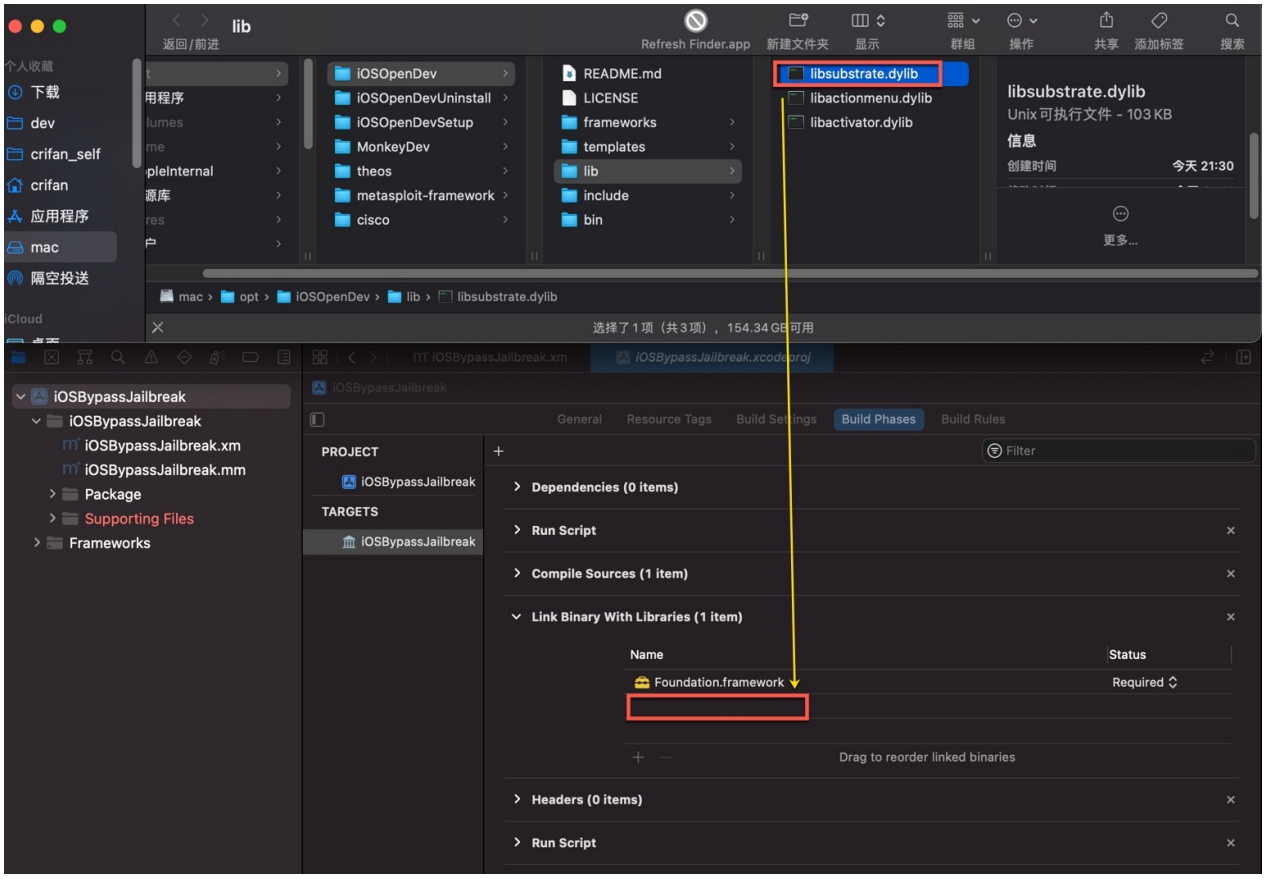
去掉 `Deploymen Info` 中的 `Mac` 和确保iOS版本一致

此处，先去做第一个配置方面的改动：

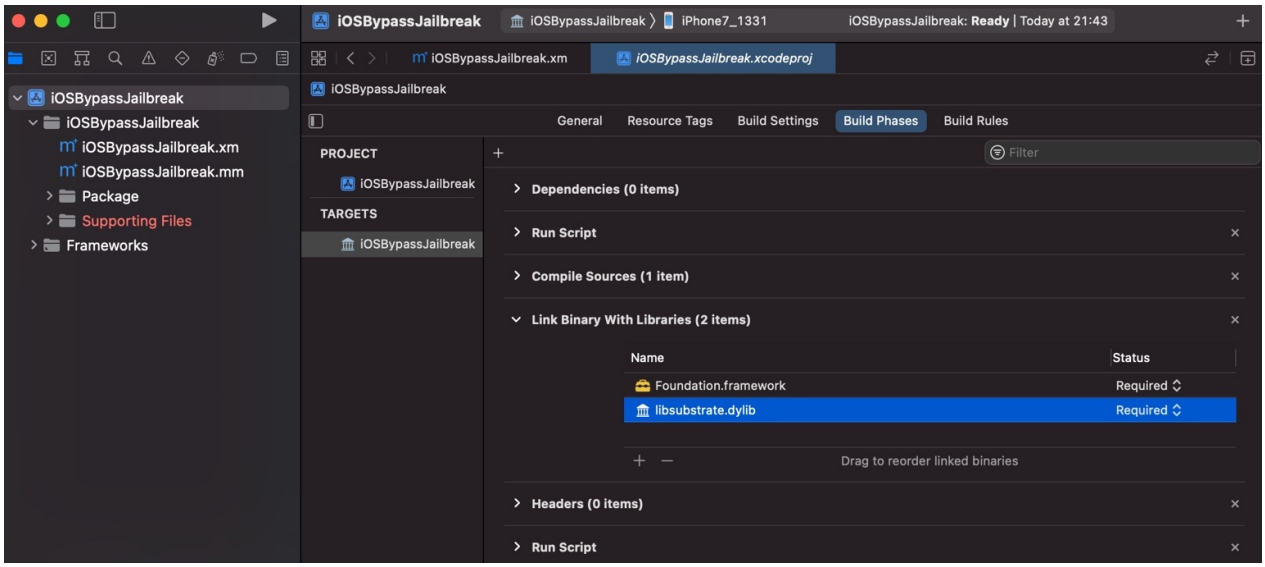
- `TARGETS` -> `General` -> `Deploymen Info` ： 去掉 `Mac` ，因为我们开发的是 `iOS` 的插件，不需要发布到 `Mac` ，且设置好iOS的最低版本
    -
- `PROJECT` -> `Info` -> `Deployment Target` -> `iOS Deployment Target` ， 也要同步设置 `iOS` 的最低版本
    -

导入依赖库 `libsubstrate.dylib`

把 `libsubstrate.dylib` (一般在 `/opt/iOSOpenDev/lib/libsubstrate.dylib`)：



导入到项目中的： `Targets` -> `YourProjectName` -> `Build Phases` -> `Link Binary With Libraries`



设置被hook的app包名或二进制文件名

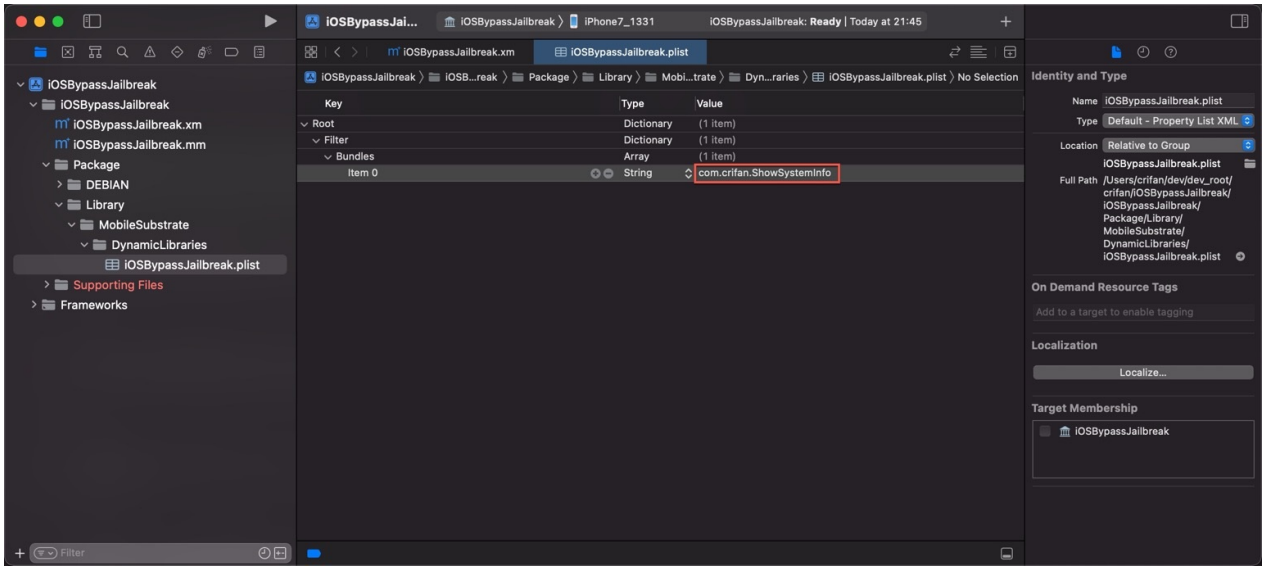
去把要hook的，被拦截的app的包名，加到被hook的包名的列表中：

`YourProjectName` -> `YourProjectName` -> `Package` -> `Libarary` -> `MobileSubstrate` -> `DynamicLibraries` -> `CurrentProjectBundleIdentifier.plist`

在 `Root` -> `Filter` -> `Bundles` ，会看到 `Item 0` ：

- `Type` ： `String`
- `Value` ： 填入你要hook的app的包名
  - 举例

    - `com.crifan.ShowSystemInfo`



- 另外
  - 如果要新增一行
    - 移动到 `Item 0` 所在的行，会看到出现个 `=` `加号` ，点击 `加号` ，会新增一行

如何hook二进制?

如果需要hook二进制，则是新建 `Array` 类型的（和 `Bundles` 并列的） `Executables` 子项，再加上对应的二进制文件名

举例：

- hook二进制： `akd` = `AuthKit.framework` 的daemon进程

  -

此时，对应的 `plist` 文件内容是：

- jailAppleAccount/Package/Library/MobileSubstrate/DynamicLibraries/jailAppleAccount.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>Filter</key>
  <dict>
    <key>Executables</key>
    <array>
      <string>akd</string>
      <string>amsaccountsd</string>
      <string>AppleMediaServices</string>
      <string>AppleAccount</string>
      <string>Preferences</string>
    </array>
    <key>Bundles</key>
    <array>
      <string>com.apple.Preferences</string>
    </array>
  </dict>
</dict>
</plist>
```

更新iOSOpenDev相关设置

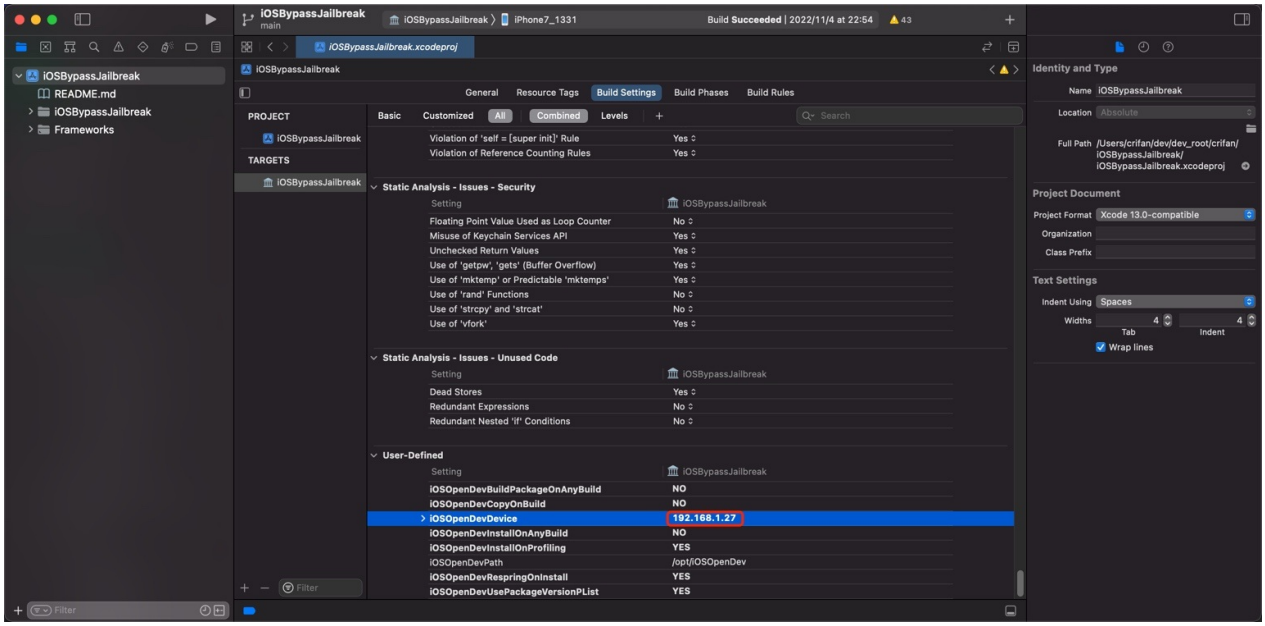
设置iPhone的IP

把此处要把iOS调试设备（iPhone）中的IP地址：

此处是：`192.168.1.27`

去加到配置中去：

- `iOSOpenDevDevice` = `192.168.1.27`



更新iOSOpenDevUsePackageVersionPList为NO，确保版本更新能生效

以及参考后续的：

control的Version版本号的改动会丢失

再去从：

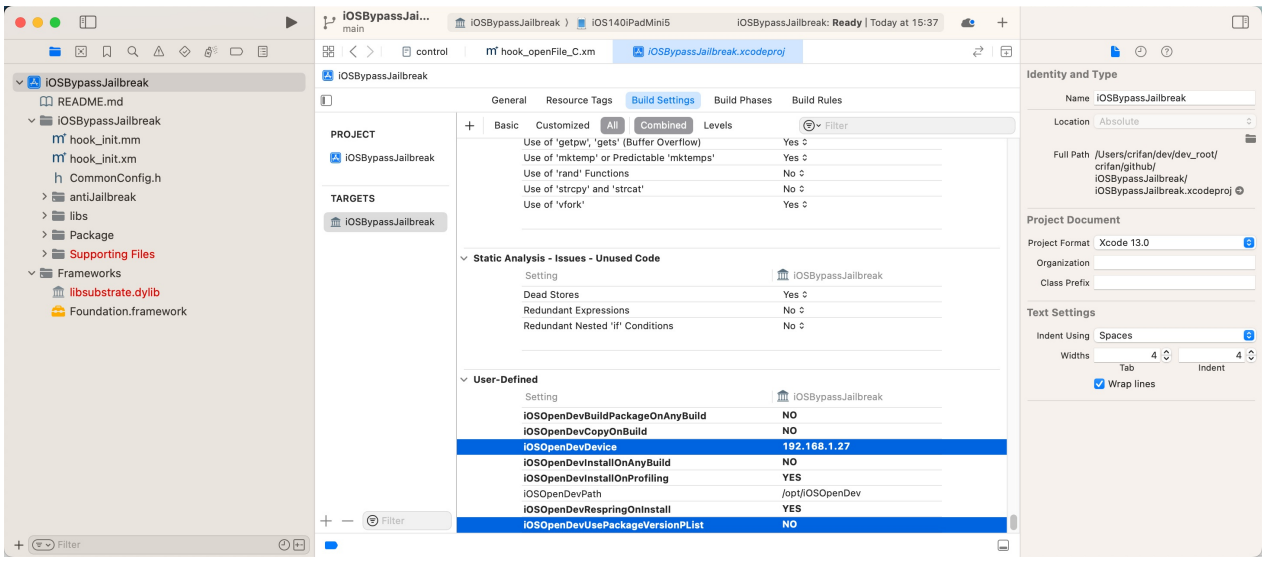
- `iOSOpenDevUsePackageVersionPList = YES`

改为：

- `iOSOpenDevUsePackageVersionPList = NO`

更新后的配置是：

```
iOSOpenDevBuildPackageOnAnyBuild = NO
iOSOpenDevCopyOnBuild = NO
iOSOpenDevDevice = 192.168.1.27
iOSOpenDevInstallOnAnyBuild = NO
iOSOpenDevInstallOnProfiling = YES
iOSOpenDevPath = /opt/iOSOpenDev
iOSOpenDevRespringOnInstall = YES
iOSOpenDevUsePackageVersionPList = NO
```



另外，理论上，去把对应变量加到环境变量：

```
➜  ~ cat ~/.zshrc | grep iOSOpenDevDevice
export iOSOpenDevDevice=192.168.1.27
```

效果应该也是一样的。

确保ssh登录且是ssh免密登录

- 此处要弄好SSH登录，且是ssh免密登录
  - 背景
    - 在 `Xcode` + `iOSOpenDev` 的编译安装最后阶段，涉及到，自动通过ssh访问iPhone设备，把生成的 `.deb` 插件的文件下载和安装到iPhone中
    - 此时就需要先准备好环境：确保 `Mac` 中可以，ssh的免密登录iPhone
  - 概述
    - 先用ssh登录一次iPhone
      - 命令

        ```
        ssh root@192.168.1.27
        ```

      - 输入密码
        - `OpenSSH` 的默认密码是： `alpine`
      - 即可登录到iPhone中
    - 把ssh的key拷贝到iPhone中
      - 命令

        ```
        ssh-copy-id root@192.168.1.27
        ```

      - 输入密码： `alpine`
  - 详解
    - ssh免密登录 · iOS越狱开发：常用越狱插件

搞懂.xm和.mm文件的逻辑

此处，在真正，新增hook代码文件，写hook代码之前，要去：

搞懂 `iOSOpenDev` 中 `.xm` 和 `.mm` 文件的逻辑

此处核心内容是：

- `.xm`：原始的hook插件的代码
- `.mm`：从 `.xm` 自动（在 `Build` 后）自动生成的文件

->

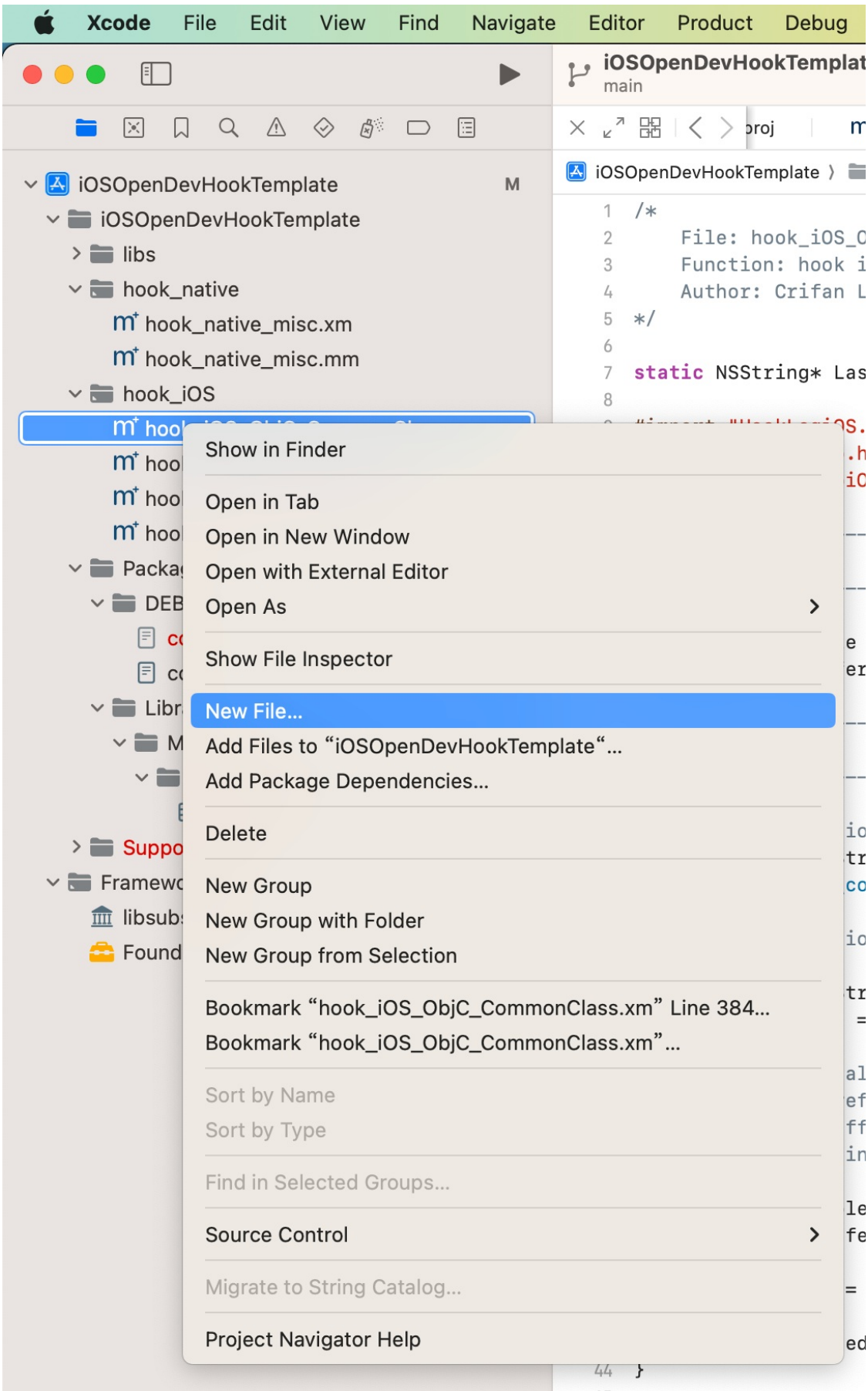
- 所以 = 结果
  - iOSOpenDev新增（hook代码逻辑的）文件时
    - 是：`.xm` 文件
    - 不是：`.mm` 文件
  - 写hook插件=添加hook代码逻辑时，改动的文件
    - 是：`.xm` 文件
    - 而不是：`.mm` 文件
  - 你每次改动更新 `.xm` 后
    - iOSOpenDev内部会自动从最新的 `.xm` 生成最新的 `.mm` 文件
      - 因此，旧的 `.mm` 文件（的内容）会被新的 `.mm` 文件（的内容）覆盖掉
        - 所以即使，如果，你之前改动了 `.mm` 文件，也是没用的，会被覆盖掉的
  - Xcode最终去编译（内部其实是iOS的 `clang` 编译器）时
    - 不支持：`.xm`
      - 万一 `Xcode` 的 `Compile Sources` 中（由于失误而）加入了 `.xm` 文件，则会导致编译报错
        - 具体详见：Compile Sources中误添加了不支持的.xm
    - 只支持：`.mm`
      - 所以接下来，要去把 `.mm` 加到 `Compile Souces`，供Xcode最终编译用
  - 首次（`Xcode` -> `Build`）编译时，会从 `.xm`，生成额外的 `.mm` 文件
    - 首次时=只用做一次
      - 概述
        - 需要你去 `Xcode` ->右键-> `Add Files to`
        - 去把 `.mm` 文件加进来（导入进来，弹框选项记得选择：`Copy items if needed`
        - 此时，对应的 `Xcode` -> `Targets` -> `Build Phases` -> `Compile Sources` 中，就可以看到对应的 `.mm` 文件了
      - 详见
        - 如何新增xm文件
    - 注
      - 后续则无需重复添加 `.mm` 文件

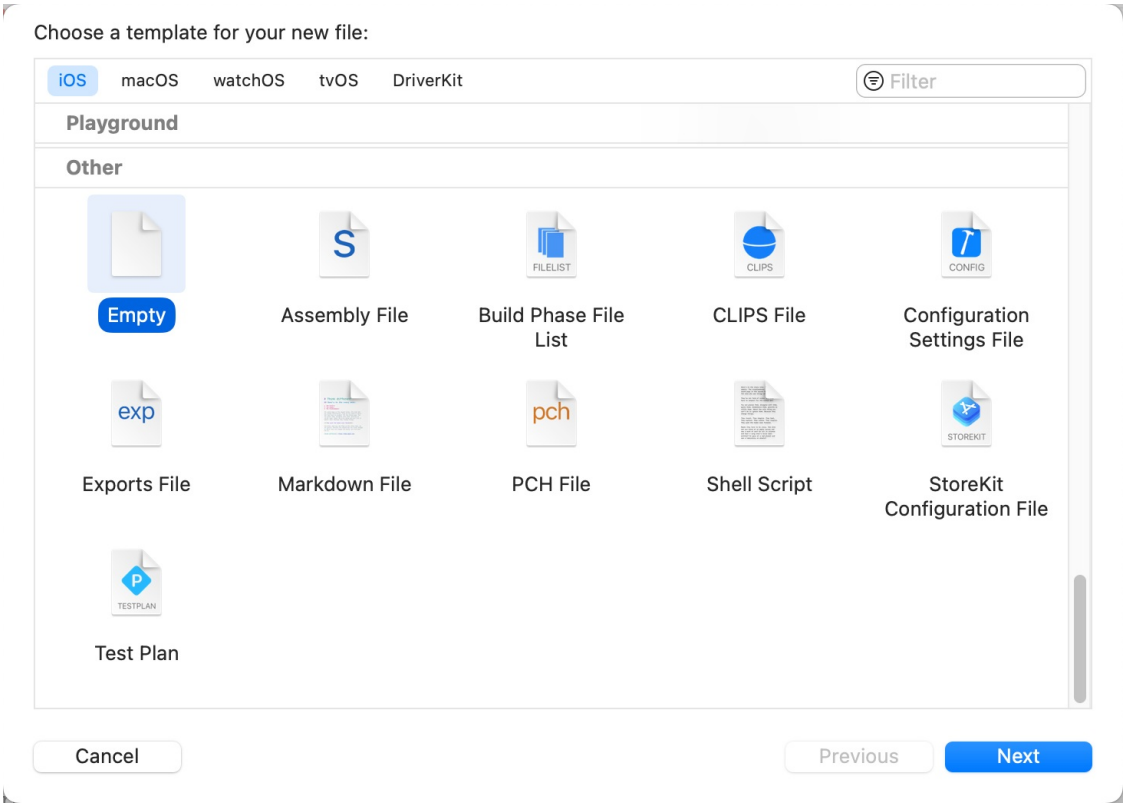
如何新增(`.xm` 和 `.mm`)文件

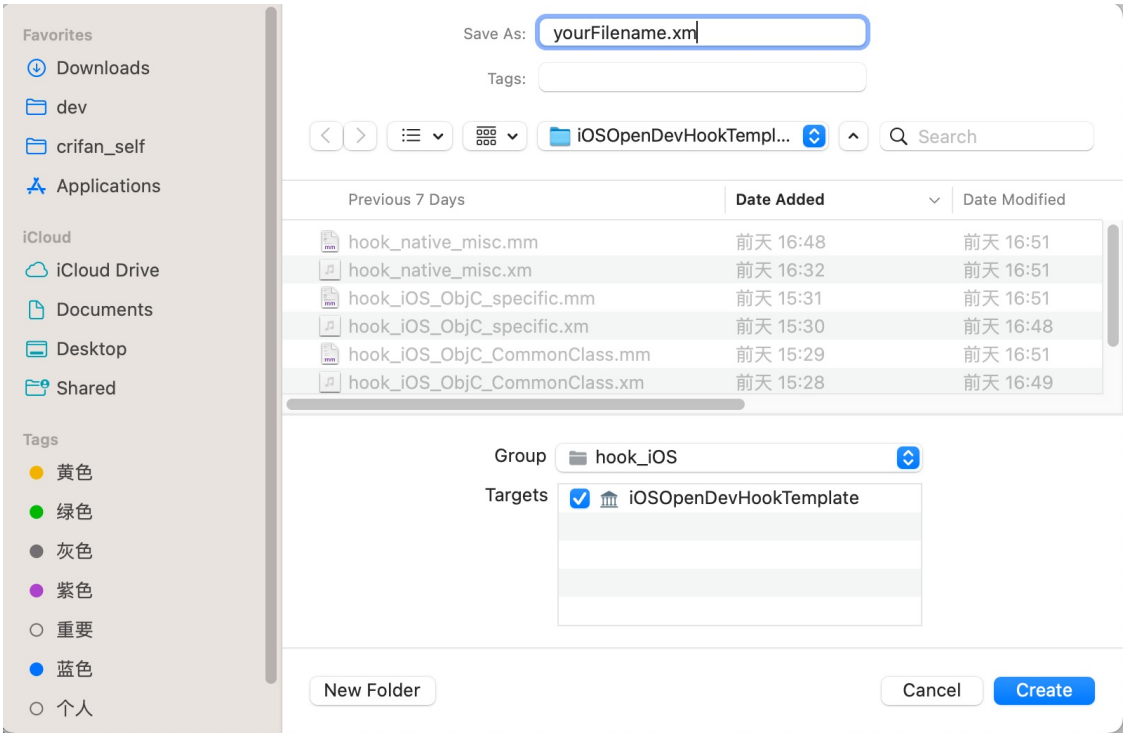
对于想要新增hook代码逻辑，而去新增 `.xm` 文件的相关逻辑是：

具体步骤是：

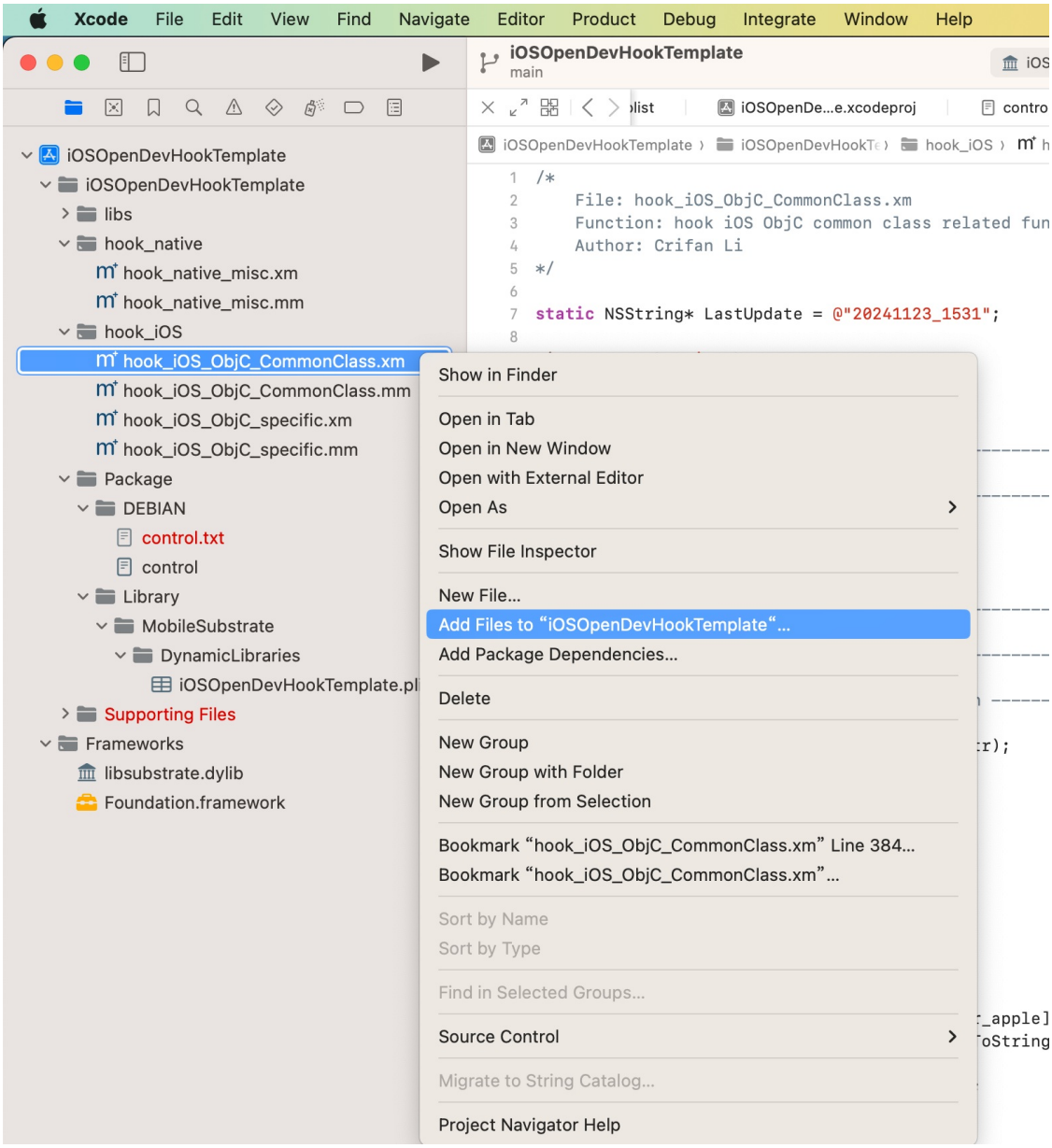
- 新建 `.xm` 文件
  - `Xcode` ->选中要新增文件所属的位置 -> 右键 -> `New File` -> `iOS` -> `Other` -> `Empty` ->输入文件名： `yourFilename.xm` -> `Create`

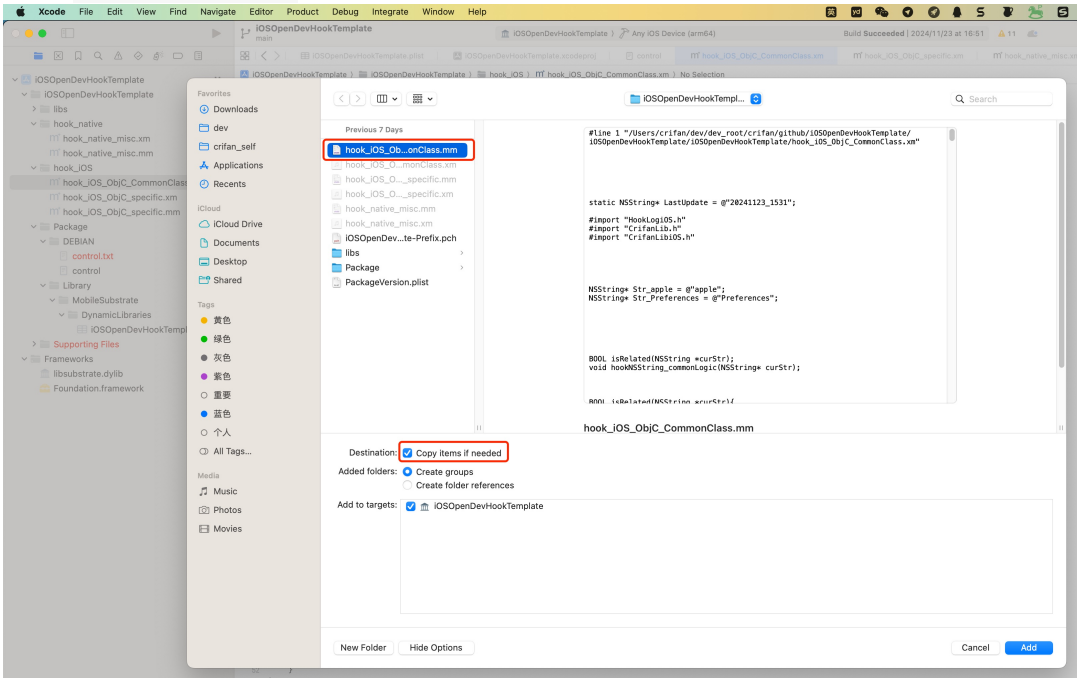

Xcode File Edit View Find Navigate Editor Product Debug
iOSOpenDevHookTemplat
main
iOSOpenDevHookTemplate
M
iOSOpenDevHookTemplate
libs
hook_native
hook_native_misc.xm
hook_native_misc.mm
hook_iOS
Show in Finder
Open in Tab
Open in New Window
Open with External Editor
Open As
Show File Inspector
New File...
Add Files to “iOSOpenDevHookTemplate“...
Add Package Dependencies...
Delete
New Group
New Group with Folder
New Group from Selection
Bookmark “hook_iOS_ObjC_CommonClass.xm” Line 384...
Bookmark “hook_iOS_ObjC_CommonClass.xm”...
Sort by Name
Sort by Type
Find in Selected Groups...
Source Control
Migrate to String Catalog...
Project Navigator Help
/*
File: hook_iOS_C
Function: hook i
Author: Crifan L
*/
static NSString* Las

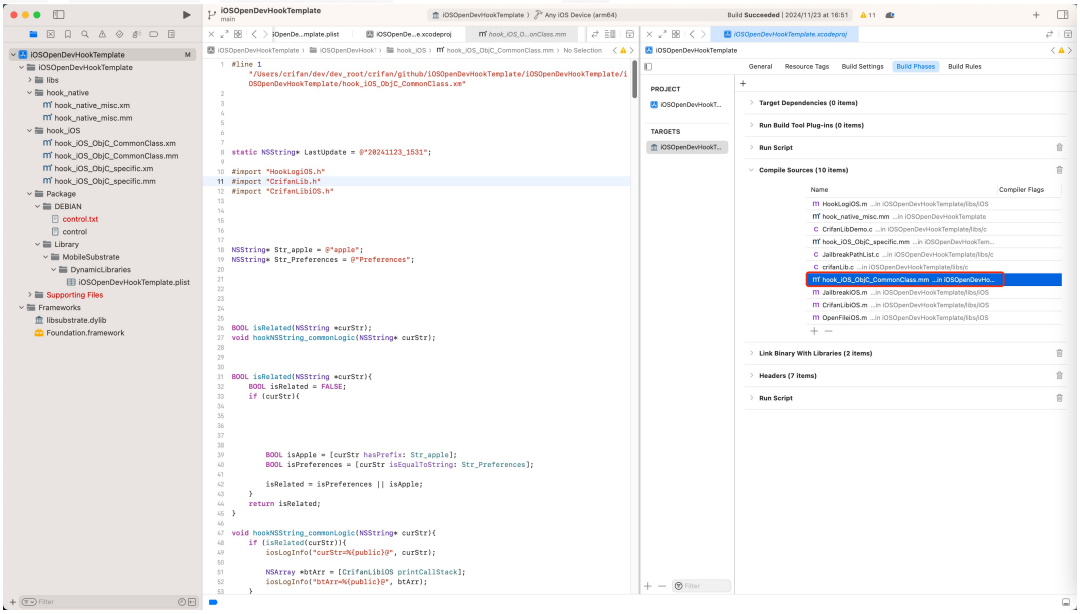




    - 此时项目列表中会新增出对应的 `.xm` 文件
- 编译 -> 会生成对应 `.mm` 文件
    - 【可选】如果遇到报错，就先Clean几次
        - `Xcode` -> `Product` -> `Clean Build Folder`
    - （然后再） `Xcode` -> `Product` -> `Build`
        - 会从 `yourFilename.xm` 生成 `yourFilename.mm`
            - 注：此时Xcode项目中是看不到的，但是文件系统中（比如通过Finder）是可以看到 `.mm` 文件的
- 把 `.mm` 文件加到 `Compile Sources` 中
    - 右键-> `Add Files to {yourProjectName}` -> 选择（刚新生成的） `yourFilename.mm`

- 

- 其中勾选： `Copy items if needed`
- 

- 项目文件列表中，即可新增对应文件 `yourFilename.mm`
- 项目的待编译的 `Compile Sources` 文件中，也包含了对应的 `.mm` 文件
  - `Targets -> Build Phase -> Compile Sources` 中有了刚加入的 `.mm` 文件
  - 
- 注
  - 如果有需要，也可以通过点击 `Compile Sources` 中的 `加号` = ，去新增导入进来（对应的或其他的 `.mm` 文件）
    - 这样后续编译代码时，才能真正编译到对应hook代码

写hook插件代码

新建 `iOSOpenDev` 的项目中的 `.xm` 文件（此处是 `iOSBypassJailbreak.xm` ）生成的默认代码，来自模板，一般是：

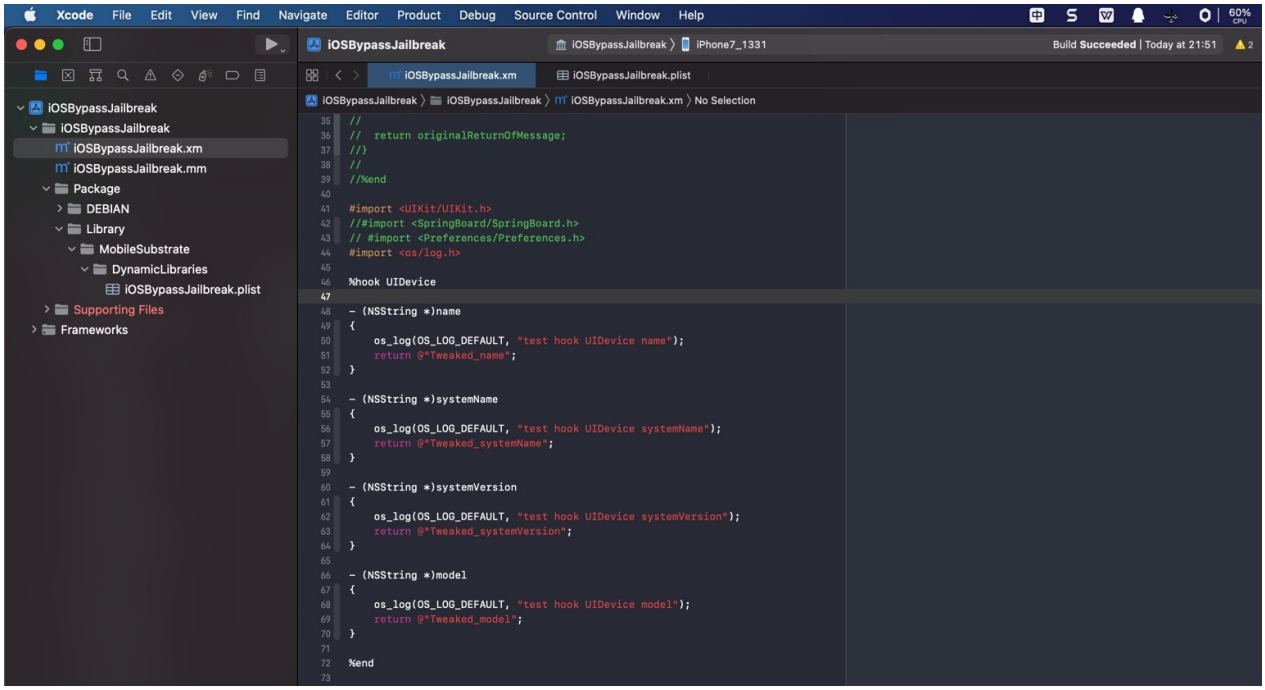
```
// Logos by Dustin Howett
// See http://iphonedevwiki.net/index.php/Logos

#error iOSOpenDev post project creation from template requirements (remove these lines after completed) -- \
   Link to libsubstrate.dylib: \
   (1) go to TARGETS > Build Phases > Link Binary With Libraries and add /opt/iOSOpenDev/lib/libsubstrate.dylib \
   (2) remove these lines from *.xm files (not *.mm files as they're automatically generated from *.xm files)

%hook ClassName

+ (id)sharedInstance
{
   %log;

   return %orig;
}

- (void)messageWithNoReturnAndOneArgument:(id)originalArgument
{
   %log;

   %orig(originalArgument);

   // or, for exmaple, you could use a custom value instead of the original argument: %orig(customValue);
}

- (id)messageWithReturnAndNoArguments
{
   %log;

   id originalReturnOfMessage = %orig;

   // for example, you could modify the original return value before returning it: [SomeOtherClass doSomethingToThisO
bject:originalReturnOfMessage];

   return originalReturnOfMessage;
}

%end
```

去删除掉，或注释掉，改为自己的hook的代码。

附录

demo示例代码

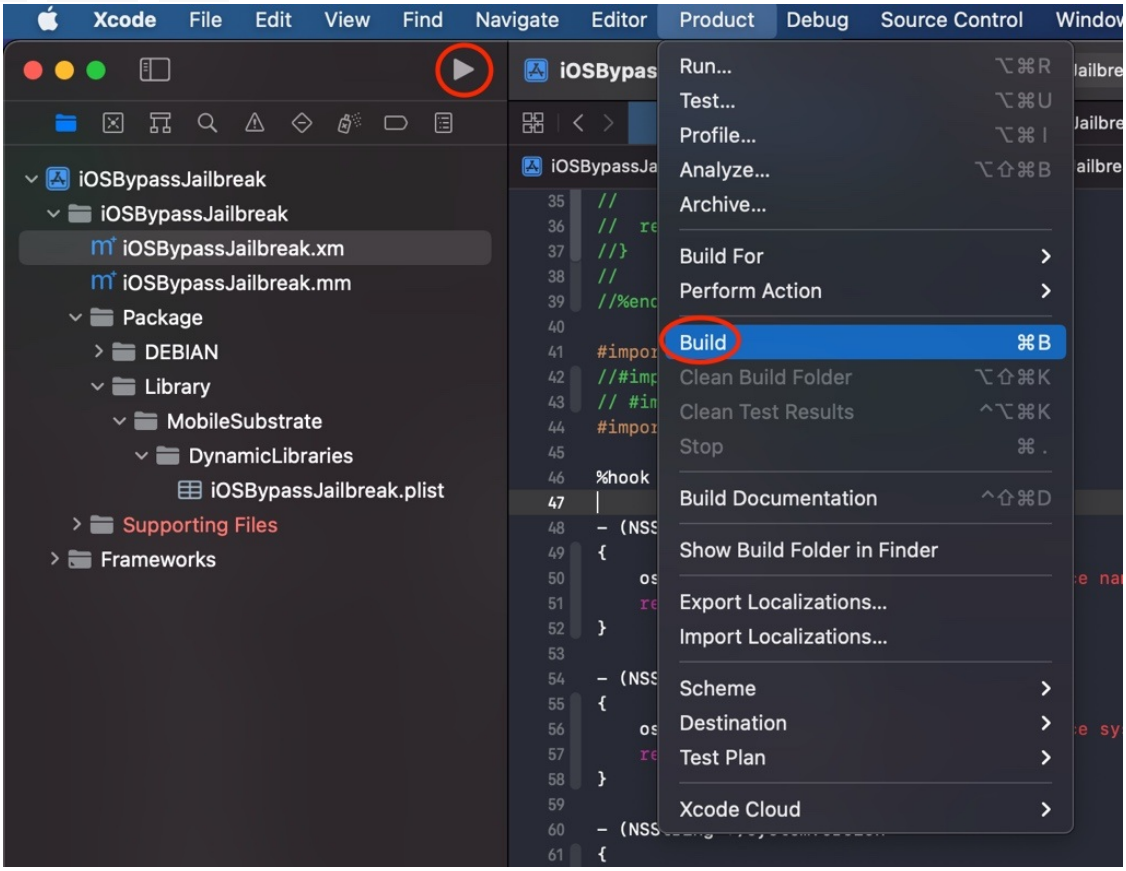
比如此处仅用于演示的代码：



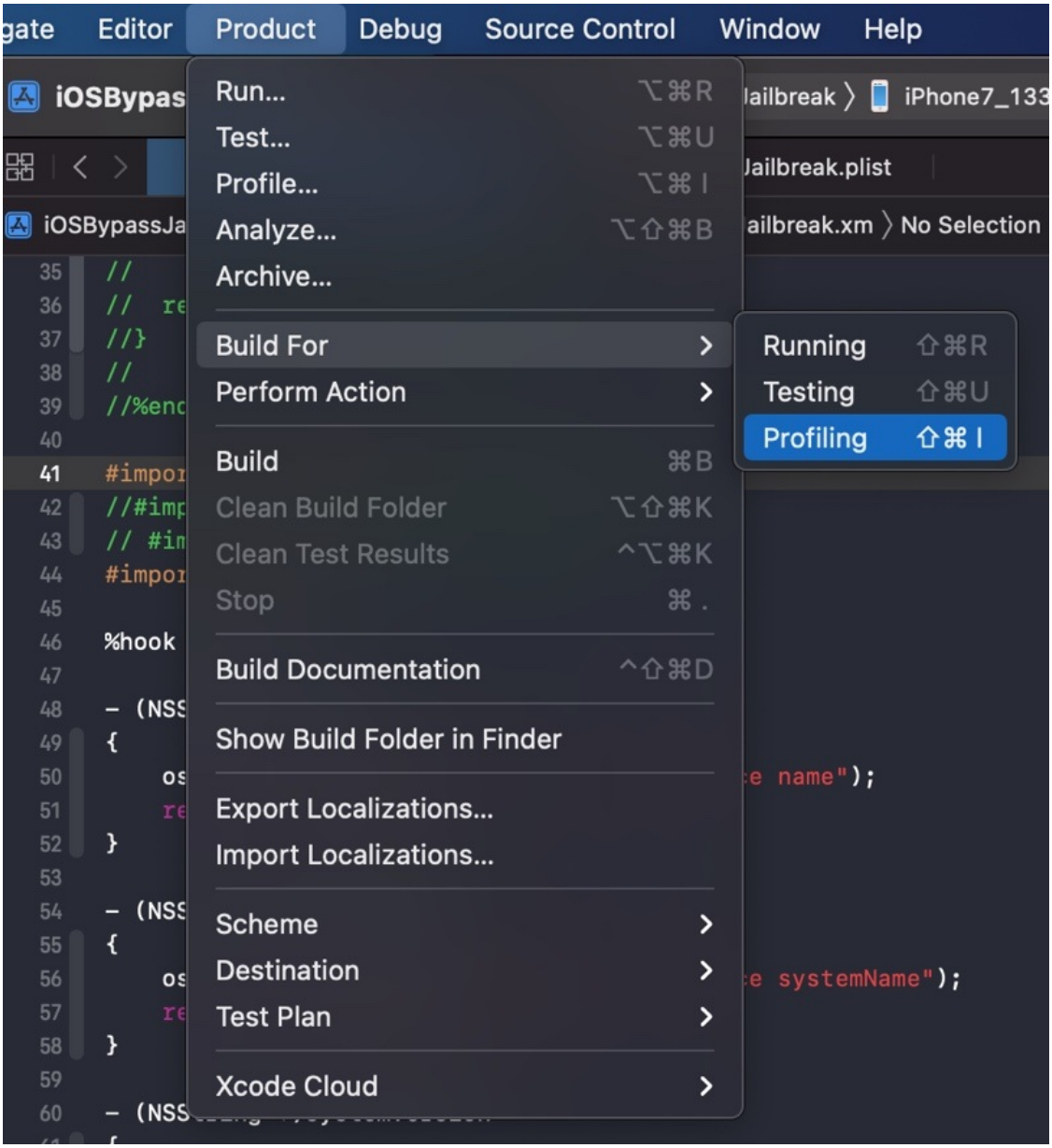

```
#import <UIKit/UIKit.h>
//#import <SpringBoard/SpringBoard.h>
// #import <Preferences/Preferences.h>
#import <os/log.h>

%hook UIDevice

- (NSString *)name
{
    os_log(OS_LOG_DEFAULT, "test hook UIDevice name");
    return @"Tweaked_name";
}

- (NSString *)systemName
{
    os_log(OS_LOG_DEFAULT, "test hook UIDevice systemName");
    return @"Tweaked_systemName";
}

- (NSString *)systemVersion
{
    os_log(OS_LOG_DEFAULT, "test hook UIDevice systemVersion");
    return @"Tweaked_systemVersion";
}

- (NSString *)model
{
    os_log(OS_LOG_DEFAULT, "test hook UIDevice model");
    return @"Tweaked_model";
}

%end
```

调试插件代码

接下来就是典型的插件开发流程了：

- **写代码** = 写hook代码= 写tweak插件代码 = 改动 `.xm` 文件
- **编译**代码 -》 确保语法没错，可以正常编译
  - `Xcode` -> `Product` -> `Build`
    - 

- **调试和运行** -》 把hook插件代码编译所生成的插件(`.deb` 文件)安装到iOS设备(iPhone)中，测试插件效果
  - `Product` -> `Build For` -> `Profiling`

- 

确认插件安装成功

- iPhone中看到自己的插件
  - `Cydia` -> `已安装` -> `最近` 能看到自己的插件：

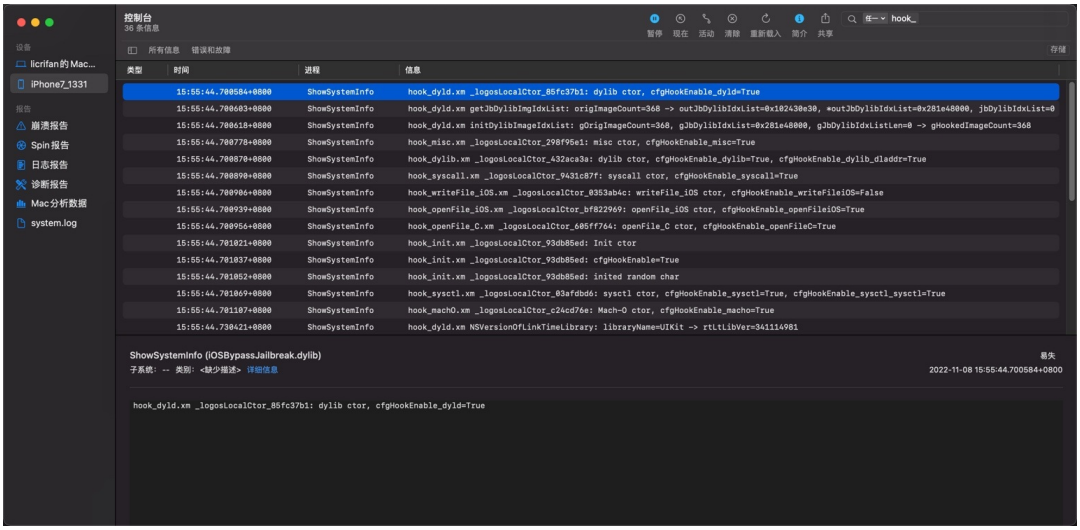
- 点击插件，可以看到插件基本信息

- 点击插件的文件，可以看到文件列表

确认插件的确正常工作

- 打开被测试的=被hook的app，看到此处测试代码：更改信息信息，显示是我们hook代码中的值，表示hook成功

- 查看对应log日志
  - `Xcode` -> `Window` -> `Devices and Simulators` -> `Devices` ->选中 `Connected` 中自己的iPhone设备-> `Open Console` ->打开 `Console` = `控制台` ，显示出对应iPhone的log日志
    - 其中就有你的插件的log日志
      - 如果没有，则自己去右上角，搜索对应关键字，即可搜到
    - 此处贴出，后续更新了代码后的相关log



带界面的插件开发流程

TODO：

- 【未解决】用XCode开发iOS的app用于配置改机软件参数
- 【已解决】越狱iPhone中安装deb包后iOS的反越狱插件没生效
- 【已解决】把iOS的app和iOS的tweak插件打包成独立的deb安装包
- 【记录】越狱iPhone中安装iOS的app和tweak合并出的deb安装包
- 【已解决】越狱iOS如何用Theos开发带GUI图形界面的插件
- 【记录】重新给iOSOpenDev的tweak加app打包deb看看app是否有权限写入Preferences目录
- 【已解决】iOS的writeToURL报错：NSCocoaErrorDomain Code 513 You don’t have permission to save the file in the folder Preferences
- 【已解决】iOSOpenDev的tweak中读取app保存出的配置文件参数
- 【已解决】iOSOpenDev的XCode编译报错：An empty identity is not valid when signing a binary for the product type Application
- 【已解决】对比研究FakeWeChatLoc和自己的XCode项目的目录结构区别
- 【已解决】给iOS的XCode项目中新增iOSOpenDev的Project Navigator的目录和文件

前提和目标

- 前提
  - 已实现：基于XCode用iOSOpenDev的Logos Tweak去创建出tweak插件
  - 已实现：用XCode开发出普通的带UI界面的iOS的app
    - 比如：实现了机型选择的功能
- 目标
  - 和tweak和app合并成单个deb安装包=单个tweak插件（安装出来后，带UI界面的tweak插件）

基本流程

TODO：

- 【已解决】XCode 13.1中Build Settings中Code Signing Identity没有Don't Sign Code选项
- 【已解决】XCode中编译设置参数中如何新增User-Defined自定义参数
- 【已解决】XCode项目中新增iOSOpenDev的Package目录到Target目录中

把Xcode中iOS的app项目转换成iOSOpenDev的项目

把普通的XCode项目，去改造成iOSOpenDev的项目：

核心步骤：

只需参考官网教程：

Convert to iOSOpenDev Project · kokoabim/iOSOpenDev Wiki (github.com)

只是有几个细节，需要更新和补充：

最新XCode（13.1）中：Code Signing Identity没有Don't Sign Code选项

解决办法：给 `PROJECT` -> `Build Settings` -> `User-Defined` ->增加参数：

- `CODE_SIGNING_ALLOWED=NO`

详见：

- 【已解决】XCode 13.1中Build Settings中Code Signing Identity没有Don't Sign Code选项

新版XCode中找不到新建User-Defined参数的入口

解决办法：选择 `PROJECT` （和 或 `TARGETS` 中的某个 `target` ）-> `Build Settings` -》最顶部（和 `Basic` 、 `Customized` 、 `All` 、 `Combined` 所在的同）一行的最右边有个加号 -》 `Add User-Defined Setting`

详见：

- 【已解决】XCode中编译设置参数中如何新增User-Defined自定义参数

给Target的目录中新建Package

要点：选中自己项目的 `Target` 目录-》右键-》 `New Group`

即可新建组=Group=子目录

详见：

- 【已解决】XCode项目中新增iOSOpenDev的Package目录到Target目录中

app和tweak之间的通信

对于iOSOpenDev的app，想要和tweak插件之间通信，主要是互相共享配置参数，此处是通过：**配置文件**

具体做法是：

app端

`MuJiaBaiHuoApp的ViewController.m`

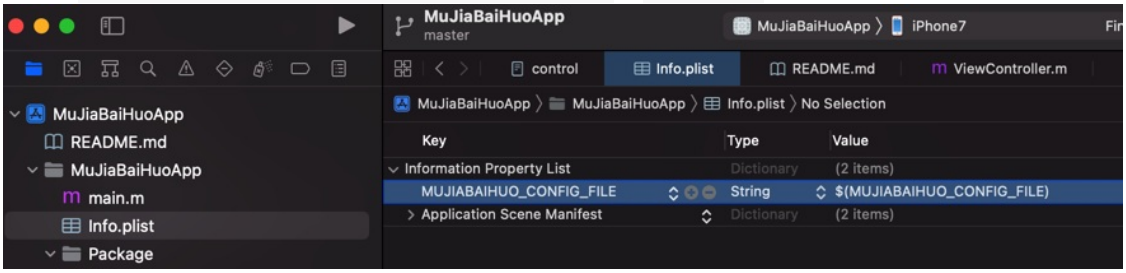
写入配置：

```
- (void) saveConfig:(NSDictionary *) curCfgDict {
    NSLog (@"curCfgDict=%@", curCfgDict);
    NSString *curCfgFile = [[NSBundle mainBundle] objectForInfoDictionaryKey:@"MUJIABAIHUO_CONFIG_FILE"];
    NSLog (@"curCfgFile=%@", curCfgFile);
    NSString *curCfgFileUrl = [NSString stringWithFormat:@"file://%@", curCfgFile];
    NSLog (@"curCfgFileUrl=%@", curCfgFileUrl);
    NSURL *curCfgUrl = [NSURL URLWithString:curCfgFileUrl];
    NSLog (@"curCfgUrl=%@", curCfgUrl);
    NSError *writeErr = NULL;
    BOOL isWriteOk = [curCfgDict writeToURL:curCfgUrl error:&writeErr];
    NSLog(@"isWriteOk=%d", isWriteOk);
    if (isWriteOk == NO){
        NSLog(@"writeErr=%@", writeErr);
    }
}
```

到对应的配置文件：

`/var/mobile/Library/Preferences/MuJiaBaiHuo.plist`

注：

- `MUJIABAIHUO_CONFIG_FILE` 是加的 `User-Defined` 的参数
  -
  - 且：同时加 `MUJIABAIHUO_CONFIG_FILE=$(MUJIABAIHUO_CONFIG_FILE)` 到 `info.plist` ，才生效
    - 

tweak端

`MuJiaBaiHuoTweak的MuJiaBaiHuoTweak.xm`

```
const NSString *CONFIG_FILE = @"/var/mobile/Library/Preferences/MuJiaBaiHuo.plist";

- (NSString *)model
{
    os_log(OS_LOG_DEFAULT, "MuJiaBaiHuoTweak hook UIDevice model");
    NSString *hookedModel = @"Tweaked_model";
    NSDictionary *curCfgDict = [[NSDictionary alloc] initWithContentsOfFile:(NSString *)CONFIG_FILE];
    os_log(OS_LOG_DEFAULT, "MuJiaBaiHuoTweak curCfgDict=%{public}@", curCfgDict);
    if (curCfgDict) {
        NSString *phoneIdStr = [curCfgDict objectForKey:@"phoneId"];
        os_log(OS_LOG_DEFAULT, "MuJiaBaiHuoTweak phoneIdStr=%{public}@", phoneIdStr);
        hookedModel = phoneIdStr;
        os_log(OS_LOG_DEFAULT, "MuJiaBaiHuoTweak hookedModel=%{public}@", hookedModel);
```

```
    }

    os_log(OS_LOG_DEFAULT, "MuJiaBaiHuoTweak return hookedModel=%{public}@", hookedModel);
    return hookedModel;
}
```

即可：

从配置文件中

`/var/mobile/Library/Preferences/MuJiaBaiHuo.plist`

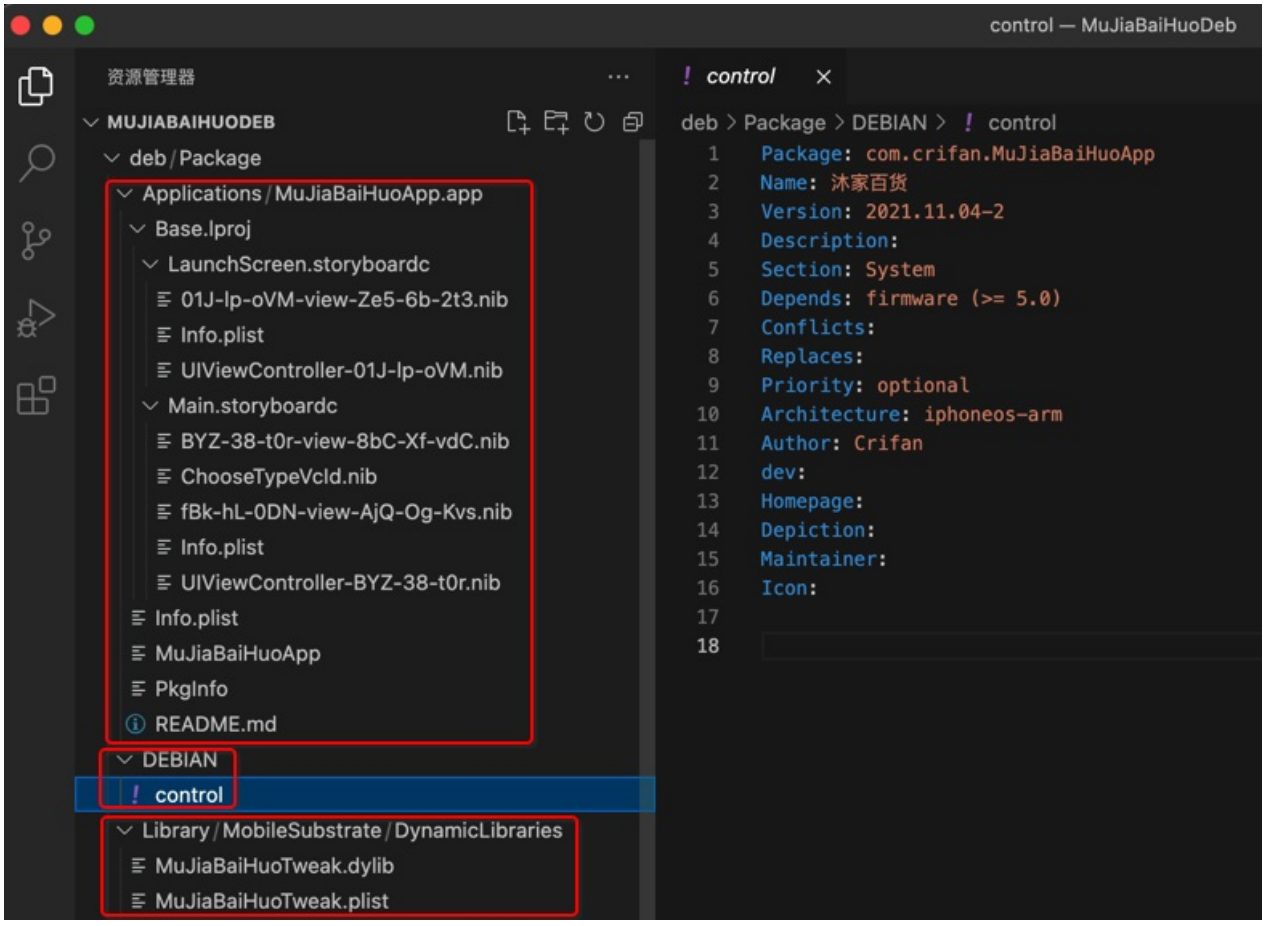
读取出之前保存的 `NSDictionary`，获取到参数 `phoneId` 的值。

把tweak和app合并成单个deb文件

此处把：

- **tweak**：的Package包中的Library文件夹
  - 去掉DEBIAN目录
    - 中的control
      - 无需tweak的control
- **app**：的Package的目录中的所有内容
  - Applications
  - DEBIAN
    - control
      - 需要app的control文件

合并到一起后的效果：



对应的目录结构：

```
➜  MuJiaBaiHuoDeb tree deb/Package
deb/Package
├── Applications
│   └── MuJiaBaiHuoApp.app
│       ├── Base.lproj
│       │   ├── LaunchScreen.storyboardc
│       │   │   ├── 01J-lp-oVM-view-Ze5-6b-2t3.nib
│       │   │   ├── Info.plist
│       │   │   └── UIViewController-01J-lp-oVM.nib
│       │   └── Main.storyboardc
│       │       ├── BYZ-38-t0r-view-8bC-Xf-vdC.nib
│       │       ├── ChooseTypeVcId.nib
│       │       ├── Info.plist
│       │       ├── UIViewController-BYZ-38-t0r.nib
│       │       └── fBk-hL-0DN-view-AjQ-Og-Kvs.nib
│       ├── Info.plist
│       ├── MuJiaBaiHuoApp
│       ├── PkgInfo
│       └── README.md
├── DEBIAN
│   └── control
└── Library
    └── MobileSubstrate
        └── DynamicLibraries
            ├── MuJiaBaiHuoTweak.dylib
            └── MuJiaBaiHuoTweak.plist

9 directories, 15 files
```

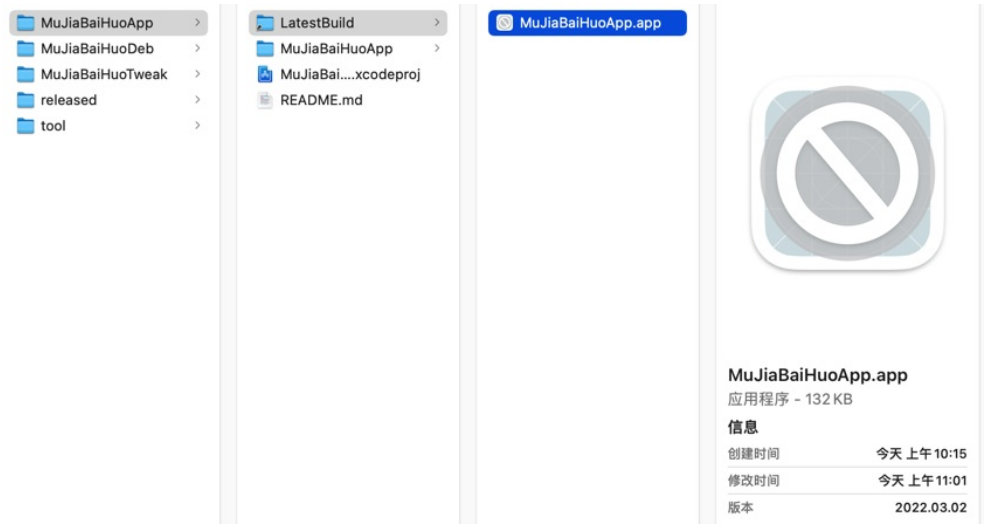
Finder中的效果：

- Applications
  -
- DEBIAN
  -
- Library
  -

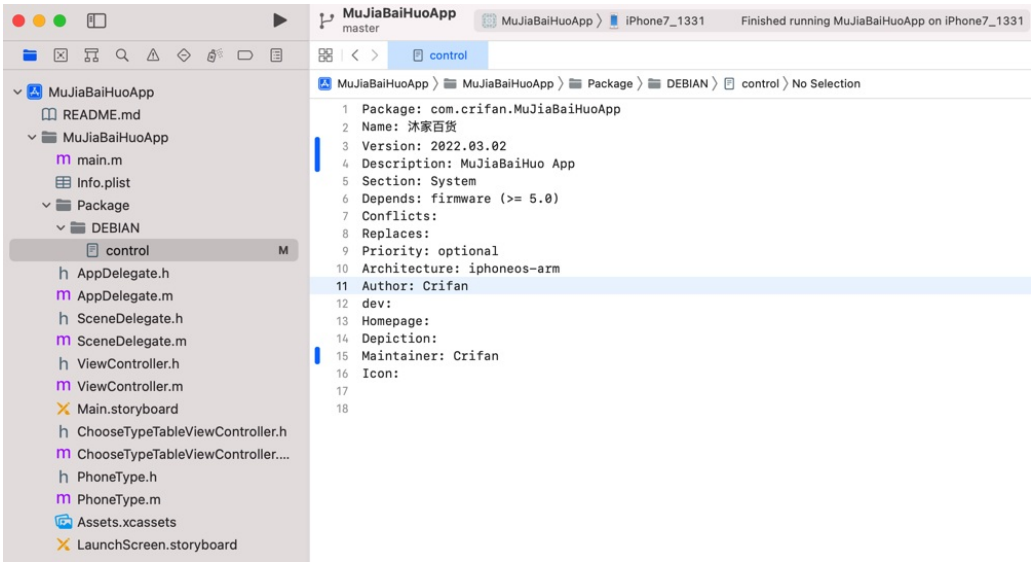
关于目录中的各个文件的详细解释：

- `Package`：要打包的根目录
  - `Applications`

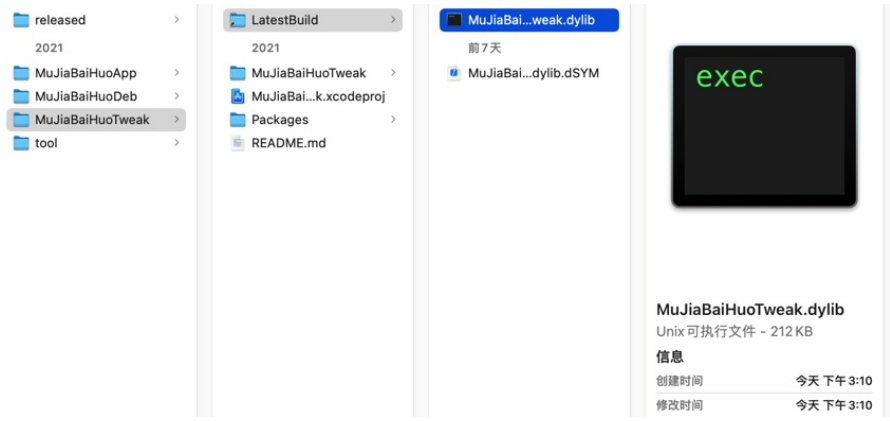
- `MuJiaBaiHuoApp.app`
  - 拷贝自：iOS的**app**的XCode项目下的LatestBuild
    - LatestBuild中保存了每次最新编译之后的版本



- `DEBIAN`
  - `control`
    - 用的是：iOS的**app**的Package/DEBIAN/control，不是iOS的tweak的Package/DEBIAN/control



- `Library`
  - `MobileSubstrate`
    - `DynamicLibraries`
      - `MuJiaBaiHuoTweak.dylib`
        - 拷贝自：iOS的**tweak**的XCode项目下的LatestBuild
          - LatestBuild中保存了每次最新编译之后的版本



- `MuJiaBaiHuoTweak.plist`
  - 拷贝自iOS的tweak中的 `Library/MobileSubstrate/DynamicLibraries`

然后写个shell脚本：

- `buildPackage.sh`

```
#!/bin/bash
find ./Package -name ".DS_Store" -depth -exec rm {} \;
dpkg-deb -Zgzip -b Package MuJiaBaiHuo.deb
```

- 注：其实核心就一句：`dpkg-deb -Zgzip -b Package MuJiaBaiHuo.deb`

去加上可执行权限：

```
chmod +x ./buildPackage.sh
```

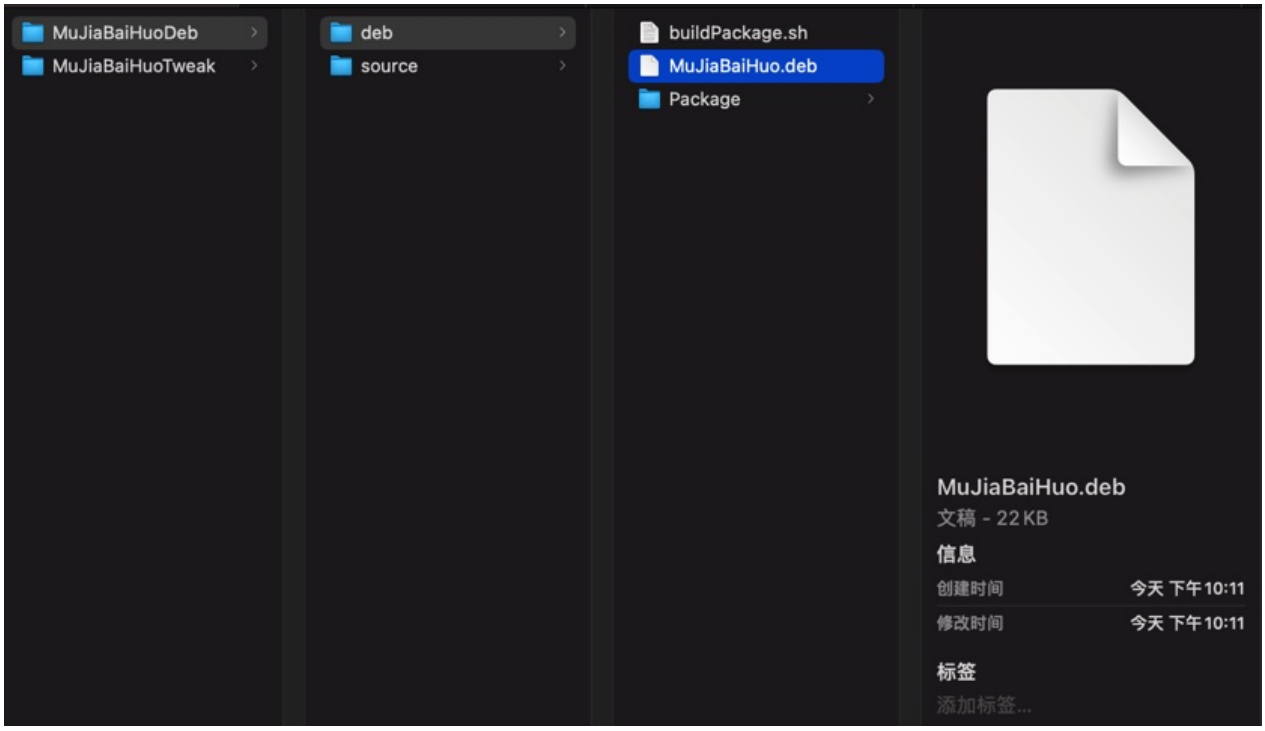
运行脚本去打包出deb

```
./buildPackage.sh
```

- log输出举例

```
➜  20220302 pwd
/Users/crifan/dev/DevRoot/zry/MuJiaBaiHuo/MuJiaBaiHuoDeb/20220302
➜  20220302 ./buildPackage.sh
dpkg-deb: 正在 'MuJiaBaiHuo.deb' 申构建软件包 'com.crifan.mujiabaihuoapp'。
```

即可得到deb插件安装包：`MuJiaBaiHuo.deb`



注：

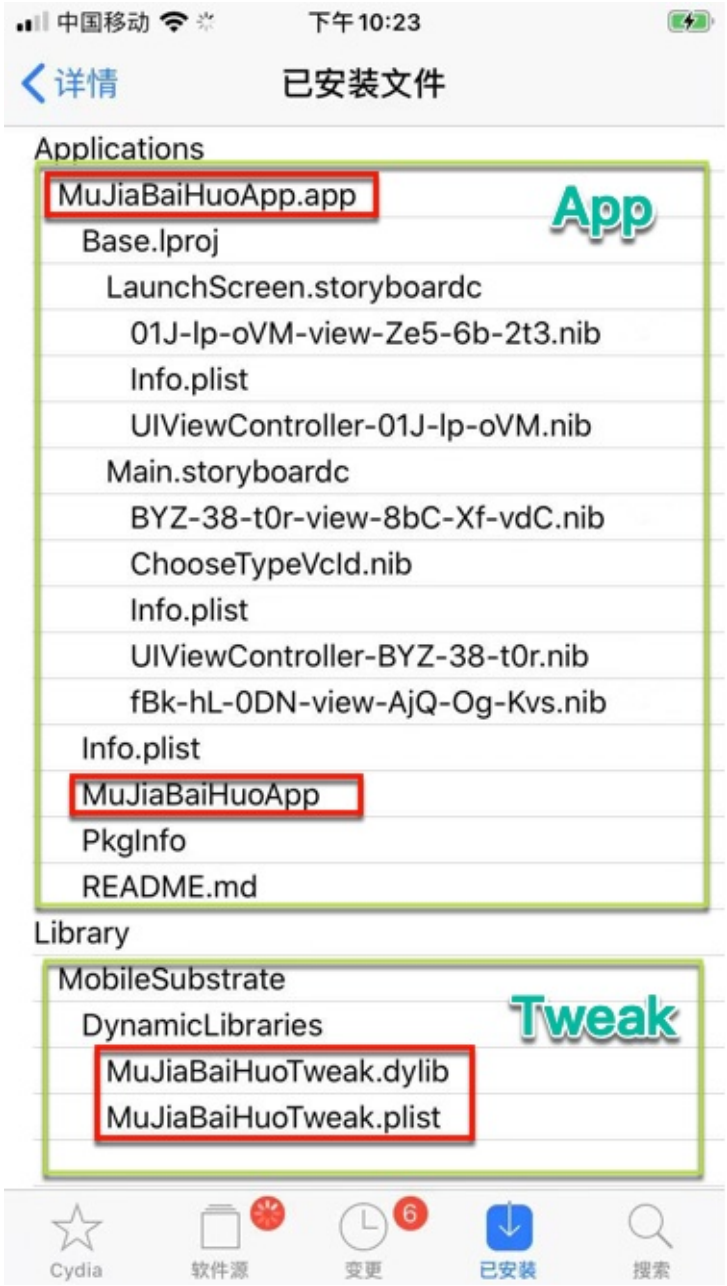
- 如何安装
  - 通过`Filza`或命令行`dpkg`（命令是`dpkg -i filename.deb`）去安装deb文件，即可安装到iPhone中。

使用效果

插件安装后的效果

Cydia中可以看到安装后的插件：

其中详情中，点击 `文件系统内容` ，可以看到包含的详细内容：



其中就有 `Applications` 和 `Library`

插件的使用效果

iOS的app部分：

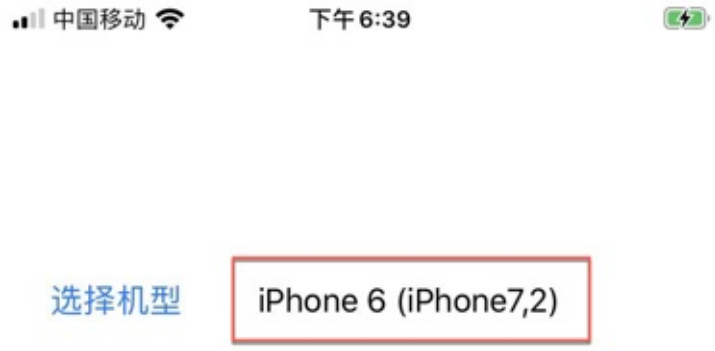
桌面上的Logo=iOS的app的图标：


中国移动
下午11:14
文件
查找
快捷指令
iTunes Store
提示
通讯录
Watch
实用工具
音乐
沐家百货
1

点击启动后，显示主界面：

点击 选择机型，出现：机型列表

去选择一个：`iPhone 6(iPhone7,2)`



然后回去打开，被hook的普通的iOS的app：`ShowSysInfo`

可以检测当前机型，是我们所hook选择的机型：

-》说明上述的：tweak插件，iOS的app（用于配合插件做配置），是生效的。

注：

对应保存到了配置文件中的内容是：

```
phoneId=iPhone7,2
```

-》至此，跑通了：

- 带UI界面的tweak插件=单个deb文件，集成了包含了
  - app
    - 实现用户UI界面
    - 选择配置，保存配置到配置文件 `/var/mobile/Library/Preferences/MuJiaBaiHuo.plist`
  - tweak
    - 实现hook功能
    - 返回的值，根据配置文件中保存的值决定

常见问题

安装器遇到了一个错误，导致安装失败

安装到最后，报错：`安装失败 安装器遇到了一个错误，导致安装失败`

解决办法：

其实此时 `iOSOpenDev` 的主体文件已安装到了默认的位置 `/opt` 中，接着去用工具初始化即可解决问题：

```
cd /opt/iOSOpenDevSetup/bin
sudo ./iod-setup base
sudo ./iod-setup sdk -sdk iphoneos
```

PrivateFramework directory not found XCode iPhoneOS15.0.sdk

`iod-setup sdk -sdk iphoneos` 时报错：

```
➜  bin sudo ./iod-setup sdk -sdk iphoneos
Setting up iPhoneOS 15.0 SDK...
Modifying SDK settings...
Symlinking to private frameworks header files...
PrivateFramework directory not found: /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS15.0.sdk/System/Library/PrivateFrameworks
```

原因：

此处是比较新的 `XCode 13` 和对应的 `iOS 15`

->而最新版XCode和iOS早已将私有库PrivateFrameworks移走了

->即 `iPhoneOSxx.xx.sdk/System/Library/` 下面没有 `PrivateFrameworks` 了

解决办法：

- 自己之后是否用到私有库PrivateFrameworks
  - 否
    - 直接新建一个空目录即可

      ```
      cd /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS15.0.sdk/System/Library
      sudo mkdir PrivateFrameworks
      ```

  - 是
    - 除了新建目录外，还要把相关iPhoneOS版本的私有库的内容放过去
      - 先要找到相关iPhoneOS的PrivateFrameworks
        - 举例
          - `iPhoneOS 9.2` 的 `sdk`，可以从这里下载到：
            - zhangkn/knPrivateFrameworks: /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS9.2.sdk/System/Library/PrivateFrameworks (github.com)

File not found XCode Specifications iPhoneOSPackageTypes.xcspec

`iod-setup sdk -sdk iphoneos` 报错：

```
➜  bin sudo ./iod-setup sdk -sdk iphoneos
Password:
Setting up iPhoneOS 15.0 SDK...
Modifying SDK settings...
Symlinking to private frameworks header files...
Adding specifications to platform...
File not found: /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/Library/Xcode/Specif
ications/iPhoneOSPackageTypes.xcspec
```

原因：

找不到specifications

解决办法：

下载别人给的：

- 4个iPhoneOS的spec文件
- 4个iPhoneSimulator的spec文件

分别放到对应位置，即可。

下载来源：

- 来源1：
  - iosopendev专用Specifications.zip
- 来源2：
  - 越狱开发:用iosOpenDev配置越狱开发环境 编写第一个hello world_我的杯洗具的博客-CSDN博客

下载后，可以看到Specifications中有8个spec。

分别新建Specifications目录：

```
sudo mkdir /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/Library/Xcode/Specificati
ons
sudo mkdir /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/Library/Xcode/Spec
ifications
```

再去

- 移动文件
  - 把
    - 4个 `iPhoneOS` 的文件
      - iPhoneOSPackageTypes.xcspec
      - iPhoneOSPackageTypes.xcspec.iOSOpenDev
      - iPhoneOSProductTypes.xcspec
      - iPhoneOSProductTypes.xcspec.iOSOpenDev
    - 放到：
      - /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/Library/Xcode/Specifications
  - 把：
    - 4个 `iPhoneSimulator` 的文件
      - iPhone Simulator PackageTypes.xcspec
      - iPhone Simulator PackageTypes.xcspec.iOSOpenDev

        - iPhone Simulator ProductTypes.xcspec
        - iPhone Simulator ProductTypes.xcspec.iOSOpenDev
    - 放到：
        - /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/Library/Xcode/Specifications

放好后是：

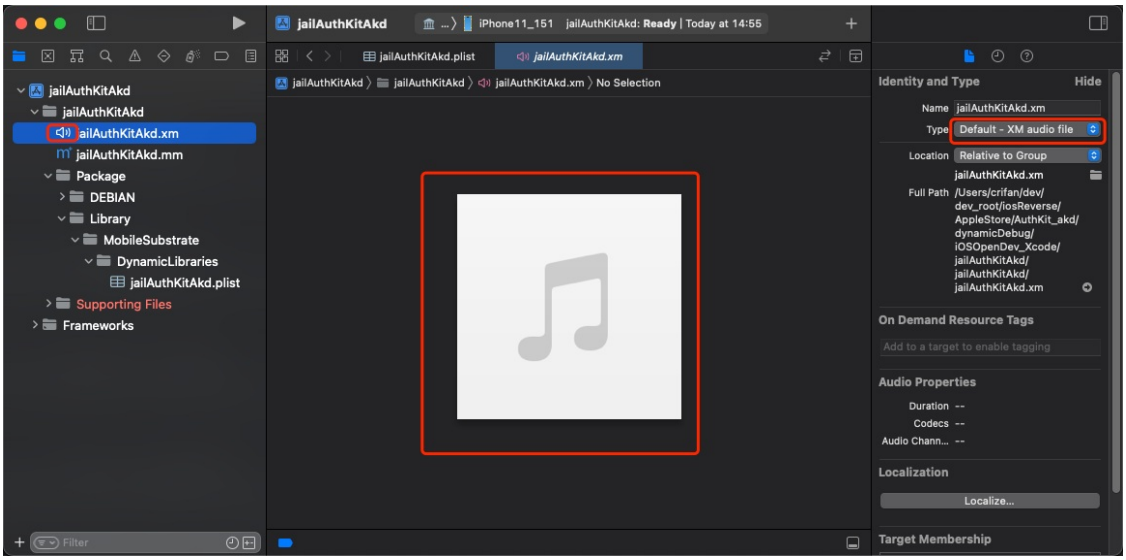
```
➜  Xcode ll /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/Library/Xcode/Specifications
total 48
-rwxr-xr-x@ 1 crifan  wheel   3.2K 12 24  2015 iPhoneOSPackageTypes.xcspec
-rwxr-xr-x@ 1 crifan  wheel   5.4K 12 24  2015 iPhoneOSPackageTypes.xcspec.iOSOpenDev
-rwxr-xr-x@ 1 crifan  wheel   4.0K 12 24  2015 iPhoneOSProductTypes.xcspec
-rwxr-xr-x@ 1 crifan  wheel   6.4K 12 24  2015 iPhoneOSProductTypes.xcspec.iOSOpenDev
➜  Xcode ll /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/Library/Xcode/Specifications
total 48
-rwxr-xr-x@ 1 crifan  wheel   3.4K 12 24  2015 iPhone Simulator PackageTypes.xcspec
-rwxr-xr-x@ 1 crifan  wheel   6.9K 12 24  2015 iPhone Simulator PackageTypes.xcspec.iOSOpenDev
-rwxr-xr-x@ 1 crifan  wheel   3.4K 12 24  2015 iPhone Simulator ProductTypes.xcspec
-rwxr-xr-x@ 1 crifan  wheel   6.1K 12 24  2015 iPhone Simulator ProductTypes.xcspec.iOSOpenDev
```

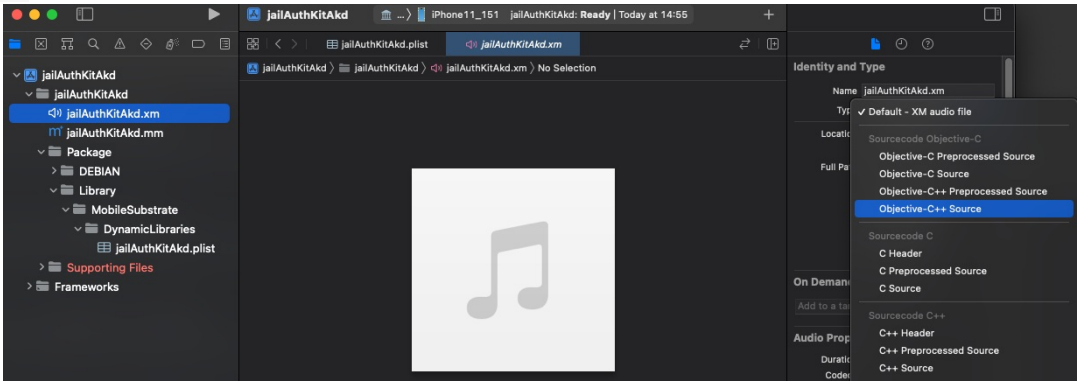
另外，新建usr的bin目录：

```
sudo mkdir /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/usr/bin
```

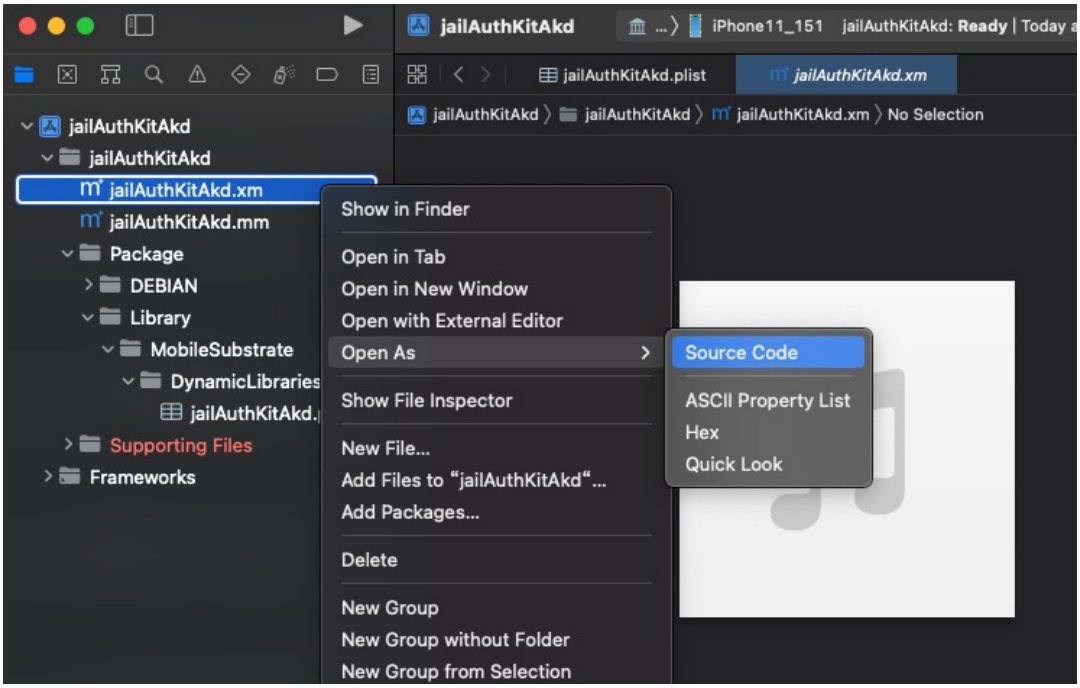
即可。

`.xm` 被识别为Audio音频文件

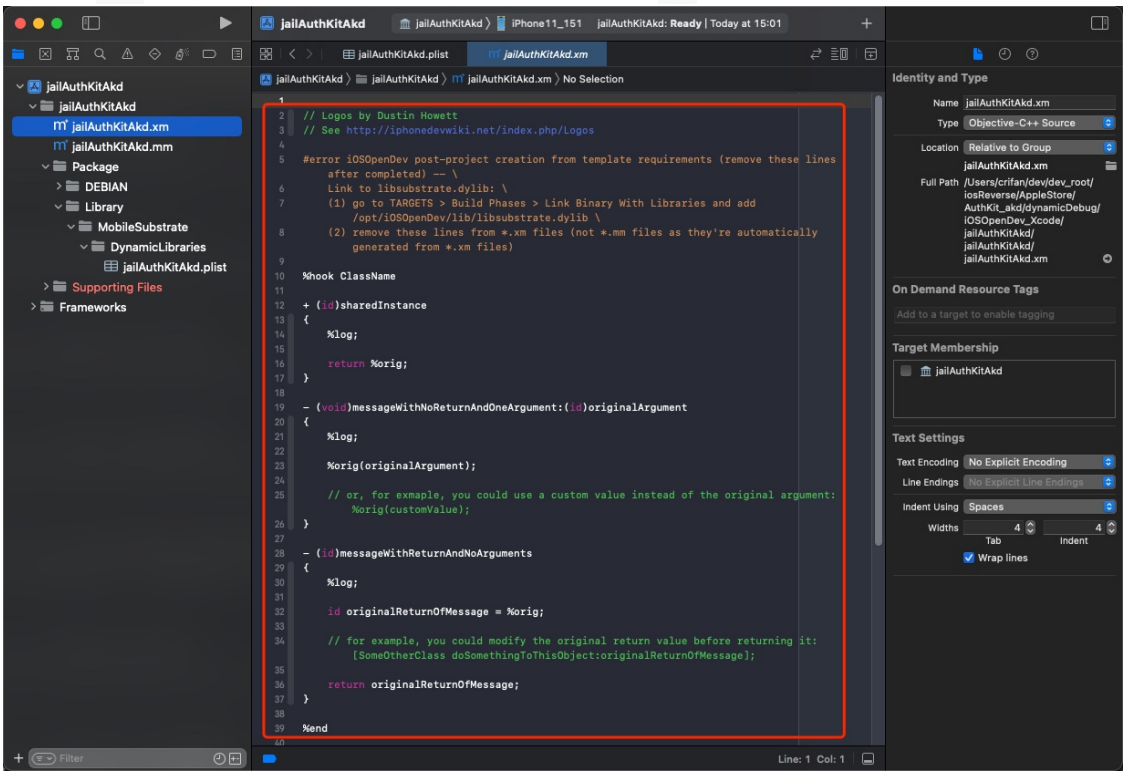
- 问题：`iOSOpenDev` 的 `Xcode` 中，默认的 `.xm` 被识别成**音频**文件，无法显示对应的源代码
  - 具体现象
    - xm的文件的图标是 `小喇叭`
    - 且右边显示的是：音乐的图标
    - 右边文件类型Type显示是：`Default - XM audio file`
  - 图
    - 

- 解决办法：
  - 先去：改变.xm的文件类型
    - Xcode右边的文件属性-> `Type` ，从 `Default - XM audio file` 改为 `Objective-C++ Source` （或 `Objective-C Source` ）
    - 

  - 再去改变 `.xm` 的文件的打开方式
    - Xcode左边文件列表->右键 `.xm` 文件-> `Open As` -> `Source Code`



- 即可正常显示 `.xm` 为ObjC的代码，并且带语法高亮了，且文件图标是 `.m` 的图标



- `.xm` 文件，默认会被 `Xcode` 识别为 `音频文件`

Expected unqualified-id

此错误，有**2**种 `情况` = `可能性` ：

- 无法识别的特殊字符
- `Compile Sources` 不支持.xm作为源码去编译

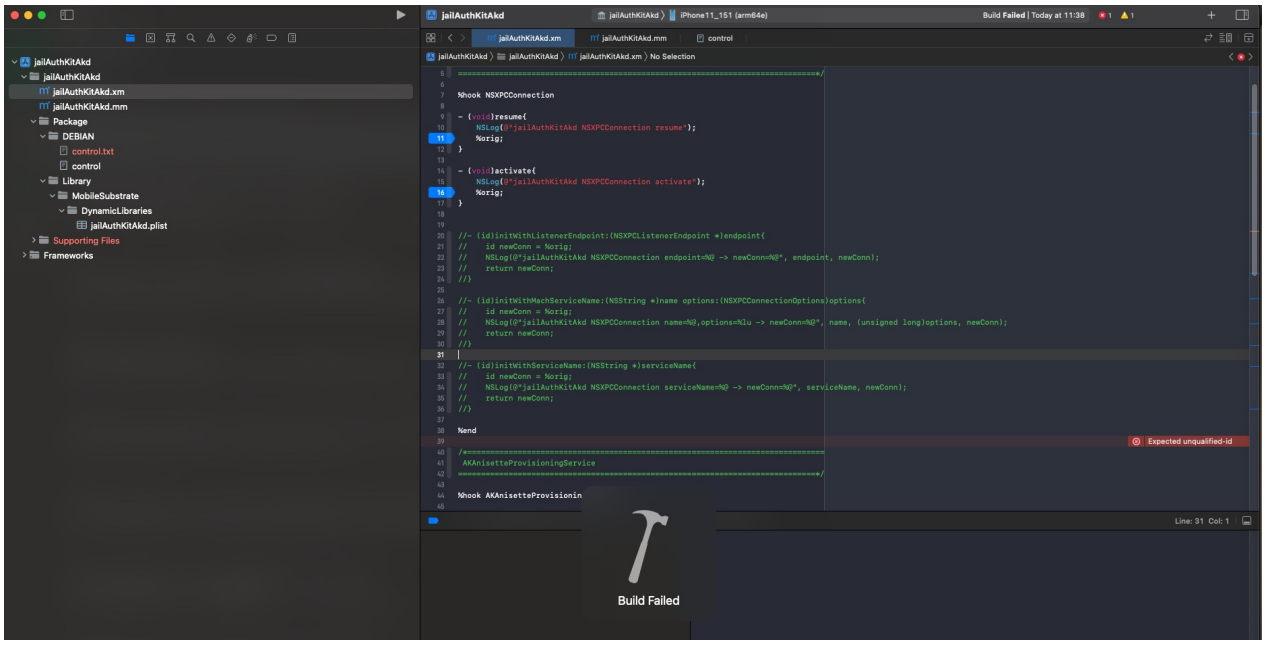
下面分别详细解释：

无法识别的特殊字符

现象

iOSOpenDev的 `.xm` 文件编译报错：

```
Expected unqualified-id
```

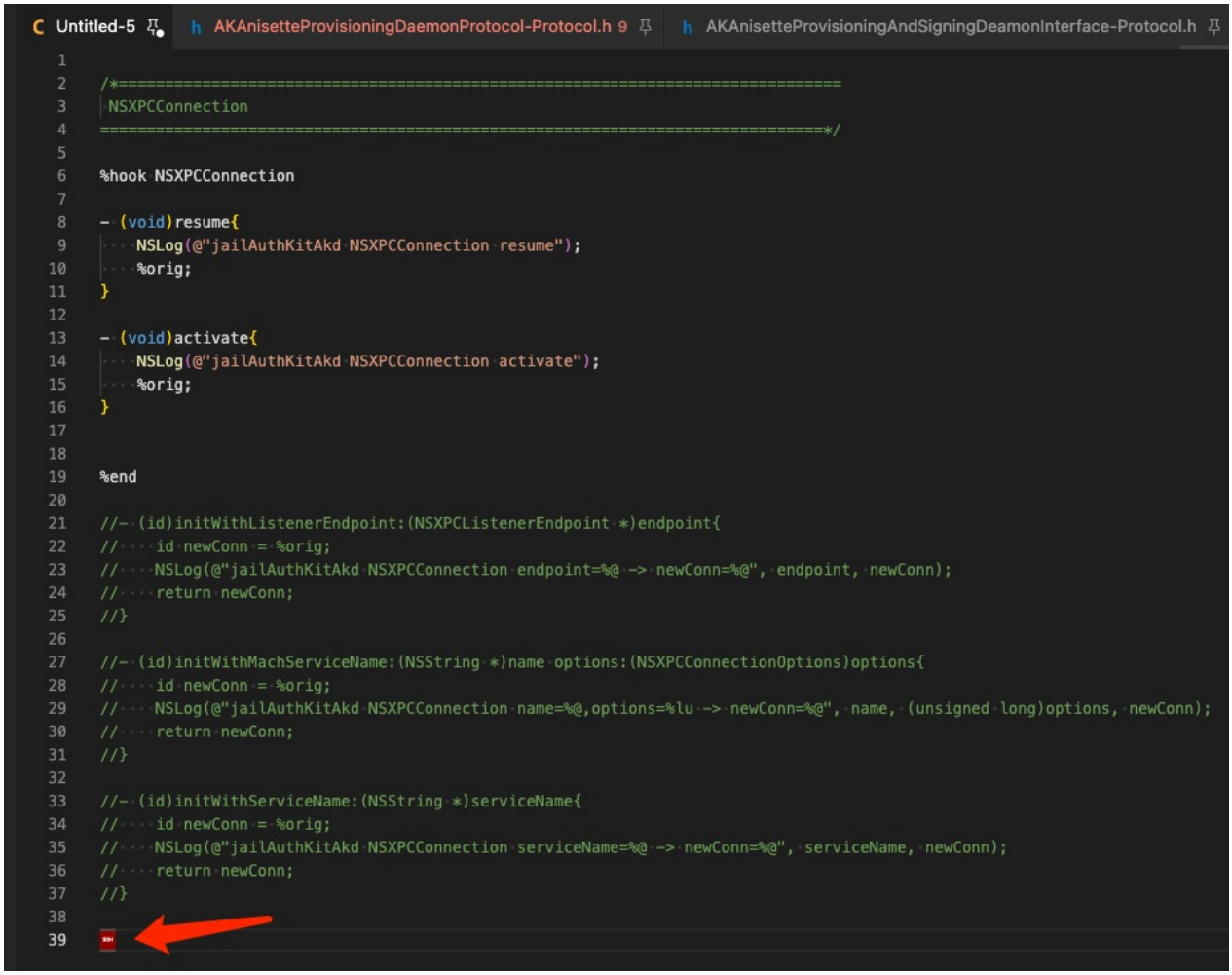


原因

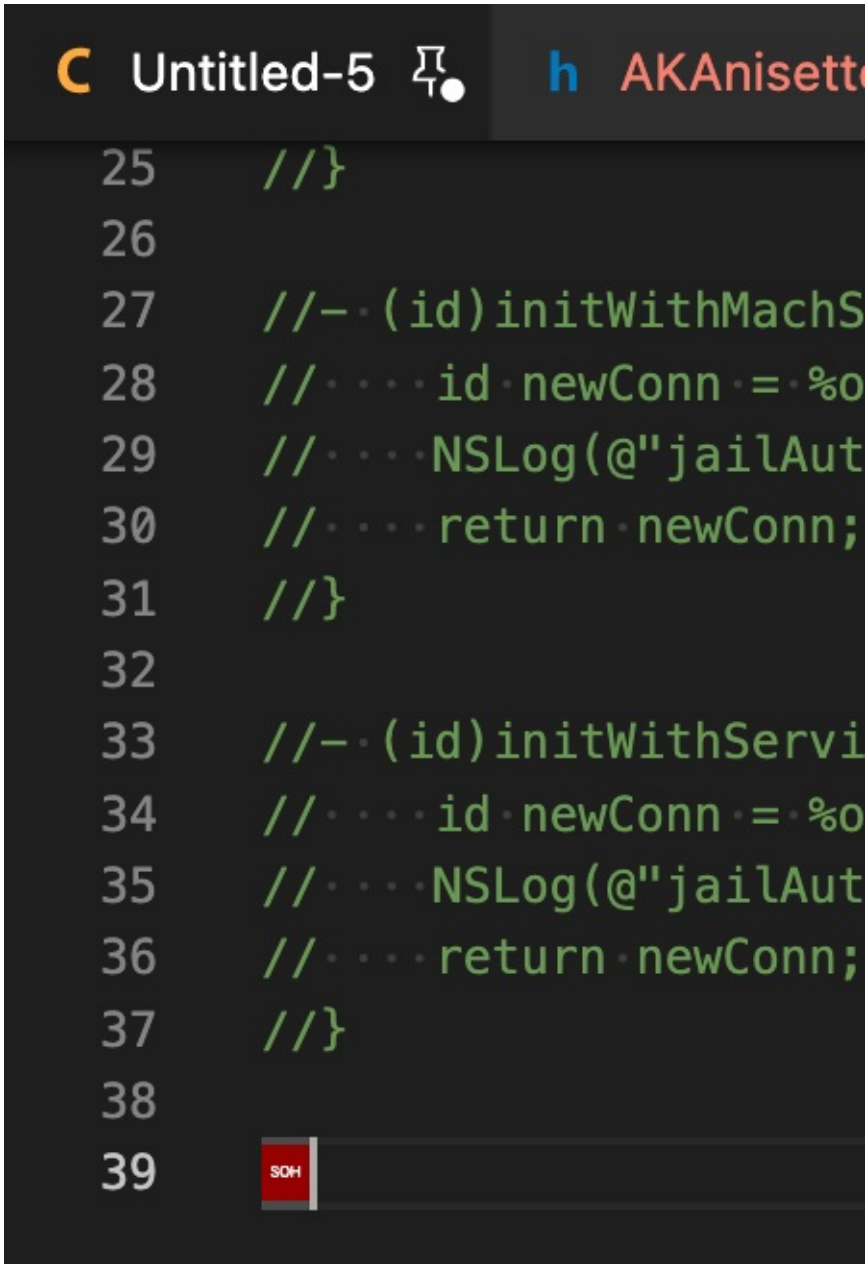
此处（由于之前某些操作，比如从别处复制粘贴过来一些内容，而导致的） `.xm` 源码中，有，此处不支持的，特殊的，控制字符

此处Xcode源码编译器中：无法看到此（不可见的）特殊字符

可以换用VSCode去打开查看到，特殊的，控制字符：



放大显示效果，可以看到此处特殊字符=控制字符=不可见字符，是：`SOH` == `Start of Header` == `标题开始`



- 注：
  - 关于特殊的、不可见的、控制字符的细节，详见：字符编码详解

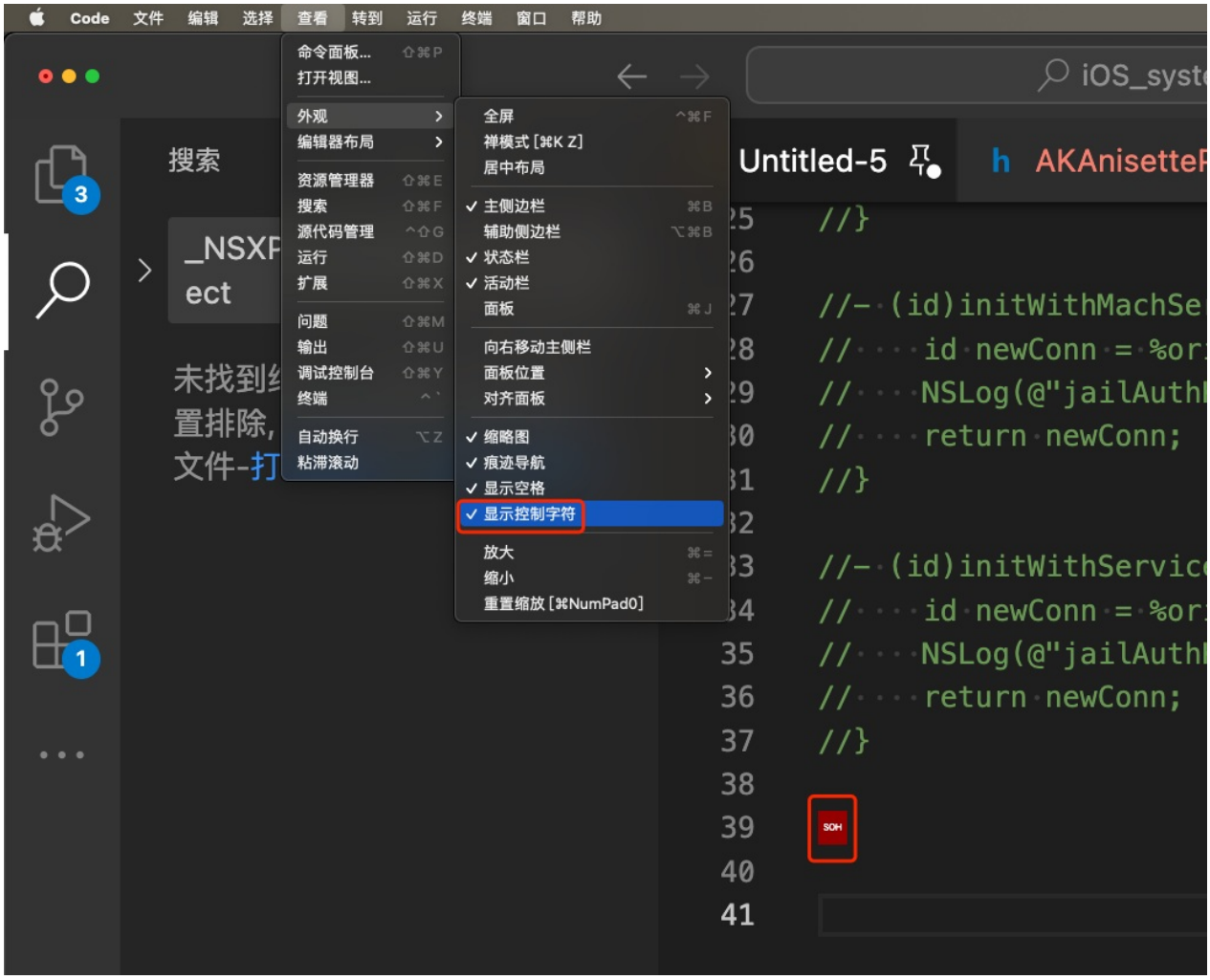
解决办法

（借助于VSCode，在能看到）此特殊的不可见的控制字符（的前提下），去删除掉

具体步骤

复制Xcode中的代码，粘贴到VSCode中

然后：VSCode-》查看-》外观-》显示控制字符-》找到对应不支持的特殊的控制字符，去删除即可



注：

- 关于VSCode支持显示不可见的控制字符，详见：代码编辑器常用功能 · 史上最好用的编辑器：VSCode

`Compile Sources` 中误添加了不支持的 `.xm`

现象

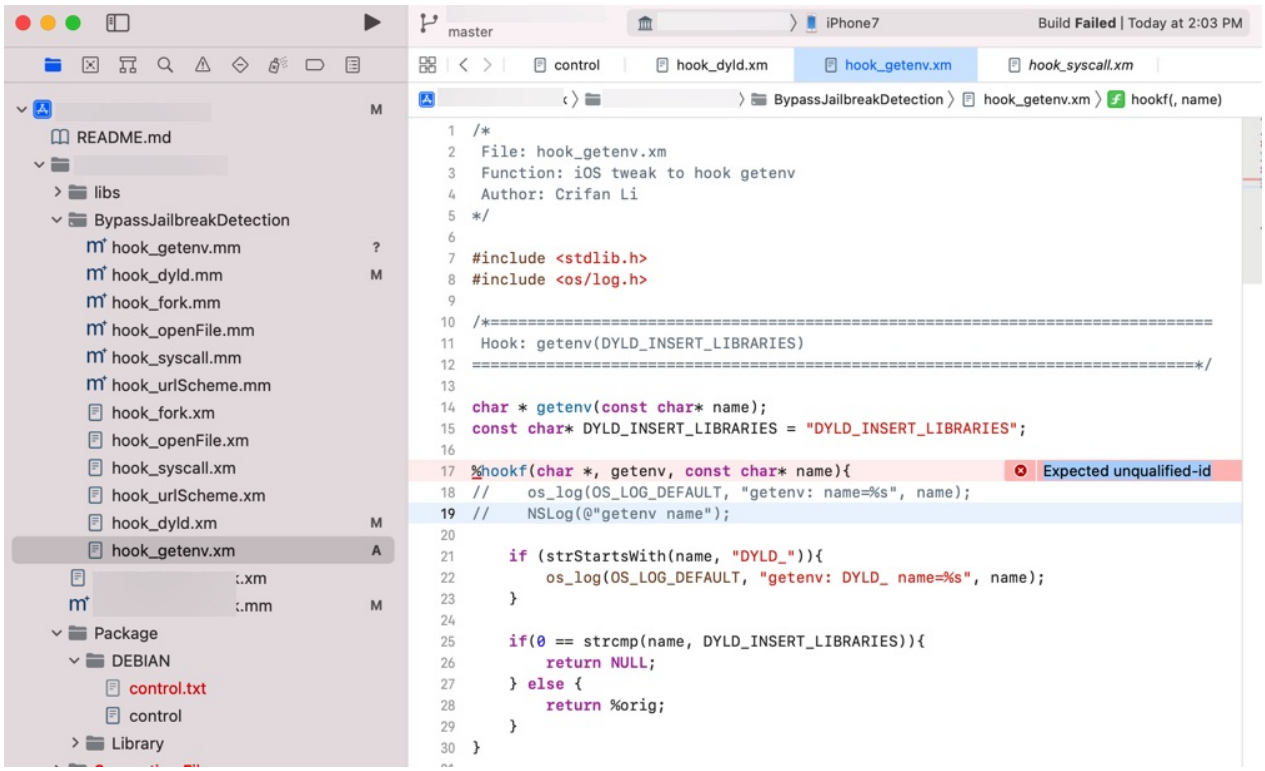
iOSOpenDev中，新增 `.xm` 文件后，尝试去编译

结果代码：

```
%hookf(char *, getenv, const char* name){
```

其中的 `%hookf` ，编译报错：

```
Expected unqualified-id
```



原因

直接原因

iOS的clang编译器无法识别（Logos的）语法：%hookf

根本原因

之前新增 `.xm` 文件时，无意间，不知道是哪里不小心，（把）导致了 `.xm` ，被加到了Xcode（+iOSOpenDev）中的 `Compile Sources` 中了

而此处，iOSOpenDev的最终的所支持的代码逻辑对应的源代码文件的格式：

- 不支持： `.xm`
- 只支持： `.mm`

而此处，背后的 `.xm` 和 `.mm` 的逻辑，详见：

.xm和.mm文件的逻辑

解决办法

1. 去 `Compile Sources` 中，把 `.xm` 文件移除掉
2. Xcode重新编译 `Build`
   - 目的：从 `.xm` 中编译生成对应的 `.mm` 文件
3. 再去把新生成的 `.mm` 文件，加到 `Compile Sources` 中，即可正常编译

具体步骤

`Xcode` -> `Targets` -> `Build Phases` -> `Compile Sources`

如果有 `.xm` 文件：则（点击 ）去删除掉

再去 `Xcode` -> `Build`：

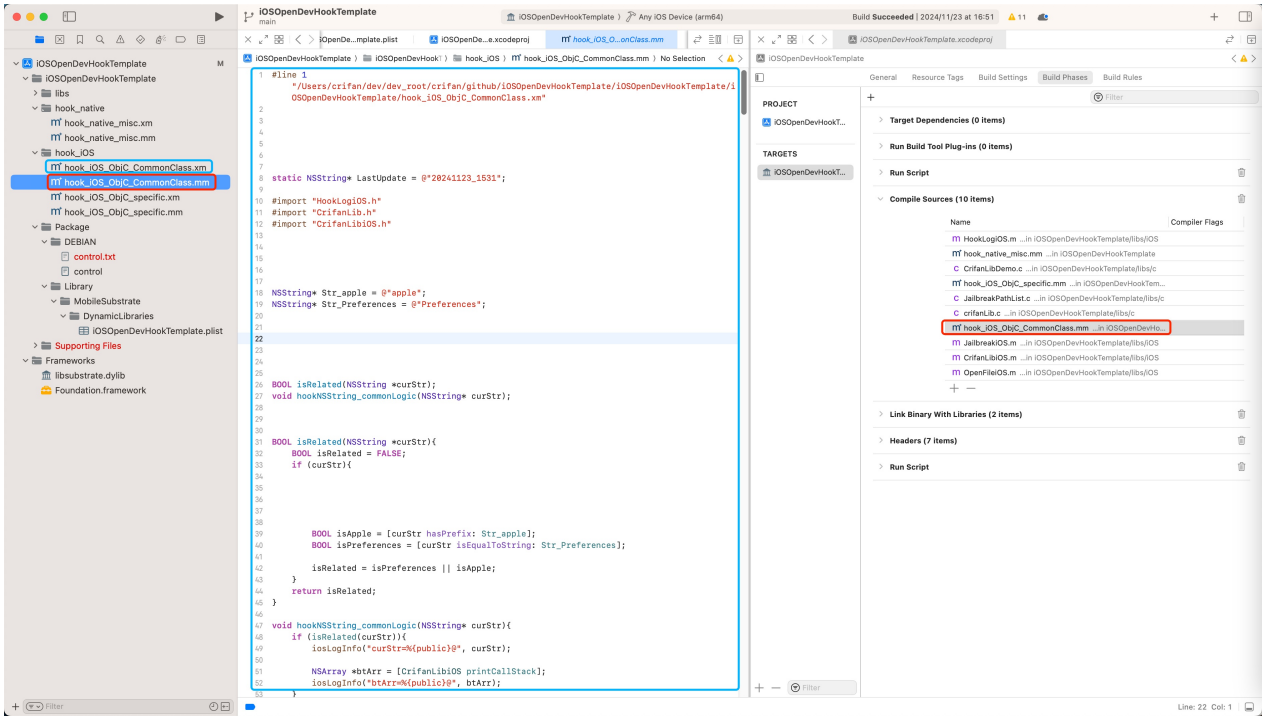
对于新增的 `.xm` 文件，则会生成新的对应的 `.mm` 文件

然后再去：

`Xcode` ->项目主目录->右键-> `Add Files to ...` -> 选择新生成的对应的 `.mm` 文件，且勾选： `Copy items if needed` -> `Add`，即可把 `.mm` 加到项目中

最后： `Xcode` -> `Targets` -> `Build Phases` -> `Compile Sources`

只保留正常的 `.mm` 文件（和其他的 `.c` 、 `.m` 等文件）：



Host key verification failed

- 现象：

Xcode编译期间报错：

```
Preparing to run Xcode Build Phase...
Signing /Users/crifan/Library/Developer/Xcode/DerivedData/iOSBypassJailbreak-bfqgivvncccwmeaykhtbtvgylkkq/Build/Products/Release-iphoneos/iOSBypassJailbreak.dylib with ldid... Done.
Copying /Users/crifan/Library/Developer/Xcode/DerivedData/iOSBypassJailbreak-bfqgivvncccwmeaykhtbtvgylkkq/Build/Products/Release-iphoneos/iOSBypassJailbreak.dylib to package directory at /Users/crifan/dev/dev_root/crifan/iOSBypassJailbreak/iOSBypassJailbreak/Package/Library/MobileSubstrate/DynamicLibraries...
Preparing to build package...
Setting control file /Users/crifan/dev/dev_root/crifan/iOSBypassJailbreak/iOSBypassJailbreak/Package/DEBIAN/control Version field to 1.0-1 using /Users/crifan/dev/dev_root/crifan/iOSBypassJailbreak/iOSBypassJailbreak/PackageVersion.plist... Done.
Building package ... Done.
Creating zip /Users/crifan/dev/dev_root/crifan/iOSBypassJailbreak/Packages/com.crifan.iOSBypassJailbreak_1.0-1_iphoneos-arm.zip... Done.
Host key verification failed.
Failed to create directory /var/root/iOSOpenDevPackages on device 192.168.1.27
Command PhaseScriptExecution failed with a nonzero exit code
```

- 原因：没有ssh免密登录
- 解决办法：设置好ssh免密登录
- 具体步骤概述
  1. 先

     ```
     ssh root@192.168.1.27
     ```

     - 默认密码：`alpine`
  2. 再

     ```
     ssh-copy-id root@192.168.1.27
     ```

scp: dest open ""/var/root/iOSOpenDevPackages/xxx.deb"": No such file or directory

现象

- Mac M2 Max中，用iOSOpenDev去编译deb插件到iPhone8，结果报错：
  -

```
Creating zip /Users/crifan/dev/dev_root/iosReverse/AppleStore/dynamicDebug/iOSOpenDev/jailAppleAccount/Packages/com.crifan.jailAppleAccount_1.7.4_iphoneos-arm.zip... Done.
scp: dest open ""/var/root/iOSOpenDevPackages/com.crifan.jailAppleAccount_1.7.4_iphoneos-arm.deb"": No such file or directory
scp: failed to upload file /Users/crifan/dev/dev_root/iosReverse/AppleStore/dynamicDebug/iOSOpenDev/jailAppleAccount/Packages/com.crifan.jailAppleAccount_1.7.4_iphoneos-arm.deb to "/var/root/iOSOpenDevPackages/com.crifan.jailAppleAccount_1.7.4_iphoneos-arm.deb"
Failed to copy file /Users/crifan/dev/dev_root/iosReverse/AppleStore/dynamicDebug/iOSOpenDev/jailAppleAccount/Packages/com.crifan.jailAppleAccount_1.7.4_iphoneos-arm.deb to device 192.168.2.13 at directory /var/root/iOSOpenDevPackages
Command PhaseScriptExecution failed with a nonzero exit code
```

作为对比：

旧的Mac（Intel的Mac），则没遇到这个错误。

对应细节是：

- 旧Mac：正常编译
  - 之前的Xcode版本：`v13.2.1`
  - scp：`3MB`
    - FAT格式，支持 `x86_64` 和 `arm64e`
- 新Mac：会报错
  - 错误信息：`scp: dest open ""/var/root/iOSOpenDevPackages/xxx.deb"": No such file or directory`
  - 最新Xcode版本：`v14.3`
  - scp：`416KB`
    - FAT格式，支持 `x86_64` 和 `arm64e`

原因

iOSOpenDev插件编译后安装deb期间，底层过程是对应脚本控制的

- `/opt/iOSOpenDev/bin/iosod`

其底层负责此处拷贝的命令是：

```
function copyFileToDevice() # args: sourceFile, targetDir, hostAddress, hostPort
{
  ...
  scp -P$hostPort "$sourceFile" root@$hostAddress:\"$targetFilePath\"" || \
```

而报错的原因，估计是：

新版Mac的CPU是ARM的Apple Silcon，对应的很多二进制，也是arm版本的

对应的scp，估计也是arm版本，其和旧的X86的scp，估计不太一样？

导致对于此处 scp中，带双引号的路径和文件，支持不够好，无法识别，所以报错找不到文件

解决办法

- 去掉scp命令中的（文件和目录中的）双引号

具体步骤

把 `/opt/iOSOpenDev/bin/iosod` 中的：

```
  scp -P$hostPort "$sourceFile" root@$hostAddress:\"$targetFilePath\"" || \
```

改为：

```
  scp -P$hostPort $sourceFile root@$hostAddress:$targetFilePath || \
```

An empty identity is not valid when signing a binary for the product type 'Dynamic Library'

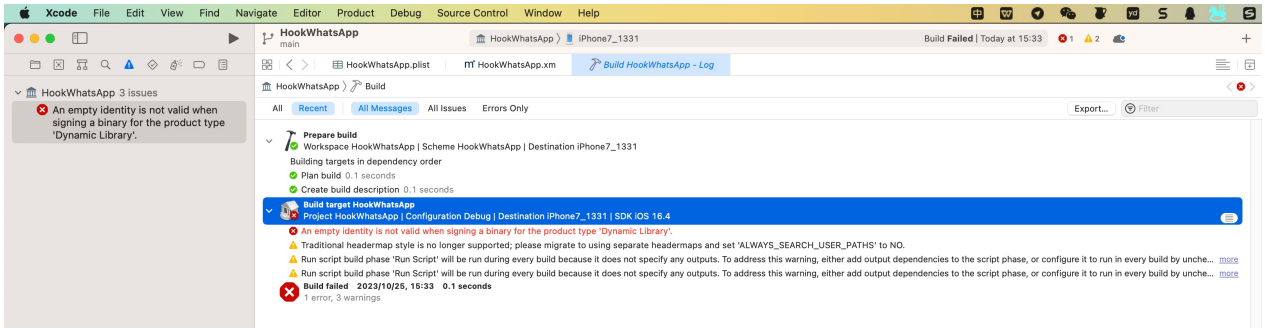
现象

`M2 Max Mac` + `Xcode 14.3.1` 中，新建iOSOpenDev项目，去Build编译，报错：

```
An empty identity is not valid when signing a binary for the product type 'Dynamic Library'
```

拷贝出的详细错误信息：

```
Build target HookWhatsApp of project HookWhatsApp with configuration Debug
error: An empty identity is not valid when signing a binary for the product type 'Dynamic Library'. (in target 'HookWhatsApp' from project 'HookWhatsApp')
'''
An empty identity is not valid when signing a binary for the product type 'Dynamic Library'.
```



原因

- 不是很清楚
  - 只是大概知道，属于Xcode的自动管理codesign签名方面的问题

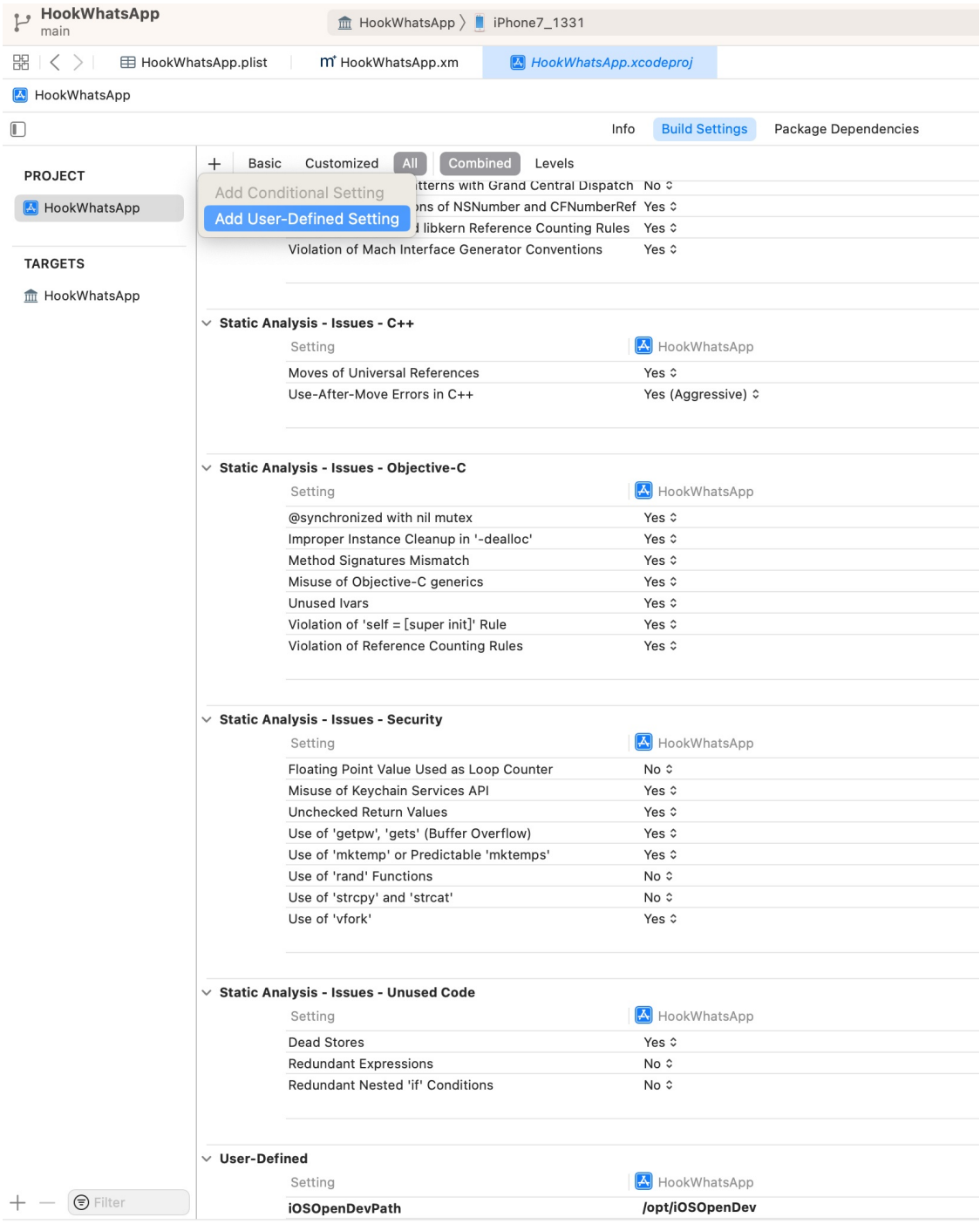
解决办法

- 思路1：禁用自动签名
- 思路2：给 `identity` 设置某个合适的值（这样就不是empty空了）

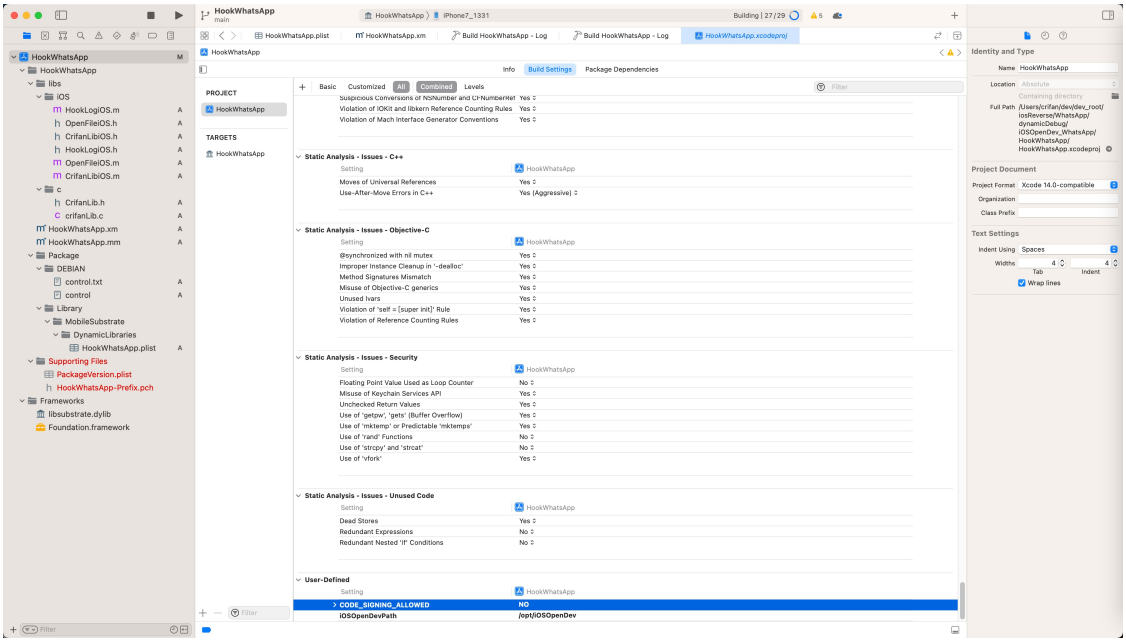
具体步骤

思路1：禁用自动签名

- `Xcode` -> `PROJECT` -> `{YourProjectName}` -> `Build Settings` -> `User-Defined` ->
  - (点击左上角的 `加号` = -> `Add User-Defined Setting`)



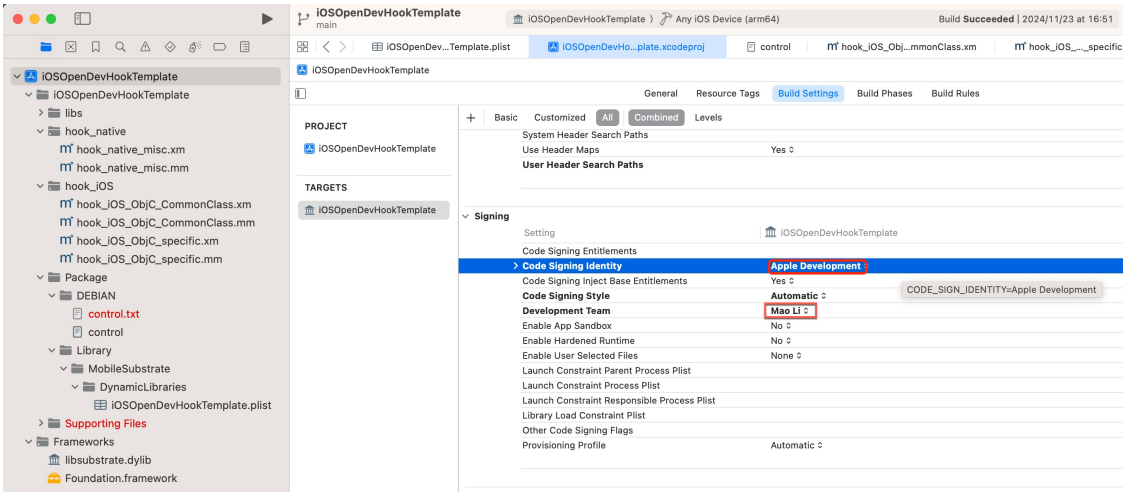
- 新增选项: `CODE_SIGNING_ALLOWED` = `NO`



- 额外说明
  - 如果还不行，多试几次Clean：
    - `Xcode -> Product -> Clean Build Folders`
    - `Xcode -> Product -> Clean All Issues`

思路2：给Identity设置某个合适的值

- `Xcode -> TARGETS -> {YourProjectName} -> Build Settings -> Signing`
  - 设置相关参数值
    - `Code Sign Identity` 设置为：（`Automatic` 中的）`Apple Development`
    - `Development Team` 设置为：你自己的Apple开发者账号 = 此处是：`Mao Li`
  - 效果图



不要设置Code Sign Identity为 `Apple Development: xxx`

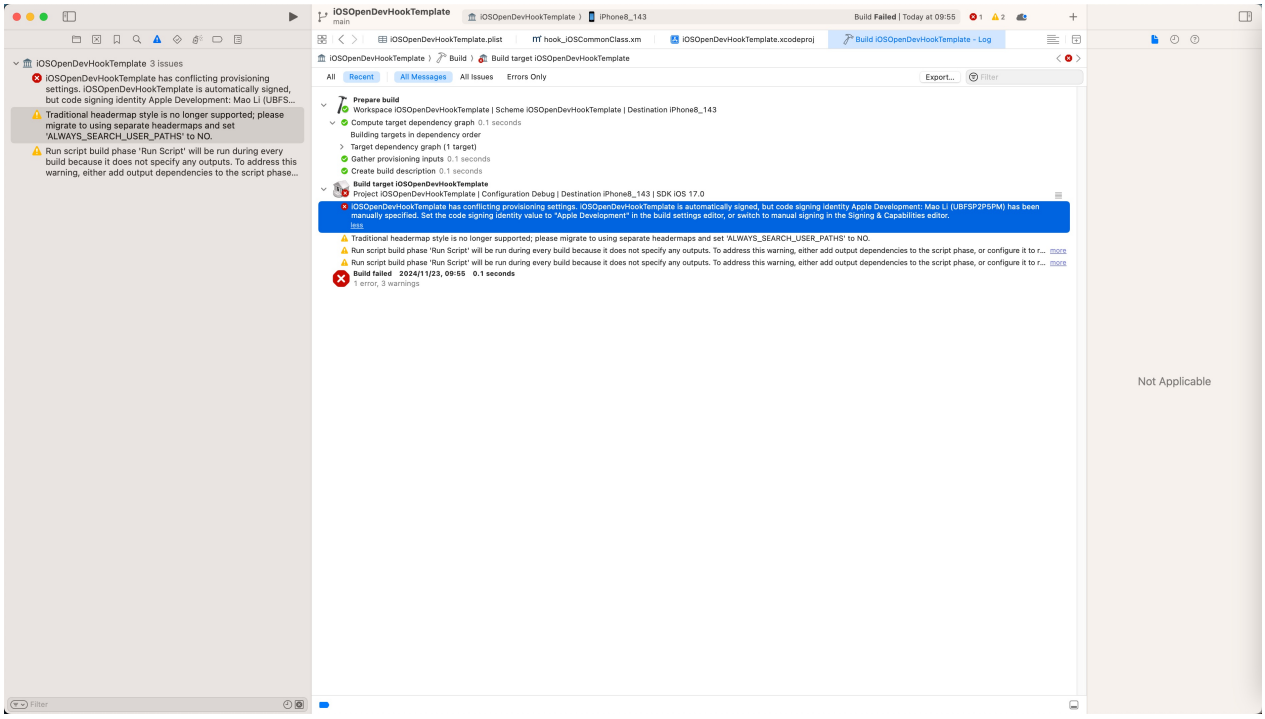
此处之前设置了：

- `Xcode -> TARGETS -> {YourProjectName} -> Build Settings -> Signing -> Code Sign Identity` ->设置为（`Certificates in Keychain` 中的）`Apple Development: Mao Li (UBFSP2P5PM)`

-

结果会报错：

```
/Users/crifan/dev/dev_root/crifan/github/iOSOpenDevHookTemplate/iOSOpenDevHookTemplate/iOSOpenDevHookTemplate.xcodeproj iOSOpenDevHookTemplate has conflicting provisioning settings. iOSOpenDevHookTemplate is automatically signed, but code signing identity Apple Development: Mao Li (UBFSP2P5PM) has been manually specified. Set the code signing identity value to "Apple Development" in the build settings editor, or switch to manual signing in the Signing & Capabilities editor.
```



然后改为上面说的（`Automatic`中的）`Apple Development`，才彻底解决了此处报错的问题。

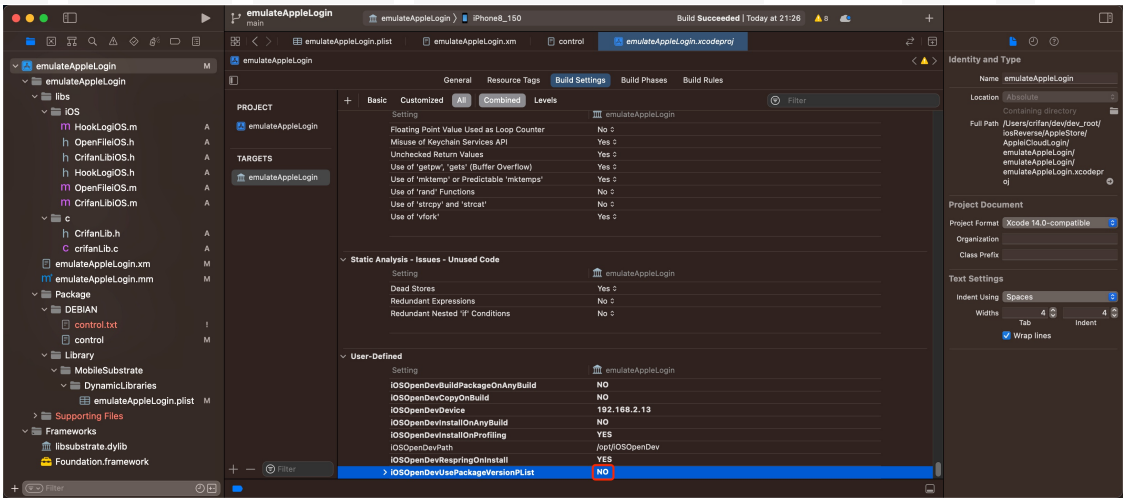
control的Version版本号的改动会丢失

- 现象：

项目中的 `.plist` 中的 `Version` 的值，默认是 `1.0-1`

当想要去改动版本号，比如改为 `2023.07.19.2126`，结果重新编译后，改动后的Version值丢失，又恢复到之前的默认值 `1.0-1` 了

- 解决办法
  - `TARGETS` -> `Build Settings` -> `User-Defined` -> `iOSOpenDevUsePackageVersionPList` 从（默认的） `YES` 改为 `NO`



安装插件后桌面上看不到iOS的app图标

- 现象：带UI界面的tweak或app，（通过Filza）安装到iPhone中后，桌面上看不到iOS的app的图标
- 原因：iPhone的UI界面没有刷新=icon图标没有刷新
- 解决办法：
  - `uicache`
    - 如果是通过Filza安装deb的话
      - Filza的安装完成界面的点击右上角：`动作` -> 选择：`UIcache`
        - 图
        - 

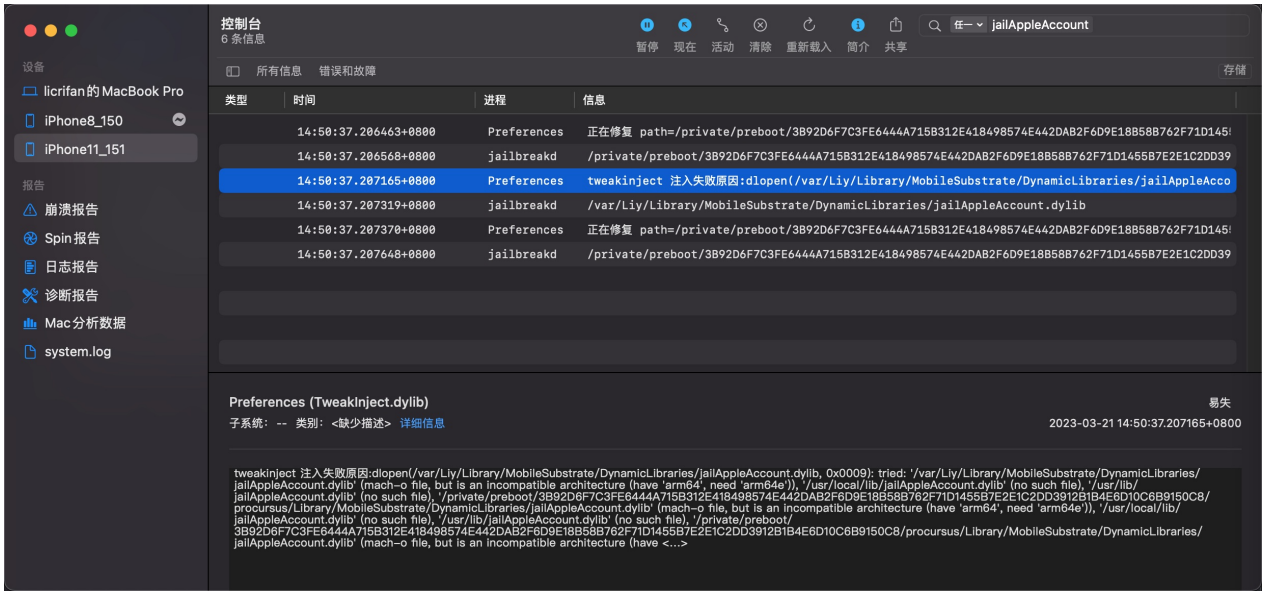
      - 稍等片刻-》桌面上即可出现iOS的app的logo图标了
        - 注：此时点击 `注销` = `Respring` = `重启SpringBoard`，虽然理论上可行，但实际是无效的，无法让桌面出现app图标的
    - 如果不是，则可以单独命令行去运行：`uicache`

mach-o file, but is an incompatible architecture have 'arm64', need 'arm64e'

现象

iOSOpenDev的Xcode编译出了插件dylib插件，但是启动加载时报错：

```
'/private/preboot/xxx/procursus/Library/MobileSubstrate/DynamicLibraries/jailAppleAccount.dylib' (mach-o file, but is an incompatible architecture (have 'arm64', need 'arm64e'))
```



原因

此处目标设备 `iPhone11` 的 `CPU` 是 `A12`，其架构是 `arm64e` 的，而插件代码编译出的架构是针对 `arm64` 的，不兼容，所以报错

解决办法

- Xcode中去把架构改为（包含=支持） `arm64e`

具体步骤

- `Xcode` -> `TARGETS` -> `YourProjectName` -> `Build Settings` -> `Architectures` -> `Architectures`
  - 从默认的： `$(ARCHS_STANDARD)` == `arm64, armv7`
  - 改为： `Other` 的 `arm64 arm64e`

如此，即可确认所编译出来的代码（插件），支持arm64e了。

注

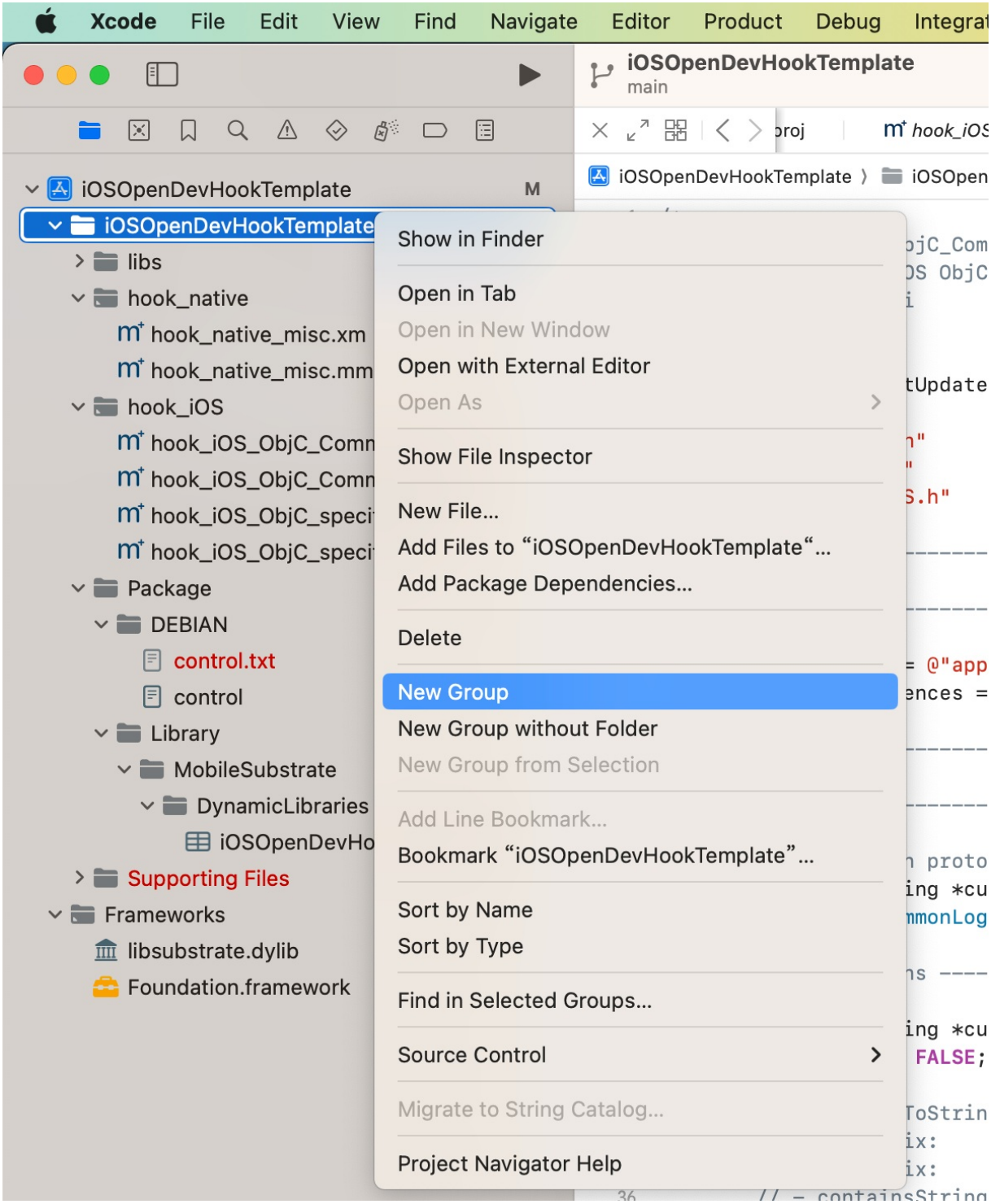
- 如果额外引用到库文件，则也要确保库文件是支持此处的arm64e的
  - 比如此处遇到 `libsubstrate.dylib`，就是：

- 默认（iOSOpenDev自带的）不支持arm64e，最后是另外找支持arm64e的
  - 比如
    - XinaA15越狱后的iPhone11中有
      - `/private/preboot/3B92D6F7C3FE6444A715B312E418498574E442DAB2F6D9E18B58B762F71D1455B7E2E1C2DD3912B1B4E6D10C6B9150C8/procursus/usr/lib/libsubstrate.dylib`
        - 大小：218KB
        - (就是我们要的)FAT格式的，支持2种架构：`arm64` 和 `arm64e`
- 去拷贝替换掉原先的：`Mac` 中的 `/opt/iOSOpenDev/lib/libsubstrate.dylib`
- 才顺利编译和链接，才能确保插件正常工作

Failed Logos Processor outputted Could not open xm file

现象

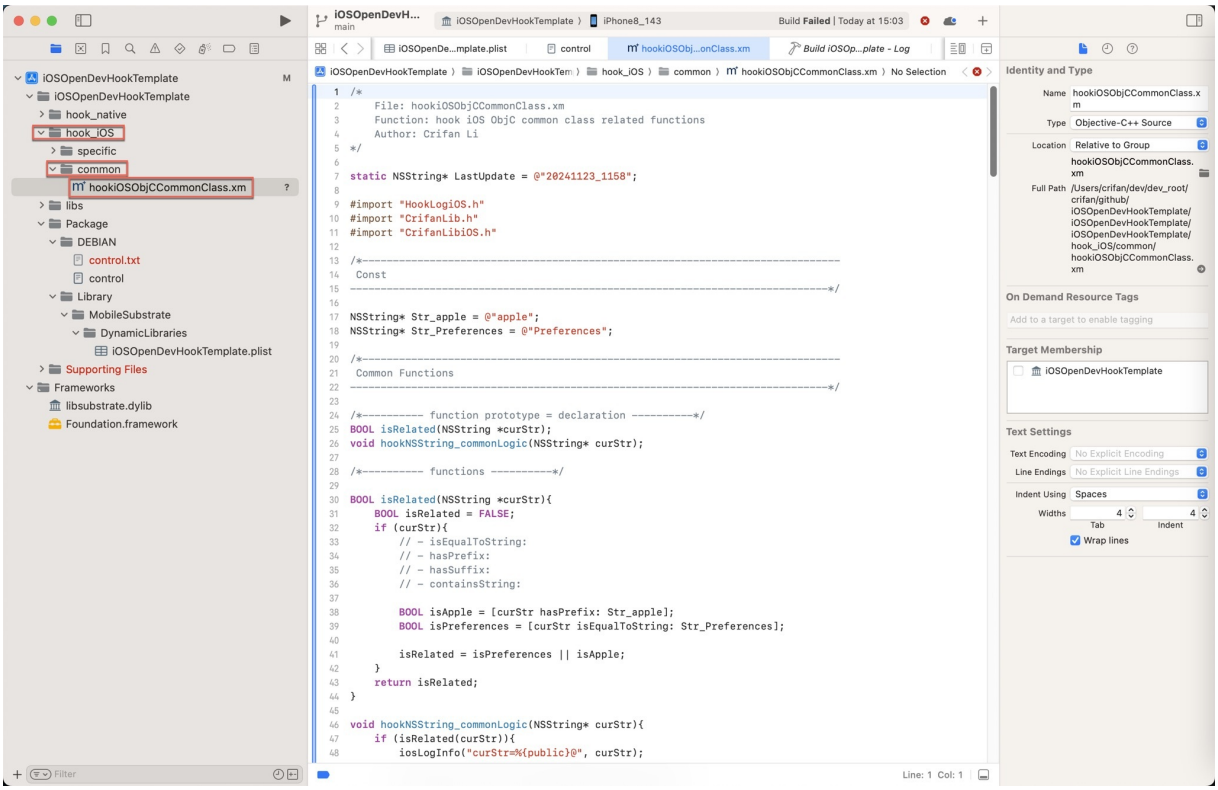
之前iOSOpenDev中，主项目目录下面，新建**子目录**：

- 

在子目录中，新增 `.xm` 文件：

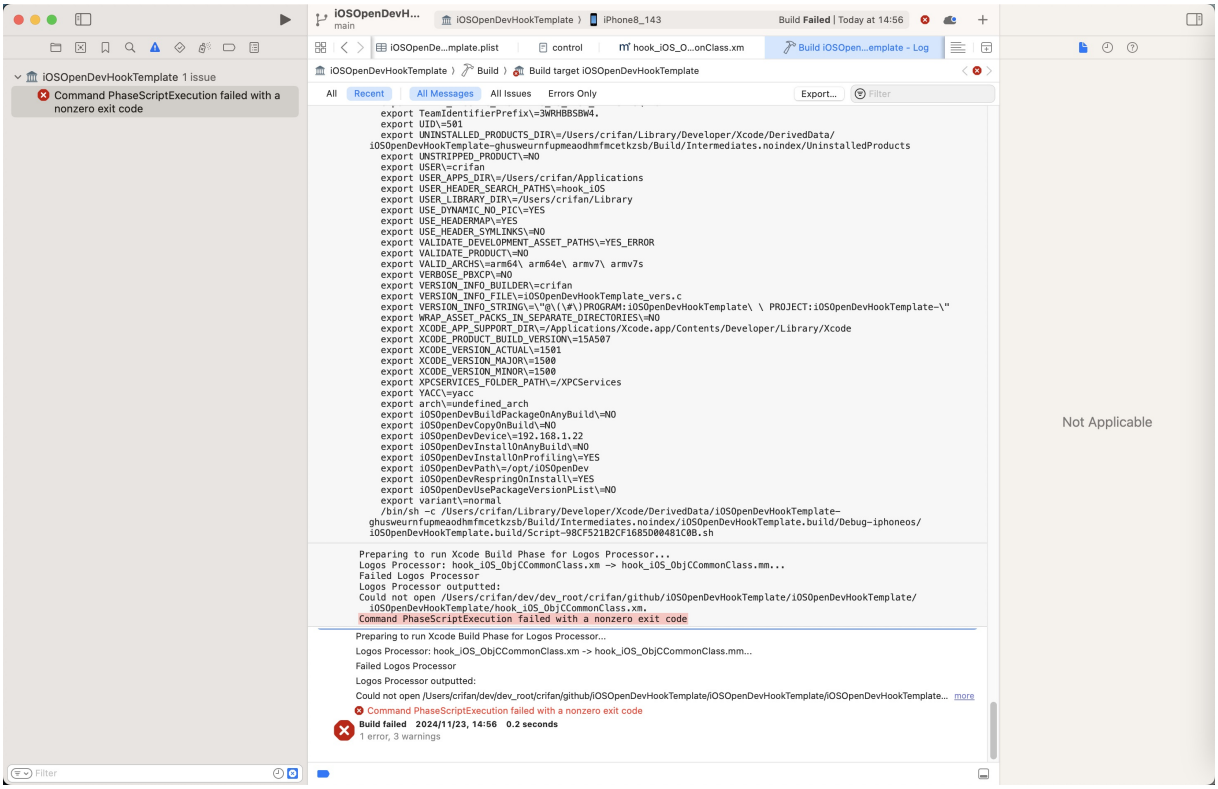
- iOSOpenDevHookTemplate
  - hook_iOS

- common
  - hook_iOS_ObjCCommonClass.xm



会导致编译报错：

```
Preparing to run Xcode Build Phase for Logos Processor...
Logos Processor: hook_iOS_ObjCCommonClass.xm -> hook_iOS_ObjCCommonClass.mm...
Failed Logos Processor
Logos Processor outputted:
Could not open /Users/crifan/dev/dev_root/crifan/github/iOSOpenDevHookTemplate/iOSOpenDevHookTemplate/iOSOpenDevHookTemplate/hook_iOS_ObjCCommonClass.xm.
Command PhaseScriptExecution failed with a nonzero exit code
```

- 

原因

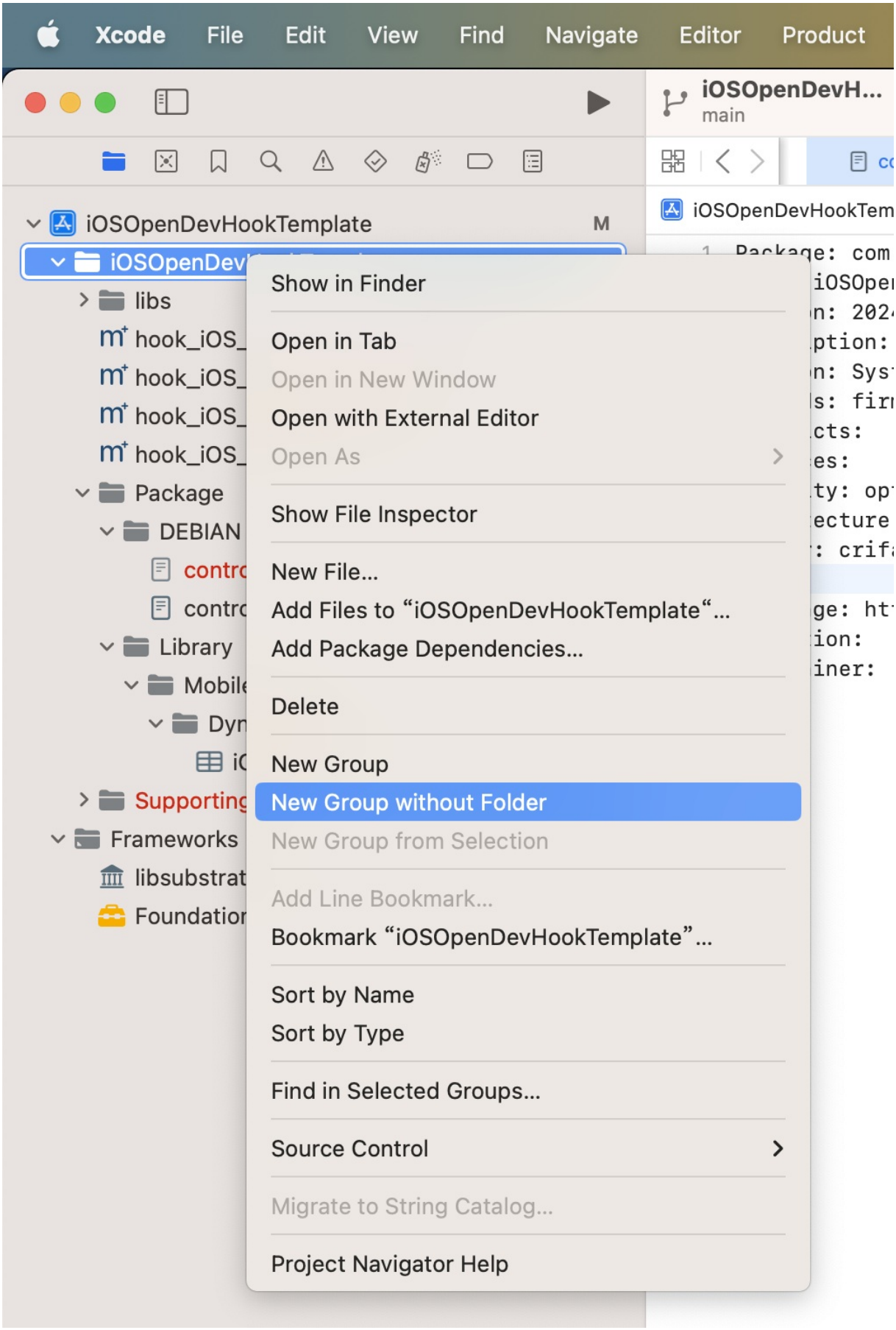
（看来是）iOSOpenDev中，不支持子目录 == 如果把 `.xm` 放在子目录中，则 `Logos Processor` 则会找不到，导致编译报错

解决办法

【推荐】思路1：用虚拟目录

Xcode中，右键新建目录时：

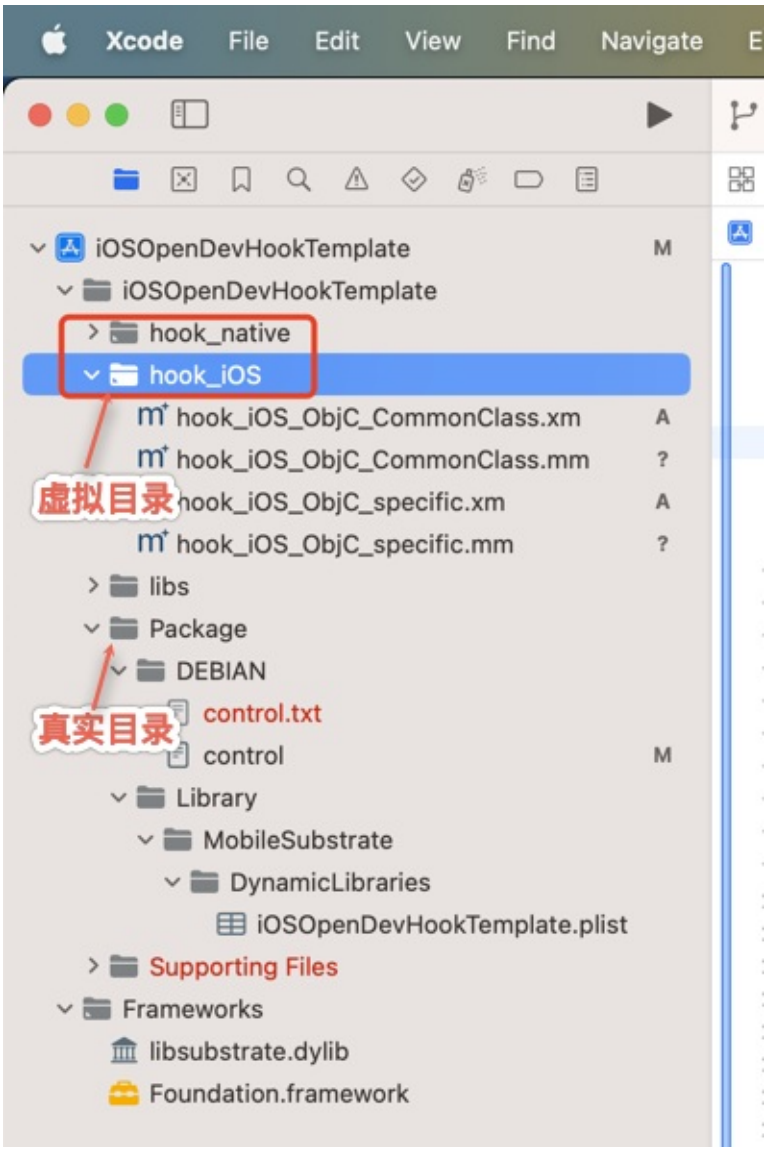
- 不用：`New Group`
  - -> 创建物理上的，文件系统上，真实的：子目录
- 改用：`New Group without folder`
  - -> Xcode中虚拟的子目录 = 物理上的，文件系统上的子目录：是没有的，是不存在的



效果

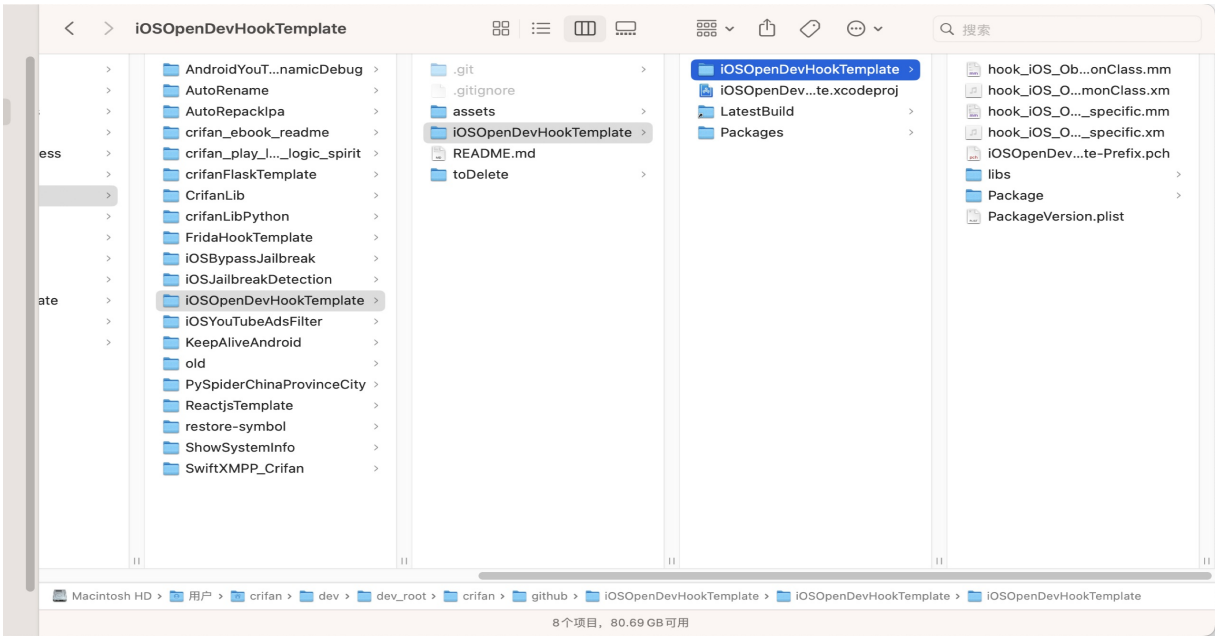
（1）Xcode中文件夹图标

虚拟文件夹 和 真实文件夹：图标，略有不同：

- 文件夹左下角的小三角
  - 虚拟文件夹：有
  - 真实文件夹：没有
- 

（2）Xcode中虚拟的子目录 = 物理上的，文件系统上的子目录：是没有的，是不存在的

Finder中，是看不到（Xcode中的虚拟的）子目录的：

- 

【不推荐】思路2：放弃创建子目录

Xcode中的iOSOpenDev中，新增文件时：不用 `New Group`，即：

只能把所有 `.xm`（和对应的 `.mm` 文件，都直接放到根目录下，没有子目录了

-》虽然当文件多时，逻辑上不够清晰，但是至少能凑合用

心得

TODO：

- 【未解决】XCode调试警告：was compiled with optimization stepping may behave oddly variables may not be available
- 【未解决】iOSOpenDev的XCode的tweak插件编译尝试去掉优化加上调试信息
-
- 【已解决】iOSOpenDev的XCode调试iPhone6报错：Unable to install The application could not be verified
- 【已解决】XCode中删除掉User-Defined的自定义参数
-
- 【已解决】iOSOpenDev的XCode的iOS的tweak插件中实现ObjC的通用全局函数
- 【已解决】iOSOpenDev的XCode的iOS插件运行报错：ImageLoaderMachO doModInitFunctions和_logosLocalInit
- 【已解决】iOS代码报错：objc Class is implemented in both app and dylib One of the two will be used Which one is undefined
-
- 【已解决】iOSOpenDev的XCode项目编译报错：iPhone Developer no identity found
- 【已解决】调试iOSOpenDev的XCode的iOS的app
- 【已解决】研究iOSOpenDev的XCode项目编译过程以确保如何链接自定义.c文件的.o文件
- 【已解决】iOSOpenDev的XCode项目偶尔编译非常慢卡死
- 【已解决】iOSOpenDev的XCode中新增.c和.h文件并正常编译
- 【已解决】如何把XCode的iOS的app项目转换成iOSOpenDev的项目
- 【已解决】对比研究FakeWeChatLoc和自己的XCode项目的目录结构区别
- 【已解决】iOSOpenDev的XCode调试iPhone7报错：Unable to install A system application with the given bundle identifier is already installed on the device and cannot be replaced
- 【记录】更新iOSOpenDev的Logos插件的code signing签名配置
- 【已解决】XCode中iOSOpenDev开发插件代码报错：No matching function for call to strcpy
- 【已解决】XCode中iOSOpenDev的Tweak项目中Build Settings中User-Defined中添加和引用变量THEOS
- 【记录】研究XCode中clang编译mm文件的过程和编译参数
- 【记录】深究为何此处XCode编译strcpy会报错No matching function for call to
- 【未解决】把之前theos的tweak改机剩余功能移植到iOSOpenDev的XCode中
-

`.xm` 文件和 `.mm` 文件

TODO：

- 【已解决】Xcode中xm源码中无法看到和添加断点
- 【已解决】iOSOpenDev的XCode中.xm文件包含.c中函数找不到报错：Undefined symbols for architecture arm64 referenced from
- 【已解决】XCode的iOSOpenDev项目报错：Failed Logos Processor Could not open xm
- 【已解决】iOSOpenDev的XCode中如何把Tweak的xm代码拆分成多个文件模块
- 【未解决】iOSOpenDev的iosod的bug修复：Logos的预处理不支持group子目录中的xm文件

代码高亮

- 【已解决】iOSOpenDev的XCode中新增xm文件设置为Logos语法高亮但无效
- 【已解决】让XCode的iOSOpenDev中Logos的xm文件支持语法高亮

iOSOpenDev内部逻辑和过程

TODO：

- 【未解决】研究iOSOpenDev的XCode项目编译过程以确保如何链接自定义.c文件的.o文件
- 【已解决】XCode编译iOSOpenDev的Logo Tweak项目报错： Command PhaseScriptExecution failed with a nonzero exit code Failed to locate Logos Processor
- 【未解决】XCode中编译iOSOpenDev的Logos的Tweak时shell从sh换为zsh
- 【已解决】给iOS的XCode项目中新增iOSOpenDev的Project Navigator的目录和文件
- 【已解决】XCode项目中新增iOSOpenDev的Package目录到Target目录中
- 【已解决】XCode中如何把libsubstrate.dylib动态库导入到Link Binary With Libraries

如何卸载带UI的插件app

- 通过Cydia去卸载已安装的（带UI界面的）插件tweak即可

步骤：

`Cydia` -> `已安装` -> `最近` ->找到插件->进入详情页


中国移动
下午11:08
已安装
详情
卸载
沐家百货
2021.11.04-2
更改软件包设置
作者
Crifan
已安装软件包
版本
2021.11.04-2
文件系统内容
com.crifan.mujiabaihuoapp
· 系统
Cydia
软件源
变更
已安装
搜索

点击右上角的 卸载 ：

即可卸载掉插件。

相关教程和代码

- 相关教程
  - 推荐用这个方式去iOS逆向调试
    - iOS逆向调试：Xcode+iOSOpenDev
- 代码
  - iOSOpenDev的hook插件代码开发的模板
    - iOSOpenDevHookTemplate
      - Github
        - https://github.com/crifan/iOSOpenDevHookTemplate
          - crifan/iOSOpenDevHookTemplate: Crifan's iOSOpenDev Hook Template, for common iOS ObjC Class, native C functions, other misc
      - 代码
        - hook_iOS_ObjC_CommonClass.xm
          - https://github.com/crifan/iOSOpenDevHookTemplate/blob/main/iOSOpenDevHookTemplate/iOSOpenDevHookTemplate/hook_iOS_ObjC_CommonClass.xm
        - hook_iOS_ObjC_specific.xm
          - https://github.com/crifan/iOSOpenDevHookTemplate/blob/main/iOSOpenDevHookTemplate/iOSOpenDevHookTemplate/hook_iOS_ObjC_specific.xm
        - hook_native_misc.xm
          - https://github.com/crifan/iOSOpenDevHookTemplate/blob/main/iOSOpenDevHookTemplate/iOSOpenDevHookTemplate/hook_native_misc.xm

附录

下面列出相关参考资料。

参考资料

- 【整理】iOS越狱插件开发工具：iOSOpenDev
- 【已解决】把iOS的app和iOS的tweak插件打包成独立的deb安装包
- 【已解决】把iOSOpenDev的tweak加app的deb文件安装到已越狱的iPhone中
- 【已解决】如何把普通iOS的app的XCode项目和iOSOpenDev的Logos插件tweak集成到一起
- 【已解决】如何把XCode的iOS的app项目转换成iOSOpenDev的项目
- 【已解决】给iOSOpenDev的app和tweak用配置文件互相通信
- 【已解决】已越狱iPhone中卸载tweak加app的deb插件
- 【记录】确认iPhone中安装后的tweak加app是否正常使用
- 【已解决】把iOSOpenDev的tweak插件和app合并打包成deb文件
- 【已解决】把iOSOpenDev的tweak加app的deb文件安装到已越狱的iPhone中
- 【已解决】iOSOpenDev的XCode去Build For Profiling安装后iPhone桌面上找不到iOS的app的图标
- 【已解决】用Filza安装tweak加app的deb后iPhone桌面中仍没出现iOS的app的logo图标
- 【已解决】ssh登录iPhone失败：Host key verification failed
- 【已解决】Mac中安装iOSOpenDev
- 【已解决】Mac中安装iOSOpenDev报错：安装器遇到了一个错误，导致安装失败
- 【已解决】Mac中初始化iOSOpenDev环境并新建插件项目
- 【已解决】iOSOpenDev设置SDK报错：File not found XCode Specifications iPhoneOSPackageTypes.xcspec
- 【已解决】iOSOpenDev设置SDK报错：PrivateFramework directory not found XCode iPhoneOS15.0.sdk
- 【已解决】用iOSOpenDev去开发带GUI图形界面的iOS的app和tweak插件集成在一起的插件deb包
- 【已解决】iOS逆向：如何去hook一个进程而不是带包名的iOS的app
- 【已解决】XCode中iOSOpenDev中修改control的Version版本号无效会被重置
- 【未解决】iOS逆向akd：新建iOSOpenDev的Xcode插件项目
- 【已解决】iOSOpenDev的插件dylib注入iPhone11失败：mach-o file but is an incompatible architecture have arm64 need arm64e
- 【已解决】寻找支持arm64e的libsubstrate.dylib
- 【已解决】Xcode项目中引用新的libsubstrate.dylib无效：始终链接是旧的库文件
- 【已解决】Mac M2 Max中iOSOpenDev编译报错：An empty identity is not valid when signing a binary for the product type Dynamic Library
- 【已解决】XCode中删除用户自定义配置User-Defined中的 CODE_SIGNING_ALLOWED=NO
- 【已解决】Mac中iOSOpenDev编译报错scp dest open /var/root/iOSOpenDevPackages/xxx.deb No such file or directory
- 【已解决】iOSOpenDev的XCode中xm代码%hookf编译报错：Expected unqualified-id
- 【已解决】Xcode中Logos插件hook代码编译报错：Expected unqualified-id
- 【规避解决】Xcode中iOSOpenDev编译报错：Failed Logos Processor outputted Could not open sub folder xm
-
- iosOpenDev-install 失败官方wiki无法解决看这里（尝试有效） - PoloKey - 博客园 (cnblogs.com)
- zhangkn/knPrivateFrameworks: /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS9.2.sdk/System/Library/PrivateFrameworks (github.com)
- iosopendev专用Specifications.zip
- 越狱开发:用iosOpenDev配置越狱开发环境 编写第一个hello world_我的杯洗具的博客-CSDN博客
- jackrex/FakeWeChatLoc: 手把手教你制作一款iOS越狱App (github.com)
- ios - An empty identity is not valid when signing a binary for the product type 'Application' in xcode version 10.2 - Stack Overflow
-