

目录

前言	1.1
调试断点概览	1.2
通用逻辑	1.3
背景知识	1.3.1
得到函数的实际地址	1.3.1.1
普通断点	1.3.2
高级用法	1.3.3
指定模块	1.3.3.1
条件判断	1.3.3.2
举例	1.3.3.2.1
Xcode图形界面中的断点	1.4
lldb命令行中的断点	1.5
breakpoint的help语法	1.5.1
常见问题	1.6
断点没触发	1.6.1
Xcode	1.6.2
hook代码的断点	1.6.2.1
lldb	1.6.3
经验心得	1.7
通用	1.7.1
导致卡死	1.7.1.1
条件判断断点	1.7.2
表达式写法	1.7.2.1
函数名和地址逻辑一致	1.7.2.2
ObjC	1.7.3
Xcode	1.7.4
调试多个函数断点	1.7.4.1
临时关闭所有断点	1.7.4.2
断点是否加成功	1.7.4.3
断点加不上	1.7.4.3.1
断点能加上且能触发	1.7.4.4
lldb无名函数	1.7.5
恢复符号表	1.7.5.1
附录	1.8

iOS逆向之动态调试：断点

- 最新版本: v1.5.0
- 更新时间: 20231026

简介

介绍iOS逆向期间涉及到的断点的各方面的知识和心得。先对于iOS逆向调试期间的断点进行概览介绍。接着介绍Xcode图形界面和lldb命令行中的断点的通用逻辑，包括相关背景知识，比如如何得到函数的实际地址；以及介绍普通断点的通用逻辑；接着介绍普通断点如何加断点，以及断点的一些高级用法，比如指定模块、条件判断断点和对应例子；接着介绍Xcode图形界面和lldb命令行中的关于断点的各自特有的内容。包括lldb中breakpoint的帮助语法。接着整理常见问题，包括断点没触发、Xcode中hook代码的断点以及lldb；以及各种经验和心得，包括通用的，比如加断点导致卡死等；以及ObjC方面的断点，和Xcode的各种心得，包括如何同时调试多个断点、临时关闭所有断点、确认断点是否加上、确保断点能加上且能触发等；以及lldb无名函数和恢复符号表；最后贴上附录资料。

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_debug_breakpoint: iOS逆向之动态调试：断点](#)

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逆向之动态调试：断点 book.crifan.org](#)
- [iOS逆向之动态调试：断点 crifan.github.io](#)

离线下载阅读

- [iOS逆向之动态调试：断点 PDF](#)
- [iOS逆向之动态调试：断点 ePUB](#)
- [iOS逆向之动态调试：断点 Mobi](#)

版权和用途说明

此电子书教程的全部内容，如无特别说明，均为本人原创。其中部分内容参考自网络，均已备注了出处。如发现有侵权，请通过邮箱联系我 [admin 艾特 crifan.com](mailto:admin@crifan.com)，我会尽快删除。谢谢合作。

各种技术类教程，仅作为学习和研究使用。请勿用于任何非法用途。如有非法用途，均与本人无关。

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](#)

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-10-30 22:10:41

调试断点概览

iOS逆向期间，尤其是用：

- Xcode 的图形界面= GUI
- lldb (+ debugserver) 的命令行= command line

的动态调试期间，涉及到很多断点相关的心得，整理如下。

而本身部分内容，之前是单独属于 Xcode 调试 和 lldb+debugserver 中的，但是此处发现有很多东西两者关系很紧密，且有其他关联，所以统一放在一起介绍，可以加深理解断点背后的逻辑。

- 断点 = Breakpoint
 - 是什么：程序中为了调试而故意停止或者暂停的地方
 - 用途：程序停下来后，可以查看变量的值，调试搞懂程序的逻辑

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook 最后更新：

2023-07-07 23:45:35

通用逻辑

下面介绍，Xcode图形界面 和 lldb命令行 中 的断点的通用的逻辑。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-07 21:40:26

背景知识

查看进程PID

- 查看进程PID

 - 方式1

 - Mac 中

```
frida-ps -Uia
```

 - 方式2

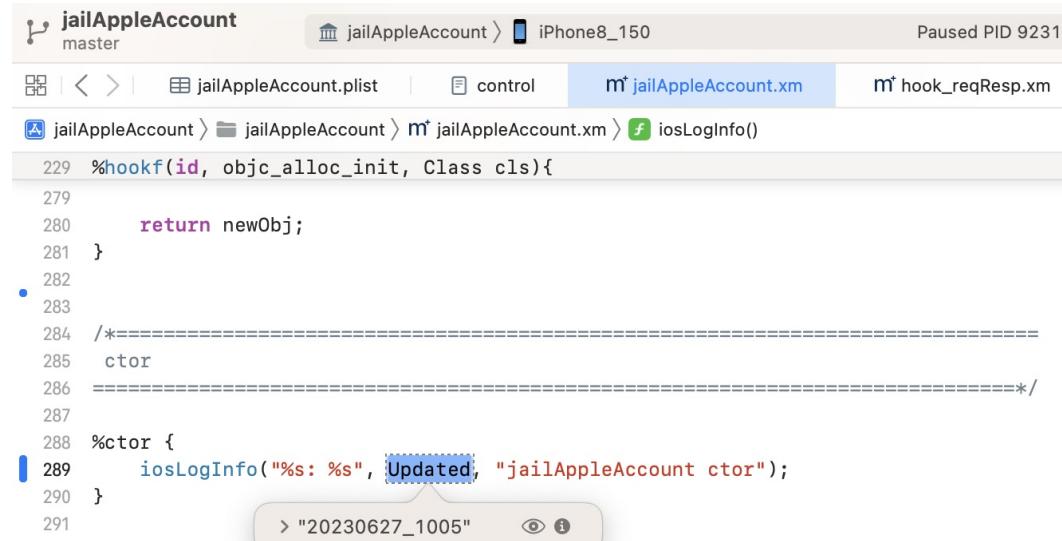
 - iPhone 的 ssh 中

```
ps -A | grep Preferences
```

确保最新的dylib插件动态库被加载

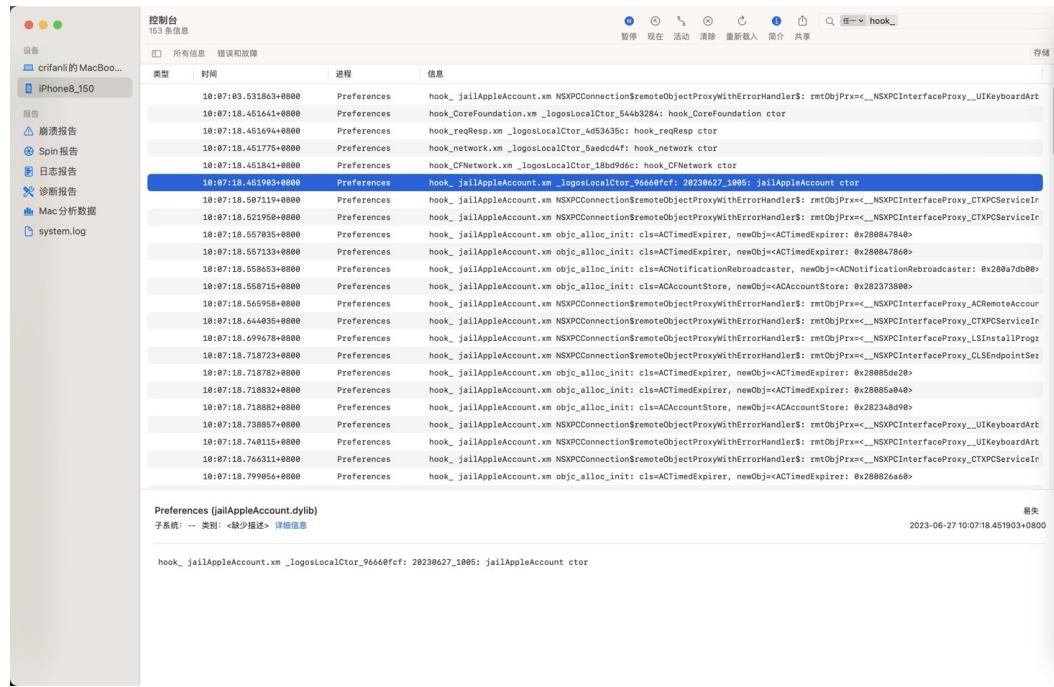
- 小技巧：想要确保：最新的dylib插件动态库被加载了 -》 即可实现hook代码断点可以加上，且能触发断点
 - 可以通过 Console.app = 控制台 中是否输出相关ctor日志确认dylib是否已加载，是否是新版本
 - 把ctor中代码加上最新版字符串

```
static char* Updated = "20230627_1005";
...
%ctor {
    iosLogInfo("%s: %s", Updated, "jailAppleAccount ctor");
}
```



 - 此处能输出最新版字符串，即表示dylib加载了，且是最新版

```
默认 10:07:18.451903+0800 Preferences hook_jailAppleAccount.xm _logosLoc
alCtor_96660fcf: 20230627_1005: jailAppleAccount ctor
```



iOS中ObjC中的Block的函数名

Block的invoke函数名，往往是：

- Block所在原函数名 + _block
 - 举例
 - 原ObjC函数名： -[MLHAMQueuePlayerSegmentList updatePeriodCurrentTimeForSegment:]
 - 对应block函数名： -[MLHAMQueuePlayerSegmentList updatePeriodCurrentTimeForSegment:]_block

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：

2023-07-06 22:42:20

得到函数的实际地址

iOS逆向调试期间，往往需要搞清楚某个函数的实际地址，然后用于后续通过地址添加断点而函数的实际地址，有多种方式可以获取到：

通过po查看类的描述，得到函数实际地址

举例说明：

当某个函数

```
-[__NSXPCInterfaceProxy_AKAppleIDAuthenticationDaemonProtocol  
authenticateWithContext:completion:]
```

去通过Xcode的UI界面去（调试期间，实时去）添加断点后，发现加不上断点 == 具体表现是断点的底色不是蓝色，而是灰色，且是虚线边框

而想要去加断点的话，也通过函数模糊搜索：

```
im lookup -r -n "authenticateWithContext:completion:"
```

的确搜不到该函数

此时，想要给该函数加断点的话：

可以通过函数地址加断点，函数地址可以从po查看实例的描述中找到

```
(lldb) po [0x28211abc0 _shortMethodDescription]  
<__NSXPCInterfaceProxy_AKAppleIDAuthenticationDaemonProtocol: 0x28211abc0>:  
in __NSXPCInterfaceProxy_AKAppleIDAuthenticationDaemonProtocol:  
    Instance Methods:  
    ...  
        - (void) authenticateWithContext:(id)arg1 completion:(id)arg2; (0xb74ee6818326c  
cc9c)  
    (_NSXPCDistantObject ...)
```

- 该函数的实际地址： 0xb74ee6818326cc9c

由此后续去通过该函数的实际地址添加断点：

```
breakpoint set -a 0xb74ee6818326cc9c  
  
br s -a 0xb74ee6818326cc9c
```

即可顺利加上断点，并真正触发到该断点。

计算出函数的实际地址

- 背景：已知函数在二进制内部的偏移量
- 希望：计算出函数的实际的地址
 - 计算公式： 函数实际地址 = 二进制的ALSR基地址 + 函数二进制内偏移量

举例

RzGame 中的 sub_1000B0770 函数，起始地址是： 0x00000100157BB0

而二进制 RzGame 的ALSR基地址，通过：

```
(lldb) image list -o -f | grep RzGame
[ 0] 0x0000000000c04000 /Users/crifan/Library/Developer/Xcode/DerivedData/udg-cuzlxqfn
klemxfexjbcfnupseasc/Build/Products/Debug-iphoneos/udg.app/RzGame
```

查看到是： 0x0000000000c04000

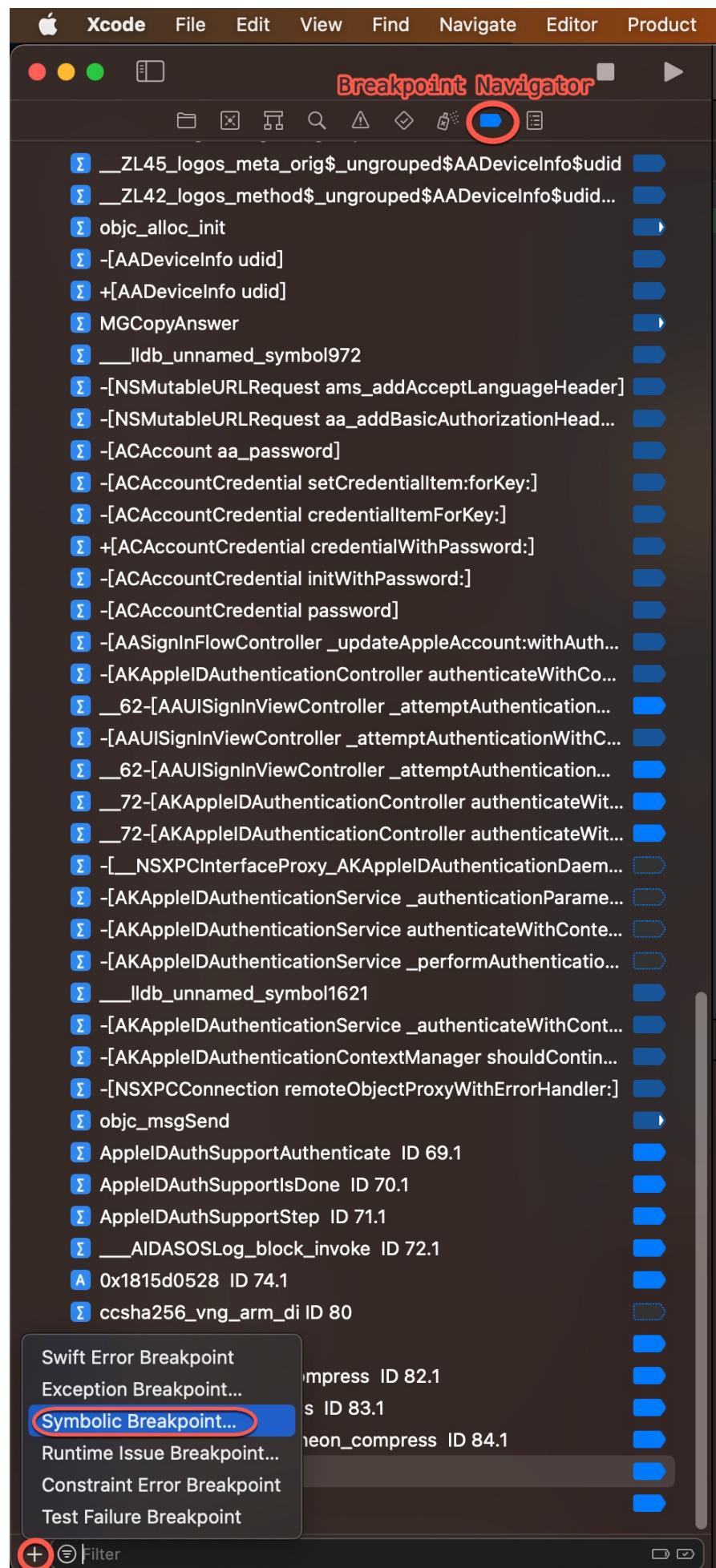
所以：

- sub_1000B0770 函数实际地址 = 二进制的ALSR基地址 + 函数二进制内偏移量 =
0x0000000000c04000 + 0x00000100157BB0 = 0x0000000100d5bbb0

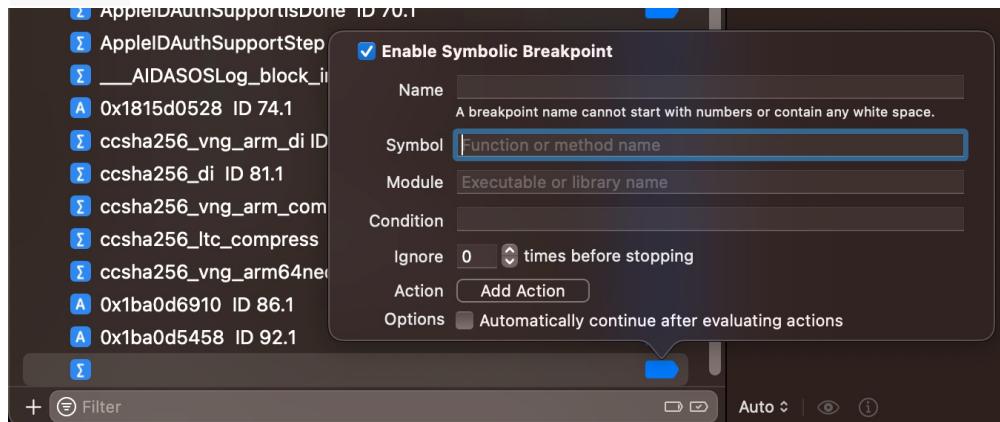
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-06 22:04:54

普通断点

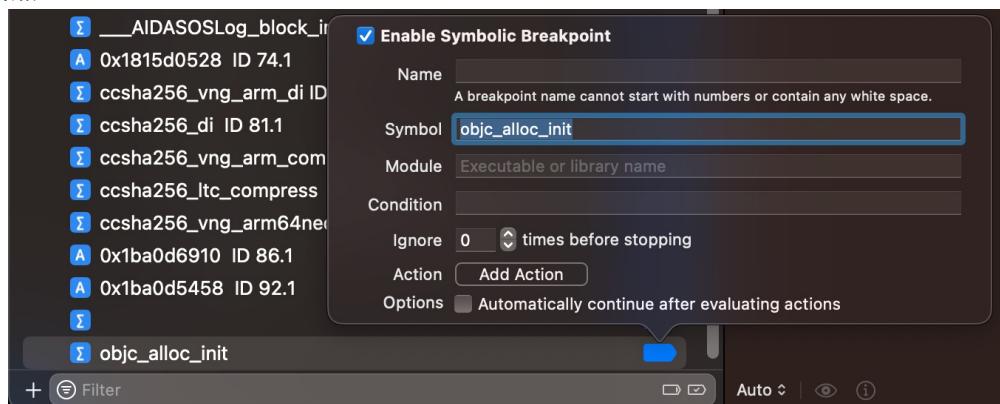
- 加断点的方式
 - 通过函数名
 - 举例
 - C语言函数
 - `objc_alloc_init`
 - `UIApplicationMain`
 - `awemeMain`
 - iOS的ObjC函数
 - `+[NSString stringWithFormat:]`
 - `-[AppDelegate application:didFinishLaunchingWithOptions:]`
 - `-[AWEUserRecommendFollowButton updateWithFollowStatus:followerStatus:preferredTitle:]`
 - `-[AAUISignInController performAuthenticationForAccount:serviceType:inViewController:completion:]`
 - 通过地址
 - 举例
 - `0x0000000100d5bbb0`
 - `0x1830c6a80`
 - 如何加断点
 - Xcode图形界面
 - 通过函数名加断点
 - `Xcode -> Breakpoint Navigator ->点击 左下角的加号 = Create a breakpoint -> Symbolic Breakpoint`
 -



- 出现 符号断点 的弹框页面

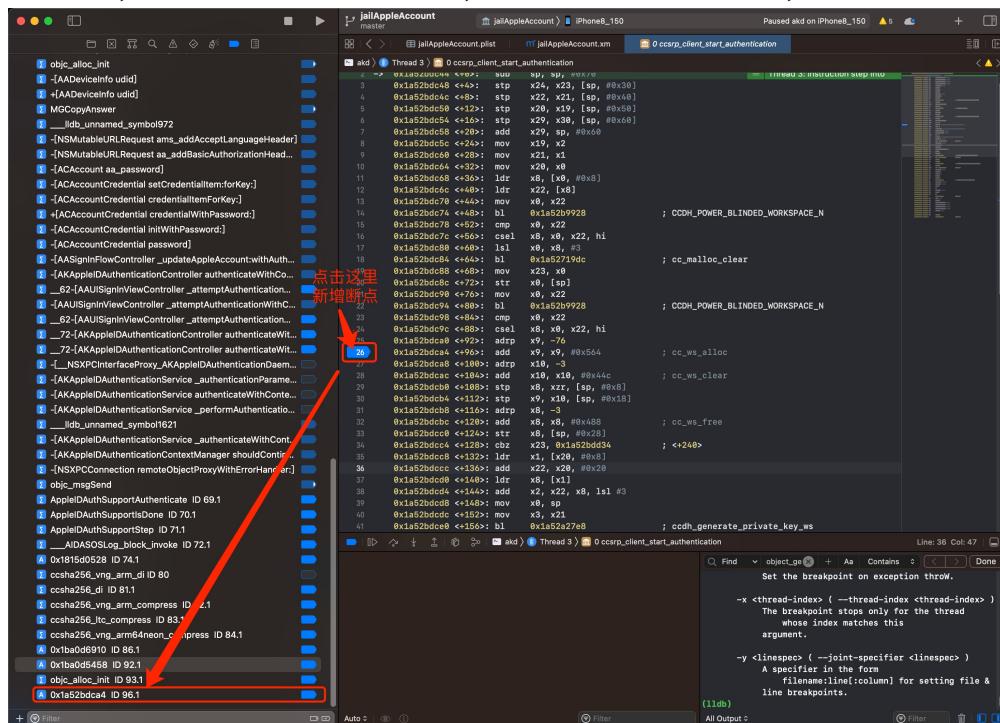


- 在 Symbol 中输入对应的函数名，比如 `objc_alloc_init`，即可添加完毕普通的函数名的断点

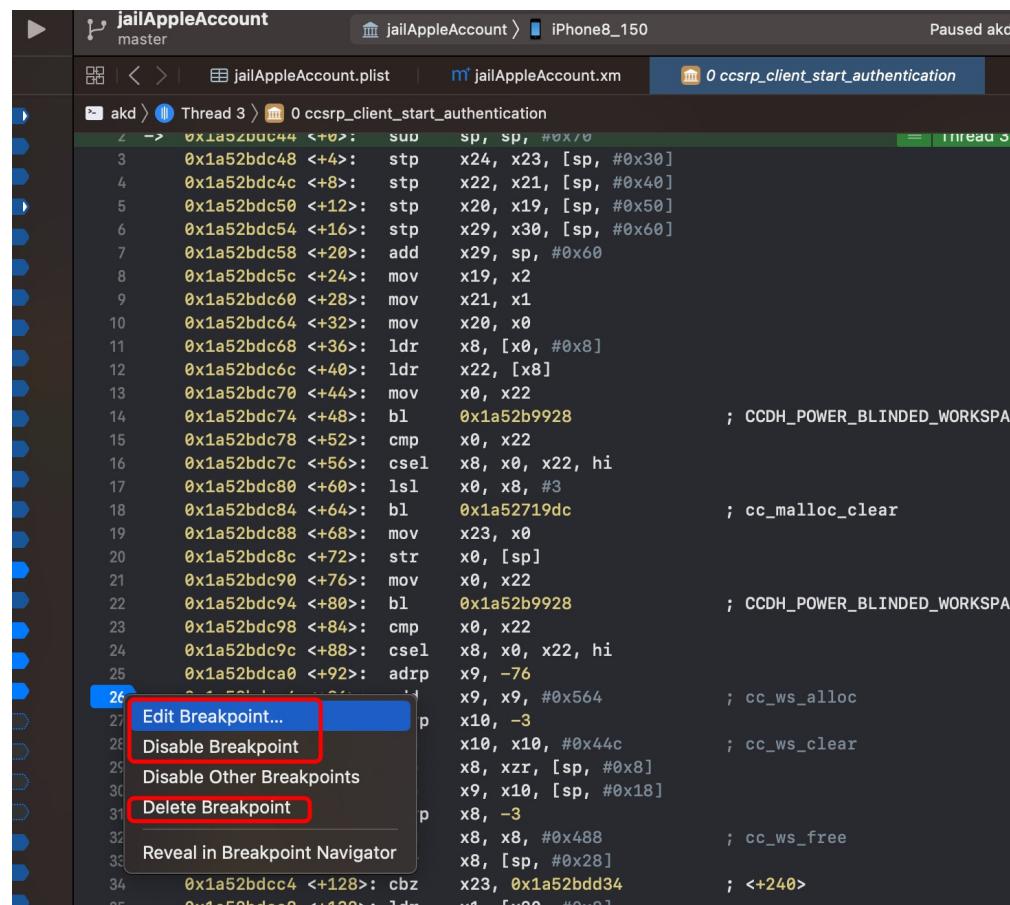


- 给某函数地址=某行汇编代码 加断点

- 对于调试期间，左键点击Xcode调试界面中，汇编代码的所在行的左边，即可新增断点



- 另外，如果需要编辑或删除，右键断点即可操作



- lldb命令行

- 快捷方式添加

```
b FunctionName
b SomeAddress
```

- 举例

```
bobjc_alloc_init
b 0x1830c6a80
```

- 完整命令添加

```
breakpoint set --name FunctionName
breakpoint set --address SomeAddress
```

- 常用命令的缩写

```
br s -n FunctionName
br s -a SomeAddress
```

- 举例

```
breakpoint set --name objc_alloc_init
breakpoint set --name "-[NSString stringByAppendingString:]"
breakpoint set --address 0x1830c6a80
```

- 输出效果

```
(lldb) breakpoint set -a 0x0000000103400010
Breakpoint 12: where = AppleStoreCore`static AppleStoreCore.User.initialize() -> (), address = 0x0000000103400010
```

- 语法详见：[breakpoint的help语法](#)

高级用法

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-06-29 21:33:26

指定模块

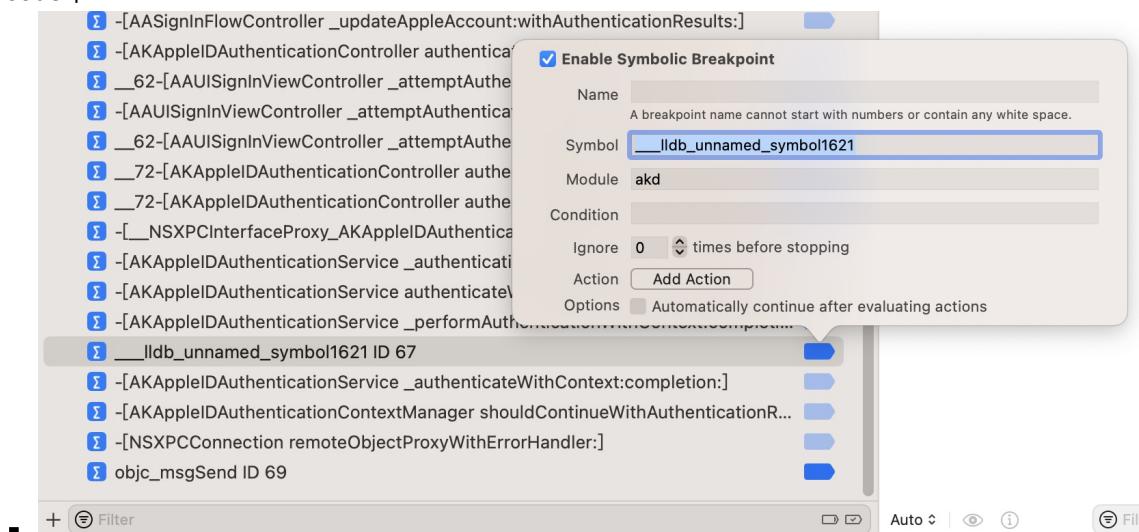
- 概述
 - 加断点时，可以指定要去查找的 库 = Module = 模块 = 二进制 = 库文件
 - Xcode图形界面：设置断点的**Module**=某个 模块名 = 二进制文件名 = 库文件名
 - lldb命令行：加断点的参数时加上 `-s libName == --shlib libName`
- 详解

举例

akd中的__lldb_unnamed_symbol1621

- akd中的__lldb_unnamed_symbol1621

- 参数
 - Symbol : __lldb_unnamed_symbol1621
 - Module : akd
- Xcode中



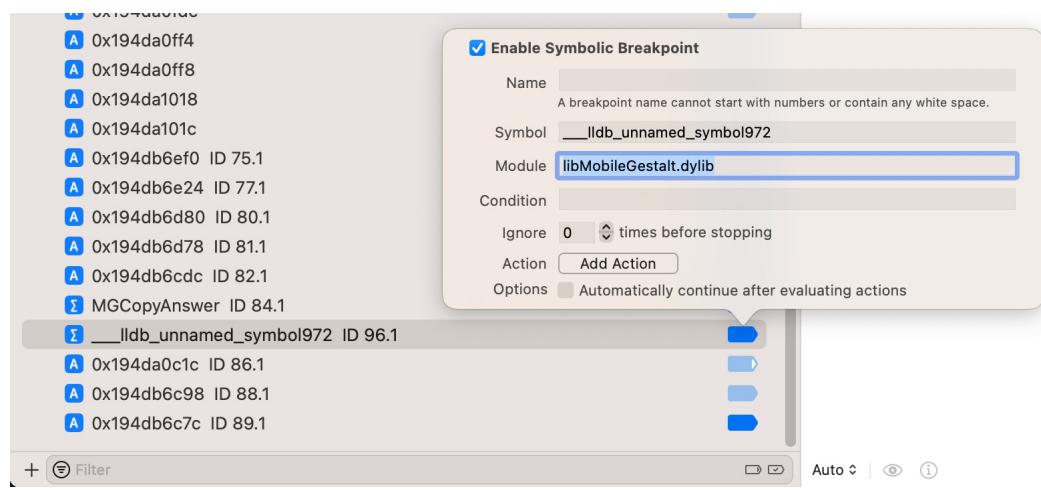
libMobileGestalt.dylib`__lldb_unnamed_symbol1972

给函数：

- libMobileGestalt.dylib`__lldb_unnamed_symbol1972
 - 即： libMobileGestalt.dylib 中的 (lldb无名函数) __lldb_unnamed_symbol1972

加断点：

- Xcode图形界面
 - Xcode-》 Breakpoint Navigator -》 点击 加号 -》 Symbolic Breakpoint -》
 - Name : __lldb_unnamed_symbol1972
 - Module : libMobileGestalt.dylib
 - 如图



- 注意

- 不是直接添加 `libMobileGestalt.dylib`__lldb_unnamed_symbol972`
- 也不能 == 如果手动转换为 `__lldb_unnamed_symbol972$libMobileGestalt`，也是无效的

- lldb命令行

```
br s -n "__lldb_unnamed_symbol972" -s libMobileGestalt.dylib
```

- 其中

- br = breakpoint
- s = set
- -n = --name
- -s = --shlib

- 输出

```
(lldb) br s -n "__lldb_unnamed_symbol972" --shlib libMobileGestalt.dylib
Breakpoint 95: where = libMobileGestalt.dylib`__lldb_unnamed_symbol972, address
s = 0x0000000194db6c40
```

正常触发断点的效果：

- Xcode图形界面

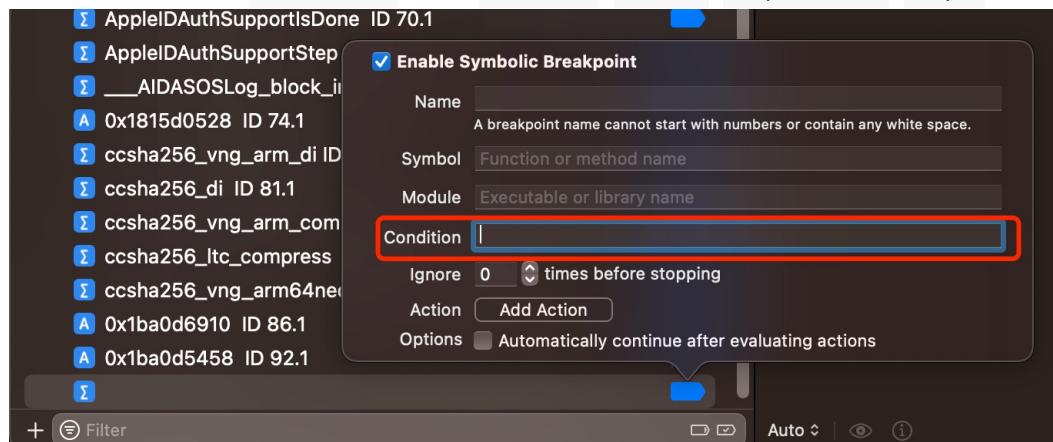
◦

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-11 23:34:19

条件判断

- 带条件判断的断点 = conditional breakpoint = 给普通的断点，加上条件判断
 - 条件判断添加方式

- Xcode图形界面：断点的 Condition 中加上 判断语句 = 表达式 = expression = expr



- lldb命令行：加断点时，加上 -c <expr> == --condition <expr>

- 语法

```
breakpoint set --name <function_name> -c <expr>
```

- 说明

- c <expr>

- 特殊：如果 <expr> 中包含双引号 "，则最外层用单引号 '

- 举例

```
br s -n "objc_alloc_init" -c '(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]'
```

- 参数引用

- \$x0 : 引用寄存器 x0 的值

- 引用值 = 条件判断的断点中引用对应的值的写法

- 寄存器：\$x0、\$x1、\$x2 等等

- 适用于：任何地方，包括普通函数和某个地址(某行汇编代码)

- 参数：第一个参数：\$arg1、第二个参数：\$arg2、第三个参数：\$arg3，等等

- 适用于：普通函数 (才有参数)

更改判断条件

如果之前已给断点加了条件判断，而想要去更改条件：

- Xcode图形界面：右键某断点-> Edit Breakpoint -> 直接修改 condition 的值即可
- lldb命令行
 - 核心思路：用 br modify -c <new_expr>

lldb中更改条件判断

举例说明：

lldb命令行中，之前加了带条件判断的断点：

```
br s -n "objc_alloc_init" -c '(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]'
```

加了后，对应的断点编号是： 3

```
(lldb) br list
...
3: name = 'objc_alloc_init', locations = 1, resolved = 1, hit count = 19
Condition: (bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]

3.1: where = libobjc.A.dylib`objc_alloc_init, address = 0x0000000019cbd3c3c, resolved,
hit count = 19
```

此处：想要修改该断点的condition条件判断

具体写法是：用modify

```
br modify 3 -c '(int)strcmp((char *)class_getName($x0), "AADeviceInfo")==0'
```

修改后，再去查看，即可看到：

```
(lldb) br list
...
3: name = 'objc_alloc_init', locations = 1, resolved = 1, hit count = 26
Condition: (int)strcmp((char *)class_getName($x0), "AADeviceInfo")==0

3.1: where = libobjc.A.dylib`objc_alloc_init, address = 0x0000000019cbd3c3c, resolved,
hit count = 26
```

- condition已经变成新设置的条件了：

- Condition: (int)strcmp((char *)class_getName(\$x0), "AADeviceInfo")==0

The screenshot shows a macOS application window titled "lldb". The title bar includes tabs for "lldb (lldb)" and other open files like ".Account/debug", ".k/fromiPhone8", ".t/LatestBuild", ".Applications", and ".es_app/binary". The main pane displays the following text:

```
-T <thread-name> ( --thread-name <thread-name> )
    The breakpoint stops only for the thread whose thread name matches this argument.

-c <expr> ( --condition <expr> )
    The breakpoint stops only if this condition expression evaluates to true.

-d ( --disable )
    Disable the breakpoint.

-e ( --enable )
    Enable the breakpoint.

-i <count> ( --ignore-count <count> )
    Set the number of times this breakpoint is skipped before stopping.

-o <boolean> ( --one-shot <boolean> )
    The breakpoint is deleted the first time it stop causes a stop.

-q <queue-name> ( --queue-name <queue-name> )
    The breakpoint stops only for threads in the queue whose name is given by this argument.

-t <thread-id> ( --thread-id <thread-id> )
    The breakpoint stops only for the thread whose TID matches this argument. The token 'current' resolves to the current thread's ID.

-x <thread-index> ( --thread-index <thread-index> )
    The breakpoint stops only for the thread whose index matches this argument.

This command takes options and free-form arguments. If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- ' between the end of the command options and the beginning of the arguments.
```

Below this is a list of breakpoints:

```
(lldb) br modify 3 -c '(int)strcmp((char *)class_getName($x0),"AADeviceInfo")==0'
(lldb) br list
Current breakpoints:
1: name = '+[AADeviceInfo uidid]', locations = 1, resolved = 1, hit count = 0
  1.1: where = AppleAccount+[AADeviceInfo(Deprecated) uidid], address = 0x000000019759a558, resolved, hit count = 0

2: name = '-[AADeviceInfo uidid]', locations = 1, resolved = 1, hit count = 0
  2.1: where = AppleAccount`-[AADeviceInfo uidid], address = 0x0000000197599970, resolved, hit count = 0

3: name = 'objc_alloc_init', locations = 1, resolved = 1, hit count = 26
Condition: (int)strcmp((char *)class_getName($x0),"AADeviceInfo")==0
  3.1: where = libobjc.A.dylib`objc_alloc_init, address = 0x000000019cbd3c3c, resolved, hit count = 26
```

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:
2023-07-08 21:36:24

条件判断的断点举例

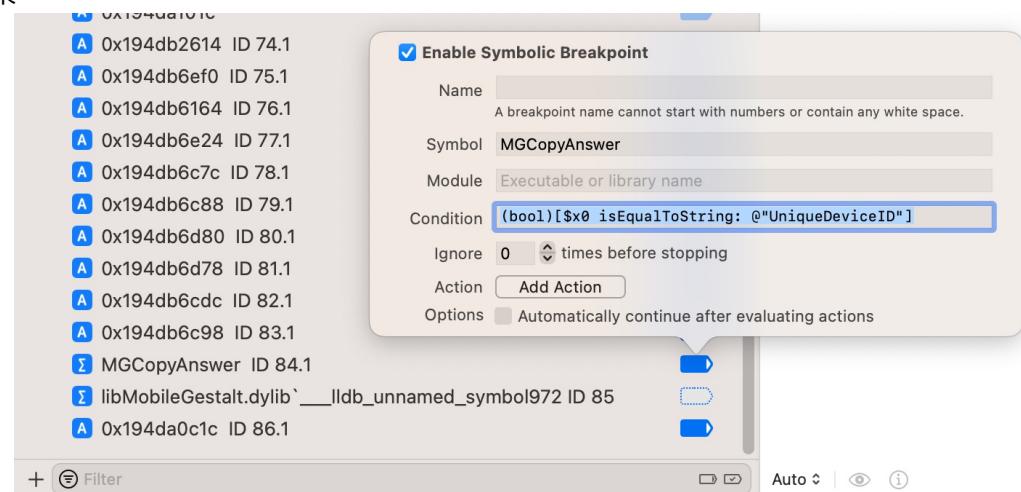
函数

判断 MGCopyAnswer 的传入参数是否是字符串 UniqueDeviceID

- 判断 MGCopyAnswer 的传入参数是否是字符串 UniqueDeviceID
 - 条件判断表达式：

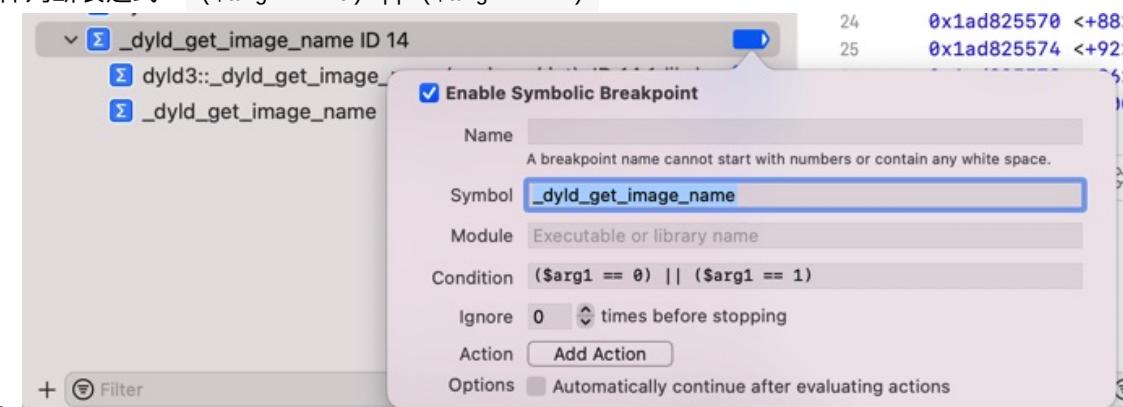
```
(bool)[$x0 isEqualToString: @"UniqueDeviceID"]
```

■ 效果

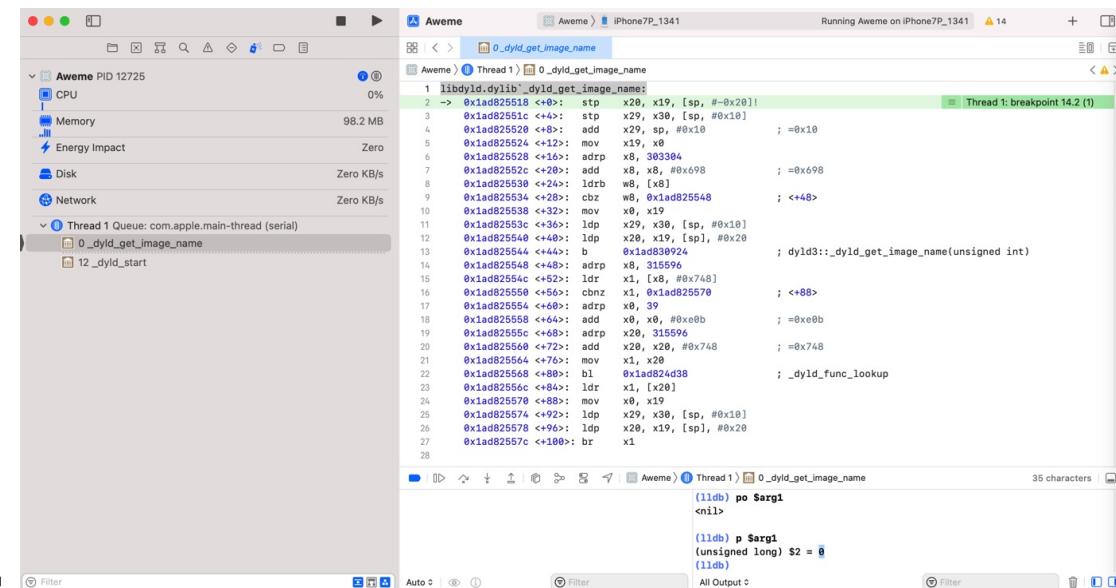


判断 _dyld_get_image_name 的第一个参数是 0 或 1

- 判断 _dyld_get_image_name 的第一个参数是 0 或 1
 - 条件判断表达式： `($arg1 == 0) || ($arg1 == 1)`

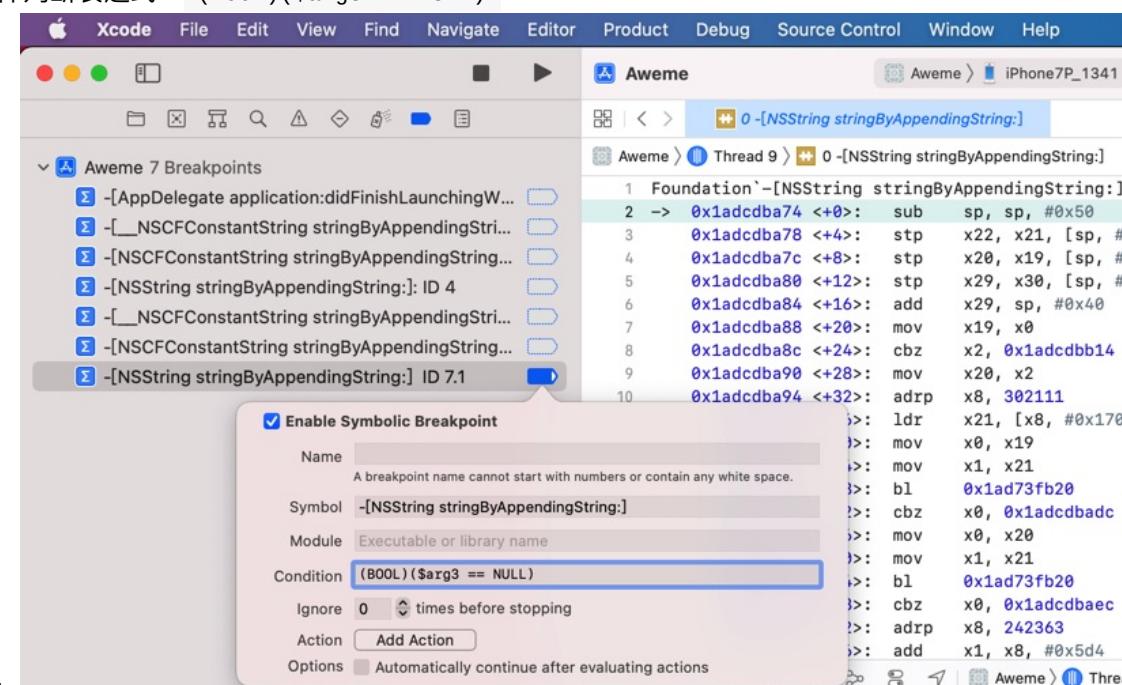


- 断点生效的效果



判断 -[NSString stringByAppendingString:] 传入参数是否为空

- 判断 `-[NSString stringByAppendingString:]` 传入参数是否为空
 - 注：此处是 `objc_msgSend`，所以第一个参数是 `id = self`，第二个参数是 `selector`，第三个参数才是真正的传入的值
 - 条件判断表达式： `(BOOL)($arg3 == NULL)`



判断 `objc_alloc_init` 输入参数是类 `AADeviceInfo` 的 class

- 判断 `objc_alloc_init` 输入参数是类 `AADeviceInfo` 的 class
 - 条件过滤表达式写法
 - 三种写法

```
■ NSStringFromClass
(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]
```

- class_getName

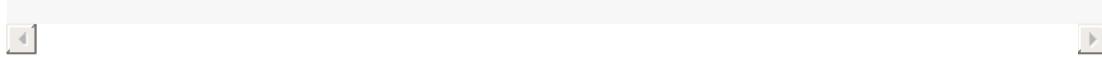

```
(int)strcmp((char *)class_getName($x0), "AADeviceInfo") == 0
```

- object_getClassName


```
(int)strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0
```

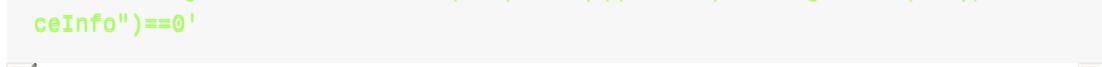
- => lldb命令行中
 - NSStringFromClass


```
br s -n "objc_alloc_init" -c '(bool)[NSStringFromClass($x0) isEqualToString:@"AADeviceInfo"]'
```



 - class_getName

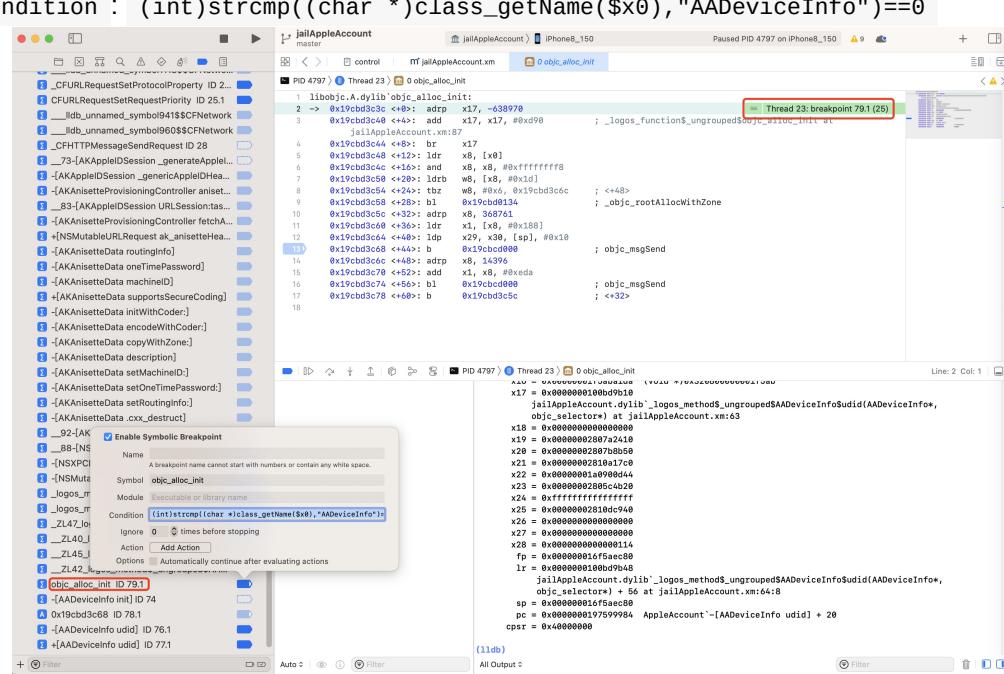

```
br s -n "objc_alloc_init" -c '(int)strcmp((char *)class_getName($x0), "AADeviceInfo") == 0'
```



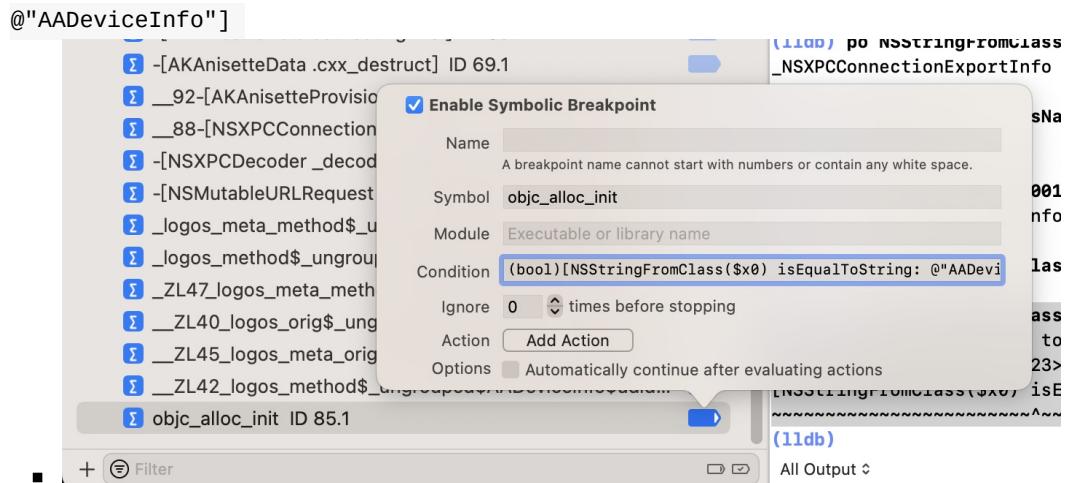
 - object_getClassName


```
br s -n "objc_alloc_init" -c '(int)strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0'
```



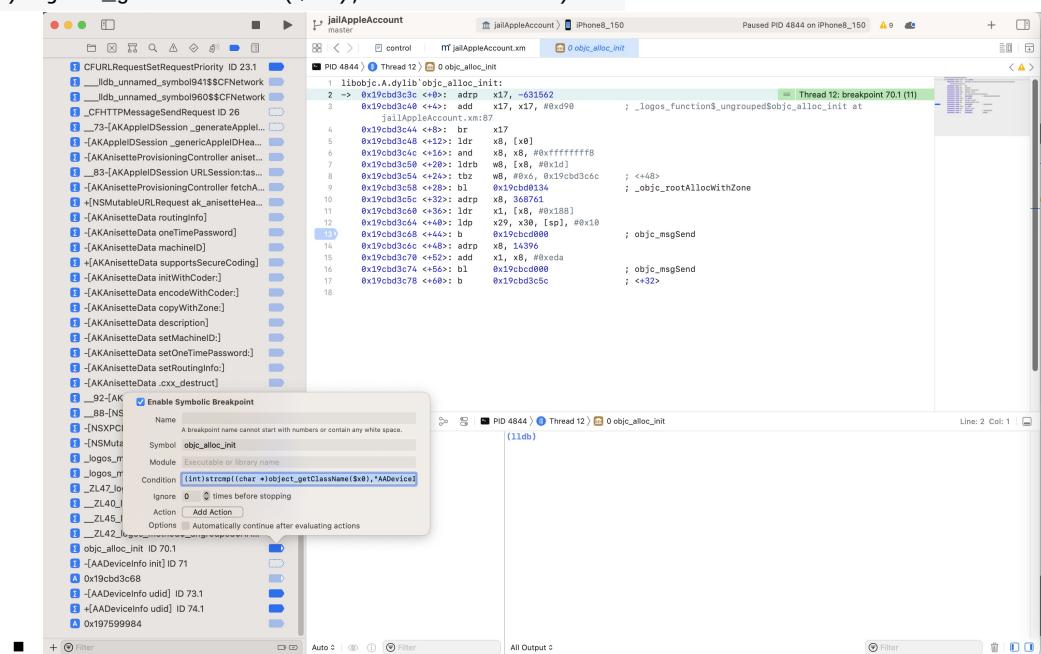
- => Xcode图形界面中
 - class_getName
 - Condition : (int)strcmp((char *)class_getName(\$x0), "AADeviceInfo") == 0

 - NSStringFromClass
 - Condition : (bool)[NSStringFromClass(\$x0) isEqualToString:@"AADeviceInfo"]



■ object_getClassName

- Condition : `(int)strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0`



某行汇编代码

`libMobileGestalt.dylib` 的 `__lldb_unnamed_symbol317` 中的 +28 行汇编代码

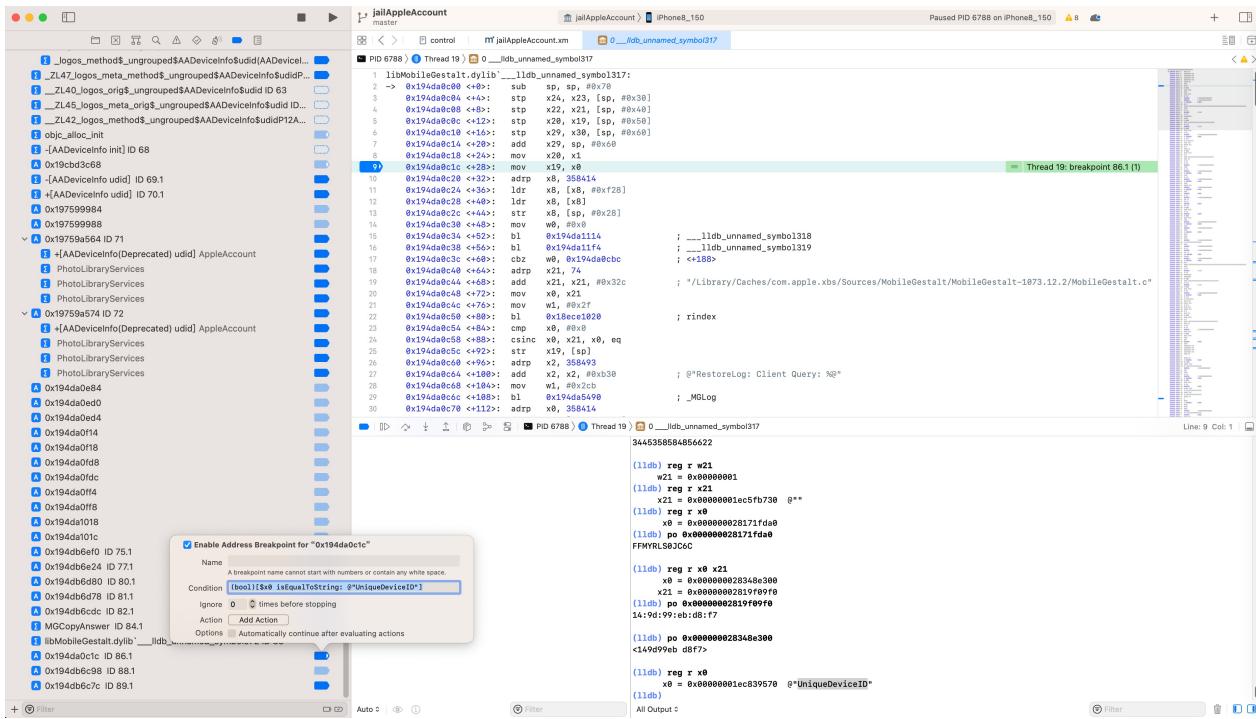
给 `libMobileGestalt.dylib` 的 `__lldb_unnamed_symbol317` 中的 +28 行汇编代码

```
0x194da0c1c +28:    mov     x19, x0
```

加条件判断断点：

```
(bool)[NSStringFromClass($x0) isEqualToString: @"UniqueDeviceID"]
```

断点触发效果：



objc_alloc_init 的 +44 行的 objc_msgSend 行的汇编代码

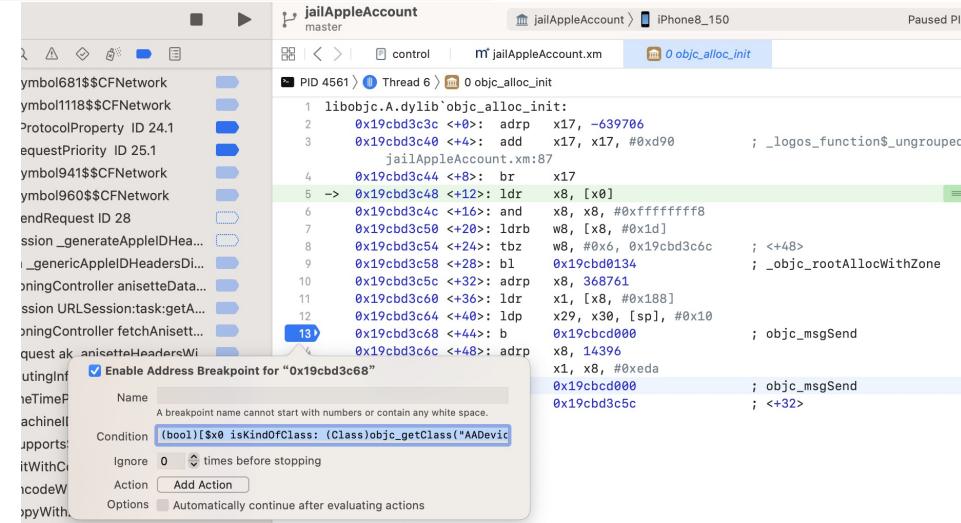
- objc_alloc_init 的 +44 行的 objc_msgSend 行的汇编代码

- 判断输入参数是类 AADeviceInfo 的 Instance 实例

- 条件判断表达式

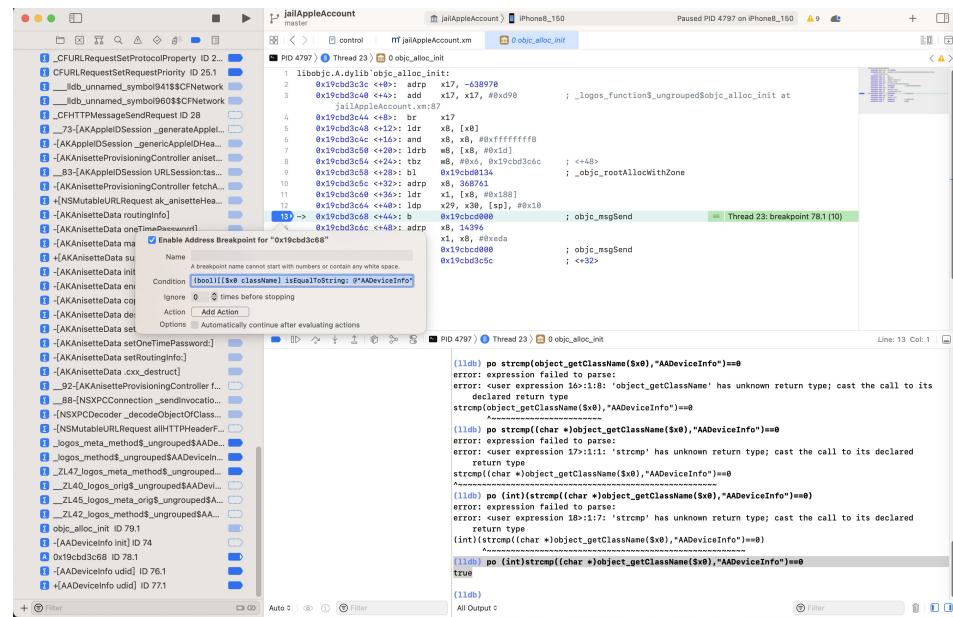
- objc_getClass

- Condiction : `(bool)[$x0 isKindOfClass:(Class)objc_getClass("AADeviceInfo")]`



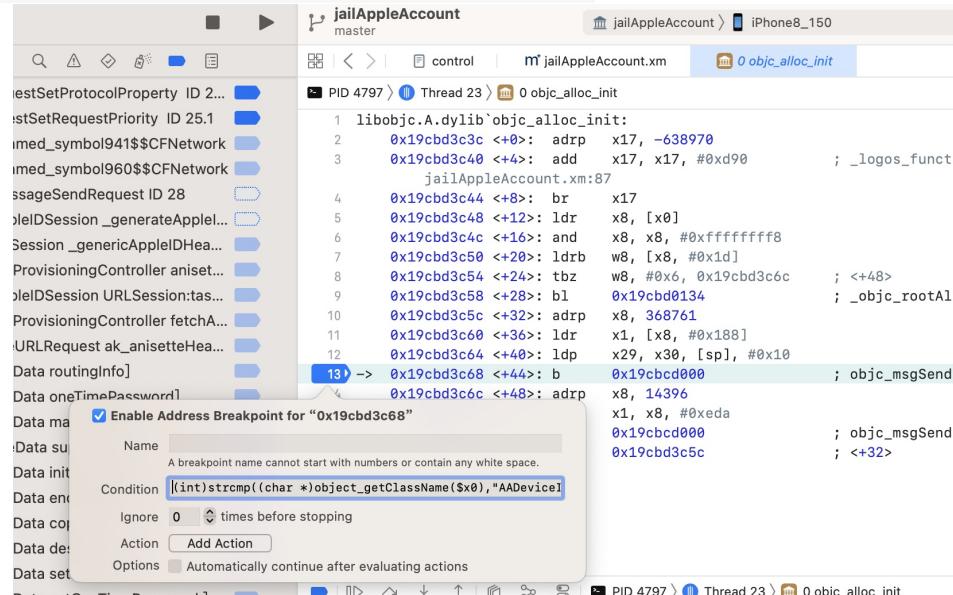
- className

- Condiction : `(bool)[[$x0 className] isEqualToString:@"AADeviceInfo"]`

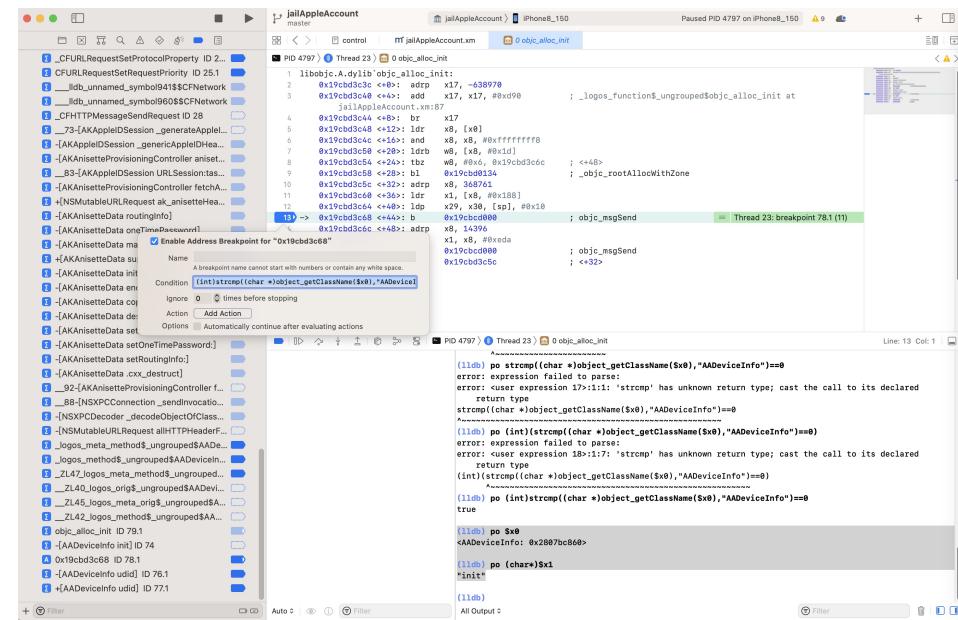


■ object_getClassName

- Condition : (int)strcmp((char *)object_getClassName(\$x0), "AADeviceInfo") == 0



■ 触发效果



88 - [NSXPConnection _sendInvocation:orArguments:count:methodSignature:selector:withProxy:]_block_invoke_3 中的 +208 行汇编代码

- 汇编代码

```
Foundation`__88 [NSXPConnection _sendInvocation:orArguments:count:methodSignature:selector:withProxy:]_block_invoke_3:
...
0x1830c6a6c <+188>: ldr    x3, [x19, #0x38]
0x1830c6a70 <+192>: ldp    x0, x4, [x19, #0x20]
0x1830c6a74 <+196>: adrp   x8, 297427
0x1830c6a78 <+200>: add    x1, x8, #0x5c6           ; =0x5c6
0x1830c6a7c <+204>: mov    x2, x21
-> 0x1830c6a80 <+208>: bl     0x1815b9dc8          ; symbol stub for: objc_ms
gSend
```

- 通过调试看到值

```
(lldb) reg r x0
x0 = 0x0000000283265c20
(lldb) po 0x0000000283265c20
• NSXPConnection: 0x283265c20 connection to service with pid 125 named com.apple.ak.anisette.xpc
```

- 想要加上条件判断实现：
 - x0中的NSXPConnection的属性serviceName值是：com.apple.ak.anisette.xpc
- 研究条件判断写法的过程
 - lldb中的po调试
 - po ((NSXPConnection*)\$x0).serviceName
 - iOS的ObjC的写法
 - [(NSXPConnection*)\$x0 serviceName]
 - 类似C语言的写法

- `(NSXPConnection*)$x0).serviceName`

- 条件判断的表达式写法

```
[[((NSXPConnection*)$x0 serviceName) isEqualToString: @"com.apple.ak.anisette.xpc"]
```

◦

- 断点生效效果

◦

- 但是

- 另外2种C语言的写法:

- `strcmp` 写法

```
(bool)((int)strcmp(((NSXPConnection*)$x0).serviceName, "com.apple.ak.anisette.xpc") == 0)
```



- `strstr` 写法

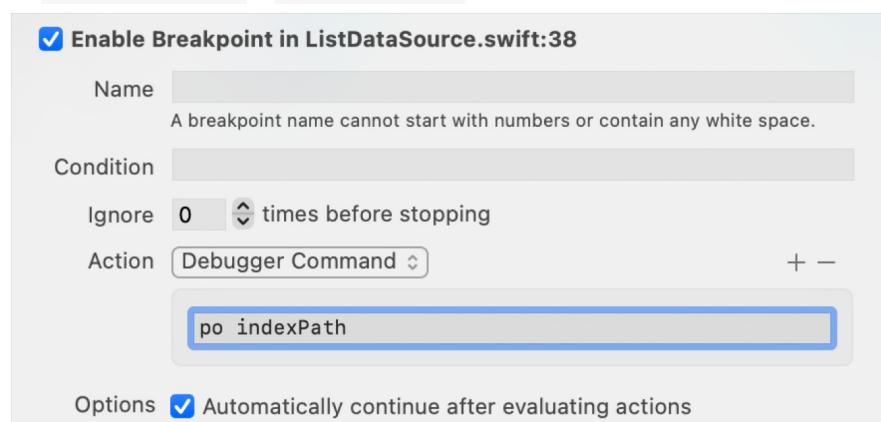
```
strstr((const char*)(((NSXPConnection*)$x0).serviceName), "com.apple.ak.anis  
ette.xpc") != NULL
```

- -》 最终无效 -》 无法触发条件断点

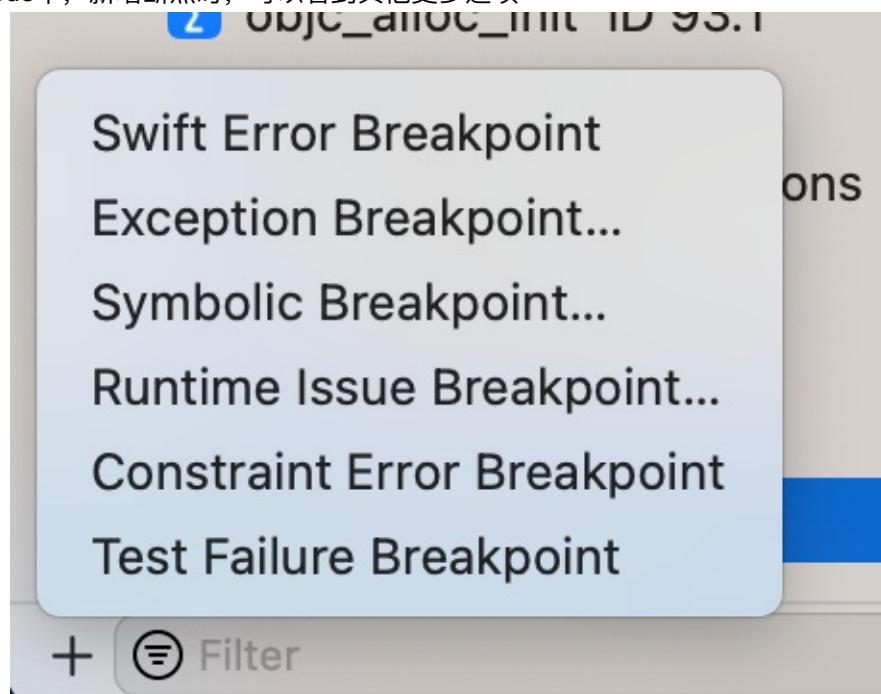
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:
2023-07-10 16:53:00

Xcode图形界面中的断点

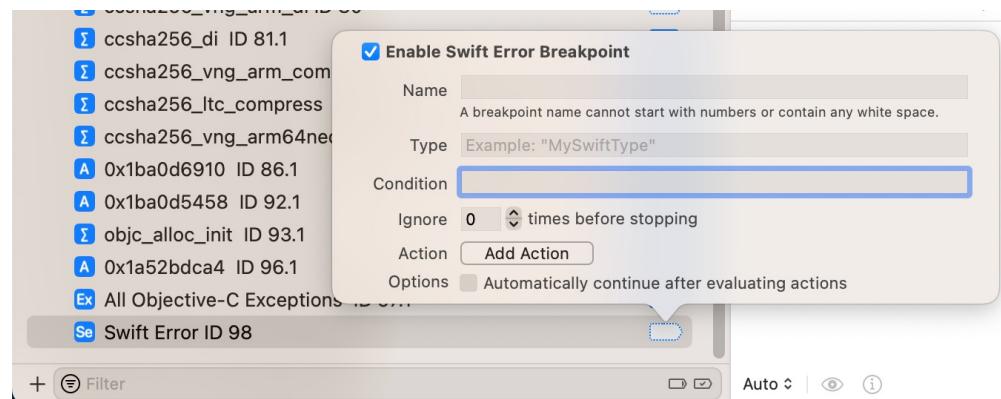
- Xcode特有的
 - 当断点发生时，支持更多的设置
 - ignore
 - 忽略几次之后，再触发断点
 - action
 - 当触发断点时，执行额外某些动作
 - 举例
 - 触发某个音乐文件，比如响铃一声之类的
 - Action= Debug Command 为 po indexPath 去打印当前索引值



- 全局的异常类的断点
 - Xcode中，新增断点时，可以看到其他更多选项

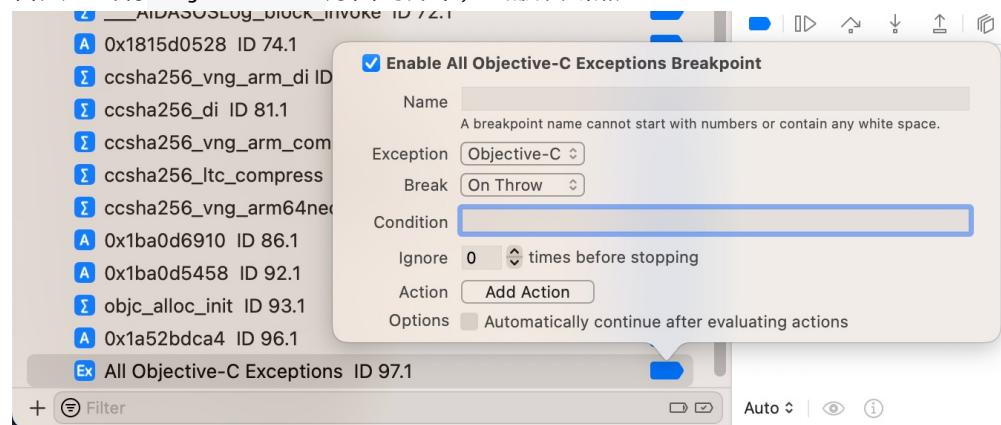


- 包括
 - Swift Error Breakpoint
 - 含义：当有 Swift 代码方面的错误，会触发断点



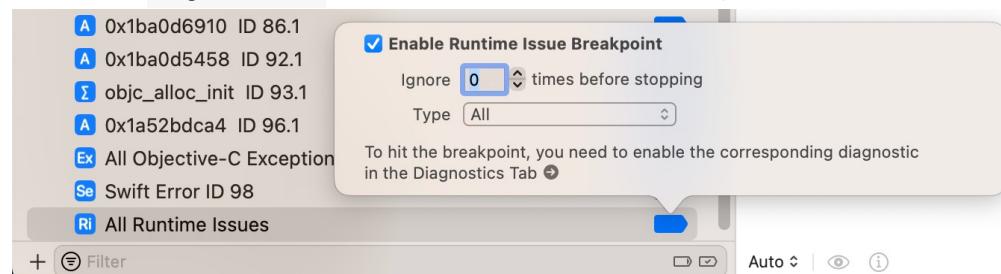
■ Exception Breakpoint = Objective-C Exceptions Breakpoint

- 含义：当有 Objective-C 方面的异常，会触发断点



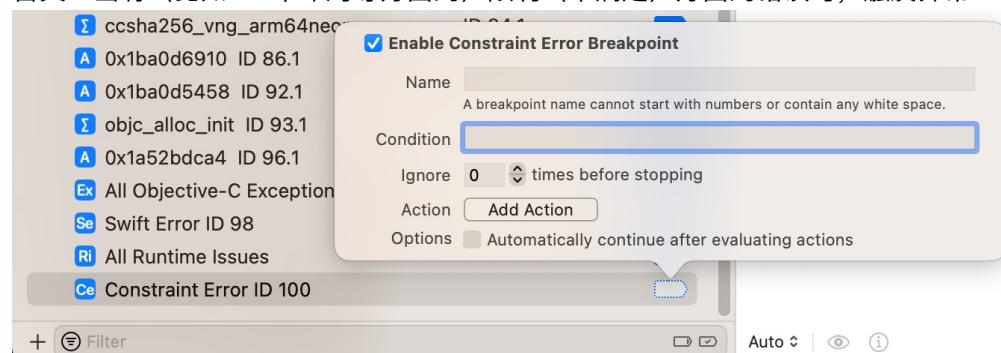
■ Runtime Issue Breakpoint

- 含义：当有 Objective-C 的运行时（Runtime）方面的问题，会触发断点



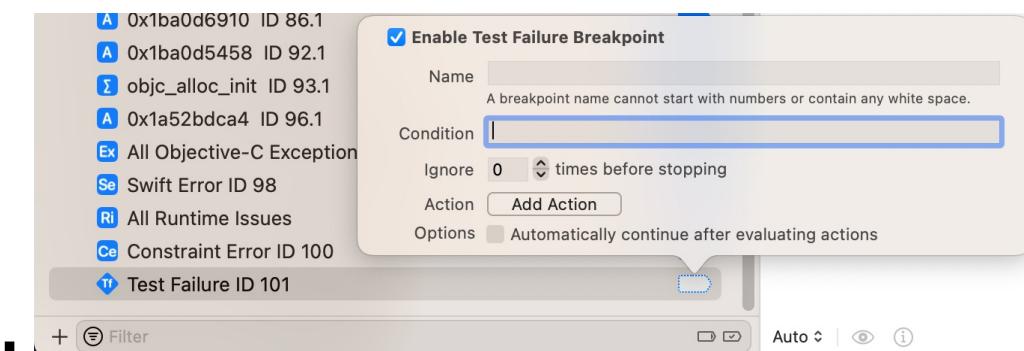
■ Constraint Error Breakpoint

- 含义：当有（比如iOS中布局等方面）限制（不满足）方面的错误时，触发异常？



■ Test Failure Breakpoint

- 含义：当项目代码中有测试代码时，且运行时测试代码运行发现条件不满足，测试失败时，触发断点



crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:
2023-07-11 23:36:41

lldb命令行中的断点

- lldb命令行中

- 。有2处地方=场景

- lldb+debugserver 中的: lldb命令行

```
Process 9971 exited with status = 0 (0x00000000)
(lldb) process connect connect://192.168.2.13:20235
(lldb) bt
error: invalid process
(lldb) exit
cristian@liricifando-MacBook-Pro: ~ > lldb
The "write" command has been loaded and is ready for use.
(lldb) process connect connect://192.168.2.13:20235
Process 10065 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = signal SIGSTOP
  Frame #0: 0x00000001bd808b70 libsystem_kernel.dylib mach_msg_trap + 8
libsystem_kernel.dylib mach_msg_trap:
-> 0x1bd808b70 =>>> ret

libsystem_kernel.dylib mach_msg_overwrite_trap:
 0x1bd808b74 <-> mov x16, #-0x20
 0x1bd808b78 <-> svc #0x80
 0x1bd808b7c <-> ret
Target 0: (Preferences) stopped.
(lldb) bt
* thread #1, queue = 'com.apple.main-thread', stop reason = signal SIGSTOP
* frame #0: 0x00000001bd808b70 libsystem_kernel.dylib mach_msg_trap + 8
  frame #1: 0x00000001bd809194 libsystem_kernel.dylib mach_msg + 72
  frame #2: 0x0000000184232a00 CoreFoundation _CFRunLoopServiceMachinePort + 368
  frame #3: 0x0000000184236b00 CoreFoundation _CFRunLoopRun + 1184
  frame #4: 0x0000000184236b90 CoreFoundation _CFRunLoopRunSpecific + 572
  frame #5: 0x0000000184236b90 GraphicsServices GSRunOnMainThread + 160
  frame #6: 0x00000000918647f01 UIKitCore _UIApplicationRun + 1080
  frame #7: 0x00000000918647f20 UIKitCore UIApplicationMain + 2060
  frame #8: 0x0000000100d4dd4 Preferences lldb_unnamed_symbol1233$preferences + 56
  frame #9: 0x0000000091841980 dyld_start + 444
(lldb) br s -n "[NSMutableDictionary objectForKey:@"Properties"]"
Breakpoint 1: where = StoreServices-[NSMutableDictionary objectForKey:@"Properties"], address = 0x0000000196127e9c
(lldb) c
Process 10065 resuming
(lldb) 
```

- Xcode 中右下角中的：lldb 命令行窗口

- ### 3 加断点

■ 普通断占

■ 通过函数名添加

— 10 —

hr s =n FunctionName

■ 举例

■ iOS的ObjC函数

```
breakpoint set -n "-[AAUISignInController _performAuthenticationForAccount:serviceType:inViewController:completion:]"
```



- 或
- `s = symbol`

```
br s -n FunctionName
```

- 通过地址添加

```
br s -a FunctionAddress
```

- 完整写法

```
breakpoint set --address FunctionAddress
```

- 举例

```
(lldb) br s -a 0x0000000100d5bbbb0
Breakpoint 4: where = RzGame`__lldb_unnamed_symbol1275$$RzGame + 685
120, address = 0x0000000100d5bbbb0
```

- 查看已有断点 (的列表)

```
br list
```

- 注: `br list = breakpoint list`

Ildb中breakpoint的帮助语法

- b = breakpoint的缩写

```
(lldb) help
Debugger commands:

...
breakpoint      -- Commands for operating on breakpoints (see 'help b' for shorthand.)
...
Current command abbreviations (type 'help command alias' for more info):
...
b              -- Set a breakpoint using one of several shorthand formats.
...
```

b

help b

```
(lldb) help b
Set a breakpoint using one of several shorthand formats. Expects 'raw' input (see 'help raw-input')

Syntax:
__regexp-break <filename : linenumber : column>
    main.c:12:21           // Break at line 12 and column 21 of main.c

__regexp-break <filename : linenumber>
    main.c:12            // Break at line 12 of main.c

__regexp-break <linenumber>
    12                  // Break at line 12 of current file

__regexp-break 0x address
    0x1234000           // Break at address 0x1234000

__regexp-break <name>
    main                // Break in 'main' after the prologue

__regexp-break &<name>
    main                // Break at first instruction in 'main'

__regexp-break <module><name>
    libc.so malloc       // Break in 'malloc' from 'libc.so'

__regexp-break / source-regex /
    /break here/         // Break on source lines in current file
                           // containing text 'break here'.

'b' is an abbreviation for '__regexp-break'
```

breakpoint

help breakpoint

```
→ ~ lldb
(lldb) help breakpoint
Commands for operating on breakpoints (see 'help b' for shorthand.)

Syntax: breakpoint <subcommand> [<command-options>]

The following subcommands are supported:

    clear    -- Delete or disable breakpoints matching the specified source file and line.
    command   -- Commands for adding, removing and listing LLDB commands executed when a breakpoint is hit.
    delete    -- Delete the specified breakpoint(s). If no breakpoints are specified, delete them all.
    disable   -- Disable the specified breakpoint(s) without deleting them. If none are specified, disable all breakpoints.
    enable    -- Enable the specified disabled breakpoint(s). If no breakpoints are specified, enable all of them.
    list      -- List some or all breakpoints at configurable levels of detail.
    modify    -- Modify the options on a breakpoint or set of breakpoints in the executable. If no breakpoint is specified, acts on the last created breakpoint. With the exception
                of -e, -d and -i, passing an empty argument clears the modification.
    name      -- Commands to manage breakpoint names
    read      -- Read and set the breakpoints previously saved to a file with "breakpoint write".
    set       -- Sets a breakpoint or set of breakpoints in the executable.
    write     -- Write the breakpoints listed to a file that can be read in with "breakpoint read".
                If given no arguments, writes all breakpoints.

For more help on any particular subcommand, type 'help <command> <subcommand>'.
```

help breakpoint set

```
(lldb) help breakpoint set
Sets a breakpoint or set of breakpoints in the executable.

Syntax: breakpoint set <cmd-options>

Command Options Usage:
breakpoint set [-DHd] [-l <linenum> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-R <address>] [-N <breakpoint-name>] [-u <column>] [-f <filename>] [-m <boolean>] [-s <shlib-name>] [-K <boolean>]
breakpoint set [-DHd] -a <address-expression> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-N <breakpoint-name>] [-s <shlib-name>]
breakpoint set [-DHd] -n <function-name> [-G <boolean>] [-C <command>] [-c <expr>] [-
```

```

i count] [-o boolean] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-R <address>] [-N <breakpoint-name>] [-f <filename>] [-L <source-language>]
] [-s <shlib-name>] [-K <boolean>]

breakpoint set [-DHd] -F <fullname> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-R <address>] [-N <breakpoint-name>] [-f <filename>] [-L <source-language>] [-s <shlib-name>] [-K <boolean>]

breakpoint set [-DHd] -S <selector> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-R <address>] [-N <breakpoint-name>] [-f <filename>] [-L <source-language>] [-s <shlib-name>] [-K <boolean>]

breakpoint set [-DHd] -M <method> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-R <address>] [-N <breakpoint-name>] [-f <filename>] [-L <source-language>] [-s <shlib-name>] [-K <boolean>]

breakpoint set [-DHd] -r <regular-expression> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-R <address>] [-N <breakpoint-name>] [-f <filename>] [-L <source-language>] [-s <shlib-name>] [-K <boolean>]

breakpoint set [-DHd] -b <function-name> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-R <address>] [-N <breakpoint-name>] [-f <filename>] [-L <source-language>] [-s <shlib-name>] [-K <boolean>]

breakpoint set [-ADHd] -p <regular-expression> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-N <breakpoint-name>] [-f <filename>] [-m <boolean>] [-s <shlib-name>] [-X <function-name>]

breakpoint set [-DHd] -E <source-language> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-N <breakpoint-name>] [-O <type-name>] [-h <boolean>] [-w <boolean>]

breakpoint set [-DHd] -P <python-class> [-k <none>] [-v <none>] [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-N <breakpoint-name>] [-f <filename>] [-s <shlib-name>]

breakpoint set [-DHd] -y <linespec> [-G <boolean>] [-C <command>] [-c <expr>] [-i <count>] [-o <boolean>] [-q <queue-name>] [-t <thread-id>] [-x <thread-index>] [-T <thread-name>] [-R <address>] [-N <breakpoint-name>] [-m <boolean>] [-s <shlib-name>] [-K <boolean>]

-A ( --all-files )
    All files are searched for source pattern matches.

-C <command> ( --command <command> )
    A command to run when the breakpoint is hit, can be provided more than once, the commands will get run in order left to right.

-D ( --dummy-breakpoints )
    Act on Dummy breakpoints - i.e. breakpoints set before a file is provided, which prime new targets.

-E <source-language> ( --language-exception <source-language> )
    Set the breakpoint on exceptions thrown by the specified language (without options, on throw but not catch.)

-F <fullname> ( --fullname <fullname> )
    Set the breakpoint by fully qualified function names. For C++ this means na

```

mespaces and all arguments, and **f**or Objective-C this means a full **f**unction prototype with **c**lass
and selector. Can be repeated multiple **t**imes to make one breakpoint **f**or multiple names.

-G <boolean> (**--auto-continue** <boolean>)
The breakpoint will auto-continue after running its commands.

-H (**--hardware**)
Require the breakpoint to use hardware breakpoints.

-K <boolean> (**--skip-prologue** <boolean>)
skip the prologue **i**f the breakpoint is at the beginning of a function. If not **s**et the target.skip-prologue setting is used.

-L <source-language> (**--language** <source-language>)
Specifies the Language to use when interpreting the breakpoint's expression (note: currently only implemented for setting breakpoints on identifiers). If not set the target.language setting is used.

-M <method> (**--method** <method>)
Set the breakpoint by C++ method names. Can be repeated multiple times to make one breakpoint for multiple methods.

-N <breakpoint-name> (**--breakpoint-name** <breakpoint-name>)
Adds this to the list of names for this breakpoint.

-O <type-name> (**--exception-typename** <type-name>)
The breakpoint will only stop if an exception Object of this type is thrown. Can be repeated multiple times to stop for multiple object types. If you just specify the type's base name it will match against that type in all modules, or you can specify the full type name including modules. Other submatches are not supported at present. Only supported for Swift at present.

-P <python-class> (**--script-class** <python-class>)
The name of the class that will manage a scripted breakpoint.

-R <address> (**--address-slide** <address>)
Add the specified offset to whatever address(es) the breakpoint resolves to. At present this applies the offset directly as given, and doesn't try to align it to instruction boundaries.

-S <selector> (**--selector** <selector>)
Set the breakpoint by ObjC selector name. Can be repeated multiple times to make one breakpoint for multiple Selectors.

-T <thread-name> (**--thread-name** <thread-name>)
The breakpoint stops only for the thread whose thread name matches this argument.

-X <function-name> (**--source-regexp-function** <function-name>)
When used with '-p' limits the source regex to source contained in the named functions. Can be repeated multiple times.

```

-a <address-expression> ( --address <address-expression> )
    Set the breakpoint at the specified address. If the address maps uniquely
    to a particular binary, then the address will be converted to a fileaddress, so that th
    e
        breakpoint will track that binary+offset no matter where the binary eventua
        lly loads. Alternately, if you also specify the module - with the -s option - then the
        address
            will be treated as a file address in that module, and resolved accordingly.
        Again, this will allow lldb to track that offset on subsequent reloads. The module n
        eed not
            have been loaded at the time you specify this breakpoint, and will get reso
        lved when the module is loaded.

-b <function-name> ( --basename <function-name> )
    Set the breakpoint by function basename (C++ namespaces and arguments will
    be ignored). Can be repeated multiple times to make one breakpoint for multiple symbol
    s.

-c <expr> ( --condition <expr> )
    The breakpoint stops only if this condition expression evaluates to true.

-d ( --disable )
    Disable the breakpoint.

-f <filename> ( --file <filename> )
    Specifies the source file in which to set this breakpoint. Note, by defaul
    t lldb only looks for files that are #included if they use the standard include file ex
    tensions.
        To set breakpoints on .c/.cpp/.m/.mm files that are #included, set target.i
    nline-breakpoint-strategy to always.

-h <boolean> ( --on-catch <boolean> )
    Set the breakpoint on exception catch.

-i <count> ( --ignore-count <count> )
    Set the number of times this breakpoint is skipped before stopping.

-k <none> ( --structured-data-key <none> )
    The key for a key/value pair passed to the implementation of a scripted bre
    akpoint. Pairs can be specified more than once.

-l <linenum> ( --line <linenum> )
    Specifies the line number on which to set this breakpoint.

-m <boolean> ( --move-to-nearest-code <boolean> )
    Move breakpoints to nearest code. If not set the target.move-to-nearest-cod
    e setting is used.

-n <function-name> ( --name <function-name> )
    Set the breakpoint by function name. Can be repeated multiple times to mak
    e one breakpoint for multiple names.

-o <boolean> ( --one-shot <boolean> )
    The breakpoint is deleted the first time it stop causes a stop.

-p <regular-expression> ( --source-pattern-regexp <regular-expression> )

```

Set the breakpoint by specifying a regular expression which is matched against the source text in a source file or files specified with the -f can be specified more than once. If no source files are specified, uses the current default source file. If you want to match against all source files, pass the --all-files option.

-q <queue-name> (--queue-name <queue-name>)
The breakpoint stops only for threads in the queue whose name is given by this argument.

-r <regular-expression> (--func-regex <regular-expression>)
Set the breakpoint by function name, evaluating a regular-expression to find the function name(s).

-s <shlib-name> (--shlib <shlib-name>)
Set the breakpoint only in this shared library. Can repeat this option multiple times to specify multiple shared libraries.

-t <thread-id> (--thread-id <thread-id>)
The breakpoint stops only for the thread whose TID matches this argument. The token 'current' resolves to the current thread's ID.

-u <column> (--column <column>)
Specifies the column number on which to set this breakpoint.

-v <none> (--structured-data-value <none>)
The value for the previous key in the pair passed to the implementation of a scripted breakpoint. Pairs can be specified more than once.

-w <boolean> (--on-throw <boolean>)
Set the breakpoint on exception throw.

-x <thread-index> (--thread-index <thread-index>)
The breakpoint stops only for the thread whose index matches this argument.

-y <linespec> (--joint-specifier <linespec>)
A specifier in the form filename:line[:column] for setting file & line breakpoints.

help breakpoint delete

```
(lldb) help breakpoint delete
Delete the specified breakpoint(s). If no breakpoints are specified, delete them all.

Syntax: breakpoint delete <cmd-options> [<breakpt-id | breakpt-id-list>]

Command Options Usage:
breakpoint delete [-Ddf] [<breakpt-id | breakpt-id-list>]

-D ( --dummy-breakpoints )
Delete Dummy breakpoints - i.e. breakpoints set before a file is provided,
which prime new targets.
```

```
-d ( --disabled )
    Delete all breakpoints which are currently disabled. When using the disabled option any breakpoints listed on the command line are EXCLUDED from deletion.

-f ( --force )
    Delete all breakpoints without querying for confirmation.

This command takes options and free-form arguments. If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- ' between the end of the
command options and the beginning of the arguments.
```

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:
2023-07-10 17:36:15

常见问题

条件判断断点

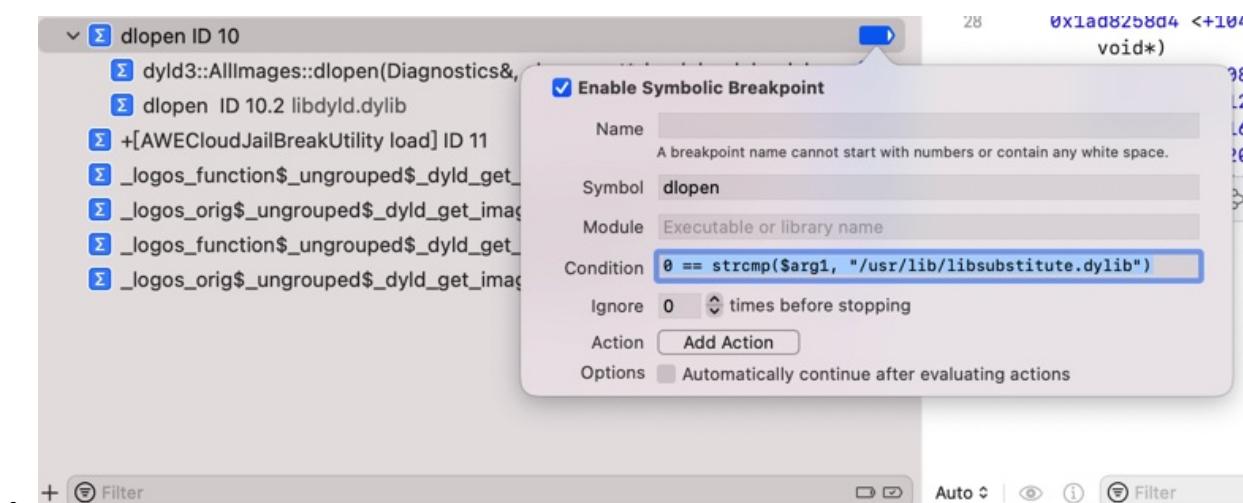
Couldn't parse conditional expression error user expression has unknown return type cast the call to its declared return type

`0 == strcmp($arg1, "/usr/lib/libsubstitute.dylib")`

- 问题

XCode的lldb中，加断点的条件判断：

```
0 == strcmp($arg1, "/usr/lib/libsubstitute.dylib")
```



报错：

```
Stopped due to an error evaluating condition of breakpoint 10.2: "0 == strcmp($arg1, "/usr/lib/libsubstitute.dylib")"
Couldn't parse conditional expression:
error: <user expression 7>:1:6: 'strcmp' has unknown return type; cast the call to its
declared return type
0 == strcmp($arg1, "/usr/lib/libsubstitute.dylib")
^-----
```

- 原因

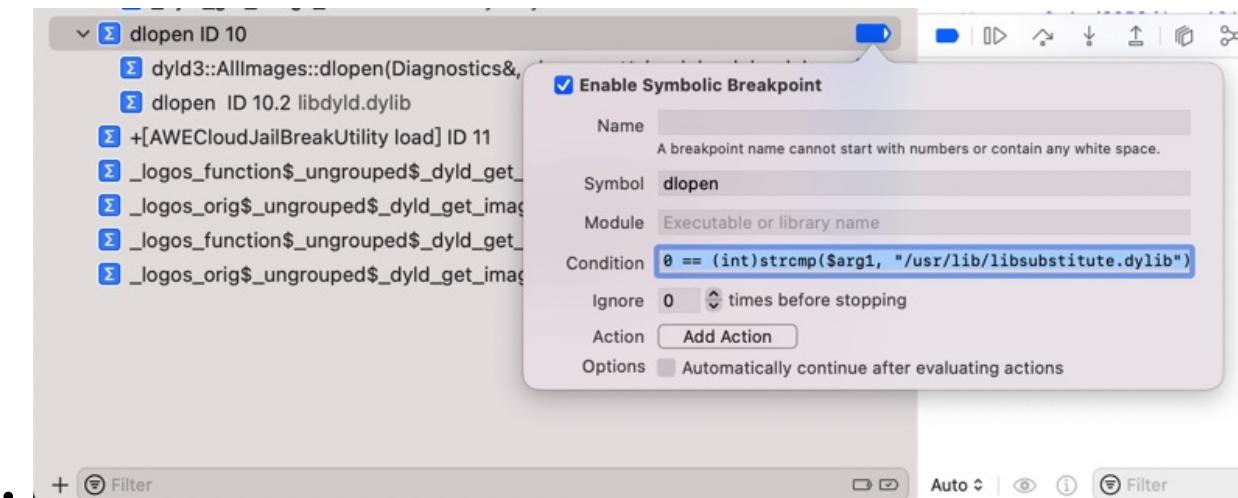
此处无法识别 `strcmp` 的返回值类型 `size_t`

注： `size_t` 本身一般是 `unsigned int` 或 `unsigned long`

- 解决办法

加上强制转换，为普通的int类型：

```
0 == (int)strcmp($arg1, "/usr/lib/libsubstitute.dylib")
```



判断objc_alloc_init中+44行的汇编代码中参数是否是AADeviceInfo的实例

- 写法1: `(bool)[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]`

之前经过 lldb 命令测试发现 `objc_getClass` 返回的结果类型无法识别:

```
(lldb) po [$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
error: expression failed to parse:
warning: user expression 25 :1:2: receiver type 'unsigned long' is not 'id' or interface pointer, consider casting it to 'id'
[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
^~~
error: user expression 25 :1:21: 'objc_getClass' has unknown return type; cast the call to its declared return type
[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
```

所以最后要改为:

```
(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
```

才至少确保语法上是正确的 (至少po可以正常解析执行)

```
(lldb) po [$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
nil
```

- 写法2: `[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]`

表达式最前面没有加上bool强制转换, 所以Xcode无法识别 (是条件判断类型的语句)

- 写法3: `(bool)[NSStringFromClass($x0) isEqualToString: @\"AADeviceInfo\"]`
 - 此处已经是 `AADeviceInfo` 的 `Instance`, 而不是 `class`, 所以不能用 `NSStringFromClass`
 - 因为函数定义是:
 - `NSString * NSStringFromClass(Class aClass);`

- `NSStringFromClass` 的参数，应该是 `Class`，而非 `Instance`
- 详见：
 - 【已解决】iOS的ObjC中如何获取Class类名

iOS的ObjC相关

WARNING: Unable to resolve breakpoint to any actual locations

- 问题

lldb 中给 ObjC 函数加断点：

```
br s -n "-[NSMutableURLRequest setValue:forHTTPHeaderField:]"
```

报错： WARNING: Unable to resolve breakpoint to any actual locations.

- 直接原因

当前被调试的二进制中，的确没有这个类的函数，可供加断点，所以报错。

- 深层次原因

此处的被lldb调试的对象，iOS的app：`Preferences`（或者是此刻iOS中系统的库），是没有包含调试的信息的，或者是经过特殊处理了，去掉了可以调试的信息

导致此处ObjC的类的函数：

```
-[NSMutableURLRequest setValue:forHTTPHeaderField:]
```

内部找不到，所以就加不上断点。

- 解决办法：没法解决
- 规避办法

此处特殊的，可以去找：其他的，个别的，继承了该类的函数，去加断点

比如：

```
image lookup -rn "setValue:forHTTPHeaderField:"
```

找到的部分类，有此函数，所以可以去加断点：

```
br s -n "-[SSMutableURLRequestProperties setValue:forHTTPHeaderField:]"
```

给带Deprecated的函数名加断点加不上断点

之前试过给ObjC函数：

```
+[AADeviceInfo (Deprecated) udid]
```

去加断点，发现加不上

后来才知道，其实是：

- 之前没加上断点，是另外的原因
 - 调试目标和hook目标不一致
 - 具体解决办法，详见：[断点能加上且能触发](#)
- 此处能加上断点，用的函数名是不带 `deprecated` 字眼的

```
+[AADeviceInfo udid]
```

- 如果起查找函数，可以发现底层函数就是带 `Deprecated` 字眼的函数 = 真正触发时，Xcode中显示的也是带`Deprecated`字眼的函数

```
(lldb) image lookup -vn "+[AADeviceInfo udid]"
1 match found in /Users/crifan/Library/Developer/Xcode/iOS DeviceSupport/15.0 (19A3
46)/Symbols/System/Library/PrivateFrameworks/AppleAccount.framework/AppleAccount:
    Address: AppleAccount[0x00000000191e0e558] (AppleAccount.__TEXT.__text + 176
432)
        Summary: AppleAccount`+[AADeviceInfo(Deprecated) udid]
        Module: file = "/Users/crifan/Library/Developer/Xcode/iOS DeviceSupport/15.
0 (19A346)/Symbols/System/Library/PrivateFrameworks/AppleAccount.framework/AppleAcc
ount", arch = "arm64"
        Symbol: id = {0x000000473}, range = [0x0000000195ce6558-0x0000000195ce659c),
name=+[AADeviceInfo(Deprecated) udid]"
```

其他

warning: failed to set breakpoint site at 0x1b1750624 for breakpoint 66.1: error sending the breakpoint request

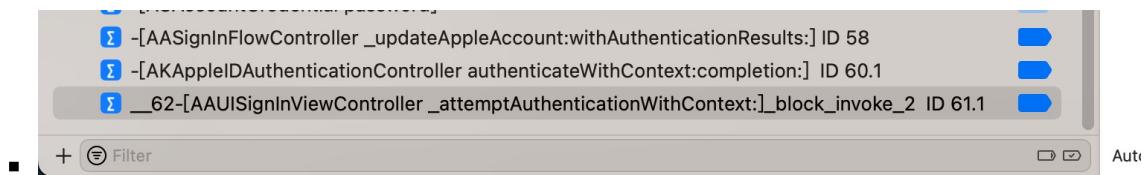
- 问题

Xcode中加断点时报错：

```
warning: failed to set breakpoint site at 0x1b1750624 for breakpoint 66.1: error sending the breakpoint request
```



- 原因：调试环境被破坏了
 - 细节：之前Mac中，正常用Xcode调试USB连接的iPhone设备中的内容，但是后来，关闭了Mac屏幕，内部应该就是去休眠了
 - 所以会导致Mac中和iPhone中部分程序已休眠，或者是停止运行，断开连接等异常情况
 - 所以即使重新打开Mac屏幕，重新继续去Xcode中去调试，此时环境也不对了，被破坏了，所以会出现各种异常情况
- 解决办法
 - 重新用Mac连接iPhone，加上Xcode去，重新启动调试环境，即可正常继续调试：添加断点等等



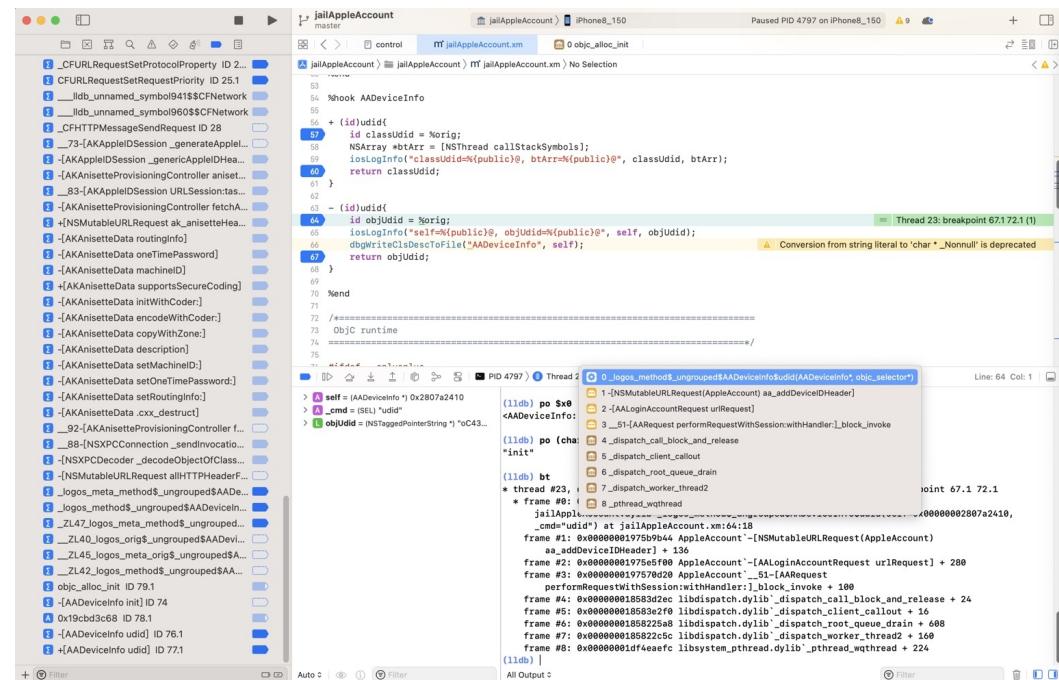
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:
2023-07-10 16:27:36

断点没触发

举例

AppleAccount的函数: +[AADeviceInfo udid] 和 -[AADeviceInfo udid]

- 现象
 - (iOSOpenDev的) hook插件可以触发
 - 但是: debugserver+lldb调试时, 断点可以加上, 但是却无法触发 (断点)
- 原因: 调试目标不匹配
 - iOSOpenDev的插件的hook目标是: com.apple.Preferences =设置app
 - Xcode调试目标是: akd 二进制
- 解决办法: 确保目标一致
 - 插件代码的hook目标是: com.apple.Preferences =设置app=Preferences
 - Xcode去调试的目标是: Preferences 进程 (的 PID)
 - 注: 此处Preferences的PID=4797



-[__NSCFConstantString stringByAppendingString:]

XCode调试抖音ipa崩溃, 发现崩溃日志是:

```
Terminating app due to uncaught exception 'NSInvalidArgumentException', reason: '*** -[__NSCFConstantString stringByAppendingString:]: nil argument'
```

以为对应symbol函数是:

```
-[__NSCFConstantString stringByAppendingString:]
```

去添加断点，结果：断点没触发

最后确认，问题的原因和解决办法分别是：

- 此处有2层错误
 - 第一层是：语法错误 -> 函数名有误
 - 末尾多了个冒号 :

```
-[__NSCFConstantString stringByAppendingString]:
```

- 因此：实际上函数断点也没加上，更不会触发断点

- 应该改为：

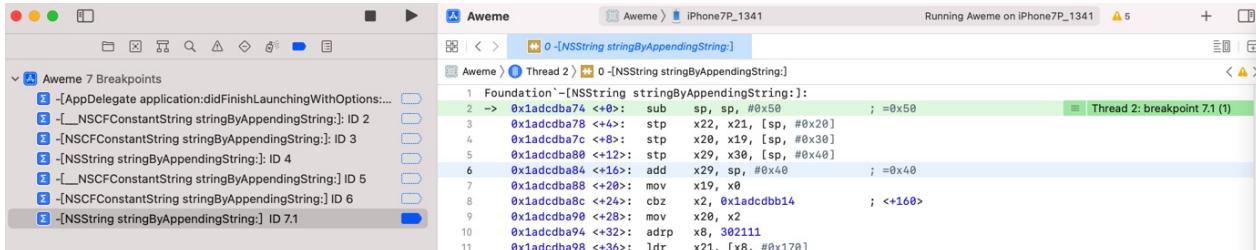
```
-[__NSCFConstantString stringByAppendingString]
```

- 第二层是：类名搞错了

- 不是 `__NSCFConstantString`，而是 `NSString`
 - 此处函数（的类）搞错了，此处实际上应该用（`__NSCFConstantString` 的所属的 `CF = CoreFoundation`）对应的 `NS` 的类：`NSString`
- 所以应该写成

```
-[NSString stringByAppendingString]:
```

即可正常触发断点，效果是：



objc_alloc_init

Xcode中加的条件判断的断点：

- 函数 `objc_alloc_init`
 - `(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]`
- 函数 `objc_alloc_init` 内的汇编代码
 - `(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]`

没有生效的原因：

(1)

- 函数 `objc_alloc_init`
 - `(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]`
 - 没生效的原因

`objc_alloc_init` 传入参数是 `Class`，此处 `isKindOfClass` 不适用于 `Class`，只适用于 `Instance`，所以无效。

(2) 对于条件判断写法:

- `(bool)[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]`

之前经过lldb命令测试发现objc_getClass返回的结果类型无法识别:

```
(lldb) po [$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
error: expression failed to parse:
warning: user expression 25 :1:2: receiver type 'unsigned long' is not 'id' or interface pointer, consider casting it to 'id'
[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
~~
error: user expression 25 :1:21: 'objc_getClass' has unknown return type; cast the call to its declared return type
[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
```

所以最后要改为:

```
(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
```

才至少确保语法上是正确的 (至少po可以正常解析执行)

```
(lldb) po [$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
nil
```

(3) 之前的:

- 函数objc_alloc_init 内的
 - 汇编代码

```
libobjc.A.dylib`objc_alloc_init:
...
0x19cbd3c68 <+44>: b      0x19cbcd000          ; objc_msgSend
```

- 的条件判断

- `(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]`
- 没有生效

应该就是此处最终找到的原因:

实际上是代码本身没运行到, 所以断点没触发

而断点的条件判断:

- `(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]`

本身, 对于

```
0x19cbd3c68 <+44>: b 0x19cbcd000 ; objc_msgSend
```

来说是正确的, 没有问题的。

而实际上是代码本身没运行到, 是由于:

- 之前的：插件hook的目标（`com.apple.Preferences`）和Xcode调试的目标（`akd`）不一致，导致的

此处已经解决了此处问题，确保：

- 现在的：插件hook的目标（`com.apple.Preferences`）和Xcode调试的目标（`com.apple.Preferences`的PID）是一致的

就确保代码能执行到，可以触发，条件判断语法是正确的断点了。

(4) 对于最新的断点：

- `objc_alloc_init`
 - （`bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]`

后续经测试确认，也是可以正常触发断点的。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-10 15:54:10

Xcode

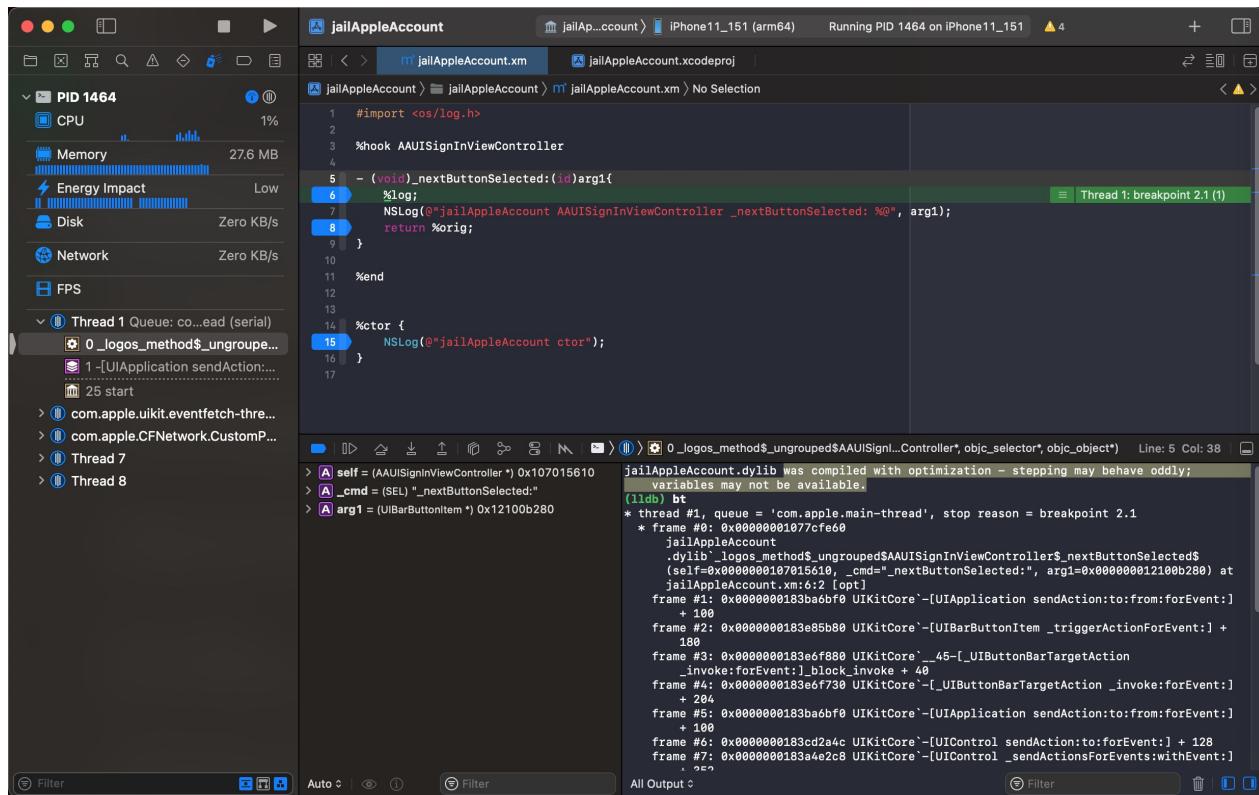
此处整理Xcode中断点相关的常见问题：

**was compiled with optimization - stepping may behave oddly;
variables may not be available**

- 问题

Xcode中编译的iOSOpenDev的dylib插件，去调试时，右下角调试窗口输出警告信息：

```
jailAppleAccount.dylib was compiled with optimization - stepping may behave oddly;
variables may not be available.
```



- 原因

Xcode在编译代码时（默认就）启用了优化：

```
Xcode -> Targets -> xxx -> Build Settings -> Apple Clang - Code Generation -> Optimization Level 中的 Release , 默认是： Fastest, Smallest [-Os]
```

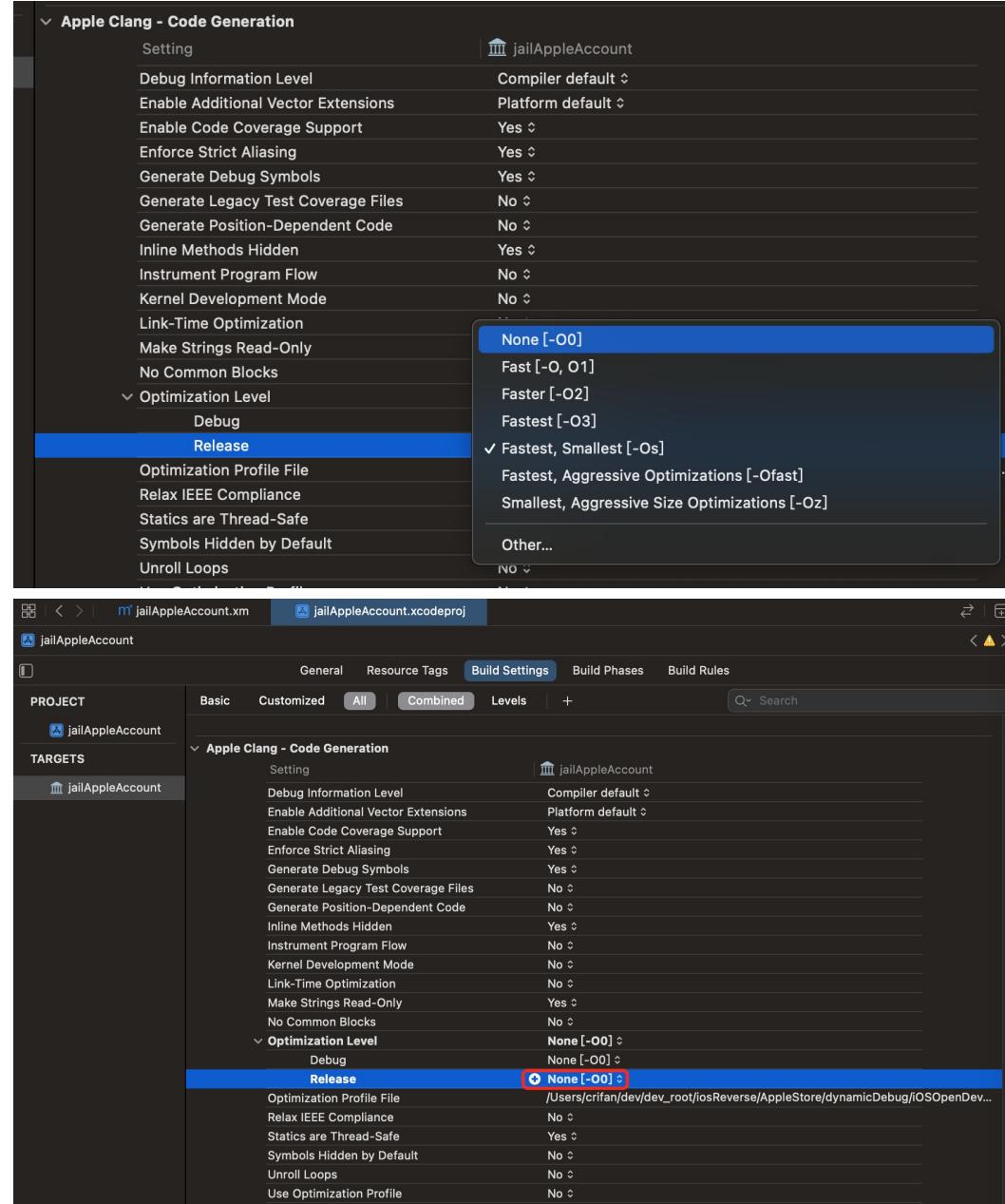
含义是：程序尽可能的快，文件尽可能的小。

涉及到的内部优化，就可能会把部分调试相关内容优化去掉，从而可能导致调试时出现上述的现象：

- stepping may behave oddly
 - 单步（进入或跳过）调试时，会出现奇怪的现象
 - 因为部分代码可能被优化掉了
 - 从而该代码单步执行可能执行不到，或者和源码对不上

- variables may not be available
 - 部分变量可能会被优化掉
- 解决办法：去掉优化，保留调试信息
- 具体步骤：
 - 把上述中，Release 的 Optimization Level 的值改为 None [-O0]

■ 图



- 表示：不（做任何）优化
- 从而保留了调试信息
- 后续
- 单步调试就正常了，和代码对得上了
 - 变量值也不会丢失了

Xcode触发断点后长时间无操作好像调试会断开

- 现象

用Xcode去Attach到Preferences进程，然后能触发到之前hook代码的断点，可以正常调试了。

触发断点后，暂停了很多秒后，且期间没有任何操作后：

好像断点会丢失，具体现象是，app好像自动接着运行了，但此处断点还停在代码断点处

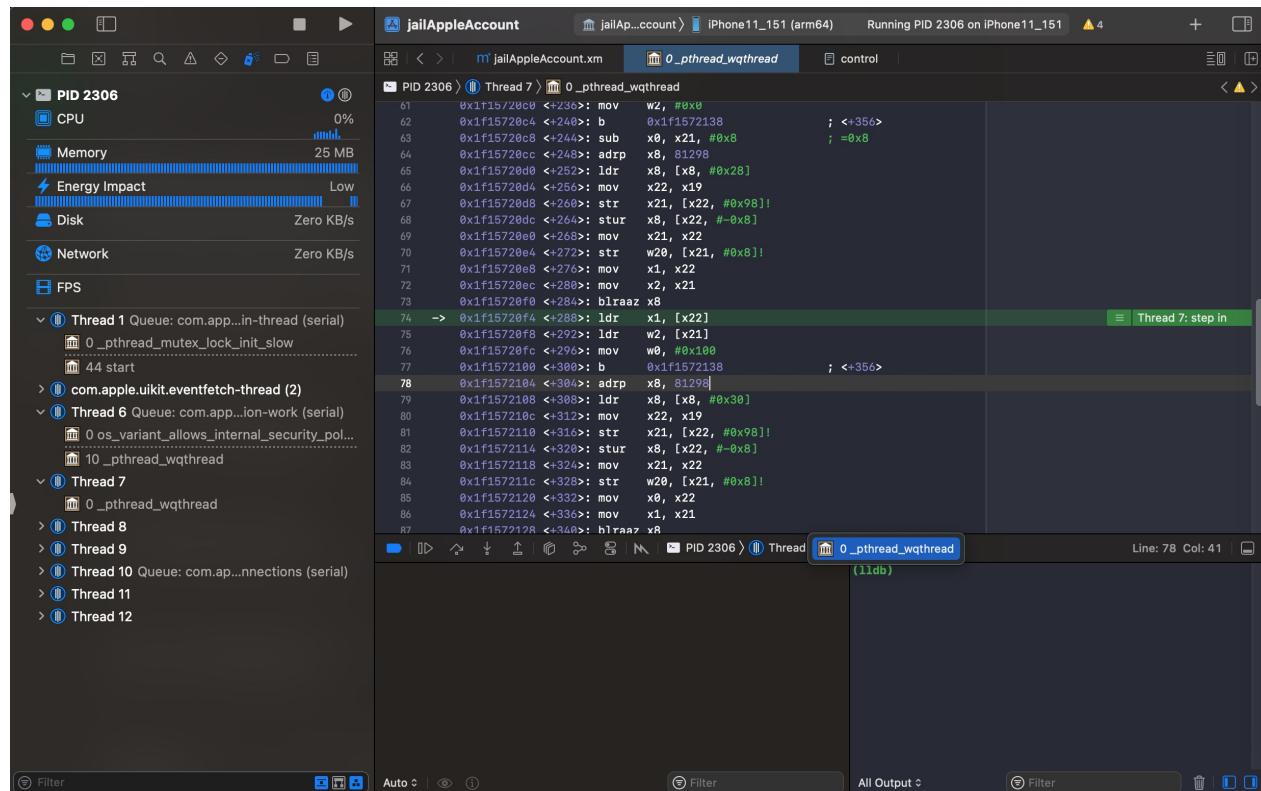
后来，又出现类似问题：

长时间没任何操作，app页面一闪，然后好像就和当前Xcode断开连接了，而无法继续正常调试了：

虽然Xcode中可以单步调试，但是实际上已经断开了

-》 单步调试的结果：，都是不正常的：

比如连续多次调试后，此处进入了一个不知名的进程：



且也看不到调用堆栈了。

- 原因：暂时不能完全确定就是Xcode的问题或bug

- 尝试解决：

- 期间去试过：

- 加大连接超时的时间方面的设置： `plugin.process.gdb-remote.packet-timeout = 1200`

```
crifan@licrifandeMacBook-Pro ~ cat ~/lldbinit
command script import /usr/local/opt/chisel/libexec/fbchiselldb.py
command script import "~/Library/Application Support/Realm/rlm_lldb.py" --all
low-reload
settings set plugin.process.gdb-remote.packet-timeout 1200
```

- 好像问题有所缓解

- 后续暂时没怎么遇到这个问题，算是基本解决了
- 注：但可能还会偶尔复发，暂时去不确定。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-10 16:39:22

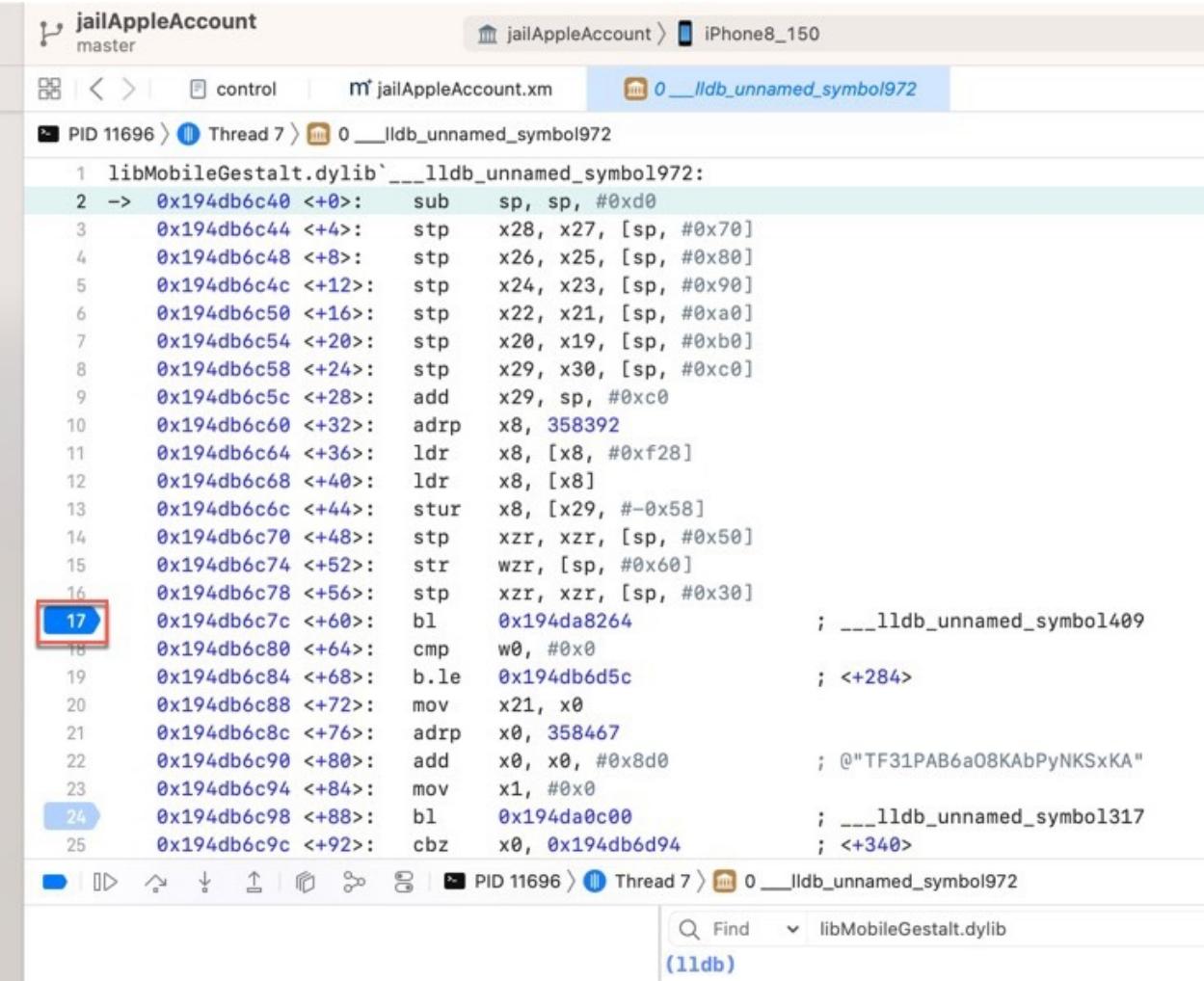
hook代码的断点

此处整理，Xcode中，对于（用iOSOpenDev等工具去写）hook代码时，加的断点方面的常见问题和相关经验心得。

给当前行加断点却看不到

Xcode中的汇编代码加断点后，默认改行左边会显示蓝色底色，表示加了断点了：

比如：



```

jailAppleAccount
master          jailAppleAccount > iPhone8_150
control      m jailAppleAccount.xm  0 __lldb_unnamed_symbol972

PID 11696 > Thread 7 > 0 __lldb_unnamed_symbol972

1 libMobileGestalt.dylib`__lldb_unnamed_symbol972:
2 -> 0x194db6c40 <+0>:    sub    sp, sp, #0xd0
3     0x194db6c44 <+4>:    stp    x28, x27, [sp, #0x70]
4     0x194db6c48 <+8>:    stp    x26, x25, [sp, #0x80]
5     0x194db6c4c <+12>:   stp    x24, x23, [sp, #0x90]
6     0x194db6c50 <+16>:   stp    x22, x21, [sp, #0xa0]
7     0x194db6c54 <+20>:   stp    x20, x19, [sp, #0xb0]
8     0x194db6c58 <+24>:   stp    x29, x30, [sp, #0xc0]
9     0x194db6c5c <+28>:   add    x29, sp, #0xc0
10    0x194db6c60 <+32>:  adrp   x8, 358392
11    0x194db6c64 <+36>:  ldr    x8, [x8, #0xf28]
12    0x194db6c68 <+40>:  ldr    x8, [x8]
13    0x194db6c6c <+44>:  stur   x8, [x29, #-0x58]
14    0x194db6c70 <+48>:  stp    xzr, xzr, [sp, #0x50]
15    0x194db6c74 <+52>:  str    wzr, [sp, #0x60]
16    0x194db6c78 <+56>:  stp    xzr, xzr, [sp, #0x30]
17    0x194db6c7c <+60>:  bl    0x194da8264 ; __lldb_unnamed_symbol409
18    0x194db6c80 <+64>:  cmp    w0, #0x0
19    0x194db6c84 <+68>:  b.le  0x194db6d5c ; <+284>
20    0x194db6c88 <+72>:  mov    x21, x0
21    0x194db6c8c <+76>:  adrp   x0, 358467
22    0x194db6c90 <+80>:  add    x0, x0, #0x8d0 ; @"TF31PAB6aO8KAbPyNKSxKA"
23    0x194db6c94 <+84>:  mov    x1, #0x0
24    0x194db6c98 <+88>:  bl    0x194da0c00 ; __lldb_unnamed_symbol317
25    0x194db6c9c <+92>:  cbz    x0, 0x194db6d94 ; <+340>

Find  libMobileGestalt.dylib
(11db)

```

但是，当给，调试时函数调用堆栈中，上层某层函数的汇编代码的下一行将要执行的汇编代码，去加断点时，结果：

点击了左边的空白处，但是：没有出现（表示断点添加成功的）蓝色底色：

```

111 0x1102d331c <+436>: strb w21, [sp, #0x68]
112 0x1102d3320 <+440>: str x23, [sp, #0x50]
113 0x1102d3324 <+444>: adrp x8, -46460
114 0x1102d3328 <+448>: ldr x20, [x8, #0x660]
115 0x1102d332c <+452>: mov x0, x23
116 0x1102d3330 <+456>: bl 0x112925950
117 0x1102d3334 <+460>: mov x21, x0
118 0x1102d3338 <+464>: add x2, sp, #0x30 ; =0x30
119 0x1102d333c <+468>: mov x0, x25
120 0x1102d3340 <+472>: mov x1, x20
121 0x1102d3344 <+476>: bl 0x1129258fc
122 0x1102d3348 <+480>: bl 0x1102d3a68 ; ___lldb_unnamed_symbol1462841$$AwemeCore Thread...
123 0x1102d334c <+484>: mov x20, x0
124 0x1102d3350 <+488>: str x26, [sp, #0x8]
125 0x1102d3354 <+492>: adr x8, #0x130 ; ___lldb_unnamed_symbol1462806$$AwemeCore
126 0x1102d3358 <+496>: nop

```

以为是：没有成功（给该行汇编代码）加上断点

其实就是没有成功添加断点，且

- 是无法给该行（断点时，上层某函数的，将要执行的下一行汇编代码）加上断点的
 - 因此：此时右键该行左边空白处，是看不到=无法出现右键菜单的
 - 对比：其他普通已经加上断点的地方出现的，右键是可以出现，编辑断点、删除断点等功能的右键菜单的

xm源码中无法看到和添加断点

- 问题

在Xcode从 13.1 升级到 13.2.1 后，之前已用 Logos 写了很多hook代码，比如 youtubeDylib.xm，且其中给hook代码加了很多断点，但是此处发现：

```

100 //-(void)playerLoop:(CDUnknownBlockType)arg1{
101   iosLogInfo("arg1.value < %f", arg1.value);
102   %orig;
103 }
104 -(void)setPlayerPaused:(_Bool)arg1{
105   iosLogInfo("arg1=%s", boolToStr(arg1));
106   %orig;
107 }
108 //-(void)setPlayerStatusObserver:(CDUnknownBlockType)arg1{
109   -(void)setPlayerStatusObserver:(id)arg1{
110     iosLogInfo("arg1=%@", arg1);
111     %orig;
112   }
113 //-(void)setCurrentTimeObserver:(CDUnknownBlockType)arg1{
114   -(void)setCurrentTimeObserver:(id)arg1{
115     iosLogInfo("arg1=%@. arg1");
116   }

```

断点都看不到了，所以点击右键也没反应

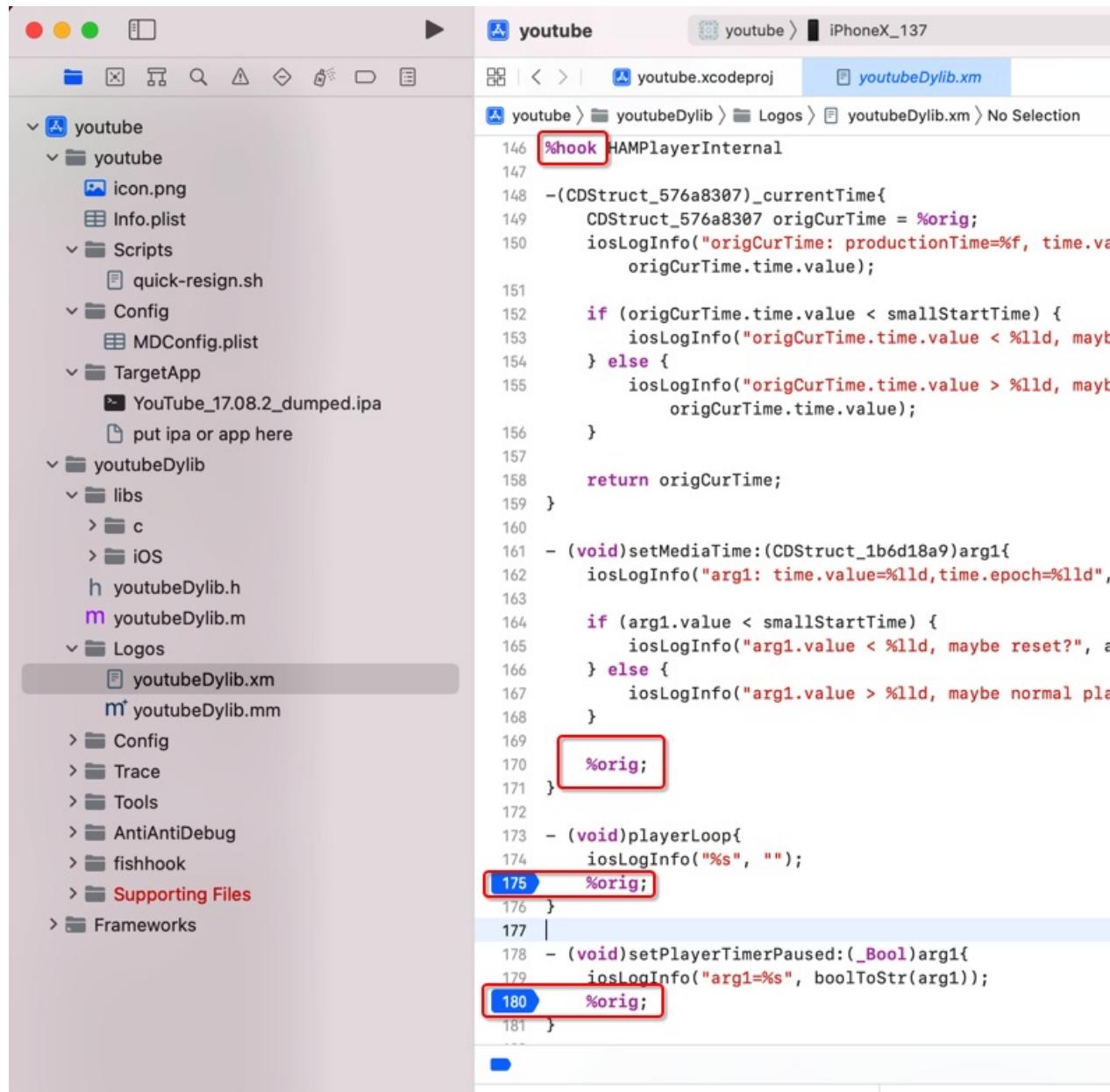
- 原因：Xcode的版本升级，把之前安装的 Logos 语法高亮的插件覆盖没了，导致语法高亮失效，断点也看不到
- 解决办法：让Xcode识别xm文件，即可显示和操作断点
- 具体步骤

Xcode中安装Logos语法高亮的插件：

<https://github.com/brendonjkding/Logos-Xcode11.git>

效果：正常情况下，切换到Logos后，语法高亮就正常了：

xm文件中的 %orig 、 %hook 等正常的高亮显示了，且断点也正常显示和操作



```

youtube
├── youtube
│   ├── icon.png
│   └── Info.plist
└── Scripts
    └── quick-resign.sh
Config
└── TargetApp
    └── YouTube_17.08.2_dumped.ipa
    └── put ipa or app here
youtubeDylib
└── libs
    ├── c
    ├── iOS
    │   └── youtubeDylib.h
    └── youtubeDylib.mm
Logos
└── youtubeDylib.xm
146 %hook HAMPlayerInternal
147
148 -(CDStruct_576a8307)_currentTime{
149     CDStruct_576a8307 origCurTime = %orig;
150     iosLogInfo("origCurTime: productionTime=%f, time.va
        origCurTime.time.value);
151
152     if (origCurTime.time.value < smallStartTime) {
153         iosLogInfo("origCurTime.time.value < %lld, mayb
154     } else {
155         iosLogInfo("origCurTime.time.value > %lld, mayb
            origCurTime.time.value);
156     }
157
158     return origCurTime;
159 }
160
161 - (void)setMediaTime:(CDStruct_1b6d18a9)arg1{
162     iosLogInfo("arg1: time.value=%lld,time.epoch=%lld",
163
164     if (arg1.value < smallStartTime) {
165         iosLogInfo("arg1.value < %lld, maybe reset?", a
166     } else {
167         iosLogInfo("arg1.value > %lld, maybe normal pla
168     }
169
170     %orig;
171 }
172
173 - (void)playerLoop{
174     iosLogInfo("%s", "");
175     %orig;
176 }
177
178 - (void)setPlayerTimerPaused:(_Bool)arg1{
179     iosLogInfo("arg1=%s", boolToStr(arg1));
180     %orig;
181 }

```

注：如果偶尔会遇到其他的：Logos语法高亮有问题，或者是xm断点不显示了，则关闭文件后重新打开，多试试几次，估计就可以了。至少暂时是这样。

给函数加hook代码的同时加函数名断点会导致EXC_BREAKPOINT的崩溃

iOS逆向期间，之前遇到一些次数的特殊情况：

对于某个函数，如果去加上了hook代码之后，额外再加上该函数的函数名的断点，则往往就会导致崩溃：

dladdr

之前给dladdr加了hook：

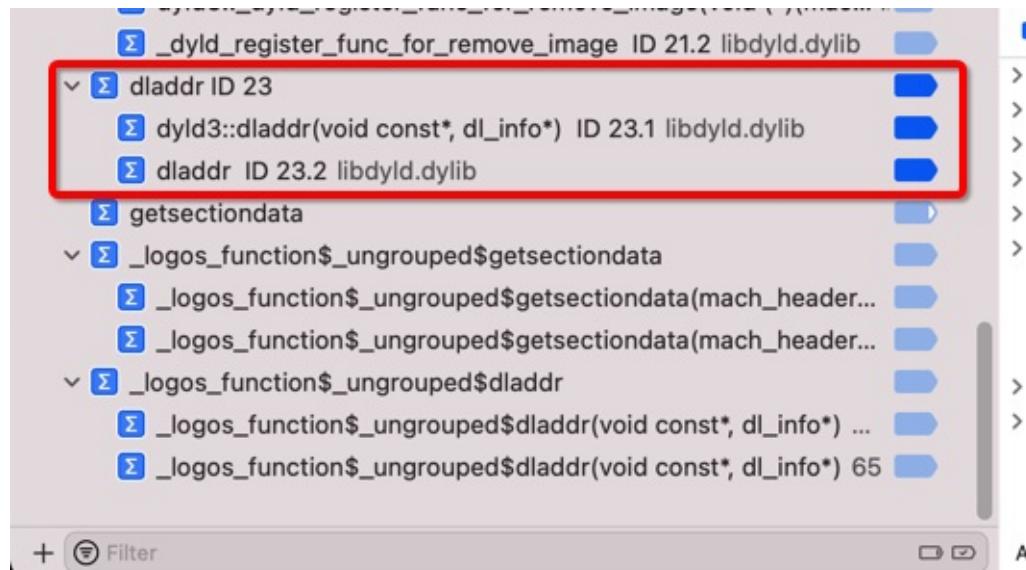
```

57
58 int dladdr(const void *, Dl_info *);
59 //int dladdr(void *, Dl_info *);
60 //extern int dladdr(const void *, Dl_info *);
61
62 //hookf(int, dladdr, void *addr, Dl_info *info){
63 %hookf(int, dladdr, const void *addr, Dl_info *info){
64     iosLogDebug("addr=%p,info=%p", addr, info);
65     int finalRet = DLADDR_FAILED;
66
67     if (NULL == addr) {
68         iosLogInfo("addr is %s", "NULL");
69     } else {
70         void* origAddr = (void*)addr;
71
72         bool isHookedAddr = isHookedDladdrAddress(addr);
73         if (isHookedAddr) {
74             origAddr = hookedToOrigDladdrAddr(addr);
75
76             iosLogDebug("addr=%p -> isHookedAddr=%s -> origAddr=%p", addr,
77                         boolToStr(isHookedAddr), origAddr);
78
79             if (NULL == origAddr) {
80                 iosLogInfo("addr=%p -> isHookedAddr=%s -> origAddr=%p", addr,
81                             boolToStr(isHookedAddr), origAddr);
82             }
83         }
84
85         //    int origRet = %orig;
86
86 //        int origRet = DLADDR_FAILED;
87 //        if (NULL == origAddr) {
88 //            origRet = DLADDR_FAILED;
89 //        } else {
90 //            origRet = %orig(origAddr, info);
91 //        }
92
93         int origRet = %orig(origAddr, info);
94         finalRet = origRet;
95
95     bool isNotHookedAddr = !isHookedAddr;
96     bool isNeedHook = cfgHookEnable_dylib_dladdr && isNotHookedAddr;
97     if (isNeedHook) {
98         //    if (dladdrRetInt > 0) {
99             //<DLADDR_FAILED> -> %orig

```

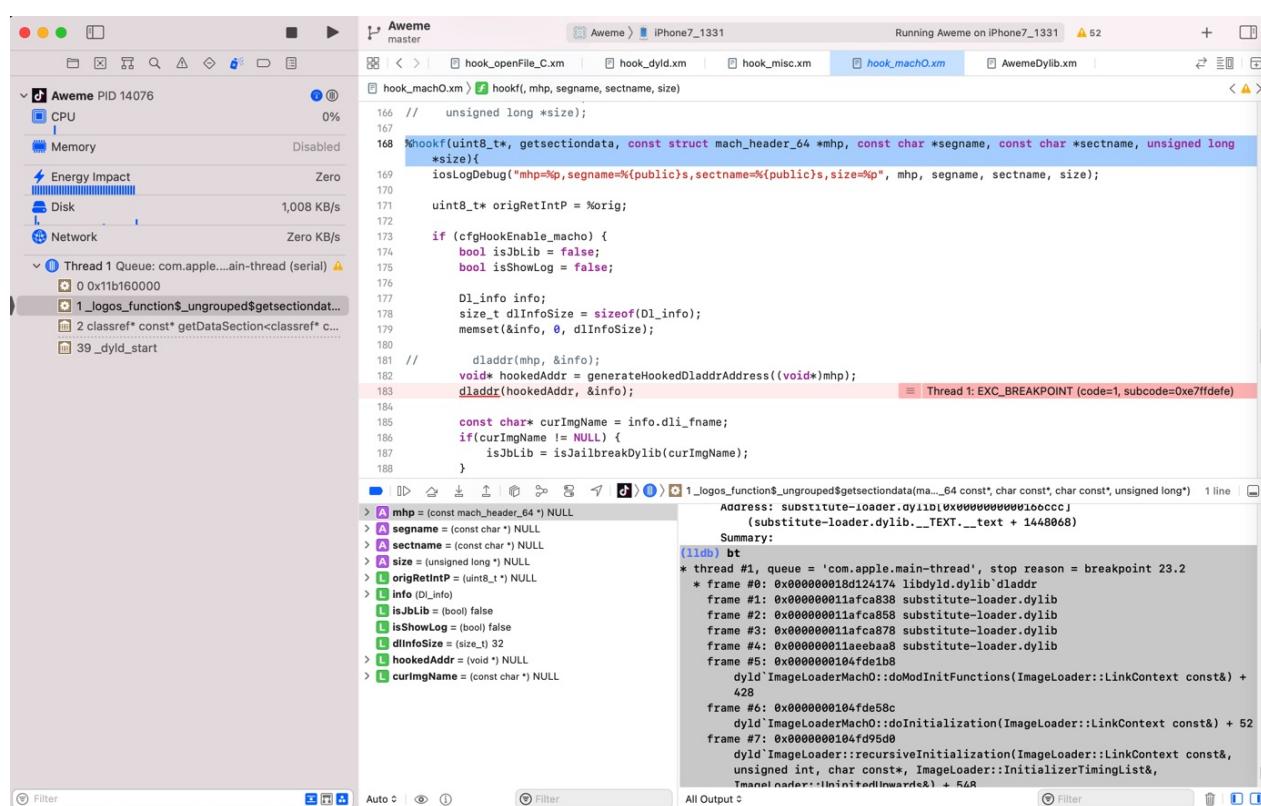
53 characters |

然后也给了 dladdr 函数名加了断点：

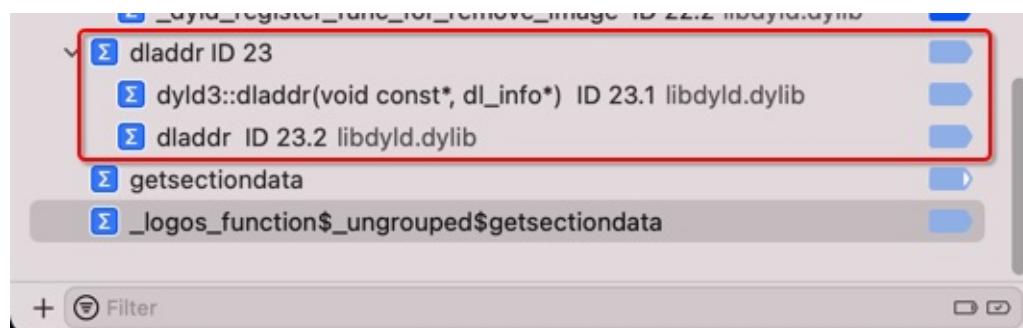


结果调试期间，`getsectiondata` 中调用 `dladdr` 的地方，就崩溃了：

Thread 1: EXC_BREAKPOINT (code 1, subcode=0xe7ffdefe)



后来去，禁用了 `dladdr` 的断点：



就不会出现之前的崩溃了。

_dyld_register_func_for_add_image

之前已给 `_dyld_register_func_for_add_image` 加了hook代码：

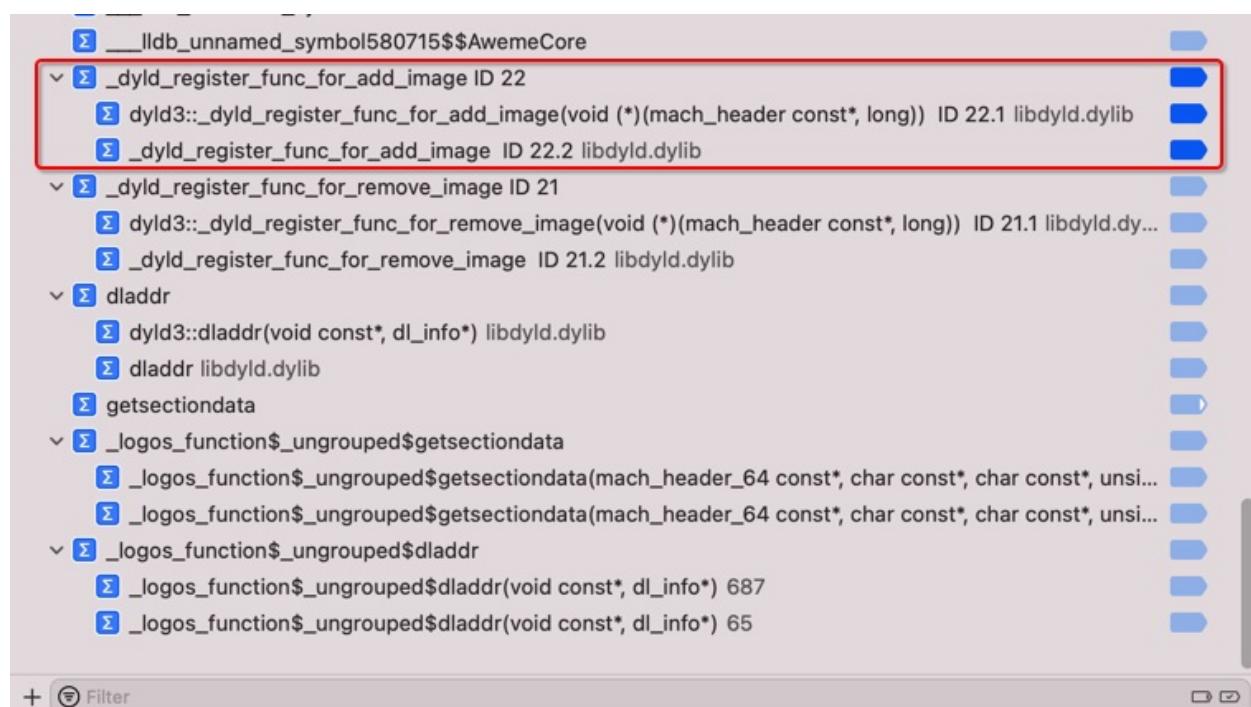
```
%hookf(void, _dyld_register_func_for_add_image, void (*func)(const struct mach_header*
mh, intptr_t vmaddr_slide)){
//    iosLogInfo("%sfunc=%p -> Omitted", HOOK_PREFIX(cfgHookEnable_dyld), func);
iosLogInfo("%sfunc=%p", HOOK_PREFIX(cfgHookEnable_dyld), func);

#ifndef XCODE_DEBUG
    orig;
//    %orig(func);
#endif
}
```

且也同时给函数名：

```
_dyld_register_func_for_add_image
```

加了断点：

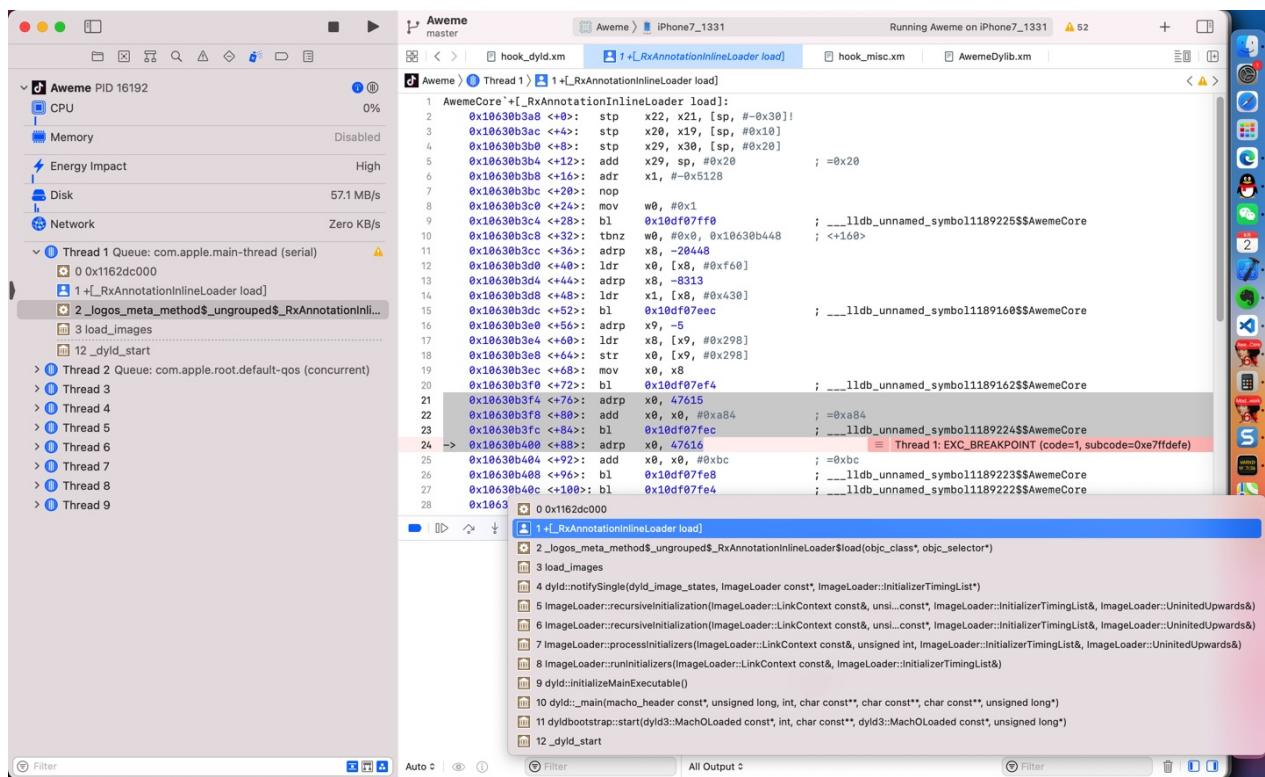


然后调试后，就会出现：

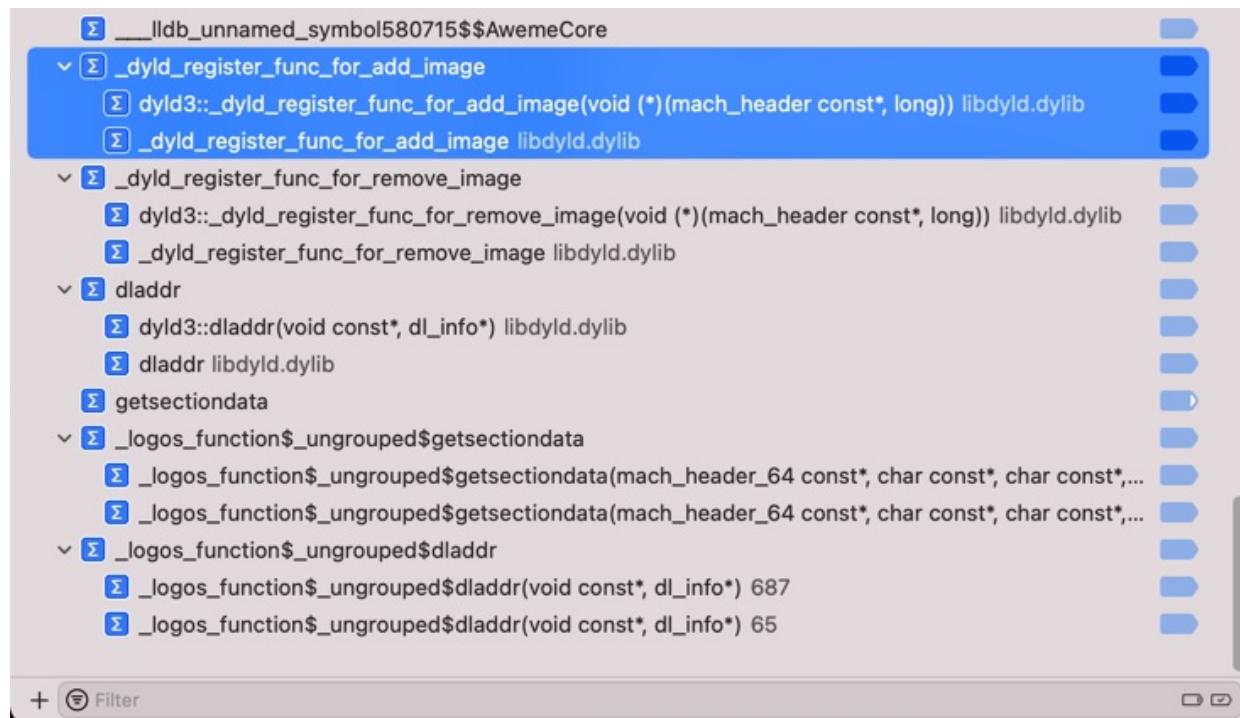
```
%hookF(void, _dyld_register_func_for_add_image, void (*func)(const struct mach_header* mh, intptr_t vmaddr_slide)){
    ...
    %orig;
    ...
}
```

其中的 `%orig` 去运行，就会崩溃：

Thread 1: EXC_BREAKPOINT (code 1, subcode 0xe7fffdef)



而去掉（此处是禁用）断点后：



崩溃问题就消失了。

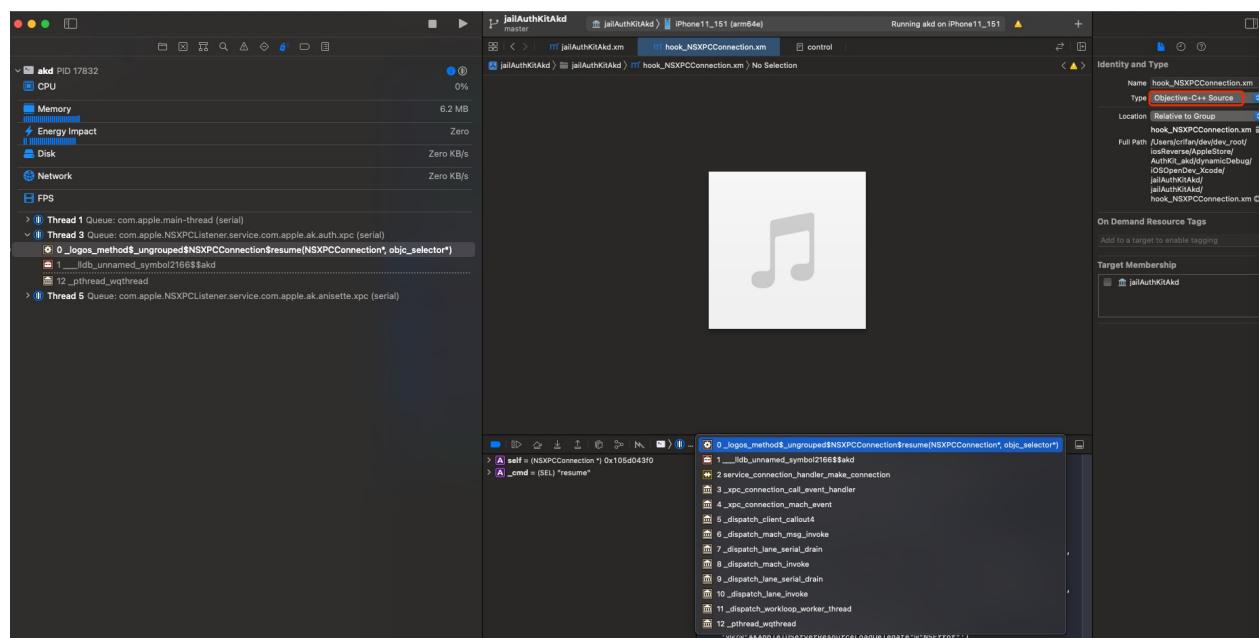
触发到xm源码中函数断点但是显示音频文件图标

- 问题

此处遇到一个奇怪问题：

Xcode 中 `iOSOpenDev` 的 `logos` 插件代码 `.xm` 文件中，给（`NSXPConnection` 的 `resume`）函数写了 hook 代码，且加上了断点，去调试，结果：

虽然是触发了断点，但是却显示出了音频文件图标，而不是 `.xm` 源码的断点的地方



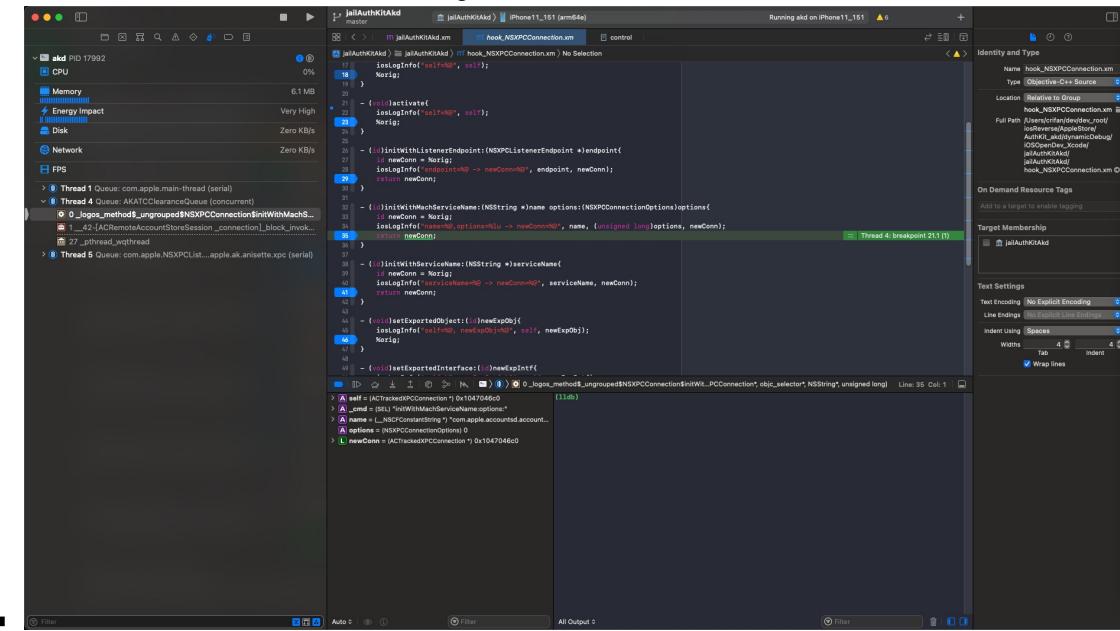
- 另：此处已经确认文件类型设置无误

- 已点击Xcode右上角的文件类型 Type，已经设置 xm 是 Objective-C++ Source

- 而不是默认的音频文件类型了。

但是不知道此处为何调试触发断点时，显示的是音频文件图标。

- 原因：经过调试研究确认，基本上确认是，Xcode的偶发性的bug而已。
 - 解决办法：重启Xcode
 - 即可正常触发断点，Xcode界面跳转到Logos的hook代码处



crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:
2023-07-10 16:33:42

lldb

Xcode中可用的断点在lldb中却不可用

此处用 debugserver 去调试设置 Preferences 的app:

- iPhone8

```
/usr/bin/debugserver 0.0.0.0:20235 -a 255
```

- Mac

```
lldb
process connect connect://192.168.2.13:20235
```

然后去:

- 给 objc_alloc_init 加上带条件判断的断点:

```
br s -n "objc_alloc_init" -c '(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]'
br s -n "objc_alloc_init" -c '(int)strcmp((char *)class_getName($x0), "AADeviceInfo")==0'
```

- 注: 这2个带条件判断的断点

```
(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]
(int)strcmp((char *)class_getName($x0), "AADeviceInfo")==0
```

- 之前在Xcode图形界面中, 都是可以正常工作的: 能添加的上断点, 且能触发断点

就会导致:

- 设置app出现: 白屏, 卡死
 - 有时候会: 过了段时间(很多分钟后), 设置app又恢复正常操作, 但是始终会:
- 报错
 - warning: hit breakpoint while running function, skipping commands and conditions to prevent recursion

至此，无法正常用上述2个条件判断断点，通过debugserver+lldb调试。

所以最终放弃。

暂时不清楚：

- 为何之前Xcode中可以正常工作的2个断点，在lldb命令行中却无法工作

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:

2023-07-10 17:59:39

经验心得

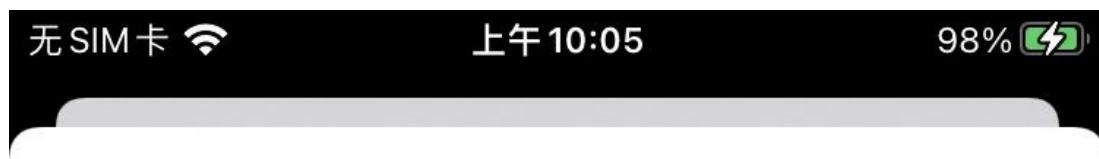
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-10 15:32:27

通用心得

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-10 17:46:01

导致卡死

- 加了断点，导致程序卡死
 - 之前遇到主要有几种情况
 - (1) 断点个数很多
 - (2) (某个或某些断点) 被触发的次数很多=被触发的频率很高
 - (2) 带条件判断，且判断逻辑很复杂
 - 现象和原因
 - 现象：
 - app卡死，界面操作无反应
 - 或者是卡顿：操作后需要很久（几十秒，甚至几分钟后）才有反应
 - 原因
 - 断点被触发了很多次
 - 且/或每次触发到时，都要执行条件判断
 - 由于逻辑复杂，所以判断逻辑执行起来很耗时，综合下来，就会导致程序运行速度很慢，极其的慢
 - 规避办法
 - (临时) 禁用或删除（带复杂逻辑的条件判断的）断点
 - 核心思路：平时用不到时，尽量临时关闭该断点，只有必要时才开启，以减少对程序运行速度的影响
- 举例
 - 举例1：Apple账号登录界面卡死，都无法弹出键盘，导致无法继续输入Apple ID账号
 -



取消

下一步

Apple ID

使用 Apple ID 登录以使用 iCloud 和其他
Apple 服务。

Apple ID

电子邮件或电话

[没有或忘记 Apple ID?](#)



Apple ID 是您用于访问所有 Apple 服务的帐
户。iCloud 使用无线数据。



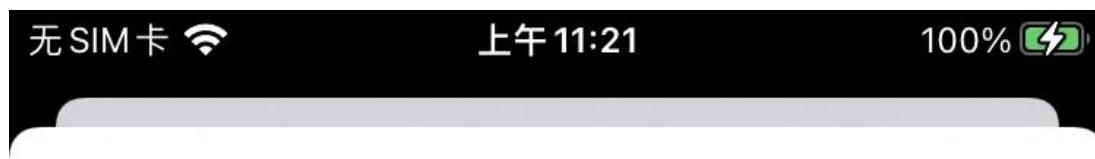
您的 Apple ID 信息用于登录时启用 Apple 服务，其中包括 iCloud 云备
份，该服务可自动备份设备上的数据，以便需要时进行替换或恢复。
您的设备序列号可能被用于检查服务的使用资格。 [了解数据的管理方
式...](#)

- 原因：

- 之前加了，条件判断的断点，且有2个
 - `objc_alloc_init`
 - `(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]`
 - 某地址(`0x19cbd3c68`)
 - `(bool)[[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]]`
- 导致此处卡死
- 解决办法
 - **临时禁用**这2个带条件判断的断点

- 举例2：输入Apple ID账号后点击下一步，界面卡死

-



取消

下一步

Apple ID

使用 Apple ID 登录以使用 iCloud 和其他
Apple 服务。

Apple ID

admin@crifan.com



[没有或忘记 Apple ID?](#)



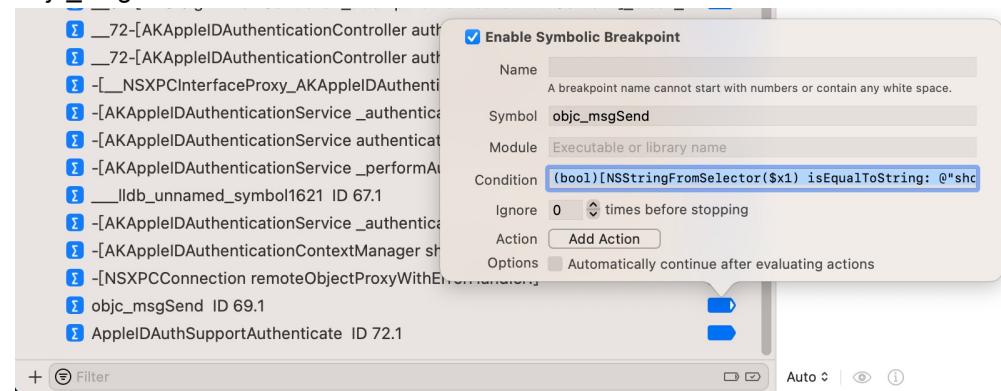
Apple ID 是您用于访问所有 Apple 服务的帐
户。iCloud 使用无线数据。



您的 Apple ID 信息用于登录时启用 Apple 服务，其中包括 iCloud 云备
份，该服务可自动备份设备上的数据，以便需要时进行替换或恢复。
您的设备序列号可能被用于检查服务的使用资格。 [了解数据的管理方
式...](#)

- 背后原因：

- 加了objc_msgSend的条件判断的断点



- 解决办法：临时禁用此条件判断的断点

- -[NSXPConnection remoteObjectProxyWithErrorHandler:]
 - objc_msgSend
 - AppleIDAuthSupportAuthenticate

- 举例3：

- XCode+MonkeyDev调试抖音卡死在登录页：



- 原因：
 - 断点太多 + 给一个被调用频率很高的函数（`-[NSString stringByAppendingString:]`）加了断点
- 解决办法：临时去掉断点
 - 等进入抖音主页后，再恢复断点

导致卡死

条件判断断点

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-11 23:16:59

断点的条件判断表达式的写法的心得

关于条件判断的断点的表达式写法的心得：

要用正确的函数

- 条件判断语句：要用正确的函数

- 举例

- objc_alloc_init
 - 其定义是
 - id objc_alloc_init(Class cls)
 - 传入参数是 Class (而不是 Instance)
 - 条件判断的写法
 - 所以只能用，应该用： NSStringFromClass
 - 定义： NSString * NSStringFromClass(Class aClass);
 - 而不应该用： isKindOfClass
 - 定义：是针对 NSObject 的 - (BOOL)isKindOfClass:(Class)aClass;
 - 因为此时 \$x0 是个 class , 不是 Instance = Object
 - 所以结论是：objc_alloc_init的断点的条件判断
 - 正确写法：

```
(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]
```

- 错误写法：

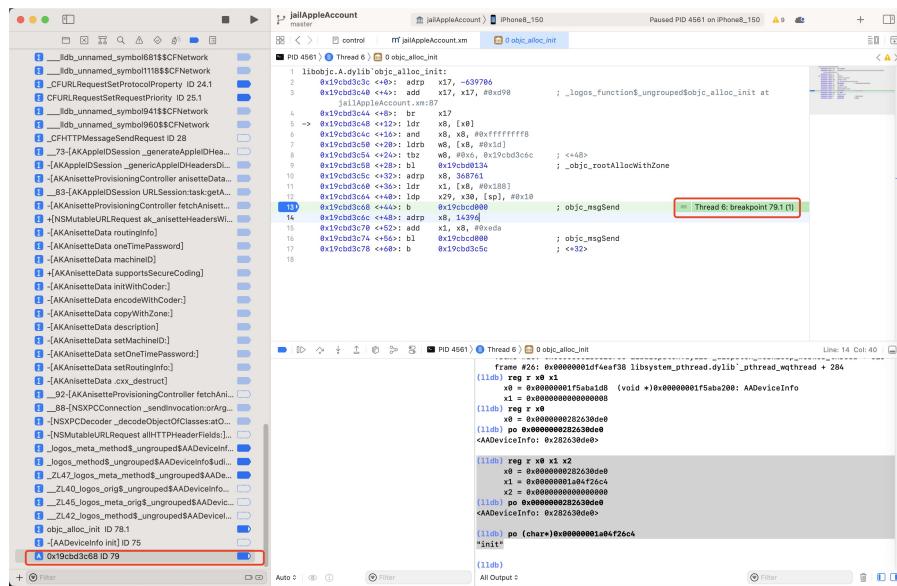
```
(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
```

- objc_alloc_init 函数中的 +44 行的 objc_msgSend

- 背景

- 代码是

- 0x19cbd3c68 <+44>: b 0x19cbcd000 ; objc_msgSend



- 此时: `x0` 是 `AADeviceInfo` 的 `Instance = obj`
- 条件判断的写法
 - 所以不能用: `NSStringFromClass`
 - 只能用 `isKindOfClass`
 - 结论
 - `objc_alloc_init` 函数中的 +44 行的 `objc_msgSend` 的断点的条件判断
 - 正确写法:

```
(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
]
```

- 错误写法:

```
(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]
]
```

最前面加bool类型转换

- 条件判断语句: 往往最前面还要加个布尔类型强制转换 (才能起到判断作用)
 - 无效的写法:

```
[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]
[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
]
```

- 有效的写法:

```
(bool)[NSStringFromClass($x0) isEqualToString: @"AADeviceInfo"]
(bool)[$x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
]
```

借助于lldb确认表达式语法无误

- 条件判断语句: 当不确定代码是否正确时, 可以借助于lldb命令行的po去执行, 看看是否有语法错误

objc_alloc_init函数的条件判断表达式写法

当，不确定自己的写法，比如：

- objc_alloc_init 的断点的条件判断

```
[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
```

是否有误，可以去用 po 试试：

```
(lldb) po [$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
error: expression failed to parse:
warning: user expression 25 :1:2: receiver type 'unsigned long' is not 'id' or interface pointer, consider casting it to 'id'
[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
^~~
error: user expression 25 :1:21: 'objc_getClass' has unknown return type; cast the call to its declared return type
[$x0 isKindOfClass: objc_getClass("AADeviceInfo")]
```

结果提示有误，意思是不清楚 objc_getClass 返回值的类型

所以后来才去加上 class 的类型转换的

```
(bool)[<x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
```

此处再去执行，至少语法上是对的，不报错了：

```
(lldb) po [<x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
nil
```

所以确定，最后的正确写法是：

```
(bool)[<x0 isKindOfClass: (Class)objc_getClass("AADeviceInfo")]
```

objc_alloc_init的+44行汇编代码的条件判断表达式写法

对于：

```
libobjc.A.dylib`objc_alloc_init:
...
-> 0x19cbd3c68 <+ 44 : b      0x19cbcd000          ; objc_msgSend
```

的尝试添加条件判断的断点的过程：

```
(lldb) po (int)strcmp(class_getName($x0), "AADeviceInfo") == 0
error: expression failed to parse:
error: user expression 10 :1:13: 'class_getName' has unknown return type; cast the call to its declared return type
```

```
(int)strcmp(class_getName($x0), "AADeviceInfo") == 0
^-----
```

无法识别 `class_getName` 返回值类型，所以去加上类型转换

```
(lldb) po (int)strcmp((const char *)class_getName($x0), "AADeviceInfo") == 0
error: Execution was interrupted, reason: EXC_BAD_ACCESS (code 1, address=0x4af7fbe24f30).
The process has been returned to the state before expression evaluation.
```

却又无法执行

```
(lldb) po class_getName($x0)
error: Execution was interrupted, reason: EXC_BAD_ACCESS (code 1, address=0x4af7fbe24f30).
The process has been returned to the state before expression evaluation.
```

继续尝试：

```
(lldb) po class_getName($x0)
error: Execution was interrupted, reason: EXC_BAD_ACCESS (code 1, address=0x4af7fbe24f30).
The process has been returned to the state before expression evaluation.
```

发现是 `class_getName` 函数无法正常执行

后来找到定义是：

```
const char * class_getName(Class cls);
```

-》 所以此处是函数用法有误：

```
class_getName 参数是 Class，而此处是 Instance，所以无法正常运行
```

换别的函数 `object_getClassName` 试试

```
(lldb) po [object_getClassName($x0) isEqualToString: @"AADeviceInfo"]
error: Execution was interrupted, reason: EXC_BAD_ACCESS (code 1, address=0x665444160).
The process has been returned to the state before expression evaluation.
```

却也无法运行，但另外发现：

```
(lldb) po object_getClassName($x0)
0x0000000197633c50
```

`object_getClassName` 函数本身，是可以正常运行的

然后找到了定义：

```
const char * object_getClassName(id obj);
```

发现此处是：返回值是 `char *`，而不是 `NSString *`

所以要换用 `char *` 的判断逻辑

```
(lldb) po (char *)object_getClassName($x0)  
"AADeviceInfo"
```

确保可以输出字符串后，然后换用 `strcmp` 去对比

```
(lldb) po strcmp(object_getClassName($x0), "AADeviceInfo") == 0  
error: expression failed to parse:  
error: user expression 16 :1:8: 'object_getClassName' has unknown return type; cast the  
call to its declared return type  
strcmp(object_getClassName($x0), "AADeviceInfo") == 0  
^~~~~~
```

仍然报错，所以也要给 `object_getClassName` 加上返回值类型转换，才可以：

```
(lldb) po strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0  
error: expression failed to parse:  
error: user expression 17 :1:1: 'strcmp' has unknown return type; cast the call to its  
declared return type  
strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0  
^~~~~~
```

但是又报错 `strcmp` 返回值类型无法识别

所以也要加上类型转换

```
(lldb) po (int)(strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0)  
error: expression failed to parse:  
error: user expression 18 :1:7: 'strcmp' has unknown return type; cast the call to its  
declared return type  
(int)(strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0)  
^~~~~~
```

发现搞错了，`int` 类型转换是需要针对 `strcmp`，而不是表达式

```
strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0
```

所以 `int` 类型转换放在 `strcmp` 前面即可：

```
(lldb) po (int)strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0  
true
```

终于是，通过 `po` 辅助而确认了此处的断点的条件判断的正确写法是：

```
(int)strcmp((char *)object_getClassName($x0), "AADeviceInfo") == 0
```

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-11 23:19:12

函数名和地址逻辑一致

关于断点的条件判断的表达式的，对于：

- Symbol=函数名
- Address=地址=汇编代码地址

的写法上，逻辑是一致的：

- 给函数加带条件判断的断点和给某行汇编加带条件判断的断点，逻辑也是一样的

举例

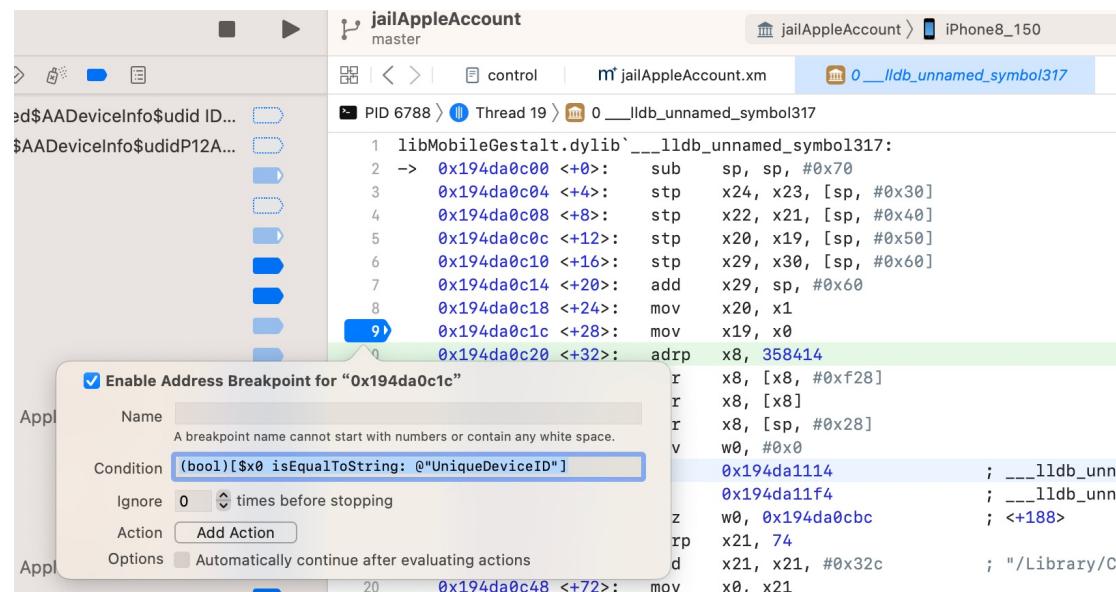
函数MGCopyAnswer和__lldb_unnamed_symbol317的某行汇编代码

- 前提
 - 此处都是去判断 `x0` 寄存器变量的值是否是 `UniqueDeviceID` 的 `NSString`
- 对于
 - 函数： MGCopyAnswer
 - 地址=某行汇编代码 = `__lldb_unnamed_symbol317` 中的 +28 行的：`0x194da0c1c <+28>: mov x19, x0`
- 加条件判断的断点，都可以用一样的写法：

```
(bool)[$x0 isEqualToString: @"UniqueDeviceID"]
```

- 效果
 - 函数： MGCopyAnswer

 A screenshot of the Xcode interface showing the Breakpoints list. A new symbolic breakpoint for the function MGCopyAnswer has been created. The breakpoint's condition is set to `(bool)[[$x0 isEqualToString: @"UniqueDeviceID"]]`. The breakpoint is highlighted with a blue border.
 - 地址=某行汇编代码= `__lldb_unnamed_symbol317` 中的 +28 行的：`0x194da0c1c <+28>: mov x19, x0`



crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新:

2023-07-11 23:23:23

ObjC函数断点

给某个ObjC的类的所有方法都加上断点

- 核心思路：用 -r 正则
- 语法解释

```
-r <regular-expression> ( --func-regex <regular-expression> )
Set the breakpoint by function name, evaluating a
regular-expression to find the function name(s).
```

- 举例

```
breakpoint set -r '\[ClassName .*]$'

br s -r '\[UIViewController .*]$" -c '(BOOL)[(id)$arg1 isKindOfClass:[CustomVC class]]'
```

如何搞懂得到正确的ObjC函数名

想要通过函数名给ObjC函数去加断点之前，先要：搞懂ObjC函数名

举例说明：

对于函数：

```
@interface AWEUserRecommendFollowButton : AWEUIListCellActionButton < AWEUserRecommendFollowButtonProtocol

. . .
- (void)updateWithFollowStatus:(long long)arg1 followerStatus:(long long)arg2 preferredTitle:(id)arg3;
- (void)updateWithFollowStatus:(long long)arg1 followerStatus:(long long)arg2;
```

想要去写出正确的函数名=可以用于后续Xcode中加断点的ObjC函数名：

- 正确写法

```
- [AWEUserRecommendFollowButton updateWithFollowStatus:followerStatus:preferredTitle:]
- [AWEUserRecommendFollowButton updateWithFollowStatus:followerStatus:]
```

- 具体处理过程

去掉函数名，在函数参数往后，其中的类型和变量名

比如：

```
- (void)updateWithFollowStatus:(long long)arg1 followerStatus:(long long)arg2;
```

去掉 `updateWithFollowStatus` 函数的参数往后的的变量类型和变量名，就是：

```
updateWithFollowStatus:(long long)arg1 followerStatus:(long long)arg2;
```

->

```
updateWithFollowStatus: followerStatus:
```

然后去掉多余的空格：

```
updateWithFollowStatus followerStatus:
```

再去加上前面的类：

- 语法是：

```
+或- 中括号 内部是类名
```

◦ ->

```
+/- [ClassNameOrInstanceName xxx]
```

逻辑是：

- 对于 Class = 类 是 = 加号
- 对于 Instance = 实例 是 = 减号

此处是Instance实例，所以是 减号

```
- [AWEUserRecommendFollowButton xxx]
```

- 注意：

- 加号 或减号 ， 和后面的 左中括号 = [中间，不能有空格
 - 即，下面写法是错误的：

```
- [AWEUserRecommendFollowButton xxx]
```

最后再去把函数部分加进来，就是：

```
- [AWEUserRecommendFollowButton updateWithFollowStatus:followerStatus:]
```

就是正确的，完整的函数名，可以加到Xcode中的断点了。

给被hook的函数加断点

对于Logos的hook函数的代码：

- `hook_dyld.xm`

```
%hookf(const char*, _dyld_get_image_name, uint32_t image_index){
    ...
    const char* imgName = %orig(image_index);
```

想要去XCode中加函数的符号断点，则：

- 不是直接加原始函数名： `_dyld_get_image_name`
- 而是要找到Logos的hook后的实际的函数名：

- `hook_dyld.mm`

```
__unused static const char* (_logos_orig$_ungrouped$_dyld_get_image_name)(uint
32_t image_index); __unused static const char* _logos_function$_ungrouped$_dyld
_get_image_name(uint32_t image_index){

    ...
    _logos_orig$_ungrouped$_dyld_get_image_name(image_index);
    ...
}
```

->

- 原函数名： `_dyld_get_image_name`
 - Logos处理后=hook后
 - 原函数变成： `_logos_orig$_ungrouped$_dyld_get_image_name`
 - hook的被替换的函数叫做： `_logos_function$_ungrouped$_dyld_get_image_name`

-> 所以想要去加断点时，应该用：

```
_logos_function$_ungrouped$_dyld_get_image_name
```

- Xcode中添加该断点的效果

◦

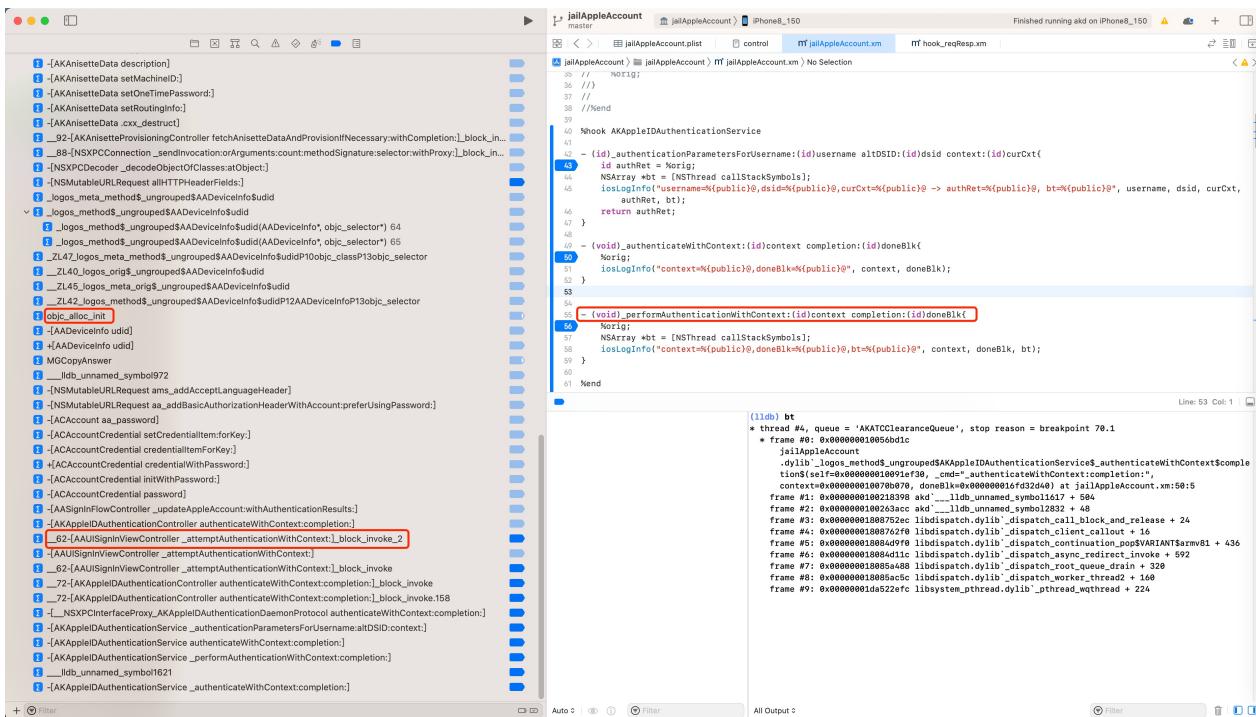
Xcode

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-07-10 16:05:48

如何用Xcode去调试多个函数，且确保断点能触发到

此处经过长期调试，发现一个心得：

- 需求：希望能在同一个 Xcode 的项目（iosOpenDev 的插件的 Xcode 项目）中，能同时调试多个函数的断点
 - 比如：
 - `objc_alloc_init`：函数断点 + hook 代码断点
 - `-[AKAppleIDAuthenticationService _performAuthenticationWithContext:completion:]`：hook 代码断点
 - `__62-[AAUISignInViewController _attemptAuthenticationWithContext:]_block_invoke_2`：函数断点



但是之前调试起来却又发现，会遇到各种问题：

- 有时候是
 - `-[AKAppleIDAuthenticationService _performAuthenticationWithContext:completion:]`：hook 代码断点，能触发
 - 但是 `__62-[AAUISignInViewController _attemptAuthenticationWithContext:]_block_invoke_2`：函数断点无法触发
- 有时候却又是
 - `__62-[AAUISignInViewController _attemptAuthenticationWithContext:]_block_invoke_2`：函数断点，能触发
 - 但是 `-[AKAppleIDAuthenticationService _performAuthenticationWithContext:completion:]`：hook 代码断点，无法触发

等等各种奇怪、特殊的现象

最后经过总结发现：

- 经验心得：Xcode中确保断点能加上且能触发

后，基本上确认：

其核心有2个：

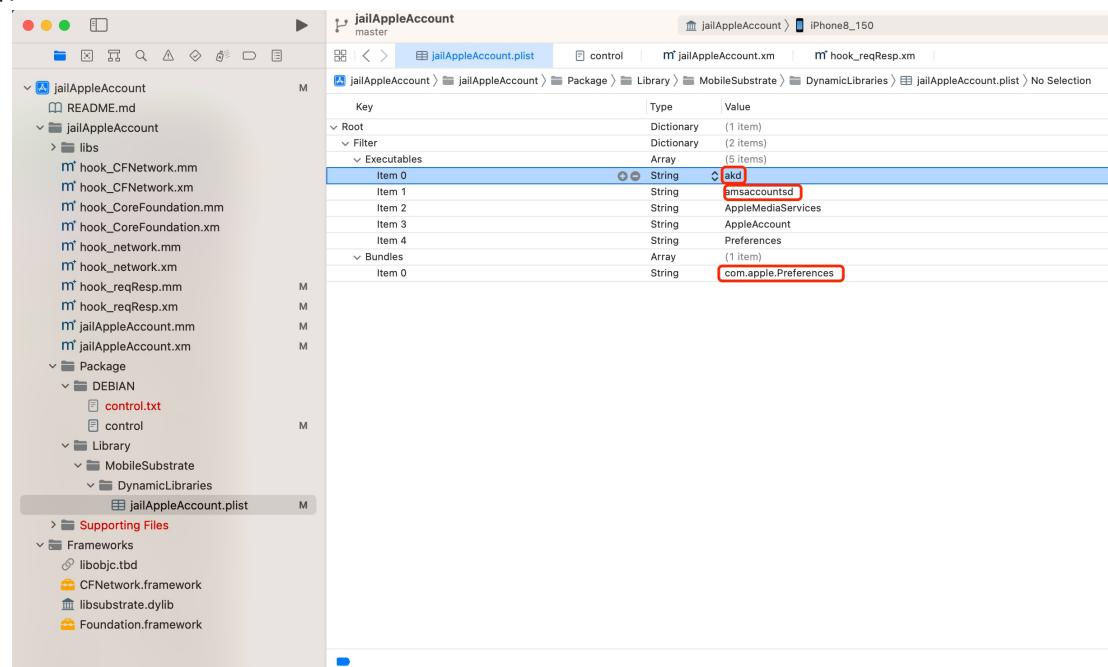
- 能搞清楚：该函数所被触发的目标
 - 确保Xcode的调试目标是：该函数所被触发的目标

但是实际上，往往却：

很难搞清楚 某函数 真正被触发 所属的目标

所以此时的心得就是：

- (1) 给 Xcode 的 (iosOpenDev 插件) 的hook目标，多加上几个目标
 - 目的：基本上能确保覆盖到被调试的多个函数，所涉及到的不同的目标
 - 效果：



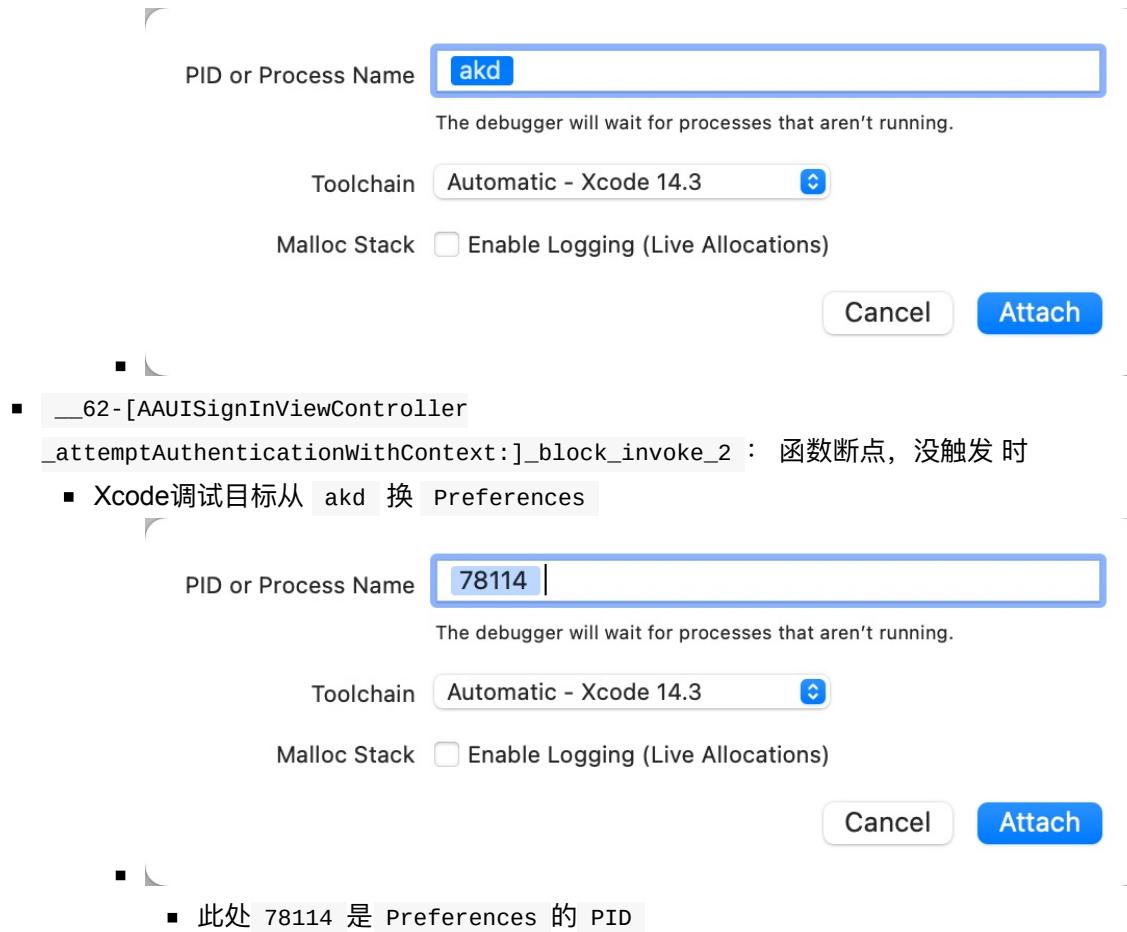
■ 包括了：

- Executable = 二进制 的： akd 、 amsaccountsds
- Bundle = app 的： com.apple.Preferences

- (2) Xcode 调试时：如果断点（没加上或）没触发，及时去试试，换其他调试目标-》往往就可以触发断点了

◦ 比如

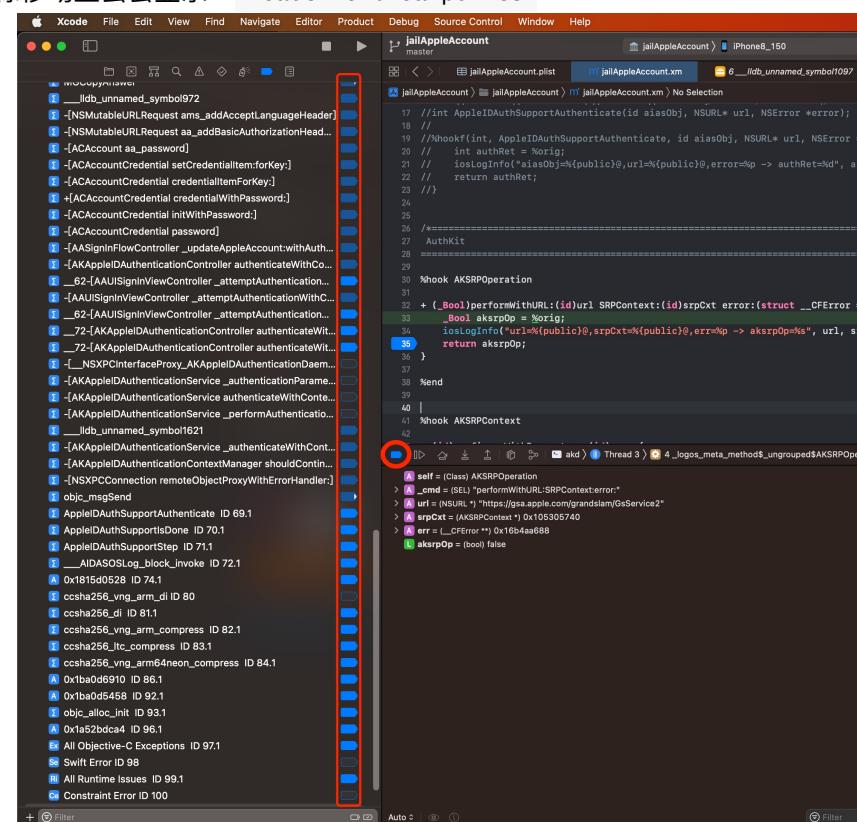
- -[AKAppleIDAuthenticationService
_performAuthenticationWithContext:completion:] : hook代码断点，没触发时
- Xcode 调试目标从 Preferences 换 akd



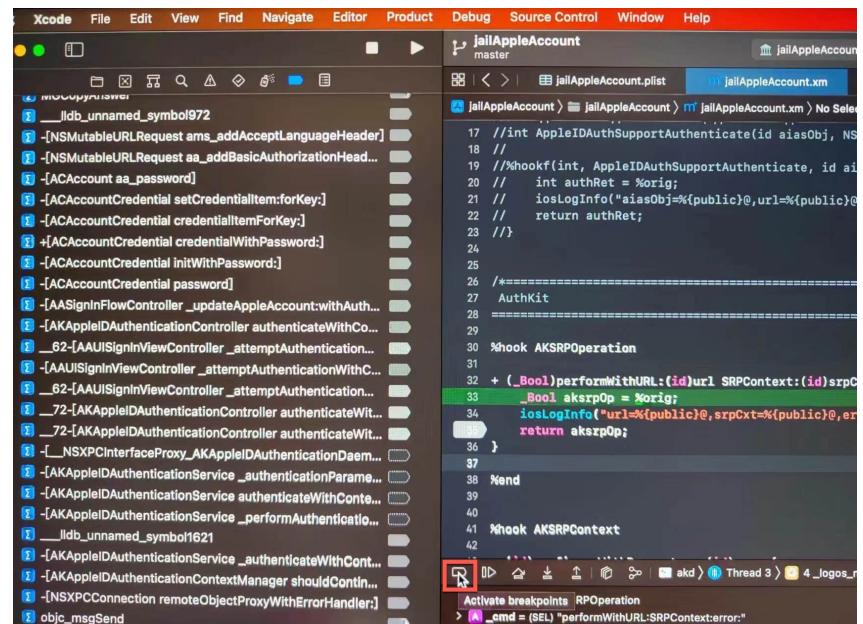
crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:
2023-07-10 17:44:29

临时关闭所有断点

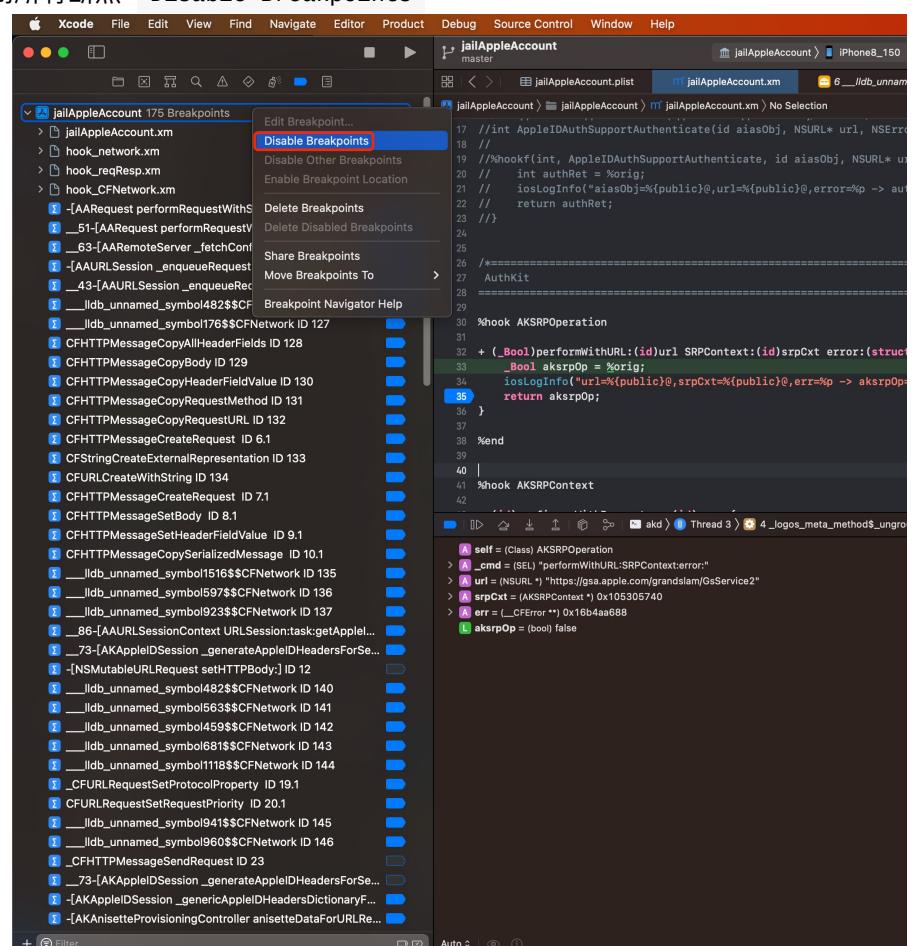
- (一键) 临时关闭所有断点 + 后续恢复所有断点
 - 背景: 有个需求, 调试时, 某些时候
 - 比如
 - 希望继续让程序尽快运行, 但又不希望已加的多个断点, 对程序产生影响
 - 比如
 - 某些会触发多次 (设置加入了额外判断条件) 的断点, 会降低程序运行速度, 甚至导致程序卡死
 - 所以希望, 可以临时的, 暂时禁用所有断点, 且待会在需要时, 可以恢复所有断点
 - 即: 临时禁用所有断点 (且之后可以恢复被临时禁用的所有断点)
 - 解决办法
 - 推荐方法
 - Xcode中下方的调试控制面板中的: 断点控制按钮, 点击可以实现
 - 对应的效果
 - Deactive breakpoints =临时暂停所有断点 (且保留之前的状态)
 - 鼠标移动上去会显示: Deactive breakpoints



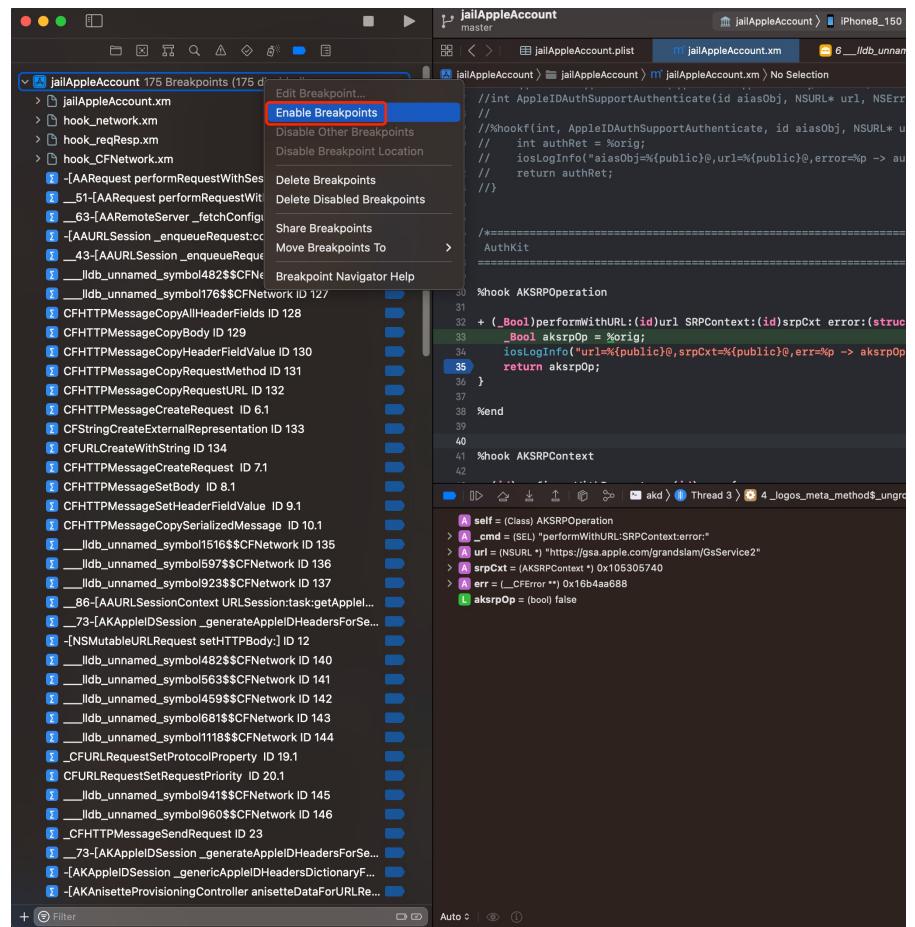
- Active breakpoints =激活 (重新启用=重新开启) 所有断点 (且恢复之前的状态)
 - 鼠标移动上去会显示: Active breakpoints



- 优势是：可以保持之前所有的断点的状态（是激活还是非激活）
- 凑合用（但效果不够好）的方法
 - Xcode中的 Breakpoint Navigator ->右键对应（正在调试的项目），即可出现右键菜单，支持
 - 关闭或开启所有断点
 - 关闭所有断点= Disable Breakpoints



- 恢复（开启）所有断点 = Enable Breakpoints



- 缺点是：是 Disable / Enable ，关闭或开启，就：丢失了之前的各个断点的状态
- 举例：
 - 之前有175个断点，其中只有85个是激活状态，剩余均为未激活，用了 Disable / Enable 后，结果全部175个断点就全是激活了（其中激活了很多，想要暂时关闭，但却又不想要删除的，原先是非激活状态的断点）

crifan.org, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新：
2023-07-10 16:07:40

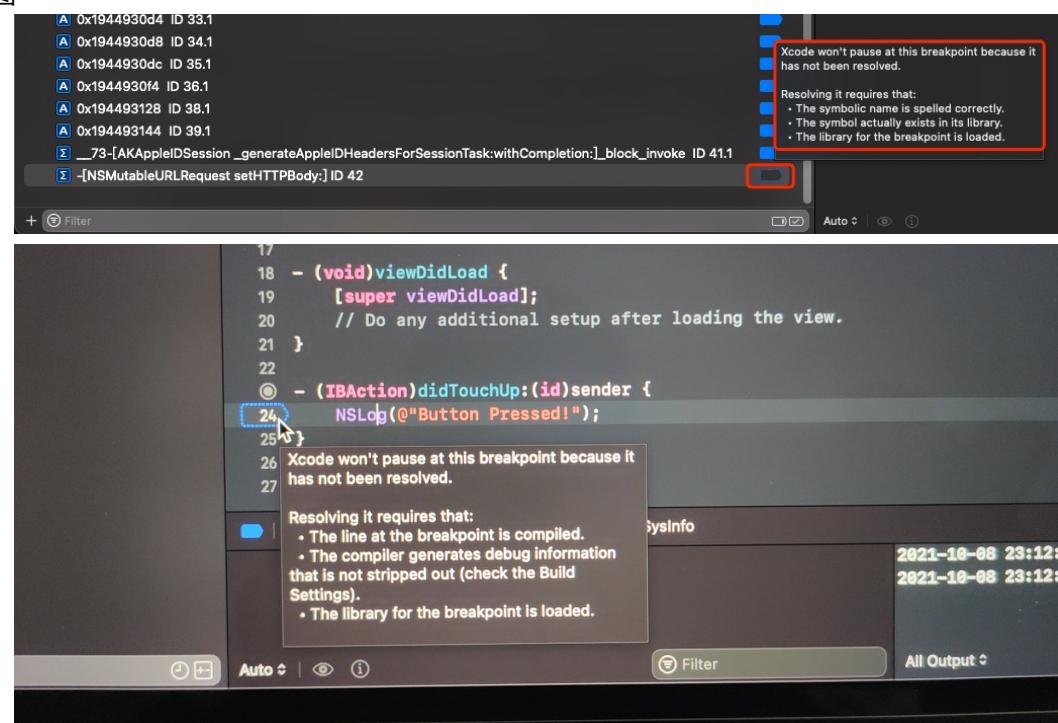
Xcode中判断断点是否添加成功

Xcode图形界面中，当给普通的 Symbolic Breakpoint 中输入 Symbol = 符号名 = 函数名 后，Xcode内部会自动去尝试匹配和查找对应符号

所以会出现：

- 有时候，过了几秒后，断点右边的状态显示
 - 不是普通的：成功加上断点的=显示的底色是实心的蓝色
 - 而是显示=特殊的，蓝色的虚线框，底色是空白（Dark模式时的深色）

■ 如图



■ 鼠标移动上去，会有提示

```
Xcode won't pause at this breakpoint because it has not been resolved.

Resolving it requires that:
  The symbolic name is spelled correctly.
  The symbol actually exists in its library.
  The library for the breakpoint is loaded.
```

■ 此种状态，就表示

- 断点没有添加成功 = 断点添加失败 = 断点加不上 = 掉断点

Xcode中断点加不上

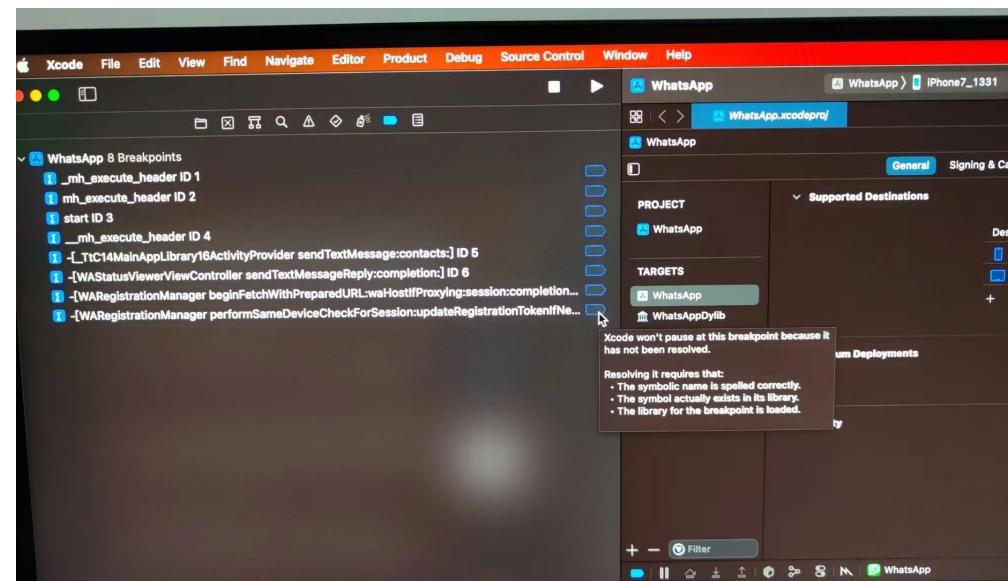
Xcode中断点加不上的原因

- 一般 断点添加失败 = 加不上断点 = 断点加不上，有几种常见的原因
 - The symbolic name is spelled correctly = 函数名有误
 - 一种是：笔误型的，函数名写错了
 - 举例
 - 把ObjC的加号误写成了减号
 - + [AKSRPOperation performWithURL:SRPContext:error:] -> -
 - [AKSRPOperation performWithURL:SRPContext:error:]
 - 把ObjC的函数末尾多写了个冒号：
 - [__NSCFConstantString stringByAppendingString:]: -> -
 - [__NSCFConstantString stringByAppendingString:]
 - The symbol not exists in the library = 符号不存在于当前（所有的已加载的）库中
 - 比如
 - ObjC中，调试 ak d时，函数 -[AKAppleIDAuthenticationContextManager shouldContinueWithAuthenticationResults:error:forContextID:completion:] 加不上断点
 - 原因，该函数其实是存在于另外的二进制=库=app: Preference 中，需要把调试目标从 akd 改为 Preferences 才行，详见：[断点能加上且能触发](#)
 - The library for the breakpoint is not loaded = 符号所在的二进制没有被加载
 - iOS的ObjC的二进制中，本身就没有该函数名=符号，因为二进制中的符号表已经被去掉了

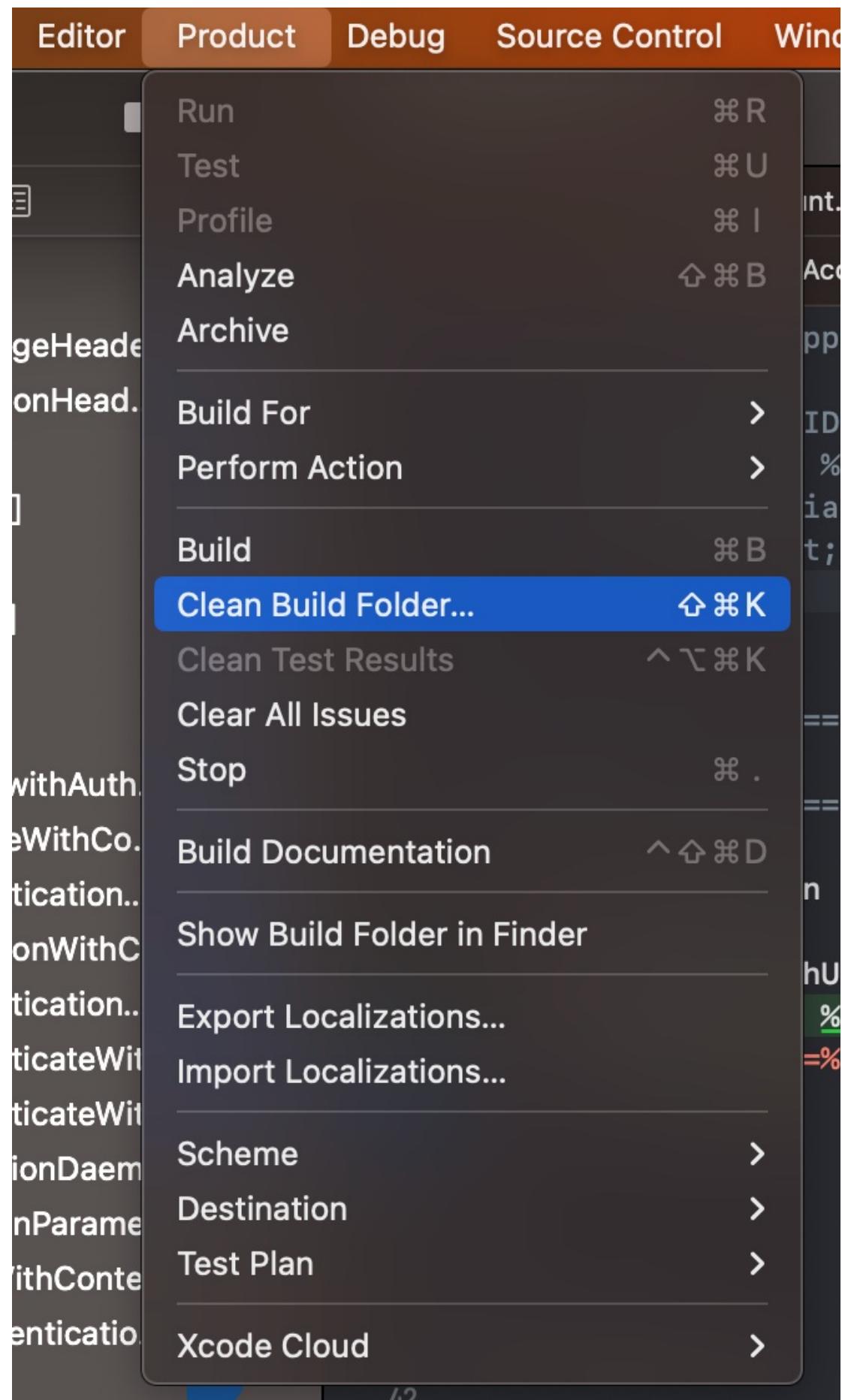
Xcode中断点加不上的解决办法

对应的，Xcode中，加不上断点时，一些常见的解决办法：

- 原因：iOS的ObjC的二进制中，符号表已经去掉了
 - 举例
 - 抖音
 - 现象： AwemeCore 中，已经没有符号表了，所以去给函数 -[AWEUserRecommendMultiTagsView followBtnClicked:] 加断点，加不上
 - 
 - WhatsApp
 - 现象： WhatsApp 中，去给函数 -[WARegistrationManager performSameDeviceCheckForSession:updateRegistrationTokenIfNecessary:withCompletion:] 加断点，但加不上

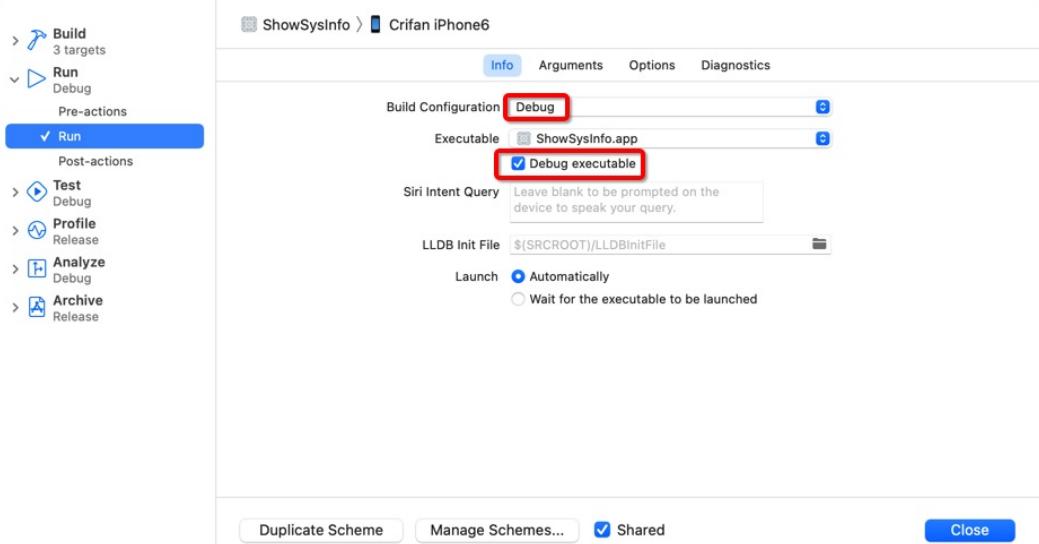


- 解决办法: [恢复符号表](#)
- 原因: Xcode本身有bug
 - 解决办法: 清除当前已编译内容
 - 具体步骤: Xcode -> Product -> Clean Build Folder
 - 注: 如果需要, 可以多试几次

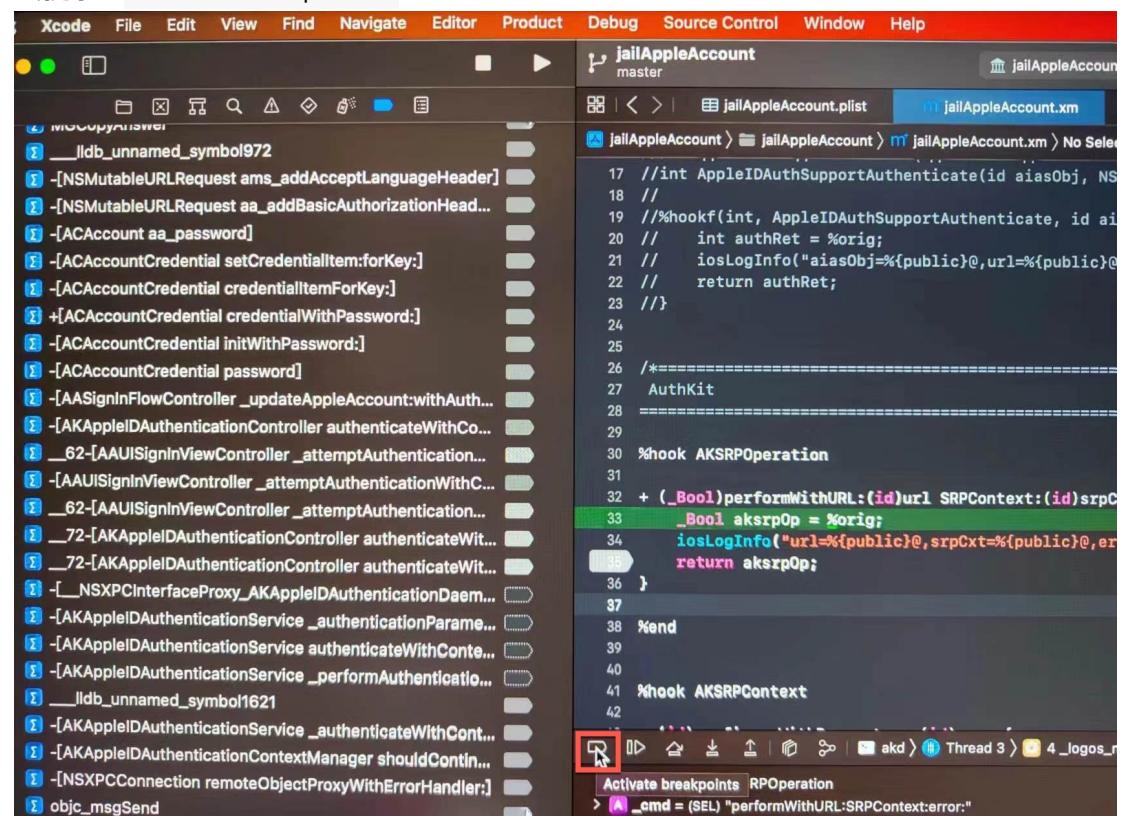


- 原因：之前（无辜，不小心）关闭了 调试

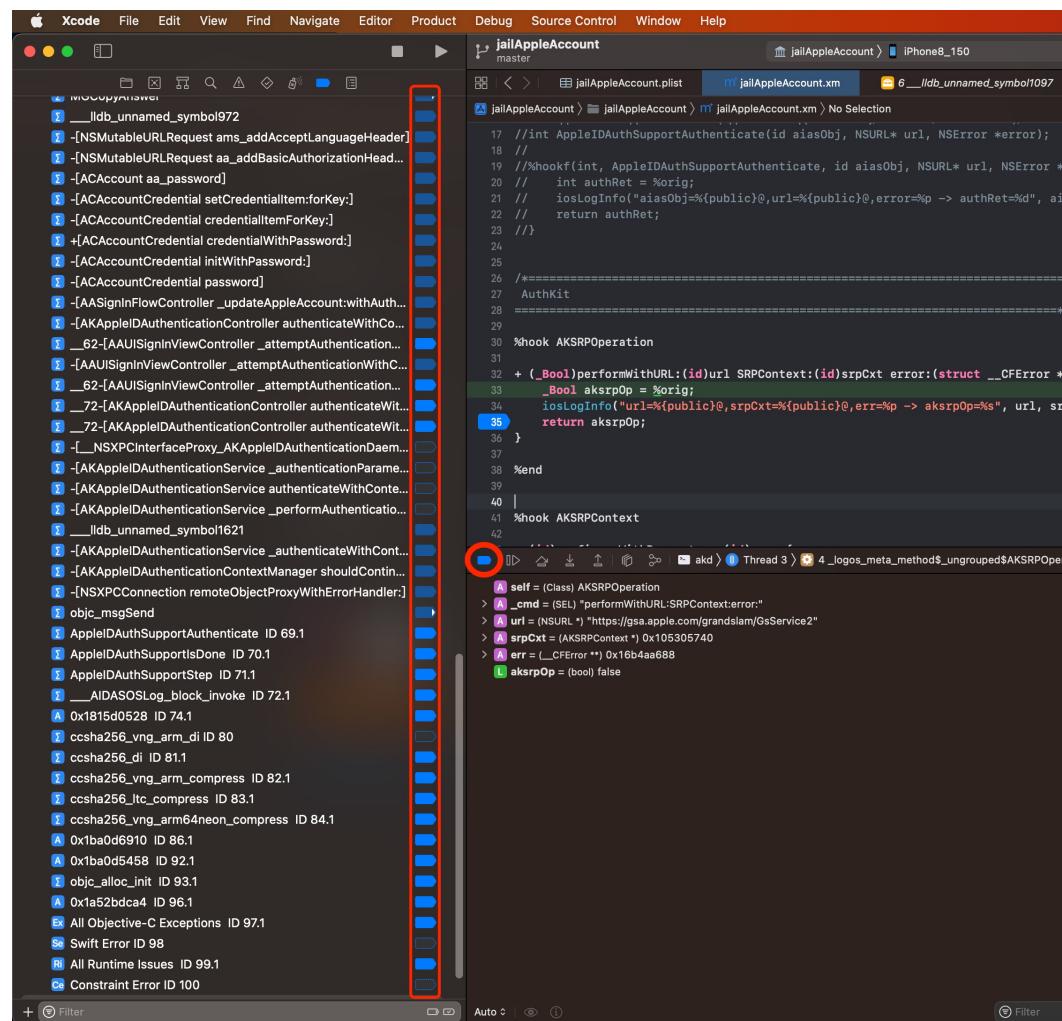
- 解决办法：（重新）开启调试
- 具体步骤：`Xcode -> Product -> Scheme -> Edit Scheme -> Run -> Info`
 - 把 Build Configuration 设置为 Debug，且勾选 Debug executable



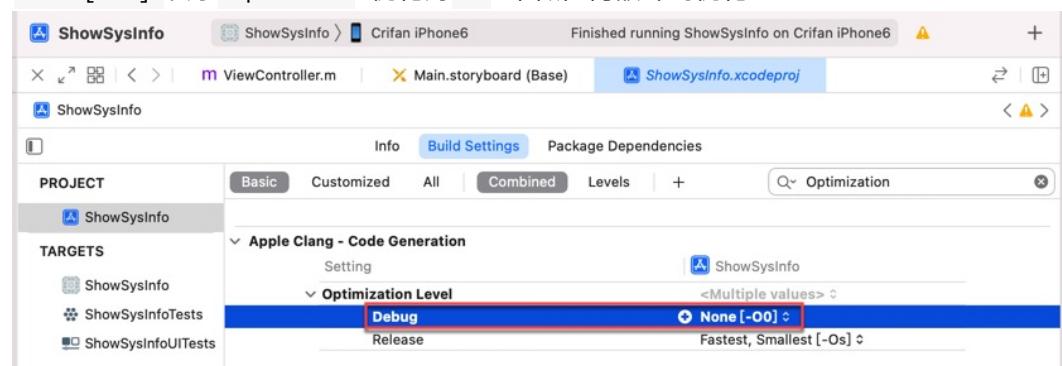
- 原因：之前（不小心）关闭了（临时）所有的断点
 - 此时：如果切换到断点试图，也会看到，所有断点已（从之前的蓝色）变成灰色了 + 鼠标移动上去，会提示：`Active Breakpoints`



- 解决办法：点击该（临时关闭或开启所有的断点）按钮，开启所有断点
 - 点击后，所有断点就恢复蓝色了



- 原因: Debug 调试模式下, 调试信息被优化掉了
 - 解决办法: 恢复 Debug 调试模式下, 不做任何优化
 - 具体步骤: Xcode -> PROJECT -> <YourProjectName> -> Build Settings -> Apple Clang -> Code Generation -> Optimization Level -> Debug 设置为 None[-O0]
 - 注: None[-O0] 表示 Optimize = 优化为 0 = 不做任何额外的优化



Xcode中确保断点能加上且能触发

此处整理经验心得：Xcode中确保断点能加上且能触发

- 暂且不包括（debugserver+lldb中） lldb 加断点的情况，只考虑Xcode调试期间的断点相关情况

先介绍背景：

- Xcode中断点类型
 - Xcode通过函数名加断点=函数名的断点
 - Xcode中的插件hook代码的断点=hook代码的断点

再介绍具体心得：

- Xcode中确保断点能加上且能触发
 - 概述
 - 函数名的断点
 - 加不上断点
 - 原因
 - 所属二进制中不存在该symbol符号
 - 解决办法：调试找出底层函数（lldb无名函数），给底层函数加断点
 - 能加上断点，但没触发
 - 原因
 - Xcode调试的目标不是函数所属的被调用的目标
 - 解决办法：确保Xcode调试目标是=等于函数被调用的所属目标
 - hook代码的断点
 - 加不上断点
 - 原因
 - 加了hook代码的dylib的最新插件没被加载
 - 解决办法：确保最新的dylib被加载
 - 函数被触发的目标和Xcode调试目标不一致
 - 解决办法：确保函数被触发目标和Xcode调试目标一致
 - 具体办法：把Xcode调试目标改为函数被触发的目标
 - 能加上断点，但没触发
 - 原因
 - 函数名搞错了
 - 解决办法：使用正确的函数
 - hook目标和Xcode调试目标都不对
 - 解决办法：把hook的目标和Xcode调试目标，都改为：函数被触发的目标

- 详解

- 函数名的断点
 - 加不上断点
 - 可能原因
 - 所属二进制中不存在该symbol符号
 - 举例
 - -[AKAppleIDAuthenticationService

_authenticateWithContext:completion:]

■ 现象



■ 如何验证: image lookup找不到该函数

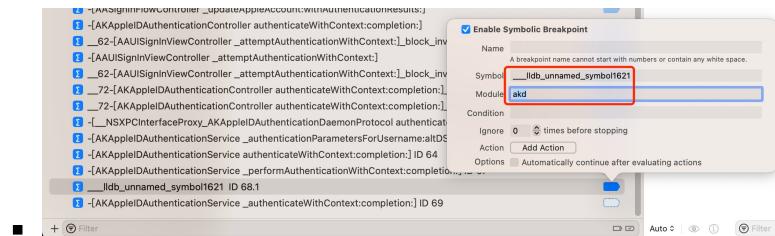
```
(lldb) image lookup -vn "[AKAppleIDAuthenticationService _authenticateWithContext:completion:]"
(lldb)
```

■ 解决办法: 经调试, 找到底层真正的函数, 此处的lldb的无名的函数: akd 的 __lldb_unnamed_symbol1621 , 给此lldb无名函数加断点

■ 写法

- Symbol : __lldb_unnamed_symbol1621
- Module : akd

■ 效果



■ 能加上断点, 但没触发

■ 可能原因

- Xcode调试的目标 不是 函数所属的被调用的目标

■ 举例

```
■ __62-[AAUISignInViewController _attemptAuthenticationWithContext:]_block_invoke_2
```

■ 现象



■ 细节:

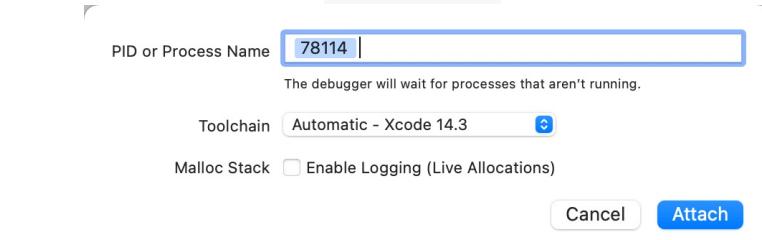
- Xcode调用的目标: akd
- 函数 (__62-[AAUISignInViewController _attemptAuthenticationWithContext:]_block_invoke_2)
被调用的所属目标: Preferences

■ 解决办法

- 确保Xcode调试目标 是 函数被调用的所属目标 -> 此处都

是: Preferences

- 具体步骤: 让Xcode调试 Preferences (而不是akd)



■ 此处的 78114 是 Preferences 的 PID

- -[NSMutableURLRequest ams_addAcceptLanguageHeader]

■ 现象

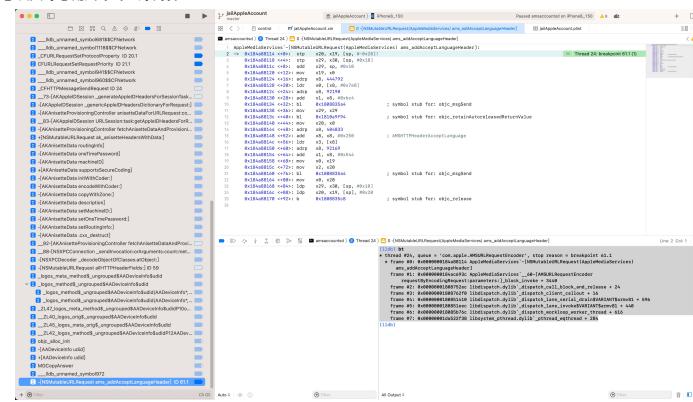
- hook插件 jailAppleAccount 中, hook目标

是: com.apple.Preferences , 以及Xcode调试Preferences, 但是断点没触发

■ 解决办法

- 把Xcode的调试目标改为 amsaccountsds

■ 即可顺利触发断点



■ 具体解释

- 后来确认此处函数 -[NSMutableURLRequest ams_addAcceptLanguageHeader] 所属的二进制是 AppleMediaServices
- 所以要去把调试目标换成 (和 AppleMediaServices 密切相关的) AppleMediaServices.framework 的 amsaccountsds

◦ hook代码的断点

- 加不上断点

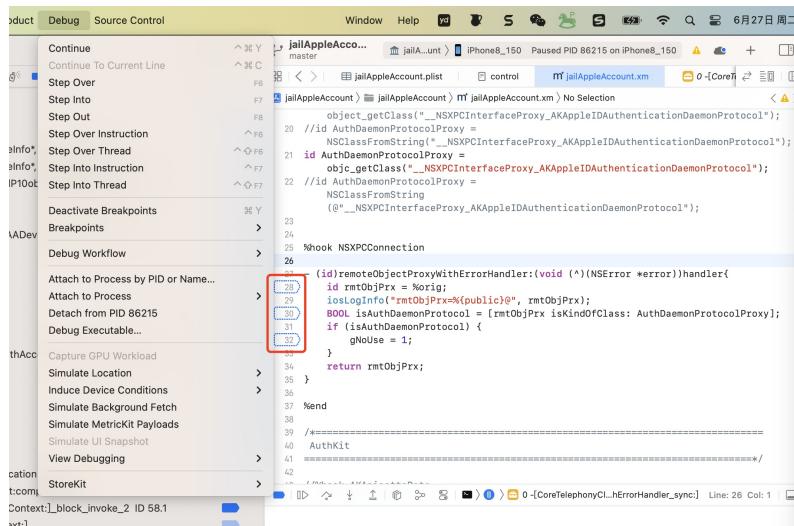
■ 可能原因

- 加了hook代码的dylib的最新插件没被加载

■ 举例

- -[NSXPCCConnection remoteObjectProxyWithErrorHandler:]

■ 现象:



■ 注：86215 是 Preferences 的 PID

■ 此时：Xcode 调试目标=被 hook 目标：Preferences

■ 原因：包含了新版的hook代码的dylib并没有被加载

■ 细节：此处Xcode中iOSOpenDev的dylib插件，编译安装后，虽然 Respring =重启桌面，但是其实 Preferences = 设置 app，并没有重启，导致包含了新版的hook代码的dylib并没有被加载

■ 解决办法：重启此处被hook目标 Preferences

■ 具体步骤：杀掉 Preferences，重启 Preferences

■ 确保：最新的dylib被加载

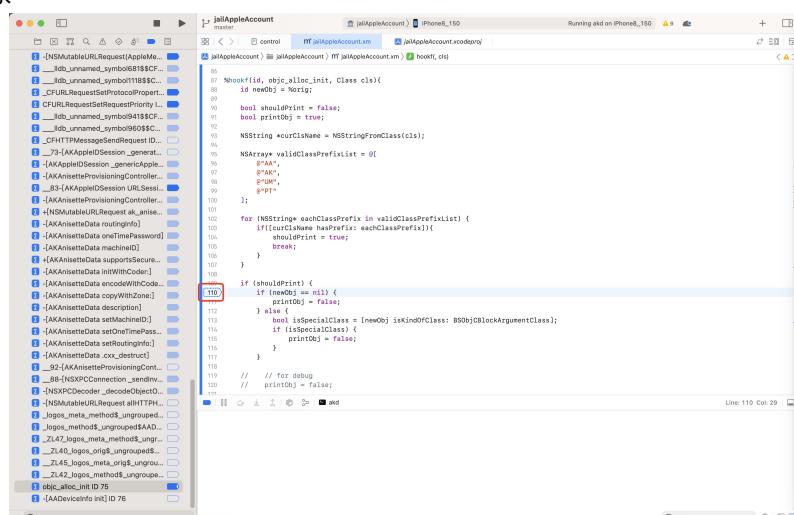
■ 详见：背景知识

■ 函数被触发的目标 和 Xcode调试目标 不一致

■ 举例

■ objc_alloc_init

■ 现象



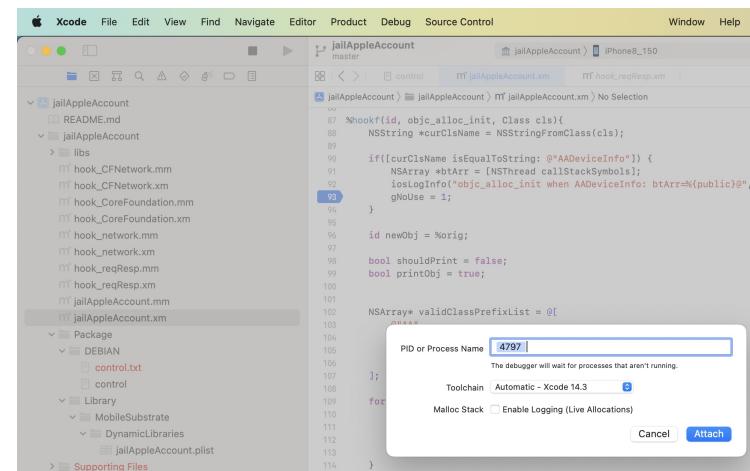
■ 细节

■ 函数objc_alloc_init被触发的目标是：Preferences

■ Xcode调试的目标：akd

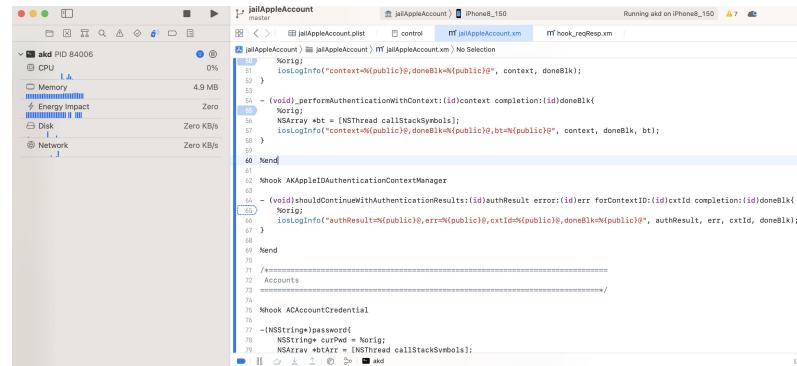
■ 解决办法：

■ 确保Xcode调试目标 和 函数被触发目标 一致，此处都是 Preferences



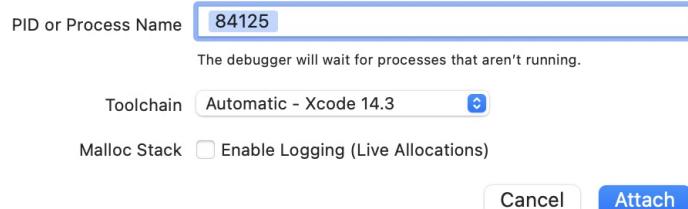
■ 注：此处 4797 是 Preferences 的 PID

- - [AKAppleIDAuthenticationContextManager
shouldContinueWithAuthenticationResults:error:forContextID:completion:]
- 现象



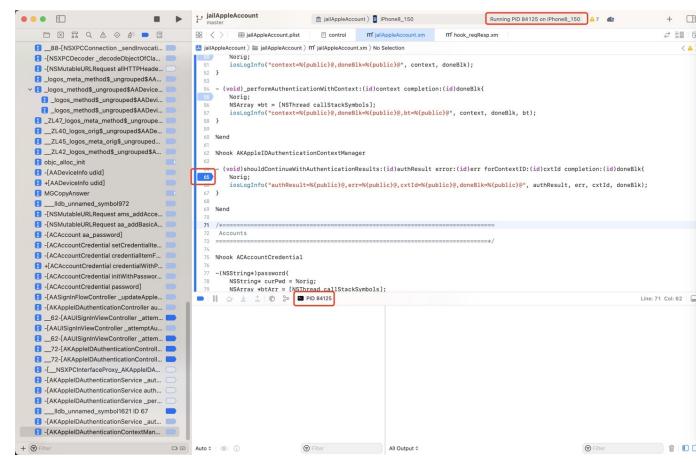
■ 细节：

- Xcode 调试目标：akd
- 函数被调用到的目标：Preferences
- 解决办法：
- 改 Xcode 调试目标为 Preferences

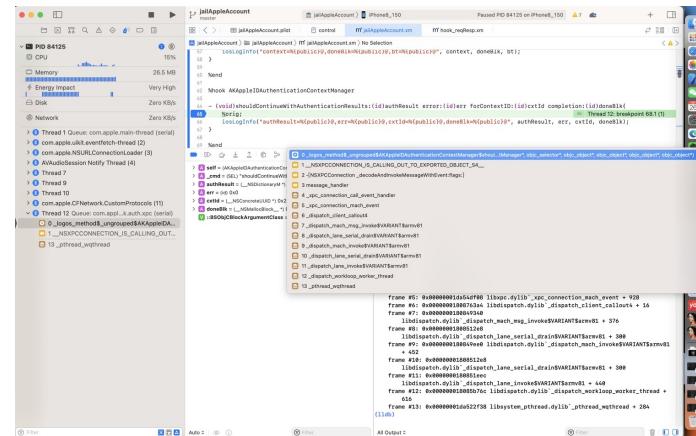


■ 注：此时 Preferences 的 PID 是 84125

- 效果：即可
- 能加上 hook 函数的断点



■ 且hook函数的断点可以触发



■ 能加上断点，但没触发

- 可能原因
 - 函数名搞错了
 - 举例

- - [AKAppleIDAuthenticationService
_authenticationParametersForUsername:altDSID:context:]

■ 现象



■ 原因

- 调试期间，看错函数名了，误把

- - [AKAppleIDAuthenticationService
_authenticateWithContext:completion:]

- 看错成：

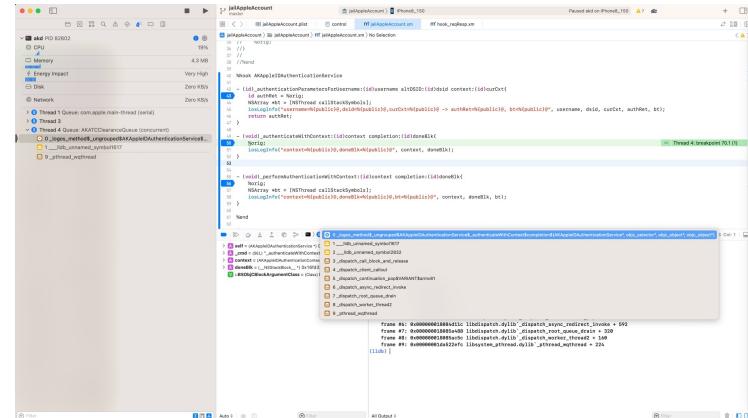
- - [AKAppleIDAuthenticationService
_authenticationParametersForUsername:altDSID:context:]

■ 解决办法：

- 改为此处正确的函数：

- [AKAppleIDAuthenticationService
_authenticateWithContext:completion:]

- 去加hook代码，加断点



- -》最后就可以正常触发hook代码的断点了

- hook目标和Xcode调试目标都不对

- 举例

- [AKAppleIDAuthenticationService
_performAuthenticationWithContext:completion:]

- 现象



- 细节：函数 -[AKAppleIDAuthenticationService

- _performAuthenticationWithContext:completion:] 是在 akd`` (AuthKit 的 daemon程序) 中被触发的

- 此时的

- hook目标是： Preferences
 - Xcode调试目标： Preferences

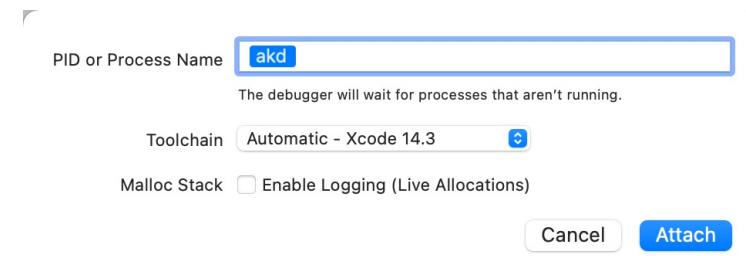
- 解决办法

- 把hook目标改为： akd

- 注：iosOpenDev 中的插件，可以加多个hook目标，所以此处：额外再加上akd即可

Key	Type	Value
Root	Dictionary	(1 item)
Filter	Dictionary	(2 items)
Executables	Array	(5 items)
Item 0	String	akd
Item 1	String	amsaccounts
Item 2	String	AppleMediaServices
Item 3	String	AppleAccount
Item 4	String	Preferences
Bundles	Array	(1 item)
Item 0	String	com.apple.Preferences

- 把Xcode调试目标改为： akd



crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:
2023-07-10 15:45:11

lldb中的无名函数

iOS逆向期间，常会遇到： `__lldb_unnamed_symbolxxx` 的函数，其中 `xxx` 是数字编号

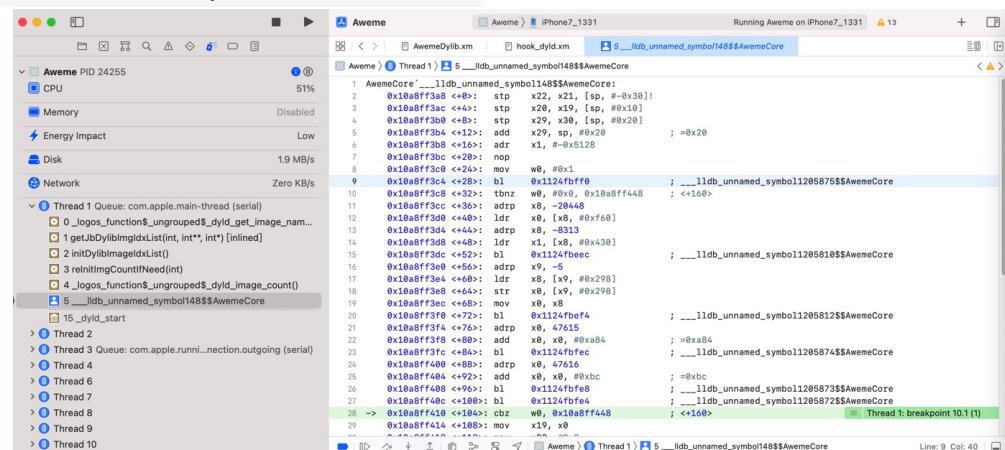
根据经验，可以分为2类：

- lldb无名函数

- 函数名（末尾）中包含模块名的：`__lldb_unnamed_symbolxxx$BinaryFileName`

- 举例

- `__lldb_unnamed_symbol12575$akd`
- `__lldb_unnamed_symbol148$AwemeCore`



- 函数名中不包含模块名的

- 举例

- `__lldb_unnamed_symbol1972`

对此，给这类函数加断点，略有不同：

- 给函数名（末尾）中包含模块名的 加断点：

- 方式1：直接用函数名加断点即可
 - 说明：因为函数名已包含模块，所以能定位到对应函数
- 方式2：计算出函数实际地址，再通过实际地址加断点

- 给函数名中不包含模块名的 加断点

- 需要指定模块名才可以
 - 具体做法，详见：[指定模块](#)

下面举例说明：

举例

函数名（末尾）中包含模块名的lldb无名函数

`__lldb_unnamed_symbol12575$akd`

想要给 `__lldb_unnamed_symbol12575$akd` 加断点：

方式1：直接通过（无名函数的）函数名

- Xcode图形界面中
 - 添加断点 `__lldb_unnamed_symbol12575$$akd`
- lldb命令行中

```
(lldb) b __lldb_unnamed_symbol12575$$akd
Breakpoint 1: where = akd`__lldb_unnamed_symbol12575$$akd, address = 0x00000000102e70460
```

方式2：通过无名函数的实际地址

如[得到函数的实际地址](#)所介绍，先去：

- 计算出模块的ALSR基地址

```
(lldb) image list -o -f | grep akd
[ 0] 0x00000000002dd0000 /System/Library/PrivateFrameworks/AuthKit.framework/akd(0x0000000102dd0000)
```

得到：

- akd的ALSR：`0x00000000002dd0000`

再去：

- 加了VM address后是：`0x00000000102dd0000`

当前无名函数的二进制内偏移量=文件偏移量=file offset：`0xA0460`

算出最终的：

- 函数的实际地址：`0x102E70460 == 0x00000000102e70460`

如果要去通过地址加断点，就是：

```
b 0x00000000102e70460
br s -a 0x00000000102e70460
```

恢复符号表

对于之前所介绍的 [lldb无名函数](#)，如果有机会，可以去恢复符号表，就可以：

把相关lldb无名函数，变成对应的有名字的函数了。

由此，则就可以：通过函数名去添加断点了。

举例

抖音的AwemeCore

给抖音的AwemeCore恢复符号表

用[HeiTanBc版本的restore-symbol](#)去恢复符号表：

具体步骤：

下载、编译、确认：

```
git clone --recursive https://github.com/HeiTanBc/restore-symbol.git
cd restore-symbol
make
./restore-symbol
```

使用：

```
→ AwemeCore.framework /Users/crifan/dev/DevSrc/iOS/restore-symbol/HeiTanBc/restore-symbol/restore-symbol AwemeCore_noSymbol -o AwemeCore_restoredSymbol_HeiTanBc
  Start
Scan OC method in mach-o-file.
Scan OC method finish.
restore 329610 symbols
  Finish
```

抖音的AwemeCore恢复符号表后的效果

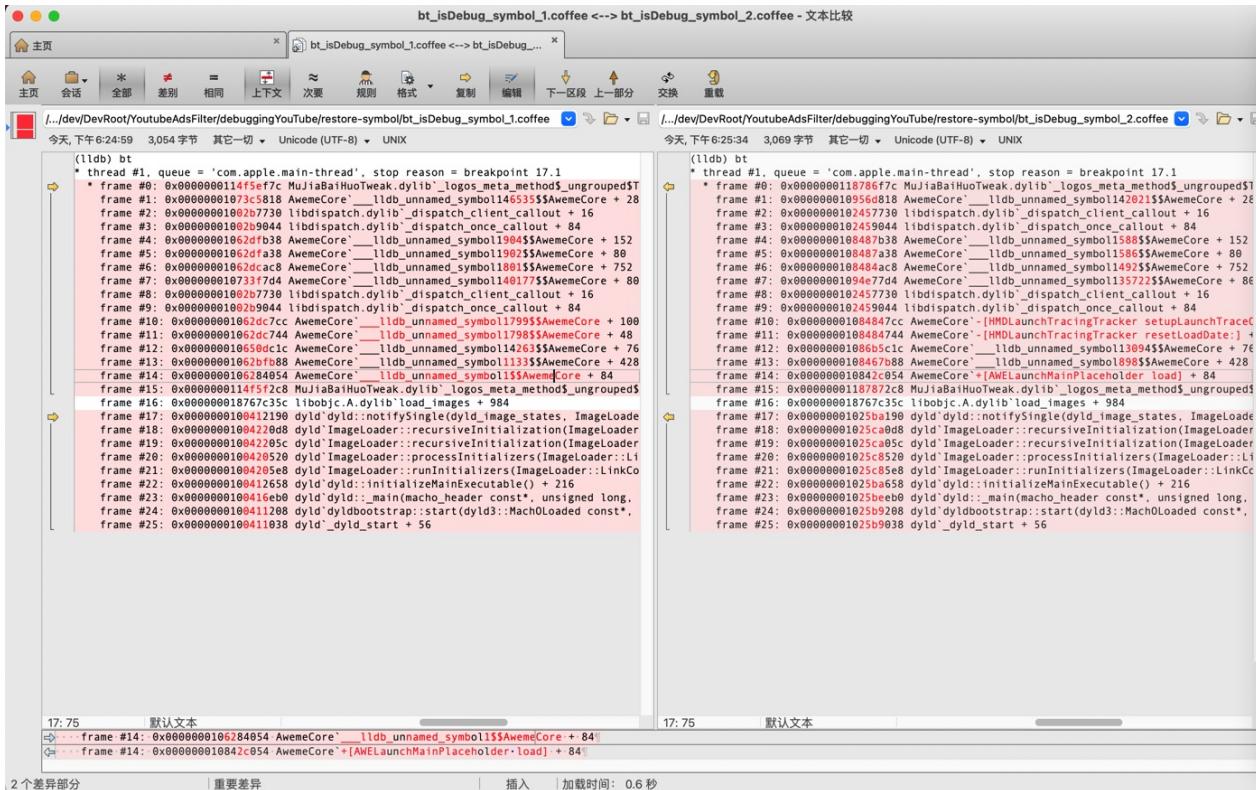
之前通过恢复符号表，把部分AwemeCore的函数，从lldb无名函数：

```
frame #10: 0x00000001062dc7cc AwemeCore`__lldb_unnamed_symbol1799$$AwemeCore + 100
frame #11: 0x00000001062dc744 AwemeCore`__lldb_unnamed_symbol1798$$AwemeCore + 48
frame #14: 0x0000000106284054 AwemeCore`__lldb_unnamed_symbol1$$AwemeCore + 84
```

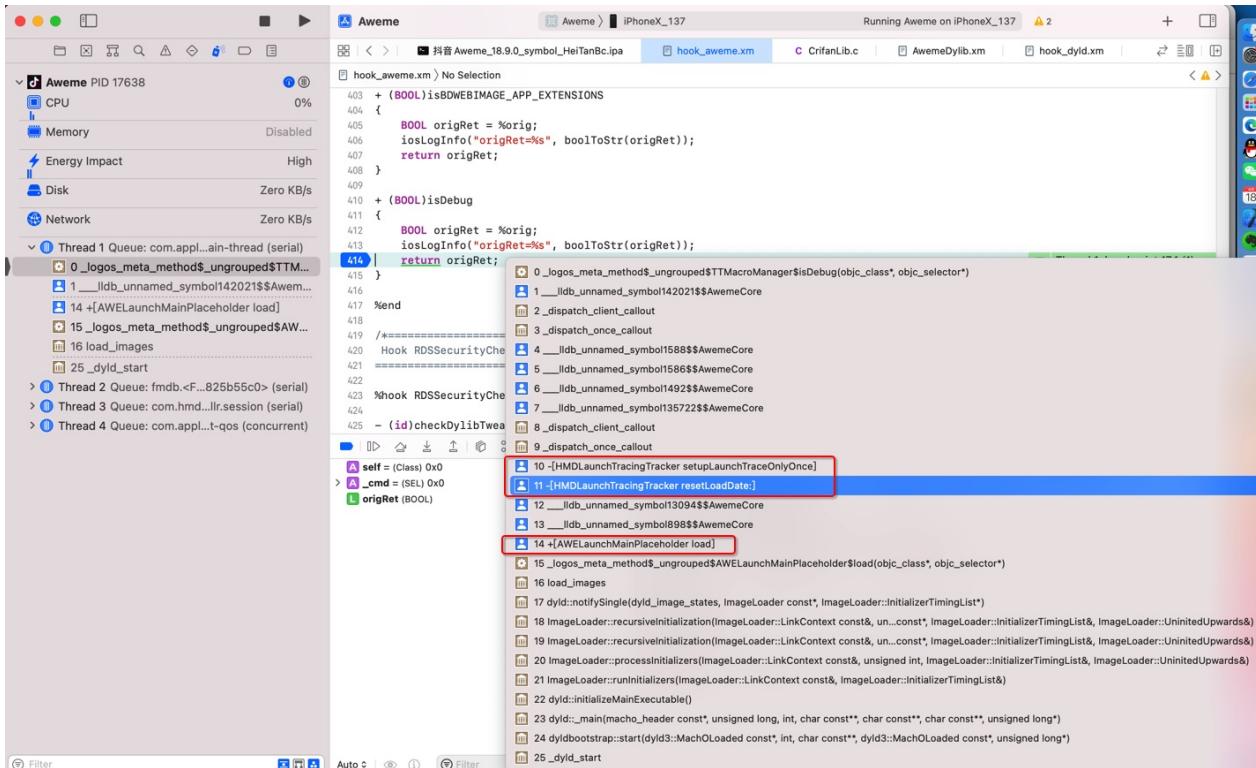
变成了有名字的函数：

```
frame #10: 0x00000001084847cc AwemeCore`-[HMDLaunchTracingTracker setupLaunchTraceOnlyOnce] + 100
frame #11: 0x0000000108484744 AwemeCore`-[HMDLaunchTracingTracker resetLoadDate:] + 48
frame #14: 0x000000010842c054 AwemeCore`+[AWELaunchMainPlaceholder load] + 84
```

对比效果：



恢复符号表后的函数调用堆栈：



由此：

- 之前：给函数断点，只能用lldb无名函数 `__lldb_unnamed_symbol1$$AwemeCore` 或计算出实际地址，通过地址去加断点

- 现在：给函数断点，就可以换用对应函数名 `+[AWELaunchMainPlaceholder load]` 了

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新:

2023-07-09 22:27:05

附录

下面列出相关参考资料。

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-06-29 21:26:17

参考资料

- 【已解决】给抖音恢复符号表便于Xcode调试加断点
- 【未解决】iOS逆向WhatsApp: SharedModules中的函数加不上如何加上断点且确保能触发
- 【已解决】用HeiTanBc的restore-symbol去给抖音AwemeCore恢复符号表
-
- 【已解决】iOS逆向Apple账号: Xcode中AKAppleIDAuthenticationService多个函数的断点无效加不上且hook代码没触发
- 【已解决】iOS逆向Apple账号: objc_alloc_init的hook代码加不上断点断点无效
- 【已解决】iOS逆向调试: 如何才能真正确保断点能触发到
- 【已解决】iOS逆向Apple账号: -[NSXPCCConnection remoteObjectProxyWithErrorHandler:] 的hook代码断点加不上
- 【未解决】iOS逆向Apple账号: -[AKAppleIDAuthenticationContextManager shouldContinueWithAuthenticationResults:error:forContextID:completion:]
- 【未解决】XCode和lldb如何根据函数地址加断点
- 【已解决】Xcode的lldb中如何根据地址给函数加断点
- 【已解决】lldb命令行中通过函数名或函数地址加断点
- 【整理】iOS逆向心得: 通过po描述中当前函数地址加断点
- 【已解决】iOS逆向akd: 找arm64的akd函数sub_1000A0460的断点地址
- 【整理】YouTube的Block代码段的断点和函数名
- 【规避解决】lldb中给ObjC函数加断点报错: WARNING Unable to resolve breakpoint to any actual locations
- 【已解决】iOS逆向: Xcode中给lldb无名函数加断点指定属于哪个库文件
- 【已解决】XCode的断点条件判断中如何获取iOS的ObjC函数的参数值
- 【已解决】XCode的lldb断点中查看NSFileManager的fileExistsAtPath的参数值
- 【已解决】研究抖音崩溃MetaSec appID原因: 给UIApplicationMain加符号断点
- 【已解决】XCode的lldb中加符号断点报错: strcmp has unknown return type cast the call to its declared return type
- 【已解决】XCode中如何给iOS的ObjC的汇编代码的断点加上条件判断
- 【基本解决】lldb命令行中如何给ObjC的断点加上条件判断
- 【部分解决】Xcode中给ObjC函数加上条件断点
- 【已解决】XCode中如何给符号断点加上判断条件
- 【未解决】Xcode中给iOS的ObjC的ARM汇编代码加上带条件判断执行的断点
- 【已解决】iOS逆向: Xcode中给汇编代码断点加条件判断
- 【已解决】Xcode中给objc_alloc_init断点加上条件判断传入的类是否是AADeviceInfo
- 【已解决】iOS逆向: objc_alloc_init中给汇编代码加断点条件判断是否是类AADeviceInfo
- 【已解决】iOS的ObjC中如何获取Class类名
- 【已解决】iOS逆向: 给objc_alloc_init函数的断点的条件判断是否是类AADeviceInfo尝试更多其他写法
- 【已解决】iOS逆向Apple账号: objc_alloc_init的汇编代码断点没有触发到的原因
- 【已解决】iOS逆向: 给objc_alloc_init的+44行汇编代码的断点加条件判断的是是否是类AADeviceInfo的更多种写法
- 【已解决】lldb调试时给已有断点更改condition条件判断
- 【已解决】给XCode调试抖音Aweme恢复符号表

- 【已解决】用HeiTanBc的restore-symbol去给抖音AwemeCore恢复符号表
- 【整理】抖音AwemeCore恢复符号表的效果举例
- 【已解决】给YouTube恢复符号表方便Xcode调试
- 【已解决】XCode调试警告：was compiled with optimization stepping may behave oddly variables may not be available
- 【已解决】Xcode编译dylib越狱插件运行提示：was compiled with optimization stepping may behave oddly
- 【记录】Xcode中加特殊的全局异常的断点
- 【未解决】Xcode中无法给下一行将要运行的汇编指令加断点
- 【已解决】iOS逆向时+[AADeviceInfo(Deprecated) udid]类名中带Deprecated如何加断点和调试
- 【未解决】iOS逆向Apple账号：debugserver+lldb调试+[AADeviceInfo udid]函数逻辑
- 【整理】Xcode调试心得：判断Xcode中添加断点是否生效
- 【已解决】XCode中加的符号断点无效
- 【已解决】Xcode中加断点无效：XCode won't pause at this breakpoint because it has not been resolved
- 【已解决】XCode中一键关闭所有断点和一键恢复所有断点
- 【已解决】iOS逆向Apple账号：objc_alloc_init的hook代码加不上断点断点无效
- 【已解决】Xcode中xm源码中无法看到和添加断点
- 【已解决】让XCode的iOSOpenDev中Logos的xm文件支持语法高亮
- 【已解决】iOS逆向：Xcode写hook插件看能否调试触发断点
- 【已解决】如何才能让Xcode中hook代码断点被触发实时调试插件hook代码
- 【已解决】XCode调试抖音ipa：给用Logos去hook的函数_dyld_get_image_name加符号断点
- 【已解决】iOS的ObjC函数如何正确转换成签名加到Xcode断点
- 【已解决】iOS逆向：给amsaccounts重签名以使得Xcode可以调试amsaccounts
- 【未解决】iOS逆向Apple账号：debugserver+lldb调试+[AADeviceInfo udid]函数逻辑
- 【未解决】iOS逆向Apple账号：换tweak插件去调试+[AADeviceInfo udid]
- 【已解决】Xcode中调试时给+[AADeviceInfo udid]的hook代码加断点失效加不上
- 【已解决】iOS逆向时lldb中添加了ObjC函数的断点却没触发到
- 【已解决】iOS逆向Apple账号：+[AADeviceInfo udid]断点没生效换调试对象Preferences
- 【已解决】iOS逆向Apple账号：给objc_alloc_init汇编代码中的AADeviceInfo init加断点调试是否触发
- 【未解决】iOS逆向Apple账号：用debugserver+lldb去调试-[AALoginAccountRequest urlRequest]看断点是否触发
- 【未解决】iOS逆向Apple账号：Xcode调试找不到和没有触发NSURLRequest相关断点
- 【未解决】Xcode中如何给iOS的Swift函数加断点：AppleStoreCore的User的initialize
- 【未解决】iOS逆向AppleStore证书无效：多个SSL函数的断点都没触发到
- 【已解决】研究YouTube逻辑：触发不到断点MLOneisieRequestFactory的 onesieRequestForPlayerRequest:dataLoader:context:cryptor:requestNumber:error:
- 【未解决】Xcode调试抖音：触发断点后继续运行却卡死
- 【已解决】XCode+MonkeyDev调试抖音不崩溃却卡死禁用断点始终停在Foundation的NSString的 stringByAppendingString
- 【未解决】Xcode中已禁用所有断点但仍会随机暂停执行在某处
- 【未解决】Xcode调试抖音会随机停在某处：strEndsWith的SIGTRAP
- 【未解决】Xcode调试抖音会随机停在某处：dladdr的SIGTRAP
- 【未解决】通过XCode给stringByAppendingString加断点调试寻找抖音崩溃原因
- 【基本解决】XCode断点调试_dyld_register_func_for_add_image后续调用导致抖音崩溃
- 【整理】iOS逆向心得：给函数加了hook同时加断点会导致EXC_BREAKPOINT的崩溃

- 【已解决】Xcode调试抖音的_RxAnnotationInlineLoader的load崩溃：Thread 1 EXC_BREAKPOINT code 1 subcode 0xe7ffdefe
- 【已解决】Xcode调试hook代码时dladdr崩溃报错：Thread 1 EXC_BREAKPOINT code 1 subcode 0xe7ffdefe
- 【已解决】Xcode中加断点lldb命令行报错：warning failed to set breakpoint site at for breakpoint error sending the breakpoint request
- 【无需解决】iOS逆向调试：Xcode触发断点后长时间无操作好像调试会断开
- 【已解决】Xcode调试抖音：卡死了运行不到汇编代码和函数的断点
- 【未解决】XCode如何给MonkeyDev调试ipa程序加上断点
- 【未解决】通过XCode给stringByAppendingString加断点调试寻找抖音崩溃原因
- 【未解决】XCode和lldb中如何通过类名打断点：AWECloudJailBreakUtility
- 【未解决】XCode和lldb中给抖音入口C语言函数_awemeMain加断点
- 【已解决】研究抖音崩溃MetaSec appId原因：给UIApplicationMain加符号断点
- 【未解决】研究AWECloudJailBreakUtility详情：用XCode给各个函数加上符号断点
- 【已解决】iOS逆向：给__lldb_unnamed_symbol317和MGCopyAnswer添加条件判断的断点
- 【未解决】lldb命令行中给objc_alloc_init的断点条件判断是否是AADeviceInfo
- 【未解决】lldb中条件判断的断点出现警告：warning hit breakpoint while running function skipping commands and conditions to prevent recursion
- 【整理】Xcode的lldb调试心得：可能被多次调用的函数不能轻易给后续加断点再继续运行
- 【已解决】Xcode中lldb中b list不是breakpoint list
- 【整理】lldb的语法和用法
- 【已解决】iOS逆向：Xcode调试Preferences时界面卡死无法操作
- 【整理】iOS逆向：ObjC底层函数objc_alloc_init
- 【整理】iOS逆向：ObjC类的常用方法
-
- [ios - lldb breakpoint on all methods in class objective c - Stack Overflow](#)
- [不用反汇编！LLDB对Objective-C函数下断点的黑科技 - 干货分享 - 睿论坛 \(iosre.com\)](#)
- [断点 - 维基百科，自由的百科全书](#)
- [Breakpoint - Wikipedia](#)
- [Tutorial — The LLDB Debugger \(llvm.org\)](#)
- [GDB to LLDB command map — The LLDB Debugger \(llvm.org\)](#)
- [stringByAppendingString: | Apple Developer Documentation](#)
- [Why aren't Xcode breakpoints functioning? - Stack Overflow](#)
- [Setting breakpoints to pause your running app | Apple Developer Documentation](#)
-

crifan.org, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：
2023-10-26 09:44:13