目录

Xcode开发：调试心得

- 最新版本：`v0.6`
- 更新时间：`20221031`

简介

此处介绍Xcode开发中调试方面的心得。先是Xcode调试的概览；再介绍调试心得，包括调试区的介绍、调试快捷键、断点、调试中、Report Navigator、Build Phases；断点包括条件断点、给汇编加断点；调试中包括日志输出、函数调用堆栈、断点触发详情、代码停下来的原因、箭头指向的指令、切换进程视图。

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/xcode_dev_debug_summary: Xcode开发：调试心得

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

- Xcode开发：调试心得 book.crifan.org
- Xcode开发：调试心得 crifan.github.io

离线下载阅读

- Xcode开发：调试心得 PDF
- Xcode开发：调试心得 ePub
- Xcode开发：调试心得 Mobi

版权和用途说明

鸣谢

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

更多其他电子书

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

Xcode调试概览

在涉及到Apple苹果相关的开发，不论是

- 不同端
  - Mac
  - iOS
- 不同开发语言
  - ObjC
  - Swift
- 不同方向
  - 正向：安全防护
  - 逆向：破解

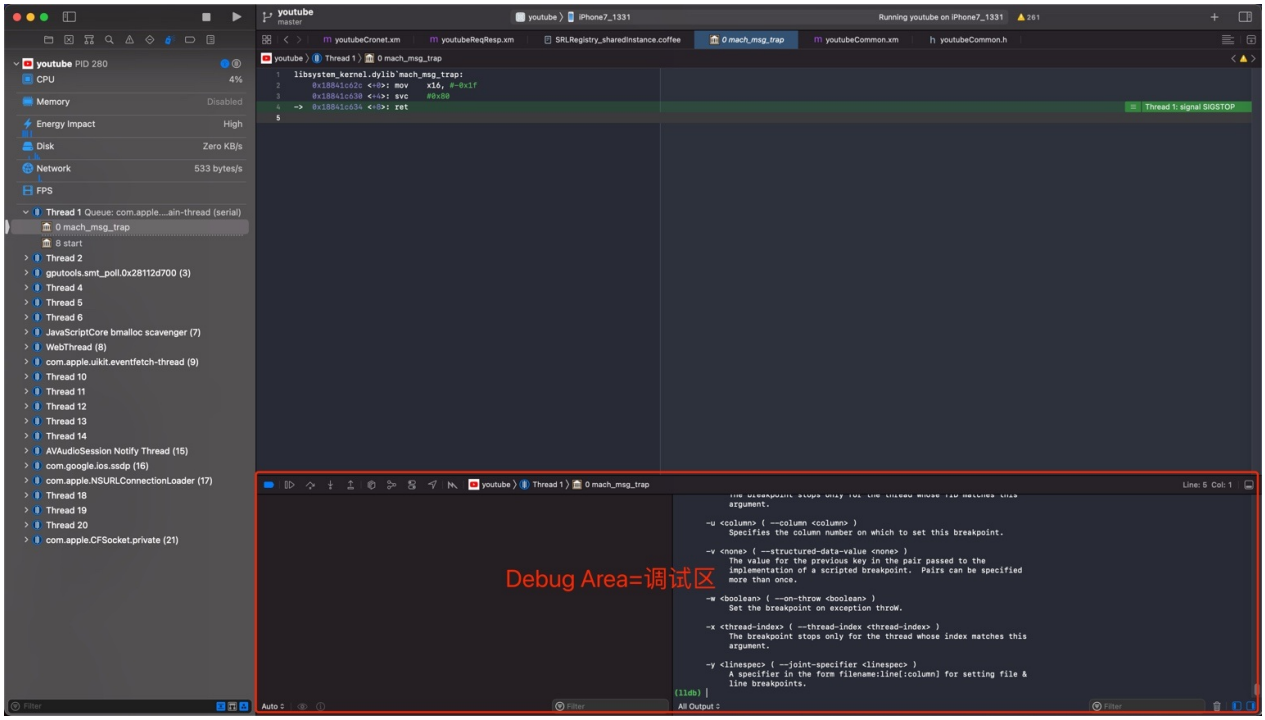
往往都要用到Apple的IDE：`Xcode`

而此处整理Xcode开发期间中，涉及到调试方面的心得。

Xcode调试心得

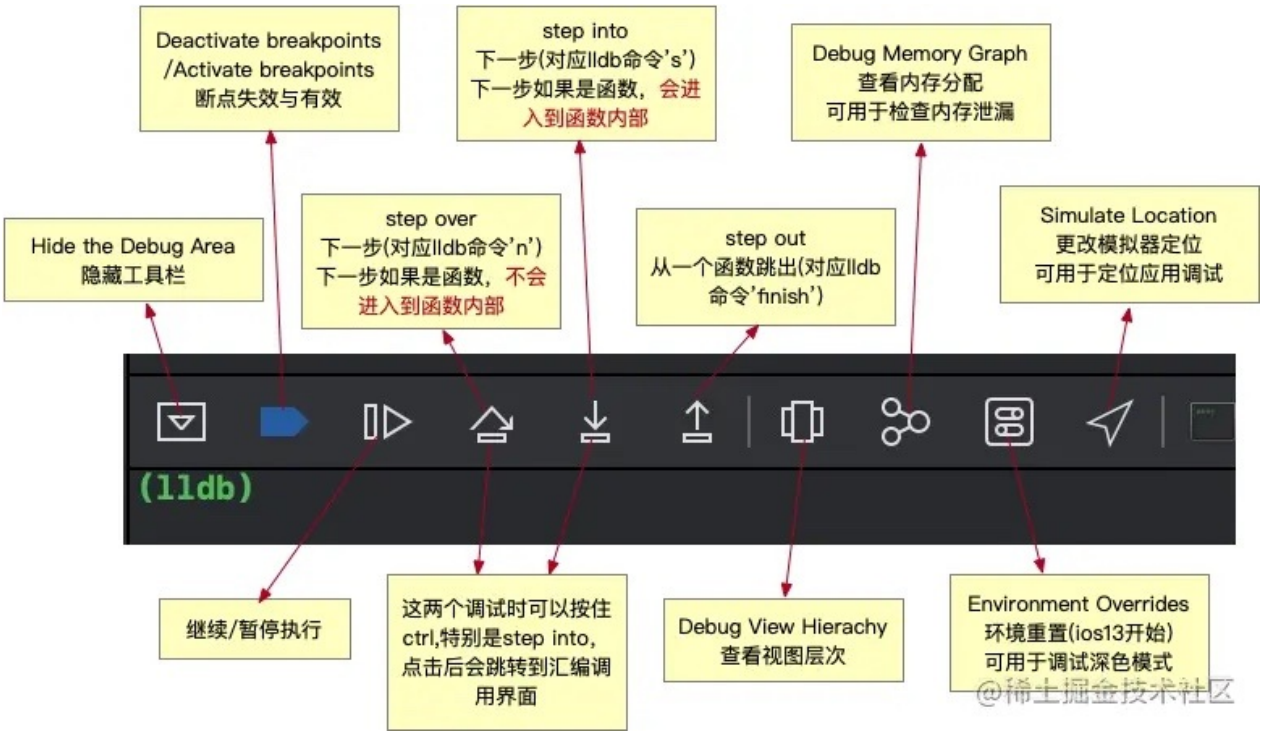
调试区

对于Xcode调试期间的界面，一般长这样：



其中右下角的区域是Xcode的 `debug区` = `调试区`

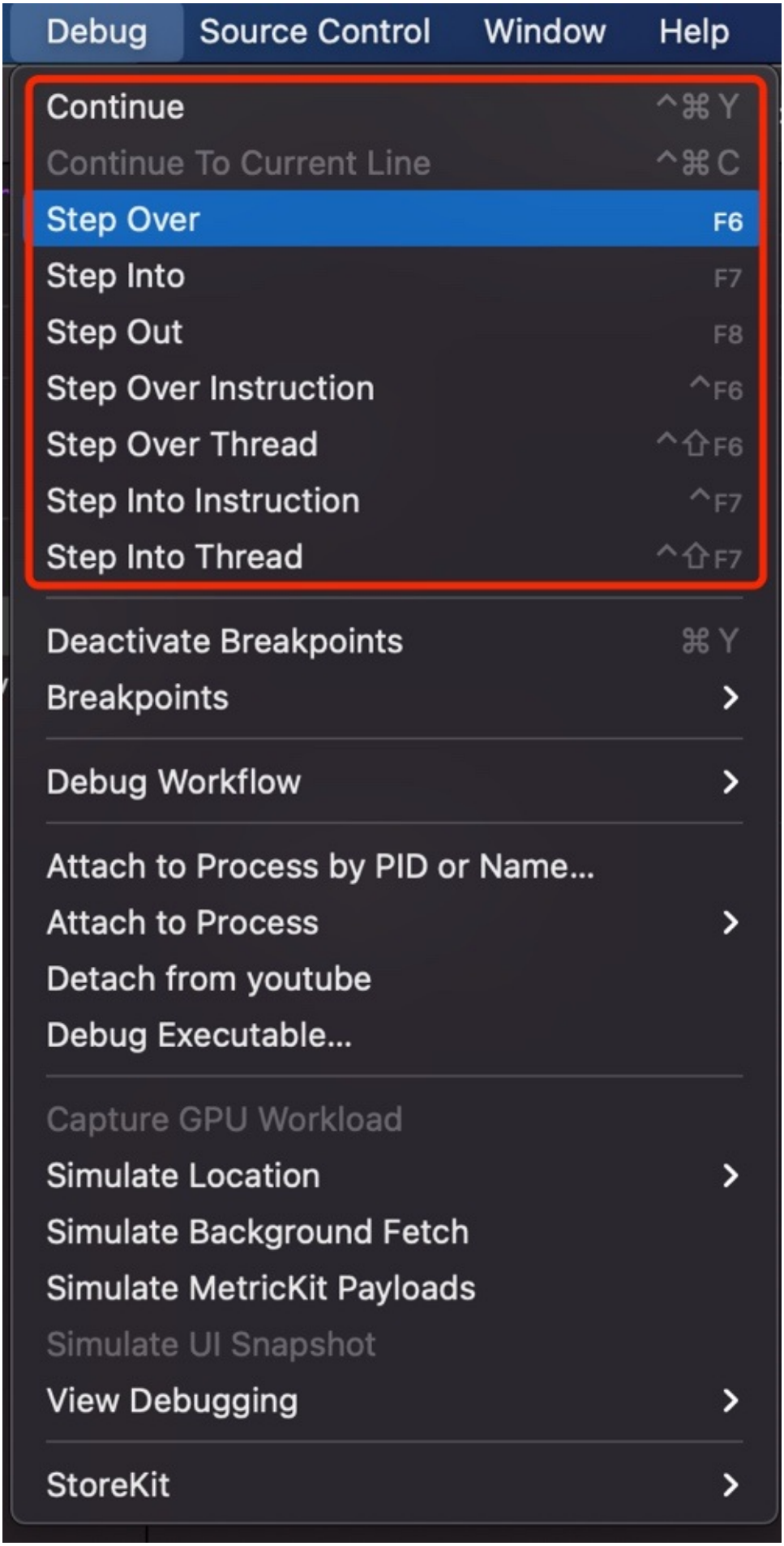
此处对于 `Xcode调试区` 的 按钮 = 功能 进行概述：



快捷键

Xcode调试相关快捷键

- Xcode调试时的相关快捷键
  - 文字
    - `Step Over`：`F6`
      - `Step Over Instruction`：`Ctrl+F6`
      - `Step Over Thread`：`Ctrl+Shift+F6`
    - `Step Into`：`F7`
      - `Step Into Instruction`：`Ctrl+F7`
      - `Step Into Thread`：`Ctrl+Shift+F7`
    - `Step Out`：`F8`
  - 图

- 
Debug
Source Control
Window
Help
Continue ^⌘Y
Continue To Current Line ^⌘C
Step Over F6
Step Into F7
Step Out F8
Step Over Instruction ^F6
Step Over Thread ^⇧F6
Step Into Instruction ^F7
Step Into Thread ^⇧F7
Deactivate Breakpoints ⌘Y
Breakpoints
Debug Workflow
Attach to Process by PID or Name...
Attach to Process
Detach from youtube
Debug Executable...
Capture GPU Workload
Simulate Location
Simulate Background Fetch
Simulate MetricKit Payloads
Simulate UI Snapshot
View Debugging
StoreKit

F7无效

Mac中此处Xcode调试的单步进入的快捷键：F7 无效

发现是被其他占用了。

经过查找发现是旧版有道词典占用的。

解决办法：

- 彻底卸载旧版有道词典
  - 因为即使没开启 取词和划词 也会占用F7快捷键
- 卸载旧版，安装新版有道词典
  - 没开启取词划词，就不会占用（F7等）快捷键

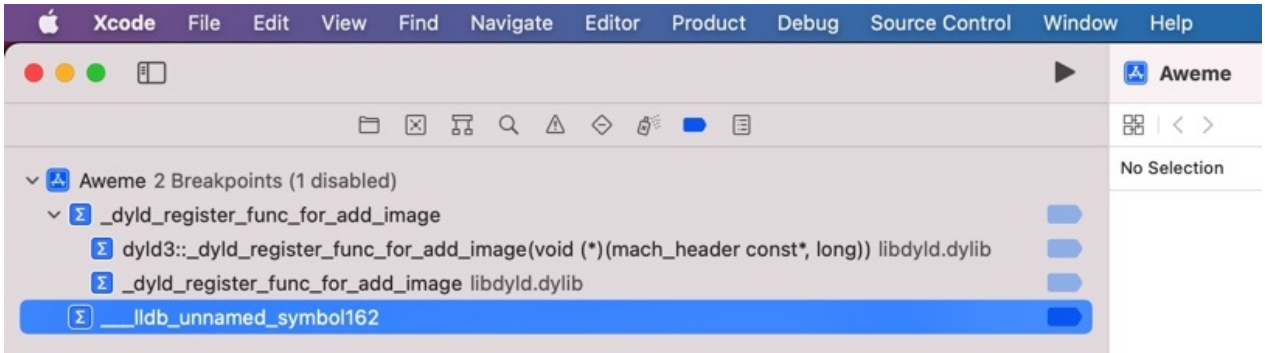
断点

调试加断点时函数名要准确，否则加不上

想要给函数

`___lldb_unnamed_symbol162`

加上断点：



后来发现，前面的写法是错误的

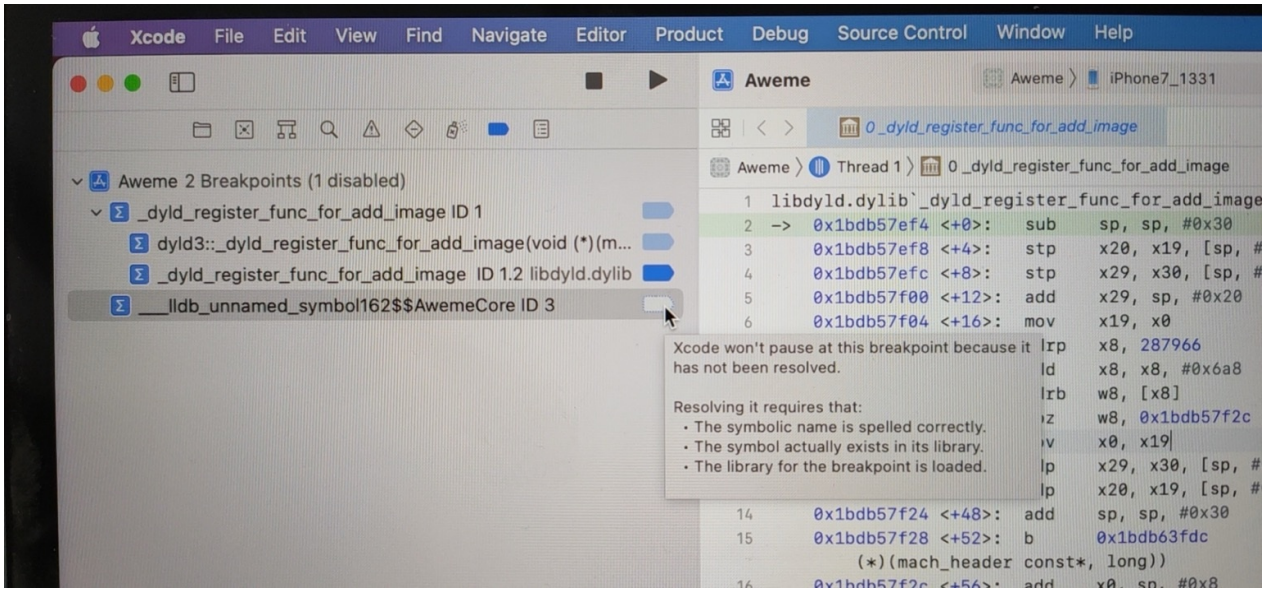
应该改为：

`___lldb_unnamed_symbol162$$AwemeCore`

不过然后又发现，好像没有生效

且对于为何没有生效，还有额外的提示：

```
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.
```

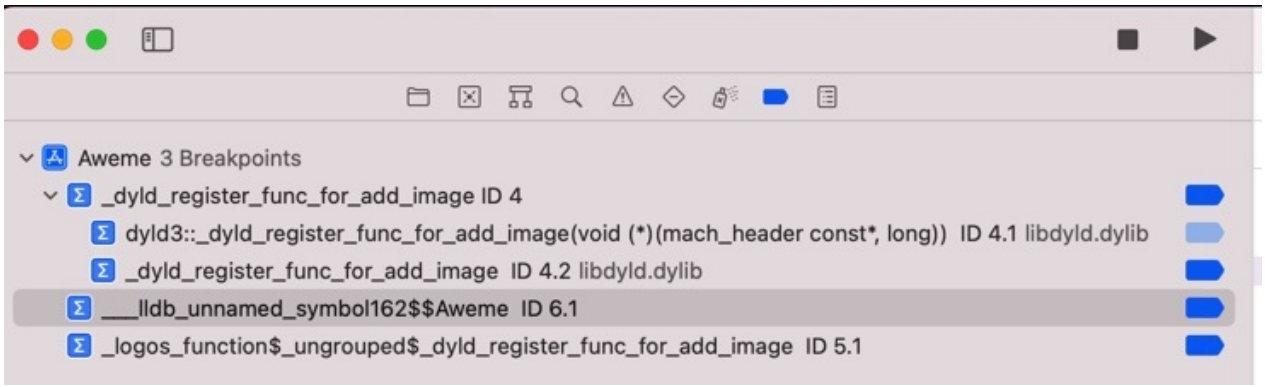


再然后发现是：

是函数所属的二进制弄错了

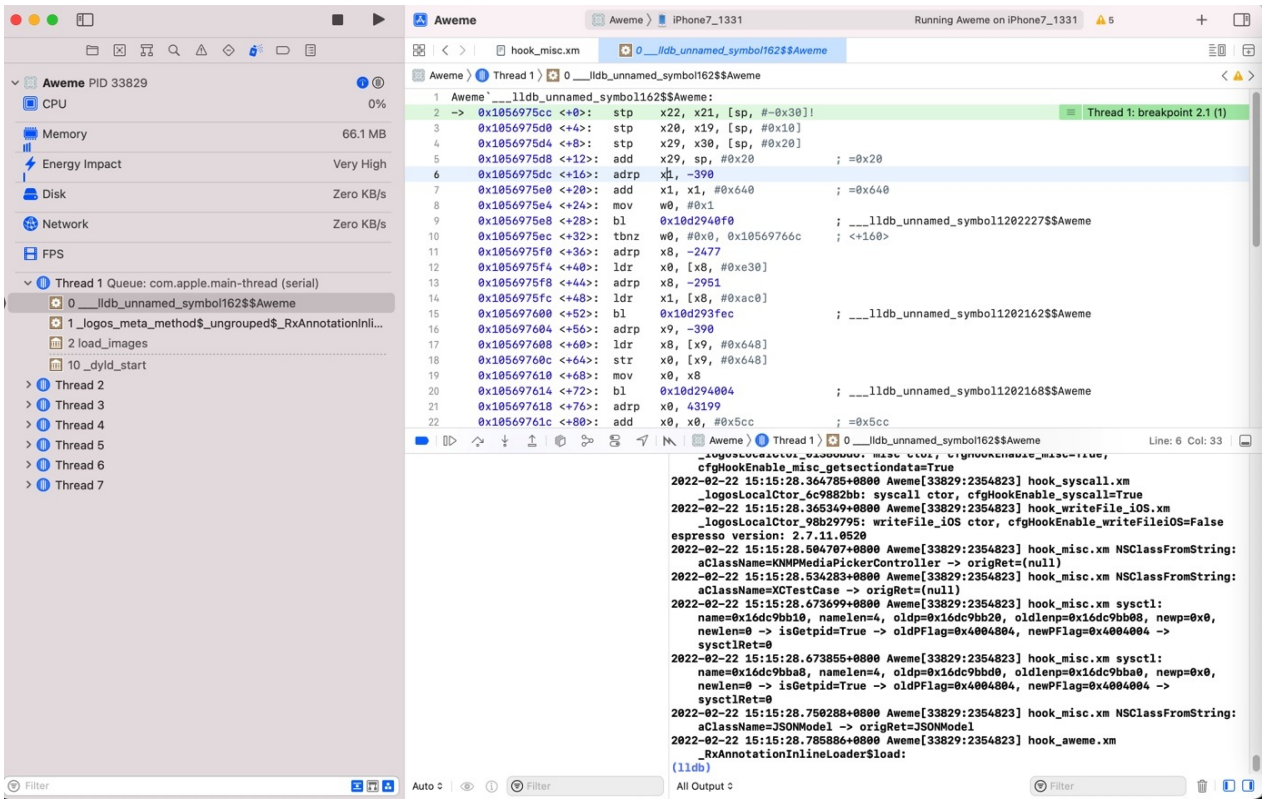
应该改为：

`___lldb_unnamed_symbol162$$Aweme`



就正确了。

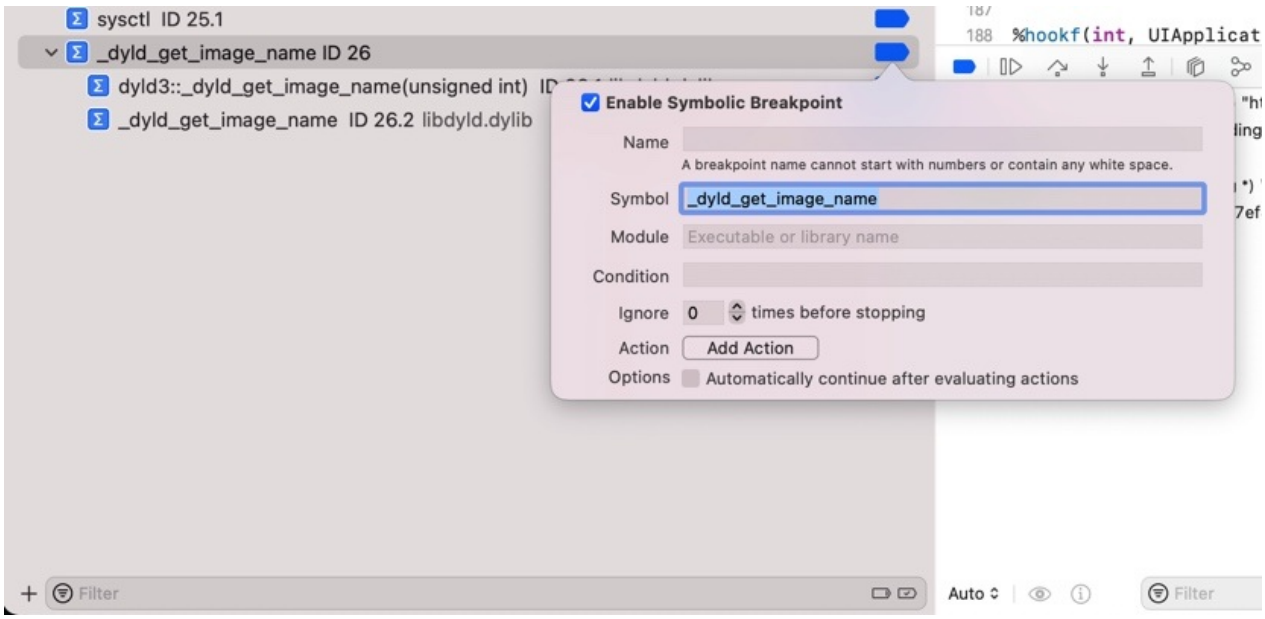
后续断点才能正常生效：



添加符号断点时，会自动搜索到匹配的函数

给Xcode添加符号断点：

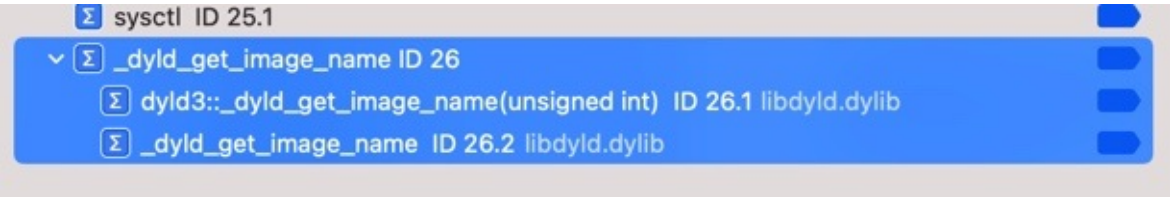
`_dyld_get_image_name`



回车确认后发现：

自动会出现2个（子）断点：

- `dyld3::_dyld_get_image_name(unsigned int) ID 26.1 libdyld.dylib`
- `_dyld_get_image_name ID 26.2 libdyld.dylib`



看起来好像是：

会根据当前符号，自动去寻找匹配到的函数

效果不错。

自动补全

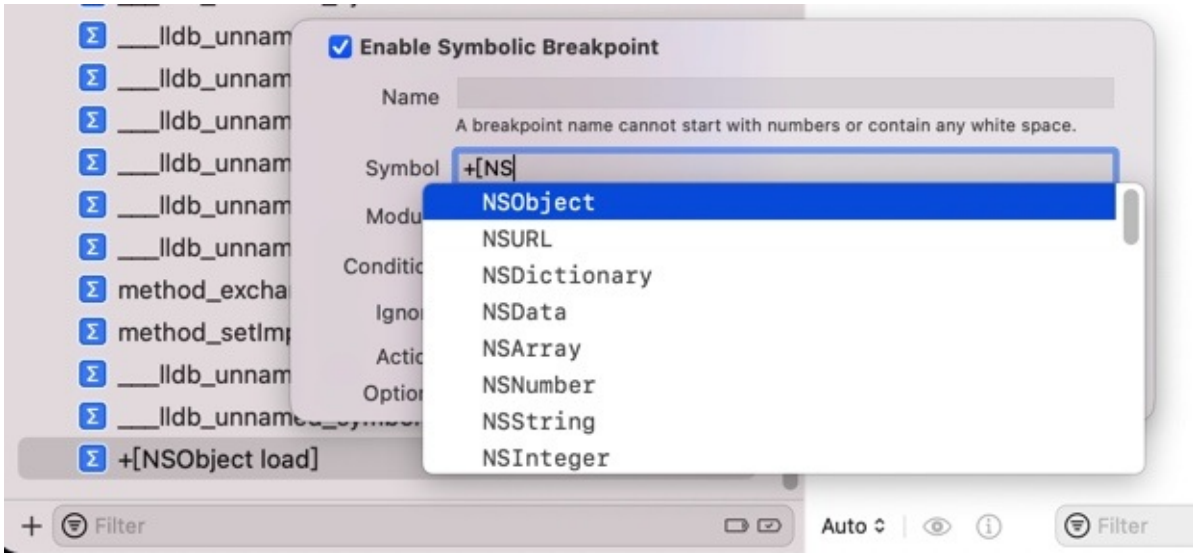
Xcode中添加断点的输入框中，竟然也支持自动补全

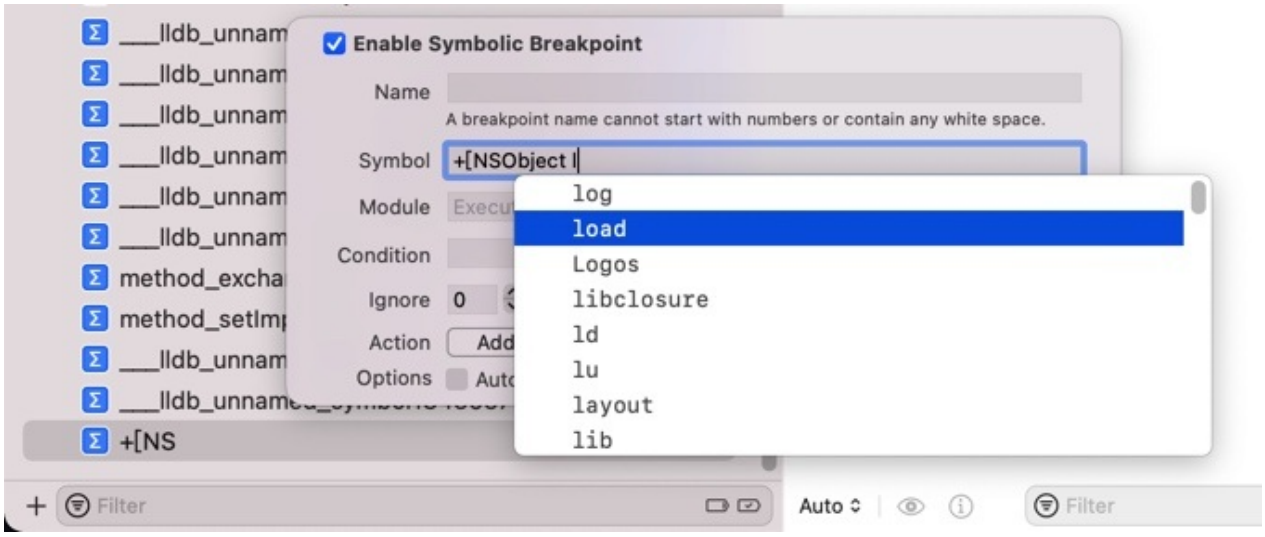
折腾：

【未解决】研究抖音越狱检测逻辑：NSObject的load

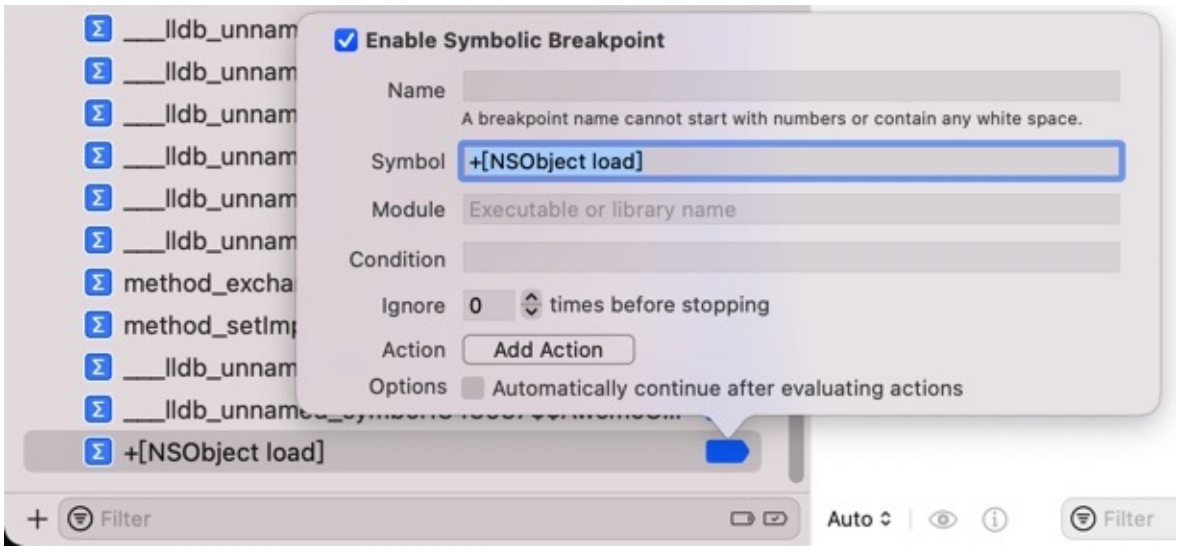
期间，发现个心得：

Xcode的符号断点的输入框中，也支持 动态匹配 自动补全：





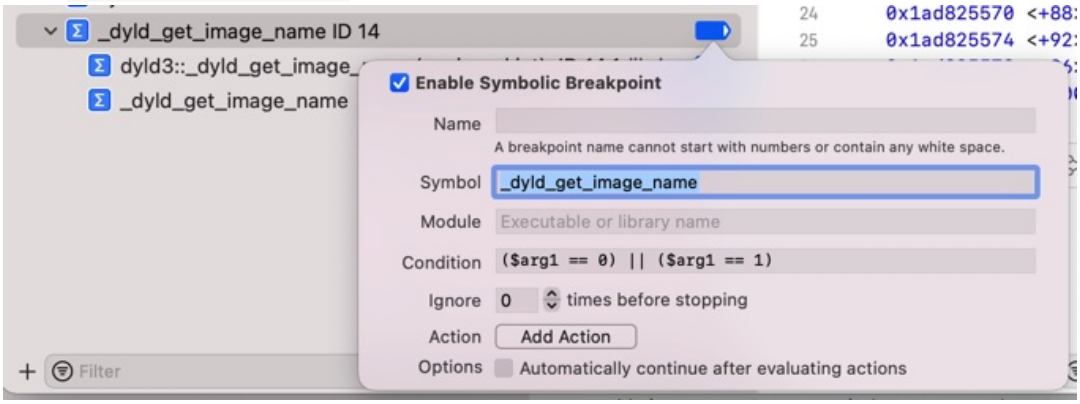
输入完后的效果：



条件判断

TODO：

- 【已解决】XCode中如何给符号断点加上判断条件
- 【未解决】通过XCode给stringByAppendingString加断点调试寻找抖音崩溃原因
- 【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

- 举例
  - Symbol: `_dyld_get_image_name`
    - Condition: `($arg1 == 0) || ($arg1 == 1)`
    - 

给汇编加断点

TODO：

- 【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

给汇编加断点

调试二进制 = ARM汇编代码，可以给汇编加上断点：

其实本身很简答：点击每行的最左边，即可加断点

只不过有个小缺点，有时候是个大缺点：由于断点开启后，该行最左边会加上白色背景，但是本身界面就是白色背景，导致：你根本不知道自己加了断点

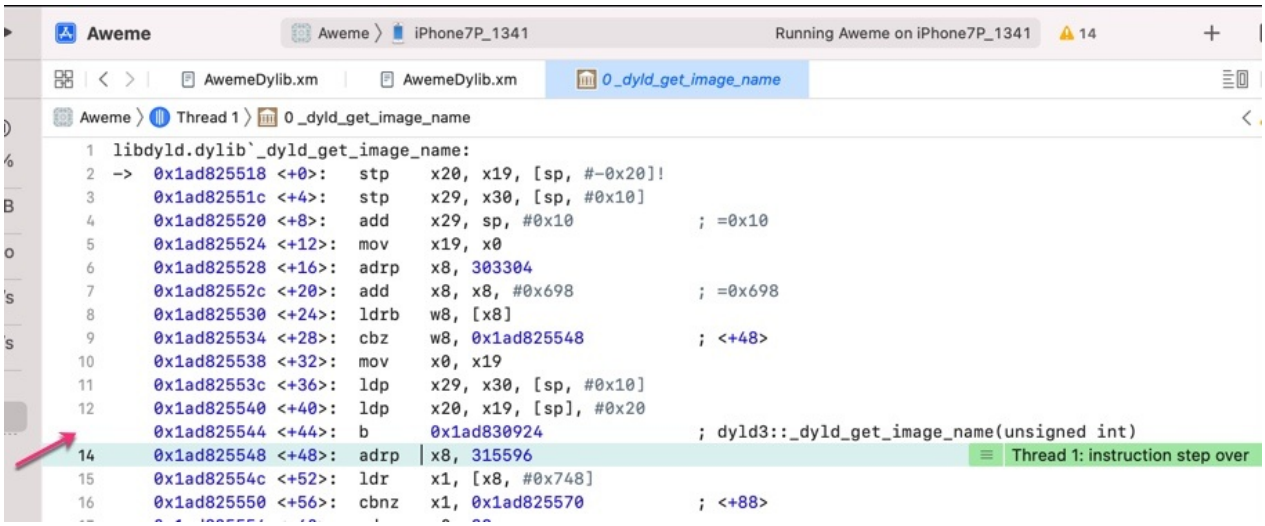
只有仔细看，才能看出来：此时的行号没了

然后此时右键，能看到断点相关菜单功能，也才能，确认的确加了断点

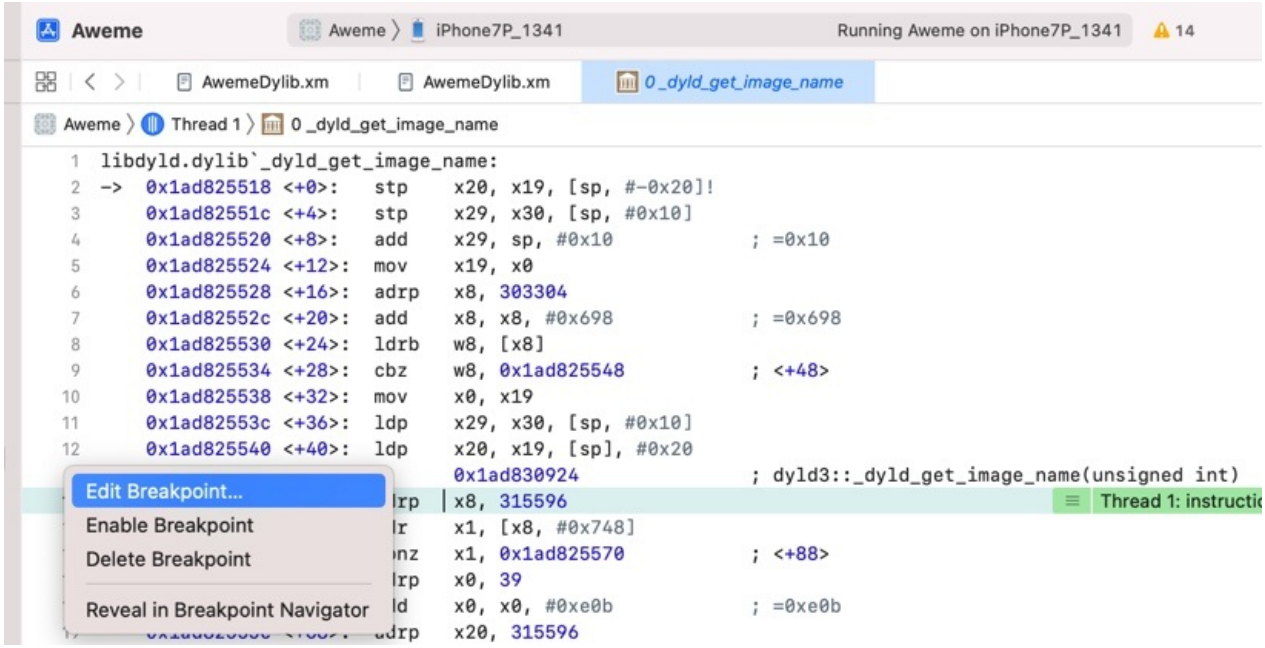
举例：

点击行号 `13` ，可以加上 汇编代码的断点：

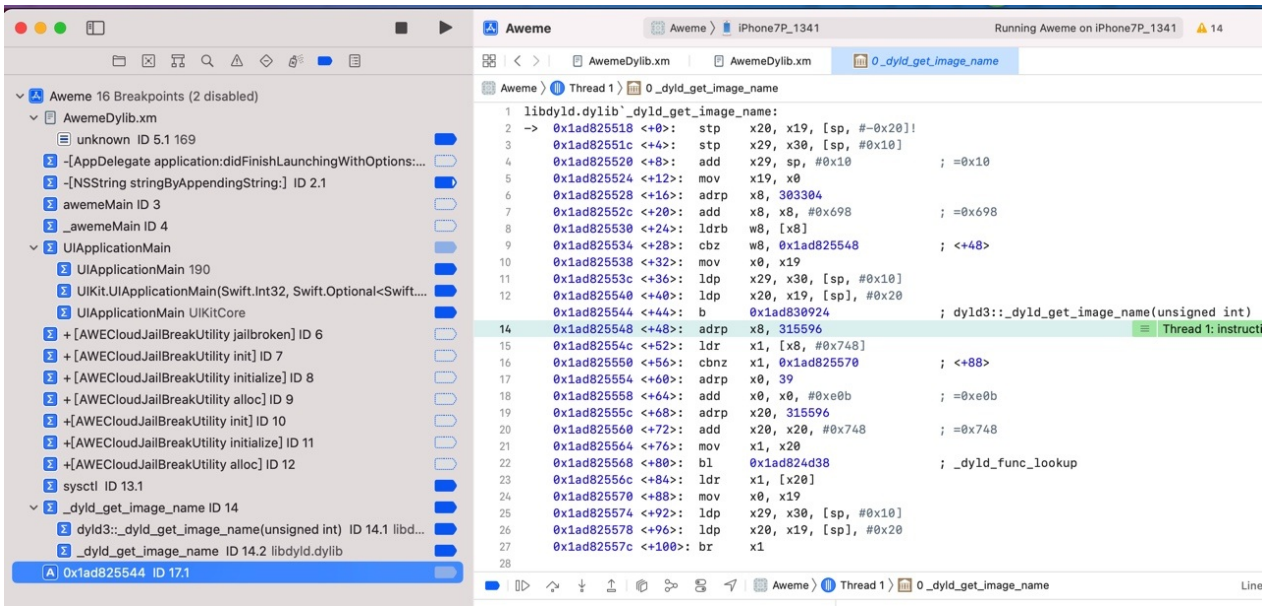
此时 `13` 的行号，就看不见了：



但是可以右键发现是加了断点的：



另外，切换到断点类别中的断点列表，也能看到新的 汇编代码 的断点：



此处断点名就是：地址：

```
0x1ad825544
```

调试中

日志输出

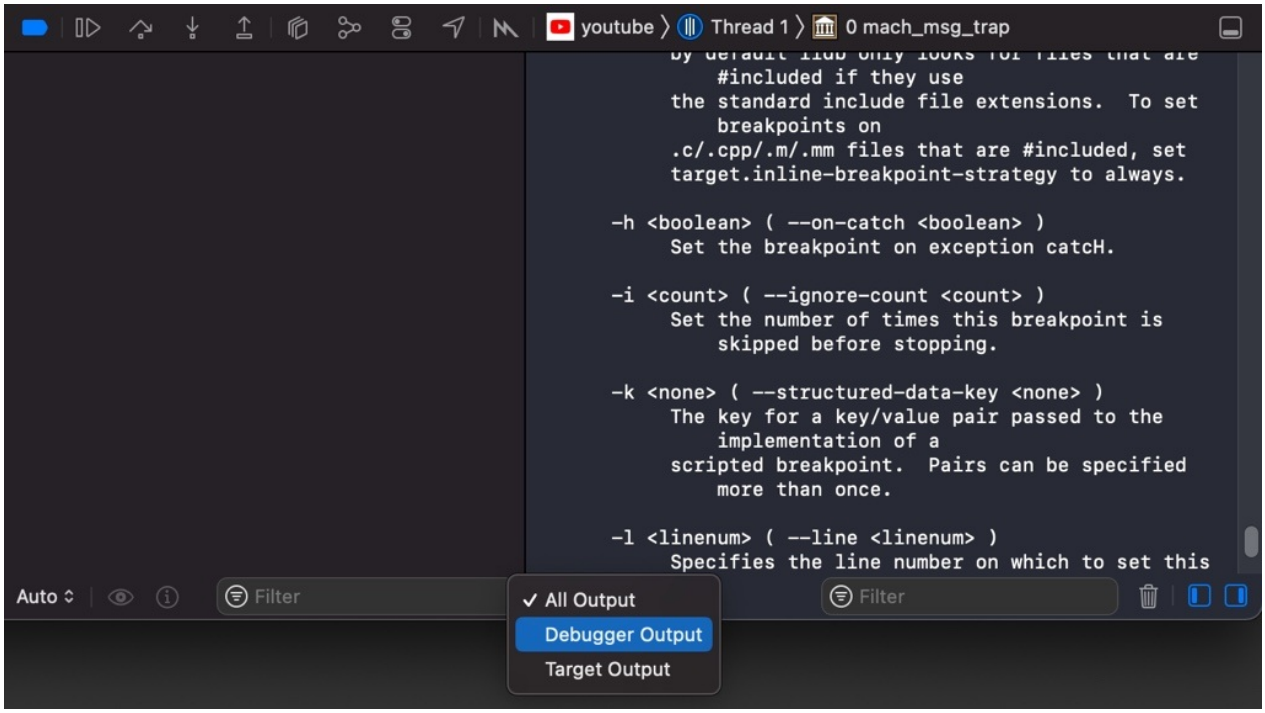
调试区中log日志输出为空

- 现象

偶尔遇到过，调试区中，输出log的地方是空白 = 所有日志都没有输出 = 看不到日志

- 原因

后来发现是，之前某次不小心，把默认的 `All Output` 改为了其他的选项了，比如 `Debugger Output` 或 `Target Output`



- 解决办法

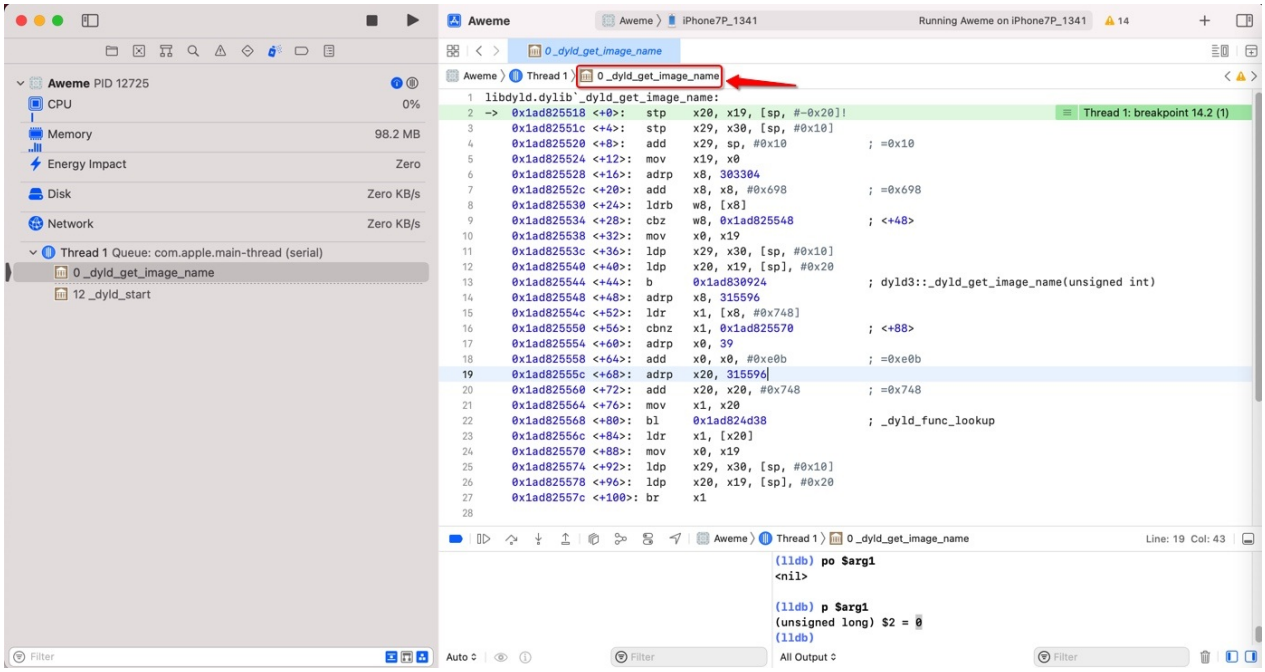
改回默认的 `All Output` ，即可正常看到全部log了。

函数调用堆栈

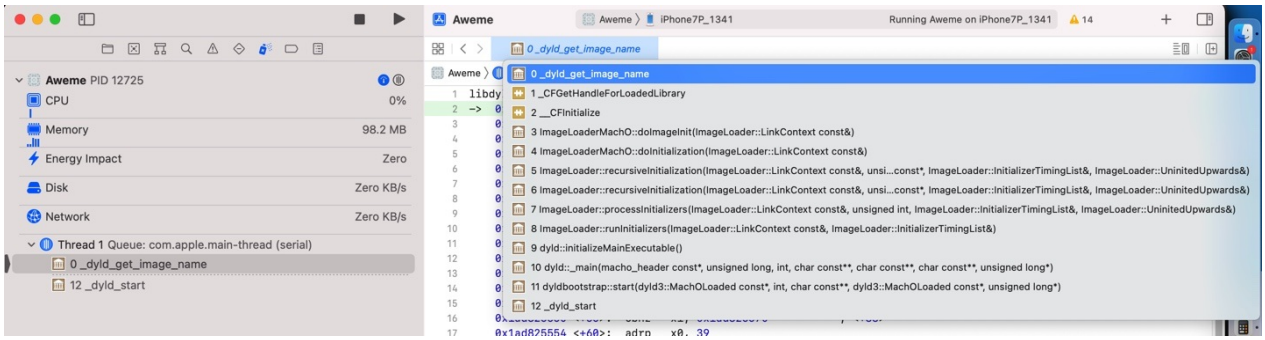
点击顶部函数名查看函数调用顺序=函数调用堆栈

无意间发现：

点击 调试界面的顶部的 App名字同行的位置的 最后的 `函数名` ：

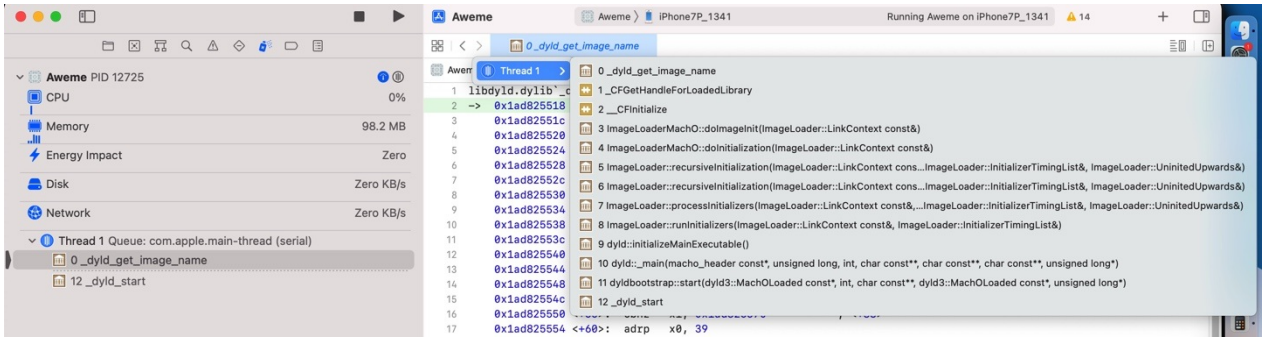


可以列出：函数调用顺序 = 函数调用堆栈：



-》方便调试。

类似的，点击 `Thread` 也可以列出函数：



-》看来是：

方便直接切换到不同的代码执行的地方：

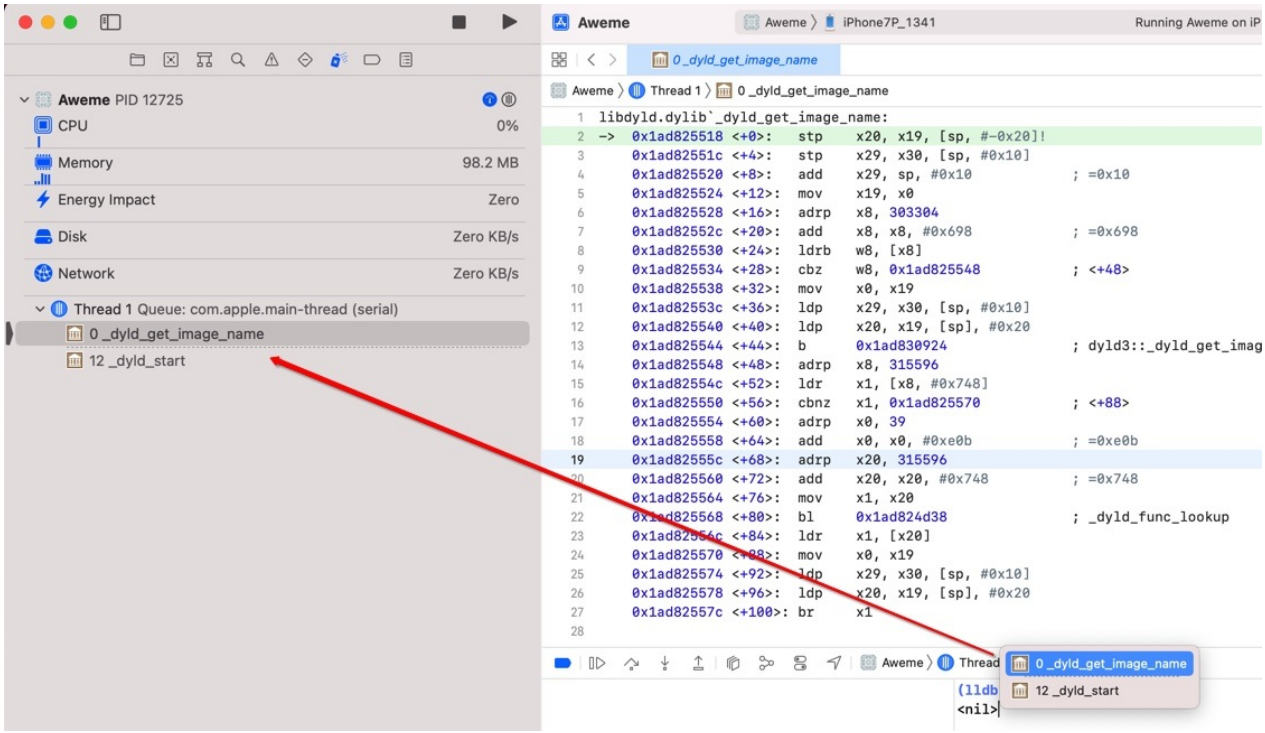
- App
  - Thread
    - Function

另外：

调试界面的底部：

直接点击，会显示出：

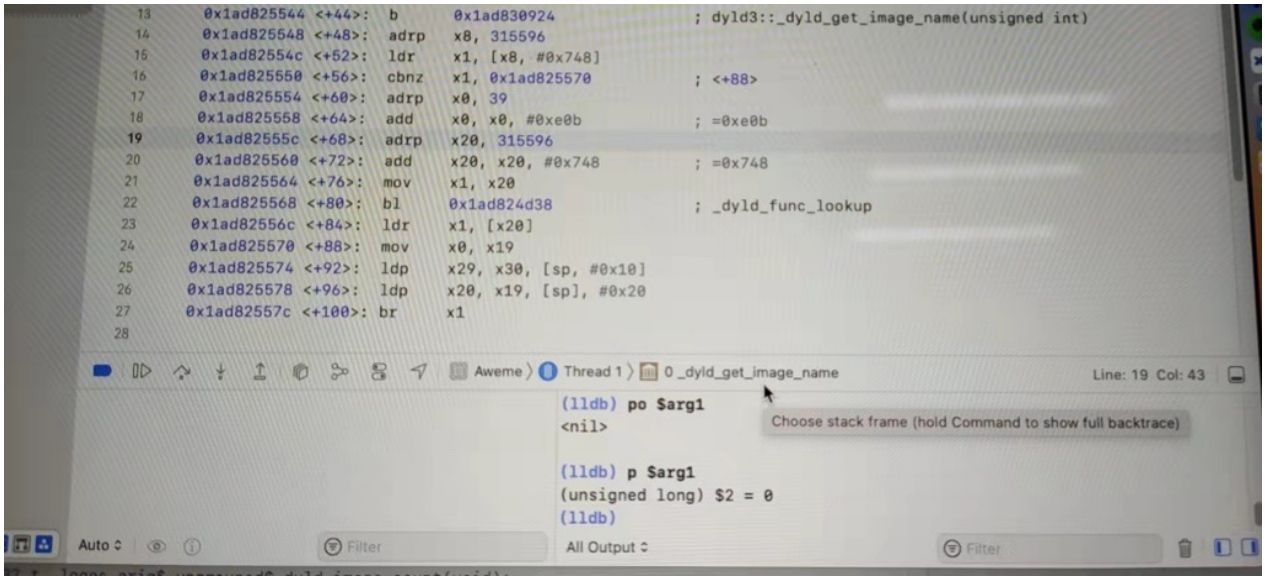
和调试左上角列出的一样的，只有2个函数：



后来发现：

鼠标移动上去，会提示：

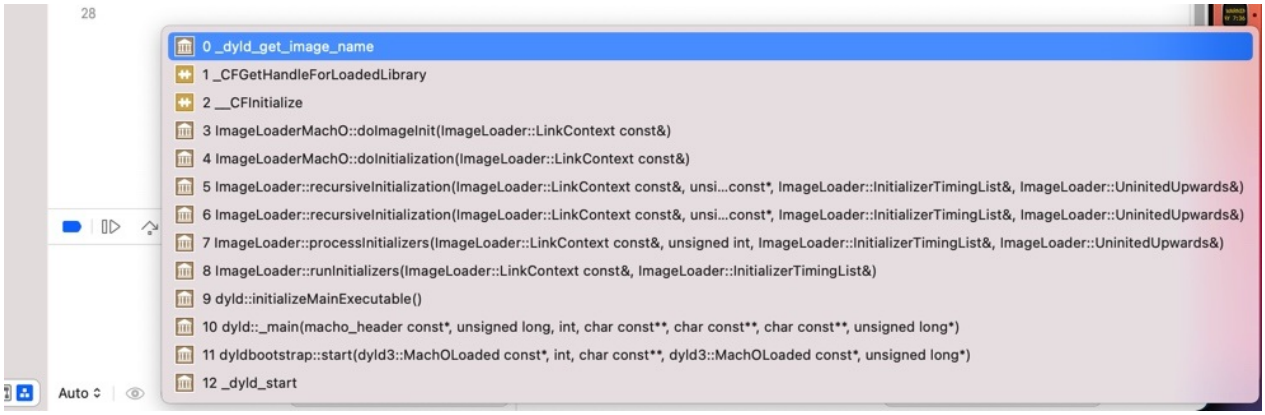
`Choose stack frame (hold Command to show full backtrace)`



然后试试：

`Command + 点击`

可以出现和前面一样的，完整的，函数调用堆栈：



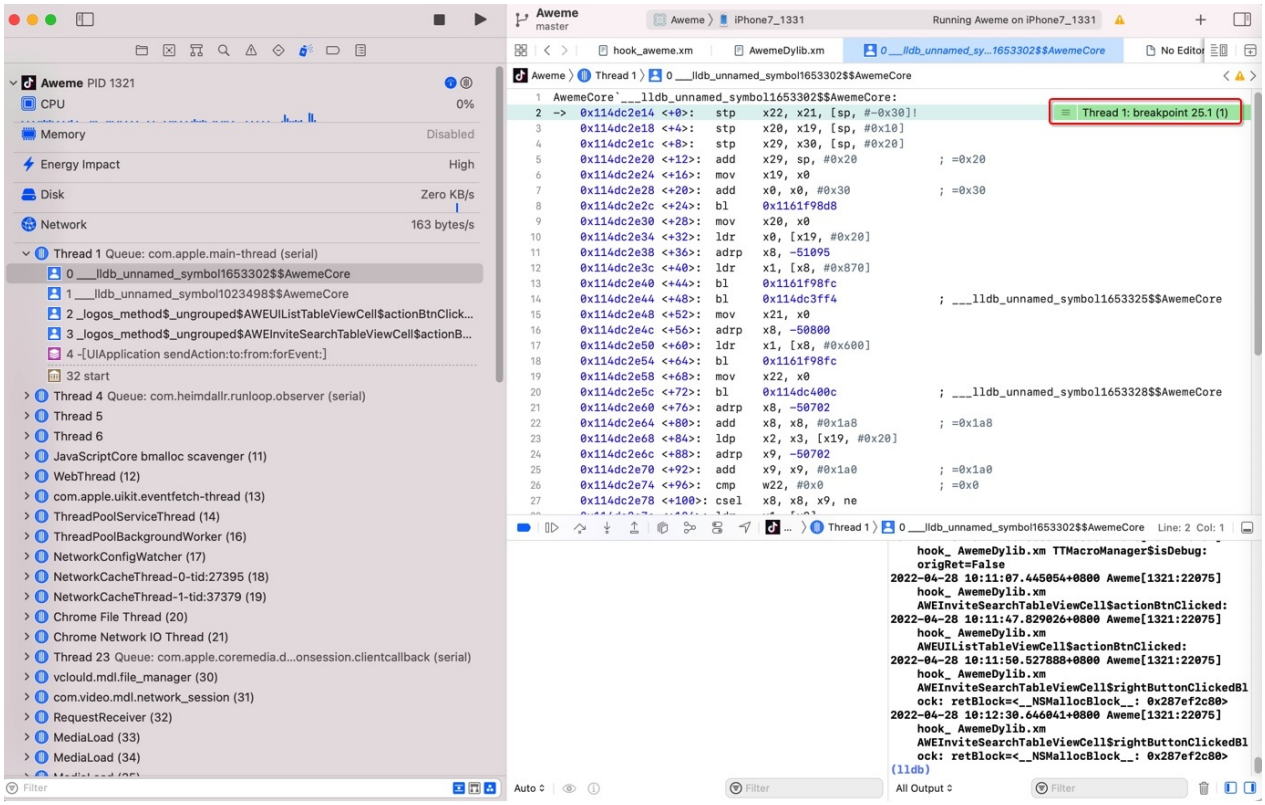
-》然后也知道了：

- `函数调用堆栈`
  - 英文专业叫法：`backtrace`
    - 此处是 `完整的backtrace` ，所以叫：`full backtrace`
    - 所以LLDB调试 函数调用堆栈 缩写是：
      - `bt` = `backtrace`

断点触发详情

Xcode触发断点时 还带提示第几次触发

举例：

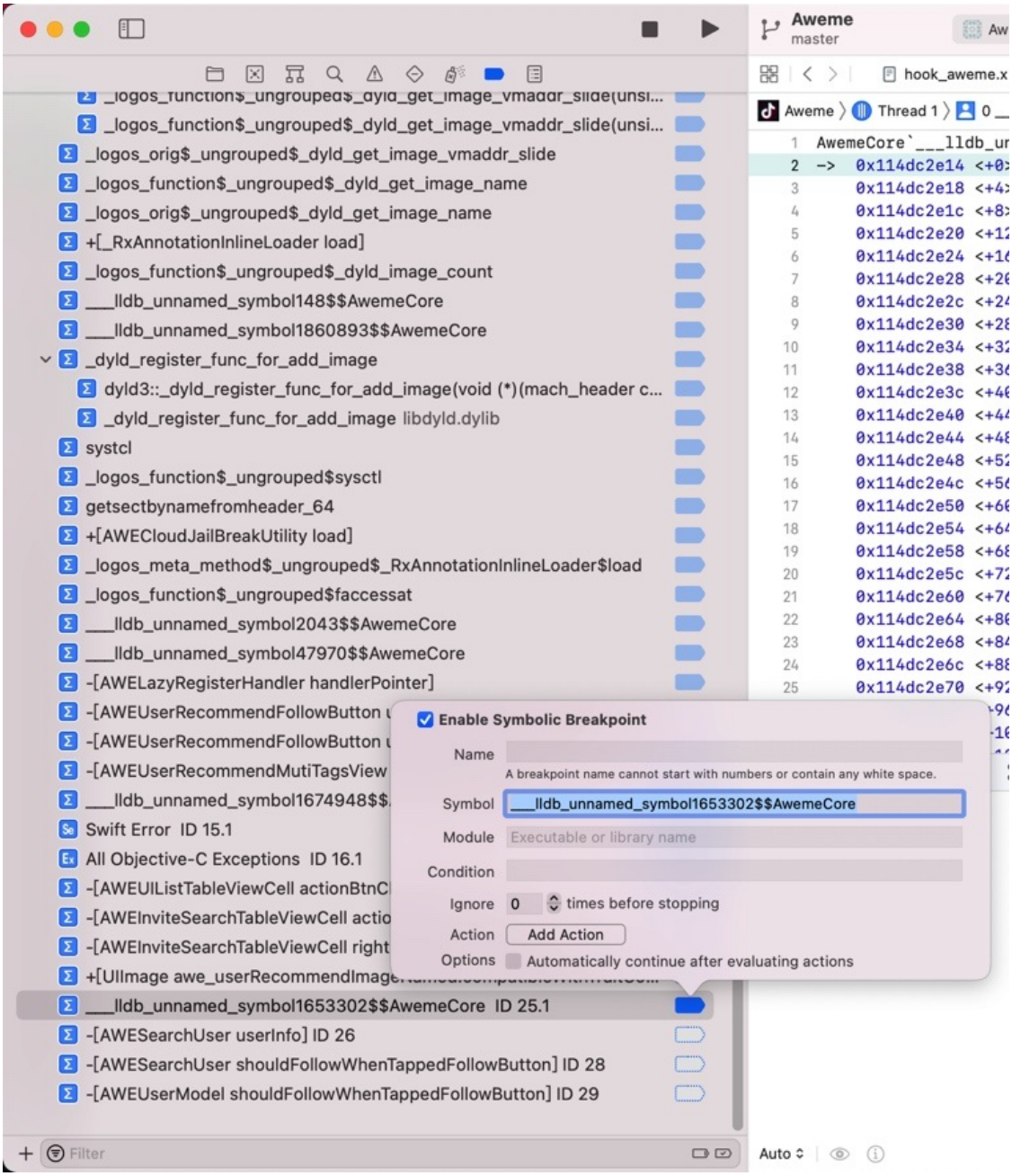


断点触发到 `___lldb_unnamed_symbol1653302$$AwemeCore` 时，显示的：

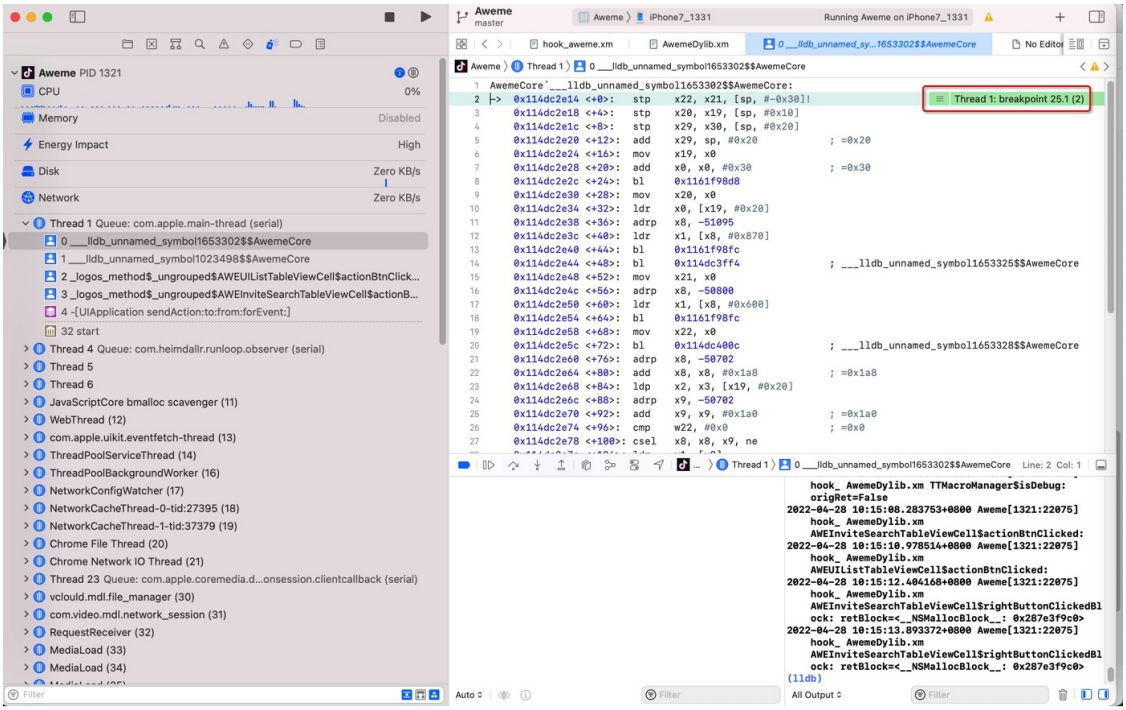
`Thread 1: breakpoint 25.1 (1)`

表示：

- 所属线程：Thread 1
- 代码停下原因：断点breakpoint
- 具体是哪个断点：25.1
    - 切换到断点列表中可以看到是：

- 

  - 对应着是自己加的符号断点
    - ___lldb_unnamed_symbol1653302$$AwemeCore
- (1)：表示此处断点触发了第一次
  - 后续如果再次触发，数字会依次增加，比如 `(2)`、`(3)` 等等



停下来的原因

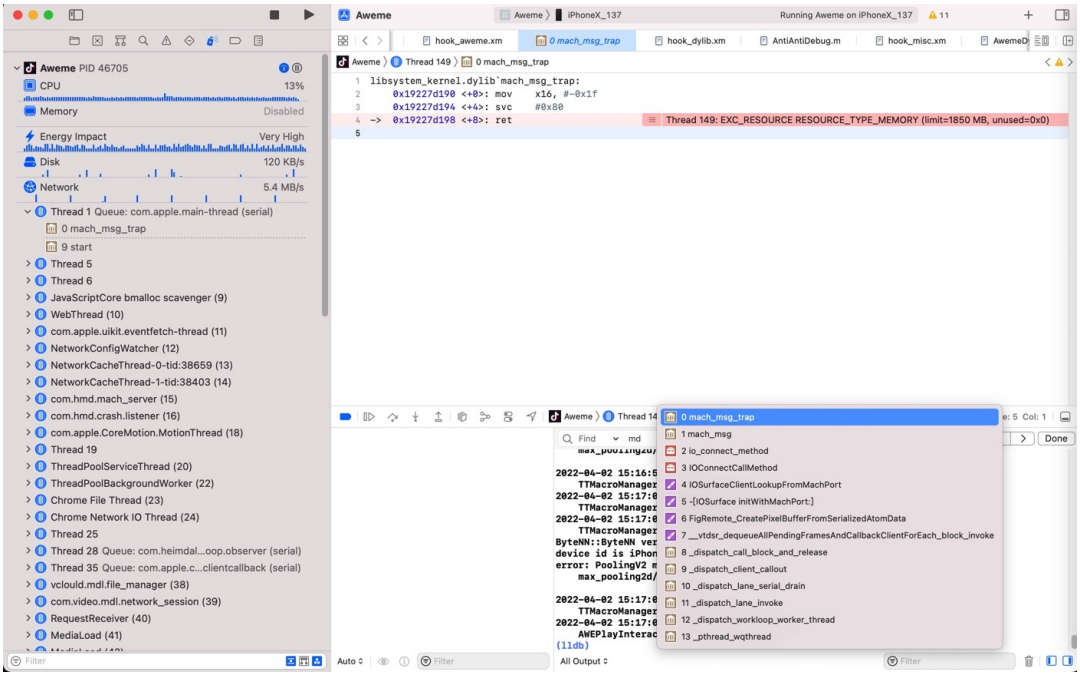
Xcode代码停下来，会显示具体原因

用Xcode去调试程序时：

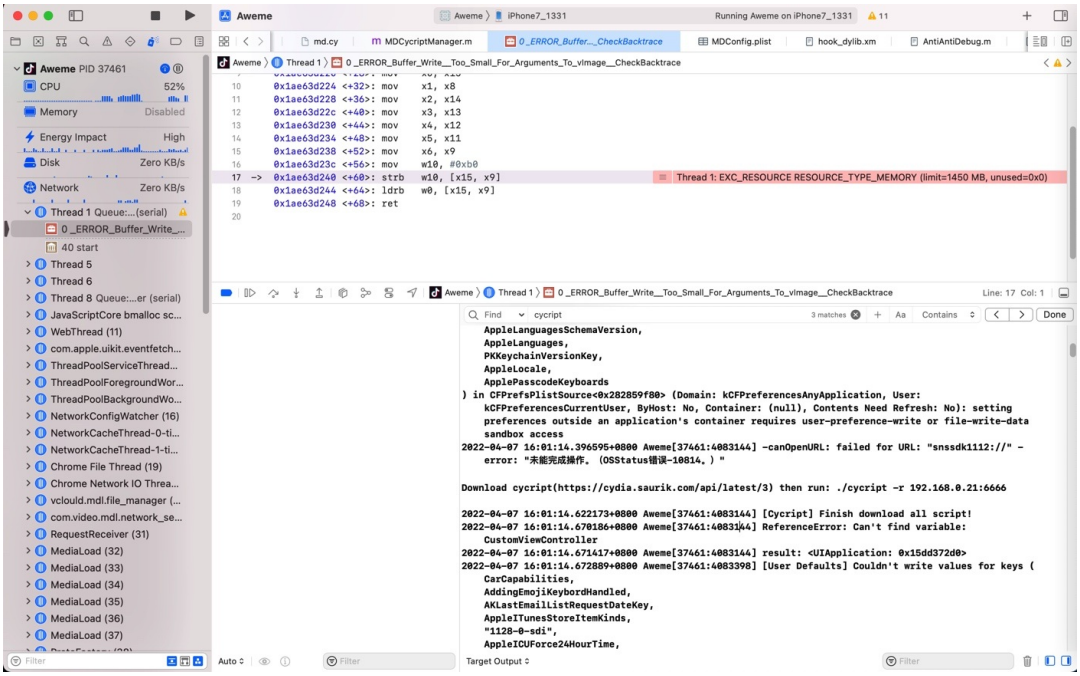
代码有时候会停下来 暂定运行

此时可以看出对应的代码停下来的原因：

- SIGTRAP：进入了某个trap 陷阱？ 触发了某个条件而进入了陷阱？
  -
- breakpoint = 之前自己设置的某个断点，生效了，触发了某个断点，导致代码停下来
  -
- SIGABRT=abort终止 = 发生了某些异常导致程序终止

-
- 异常：
  - EXC_RESOURCE RESOURCE_TYPE_MEMORY
    - Thread 149: EXC_RESOURCE RESOURCE_TYPE_MEMORY (limit=1850 MB, unused=0x0)
    - 

    - Thread 1: EXC_RESOURCE RESOURCE_TYPE_MEMORY (limit=1450 MB, unused=0x0)



- 等等

总之：

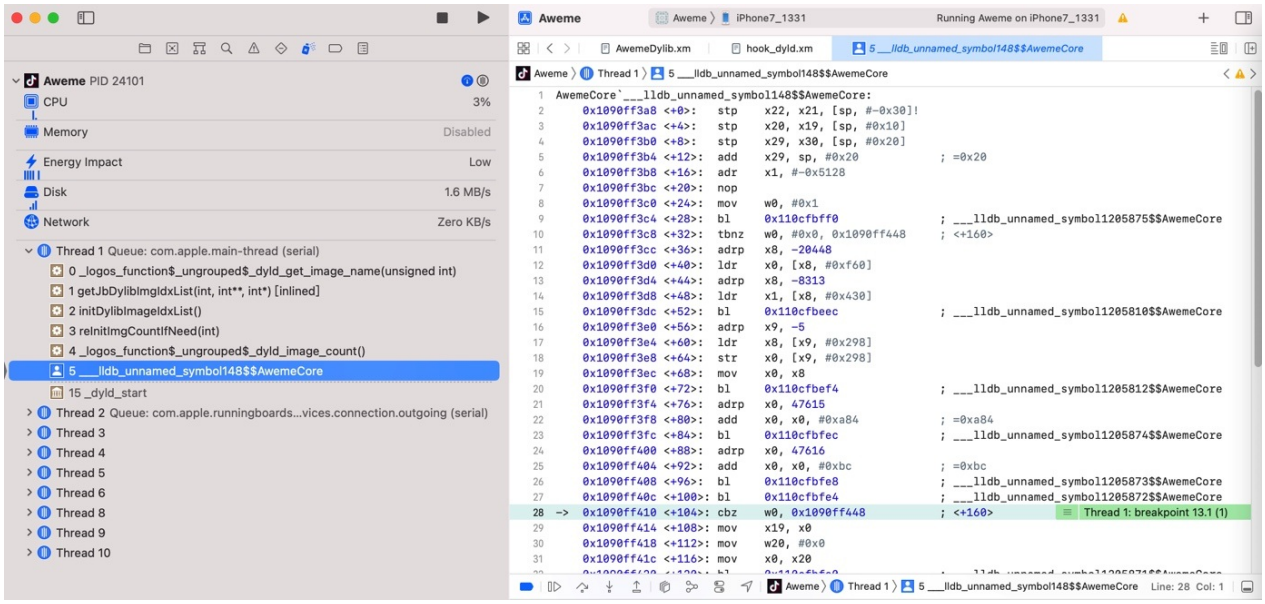
Xcode调试期间，代码停下来，一般会显示对应的原因的。

箭头指向的指令

当调试时，代码停下来时，会看到的汇编代码中的有个： `箭头` = `->`

而 `->` 指向的是将要运行的指令代码，不是当前正在运行的代码

举例：




```
AwemeCore`___lldb_unnamed_symbol148$$AwemeCore:
• • •
    0x1090ff404 <+92> : add    x0, x0, #0xbc          ; =0xbc
    0x1090ff408 <+96> : bl     0x110cfbfe8           ; ___lldb_unnamed_symbol1205873$$AwemeCore
    0x1090ff40c <+100> : bl    0x110cfbfe4           ; ___lldb_unnamed_symbol1205872$$AwemeCore
->  0x1090ff410 <+104> : cbz   w0, 0x1090ff448        ; <+160>
```

当前 `->` 指向的：

```
->  0x1090ff410 <+104> : cbz   w0, 0x1090ff448        ; <+160>
```

是 将要运行的 下一行的代码=指令

而当前正在运行的指令，对应着箭头 `->` 的上面的一行：

```
0x1090ff40c <+100> : bl    0x110cfbfe4           ; ___lldb_unnamed_symbol1205872$$AwemeCore
```

切换进程视图

在不同视图之间切换：

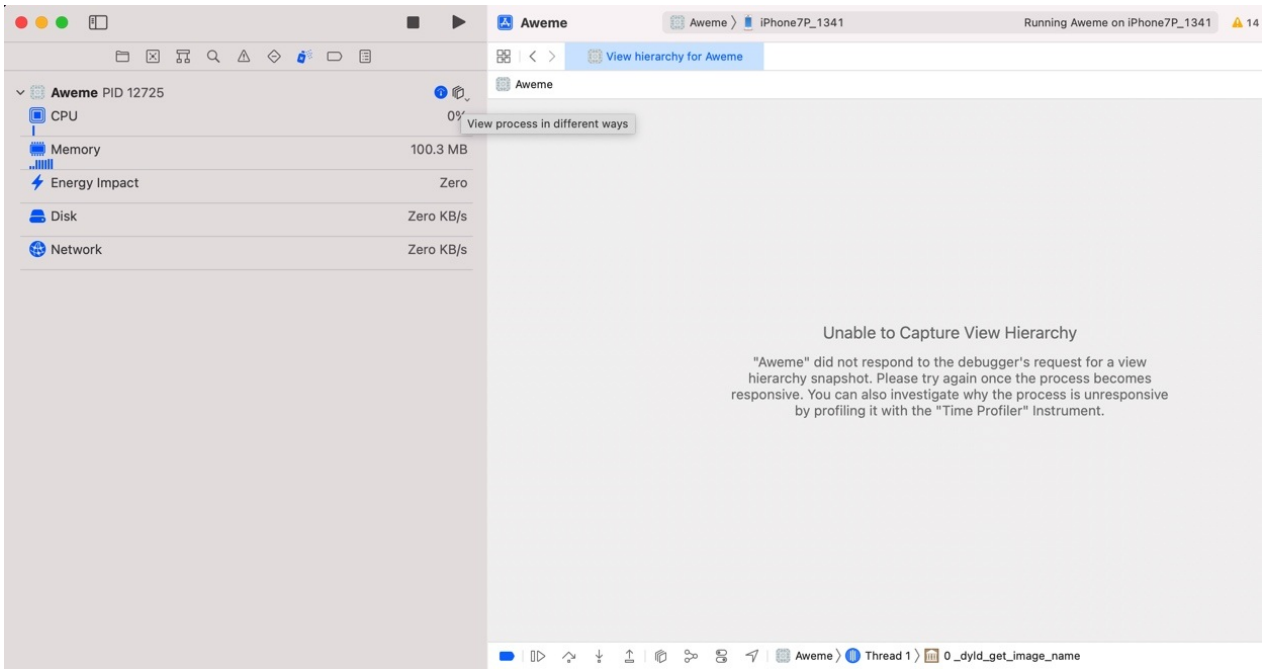
在调试期间，新点击了：

- `Debug Memory Graph`
- `Debug View Hierarchy`

后，回不去调试的Thread了

后来发现了，是点击：

`View process in different ways`



即可看到几种方式：

- 当前的是：`View UI Hierarchy`
  -

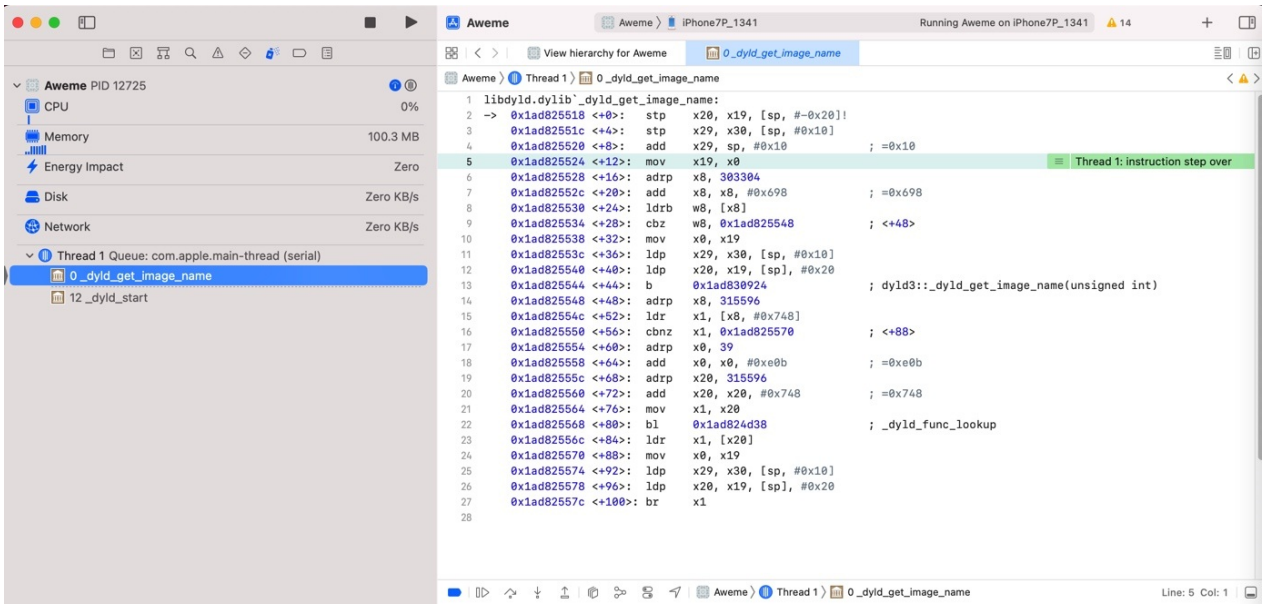
切换到：

- `View Process by Thread`

-

就是Debug默认的，常见的形式了：

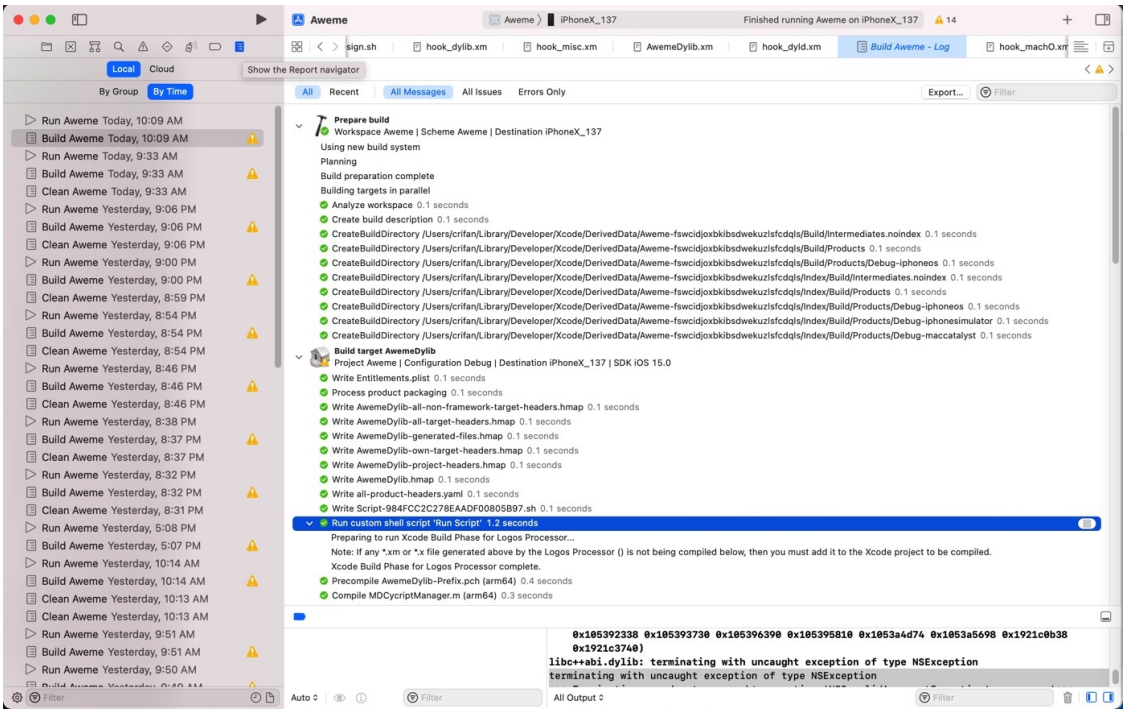
以线程方式查看进程，其中能看到函数调用堆栈的内容：

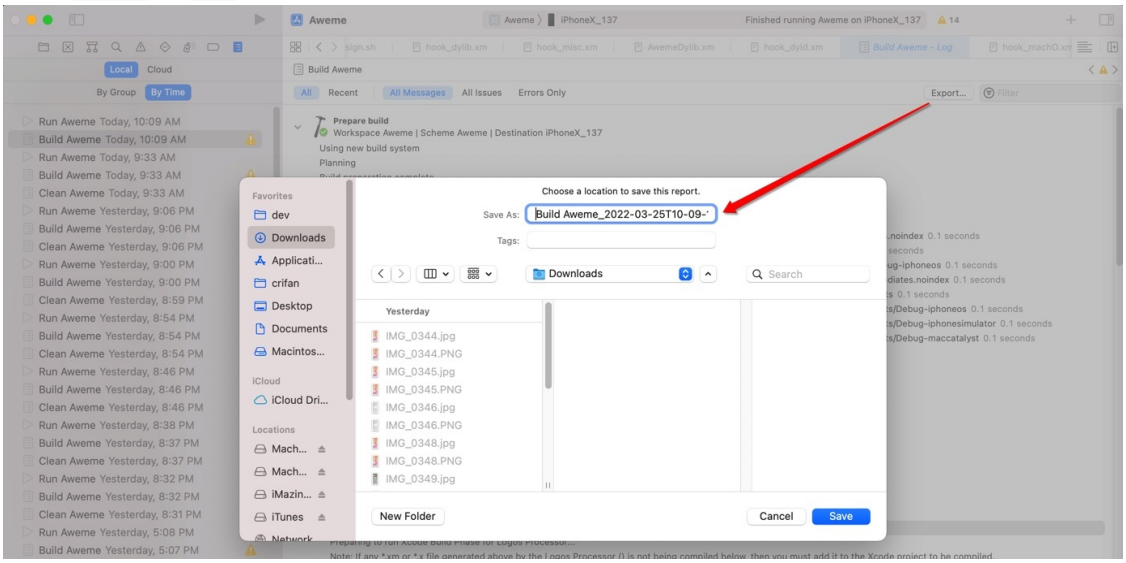


Report Navigator

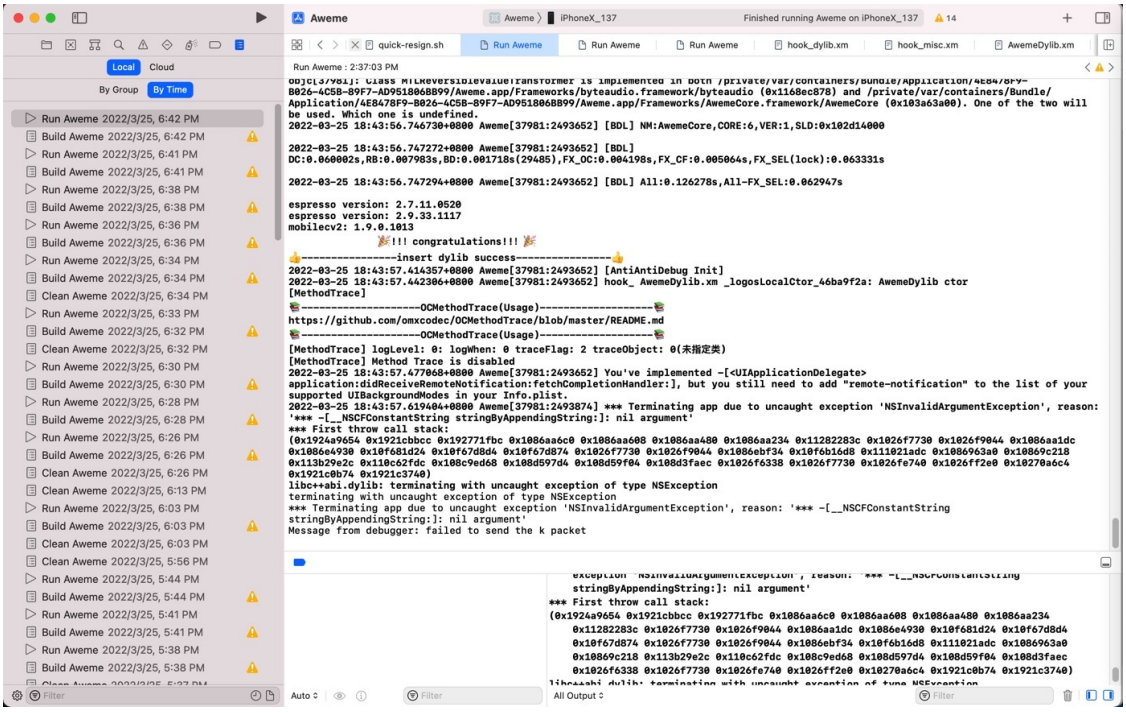
对于项目的编译和运行的详细过程，可以通过Report navigator中查看

`Xcode` -> `Show the Report navigator` -> 点击对应 `Build` 或 `Run` ->右边即可显示出编译或运行的详细过程log日志

- Build
  - 举例
    - 

  - 导出：点击 `Export` 导出 `txt` 格式的日志
    - 

- Run
  - 举例

- 

  - 导出：全选 -> 复制 -> 粘贴导出

TODO：

【已解决】XCode项目的编译链接安装等内部详细日志和过程

Build Phases

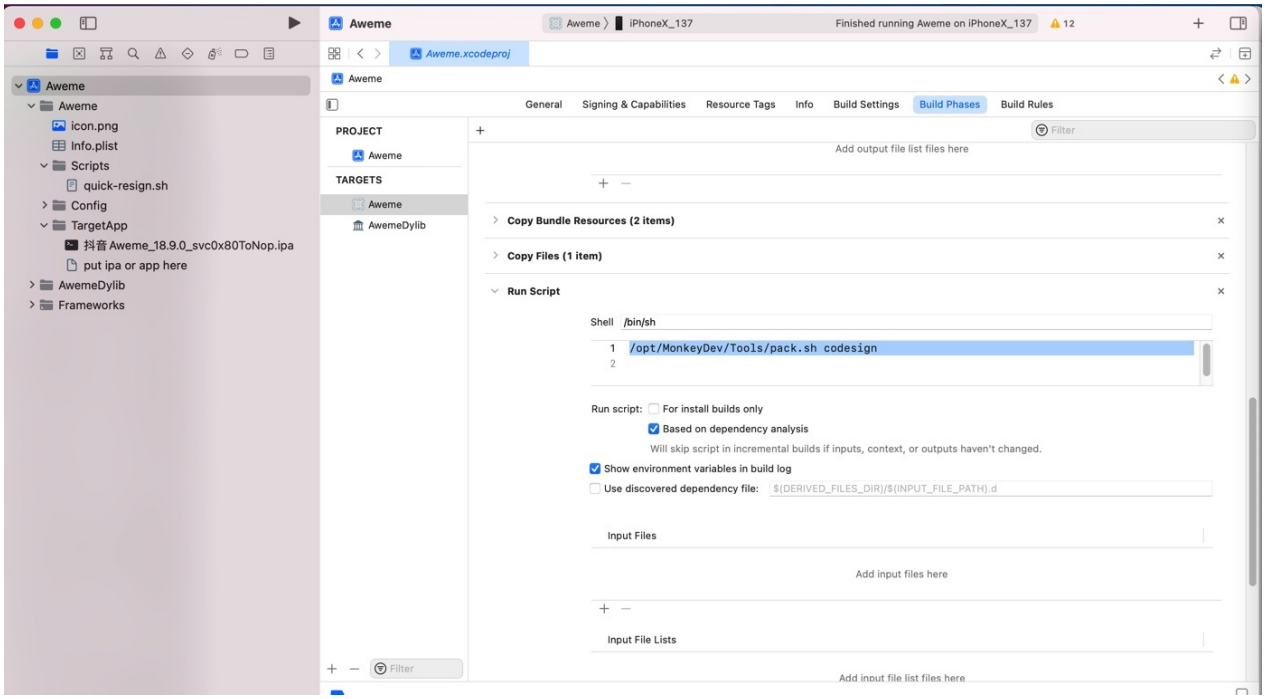
Run Script

Run Script对应着Build中的Script-xxx.sh

`Xcode -> Target -> Build Phases -> Run Script`

其中输入的要运行的脚本代码

`/opt/MonkeyDev/Tools/pack.sh`



->最终对应着：

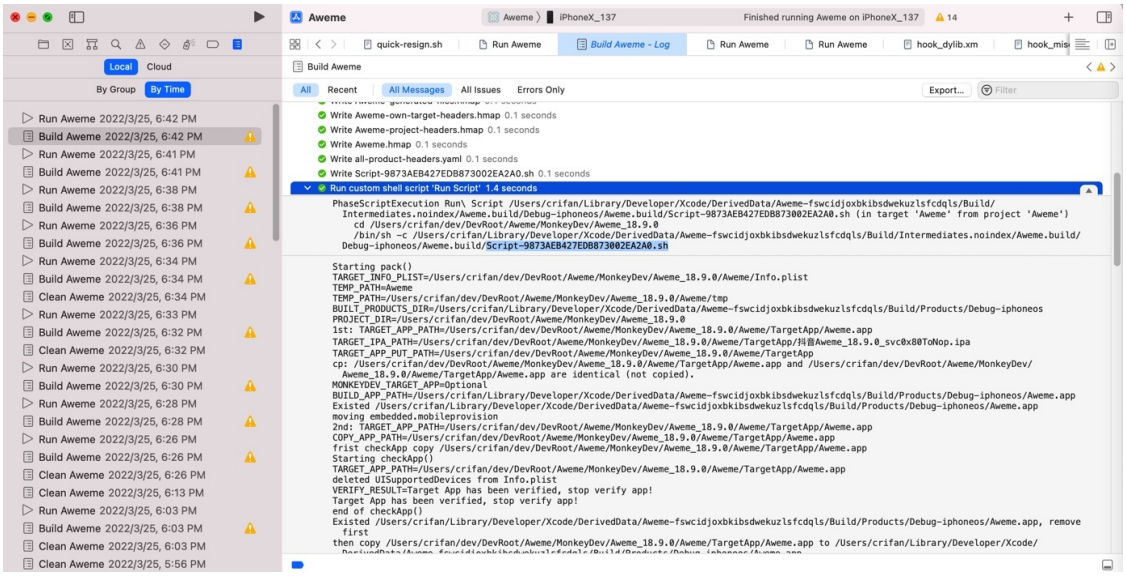
Build日志中的：

- `Script-9873AEB427EDB873002EA2A0.sh`
  - 完整路径：`/Users/crifan/Library/Developer/Xcode/DerivedData/Aweme-fswcidjoxbkibsdwekuzlsfcdqls/Build/Intermediates.noindex/Aweme.build/Debug-iphoneos/Aweme.build/Script-9873AEB427EDB873002EA2A0.sh`
  - 脚本内容是：

    ```
    #!/bin/sh
    /opt/MonkeyDev/Tools/pack.sh
    ```

  - 对应的运行脚本的命令是

    ```
    PhaseScriptExecution Run\ Script /Users/crifan/Library/Developer/Xcode/DerivedData/Aweme-fswcidjoxbkibsdwekuzlsfcdqls/Build/Intermediates.noindex/Aweme.build/Debug-iphoneos/Aweme.build/Script-9873AEB427EDB873002EA2A0.sh (in target 'Aweme' from project 'Aweme')
        cd /Users/crifan/dev/DevRoot/Aweme/MonkeyDev/Aweme_18.9.0
        /bin/sh -c /Users/crifan/Library/Developer/Xcode/DerivedData/Aweme-fswcidjoxbkibsdwekuzlsfcdqls/Build/Intermediates.noindex/Aweme.build/Debug-iphoneos/Aweme.build/Script-9873AEB427EDB873002EA2A0.sh
    ```

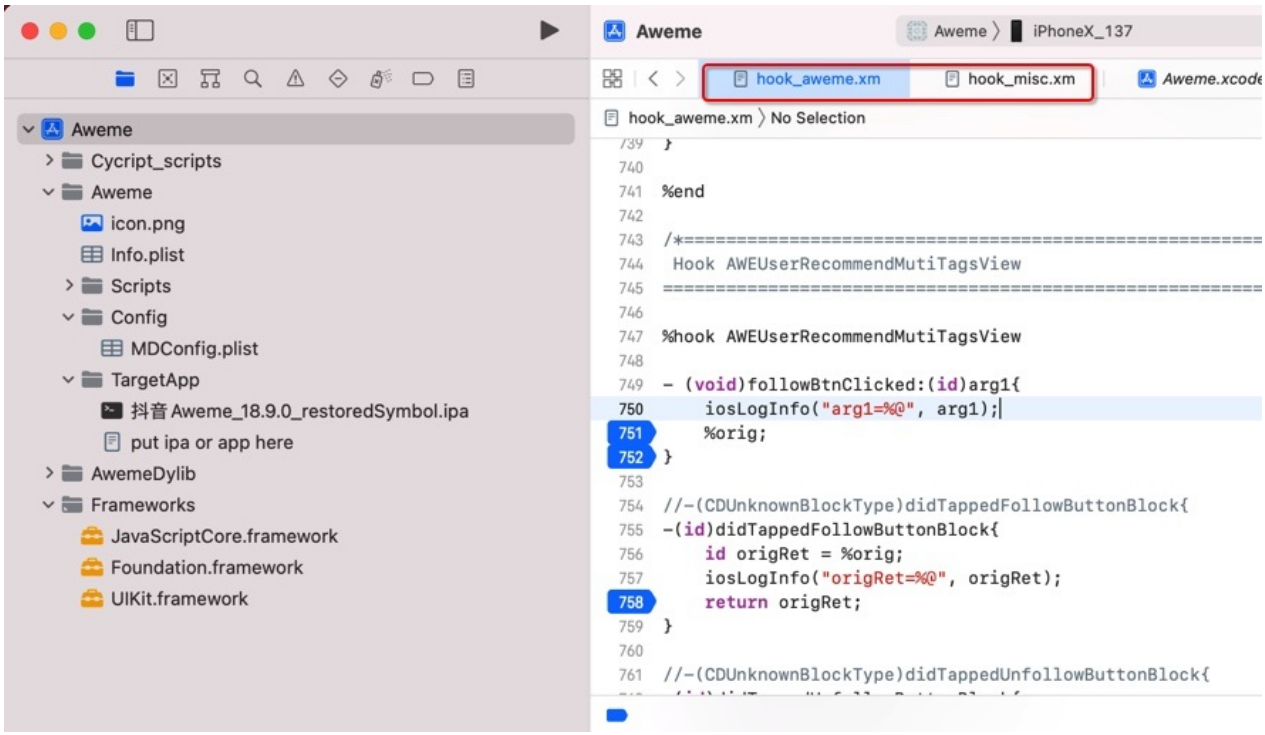


其他

可以通过打开文件，打开另外一个项目中的文件

之前调试抖音的Xcode项目，期间会触发断点运行到，另外一个插件的代码

所以之前是当前Xcode项目自动打开了另外项目中的文件：



此处是希望继续保持的状态。

另外也担心不小心关闭掉了打开的文件，好像就没机会再去打开到对应代码文件了？

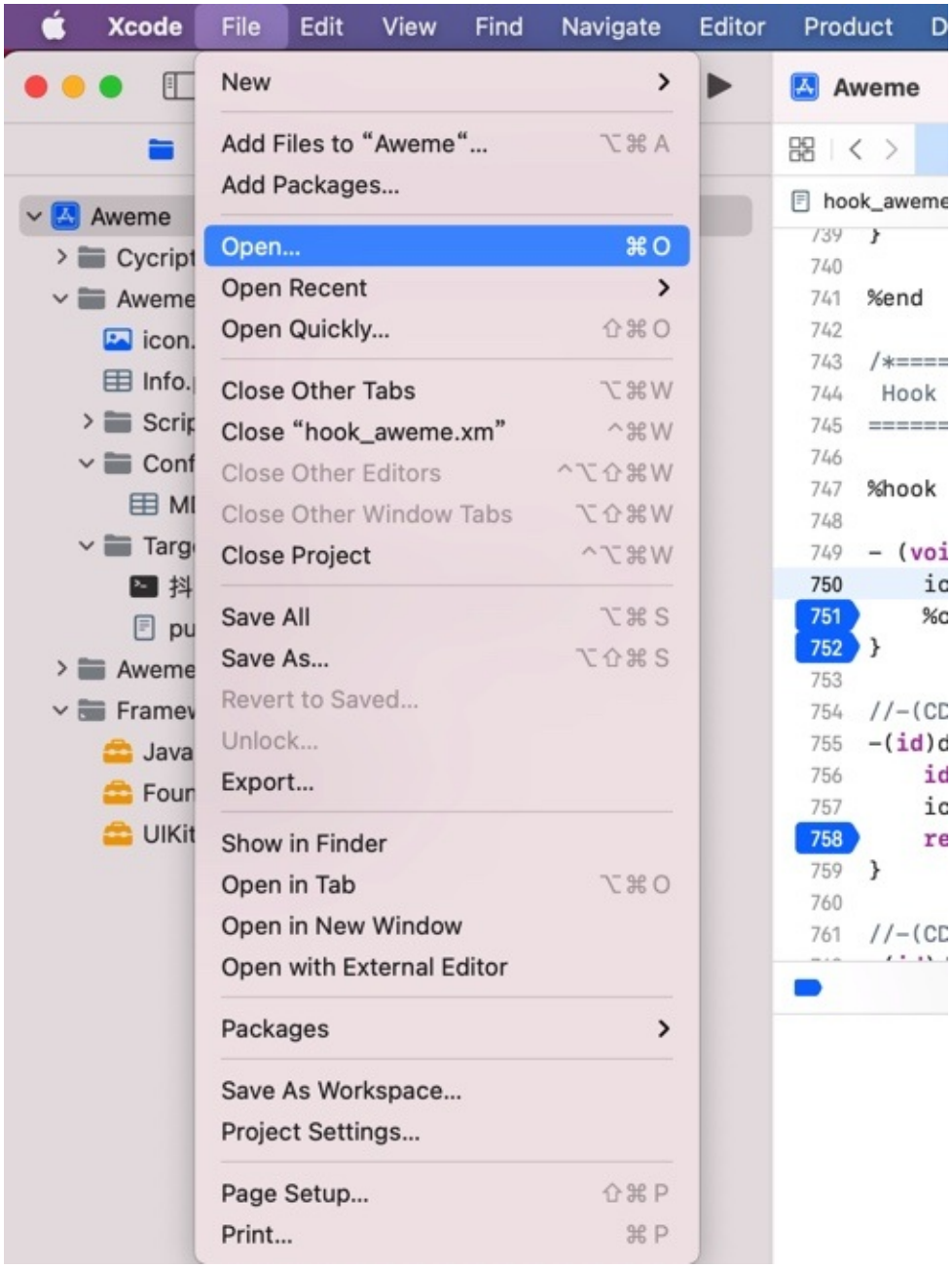
比如想要添加新的断点，然后就不方便再去重新找到另外项目中的代码文件，再去新增断点

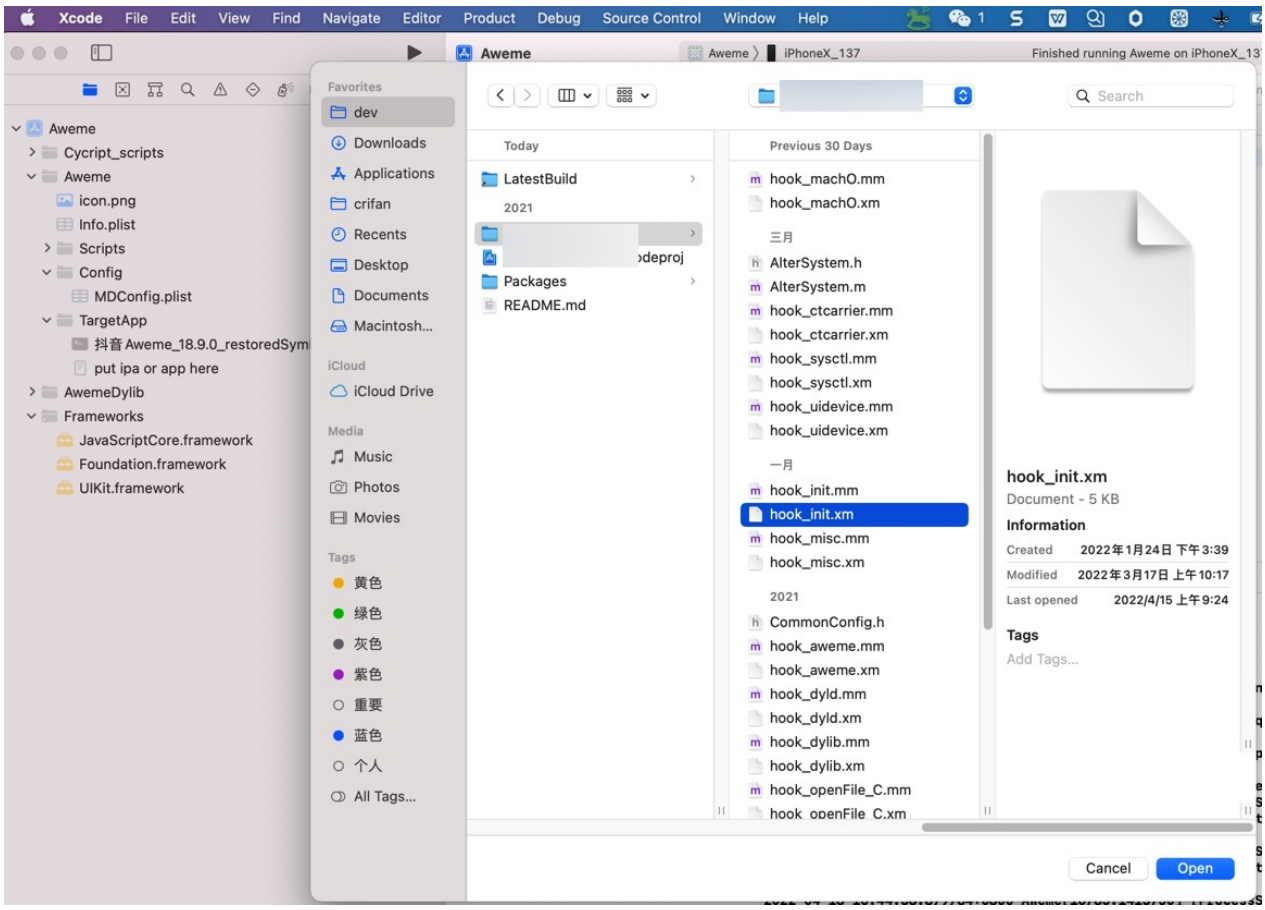
后来突然想到：

其实可以通过Xcode中的，打开文件，的功能，去打开，任何位置，包括别的Xcode项目中的文件的

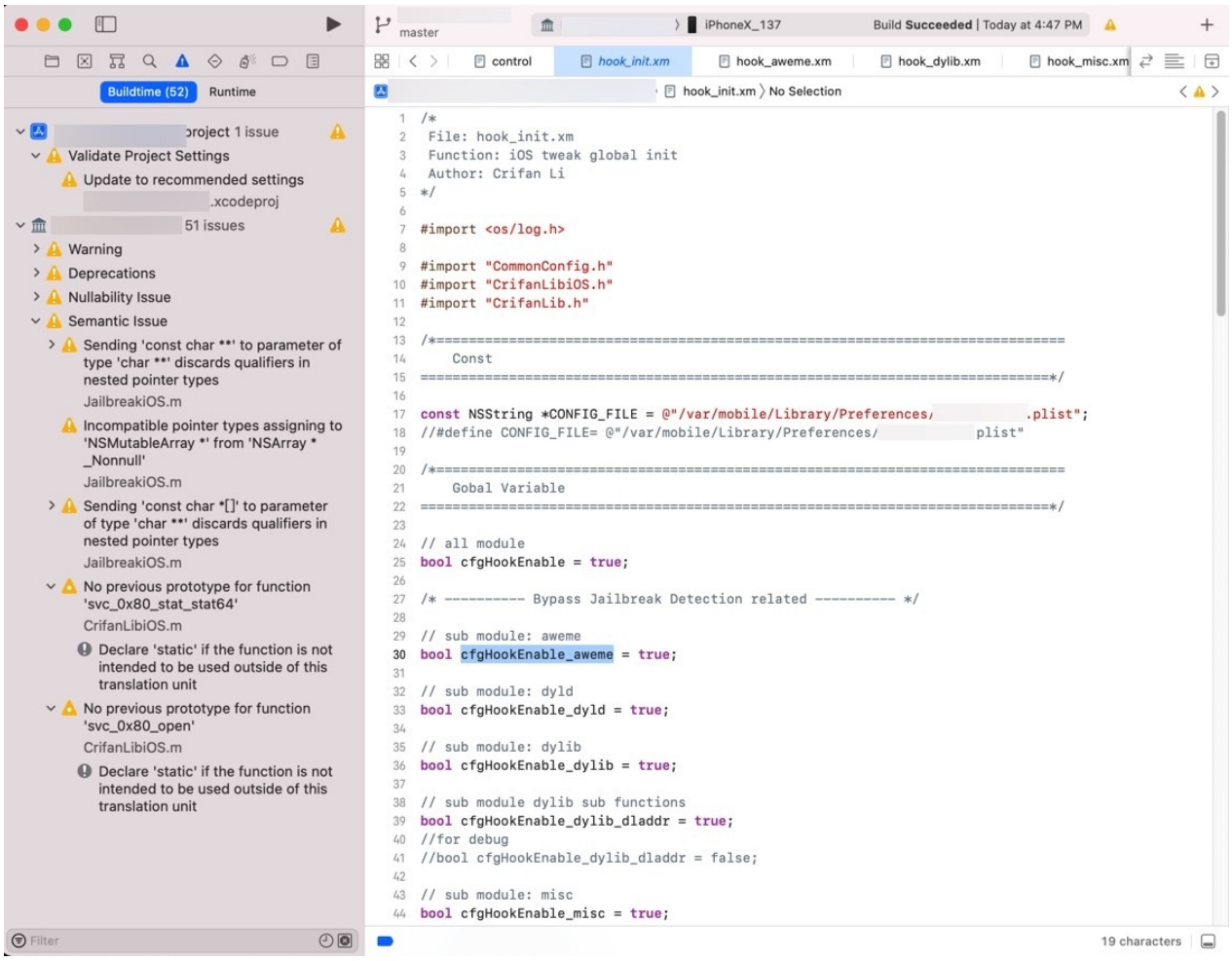
去试试，此处故意去打开一个，之前没有打开的文件：

`hook_init.xm`


Xcode
File
Edit
View
Find
Navigate
Editor
Product
New
Add Files to “Aweme“... ⌥⌘A
Add Packages...
Open... ⌘O
Open Recent
Open Quickly... ⇧⌘O
Close Other Tabs ⌥⌘W
Close “hook_aweme.xm” ^⌘W
Close Other Editors ^⌥⇧⌘W
Close Other Window Tabs ⌥⇧⌘W
Close Project ^⌥⌘W
Save All ⌥⌘S
Save As... ⌥⇧⌘S
Revert to Saved...
Unlock...
Export...
Show in Finder
Open in Tab ⌥⌘O
Open in New Window
Open with External Editor
Packages
Save As Workspace...
Project Settings...
Page Setup... ⇧⌘P
Print... ⌘P
Aweme
hook_aweme
%end
Hook
%hook



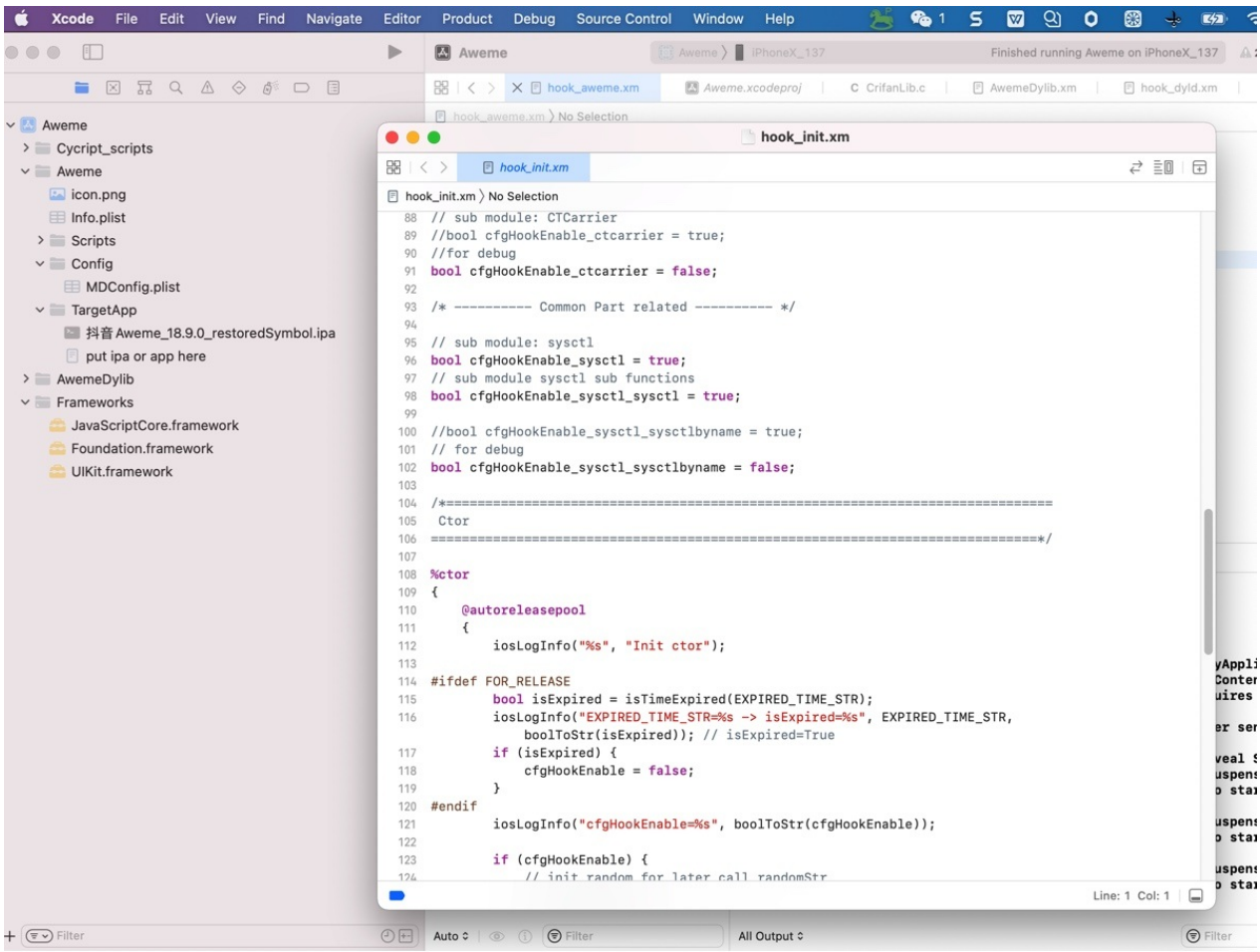
但是竟然是跳转到：另外该Xcode项目中了



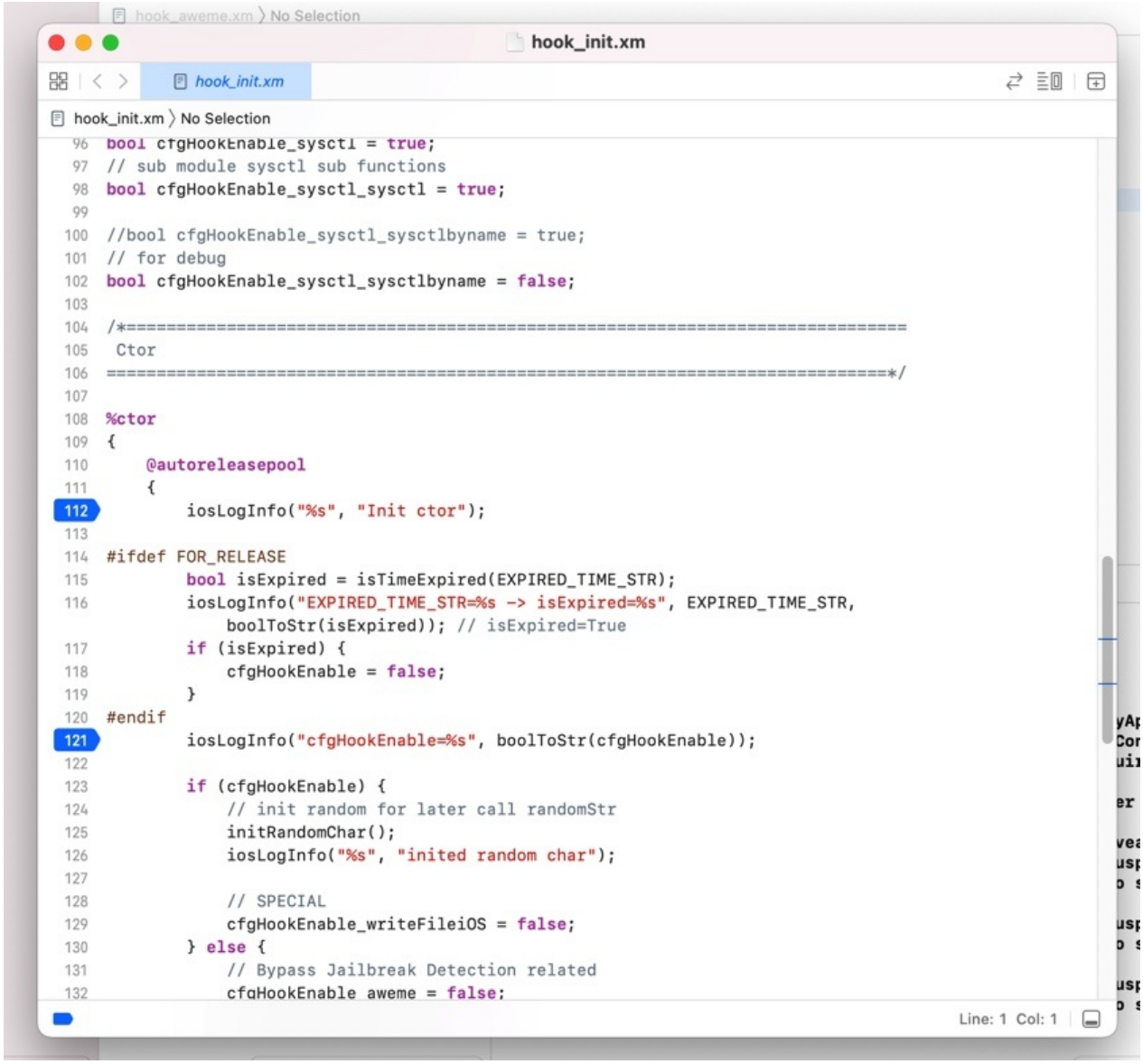
而此处，当前Xcode项目中，并没有打开。

那故意把另外的Xcode项目关闭后，再去试试

会弹框显示，单独显示



然后继续去试了试，发现也是可以单独加断点的：



-》基本上满足了我们的要求：

Xcode的A项目中，打开B项目中的文件

用途是：

A项目中的Xcode部分断点，会触发B项目中的文件的代码生效，用于调试查看B项目源码运行情况

注：B项目编译期间，是保留了 `symbol` 的，所以是有机会源码调试的。

附录

下面列出相关参考资料。

参考资料

- 【已解决】Mac中F7快捷键被占用
- 【无需解决】Xcode调试log日志窗口不输出log日志了
-
- lldb常用命令与调试技巧 - 掘金 (juejin.cn)
-