

# Swift源码编译

## 编译环境

## 编译步骤

第一步：clone swift源码

第二步：update-checkout

第三步：编译

第四步：使用 VSCode 来调试 Swift。

## 编译环境

- Xcode version 11.5
- Python 2.x
- brew install cmake ninja

## 编译步骤

### 第一步：clone swift源码

```
1 git clone --branch swift-5.2.4-RELEASE https://github.com/apple/swift.git
```

这里我编译的是 `swift-5.2.4-Release`，因为在编译源码的时候，最新的就是 `5.2.4 RELEASE`。

如果想要编译最新的，大家可以自行在官网上找到[分支](#)，同时注意对应的 `Xcode` 版本要匹配（这个在官方文档编译 `Swift` 的时候会有说明）。目前找到的最新分支应该是 `5.3.1-release`。

### 第二步： `update-checkout`

确保你当前的目录是在 `swift-source` 目录下，然后执行如下命令：

```
1 ./swift/utils/update-checkout --tag swift-5.2.4-RELEASE --clone
```

这一步非常重要，因为 `update-checkout` 会 `clone` 编译 `swift` 相关的库，不然后面在编译 `swift` 源码的过程中一定会失败。

注意：这里我提供了执行了第一步和第二步后的源码文件，如下。大家下载之后，执行第三步即可

- 1 链接：[https://pan.baidu.com/s/1rkKm\\_tXwGEp5xfWrvxvyYA](https://pan.baidu.com/s/1rkKm_tXwGEp5xfWrvxvyYA)
- 2 密码：2sev

当然这里也有编译好 (5.3.1) 的第一步和第二步的源码，下载直接执行第三步就行：

<input type="checkbox"/>	 swift-source.zip	2.33GB	2020.11.28 15:31
<input type="checkbox"/>	 swift-source-5.3.1-第二步.zip	2.43GB	2020.12.04 19:06
<input type="checkbox"/>	 5.3.1第三步.zip	10.95GB	2020.12.04 19:08

也有编译好的 5.3.1 的代码，可以找一个空闲的移动硬盘，下载解压

注意：这里需要将你的盘符路径修改为： `Volumes/MobileHDD`，确保访问 `Swift-Source` 源码的路径是： `Volumes/MobileHDD/swift-source/` 即可，这是最便捷的方法！！

<input type="checkbox"/>	 swift-source.zip	   2.33GB	2020.11.28 15:31
<input type="checkbox"/>	 swift-source-5.3.1-第二步.zip	2.43GB	2020.12.04 19:06
<input type="checkbox"/>	 5.3.1第三步.zip	10.95GB	2020.12.04 19:08

### 第三步：编译

编译过程中既可以使用 `ninja`，也可以使用 `xcode` 来进行编译。但是实际测试过程中 `xcode` 编译之后的支持性不是特别的好。所以这里就推荐使用 `ninja` 来作为编译工具

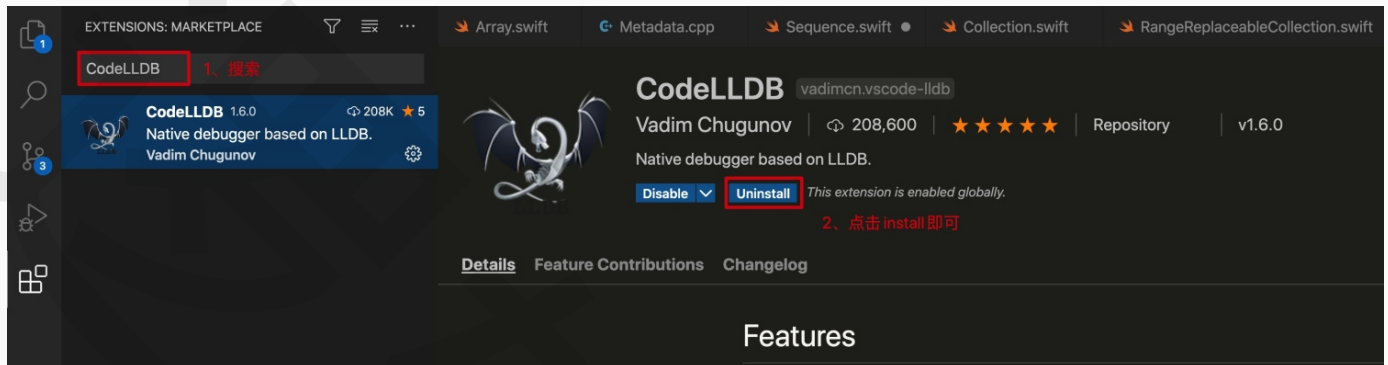
如果还有想要使用 `xcode` 编译的，大家可以自己去官网文档当中来进行编译。

执行如下命令进行编译：

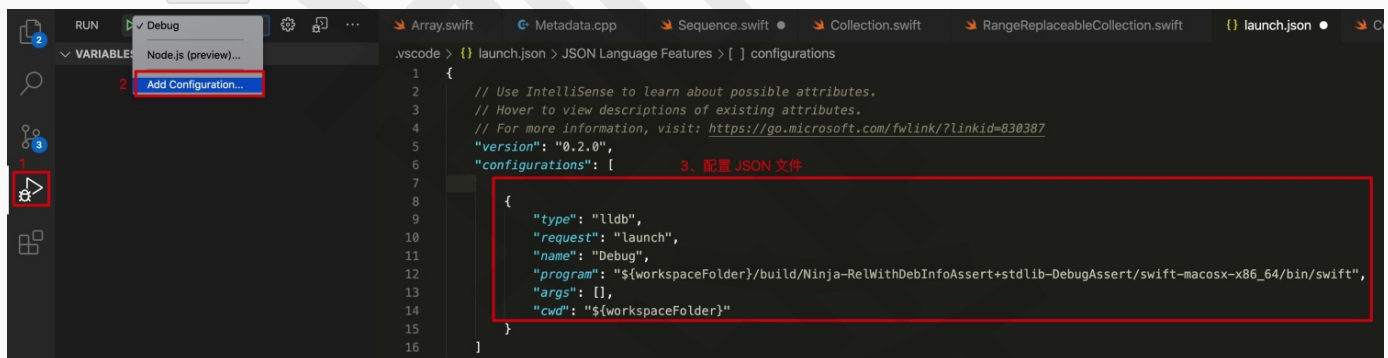
```
1 ./swift/utils/build-script -r --debug-swift-stdlib --lldb
```

#### 第四步：使用 `VSCode` 来调试 `Swift`。

首先我们需要安装一个插件 `CodeLLDB`



接下来配置 `JSON` 文件



```
1 {
2   "version": "0.2.0",
3   "configurations": [
4
5     {
6       "type": "lldb",
7       "request": "launch",
8       "name": "Debug",
9       "program": "${workspaceFolder}/build/Ninja-RelWithDeb
10 InfoAssert+stdlib-DebugAssert/swift-macosx-x86_64/bin/swift",
11       "args": [],
12       "cwd": "${workspaceFolder}"
13     }
14   ]
15 }
```

注意：上述 program 的文件路径和你编译的文件路径相同即可

run 起来之后：

```

1  ; id = {0x00000000}, range = [0x0000000000001000-0x0000000000001062), mangled="
2  ; Source location: unknown
3  10F319000: 5F                                popq   %rdi
4  10F319001: 6A 00                            pushq  $0x0
5  10F319003: 48 89 E5                        movq   %rsp, %rbp
6  10F319006: 48 83 E4 F0                      andq   $-0x10, %rsp
7  10F31900A: 48 83 EC 10                      subq   $0x10, %rsp
8  10F31900E: 8B 75 08                        movl   0x8(%rbp), %esi
9  10F319011: 48 8D 55 10                      leaq   0x10(%rbp), %rdx
10 10F319015: 48 8D 0D E4 EF FF FF          leaq   -0x101c(%rip), %rcx
11 10F31901C: 4C 8D 45 F8                      leaq   -0x8(%rbp), %r8
12 10F319020: E8 3D 00 00 00                callq  0x10f319062 ; dyldbootstrap::start
13 10F319025: 48 8B 7D F8                      movq   -0x8(%rbp), %rdi
14 10F319029: 48 83 FF 00                      cmpq   $0x0, %rdi
15 10F31902D: 75 10                          jne    0x10f31903f ; <+63>
16 10F31902F: 48 89 EC                        movq   %rbp, %rsp
17 10F319032: 48 83 C4 08                      addq   $0x8, %rsp

```

过掉断点之后：

```

1  ; id = {0x00000000}, range = [0x0000000000001000-0x0000000000001062), mangled="_dyld
2  ; Source location: unknown
3  10F319000: 5F                                popq   %rdi
4  10F319001: 6A 00                            pushq  $0x0
5  10F319003: 48 89 E5                        movq   %rsp, %rbp
6  10F319006: 48 83 E4 F0                      andq   $-0x10, %rsp
7  10F31900A: 48 83 EC 10                      subq   $0x10, %rsp
8  10F31900E: 8B 75 08                        movl   0x8(%rbp), %esi
9  10F319011: 48 8D 55 10                      leaq   0x10(%rbp), %rdx
10 10F319015: 48 8D 0D E4 EF FF FF          leaq   -0x101c(%rip), %rcx
11 10F31901C: 4C 8D 45 F8                      leaq   -0x8(%rbp), %r8
12 10F319020: E8 3D 00 00 00                callq  0x10f319062 ; dyldbootstrap::start(dyld
13 10F319025: 48 8B 7D F8                      movq   -0x8(%rbp), %rdi
14 10F319029: 48 83 FF 00                      cmpq   $0x0, %rdi
15 10F31902D: 75 10                          jne    0x10f31903f ; <+63>
16 10F31902F: 48 89 EC                        movq   %rbp, %rsp
17 10F319032: 48 83 C4 08                      addq   $0x8, %rsp

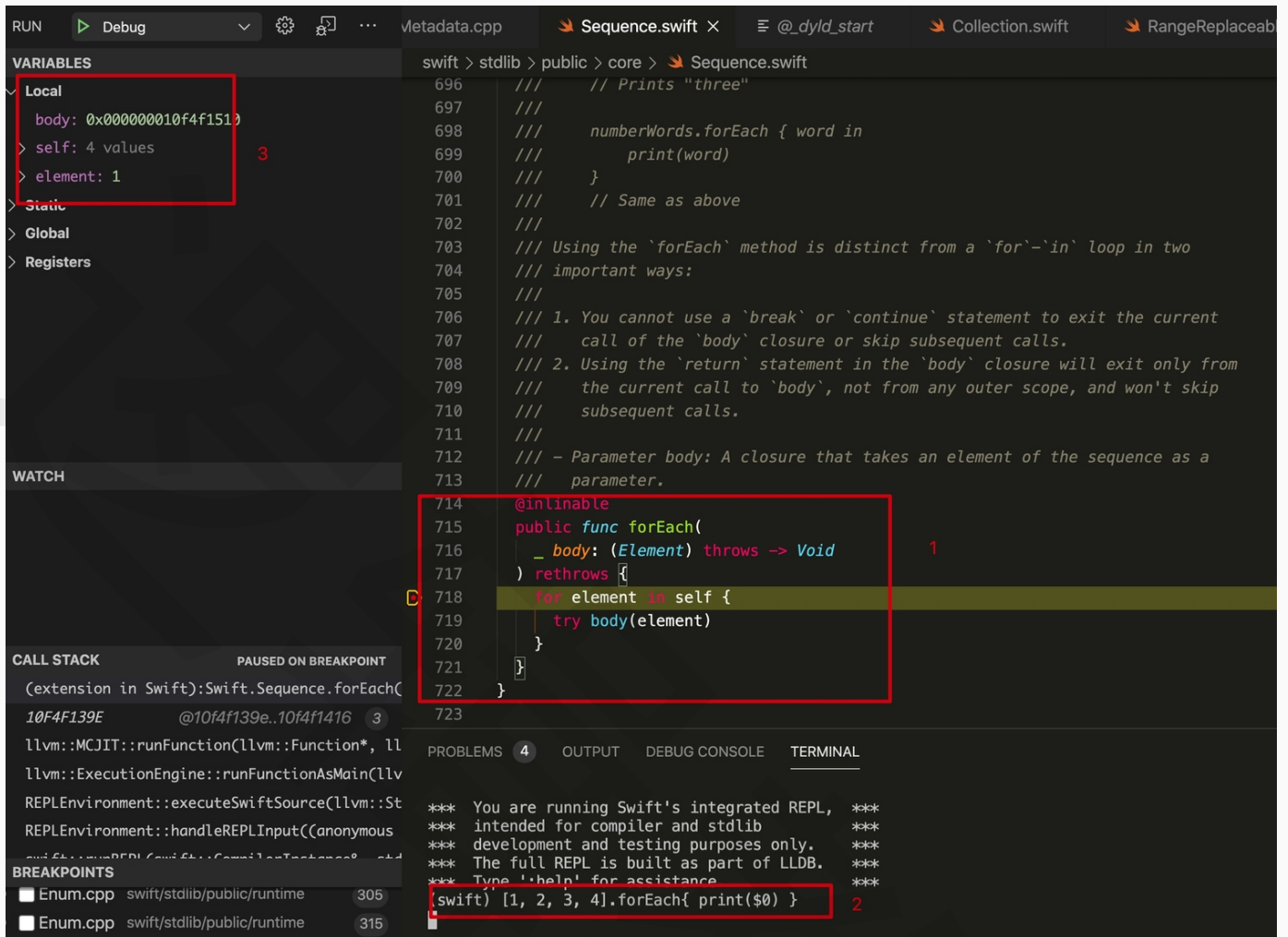
```

```

*** You are running Swift's integrated REPL, ***
*** intended for compiler and stdlib          ***
*** development and testing purposes only.     ***
*** The full REPL is built as part of LLDB.   ***
*** Type ':help' for assistance.              ***
(swift)

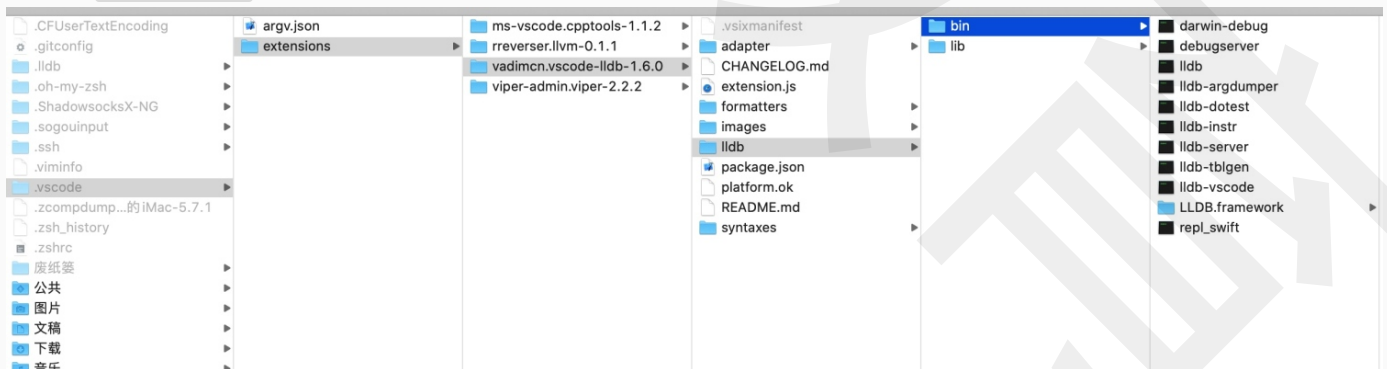
```

e.g:

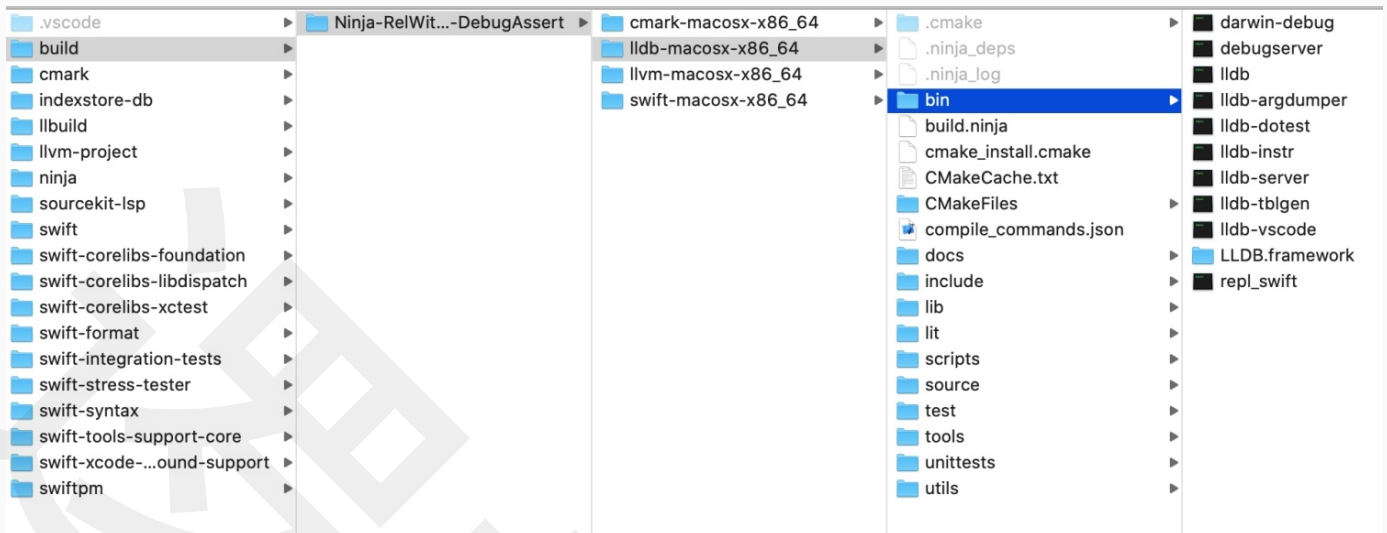


在调试 .swift 文件的时候，可能区域 3 不显示，解决办法如下：

先找到 CodeLLDB 的安装目录：



然后找到编译过后的 LLD 文件目录,把这里面的文件全部拷贝到上面 CodeLLDB 的目录!



同时修改 CodeLLDB 的 lib 文件下面的 dylib 文件

