

쉽고 재미있는 iOS 디버깅

안정민 한국카카오은행

목차

- LLDB 소개
- LLDB 명령어
- Chisel 명령어

LLDB 명령어

(Execution Commands)

Execution Commands

▼ HelloWorld PID 71438

CPU 0%

Memory 55 MB

Disk 6 MB/s

Network Zero KB/s

▼ Thread 1 Queue: com....thread (serial)

- 0 ViewController.viewDidLoad()
- 20 UIApplicationMain
- 21 main
- 22 start
- 23 start

▶ Thread 2

▶ Thread 3

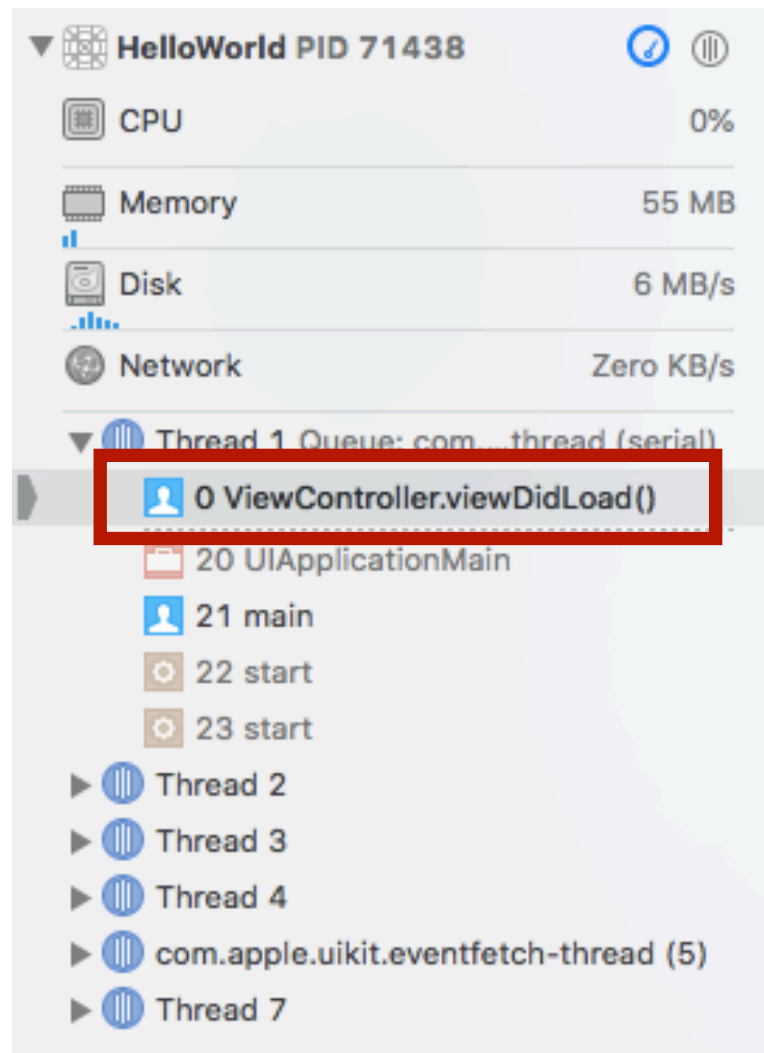
▶ Thread 4

▶ com.apple.uikit.eventfetch-thread (5)

▶ Thread 7

```
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         var count = 1
16         print("Hello world first")
17         print("Hello world second")
18         count += 1
19         a()
20         print("count : \(count)")
21         count += 1
22         print("Hello world third")
23         print("count : \(count)")
24     }
25
```

Execution Commands



▼ HelloWorld PID 71438

- CPU 0%
- Memory 55 MB
- Disk 6 MB/s
- Network Zero KB/s

▼ Thread 1 Queue: com... thread (serial)

- 0 ViewController.viewDidLoad()
- 20 UIApplicationMain
- 21 main
- 22 start
- 23 start
- Thread 2
- Thread 3
- Thread 4
- com.apple.uikit.eventfetch-thread (5)
- Thread 7

```
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         var count = 1
16         print("Hello world first")
17         print("Hello world second")
18         count += 1
19         a()
20         print("count : \(count)")
21         count += 1
22         print("Hello world third")
23         print("count : \(count)")
24     }
25
```

Execution Commands

/// 현재 스레드 목록을 보여주기

(lldb) thread list

/// Thread 2로 이동하기

(lldb) thread **select** 2

Execution Commands

/// 현재 스레드의 `stack backtrace`를 보여주기

```
(lldb) thread backtrace
```

```
(lldb) bt
```

/// 모든 스레드의 `stack backtrace`를 보여주기

```
(lldb) thread backtrace all
```

```
(lldb) bt all
```

/// 현재 스레드의 `frame`에서 1~5번 `backtrace`를 보여주기

```
(lldb) thread backtrace -c 5
```

```
(lldb) bt 5
```

Execution Commands

/// 현재 스레드에서 특정 stack frame index를 선택하기

```
(lldb) frame select 3
```

```
(lldb) fr s 3
```

/// 현재 스레드에서 현재 선택된 frame의 정보 출력하기

```
(lldb) frame info
```

```
(lldb) fr info
```

/// 현재 선택된 stack frame에서 위 아래로 이동

```
(lldb) up
```

```
(lldb) up [이동할 Frame 수]
```

```
(lldb) frame select --relative=1
```

```
(lldb) down
```

```
(lldb) down [이동할 Frame 수]
```

```
(lldb) frame select --relative=-1
```

```
(lldb) fr s -r-1
```


Execution Commands

/// local 변수 `bar` 내용 출력하기

```
(lldb) frame variable bar
```

```
(lldb) fr v bar
```

/// local 변수 `bar`을 hex로 내용 출력하기

```
(lldb) frame variable --format x bar
```

```
(lldb) fr v -f x bar
```

/// Object의 Description 출력하기

```
(lldb) frame variable -0 self
```

/// global 변수 `baz` 내용 출력하기

```
(lldb) target variable baz
```

```
(lldb) ta v baz
```

/// 현재 소스 파일에서 정의된 global/static 변수 출력하기

```
(lldb) target variable
```

```
(lldb) ta v
```

Execution Commands

* Step Over – 현재 선택된 Frame에서
소스 수준의 한 단계를 진행.

(lldb) thread step-over

(lldb) next

(lldb) n

```
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         var count = 1
16         print("Hello world first")
17         print("Hello world second")
18         count += 1
19     }
20 }
```

Thread 1: bre...

0 ViewController.viewDidLoad()

(lldb) next

Execution Commands

* Step Into - 현재 선택된 Frame에서
소스 수준의 한 단계 안으로 들어감.

(lldb) thread step-in

(lldb) step

(lldb) s

```
15         var count = 1
16         print("Hello world first")
17         print("Hello world second")
18         count += 1
19         a()
20         print("count : \(count)")
21         count += 1
22         print("Hello world third")
23         print("count : \(count)")
24     }
25
26     func a() {
27         print("Hello world Alpha")
28     }
29 }
```

Thread 1: breakpoint 1.1

0 ViewController.viewDidLoad()

(lldb)

Execution Commands

* Step Out - 현재 선택된 Frame에서
벗어남.

(lldb) thread step-out

(lldb) finish

```
14 super.viewDidLoad(),
15 var count = 1
16 print("Hello world first")
17 print("Hello world second")
18 count += 1
19 a()
20 print("count : \(count)")
21 count += 1
22 print("Hello world third")
23 print("count : \(count)")
24 }
25
26 func a() {
27     print("Hello world Alpha")
28 }
```

Thread 1: step...

(lldb) |

Execution Commands

* Thread Until - 특정 줄/주소 등 전까지
실행

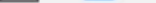
```
(lldb) thread until 22
```

```
(lldb) thread until --frame 2 10
```

(특정 프레임의 특정 라인까지 실행)

```
11 class ViewController: UIViewController {
12
13     override func viewDidLoad() {
14         super.viewDidLoad()
15         var count = 1
16         print("Hello world first")
17         print("Hello world second")
18         count += 1
19         a()
20         print("count : \(count)")
21         count += 1
22         print("Hello world third")
23         print("count : \(count)")
24     }
25 }
```

Thread 1: bre...


HelloWorld > Thread 1 > 0 ViewController.viewDidLoad()

(11db)

I

Execution Commands

* 특정 라인/주소으로 이동함.

(lldb) thread jump --line 10

(lldb) thread jump --by 5

(lldb) thread jump --by -5

```
13      override func viewDidLoad() {
14          super.viewDidLoad()
15          var count = 1
16          print("Hello world first")
17          print("Hello world second")
18          count += 1
19          a()
20          print("count : \(count)")
21          count += 1
22          print("Hello world third")
23          print("count : \(count)")
24      }
25
```

Thread 1: bre...

0 ViewController.viewDidLoad()

(lldb)

Execution Commands

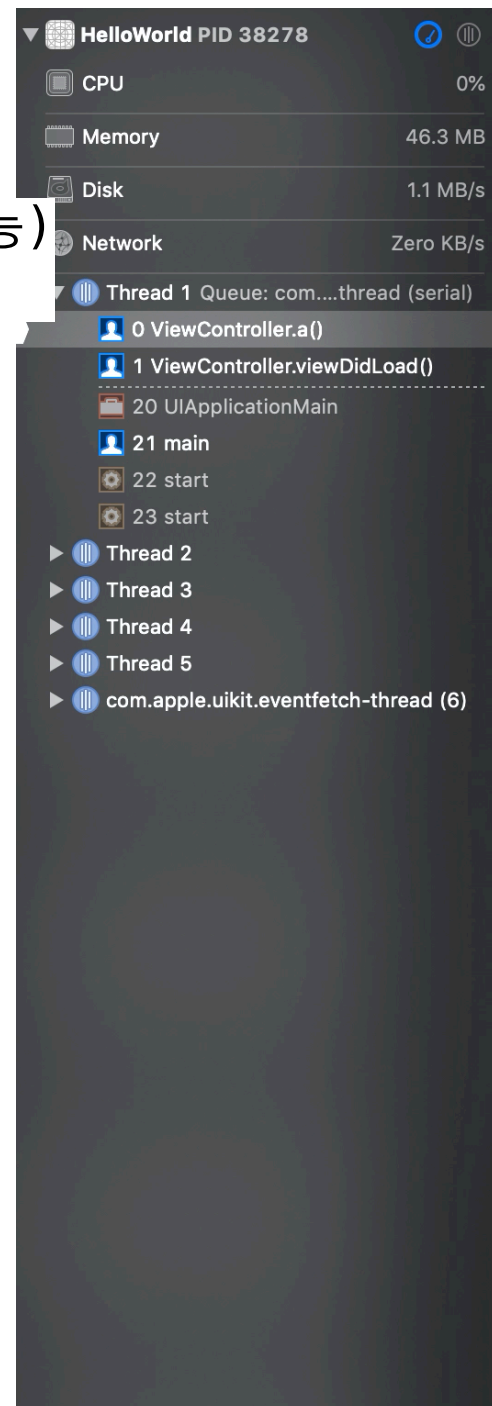
* 현재 frame에서 특정 값을 반환

(단, Swift는 Int, Void만 반환 가능)

(lldb) thread return

(lldb) thread return 0

(lldb) thread return "aa"



```
13 class ViewController: UIViewController {
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17         var count = 1
18         print("Hello world first")
19         print("Hello world second")
20         count += 1
21         a()
22         print("count : \(count)")
23         count += 1
24         print("Hello world third")
25         print("count : \(count)")
26     }
27
28     func a() {
29         print("Hello world Alpha")
30     }
31 }
32
33 class SecondViewController: UIViewController {}
```

Thread 1: break...

Hello world first
Hello world second
(lldb) |

LLDB 명령어

(Evaluating Expression)

Evaluating Expression

변수 선언하기

```
(lldb) settings set target.language swift
```

```
(lldb) expr var $foo = 10
```

```
(lldb) expr $foo // Output: 10
```

```
(lldb) expr $foo += 1
```

```
(lldb) expr $foo // Output: 11
```

Evaluating Expression

표현식 계산하기

```
(lldb) expression 1 + 2
```

```
(lldb) expr 1 + 2
```

```
(lldb) e 1 + 2
```

```
(lldb) expression -- 1 + 2
```

```
(lldb) print 1 + 2
```

```
(lldb) p 1 + 2
```

```
(lldb) expression -Object-description -- 1 + 2
```

```
(lldb) expression -0 -- 1 + 2
```

```
(lldb) po 1 + 2
```

Evaluating Expression

Command	Alias for	Execute
po [expression]	expression -Object-description --	Code를 실행하고, debugDescription 또는 description을 출력하며 LLDB formatter를 사용하여 출력
p [expression]	expression --	Code를 실행하며 LLDB formatter를 사용하여 출력
fr v [name]	frame variable	Code를 실행하지 않으며, LLDB formatter를 사용하여 출력

Evaluating Expression

/// 일시 정지한 후, 특정 메모리 주소의 값을 출력하기

/// Swift

```
(lldb) expr -l swift -- import UIKit
```

```
(lldb) expr -l swift -- import HelloWorld
```

```
(lldb) expr -l swift -- let $vc =
```

```
    unsafeBitCast(0x7fbdb852ae70, to: ViewController.self)
```

```
(lldb) po $vc
```

```
(lldb) po $vc.view
```

/// 한번에 코드 입력하기

```
(lldb) expr -l Swift --
```

```
1 import UIKit
```

```
2 import HelloWorld
```

```
3 let $vc = unsafeBitCast(0x7fe75a70bb40, to:
```

```
    ViewController.self)
```

```
4 print($vc)
```

Evaluating Expression

```
/// 새로운 ViewController 만들어 Push 하기
(lldb) ex -l swift -- let $vc = unsafeBitCast(0x7fce5dd13010, to: UIViewController.self)
(lldb) ex -l swift -- let $newvc = UIViewController()
(lldb) ex -l swift -- $newvc.view.backgroundColor = UIColor.red
(lldb) ex -l swift -- $vc.navigationController?.pushViewController($newvc, animated: true)
```

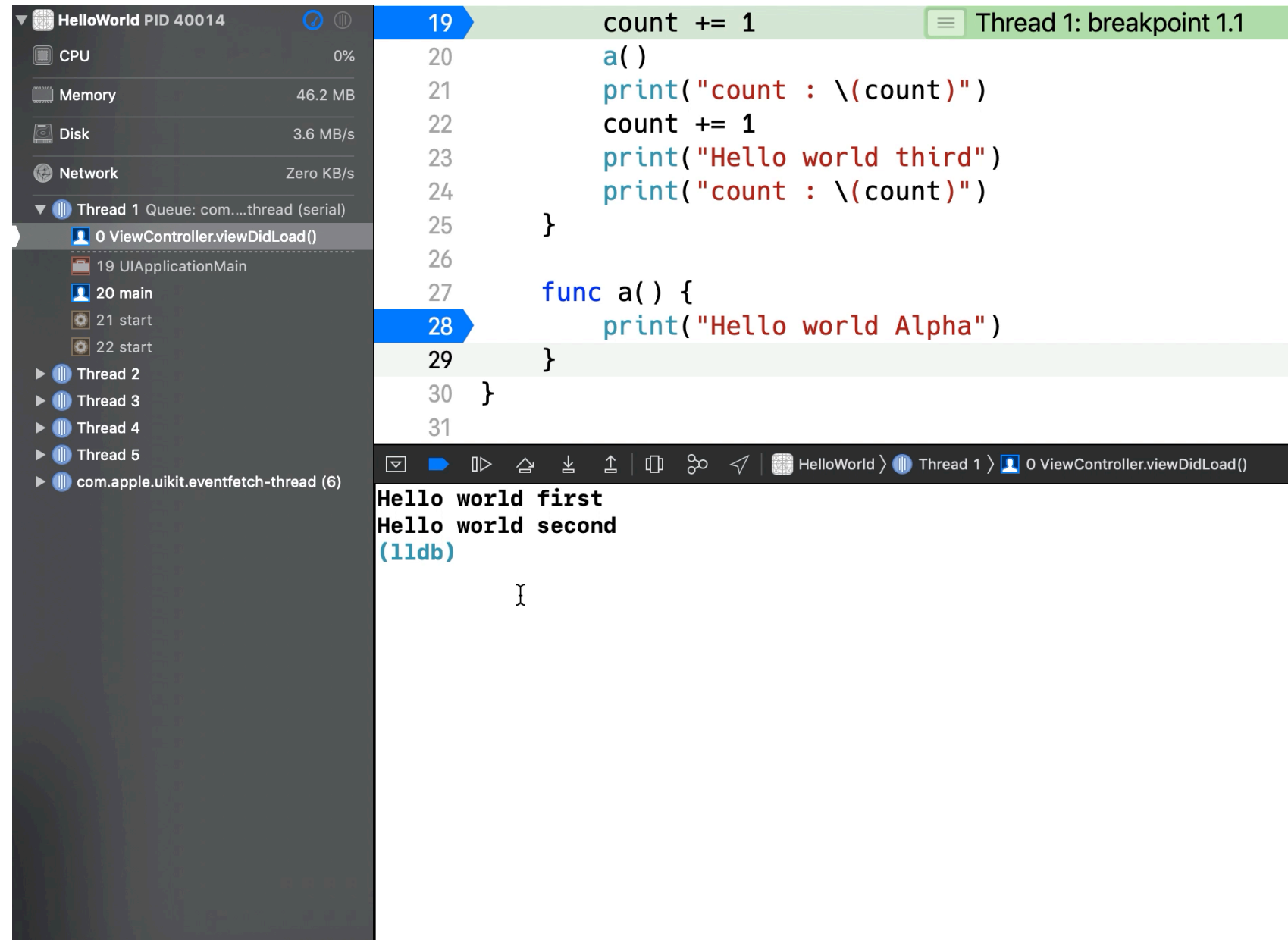
Evaluating Expression

* 중단된 상태에서 중단점이 걸린 코드를
실행했을 때, 중단점이 걸리도록 하기

```
(lldb) expr --ignore-
```

```
breakpoints false -- a()
```

```
(lldb) ex -i false -- a()
```



The screenshot shows the Xcode IDE with a Swift file open. The file contains the following code:

```
19 count += 1
20 a()
21 print("count : \(count)")
22 count += 1
23 print("Hello world third")
24 print("count : \(count)")
25 }
26
27 func a() {
28     print("Hello world Alpha")
29 }
30 }
31
```

A breakpoint is set at line 19, labeled "Thread 1: breakpoint 1.1". The left sidebar shows the project structure with "Thread 1 Queue: com....thread (serial)" expanded, showing the call stack:

- 0 ViewController.viewDidLoad()
- 19 UIApplicationMain
- 20 main
- 21 start
- 22 start

The bottom console shows the output of the program:

```
Hello world first
Hello world second
(lldb)
```

LLDB 명령어 (BreakPoint)

BreakPoint

/// viewDidLoad 이름인 모든 함수에 breakpoint를 설정하기 - Swift

```
(lldb) breakpoint set --name viewDidLoad
```

```
(lldb) br s -n viewDidLoad
```

```
(lldb) b viewDidLoad
```

/// viewDidLoad 이름인 모든 함수에 breakpoint를 설정하기 - Objc

```
(lldb) breakpoint set --name "-[UIViewController viewDidLoad]"
```

/// 특정 파일 특정 줄에 breakpoint 설정하기

```
(lldb) breakpoint set --file ViewController.swift --line 28
```

```
(lldb) br s -f ViewController.swift -l 28
```

```
(lldb) b ViewController.swift:28
```

/// 현재 파일의 특정 줄에 breakpoint 설정하기

```
(lldb) breakpoint set --line 28
```

```
(lldb) br s -l 28
```

```
(lldb) b 28
```


BreakPoint

/// 특정 이름을 가진 Select에 breakpoint 설정하기

```
(lldb) breakpoint set --selector dealloc
```

/// breakpoint에 count값이 2보다 크면 중단되도록 조건 설정

```
(lldb) b 23
```

```
(lldb) br modify --condition "count > 2"
```

```
(lldb) br modify -c "count > 2"
```

/// 2번까지는 무시하고, 3번째부터는 중단점이 걸리도록 설정

```
(lldb) b 24
```

```
(lldb) br modify --ignore-count 2
```

```
(lldb) br modify -i 2
```

/// breakpoint 목록 보기

```
(lldb) breakpoint list
```

```
(lldb) br l
```

/// breakpoint 지우기

```
(lldb) breakpoint delete 1
```

```
(lldb) br del 1
```

LLDB 명령어 (Disassemble)

Disassemble

/// 현재 스레드와 stack frame에 현재 함수를 disassemble하여 보여줌.

```
(lldb) disassemble
```

```
(lldb) di
```

/// main 이라는 함수를 disassemble하여 보여줌.

```
(lldb) disassemble --name main
```

```
(lldb) di -n main
```

/// 지정된 주소범위의 명령어 코드를 출력함.

```
(lldb) disassemble --start-address 0x10363f38f --end-address 0x10363f3cd
```

```
(lldb) di -s 0x10363f38f -e 0x10363f3cd
```

/// 현재 스레드와 stack frame에 현재 함수 코드에 해당하는 명령어를 코드와 같이 출력함.

```
(lldb) disassemble --mixed
```

```
(lldb) di -m
```

/// 현재 스레드와 stack frame에 현재 중단된 지점의 코드를 disassemble하여 보여줌.

```
(lldb) disassemble --line
```

```
(lldb) di -l
```

LLDB 명령어

(Script - Python REPL)

Evaluating Expression

/// Python을 LLDB Script로 사용할 수 있음.

```
(lldb) script print 1 + 2 // Output: 3
```

```
(lldb) script import os
```

```
(lldb) script print os.getcwd()
```

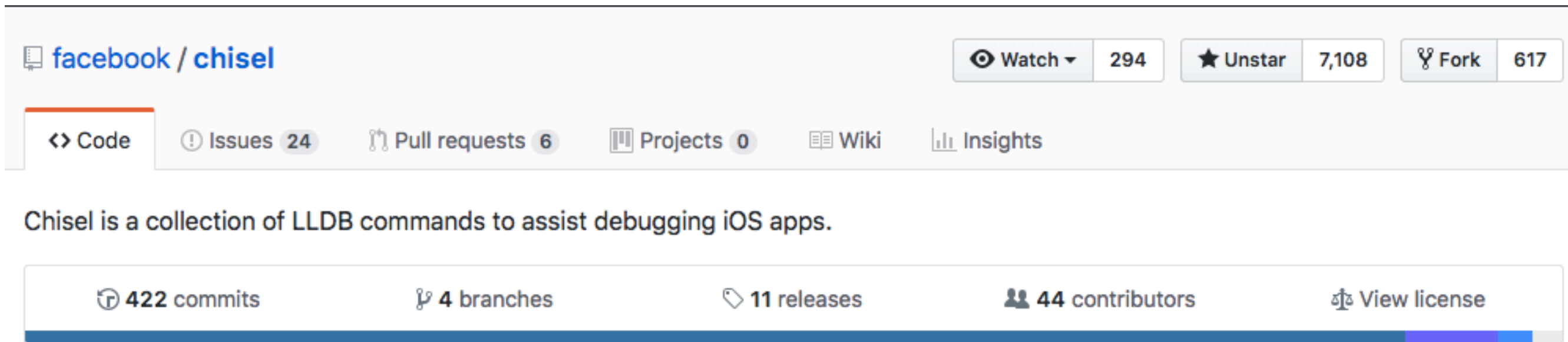
/// import - 필요한 script 소스를 import하여 사용함.

```
(lldb) command script import ~/myCommands.py
```

```
$ echo 'command script import ~/myCommands.py' >>  
  ~/.lldbinit
```

Chisel

Chisel 소개



The screenshot shows the GitHub repository page for 'facebook/chisel'. At the top, the repository name 'facebook / chisel' is displayed. To the right, there are buttons for 'Watch' (294), 'Unstar' (7,108), and 'Fork' (617). Below these, a navigation bar includes links for 'Code', 'Issues' (24), 'Pull requests' (6), 'Projects' (0), 'Wiki', and 'Insights'. A description states: 'Chisel is a collection of LLDB commands to assist debugging iOS apps.' At the bottom, statistics are listed: '422 commits', '4 branches', '11 releases', '44 contributors', and a 'View license' link.

facebook / chisel

Watch 294 Unstar 7,108 Fork 617

Code Issues 24 Pull requests 6 Projects 0 Wiki Insights

Chisel is a collection of LLDB commands to assist debugging iOS apps.

422 commits 4 branches 11 releases 44 contributors View license

- Facebook에서 만듦(<https://github.com/facebook/chisel>)
- 파이썬으로 작성되어 있음.
- iOS 디버깅을 위한 LLDB 명령어 모음

Chisel 소개 - Commands

- Print Commands
- Find Commands
- Visualize Command
- Display Commands
- Autolayout Commands
- Flicker Commands
- Accessibility Commands

Chisel 명령어 (Print Commands)

Print Commands

```
/// UIWindow에 표시되는 모든 UIView를 출력하는 명령어  
(lldb) pviews
```

```
/// UIWindow에 표시하는 모든 UIViewController를 출력하는 명령어  
(lldb) pvc
```

```
/// 현재 화면에 나타난 최상위의 UITableView를 출력하는 명령어  
(lldb) ptv
```

```
/// 현재 화면에 나타난 최상위의 UITableView에 visible cell을 출력하는  
명령어  
(lldb) pcells
```

Print Commands

/// 해당 인스턴스의 상속 계층 구조를 보여주는 명령어

```
(lldb) pclass 0x7f9950708cd0
```

/// 객체 내부를 보여주는 명령어

```
(lldb) pinternals 0x7fada0624840
```

/// KeyPath를 이용하여 값을 출력하는 명령어

```
(lldb) pvc
```

```
<UITableViewController 0x7fada0406ae0>, state: appeared,  
    view: <UITableView 0x7fada085dc00>
```

```
(lldb) pkp 0x7fada0406ae0 .view.backgroundColor
```

```
UIExtendedSRGBColorSpace 1 1 1 1
```

```
(lldb) pkp 0x7fada0406ae0 .view.isHidden
```

```
0
```

Chisel 명령어 (Find Commands)

Find Commands

/// 정규식을 이용하여 특정 UIViewController 클래스를 찾아 출력하는 명령어

```
(lldb) fvc ViewController
```

/// 현재 표시되는 화면에서 정규식을 사용하여 특정 UIView 클래스를 찾아 출력하는 명령어

```
(lldb) fv TableView
```

Chisel 명령어 (Visualize Commands)

Visualize Commands

/// UIImage, CGImageRef, UIView, CALayer를 이미지로 만들어
Preview로 열어 보여주는 명령어

```
(lldb) visualize self.view
```

```
(lldb) pviews
```

```
(lldb) visualize 0x7f8527501000
```

Chisel 명령어 (Display Commands)

Display Commands

```
/// 해당 View 또는 Layer에 border를 설정하거나 끄는 명령어  
(lldb) pviews  
(lldb) border 0x7f80f2d11b40 --color blue --width 10  
(lldb) unborder 0x7f80f2d11b40
```

```
/// 사각형을 View 또는 Layer 위에 노출시키거나 끄는 명령어  
(lldb) pviews  
(lldb) mask 0x7fdfeeee04200 --color red  
(lldb) unmask 0x7fdfeeee04200
```

```
/// 특정 UIView나 CALayer를 숨기거나 보여주는 명령어  
(lldb) pviews  
(lldb) hide 0x7f80f2c01f20  
(lldb) show 0x7f80f2c01f20
```

Display Commands

```
/// 애니메이션을 느리게 또는 빠르게 하는 명령어
/// 애니메이션 속도를 느리게 하거나 빠르게 함.
(lldb) slowanim [속도, 기본값 1]
```

```
// 변경된 속도를 원상복구 함.
(lldb) unslowanim
```

```
/// 특정 UIViewController를 present하거나 dismiss 하는 명령어
(lldb) pvc
(lldb) expr -l objc -- UIViewController *$vc =
    (UIViewController *)0x7f80f2e16640
(lldb) dismiss $vc
```

```
/// 즉각적으로 화면을 다시 그리도록 하는 명령어
(lldb) caflush
```

Chisel 명령어 (Autolayout Commands)

Autolayout Commands

/// SubView들의 Hierarchy를 출력하는 명령어, 기본은 Key Window로부터 시작함.

/// 만약 Autolayout 에러가 발생하는 경우, AMBIGUOUS LAYOUT라고 표시됨.

```
(lldb) paltrace
```

```
(lldb) paltrace 0x7fada085dc00
```

/// Ambiguous Layouts View들만 border를 설정하거나 끄는 명령어

```
(lldb) paltrace
```

```
(lldb) alamborder
```

```
(lldb) alamunborder
```

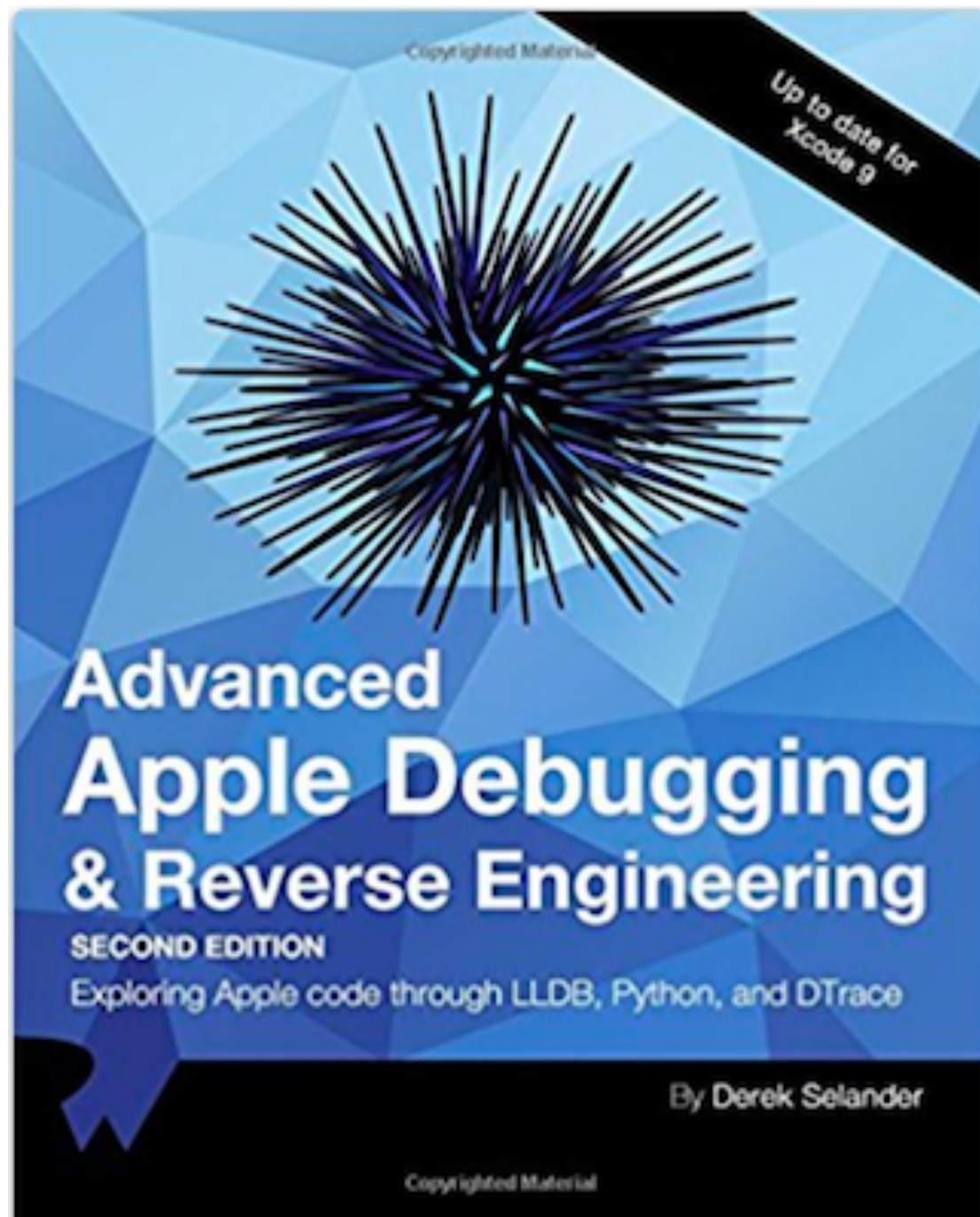
Chisel 명령어 (Accessibility Commands)

Accessibility Commands

/// 접근성이 설정되어 있는 모든 View를 출력하는 명령어
(lldb) pa11y

/// 모든 View의 접근성 식별자를 출력하는 명령어
(lldb) pa11yi

추천도서



**Advanced Apple Debugging
and Reverse Engineering**

from. Raywenderlich

QnA

Reference

[Apple - LLDB Quick Start Guide](https://developer.apple.com/library/archive/documentation/IDEs/Conceptual/gdb_to_lldb_transition_guide/document/Introduction.html#//apple_ref/doc/uid/TP40012917-CH1-SW1)

[UIKonf18 – Day 1 – Carola Nitz – Advanced Debugging Techniques](<https://www.youtube.com/watch?v=578YdS2sNqk>)

[Advanced Debugging with Xcode and LLDB](<https://developer.apple.com/videos/play/wwdc2018/412>)

[Chisel](<https://github.com/facebook/chisel>)

[LLDB Chisel Commands](https://kapeli.com/cheat_sheets/LLDB_Chisel_Commands.docset/Contents/Resources/Documents/index)

[More than `po`: Debugging in lldb](<https://www.slideshare.net/micheletitolo/more-than-po-debugging-in-lldb>)

[Debugging RubyMotion applications](<http://ruby-korea.github.io/RubyMotionDocumentation/articles/debugging/>)

[Xcode LLDB 디버깅 테크닉](<https://www.letmecompile.com/xcode-lldb-%EB%94%94%EB%B2%84%EA%B9%85-%ED%85%8C%ED%81%AC%EB%8B%89/>)

[LLDB Debugging Cheat Sheet](<https://gist.github.com/alanzeino/82713016fd6229ea43a8>)

[Debugging a Debugger](<http://idrisr.com/2015/10/12/debugging-a-debugger.html>)

https://kapeli.com/cheat_sheets/LLDB_Chisel_Commands.docset/Contents/Resources/Documents/index

<http://ios.137422.xyz/83589/>

<https://www.slideshare.net/YiyingTseng/debug-lldb-86558535>

<https://medium.com/flawless-app-stories/debugging-swift-code-with-lldb-b30c5cf2fd49>

<https://dailyhotel.io/advanced-debugging-with-xcode-and-lldb-8e8dc5326167>

<https://pspdfkit.com/blog/2018/how-to-extend-lldb-to-provide-a-better-debugging-experience/>



Let's Swift 2018