作者 小笨狼 (/users/1f93e3b1f3da) 2015.12.26 16:28*
写了35945字，被797人关注，获得了367个喜欢
(/users

小笨狼的LLDB技巧:chisel

不玩LLDB，不知道chisel有多强大。chisel之于LLDB，就像iPhone之于手机，前者几乎给后者重新下了一次定义。如果你还不知道什么是LLDB，请看我上一篇文章小笨狼与LLDB的故事 (http://www.jianshu.com/p/e89af3e9a8d7)

安装

安装Homebrew

chisel的安装需要使用Homebrew，如果还没有安装Homebrew，可以使用下面的命令安装，如果你已经安装了，可以跳过这一步

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/inst
```

用Homebrew安装chisel

搞定Homebrew之后，你就可以用他来安装chisel了

```
brew update
brew install chisel
```

官网的命令里有一个 `brew update` ，是用来更新Homebrew版本的，如果你是新安装的Homebrew，可以省略掉这条命令

在 `~/.lldbinit` 中添加命令

安装好之后，terminal中会出现这个东西

```
==> Caveats
Add the following line to ~/.lldbinit to load chisel when Xcode launches:
  command script import /usr/local/opt/chisel/libexec/fblldb.py
```

意思是需要把这个命令 `command script import /usr/local/opt/chisel/libexec/fblldb.py` 加到 `~/.lldbinit` 文件中

我们将命令其添加进去即可。如果你已经添加过一次了，不需要再次添加

```
touch ~/.lldbinit
echo "command script import /usr/local/opt/chisel/libexec/fblldb.py" >> ~/.lldbinit
```

重启一下Xcode，安装完成

更新

如果你想更新chisel，只需要输入更新的命令即可

```
brew upgrade chisel
```

命令

Autolayout

autolayout 中 有 一 种 bug 叫 Ambiguous Layouts (https://developer.apple.com/library/tvos/documentation/UserExperience/Conceptual/AutolayoutPG/AmbiguousLayouts.html#//apple_ref/doc/uid/TP40010853-CH18-SW1)， 意思是你设置的约束不足以确定view的位置或大小。比如你只设置了X轴的位置，没有设置Y轴的位置

autolayout提供了专门判断和查找这类问题的方法:

```
hasAmbiguousLayout. Available for both iOS and OS X. Call this method on a misplaced
_autolayoutTrace. Available as a private method in iOS. Call this method on a view.
```

- `hasAmbiguousLayout` 用于判断是否存在 `Ambiguous Layouts`
- `_autolayoutTrace` 用于查找存在的 `Ambiguous Layouts`

但是即使有查找的方法，真正去做这个事儿也比较费时费力的，这时候chisel给我们提供了更为方便的命令

alamborder

给存在 `Ambiguous Layouts` 的view加上border，方便查找哪些View存在问题

```
Syntax: alamborder [--color=color] [--width=width]
```

- `--color` / `-c`：border的颜色，参数为string类型，比如'red'，'green'，'magenta'等，不设置默认为红色。
- `--width` / `-w`：border的宽度，参数为CGFloat类型，不设置默认宽度为2。

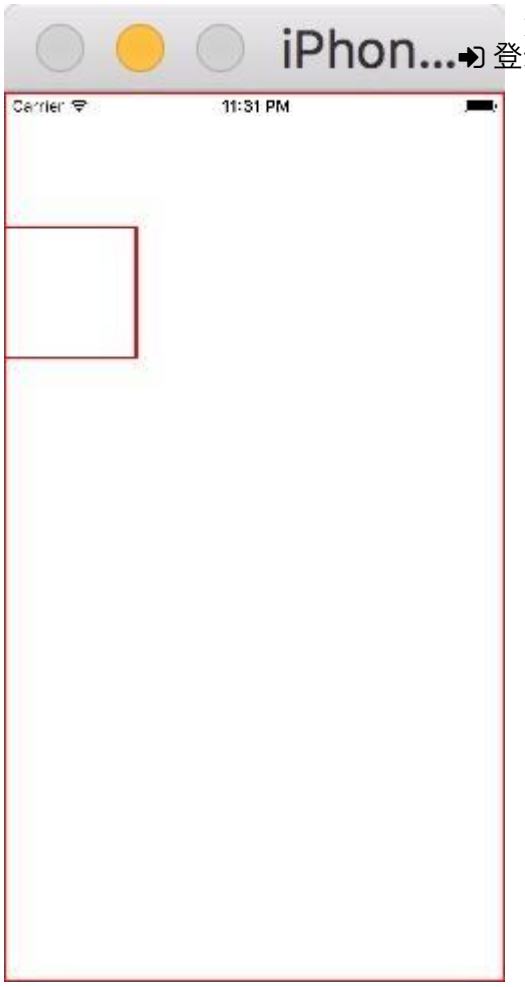
e.g: 假设我们写了这么一段代码，可以明显看出，我们没有设置X轴的位置。

```
UIView *subview = [UIView new];
[self.view addSubview:subview];
[subview mas_makeConstraints:^(MASConstraintMaker *make) {
    make.top.offset(100);
    make.size.equalTo(@100);
}];
```

运行代码之后，在LLDB控制台输入 `alamborder`

```
(lldb) alamborder
```

所有带有 `Ambiguous Layouts` 的view立即会被渲染上红色border



可以看到，subview的边框已经变为红色，另外只要有一个View存在 `Ambiguous Layouts`，UIWindow的边框也会变为红色，这就有效的避免了宽度或者高度为0的 `Ambiguous Layouts` 不宜察觉的缺陷。

alamunborder

将 `alamborder` 设置的border去掉.

语法:

```
Syntax: alamunborder
```

e.g: 刚刚设置的border，在lldb控制台输入 `alamunborder` 即可去掉边框

```
(lldb) alamborder
```

paltrace

打印某个View的autolayout详细信息，相当于调用 `_autolayoutTrace`

语法:

```
Syntax: paltrace <view>
```

- `<view>`：需要打印详细信息的view，不传参数默认为keyWindow

e.g: 查看一下keyWindow上有哪个view存在 `Ambiguous Layouts`

```
(lldb) paltrace
•UIWindow:0x7ff450d2fb50 - AMBIGUOUS LAYOUT
|   •UIView:0x7ff450e14430
|   |   *_UILayoutGuide:0x7ff450d30e90
|   |   *_UILayoutGuide:0x7ff450d31230
|   |   *UIView:0x7ff450d32870- AMBIGUOUS LAYOUT for UIView:0x7ff450d32870.minX{id: 

Legend:
    * - is laid out with auto layout
    + - is laid out manually, but is represented in the layout engine because transl
    • - layout engine host
```

我们可以看到， `UIView:0x7ff450d32870` 存在 `Ambiguous Layouts` ， 原因是缺少 `minX` 。也就是没有设置X轴的位置

Print

在LLDB中，我们执行的最多的可能就是打印操作了，chisel专门为这类操作封装了一些打印命令

pviews

循环打印view层级，正常情况下等效于调用 `recursiveDescription` 命令

```
// 下面2条命令等效
(lldb) po [self.view recursiveDescription]
(lldb) pviews self.view
```

语法:

```
pviews [--up] [--depth=depth] <aView>
```

- `--up` / `-u`：以view为起始位置，向上打印，直到打印到window层
- `--depth` / `-d`：传入int类型，表示打印的层数，0表示没有限制

e.g: 打印一下self.view层级

```
(lldb) pviews self.view
<UIView: 0x7fee7ae1fa60; frame = (0 0; 375 667); autoresize = W+H; layer = <CALayer:
   | <UIButton: 0x7fee7ae1dd90; frame = (54 244; 46 30); opaque = NO; autoresize = RM
   | <UIView: 0x7fee7ae1f2e0; frame = (35 312; 240 128); autoresize = RM+BM; layer =
   | <_UILayoutGuide: 0x7fee7ae1fc20; frame = (0 0; 0 0); hidden = YES; layer = <CALa
   | <_UILayoutGuide: 0x7fee7ae20b30; frame = (0 0; 0 0); hidden = YES; layer = <CALa
   | <UILabel: 0x7fee7ae1d3f0; frame = (0.4 150; 58.25 20.5); text = 'aaa'; userInte
```

pvc

循环打印viewController的层级

语法:

```
Syntax: pvc <aViewController>
```

`<aViewController>`：表示要打印的viewController，不传参数默认viewController为当前的VC

e.g: 打印一下当前VC

```
(lldb) pvc
<UINavigationController 0x7fce1a03d800>, state: appeared, view: <UILayoutContainerVi
   | <ViewController 0x7fce1c03a1d0>, state: disappeared, view: <UIView 0x7fce19517b
   | <TableViewController 0x7fce1951f7b0>, state: appeared, view: <UIView 0x7fce1c05
```

pclass

循环打印class的继承关系

语法:

```
Syntax: pclass <object>
```

`<object>`：要打印继承关系的对象

e.g: 打印一个View对象的继承关系

```
(lldb) pclass [UIView new]
UIView
   | UIResponder
   |    | NSObject
```

presponder

打印响应链

语法:

```
Syntax: presponder <startResponder>
```

`<startResponder>`：UIResponder对象，响应链开始位置

e.g: 打印一个tableView的响应链

```
(lldb) presponder tableView
<UITableView: 0x7fde54810e00; frame = (0 0; 0 0); clipsToBounds = YES; gestureRecogn
   | <UIView: 0x7fde5255c710; frame = (0 0; 600 600); autoresize = W+H; layer = <CAL
   |    | <TableViewController: 0x7fde52491310>
```

ptv

打印屏幕中显示的tableView，主要是与 `pcells` 联合使用。如果有多个tableView，打印View层级中最上面的一个

语法:

```
Syntax: ptv
```

e.g: 看看当前最上面是哪个tableView

```
(lldb) ptv
<UITableView: 0x7fde52811800; frame = (0 0; 414 736); clipsToBounds = YES; autoresiz
```

pcells

打印tableView中当前可见的cell，如果有多个tableView，打印View层级中最上面的tableView的可见cell

语法:

```
Syntax: pcells
```

e.g: 看看当前可见的cell有哪些

```
(lldb) pcells
<__NSArrayI 0x7fde52565a00>(
<UITableViewCell: 0x7fde52551180; frame = (0 0; 414 44); text = 'BasicViewController
<UITableViewCell: 0x7fde5255bea0; frame = (0 44; 414 44); text = 'DateViewController
<UITableViewCell: 0x7fde5255e2d0; frame = (0 88; 414 44); text = 'PPTViewController'
<UITableViewCell: 0x7fde5255fce0; frame = (0 132; 414 44); text = 'TableViewControll
)
```

pinternals

打印一个对象内部的成员变量，这个方法我一般用来看model属性

语法:

```
Syntax: pinternals <object>
```

- `<object>`：需要打印内部成员变量的对象

e.g: 我们来看看一个 `model` 内部属性的值

```
(lldb) pinternals model
(Model) $5 = {
  _name = 0x000000010dd1c0a0 @"老鼠爱大米"
  _URL = nil
  _array = nil
  _dictionary = nil
  _string = nil
  _model = nil
}
```

pdata

对编码过的NSData进行解码打印，等效于调用 `-[NSString initWithData:encoding:]`

语法:

```
Syntax: pdata [--encoding=encoding] <data>
```

- `<data>`：需要打印的data，NSData类型
- `--encoding` / `-e`：编码类型，如果缺省默认为utf8，主要支持的类型有:

```
- ascii,
- utf8,
- utf16, unicode,
- utf16l (Little endian),
- utf16b (Big endian),
- utf32,
- utf32l (Little endian),
- utf32b (Big endian),
- latin1, iso88591 (88591),
- latin2, iso88592 (88592),
- cp1251 (1251),
- cp1252 (1252),
- cp1253 (1253),
- cp1254 (1254),
- cp1250 (1250),
```

e.g: 将一个utf8的NSData打印

```
(lldb) pdata -e=utf8 data
老鼠爱大米
```

pkp

通过 `-valueForKeyPath:` 打印 `key path` 对应的值。

语法:

```
Syntax: pkp <keypath>
```

- `<keypath>`：需要打印的路径，如 `self.view`

说明：以前打印属性一般都用 `po obj.xxx`，现在我想用 `pkp obj.xxx` 是一个更好的选择了，因为 `po obj.xxx` 是调用getter方法，如果没有getter方法就无法打印了。`pkp obj.xxx` 不仅会调用getter方法，没有getter方法还会去查找成员变量

e.g: 打印一下self.view

```
(lldb) pkp self.view
<UIView: 0x7fd1da52d5d0; frame = (0 0; 375 667); autoresize = W+H; layer = <CALayer:
```

pivar

打印对象成员变量

```
Syntax: pivar <object> <ivarName>
```

- `<object>`：id类型，要打印成员变量的对象
- `<ivarName>`：成员变量的名称，注意：如果是属性，对应成员变量的名字默认有 _ 前缀.

说明：个人觉得这个方法有点鸡肋， `pinternals` 一下子可以打印出所有的成员变量，用起来更方便，如果你只想打印某一个成员变量，用 `pkp` 应该更爽

e.g: 继续打印self.view

```
(lldb) pivar self _view
<UIView: 0x7fd1da52d5d0; frame = (0 0; 375 667); autoresize = W+H; layer = <CALayer:
```

pca

从渲染服务器的角度来打印 `layer tree` ， 命令的完整名字是 `PrintCoreAnimationTree` ， 相当于调用 `po [NSString stringWithCString:(char *)CARenderServerGetInfo(0, 2, 0)]`

语法:

```
Syntax: pca
```

说明：这个命令我也没用过，可能在某些情况下会有用处

e.g: 试一下pca

```
(lldb) pca
== context 956ecbbf; level 0; pid 28092 [/Applications/Xcode.app/Contents/Developer/
(layer [375 667 0] [0 0 375 667] [0.5 0.5 0]
  (transform [2 -0 0 0; 0 2 0 0; 0 0 1 0; 0 0 0 1])
 (rasterizationScale 2)
  (sublayers (array
    (layer [187.5 333.5 0] [0 0 375 667] [0.5 0.5 0]
      (masksToBounds true)
      (sublayers (array
        (layer [187.5 333.5 0] [0 0 375 667] [0.5 0.5 0]
          (backgroundColor #000000ff)
          (sublayers (array
            (layer [187.5 333.5 0] [0 0 375 667] [0.5 0.5 0]
              (sublayers (array
                (layer [187.5 333.5 0] [0 0 375 667] [0.5 0.5 0]
                  (sublayers (array
                    (layer [187.5 333.5 0] [0 0 375 667] [0.5 0.5 0]
                      (sublayers (array
                        (layer [187.5 333.5 0] [0 0 375 667] [0.5 0.5 0]
                          (masksToBounds true)
                          (edgeAntialiasingMask 0)
                          (subclass (layer-host adc504e5
                            ....东西太多，省略部分内容.........
```

panim

显示是否正在执行动画，相当于调用 `p (BOOL)[UIView _isInAnimationBlock]`

语法:

```
Syntax: panim
```

说明：这个命令也并不常用

e.g: 在动画中，我们打印一下:

```
(lldb) panim
(BOOL) $0 = YES
```

Find

debug的时候，我们经常需要查找一些东西，比如View，viewController等。

fvc

语法：

```
Syntax: fvc [--name=classNameRegex] [--view=view]
```

- `--name` / `-n`：string类型参数，根据viewController的Class名字查找viewController
- `--view` / `-v`：UIView类型参数，根据viewController拥有的view查找viewController

说明：上面2个option不能同时使用，只能使用某一个

e.g: 我们先根据名字查找一下VC

```
(lldb) fvc --name=viewcontroller
0x7fd01a90f310 ViewController
```

e.g: 如果我们知道VC的view地址，也可以根据view来查找VC

```
(lldb) fvc --view=0x7fd0194194d0
Found the owning view controller.
<ViewController: 0x7fd01a90f310>
```

fv

根据view的class名字查找view

语法：

```
Syntax: fv <classNameRegex>
```

- `<classNameRegex>`：view的class名称

e.g: 查找一下屏幕上的UILabel

```
(lldb) fv uilabel
0x7fd01a91dc10 UILabel
```

taplog

将点击的view打印出来，这个命令对于查找哪个view非常有帮助

```
Syntax: taplog
```

说明：要查看的view必须能接收点击事件，也就是他的 `userInteractionEnabled` 必须为YES才能被找到，UILabel和UIImageView默认 `userInteractionEnabled` 为NO。

用法：我们需要先将程序暂停，输入 `taplog` ，程序会自己运行，这时候点击你需要查看的view，控制台上就会显示出你刚刚点击的view相关信息

e.g: 我们先将程序暂停，输入 `taplog`

```
(lldb) taplog
Process 28421 resuming
```

程序会自己运行，我们再点击一个UIButton：

```
<UIButton: 0x7fe6785284e0; frame = (54 244; 46 30); opaque = NO; autoresize = RM+BM;
```

flicker

将view闪烁一下，以便于查找view的位置

语法：

```
Syntax: flicker <viewOrLayer>
```

- `<viewOrLayer>` 需要闪烁的view或者layer

e.g: 我们来看看self.subView的位置

```
(lldb) flicker self.subView
```

在view层级中搜索view，并显示出来

语法：

```
Syntax: vs <view>
```

- `<view>` :要查找的view

说明：相比 `fv` ， `vs` 主要用于显示view在屏幕上的位置，2个命令可以配合使用

e.g: 假设我们要找屏幕上的一个view

首先用 `fv` 查找 `UIView` 类型的view

```
(lldb) fv uiview
0x7fbcf37228d0 UIView
0x7fbcf3725e90 UIView
```

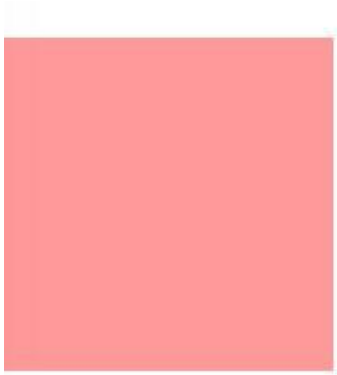
然后看看这2个view到底哪个是我们想要找的view

```
(lldb) vs 0x7fbcf3725e90

Use the following and (q) to quit.
(w) move to superview
(s) move to first subview
(a) move to previous sibling
(d) move to next sibling
(p) print the hierarchy

<UIView: 0x7fbcf3725e90; frame = (0 100; 100 100); layer = <CALayer: 0x7fbcf3712a40>
```

输入命令后他会帮我们在屏幕上用粉红色标志出来 `vs` 的view



- `w`：移动到superview
- `s`：移动到第一个subview
- `a`：移动到前面的同级view
- `d`：移动到后面的同级view
- `p`：打印出层级
- `q`：退出

如果这个不是我们要找的view，可以使用 `w`，`s`，`a`，`d`，`p` 命令继续查找

Display

debug的时候，可能有一小半的工作是跟UI打交道，关于UI显示上的东西，也有几个命令

caflush

刷新UI界面。一般我们用LLDB命令改变UI，UI并不会立即更新，我们需要使用 `caflush` 刷新界面

语法:

```
Syntax: caflush
```

e.g: 我们用命令将label的背景色改为红色

```
(lldb) fv uilabel
0x7fb3919189d0 UILabel
(lldb) e [((UILabel*)0x7fb3919189d0) setBackgroundColor:[UIColor redColor]]
(lldb) caflush
```

border

给View或者layer加上border

语法:

```
Syntax: border [--color=color] [--width=width] <viewOrLayer>
```

- `--color` / `-c`：边框颜色，string类型，比如:'red', 'green' 'magenta'等，不设置默认为红色
- `--width` / `-w`：边框宽度，不设置默认为2
- `<viewOrLayer>`：需要设置边框的view或者layer

e.g: 给刚刚的label加上边框

```
(lldb) fv uilabel
0x7fe713901f10 UILabel
(lldb) border 0x7fe713901f10
```

unborder

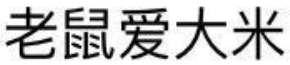
去掉view或者layer的border

语法:

```
Syntax: unborder <viewOrLayer>
```

e.g: 将刚刚加上的border去掉

```
(lldb) unborder 0x7fe713901f10
```



mask

给view添加一个半透明的矩形mask，用来查看view的位置

语法:

```
Syntax: mask [--color=color] [--alpha=alpha] <viewOrLayer>
```

- `--color` / `-c`：mask的颜色，string类型，比如:'red', 'green', 'magenta'等，不设置默认为红色
- `--alpha` / `-a`：mask的透明度，不设置默认为0.5
- `<viewOrLayer>`：需要添加mask的view或者layer

e.g: 假如label是隐藏的，我们给他添加一个mask，看看他的位置在哪儿

```
(lldb) fv uilabel
0x7fe713901f10 UILabel
(lldb) mask 0x7fe713901f10
```

unmask

将添加的mask去掉

语法:

```
Syntax: unmask <viewOrLayer>
```

- `<viewOrLayer>`：需要去掉mask的view或者layer

e.g: 我们将刚刚添加的mask去掉

```
(lldb) unmask 0x7fe713901f10
```

使用命令之后，我们可以看到什么都没有了，因为label是hidden的

show

显示一个view或者layer，相当于执行 `view.hidden = NO`

语法:

```
Syntax: show <viewOrLayer>
```

- `<viewOrLayer>`：需要显示的view或者layer

e.g: 将刚刚的label显示出来

```
(lldb) show 0x7fe713901f10
```

老鼠爱大米

hide

隐藏一个view或者layer，相当于执行 `view.hidden = YES`

语法:

```
Syntax: hide <viewOrLayer>
```

- `<viewOrLayer>`：需要隐藏的view或者layer

e.g: 又把label隐藏

```
(lldb) hide 0x7fe713901f10
```

可以看到label位置什么都没有了

slowanim

减慢动画的速度

语法:

```
Syntax: slowanim <speed>
```

- `<speed>`：动画的速度，速度越大，动画越快。1表示原始速度。不传参数默认为0.1

e.g: 原始动画我们设置为1s

```
[UIView animateWithDuration:1 animations:^{
    self.subView.frame = frame;
}];
```

暂停程序，将动画放慢5倍

我们可以看到动画变慢了:

unslowanim

取消 `slowanim` 效果，将动画速度变为正常

语法:

```
Syntax: unslowanim
```

e.g: 我们将刚刚的 `slowanim` 效果取消

```
(lldb) unslowanim
```

Preview

预览功能，帮助我们用命令查看一个view或者图片的真正样子

visualize

用预览App打开UIImage, CGImageRef, UIView, CALayer等对象

语法:

```
Syntax: visualize <target>
```

- `<target>`：需要预览的对象，id类型

e.g: 我们来看看某个image的样子

Debug

LLDB主要用于Debug，chisel怎么可能缺少debug相关命令呢?

wivar

为对象的成员变量设置 `watchpoint` ， 更多 `watchpoint` 相关知识请阅读<小笨狼与LLDB的故事> (http://www.jianshu.com/p/e89af3e9a8d7)

语法:

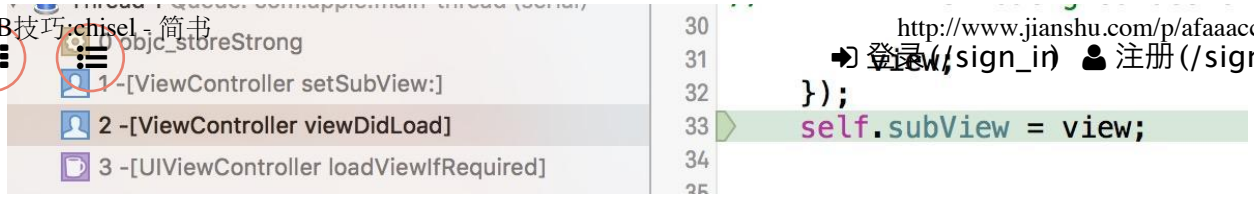
```
Syntax: wivar <object> <ivarName>
```

- `<object>`：需要为成员变量设置 `watchpoint` 的对象。id类型
- `<ivarName>`：成员变量的名字，注意一般属性对应的成员变量带有 _ 前缀

e.g: 为self.subView设置watchpoint

```
(lldb) wivar self _subView
Remember to delete the watchpoint using: watchpoint delete 1
```

这时候，_subView值改变就会中断程序



bmessage

根据方法名设置断点

语法:

```
Syntax: bmessage <expression>
```

- `<expression>`：设置断点的方法名，如： `-[MyView setFrame:]`，`+[MyView awesomeClassMethod]`，`-[0xabcd1234 setFrame:]` 等

说明：一般设置断点，如果这个方法本类没有实现，是父类实现的，断点是无效的。`bmessage` 有效避免了这种缺陷，即使本类没有实现，也能设置上断点

e.g: 给self中的 `btnAction:` 方法设置一个断点

```
(lldb) bmessage -[self btnAction:]
Setting a breakpoint at -[ViewController btnAction:] with condition (void*)(id)$rdi =
Breakpoint 4: where = TLLDB`-[ViewController btnAction:] at ViewController.m:42, add
```

binside

通过一个相对地址，给framework(library)设置断点

语法:

```
Syntax: binside <address>
```

- `<address>`：framework的相对地址

pinvocation

打印方法调用堆栈，仅支持x86

语法:

```
Syntax: pinvocation [--all]
```

- `bt /a`: 表示打印所有堆栈，不设置默认只打印当前堆栈

说明：与 `bt` 命令类似，不过信息比 `bt` 打印得更详细，遗憾的是只能支持x86

- `pinvocation`: 打印一下当前堆栈

```
(lldb) pinvocation
frame #0: 0x000962aa TMasonry`-[ViewController viewDidLoad](self=0x7bf2d9c0, _cmd="v
NSInvocation: 0x7bf433e0
self: 0x7bf2d9c0
```

打印所有堆栈：

```
(lldb) pinvocation -a
frame #0: 0x000962aa TMasonry`-[ViewController viewDidLoad](self=0x7bf2d9c0, _cmd="v
NSInvocation: 0x7d2bb050
self: 0x7bf2d9c0
---------------------------------
frame #1: 0x008062ae UIKit`-[UIViewController _sendViewDidLoadWithAppearanceProxyObj
NSInvocation: 0x7be18a50
self: 0x7bf2d9c0
---------------------------------
frame #2: 0x0080adce UIKit`-[UIViewController loadViewIfRequired] + 1384
NSInvocation: 0x7bf0cc40
self: 0x7bf2d9c0
---------------------------------
frame #3: 0x008569f9 UIKit`-[UINavigationController _layoutViewController:] + 52
NSInvocation: 0x7d340c90
self: 0x7c89ee00

Argument:
0xbff69108, address of @} 0x7bf2d9c0
---------------------------------
frame #4: 0x008572b1 UIKit`-[UINavigationController _updateScrollViewFromViewControl
NSInvocation: 0x7d340cc0
self: 0x7c89ee00

2 Arguments:
0xbff69158, address of @} 0x0
0xbff6915c, address of @} 0x7bf2d9c0
---------------------------------
frame #5: 0x00857458 UIKit`-[UINavigationController _startTransition:fromViewControl
NSInvocation: 0x7bf24870
self: 0x7c89ee00

3 Arguments:
0xbff69298, address of i} 0
0xbff6929c, address of @} 0x0
0xbff692a0, address of @} 0x7bf2d9c0
---------------------------------
frame #6: 0x00858854 UIKit`-[UINavigationController _startDeferredTransitionIfNeeded
NSInvocation: 0x7bf16b50
self: 0x7c89ee00

Argument:
0xbff692f8, address of @} 0x0
---------------------------------
frame #7: 0x00859ada UIKit`-[UINavigationController __viewWillLayoutSubviews] + 68
NSInvocation: 0x7be18930
self: 0x7c89ee00
---------------------------------
frame #8: 0x00a35c4a UIKit`-[UILayoutContainerView layoutSubviews] + 252
NSInvocation: 0x7bf26ab0
self: 0x7bf40d40
```

```
frame #9: 0x0070b008 UIKit`-[UIView(CALayerDelegate) layoutSublayersOfLayer:] + 810
NSInvocation: 0x7bf25da0
self: 0x7bf40d40

Argument:
0xbff693b8, address of @} 0x7bf40f80
---------------------------------
frame #10: 0x01b23059 libobjc.A.dylib`-[NSObject performSelector:withObject:] + 70
NSInvocation: 0x7d2bb6c0
self: 0x7bf40d40

2 Arguments:
0xbff693d8, address of :} layoutSublayersOfLayer:
0xbff693dc, address of @} 0x7bf40f80
---------------------------------
frame #11: 0x0496b80a QuartzCore`-[CALayer layoutSublayers] + 144
NSInvocation: 0x7bf134c0
self: 0x7bf40f80
---------------------------------
frame #12: 0x0495f4ee QuartzCore`CA::Layer::layout_if_needed(CA::Transaction*) + 388
---------------------------------
frame #13: 0x0495f352 QuartzCore`CA::Layer::layout_and_display_if_needed(CA::Transac
---------------------------------
frame #14: 0x04951e8b QuartzCore`CA::Context::commit_transaction(CA::Transaction*) +
---------------------------------
frame #15: 0x04985e03 QuartzCore`CA::Transaction::commit() + 561
---------------------------------
frame #16: 0x049866c4 QuartzCore`CA::Transaction::observer_callback(__CFRunLoopObser
---------------------------------
frame #17: 0x01f66ffe CoreFoundation`__CFRUNLOOP_IS_CALLING_OUT_TO_AN_OBSERVER_CALLB
---------------------------------
frame #18: 0x01f66f5e CoreFoundation`__CFRunLoopDoObservers + 398
---------------------------------
frame #19: 0x01f5c108 CoreFoundation`CFRunLoopRunSpecific + 504
---------------------------------
frame #20: 0x01f5befb CoreFoundation`CFRunLoopRunInMode + 123
---------------------------------
frame #21: 0x00639206 UIKit`-[UIApplication _run] + 540
NSInvocation: 0x7d21fc00
self: 0x7d30bfe0
---------------------------------
frame #22: 0x0063ebfa UIKit`UIApplicationMain + 160
---------------------------------
frame #23: 0x00096a0a TMasonry`main(argc=1, argv=0xbff6a898) + 138 at main.m:14
---------------------------------
frame #24: 0x031caa21 libdyld.dylib`start + 1
---------------------------------
```


Accessibility

使用，UIAccessibility 详细资料可以阅读UIAccessibility (http://nshipster.cn/uiaccessibility/)

fa11y

根据view的accessibilityLabel (https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAccessibility_Protocol/#//apple_ref/occ/instp/NSObject/accessibilityLabel)查找view

语法:

```
Syntax: fa11y <labelRegex>
```

- <labelRegex>：需要匹配的text

说明：UILabel，UIButton的 accessibilityLabel 等于title

e.g: 我们根据显示的文案，查找相应的控件

```
(lldb) fa11y 妈妈
(UILabel 0x176b5bd0) 妈妈叫你回家吃饭了
```

pa11y

打印view层级中所有的 accessibilityLabel

语法:

```
Syntax: pa11y <aView>
```

- <aView>：需要打印层级的View，UIView类型

e.g: 我们打印一下self.view层级中所有的 accessibilityLabel

```
(lldb) pa11y self.view
UIView self.view
   | (UIButton 0x176b4600) Button
   | (UILabel 0x176b5bd0) 妈妈叫你回家吃饭了
```

chisel的命令就介绍到这里，东西挺多的，没必要全部都记住，我把这个写出来就打算当做一个文档，以后记不起哪个命令就查一下自己的博客。

不过chisel确实是个好东西，工欲善其事，必先利其器。要想要方便的调bug，chisel绝对值得你拥有

Reference

- chisel (https://github.com/gkassabli/chisel)
- 小笨狼与LLDB的故事 (http://www.jianshu.com/p/e89af3e9a8d7)
- Ambiguous Layouts (https://developer.apple.com/library/tvos/documentation/UserExperience/Conceptual/AutolayoutPG/AmbiguousLayouts.html#//apple_ref/doc/uid/TP40010853-CH18-SW1)
- UIAccessibility (http://nshipster.cn/uiaccessibility/)
- accessibilityLabel (https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIAccessibility_Protocol/#//apple_ref/occ/instp/NSObject/accessibilityLabel)

About Me

个人博客 (http://jiangliancheng.gitcafe.io/) 简书 (http://www.jianshu.com/users/1f93e3b1f3da) 微博 (http://weibo.com/u/5592239559) QQ群：159974494

欢迎大家关注我，共同学习iOS~

曾樑 (/users/e1fcce9bdf77)
2楼 · 2015.12.26 17:13 (/p/afaaacc55460/comments/1104602#comment-1104602)
挺全的
♡ 喜欢(0) 回复

刘畅畅可爱多 (/users/d809e1e0eff4)
3楼 · 2015.12.26 17:25 (/p/afaaacc55460/comments/1104670#comment-1104670)
那只狗吓死人
♡ 喜欢(0) 回复

> 小笨狼 (/users/1f93e3b1f3da)： @刘畅畅可爱多 (/users/d809e1e0eff4) 哈哈，你不觉得这家伙很逗比么
> 2015.12.26 17:38 (/p/afaaacc55460/comments/1104743#comment-1104743) 回复
>
> 添加新回复

张无忌_ (/users/1a0353613239)
4楼 · 2015.12.26 19:31 (/p/afaaacc55460/comments/1105314#comment-1105314)
很全，很棒。
♡ 喜欢(0) 回复

> 小笨狼 (/users/1f93e3b1f3da)： @张无忌_ (/users/1a0353613239) 谢谢^ω^
> 2015.12.26 19:44 (/p/afaaacc55460/comments/1105426#comment-1105426) 回复
>
> 添加新回复

从今以后 (/users/56f178cefae5)
5楼 · 2015.12.27 16:39 (/p/afaaacc55460/comments/1110373#comment-1110373)
😂 业界良心
♡ 喜欢(0) 回复

庸者的救赎 (/users/0726f4d689a3)
6楼 · 2015.12.28 04:15 (/p/afaaacc55460/comments/1113745#comment-1113745)

♡ 喜欢(0) 回复

xi_lin (/users/82e193381480)
7楼 · 2016.01.04 16:49 (/p/afaaacc55460/comments/1158796#comment-1158796)

♡ 喜欢(0) 回复

xi_lin (/users/82e193381480)
8楼 · 2016.01.04 16:50 (/p/afaaacc55460/comments/1158799#comment-1158799)

可以再说明一下alamunborder的使用吗？我用paltrace打印出来的结果里有提示AMBIGUOUS LAYOUT，但是用alamunborder没有效果，而且一回车程序就卡死了，无法continue

♡ 喜欢(0) 回复

> 小笨狼 (/users/1f93e3b1f3da)： @xi_lin (/users/82e193381480) 没有效果会不会是view的宽或者高是0，至于卡死得具体看情况才知道原因
> 2016.01.04 18:17 (/p/afaaacc55460/comments/1159364#comment-1159364) 回复
>
> 添加新回复

ColeXm (/users/4c8ed0eab359)
9楼 · 2016.01.06 14:43 (/p/afaaacc55460/comments/1172080#comment-1172080)

非常好，感谢分享。

♡ 喜欢(0) 回复

be47022ef6ea (/users/be47022ef6ea)
10楼 · 2016.01.20 08:36 (/p/afaaacc55460/comments/1276607#comment-1276607)

感谢🙏作者满满的干货👍

♡ 喜欢(0) 回复

登录后发表评论 (/sign_in)

简书程序员大本营 投稿须知： 1.本专题仅收录与程序有关的文章。 2.请在代码板里写代码，尽量保证可看性。

5256篇文章 (/collection/NEt52a) · 50380人关注

(/collection/NEt52a)

iOS Developer (/collection/3233d1a249ca)

分享 iOS 开发的知识，解决大家遇到的问题，讨论iOS开发的前沿，欢迎大家投稿~

添加关注 (/sign_in)

1700篇文章 (/collection/3233d1a249ca) · 11698人关注

(/collection/3233d1a249ca)

iOS开发技巧 (/collection/19dbe28002a3)

【简介】 专题内容主要包括OC、swift等涉及到iOS开发进阶的内容。 swift可以关注下我的另一个专题： swift开发...

添加关注 (/sign_in)

800篇文章 (/collection/19dbe28002a3) · 9813人关注

(/collection/19dbe28002a3)