Debugging with LLDB

Session 415

Greg Clayton
LLDB Architect

LLDB

Year in review

JANUARY FEBRUARY

MAY JUNE JULY

MAY JUNE 2011 JULY

WWDC2011

LLDB was available in Xcode seeds

MAY JUNE JULY

WWDC2011

LLDB was available in Xcode seeds

EMBER DECEMBER JANUARY

EMBER DECEMBER 2011 JANUARY

Xcode 4.3

LLDB became the default debugger

EMBER DECEMBER JANUARY

Xcode 4.3

LLDB became the default debugger

MAY JUNE JULY

MAY JUNE 2012 JULY

WWDC2012

Xcode 4.5 seeded

MAY JUNE 2012 JULY

WWDC2012

Xcode 4.5 seeded

Vastly improved LLDB

LLDB Improvements

Fixes and features

LLDB Improvements

Fixes and features

LLDB Improvements

Fixes and features

- Improved Objective-C debugging support
  - Objective-C property syntax
  - Full Objective-C class definitions

LLDB Improvements

Fixes and features

- Improved Objective-C debugging support
  - Objective-C property syntax
  - Full Objective-C class definitions
- Data formatters now in LLDB
  - Objective-C
  - C++ STL types and collections

LLDB Improvements

Fixes and features

- Improved Objective-C debugging support
  - Objective-C property syntax
  - Full Objective-C class definitions
- Data formatters now in LLDB
  - Objective-C
  - C++ STL types and collections
- Watchpoints for desktop and iOS

LLDB Improvements

Fixes and features

- Improved Objective-C debugging support
  - Objective-C property syntax
  - Full Objective-C class definitions
- Data formatters now in LLDB
  - Objective-C
  - C++ STL types and collections
- Watchpoints for desktop and iOS
- Improved Python scripting

Overview

Introduction LLDB in depth Examples Conclusion

Overview

Introduction

LLDB in depth

Examples

Conclusion

Introduction

Why create LLDB?

Introduction

Why create LLDB?

- Wanted better debugger

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?
  - Architecture

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?
    - Architecture
    - Parses information in large chunks

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?
  - Architecture
  - Parses information in large chunks
  - GDB was not designed to vend an API

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?
  - Architecture
  - Parses information in large chunks
  - GDB was not designed to vend an API
  - Global variables contain program state

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?
  - Architecture
  - Parses information in large chunks
  - GDB was not designed to vend an API
  - Global variables contain program state
  - Different GDB binaries for each architecture

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?
  - Architecture
  - Parses information in large chunks
  - GDB was not designed to vend an API
  - Global variables contain program state
  - Different GDB binaries for each architecture
    - Pervasive preprocessor macros

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?
  - Architecture
  - Parses information in large chunks
  - GDB was not designed to vend an API
  - Global variables contain program state
  - Different GDB binaries for each architecture
    - Pervasive preprocessor macros
  - Issues with expression parser

Introduction

Why create LLDB?

- Wanted better debugger
- What was wrong with GDB?
    - Architecture
    - Parses information in large chunks
    - GDB was not designed to vend an API
    - Global variables contain program state
    - Different GDB binaries for each architecture
        - Pervasive preprocessor macros
    - Issues with expression parser
        - Objective-C properties

Introduction

Design goals

Introduction

Design goals

- Performance
- Memory
- Customizable
- Compiler integration
- Modern architecture

Introduction

Design goals

- Performance
- Memory
- Customizable
- Compiler integration
- Modern architecture

Introduction

Customizable

Introduction
Customizable

- Variable and value display
    - Formats
    - Summaries
    - Synthetic instance variables

Introduction
Customizable

- Variable and value display
  - Formats
  - Summaries
  - Synthetic instance variables
- Commands
  - Aliases
  - User-defined

Introduction

Customizable

- Variable and value display
  - Formats
  - Summaries
  - Synthetic instance variables
- Commands
  - Aliases
  - User-defined
- Prompts

Introduction

Why do compiler integration?

Introduction

Why do compiler integration?

- What do debuggers do?

Introduction

Why do compiler integration?

- What do debuggers do?
    - Debuggers invent their own type representation

Introduction

Why do compiler integration?

- What do debuggers do?
  - Debuggers invent their own type representation
  - Expression parsers use these types

Introduction

Why do compiler integration?

- What do debuggers do?
  - Debuggers invent their own type representation
  - Expression parsers use these types
  - Strive for compiler level of accuracy

Introduction

Why do compiler integration?

- What do debuggers do?
  - Debuggers invent their own type representation
  - Expression parsers use these types
  - Strive for compiler level of accuracy
  - Expression parser needs to be updated

Introduction

Why do compiler integration?

- What do debuggers do?
  - Debuggers invent their own type representation
  - Expression parsers use these types
  - Strive for compiler level of accuracy
  - Expression parser needs to be updated
- How hard can it be to write a good C++ parser?

Introduction

Compiler integration in LLDB

Introduction

Compiler integration in LLDB

- Full Clang compiler built in

Introduction

Compiler integration in LLDB

- Full Clang compiler built in
- LLDB converts debugging information into native Clang types

Introduction

Compiler integration in LLDB

- Full Clang compiler built in
- LLDB converts debugging information into native Clang types
  - Use Clang AST data structures for types

Introduction

Compiler integration in LLDB

- Full Clang compiler built in
- LLDB converts debugging information into native Clang types
  - Use Clang AST data structures for types
- Use the compiler to parse expressions

Introduction

Compiler integration in LLDB

- Full Clang compiler built in
- LLDB converts debugging information into native Clang types
  - Use Clang AST data structures for types
- Use the compiler to parse expressions
  - Attain what other debuggers strive for

Introduction

Compiler integration in LLDB

- Full Clang compiler built in
- LLDB converts debugging information into native Clang types
  - Use Clang AST data structures for types
- Use the compiler to parse expressions
  - Attain what other debuggers strive for
  - Complete language support

Introduction

Compiler integration in LLDB

- Full Clang compiler built in
- LLDB converts debugging information into native Clang types
    - Use Clang AST data structures for types
- Use the compiler to parse expressions
    - Attain what other debuggers strive for
    - Complete language support
    - Accurate error reporting

Introduction

Compiler integration in LLDB

- Full Clang compiler built in
- LLDB converts debugging information into native Clang types
    - Use Clang AST data structures for types
- Use the compiler to parse expressions
    - Attain what other debuggers strive for
    - Complete language support
    - Accurate error reporting
    - Free compiler features
        - Objective-C Literals
        - C++11

Introduction

Modern architecture

Introduction

Modern architecture

- Clean object-oriented design

Introduction

Modern architecture

- Clean object-oriented design
  - Encapsulation

Introduction

Modern architecture

- Clean object-oriented design
    - Encapsulation
    - Plug-ins

Introduction

Modern architecture

- Clean object-oriented design
  - Encapsulation
  - Plug-ins
- Designed for today’s debugging requirements

Introduction

Modern architecture

- Clean object-oriented design
  - Encapsulation
  - Plug-ins
- Designed for today's debugging requirements
  - Multi-threaded programs

Introduction

Modern architecture

- Clean object-oriented design
  - Encapsulation
  - Plug-ins
- Designed for today's debugging requirements
  - Multi-threaded programs
  - Stay in sync with compiler

Introduction

Modern architecture

- Clean object-oriented design
  - Encapsulation
  - Plug-ins
- Designed for today's debugging requirements
  - Multi-threaded programs
  - Stay in sync with compiler
- LLDB is a framework

Introduction

Modern architecture

- Clean object-oriented design
  - Encapsulation
  - Plug-ins
- Designed for today’s debugging requirements
  - Multi-threaded programs
  - Stay in sync with compiler
- LLDB is a framework
  - Provides an API to the debugger

Introduction

Modern architecture

- Clean object-oriented design
  - Encapsulation
  - Plug-ins
- Designed for today's debugging requirements
  - Multi-threaded programs
  - Stay in sync with compiler
- LLDB is a framework
  - Provides an API to the debugger
  - Scriptable with Python

Overview

Debugging with LLDB

Introduction

LLDB in depth

Examples

Conclusion

Overview

Debugging with LLDB

LLDB in depth

Introduction

Examples

Conclusion

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

Getting Started

First commands

Getting Started

First commands

```
% xcrun lldb
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
(lldb) run
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
(lldb) run
(lldb) bt
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
(lldb) run
(lldb) bt
(lldb) step
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
(lldb) run
(lldb) bt
(lldb) step
(lldb) step
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
(lldb) run
(lldb) bt
(lldb) step
(lldb) step
(lldb) print argc
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
(lldb) run
(lldb) bt
(lldb) step
(lldb) step
(lldb) print argc
(lldb) next
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
(lldb) run
(lldb) bt
(lldb) step
(lldb) step
(lldb) print argc
(lldb) next
(lldb) next
```

Getting Started

First commands

```
% xcrun lldb
(lldb) file a.out
(lldb) b main
(lldb) run
(lldb) bt
(lldb) step
(lldb) step
(lldb) print argc
(lldb) next
(lldb) next
(lldb) q
```

Getting Started

First commands

Getting Started

First commands

```
% xcrun lldb
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
(lldb) process launch
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
(lldb) process launch
(lldb) thread backtrace
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
(lldb) process launch
(lldb) thread backtrace
(lldb) thread step-in
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
(lldb) process launch
(lldb) thread backtrace
(lldb) thread step-in
(lldb) thread step-in
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
(lldb) process launch
(lldb) thread backtrace
(lldb) thread step-in
(lldb) thread step-in
(lldb) expression argc
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
(lldb) process launch
(lldb) thread backtrace
(lldb) thread step-in
(lldb) thread step-in
(lldb) expression argc
(lldb) thread step-over
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
(lldb) process launch
(lldb) thread backtrace
(lldb) thread step-in
(lldb) thread step-in
(lldb) expression argc
(lldb) thread step-over
(lldb) thread step-over
```

Getting Started

First commands

```
% xcrun lldb
(lldb) target create a.out
(lldb) breakpoint set --name main
(lldb) process launch
(lldb) thread backtrace
(lldb) thread step-in
(lldb) thread step-in
(lldb) expression argc
(lldb) thread step-over
(lldb) thread step-over
(lldb) quit
```

Getting Started
Command interpreter

Getting Started

Command interpreter

- GDB command interpreter had issues

Getting Started
Command interpreter

- GDB command interpreter had issues
  - Inconsistent syntax

Getting Started

Command interpreter

- GDB command interpreter had issues
  - Inconsistent syntax
  - Overloaded arguments

Getting Started
Command interpreter

- GDB command interpreter had issues
  - Inconsistent syntax
  - Overloaded arguments
- LLDB command interpreter

Getting Started

Command interpreter

- GDB command interpreter had issues
  - Inconsistent syntax
  - Overloaded arguments
- LLDB command interpreter
  - Consistent syntax

Getting Started
Command interpreter

- GDB command interpreter had issues
  - Inconsistent syntax
  - Overloaded arguments
- LLDB command interpreter
  - Consistent syntax
  - Use options instead of overloading

Getting Started

Command interpreter

- GDB command interpreter had issues
  - Inconsistent syntax
  - Overloaded arguments
- LLDB command interpreter
  - Consistent syntax
  - Use options instead of overloading
    - Targeted autocompletion

Getting Started

Command interpreter

- GDB command interpreter had issues
  - Inconsistent syntax
  - Overloaded arguments
- LLDB command interpreter
  - Consistent syntax
  - Use options instead of overloading
    - Targeted autocompletion
  - Discoverable commands

Getting Started
Command interpreter

- GDB command interpreter had issues
  - Inconsistent syntax
  - Overloaded arguments
- LLDB command interpreter
  - Consistent syntax
  - Use options instead of overloading
    - Targeted autocompletion
  - Discoverable commands
  - Built-in documentation

Command Syntax

Noun and verb

```
<noun> <verb>

(lldb) target create
(lldb) breakpoint set
(lldb) process launch
(lldb) thread step-in
(lldb) frame variable
```

Command Syntax

Shell style options

```
<noun> <verb> [options]

(lldb) target create --arch i386
(lldb) breakpoint set --name main
(lldb) process launch --stop-at-entry
(lldb) thread step-in
(lldb) frame variable --format hex
```

Command Syntax

Arguments

```
<noun> <verb> [options] [argument [argument...]]

(lldb) target create --arch i386 /bin/ls
(lldb) breakpoint set --name main
(lldb) process launch --stop-at-entry -- -lAF /tmp
(lldb) thread step-in
(lldb) frame variable --format hex argc argv[0]
```

Command Syntax

Arguments

```
<noun> <verb> [options] [argument [argument...]]

(lldb) target create /bin/ls --arch i386
(lldb) breakpoint set --name main
(lldb) process launch --stop-at-entry -- -lAF /tmp
(lldb) thread step-in
(lldb) frame variable --format hex argc argv[0]
```

Command Syntax

Shortest match

```
<noun> <verb> [options] [argument [argument...]]

(lldb) ta c /bin/ls --arch i386
(lldb) breakpoint set --name main
(lldb) process launch --stop-at-entry -- -lAF /tmp
(lldb) thread step-in
(lldb) frame variable --format hex argc argv[0]
```

Command Syntax

Shortest match

```
<noun> <verb> [options] [argument [argument...]]

(lldb) ta c /bin/ls --arch i386
(lldb) br s --name main
(lldb) process launch --stop-at-entry -- -lAF /tmp
(lldb) thread step-in
(lldb) frame variable --format hex argc argv[0]
```

Command Syntax

Shortest match

```
<noun> <verb> [options] [argument [argument...]]

(lldb) ta c /bin/ls --arch i386
(lldb) br s --name main
(lldb) pro la --stop-at-entry -- -lAF /tmp
(lldb) thread step-in
(lldb) frame variable --format hex argc argv[0]
```

Command Syntax

Shortest match

```
<noun> <verb> [options] [argument [argument...]]

(lldb) ta c /bin/ls --arch i386
(lldb) br s --name main
(lldb) pro la --stop-at-entry -- -lAF /tmp
(lldb) th step-in
(lldb) frame variable --format hex argc argv[0]
```

Command Syntax

Shortest match

```
<noun> <verb> [options] [argument [argument...]]

(lldb) ta c /bin/ls --arch i386
(lldb) br s --name main
(lldb) pro la --stop-at-entry -- -lAF /tmp
(lldb) th step-in
(lldb) fr v --format hex argc argv[0]
```

Command Syntax

Short options

```
<noun> <verb> [options] [argument [argument...]]

(lldb) ta c /bin/ls -a i386
(lldb) br s -n main
(lldb) pro la -s -- -lAF /tmp
(lldb) th step-in
(lldb) fr v -f x argc argv[0]
```

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

Terminology

`target`

Terminology

`target`

```
(lldb) file a.out
```

Terminology

`target`

```
(lldb) file a.out
(lldb) target create a.out
```

Terminology

target

```
(lldb) file a.out
(lldb) target create a.out
(lldb) target
```

Terminology

`target`

```
(lldb) file a.out
(lldb) target create a.out
(lldb) target
Available completions:
    create
    delete
    list
    modules
    select
    stop-hook
    symbols
    variable
```

Terminology

`target`

Terminology

`target`

```
(lldb) target create /bin/ls
```

Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
```

Terminology

`target`

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp
```

Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp

(lldb) target create /bin/cat
```

Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp

(lldb) target create /bin/cat
(lldb) breakpoint set --name free
```

Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp

(lldb) target create /bin/cat
(lldb) breakpoint set --name free
(lldb) process launch -- /tmp/test.txt
```

Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp

(lldb) target create /bin/cat
(lldb) breakpoint set --name free
(lldb) process launch -- /tmp/test.txt

(lldb) target list
Current targets:
  target #0: /bin/ls (arch=x86_64,pid=18879,state=stopped
* target #1: /bin/cat (arch=x86_64,pid=18885,state=stoppe
```

Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp

(lldb) target create /bin/cat
(lldb) breakpoint set --name free
(lldb) process launch -- /tmp/test.txt

(lldb) target list
Current targets:
  target #0: /bin/ls (arch=x86_64,pid=18879,state=stopped
* target #1: /bin/cat (arch=x86_64,pid=18885,state=stoppe
(lldb) target select 0
```

Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp

(lldb) target create /bin/cat
(lldb) breakpoint set --name free
(lldb) process launch -- /tmp/test.txt

(lldb) target list
Current targets:
  target #0: /bin/ls (arch=x86_64,pid=18879,state=stopped
* target #1: /bin/cat (arch=x86_64,pid=18885,state=stoppe
(lldb) target select 0
(lldb) thread backtrace
```

Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp

(lldb) target create /bin/cat
(lldb) breakpoint set --name free
(lldb) process launch -- /tmp/test.txt

(lldb) target list
Current targets:
  target #0: /bin/ls (arch=x86_64,pid=18879,state=stopped
* target #1: /bin/cat (arch=x86_64,pid=18885,state=stoppe
(lldb) target select 0
(lldb) thread backtrace
(lldb) target select 1
```

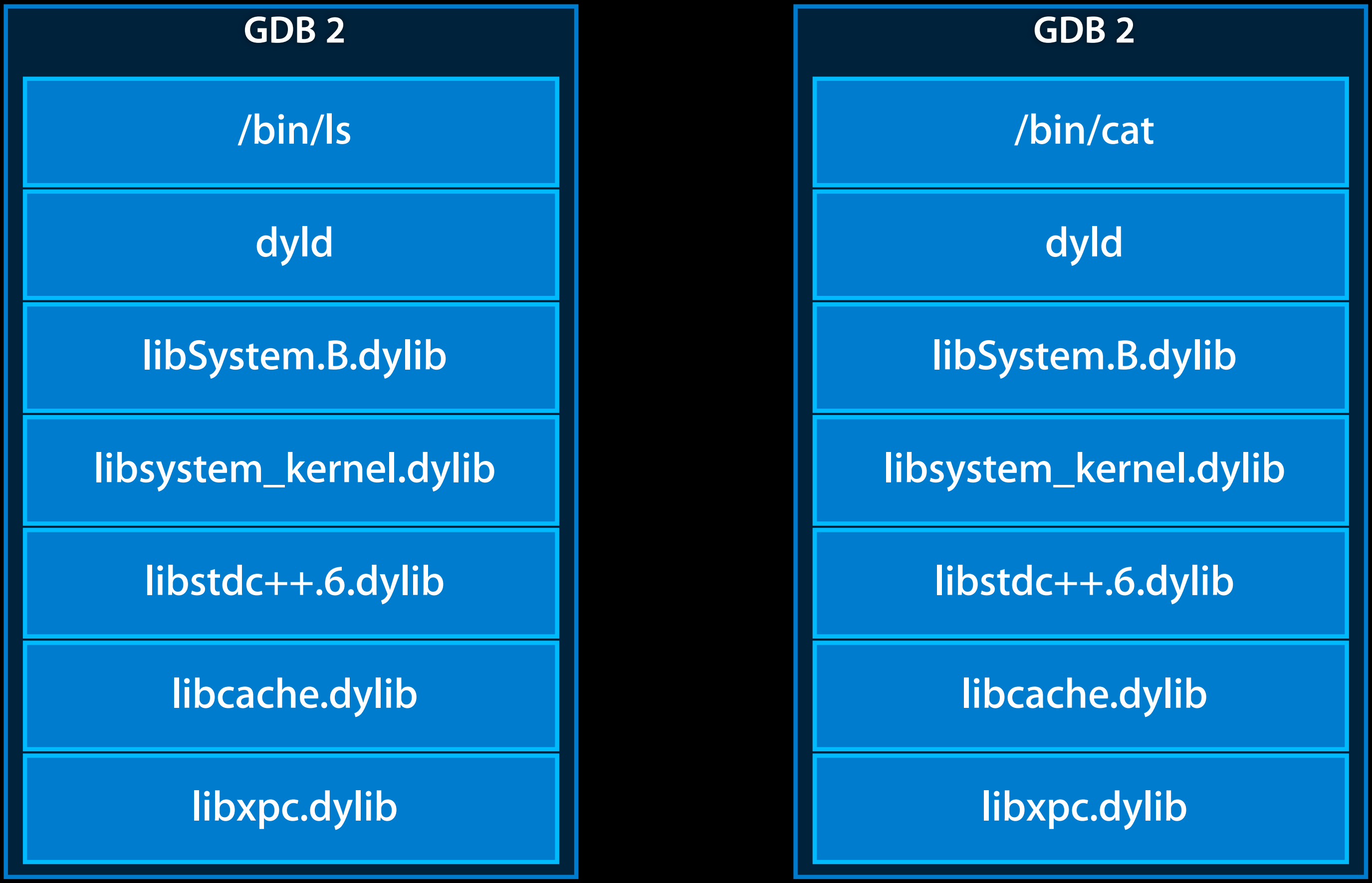
Terminology

target

```
(lldb) target create /bin/ls
(lldb) breakpoint set --name malloc
(lldb) process launch -- -lAF /tmp

(lldb) target create /bin/cat
(lldb) breakpoint set --name free
(lldb) process launch -- /tmp/test.txt

(lldb) target list
Current targets:
  target #0: /bin/ls (arch=x86_64,pid=18879,state=stopped
* target #1: /bin/cat (arch=x86_64,pid=18885,state=stoppe
(lldb) target select 0
(lldb) thread backtrace
(lldb) target select 1
(lldb) thread backtrace
```

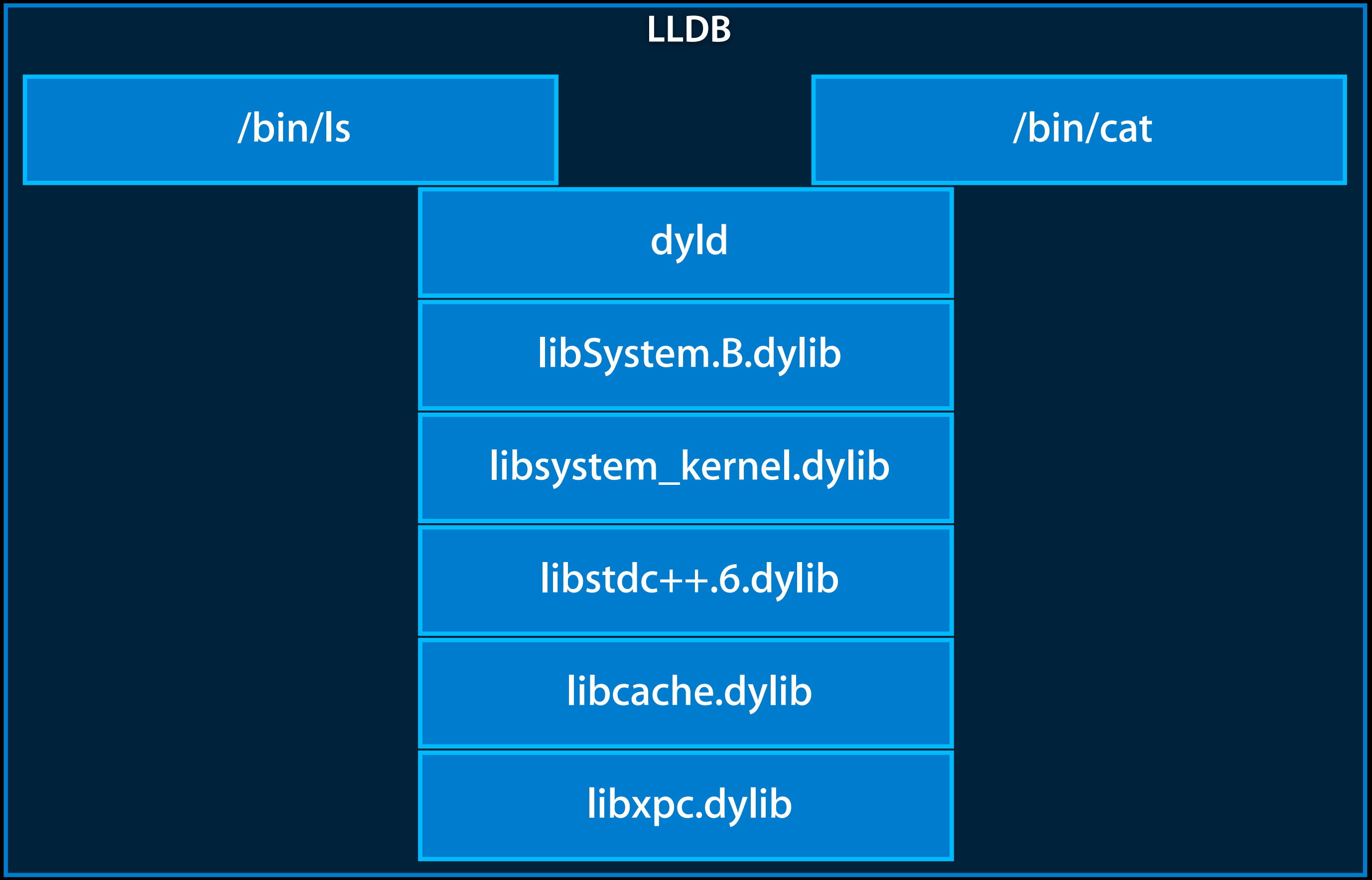
Memory Usage

GDB



Memory Usage

LLDB



Terminology

`process`

Terminology

process

```
(lldb) run <arg1> <arg2> ...
```

Terminology

process

```
(lldb) run <arg1> <arg2> ...
(lldb) process launch -- <arg1> <arg2> ...
```

Terminology

process

```
(lldb) run <arg1> <arg2> ...
(lldb) process launch -- <arg1> <arg2> ...
```

Terminology

process

```
(lldb) run <arg1> <arg2> ...
(lldb) process launch -- <arg1> <arg2> ...
```

Terminology

process

```
(lldb) run <arg1> <arg2> ...
(lldb) process launch -- <arg1> <arg2> ...
(lldb) process
```

Terminology

process

```
(lldb) run <arg1> <arg2> ...
(lldb) process launch -- <arg1> <arg2> ...
(lldb) process
Available completions:
	attach
	connect
	continue
	detach
	handle
	interrupt
	kill
	launch
	load
	signal
	status
	unload
```

Terminology

`process`

Terminology

process

```
(lldb) process attach --pid 123
```

Terminology

process

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
```

Terminology

process

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
(lldb) target create /Applications/Safari.app
```

Terminology

process

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
(lldb) target create /Applications/Safari.app
(lldb) process attach
```

Terminology

process

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
(lldb) target create /Applications/Safari.app
(lldb) process attach
(lldb) target create com.apple.my_xpc_service
```

Terminology

process

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
(lldb) target create /Applications/Safari.app
(lldb) process attach
(lldb) target create com.apple.my_xpc_service
(lldb) process attach --waitfor
```

Terminology

process

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
(lldb) target create /Applications/Safari.app
(lldb) process attach
(lldb) target create com.apple.my_xpc_service
(lldb) process attach --waitfor
(lldb) process continue
```

Terminology

process

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
(lldb) target create /Applications/Safari.app
(lldb) process attach
(lldb) target create com.apple.my_xpc_service
(lldb) process attach --waitfor
(lldb) process continue
(lldb) continue
```

Terminology

`process`

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
(lldb) target create /Applications/Safari.app
(lldb) process attach
(lldb) target create com.apple.my_xpc_service
(lldb) process attach --waitfor
(lldb) process continue
(lldb) continue
(lldb) c
```

Terminology

process

```
(lldb) process attach --pid 123
(lldb) process attach --name Safari
(lldb) target create /Applications/Safari.app
(lldb) process attach
(lldb) target create com.apple.my_xpc_service
(lldb) process attach --waitfor
(lldb) process continue
(lldb) continue
(lldb) c
^C
```

Terminology

`thread`

Terminology

`thread`

```
(lldb) thread
```

Terminology

thread

```
(lldb) thread
Available completions:
    backtrace
    continue
    list
    select
    step-in
    step-inst
    step-inst-over
    step-out
    step-over
    until
```

Terminology

thread

```
(lldb) thread
Available completions:
   backtrace
   continue
   list
   select
   step-in
   step-inst
   step-inst-over
   step-out
   step-over
   until
(lldb) thread list
```

Terminology

thread

```
(lldb) thread
Available completions:
   backtrace
   continue
   list
   select
   step-in
   step-inst
   step-inst-over
   step-out
   step-over
   until
(lldb) thread list
(lldb) thread select 12
```

Terminology

thread

```
(lldb) thread
Available completions:
   backtrace
   continue
   list
   select
   step-in
   step-inst
   step-inst-over
   step-out
   step-over
   until
(lldb) thread list
(lldb) thread select 12
(lldb) thread backtrace
```

Terminology

thread

```
(lldb) thread
Available completions:
   backtrace
   continue
   list
   select
   step-in
   step-inst
   step-inst-over
   step-out
   step-over
   until
(lldb) thread list
(lldb) thread select 12
(lldb) thread backtrace
(lldb) bt
```

Terminology

thread

```
(lldb) thread
Available completions:
   backtrace
   continue
   list
   select
   step-in
   step-inst
   step-inst-over
   step-out
   step-over
   until
(lldb) thread list
(lldb) thread select 12
(lldb) thread backtrace
(lldb) bt
(lldb) bt all
```

Terminology

`frame`

Terminology

`frame`

```
(lldb) frame
```

Terminology

`frame`

```
(lldb) frame
Available completions:
    info
    select
    variable
```

Terminology

`frame`

```
(lldb) frame
Available completions:
    info
    select
    variable
(lldb) frame select 12
```

Terminology

`frame`

```
(lldb) frame
Available completions:
   info
   select
   variable
(lldb) frame select 12
(lldb) f 12
```

Terminology

`frame`

```
(lldb) frame
Available completions:
   info
   select
   variable
(lldb) frame select 12
(lldb) f 12
(lldb) up
```

Terminology

`frame`

```
(lldb) frame
Available completions:
   info
   select
   variable
(lldb) frame select 12
(lldb) f 12
(lldb) up
(lldb) down
```

Terminology

`frame`

```
(lldb) frame
Available completions:
   info
   select
   variable
(lldb) frame select 12
(lldb) f 12
(lldb) up
(lldb) down
(lldb) frame variable
```

Terminology

`frame`

```
(lldb) frame
Available completions:
    info
    select
    variable
(lldb) frame select 12
(lldb) f 12
(lldb) up
(lldb) down
(lldb) frame variable
(gdb) info locals
```

Terminology

frame

```
(lldb) frame
Available completions:
   info
   select
   variable
(lldb) frame select 12
(lldb) f 12
(lldb) up
(lldb) down
(lldb) frame variable
(gdb) info locals
(gdb) info args
```

Terminology

`modules`

Terminology

`modules`

```
(lldb) target modules list
```

Terminology

modules

```
(lldb) target modules list
(gdb) info shared
```

Terminology

modules

```
(lldb) target modules list
(gdb) info shared
(lldb) target modules list [file1 ...]
```

Terminology

modules

```
(lldb) target modules list
(gdb) info shared
(lldb) target modules list [file1 ...]
(lldb) target modules dump symtab [file1 ...]
```

Terminology

modules

```
(lldb) target modules list
(gdb) info shared
(lldb) target modules list [file1 ...]
(lldb) target modules dump symtab [file1 ...]
(lldb) target modules dump sections [file1 ...]
```

Terminology

modules

```
(lldb) target modules list
(gdb) info shared
(lldb) target modules list [file1 ...]
(lldb) target modules dump symtab [file1 ...]
(lldb) target modules dump sections [file1 ...]
(lldb) target modules lookup --address <address>
```

Terminology

modules

```
(lldb) target modules list
(gdb) info shared
(lldb) target modules list [file1 ...]
(lldb) target modules dump symtab [file1 ...]
(lldb) target modules dump sections [file1 ...]
(lldb) target modules lookup --address <address>
(lldb) target modules lookup --type <name>
```

Help

```
syntax: help [command]
```

Help

```
syntax: help [command]
```

```
(lldb) help memory read
```

Help

`syntax: help [command]`

```
(lldb) help memory read
   Read from the memory of the process being debugged.

Syntax: memory read <cmd-options> <start-address> [<end-address>]

Command Options Usage:
  memory read [-A] [-f <format>] [-c <count>] [-G <gdb-format>] [-s <byte-siz
  memory read [-bA] [-f <format>] [-c <count>] [-s <byte-size>] [-o <path>] <
  memory read [-AFLORT] -t <none> [-f <format>] [-c <count>] [-G <gdb-format>

       -A  ( --append-outfile )
            Append to the the file specified with '--outfile <path>'.

       -D <count>  ( --depth <count> )
            Set the max recurse depth when dumping aggregate types (default

       -F  ( --flat )
            Display results in a flat format that uses expression paths for
```

```
memory read [-A] [-f <format>] [-c <count>] [-G <gdb-format>] [-s <byte-siz
memory read [-bA] [-f <format>] [-c <count>] [-s <byte-size>] [-o <path>] <
memory read [-AFLORT] -t <none> [-f <format>] [-c <count>] [-G <gdb-format>

     -A  ( --append-outfile )
          Append to the the file specified with '--outfile <path>'.

     -D <count>  ( --depth <count> )
          Set the max recurse depth when dumping aggregate types (default

     -F  ( --flat )
          Display results in a flat format that uses expression paths for

     -f <format>  ( --format <format> )
          Specify a format to be used for display.

     -L  ( --location )
          Show variable location information.
```

```
memory read [-A] [-f <format>] [-c <count>] [-G <gdb-format>] [-s <byte-siz
memory read [-bA] [-f <format>] [-c <count>] [-s <byte-size>] [-o <path>] <
memory read [-AFLORT] -t <none> [-f <format>] [-c <count>] [-G <gdb-format>

     -A  ( --append-outfile )
          Append to the the file specified with '--outfile <path>'.

     -D <count>  ( --depth <count> )
          Set the max recurse depth when dumping aggregate types (default

     -F  ( --flat )
          Display results in a flat format that uses expression paths for

     -f <format>  ( --format <format> )
          Specify a format to be used for display.

     -L  ( --location )
          Show variable location information.
```

Help

```
syntax: help <option-type>
```

Help

```
syntax: help <option-type>
```

```
(lldb) help <format>
```

Help

`syntax: help <option-type>`

```
(lldb) help <format>
<format> -- One of the format names (or one-character names) that can be used
to show a variable's value:
            "default"
            'B' or "boolean"
            'b' or "binary"
            'y' or "bytes"
            'Y' or "bytes with ASCII"
            'c' or "character"
            'C' or "printable character"
            'F' or "complex float"
            's' or "c-string"
            'd' or "decimal"
            'E' or "enumeration"
            'x' or "hex"
            'f' or "float"
            'o' or "octal"
```

Apropos

```
syntax: apropos <keyword>
```

Apropos

```
syntax: apropos <keyword>
```

```
(lldb) apropos thread
```

Apropos

syntax: apropos <keyword>

```
(lldb) apropos thread
The following commands may relate to 'thread':
breakpoint command add -- Add a set of commands to a breakpoint, to be execut
breakpoint modify      -- Modify the options on a breakpoint or set of breakp
breakpoint set         -- Sets a breakpoint or set of breakpoints in the exec
frame                  -- A set of commands for operating on the current thre
frame info             -- List information about the currently selected frame
frame select           -- Select a frame by index from within the current thr
log enable             -- Enable logging for a single log channel.
process continue       -- Continue execution of all threads in the current pr
register               -- A set of commands to access thread registers.
target stop-hook add   -- Add a hook to be executed when the target stops.
thread                 -- A set of commands for operating on one or more thre
thread backtrace       -- Show the stack for one or more threads.  If no thre
thread continue        -- Continue execution of one or more threads in an act
thread list            -- Show a summary of all current threads in a process.
thread select          -- Select a thread as the currently active thread.
```

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

LLDB in Depth

Customizing commands

- Simple aliases
- Regular expression aliases
- User-defined

Aliasing

Aliasing

```
(lldb) command alias <name> <command> [<arg1> <arg2> ...]
```

Aliasing

```
(lldb) command alias <name> <command> [<arg1> <arg2> ...]
(lldb) command alias up frame select --relative=1
```

Aliasing

```
(lldb) command alias <name> <command> [<arg1> <arg2> ...]
(lldb) command alias up frame select --relative=1
(lldb) command alias down frame select -r-1
```

Aliasing

```
(lldb) command alias <name> <command> [<arg1> <arg2> ...]
(lldb) command alias up frame select --relative=1
(lldb) command alias down frame select -r-1
(lldb) command alias disasm-range
```

Aliasing

```
(lldb) command alias <name> <command> [<arg1> <arg2> ...]
(lldb) command alias up frame select --relative=1
(lldb) command alias down frame select -r-1
(lldb) command alias disasm-range
         disassemble
```

Aliasing

```
(lldb) command alias <name> <command> [<arg1> <arg2> ...]
(lldb) command alias up frame select --relative=1
(lldb) command alias down frame select -r-1
(lldb) command alias disasm-range
         disassemble
           --start-address %1
```

Aliasing

```
(lldb) command alias <name> <command> [<arg1> <arg2> ...]
(lldb) command alias up frame select --relative=1
(lldb) command alias down frame select -r-1
(lldb) command alias disasm-range
         disassemble
           --start-address %1
           --end-address %2
```

Regular Expression Aliases

Power user feature

Regular Expression Aliases

Power user feature

- Specify a command alias name

Regular Expression Aliases

Power user feature

- Specify a command alias name
- One or more regular expressions with substitutions

```
s/<regex>/<subst>/
```

Regular Expression Aliases

Power user feature

- Specify a command alias name
- One or more regular expressions with substitutions

```
s/<regex>/<subst>/
```

- When the alias is used

Regular Expression Aliases

Power user feature

- Specify a command alias name
- One or more regular expressions with substitutions

```
s/<regex>/<subst>/
```

- When the alias is used
    - Arguments following alias are matched against regular expressions

Regular Expression Aliases

Power user feature

- Specify a command alias name
- One or more regular expressions with substitutions

  ```
  s/<regex>/<subst>/
  ```
- When the alias is used
  - Arguments following alias are matched against regular expressions
  - First regular expression to match wins

Regular Expression Aliases

Power user feature

- Specify a command alias name
- One or more regular expressions with substitutions

```
s/<regex>/<subst>/
```

- When the alias is used
  - Arguments following alias are matched against regular expressions
  - First regular expression to match wins
    - Substitutions are performed

Regular Expression Aliases

Power user feature

- Specify a command alias name
- One or more regular expressions with substitutions

```
s/<regex>/<subst>/
```

- When the alias is used
  - Arguments following alias are matched against regular expressions
  - First regular expression to match wins
    - Substitutions are performed
    - New command is executed

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
     “s/^([0-9]+)$/frame select %1/”
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
     “s/^([0-9]+)$/frame select %1/”
     “s/^([+-][0-9]+)$/frame select --relative=%1/”
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
     "s/^([0-9]+)$/frame select %1/"
     "s/^([+-][0-9]+)$/frame select --relative=%1/"
     "s/^(.*)$/frame variable %1/"
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
     “s/^([0-9]+)$/frame select %1/”
     “s/^([+-][0-9]+)$/frame select --relative=%1/”
     “s/^(.*)$/frame variable %1/”
(lldb) f 12
frame select 12
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
     “s/^([0-9]+)$/frame select %1/”
     “s/^([+-][0-9]+)$/frame select --relative=%1/”
     “s/^(.*)$/frame variable %1/”
(lldb) f 12
frame select 12
(lldb) f +2
frame select --relative=+2
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
     “s/^([0-9]+)$/frame select %1/”
     “s/^([+-][0-9]+)$/frame select --relative=%1/”
     “s/^(.*)$/frame variable %1/”
(lldb) f 12
frame select 12
(lldb) f +2
frame select --relative=+2
(lldb) f -1
frame select --relative=-1
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
     “s/^([0-9]+)$/frame select %1/”
     “s/^([+-][0-9]+)$/frame select --relative=%1/”
     “s/^(.*)$/frame variable %1/”
(lldb) f 12
frame select 12
(lldb) f +2
frame select --relative=+2
(lldb) f -1
frame select --relative=-1
(lldb) f
frame variable
```

Regular Expression Aliases

```
syntax: command regex <name> [s/<regex>/<subst>/ ...]
```

```
(lldb) command regex f
     “s/^([0-9]+)$/frame select %1/”
     “s/^([+-][0-9]+)$/frame select --relative=%1/”
     “s/^(.*)$/frame variable %1/”
(lldb) f 12
frame select 12
(lldb) f +2
frame select --relative=+2
(lldb) f -1
frame select --relative=-1
(lldb) f
frame variable
(lldb) f argc argv
frame variable argc argv
```

User-Defined Commands

User-Defined Commands

- Make it yourself with Python

User-Defined Commands

- Make it yourself with Python
- Write Python module with a command function

```
def <function>(debugger, command, result, dict)
```

User-Defined Commands

- Make it yourself with Python
- Write Python module with a command function

```
def <function>(debugger, command, result, dict)
```

- Import module into LLDB

User-Defined Commands

- Make it yourself with Python
- Write Python module with a command function

```
def <function>(debugger, command, result, dict)
```

- Import module into LLDB
- Bind Python function to command

Python Command

Add “`ls`” command

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
```

Python Command

Add “`ls`” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
(lldb) command script add
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
(lldb) command script add
          --function lldbshell.ls_cmd
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
(lldb) command script add
          --function lldbshell.ls_cmd
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
(lldb) command script add
          --function lldbshell.ls_cmd
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
(lldb) command script add
          --function lldbshell.ls_cmd
```

Python Command

Add "ls" command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
(lldb) command script add
          --function lldbshell.ls_cmd
          ls
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
(lldb) command script add
          --function lldbshell.ls_cmd
          ls
```

Python Command

Add “ls” command

```
% cat /tmp/lldbshell.py
#!/usr/bin/python
import lldb
import commands
def ls_cmd(debugger, command, result, dict):
    shell_cmd = '/bin/ls %s' % command
    shell_result = commands.getoutput(shell_cmd)
    result.PutCString(shell_result)
% xcrun lldb
(lldb) command script import /tmp/lldbshell.py
(lldb) command script add
          --function lldbshell.ls_cmd
          ls
```

Using New Python Command

Using New Python Command

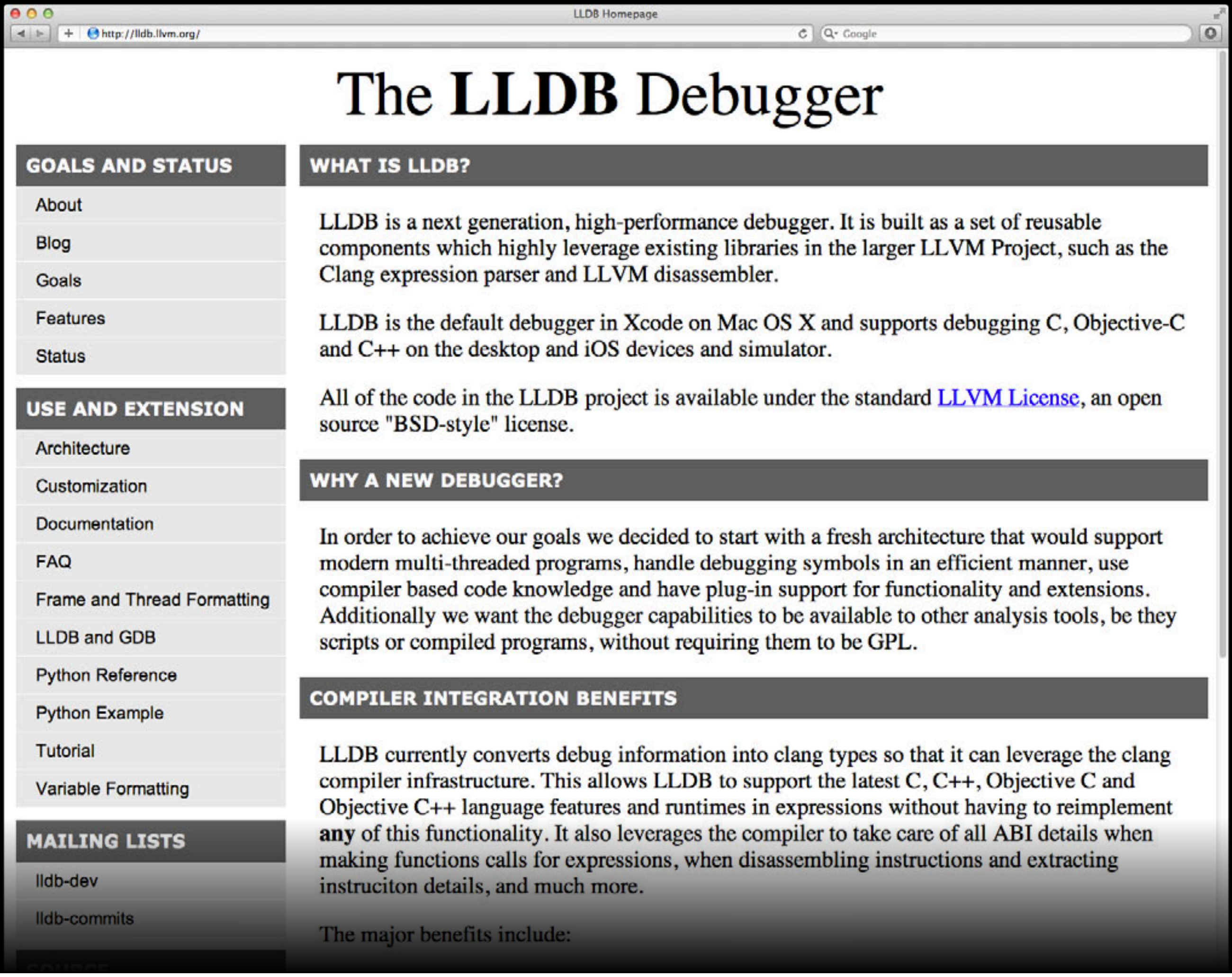
```
(lldb) ls -lAF /tmp/
```

Using New Python Command

```
(lldb) ls -lAF /tmp/
total 0
drwx------  3 root        wheel  102 May 11 09:40 launch-
drwx------  3 root        wheel  102 May 22 15:46 launch-
drwx------  3 root        wheel  102 May 13 14:34 launch-
drwx------  3 root        wheel  102 May 21 19:45 launch-
drwx------  3 local       wheel  102 May 21 19:31 launch-
drwx------  3 local       wheel  102 May 20 20:15 launch-
drwx------  3 local       wheel  102 May 11 19:37 launch-
drwx------  3 root        wheel  102 May 20 11:24 launch-
drwx------  3 root        wheel  102 May 11 09:40 launch-
drwx------  3 root        wheel  102 May 13 14:34 launch-
drwx------  3 root        wheel  102 May 12 19:17 launch-
drwx------  3 local       wheel  102 May 21 19:31 launch-
drwx------  3 local       wheel  102 May 22 14:03 launch-
drwx------  3 local       wheel  102 May 20 10:53 launch-
drwx------  3 root        wheel  102 May 13 19:26 launch-
```

Python Reference

Python Reference



LLDB Homepage

http://lldb.llvm.org/

The **LLDB** Debugger

GOALS AND STATUS

- About
- Blog
- Goals
- Features
- Status

USE AND EXTENSION

- Architecture
- Customization
- Documentation
- FAQ
- Frame and Thread Formatting
- LLDB and GDB
- Python Reference
- Python Example
- Tutorial
- Variable Formatting

MAILING LISTS

- lldb-dev
- lldb-commits

WHAT IS LLDB?

LLDB is a next generation, high-performance debugger. It is built as a set of reusable components which highly leverage existing libraries in the larger LLVM Project, such as the Clang expression parser and LLVM disassembler.

LLDB is the default debugger in Xcode on Mac OS X and supports debugging C, Objective-C and C++ on the desktop and iOS devices and simulator.

All of the code in the LLDB project is available under the standard LLVM License, an open source "BSD-style" license.

WHY A NEW DEBUGGER?

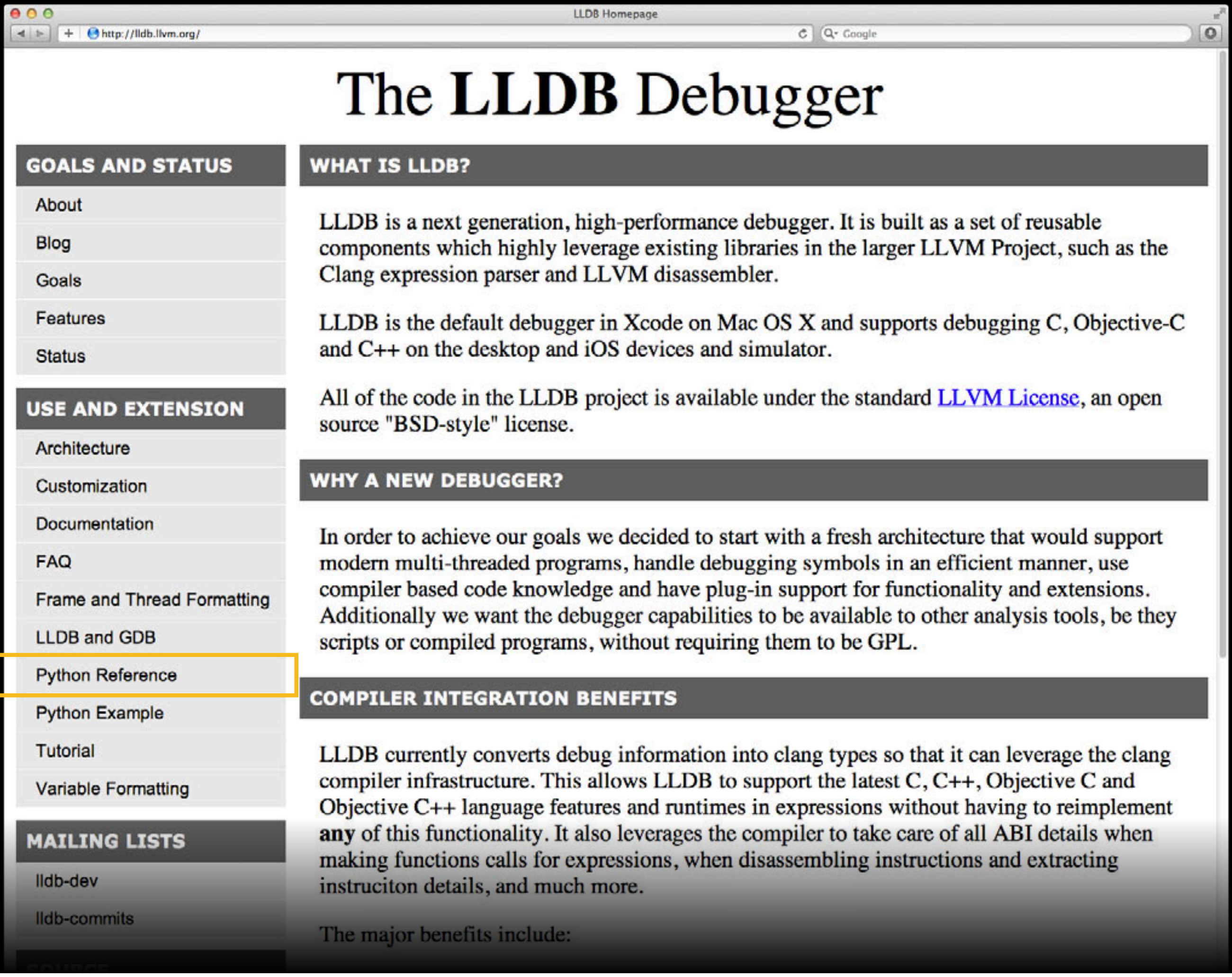
In order to achieve our goals we decided to start with a fresh architecture that would support modern multi-threaded programs, handle debugging symbols in an efficient manner, use compiler based code knowledge and have plug-in support for functionality and extensions. Additionally we want the debugger capabilities to be available to other analysis tools, be they scripts or compiled programs, without requiring them to be GPL.

COMPILER INTEGRATION BENEFITS

LLDB currently converts debug information into clang types so that it can leverage the clang compiler infrastructure. This allows LLDB to support the latest C, C++, Objective C and Objective C++ language features and runtimes in expressions without having to reimplement **any** of this functionality. It also leverages the compiler to take care of all ABI details when making functions calls for expressions, when disassembling instructions and extracting instruciton details, and much more.

The major benefits include:

Python Reference


LLDB Homepage

http://lldb.llvm.org/

The LLDB Debugger

GOALS AND STATUS

- About
- Blog
- Goals
- Features
- Status

USE AND EXTENSION

- Architecture
- Customization
- Documentation
- FAQ
- Frame and Thread Formatting
- LLDB and GDB
- Python Reference
- Python Example
- Tutorial
- Variable Formatting

MAILING LISTS

- lldb-dev
- lldb-commits

WHAT IS LLDB?

LLDB is a next generation, high-performance debugger. It is built as a set of reusable components which highly leverage existing libraries in the larger LLVM Project, such as the Clang expression parser and LLVM disassembler.

LLDB is the default debugger in Xcode on Mac OS X and supports debugging C, Objective-C and C++ on the desktop and iOS devices and simulator.

All of the code in the LLDB project is available under the standard LLVM License, an open source "BSD-style" license.

WHY A NEW DEBUGGER?

In order to achieve our goals we decided to start with a fresh architecture that would support modern multi-threaded programs, handle debugging symbols in an efficient manner, use compiler based code knowledge and have plug-in support for functionality and extensions. Additionally we want the debugger capabilities to be available to other analysis tools, be they scripts or compiled programs, without requiring them to be GPL.

COMPILER INTEGRATION BENEFITS

LLDB currently converts debug information into clang types so that it can leverage the clang compiler infrastructure. This allows LLDB to support the latest C, C++, Objective C and Objective C++ language features and runtimes in expressions without having to reimplement **any** of this functionality. It also leverages the compiler to take care of all ABI details when making functions calls for expressions, when disassembling instructions and extracting instruciton details, and much more.

The major benefits include:

Initialization Files

~/.lldbinit

Initialization Files

~/.lldbinit

- Add commands to be executed just after LLDB launches

Initialization Files

~/.lldbinit

- Add commands to be executed just after LLDB launches
- Common uses include
    - Aliases
    - Default settings
    - Type formatting
    - Importing Python modules

Initialization Files

~/.lldbinit

- Add commands to be executed just after LLDB launches
- Common uses include
  - Aliases
  - Default settings
  - Type formatting
  - Importing Python modules
  - Bind Python commands

Initialization Files Rules

.lldbinit with “Xcode”

Initialization Files Rules

.lldbinit with “Xcode”

1 Default initialization file

`~/.lldbinit-Xcode`

Initialization Files Rules

.lldbinit with “Xcode”

1 Default initialization file

`~/.lldbinit`

Initialization Files Rules

.lldbinit with “Xcode”

1. Default initialization file
2. Load the executable

```
~/.lldbinit
```

Initialization Files Rules

.lldbinit with “lldb”

Initialization Files Rules

.lldbinit with “lldb”

```
% cd /tmp
```

Initialization Files Rules

.lldbinit with “lldb”

```
% cd /tmp
% xcrun lldb /bin/ls
```

Initialization Files Rules

.lldbinit with “lldb”

```
% cd /tmp
% xcrun lldb /bin/ls
(lldb)
```

1 **Default initialization file** `~/.lldbinit-lldb`

Initialization Files Rules

.lldbinit with "lldb"

```
% cd /tmp
% xcrun lldb /bin/ls
(lldb)
```

1 Default initialization file `~/.lldbinit`

Initialization Files Rules

.lldbinit with “lldb”

```
% cd /tmp
% xcrun lldb /bin/ls
(lldb)
```

1 Default initialization file `~/.lldbinit`

Initialization Files Rules

.lldbinit with "lldb"

```
% cd /tmp
% xcrun lldb /bin/ls
(lldb)
```

1. Default initialization file `~/.lldbinit`
2. Load the file "/bin/ls"

Initialization Files Rules

.lldbinit with "lldb"

```
% cd /tmp
% xcrun lldb /bin/ls
(lldb)
```

1. Default initialization file `~/.lldbinit`
2. Load the file "/bin/ls"
3. Current working directory

Initialization Files Rules

.lldbinit with "lldb"

```
% cd /tmp
% xcrun lldb /bin/ls
(lldb)
```

1. Default initialization file `~/.lldbinit`
2. Load the file "/bin/ls"
3. Current working directory `/tmp/.lldbinit`

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

Launching Programs

Arguments

Launching Programs

Arguments

```
% xcrun lldb print-args 1 2 3
```

Launching Programs

Arguments

```
% xcrun lldb print-args 1 2 3
(lldb) run
```

Launching Programs

Arguments

```
% xcrun lldb print-args 1 2 3
(lldb) run
argv[ 0] = '/tmp/print-args'
argv[ 1] = '1'
argv[ 2] = '2'
argv[ 3] = '3'
```

Launching Programs

Arguments

```
% xcrun lldb print-args 1 2 3
(lldb) run
argv[ 0] = '/tmp/print-args'
argv[ 1] = '1'
argv[ 2] = '2'
argv[ 3] = '3'
(lldb) run 4 5 6
```

Launching Programs

Arguments

```
% xcrun lldb print-args 1 2 3
(lldb) run
argv[ 0] = '/tmp/print-args'
argv[ 1] = '1'
argv[ 2] = '2'
argv[ 3] = '3'
(lldb) run 4 5 6
argv[ 0] = '/tmp/print-args'
argv[ 1] = '4'
argv[ 2] = '5'
argv[ 3] = '6'
```

Launching Programs

Setting environment variables

Launching Programs

Setting environment variables

```
% export MallocStackLogging=1
```

Launching Programs

Setting environment variables

```
% export MallocStackLogging=1
% xcrun lldb /bin/cat
```

Launching Programs

Setting environment variables

```
% export MallocStackLogging=1
% xcrun lldb /bin/cat
(lldb) process launch
```

Launching Programs

Setting environment variables

```
% export MallocStackLogging=1
% xcrun lldb /bin/cat
(lldb) process launch
^C
```

Launching Programs

Setting environment variables

```
% export MallocStackLogging=1
% xcrun lldb /bin/cat
(lldb) process launch
^C
(lldb) p (char *)getenv("MallocStackLogging")
(const char *) $2 = 0x00007fff5fbffa21 "1"
```

Launching Programs

Setting environment variables

Launching Programs

Setting environment variables

```
% xcrun lldb /bin/cat
```

Launching Programs

Setting environment variables

```
% xcrun lldb /bin/cat
(lldb) process launch --environment MallocStackLogging=1
```

Launching Programs

Setting environment variables

```
% xcrun lldb /bin/cat
(lldb) process launch --environment MallocStackLogging=1
(lldb) process launch -v MallocStackLogging=1
```

Launching Programs

Setting environment variables

```
% xcrun lldb /bin/cat
(lldb) process launch --environment MallocStackLogging=1
(lldb) process launch -v MallocStackLogging=1
                      -v MallocStackLoggingNoCompact=2
```

Launching Programs

Launch in terminal

Launching Programs

Launch in terminal

```
(lldb) target create /bin/cat
```

Launching Programs

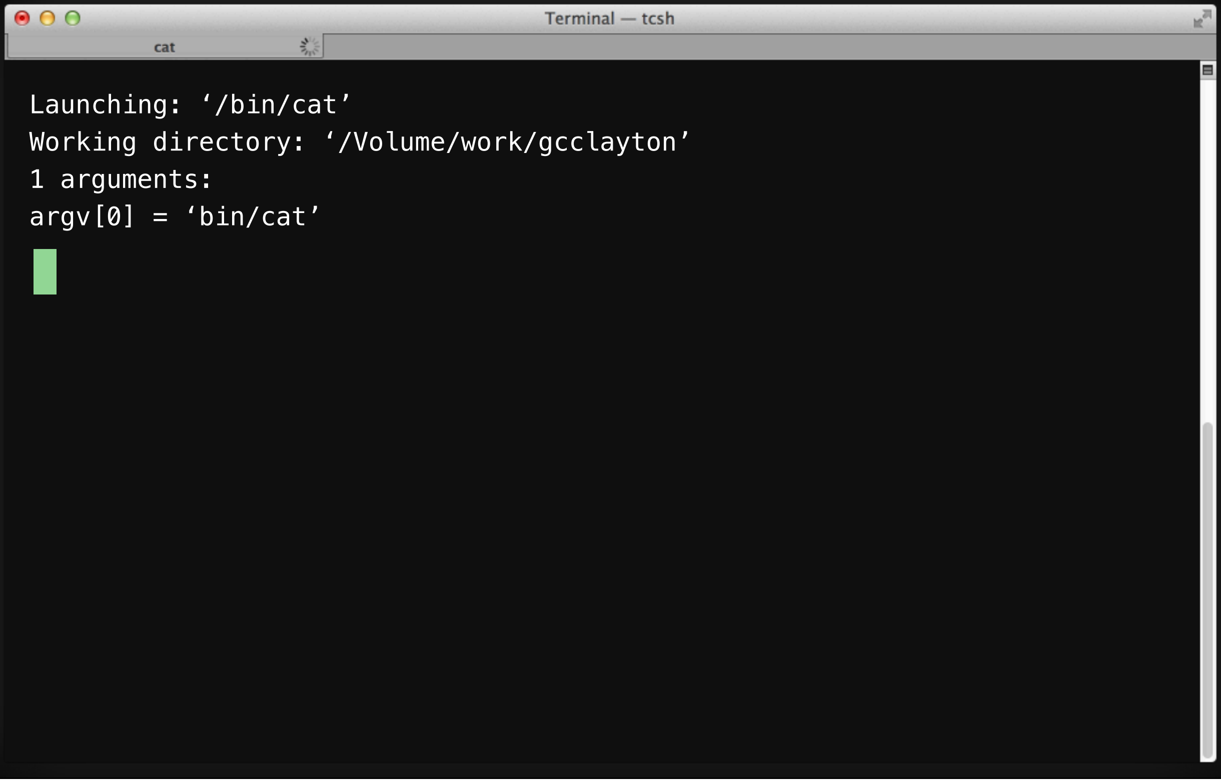
Launch in terminal

```
(lldb) target create /bin/cat
(lldb) process launch --tty
```

Launching Programs

Launch in terminal

```
(lldb) target create /bin/cat
(lldb) process launch --tty
```



```
Launching: ‘/bin/cat’
Working directory: ‘/Volume/work/gcclayton’
1 arguments:
argv[0] = ‘bin/cat’
```

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

LLDB in Depth

LLDB

- Getting started
- Terminology
- Customizing commands
- Launching programs
- Debug session

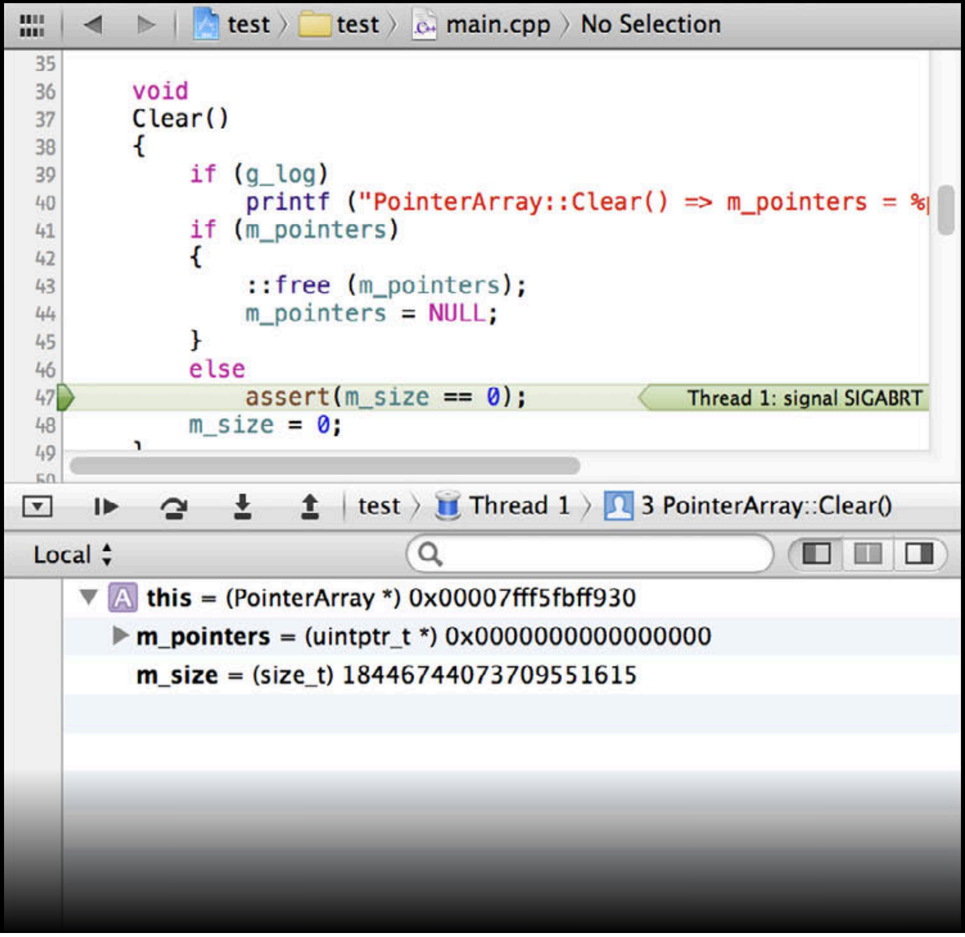
Debug Session

Example class

```
class PointerArray
 {
 protected:
      uintptr_t * m_pointers;
      size_t      m_size;
 };
```

Debug Session

Bug detected


test 〉 test 〉 main.cpp 〉 No Selection

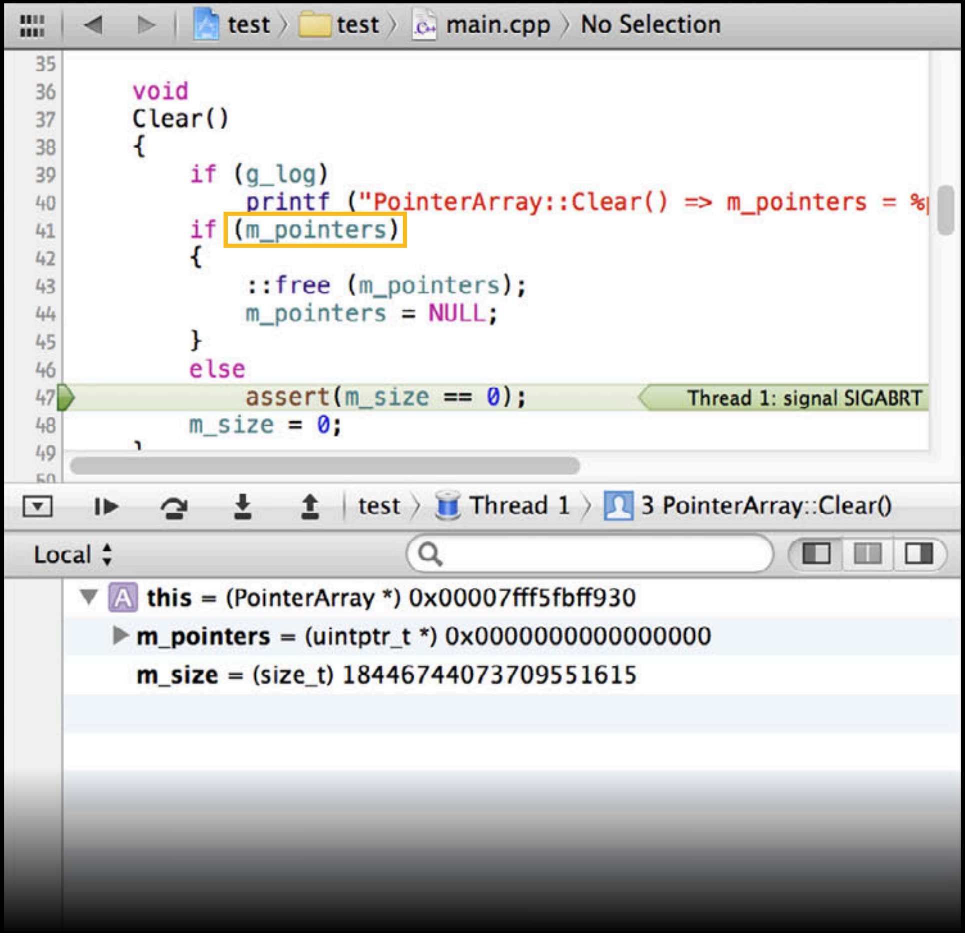
```
    void
    Clear()
    {
        if (g_log)
            printf ("PointerArray::Clear() => m_pointers = %
        if (m_pointers)
        {
            ::free (m_pointers);
            m_pointers = NULL;
        }
        else
            assert(m_size == 0);                    Thread 1: signal SIGABRT
        m_size = 0;
```

test 〉 Thread 1 〉 3 PointerArray::Clear()
Local

```
▼ this = (PointerArray *) 0x00007fff5fbff930
  ▶ m_pointers = (uintptr_t *) 0x0000000000000000
    m_size = (size_t) 18446744073709551615
```

Debug Session

Bug detected

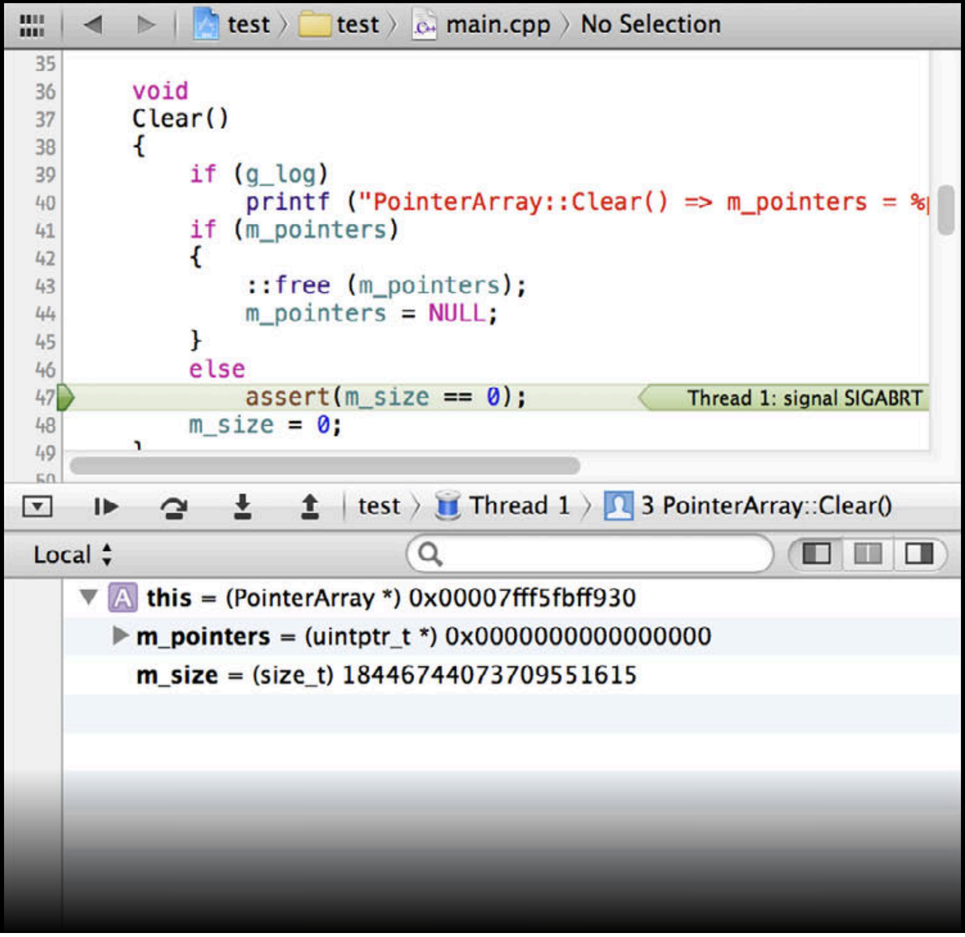

test › test › main.cpp › No Selection
test › Thread 1 › 3 PointerArray::Clear()
Local

```
    void
    Clear()
    {
        if (g_log)
            printf ("PointerArray::Clear() => m_pointers = %
        if (m_pointers)
        {
            ::free (m_pointers);
            m_pointers = NULL;
        }
        else
            assert(m_size == 0);          Thread 1: signal SIGABRT
        m_size = 0;
```

```
▼ this = (PointerArray *) 0x00007fff5fbff930
  ▶ m_pointers = (uintptr_t *) 0x0000000000000000
    m_size = (size_t) 18446744073709551615
```

Debug Session

Bug detected



test 〉 test 〉 main.cpp 〉 No Selection

```
    void
    Clear()
    {
        if (g_log)
            printf ("PointerArray::Clear() => m_pointers = %
        if (m_pointers)
        {
            ::free (m_pointers);
            m_pointers = NULL;
        }
        else
            assert(m_size == 0);
        m_size = 0;
```

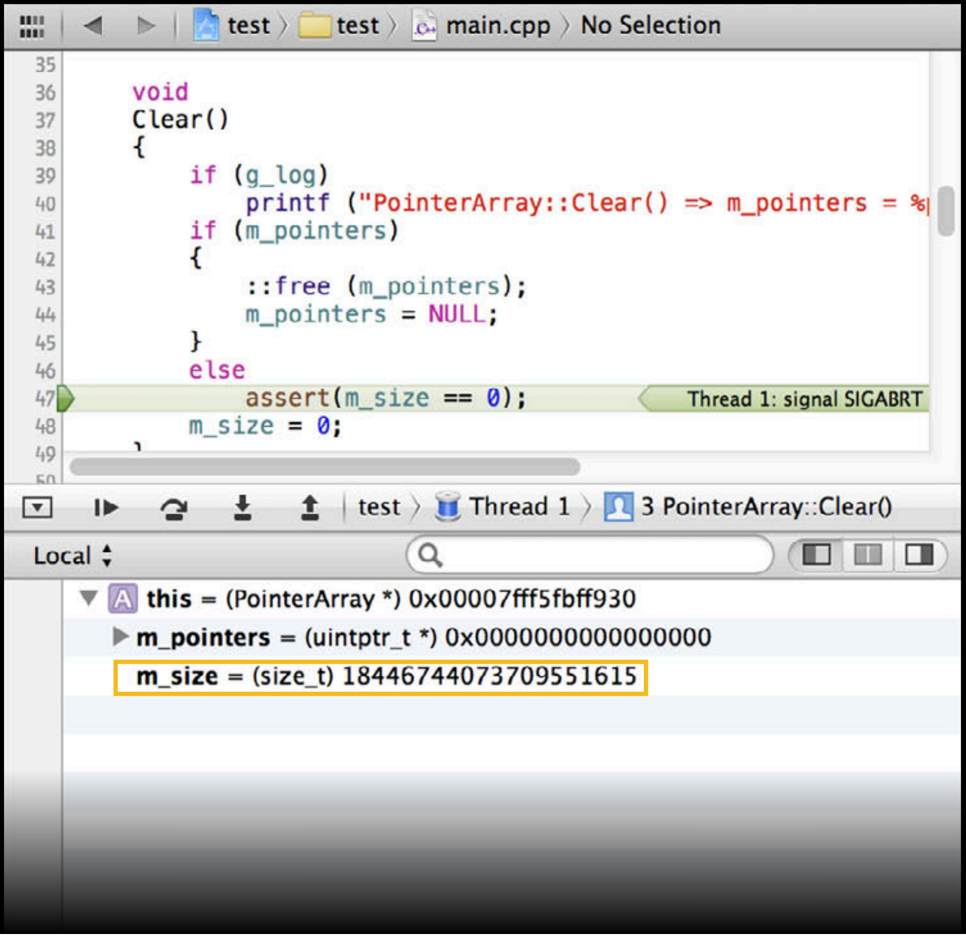
Thread 1: signal SIGABRT

test 〉 Thread 1 〉 3 PointerArray::Clear()
Local

```
▼ this = (PointerArray *) 0x00007fff5fbff930
  ▶ m_pointers = (uintptr_t *) 0x0000000000000000
    m_size = (size_t) 18446744073709551615
```

Debug Session

Bug detected



test 〉 test 〉 main.cpp 〉 No Selection

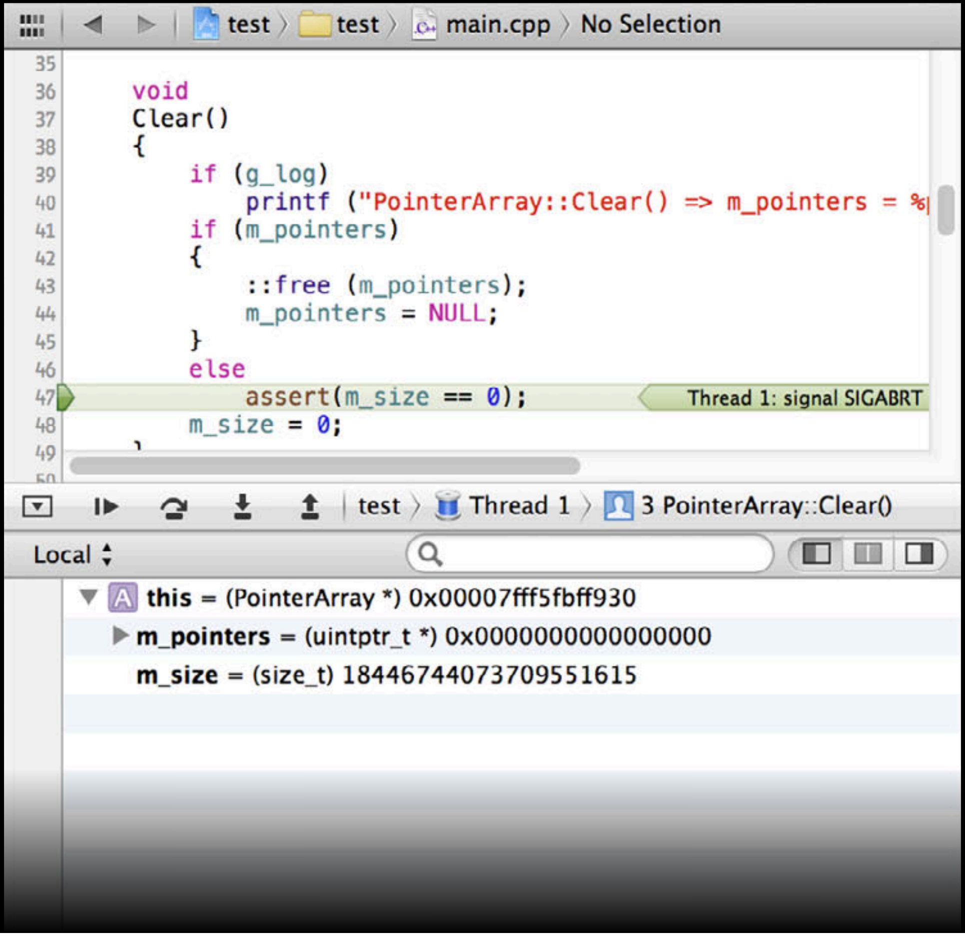
```
    void
    Clear()
    {
        if (g_log)
            printf ("PointerArray::Clear() => m_pointers = %
        if (m_pointers)
        {
            ::free (m_pointers);
            m_pointers = NULL;
        }
        else
            assert(m_size == 0);              Thread 1: signal SIGABRT
        m_size = 0;
```

test 〉 Thread 1 〉 3 PointerArray::Clear()
Local

```
▼ this = (PointerArray *) 0x00007fff5fbff930
  ▶ m_pointers = (uintptr_t *) 0x0000000000000000
    m_size = (size_t) 18446744073709551615
```

Debug Session

Bug detected

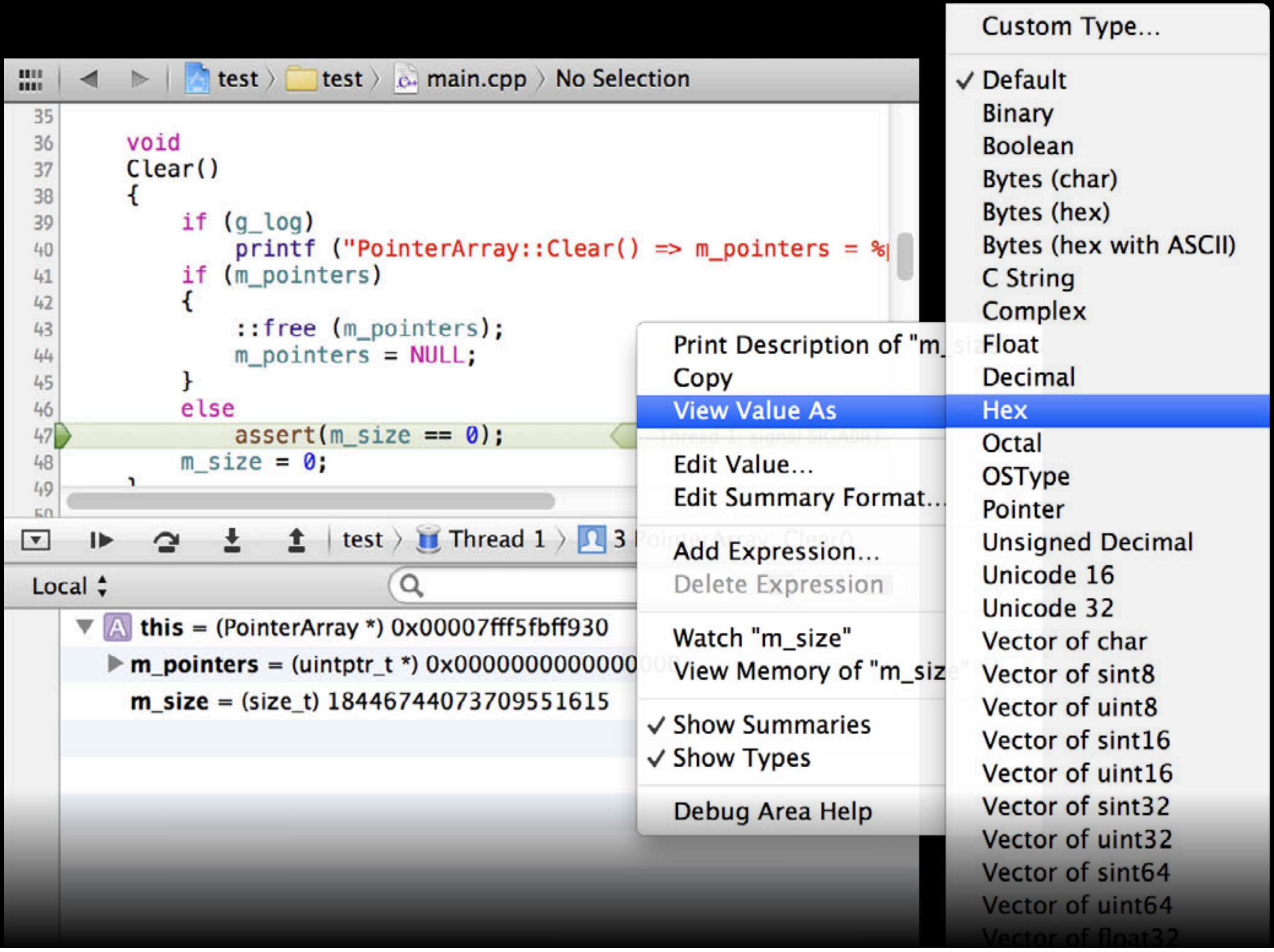


```
    void
    Clear()
    {
        if (g_log)
            printf ("PointerArray::Clear() => m_pointers = %
        if (m_pointers)
        {
            ::free (m_pointers);
            m_pointers = NULL;
        }
        else
            assert(m_size == 0);          Thread 1: signal SIGABRT
        m_size = 0;
```

```
▼ this = (PointerArray *) 0x00007fff5fbff930
  ▶ m_pointers = (uintptr_t *) 0x0000000000000000
    m_size = (size_t) 18446744073709551615
```

Debug Session

Bug detected



```
void
Clear()
{
    if (g_log)
        printf ("PointerArray::Clear() => m_pointers = %
    if (m_pointers)
    {
        ::free (m_pointers);
        m_pointers = NULL;
    }
    else
        assert(m_size == 0);
    m_size = 0;
```

```
▼ this = (PointerArray *) 0x00007fff5fbff930
  ▶ m_pointers = (uintptr_t *) 0x0000000000000
    m_size = (size_t) 18446744073709551615
```

Formatting Variable Values

Command line

Formatting Variable Values

Command line

```
(lldb) frame variable --format hex this->m_size
```

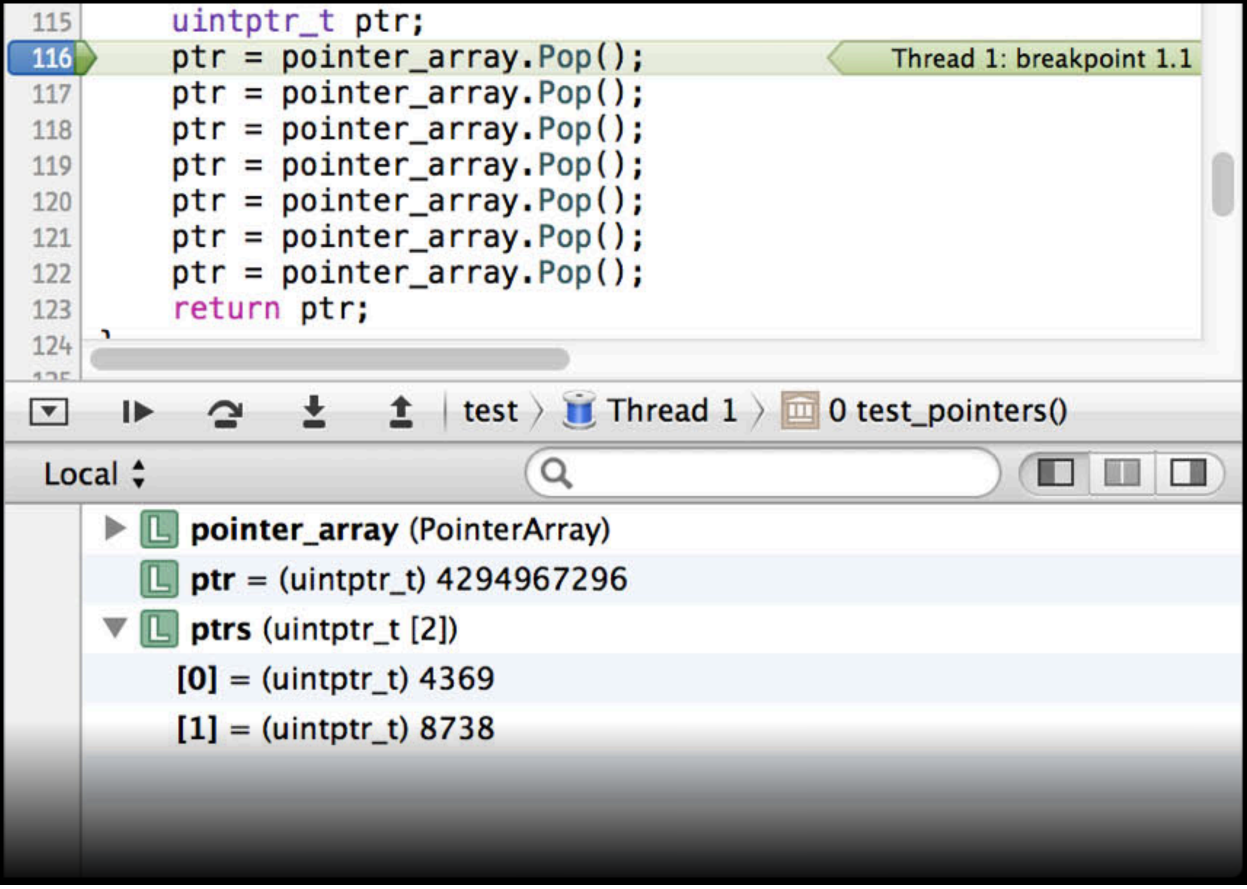
Formatting Variable Values

Command line

```
(lldb) frame variable --format hex this->m_size
(size_t) this->m_size = 0xffffffffffffffff
```

Debug Session

Wrong default formats



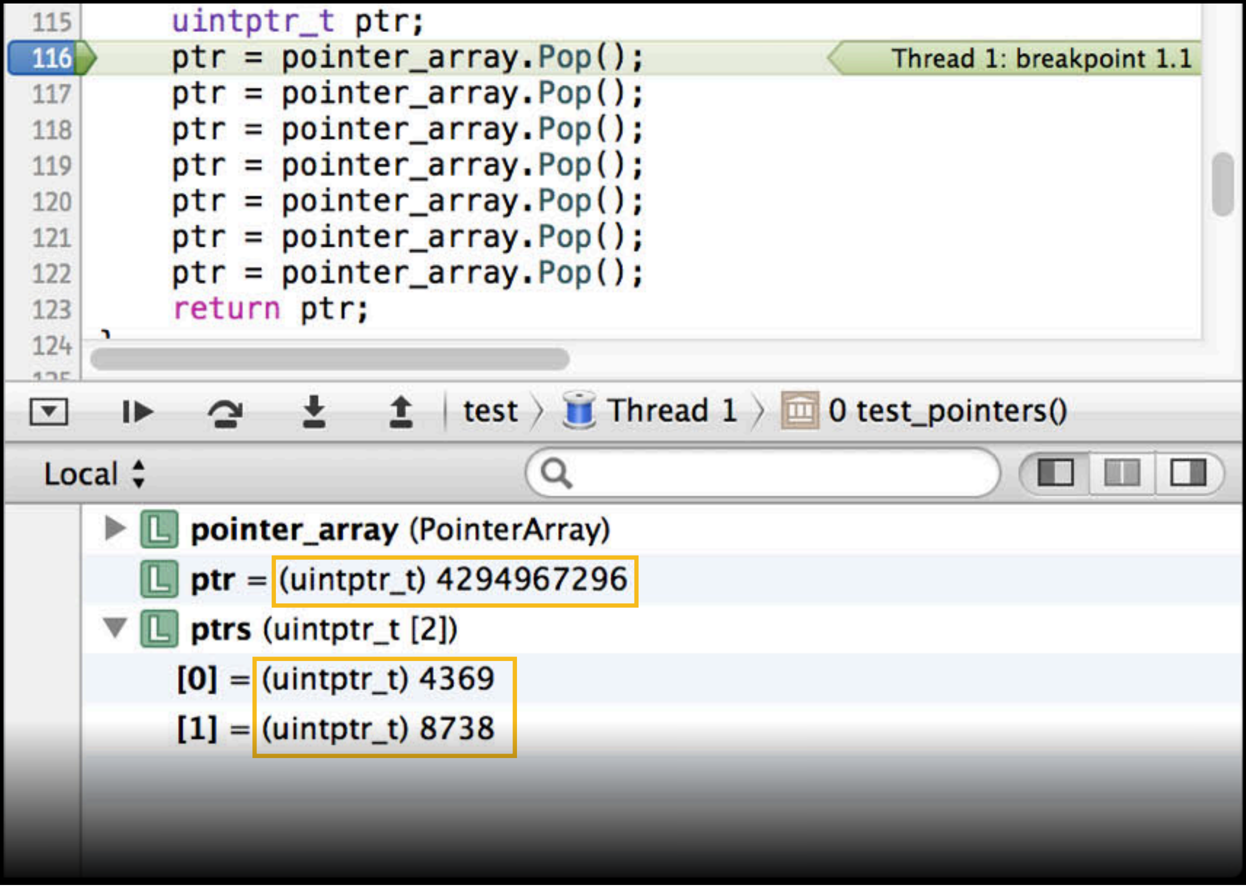
```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
```

Thread 1: breakpoint 1.1

```
pointer_array (PointerArray)
ptr = (uintptr_t) 4294967296
ptrs (uintptr_t [2])
  [0] = (uintptr_t) 4369
  [1] = (uintptr_t) 8738
```

Debug Session

Wrong default formats

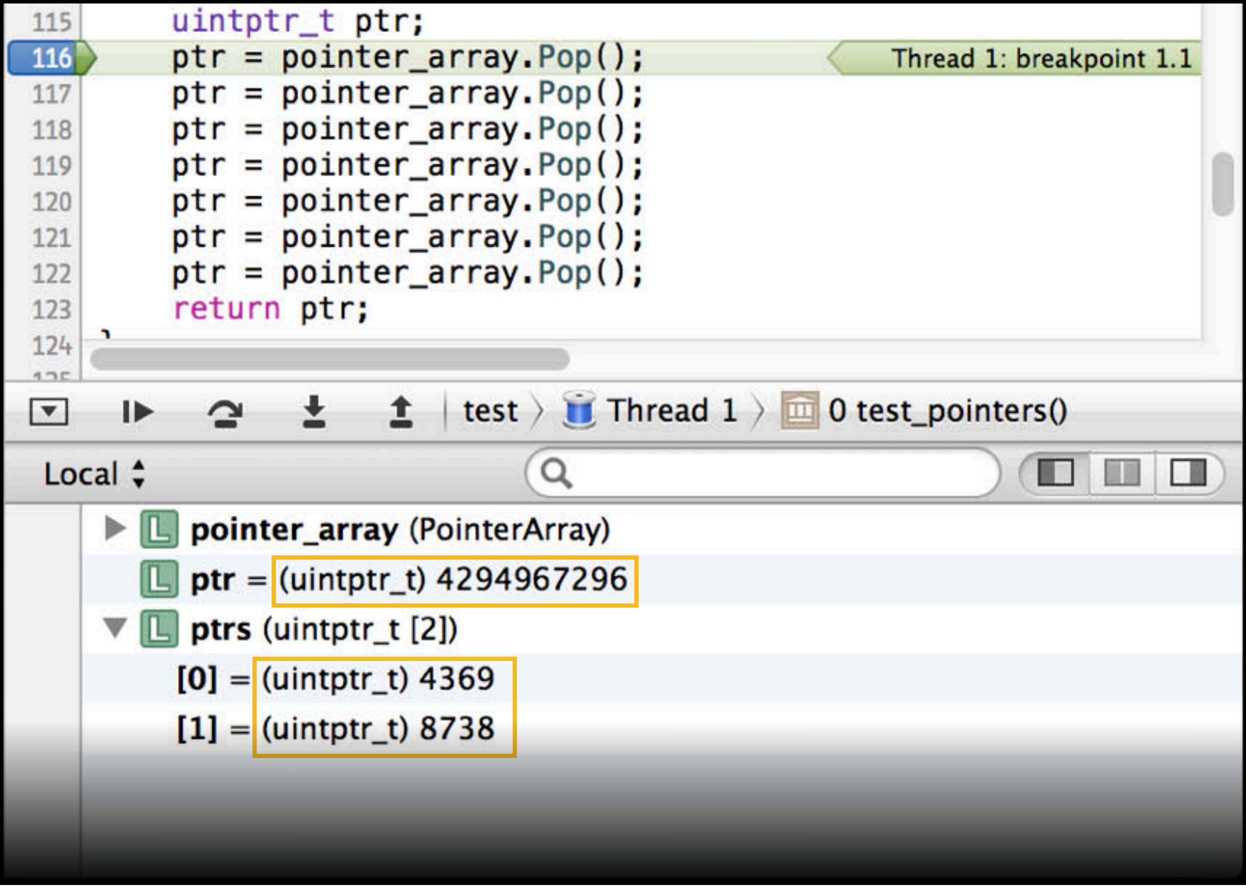


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
}
```

```
▶ pointer_array (PointerArray)
  ptr = (uintptr_t) 4294967296
▼ ptrs (uintptr_t [2])
    [0] = (uintptr_t) 4369
    [1] = (uintptr_t) 8738
```

Debug Session

Wrong default formats



```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
```

```
▶ pointer_array (PointerArray)
  ptr = (uintptr_t) 4294967296
▼ ptrs (uintptr_t [2])
    [0] = (uintptr_t) 4369
    [1] = (uintptr_t) 8738
```

Debug Session

Setting default formats

Debug Session

Setting default formats

```
(lldb) type format add
```

Debug Session

Setting default formats

```
(lldb) type format add
           --format hex
```

Debug Session

Setting default formats

```
(lldb) type format add
           --format hex
           uintptr_t intptr_t off_t
```

Debug Session

Setting default formats

```
(lldb) type format add
           --format hex
           uintptr_t intptr_t off_t
(lldb) print ptr
(uintptr_t) ptr = 0x0000000100000000
```

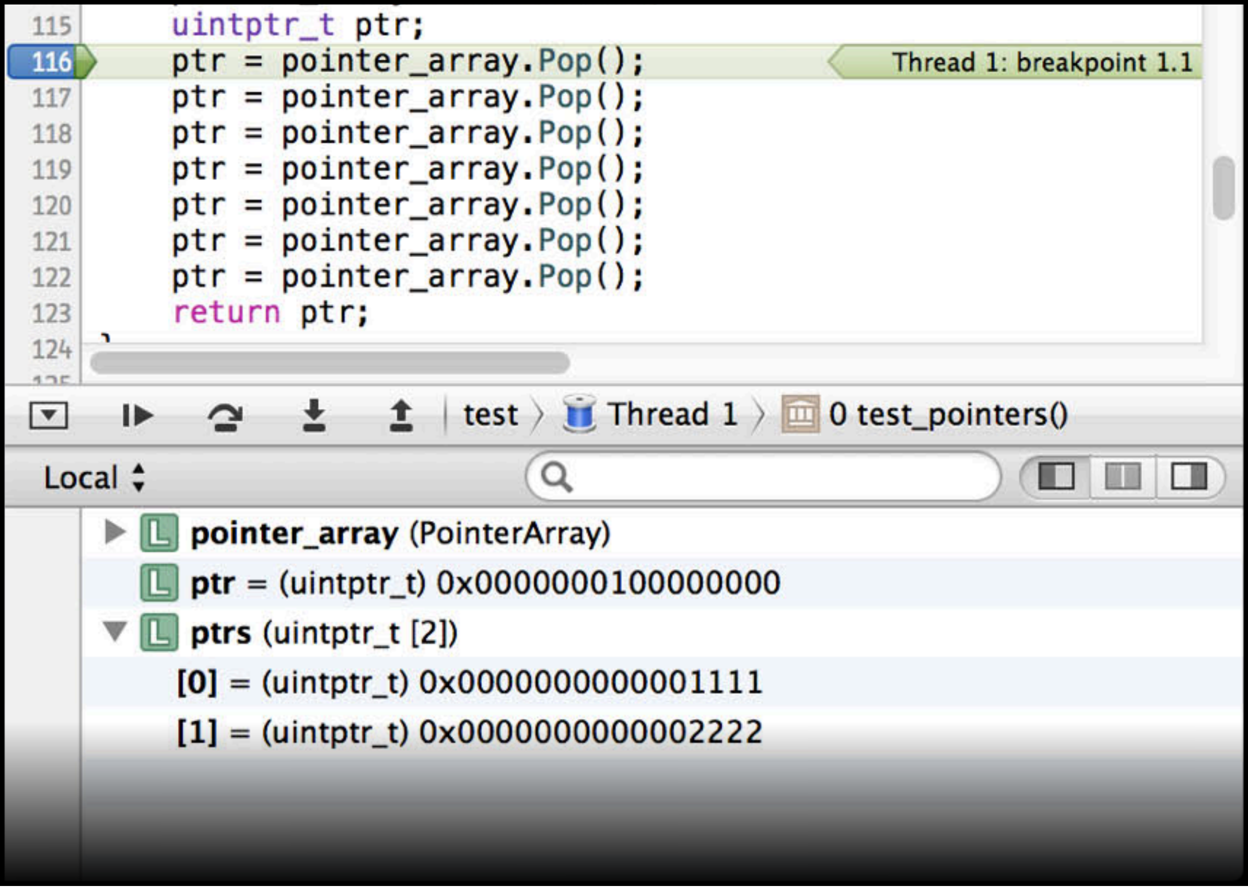
Debug Session

Setting default formats

```
(lldb) type format add
           --format hex
           uintptr_t intptr_t off_t
(lldb) print ptr
(uintptr_t) ptr = 0x0000000100000000
(lldb) print ptrs
(uintptr_t [2]) ptrs = {
  [0] = 0x0000000000001111
  [1] = 0x0000000000002222
}
```

Debug Session

Corrected default formats

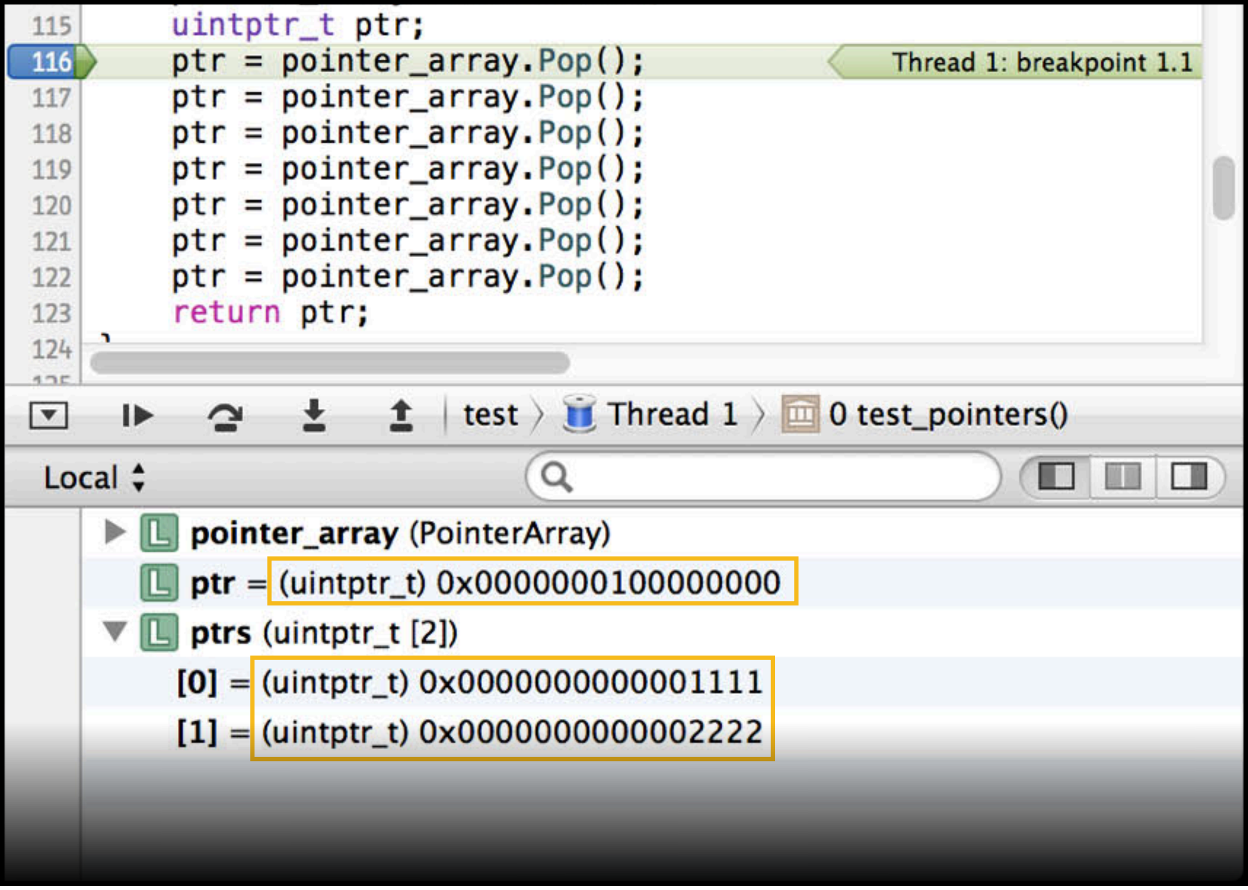


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
```

```
pointer_array (PointerArray)
ptr = (uintptr_t) 0x0000000100000000
ptrs (uintptr_t [2])
  [0] = (uintptr_t) 0x0000000000001111
  [1] = (uintptr_t) 0x0000000000002222
```

Debug Session

Corrected default formats

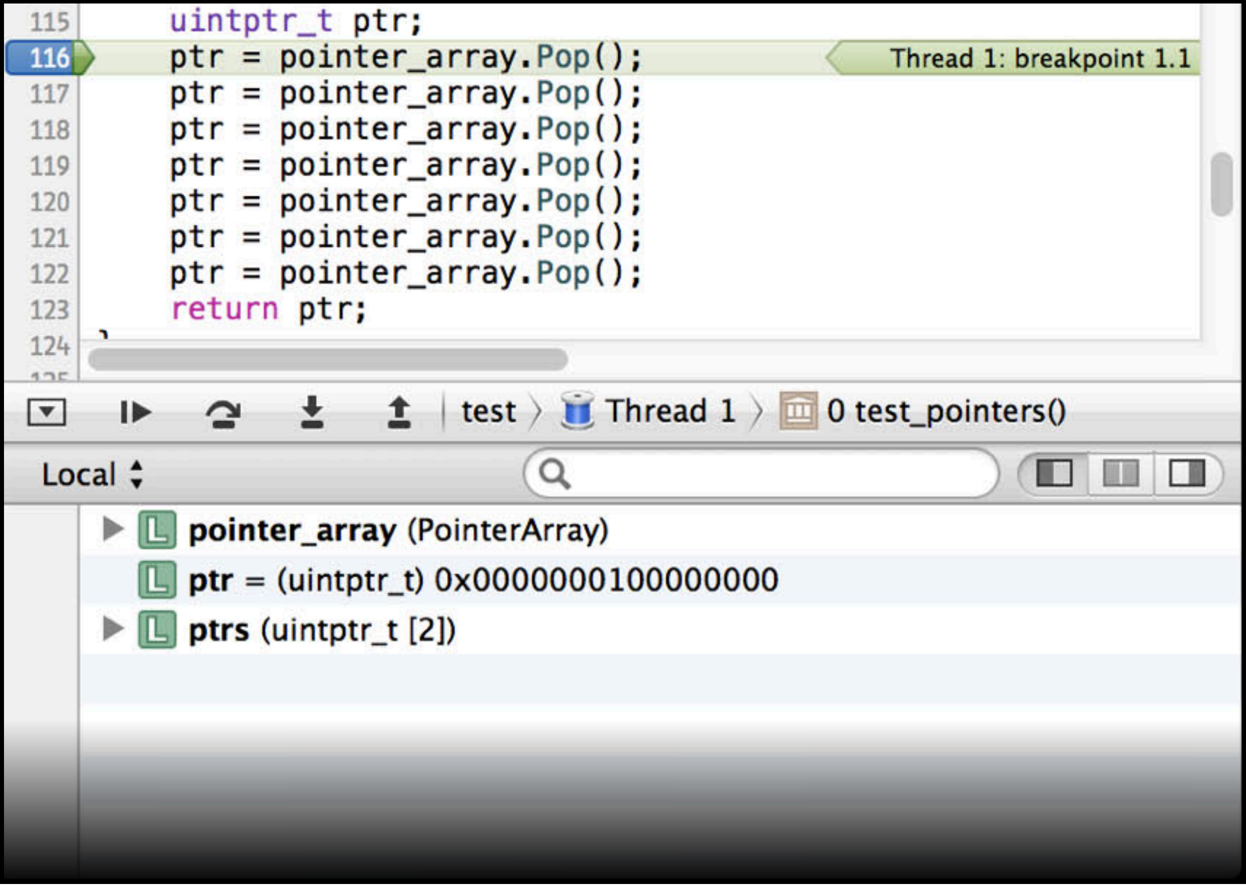


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();                    Thread 1: breakpoint 1.1
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
}
```

```
test > Thread 1 > 0 test_pointers()

Local

▶ pointer_array (PointerArray)
  ptr = (uintptr_t) 0x0000000100000000
▼ ptrs (uintptr_t [2])
    [0] = (uintptr_t) 0x0000000000001111
    [1] = (uintptr_t) 0x0000000000002222
```

Debug Session

Adding type summaries

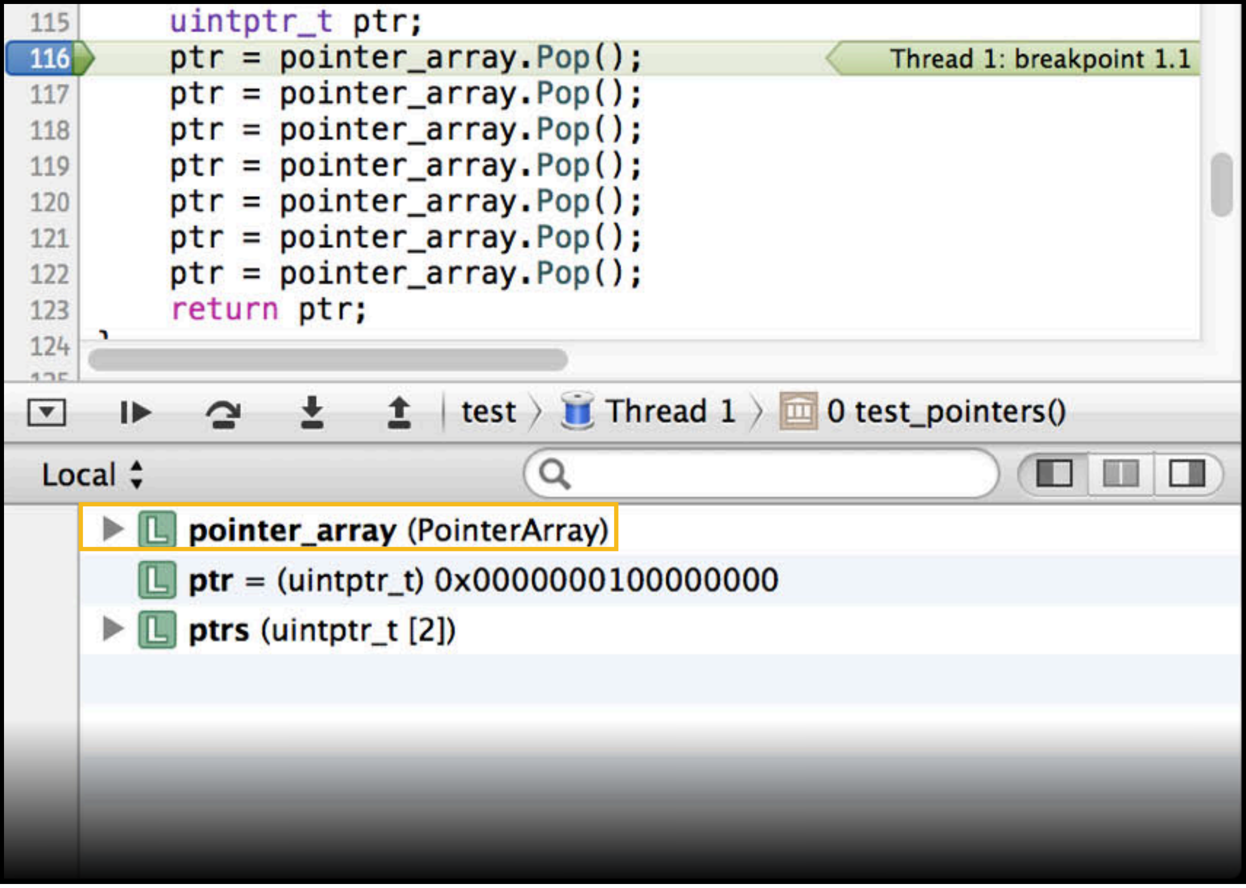


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();          Thread 1: breakpoint 1.1
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
```

```
pointer_array (PointerArray)
ptr = (uintptr_t) 0x0000000100000000
ptrs (uintptr_t [2])
```

Debug Session

Adding type summaries

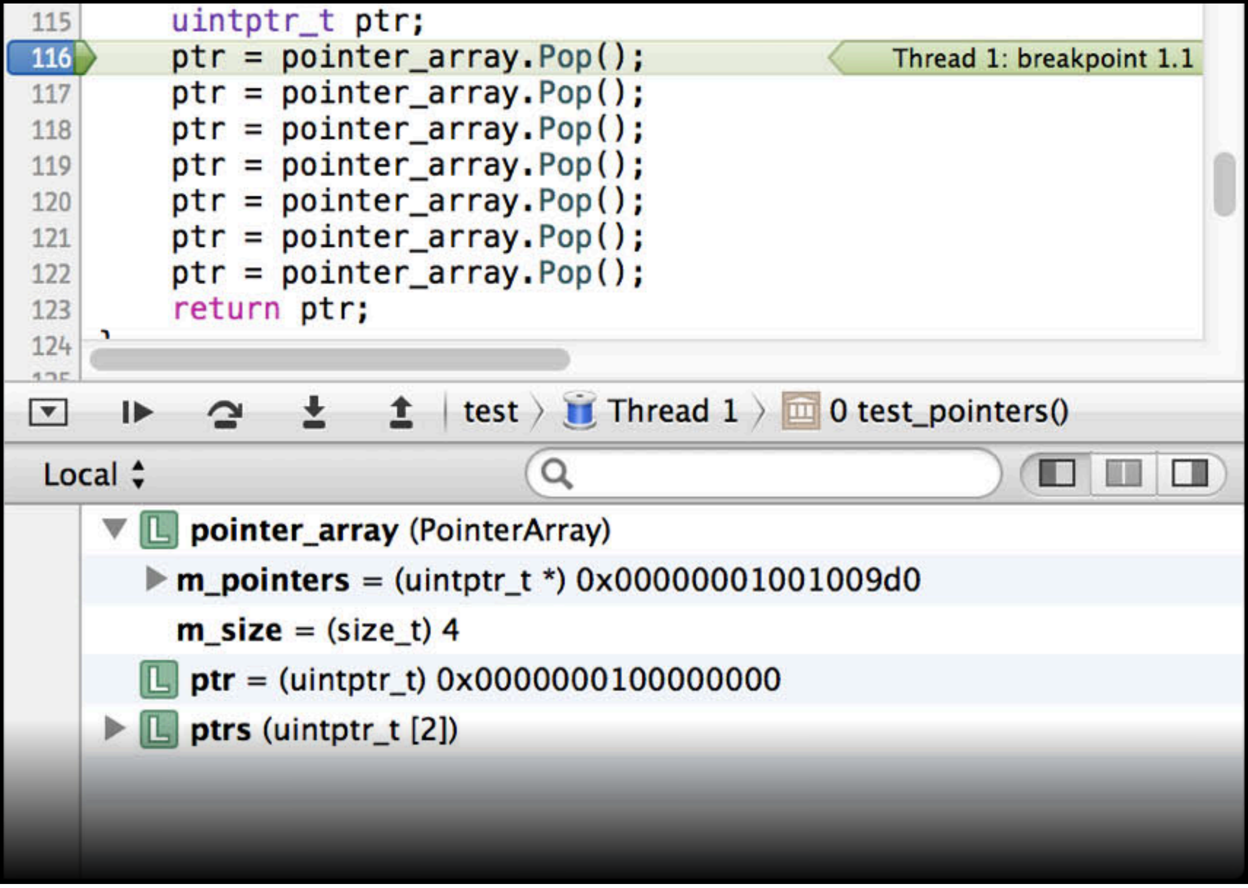


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
}
```

```
pointer_array (PointerArray)
ptr = (uintptr_t) 0x0000000100000000
ptrs (uintptr_t [2])
```

Debug Session

Adding type summaries



```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
}
```

```
▼ pointer_array (PointerArray)
    ▶ m_pointers = (uintptr_t *) 0x00000001001009d0
      m_size = (size_t) 4
  ptr = (uintptr_t) 0x0000000100000000
▶ ptrs (uintptr_t [2])
```

Debug Session

```
type summary add
```

Debug Session

`type summary add`

```
(lldb) type summary add
```

Debug Session

`type summary add`

```
(lldb) type summary add
       -s "size=${var.m_size} ${var.m_pointers}"
```

Debug Session

`type summary add`

```
(lldb) type summary add
       -s "size=${var.m_size} ${var.m_pointers}"
       PointerArray
```

Debug Session

`type summary add`

```
(lldb) type summary add
       -s "size=${var.m_size} ${var.m_pointers}"
       PointerArray
(lldb) frame variable pointer_array
(PointerArray) pointer_array = size=1 0x100100990
```

Type Summary Strings

Summary string syntax

Type Summary Strings

Summary string syntax

- String can contain `plain text` and `variables` and `formats`

Type Summary Strings

Summary string syntax

- String can contain `plain text` and `variables` and `formats`
- Variables references

```
${var[path][%<format>]}
```

Type Summary Strings

Summary string syntax

- String can contain `plain text` and `variables` and `formats`
- Variables references

```
${var[path][%<format>]}
```

- `Formats` characters can be listed with:

```
(lldb) help <format>
```

Type Summary Strings

Summary string syntax

- String can contain `plain text` and `variables` and `formats`
- Variables references

```
${var[path][%<format>]}
```

- `Formats` characters can be listed with:

```
(lldb) help <format>
```

- Example summary strings

Type Summary Strings

Summary string syntax

- String can contain `plain text` and `variables` and `formats`
- Variables references

```
    ${var[path][%<format>]}
```

- `Formats` characters can be listed with:

```
    (lldb) help <format>
```

- Example summary strings

```
    "natural = ${var}, octal = ${var%o} , hex = ${var%x}"
```

Type Summary Strings

Summary string syntax

- String can contain `plain text` and `variables` and `formats`
- Variables references

```
${var[path][%<format>]}
```

- `Formats` characters can be listed with:

```
(lldb) help <format>
```

- Example summary strings

```
“natural = ${var}, octal = ${var%o} , hex = ${var%x}”
“( x = ${var.x}, y = ${var.y} )”
```

Type Summary Strings

Summary string syntax

- String can contain `plain text` and `variables` and `formats`
- Variables references

```
${var[path][%<format>]}
```

- `Formats` characters can be listed with:

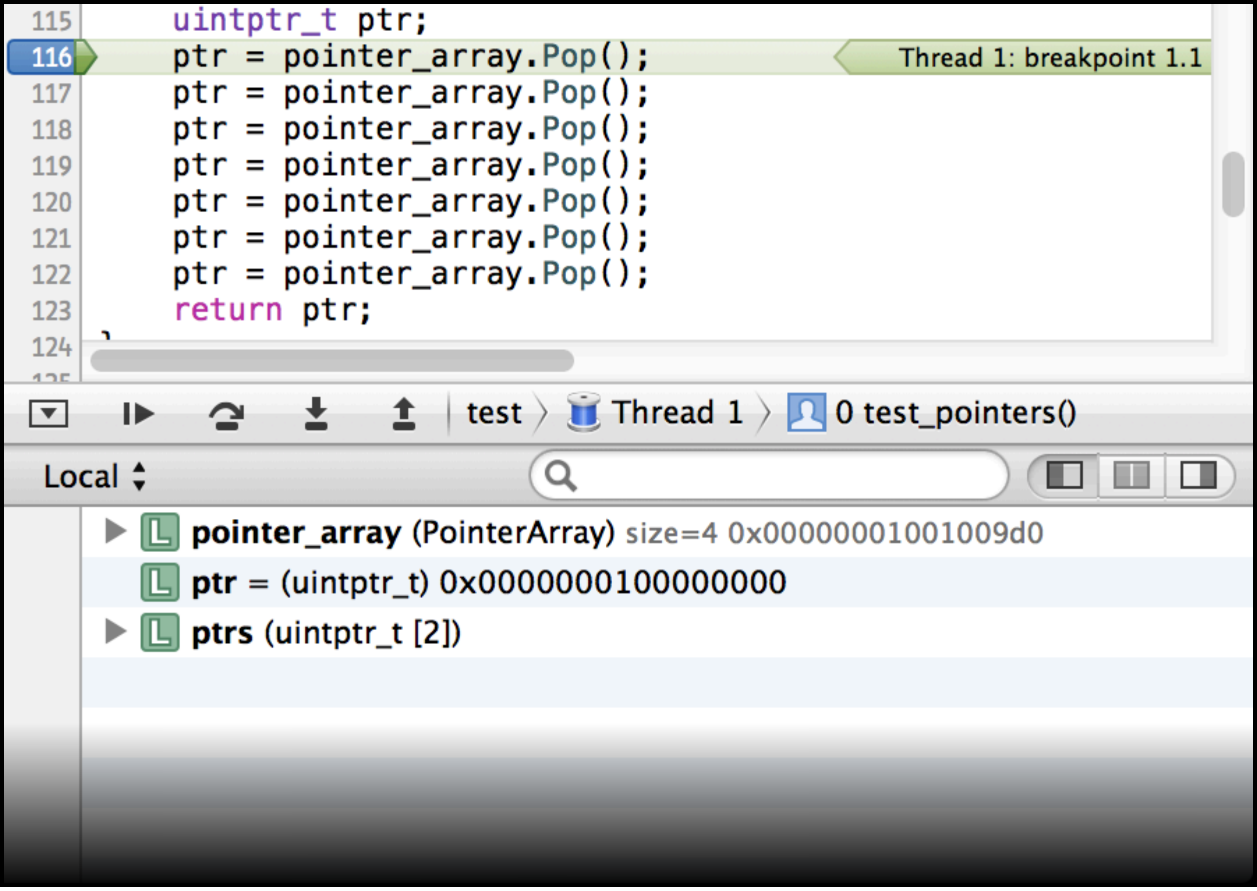
```
(lldb) help <format>
```

- Example summary strings

```
“natural = ${var}, octal = ${var%o} , hex = ${var%x}”
“( x = ${var.x}, y = ${var.y} )”
“string = ${var._M_dataplus._M_p%s}”
```

Debug Session

Type summaries in Xcode



```
    uintptr_t ptr;
    ptr = pointer_array.Pop();                    Thread 1: breakpoint 1.1
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
}
```

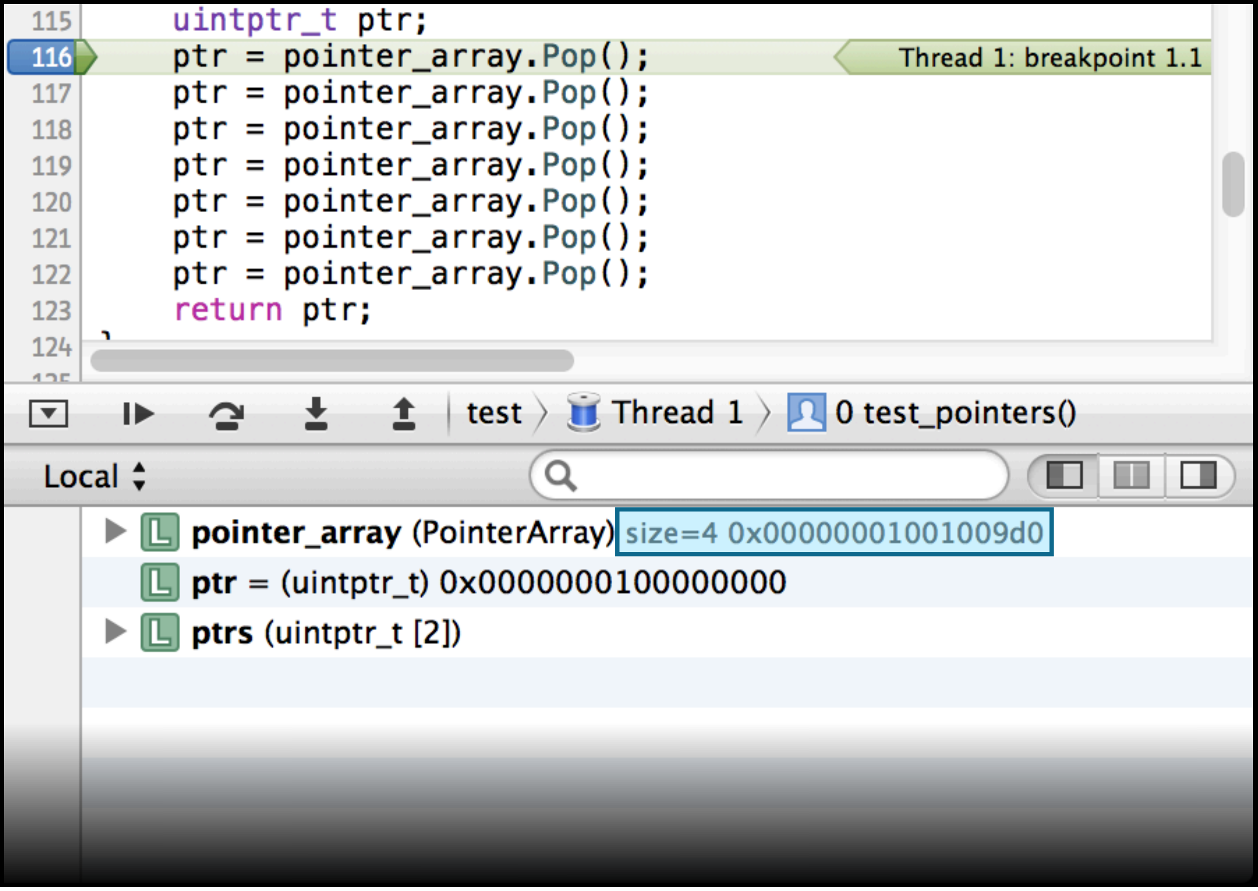
test 〉 Thread 1 〉 0 test_pointers()

Local

```
▶ pointer_array (PointerArray) size=4 0x00000001001009d0
  ptr = (uintptr_t) 0x0000000100000000
▶ ptrs (uintptr_t [2])
```

Debug Session

Type summaries in Xcode

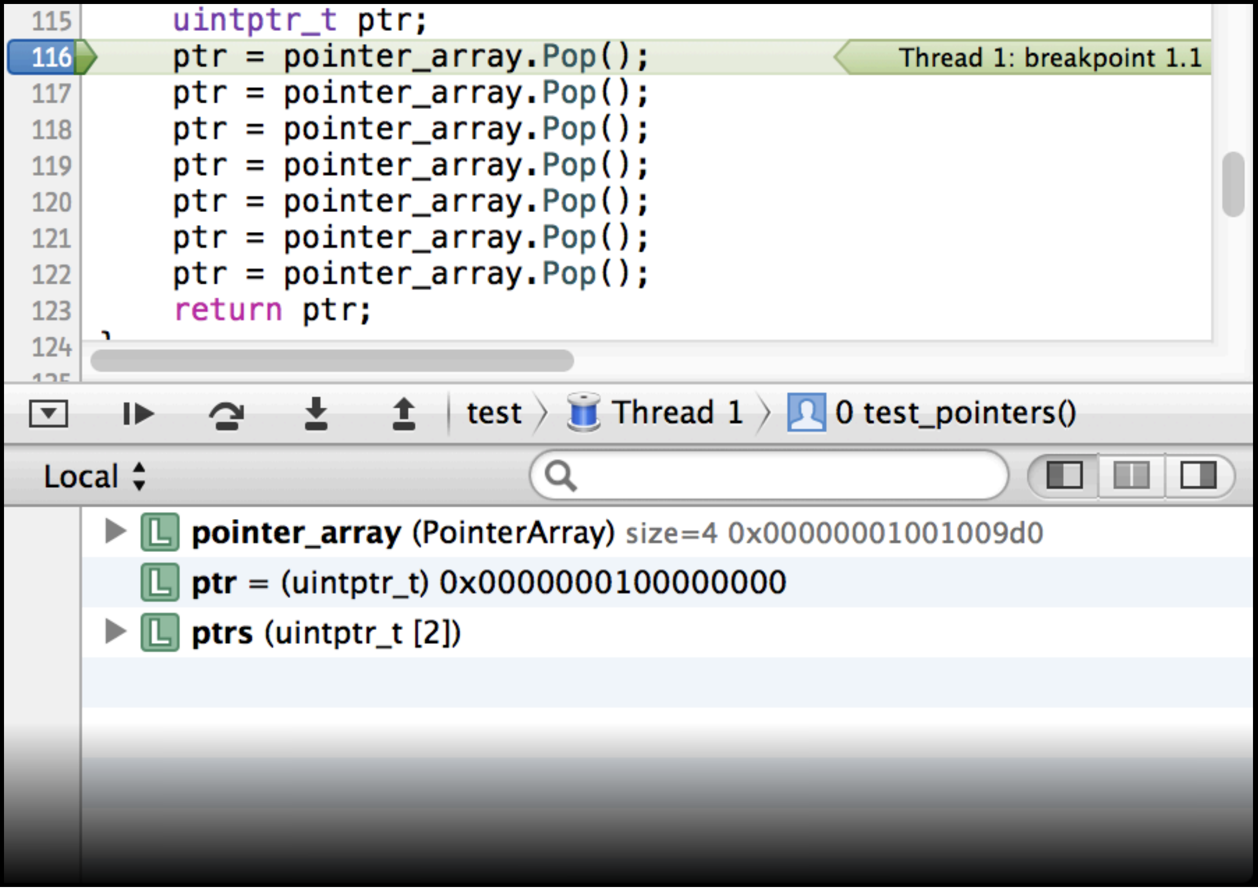


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();                    Thread 1: breakpoint 1.1
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
}
```

```
pointer_array (PointerArray) size=4 0x00000001001009d0
ptr = (uintptr_t) 0x0000000100000000
ptrs (uintptr_t [2])
```

Debug Session

Type summaries in Xcode

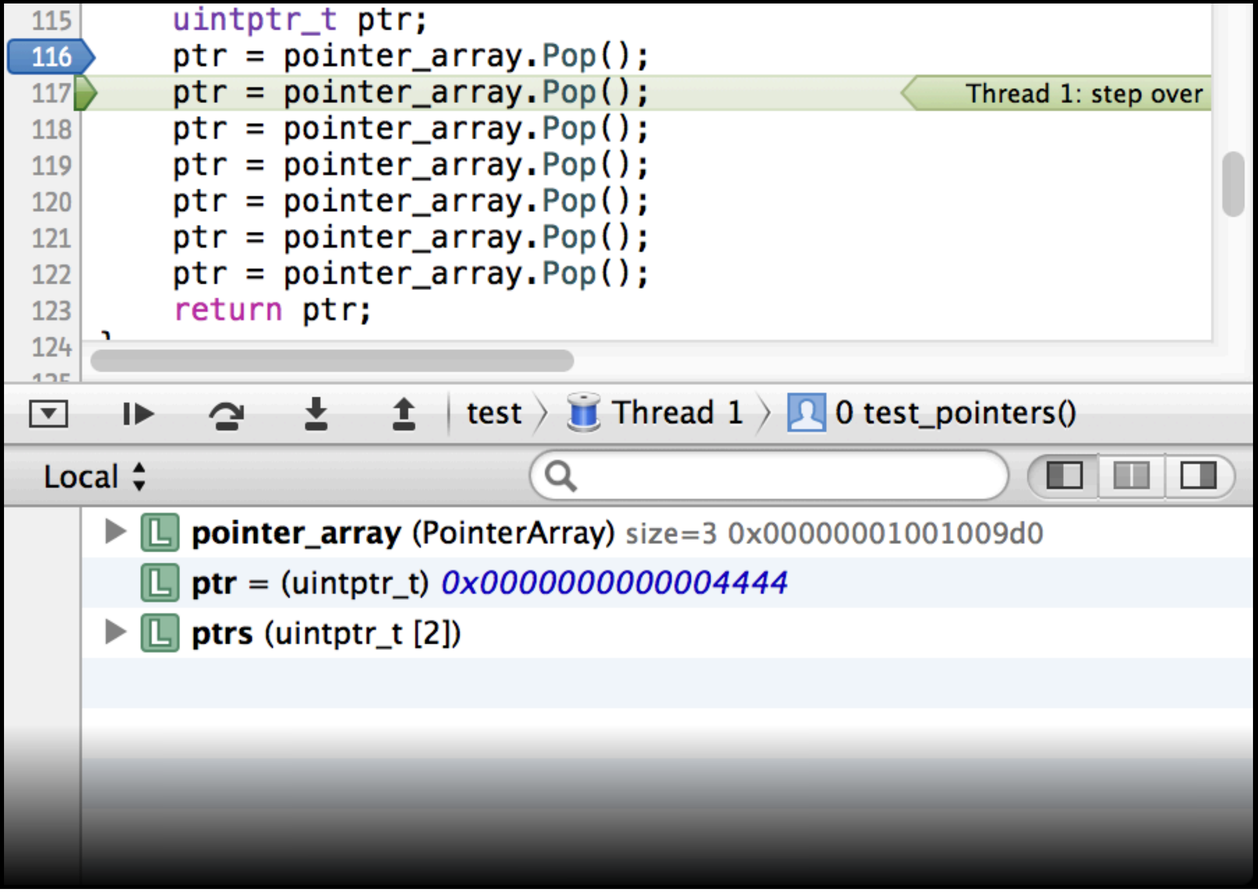


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();                    Thread 1: breakpoint 1.1
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
```

```
▶ pointer_array (PointerArray) size=4 0x00000001001009d0
  ptr = (uintptr_t) 0x0000000100000000
▶ ptrs (uintptr_t [2])
```

Debug Session

Type summaries in Xcode

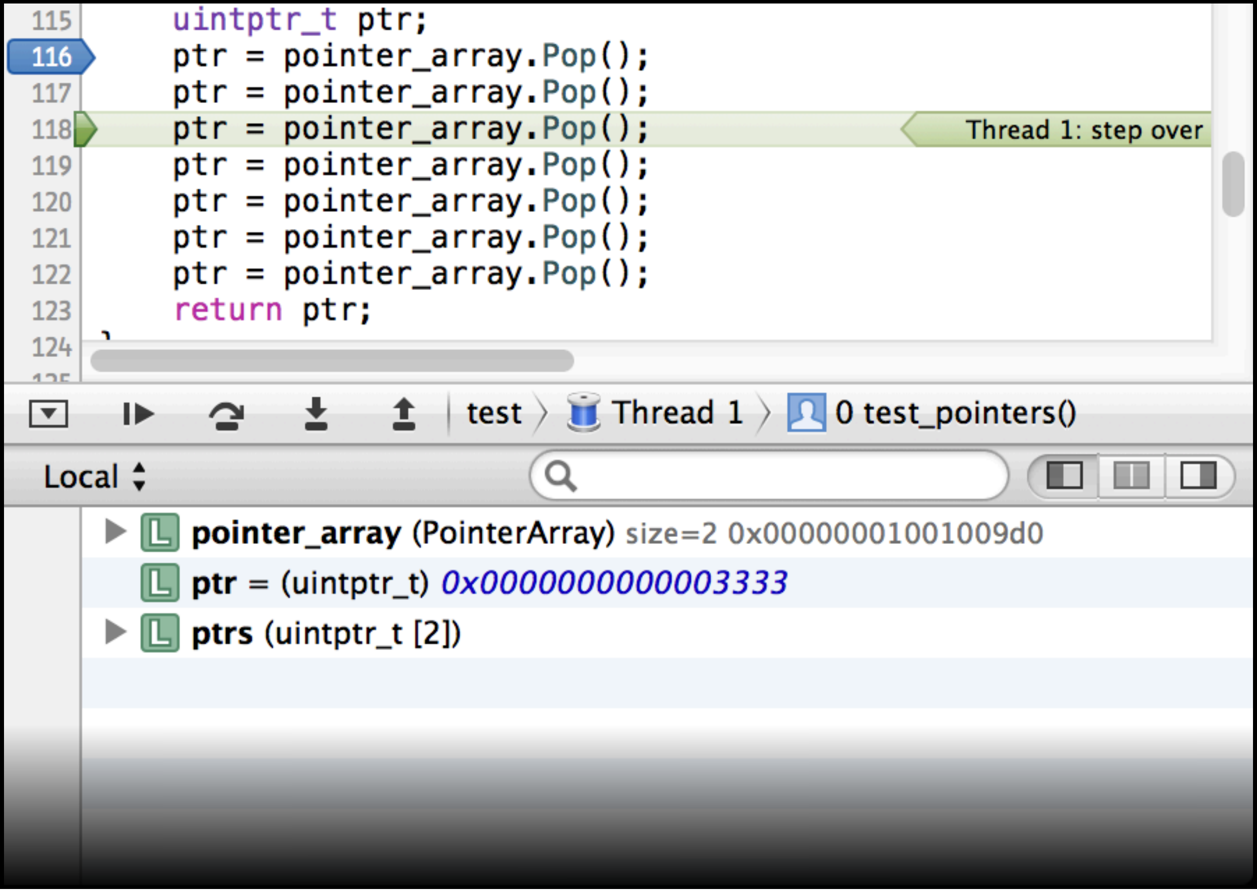


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
```

```
pointer_array (PointerArray) size=3 0x00000001001009d0
ptr = (uintptr_t) 0x0000000000004444
ptrs (uintptr_t [2])
```

Debug Session

Type summaries in Xcode

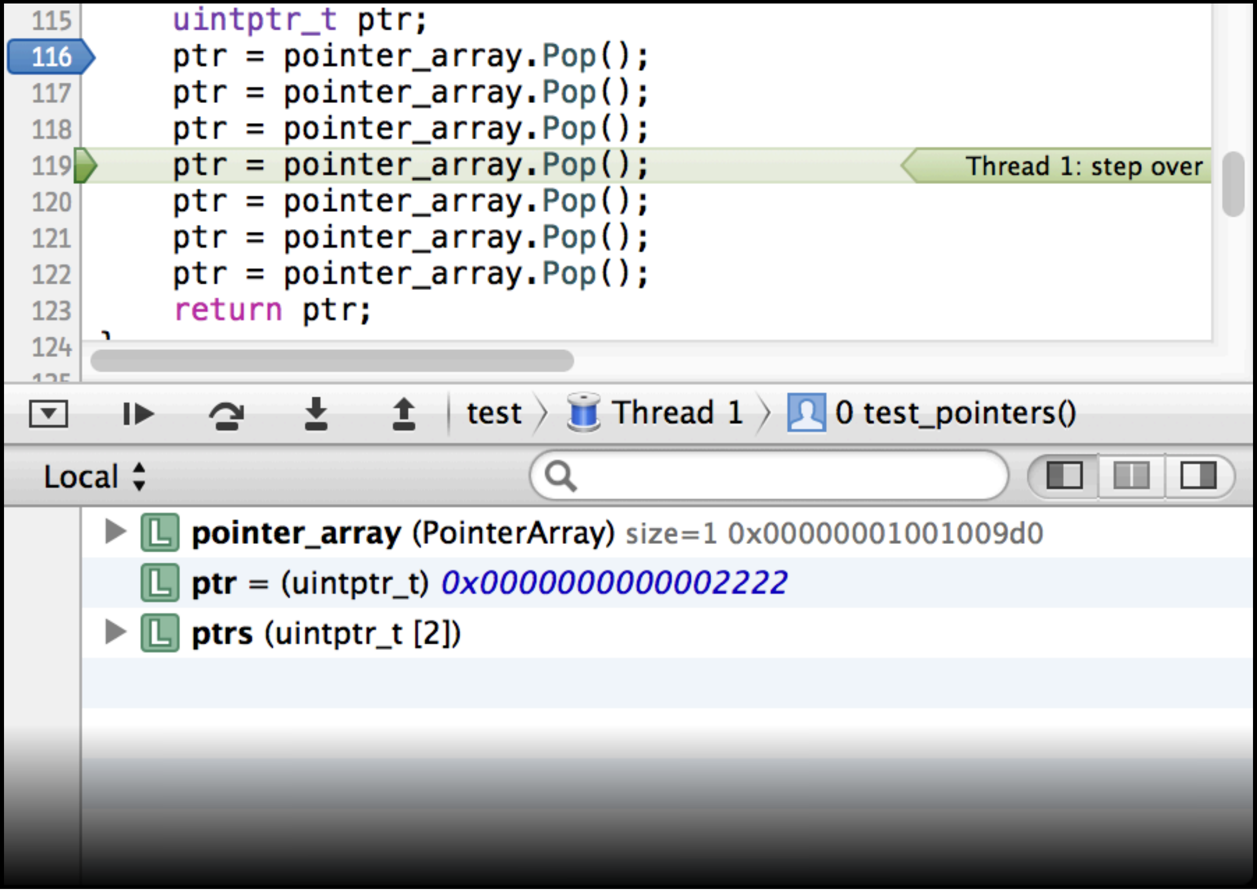


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
```

```
pointer_array (PointerArray) size=2 0x00000001001009d0
ptr = (uintptr_t) 0x0000000000003333
ptrs (uintptr_t [2])
```

Debug Session

Type summaries in Xcode

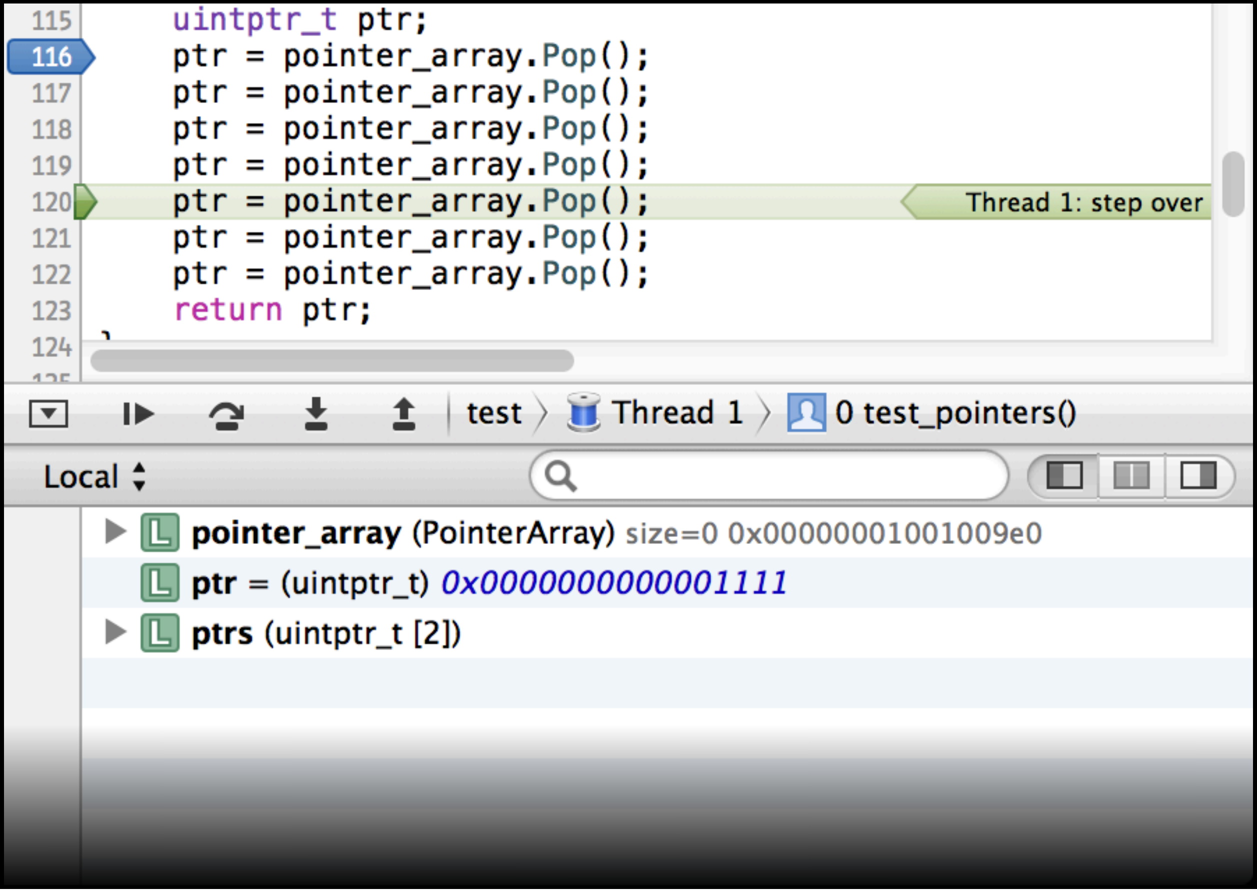


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
}
```

```
pointer_array (PointerArray) size=1 0x00000001001009d0
ptr = (uintptr_t) 0x0000000000002222
ptrs (uintptr_t [2])
```

Debug Session

Type summaries in Xcode

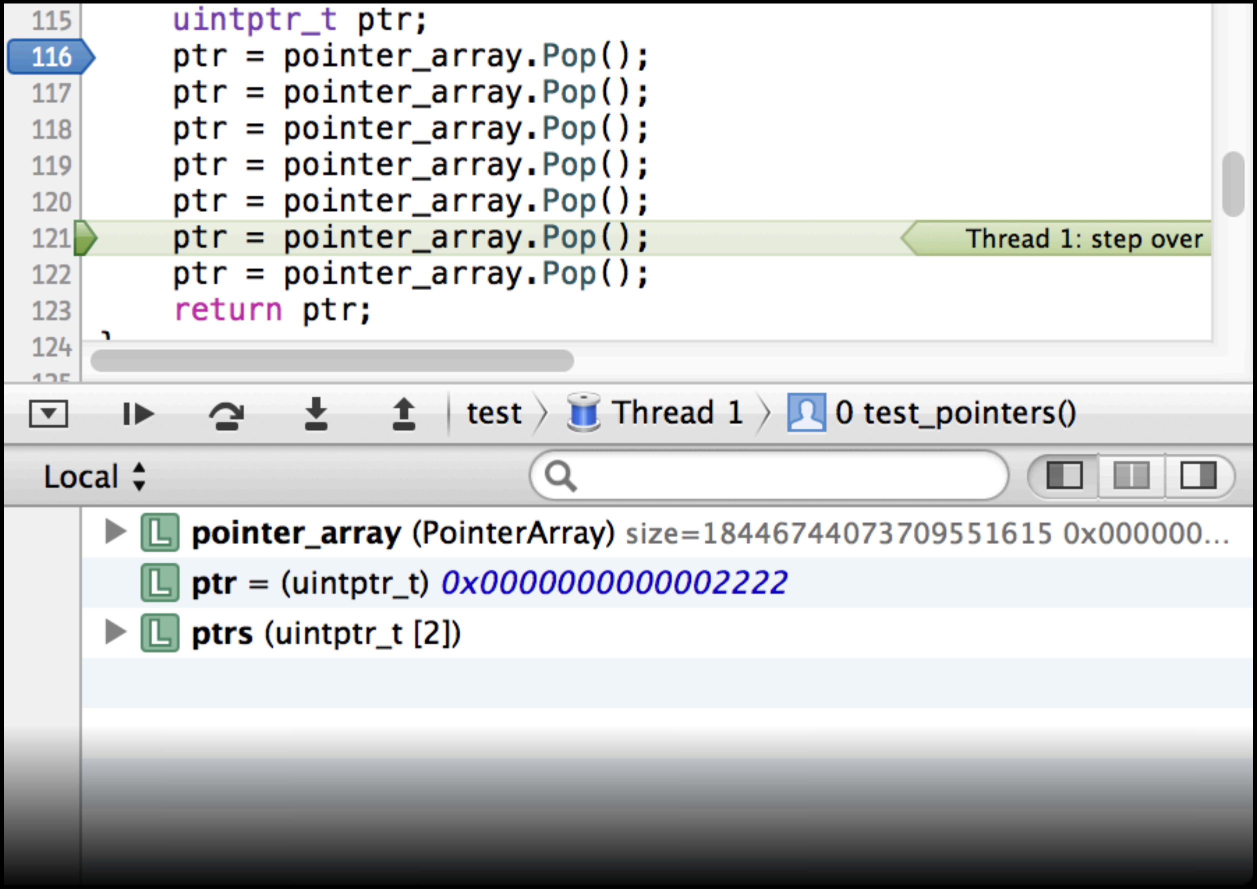


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();        Thread 1: step over
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    return ptr;
}
```

```
pointer_array (PointerArray) size=0 0x00000001001009e0
ptr = (uintptr_t) 0x0000000000001111
ptrs (uintptr_t [2])
```

Debug Session

Type summaries in Xcode

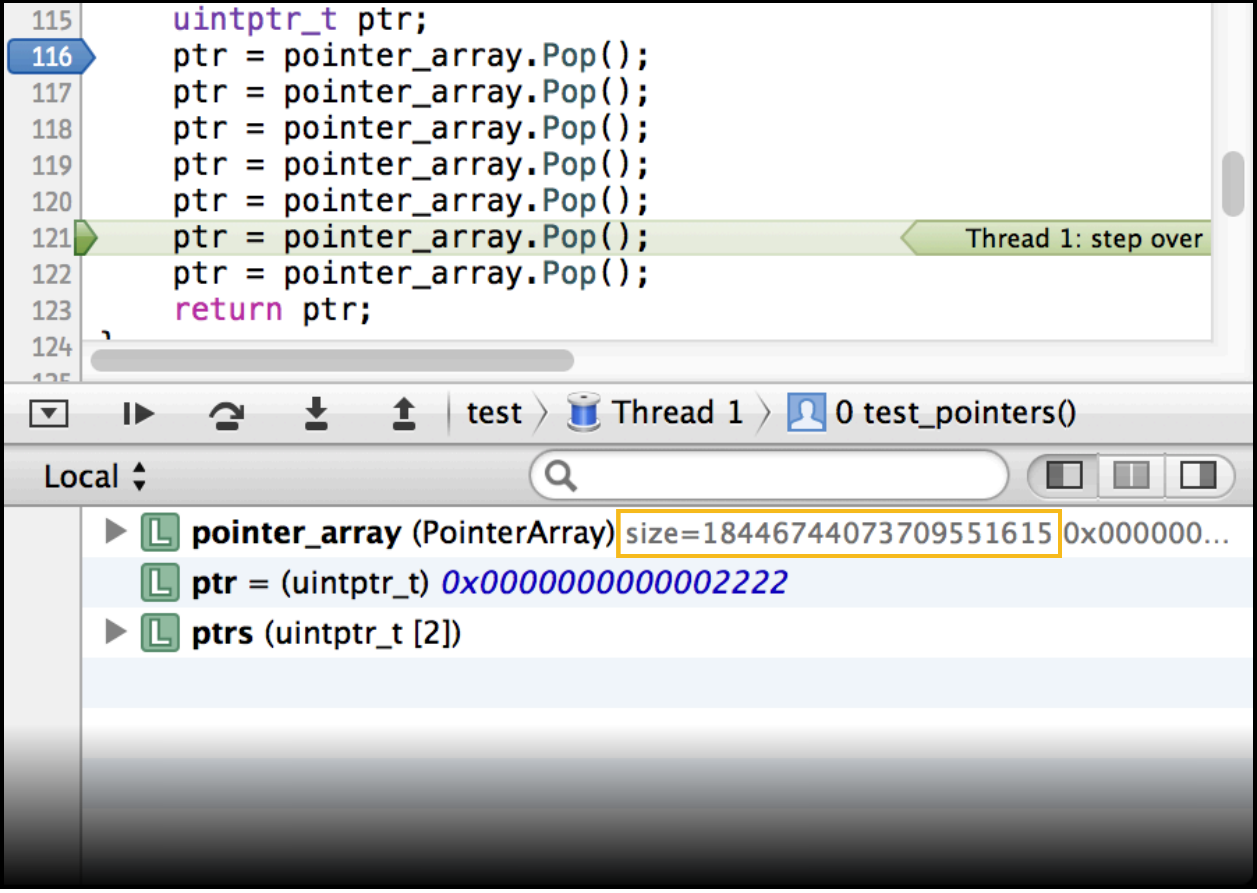


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();                Thread 1: step over
    ptr = pointer_array.Pop();
    return ptr;
```

```
pointer_array (PointerArray) size=18446744073709551615 0x000000...
ptr = (uintptr_t) 0x0000000000002222
ptrs (uintptr_t [2])
```

Debug Session

Type summaries in Xcode

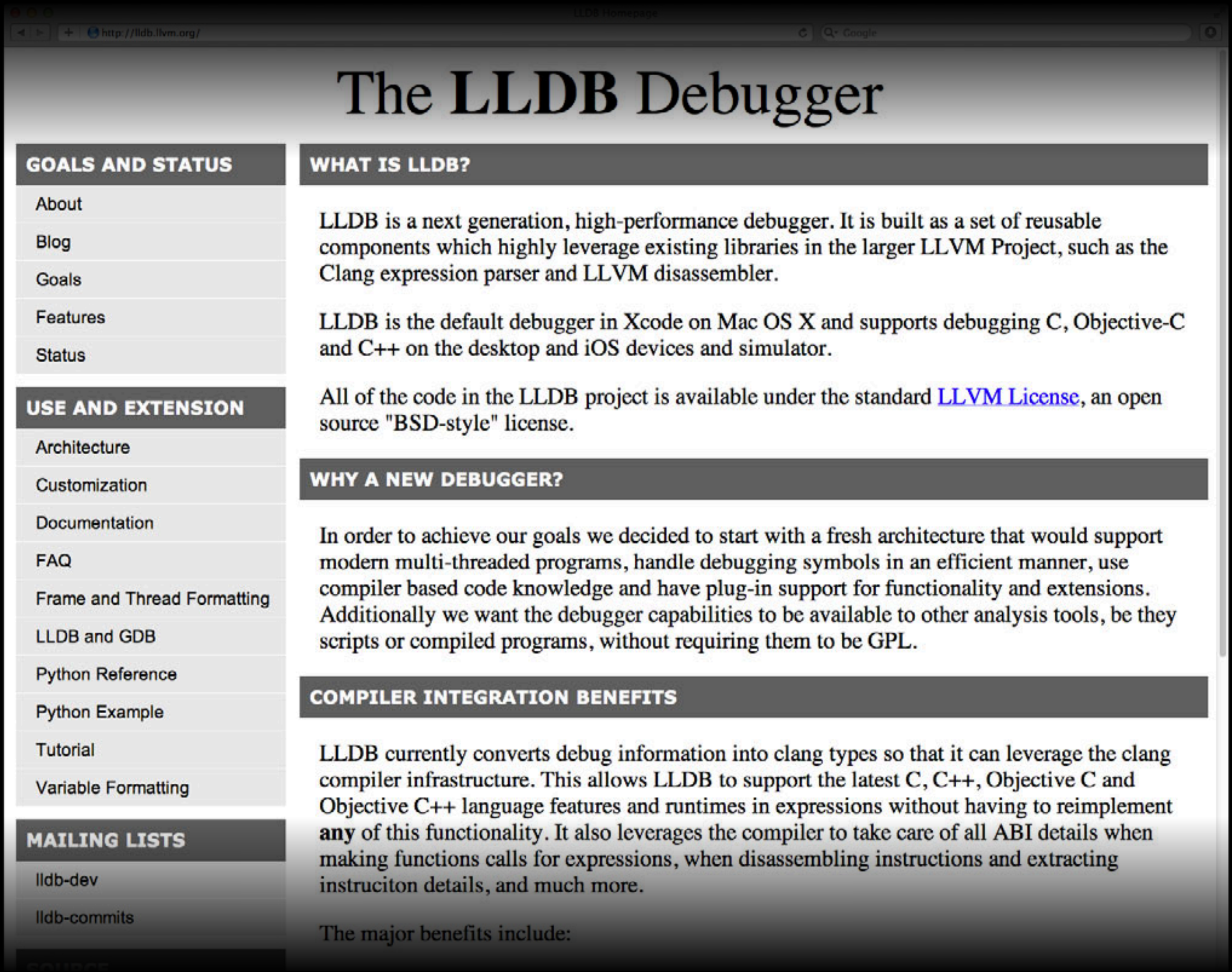


```
    uintptr_t ptr;
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();
    ptr = pointer_array.Pop();                    Thread 1: step over
    ptr = pointer_array.Pop();
    return ptr;
```

```
▶ pointer_array (PointerArray) size=18446744073709551615 0x000000...
  ptr = (uintptr_t) 0x0000000000002222
▶ ptrs (uintptr_t [2])
```

Variable Formatting Reference

Variable Formatting Reference



The LLDB Debugger

GOALS AND STATUS

- About
- Blog
- Goals
- Features
- Status

USE AND EXTENSION

- Architecture
- Customization
- Documentation
- FAQ
- Frame and Thread Formatting
- LLDB and GDB
- Python Reference
- Python Example
- Tutorial
- Variable Formatting

MAILING LISTS

- lldb-dev
- lldb-commits

WHAT IS LLDB?

LLDB is a next generation, high-performance debugger. It is built as a set of reusable components which highly leverage existing libraries in the larger LLVM Project, such as the Clang expression parser and LLVM disassembler.

LLDB is the default debugger in Xcode on Mac OS X and supports debugging C, Objective-C and C++ on the desktop and iOS devices and simulator.

All of the code in the LLDB project is available under the standard LLVM License, an open source "BSD-style" license.

WHY A NEW DEBUGGER?

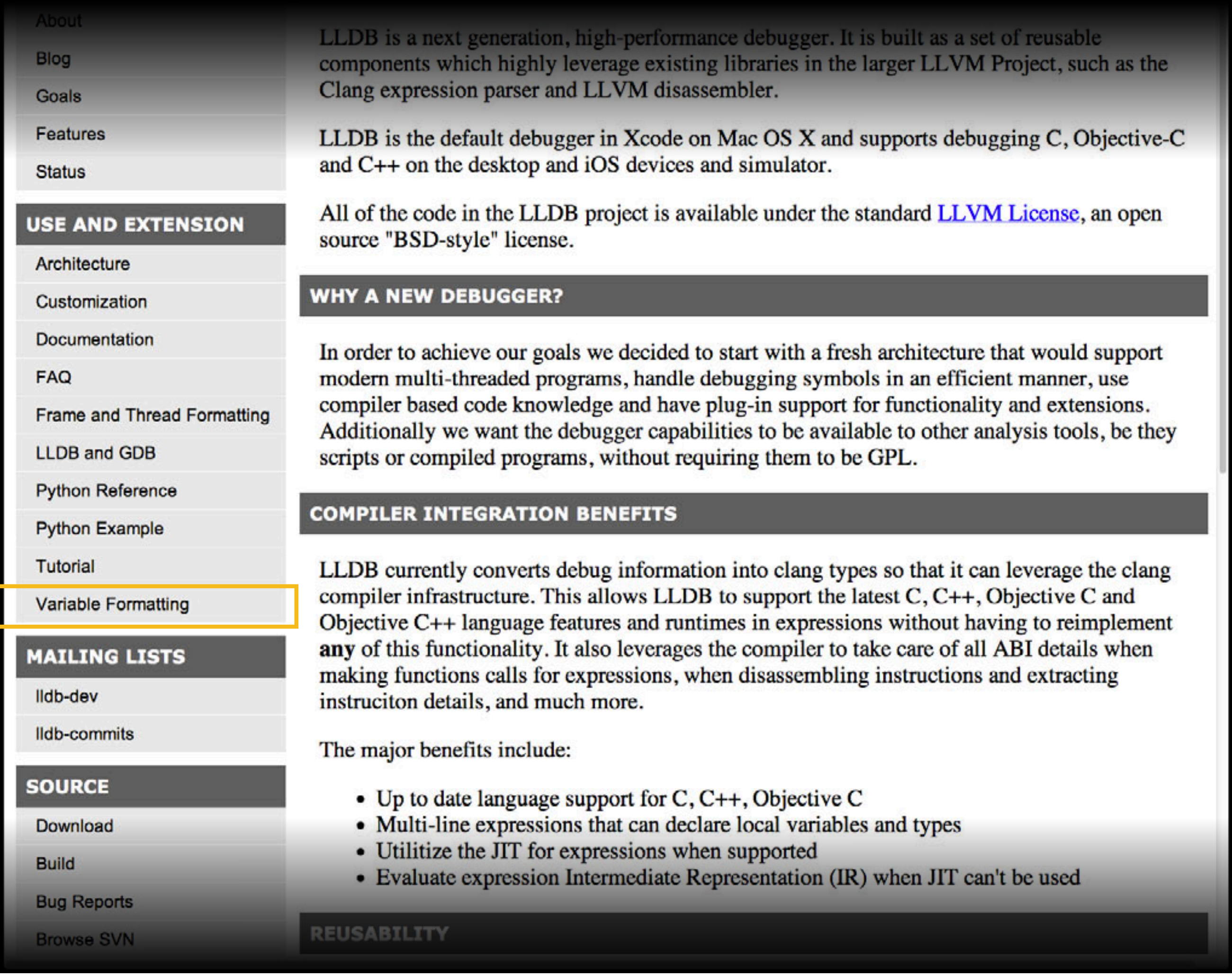
In order to achieve our goals we decided to start with a fresh architecture that would support modern multi-threaded programs, handle debugging symbols in an efficient manner, use compiler based code knowledge and have plug-in support for functionality and extensions. Additionally we want the debugger capabilities to be available to other analysis tools, be they scripts or compiled programs, without requiring them to be GPL.

COMPILER INTEGRATION BENEFITS

LLDB currently converts debug information into clang types so that it can leverage the clang compiler infrastructure. This allows LLDB to support the latest C, C++, Objective C and Objective C++ language features and runtimes in expressions without having to reimplement **any** of this functionality. It also leverages the compiler to take care of all ABI details when making functions calls for expressions, when disassembling instructions and extracting instruciton details, and much more.

The major benefits include:

Variable Formatting Reference



LLDB is a next generation, high-performance debugger. It is built as a set of reusable components which highly leverage existing libraries in the larger LLVM Project, such as the Clang expression parser and LLVM disassembler.

LLDB is the default debugger in Xcode on Mac OS X and supports debugging C, Objective-C and C++ on the desktop and iOS devices and simulator.

All of the code in the LLDB project is available under the standard LLVM License, an open source "BSD-style" license.

WHY A NEW DEBUGGER?

In order to achieve our goals we decided to start with a fresh architecture that would support modern multi-threaded programs, handle debugging symbols in an efficient manner, use compiler based code knowledge and have plug-in support for functionality and extensions. Additionally we want the debugger capabilities to be available to other analysis tools, be they scripts or compiled programs, without requiring them to be GPL.

COMPILER INTEGRATION BENEFITS

LLDB currently converts debug information into clang types so that it can leverage the clang compiler infrastructure. This allows LLDB to support the latest C, C++, Objective C and Objective C++ language features and runtimes in expressions without having to reimplement **any** of this functionality. It also leverages the compiler to take care of all ABI details when making functions calls for expressions, when disassembling instructions and extracting instruciton details, and much more.

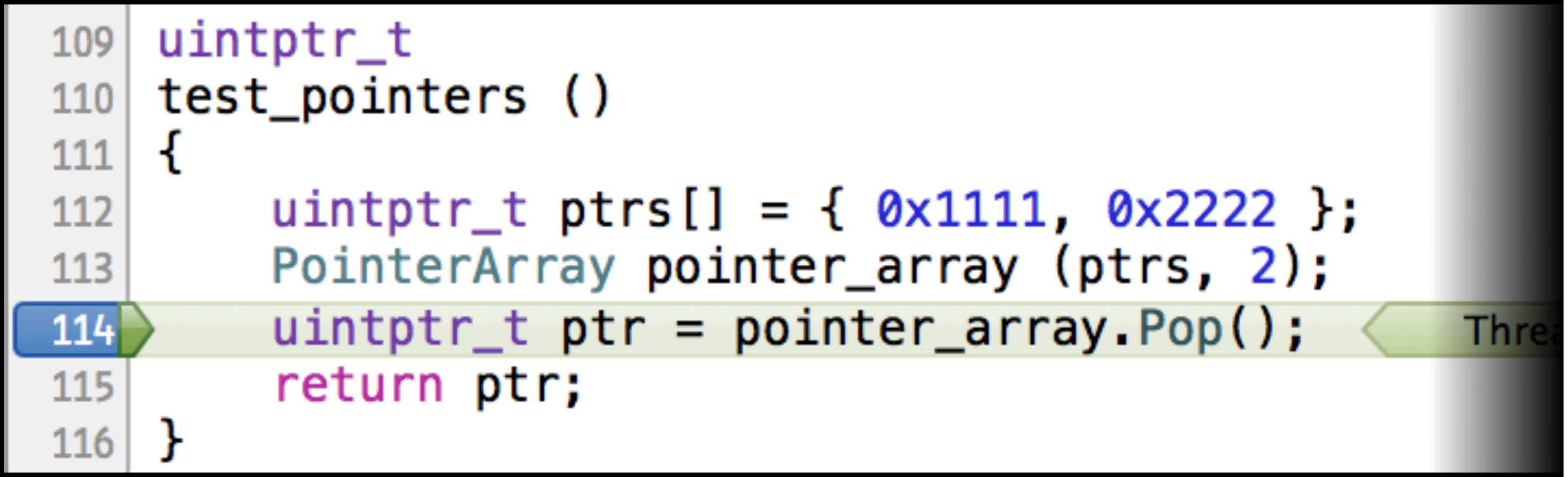
The major benefits include:

- Up to date language support for C, C++, Objective C
- Multi-line expressions that can declare local variables and types
- Utilitize the JIT for expressions when supported
- Evaluate expression Intermediate Representation (IR) when JIT can't be used

REUSABILITY

Debug Session

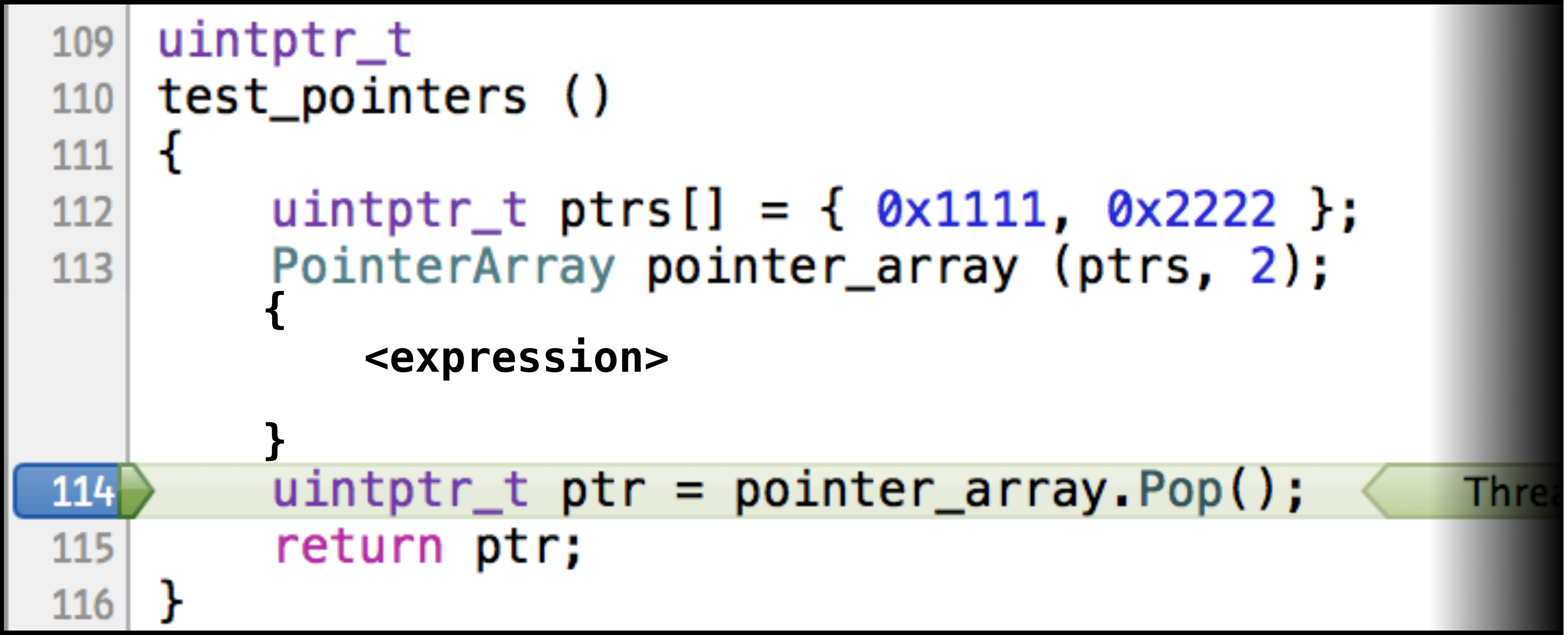
Expressions



```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

Debug Session

Expressions



```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    {
        <expression>

    }
    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

```
(lldb) expression <expression>
```

Expressions

What are expressions?

Expressions

What are expressions?

- Expressions evaluate statements as if they were code

Expressions

What are expressions?

- Expressions evaluate statements as if they were code
- Results are displayed and stored in convenience variables

Expressions

What are expressions?

- Expressions evaluate statements as if they were code
- Results are displayed and stored in convenience variables
- Expressions can do many things

Expressions

What are expressions?

- Expressions evaluate statements as if they were code
- Results are displayed and stored in convenience variables
- Expressions can do many things
    - Arithmetic

Expressions

What are expressions?

- Expressions evaluate statements as if they were code
- Results are displayed and stored in convenience variables
- Expressions can do many things
    - Arithmetic
    - Function calls

Expressions

What are expressions?

- Expressions evaluate statements as if they were code
- Results are displayed and stored in convenience variables
- Expressions can do many things
  - Arithmetic
  - Function calls
  - Casting

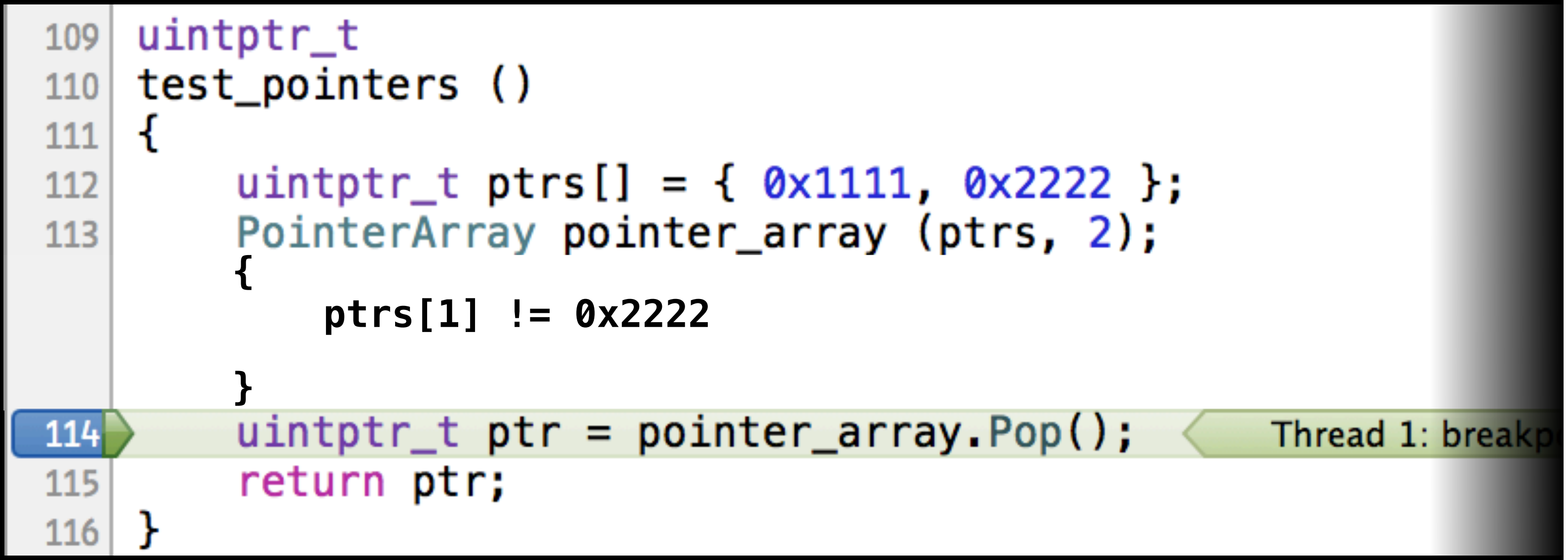
Expressions

What are expressions?

- Expressions evaluate statements as if they were code
- Results are displayed and stored in convenience variables
- Expressions can do many things
  - Arithmetic
  - Function calls
  - Casting
  - With LLDB, much more…

Debug Session

Simple expression

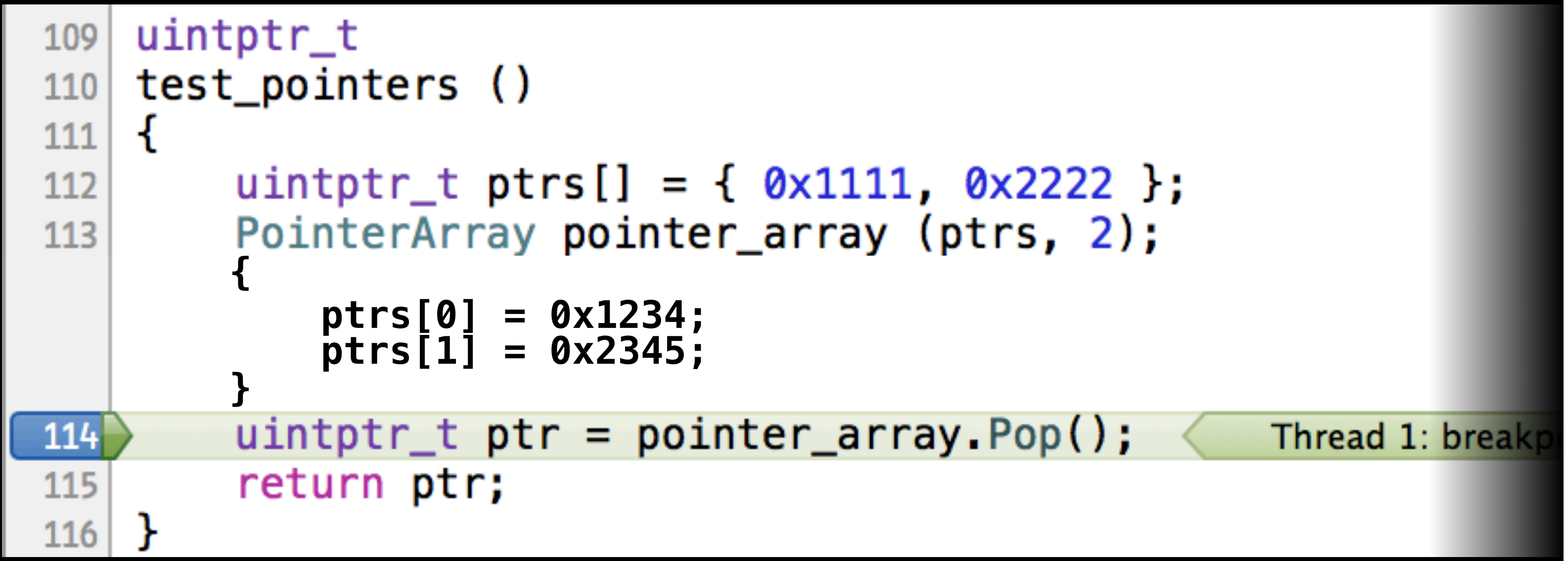


```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    {
        ptrs[1] != 0x2222
    }
    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

```
(lldb) expression ptrs[1] != 0x2222
```

Debug Session

Multiple statements

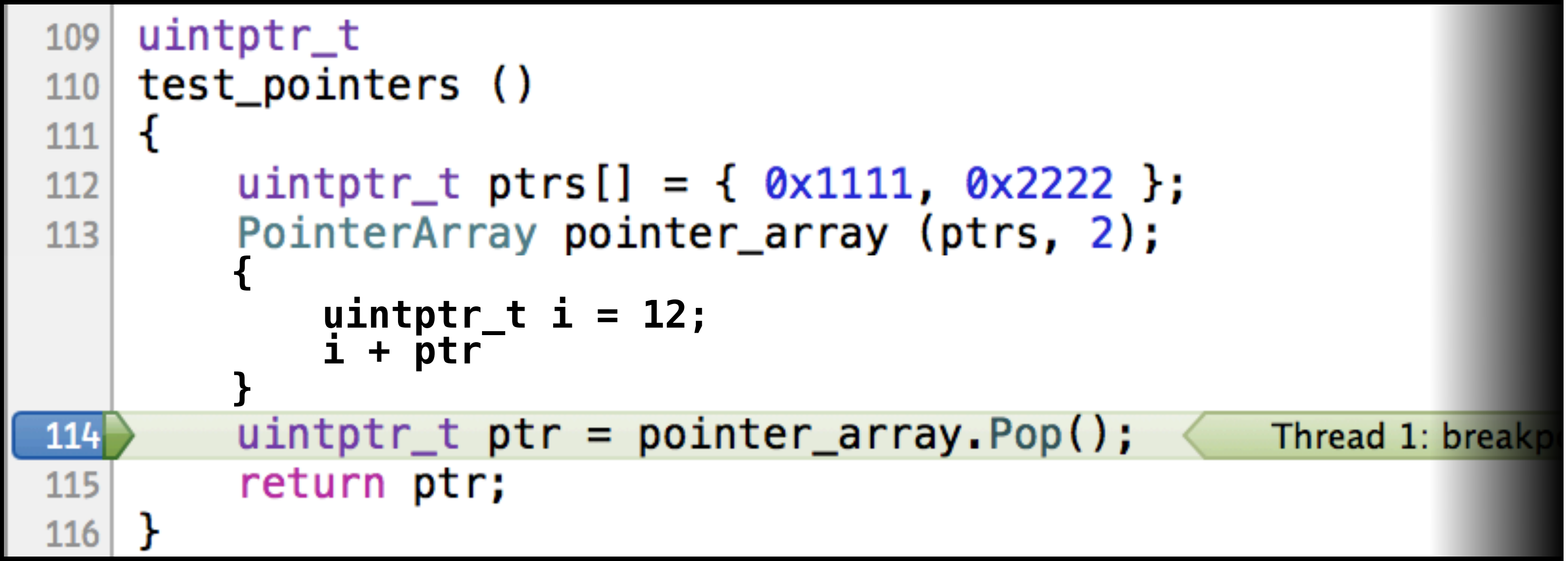


```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    {
        ptrs[0] = 0x1234;
        ptrs[1] = 0x2345;
    }
    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

```
(lldb) ptrs[0] = 0x1234; ptrs[1] = 0x2345;
<no result>
```

Debug Session

Expression local variables

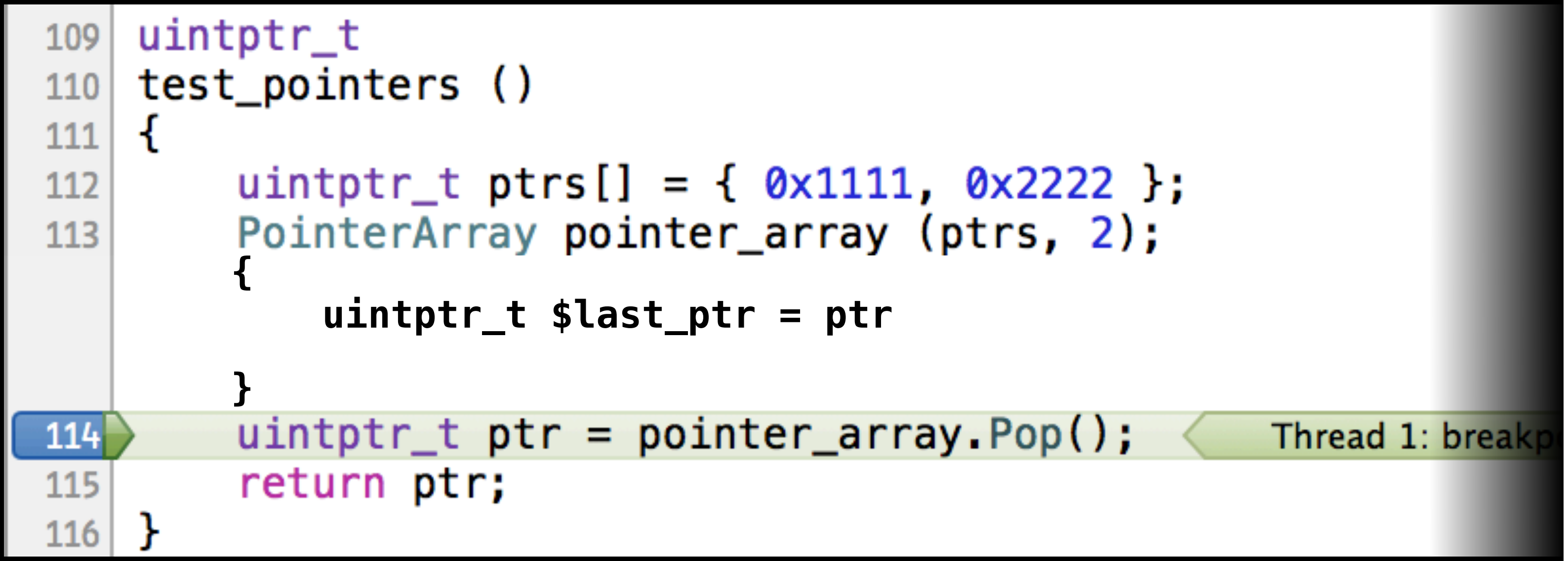


```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    {
        uintptr_t i = 12;
        i + ptr
    }
    uintptr_t ptr = pointer_array.Pop();      Thread 1: breakp
    return ptr;
}
```

```
(lldb) uintptr_t i = 12; i + ptr
(float) $1 = 14.2
```

Debug Session

Expression global variables

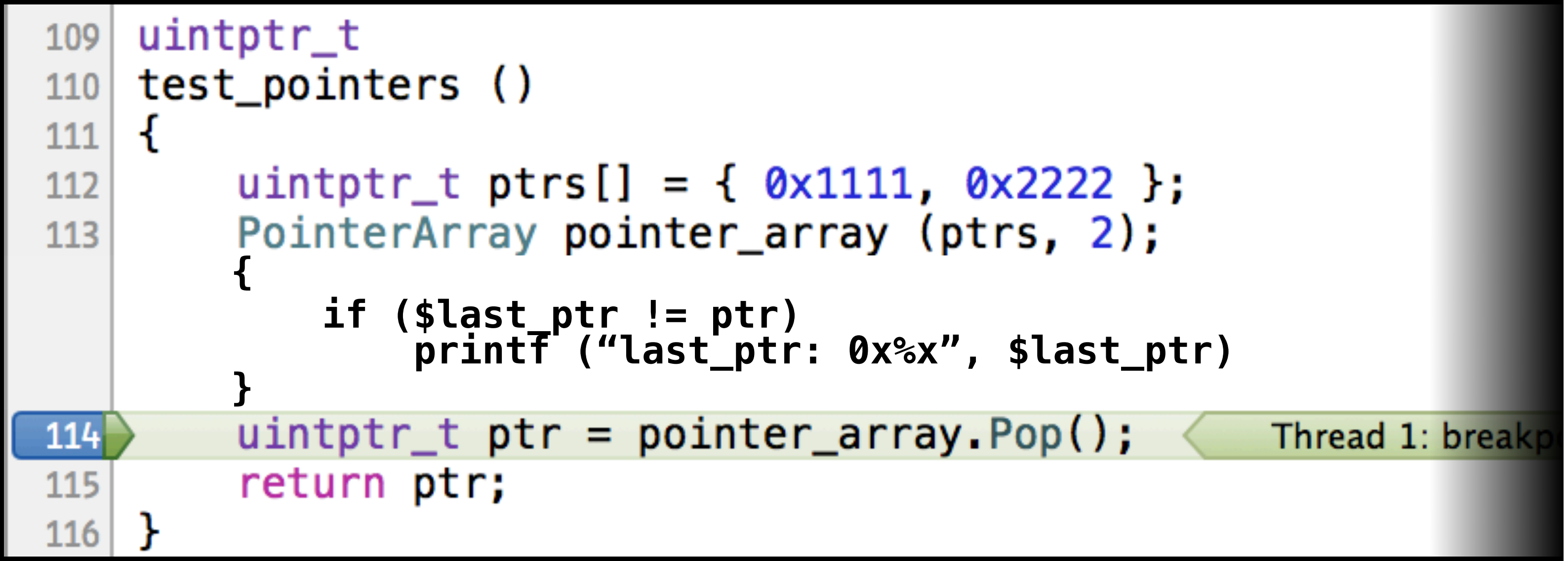


```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    {
        uintptr_t $last_ptr = ptr
    }
    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

```
(lldb) expression uintptr_t $last_ptr = ptr
<no result>
```

Debug Session

Flow control

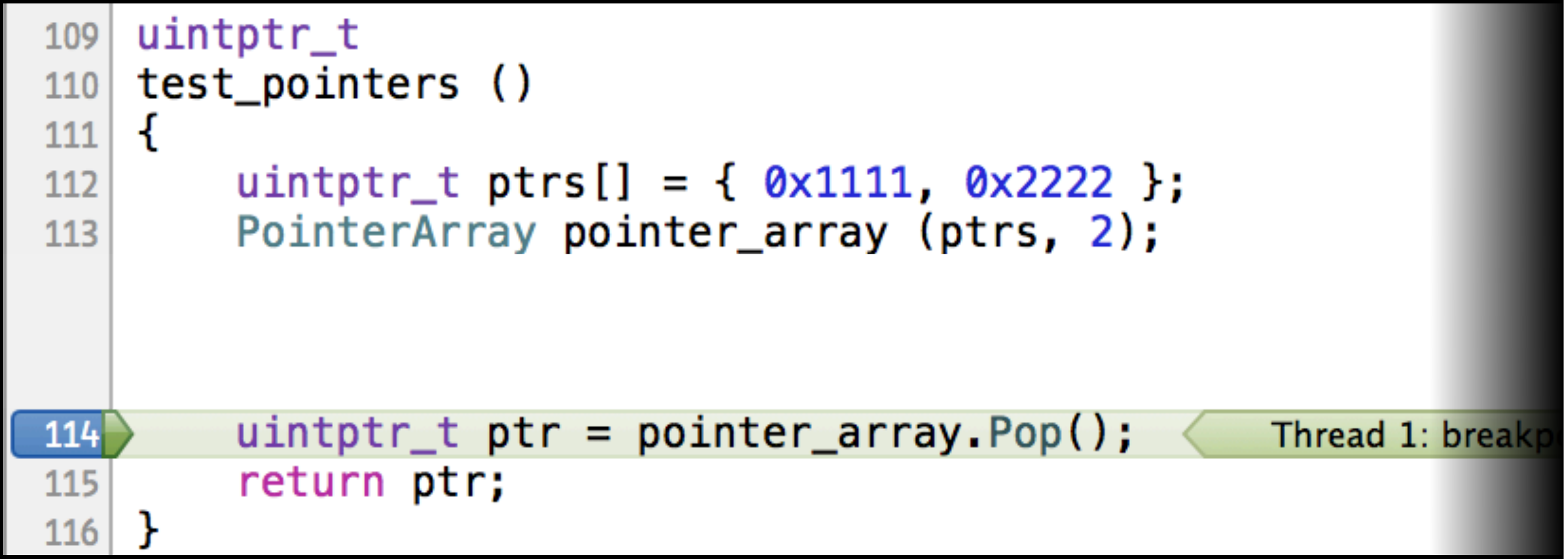


```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    {
        if ($last_ptr != ptr)
            printf (“last_ptr: 0x%x”, $last_ptr)
    }
    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

```
(lldb) expression if ($last_ptr != ptr)
                         printf (“last_ptr: 0x%x”, $last_ptr)
```

Debug Session

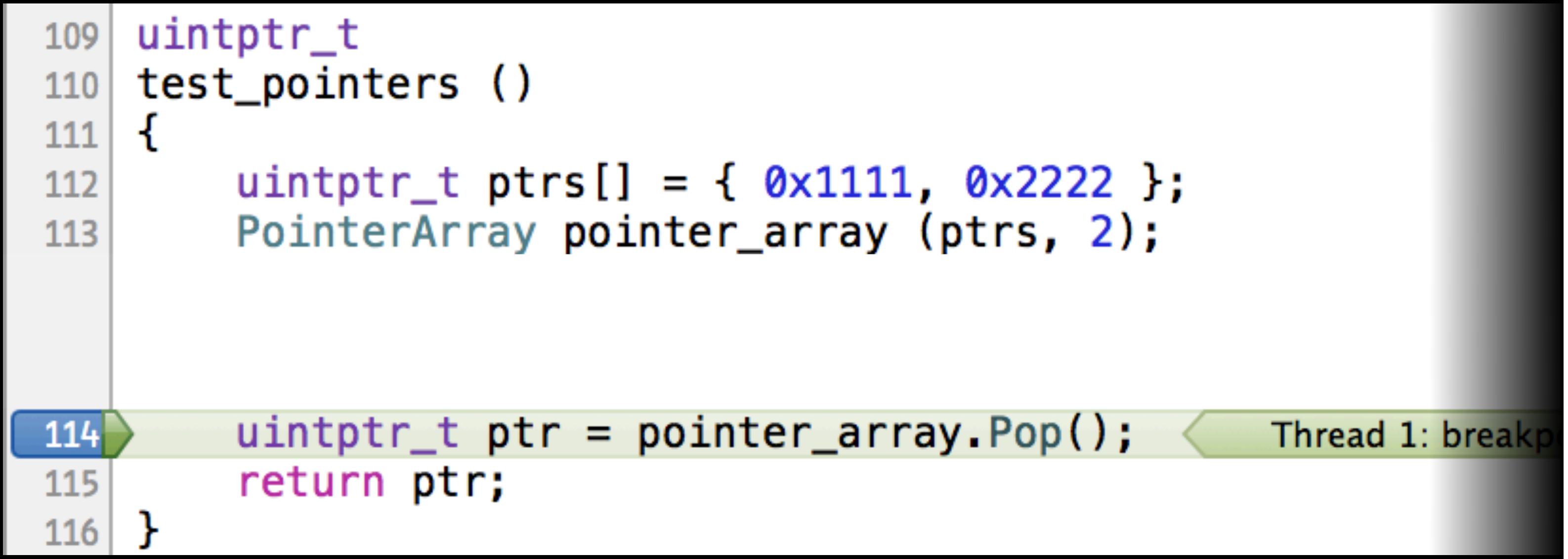
Stopping in expression code



```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);



    uintptr_t ptr = pointer_array.Pop();          Thread 1: breakp
    return ptr;
}
```

Debug Session

Stopping in expression code

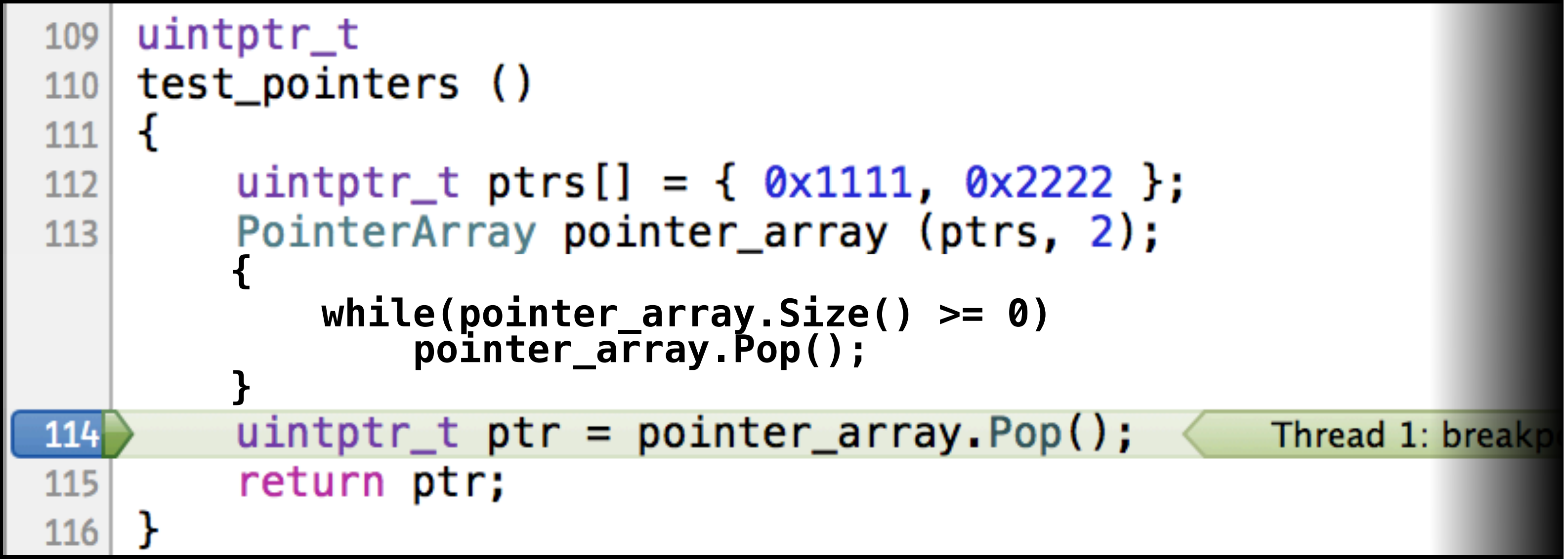


```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);



    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

```
(lldb) expression --unwind-on-error=0 --
```

Debug Session

Stopping in expression code

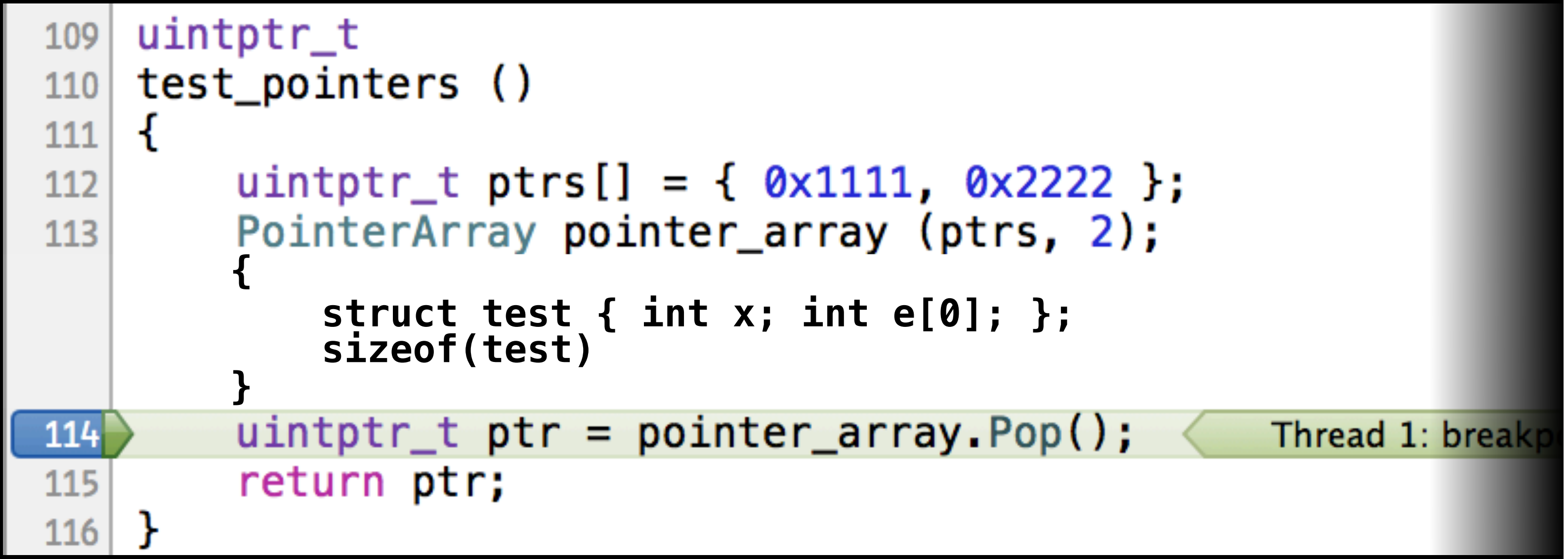


```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    {
        while(pointer_array.Size() >= 0)
            pointer_array.Pop();
    }
    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

```
(lldb) expression --unwind-on-error=0 --
while(pointer_array.Size() >= 0)
    pointer_array.Pop()
```

Debug Session

Define local types



```
uintptr_t
test_pointers ()
{
    uintptr_t ptrs[] = { 0x1111, 0x2222 };
    PointerArray pointer_array (ptrs, 2);
    {
        struct test { int x; int e[0]; };
        sizeof(test)
    }
    uintptr_t ptr = pointer_array.Pop();
    return ptr;
}
```

```
(lldb)) expression struct test { int x; int e[0]; };
                   sizeof(test)
```

Debug Session

Multiple line expressions

Debug Session

Multiple line expressions

```
(lldb) expression
```

Debug Session

Multiple line expressions

```
(lldb) expression
Enter expressions, then terminate with an empty line to evaluate:
```

Debug Session

Multiple line expressions

```
(lldb) expression
Enter expressions, then terminate with an empty line to evaluate:
const char *arg;
```

Debug Session

Multiple line expressions

```
(lldb) expression
Enter expressions, then terminate with an empty line to evaluate:
const char *arg;
int i=0;
```

Debug Session

Multiple line expressions

```
(lldb) expression
Enter expressions, then terminate with an empty line to evaluate:
const char *arg;
int i=0;
while (arg = argv[i++])
```

Debug Session

Multiple line expressions

```
(lldb) expression
Enter expressions, then terminate with an empty line to evaluate:
const char *arg;
int i=0;
while (arg = argv[i++])
  (int)puts(arg);
```

Debug Session

Multiple line expressions

```
(lldb) expression
Enter expressions, then terminate with an empty line to evaluate:
const char *arg;
int i=0;
while (arg = argv[i++])
  (int)puts(arg);
i
```

Debug Session

Multiple line expressions

```
(lldb) expression
Enter expressions, then terminate with an empty line to evaluate:
const char *arg;
int i=0;
while (arg = argv[i++])
  (int)puts(arg);
i

/tmp/prints-args
4
5
6
(int) $3 = 4
(lldb)
```

Expressions

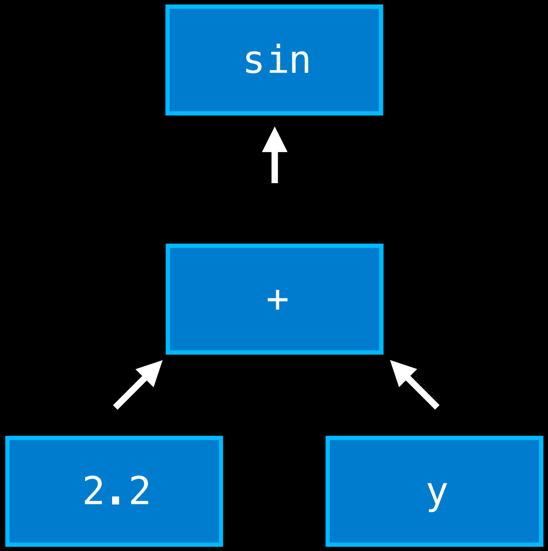
Other debuggers

```
(gdb) p sin(2.2 + y)
(double) $0 = 0.808496
```

Expressions

Other debuggers

```
(gdb) p sin(2.2 + y)
(double) $0 = 0.808496
```



Expressions

LLDB with Clang

```
(lldb) expression sin(2.2 + y)
(double) $0 = 0.808496
```

Expressions

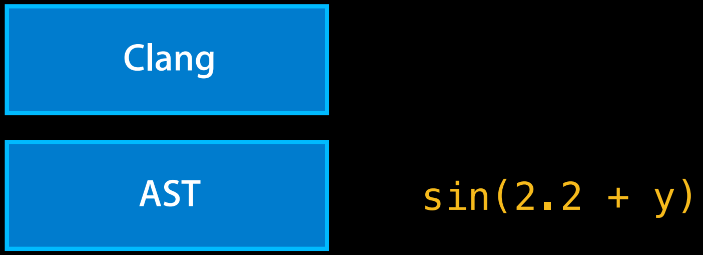
LLDB with Clang

```
(lldb) expression sin(2.2 + y)
(double) $0 = 0.808496
```

Expressions

LLDB with Clang

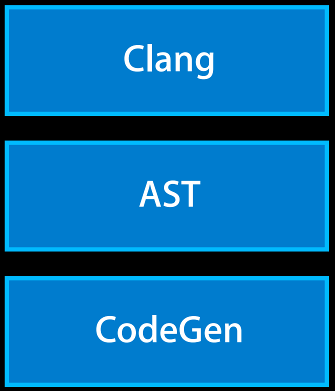
```
(lldb) expression sin(2.2 + y)
(double) $0 = 0.808496
```



Expressions

LLDB with Clang

```
(lldb) expression sin(2.2 + y)
(double) $0 = 0.808496
```

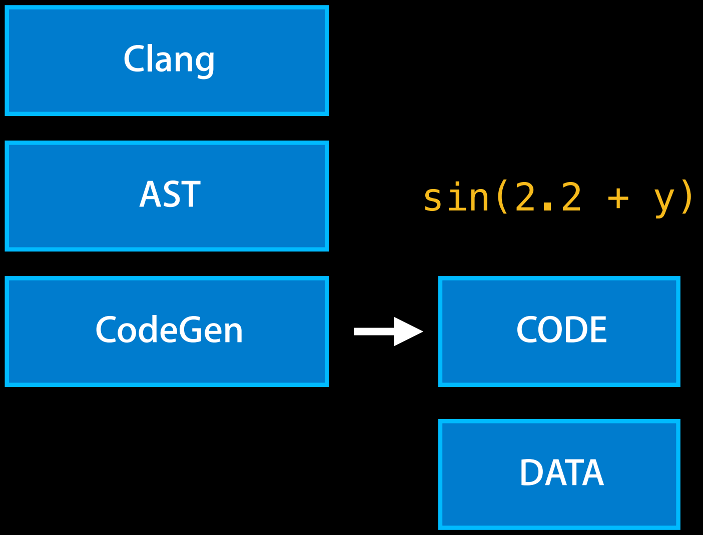


```
sin(2.2 + y)
```

Expressions

LLDB with Clang

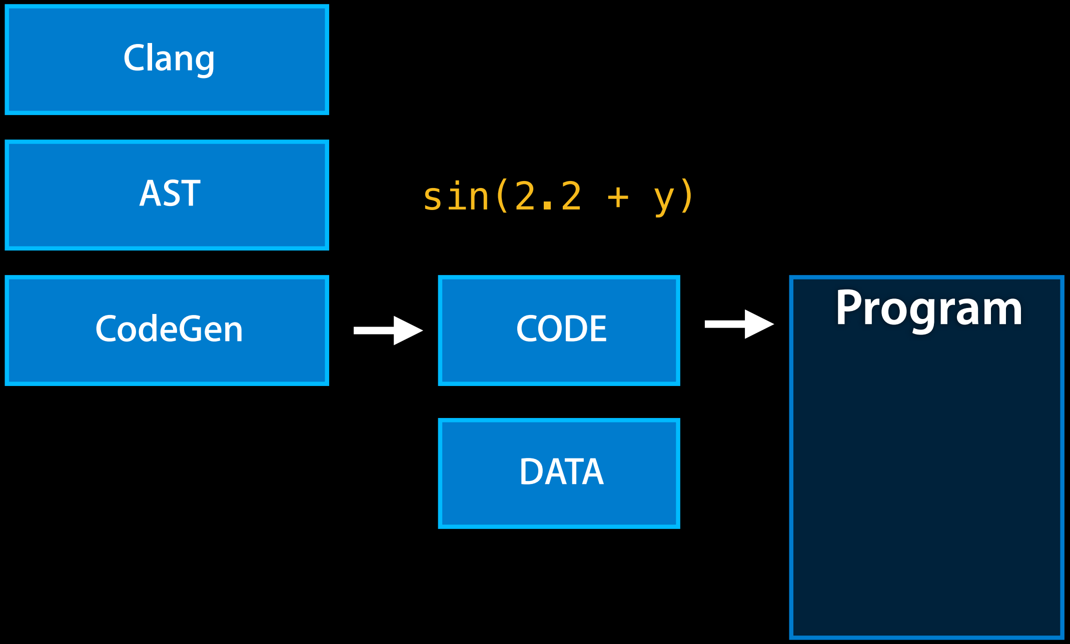
```
(lldb) expression sin(2.2 + y)
(double) $0 = 0.808496
```



Expressions

LLDB with Clang

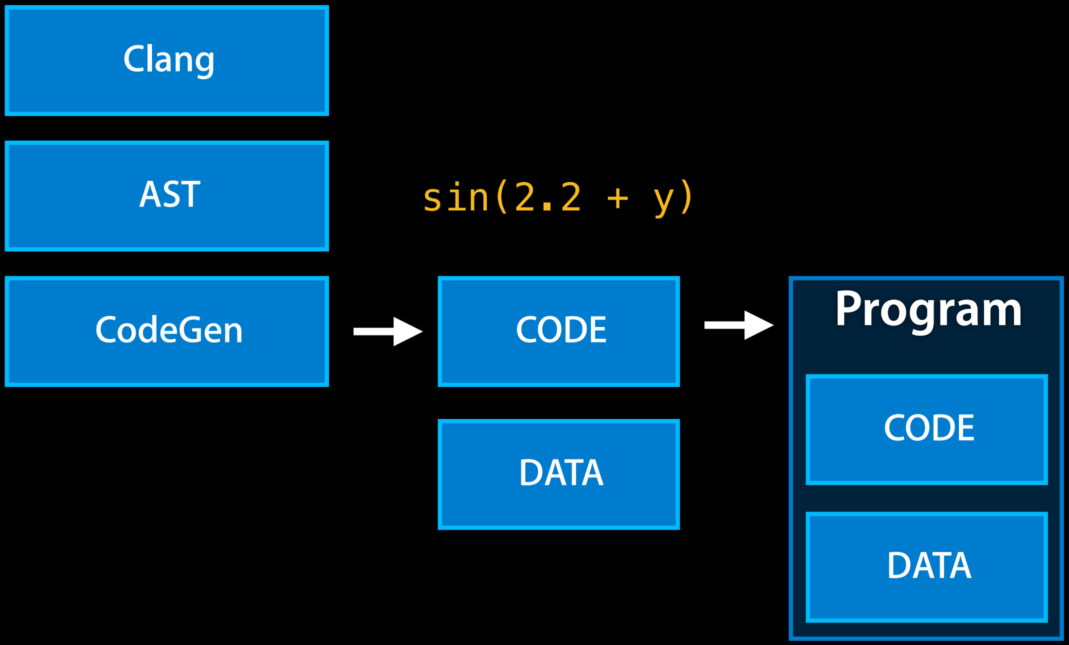
```
(lldb) expression sin(2.2 + y)
(double) $0 = 0.808496
```



Expressions

LLDB with Clang

```
(lldb) expression sin(2.2 + y)
(double) $0 = 0.808496
```



Injecting Checks into Expressions

Compiler integration

```
(lldb) p item->value + [value floatValue]
```

Injecting Checks into Expressions

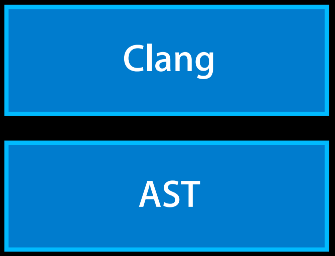
Compiler integration

```
(lldb) p item->value + [value floatValue]
```

Injecting Checks into Expressions

Compiler integration

```
(lldb) p item->value + [value floatValue]
```

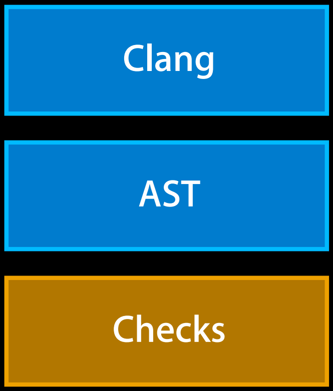


```
item->value + [value floatValue]
```

Injecting Checks into Expressions

Compiler integration

```
(lldb) p item->value + [value floatValue]
```

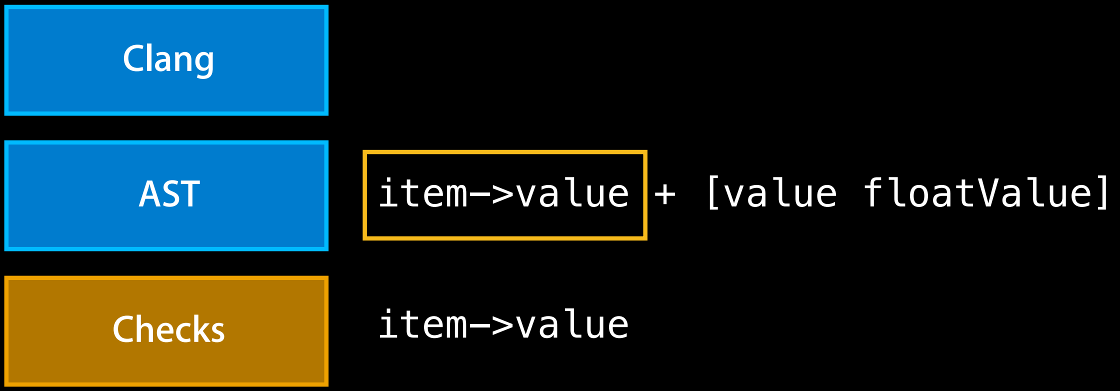


```
item->value + [value floatValue]
```

Injecting Checks into Expressions

Compiler integration

```
(lldb) p item->value + [value floatValue]
```

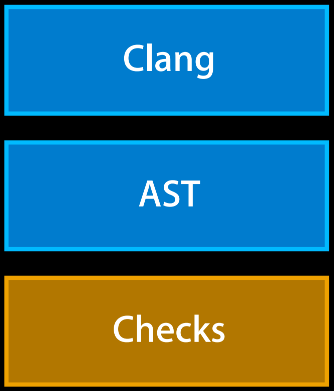


```
item->value + [value floatValue]
item->value
```

Injecting Checks into Expressions

Compiler integration

```
(lldb) p item->value + [value floatValue]
```



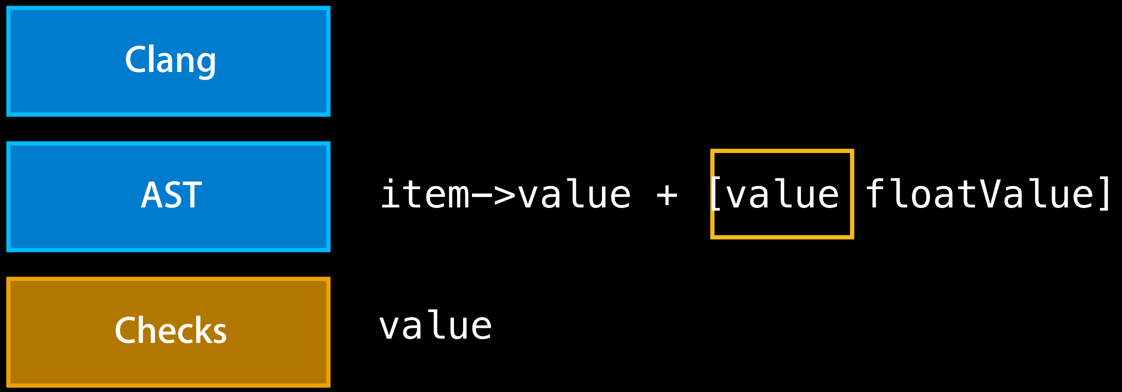
```
item->value + [value floatValue]
```

```
___check_ptr(item)->value
```

Injecting Checks into Expressions

Compiler integration

```
(lldb) p item->value + [value floatValue]
```

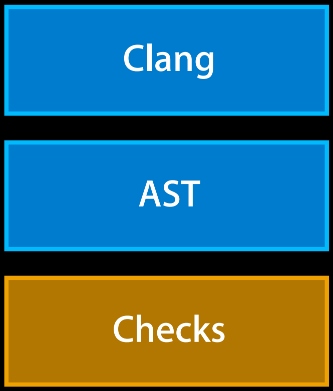


```
item->value + [value floatValue]
value
```

Injecting Checks into Expressions

Compiler integration

```
(lldb) p item->value + [value floatValue]
```

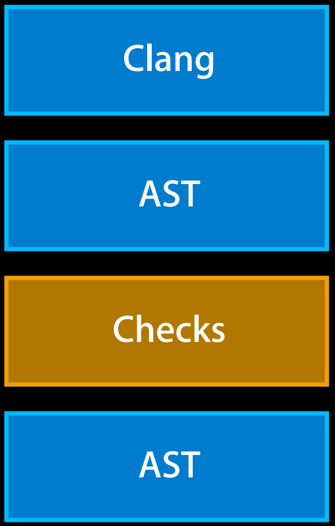


```
item->value + [value floatValue]
___check_obj(value)
```

Injecting Checks into Expressions

Compiler integration

```
(lldb) p item->value + [value floatValue]
```



```
item->value + [value floatValue]
```

```
___check_obj(value)
```

```
___check_ptr(item)->value +
[___check_obj(value) floatValue]
```

Overview

Debugging with LLDB

LLDB in depth

Introduction

Examples

Conclusion

Overview

Debugging with LLDB

Examples

Introduction LLDB in depth Conclusion

Symbolication

Symbolicating a crash log

- Manually
- Using built-in Python module

Symbolication
Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target create a.out
```

Symbolication
Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target create a.out
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target create a.out
```

Symbolication
Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target create a.out
(lldb) target modules add /usr/lib/dyld
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target create a.out
(lldb) target modules add /usr/lib/dyld
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target create a.out
(lldb) target modules add /usr/lib/dyld
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target modules load --file a.out __TEXT 0x104cc0000
(lldb) target modules load --file dyld  __TEXT 0x7fff648c0000
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target modules load --file a.out __TEXT 0x104cc0000
(lldb) target modules load --file dyld  __TEXT 0x7fff648c0000
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target modules load --file a.out __TEXT 0x104cc0000
(lldb) target modules load --file dyld  __TEXT 0x7fff648c0000
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target modules lookup --address 0x0000000104cc146a
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target modules lookup --address 0x0000000104cc146a
```

Symbolication

Manual

```
Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_c.dylib 0x00007fff8ad164f0 strlen + 16
1   a.out             0x0000000104cc146a main + 34 (char_traits.h:257)
2   a.out             0x0000000104cc1440 start + 52

Binary Images:
       0x104cc0000 -        0x104cc1fff  a.out (??? - ???) <626A790D-54
    0x7fff648c0000 -     0x7fff648f4baf  dyld (195.6 - ???) <0CD1B35B-
    0x7fff87f5e000 -     0x7fff87f68ff7  liblaunch.dylib (392.38.0 -
    0x7fff8800c000 -     0x7fff8800dff7  libsystem_sandbox.dylib (???
    0x7fff88426000 -     0x7fff8842efff  libsystem_dnssd.dylib (???
    0x7fff886d2000 -     0x7fff886e0fff  libdispatch.dylib (187.9.0 -
    0x7fff88708000 -     0x7fff88709fff  libunc.dylib (24.0.0 - compati
```

```
(lldb) target modules lookup --address 0x0000000104cc146a
```

Symbolication

Manual

```
(lldb) target modules lookup -a 0x0000000104cc146a
Address: a.out[0x000000010000146a] (a.out.__TEXT.__text + 94)
Summary: a.out`main + 34 [inlined] char_traits<char>::length(char const*) + 5
         a.out`main + 29 [inlined] string::assign(char const*) at main.cpp:11
         a.out`main + 29 at main.cpp:11
```

Symbolication

Manual

```
(lldb) target modules lookup -a 0x0000000104cc146a
Address: a.out[0x000000010000146a] (a.out.__TEXT.__text + 94)
Summary: a.out`main + 34 [inlined] char_traits<char>::length(char const*) + 5
         a.out`main + 29 [inlined] string::assign(char const*) at main.cpp:11
         a.out`main + 29 at main.cpp:11
```

Symbolication

Using with Python module

Symbolication

Using with Python module

```
(lldb) script import lldb.macosx.crashlog
```

Symbolication

Using with Python module

```
(lldb) script import lldb.macosx.crashlog
"crashlog" command installed, type "crashlog --help" for detailed help
```

Symbolication

Using with Python module

```
(lldb) script import lldb.macosx.crashlog
"crashlog" command installed, type "crashlog --help" for detailed help
(lldb) crashlog /tmp/a.crash
```

Symbolication

Using with Python module

```
(lldb) script import lldb.macosx.crashlog
"crashlog" command installed, type "crashlog --help" for detailed help
(lldb) crashlog /tmp/a.crash
Getting symbols for 626A790D-54BA-3B1F-9689-095C4A5B35FC /tmp/a.out... ok
Thread[0] EXC_BAD_ACCESS (SIGSEGV) (KERN_INVALID_ADDRESS at
0x0000000000000000)
[  0] 0x00007fff8ad164f0 libsystem_c.dylib`strlen + 16

     0x00007fff8ad164e4:     movl %edi, %ecx
     0x00007fff8ad164e6:     movq %rdi, %rdx
     0x00007fff8ad164e9:     andq $-16, %rdi
     0x00007fff8ad164ed:      orl $-1, %eax
 ->  0x00007fff8ad164f0:  pcmpeqb (%rdi), %xmm0
     0x00007fff8ad164f4:     andl $15, %ecx
     0x00007fff8ad164f7:     shll %cl, %eax
     0x00007fff8ad164f9: pmovmskb %xmm0, %ecx
     0x00007fff8ad164fd:     andl %eax, %ecx
```

Symbolication

Using with Python module

```
[  1] 0x0000000104cc1469 a.out`main [inlined] std::char_traits<char>::length(
[  1] 0x0000000104cc1465 a.out`main [inlined] std::string::assign(char const*
[  1] 0x0000000104cc1465 a.out`main + 29 at main.cpp:11
[  2] 0x0000000104cc143f a.out`start + 51
```

Symbolication

Using with Python module

```
[  1] 0x0000000104cc1469 a.out`main [inlined] std::char_traits<char>::length(
[  1] 0x0000000104cc1465 a.out`main [inlined] std::string::assign(char const*
[  1] 0x0000000104cc1465 a.out`main + 29 at main.cpp:11
[  2] 0x0000000104cc143f a.out`start + 51
```

LLDB Python Package

Take a look at the built-in modules

LLDB Python Package

Take a look at the built-in modules

- Xcode.app/Contents/SharedFrameworks/LLDB.framework

LLDB Python Package

Take a look at the built-in modules

- Xcode.app/Contents/SharedFrameworks/LLDB.framework
    - Resources/Python/lldb

LLDB Python Package

Take a look at the built-in modules

- Xcode.app/Contents/SharedFrameworks/LLDB.framework
  - Resources/Python/lldb
    - lldb/formatters/cpp

LLDB Python Package

Take a look at the built-in modules

- Xcode.app/Contents/SharedFrameworks/LLDB.framework
  - Resources/Python/lldb
    - lldb/formatters/cpp
    - lldb/formatters/objc

LLDB Python Package

Take a look at the built-in modules

- Xcode.app/Contents/SharedFrameworks/LLDB.framework
  - Resources/Python/lldb
    - lldb/formatters/cpp
    - lldb/formatters/objc
    - lldb/utils/symbolication.py

LLDB Python Package

Take a look at the built-in modules

- Xcode.app/Contents/SharedFrameworks/LLDB.framework
    - Resources/Python/lldb
        - lldb/formatters/cpp
        - lldb/formatters/objc
        - lldb/utils/symbolication.py
        - lldb/macosx/crashlog.py

LLDB Python Package

Take a look at the built-in modules

- Xcode.app/Contents/SharedFrameworks/LLDB.framework
  - Resources/Python/lldb
    - lldb/formatters/cpp
    - lldb/formatters/objc
    - lldb/utils/symbolication.py
    - lldb/macosx/crashlog.py
    - lldb/macosx/heap.py

Overview

Debugging with LLDB

Examples

Introduction LLDB in depth Conclusion

Overview

Debugging with LLDB

Conclusion

Introduction LLDB in depth Examples

Conclusion

Wrapping things up

Conclusion

Wrapping things up

- Customizable

Conclusion

Wrapping things up

- Customizable
  - Type formats

Conclusion

Wrapping things up

- Customizable
  - Type formats
  - Type summaries

Conclusion

Wrapping things up

- Customizable
  - Type formats
  - Type summaries
- Improved multi-threaded debugging

Conclusion

Wrapping things up

- Customizable
  - Type formats
  - Type summaries
- Improved multi-threaded debugging
- Compiler integration

Conclusion

Wrapping things up

- Customizable
  - Type formats
  - Type summaries
- Improved multi-threaded debugging
- Compiler integration
  - Rethink what you can do with expressions

Conclusion

Wrapping things up

- Customizable
  - Type formats
  - Type summaries
- Improved multi-threaded debugging
- Compiler integration
  - Rethink what you can do with expressions
- LLDB.framework

Conclusion

Wrapping things up

- Customizable
  - Type formats
  - Type summaries
- Improved multi-threaded debugging
- Compiler integration
  - Rethink what you can do with expressions
- LLDB.framework
- Python integration

More Information

Michael Jurewitz
Developer Tools Evangelist
jury@apple.com

Documentation
LLDB Website
http://lldb.llvm.org

Python Reference
http://lldb.llvm.org/python-reference.html

Variable Formats and Summaries
http://lldb.llvm.org/varformats.html

Apple Developer Forums
http://devforums.apple.com

Labs

LLDB Lab	Developer Tools Lab C Friday 11:30AM

WWDC2012