QuickDocs | Low Level Debugger (LLDB)

> ***January 2020***

Contents

🏠 | ⬅️ | ➡️

1. Overview / Cheat Sheet

`[Search Tags: >overview >cheatsheet >brief >review >revision >reminder >quickreminder >viewover >fastreview >quickreview]`

🏠 | ⬅️ | ➡️

1.1. Beginner Usage Overview

[Search Tags: >]

> TODO: ## 1.1. Beginner Usage Overview

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Quisque id diam vel quam elementum pulvinar. Orci nulla pellentesque dignissim enim. Magna fringilla urna porttitor rhoncus dolor purus. Mollis nunc sed id semper risus in hendrerit gravida rutrum. Faucibus turpis in eu mi bibendum. Ultrices neque ornare aenean euismod elementum. Consectetur lorem donec massa sapien faucibus. At imperdiet dui accumsan sit amet nulla facilisi morbi tempus. Rhoncus urna neque viverra justo nec ultrices dui. Sed faucibus turpis in eu mi bibendum.

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | n/a | n/a | n/a |

🏠 | ⬅️ | ➡️

1.2. Intermediate Usage Overview

[Search Tags: >]

> TODO: ## 1.2. Intermediate Usage Overview

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Quisque id diam vel quam elementum pulvinar. Orci nulla pellentesque dignissim enim. Magna fringilla urna porttitor rhoncus dolor purus. Mollis nunc sed id semper risus in hendrerit gravida rutrum. Faucibus turpis in eu mi bibendum. Ultrices neque ornare aenean euismod elementum. Consectetur lorem donec massa sapien faucibus. At imperdiet dui accumsan sit amet nulla facilisi morbi tempus. Rhoncus urna neque viverra justo nec ultrices dui. Sed faucibus turpis in eu mi bibendum.

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Documentation Archive | Apple | GDB and LLDB Command Examples |

🏠 | ⬅️ | ➡️

1.3. Cheat Sheet

`[Search Tags: >]`

> TODO: ## 1.3. Cheat Sheet

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Quisque id diam vel quam elementum pulvinar. Orci nulla pellentesque dignissim enim. Magna fringilla urna porttitor rhoncus dolor purus. Mollis nunc sed id semper risus in hendrerit gravida rutrum. Faucibus turpis in eu mi bibendum. Ultrices neque ornare aenean euismod elementum. Consectetur lorem donec massa sapien faucibus. At imperdiet dui accumsan sit amet nulla facilisi morbi tempus. Rhoncus urna neque viverra justo nec ultrices dui. Sed faucibus turpis in eu mi bibendum.

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | PDF | nesono | `lldb` Cheat Sheet |

🏠 | ⬅️ | ➡️

2. What is it ?

`[Search Tags: >lldb.what? >lldbwhat? >lldb.who? >lldb.why? >lldb.whocares? >lldb.whycare? >lldb.? >lldb? >wat >woot >wut >whatisit ?isit >whatsit >about >description >whycare >caring? >info >intro >lldb.whatisit >lldb.whycare? >lldb.whyshouldicare?]`

What is it ?

- LLDB, is a program – that provides an interface for you to examine the execution of a ***[running]*** program – instruction by instruction, or step by step, or breakpoint by breakpoint.

  > Didnt understand ? Try this:
  >
  > LLDB, short for ***Low-Level-De-Bugger***, is a program – **that provides an interface** *(a way/channel/portal/access-point/control-panel)* **for you to examine the execution** *([individual] actions/steps taken-by/performed-by)* **of a** *[-nother]* **program** – *[CPU]* **instruction by instruction, or** *[source code]* **statement by statement, or breakpoint** *(checkpoint(s) set by you, in the source code)* **by breakpoint.**

- "`lldb` is the default debugger in Xcode on macOS and supports debugging C, Objective-C and C++ on the desktop and iOS devices and simulator."

Why should I care ?

- *The short answer;* because it will save you ***(THOUSANDS of)*** hours of debugging [, and by extention, of your ***LIFE***].

  > What's "***Debugging***" ?
  >
  > Short, oversimplified answer; the process of locating bugs ***[in source code]***, finding their cause(s) and patching (i.e fixing/removing) them.

> ***Furthur Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Documentation | LLDB | The LLDB Debugger \| Official Website |
> | 2 | Encyclopedia | Wikipedia | LLDB (debugger) |

🏠 | ⬅️ | ➡️

3. How do I use it ?

`[Search Tags: >lldb.usage > lldb.use >use >usage >how? >howtouse >usinglldb >howtouse >howdoiuseit >howtousage >lldbhowtouse >uselldb >lldbusage]`

TODO : # 3. How do I use it ?

- ***To use LLDB***:

```
$> clang <source-code-files> -g -O0
$> lldb <execuable>
$> (lldb) b main
$> (lldb) r <arg-1> <arg-2> ... <arg-n>
$> (lldb) gui
```

> In words;
>
> 1. **Compile Program** (§3.1) with **Debug Flags** (§3.1.1)
> 2. **Launch LLDB** (§3.2)
> 3. **Load LLDB** *[with your program]* (§3.3)
> 4. **Setup LLDB** (§3.4) (e.g. entry-point, breakpoints, watchpoints) *[for your program]*
> 5. **Run your program** (§3.5) *(within LLDB)*
> 6. **Launch Graphical User Interface** (3.6) mode
> 7. **Examine the Execution** (§3.7) *[of your program]*.

Annotated Command Line:

```
$> # 1. Compile
$> clang <source-code> -g -O0
$>
$> # 2,3. Launch & Load
$> lldb <execuable>
$>
$> # 4. Setup
$> (lldb) b main  # Choose your entry-point, I chose the `int
main()` function
$> (lldb) b ...   # Add more breakpoints if you want
$>
$> # 5. Run (optionally, with arguments)
$> (lldb) run <arg-1> <arg-2> ... <arg-n>
$>
$> # 6. Launch 'gui' mode
$> (lldb) gui
$>
$> # 7. Examine Code
```

*[**Note**:*

- *There is also a brief section on a set of [compiler] flags, which, basically, make up the other half of the debugging tools/weapons available to you – **the** `"fsanitize"` **family** (§3.1.2) [of flags]. Don't miss it, you'll miss out on a LOT !*

*- **end note**]*

Ressources:

#	Type	Author	Link
1	Documentation	LLDB	(Official) Tutorial
2	Documentation	Apple	LLDB Quick Start Tutorial

🏠 | ⬅️ | ➡️

3.1. Compile Program

`[Search Tags: >lldb.compileprogram >compileprogram >db.compileprogram >compilationstep >compilestep >compprogram]`

Contents

- 2 Sanitize Flags: `fsanitize` family

- ***LLDB works by loading it with** (i.e setting it up to run) **an execuable** [file]*.
- First thing would be to compile a program; and when debugging, don't forget to compile with the ***debugging*** (§3.1.1) and ***sanitizer*** (3.1.2) *[compiler]* flags.
- ***Demonstration:***

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Manual Page | MacOS | `man clang` |

🏠 | ⬅️ | ➡️

3.1.1. Debug Flags: `-g` & `-O0`

`[Search Tags: >compileprogram >compileprg >programcompilation >compilation >lldb.compile.flags >lldb.compile.debugflags >lldb.debugflags >lldb.flags >debugflags >dbflags >deflags >bugflags >dbgflags >dbugflags]`

- ***To see source code in the "examination phase"** (instead of assembly code)*, you'll have to add *[in the compilation step]* the *[compiler]* flag *(specific to the compiler used)* that ***generates debug information***; but if your goal is to see ***assembly***, then omit this step.

> For `clang` and `gcc` *[compilers]*, the flag is: `-g`.

- ***To make sure that no source code is not optimized away*** *(i.e modified by the compiler in order to optimize the [speed and efficiency of a] program)*, you'll have to add *[in the compilation step]* the *[compiler]* flag *(specific to the compiler used)* that ***turns off optimizations***; if you don't do this, then during "*examination phase*", `lldb` might appear to be excuting *[the source code]* in a non-linear manner *(i.e it will (seem to) jump (skip) over some lines and loops, etc)*.

> For `clang` and `gcc` *[compilers]*, the flag is: `-O0`
> Uppercase letter /Oh/ `O`, followed by, the digit /zero/ `0`.

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Manual Page | Unix / Linux / MacOS | `man clang` |
> | 2 | Documentation | Clang | Compiler :: Debug Options |
> | 3 | Documentation | Clang | Compiler :: Documentation |
> | 4 | Documentation | GNU | Compiler :: Debug Options |
> | 5 | Documentation | GNU | Compiler :: Documentation |

🏠 | ⬅️ | ➡️

3.1.2. Sanitizer Flags: `-fsanitize` family

`[Search Tags: >fsanitize=address >fsanitize=undefined >fsanitize=memory >fsanitize=leaks >fsanitize=threads >fsanitize=dataflag >fsanitize=cfi >fsanitize=safestack >fsanitize=safe-stack >fsanitize=data-flow >sanitizers >sanitizerflags >sanitizeflags >sanitizer.flags >flags.sanitizers >debug.sanitizers >debugsanitizers >debugfsanitizers >memoryflags >leakflags >leaksflags >threadflags >dataflowflags >undefinedbehaviorflags >safestackflags >cfiflags >memory.flags >leaks.flags >leak.flags >thread.flags >dataflow.flags >undefinedbehavior.flags >safestack.flags >cfi.flags] >flags.memory >flags.leak >flags.thread >flags.dataflow >flags.undefinedbehavior >flags.safestack >flags.cfi`

> *This section is unrelated to LLDB, but related to debugging.*

- The `fsanitize` family of *[compiler]* flags, is an extraordinarily helpful set of *[compiler]* flags, with regards to debugging. They enable *[compiler]* *runtime* checks — ***which are disabled by default*** — that detect and help avoid bugs. If a check fails, a diagnostic message is produced *(at runtime)* explaining the problem.

  Each *[sanitizer]* performs multiple *(different)* checks, for example: the *UndefinedBehaviorSanitizer* — enabled by *`-fsanitize=undefined`* — performs all the checks listed here (or here (just another good

resource)).

> *[**Note:** For the better diagnostic messages, compile with the `-g` [compiler] flag; the `-O0` [compiler] flag comes naturally along as well, to disable [compiler] optimizations. - **end note**]*

- ***Enable sanitizer** [checks] **:***

> ***Synopsis:***
>
> ```
> $> <compile-command> [-g -O0] [-fsanitize=<sanitizer-flag> ...]
> ```
>
> ***Option(s):***
>
> | Sanitizer | Flag | Description |
> | --- | --- | --- |
> | UndefinedBehaviorSanitizer | `undefined` | *A detector for **undefined behavior**.* |
> | AddressSanitizer | `address` | *A detector for **memory errors** (e.g. segmentation faults).* |
> | MemorySanitizer | `memory` | *A detector for **uninitialized reads**.* |
> | LeakSanitizer | `leak` | *A detector for **memory leakage**.* |
> | ThreadSanitizer | `thread` | *A detector for **data-race**.* |
> | DataFlowSanitizer | `dataflow` | *A general **data flow analysis**.* |
> | Control Flow Integry | `cfi` | ***Control flow** checks.* |
> | SafeStack | `safe-stack` | *Protection against **stack-based memory** corruption errors.* |
>
> ***Example:***
>
> ```
> $> gcc source.c -g -O0 -fsanitize=address -fsanitize=undefined
> ```

> *[**Note:***
>
> - *It is not possible to have more than one the following sanitizers: `-fsanitize=address`, `-fsanitize=thread`, and `-fsanitize=memory`, at the same time.*
> - *Not all sanitizers are, always, supported, on all machines. Just try them.*
>
> *- **end note**]*

> ***Further Reading:***

#	Type	Author	Link
1	Documentation	Apple	Sanitizers Family
2	Documentation	Clang	Sanitizers Family
3	Documentation	GNU	Sanitizers Family (scroll down)

🏠 | ⬅️ | ➡️

3.2. Run LLDB

`[Search Tags: >lldb.run >debugger.run >lldbrun >debuggerrun >run.lldb >run.debugger >runlldb >rundebugger]`

Run the `lldb` debugger *[program]* by typing *[in your command prompt]*:

```
$> lldb
(lldb)
```

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`man lldb`

🏠 | ⬅️ | ➡️

3.3. Load LLDB

`[Search Tags: >loadprogram >loadprocess >loadexecutable >programload >processload >executableload >lldb.loadprogram >loadlldb >loaddebugger >debugger.loadprogram >db.loadprogram >programload >program.load >lldb.unloadprogram >unloadlldb >unloaddebugger >debugger.unloadprogram >db.unloadprogram >programunload >program.unload]`

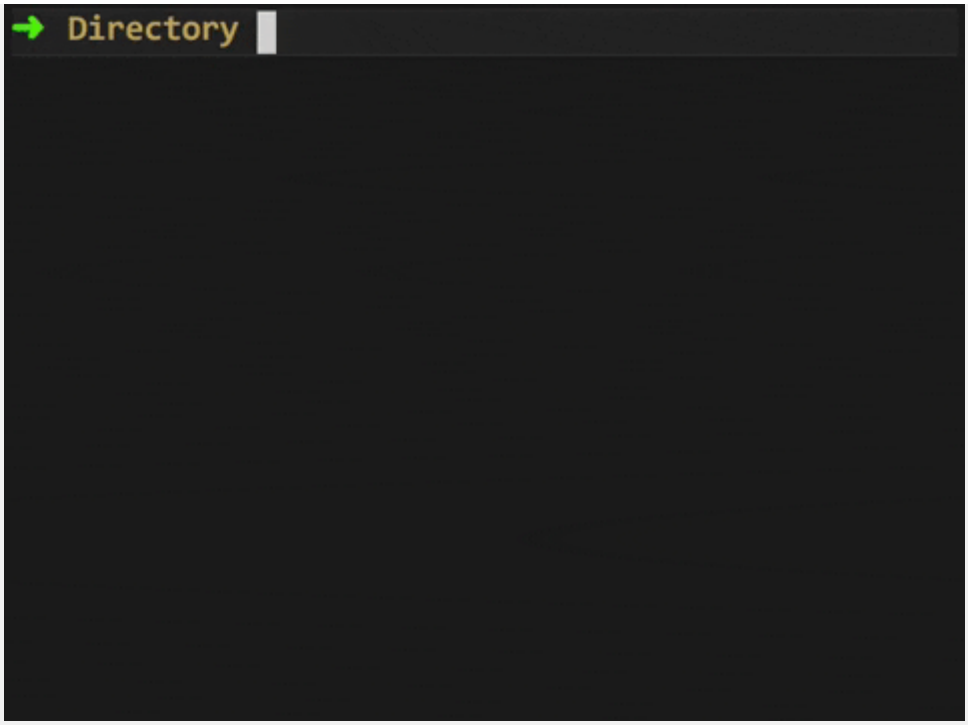
Contents

- `lldb` must *[then]* be informed of which program we intend to debug – this step is referred to as: ***"loading a program"*** . The following subsection explores the basic commands (§3.3.1) of loading programs.

  > In `lldb`'s lexicon, a ***"program [executable] intended for debugging"*** is a referred to as a ***"[debugger] target"***.

> So technically you could say; we are ***loading*** *[`lldb` with]* a ***debugger target***; i.e setting *[`lldb`]* up to target a specific process *(program)*, for debugging/examination-phase.

- ***Demonstration:***



> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Documentation | LLDB | (Official) Tutorial :: Loading a Program into LLDB |
> | 2 | Manual Page | Unix / Linux / MacOS | `(lldb) help target` |

🏠 | ⬅️ | ➡️

3.3.1. Basic Commands

`[Search Tags: >lldbloadcomands >lldb.load.commands >lldbloadcmds >lldb.load.cmds >debuggerloadcomands >debugger.load.commands >debuggerloadcmds >debugger.load.cmds >loadcomands >load.commands >loadcmds >load.cmds >loadbasiccommands >loadbasics >ldbasiccommands >ldbasics >basiccommandsload >basicsload >basiccommandsld >basicsld >loadbasiccmds >loadbasics >ldbasiccmds >ldbasics >basiccmdsload >basicsload >basiccmdsld >basicsld >loadbsccommands >loadbscs >ldbsccommands >ldbscs >bsccommandsload >bscsload >bsccommandsld >bscsld]`

Commands to:

- **Create a target *(i.e load a program executable)*:**

> [Search Tags: >crtarget >crttarget >createtarget >targetcreate >ldtarget >lodtarget >loadtarget >targetload >crtargt >crttargt >createtargt >targtcreate >ldtargt >lodtargt >loadtargt >targtload >crtrgt >crttrgt >createtrgt >trgtcreate >ldtrgt >lodtrgt >loadtrgt >trgtload >crtr >crttr >createtr >trcreate >ldtr >lodtr >loadtr >trload]

> ***From outside [`lldb`]:***
>
> > ***Synopsis:***
> >
> > ```
> > $> lldb --file <program-executable-filename> [<arg> ...]
> > ```
> >
> > ***Example(s):***
> >
> > ```
> > $> lldb --file a.out "arg1" "arg2"    # OR
> > $> lldb -f a.out "arg1" "arg2"        # OR
> > $> lldb a.out "arg1" "arg2"
> > ```

> ***From inside [`lldb`]:***
>
> > ***Synopsis:***
> >
> > ```
> > (lldb) target create <program-executable-filename>
> > ```
> >
> > ***Example(s):***
> >
> > ```
> > (lldb) file a.out                    # OR
> > (lldb) target create a.out           # OR
> > (lldb) ta cr a.out                   # OR
> > (lldb) ta c a.out
> > ```

- **List *(all)* targets:**

> [Search Tags: >lsttarget >listtarget >lstarget >targetlst >targetlist >targetls >lsttargt >listtargt >lstargt >targtlst >targtlist >targtls >lsttrgt >listtrgt >lstrgt >trgtlst >trgtlist >trgtls >lsttr >listtr >lstr >trlst >trlist >trls >litarget >targetli >litargt >targtli >litrgt >trgtli >litr >trli]

> ***Synopsis:***
>
> ```
> (lldb) target list
> ```
>
> ***Example(s):***
>
> ```
> (lldb) target list
> (lldb) ta l
> ```

- **Select *[a different]* target *[as current target]*:**

> [Search Tags: >selcttarget >setarget >selecttarget >targetselect >selcttargt >setargt >selecttargt >targtselect >selcttrgt >setrgt >selecttrgt >trgtselect >selcttr >setr >selecttr >trselect]

> ***Synopsis:***
>
> ```
> (lldb) target select <target-index>
> ```
>
> ***Example(s):***
>
> ```
> (lldb) target select 3       # select the third target, as
> current debugging target
> (lldb) ta se 5
> ```

- **Delete target *(i.e unload an program execuable)*:**

> [Search Tags: >detarget >deltarget >deletetarget >targetdelete >detargt >deltargt >deletetargt >targtdelete >detrgt >deltrgt >deletetrgt >trgtdelete >detr >deltr >deletetr >trdelete >ultarget >unltarget >unletetarget >targetunload >ultargt >unltargt >unloadtargt >targtunload >ultrgt >unltrgt >unloadtrgt >trgtunload >ultr >unltr >unloadtr >trunload]

> ***Synopsis:***

```
(lldb) target delete [<target-ids>]
(lldb) target delete [--all]
```

Example(s):

```
(lldb) target delete 3       # delete the target with `target-
id`: #3
(lldb) tar del 7 5 2         # delete the list of targets: #7 #5
#2
(lldb) ta de --all           # delete all targets
(lldb) ta d -a               # delete all targets
```

---

*[**Section Notes**:*

- "From **outside** *[`lldb`]*"; i.e before, and at the same time as, launch *[of `lldb`] [, on the [terminal] command prompt]*.
- "From **inside** *[`lldb`]*"; i.e after launch *[of `lldb`] [, on the [`lldb`] command prompt]*.
- *`[<arg> ...]`* represents the argument(s) that you pass to a program.
- `file` is an *(built-in)* alias/abbreviation for `target create`, see `help file`.
- To load/debug Python *[scripts]*:

  ```
  $> lldb -f python -- <script>
  ```

  or

  ```
  (lldb) ta cr /path/to/<python>
  (lldb) r <script>
  ```

  > *This also works for other scripting language interpreters and their [type of] scripts.*

*- **end section notes**]*

Further Reading:

#	Type	Author	Link
1	Documentation	LLDB	(Official) Tutorial :: Loading a Program into LLDB

#	Type	Author	Link
2	Manual Page	Unix / Linux / MacOS	`man lldb`
3	Manual Page	Unix / Linux / MacOS	`(lldb) help target create`
4	Manual Page	Unix / Linux / MacOS	`(lldb) help target list`
5	Manual Page	Unix / Linux / MacOS	`(lldb) help target select`
6	Manual Page	Unix / Linux / MacOS	`(lldb) help target delete`

🏠 | ⬅️ | ➡️

3.4. Setup LLDB

`[Search Tags: >lldb.setup >debugger.setup >lldbsetup >debuggersetup >setuplldb >setupdebugger]`

- `lldb` *[then]* gives you the possibility of setting up *breakpoints* (§3.4.1) — locations in your program to pause execution at *[and examine the current state of things]* — and *watchpoints* (§3.4.2) — a logger, for certain variables and/or memory-regions, logging *[incoming]* operations (`reads` & `writes`).

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help breakpoint`
2	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint`
3	Documentation	LLDB	(Official) Tutorial :: Setting Breakpoints
5	Documentation	LLDB	(Official) Tutorial :: Setting Watchpoints

🏠 | ⬅️ | ➡️

3.4.1. Breakpoints

`[Search Tags: >lldb.breakpoints >debugger.breakpoints >lldbbreakpoints >debuggerbreakpoints >sectionbreakpoint >breakpointsections >sectionbreakpt >breakptsections >sectionbrkpt >brkptsections >sectionbrpt >brptsections >sectionbr >brsections]`

Contents

> *"In software development, a breakpoint is an intentional stopping or pausing place in a program, put in place for debugging purposes. It is also sometimes simply referred to as a 'pause'.*
>
> *More generally, a breakpoint is a means of acquiring knowledge about a program during its execution. During the interruption, the programmer inspects the test environment (general purpose registers, memory, logs, files, etc.) to find out whether the program is functioning as expected. In practice, a breakpoint consists of one or more conditions that determine when a program's execution should be interrupted. [...]"*
>
> — *Wikipedia :: Breakpoints*

The following subsections dive into the ***basic*** (§3.4.1.1) as well as [slightly] more ***advanced commands*** (§3.4.1.2) ***[for operating on breakpoints]***, then into the available ***breakpoint options*** (§3.4.1.3) and how to utilise them, also, we discover the [amazing] usefulness and power of ***breakpoint names*** (§3.4.1.4) and that of configuring their options, additionally we briefly see the breakpoint facilites for multi-threaded programs (§3.4.1.5) and, finally, a few ***C++ (§3.4.1.6) [specific] breakpoint commands***.

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Encyclopedia | Wikipedia | Breakpoints |
> | 2 | Documentation | LLDB | (Official) Tutorial :: Setting Breakpoints |
> | 3 | Documentation | LLDB | (Official) Tutorial :: Breakpoint Names |
> | 4 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint` |

🏠 | ⬅️ | ➡️

3.4.1.1. Basic Commands

`[Search Tags: >basicbreakpoint >basicsbreakpoint >breakpointbasics >basicbreakpt >basicsbreakpt >breakptbasics >basicbrkpt >basicsbrkpt >brkptbasics >basicbrpt >basicsbrpt >brptbasics >basicbr >basicsbr >brbasics]`

Contents

> *(Basic) Commands for operating on breakpoints.*

- ***Set a breakpoint, on a function:***

> [Search Tags: >brset >setbr >sbr >sebr >breakpointset >breakpointfunctions >breakpointfuncs >breakpointfts >breakpointfcs >breakpointmain >setbreakpoint >brkptset >setbrkpt >breakptset >setbreakpt >brsetfunctions >brfunctions >brfuncs >brfts >brfcs >brsfunctions >brsfuncs >brsfts >brsfcs >brsmain]

> ***Synopsis:***

```
breakpoint set --name <function-name>
b <function-name>
```

> ***Example(s):***

```
(lldb) breakpoint set --name main
(lldb) br s -n main
(lldb) b main
```

> *[**Note**: Only the function **itself** has a breakpoint set on it – call-sites [of the said function] are ignored. - **end note**]*

- ***Set a breakpoint, on a [source code] line:***

> [Search Tags: >brsrcs >brssrcs >brsetsrcs >breakpointsetsrcs >brsources >brssources >brsetsources >breakpointsetsources >brsetfiles >brfiles >brsrcs >brpages >brpgs >brsfiles >brssrcs >brspages >brspgs] >brsetlines >brlines >brlis >brpages >brslines >brslis >brspages >brsls >breakpointlines >breakpointlis >breakpointls >breakpointfcs >breakpointmainlines

> ***Synopsis:***

```
breakpoint set --file <filename> --line <line-number>
b <filename>:<line-number>
```

> ***Example(s):***

```
(lldb) breakpoint set -f hello.c -l 10
(lldb) br s -f hello.c -l 10
(lldb) b hello.c:10
```

- ***List breakpoints:***

> [Search Tags: >breakpointlist >breakpointls >listbreakpoint >lstbreakpoint >lsbreakpoint >libreakpoint >breakptlist >breakptls >listbreakpt >lstbreakpt >lsbreakpt >libreakpt >brkptlist >brkptls >listbrkpt >lstbrkpt >lsbrkpt >librkpt >brptlist >brptls >listbrpt >lstbrpt >lsbrpt >librpt >brlist >brls >listbr >lstbr >lsbr >libr]

> ***Synopsis:***

```
breakpoint list -[bfv] [<breakpt-id> ...]
```

> ***Example(s):***

```
(lldb) breakpoint list --brief 3 2    # --brief    (minimum
description)
(lldb) br l -f 1                      # --full     (full
description, default
(lldb) br l -v                        # --verbose  (extensive
description)
```

- ***Delete breakpoint(s):***

> [Search Tags: >brdelete >deletebr >deletebrpt >deletebrkpt >deletebreakpt >deletebreakpoint >brdelete >brptdelete >brkptdelete >breakptdelete >breakpointdelete >debr >deletebr >deletebreakpoint >brunload >uldbr >breakpointunload >unloadbreakpoint >delbr >brdel >delbrpt >brptdel >delbrkpt >brkptdel >delbreakpt >breakptdel >delbreakpoint >breakpointdel]

> ***Synopsis:***

```
breakpoint delete [<breakpt-ids | breakpt-name>]
```

> ***Example(s):***

```
(lldb) breakpoint delete 5
(lldb) br de 1 2 3
(lldb) br d                           # delete all breakpts
```

> *[**Note**:*
>
> - *If no breakpoint [id] is specified, [the command will] delete them [the current breakpoints] all.*
> - `lldb`*, automatically, deletes breakpoints of targets that are deleted.*
>
> *- **end note**]*

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint set` |
> | 2 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint list` |
> | 3 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint delete` |

🏠 | ⬅️ | ➡️

3.4.1.2. Advanced Commands

`[Search Tags: >advancedbreakpoint >breakpointadvanceds >advancedbreakpt >breakptadvanceds >advancedbrkpt >brkptadvanceds >advancedbrpt >brptadvanceds >advancedbr >bradvanceds >advbreakpoint >breakpointadvs >advbreakpt >breakptadvs >advbrkpt >brkptadvs >advbrpt >brptadvs >advbr >bradvs]`

Commands to:

> *[A lil' more] Advanced commands for operating on breakpoints.*

- ***Enable / Disable breakpoints:***

> `[Search Tags: >breakpointenable >breakptenable >brkptenable >brptenable >brenable >breakpointdisable >breakptdisable >brkptdisable >brptdisable >brdisable >enablebreakpoint >enablebreakpt >enablebrkpt >enablebrpt >enablebr >disablebreakpoint >disablebreakpt >disablebrkpt >disablebrpt >disablebr]`
>
> ***Synopsis:***

```
(lldb) breakpoint disable [<breakpt-id | breakpt-name> ...]
(lldb) breakpoint enable  [<breakpt-id | breakpt-name> ...]
```

Example(s):

```
(lldb) breakpoint disable 1
(lldb) br di 1
```

```
(lldb) breakpoint disable 3.*      # disable all breakpoints of
ID 3.
(lldb) br di 3.*
```

```
(lldb) breakpoint enable 2 6 3.2   # enable breakpoints: 2, 6 and
3.2
(lldb) br en 2 6 3.2
```

```
(lldb) breakpoint enable 'funcs'   # enable breakpoints who have
'funcs' in their list of names
(lldb) br en 'funcs'
```

*[**Note**:*

- *To enable only certain locations of a logical breakpoint, use the breakpoint disable command, passing the breakpoint ID followed by a dot-separated wildcard character (`*`), e.g.: `1.*` or `3.*`.*
- *It is also possible to set, initialy disabled, breakpoints:*

  ```
  (lldb) breakpoint set <breakpt-definition> [--disable]
  ```

*- **end note**]*

- ***Set a breakpoint, on function(s), using regular-expressions:***

```
[Search Tags: >regexftbreakpoint >regexfcbreakpoint >regexfuncbreakpoint
>regexfunctionbreakpoint >regexftbreakpt >regexfcbreakpt >regexfuncbreakpt
>regexfunctionbreakpt >regexftbrkpt >regexfcbrkpt >regexfuncbrkpt >regexfunctionbrkpt
>regexftbrpt >regexfcbrpt >regexfuncbrpt >regexfunctionbrpt >regexftbr >regexfcbr
>regexfuncbr >regexfunctionbr >breakpointregexfts >breakpointregexfcs
```

> >breakpointregexfuncs >breakpointregexfunctions >breakptregexfts >breakptregexfcs >breakptregexfuncs >breakptregexfunctions >brkptregexfts >brkptregexfcs >brkptregexfuncs >brkptregexfunctions >brptregexfts >brptregexfcs >brptregexfuncs >brptregexfunctions >brregexfts >brregexfcs >brregexfuncs >brregexfunctions]

> ***Synopsis:***

```
breakpoint set --func-regex <regular-expression>
```

> ***Example(s):***

```
(lldb) breakpoint set --func-regex 'Parser.{3,4,5}_Command\(\)'
(lldb) br s -r "Parser.{3,4,5}_Command\(\)"
```

> *[**Note**: Function call-sites also count as matches, and get a breakpoint. - **end note**]*

- ***Set a breakpoint, on line(s), in file(s), using regular-expressions:***

> [Search Tags: >regexsrcbreakpoint >regexsrcebreakpoint >regexsourcebreakpoint >regexsrcbreakpt >regexsrcebreakpt >regexsourcebreakpt >regexsrcbrkpt >regexsrcebrkpt >regexsourcebrkpt >regexsrcbrpt >regexsrcebrpt >regexsourcebrpt >regexsrcbr >regexsrcebr >regexsourcebr >breakpointregexsrcs >breakpointregexsrces >breakpointregexsources >breakptregexsrcs >breakptregexsrces >breakptregexsources >brkptregexsrcs >brkptregexsrces >brkptregexsources >brptregexsrcs >brptregexsrces >brptregexsources >brregexsrcs >brregexsrces >brregexsources]

> ***Synopsis:***

```
breakpoint set --all-files --source-pattern-regexp <regular-
expression>                # Search in all files
breakpoint set [--file <files> ...] --source-pattern-regexp
<regular-expression>       # Search (only) in specified files
```

> ***Example(s):***

```
(lldb) breakpoint set --all-files --source-pattern-regexp 'return
\(FAILURE\);'
(lldb) br s -A -p 'return \(FAILURE\);'
```

```
(lldb) breakpoint set --file 'core.c' --file 'cleanup.c' --
source-pattern-regexp 'if \('
(lldb) br s -f core.c -f cleanup.c -p 'if \('
```

*[**Note:***

- *Source file(s) are specified with the `-f` option. The `-f` option can be specified more than once. If no source files are specified, uses the current "default source file".*
- *You cannot specific multiple regex-patterns at the same time; i.e you cannot do the following : `... -p <pattern> -p <pattern>`. It [`lldb`] will only search for the last pattern.*

*- **end note**]*

- ***Modify [existent] breakpoint(s) [option(s)] :***

```
[Search Tags: >modbreakpoint >modifbreakpoint >mobreakpoint >mbreakpoint
>modifybreakpoint >breakpointmodify >modbreakpt >modifbreakpt >mobreakpt >mbreakpt
>modifybreakpt >breakptmodify >modbrkpt >modifbrkpt >mobrkpt >mbrkpt >modifybrkpt
>brkptmodify >modbrpt >modifbrpt >mobrpt >mbrpt >modifybrpt >brptmodify >modbr
>modifbr >mobr >mbr >modifybr >brmodify >brmoattributes >brmoattribs >brmoconditions
>brmoconds >brmodifyattributes >brmodifyattribs >brmodifyconditions >brmodifyconds]
```

Brief

`modify` *lets one modify previously* `set` *[breakpoint] options [of [existent] breakpoint(s)].*

Synopsis:

```
breakpoint modify <cmd-options> [<breakpt-id | breakpt-name> ...]
```

Command Options:

Command Options	Abrv.	Description
`--disable`	`-d`	*Disable the breakpoint.*
`--enable`	`-e`	*Enable the breakpoint.*
`--condition <condition-expr>`	`-c`	*The breakpoint stops only if this condition expression evaluates to true.*

Command Options	Abrv.	Description
`--ignore-count <count>`	`-i`	*Set the number of times this breakpoint is skipped before stopping.*
`--auto-continue <boolean>`	`-G`	*The breakpoint will auto-continue after running its commands.*
`--one-shot <boolean>`	`-o`	*The breakpoint is deleted the first time it stop causes a stop.*
`--thread-index <thread-index>`	`-x`	*The breakpoint stops only for the thread whose index matches this argument.*
`--thread-id <thread-id>`	`-t`	*The breakpoint stops only for the thread whose TID matches this argument.*
`--thread-name <thread-name>`	`-T`	*The breakpoint stops only for the thread whose thread name matches this argument.*

Example(s):

```
(lldb) breakpoint modify --disable 3
(lldb) br m -d 3
```

```
(lldb) breakpoint modify --condition 'ac < 2' --one-shot true 4 2
7
(lldb) br m -c 'ac < 2' -o true 4 2 7
```

```
(lldb) breakpoint modify --condition '' 'controlFlow'     #
clears any existent condition, for all breakpoints who have
'controlFlow' in their list of names
(lldb) br m -c '' 'controlFlow'
```

```
(lldb) breakpoint modify --thread-id 6 8                    #
change the thread assigned for the breakpoint of id: 8
(lldb) br m -t 6 8
```

*[**Note:***

- *Passing an empty argument (i.e: `''`) [to flags] clears the modification(s) — except for `--enable` (`-e`), `--disable` (`-d`) and `--ignore-count` (`-i`) [flags], .*
- *If no breakpoint is specified, acts on the last created breakpoint.*

> - *To set/modify breakpoint commands, see `breakpoint command`.*
>
> *- **end note**]*

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint set` |
> | 2 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint enable` |
> | 3 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint disable` |
> | 4 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint modify` |

🏠 | ⬅️ | ➡️

3.4.1.3. Options

`[Search Tags: >optsbreakpoint >breakpointopts >breakpointops >opsbreakpoint >optionsbreakpoint >breakpointoptions >optsbreakpt >breakptopts >breakptops >opsbreakpt >optionsbreakpt >breakptoptions >optsbrkpt >brkptopts >brkptops >opsbrkpt >optionsbrkpt >brkptoptions >optsbrpt >brptopts >brptops >opsbrpt >optionsbrpt >brptoptions >optsbr >bropts >brops >opsbr >optionsbr >broptions]`

Commands for:

> *(**Know that**) — "Breakpoints carry two orthognal sets of information: one specifies where to set the breakpoint, and the other how to react when the breakpoint is hit. The latter set of information (e.g. commands, conditions hit-count, auto-continue…) we call breakpoint options."*
>
> *— LLDB :: Tutorial :: Breakpoint Names*

> *[**Note**: We'll refer to options that are neither [breakpoint] conditions nor [breakpoint] commands as: "[breakpoint] attributes", e.g.: hit-count, auto-continue, etc… - **end note**]*

- **Breakpoint Conditions:**

> [Search Tags: >breakpointsetcondition >breakptsetcondition >brkptsetcondition >brptsetcondition >brsetcondition >breakpointaddcondition >breakptaddcondition >brkptaddcondition >brptaddcondition >braddcondition >conditionbreakpoint >conditionbreakpt >conditionbrkpt >conditionbrpt >conditionbr >breakpointconditions >breakptconditions >brkptconditions >brptconditions >brconditions >breakpointconds >breakptconds >brkptconds >brptconds >brconds]

  - **Set a conditional breakpoint:**

> ***Synopsis:***
>
> ```
> breakpoint set <breakpt-definition> [--condition <expr>]
> ```
>
> ***Example(s):***
>
> ```
> (lldb) breakpoint set --line 14 --condition 'argc < 2'
> (lldb) br s -l 14 -c 'argc < 2'
> ```
>
> ```
> (lldb) breakpoint set --name baz --condition '(int)strcmp(y,
> "hello") == 0'
> (lldb) br s -n baz -c '(int)strcmp(y, "hello") == 0'
> ```

  - **Add/Modify breakpoint condition *[of an existing breakpoint]*:**

> ***Synopsis:***
>
> ```
> breakpoint modify [--condition <expr>] [<breakpt-id |
> breakpt-name> ...]
> ```
>
> ***Example(s):***

```
(lldb) breakpoint modify --condition 'my_var == 42' 3     #
add condition to breakpt with ID: 3
(lldb) br m -c 'my_var == 42' 4 2 8
```

```
(lldb) breakpoint modify --condition '' 'controlFlow'     #
clears any existent condition, for all breakpoints who have
'controlFlow' in their list of names
(lldb) br m -c '' 'controlFlow'
```

- **Breakpoint Commands:**

```
[Search Tags: >commandbreakpoint >commandsbreakpoint >breakpointcommands
>commandbreakpt >commandsbreakpt >breakptcommands >commandbrkpt >commandsbrkpt
>brkptcommands >commandbrpt >commandsbrpt >brptcommands >commandbr >commandsbr
>brcommands >cmdbreakpoint >cmdsbreakpoint >breakpointcmds >cmdbreakpt >cmdsbreakpt
>breakptcmds >cmdbrkpt >cmdsbrkpt >brkptcmds >cmdbrpt >cmdsbrpt >brptcmds >cmdbr
>cmdsbr >brcmds >breakpointscrpt >scrptbreakpoint >scriptbreakpoint
>breakpointscripts >breakptscrpts >scrptbreakpts >scriptbreakpts >breakptscripts
>brkptscrpts >scrptbrkpts >scriptbrkpts >brkptscripts >brptscrpts >scrptbrpts
>scriptbrpts >brptscripts >brscrpts >scrptbr >scriptbr >brscript]
```

  - **Add/Modify *[breakpoint]* command(s):**

```
[Search Tags: >breakpointaddcommands >breakpointaddcmds >breakpointacmds
>breakptaddcommands >breakptaddcmds >breakptadcmds >brkptaddcommands
>brkptaddcmds >brkptadcmds >brptaddcommands >brptaddcmds >brptadcmds
>braddcommands >braddcmds >bradcmds >breakpointaddscripts >breakpointaddscrpts
>breakpointadscrpts >breakptaddscripts >breakptaddscrpts >breakptadscrpts
>brkptaddscripts >brkptaddscrpts >brkptadscrpts >brptaddscripts >brptaddscrpts
>brptadscrpts >braddscripts >braddscrpts >bradscrpts >breakpointsetcommands
>breakpointsetcmds >breakpointacmds >breakptsetcommands >breakptsetcmds
>breakptsecmds >brkptsetcommands >brkptsetcmds >brkptsecmds >brptsetcommands
>brptsetcmds >brptsecmds >brsetcommands >brsetcmds >brsecmds
>breakpointsetscripts >breakpointsetscrpts >breakpointsescrpts
>breakptsetscripts >breakptssetscrpts >breakptssescrpts >brkptssetscripts
>brkptssetscrpts >brkptssescrpts >brptssetscripts >brptssetscrpts
>brptssescrpts >brsetscripts >brsetscrpts >brsescrpts]
```

Synopsis:

```
breakpoint command add [--script-type <type>] [<breakpt-id |
breakpt-name> ...]
```

Then you are prompted:

```
> Enter your debugger command(s). Type 'DONE' to end.
> <lldb-command>
> <lldb-command>       # one command per line
> ...
> DONE
```

Example(s):

```
(lldb) breakpoint command add 2 4 1                    # add
command to breakpoints [of ID]: 2, 4 and 1
Enter your debugger command(s). Type 'DONE' to end.
> thread backtrace
> frame variable
> DONE

(lldb) br co a 2 4 1
Enter your debugger command(s). Type 'DONE' to end.
> bt               # alias, see: help bt
> fr v             # shorthand, for "frame variable"
> DONE
```

```
(lldb) breakpoint command add --script-type python 4
Enter your Python command(s). Type 'DONE' to end.
> print "Hit this breakpoint!"
> DONE
```

```
(lldb) script
>>> bp_count = 0
>>> quit()
...
(lldb) br co add -s python 1.1
Enter your Python command(s). Type 'DONE' to end.
> global bp_count
> bp_count = bp_count + 1
> print "Hit this breakpoint " + repr(bp_count) + " times!"
> DONE
```

[Note:

- In this case, since there is a reference to a global variable, `bp_count`, you will also need to make sure `bp_count` exists and is initialized:

```
(lldb) script
>>> bp_count = 0
>>> quit()
```

  Your Python code, however organized, can optionally return a value. If the returned value is `False`, that tells LLDB not to stop at the breakpoint to which the code is associated. Returning anything other than `False`, or even returning None, or even omitting a return statement entirely, will cause `lldb` to stop.

- You can, alternatively, specify one-liner commands with the: `--one-liner` or `-o` option, followed by the desired `<command>`.

- ***end note]***

- **List *[breakpoint]* command(s):**

> `[Search Tags: >breakpointlistcommands >breakpointlistcmds >breakpointlscmds >breakptlistcommands >breakptlistcmds >breakptlscmds >brkptlistcommands >brkptlistcmds >brkptlscmds >brptlistcommands >brptlistcmds >brptlscmds >brlistcommands >brlistcmds >brlscmds >breakpointlicmds >breakptlicmds >brkptlicmds >brptlicmds >brlicmds >lsbreakptcmds >lsbrkptcmds >lsbrptcmds >lsbreakptcmds >lsbrkptcmds >lsbrptcmds >lstbreakptcmds >lstbrkptcmds >lstbrptcmds >lstbrcmds >listbreakptcmds >listbrkptcmds >listbrptcmds >listbrcmds >lsbrcmds >lsbreakptcommands >lsbrkptcommands >lsbrptcommands >lsbreakptcommands >lsbrkptcommands >lsbrptcommands >lstbreakptcommands >lstbrkptcommands >lstbrptcommands >lstbrcommands >listbreakptcommands >listbrkptcommands >listbrptcommands >listbrcommands >lsbrcommands]`

> ***Synopsis:***

```
breakpoint command list <breakpt-id>
```

> ***Example(s):***

```
(lldb) breakpoint command list 1.1
(lldb) br co li 1.1
```

- **Delete *[breakpoint]* command(s):**

> [Search Tags: >breakpointdeletecommands >breakpointdeletecmds >breakpointdecmds >breakptdeletecommands >breakptdeletecmds >breakptdecmds >brkptdeletecommands >brkptdeletecmds >brkptdecmds >brptdeletecommands >brptdeletecmds >brptdecmds >brdeletecommands >brdeletecmds >brdecmds]

> ***Synopsis:***
>
> ```
> breakpoint command delete <breakpt-id>
> ```
>
> ***Example(s):***
>
> ```
> (lldb) breakpoint command delete 1.1
> (lldb) br co de 1.1
> ```

- **Breakpoint Attributes:**

> [Search Tags: >attributebreakpoint >attribbreakpoint >atbbreakpoint >breakpointatbs >breakpointatts >breakpointattributes >breakpointattribs >attributebreakpt >attribbreakpt >atbbreakpt >breakptatbs >breakptatts >breakptattributes >breakptattribs >attributebrkpt >attribbrkpt >atbbrkpt >brkptatbs >brkptatts >brkptattributes >brkptattribs >attributebrpt >attribbrpt >atbbrpt >brptatbs >brptatts >brptattributes >brptattribs >attributebr >attribbr >atbbr >bratbs >bratts >brattributes >brattribs]

  - **Set a breakpoint, with attribute(s):**

> ***Synopsis:***
>
> ```
> breakpoint set <breakpt-definition> <conditions>
> [<attribute> <boolean> ...]
> ```
>
> ***Command Options:***

Attribute	Abrv.	Description
`--ignore-count <count>`	`-i`	Set the number of times this breakpoint is skipped before stopping.
`--auto-continue <boolean>`	`-G`	The breakpoint will auto-continue after running its commands.

Attribute	Abrv.	Description
`--one-shot <boolean>`	`-o`	The breakpoint is deleted the first time it stops causes a stop.
`--move-to-nearest-code <boolean>`	`-m`	Move breakpoints to nearest code.

Example(s):

```
(lldb) breakpoint set --name foo --ignore-count 5 --one-shot
true
(lldb) br s -n foo -i 5 -o true
```

```
(lldb) breakpoint set --name bar --condition 'argc > 3' --
auto-continue true --ignore-count 5
(lldb) br s -n bar -c 'argc > 3' -G true -i 5
```

- **Add/Modify breakpoint attributes:**

Synopsis:

```
breakpoint modify <attributes> [<breakpt-id | breakpt-name>
...]
```

Example(s):

```
(lldb) breakpoint modify --auto-continue true 1.1
(lldb) br m -G true 1.1
```

```
(lldb) breakpoint modify --one-shot true 4 2 7
(lldb) br m -o true 4 2 7
```

Further Reading:

#	Type	Author	Link

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help breakpoint set`
2	Manual Page	Unix / Linux / MacOS	`(lldb) help breakpoint command`
3	Manual Page	Unix / Linux / MacOS	`(lldb) help breakpoint command add`
4	Manual Page	Unix / Linux / MacOS	`(lldb) help breakpoint command list`
5	Manual Page	Unix / Linux / MacOS	`(lldb) help breakpoint command delete`

🏠 | ⬅️ | ➡️

3.4.1.4. Names

`[Search Tags: >namedbreakpoint >nbreakpoint >namesbreakpoint >breakpointnames >namedbreakpt >nbreakpt >namesbreakpt >breakptnames >namedbrkpt >nbrkpt >namesbrkpt >brkptnames >namedbrpt >nbrpt >namesbrpt >brptnames >namedbr >nbr >namesbr >brnames]`

Commands to:

Breakpoint name is an extremely powerful *[`lldb`]* feature. It allows us to create a breakpoint "profile", so to speak, — a set, of breakpoint options — referrable by `name`. Later on, when we create *(`set`)* breakpoints, we can choose to have them *[the breakpoints we are creating (`set`'ing)]* inherit *(have added to their list of `names`)* one or more *[profile]* `names`, i.e one or more of these *[pre-defined]* sets of options. Any modification to a *[breakpoint]* profile immediately applies to all breakpoints that inherit that profile *(have that [profile] `name` in their list of [profile] `names`)*.

Breakpoint `names` *(profiles)* live independantly of breakpoints that inherit them, and *[existent]* breakpoints all together, allowing them to persist even after all breakpoints are deleted. Down below are discussed the breakpoint commands used to **create** / **list** / **delete** / **configure** breakpoint `names` *(profiles)* and how to make your breakpoint(s) inherit/be-named *(have added to their list of `names` (profiles))* a `name` *(profile)*.

> *For more on **breakpoint names**, and why they are the best, powerful and modular way of using breakpoints, see: (Official) Tutorial :: Breakpoint Names.*

- **Add breakpoint name(s):**

> [Search Tags: >breakpointsetnames >breakptsetnames >brkptsetnames >brptsetnames >brsetnames >setbreakpointnames >setbreakptnames >setbrkptnames >setbrptnames >setbrnames >namebreakpoints >namebreakpts >namebrkpts >namebrpts >namebrs >addbreakpointnames >breakpointaddnames >breakpointanames >addbreakptnames >breakptaddnames >breakptanames >addbrkptnames >brkptaddnames >brkptanames >addbrptnames >brptaddnames >brptanames >addbrnames >braddnames >branames >createbreakpointnames >breakpointcreatenames >createbreakptnames >breakptcreatenames >createbrkptnames >brkptcreatenames >createbrptnames >brptcreatenames >createbrnames >brcreatenames]

Synopsis:

> *At creation (seting) [of breakpoint]:*

```
breakpoint set <breakpt-definition> --breakpoint-name
<breakpt-name>
```

> *After creation (seting) [of breakpoint]:*

```
breakpoint name add --name <breakpt-name> [<breakpt-id |
breakpt-name> ...]
```

Example(s):

```
(lldb) breakpoint name add --name 'controlFlow'
(lldb) br n a -N 'controlFlow'
```

> *To clarify — we are just creating an [un-configured] breakpoint name (profile), namely: "controlFlow" — it is (implicitly) added to the last created (i.e set) breakpoint.*

```
(lldb) breakpoint set --name foo --breakpoint-name 'funcs'
(lldb) br s -n foo -N 'funcs'
```

> *To clarify — we are adding a breakpoint name — namely 'funcs' — to the list of names of the breakpoint [as we set it].*

```
(lldb) breakpoint set --all-files --source-pattern-regexp 'return
\(FAILURE\);' --breakpoint-name 'failure'
(lldb) br s -A -p 'return \(FAILURE\);' -N 'failure'
```

> *To clarify — we are adding a breakpoint name to the list of names of the breakpoint(s) [as we `set` it/them].*

```
(lldb) breakpoint name add --name 'funcs' 3 2 7
(lldb) br n a -N 'funcs' 3 2 7
```

> *To clarify — we are `add`ing a breakpoint name to the list of names of the breakpoints [of id]: 3, 2 and 7.*

```
(lldb) breakpoint name add --name 'important' 'funcs'
'controlFlow'
(lldb) br n a -N 'important' 'funcs' 'controlFlow'
```

> *To clarify — we are adding a breakpoint name to the list of names of the breakpoints that have in their list of names the name: `'funcs'` and/or `'controlFlow'`.*

[Note:

- *Every created (`add`'ed) `name`, if not given any `<breakpt-id | breakpt-name>` will (implicitly) be added to [the list of names of] the last created breakpoint — evidently you must have at least one [existent] breakpoint before creating (`add`'ing) breakpoint `names`.*

*- **end note**]*

- ***List [breakpoint] names:***

> `[Search Tags: >libreakpointnames >lsbreakpointnames >listbreakpointnames >breakpointlinames >breakpointlsnames >breakpointlistnames >libreakptnames >lsbreakptnames >listbreakptnames >breakptlinames >breakptlsnames >breakptlistnames >librkptnames >lsbrkptnames >listbrkptnames >brkptlinames >brkptlsnames >brkptlistnames >librptnames >lsbrptnames >listbrptnames >brptlinames >brptlsnames >brptlistnames >librnames >lsbrnames >listbrnames >brlinames >brlsnames >brlistnames >breakpointnamels >breakpointnameli >breakpointnamelist >breakptnamels >breakptnameli >breakptnamelist >brkptnamels >brkptnameli >brkptnamelist >brptnamels >brptnameli >brptnamelist >brnamels >brnameli >brnamelist >breakpointnals >breakpointnls >breakpointnali >breakpointnli >breakpointnalist >breakpointnlist >breakptnals >breakptnls >breakptnali >breakptnli >breakptnalist >breakptnlist >brkptnals >brkptnls >brkptnali >brkptnli >brkptnalist >brkptnlist >brptnals >brptnls >brptnali >brptnli >brptnalist >brptnlist >brnals >brnls >brnali >brnli >brnalist >brnlist]`

> ***Synopsis:***

```
breakpoint name list
```

Example(s):

```
(lldb) breakpoint name list
(lldb) br n l
```

- ***Delete [breakpoint] name:***

```
[Search Tags: >debreakpointnames >delbreakpointnames >deletebreakpointnames
>breakpointlinames >breakpointlsnames >breakpointdeletenames >debreakptnames
>delbreakptnames >deletebreakptnames >breakptlinames >breakptlsnames
>breakptdeletenames >debrkptnames >delbrkptnames >deletebrkptnames >brkptlinames
>brkptlsnames >brkptdeletenames >debrptnames >delbrptnames >deletebrptnames
>brptlinames >brptlsnames >brptdeletenames >debrnames >delbrnames >deletebrnames
>brlinames >brlsnames >brdeletenames >breakpointunname >breakptunname >brkptunname
>brptunname >brunname >unnamebreakpoints >unnamebreakpts >unnamebrkpts >unnamebrpts
>unnamebrs >removebreakpointnames >breakpointremovenames >removebreakptnames
>breakptremovenames >removebrkptnames >brkptremovenames >removebrptnames
>brptremovenames >removebrnames >brremovenames >breakpointremvenames
>breakpointremvnames >breakpointrmvnames >breakpointrmnames >breakptremvenames
>breakptremvnames >breakptrmvnames >breakptrmnames >brkptremvenames >brkptremvnames
>brkptrmvnames >brkptrmnames >brptremvenames >brptremvnames >brptrmvnames
>brptrmnames >brremvenames >brremvnames >brrmvnames >brrmnames >breakpointnamedelete
>breakpointnadelete >breakpointndelete >breakptnamedelete >breakptnadelete
>breakptndelete >brkptnamedelete >brkptnadelete >brkptndelete >brptnamedelete
>brptnadelete >brptndelete >brnamedelete >brnadelete >brndelete]
```

Synopsis:

```
breakpoint name delete [--name <breakpt-name>] [<breakpt-id>]
```

Example(s):

```
(lldb) breakpoint name delete --name 'funcs' 3 2     # remove
'funcs' name from the breakpoints [of id]: 3, 2
(lldb) br n d -N 'funcs' 3 2
```

*[**Note:** You can only delete a single [breakpoint] name at a time [from the list of names of a breakpoint]. - **end note**]*

- ***Configure [breakpoint] names:***

> [Search Tags: >confbreakpointnames >configbreakpointnames >confbreakpointnames >configbreakpointnames >configurebreakpointnames >breakpointconfnames >breakpointconfignames >breakpointconfigurenames >confbreakptnames >configbreakptnames >confbreakptnames >configbreakptnames >configurebreakptnames >breakptconfnames >breakptconfignames >breakptconfigurenames >confbrkptnames >configbrkptnames >confbrkptnames >configbrkptnames >configurebrkptnames >brkptconfnames >brkptconfignames >brkptconfigurenames >confbrptnames >configbrptnames >confbrptnames >configbrptnames >configurebrptnames >brptconfnames >brptconfignames >brptconfigurenames >confbrnames >configbrnames >confbrnames >configbrnames >configurebrnames >brconfnames >brconfignames >brconfigurenames >breakpointnconfnames >breakpointnameconfnames >breakpointnconfignames >breakpointnameconfignames >breakpointnconfigurenames >breakpointnameconfigurenames >breakptnconfnames >breakptnameconfnames >breakptnconfignames >breakptnameconfignames >breakptnconfigurenames >breakptnameconfigurenames >brkptnconfnames >brkptnameconfnames >brkptnconfignames >brkptnameconfignames >brkptnconfigurenames >brkptnameconfigurenames >brptnconfnames >brptnameconfnames >brptnconfignames >brptnameconfignames >brptnconfigurenames >brptnameconfigurenames >brnconfnames >brnameconfnames >brnconfignames >brnameconfignames >brnconfigurenames >brnameconfigurenames]

> ***Brief:***
>
> *`breakpoint name configure` allows to configure the options of breakpoints who have in their list of names anyone of the `<breakpoint-names>` provided.*
>
> ***Synopsis:***
>
> ```
> breakpoint name configure <cmd-options> [<breakpt-name> ...]
> # i.e, expanded below
> breakpoint name configure <condition> [<command> ...]
> [<attribute> ...] [<breakpt-name> ...]
> ```
>
> ***Example(s):***
>
> ```
> (lldb) breakpoint name configure --condition '2 <= ac && ac <=
> 10' --command bt --command 'fr v' --auto-continue true
> controlFlow
> (lldb) br n c -c '2 <= ac && ac <= 10' -C bt -C 'fr v' -G
> controlFlow
> ```

> > *To clarify — we are asking `lldb` to configure all breakpoints, named `controlFlow`, to break (only) when: `2 <= ac <= 10`, and following a break, to run the following [`lldb`] commands: `bt` and `fr v`, and to then `continue` execution [of the program] automatically.*

> *[**Note:***
>
> - *If you provide a breakpoint id, with the `--breakpoint-id` or `-B` option [followed by `<breakpt-ids>`], the options will be copied from the breakpoint, otherwise only the options specified will be set on the name.*
>
> *- **end note**]*

> *Deeper Look at Breakpoint Configuration Commands:*

- ***Disbale / Enable [named breakpoints] :***

> > `[Search Tags: >breakpointconfigureenable >breakpointconfiguredisable >configurebreakpointenable >configurebreakpointdisable >breakpointconfigenable >breakpointconfigdisable >configbreakpointenable >configbreakpointdisable >breakpointconfenable >breakpointconfdisable >confbreakpointenable >confbreakpointdisable >breakptconfigureenable >breakptconfiguredisable >configurebreakptenable >configurebreakptdisable >breakptconfigenable >breakptconfigdisable >configbreakptenable >configbreakptdisable >breakptconfenable >breakptconfdisable >confbreakptenable >confbreakptdisable >brkptconfigureenable >brkptconfiguredisable >configurebrkptenable >configurebrkptdisable >brkptconfigenable >brkptconfigdisable >configbrkptenable >configbrkptdisable >brkptconfenable >brkptconfdisable >confbrkptenable >confbrkptdisable >brptconfigureenable >brptconfiguredisable >configurebrptenable >configurebrptdisable >brptconfigenable >brptconfigdisable >configbrptenable >configbrptdisable >brptconfenable >brptconfdisable >confbrptenable >confbrptdisable >brconfigureenable >brconfiguredisable >configurebrenable >configurebrdisable >brconfigenable >brconfigdisable >configbrenable >configbrdisable >brconfenable >brconfdisable >confbrenable >confbrdisable]`

> > ***Synopsis:***
>
> ```
> breakpoint name configure [--disable] [--enable] [<breakpt-
> name> ...]
> ```
>
> ***Example(s):***

```
(lldb) breakpoint name configure --disable 'funcs'         #
disable all breakpoints named: 'funcs'
(lldb) br n c -d 'funcs'
```

```
(lldb) breakpoint name configure --enable 'controlFlow'    #
enable all breakpoints named: 'controlFlow'
(lldb) br n c -e 'controlFlow'
```

- ***Configure condition:***

```
[Search Tags: >breakpointconfigureconditions >breakpointconfigureconds
>configurebreakpointconditions >configurebreakpointconds
>breakpointconfigconditions >breakpointconfigconds >configbreakpointconditions
>configbreakpointconds >breakpointconfconditions >breakpointconfconds
>confbreakpointconditions >confbreakpointconds >breakptconfigureconditions
>breakptconfigureconds >configurebreakptconditions >configurebreakptconds
>breakptconfigconditions >breakptconfigconds >configbreakptconditions
>configbreakptconds >breakptconfconditions >breakptconfconds
>confbreakptconditions >confbreakptconds >brkptconfigureconditions
>brkptconfigureconds >configurebrkptconditions >configurebrkptconds
>brkptconfigconditions >brkptconfigconds >configbrkptconditions
>configbrkptconds >brkptconfconditions >brkptconfconds >confbrkptconditions
>confbrkptconds >brptconfigureconditions >brptconfigureconds
>configurebrptconditions >configurebrptconds >brptconfigconditions
>brptconfigconds >configbrptconditions >configbrptconds >brptconfconditions
>brptconfconds >confbrptconditions >confbrptconds >brconfigureconditions
>brconfigureconds >configurebrconditions >configurebrconds >brconfigconditions
>brconfigconds >configbrconditions >configbrconds >brconfconditions
>brconfconds >confbrconditions >confbrconds]
```

Synopsis:

```
breakpoint name configure --condition <condition-epxr>
[<breakpt-name> ...]
```

Example(s):

```
(lldb) breakpoint name configure --condition 'argc > 2'
(lldb) br n c -c 'argc > 2'
```

- ***Configure command(s):***

> [Search Tags: >breakpointconfigurecommands >breakpointconfigurecmds >configurebreakpointcommands >configurebreakpointcmds >breakpointconfigcommands >breakpointconfigcmds >configbreakpointcommands >configbreakpointcmds >breakpointconfcommands >breakpointconfcmds >confbreakpointcommands >confbreakpointcmds >breakptconfigurecommands >breakptconfigurecmds >configurebreakptcommands >configurebreakptcmds >breakptconfigcommands >breakptconfigcmds >configbreakptcommands >configbreakptcmds >breakptconfcommands >breakptconfcmds >confbreakptcommands >confbreakptcmds >brkptconfigurecommands >brkptconfigurecmds >configurebrkptcommands >configurebrkptcmds >brkptconfigcommands >brkptconfigcmds >configbrkptcommands >configbrkptcmds >brkptconfcommands >brkptconfcmds >confbrkptcommands >confbrkptcmds >brptconfigurecommands >brptconfigurecmds >configurebrptcommands >configurebrptcmds >brptconfigcommands >brptconfigcmds >configbrptcommands >configbrptcmds >brptconfcommands >brptconfcmds >confbrptcommands >confbrptcmds >brconfigurecommands >brconfigurecmds >configurebrcommands >configurebrcmds >brconfigcommands >brconfigcmds >configbrcommands >configbrcmds >brconfcommands >brconfcmds >confbrcommands >confbrcmds]

> ***Synopsis:***
>
> ```
> breakpoint name configure --command <command> [<breakpt-
> name> ...]
> ```
>
> ***Example(s):***
>
> ```
> (lldb) breakpoint name configure --command 'bt'
> (lldb) br n c
> ```
>
> ```
> (lldb) breakpoint name configure --command 'bt' --command
> 'frame view'
> (lldb) br n c
> ```

- ***Configure attribute(s):***

> [Search Tags: >breakpointconfigureattributes >breakpointconfigureattribs >configurebreakpointattributes >configurebreakpointattribs >breakpointconfigattributes >breakpointconfigattribs

```
>configbreakpointattributes >configbreakpointattribs >breakpointconfattributes
>breakpointconfattribs >confbreakpointattributes >confbreakpointattribs
>breakptconfigureattributes >breakptconfigureattribs
>configurebreakptattributes >configurebreakptattribs >breakptconfigattributes
>breakptconfigattribs >configbreakptattributes >configbreakptattribs
>breakptconfattributes >breakptconfattribs >confbreakptattributes
>confbreakptattribs >brkptconfigureattributes >brkptconfigureattribs
>configurebrkptattributes >configurebrkptattribs >brkptconfigattributes
>brkptconfigattribs >configbrkptattributes >configbrkptattribs
>brkptconfattributes >brkptconfattribs >confbrkptattributes >confbrkptattribs
>brptconfigureattributes >brptconfigureattribs >configurebrptattributes
>configurebrptattribs >brptconfigattributes >brptconfigattribs
>configbrptattributes >configbrptattribs >brptconfattributes >brptconfattribs
>confbrptattributes >confbrptattribs >brconfigureattributes >brconfigureattribs
>configurebrattributes >configurebrattribs >brconfigattributes >brconfigattribs
>configbrattributes >configbrattribs >brconfattributes >brconfattribs
>confbrattributes >confbrattribs]
```

Synopsis:

```
breakpoint name configure [-i <boolean>] [-G <boolean>] [-o
<boolean>] [<breakpt-name> ...]
```

Command Options:

Attribute	Abrv.	Description
`--ignore-count <count>`	`-i`	Set the number of times this breakpoint is skipped before stopping; this is what is referred to as the *hit-count* option.
`--auto-continue <boolean>`	`-G`	The breakpoint will auto-continue after running its commands.
`--one-shot <boolean>`	`-o`	The breakpoint is deleted the first time it stops causes a stop.

Example(s):

```
(lldb) breakpoint name configure --one-shot true 'funcs'
(lldb) br n c -o true 'funcs'
```

- ***Configure [assigned] thread(s):***

> [Search Tags: >breakpointconfigurethrds >breakpointconfigurethreads >configurebreakpointthrds >configurebreakpointthreads >breakpointconfigthrds >breakpointconfigthreads >configbreakpointthrds >configbreakpointthreads >breakpointconfthrds >breakpointconfthreads >confbreakpointthrds >confbreakpointthreads >breakptconfigurethrds >breakptconfigurethreads >configurebreakptthrds >configurebreakptthreads >breakptconfigthrds >breakptconfigthreads >configbreakptthrds >configbreakptthreads >breakptconfthrds >breakptconfthreads >confbreakptthrds >confbreakptthreads >brkptconfigurethrds >brkptconfigurethreads >configurebrkptthrds >configurebrkptthreads >brkptconfigthrds >brkptconfigthreads >configbrkptthrds >configbrkptthreads >brkptconfthrds >brkptconfthreads >confbrkptthrds >confbrkptthreads >brptconfigurethrds >brptconfigurethreads >configurebrptthrds >configurebrptthreads >brptconfigthrds >brptconfigthreads >configbrptthrds >configbrptthreads >brptconfthrds >brptconfthreads >confbrptthrds >confbrptthreads >brconfigurethrds >brconfigurethreads >configurebrthrds >configurebrthreads >brconfigthrds >brconfigthreads >configbrthrds >configbrthreads >brconfthrds >brconfthreads >confbrthrds >confbrthreads]

> ***Synopsis:***

```
(lldb) breakpoint name configure [ --thread-index <index> ]
[<breakpt-name> ...]
                                 [ --thread-name <name> ]
[<breakpt-name> ...]
                                 [ --thread-id <tid> ]
[<breakpt-name> ...]
```

> ***Example(s):***

```
(lldb) breakpoint name configure --thread-index 3 'funcs'
(lldb) br n c -x 3 'funcs'
```

```
(lldb) breakpoint name configure --thread-name
'centralServer' 'controlFlow'
(lldb) br n c -T 'centralServer' 'controlFlow'
```

```
(lldb) breakpoint name configure --thread-id 483413
'returns'
(lldb) br n c -t 483413 'returns'
```

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Documentation | LLDB | (Official) Tutorial :: Breakpoint Names |
> | 2 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint name` |
> | 3 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint name add` |
> | 3 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint name list` |
> | 3 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint name configure` |
> | 3 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint name delete` |

🏠 | ⬅️ | ➡️

3.4.1.5. Multi-Threaded Programs

`[Search Tags: >breakpointthreads >threadbreakpoint >multithreadedbreakpoint >multithreadbreakpoint >breakptthreads >threadbreakpt >multithreadedbreakpt >multithreadbreakpt >brkptthreads >threadbrkpt >multithreadedbrkpt >multithreadbrkpt >brptthreads >threadbrpt >multithreadedbrpt >multithreadbrpt >brthreads >threadbr >multithreadedbr >multithreadbr]`

- ***Set breakpoint, on** [a specific] **thread:***

> ***Synopsis:***
>
> ```
> (lldb) breakpoint set <breakpt-definition> [ --thread-index
> <index> ]     # by index (in the process)
> (lldb) breakpoint set <breakpt-definition> [ --thread-name <name>
> ]       # by name
> (lldb) breakpoint set <breakpt-definition> [ --thread-id <tid> ]
> # by tid (in the computer)
> ```
>
> ***Example(s):***
>
> ```
> (lldb) breakpoint set --name baz --thread-index 3
> (lldb) br s -n baz -x 3
> ```
>
> ```
> (lldb) breakpoint set --name foo --thread-name 'centralServer'
> (lldb) br s -n foo -T 'centralServer'
> ```

```
(lldb) breakpoint set --name bar --thread-id 483413
(lldb) br s -n bar -t 483413
```

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help breakpoint set`

🏠 | ⬅️ | ➡️

3.4.1.6. C++ Programs

`[Search Tags: >breakpointcpp >cppbreakpoint >breakptcpp >cppbreakpt >brkptcpp >cppbrkpt >brptcpp >cppbrpt >brcpp >cppbr >breakpointc++ >c++breakpoint >breakptc++ >c++breakpt >brkptc++ >c++brkpt >brptc++ >c++brpt >brc++ >c++br]`

Contents

- **Set a breakpoint, on function(s), using fullname:**

`[Search Tags: >breakpointfullname >breakptfullname >brkptfullname >brptfullname >brfullname >fullnamebreakpoint >fullnamebreakpt >fullnamebrkpt >fullnamebrpt >fullnamebr >breakpointfullnamefunc >breakpointfullnamefunction >breakpointfuncfullname >breakpointfunctionfullname >breakptfullnamefunc >breakptfullnamefunction >breakptfuncfullname >breakptfunctionfullname >brkptfullnamefunc >brkptfullnamefunction >brkptfuncfullname >brkptfunctionfullname >brptfullnamefunc >brptfullnamefunction >brptfuncfullname >brptfunctionfullname >brfullnamefunc >brfullnamefunction >brfuncfullname >brfunctionfullname]`

Synopsis:

```
(lldb) breakpoint set --fullname <full-function-name>
```

Example(s):

```
(lldb) breakpoint set --fullname 'Temperature::getter'
(lldb) br s -F 'Temperature::getter'
```

- **Set a breakpoint, on function(s), using basename:**

> [Search Tags: >breakpointbasename >breakptbasename >brkptbasename >brptbasename >brbasename >basenamebreakpoint >basenamebreakpt >basenamebrkpt >basenamebrpt >basenamebr]

> ***Synopsis:***
>
> ```
> (lldb) breakpoint set --basename <base-function-name>
> ```
>
> ***Example(s):***
>
> ```
> (lldb) breakpoint set --basename 'getter'
> (lldb) br s -b 'getter'
> ```

> *[**Note**: Both, **namespace functions** and **class methods** with the given basename will have a breakpoint set on them. - **end note**]*

- **Set a breakpoint, on [class] method(s):**

> [Search Tags: >breakpointmethods >breakptmethods >brkptmethods >brptmethods >brmethods >methodsbreakpoint >methodsbreakpt >methodsbrkpt >methodsbrpt >methodsbr >breakpointclassmethods >breakptclassmethods >brkptclassmethods >brptclassmethods >brclassmethods >classmethodsbreakpoint >classmethodsbreakpt >classmethodsbrkpt >classmethodsbrpt >classmethodsbr]

> ***Synopsis:***
>
> ```
> (lldb) breakpoint set --method <method>
> ```
>
> ***Example(s):***
>
> ```
> (lldb) breakpoint set --method 'getter'
> (lldb) br s -M 'getter'
> ```

- **Set a breakpoint, on (all) exceptions [on `catch` and/or `throws`]:**

> `[Search Tags: >breakpointexceptions >breakptexceptions >brkptexceptions >brptexceptions >brexceptions >exceptionsbreakpoint >exceptionsbreakpt >exceptionsbrkpt >exceptionsbrpt >exceptionsbr]`

> ***Synopsis:***
>
> ```
> breakpoint set --language-exception <source-code-language> [--on-catch <true | false>] [--on-throw <true | false>]
> ```
>
> ***Example(s):***
>
> ```
> breakpoint set --language-exception c++
> # on default setting
> br s -E c++
> ```
>
> ```
> breakpoint set --language-exception c++ --on-catch true --on-throw true        # on throws
> br s -E c++ -h true -w true
> ```
>
> ```
> breakpoint set --language-exception c++ --on-catch True --on-throw False       # on catches
> br s -E c++ -h True -w False
> ```

> *[**Note**: To set a breakpoint on specific exception objects, there exists the `--exception-typename` (`-O`) option, but it is unfortunately only supported for **Swift**, at the moment (22/01/2020). - **end note**]*

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Manual Page | Unix / Linux / MacOS | `(lldb) help breakpoint set` |

🏠 | ⬅️ | ➡️

3.4.2. Watchpoints

[Search Tags: >lldb.watchpoints >debugger.watchpoints >lldbwatchpoints >debuggerwatchpoints >secwatchpoint >sectwatchpoint >sectionwatchpoint >watchpointsection >secwatchpt >sectwatchpt >sectionwatchpt >watchptsection >secwapt >sectwapt >sectionwapt >waptsection >secwa >sectwa >sectionwa >wasection]

Contents

> *(**Definition**) Watchpoint: a (special) kind of breakpoint (debugging mechanism) whereby execution is suspended every time a specified variable or memory-location is accessed for reading and/or writing.*
>
> — *Wikitionary :: Watchpoint*

> *[**Note**: Before the execution of a [targeted] program, watchpoints can only be set on global variables — once [the [targeted] program is] launched, watchpoints can be set on any variable/memory-location. - **end note**]*

The following subsections dive into the ***basic*** (§3.4.2.1) as well as [slightly] more ***advanced commands*** (§3.4.2.2) *[for operating on watchpoints]*, then into the available ***watchpoint options*** (§3.4.2.3) and how to utilise them.

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Manual Page | Unix / Linux / MacOS | `(lldb) help watchpoint [<command>]` |
> | 2 | Documentation | LLDB | (Official) Tutorial :: Setting Watchpoints |
> | 3 | Documentation | LLDB | GDB to LLDB Command Map |

🏠 | ⬅️ | ➡️

3.4.2.1. Basic Commands

[Search Tags: >basicwatchpoint >basicswatchpoint >watchpointbasics >basicwatchpt >basicswatchpt >watchptbasics >basicwapt >basicswapt >waptbasics >basicwapt >basicswapt >waptbasics >basicwa >basicswa >wabasics]

Commands to:

> *(Basic) Commands for operating on watchpoints.*

- **Set a watchpoint:**

> [Search Tags: >createwatchpoint >crwatchpoint >sewatchpoint >swatchpoint >watchpointcreates >setwatchpoint >watchpointsets >createwatchpt >crwatchpt >sewatchpt >swatchpt >watchptcreates >setwatchpt >watchptsets >createwapt >crwapt >sewapt >swapt >waptcreates >setwapt >waptsets >createwa >crwa >sewa >swa >wacreates >setwa >wasets]

> ***Synopsis:***
>
> > *on a variable:*
> >
> > ```
> > watchpoint set variable [--watch <watch-type>] [--size
> > <byte-size>] <variable-name>
> > ```
> >
> > *on an address [by supplying an expression]:*
> >
> > ```
> > watchpoint set expression [--watch <watch-type>] [--size
> > <byte-size>] -- <expr>
> > watchpoint set expression <expr>
> > ```
>
> ***Option(s):***
>
> ```
> -s <byte-size> ( --size <byte-size> )
>               Number of bytes to use to watch a region.
>               Values: 1 | 2 | 4 | 8
> ```
>
> ```
> -w <watch-type> ( --watch <watch-type> )
>               Specify the type of watching to perform.
>               Values: read | write | read_write
> ```
>
> ***Example(s):***
>
> ```
> (lldb) watchpoint set variable --watch read_write my_var
> (lldb) wa s v -w read_write my_var
> ```

```
(lldb) watchpoint set variable   --watch write --size 8 --
my_PtrToLongInt
(lldb) wa s v -w write -s 8 -- my_PtrToLongInt
```

```
(lldb) watchpoint set expression --watch write --size 8 --
my_PtrToLongInt
(lldb) wa s e -w write -s 8 -- my_PtrToLongInt
```

```
(lldb) watchpoint set expression --watch read --size 4 --
0x00007ffeefbff510
(lldb) wa s e -w read -s 4 -- 0x00007ffeefbff510
```

*[**Note**:*

We say:

- *`breakpoint set variable` – sets watctpoints to watch for `<watch-type>` accesses on the `<size>`-byte **variable**, `<variable-name>`.*
- *`breakpoint set expression` – sets watctpoints to watch for `<watch-type>` accesses on the `<size>`-byte **region, pointed to by the address** `<expr>`.*

*- **end note**]*

- **List watchpoints:**

```
[Search Tags: >watchpointlist >watchpointls >listwatchpoint >lstwatchpoint
>lswatchpoint >liwatchpoint >watchptlist >watchptls >listwatchpt >lstwatchpt
>lswatchpt >liwatchpt >waptlist >waptls >listwapt >lstwapt >lswapt >liwapt >walist
>wals >listwa >lstwa >lswa >liwa]
```

Synopsis:

```
watchpoint list -[bfv] [<watchpt-ids>]
```

Example(s):

```
(lldb) watchpoint list -b 4 5 2       # --brief    (minimum
description)
(lldb) wa l -f 8                      # --full     (full
description, default)
```

```
(lldb) w l -v                              # --verbose  (extensive
description)
```

- **Delete watchpoint(s):**

> [Search Tags: >wadelete >deletewa >deletewapt >deletewatchpt >deletewatchpoint >wadelete >waptdelete >watchptdelete >watchpointdelete >dewa >deletewa >deletetarge >waunload >unloadwatchpoints >uldwatchpoints >watchpointunload >unloadwatchpoint >delwa >wadel >delwapt >waptdel >delwatchpt >watchptdel >delwatchpoint >watchpointdel]

> ***Synopsis:***
>
> ```
> watchpoint delete [<watchpt-ids>]
> ```
>
> ***Example(s):***
>
> ```
> (lldb) watchpoint delete 1 2 3
> (lldb) wa de 5
> ```

> *[**Note**: If no watchpoints are specified, delete them all. - **end note**]*

> ***Further Reading:***
>
#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint set`
2	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint list`
3	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint delete`

🏠 | ⬅️ | ➡️

3.4.2.2. Advanced Commands

[Search Tags: >advancedwatchpoint >watchpointadvanceds >advancedwatchpt >watchptadvanceds >advancedwapt >waptadvanceds >advancedwa >waadvanceds >advwatchpoint >watchpointadvs >advwatchpt >breakptadvs >advwapt >waptadvs >advwa >waadvs]

Commands to:

> *[A lil' more] Advanced commands for operating on watchpoints.*

- Enable / Disable watchpoints:

> `[Search Tags: >watchpointenable >watchptenable >waptenable >waenable >watchpointdisable >watchptdisable >waptdisable >wadisable >enablewatchpoint >enablewatchpt >enablewapt >enablewa >disablewatchpoint >disablewatchpt >disablewapt >disablewa]`

> ***Synopsis:***

```
(lldb) watchpoint disable <watchpt-ids | watchpt-names>
(lldb) watchpoint enable  <watchpt-ids | watchpt-names>
```

```
(lldb) watchpoint modify [--disable] [--enable] <watchpt-ids |
watchpt-names>
```

> ***Example(s):***

```
(lldb) watchpoint disable 1
(lldb) br di 1
```

```
(lldb) watchpoint disable 3.*      # disable all watchpoints of
ID 3.
(lldb) br di 3.*
```

```
(lldb) watchpoint enable 2 6 3.2    # enable watchpoints: 2, 6
and 3.2
(lldb) br en 2 6 3.2
```

> ***[Note:***
>
> - *To enable only certain locations of a logical watchpoint, use the watchpoint disable command, passing the watchpoint ID followed by a dot-separated wildcard character (*), e.g. `1.*` or `3.*`.*
> - *It is also possible to set, initialy disabled, watchpoints:*

```
(lldb) watchpoint set <watchpt-definition> [--disable]
```

*- **end note**]*

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint enable`
2	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint disable`

🏠 | ⬅️ | ➡️

3.4.2.3 Options

`[Search Tags: >optswatchpoint >watchpointopts >watchpointops >opswatchpoint >optionswatchpoint >watchpointoptions >optswatchpt >watchptopts >watchptops >opswatchpt >optionswatchpt >watchptoptions >optswapt >waptopts >waptops >opswapt >optionswapt >waptoptions >optswa >waopts >waops >opswa >optionswa >waoptions]`

Commands to:

- 1 Add/Modify Watchpoint Conditions
- 2 Add/Modify Watchpoint Commands
- 3 Add/Modify Watchpoint Attributes

*[**Note**: We'll refer to options that are neither [watchpoint] conditions nor [watchpoint] commands as: "[watchpoint] attributes" - **end note**]*

- **Add/modify [watchpoint] condition:**

Synopsis:

```
watchpoint modify --condition <condition-expr> [<watch-id> ...]
```

Example(s):

```
watchpoint modify --condition 'my_var > 10' 1
wa mo -c 'my_var > 10' 1
```

```
watchpoint modify --condition 'my_ptr == NULL' 2
wa mo -c 'my_ptr == NULL' 2
```

```
watchpoint modify --condition '' 1        # clear watchpoint
condition
wa mo -c '' 1
```

- **Add/modify [watchpoint] command(s):**

> `watchpoint command` is identical to `breakpoint command` — it goes without saying, simply replace `breakpoint` with `watchpoint`.

- **Add/modify [watchpoint] attribute(s):**

> *Watchpoints can only add the `ignore-count` attribute; i.e the number of times this watchpoint is skipped before stopping.*

> ***Synopsis:***

```
watchpoint ignore --ignore-count <count> [<watchpt-id> ...]
```

> ***Example(s):***

```
watchpoint ignore --ignore-count 10 3 7 6
wa i -i 10 3 7 6
```

> *[**Note:** If no watchpoints are specified, set them all. - **end note**]*

> ***Further Reading:***

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint modify`
2	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint command`

#	Type	Author	Link
3	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint command add`
4	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint command list`
5	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint command delete`
1	Manual Page	Unix / Linux / MacOS	`(lldb) help watchpoint ignore`

🏠 | ⬅️ | ➡️

3.5. Start Debugging

`[Search Tags: >lldb.startdebug >lldb.debug >debugger.startdebug >debugger.debug >lldbstartdebug >lldbdebug >debuggerstartdebug >debuggerdebug >startdebugging >strtdebugging >startlldbdebuggings >startprogram >startprocess >startprogdebugging >startprcsdebugging]`

- There are two ways to start debugging a process (running program):
  - `launch` – Launching one (§3.5.1)
  - `attach` – Attaching to ***[an already running]*** one (§3.5.2)
- Also, with regards to programs that are `launched gives the possibility of configuring things like: where you want the process to be run *(terminal, shell)*, setting environment variables, setting the current working directory, redirecting` stdin/out/err`, etc – all of which is dicussed in **Advanced Program Configurations** (§3.5.3).

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help process [<command>]`

🏠 | ⬅️ | ➡️

3.5.1. Launch

`[Search Tags: >prlasection >prsection >processsection >prcssection lasection >lnchsection >lchsection >launchsection >launchprocess >launchprcs >lprocess >lnchprcs >launchprograms >launchprogs >lprograms >lnchprgs >laprogs >laprs >laprs >programlaunch >progrmlaunch >proglaunch >programlnch >progrmlnch >proglnch >programla >progrmla >progla >processlaunch >prcslaunch >processlnch >prcslnch >processla >prcsla >prlaunch >prlnch >prla]`

Contents

- **Launch *[a program]*:**

> ***Synopsis:***
>
> ```
> process launch -- [<arg> ...]
> run [<arg> ...]                                    # alias
> r [<arg> ...]                                      # alias
> ```
>
> ***Example(s):***
>
> ```
> (lldb) process launch                              # without
> arguments
> (lldb) pr la -- "arg1" "arg2" "youGetThePoint"     # with
> arguments
> (lldb) run "arg1" "arg2" "youGetThePoint"          # with
> arguments
> (lldb) r                                           # without
> arguments
> ```

> ***[Note:***
>
> - *run* is an alias for '*process launch --shell-expand-args true --*', see *h run*.
> - *r* is an alias for *'run'* , see *h r*.
>
> *- **end note]***

- **Launch configurations:**

> [Search Tags: >configlaunchprocess >launchprocessconfigurations >configlaunchprcs >launchprcsconfigurations >configlprocess >lprocessconfigurations >configlnchprcs >lnchprcsconfigurations >configlaunchprograms >launchprogramsconfigurations >configlaunchprogs >launchprogsconfigurations >configlprograms >lprogramsconfigurations >configlnchprgs >lnchprgsconfigurations >configlaprogs >laprogsconfigurations >configlaprs >laprsconfigurations >configlaprs >laprsconfigurations >configprogramlaunch >programlaunchconfigurations >configprogrmlaunch >progrmlaunchconfigurations >configproglaunch >proglaunchconfigurations >configprogramlnch >programlnchconfigurations

> >configprogrmlnch >progrmlnchconfigurations >configproglnch >proglnchconfigurations >configprogramla >programlaconfigurations >configprogrmla >progrmlaconfigurations >configprogla >proglaconfigurations >configprocesslaunch >processlaunchconfigurations >configprcslaunch >prcslaunchconfigurations >configprocesslnch >processlnchconfigurations >configprcslnch >prcslnchconfigurations >configprocessla >processlaconfigurations >configprcsla >prcslaconfigurations >configprlaunch >prlaunchconfigurations >configprlnch >prlnchconfigurations >configprla >prlaconfig >launchconfigurations >launchconfigs >lnchconfigurations >lnchconfigs >laconfigurations >laconfigs]

- **Run *[program]* in *[different]* shell or terminal:**

> [Search Tags: >termlaunchprocesses >termlaunchprograms >terminallaunchprocesses >terminallaunchprograms >shelllaunchprocesses >shelllaunchprograms >launchshell >shelllaunch >launchshll >shlllaunch >launchterminal >terminallaunch >lnchshell >shelllnch >lnchshll >shlllnch >lnchterminal >terminallnch >lashell >shellla >lashll >shllla >laterminal >terminalla >launchconfigshell >shellconfiglaunch >launchconfigshll >shllconfiglaunch >launchconfigterminal >terminalconfiglaunch >lnchconfigshell >shellconfiglnch >lnchconfigshll >shllconfiglnch >lnchconfigterminal >terminalconfiglnch >laconfigshell >shellconfigla >laconfigshll >shllconfigla >laconfigterminal >terminalconfigla >termlaunch >termlnch >termla >termconfiglaunch >termconfiglnch >termconfigla]

> ***Synopsis:***
>
> ```
> process launch --shell=[<filename>] -- [<arg> ...]   # shell
> process launch --tty -- [<arg> ...]                  #
> terminal
> ```
>
> ***Example(s):***
>
> ```
> process launch --shell=/dev/ttys002 -- "arg1" "arg2"
> pr la -c=/dev/ttys003
> ```
>
> ```
> (lldb) process launch --tty -- "arg1" "arg2"
> (lldb) pr la -t
> ```

> *[**Note**: Not supported on all platforms. - **end note**]*

- **Set environment variables:**

> [Search Tags: >envlaunchprocesses >envlaunchprograms >environmentlaunchprocesses >environmentlaunchprograms >launchenvironment >environmentlaunch >lnchenvironment >environmentlnch >laenvironment >environmentla >launchconfigenvironment >environmentconfiglaunch >lnchconfigenvironment >environmentconfiglnch >laconfigenvironment >environmentconfigla >envlaunch >envlnch >envla >envconfiglaunch >envconfiglnch >envconfigla >configenvironment >configureenvironment]

> ***Synopsis:***
>
> ```
> process launch --environment <NAME>=<VALUE> -- [<arg> ...]
> ```
>
> ***Example(s):***
>
> ```
> process launch --environment BIG_ENDIAN=true -- "arg1"
> pr la -v BIG_ENDIAN=true -- "arg1"
> ```
>
> ```
> process launch --environment GREET="hello" --environment
> FAREWELL="bye" -- "arg1" "arg2"
> pr la -v GREET="hello" -v FAREWELL="bye" -- "arg1" "arg2"
> ```

- **Set current working directory *(cwd)*:**

> [Search Tags: >cwdlaunchprocesses >cwdlaunchprograms >cwdlaunch >cwdlnch >cwdla >cwdlaunchprocesses >cwdlaunchprograms >cwdconfiglaunch >cwdconfiglnch >cwdconfigla >launchcwd >lnchcwd >lacwd >launchconfigcwd >lnchconfigcwd >laconfigcwd >configcwd >configurecwd]

> ***Synopsis:***
>
> ```
> process launch --working-dir <directory> -- [<arg> ...]
> ```
>
> ***Example(s):***
>
> ```
> process launch --working-dir /Volumes/Driver/Core/ -- "arg1"
> "arg2"
> pr la -w /Volumes/Driver/Core/ -- "arg1" "arg2"
> ```

- **Redirect *[program]* standard in/out/err streams:**

> [Search Tags: >redirstreams >redirstdstreams >redirectprstdstreams >redirectprogstreams >redirectprogstdstreams >redirprcsstreams >redirprcsstdstreams >redirprogramstdstreams >redirprogramstreams >redirectprogramstdstreams >redirectprogramstreams redirectprocessstreams >redirectprocessstdstreams redirstreams >redirectstreams >redirlaunchprocesses >redirectlaunchprocesses >redirlaunchprograms >redirectlaunchprograms >redirlaunch >redirectlaunch >redirlnch >redirectlnch >redirla >redirectla >redirlaunchprocesses >redirectlaunchprocesses >redirlaunchprograms >redirectlaunchprograms >redirconfiglaunch >redirectconfiglaunch >redirconfiglnch >redirectconfiglnch >redirconfigla >redirectconfigla >launchredirectstreams >launchredirectstdstreams >lnchredirectstreams >lnchredirectstdstreams >laredirectstreams >laredirectstdstreams >launchconfigredirectstreams >launchconfigredirectstdstreams >lnchconfigredirectstreams >lnchconfigredirectstdstreams >laconfigredirectstreams >laconfigredirectstdstreams >configredirectstreams >configredirectstdstreams >configureredirectstreams >configureredirectstdstreams >processlaunchredirectstdstreams >processlaunchredirectstreams >prcslaunchredirectstdstreams >prcslaredirectstdstreams >prlaredirectstdstreams >prlaredirstdstreams >prlaredirstdstdstreams >prcslnchredirectstdstreams >prcslaunchredirectstreams >prcslaredirectstreams >prlaredirectstreams >prlaredirstreams >prlaredirstdstreams >prcslnchredirectstreams]

> ***Synopsis:***
>
> ```
> process launch [--stdin <filename>] [--stdout <filename>] [--stderr <filename>] -- [<arg> ...]
> ```
>
> ***Example(s):***
>
> ```
> (lldb) process launch --stdin   file-1.txt  # text file, w/ name: 'file-1.txt'
> (lldb) process launch --stdout  file-2.txt
> (lldb) process launch --stderr  file-3.txt
> ```
>
> ```
> (lldb) pr la -i  /dev/ttys001               # terminal shell [device] file, w/ name: '/dev/ttys001'
> ```

```
(lldb) pr la -o  /dev/ttys002
(lldb) pr la -e  /dev/ttys003
```

```
(lldb) process launch -i /dev/ttys001 -o outFile.log -e
errFile.log -- "arg1" "arg2" "youGetThePoint"
(lldb) pr la -i /dev/ttys001 -o outFile.log -e errFile.log -
- "arg1" "arg2" "youGetThePoint"
```

*[**Note**:*

- *To clarify [, this [last] example] —*
    - *we redirect [the] standard input [stream] (`stdin`) [of the program] to (i.e to be or come or be-given from or originate from) a [terminal [device]] file, by the name of: `/dev/ttys001` – this is a running [terminal] shell [instance],*
    - *we redirect the standard output (`stdout`) to [be written/printed to] a `.log` file, by the name of: "`outFile`",*
    - *we do the same [, as `stdout`,] with the standard error (`stderr`), this time, the file goes by the name: "`errFile`",*
    - *we delimit the `lldb` command options that we have given, from [, that which we will give as,] program arguments, with: "`--`" [, `lldb`'s parser delimiter],*
    - *we pass [to our program] three arguments.*

*- **end note**]*

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help process launch`
2	Documentation	LLDB	(Official) Tutorial :: Starting or Attaching to Your Program
3	Encyclopedia	Wikipedia	Device files
4	Q&A Forum	StackOverflow	How to get the current terminal name ?
5	Article	`opensource.com`	Managing devices in Linux
6	Q&A Forum	StackOverflow	What is the difference between shell, console and terminal ?

3.5.2. Attach

[Search Tags: >atsection >atchsection >achsection >attchsection >attachsection >attchprocess >attachprocess >attchprcs >attachprcs >aprocess >atchprcs >attchprograms >attachprograms >attchprogs >attachprogs >aprograms >atchprgs >atprogs >attchprs >attachprs >programattch >programattach >progrmattch >progrmattach >progattch >progattach >programatch >progrmatch >progatch >programat >progrmat >progat >processattch >processattach >prcsattch >prcsattach >processatch >prcsatch >processat >prcsat >prattch >prattach >pratch >prat]

> lldbattachprocess >lldbattachprogram >lldbattachtoprocess >lldbattachtoprogram >lldb.attachprocess >lldb.attachprogram >lldb.attachtoprocess >lldb.attachtoprogram >debuggerattachprocess >debuggerattachprogram >debuggerattachtoprocess >debuggerattachtoprogram >debugger.attachprocess >debugger.attachprogram >debugger.attachtoprocess >debugger.attachtoprogram

Contents

> ***TL;DR:***
> *"Attach" means, take over control of a process (running program), at the instruction that the process has reached.*

> ***FULL:***
> *Every time you start a new application, you create one or more processes. A process is simply executable code that is loaded into memory. The CPU reads and executes the instructions to perform the tasks you ask the application to do. When the CPU loads your application into memory, it assigns each process the application creates a Process IDentifier (PID), which is pronounced pid (think of lid with a p instead of an l). The PID is simply a number associated with the process for easy identification.*

> *In most cases, you debug an application by running it in the IDE in debug mode. However, there are some situations where you must debug the application in a different way — by attaching to its process.* ***Attaching to the process means telling the CPU to send the instructions in the executable code to a debugger before they're executed by the CPU.*** *In other words, you place the debugger between the executable code and the CPU.*
>
> *— Attaching to a Running Process Using CodeBlocks*

> *Command for attaching to processes:*

- **Attach to process:**

  > ***Synopsis:***

```
process attach [--pid <pid>]                              # by
process identifier (pid)
process attach [--wait-for] [--name <program-name>]    # by
program name
```

Example(s):

```
(lldb) process attach --pid 2432                    # attach to
currently running process by identifier (pid)
(lldb) pr a -p 2432
```

```
(lldb) process attach --name a.out                  # attach to
currently running process by program name
(lldb) pr a -n a.out
```

```
(lldb) process attach --waitfor --name a.out        # wait for,
and, attach to, the next process, whose program name is: 'a.out'
(lldb) pr a -w -n a.out
```

- **Attach to remote process:**

Brief:

Connect to a process via remote GDB server. If no host is specifed, localhost is assumed.

Synopsis:

```
gdb-remote <host>:<port>
```

Example(s):

```
(lldb) gdb-remote 8000
```

> *To clarify — we attach to a remote GDB protocol server running on the **local system** (i.e `localhost`), port `8000`.*

```
(lldb) gdb-remote eorgadd:8000
```

> *To clarify — Attach to a remote GDB protocol server running on the system `eorgadd`, port `8000`.*

```
(lldb) gdb-remote 216.3.128.12:8000
```

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help process attach`
2	Manual Page	Unix / Linux / MacOS	`(lldb) help gdb-remote`
3	Documentation	Apple	GDB and LLDB Command Examples
4	Encyclopedia	Wikipedia	`gdbserver`
5	Documentation	`sourceware.org/gdb`	Using the `gdbserver` Program
6	Tutorial	TheGeekStuff	How to Debug Programs on Remote Server using GDBServer Example

---

🏠 | ⬅️ | ➡️

3.6. Graphical User Interface (GUI)

[Search Tags: >lldbgui >lldbgraphicalui >lldbgraphicaluserinterface >lldb.gui >lldb.graphicalui >lldb.graphicaluserinterface >debuggergui >debuggergraphicalui >debuggergraphicaluserinterface >debugger.gui >debugger.graphicalui >debugger.graphicaluserinterface >graphicaluserinterface >graphicalui >userinterface >interface >gmode >graphicalmode >gumode >guimodes >modegui >modegraphicaluserinterface >graphicaluserinterfacemodes]

Contents

About

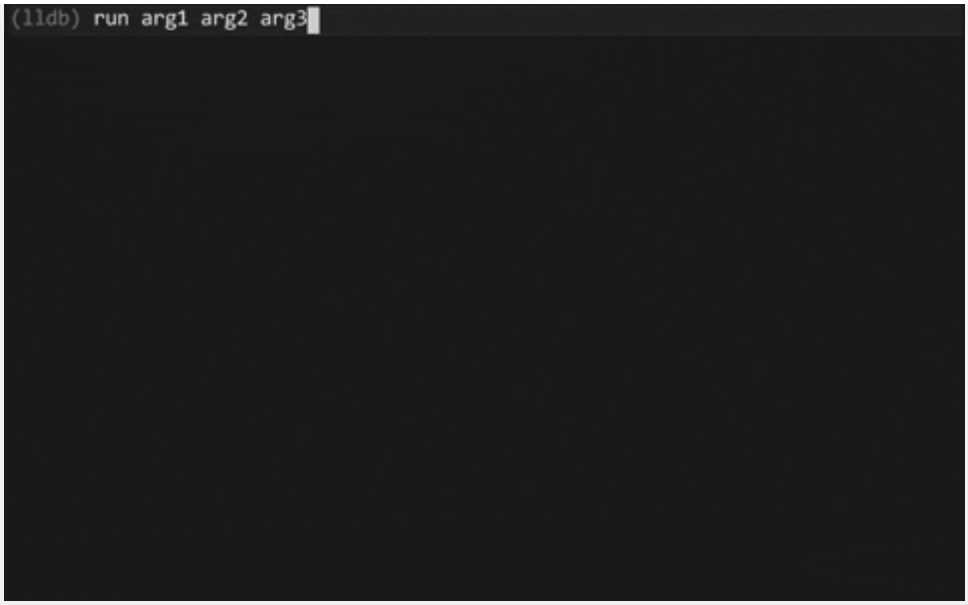
- The graphical user interface *[mode of `lldb`]*, or `gui` for short, is what it says it is — namely, a user interface for `lldb` that is **graphical** *[, rather than textual (**command prompt**)]*.

- Its advantages are **MASSIVE**. Here are a few:

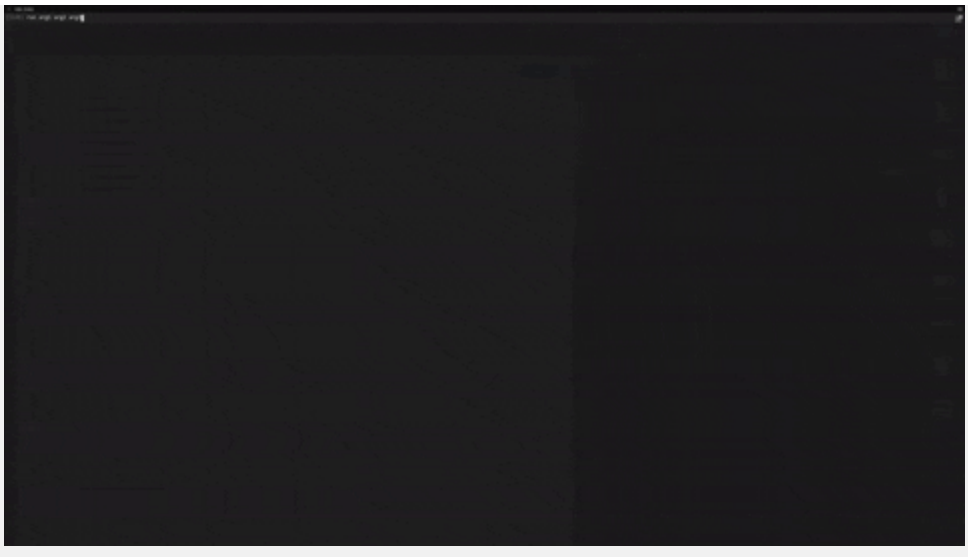
    - You no longer have to constantly auto/man-ually print the source code, value of variables/registers and backtrace, after every instruction/line executed.
    - You can quickly & with little to no effort navigate through code, variables/registers, threads, stack frames.
    - You can see the value of variables/registers in different formats: from decimal, to hex, to binary, etc...
    - You can enjoy a graphical layout, who doesn't rather have that ? — *Yeah, cause you're weird.*

    There are more... but this should suffice.

- *[Anyways]* Here's what it looks like:

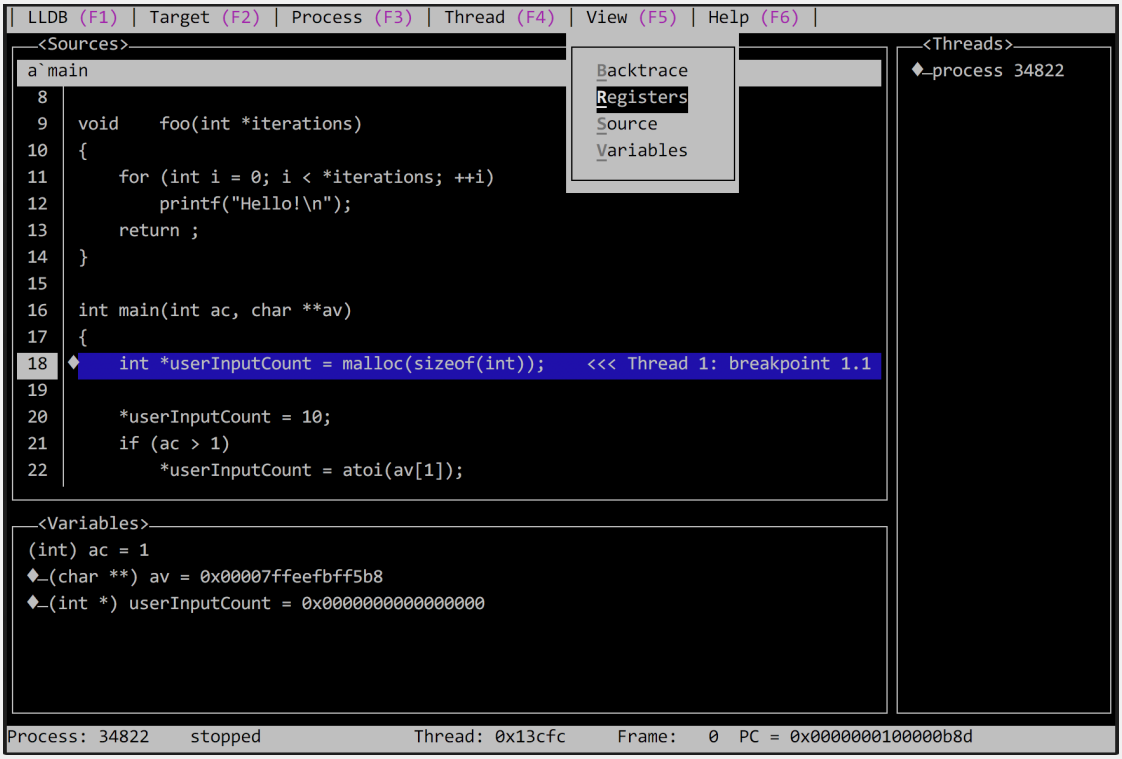
    

    > *This is a small terminal window only in order to be able to make a `gif` who's text is discernable. Realistically, on a bigger terminal screen, it'll look like this:*

- It *[the `gui`]* has 3 *[by default, but up to 4]* screen subdivisions, called views (or window panes):

Window Pane	Description
Source	Displays, *[currently executing]* and surrounding, *[assembly]* instructions, or if the `-g` flag was used *[in compilation]*, source code.
Variables	Displays variables *[, belonging to the current stack frame,]* and their currently held values.
Registers	Displays registers *[, belonging to the current stack frame,]* and their currently held values.
Backtrace *(Threads & Stack Frames)*	Displays the process's current threads and their stack frames.

> ***[Note:***
>
> - *The "registers" window pane is not visible on entry of the `gui`, you must toggle it on, from: `>Source Code View > Top Menu > Views > Registers`.*
>
> 
>
>
> *- **end note**]*

🏠 | ⬅️ | ➡️

3.6.1. Usage Commands

`[Search Tags: >guiusagecommands >guiusagecmds >usagecommandsgui >usagecmdsgui >graphicaluserinterfaceusagecommands >graphicaluserinterfaceusagecmds >usagecommandsgraphicaluserinterface >usagecmdsgraphicaluserinterface >guicommands >guicmds >commandsgui >cmdsgui >graphicaluserinterfacecommands >graphicaluserinterfacecmds >commandsgraphicaluserinterface >cmdsgraphicaluserinterface]`

- **Enter & Exit `GUI` *[mode]*:**

> `[Search Tags: >entergui >entergraphicaluserinterface >quitgui >closegui >leavegui exitgui >quitgraphicaluserinterface >closegraphicaluserinterface >leavegraphicaluserinterface >exitgraphicaluserinterface]`

[After starting your program,] to enter `GUI` *[mode]*, type "`gui`" *[in the `lldb` command prompt]* — like so:

```
(lldb) run [<arg> ...]
.
.
.
(lldb) gui
```

To leave `GUI` *[mode]*, press *[the]* `esc` *(escape) [key] — from anywhere, and any window pane*.

```
Press, 'esc'.
```

- **Common *[Help Menu]*:**

> `[Search Tags: >commonguihelpmenu >commonguimenu >commonguipage >commonhelpmenu >commonmenu >commonpage >generalguihelpmenu >generalguimenu >generalguipage >generalhelpmenu >generalmenu >generalpage >guigeneralhelpmenupage >guigeneralmenuhelppage >guigeneralmenuhelppage >helpmenupageguigeneral >menuhelppageguigeneral >menuhelppageguigeneral >helpmenupagegeneral >menuhelppagegeneral >menuhelppagegeneral >helpmenugeneral >menuhelpgeneral >menuhelpgeneral >helpgeneral >menugeneral >generalhelpmenupage >generalmenuhelppage >generalmenuhelppag >guigeneralcommands >guigeneralcmds >commandsguigeneral >cmdsguigeneral >commandsgeneral >cmdsgeneral >generalcommands >generalcmds >generalpage >generalpage >pagegeneral >pagegeneral >commonhelpmenupage >commonmenuhelppage >commonmenuhelppage >common >common >common >common >common >common >common >common >common >common >commonhelpmenupage >commonmenuhelppage >commonmenuhelppag >commoncommands >commoncmds >common >common >common >common >commoncommands >commoncmds >commonpage >commonpage >common >common]`

Key	Action
tab	Select next view
h	Show help dialog with view specific key bindings
,	Page up

Key	Action
.	Page down
up	Select previous
down	Select next
left	Unexpand or select parent
right	Expand
page-up	Page up
page-down	Page down
esc	Quit `gui` mode

*[**Note:** Aside from these common keyboard shortcut commands, views also have specific keyboard shortcut commands of their own, press `h` to open a dialog to display them.*

- Here are the remaining *[view]* help menus:

    1. Source Help Menu
    2. Variables & Registers Help Menu
    3. Backtrace Help Menu

*- **end note**]*

- **Source Code *[Help Menu]*:**

```
[Search Tags: >srcguihelpmenu >srcguimenu >srcguipage >srchelpmenu >srcmenu >srcpage
>sourceguihelpmenu >sourceguimenu >sourceguipage >sourcehelpmenu >sourcemenu
>sourcepage>sourcecodeguihelpmenu >sourcecodeguimenu >sourcecodeguipage
>sourcecodehelpmenu >sourcecodemenu >sourcecodepage>guisourcecodehelpmenupage
>guisourcecodemenuhelppage >guisourcecodemenuhelppage >helpmenupageguisourcecode
>menuhelppageguisourcecode >menuhelppageguisourcecode >helpmenupagesourcecode
>menuhelppagesourcecode >menuhelppagesourcecode >helpmenusourcecode
>menuhelpsourcecode >menuhelpsourcecode >helpsourcecode >menusourcecode
>sourcecodehelpmenupage >sourcecodemenuhelppage >sourcecodemenuhelppag
>guisourcecodecommands >guisourcecodecmds >commandsguisourcecode >cmdsguisourcecode
>commandssourcecode >cmdssourcecode >sourcecodecommands >sourcecodecmds
>sourcecodepage >sourcecodepage >pagesourcecode >pagesourcecode
>guisourcehelpmenupage >guisourcemenuhelppage >guisourcemenuhelppage
>helpmenupageguisource >menuhelppageguisource >menuhelppageguisource
>helpmenupagesource >menuhelppagesource >menuhelppagesource >helpmenusource
>menuhelpsource >menuhelpsource >helpsource >menusource >sourcehelpmenupage
>sourcemenuhelppage >sourcemenuhelppag >guisourcecommands >guisourcecmds
```

```
>commandsguisource >cmdsguisource >commandssource >cmdssource >sourcecommands
>sourcecmds >sourcepage >sourcepage >pagesource >pagesource]
```

Key	Action
`return`	Run to selected line with one shot breakpoint
`up`	Select previous source line
`down`	Select next source line
`page-up`	Page up
`page-down`	Page down
`b`	Set breakpoint on selected source/disassembly line
`c`	Continue process
`d`	Detach and resume process
`D`	Detach with process suspended
`h`	Show help dialog
`k`	Kill process
`n`	Step over (source line)
`N`	Step over (single instruction)
`o`	Step out
`s`	Step in (source line)
`S`	Step in (single instruction)
`,`	Page up
`.`	Page down

- **Variables & Registers *[Help Menu]*:**

```
[Search Tags: >variablesguihelpmenu >variableguihelpmenu >variablesguimenu
>variableguimenu >variablesguipage >variableguipage >variableshelpmenu
>variablehelpmenu >variablesmenu >variablemenu >variablespage >variablepage
>varguihelpmenu >varguimenu >varguipage >varhelpmenu >varmenu >varpage
>varsguihelpmenu >varsguimenu >varsguipage >varshelpmenu >varsmenu >varspage
>registerguihelpmenu >registerguimenu >registerguipage >registerhelpmenu
>registermenu >registerpage >regguihelpmenu >regsguihelpmenu >regguimenu >regsguimenu
>regguipage >regsguipage >reghelpmenu >regshelpmenu >regmenu >regsmenu >regpage
>regspage >guivariablehelpmenupage >guivariablemenuhelppage >guivariablemenuhelppage
>helpmenupageguivariable >menuhelppageguivariable >menuhelppageguivariable
```

```
>helpmenupagevariable >menuhelppagevariable >menuhelppagevariable >helpmenuvariable
>menuhelpvariable >menuhelpvariable >helpvariable >menuvariable >variablehelpmenupage
>variablemenuhelppage >variablemenuhelppag >guivariablecommands >guivariablecmds
>commandsguivariable >cmdsguivariable >commandsvariable >cmdsvariable
>variablecommands >variablecmds >variablepage >variablepage >pagevariable
>pagevariable >guiregistershelpmenupage >guiregistersmenuhelppage
>guiregistersmenuhelppage >helpmenupageguiregisters >menuhelppageguiregisters
>menuhelppageguiregisters >helpmenupageregisters >menuhelppageregisters
>menuhelppageregisters >helpmenuregisters >menuhelpregisters >menuhelpregisters
>helpregisters >menuregisters >registershelpmenupage >registersmenuhelppage
>registersmenuhelppag >guiregisterscommands >guiregisterscmds >commandsguiregisters
>cmdsguiregisters >commandsregisters >cmdsregisters >registerscommands >registerscmds
>registerspage >registerspage >pageregisters >pageregisters >guivarhelpmenupage
>guivarmenuhelppage >guivarmenuhelppage >helpmenupageguivar >menuhelppageguivar
>menuhelppageguivar >helpmenupagevar >menuhelppagevar >menuhelppagevar >helpmenuvar
>menuhelpvar >menuhelpvar >helpvar >menuvar >varhelpmenupage >varmenuhelppage
>varmenuhelppag >guivarcommands >guivarcmds >commandsguivar >cmdsguivar >commandsvar
>cmdsvar >varcommands >varcmds >varpage >varpage >pagevar >pagevar]
```

Key	Action
`up`	Select previous item
`down`	Select next item
`right`	Expand selected item
`left`	Unexpand selected item or select parent if not expanded
`page-up`	Page up
`page-down`	Page down
`A`	Format as annotated address
`b`	Format as binary
`B`	Format as hex bytes with ASCII
`c`	Format as character
`d`	Format as a signed integer
`D`	Format selected value using the default format for the type
`f`	Format as float
`h`	Show help dialog
`i`	Format as instructions
`o`	Format as octal

- **Backtrace *[Help Menu]*:**

> `[Search Tags: >backtraceguihelpmenu >backtraceguimenu >backtraceguipage >backtracehelpmenu >backtracemenu >backtracepage >btguihelpmenu >btguimenu >btguipage >bthelpmenu >btmenu btpage >guipanesthreads >guipanesthrds >guipanesthreads >guipanesthrds >guipanethreads >guipanethrds >guipanethreads >guipanethrds >guithreadspanes >guithrdspanes >guithreadspanes >guithrdspanes >guithreadspanes >guithrdspanes >guipanesstackframes >guipanesstckframes >guipanestackframes >guipanestckframes >guipanesframes >guipaneframes >guistackframes >guistckframes >guiframes >guiframes >guibacktracehelp >helpbacktrace >hbracktrace >hbt >guibt >helpbt]`

Key	Action
`up`	Select previous item
`down`	Select next item
`right`	Expand the selected item
`left`	Unexpand the selected item or select parent if
`page-up`	Page up
`page-down`	Page down
`h`	Show help dialog
`space`	Toggle item expansion
`,`	Page up
`.`	Page down

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help gui`
2	Manual Page	Unix / Linux / MacOS	`(lldb-gui-source-view) h`
3	Manual Page	Unix / Linux / MacOS	`(lldb-gui-variables-view) h`
4	Manual Page	Unix / Linux / MacOS	`(lldb-gui-registers-view) h`
5	Manual Page	Unix / Linux / MacOS	`(lldb-gui-backtrace-view) h`

🏠 | ⬅️ | ➡️

3.7. Control Process Execution

[Search Tags: >lldb.controlprocessexecution >lldbcontrolprocessexecution >debugger.controlprocessexecution >debuggercontrolprocessexecution >controlprocessexecution >contrlprocessexecution >cntrlprocessexecution >ctrlprocessexecution >controlprcsexecution >contrlprcsexecution >cntrlprcsexecution >ctrlprcsexecution >controlprcsexecs >contrlprcsexecs >cntrlprcsexecs >ctrlprcsexecs >controlsection >ctrlsection >processcontrolsection >prcsctrlsection >prcscontrolsection >processcontrolsection >lldb.controlprogramexecution >lldbcontrolprogramexecution >debugger.controlprogramexecution >debuggercontrolprogramexecution >controlprogramexecution >contrlprogramexecution >cntrlprogramexecution >ctrlprogramexecution >controlprogexecution >controlprogsexecution >contrlprogexecution >contrlprogsexecution >cntrlprogexecution >cntrlprogsexecution >ctrlprogexecution >ctrlprogsexecution >controlprogexecs >controlprogsexecs >contrlprogexecs >contrlprogsexecs >cntrlprogexecs >cntrlprogsexecs >ctrlprogexecs >ctrlprogsexecs >controlsection >ctrlsection >programcontrolsection >progctrlsection >progsctrlsection >progcontrolsection >progscontrolsection >programcontrolsection]

Contents

About

After process `launch` / `attach`, `lldb` completely hands over the control of execution of the process to you, and it *[`lldb`]* gives you commands with which to control it.

The following section will layout the **`lldb` prompt commands** offered to control the program — not the **graphical user interface commands**, those are laid out in section (§3.6.2 @ Source Code *[Commands Help Menu]*).

Control commands:

[Search Tags: >ctrlcommands >controlcommands >cntrlcommands >ctrlcmds >controlcmds >cntrlcmds >commandsctrl >commandscontrol >commandscntrl >cmdsctrl >cmdscontrol >cmdscntrl >continue >cmdcontinue >continuecmd >commandcontinue >continuecommand >continue >cmdcontinue >continuecmd >commandcontinue >continuecommand >until >cmduntil >untilcmd >commanduntil >untilcommand >step >cmdstep >stepcmd >commandstep >stepcommand >stepin >stepout >stepover >cmdstepin >cmdstepout >cmdstepover >stepincmd >stepoutcmd >stepovercmd >commandstepin >commandstepout >commandstepover >stepincommand >stepoutcommand >stepovercommand >kill >cmdkill >killcmd >commandkill >killcommand >detach >cmddetach >detachcmd >commanddetach >detachcommand]

Command	Description
`(lldb) continue`	**Continue execution** *[of all threads in the current process] [till a breakpoint is hit or termination of the process is met]*.
`(lldb) thread until <line>`	**Run until line** `<line>` or control leaves the current function.

Command	Description

`(lldb) thread step-over`
`(lldb) next`
`(lldb) n`
	Step over *[function call]* lines *[, if any]*, executing the current line and stepping, thereafter, over it, to the next one. Defaults to current thread unless specified.

`(lldb) thread step-in`
`(lldb) step`
`(lldb) s`
	Step into *[function]* calls. Defaults to current thread unless specified.

`(lldb) thread step-out` `(lldb) finish`	**Step out** of the currently selected frame *(i.e function call)*.
`(lldb) kill`	**Terminate** the current target process.
`(lldb) detach`	**Detach** from the current target process.

> ***[Note:***
>
> - For *[assembly]* **instructions**, the step in/out commands are slightly different;
>
> | Command | Description |
> |---|---|
> | `(lldb) thread step-inst-over`
`(lldb) ni` | Do a single *[instruction]* **step over**. |
> | `(lldb) thread step-inst`
`(lldb) si` | Do a single *[instruction]* **step in**. |
>
> ***- end note]***

> ***Further Reading:***
>
#	Type	Author	Link
1	Documentation	LLDB	(Official) Tutorial | GDB to LLDB command map
2	Manual Page	Unix / Linux / MacOS	`(lldb) help thread`
3	Manual Page	Unix / Linux / MacOS	`(lldb) help continue`
4	Manual Page	Unix / Linux / MacOS	`(lldb) help thread until`
5	Manual Page	Unix / Linux / MacOS	`(lldb) help thread step-in`
6	Manual Page	Unix / Linux / MacOS	`(lldb) help thread step-out`
7	Manual Page	Unix / Linux / MacOS	`(lldb) help thread step-over`

#	Type	Author	Link
8	Manual Page	Unix / Linux / MacOS	`(lldb) help thread step-inst`
9	Manual Page	Unix / Linux / MacOS	`(lldb) help thread stepi-inst-over`
10	Manual Page	Unix / Linux / MacOS	`(lldb) help kill`
11	Manual Page	Unix / Linux / MacOS	`(lldb) help detach`

🏠 | ⬅️ | ➡️

3.8. Examine

`[Search Tags: >lldb.examine >lldbexamine >debuggerexamine >debugger.examine >lldb.examination >lldbexamination >debuggerexamination >debugger.examination >lldb.examinating >lldbexaminating >debuggerexaminating >debugger.examinating >examine >examination >examinating]`

Commands for examining:

Once the program stops execution *(e.g. due to a breakpoint, watchpoint, manual stop, crash, etc ...)*, you will be able to examine *(or inspect)* the state of the process.

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help source`
2	Manual Page	Unix / Linux / MacOS	`(lldb) help thread`
3	Manual Page	Unix / Linux / MacOS	`(lldb) help frame`
4	Documentation	LLDB	(Official) Tutorial
5	Documentation	LLDB	GDB to LLDB Command Map
6	Documentation	Apple	LLDB Quick Start Tutorial

🏠 | ⬅️ | ➡️

3.8.1. Source Code

[Search Tags: >sourcecodelldb >lldb.sourcecode >lldbsourcecode >examinesourcecode >sourcecodeexamine >sourcecodeexamination >examinesourcecde >sourcecdeexamine >sourcecdeexamination >examinesrccode >srccodeexamine >srccodeexamination >examinesrccde >srccdeexamine >srccdeexamination >examinecode >examinecode >scodeexamine >scodeexamine]

> *[**Note:** `gui` mode already displays the source code automatically during execution — the following command is for `lldb`'s textual mode. - **end note**]*

- **List source code:**

> [Search Tags: >sourcelist >solist >listsource >listso >sourcedisplay >sodisplay >displaysource >displayso >sourceshow >soshow >showsource >showso >sourcelist >listsource >sourcelst >lstsource >srclist >listsrc >srclst >lstsrc >lssrc >lssource]

> ***Synopsis:***
>
> ```
> source list [--show-breakpoints] [--count <count>] [--file
> <filename>] [--line <linenum>]
> source list [--show-breakpoints] [--count <count>] [--name
> <program-symbol>]
> ```
>
> ***Example(s):***
>
> ```
> (lldb) source list --count 20 --file main.c --line 5    # list
> <count> lines from <file> starting from line <line>
> (lldb) so li -c 20 -f main.c -l 5
> ```
>
> ```
> (lldb) source list --count 25 --name foo                # list
> <count> lines having to do with <program-symbol>
> (lldb) so li -c 25 -n foo
> ```

> ***Further Reading:***
>
> | # | Type | Author | Link |
> |---|---|---|---|
> | 1 | Manual Page | Unix / Linux / MacOS | `(lldb) help source list` |
> | 2 | Manual Page | Unix / Linux / MacOS | `(lldb) help source info` |

3.8.2. State of Threads

[Search Tags: >threadexamination >examinationthread >examthreads >threadsexam >threadexam >examthrds >thrdsexam >thrdexam >examinthreads >threadsexamin >threadexamin >examinthrds >thrdsexamin >thrdexamin >examinethreads >threadsexamine >threadexamine >examinethrds >thrdsexamine >thrdexamine >examinatingthreads >threadsexaminating >threadexaminating >examinatingthrds >thrdsexaminating >thrdexaminating >threadsstate >thrdsstate >threadstate >thrdstate >thrsstate >thrstate >thsstate >thstate >tsstate >tstate >statethreads >statethrds >statethread >statethrd >statethrs]

Commands to:

- 1 List threads
- 2 List thread information
- 3 Select *[current]* thread
- 4 Thread backtrace

To inspect the current state of your process, you can start with the threads:

- **List threads:**

> [Search Tags: >listthreads >lstthreads >lsthreads >threadslist >threadlist >threadslst >threadlst >threadsls >threadls >listthrds >lstthrds >lsthrds >thrdslist >thrdlist >thrdslst >thrdlst >thrdsls >thrdls]

> ***Synopsis:***
>
> ```
> thread list
> ```
>
> ***Example(s):***
>
> ```
> (lldb) thread list
> (lldb) th l
> ```
>
> ```
> * thread #1: tid = 0xa3727, 0x0000000100003b49 a`main(ac=1,
> av=0x00007ffeefbff5b0) at threadedHello.cpp:29, queue =
> 'com.apple.main-thread', stop reason = one-shot breakpoint 3
>   thread #2: tid = 0xa37da, 0x00007fff621d5d8a
> libsystem_kernel.dylib`__semwait_signal + 10
>   thread #3: tid = 0xa37db, 0x00007fff621d5d8a
> libsystem_kernel.dylib`__semwait_signal + 10
>   thread #4: tid = 0xa37dc, 0x00007fff621d5d8a
> libsystem_kernel.dylib`__semwait_signal + 10
> ```

```
  thread #5: tid = 0xa37dd, 0x00007fff621d5d8a
libsystem_kernel.dylib`__semwait_signal + 10
  thread #6: tid = 0xa37de, 0x00007fff621d5d8a
libsystem_kernel.dylib`__semwait_signal + 10
  thread #7: tid = 0xa37df, 0x00007fff621d5d8a
libsystem_kernel.dylib`__semwait_signal + 10
  thread #8: tid = 0xa37e0, 0x00007fff621d5d8a
libsystem_kernel.dylib`__semwait_signal + 10
  thread #9: tid = 0xa37e1, 0x00007fff621d5d8a
libsystem_kernel.dylib`__semwait_signal + 10
  thread #10: tid = 0xa37e2, 0x00007fff621d5d8a
libsystem_kernel.dylib`__semwait_signal + 10
  thread #11: tid = 0xa37e3, 0x000000010013d7e0
libclang_rt.asan_osx_dynamic.dylib`__asan::ReportGenericError(uns
igned long, unsigned long, unsigned long, unsigned long, bool,
unsigned long, unsigned int, bool)
```

*[**Note:** The `*` indicates that `thread #1` is the current thread. - **end note**]*

- **Select *[current]* thread:**

`[Search Tags: >selectthreads >selctthreads >selcthreads >threadsselect >threadselect >threadsselct >threadselct >threadsslct >threadslct >slctthreads >slctthread >selectthrds >selctthrds >selcthrds >thrdsselect >thrdlist >thrdsselct >thrdselct >thrdsslct >thrdslct >slctthrds >slctthrd >sthreads >sthrds >sethreads >sethrds >selthreads >selthrds]`

Synopsis:

```
thread select <thread-index>
```

Example(s):

```
(lldb) thread select 2
(lldb) th se 2
```

*[**Note:***

- *The `select`'ed thread will be used by default in all the commands in the next section.*
- *Thread index is just the one shown in the `thread list` listing.*

*- **end note**]*

- **List thread information:**

> [Search Tags: >infothreads >threadsinfo >threadinfo >infothrds >thrdsinfo >thrdinfo >informationthreads >threadsinformation >threadinformation >informationthrds >thrdsinformation >thrdinformation >infthreads >infthrds]

> ***Synopsis:***
>
> ```
> thread info [--json] [--stop-info] [<thread-index> | all]
> ```
>
> ```
> (lldb) thread info --json
> (lldb) th i -j
> ```
>
> ***[Example] Output:***
>
> ```
> thread #1: tid = 0x9dbb2, 0x0000000100000a6c
> a`foo(iterations=0x00006020000000f0) at loopInput.c:11, queue =
> 'com.apple.main-thread', stop reason = step in
>         > {
>   "dispatch_queue_t" : 140735783837760,
>   "pthread_t" : 140735784059776,
>   "requested_qos" : {
>     "constant_name" : "QOS_CLASS_USER_INTERACTIVE",
>     "enum_value" : 33,
>     "printable_name" : "User Interactive"
>   },
>   "tsd_address" : 140735784060000
> }
> ```

> *[**Note:** The `*` indicates that `thread #1` is the current thread. - **end note**]*

- **Thread backtrace:**

> [Search Tags: >threadsbactrace >threadsbactrace >thrdbactrace >thrdsbactrace >bactracethreads >bactracethrds >backtracethread >threadbacktrace >backtrcethread >threadbacktrce >bcktrcethread >threadbcktrce >bcktracethread >threadbcktrace >bcktrthread >threadbcktr >bktrthread >threadbktr >btthread >threadbt >backtracethrds >backtracethrd >thrdsbacktrace >thrdbacktrace >backtrcethrds >backtrcethrd >thrdsbacktrce >thrdbacktrce >bcktrcethrds >bcktrcethrd >thrdsbcktrce >thrdbcktrce >bcktracethrds >bcktracethrd >thrdsbcktrace >thrdbcktrace >bcktrthrds >bcktrthrd >thrdsbcktr >thrdbcktr >bktrthrds >bktrthrd >thrdsbktr >thrdbktr >btthrds >btthrd >thrdsbt >thrdbt >stacktracethread >threadstacktrace >stacktrcethread

```
>threadstacktrce >stacktracethrds >stacktracethrd >thrdsstacktrace >thrdstacktrace
>stacktrcethrds >stacktrcethrd >thrdsstacktrce >thrdstacktrce >stcktracethread
>stktracethread >threadstcktrace >threadstktrace >stcktrcethread >stktrcethread
>threadstcktrce >threadstktrce >stcktracethrds >stktracethrds >stcktracethrd
>stktracethrd >thrdsstcktrace >thrdsstktrace >thrdstcktrace >thrdstktrace
>stcktrcethrds >stktrcethrds >stcktrcethrd >stktrcethrd >thrdsstcktrce >thrdsstktrce
>thrdstcktrce >thrdstktrce >showbacktrace >backtraceshow >showbacktrce >backtrceshow
>showbcktrace >bcktraceshow >showbcktrce >bcktrceshow >showbcktr >bcktrshow
>showbktrce >bktrceshow >showbktr >bktrshow >showbt >btshow]
```

Synopsis:

```
thread backtrace [--count <count>] [--start <frame-index>]    #
Backtrace [the first <count> frames] [starting from the frame
<frame-index> for] the current thread.
thread backtrace [all]                                        #
Show backtrace all threads.
```

Example(s):

```
(lldb) thread backtrace                          # Show the stack
backtrace for the current thread.
(lldb) th b
(lldb) bt
```

```
(lldb) thread backtrace --count 5                # Backtrace the
first 5 frames for the current thread.
(lldb) th b -c 5
(lldb) bt -c 5
(lldb) bt 5
```

```
(lldb) thread backtrace --count 2 --start 4      # Backtrace the
first 5 frames starting from the frame #4, for the current
thread.
(lldb) th b -c 2 -s 4
```

```
(lldb) thread backtrace all                      # Show the stack
backtraces for all threads.
(lldb) th b all
```

[Example] Output:

```
thread #1: tid = 0x2c03, stop reason = breakpoint 1.1, queue =
com.apple.main-thread
frame #0: 0x0000000100010d5b, where = Sketch`-[SKTGraphicView
alignLeftEdges:] + 33 at /Projects/Sketch/SKTGraphicView.m:1405
frame #1: 0x00007fff8602d152, where = AppKit`-[NSApplication
sendAction:to:from:] + 95
frame #2: 0x00007fff860516be, where = AppKit`-[NSMenuItem
_corePerformAction] + 365
frame #3: 0x00007fff86051428, where = AppKit`-[NSCarbonMenuImpl
performActionWithHighlightingForItemAtIndex:] + 121
frame #4: 0x00007fff860370c1, where = AppKit`-[NSMenu
performKeyEquivalent:] + 272
frame #5: 0x00007fff86035e69, where = AppKit`-[NSApplication
_handleKeyEquivalent:] + 559
frame #6: 0x00007fff85f06aa1, where = AppKit`-[NSApplication
sendEvent:] + 3630
frame #7: 0x00007fff85e9d922, where = AppKit`-[NSApplication run]
+ 474
frame #8: 0x00007fff85e965f8, where = AppKit`NSApplicationMain +
364
frame #9: 0x0000000100015ae3, where = Sketch`main + 33 at
/Projects/Sketch/SKTMain.m:11
frame #10: 0x0000000100000f20, where = Sketch`start + 52
```

[Note:

- *`bt` is an alias for `backtrace thread`, see `help bt`.*

*- **end note]***

Further Reading:

#	Type	Author	Link
1	Manual Page	LLDB	`(lldb) help thread list`
2	Manual Page	LLDB	`(lldb) help thread select`
3	Manual Page	LLDB	`(lldb) help thread info`
4	Manual Page	LLDB	`(lldb) help thread backtrace`
5	Documentation	LLDB	(Official) Tutorial :: Examine Thread State
6	Documentation	LLDB	Gdb to LLDB Command Map :: Examine Thread State
7	Documentation	Apple	LLDB Tutorial :: Examining Thread State
8	Encyclopedia	Wikipedia	Stack Trace

#	Type	Author	Link
9	Documentation	GNU	Backtraces

🏠 | ⬅️ | ➡️

3.8.3. State of Stack Frames

[Search Tags: >stackexamination >examinationstack >examstacks >stacksexam >stackexam >examstcks >examstks >stcksexam >stksexam >stckexam >stkexam >examinstacks >stacksexamin >stackexamin >examinstcks >examinstks >stcksexamin >stksexamin >stckexamin >stkexamin >examinestacks >stacksexamine >stackexamine >examinestcks >examinestks >stcksexamine >stksexamine >stckexamine >stkexamine >examinatingstacks >stacksexaminating >stackexaminating >examinatingstcks >examinatingstks >stcksexaminating >stksexaminating >stckexaminating >stkexaminating >frameexamination >examinationframe >examframes >framesexam >frameexam >examfrms >examfrs >frmsexam >frsexam >frmexam >frexam >examinframes >framesexamin >frameexamin >examinfrms >examinfrs >frmsexamin >frsexamin >frmexamin >frexamin >examineframes >framesexamine >frameexamine >examinefrms >examinefrs >frmsexamine >frsexamine >frmexamine >frexamine >examinatingframes >framesexaminating >frameexaminating >examinatingfrms >examinatingfrs >frmsexaminating >frsexaminating >frmexaminating >frexaminating >stacksstate >stcksstate >stackstate >stckstate >stksstate >stkstate >stsstate >ststate >tsstate >tstate >statestacks >statestcks >statestack >statestck >statestks >framesstate >frmsstate >framestate >frmstate >frsstate >frstate >stsstate >ststate >tsstate >tstate >stateframes >statefrms >stateframe >statefrm >statefrs >stackframeexamination >examinationstackframe >examstackframes >stackframesexam >stackframeexam >examinstackframes >stackframesexamin >stackframeexamin >examinestackframes >stackframesexamine >stackframeexamine >examinatingstackframes >stackframesexaminating >stackframeexaminating >stackframesstate >stackframestate >statestackframes >statestackframe]

Commands to:

- 1 Select *[current]* Stack Frame
- 2 List Stack Frame Information

- **Select *[current]* stack frame:**

> [Search Tags: >selectstacks >selctstacks >selcstacks >stacksselect >stackselect >stacksselct >stackselct >stacksslct >stackslct >slctstacks >slctstack >sstacks >sestacks >selstacks >selectstcks >selctstcks >selcstcks >stcksselect >stcklist >stcksselct >stckselct >stcksslct >stckslct >slctstcks >slctstck >sstcks >sestcks >selstcks >selectstks >selctstks >selcstks >stksselect >stklist >stksselct >stkselct >stksslct >stkslct >slctstks >slctstk >sstks >sestks >selstks >selectframes >selctframes >selcframes >framesselect >frameselect >framesselct >frameselct >framesslct >frameslct >slctframes >slctframe >sframes >seframes >selframes

```
>selectfrms >selctfrms >selcfrms >frmsselect >frmlist >frmsselct >frmselct >frmsslct
>frmslct >slctfrms >slctfrm >sfrms >sefrms >selfrms >selectfrs >selctfrs >selcfrs
>frsselect >frlist >frsselct >frselct >frsslct >frslct >slctfrs >slctfr >sfrs >sefrs
>selfrs >selectstackframes >selctstackframes >selcstackframes >stackframesselect
>stackframeselect >stackframesselct >stackframeselct >stackframesslct >stackframeslct
>slctstackframes >slctstackframe >sstackframes >sestackframes >selstackframes]
```

Synopsis:

```
frame select [--relative <offset>] [<frame-index>]
```

```
down         # Select the stack frame: current-index - 1 ; that
is called by the current stack frame.
up           # Select the stack frame: current-index + 1 ; that
called the current stack frame.
```

Example(s):

```
(lldb) fr select 4                    # Select stack frame #4 for
the current thread.
(lldb) fr s 4
(lldb) f 4
```

```
(lldb) fr select --relative 3         # Select a stack frame
relative to the the current thread.
(lldb) fr s -r 3
(lldb) f s -r 3
```

```
(lldb) down
(lldb) frame select --relative=-1     # same as 'down'
```

```
(lldb) up
(lldb) frame select --relative=1      # same as 'up'
```

[Note:

- *`f` is an alias for `frame select`, see `(lldb) help f`.*
- *`down` and `up` are aliases, see `(lldb) help down` and `(lldb) help up`.*

> *- end note]*

- **List stack frame information:**

> [Search Tags: >infostackframes >stackframesinfo >stackframeinfo >informationstackframes >stackframesinformation >stackframeinformation >infstackframes >infostacks >stacksinfo >stackinfo >informationstacks >stacksinformation >stackinformation >infstacks >infostcks >stcksinfo >stckinfo >informationstcks >stcksinformation >stckinformation >infstcks >infostks >stksinfo >stkinfo >informationstks >stksinformation >stkinformation >infstks >infoframes >framesinfo >frameinfo >informationframes >framesinformation >frameinformation >infframes >infofrms >frmsinfo >frminfo >informationfrms >frmsinformation >frminformation >inffrms >infofrs >frsinfo >frinfo >informationfrs >frsinformation >frinformation >inffrs]

> ***Synopsis:***
>
> ```
> frame info
> ```
>
> ***Example(s):***
>
> ```
> (lldb) frame info     # List information about the currently
> selected frame, in the current thread.
> (lldb) fr i
> ```
>
> ***[Example] Output:***
>
> ```
> frame #0: 0x0000000100003b49 a`main(ac=2, av=0x00007ffeefbff5a0)
> at threadedHello.cpp:29
> ```

Further Reading:

#	Type	Author	Link
1	Manual Page	Unix / Linux / MacOS	`(lldb) help frame info`
2	Manual Page	Unix / Linux / MacOS	`(lldb) help frame select`

#	Type	Author	Link
3	Documentation	LLDB	(Official) Tutorial :: Examining Stack Frame State
4	Documentation	LLDB	GDB to LLDB Command Map :: Examining Thread State
5	Documentation	Apple	LLDB Tutorial :: Examining the Stack Frame State

---

🏠 | ⬅️ | ➡️

3.8.4. State of Stack Frame Variables

`[Search Tags: >]`

Contents

- :

> `[Search Tags: >termlaunchprocesses >termlaunchprograms >terminallaunchprocesses >terminallaunchprograms >shelllaunchprocesses >shelllaunchprograms >launchshell >shelllaunch >launchshll >shlllaunch >launchterminal >terminallaunch >lnchshell >shelllnch >lnchshll >shlllnch >lnchterminal >terminallnch >lashell >shellla >lashll >shllla >laterminal >terminalla >launchconfigshell >shellconfiglaunch >launchconfigshll >shllconfiglaunch >launchconfigterminal >terminalconfiglaunch >lnchconfigshell >shellconfiglnch >lnchconfigshll >shllconfiglnch >lnchconfigterminal >terminalconfiglnch >laconfigshell >shellconfigla >laconfigshll >shllconfigla >laconfigterminal >terminalconfigla >termlaunch >termlnch >termla >termconfiglaunch >termconfiglnch >termconfigla]`

> ***Synopsis:***
>
> ```
> frame variable
> ```
>
> ***Example(s):***
>
> ```
> 
> ```
>
> ```
> 
> ```

> ***[Note: . - end note]***

-A (--show-all-children) Ignore the upper bound on the number of children to show.

-D (--depth) Set the max recurse depth when dumping aggregate types (default is infinity).

-F (--flat) Display results in a flat format that uses expression paths for each variable or member.

-G (--gdb-format) Specify a format using a GDB format specifier string.

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

-O (--object-description) Display using a language-specific description API, if possible.

-P (--ptr-depth) The number of pointers to be traversed when dumping values (default is zero).

-R (--raw-output) Don't use formatting options.

-S (--synthetic-type) Show the object obeying its synthetic provider, if available.

-T (--show-types) Show variable types when dumping values.

-V (--validate) Show results of type validators.

-Y[] (--no-summary-depth=[]) Set the depth at which omitting summary information stops (default is 1).

-Z (--element-count) Treat the result of the expression as if its type is an array of this many values.

-a (--no-args) Omit function arguments.

-c (--show-declaration) Show variable declaration information (source file and line where the variable was declared).

-d (--dynamic-type) Show the object as its full dynamic type, not its static type, if available. Values: no-dynamic-values | run-target | no-run-target

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

-g (--show-globals) Show the current frame source file global and static variables.

-l (--no-locals) Omit local variables.

-r (--regex) The argument for name lookups are regular expressions.

-s (--scope) Show variable scope (argument, local, global, static).

-y (--summary) Specify the summary that the variable output should use.

-z (--summary-string) Specify a summary string to use to format the variable output.

> ***Further Reading:***

#	Type	Author	Link
1	Documentation	LLDB	Gdb to LLDB Command Map :: Examine Variables
1	n/a	n/a	n/a