lldb version : lldb-330.0.48 Debugger commands: copy & pasted manually by NickIN Cheers! Dt. 2016-09-15

Dark Blue is the main command **orange is subcommands.** **Purple is more sub-sub-commands Blue is sub-sub-commands options. Green is alias**

```
apropos           -- Find a list of debugger commands related to a particular word/subject.
breakpoint        -- A set of commands for operating on breakpoints. Also see _regexp-break.
command           -- A set of commands for managing or customizing the debugger commands.
disassemble       -- Disassemble bytes in the current function, or elsewhere in the executable program as specified by the user.
expression        -- Evaluate an expression (ObjC++ or Swift) in the current program context, using user defined variables and variables currently in scope.
frame             -- A set of commands for operating on the current thread's frames.
gdb-remote        -- Connect to a remote GDB server.  If no hostname is provided, localhost is assumed.
gui               -- Switch into the curses based GUI mode.
help              -- Show a list of all debugger commands, or give details about specific commands.
kdp-remote        -- Connect to a remote KDP server.  udp port 41139 is the default port number.
log               -- A set of commands for operating on logs.
memory            -- A set of commands for operating on memory.
platform          -- A set of commands to manage and create platforms.
plugin            -- A set of commands for managing or customizing plugin commands.
process           -- A set of commands for operating on a process.
quit              -- Quit out of the LLDB debugger.
register          -- A set of commands to access thread registers.
script            --  Pass an expression to the script interpreter for evaluation and return the results. Drop into the interactive interpreter if no
                      expression is given.
settings          -- A set of commands for manipulating internal settable debugger variables.
source            -- A set of commands for accessing source file information
target            -- A set of commands for operating on debugger targets.
thread            -- A set of commands for operating on one or more threads within a running process.
type              -- A set of commands for operating on the type system
version           -- Show version of LLDB debugger.
watchpoint        -- A set of commands for operating on watchpoints.
```

Current command abbreviations (type 'help command alias' for more info):

```
help di         --     ('target symbols add')  Add a debug symbol file to one of the target's current modules by specifying a path to a debug symbols file, or
                       using the options to specify a module to download symbols for.
attach          --     ('_regexp-attach')  Attach to a process id if in decimal, otherwise treat the argument as a process name to attach to.
b               --     ('_regexp-break')  Set a breakpoint using a regular expression to specify the location, where <linenum> is in decimal and <address>
                       is in hex.
bt              --     ('_regexp-bt')  Show a backtrace.  An optional argument is accepted; if that argument is a number, it specifies the number of frames to
                       display.
                       If that argument is 'all', full backtraces of all threads are displayed.
c               --     ('process continue')  Continue execution of all threads in the current process.
```

```
call        --    ('expression --')  Evaluate an expression (ObjC++ or Swift) in the current program context, using user defined variables and variables
                  currently in scope.
continue    --    ('process continue')  Continue execution of all threads in the current process.
detach      --    ('process detach')  Detach from the current process being debugged.
di          --    ('disassemble')  Disassemble bytes in the current function, or elsewhere in the executable program as specified by the user.
dis         --    ('disassemble')  Disassemble bytes in the current function, or elsewhere in the executable program as specified by the user.
display     --    ('_regexp-display')  Add an expression evaluation stop-hook.
down        --    ('_regexp-down')  Go down "n" frames in the stack (1 frame by default).
env         --    ('_regexp-env')  Implements a shortcut to viewing and setting environment variables.
exit        --    ('quit')  Quit out of the LLDB debugger.
f           --    ('frame select')  Select a frame by index from within the current thread and make it the current frame.
file        --    ('target create')  Create a target using the argument as the main executable.
finish      --    ('thread step-out')  Finish executing the function of the currently selected frame and return to its call site in specified thread
                  (current thread, if none specified).
image       --    ('target modules')  A set of commands for accessing information for one or more target modules.
j           --    ('_regexp-jump')  Sets the program counter to a new address.
jump        --    ('_regexp-jump')  Sets the program counter to a new address.
kill        --    ('process kill')  Terminate the current process being debugged.
l           --    ('_regexp-list')  Implements the GDB 'list' command in all of its forms except FILE:FUNCTION and maps them to the appropriate
                  'source list'commands.
list        --    ('_regexp-list')  Implements the GDB 'list' command in all of its forms except FILE:FUNCTION and maps them to the appropriate 'source
                  list' commands.
n           --    ('thread step-over')  Source level single step in specified thread (current thread, if none specified), stepping over calls.
next        --    ('thread step-over')  Source level single step in specified thread (current thread, if none specified), stepping over calls.
nexti       --    ('thread step-inst-over')  Single step one instruction in specified thread (current thread, if none specified), stepping over calls.
ni          --    ('thread step-inst-over')  Single step one instruction in specified thread (current thread, if none specified), stepping over calls.
p           --    ('expression --')  Evaluate an expression (ObjC++ or Swift) in the current program context, using user defined variables and
                  variables currently in scope.
po          --    ('expression -O  -- ')  Evaluate an expression (ObjC++ or Swift) in the current program context, using user defined variables
                  and variables currently in scope.
print       --    ('expression --')  Evaluate an expression (ObjC++ or Swift) in the current program context, using user defined variables and
                  variables currently in scope.
q           --    ('quit')  Quit out of the LLDB debugger.
r           --    ('process launch -c /bin/sh --')  Launch the executable in the debugger.
rbreak      --    ('breakpoint set -r %1')  Sets a breakpoint or set of breakpoints in the executable.
repl        --    ('expression -r  -- ')  Evaluate an expression (ObjC++ or Swift) in the current program context, using user defined variables and
                  variables currently in scope.
run         --    ('process launch -c /bin/sh --')  Launch the executable in the debugger.
s           --    ('thread step-in')  Source level single step in specified thread (current thread, if none specified).
si          --    ('thread step-inst')  Single step one instruction in specified thread (current thread, if none specified).
step        --    ('thread step-in')  Source level single step in specified thread (current thread, if none specified).
```

```
stepi          --      ('thread step-inst')  Single step one instruction in specified thread (current thread, if none specified).
t              --      ('thread select')  Select a thread as the currently active thread.
tbreak         --      ('_regexp-tbreak')  Set a one shot breakpoint using a regular expression to specify the location, where <linenum> is in decimal
                       and <address> is in hex.
undisplay      --      ('_regexp-undisplay')  Remove an expression evaluation stop-hook.
up             --      ('_regexp-up')  Go up "n" frames in the stack (1 frame by default).
x              --      ('memory read')  Read from the memory of the process being debugged.

For more information on any command, type 'help <command-name>'.
(lldb) help image (image is an alias to 'target modules')
The following subcommands are supported:

add          -- Add a new module to the current target's modules.
dump         -- A set of commands for dumping information about one or more target modules.
list         -- List current executable and dependent shared library images.
load         -- Set the load addresses for one or more sections in a target module.
lookup       -- Look up information within executable and dependent shared library images.
search-paths -- A set of commands for operating on debugger target image search paths.
show-unwind  -- Show synthesized unwind instructions for a function.

For more help on any particular subcommand, type 'help <command> <subcommand>'.
----------------------
(lldb) help apropos
Find a list of debugger commands related to a particular word/subject.
Syntax: apropos <search-word>
----------------------
(lldb) help breakpoint
The following subcommands are supported:
clear          --      Clears a breakpoint or set of breakpoints in the executable.
command        --      A set of commands for adding, removing and examining bits of code to be executed when the breakpoint is hit
                       (breakpoint 'commands').
delete         --      Delete the specified breakpoint(s).  If no breakpoints are specified, delete them all.
disable        --      Disable the specified breakpoint(s) without removing it/them.  If no breakpoints are specified, disable them all.
enable         --      Enable the specified disabled breakpoint(s). If no breakpoints are specified, enable all of them.
list           --      List some or all breakpoints at configurable levels of detail.
modify         --      Modify the options on a breakpoint or set of breakpoints in the executable.  If no breakpoint is specified, acts on
                       the last created breakpoint.  With the exception of -e, -d and -i, passing an empty argument clears the modification.
name           --      A set of commands to manage name tags for breakpoints
set            --      Sets a breakpoint or set of breakpoints in the executable.

For more help on any particular subcommand, type 'help <command> <subcommand>'.
```

```
(lldb) help breakpoint clear
Clears a breakpoint or set of breakpoints in the executable.
Syntax: breakpoint clear <cmd-options>

Command Options Usage:
breakpoint clear -l <linenum> [-f <filename>]

-f <filename> ( --file <filename> )
Specify the breakpoint by source location in this particular file.

-l <linenum> ( --line <linenum> )
Specify the breakpoint by source location at this particular line.

------
(lldb) help breakpoint command
The following subcommands are supported:

add            --      Add a set of commands to a breakpoint, to be executed whenever the breakpoint is hit.  If no breakpoint is
                       specified, adds the commands to the last created breakpoint.
delete         --      Delete the set of commands from a breakpoint.
list           --      List the script or set of commands to be executed when the breakpoint is hit.

For more help on any particular subcommand, type 'help <command> <subcommand>'.
-----------
(lldb) help breakpoint delete
Delete the specified breakpoint(s).  If no breakpoints are specified, delete them all.
Syntax: breakpoint delete <cmd-options> [<breakpt-id | breakpt-id-list>]

Command Options Usage:
breakpoint delete [-Df] [<breakpt-id | breakpt-id-list>]

-D ( --dummy-breakpoints )
Delete Dummy breakpoints - i.e. breakpoints set before a file is provided, which prime new targets.

-f ( --force )
Delete all breakpoints without querying for confirmation.
This command takes options and free-form arguments.  If your arguments resemble option specifiers
(i.e., they start with a - or --), you must use ' -- ' between the end of the command options and the beginning of the arguments.

-----------------
```

```
(lldb) help breakpoint enable
Enable the specified disabled breakpoint(s). If no breakpoints are specified, enable all of them.
Syntax: breakpoint enable [<breakpt-id | breakpt-id-list>]
---------
lldb) help breakpoint list
List some or all breakpoints at configurable levels of detail.
Syntax: breakpoint list <cmd-options> [<breakpt-id>]

Command Options Usage:
breakpoint list [-Dbi] [<breakpt-id>]
breakpoint list [-Dfi] [<breakpt-id>]
breakpoint list [-Div] [<breakpt-id>]

-D ( --dummy-breakpoints )
List Dummy breakpoints - i.e. breakpoints set before a file is provided, which prime new targets.

-b ( --brief )
Give a brief description of the breakpoint (no location info).

-f ( --full )
Give a full description of the breakpoint and its locations.

-i ( --internal )
Show debugger internal breakpoints

-v ( --verbose )
Explain everything we know about the breakpoint (for debugging debugger bugs).

This command takes options and free-form arguments.  If your arguments resemble option specifiers
(i.e., they start with a - or --), you must use ' -- ' between the end of the command options and the beginning of the arguments.
--------
(lldb) help breakpoint modify
Modify the options on a breakpoint or set of breakpoints in the executable.  If no breakpoint is specified, acts on the last
created breakpoint.  With the exception of -e, -d and -i, passing an empty argument clears the modification.

Syntax: breakpoint modify <cmd-options> [<breakpt-id | breakpt-id-list>]

Command Options Usage:
breakpoint modify     [-De] [-i <count>] [-o <boolean>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>] [-q <queue-name>] [-c <expr>]
                      [<breakpt-id | breakpt-id-list>]
```

```
breakpoint modify [-Dd] [-i <count>] [-o <boolean>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>] [-q <queue-name>] [-c <expr>] [<breakpt-id |
breakpt-id-list>]

-D ( --dummy-breakpoints )
Sets Dummy breakpoints - i.e. breakpoints set before a file is provided, which prime new targets.

-T <thread-name> ( --thread-name <thread-name> )
The breakpoint stops only for the thread whose thread name matches this argument.

-c <expr> ( --condition <expr> )
The breakpoint stops only if this condition expression evaluates to true.

-d ( --disable )
Disable the breakpoint.

-e ( --enable )
Enable the breakpoint.

-i <count> ( --ignore-count <count> )
Set the number of times this breakpoint is skipped before stopping.

-o <boolean> ( --one-shot <boolean> )
The breakpoint is deleted the first time it stop causes a stop.

-q <queue-name> ( --queue-name <queue-name> )
The breakpoint stops only for threads in the queue whose name is given by this argument.

-t <thread-id> ( --thread-id <thread-id> )
The breakpoint stops only for the thread whose TID matches this argument.

-x <thread-index> ( --thread-index <thread-index> )
The breakpoint stops only for the thread whose index matches this argument.
This command takes options and free-form arguments.  If your arguments resemble option specifiers
(i.e., they start with a - or  --), you must use ' -- ' between the end of the command options and the beginning of the arguments.
--------
(lldb) help breakpoint name
The following subcommands are supported:
add            --      Add a name to the breakpoints provided.
delete         --      Delete a name from the breakpoints provided.
list           --      List either the names for a breakpoint or the breakpoints for a given name.
For more help on any particular subcommand, type 'help <command> <subcommand>'.
```

```
--------
(lldb) help breakpoint set
Sets a breakpoint or set of breakpoints in the executable.
Syntax: breakpoint set <cmd-options>

Command Options Usage:
breakpoint set         [-DHo] -l <linenum> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-f <filename>] [-K <boolean>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -a <address-expression> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -n <function-name> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-f <filename>] [-K <boolean>] [-L <language>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -F <fullname> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-f <filename>] [-K <boolean>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -S <selector> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-f <filename>] [-K <boolean>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -M <method> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-f <filename>] [-K <boolean>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -r <regular-expression> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-f <filename>] [-K <boolean>] [-L <language>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -b <function-name> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-f <filename>] [-K <boolean>] [-L <language>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -p <regular-expression> [-s <shlib-name>] [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>]
                       [-q <queue-name>] [-f <filename>] [-N <breakpoint-name>]
breakpoint set         [-DHo] -E <language> [-i <count>] [-c <expr>] [-x <thread-index>] [-t <thread-id>] [-T <thread-name>] [-q <queue-name>] [-w <boolean>]
                       [-h <boolean>] [-N <breakpoint-name>]

-D ( --dummy-breakpoints )
Sets Dummy breakpoints - i.e. breakpoints set before a file is provided, which prime new targets.

-E <language> ( --language-exception <language> )
Set the breakpoint on exceptions thrown by the specified language (without options, on throw but not catch.)

-F <fullname> ( --fullname <fullname> )
Set the breakpoint by fully qualified function names. For C++ this means namespaces and all arguments, and for Objective
C this means a full function prototype with class and selector.   Can be repeated multiple times to make one breakpoint
for multiple names.

-H ( --hardware )
Require the breakpoint to use hardware breakpoints.
```

-K <boolean> (--skip-prologue <boolean>)
skip the prologue if the breakpoint is at the beginning of a function. If not set the target. skip-prologue setting is used.

-L <language> (--symbol-Language <language>)
Set the breakpoint on names that belong to the specified language

-M <method> (--method <method>)
Set the breakpoint by C++ method names. Can be repeated multiple times to make one breakpoint for multiple methods.

-N <breakpoint-name> (--breakpoint-name <breakpoint-name>)
Adds this to the list of names for this breakpoint.

-S <selector> (--selector <selector>)
Set the breakpoint by ObjC selector name. Can be repeated multiple times to make one breakpoint for multiple Selectors.

-T <thread-name> (--thread-name <thread-name>)
The breakpoint stops only for the thread whose thread name matches this argument.

-a <address-expression> (--address <address-expression>)
Set the breakpoint by address, at the specified address.

-b <function-name> (--basename <function-name>)
Set the breakpoint by function basename (C++ namespaces and arguments will be ignored). Can be repeated multiple times to make one breakpoint for multiple symbols.

-c <expr> (--condition <expr>)
The breakpoint stops only if this condition expression evaluates to true.

-f <filename> (--file <filename>)
Specifies the source file in which to set this breakpoint. Note, by default lldb only looks for files that are
#included if they use the standard include file extensions. To set breakpoints on .c/.cpp/.m/.mm files that are
#included, set target.inline-breakpoint-strategy to "always".

-h <boolean> (--on-catch <boolean>)
Set the breakpoint on exception catcH.

-i <count> (--ignore-count <count>)
Set the number of times this breakpoint is skipped before stopping.

-l <linenum> (--line <linenum>)

```
Specifies the line number on which to set this breakpoint.

-n <function-name> ( --name <function-name> )
Set the breakpoint by function name.  Can be repeated multiple times to make one breakpoint for multiple names

-o ( --one-shot )
The breakpoint is deleted the first time it causes a stop.

-p <regular-expression> ( --source-pattern-regexp <regular-expression> )
Set the breakpoint by specifying a regular expression which is matched against the source text in a source file or files
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"

-q <queue-name> ( --queue-name <queue-name> )
The breakpoint stops only for threads in the queue whose name is given by this argument.

-r <regular-expression> ( --func-regex <regular-expression> )
Set the breakpoint by function name, evaluating a regular-expression to find the function name(s).

-s <shlib-name> ( --shlib <shlib-name> )
Set the breakpoint only in this shared library.  Can repeat this option multiple times to specify multiple shared
libraries.

-t <thread-id> ( --thread-id <thread-id> )
The breakpoint stops only for the thread whose TID matches this argument.

-w <boolean> ( --on-throw <boolean> )
Set the breakpoint on exception throW.

-x <thread-index> ( --thread-index <thread-index> )
The breakpoint stops only for the thread whose indeX matches this argument.
----------------------------------------------------------------------------------------------------------------------
(lldb) help command
The following subcommands are supported:
alias          --      Allow users to define their own debugger command abbreviations.  This command takes 'raw' input (no need to quote stuff).
delete         --      Allow the user to delete user-defined regular expression, python or multi-word commands.
history        --      Dump the history of commands in this session.
regex          --      Allow the user to create a regular expression command.
script         --      A set of commands for managing or customizing script commands.
source         --      Read in debugger commands from the file <filename> and execute them.
unalias        --      Allow the user to remove/delete a user-defined command abbreviation.
```

```
For more help on any particular subcommand, type 'help <command> <subcommand>'.
------------------------------------------------------------------------------------------------------------------------

(lldb) help command alias
'alias' allows the user to create a short-cut or abbreviation for long commands, multi-word commands, and commands that take particular options.
Below are some simple examples of how one might use the 'alias' command:

command alias sc script'            // Creates the abbreviation 'sc' for the 'script' command.
' command alias bp breakpoint'        // Creates the abbreviation 'bp' for the 'breakpoint' command.  Since breakpoint commands are two-word
                                      // commands, the user will still need to enter the second word after 'bp', e.g. 'bp enable' or
                                      // 'bp delete'.
' command alias bpl breakpoint list'  // Creates the abbreviation 'bpl' for the two-word command 'breakpoint list'.

An alias can include some options for the command, with the values either filled in at the time the alias is created, or specified as positional
arguments, to be filled in when the alias is invoked.  The following example shows how to create aliases with options:

' command alias bfl breakpoint set -f %1 -l %2'
This creates the abbreviation 'bfl' (for break-file-line), with the -f and -l options already part of the alias.  So if the user wants to set a breakpoint
by file and line without explicitly having to use the -f and -l options, the user can now use 'bfl' instead.  The '%1' and '%2' are positional placeholders
for the actual arguments that will be passed when the alias command is used.
The number in the placeholder refers to the position/order the actual value occupies when the alias is used.  All the occurrences of '%1' in the alias
will be replaced with the first argument, all the occurrences of '%2' in the alias will be replaced with the second argument, and so on.  This also allows
actual arguments to be used multiple times within an alias (see 'process launch' example below).

Note: the positional arguments must substitute as whole words in the resultant command, so you can't at present do something like:

command alias bcppfl breakpoint set -f %1.cpp -l %2
to get the file extension ".cpp" automatically appended.  For more complex aliasing, use the "command regex" command instead.
So in the 'bfl' case, the actual file value will be filled in with the first argument following 'bfl' and the actual line number value will be filled in with
the second argument.  The user would use this alias as follows:

 (lldb) command alias bfl breakpoint set -f %1 -l %2
          <... some time later ...>
 (lldb) bfl my-file.c 137
       This would be the same as if the user had entered 'breakpoint set -f my-file.c -l 137'.
```

```
Another example:
 (lldb)  command alias pltty  process launch -s -o %1 -e %1
 (lldb)  pltty /dev/tty0
// becomes 'process launch -s -o /dev/tty0 -e /dev/tty0'

If the user always wanted to pass the same value to a particular option, the alias could be defined with that value directly in the alias as a constant, rather
than using a positional placeholder:

command alias bl3  breakpoint set -f %1 -l 3  // Always sets a breakpoint on line 3 of whatever file is
Syntax: command alias <alias-name> <cmd-name> [<options-for-aliased-command>]
-----------
 (lldb) help command delete
 Allow the user to delete user-defined regular expression, python or multi-word commands.
  Syntax: command delete <cmd-name>
 -----------
 (lldb) help command history
 Dump the history of commands in this session.
 Syntax: command history <cmd-options>
 Command Options Usage:
 command history  [-c <unsigned-integer>] [-s <unsigned-integer>] [-e <unsigned-integer>]
 command history  [-C]
  -C ( --clear )
 Clears the current command history.

 -c <unsigned-integer> ( --count <unsigned-integer> )
 How many history commands to print.

 -e <unsigned-integer> ( --end-index <unsigned-integer> )
 Index at which to stop printing history commands.

 -s <unsigned-integer> ( --start-index <unsigned-integer> )
 Index at which to start printing history commands (or end to mean tail mode).
------------
```

```
(lldb) help command regex
Allow the user to create a regular expression command.

Syntax: command regex <cmd-name> [s/<regex>/<subst>/ ...]

Command Options Usage:
command regex [-h <none>] [-s <none>]

-h <none> ( --help <none> )
The help text to display for this command.

-s <none> ( --syntax <none> )
A syntax string showing the typical usage syntax.

This command allows the user to create powerful regular expression commands  with substitutions. The regular expressions and substitutions are specified
using the regular expression substitution format of:

s/<regex>/<subst>/

<regex> is a regular expression that can use parenthesis to capture regular  expression input and substitute the captured matches in the output using %1
for the first match, %2 for the second, and so on.

The regular expressions can all be specified on the command line if more than one argument is provided. If just the command name is provided on the command
line, then the regular expressions and substitutions can be entered on separate lines, followed by an empty line to terminate the command definition.

EXAMPLES

The following example will define a regular expression command named 'f' that will call 'finish' if there are no arguments, or 'frame select <frame-idx>' if
a number follows 'f':
 (lldb) command regex f s/^$/finish/ 's/([0-9]+)/frame select %1/'
---------------
(lldb) help command script
The following subcommands are supported:
add            --      Add a scripted function as an LLDB command.
clear          --      Delete all scripted commands.
delete         --      Delete a scripted command.
import         --      Import a scripting module in LLDB.
list           --      List defined scripted commands.
```

```
 For more help on any particular subcommand, type 'help <command> <subcommand>'.
------------------

 (lldb) help command source
 Read in debugger commands from the file <filename> and execute them.
  Syntax: command source <cmd-options> <filename>

 Command Options Usage:
command source  [-e <boolean>] [-c <boolean>] [-s <boolean>] <filename>

 -c <boolean> ( --stop-on-continue <boolean> )
 If true, stop executing commands on continue.

 -e <boolean> ( --stop-on-error <boolean> )
 If true, stop executing commands on error.

 -s <boolean> ( --silent-run <boolean> )
 If true don't echo commands while executing.

This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- ' between
the end of the command options and the beginning of the arguments.
---------------------
 (lldb) help command unalias
 Allow the user to remove/delete a user-defined command abbreviation.
  Syntax: command unalias <alias-name>
 ------------------------------------------------------------------------------------------------------------------------------
 (lldb) help disassemble
 Disassemble bytes in the current function, or elsewhere in the executable program as specified by the user.

 Syntax: disassemble [<cmd-options>]

 Command Options Usage:
 disassemble [-bmr] -s <address-expression> [-C <num-lines>] [-P <plugin>] [-F <disassembly-flavor>] [-A <arch>] [-e <address-expression>]
 disassemble [-bmr] -s <address-expression> [-C <num-lines>] [-P <plugin>] [-F <disassembly-flavor>] [-A <arch>] [-c <num-lines>]
 disassemble [-bmr] [-C <num-lines>] [-P <plugin>] [-F <disassembly-flavor>] [-A <arch>] [-c <num-lines>] [-n <function-name>]
 disassemble [-bfmr] [-C <num-lines>] [-P <plugin>] [-F <disassembly-flavor>] [-A <arch>] [-c <num-lines>]
 disassemble [-bmpr] [-C <num-lines>] [-P <plugin>] [-F <disassembly-flavor>] [-A <arch>] [-c <num-lines>]
 disassemble [-blmr] [-C <num-lines>] [-P <plugin>] [-F <disassembly-flavor>] [-A <arch>]
 disassemble [-bmr] [-C <num-lines>] [-P <plugin>] [-F <disassembly-flavor>] [-A <arch>] [-a <address-expression>]

 -A <arch> ( --arch <arch> )
```

Specify the architecture to use from cross disassembly.

-C <num-lines> (--context <num-lines>)
Number of context lines of source to show.

-F <disassembly-flavor> (--flavor <disassembly-flavor>)
Name of the disassembly flavor you want to use. Currently the only valid options are default, and for Intel architectures, att and intel.

-P <plugin> (--plugin <plugin>)
Name of the disassembler plugin you want to use.

-a <address-expression> (--address <address-expression>)
Disassemble function containing this address.

-b (--bytes)
Show opcode bytes when disassembling.

-c <num-lines> (--count <num-lines>)
Number of instructions to display.

-e <address-expression> (--end-address <address-expression>)
Address at which to end disassembling.

-f (--frame)
Disassemble from the start of the current frame's function.

-l (--line)
Disassemble the current frame's current source line instructions if there is debug line table information, else disassemble around the pc.

-m (--mixed)
Enable mixed source and assembly display.

-n <function-name> (--name <function-name>)
Disassemble entire contents of the given function name.

-p (--pc)
Disassemble around the current pc.

-r (--raw)
Print raw disassembly with no symbol information.

```
 -s <address-expression> ( --start-address <address-expression> )
 Address at which to start disassembling.
 ----------------------------------------------------------------------------------------------------------------------------
(lldb) help expression
 Evaluate an expression (ObjC++ or Swift) in the current program context, using user defined variables and variables currently in scope.  This command takes
'raw' input (no need to quote stuff).
  Syntax: expression <cmd-options> -- <expr>

Command Options Usage:
expression [-AFLORTg] [-f <format>] [-G <gdb-format>] [-l <language>] [-a <boolean>] [-i <boolean>] [-t <unsigned-integer>] [-u <boolean>] [-v[<description-
verbosity>]] [-d <none>] [-S <boolean>] [-D <count>] [-P <count>] [-Y[<count>]] [-V <boolean>] -- <expr>

expression [-AFLORTg] [-l <language>] [-a <boolean>] [-i <boolean>] [-t <unsigned-integer>] [-u <boolean>] [-d <none>] [-S <boolean>] [-D <count>] [-P <count>]
[-Y[<count>]] [-V <boolean>] -- <expr>

expression [-r] -- <expr>
expression <expr>

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

 -D <count> ( --depth <count> )
 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> ( --gdb-format <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 <count> ( --ptr-depth <count> )
 The number of pointers to be traversed when dumping values (default is zero).

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

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

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

-V <boolean> (--validate <boolean>)
Show results of type validators.

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

-a <boolean> (--all-threads <boolean>)
Should we run all threads if the execution doesn't complete on one thread.

-d <none> (--dynamic-type <none>)
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> (--format <format>)
Specify a format to be used for display.

-g (--debug)
When specified, debug the JIT code by setting a breakpoint on the first instruction and forcing breakpoints to not be ignored (-i0) and no unwinding to happen on error (-u0).

-i <boolean> (--ignore-breakpoints <boolean>)
Ignore breakpoint hits while running expressions

-l <language> (--language <language>)
The language to use when parsing the expression.

-r (--repl)
Drop into swift REPL

-t <unsigned-integer> (--timeout <unsigned-integer>)
Timeout value (in microseconds) for running the expression.

-u <boolean> (--unwind-on-error <boolean>)
Clean up program state if the expression causes a crash, or raises a signal. Note, unlike gdb hitting a breakpoint is controlled by another option (-i).

```
 -v[<description-verbosity>] ( --description-verbosity=[<description-verbosity>] )
 How verbose should the output of this expression be, if the object description is asked for.
 Values: compact | full

 Timeouts:
 If the expression can be evaluated statically (without running code) then it will be.  Otherwise, by default the expression will run on the current thread with a short timeout:  currently .25 seconds.  If it doesn't return in that time, the evaluation will be interrupted  and resumed with all threads running.  You can use the -a option to disable retrying on all  threads.  You can use the -t option to set a shorter timeout.

 User defined variables:
 You can define your own variables for convenience or to be used in subsequent expressions. You define them the same way you would define variables in C.  If the first character of  your user defined variable is a $, then the variable's value will be available in future  expressions, otherwise it will just be available in the current expression.

 Continuing evaluation after a breakpoint:
 If the "-i false" option is used, and execution is interrupted by a breakpoint hit, once  you are done with your investigation, you can either remove the expression execution frames  from the stack with "thread return -x" or if you are still interested in the expression result  you can issue the "continue" command and the expression evaluation will complete and the expression result will be available using the "thread.completed-expression" key in the thread format.

 Examples:
 expr my_struct->a = my_array[3]
 expr -f bin -- (index * 8) + 5
 expr unsigned int $foo = 5
 expr char c[] = "foo"; c[0]

IMPORTANT NOTE:  Because this command takes 'raw' input, if you use any command options you must use ' -- ' between the end of the command options and the beginning of the raw input.
-----------------------------------------------------------------------------------------------------------------------------
 (lldb) help frame
 The following subcommands are supported:
 info          --      List information about the currently selected frame in the current thread.
 select        --      Select a frame by index from within the current thread and make it the current frame.
 variable      --      Show frame variables. All argument and local variables that are in scope will be shown when no arguments are given. If any arguments are specified, they can be names of argument, local, file static and file global variables. Children of aggregate variables can be specified such as 'var->child.x'.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.
-----------------------------------------------------------------------------------------------------------------------------
```

```
(lldb) help gdb-remote
Connect to a remote GDB server.  If no hostname is provided, localhost is assumed.  This command takes 'raw' input (no need to quote stuff).
Syntax: gdb-remote [<hostname>:]<portnum>
 ----------------------------------------------------------------------------------------------------------------------------
(lldb) help gui
Switch into the curses based GUI mode.
 Syntax: gui
 ----------------------------------------------------------------------------------------------------------------------------
(lldb) help kdp-remote
Connect to a remote KDP server.  udp port 41139 is the default port number.  This command takes 'raw' input (no need to quote stuff).
Syntax: kdp-remote <hostname>[:<portnum>]
----------------------------------------------------------------------------------------------------------------------------
(lldb) help log
The following subcommands are supported:
disable -- Disable one or more log channel categories.
enable  -- Enable logging for a single log channel.
list    -- List the log categories for one or more log channels.  If none specified, lists them all.
timers  -- Enable, disable, dump, and reset LLDB internal performance timers.

For more help on any particular subcommand, type 'help <command> <subcommand>'.
----------------------------------------------------------------------------------------------------------------------------
(lldb) help memory
The following subcommands are supported:
find  -- Find a value in the memory of the process being debugged.
read  -- Read from the memory of the process being debugged.
write -- Write to the memory of the process being debugged.
For more help on any particular subcommand, type 'help <command> <subcommand>'.
 ---------
(lldb) help memory find
Find a value in the memory of the process being debugged.
Syntax: memory find <cmd-options> <address> <value> [<value> [...]]

Command Options Usage:
memory find <address> <value> [<value> [...]]
memory find  [-e <expr>] [-s <name>] [-c <count>] [-o <offset>] <address> <value> [<value> [...]]

-c <count> ( --count <count> )
How many times to perform the search.
```

```
 -e <expr> ( --expression <expr> )
 Evaluate an expression to obtain a byte pattern.

 -o <offset> ( --dump-offset <offset> )
 When dumping memory for a match, an offset from the match location to start dumping from.

 -s <name> ( --string <name> )
 Use text to find a byte pattern.

This command takes options and free-form arguments. If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- ' between
the end of the command options and the beginning of the arguments.
  --------
 (lldb) help memory read
 Read from the memory of the process being debugged.
 Syntax: memory read <cmd-options> <address-expression> [<address-expression>]

 Command Options Usage:
 memory read   [-r] [-f <format>] [-c <count>] [-G <gdb-format>] [-s <byte-size>] [-l <number-per-line>]
               [-o <filename>] <address-expression> [<address-expression>]
memory read    [-br] [-f <format>] [-c <count>] [-s <byte-size>] [-o <filename>] <address-expression> [<address-expression>]
memory read    [-AFLORTr] -t <none> [-f <format>] [-c <count>] [-G <gdb-format>] [-o <filename>] [-d <none>] [-S <boolean>] [-D <count>]
               [-P <count>] [-Y[<count>]] [-V <boolean>] <address-expression> [<address-expression>]

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

 -D <count> ( --depth <count> )
 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> ( --gdb-format <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 <count> (--ptr-depth <count>)
The number of pointers to be traversed when dumping values (default is zero).

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

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

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

-V <boolean> (--validate <boolean>)
Show results of type validators.

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

-b (--binary)
If true, memory will be saved as binary. If false, the memory is saved save as an ASCII dump that uses the format, size, count and number per line settings.

-c <count> (--count <count>)
The number of total items to display.

-d <none> (--dynamic-type <none>)
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> (--format <format>)
Specify a format to be used for display.

-l <number-per-line> (--num-per-line <number-per-line>)
The number of items per line to display.

-o <filename> (--outfile <filename>)
Specify a path for capturing command output.

-r (--force)
Necessary if reading over target.max-memory-read-size bytes.

```
 -s <byte-size> ( --size <byte-size> )
 The size in bytes to use when displaying with the selected format.

 -t <none> ( --type <none> )
 The name of a type to view memory as.
  --append-outfile
 Append to the the file specified with '--outfile <path>'.
 This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- '
between the end of the command options and the beginning of the arguments.
----------
 (lldb) help memory write
 Write to the memory of the process being debugged.
 Syntax: memory write  <cmd-options> <address> <value> [<value> [...]]
  Command Options Usage:
 memory write   [-f <format>] [-s <byte-size>] <address> <value> [<value> [...]]
 memory write   -i <filename> [-s <byte-size>] [-o <offset>] <address> <value> [<value> [...]]

 -f <format> ( --format <format> )
 Specify a format to be used for display.
  -i <filename> ( --infile <filename> )
 Write memory using the contents of a file.
  -o <offset> ( --offset <offset> )
 Start writing bytes from an offset within the input file.
  -s <byte-size> ( --size <byte-size> )
 The size in bytes to use when displaying with the selected format.

This command takes options and free-form arguments. If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the  arguments.
 ----------------------------------------------------------------------------------------------------------------------------------
 (lldb) help platform
 The following subcommands are supported:
 connect        -- Connect a platform by name to be the currently selected platform.
 disconnect     -- Disconnect a platform by name to be the currently selected platform.
 list           -- List all platforms that are available.
 process        -- A set of commands to query, launch and attach to platform processes
 select         -- Create a platform if needed and select it as the current platform.
 settings       -- Set settings for the current target's platform, or for a platform by name.
 shell          -- Run a shell command on a the selected platform.  This command takes 'raw' input (no need to quote stuff).
 status         -- Display status for the currently selected platform.
 target-install -- Install a target (bundle or executable file) to the remote end.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.
```

```
 -----------------


 (lldb) help platform connect
 Connect a platform by name to be the currently selected platform.
  Syntax: platform connect <connect-url>

 Command Options Usage:
platform connect [-irs] [-R <cmd-name>] [-P <cmd-name>] [-S <cmd-name>] [-c <path>]

 -P <cmd-name> ( --rsync-prefix <cmd-name> )
 Platform-specific rsync prefix put before the remote path.

 -R <cmd-name> ( --rsync-opts <cmd-name> )
 Platform-specific options required for rsync to work.

 -S <cmd-name> ( --ssh-opts <cmd-name> )
 Platform-specific options required for SSH to work.

 -c <path> ( --local-cache-dir <path> )
 Path in which to store local copies of files.

 -i ( --ignore-remote-hostname )
 Do not automatically fill in the remote hostname when composing the rsync command.

-r ( --rsync )
 Enable rsync.

 -s ( --ssh )
 Enable SSH.
 ------------------------
 (lldb) help platform disconnect
 Disconnect a platform by name to be the currently selected platform.
  Syntax: platform disconnect
 ------------------------
 (lldb) help platform list
 List all platforms that are available.
 Syntax: platform list

 ----------------------
```

```
(lldb) help platform process
The following subcommands are supported:
attach -- Attach to a process.
info   -- Get detailed information for one or more process by process ID.
launch -- Launch a new process on a remote platform.
list   -- List processes on a remote platform by name, pid, or many other matching attributes.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.

------------------------
(lldb) help platform select
Create a platform if needed and select it as the current platform.

Syntax: platform select <platform-name>

Command Options Usage:
platform select  [-v <none>] [-b <none>] [-S <filename>]

-S <filename> ( --sysroot <filename> )
Specify the SDK root directory that contains a root of all remote system files.

-b <none> ( --build <none> )
Specify the initial SDK build number.

-v <none> ( --version <none> )
Specify the initial SDK version to use prior to connecting.

-------------------------
(lldb) help platform settings
Set settings for the current target's platform, or for a platform by name.

Syntax: platform settings

Command Options Usage:
platform settings [-RWXdertwx] [-w <path>] [-v <perms-numeric>] [-s <perms=string>]

-R ( --group-read )
Allow group to read.
```

```
-W ( --group-write )
Allow group to write.

-X ( --group-exec )
Allow group to execute.

-d ( --world-read )
Allow world to read.
-e ( --world-exec )
Allow world to execute.
-r ( --user-read )
Allow user to read.

-s <perms=string> ( --permissions-string <perms=string> )
Give out the string value for permissions (e.g. rwxr-xr--).

-t ( --world-write )
Allow world to write.

-v <perms-numeric> ( --permissions-value <perms-numeric> )
Give out the numeric value for permissions (e.g. 757)

-w <path> ( --working-dir <path> )
The working directory for the platform.

-w ( --user-write )
Allow user to write.

-x ( --user-exec )
Allow user to execute.
--------------------------------
(lldb) help platform shell
Run a shell command on a the selected platform.  This command takes 'raw' input (no need to quote stuff).
 Syntax: platform shell <shell-command>
 Command Options Usage:
platform shell [-t <value>]
 -t <value> ( --timeout <value> )
Seconds to wait for the remote host to finish running the command.

IMPORTANT NOTE:  Because this command takes 'raw' input, if you use any command options you must use ' -- ' between the end of the command options and the
beginning of the raw input
```

```
(lldb) help platform status
Display status for the currently selected platform.
Syntax: platform status
 ---------------
(lldb) help platform target-install
Install a target (bundle or executable file) to the remote end.
 Syntax: platform target-install <local-thing> <remote-sandbox>

-------------------------------------------------------------------------------------------------------------------------------

(lldb) help plugin
The following subcommands are supported:
 load -- Import a dylib that implements an LLDB plugin.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.

-------------------------------------------------------------------------------------------------------------------------------

(lldb) help process
The following subcommands are supported:
attach    -- Attach to a process.
connect   -- Connect to a remote debug service.
continue  -- Continue execution of all threads in the current process.
detach    -- Detach from the current process being debugged.
handle    -- Show or update what the process and debugger should do with various signals received from the OS.
interrupt -- Interrupt the current process being debugged.
kill      -- Terminate the current process being debugged.
launch    -- Launch the executable in the debugger.
load      -- Load a shared library into the current process.
plugin    -- Send a custom command to the current process plug-in.
save-core -- Save the current process as a core file using an appropriate file type.
signal    -- Send a UNIX signal to the current process being debugged.
status    -- Show the current status and location of executing process.
unload    -- Unload a shared library from the current process using the index returned by a previous call to "process load".

For more help on any particular subcommand, type 'help <command> <subcommand>'.
```

```
(lldb) help process attach
Attach to a process.
 Syntax: process attach <cmd-options>

Command Options Usage:
process attach [-c] [-P <plugin>] [-p <pid>]
process attach [-ciw] [-P <plugin>] [-n <process-name>]

-P <plugin> ( --plugin <plugin> )
Name of the process plugin you want to use.

-c ( --continue )
Immediately continue the process once attached.

-i ( --include-existing )
Include existing processes when doing attach -w.

-n <process-name> ( --name <process-name> )
The name of the process to attach to.

-p <pid> ( --pid <pid> )
The process ID of an existing process to attach to.

-w ( --waitfor )
Wait for the process with <process-name> to launch.
---------------
(lldb) help process connect
Connect to a remote debug service.
 Syntax: process connect <remote-url>

Command Options Usage:
process connect [-p <plugin>]

-p <plugin> ( --plugin <plugin> )
Name of the process plugin you want to use.

-------------
```

```
 (lldb) help process continue
Continue execution of all threads in the current process.

Syntax: process continue

Command Options Usage:
process continue [-i <unsigned-integer>]

-i <unsigned-integer> ( --ignore-count <unsigned-integer> )
Ignore <N> crossings of the breakpoint (if it exists) for the currently selected thread.
-------------
(lldb) help process detach
Detach from the current process being debugged.

Syntax: process detach

Command Options Usage:
process detach [-s <boolean>]
 -s <boolean> ( --keep-stopped <boolean> )
Whether or not the process should be kept stopped on detach (if possible).
 ---------------
(lldb) help process handle
Show or update what the process and debugger should do with various signals received from the OS.
Syntax: process handle <cmd-options> [<unix-signal> [<unix-signal> [...]]]

Command Options Usage:
process handle [-s <boolean>] [-n <boolean>] [-p <boolean>] [<unix-signal> [<unix-signal> [...]]]

-n <boolean> ( --notify <boolean> )
Whether or not the debugger should notify the user if the signal is received.

-p <boolean> ( --pass <boolean> )
Whether or not the signal should be passed to the process.

-s <boolean> ( --stop <boolean> )
Whether or not the process should be stopped if the signal is received.
If no signals are specified, update them all.  If no update option is specified, list the current values.
This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the  arguments.
------------------------
```

```
(lldb) help process interrupt
Interrupt the current process being debugged.
 Syntax: process interrupt
-----------------------
 (lldb) help process kill
Terminate the current process being debugged.
Syntax: process kill
-----------------------
 (lldb) help process launch
Launch the executable in the debugger.

Syntax: process launch <cmd-options> [<run-args>]

Command Options Usage:
process launch [-s] [-A <boolean>] [-p <plugin>] [-w <directory>] [-a <arch>] [-v <none>] [-c[<filename>]] [-i <filename>] [-o <filename>] [-e <filename>]
[<run-args>]
process launch [-st] [-A <boolean>] [-p <plugin>] [-w <directory>] [-a <arch>] [-v <none>] [-c[<filename>]] [<run-args>]
process launch [-ns] [-A <boolean>] [-p <plugin>] [-w <directory>] [-a <arch>] [-v <none>] [-c[<filename>]] [<run-args>]

-A <boolean> ( --disable-aslr <boolean> )
Set whether to disable address space layout randomization when launching a process.

-a <arch> ( --arch <arch> )
Set the architecture for the process to launch when ambiguous.

-c[<filename>] ( --shell=[<filename>] )
Run the process in a shell (not supported on all platforms).

-e <filename> ( --stderr <filename> )
Redirect stderr for the process to <filename>.

-i <filename> ( --stdin <filename> )
Redirect stdin for the process to <filename>.

-n ( --no-stdio )
Do not set up for terminal I/O to go to running process.

-o <filename> ( --stdout <filename> )
Redirect stdout for the process to <filename>.

-p <plugin> ( --plugin <plugin> )
```

```
Name of the process plugin you want to use.

-s ( --stop-at-entry )
Stop at the entry point of the program when launching a process.

-t ( --tty )
Start the process in a terminal (not supported on all platforms).

-v <none> ( --environment <none> )
Specify an environment variable name/value string (--environment NAME=VALUE). Can be specified multiple times for subsequent environment entries.

-w <directory> ( --working-dir <directory> )
Set the current working directory to <path> when running the inferior.
This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- '
between the end of the command options and the beginning of the arguments.
------------------
(lldb) help process load
Load a shared library into the current process.
Syntax: process load <filename> [<filename> ...]
------------
(lldb) help process plugin
The following subcommands are supported:
packet -- Commands that deal with GDB remote packets.
For more help on any particular subcommand, type 'help <command> <subcommand>'.
-----------------
 (lldb) help process save-core
Save the current process as a core file using an appropriate file type.
Syntax: process save-core FILE
----------------
(lldb) help process signal
Send a UNIX signal to the current process being debugged.
Syntax: process signal <unix-signal>
 ------------------
(lldb) help process status
Show the current status and location of executing process.
Syntax: process status

--------------------
```

```
(lldb) help process unload
Unload a shared library from the current process using the index returned by a previous call to "process load".
 Syntax: process unload <index>
 ----------------------------------------------------------------------------------------------------------------------
(lldb) help quit
Quit out of the LLDB debugger.
 Syntax: quit
----------------------------------------------------------------------------------------------------------------------
 (lldb) help register
The following subcommands are supported:
read  -- Dump the contents of one or more register values from the current frame.  If no register is specified, dumps them all.
write -- Modify a single register value.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.
---------
(lldb) help register read
Dump the contents of one or more register values from the current frame.  If no register is specified, dumps them all.
 Syntax: register read <cmd-options> [<register-name> [<register-name> [...]]]

Command Options Usage:
register read  [-A] [-f <format>] [-G <gdb-format>] [-s <index>] [<register-name> [<register-name> [...]]]
register read  [-Aa] [-f <format>] [-G <gdb-format>] [<register-name> [<register-name> [...]]]

-A ( --alternate )
Display register names using the alternate register name if there is one.

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

-a ( --all )
Show all register sets.

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

-s <index> ( --set <index> )
Specify which register sets to dump by index.
This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- '
between the end of the command options and the beginning of the arguments.
```

```
(lldb) help register write
 Modify a single register value.
 Syntax: register write <register-name> <value>
-------------------------------------------------------------------------------------------------------------------------
 (lldb) help script
 Pass an expression to the script interpreter for evaluation and return the results. Drop into the interactive interpreter if no expression is given.  This
command takes 'raw' input (no need to quote stuff).
 Syntax: script [<script-expression-for-evaluation>]
 -------------------------------------------------------------------------------------------------------------------------
  (lldb) help settings
 The following subcommands are supported:
 append        --      Append a new value to the end of an internal debugger settings array, dictionary or string variable.  This command takes 'raw' input
                       (no need to quote stuff).
 clear         --      Erase all the contents of an internal debugger settings variables; this is only valid for variables with clearable types, i.e. strings,
                       arrays or dictionaries.
 insert-after  --      Insert value(s) into an internal debugger settings array variable, immediately after the specified element.  This command takes 'raw'
                       input (no need to quote stuff).
 insert-before --      Insert value(s) into an internal debugger settings array variable, immediately before the specified element.  This command takes 'raw'
                       input (no need to quote stuff).
 list          --      List and describe all the internal debugger settings variables that are available to the user to 'set' or 'show', or describe a
                       particular variable or set of variables (by specifying the variable name or a common prefix).
 remove        --      Remove the specified element from an array or dictionary settings variable.  This command takes 'raw' input (no need to quote stuff).
 replace       --      Replace the specified element from an internal debugger settings array or dictionary variable with the specified new value.  This command
                       takes 'raw' input (no need to quote stuff).
 set           --      Set or change the value of a single debugger setting variable.  This command takes 'raw' input (no need to quote stuff).
 show          --      Show the specified internal debugger setting variable and its value, or show all the currently set variables and their values, if nothing
                       is specified.
  For more help on any particular subcommand, type 'help <command> <subcommand>'.
  ----------
  (lldb) help settings append
 Append a new value to the end of an internal debugger settings array, dictionary or string variable.  This command takes 'raw' input (no need to quote stuff).
  Syntax: settings append <setting-variable-name> <value>
  -----------
 (lldb) help settings clear
 Erase all the contents of an internal debugger settings variables; this is only valid for variables with clearable types, i.e. strings, arrays or dictionaries.
  Syntax: settings clear <setting-variable-name>
  -------------
 (lldb) help settings insert-after
 Insert value(s) into an internal debugger settings array variable, immediately after the specified element.  This command takes 'raw' input (no need to quote
stuff).
  Syntax: settings insert-after <setting-variable-name> <setting-index> <value>
  --------
```

```
(lldb) help settings insert-before
Insert value(s) into an internal debugger settings array variable, immediately before the specified element.  This command takes 'raw' input
(no need to quote stuff).
Syntax: settings insert-before <setting-variable-name> <setting-index> <value>
----------
(lldb) help settings list
List and describe all the internal debugger settings variables that are available to the user to 'set' or 'show', or describe a particular variable or set of
variables (by specifying the variable name or a common prefix).
Syntax: settings list [<setting-variable-name | setting-prefix>]
-----------
(lldb) help settings remove
Remove the specified element from an array or dictionary settings variable.  This command takes 'raw' input (no need to quote stuff).
 Syntax: settings remove <setting-variable-name> <setting-index | setting-key>

--------------
(lldb) help settings replace
Replace the specified element from an internal debugger settings array or dictionary variable with the specified new value.  This command takes 'raw' input (no
need to quote stuff).
Syntax: settings replace  <setting-variable-name> <setting-index | setting-key> <value>
----------------
(lldb) help settings set
Set or change the value of a single debugger setting variable. This command takes 'raw' input (no need to quote stuff).

Syntax: settings set <cmd-options> -- <setting-variable-name> <value>

Command Options Usage:
settings set  -- <setting-variable-name> <value>
settings set  [-g] -- <setting-variable-name> <value>
settings set <setting-variable-name> <value>

-g ( --global )
Apply the new value to the global default value.

When setting a dictionary or array variable, you can set multiple entries
at once by giving the values to the set command.  For example:

(lldb) settings set target.run-args value1 value2 value3
(lldb) settings set target.env-vars MYPATH=~/.:/usr/bin  SOME_ENV_VAR=12345
```

```
(lldb) settings show target.run-args
[0]: 'value1'
[1]: 'value2'
[3]: 'value3'
(lldb) settings show target.env-vars
'MYPATH=~/.:/usr/bin'
'SOME_ENV_VAR=12345'

Warning:  The 'set' command re-sets the entire array or dictionary.  If you  just want to add, remove or update individual values
(or add something to the end), use one of the other settings sub-commands: append, replace, insert-before or insert-after.

This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.
------------
(lldb) help settings show
Show the specified internal debugger setting variable and its value, or show all the currently set variables and their values, if nothing is specified.
Syntax: settings show  [<setting-variable-name>]

-----------------------------------------------------------------------------------------------------------------------------
(lldb) help source
The following subcommands are supported:
 list -- Display source code (as specified) based on the current executable's debug info.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.
--------
(lldb) help source list
Display source code (as specified) based on the current executable's debug info.
 Syntax: source list <cmd-options>

Command Options Usage:
source list  [-b] [-c <count>] [-s <shlib-name>] [-f <filename>] [-l <linenum>]
source list [-b] [-c <count>] [-s <shlib-name>] [-n <symbol>]
source list [-b] [-c <count>] [-a <address-expression>]
source list [-br] [-c <count>]

-a <address-expression> ( --address <address-expression> )
Lookup the address and display the source information for the corresponding file and line.

-b ( --show-breakpoints )
Show the line table locations from the debug information that indicate valid places to set source level breakpoints.
```

```
-c <count> ( --count <count> )
The number of source lines to display.

-f <filename> ( --file <filename> )
The file from which to display source.

-l <linenum> ( --line <linenum> )
The line number at which to start the display source.

-n <symbol> ( --name <symbol> )
The name of a function whose source to display.
 -r ( --reverse )
Reverse the listing to look backwards from the last displayed block of source.

-s <shlib-name> ( --shlib <shlib-name> )
Look up the source file in the given shared library.
-------------------------------------------------------------------------------------------------------------------------
(lldb) help target
The following subcommands are supported:

create    -- Create a target using the argument as the main executable.
delete    -- Delete one or more targets by target index.
list      -- List all current targets in the current debug session.
modules   -- A set of commands for accessing information for one or more target modules.
select    -- Select a target as the current target by target index.
stop-hook -- A set of commands for operating on debugger target stop-hooks.
symbols   -- A set of commands for adding and managing debug symbol files.
variable  -- Read global variable(s) prior to, or while running your binary.

For more help on any particular subcommand, type 'help <command> <subcommand>'.

------------
(lldb) help target create
Create a target using the argument as the main executable.
 Syntax: target create <cmd-options> <filename>

Command Options Usage:
target create        [-d] [-a <arch>] [-p <platform-name>] [-v <none>] [-b <none>] [-S <filename>] [-c <filename>] [-P <path>] [-s <filename>]
                     [-r <filename>] <filename>
```

```
 -P <path> ( --platform-path <path> )
 Path to the remote file to use for this target.

 -S <filename> ( --sysroot <filename> )
 Specify the SDK root directory that contains a root of all remote system files.

 -a <arch> ( --arch <arch> )
 Specify the architecture for the target.

 -b <none> ( --build <none> )
 Specify the initial SDK build number.

 -c <filename> ( --core <filename> )
 Fullpath to a core file to use for this target.

 -d ( --no-dependents )
 Don't load dependent files when creating the target, just add the specified executable.

 -p <platform-name> ( --platform <platform-name> )
 Specify name of the platform to use for this target, creating the platform if necessary.

 -r <filename> ( --remote-file <filename> )
 Fullpath to the file on the remote host if debugging remotely.

 -s <filename> ( --symfile <filename> )
 Fullpath to a stand alone debug symbols file for when debug symbols are not in the executable.

 -v <none> ( --version <none> )
 Specify the initial SDK version to use prior to connecting.

 This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- '
between the end of the command options and the beginning of the arguments.
----------------------
 (lldb) help target delete
 Delete one or more targets by target index.
 Syntax: target delete <cmd-options>

 Command Options Usage:
 target delete  [-c <boolean>]
  -c <boolean> ( --clean <boolean> )
 Perform extra cleanup to minimize memory consumption after deleting the target.
```

```
(lldb) help target list
List all current targets in the current debug session.
 Syntax: target list
 ---------
(lldb) help target modules
The following subcommands are supported:

add          -- Add a new module to the current target's modules.
dump         -- A set of commands for dumping information about one or more target modules.
list         -- List current executable and dependent shared library images.
load         -- Set the load addresses for one or more sections in a target module.
lookup       -- Look up information within executable and dependent shared library images.
search-paths -- A set of commands for operating on debugger target image search paths.
show-unwind  -- Show synthesized unwind instructions for a function.
For more help on any particular subcommand, type 'help <command> <subcommand>'.
----------
(lldb) help target modules list
List current executable and dependent shared library images.
 Syntax: target modules list [<cmd-options>]

Command Options Usage:
target modules list    [-ghou] [-a <address-expression>] [-A[<width>]] [-t[<width>]] [-f[<width>]] [-d[<width>]] [-b[<width>]] [-s[<width>]]
                       [-S[<width>]] [-m[<width>]] [-r[<width>]] [-p[<none>]]

-A[<width>] ( --arch=[<width>] )
Display the architecture when listing images.

-S[<width>] ( --symfile-unique=[<width>] )
Display the symbol file with optional width only if it is different from the executable object file.

-a <address-expression> ( --address <address-expression> )
Display the image at this address.

-b[<width>] ( --basename=[<width>] )
Display the basename with optional width for the image object file.

-d[<width>] ( --directory=[<width>] )
Display the directory with optional width for the image object file.
```

```
-f[<width>] ( --fullpath=[<width>] )
Display the fullpath to the image object file.

-g ( --global )
Display the modules from the global module list, not just the current target.

-h ( --header )
Display the image header address as a load address if debugging, a file address otherwise.

-m[<width>] ( --mod-time=[<width>] )
Display the modification time with optional width of the module.

-o ( --offset )
Display the image header address offset from the header file address (the slide amount).

-p[<none>] ( --pointer=[<none>] )
Display the module pointer.

-r[<width>] ( --ref-count=[<width>] )
Display the reference count if the module is still in the shared module cache.

-s[<width>] ( --symfile=[<width>] )
Display the fullpath to the image symbol file with optional width.

-t[<width>] ( --triple=[<width>] )
Display the triple when listing images.

-u ( --uuid )
Display the UUID when listing images.

--------
(lldb) help target modules load
Set the load addresses for one or more sections in a target module.

Syntax: target modules load [--file <module> --uuid <uuid>] <sect-name> <address> [<sect-name> <address> ....]

Command Options Usage:
target modules load [-u <none>] [-f <name>] [-s <offset>] [<filename> [<filename> [...]]]

-f <name> ( --file <name> )
Fullpath or basename for module to load.
```

```
 -s <offset> ( --slide <offset> )
 Set the load address for all sections to be the virtual address in the file plus the offset.

 -u <none> ( --uuid <none> )
 A module UUID value.
 This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use  ' -- ' between the end of the command options and the beginning of the arguments.
 ----------
 (lldb) help target modules lookup
 Look up information within executable and dependent shared library images.

 Syntax: target modules lookup <cmd-options> [<filename> [<filename> [...]]]

 Command Options Usage:
target modules lookup  [-Av] -a <address-expression> [-o <offset>] [<filename> [<filename> [...]]]
target modules lookup  [-Arv] -s <symbol> [<filename> [<filename> [...]]]
target modules lookup  [-Aiv] -f <filename> [-l <linenum>] [<filename> [<filename> [...]]]
target modules lookup  [Airv] -F <function-name> [<filename> [<filename> [...]]]
target modules lookup  [-Airv] -n <function-or-symbol> [<filename> [<filename> [...]]]
target modules lookup  [-Av] -t <name> [<filename> [<filename> [...]]]

 -A ( --all )
 Print all matches, not just the best match, if a best match is available.

 -F <function-name> ( --function <function-name> )
 Lookup a function by name in the debug symbols in one or more target modules.

 -a <address-expression> ( --address <address-expression> )
 Lookup an address in one or more target modules.

 -f <filename> ( --file <filename> )
 Lookup a file by fullpath or basename in one or more target modules.

 -i ( --no-inlines )
 Ignore inline entries (must be used in conjunction with --file or --function).

 -l <linenum> ( --line <linenum> )
 Lookup a line number in a file (must be used in conjunction with --file).
```

```
 -n <function-or-symbol> ( --name <function-or-symbol> )
 Lookup a function or symbol by name in one or more target modules.

 -o <offset> ( --offset <offset> )
 When looking up an address subtract <offset> from any addresses before doing the lookup.

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

 -s <symbol> ( --symbol <symbol> )
 Lookup a symbol by name in the symbol tables in one or more target modules.

 -t <name> ( --type <name> )
 Lookup a type by name in the debug symbols in one or more target modules.

 -v ( --verbose )
 Enable verbose lookup information.

 This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.
 ------------
 (lldb) help target modules search-paths
 The following subcommands are supported:

 add    -- Add new image search paths substitution pairs to the current target.
 clear  -- Clear all current image search path substitution pairs from the current target.
 insert -- Insert a new image search path substitution pair into the current target at the specified index.
 list   -- List all current image search path substitution pairs in the current target.
 query  -- Transform a path using the first applicable image search path.

 For more help on any particular subcommand, type 'help <command> <subcommand>'.

 ---------------
 (lldb) help target modules show-unwind
 Show synthesized unwind instructions for a function.
  Syntax: target modules show-unwind <cmd-options>

 Command Options Usage:
target modules show-unwind    [-n <function-name>]
target modules show-unwind    [-a <address-expression>]
```

```
-a <address-expression> ( --address <address-expression> )
Show unwind instructions for a function or symbol containing an address

-n <function-name> ( --name <function-name> )
Show unwind instructions for a function or symbol name.
```

```
(lldb) help thread
The following subcommands are supported:

backtrace      --    Show the stack for one or more threads.  If no threads are specified, show the currently selected thread.
                     Use the thread-index "all" to see all threads.
continue       --    Continue execution of one or more threads in an active process.
info           --    Show an extended summary of information about thread(s) in a process.
jump           --    Sets the program counter to a new address.
list           --    Show a summary of all current threads in a process.
plan           --    A set of subcommands for accessing the thread plans controlling execution control on one or more threads.
return         --    Return from the currently selected frame, short-circuiting execution of the frames below it, with an optional return value,
                     or with the -x option from the innermost function evaluation.  This command takes 'raw' input (no need to quote stuff).
select         --    Select a thread as the currently active thread.
step-in        --    Source level single step in specified thread (current thread, if none specified).
step-inst      --    Single step one instruction in specified thread (current thread, if none specified).
step-inst-over --    Single step one instruction in specified thread (current thread, if none specified), stepping over calls.
step-out       --    Finish executing the function of the currently selected frame and return to its call site in specified thread
                     (current thread, if none specified).
step-over      --    Source level single step in specified thread (current thread, if none specified), stepping over calls.
step-scripted  --    Step as instructed by the script class passed in the -C option.
until          --    Run the current or specified thread until it reaches a given line number or leaves the current function.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.
------------------
```

```
(lldb) help thread backtrace
Show the stack for one or more threads.  If no threads are specified, show the currently selected thread.  Use the thread-index "all" to see all threads.
Syntax: thread backtrace <cmd-options>

Command Options Usage:
thread backtrace [-c <count>] [-s <frame-index>] [-e <boolean>]
 -c <count> ( --count <count> )
How many frames to display (-1 for all)

-e <boolean> ( --extended <boolean> )
Show the extended backtrace, if available

-s <frame-index> ( --start <frame-index> )
Frame in which to start the backtrace

(lldb) help thread continue
Continue execution of one or more threads in an active process.
 Syntax: thread continue <thread-index> [<thread-index> [...]]

(lldb) help thread info
Show an extended summary of information about thread(s) in a process.
 Syntax: thread info

Command Options Usage:
thread info [-js]

-j ( --json )
Display the thread info in JSON format.

-s ( --stop-info )
Display the extended stop info in JSON format.
```

```
(lldb) help thread jump
Sets the program counter to a new address.
 Syntax: thread jump

Command Options Usage:
       thread jump    [-r] -l <linenum> [-f <filename>]
       thread jump    [-r] -b <offset>
       thread jump    [-r] -a <address-expression>

-a <address-expression> ( --address <address-expression> )
Jumps to a specific address.

-b <offset> ( --by <offset> )
Jumps by a relative line offset from the current line.

-f <filename> ( --file <filename> )
Specifies the source file to jump to.

-l <linenum> ( --line <linenum> )
Specifies the line number to jump to.

-r ( --force )
Allows the PC to leave the current function.

(lldb) help thread list
Show a summary of all current threads in a process.
 Syntax: thread list

(lldb) help thread plan
The following subcommands are supported:
discard       --      Discards thread plans up to and including the plan passed as the command argument.Only user visible plans can be discarded, use the index
                      from "thread plan list" without the "-i" argument.
list          --      Show thread plans for one or more threads.  If no threads are specified, show the currently selected thread.  Use the thread-index "all"
                      to see all threads.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.
```

```
 (lldb) help thread return
 Return from the currently selected frame, short-circuiting execution of the frames below it, with an optional return value, or with the -x option from the
innermost function evaluation.  This command takes 'raw' input (no need to quote stuff).
  Syntax: thread return

 Command Options Usage:
 thread return [-x] -- [<expr>]
 thread return [<expr>]

 -x ( --from-expression )
 Return from the innermost expression evaluation.
 IMPORTANT NOTE:  Because this command takes 'raw' input, if you use any command options you must use
' -- ' between the end of the command options and the beginning of the raw input.
-----------------------
help thread select
 Select a thread as the currently active thread.
  Syntax: thread select <thread-index>

 (lldb) help thread step-in
 Source level single step in specified thread (current thread, if none specified).
  Syntax: thread step-in <cmd-options> [<thread-id>]

 Command Options Usage:
thread step-in [-a <boolean>] [-A <boolean>] [-c <count>] [-m <run-mode>] [-r <regular-expression>] [-t <function-name>] [<thread-id>]
thread step-in [-C <python-class>] [<thread-id>]

 -A <boolean> ( --step-out-avoids-no-debug <boolean> )
 A boolean value, if true stepping out of functions will continue to step out till it hits a function with debug information.

 -C <python-class> ( --python-class <python-class> )
 The name of the class that will manage this step - only supported for Scripted Step.

 -a <boolean> ( --step-in-avoids-no-debug <boolean> )
 A boolean value that sets whether stepping into functions will step over functions with no debug information.

 -c <count> ( --count <count> )
 How many times to perform the stepping operation - currently only supported for step-inst and next-inst.

 -m <run-mode> ( --run-mode <run-mode> )
 Determine how to run other threads while stepping the current thread.
 Values: this-thread | all-threads | while-stepping
```

```
 -r <regular-expression> ( --step-over-regexp <regular-expression> )
 A regular expression that defines function names to not to stop at when stepping in.

 -t <function-name> ( --step-in-target <function-name> )
 The name of the directly called function step in should stop at when stepping into.

This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- ' between
the end of the command options and the beginning of the arguments.

(lldb) help thread step-inst
 Single step one instruction in specified thread (current thread, if none specified).
  Syntax: thread step-inst <cmd-options> [<thread-id>]

 Command Options Usage:
 thread step-inst [-a <boolean>] [-A <boolean>] [-c <count>] [-m <run-mode>] [-r <regular-expression>] [-t <function-name>] [<thread-id>]
 thread step-inst [-C <python-class>] [<thread-id>]

 -A <boolean> ( --step-out-avoids-no-debug <boolean> )
 A boolean value, if true stepping out of functions will continue to step out till it hits a function with debug information.

 -C <python-class> ( --python-class <python-class> )
 The name of the class that will manage this step - only supported for Scripted Step.

 -a <boolean> ( --step-in-avoids-no-debug <boolean> )
 A boolean value that sets whether stepping into functions will step over functions with no debug information.

 -c <count> ( --count <count> )
 How many times to perform the stepping operation - currently only supported for step-inst and next-inst.

 -m <run-mode> ( --run-mode <run-mode> )
 Determine how to run other threads while stepping the current thread.
 Values: this-thread | all-threads | while-stepping

 -r <regular-expression> ( --step-over-regexp <regular-expression> )
 A regular expression that defines function names to not to stop at when stepping in.

 -t <function-name> ( --step-in-target <function-name> )
 The name of the directly called function step in should stop at when stepping into.
  This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.
```

```
(lldb) help thread step-inst-over
Single step one instruction in specified thread (current thread, if none specified), stepping over calls.
 Syntax: thread step-inst-over <cmd-options> [<thread-id>]

Command Options Usage:
thread step-inst-over [-a <boolean>] [-A <boolean>] [-c <count>] [-m <run-mode>] [-r <regular-expression>] [-t <function-name>] [<thread-id>]
thread step-inst-over [-C <python-class>] [<thread-id>]

-A <boolean> ( --step-out-avoids-no-debug <boolean> )
A boolean value, if true stepping out of functions will continue to step out till it hits a function with debug information.

-C <python-class> ( --python-class <python-class> )
The name of the class that will manage this step - only supported for Scripted Step.

-a <boolean> ( --step-in-avoids-no-debug <boolean> )
A boolean value that sets whether stepping into functions will step over functions with no debug information.

-c <count> ( --count <count> )
How many times to perform the stepping operation - currently only supported for step-inst and next-inst.

-m <run-mode> ( --run-mode <run-mode> )
Determine how to run other threads while stepping the current thread.
Values: this-thread | all-threads | while-stepping

-r <regular-expression> ( --step-over-regexp <regular-expression> )
A regular expression that defines function names to not to stop at when stepping in.

-t <function-name> ( --step-in-target <function-name> )
The name of the directly called function step in should stop at when stepping into.

This command takes options and free-form arguments. If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.

(lldb) help thread step-out
Finish executing the function of the currently selected frame and return to its call site in specified thread (current thread, if none specified).
Syntax: thread step-out <cmd-options> [<thread-id>]

Command Options Usage:
thread step-out  [-a <boolean>] [-A <boolean>] [-c <count>] [-m <run-mode>] [-r <regular-expression>] [-t <function-name>] [<thread-id>]
thread step-out [-C <python-class>] [<thread-id>]
```

```
 -A <boolean> ( --step-out-avoids-no-debug <boolean> )
 A boolean value, if true stepping out of functions will continue to step out till it hits a function with debug information.

 -C <python-class> ( --python-class <python-class> )
 The name of the class that will manage this step - only supported for Scripted Step.

 -a <boolean> ( --step-in-avoids-no-debug <boolean> )
 A boolean value that sets whether stepping into functions will step over functions with no debug information.

 -c <count> ( --count <count> )
 How many times to perform the stepping operation - currently only supported for step-inst and next-inst.

 -m <run-mode> ( --run-mode <run-mode> )
 Determine how to run other threads while stepping the current thread.
 Values: this-thread | all-threads | while-stepping

 -r <regular-expression> ( --step-over-regexp <regular-expression> )
 A regular expression that defines function names to not to stop at when stepping in.

 -t <function-name> ( --step-in-target <function-name> )
 The name of the directly called function step in should stop at when stepping into.

 This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.

 (lldb) help thread step-over
 Source level single step in specified thread (current thread, if none specified), stepping over calls.
  Syntax: thread step-over <cmd-options> [<thread-id>]

 Command Options Usage:
 thread step-over [-a <boolean>] [-A <boolean>] [-c <count>] [-m <run-mode>] [-r <regular-expression>] [-t <function-name>] [<thread-id>]
thread step-over [-C <python-class>] [<thread-id>]

 -A <boolean> ( --step-out-avoids-no-debug <boolean> )
 A boolean value, if true stepping out of functions will continue to step out till it hits a function with debug information.

 -C <python-class> ( --python-class <python-class> )
 The name of the class that will manage this step - only supported for Scripted Step.

 -a <boolean> ( --step-in-avoids-no-debug <boolean> )
 A boolean value that sets whether stepping into functions will step over functions with no debug information.
```

```
 -c <count> ( --count <count> )
 How many times to perform the stepping operation - currently only supported for step-inst and next-inst.

 -m <run-mode> ( --run-mode <run-mode> )
 Determine how to run other threads while stepping the current thread.
 Values: this-thread | all-threads | while-stepping

 -r <regular-expression> ( --step-over-regexp <regular-expression> )
 A regular expression that defines function names to not to stop at when stepping in.

 -t <function-name> ( --step-in-target <function-name> )
 The name of the directly called function step in should stop at when stepping into.

 This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.

 (lldb) help thread step-scripted
 Step as instructed by the script class passed in the -C option.

 Syntax: thread step-scripted <cmd-options> [<thread-id>]

 Command Options Usage:
thread step-scripted [-a <boolean>] [-A <boolean>] [-c <count>] [-m <run-mode>] [-r <regular-expression>] [-t <function-name>] [<thread-id>]
thread step-scripted [-C <python-class>] [<thread-id>]

 -A <boolean> ( --step-out-avoids-no-debug <boolean> )
 A boolean value, if true stepping out of functions will continue to step out till it hits a function with debug information.

 -C <python-class> ( --python-class <python-class> )
 The name of the class that will manage this step - only supported for Scripted Step.

 -a <boolean> ( --step-in-avoids-no-debug <boolean> )
 A boolean value that sets whether stepping into functions will step over functions with no debug information.

 -c <count> ( --count <count> )
 How many times to perform the stepping operation - currently only supported for step-inst and next-inst.

 -m <run-mode> ( --run-mode <run-mode> )
 Determine how to run other threads while stepping the current thread.
 Values: this-thread | all-threads | while-stepping
```

```
 -r <regular-expression> ( --step-over-regexp <regular-expression> )
 A regular expression that defines function names to not to stop at when stepping in.

 -t <function-name> ( --step-in-target <function-name> )
 The name of the directly called function step in should stop at when stepping into.

 This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.

 (lldb) help thread until
 Run the current or specified thread until it reaches a given line number or leaves the current function.

 Syntax: thread until <cmd-options> <linenum>
 Command Options Usage:
 thread until [-f <frame-index>] [-t <thread-index>] [-m <run-mode>] <linenum>

 -f <frame-index> ( --frame <frame-index> )
 Frame index for until operation - defaults to 0

 -m <run-mode> ( --run-mode <run-mode> )
 Determine how to run other threads while stepping this one
 Values: this-thread | all-threads

 -t <thread-index> ( --thread <thread-index> )
 Thread index for the thread for until operation

 This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.
-------------------------------------------------------------------------------------------------------------------------------
(lldb) help type
 The following subcommands are supported:
  category  -- A set of commands for operating on categories
 filter    -- A set of commands for operating on type filters
 format    -- A set of commands for editing variable value display options
 lookup    -- Lookup a type by name in the select target.  This command takes 'raw' input (no need to quote stuff).
 summary   -- A set of commands for editing variable summary display options
 synthetic -- A set of commands for operating on synthetic type representations
  For more help on any particular subcommand, type 'help <command> <subcommand>'.
  ---------------
```

```
(lldb) help type category
The following subcommands are supported:
define  -- Define a new category as a source of formatters.
delete  -- Delete a category and all associated formatters.
disable -- Disable a category as a source of formatters.
enable  -- Enable a category as a source of formatters.
list    -- Provide a list of all existing categories.
For more help on any particular subcommand, type 'help <command> <subcommand>'.

--------
(lldb) help type filter
The following subcommands are supported:

add    -- Add a new filter for a type.
clear  -- Delete all existing filters.
delete -- Delete an existing filter for a type.
list   -- Show a list of current filters.

For more help on any particular subcommand, type 'help <command> <subcommand>'.
(lldb) help type format
The following subcommands are supported:

add    --     Add a new formatting style for a type.
clear  --     Delete all existing format styles.
delete --     Delete an existing formatting style for a type.
info   --     This command evaluates the provided expression and shows which format is applied to the resulting value (if any).
              This command takes 'raw' input (no need to quote stuff).
list   --     Show a list of current formatting styles.
 For more help on any particular subcommand, type 'help <command> <subcommand>'.

(lldb) help type lookup
Lookup a type by name in the select target.  This command takes 'raw' input (no need to quote stuff).
 Syntax: type lookup <typename>
```

```
(lldb) help type summary
The following subcommands are supported:
add    --     Add a new summary style for a type.
clear  --     Delete all existing summary styles.
delete --     Delete an existing summary style for a type.
info   --     This command evaluates the provided expression and shows which summary is applied to the resulting value (if any).
              This command takes 'raw' input (no need to quote stuff).
list   --     Show a list of current summary styles.

For more help on any particular subcommand, type 'help <command> <subcommand>'.

(lldb) help type synthetic
The following subcommands are supported:
add    --     Add a new synthetic provider for a type.
clear  --     Delete all existing synthetic providers.
delete --     Delete an existing synthetic provider for a type.
info   --     This command evaluates the provided expression and shows which synthetic is applied to the resulting value (if any).
              This command takes 'raw' input (no need to quote stuff).
list   --     Show a list of current synthetic providers.

For more help on any particular subcommand, type 'help <command> <subcommand>'.

-------------------------------------------------------------------------------------------------------------------------------
(lldb) help version
Show version of LLDB debugger.
Syntax: version
 ------------------------------------------------------------------------------------------------------------------------------
(lldb) help watchpoint
The following subcommands are supported:

command --    A set of commands for adding, removing and examining bits of code to be executed when the watchpoint is hit (watchpoint 'commmands').
delete  --    Delete the specified watchpoint(s).  If no watchpoints are specified, delete them all.
disable --    Disable the specified watchpoint(s) without removing it/them.  If no watchpoints are specified, disable them all.
enable  --    Enable the specified disabled watchpoint(s). If no watchpoints are specified, enable all of them.
ignore  --    Set ignore count on the specified watchpoint(s).  If no watchpoints are specified, set them all.
list    --    List all watchpoints at configurable levels of detail.
modify  --    Modify the options on a watchpoint or set of watchpoints in the executable.  If no watchpoint is specified, act on the
              last created watchpoint.  Passing an empty argument clears the modification.
set     --    A set of commands for setting a watchpoint.
For more help on any particular subcommand, type 'help <command> <subcommand>'.
--------------
```

```
(lldb) help watchpoint command
The following subcommands are supported:
add    --     Add a set of commands to a watchpoint, to be executed whenever the watchpoint is hit.
delete --     Delete the set of commands from a watchpoint.
list   --     List the script or set of commands to be executed when the watchpoint is hit.

For more help on any particular subcommand, type 'help <command> <subcommand>'.

(lldb) help watchpoint delete
Delete the specified watchpoint(s).  If no watchpoints are specified, delete them all.
Syntax: watchpoint delete [<watchpt-id | watchpt-id-list>]

(lldb) help watchpoint disable
Disable the specified watchpoint(s) without removing it/them.  If no watchpoints are specified, disable them all.

Syntax: watchpoint disable [<watchpt-id | watchpt-id-list>]

(lldb) help watchpoint enable
Enable the specified disabled watchpoint(s). If no watchpoints are specified, enable all of them.

Syntax: watchpoint enable [<watchpt-id | watchpt-id-list>]

(lldb) help watchpoint ignore
Set ignore count on the specified watchpoint(s).  If no watchpoints are specified, set them all.
 Syntax: watchpoint ignore <cmd-options> [<watchpt-id | watchpt-id-list>]

Command Options Usage:
watchpoint ignore -i <count> [<watchpt-id | watchpt-id-list>]

-i <count> ( --ignore-count <count> )
Set the number of times this watchpoint is skipped before stopping.

This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --),
you must use ' -- ' between the end of the command options and the beginning of the arguments.
```

```
(lldb) help watchpoint list
List all watchpoints at configurable levels of detail.
Syntax: watchpoint list <cmd-options> [<watchpt-id | watchpt-id-list>]

Command Options Usage:
watchpoint list [-b] [<watchpt-id | watchpt-id-list>]
watchpoint list [-f] [<watchpt-id | watchpt-id-list>]
watchpoint list [-v] [<watchpt-id | watchpt-id-list>]

-b ( --brief )
Give a brief description of the watchpoint (no location info).

-f ( --full )
Give a full description of the watchpoint and its locations.

-v ( --verbose )
Explain everything we know about the watchpoint (for debugging debugger bugs).

This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- '
between the end of the command options and the beginning of the arguments.

(lldb) help watchpoint modify
Modify the options on a watchpoint or set of watchpoints in the executable.  If no watchpoint is specified, act on the last created watchpoint.  Passing an
empty argument clears the modification.

Syntax: watchpoint modify <cmd-options> [<watchpt-id | watchpt-id-list>]

Command Options Usage:
watchpoint modify [-c <expr>] [<watchpt-id | watchpt-id-list>]
-c <expr> ( --condition <expr> )
The watchpoint stops only if this condition expression evaluates to true.

This command takes options and free-form arguments.  If your arguments resemble option specifiers (i.e., they start with a - or --), you must use ' -- '
between the end of the command options and the beginning of the arguments.

(lldb) help watchpoint set
The following subcommands are supported:
expression --         Set a watchpoint on an address by supplying an expression. Use the '-w' option to specify the type of watchpoint and the '-x'
                      option to specify the byte size to watch for. If no '-w' option is specified, it defaults to write. If no '-x' option is specified, it
                      defaults to the target's pointer byte size. Note that there are limited hardware resources for watchpoints. If watchpoint setting fails,
                      consider disable/delete existing ones to free  up resources.  This command takes 'raw' input (no need to quote stuff).
```

```
variable   --        Set a watchpoint on a variable. Use the '-w' option to specify the type of watchpoint and the '-x' option to specify the byte size to
                     watch for. If no '-w' option is specified, it defaults to write. If no '-x' option is specified, it defaults to the variable's byte size.
                     Note that there are limited hardware resources for watchpoints. If watchpoint setting fails, consider disable/delete existing
                     ones to free up resources.
For more help on any particular subcommand, type 'help <command> <subcommand>'.
-----------------------------------------------------------------------------------------------------
(lldb) settings list
Top level variables:
auto-confirm             -- If true all confirmation prompts will receive their default reply.
disassembly-format       -- The default disassembly format string to use when disassembling instruction sequences.
frame-format             -- The default frame format string to use when displaying stack frame information for threads.
notify-void              -- Notify the user explicitly if an expression returns void (default: false).
prompt                   -- The debugger command line prompt displayed for the user.
script-lang              -- The script language to be used for evaluating user-written scripts.
stop-disassembly-count   -- The number of disassembly lines to show when displaying a stopped context.
stop-disassembly-display -- Control when to display disassembly when displaying a stopped context.
stop-line-count-after    -- The number of sources lines to display that come after the current source line when displaying a stopped context.
stop-line-count-before   -- The number of sources lines to display that come before the current source line when displaying a stopped context.
term-width               -- The maximum number of columns to use for displaying text.
thread-format            -- The default thread format string to use when displaying thread information.
use-external-editor      -- Whether to use an external editor or not.
use-color                -- Whether to use Ansi color codes or not.
auto-one-line-summaries  -- If true, LLDB will automatically display small structs in one-liner format (default: true).
auto-indent              -- If true, LLDB will auto indent/outdent code. Currently only supported in the REPL (default: true).
print-decls              -- If true, LLDB will print the values of variables declared in an expression. Currently only supported in the REPL (default: true).
tab-size                 -- The tab size to use when indenting code in multi-line input mode (default: 4).
escape-non-printables    -- If true, LLDB will automatically escape non-printable and escape characters when formatting strings.

'target' variables:

default-arch                     -- Default architecture to choose, when there's a choice.
expr-prefix                      -- Path to a file containing expressions to be prepended to all expressions.
prefer-dynamic-value             -- Should printed values be shown as their dynamic value.
enable-synthetic-value           -- Should synthetic values be used by default whenever available.
skip-prologue                    -- Skip function prologues when setting breakpoints by name.
source-map                       -- Source path remappings used to track the change of location between a source file when built, and where it
                                    exists on the current system.  It consists of an array of duples, the first  element of each duple is some part (starting
                                    at the root) of the path to the file when it was built, and the second is where the remainder of the original build
                                    hierarchy is rooted on the local system.  Each element of the array is checked in order and the first one that results in
                                    a match wins.
exec-search-paths                -- Executable search paths to use when locating executable files whose paths don't match the local file system.
```

```
debug-file-search-paths             -- List of directories to be searched when locating debug symbol files.
swift-framework-search-paths        -- List of directories to be searched when locating fraomeworks for Swift.
swift-module-search-paths           -- List of directories to be searched when locating modules for Swift.
max-children-count                  -- Maximum number of children to expand in any level of depth.
max-string-summary-length           -- Maximum number of characters to show when using %s in summary strings.
max-memory-read-size                -- Maximum number of bytes that 'memory read' will fetch before --force must be specified.
breakpoints-use-platform-avoid-list -- Consult the platform module avoid list when setting non-module specific breakpoints.
arg0                                -- The first argument passed to the program in the argument array which can be different from the executable itself.
run-args                            -- A list containing all the arguments to be passed to the executable when it is run.
                                    Note that this does NOT include the argv[0] which is in target.arg0.
env-vars                            -- A list of all the environment variables to be passed to the executable's environment, and their values.
inherit-env                         -- Inherit the environment from the process that is running LLDB.
input-path                          -- The file/path to be used by the executable program for reading its standard input.
output-path                         -- The file/path to be used by the executable program for writing its standard output.
error-path                          -- The file/path to be used by the executable program for writing its standard error.
detach-on-error                     -- debugserver will detach (rather than killing) a process if it loses connection with lldb.
disable-aslr                        -- Disable Address Space Layout Randomization (ASLR)
disable-stdio                       -- Disable stdin/stdout for process (e.g. for a GUI application)

inline-breakpoint-strategy          -- The strategy to use when settings breakpoints by file and line. Breakpoint locations can end up being inlined
                                    by the compiler, so that a compile unit 'a.c' might contain an inlined function from another source file. Usually this is
                                    limitted to breakpoint locations from inlined functions from header or other include files, or more accurately
                                    non-implementation source files. Sometimes code might #include implementation files and cause inlined breakpoint locations
                                    in inlined implementation files. Always checking for inlined breakpoint locations can be expensive (memory and time), so
                                    if you have a project with many headers and find that setting breakpoints is slow, then you can change this setting to
                                    headers. This setting allows you to control exactly which strategy is used when setting file and line breakpoints.

x86-disassembly-flavor              -- The default disassembly flavor to use for x86 or x86-64 targets.
use-hex-immediates                  -- Show immediates in disassembly as hexadecimal.
hex-immediate-style                 -- Which style to use for printing hexadecimal disassembly values.
use-fast-stepping                   -- Use a fast stepping algorithm based on running from branch to branch rather than instruction single-stepping.
load-script-from-symbol-file        -- Allow LLDB to load scripting resources embedded in symbol files when available.
memory-module-load-level            -- Loading modules from memory can be slow as reading the symbol tables and other data can take a long time depending on
                                    your connection to the debug target. This setting helps users control how much information gets loaded when loading
                                    modules from memory.'complete' is the default value for this setting which will load all sections and symbols by reading
                                    them from memory (slowest, most accurate). 'partial' will load sections and attempt to find function bounds without
                                    downloading the symbol table (faster, still accurate, missing symbol names).
'minimal' is the fastest setting and will load section data with no symbols, but should rarely be used as stack frames in these memory regions will be
inaccurate and not provide any context (fastest).
```

```
display-expression-in-crashlogs     -- Expressions that crash will show up in crash logs if the host system supports executable specific crash log strings
                                    and this setting is set to true.
trap-handler-names                  -- A list of trap handler function names, e.g. a common Unix user process one is _sigtramp.
sdk-path                            -- The path to the SDK used to build the current target.
module-cache-path                   -- The path to the module-cache directory.

'target.process' variables:

disable-memory-cache              -- Disable reading and caching of memory in fixed-size units.
extra-startup-command             -- A list containing extra commands understood by the particular process plugin used.
                                     For instance, to turn on debugserver logging set this to "QSetLogging:bitmask=LOG_DEFAULT;"
ignore-breakpoints-in-expressions -- If true, breakpoints will be ignored during expression evaluation.
unwind-on-error-in-expressions    -- If true, errors in expression evaluation will unwind the stack back to the state before the call.
python-os-plugin-path             -- A path to a python OS plug-in module file that contains a OperatingSystemPlugIn class.
stop-on-sharedlibrary-events      -- If true, stop when a shared library is loaded or unloaded.
detach-keeps-stopped              -- If true, detach will attempt to keep the process stopped.
memory-cache-line-size            -- The memory cache line size
```

'target.process.thread' variables:

```
step-in-avoid-nodebug  -- If true, step-in will not stop in functions with no debug information.
step-out-avoid-nodebug -- If true, when step-in/step-out/step-over leave the current frame, they will continue to step out till they come to a function with
                          debug information.  Passing a frame argument to step-out will
override this option.
step-avoid-regexp      -- A regular expression defining functions step-in won't stop in.
step-avoid-libraries   -- A list of libraries that source stepping won't stop in.
trace-thread           -- If true, this thread will single-step and log execution.
```

'interpreter' variables:

```
expand-regex-aliases         -- If true, regular expression alias commands will show the expanded command that will be executed. This can be used to debug new
                                regular expression alias commands.
prompt-on-quit               -- If true, LLDB will prompt you before quitting if there are any live processes being debugged. If false, LLDB will quit without
                                asking in any case.
stop-command-source-on-error -- If true, LLDB will stop running a 'command source' script upon encountering an error.
space-repl-prompts           -- If true, blank lines will be printed between between REPL submissions.
```

```
'plugin' variables:
 'plugin.dynamic-loader' variables:
 'plugin.dynamic-loader.darwin-kernel' variables:

load-kexts -- Automatically loads kext images when attaching to a kernel.
scan-type  -- Control how many reads lldb will make while searching for a Darwin kernel on attach.

'plugin.process' variables:

'plugin.process.kdp-remote' variables:

packet-timeout -- Specify the default packet timeout in seconds.

'plugin.process.gdb-remote' variables:

packet-timeout         -- Specify the default packet timeout in seconds.
target-definition-file -- The file that provides the description for remote target registers.

'platform' variables:

'platform.plugin' variables:

'platform.plugin.linux' variables:

use-llgs-for-local -- Control whether the platform uses llgs for local debug sessions.

'platform.plugin.darwin-kernel' variables:

search-locally-for-kexts -- Automatically search for kexts on the local system when doing kernel debugging.
kext-directories         -- Directories/KDKs to search for kexts in when starting a kernel debug session.

----------------------------------------------------------------------------------------------------
(lldb) setting show
auto-confirm (boolean)       =              false

disassembly-format (format-string) =        "{${function.initial-function}{${module.file.basename}`}{${function.name-without-args}}:}{${function.changed}
{${module.file.basename}`}{${function.name-without-args}}:}{${current-pc-arrow} }${addr-file-or-load}{ <${function.concrete-only-addr-offset-no-padding}>}: "
frame-format (format-string) = "frame #${frame.index}: ${frame.pc}{ ${module.file.basename}{`${function.name-with-args}${function.pc-offset}}}{ at
{line.file.basename}:${line.number}}\n"
```

```
notify-void (boolean)                =      false
prompt (string)                      =      "(lldb) "
script-lang (enum)                   =      python
stop-disassembly-count (int)         =      4
stop-disassembly-display (enum)      =      no-source
stop-line-count-after (int)          =      3
stop-line-count-before (int)         =      3
term-width (int)                     =      226
thread-format (format-string)        =      "thread #${thread.index}: tid = ${thread.id%tid}{, ${frame.pc}}{ ${module.file.basename}{`${function.name-with-
                                            args}${function.pc-offset}}}{ at ${line.file.basename}:${line.number}}{, name = '${thread.name}'}{, queue =
                                            '${thread.queue}'}{, activity = '${thread.info.activity.name}'}{, ${thread.info.trace_messages} messages}{, stop
                                            reason = ${thread.stop-reason}}{\nReturn value: ${thread.return-value}}{\nCompleted expression: ${thread.completed-
                                            expression}}\n"

use-external-editor (boolean)        =      false
use-color (boolean)                  =      true
auto-one-line-summaries (boolean)    =      true
auto-indent (boolean)                =      true
print-decls (boolean)                =      true
tab-size (unsigned)                  =      4
escape-non-printables (boolean)      =      true
target.default-arch (arch)           =
expr-prefix (file)                   =
prefer-dynamic-value (enum)          =      no-run-target
enable-synthetic-value (boolean)     =      true
skip-prologue (boolean)              =      true
source-map (path-map)                =
exec-search-paths (file-list)        =
debug-file-search-paths (file-list)  =
swift-framework-search-paths (file-list)    =
swift-module-search-paths (file-list)       =
max-children-count (int)                    =      256
max-string-summary-length (int)             =      1024
max-memory-read-size (int)                  =      1024
breakpoints-use-platform-avoid-list (boolean) =    true
arg0 (string)                        =
run-args (array of strings)          =
env-vars (dictionary of strings)     =
inherit-env (boolean)                =      true
input-path (file)                    =
```

```
output-path (file)                  =
error-path (file)                   =
detach-on-error (boolean)           =       true
disable-aslr (boolean)              =       true
disable-stdio (boolean)             =       false
inline-breakpoint-strategy (enum)   =       always
x86-disassembly-flavor (enum)       =       default
use-hex-immediates (boolean)        =       true
hex-immediate-style (enum)          =       c
use-fast-stepping (boolean)         =       true
load-script-from-symbol-file (enum) = warn
memory-module-load-level (enum)     =       complete
display-expression-in-crashlogs (boolean) = false
target.trap-handler-names (array of strings)=
sdk-path (file)                             =
module-cache-path (file)                    =
disable-memory-cache (boolean)              =       false
extra-startup-command (array of strings)    =
target.process.ignore-breakpoints-in-expressions (boolean) =      true
target.process.unwind-on-error-in-expressions (boolean)    =      true
python-os-plugin-path (file)                               =
target.process.stop-on-sharedlibrary-events (boolean)      =      false
target.process.detach-keeps-stopped (boolean)              =      false
memory-cache-line-size (unsigned)                          =      512
target.process.thread.step-in-avoid-nodebug (boolean)      =      true
target.process.thread.step-out-avoid-nodebug (boolean)     =      false
target.process.thread.step-avoid-regexp (regex)            =      ^std::
target.process.thread.step-avoid-libraries (file-list)     =
trace-thread (boolean)                                     =      false
interpreter.expand-regex-aliases (boolean)                 =      false
interpreter.prompt-on-quit (boolean)                       =      true
interpreter.stop-command-source-on-error (boolean)         =      true
interpreter.space-repl-prompts (boolean)                   =      false
plugin.dynamic-loader.darwin-kernel.load-kexts (boolean)   =      true
plugin.dynamic-loader.darwin-kernel.scan-type (enum)       =      fast-scan
plugin.process.kdp-remote.packet-timeout (unsigned)        =      5
plugin.process.gdb-remote.packet-timeout (unsigned)        =      1
plugin.process.gdb-remote.target-definition-file (file)    =
platform.plugin.linux.use-llgs-for-local (boolean)         =      false
platform.plugin.darwin-kernel.search-locally-for-kexts (boolean) = true
platform.plugin.darwin-kernel.kext-directories (file-list) =
```