NAME

lldb - LLDB Documentation

SYNOPSIS

lldb [*options*] *executable*

DESCRIPTION

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

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

ATTACHING

--attach-name <name>
Tells the debugger to attach to a process with the given name.

--attach-pid <pid>
Tells the debugger to attach to a process with the given pid.

-n <value>
Alias for --attach-name

-p <value>
Alias for --attach-pid

--wait-for
Tells the debugger to wait for a process with the given pid or name to launch before attaching.

-w Alias for --wait-for

COMMANDS

--batch
Tells the debugger to run the commands from -s, -S, -o & -O, and then quit.

-b Alias for --batch

-K <value>
Alias for --source-on-crash

-k <value>
Alias for --one-line-on-crash

--local-lldbinit
Allow the debugger to parse the .lldbinit files in the current working directory, unless --no-lldbinit is passed.

--no-lldbinit
Do not automatically parse any '.lldbinit' files.

--one-line-before-file <command>
Tells the debugger to execute this one-line lldb command before any file provided on the command line has been loaded.

--one-line-on-crash <command>
When in batch mode, tells the debugger to run this one-line lldb command if the target crashes.

--one-line <command>
Tells the debugger to execute this one-line lldb command after any file provided on the command line has been loaded.

-O <value>
Alias for --one-line-before-file

-o <value>
Alias for --one-line

-Q Alias for --source-quietly

--source-before-file <file>
Tells the debugger to read in and execute the lldb commands in the given file, before any file has been loaded.

--source-on-crash <file>
When in batch mode, tells the debugger to source this file of lldb commands if the target crashes.

--source-quietly
Tells the debugger to execute this one-line lldb command before any file has been loaded.

--source <file>
Tells the debugger to read in and execute the lldb commands in the given file, after any file has been loaded.

-S <value>
Alias for --source-before-file

-s <value>
Alias for --source

-x Alias for --no-lldbinit

OPTIONS

--arch <architecture>
Tells the debugger to use the specified architecture when starting and running the program.

-a <value>
Alias for --arch

--capture-path <filename>
Tells the debugger to use the given filename for the reproducer.

--capture
Tells the debugger to capture a reproducer.

--core <filename>
Tells the debugger to use the full path to <filename> as the core file.

-c <value>
Alias for --core

--debug
Tells the debugger to print out extra information for debugging itself.

-d Alias for --debug

--editor
Tells the debugger to open source files using the host’s "external editor" mechanism.

-e Alias for --editor

--file <filename>
Tells the debugger to use the file <filename> as the program to be debugged.

-f <value>
Alias for --file

--help
Prints out the usage information for the LLDB debugger.

-h Alias for --help

--no-use-colors
Do not use colors.

--replay <filename>
Tells the debugger to replay a reproducer from <filename>.

--version
Prints out the current version number of the LLDB debugger.

-v Alias for --version

-X Alias for --no-use-color

REPL

-r=<flags>
Alias for --repl=<flags>

--repl-language <language>
Chooses the language for the REPL.

--repl=<flags>
Runs lldb in REPL mode with a stub process with the given flags.

-R <value>
Alias for --repl-language

SCRIPTING

-l <value>
Alias for --script-language

--python-path
Prints out the path to the lldb.py file for this version of lldb.

-P Alias for --python-path

--script-language <language>
Tells the debugger to use the specified scripting language for user-defined scripts.

EXAMPLES

The debugger can be started in several modes.

Passing an executable as a positional argument prepares lldb to debug the given executable. To disambiguate between arguments passed to lldb and arguments passed to the debugged executable, arguments starting with a - must be passed after --.

lldb --arch x86_64 /path/to/program program argument -- --arch arvm7

For convenience, passing the executable after -- is also supported.

lldb --arch x86_64 -- /path/to/program program argument --arch arvm7

Passing one of the attach options causes **lldb** to immediately attach to the given process.

lldb -p <pid> lldb -n <process-name>

Passing --repl starts **lldb** in REPL mode.

lldb -r

Passing --core causes **lldb** to debug the core file.

lldb -c /path/to/core

Command options can be combined with these modes and cause **lldb** to run the specified commands before or after events, like loading the file or crashing, in the order provided on the command line.

lldb -O ’settings set stop-disassembly-count 20’ -o ’run’ -o ’bt’ lldb -S /source/before/file -s

/source/after/file lldb -K /source/before/crash -k /source/after/crash

Note: In REPL mode no file is loaded, so commands specified to run after loading the file (via -o or -s) will be ignored.

USING LLDB

In **lldb** there is a help command which can be used to find descriptions and examples of all **lldb** commands. To get help on "breakpoint set" you would type "help breakpoint set".

There is also an apropos command which will search the help text of all commands for a given term -- this is useful for locating a command by topic. For instance, "apropos breakpoint" will list any command that has the word "breakpoint" in its help text.

CONFIGURATION FILES

lldb reads things like settings, aliases and commands from the .lldbinit file.

First, **lldb** will try to read the application specific init file whose name is ~/.lldbinit followed by a "-" and the name of the current program. This would be ~/.lldbinit-lldb for the command line **lldb** and ~/.lldbinit-Xcode for Xcode. If there is no application specific init file, **lldb** will look for an init file in the home directory. If launched with a *REPL* option, it will first look for a REPL configuration file, specific to the REPL language. The init file should be named as follow: **.lldbinit-<language>-repl** (i.e. **.lldbinit-swift-repl**). If this file doesn't exist, or **lldb** wasn't launch with *REPL*, meaning there is neither a REPL init file nor an application specific init file, **lldb** will fallback to the global ~/.lldbinit.

Secondly, it will look for an .lldbinit file in the current working directory. For security reasons, **lldb** will print a warning and not source this file by default. This behavior can be changed by changing the target.load-cwd-lldbinit setting.

To always load the .lldbinit file in the current working directory, add the following command to ~/.lldbinit:

settings set target.load-cwd-lldbinit true

To never load the .lldbinit file in the current working directory and silence the warning, add the following command to ~/.lldbinit:

settings set target.load-cwd-lldbinit false

SEE ALSO

The LLDB project page *https://lldb.llvm.org* has many different resources for **lldb** users -- the gdb/lldb command equivalence page *https://lldb.llvm.org/use/map.html* can be especially helpful for users coming from gdb.

AUTHOR

LLVM project

COPYRIGHT