LLDB LLVM debugger

CONCEPTS

Compile with -g (and -O0 for sane stepping), otherwise LLDB has no source lines or local variables.

Debugger settings, aliases
- **Target** executable, breakpoints
  - **Process** running program
    - **Thread** bt, step
      - **Frame** variables, registers

Commands are noun verb (breakpoint set, frame select); most have short aliases such as b, n, s, c, bt, p, f.

START AND ATTACH

`lldb` *`<prog>`*
Load a program

`lldb --` *`<prog> <args>`*
Load a program with arguments

`lldb -p` *`<pid>`*
Attach to a running process

`lldb -c` *`<core> <prog>`*
Inspect a core dump

`run [args]`
Start the program

`process attach --name` *`<name>`*
Attach by process name

`settings set target.run-args` *`<args>`*
Set arguments for later runs

`settings set target.env-vars` *`<VAR>=<val>`*
Set an environment variable

`process detach`
Release the process and keep it running

`quit`
Leave LLDB

BREAKPOINTS

`b` *`<func>`*
Break at a function

`b` *`<file>:<line>`*
Break at a line of a file

`breakpoint set --name` *`<func>`* **`--condition`** *`'<cond>'`*
Break only when the condition holds

BREAKPOINTS (cont.)

`breakpoint list`
List breakpoints and hit counts

`breakpoint delete` *`<n>`*
Delete a breakpoint

`breakpoint disable` *`<n>`*
Disable a breakpoint

`breakpoint enable` *`<n>`*
Re-enable a breakpoint

`breakpoint set --func-regex` *`<regex>`*
Break on every function matching a regex

`breakpoint set --method` *`<name>`*
Break on a C++ method by base name

`breakpoint modify --condition` *`'<cond>' <n>`*
Add or change a condition

WATCHPOINTS

`watchpoint set variable` *`<var>`*
Stop when a variable changes

`watchpoint set expression --` *`<addr>`*
Watch the memory at an address

`watchpoint list`
List watchpoints

`watchpoint delete` *`<n>`*
Delete a watchpoint

RUN CONTROL

`c`
Continue until the next stop

`n`
Step over the next line

`s`
Step into the next line

`finish`
Run until the current function returns

`thread until` *`<line>`*
Run to a line in this frame

`si`
Step one instruction

`ni`
Step over one instruction (calls too)

STACK AND THREADS

`bt`
Backtrace of the current thread

`bt all`
Backtrace of every thread

`frame select` *`<n>`*
Switch to frame n

`up`
Go to the caller frame

`down`
Go back toward the callee frame

`frame info`
Show the current frame

`thread list`
List threads

`thread select` *`<n>`*
Switch to thread n

INSPECT DATA

`frame variable`
Show arguments and locals

`frame variable` *`<var>`*
Show one variable

`p` *`<expr>`*
Evaluate a C/C++ expression

MEMORY AND REGISTERS

`memory read --size 4 --format x --count 4` *`<addr>`*
Dump four 32-bit words in hex

`register read`
Show general registers

SOURCE AND SYMBOLS

`disassemble --frame`
Disassemble the current function

FROM GDB

GDB to LLDB

GDB	LLDB
`break foo`	b foo
`rbreak <re>`	breakpoint set -r <re>
`info locals`	frame variable
`info break`	breakpoint list
`watch <var>`	watchpoint set variable <var>
`info threads`	thread list
`disassemble`	disassemble --frame
`info registers`	register read