LLDB LLVM debugger

CONCEPTS

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

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

run [args]
Start the program

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

breakpoint list
List breakpoints and hit counts

breakpoint delete *<n>*
Delete a breakpoint

WATCHPOINTS

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

RUN CONTROL

c
Continue until the next stop

n
Step over the next line

s
Step into the next line

RUN CONTROL (cont.)

finish
Run until the current function returns

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

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

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