Understanding LLDB Command Syntax

You interact with LLDB by entering commands into a debugging session. Each LLDB command consists of zero or more subcommands and may additionally specify one or more options or arguments, in the following form:

```
<command> [<subcommand>...] [--<option> [<option-value>]]... [argument]...
```

Subcommands

Subcommands are space-delimited tokens that organize related operations. Typically, the final subcommand of a command is a verb that indicates the action to be performed. For example, commands related to managing breakpoints start with `breakpoint`, such as the `breakpoint list` and `breakpoint disable` commands, which list and disable breakpoints, respectively.

Arguments

A command may require one or more arguments. Arguments are space-delimited tokens that direct the action to be performed. For example, the `breakpoint disable` command requires a single argument to specify which breakpoint to disable, such as `breakpoint disable 1`, which disables the breakpoint with ID equal to `1`.

> **Note:** Specify arguments containing whitespace by surrounding them with single quotes (`' '`) or double quotes (`" "`). Within single or double quotes, you can use the backslash character (\) to escape non-delimiting quotation marks, such as `"some \"quoted\" string"`.

Options

A command may also include one or more options. Options are space-delimited tokens beginning with a double dash (`--`) that may be used in different combinations to modify the action to be performed. Some options also provide a shorthand form that uses a single dash (`-`). For example, when the `breakpoint set` command specifies the `--one-shot` (`-o`) option, as in `breakpoint set —one--shot`, the breakpoint is deleted the first time it causes the program to stop.

Some options specify a single space-delimited value to act as a named parameter for the command. For example, the `breakpoint set` command can set a breakpoint on a particular function by passing the `--name` option with the name of the function as the option value.

Some commands may require options in certain combinations. For example, the `breakpoint set` command can set a breakpoint on a particular location in code by passing the `--file` and `--line` options with the corresponding file name and line number.

> **Note:** Commands that accept options as well as freeform arguments, such as the expression command, must place a space-delimited double dash (`--`) between the last option and the first argument. This ensures that arguments resembling an option by starting with a dash (`-`) are interpreted as an argument.

Command Forms

Equivalent LLDB commands can be represented in various different forms. For example, each of the following commands perform the same action of evaluating the object description of a variable `someVariable` and printing the result:

Canonical form	`expression --object-description -- someVariable`
Abbreviated form	`e -O -- someVariable`
Alias	`po someVariable`

- The *canonical form* of a command acts as a descriptive representation of the action to be performed.
- The *abbreviated form* of a command uses short abbreviations of commands and subcommands, such as `e` for `expression`, and the short form of options, such as `-O` for `--object-description`.
- An *alias* can be created for any command subsequence, providing a convenient shorthand to perform common actions, such as `po` to evaluate and print an object expression.

For clarity, this document primarily refers to commands in their canonical form, followed by any abbreviated forms or aliases in parentheses. You’re encouraged to use the more convenient shorthand when debugging your own code.

Using Command-Line Help

LLDB provides extensive documentation within a debugger session through the `help` command.

Calling the `help` command without any arguments lists all of the available debugger commands, as well as the existing command aliases.

```
(lldb) help
Debugger commands:


  apropos           -- Find a list of debugger commands related to a particular
word/subject.
  breakpoint        -- A set of commands for operating on breakpoints.
  ...
```

You can get information about the usage of a particular command or subcommand by passing it as the argument of the `help` command. For example, to get help for the `breakpoint set` command:

```
(lldb) help breakpoint set
     Sets a breakpoint or set of breakpoints in the executable.


Syntax: breakpoint set <cmd-options>
...
```

The `help` command works for any available form of a command, including an `alias`. For example, to determine the command aliased by `po`:

```
(lldb) help po
     Evaluate an expression (ObjC++ or Swift) in the current program context, using
user defined variables and
     variables currently in scope.
...
'po' is an abbreviation for 'expression -O  -- '
```