1 LLDB

1.1 コマンド構文

GDB の自由な形式のコマンドとは異なり、LLDB は構造化されたコマンドを持ちます。すべての LLDB コマンドは以下の形をしています。

```
<noun> <verb> [-<options> [<option-value>]] [<argument> [<argument>...]]
```

- `argument`, `options`, `option-value` は全てホワイトスペースで区切られます。
- スペースを含む引数はシングルまたはダブルクォートで囲むことで保護できます。
- 引数内の"及び\は\でエスケープできます。
- バッククォートで囲んだ文字列は式として解釈され値に置き換わります。
- `--`を使用してそれ以前の引数のみをオプションとして明示できます。
- TAB による補完が可能です。
- `help` コマンドがあります。
- `apropos` もあります。
- エイリアスもあります。

```
(lldb) command alias bfl breakpoint set -f %1 -l %2
```

- 規定のエイリアスもあります。網羅的ではないです。
- `~/.ldinit` にエイリアスを書けば一般に使用できる。help にも反映される。
- GDB コマンドのエイリアスも結構ある。
- `unalias` もできる。
- `script` で Python インタプリタにアクセスできる。

1.2 プログラムを LLDB に読み込む

先ず、デバッグするプログラムを指定します。LLDB 起動時に、コマンドラインでデバッグするプログラムを指定できます。

```
$ lldb <program>
```

若しくは LLDB 起動後に `file` コマンドで指定します。

```
(lldb) file <program>
```

1.3 ブレークポイントを管理する

`help breakpoint [<subcommand>]`でブレークポイント関連のコマンドのヘルプを閲覧できます。

1.3.1 clear

指定したファイル、行数にあるブレークポイントを削除または無効化します。

文法:

```
(lldb) breakpoint clear <options>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-l, --line <linenum>`	行数を指定
`-f, --file <filename>`	ファイル名を指定

1.3.2 command

停止時のコマンドを設定します。

文法:

```
(lldb) breakpoint command <subcommand> [<subcommand-options>] <breakpt-id>
```

`subcommand` には `add, delete, list` が指定できます。

1.3.2.1 add

コマンドを追加します。

文法:

```
(lldb) breakpoint command add <options> [<breakpoint-id>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-D, --dummy-breakpoints`	ダミーブレークポイントを指定
`-o, --one-liner <cmd>`	停止時に実行するコマンドを設定
`-F, --python-function <func>`	停止時に実行する Python の関数を設定
`-s, --script-type <none>`	コマンドの言語を指定。`command, python, lua, default-script` が指定可能
`-e, --stop-on-error <bool>`	コマンド実行時エラーで停止するかの設定
`-k, --structured-data-key <none>`	The key for a key/value pair passed to the implementation of a breakpoint command. Pairs can be specified more than once.
`-v, --structured-data-value <none>`	The value for the previous key in the pair passed to the implementation of a breakpoint command. Pairs can be specified more than once.

1.3.2.2 delete

コマンドを削除します。

文法:

```
(lldb) breakpoint delete <options> [<breakpoint-id-list>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-D, --dummy-breakpoints`	ダミーブレークポイントを指定
`-d, --disabled`	現在無効な(リストで指定した以外の)すべてを指定
`-f, --force`	警告なしですべて指定

1.3.2.3 list

設定されているブレークポイントを表示します。

文法:

```
(lldb) breakpoint list <options> [<breakpoint-id>]
```

`options` に指定できるオプションは以下の通りです:

option name	description	
`-D, --dummy-breakpoints`	ダミーブレークポイントを指定	
`-b, --brief`	情報を短く表示	`bi`, `fi`, `iv` の組み合わせのみ可
`-f, --full`	すべての情報を表示	
`-i, --internal`	デバッガの内部ブレークポイントも表示	
`-v, --verbose`	わかることすべてを表示	

1.3.3 delete

ブレークポイントを削除します。

文法:

```
(lldb) breakpoint delete <options> [<breakpoint-id-list>]
```

`options` に指定できるオプションは以下の通りです:

option name	description	
`-D, --dummy-breakpoints`	ダミーブレークポイントを指定	短縮して指定可能`-Ddf`
`-d, --disabled`	現在無効な(リストで指定した以外の)すべてを指定	
`-f, --force`	警告なしですべて指定	

1.3.4 disable

ブレークポイントを無効化します。

文法:

```
```

```
(lldb) breakpoint disable [<breakpoint-id-list>]
```

1.3.5 enable

ブレークポイントを有効化します。

文法:

```
(lldb) breakpoint enable [<breakpoint-id-list>]
```

1.3.6 list

設定されているブレークポイントを表示します。

文法:

```
(lldb) breakpoint list [<options>] [<breakpoint-id>]
```

`options` に指定できるオプションは以下の通りです:

option name	description	
`-D, --dummy-breakpoints`	ダミーブレークポイントを表示	
`-b, --brief`	ブレークポイントの情報を短く表示	`bi`, `fi`, `iv` の組み合わせのみ可
`-f, --full`	ブレークポイントのすべての情報を表示	
`-i, --internal`	デバッガの内部ブレークポイントも表示	
`-v, --verbose`	ブレークポイントについてわかることすべてを表示	

1.3.7 modify

設定されているブレークポイントの内容を変更します。

文法:

```
(lldb) breakpoint modify [<options>] [<breakpoint-id-list>]
```

`options` に指定できるオプションは以下の通りです:

option name	description	
`-D, --dummy-breakpoints`	ダミーブレークポイント	まとめて `-De` のように指定可能
`-d, --disable`	ブレークポイントを無効化	
`-e, --enable`	ブレークポイントを有効化	
`-G --auto-continue <bool>`	コマンド実行後自動で再開	
`-c, --condition <cond>`	条件式 `cond` を満たすときだけ停止	
`-i, --ignore-count <n>`	ブレークポイントを無視する回数	

-o, --one-shot <bool>	一度停止したら削除	
-q, --queue-name <name>	指定したキューに入っているスレッドのみ停止	
-t, --thread-id <tid>	指定したスレッドのみ停止	
-x, --thread-index <tidx>	指定したインデクスのスレッドのみ停止	
-T, --thread-name <name>	指定したスレッドのみ停止	

1.3.8 name

ブレークポイントの名前を管理します。

文法:

```
(lldb) breakpoint name <subcommand> [<options>]
```

subcommand には add, configure, delete, list が指定できます。

1.3.8.1 add

名前を追加します。

文法:

```
(lldb) breakpoint name add <options> <breakpoint-id-list>
```

options に指定できるオプションは以下の通りです:

option name	description
-N, --name <breakpoint-name>	追加する名前

1.3.8.2 configure

名前のあるブレークポイントを編集します。ブレークポイント ID を指定した場合、オプションをコピーします。それ以外ではそのまま編集されます。

文法:

```
(lldb) breakpoint name configure [<options>] [<breakpoint-name-list>]
```

options に指定できるオプションは以下の通りです:

option name	description
-d, --disable	無効化されたブレークポイントを設置
-e, --enable	ブレークポイントを有効化
-G --auto-continue <bool>	コマンド実行後自動で再開
-C, --command <cmd>	停止時に自動実行するコマンド
-c, --condition <cond>	条件式 cond を満たすときだけ停止
-i, --ignore-count <n>	ブレークポイントを無視する回数

`-o, --one-shot <bool>`	一度停止したら削除
`-q, --queue-name <name>`	指定したキューに入っているスレッドのみ停止
`-t, --thread-id <tid>`	指定したスレッドのみ停止
`-x, --thread-index <tidx>`	指定したインデクスのスレッドのみ停止
`-T, --thread-name <name>`	指定したスレッドのみ停止
`-D, --allow-delete <bool>`	名前で削除、すべて削除を許可
`-A, --allow-disable <bool>`	名前で無効化、すべて無効化を許可
`-L, --allow-list <bool>`	明示的に指定されないリストを許可
`-B, --breakpoint-id <breakpoint-id>`	ブレークポイント ID を指定
`-H, --help-string <none>`	名前の目的の説明を設定

1.3.8.3 delete

名前を削除します。

文法:

```
(lldb) breakpoint name delete <options> <breakpoint-id-list>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-N --name <name>`	削除する名前を指定

1.3.8.4 list

名前を表示します。

文法:

```
(lldb) breakpoint name list <options>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-D, --dummy-breakpoints`	ダミーブレークポイントを表示

1.3.9 read

以前に `write` で保存したブレークポイントを読み込みます。

文法:

```
(lldb) breakpoint read <options>
```

`options` に指定できるオプションは以下の通りです:

option name	description

-f, --file <filename>	読み込むファイルを指定
-N, --breakpoint-name <name>	指定した名前のブレークポイントのみ読み込む

1.3.10 set

プログラムにブレークポイントを設置します。

文法:

```
(lldb) breakpoint set <options>
```

options に指定できるオプションは以下の通りです:

option name	description	
-A, --all-files	全てのファイルを検索	フラグ。まとめて-ADHdのように指定可能
-D, --dummy-breakpoints	ダミーのブレークポイントを設置	
-H, --hardware	ハードウェアブレークポイントを使用	
-d, --disable	無効化されたブレークポイントを設置	
-l, --line <linenum>	行番号 linenum を指定	場所指定。併用不可
-a, --address <addr>	アドレス addr を指定	
-n, --name <func>	関数名 func を指定	
-F, --fullname <name>	関数の完全修飾名を指定	
-S, --selector <selector>	Objective-C のセレクタ名を指定	
-M, --method <method>	C++のメソッド名を指定	
-r, --func-regex <reg>	正規表現 reg にマッチする関数名を持つ関数を指定	
-b, --basename <func>	関数の基本名が func の関数を指定 (C++の名前空間や引数を無視)	
-p, --source-pattern-regex <reg>	指定したファイル内のソースコードで正規表現にマッチする箇所を指定	
-E, --language-exceprion <lang>	指定した言語の例外スローを指定	
-y, --joint-specifier <linespec>	filename:line[:column]の形式でファイルと行を指定	
-k, --structured-data-key <none>	スクリプトによるブレークポイントの実装に渡されるキーと値のペアのキー。 ペアは複数指定できます。	その他のオプション。併用できないものもある
-v, --structured-data-value <none>	スクリプトによるブレークポイントの実装に渡されるキーと値のペアの値。ペアは複数指定できます。	
-G --auto-continue <bool>	コマンド実行後自動で再開	
-C, --command <cmd>	停止時に自動実行するコマンド	
-c, --condition <cond>	条件式 cond を満たすときだけ停止	

-i, --ignore-count <n>	ブレークポイントを無視する回数	
-o, --one-shot <bool>	一度停止したら削除	
-q, --queue-name <name>	指定したキューに入っているスレッドのみ停止	
-t, --thread-id <tid>	指定したスレッドのみ停止	
-x, --thread-index <tidx>	指定したインデクスのスレッドのみ停止	
-T, --thread-name <name>	指定したスレッドのみ停止	
-R, --address-slide <addr>	指定されたオフセットを、ブレークポイントが解決するアドレスに追加します。現在のところ、これは指定されたオフセットをそのまま適用し、命令境界に整列させようとはしません。	
-N, --breakpoint-name <name>	ブレークポイントの名前	
-u, --column <col>	列を指定	
-f, --file <filename>	検索するファイルを指定	
-m, --move-to-nearest-code <bool>	一番近いコードへブレークポイントを移動	
-s, --shlib <name>	共有ライブラリを指定	
-K, --skip-prologue <bool>	プロローグをスキップ	

1.3.11 write

ブレークポイントをファイルに保存します。read で読み込めます。ブレークポイントを指定しなければ全て保存されます。

文法:

```
(lldb) breakpoint write <options> [<breakpoint-id-list>]
```

options に指定できるオプションは以下の通りです:

option name	description
-a, --append	ファイルが既存ならば追加
-f, --file <filename>	保存先のファイル名

1.4 ウォッチポイントを管理する

help watchpoint [<subcommand>]でウォッチポイント関連のコマンドのヘルプを閲覧できます。

1.4.1 command

ウォッチポイントにヒットしたときに実行するコマンドを管理します。

文法:

```
```

```
(lldb) watchpoint command <subcommand> [<options>]
```

subcommand には add, delete, list が指定できます。

1.4.1.1 add

コマンドを追加します。

文法:

```
(lldb) watchpoint command add [<options>] <watchpoint-id>
```

options に指定できるオプションは以下の通りです:

option name	description
-o, --one-liner <cmd>	停止時に実行するコマンドを設定
-F, --python-function <func>	停止時に実行する Python の関数を設定
-s, --script-type <none>	コマンドの言語を指定。command, python, lua, default-script が指定可能
-e, --stop-on-error <bool>	コマンド実行時エラーで停止するかの設定

1.4.1.2 delete

コマンドを削除します。

文法:

```
(lldb) watchpoint command delete <watchpoint-id>
```

1.4.1.3 list

コマンドを表示します。

文法:

```
(lldb) watchpoint command list <watchpoint-id>
```

1.4.2 delete

ウォッチポイントを削除します。

文法:

```
(lldb) watchpoint delete [<options>] [<watchpoint-id-list>]
```

options に指定できるオプションは以下の通りです:

option name	description
-f, --force	確認なしで削除

1.4.3 disable
ウォッチポイントを無効化します。

文法:

```
(lldb) watchpoint disable [<watchpoint-id-list>]
```

1.4.4 enable
ウォッチポイントを有効化します。

文法:

```
(lldb) watchpoint enable [<watchpoint-id-list>]
```

1.4.5 ignore
イグノアカウンタを設定します。

文法:

```
(lldb) watchpoint ignore <options> <watchpoint-id-list>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-i, --ignore-count`	ウォッチポイントを無視する回数

1.4.6 list
設定されたウォッチポイントを表示します。

文法:

```
(lldb) watchpoint list [<options>] [<watchpoint-id-list>]
```

`options` に指定できるオプションは以下の通りです:

option name	description	
`-b, --brief`	短い説明を表示	オプションは併用不可
`-f, --full`	完全な説明を表示	
`-v, --verbose`	全てを表示	

1.4.7 modify
ウォッチポイントを変更します。

文法:

```
(lldb) watchpoint modify [<options>] [<watchpoint-id-list>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-c, --condition <cond>`	条件を満たすときだけ停止

1.4.8 set

ウォッチポイントを設定します。

文法:

```
(lldb) watchpoint set <subcommand> [<options>]
```

`subcommand` には `expression, variable` が設定できます。

1.4.8.1 expression

式の結果が指すアドレスにウォッチポイントを設定します。

文法:

```
(lldb) watchpoint set expression [<options>] -- <expr>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-w, --watch <type>`	ウォッチのタイプを指定。`read, write, read_write` が指定可能
`-s, --size <size>`	監視するバイト数

1.4.8.2 variable

変数にウォッチポイントを設定します。

文法:

```
(lldb) watchpoint set variable [<options>] -- <varname>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-w, --watch <type>`	ウォッチのタイプを指定。`read, write, read_write` が指定可能
`-s, --size <size>`	監視するバイト数

1.5 プロセスを制御する

LLDB でプロセスを制御するには `process` コマンドを使用します。

文法:

```
(lldb) process <subcommand> [<options>]
```

subcommand には attach, connect, continue, detach, handle, interrupt, kill, launch, load, plugin, save-core, signal, status, trace, unload が指定できます。

1.5.1 attach

プロセスに LLDB をアタッチします。

文法:

```
(lldb) process attach <options>
```

options に指定できるオプションは以下の通りです:

option name	description
-c, --continue	アタッチ後に停止せずに継続
-i, --include-existing	-w 指定時に、すでに存在するプロセスを含む
-w, --waitfor	-n で指定した名前のプロセスが起動するまで待つ
-p, --pid	プロセス ID を指定
-n, --name	名前を指定してアタッチ
-P, --plugin <plugin>	プロセスプラグインを指定

1.5.2 connect

リモートデバッグサービスに接続します。

! TODO !

確認

文法:

```
(lldb) process connect <remote-url>
```

1.5.3 continue

現在のプロセスのすべてのスレッドを継続実行します。

文法:

```
(lldb) process continue <options>
```

options に指定できるオプションは以下の通りです:

option name	description
-i, --ignore-count <count>	ignore-counter を設定します

1.5.4 detach

プロセスからデタッチします。

文法:

```
(lldb) process detach <options>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-s, --keep-stopped <bool>`	

1.5.5 handle

シグナルの LLDB での扱いを設定します。

文法:

```
(lldb) process handle <options> [<signal>...]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-n, --notify <bool>`	デバッガがシグナル受信をユーザに知らせるか
`-p, --pass <bool>`	シグナルをプロセスに渡すか
`-s, --stop <bool>`	シグナル受信時にプロセスを停止するか

`signal` を指定しない場合、すべてのシグナルに対して設定されます。

1.5.6 interrupt

現在のターゲットプロセスに割り込みを行います。

文法:

```
(lldb) process interrupt
```

1.5.7 kill

現在のターゲットプロセスを kill します。

文法:

```
(lldb) process kill
```

1.5.8 launch

プログラムを起動します。

文法:

```
(lldb) process launch <options> [<args>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-s, --stop-at-entry`	エントリポイントで停止
`-t, --tty <none>`	プロセスを開始するターミナル
`-n, --no-stdio`	標準出力を行わない
`-a, --arch <arch>`	曖昧なときにアーキテクチャを指定
`-A, --disable-aslr <bool>`	アドレス空間のランダム化を行うか
`-E, --environment <none>`	環境変数を `NAME=VALUE` の形で設定
`-P, --plugin <plugin>`	プロセスプラグイン名
`-c, --shell <filename>`	プロセスを走らせるシェル
`-e, --stderr <filename>`	標準エラー出力
`-X, --shell-expand-args <bool>`	プロセス起動時にシェルが引数を拡張するか
`-i, --stdin <filename>`	標準入力
`-o, --stdout <filename>`	標準出力
`-w, --working-dir <dir>`	プロセスのワーキングディレクトリ
`-C, --script-class <python-class>`	scripted class の管理クラス名
`-k, --structured-data-key <none>`	The key for a key/value pair passed to the implementation of a scripted process. Pairs can be specified more than once.
`-v, --structured-data-value <none>`	The value for the previous key in the pair passed to the implementation of a scripted process. Pairs can be specified more than once.

1.5.9 load

現在のプロセスに共有ライブラリをロードします。

文法:

```
(lldb) process load <options> <filename>...
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-i <path>, --install=<path>`	ターゲットに共有ライブラリをインストール

1.5.10 plugin

現在のターゲットプロセスプラグインにコマンドを渡します。

文法:

```
```

```
(lldb) process plugin <args>
```

1.5.11 save-core

現在のプロセス状態をコアファイルに保存します。

文法:

```
(lldb) process save-core <options> <filename>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-p <plugin>, --plugin-name=<plugin>`	コアファイルを生成するためのプラグイン
`-s, --style <corefile-style>`	コアファイルの保存形式。`full, modified-memory, stack` が指定可能

1.5.12 signal

UNIX のシグナルをプロセスに送信します。

文法:

```
(lldb) process signal <signal>
```

1.5.13 status

プロセスのステータスと停止位置を表示します。

文法:

```
(lldb) process status <options>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-v, --verbose`	拡張情報を含むすべてのプロセスステータスを表示

1.5.14 trace

プロセスをトレースします。

文法:

```
(lldb) process trace <subcommand> [<options>]
```

`subcommand` には `save, start, stop` が指定できます。

1.5.15 save

現在のプロセスのトレースを保存します。

文法:

```
(lldb) process trace save [<options>]
```

options に指定できるオプションは以下の通りです:

option name	description
-d, --directory <dir>	保存場所

1.5.16 start
トレースを開始します。

文法:

```
(lldb) process trace start <options>
```

1.5.17 stop
トレースを終了します。

文法:

```
(lldb) process trace stop
```

1.5.18 unload
現在のプロセスから共有ライブラリをアンロードします。

文法:

```
(lldb) process unload <index>
```

1.6 スレッドを制御する
スレッドを制御するには thread コマンドを使用します。

文法:

```
(lldb) thread <subcommand> [<options>]
```

subcommand には backtrace, continue, exception, info, jump, list, plan, return, select, siginfo, step-in, step-inst, step-inst-over, step-out, step-over, step-scripted, trace, until が指定できます。

1.6.1 backtrace
スレッドのコールスタックを表示します。

文法:

```
(lldb) thread backtrace <options>
```

options に指定できるオプションは以下の通りです:

option name	description
`-c, --count <count>`	表示するフレームの数。-1 は全て
`-e, --extended <bool>`	拡張バックトレース
`-s, --start <frame-index>`	バックトレースの開始フレーム

1.6.2 continue

プロセスを継続実行します。スレッドを指定しない場合全てのスレッドが対象になります。

文法:

```
(lldb) thread continue <thread-index>...
```

1.6.3 exception

現在の例外オブジェクトを表示します。

文法:

```
(lldb) thread exception
```

1.6.4 info

一つ以上のスレッドの概要情報を表示します。スレッドを指定しない場合、現在のスレッドが対象です。

文法:

```
(lldb) thread info <options>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-j, --json`	JSON 形式で表示
`-s, --stop-info`	JSON 形式で停止情報を表示

1.6.5 jump

プログラムカウンタを変更します。

文法:

```
(lldb) thread jump <options>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-a, --address <expr>`	ジャンプ先のアドレス
`-b, --by <offset>`	ジャンプ先のオフセット
`-f, --file <filename>`	ジャンプするファイル

`-l, --line <linenum>`	ジャンプするソース行番号
`-r, --force`	プログラムカウンタが関数の外に出ることを許可

1.6.6 list

各スレッドについての概要を表示します。`setting set thread-format` で表示をカスタマイズできます。

文法:

```
(lldb) thread list
```

1.6.7 plan

スレッドの実行計画を管理するコマンドです。

文法:

```
(lldb) thread plan <subcommand> [<options>]
```

`subcommand` には `discard, list, prune` が指定できます。

1.6.7.1 discard

スレッド計画を廃棄します。

文法:

```
(lldb) thread plan discard <index>
```

1.6.7.2 list

スレッド計画を表示します。

文法:

```
(lldb) thread plan list <options>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-i, --internal`	内部スレッド計画も表示
`-t, --thread-id`	スレッド ID を指定
`-u, --unreported`	unreported なスレッドを指定
`-v, --verbose`	より多くの情報を表示

1.6.7.3 prune

現在の unreported なスレッドのすべての計画を削除します。

文法:

```
(lldb) thread plan prune <thread-id>...
```

1.6.8 return

選択中のスタックフレームから返ります。

文法:

```
(lldb) thread return <options>
```

! TODO !

`options` に指定できるオプションは以下の通りです:

option name	description
`-x, --from-expression`	一番内側から式の値で返る

1.6.9 select

選択中のスレッドを変更します。

文法:

```
(lldb) thread select <index>
```

1.6.10 siginfo

現在の siginfo オブジェクトを表示します。

文法:

```
(lldb) thread siginfo
```

1.6.11 step-in

ソースコード上のステップを行います。関数呼び出しの場合には関数の中に入ります。スレッドを指定しない限り現在のスレッドにのみ適用されます。

文法:

```
(lldb) thread step-in [<options>] [<thread-id>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-A, --step-out-avoids-no-debug <bool>`	ステップアウト時にデバッグ情報のある関数に当たるまでステップアウトを続ける
`-a, --step-in-avoids-no-debug <bool>`	デバッグ情報のない関数にステップインせずにステップオーバーする

`-c, --count <count>`	ステップ回数
`-e, --end-linenumber <linenum>`	ステップを停止する行番号
`-m, --run-mode <mode>`	他スレッドの処理方法。`this-thread`, `all-threads`, `while-stepping` が指定可能
`-r, --step-over-regexp <regex>`	正規表現にマッチする関数をスキップ
`-t, --step-in-target <func-name>`	直接呼び出された関数のステップイン時に停止する関数名

1.6.12 step-inst

命令上のステップを行います。call は中に入ります。

文法:

```
(lldb) thread step-inst [<options>] [<thread-id>]
```

`options` に指定できるオプションは Section 1.6.11 と同じです。

1.6.13 step-inst-over

命令上のステップを行います。call はステップオーバーします。

文法:

```
(lldb) thread step-inst-over [<options>] [<thread-id>]
```

`options` に指定できるオプションは Section 1.6.11 と同じです。

1.6.14 step-out

現在のスタックフレームを抜けるまで実行し、停止します。

文法:

```
(lldb) thread step-out [<options>] [<thread-id>]
```

`options` に指定できるオプションは Section 1.6.11 と同じです。

1.6.15 step-over

ソースコード上のステップを行います。関数呼び出しはステップオーバーします。

文法:

```
(lldb) thread step-over [<options>] [<thread-id>]
```

`options` に指定できるオプションは Section 1.6.11 と同じです。

1.6.16 step-scripted

Step as instructed by the script class passed in the -C option. You can also specify a dictionary of key (-k) and value (-v) pairs that will be used to populate an SBStructuredData Dictionary, which will be passed to the constructor of the class implementing the scripted step. See the Python Reference for more details.

文法:

```
(lldb) thread step-scripted [<options>] [<thread-id>]
```

`options` には Section 1.6.11 で指定できるものに加えて以下が指定可能です。 `options` に指定できるオプションは以下の通りです:

option name	description
`-k, --structured-data-key`	The key for a key/value pair passed to the implementation of a scripted step. Pairs can be specified more than once.
`-v, --structured-data-value`	The value for the previous key in the pair passed to the implementation of a scripted step. Pairs can be specified more than once.

1.6.17 trace

トレース関連のコマンドです。

文法:

```
(lldb) thread trace <subcommand> [<options>]
```

`subcommand` には `dump, export, start, stop` が指定可能です。

1.6.17.1 dump

トレース情報を表示します。

文法:

```
(lldb) thread trace subcommand [<options>]
```

`subcommand` には `info, instructions` が指定できます。

1.6.17.1.1 info

トレースされた情報をダンプします。

文法:

```
(lldb) thread trace info [<options>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-v, --verbose`	冗長なダンプ

1.6.17.1.2 instructions

トレースされた命令をダンプします。

文法:

```
(lldb) thread trace instructions [<options>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-c, --count <count>`	表示する命令の数
`-f, --forwards`	一番古い地点から表示
`-r, --raw`	シンボル情報なし
`-s, --skip <index>`	スキップする命令数
`-t, --tsc`	可能なら命令ごとにタイムスタンプを表示

1.6.17.2 export

トレースをエクスポートします。

文法:

```
(lldb) thread trace export <plugin> [<options>]
```

1.6.17.3 start

トレースを開始します。

文法:

```
(lldb) thread trace start [<options>]
```

1.6.17.4 stop

トレースを終了します。

文法:

```
(lldb) thread trace stop [<thread-index>...]
```

1.6.18 until

指定した箇所まで実行します。

文法:

```
(lldb) thread until <options> <linenum>
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-a, --address <expr>`	このアドレスに到達するか関数を抜けるまで実行
`-f, --frane <index>`	対象のフレーム。デフォルトは 0

`-m, --run-mode <mode>`	実行モード。`this-thread, all-threads`
`-t, --thread <index>`	対象のスレッド

1.7 フレームを制御する

フレームを制御するには `frame` コマンドを使用します。

文法:

```
(lldb) frame <subcommand> [<options>]
```

`subcommand` にはが指定できます。

1.7.1 diagnose

現在の停止位置がどのようなパスでレジスタやアドレスに到達したかを判断しようとします。

文法:

```
(lldb) frame diagnose [<options>] [<frame-index>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-a, --address <addr>`	アドレス
`-o, --offset <offset>`	オフセット
`-r, --register <name>`	レジスタ

1.7.2 info

現在のフレームの情報を表示します。

文法:

```
(lldb) frame info
```

1.7.3 select

スタックフレームからフレームを選択します。

文法:

```
(lldb) frame select [<options>] [<frame-index>]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-r, --relative <offset>`	現在のフレームからのオフセットで指定

1.7.4 variable

現在のスタックフレームに存在する変数を表示します。

文法:

```
(lldb) frame variable <options> [<varname>...]
```

`options` に指定できるオプションは以下の通りです:

option name	description
`-A, --show-all-children`	上限を無視して子を表示
`-D, --depth <count>`	表示する最大深さ
`-F, --flat`	フラットフォーマットで表示
`-G, --gdb-format`	GDB フォーマットで表示
`-L, --locationn`	位置情報を表示
`-O, --object-description`	言語ごとの説明 API を使用して表示
`-P, --ptr-depth <count>`	値をダンプする際のポインタの深さ
`-R, --raw-output`	フォーマットを行わない
`-S, --synthetic-type <bool>`	synthetic provider に従うかを表示
`-T, --show-types`	型を表示
`-V, --validate <bool>`	型チェックの結果を表示
`-Y [<count>], --no-summary-depth=[<count>]`	概要情報を省略する深さ
`-Z, --element-count <count>`	型が配列あるかのように表示
`-a, --no-args`	関数の引数を省略
`-c, --show-declaration`	変数の宣言情報を表示
`-d, --dynamic-type <none>`	オブジェクトの完全な動的型を表示
`-f, --format <fmt>`	フォーマット
`-g, --show-globals`	グローバル変数を表示
`-l, --no-locals`	局所変数を省略
`-r, --regex`	変数名を正規表現として解釈
`-s, --scope`	変数のスコープを表示
`-t, --no-recognized-args`	recognized function arguments を省略
`-y, --summary <name>`	変数の出力が使用すべきサマリーを指定
`-z, --summary-string <name>`	フォーマットに使用するサマリー文