GDB

Chapter 1

1 概要

1.1 GDB とは

GDB は Gnu Project のデバッガです。

1.2 コマンド

GDB を起動すると先頭に(gdb)の表示が出て対話モードになります。

```
(gdb) <command> [<options>]
```

の形式でコマンドを実行し、様々の処理を行うことでデバッグを行います。

コマンド入力時は TAB による補完が可能です。また、曖昧さがない場合にはコマンドは先頭数文字だけの入力でも認識されます。

1.3 ヘルプ

コマンドのヘルプを help コマンドで確認できます。

```
(gdb) help [<class>|<command>|all]
```

argument name	description
class	特定のクラスのコマンドのリストを表示
command	特定のコマンドのヘルプを表示
all	全てのコマンドのリストを表示

class に指定できるもの:

class name	description
aliases	他のコマンドのエイリアス
breakpoints	特定の場所でプログラムを停止させる
data	データを検査する
files	ファイルを指定、検査する
internals	メンテナンスコマンド
obscure	ぼんやりした特徴
running	プログラムを走らせる
stack	スタックを検査する
status	ステータスの検査
support	サポート機能
tracepoint	プログラムを停止せずにトレースする
user-defined	ユーザー定義のコマンド

この一覧は引数無しで help コマンドを実行することで閲覧できます。

2 基本操作

2.1 GDB の起動

GDB を起動するには以下のコマンドをシェルで叩きます。

```
$ gdb [<options>] [<program> [<core-file>|<pid>]]
$ gdb [<options>] --args <program> [<args>...]
```

`options` に指定できるもの

option name	description
ファイル等の指定	
`--args`	実行ファイル以降の引数をプログラムに渡す
`--core <file>`	ダンプされたコアファイルを検査
`--exec <file>`	実行ファイルを検査
`--pid <pid>`	プロセス ID を指定してアタッチ
`--directory <dir>`	ソースファイル検索ディレクトリを指定
`--se <file>`	シンボルファイルおよび実行ファイルを指定
`--symbol <file>`	シンボルファイルを指定
`--readnow`	全てのシンボルを最初のアクセス時に読み込む
`--readnever`	シンボルファイルを読み込まない
`--write`	実行ファイルとコアファイルに書き込みを行う
初期化コマンドとコマンドファイル	
`-x, --command <file>`	ファイルからコマンドを実行
`-ix, --init-command <file>`	実行ファイル読み込み前に実行するコマンドファイル
`-ex, --eval-command <cmd>`	コマンドを実行
`-iex, --init-eval-command <cmd>`	実行ファイル読み込み前に実行するコマンド
`--nh`	`~/.gdbinit` を読み込まない
`--nx`	`.gdbinit` を全て読み込まない
出力と UI 制御	
`--fullname`	emacs-GDB で使用される出力情報
`--interpreter <interpreter>`	インタープリターを指定
`--tty <tty>`	デバッグするプログラムの入出力 TTY を指定
`-w`	GUI を使用
`--nw`	GUI を使用しない
`--tui`	TUI を使用
`--dbx`	DBX compatibility モード
`-q, --quiet, --silent`	GDB 開始時のメッセージを表示しない
モード	
`--batch`	バッチモードで実行
`--batch-silent`	バッチモードで実行し、出力を行わない

`--return-child-result`	デバッグするプログラムの終了コードで GDB も終了する
`--configuration`	GDB の詳細な設定を表示
`--help`	GDB 起動のヘルプを表示
`--version`	バージョン情報を表示
リモートデバッグオプション	
`-b <baudrate>`	帯域幅を指定
`-l <timeout>`	タイムアウト時間(秒)を指定
その他のオプション	
`--cd <dir>`	カレントディレクトリを変更
`-D, --data-directory <dir>`	データディレクトリを変更

2.2 プログラムの読み込み

デバッグするプログラムを GDB に読み込むには起動時の引数で指定するか、コマンドで指定することができます。

起動時には例えば

```
$ gdb ./a.out
```

のように指定し、コマンドでは

```
(gdb) file ./a.out
```

のように指定できます。

2.3 プログラムの起動

プログラムを起動するには `run` コマンドまたは `start` コマンドを使用します。 `run` コマンドは設定した停止場所まで継続実行し、`start` ではエントリポイントで停止します。

`run` コマンドの文法:

```
(gdb) run <args>
```

argument name	description
`args`	プログラムにコマンドライン引数として渡す

`start` コマンドの文法:

```
(gdb) start <args>
```

argument name	description
`args`	プログラムにコマンドライン引数として渡す

2.4 プログラムの終了
プログラムを終了するには `ctrl+c` を使用するか `kill` コマンドで終了できます。

2.5 GDB の終了
GDB を終了するには `ctrl+d` を使用するか `quit` または `exit` コマンドで終了できます。

3 プログラムの停止
デバッグ中のプログラムを停止させるにはあらかじめ停止場所や条件を設定しておく必要があります。停止箇所としてはブレークポイント、ウォッチポイント、キャッチポイントが設定できます。

なお、ブレークポイント、ウォッチポイント、キャッチポイントをまとめてブレークポイントと呼ぶこともあります。

3.1 ブレークポイント
プログラムが到達した際に停止する場所です。

3.1.1 ブレークポイントの設置
ブレークポイントを設置するには `break` コマンドを使用します。

```
(gdb) break [<location>] [thread <tnum>] [[-force-condition] if [<cond>]]
```

argument name	description
`location`	設置場所を指定します。詳細は Section 3.1.1.1 を参照
`thread <tnum>`	特定のスレッドでのみ停止。スレッド番号は `info threads` で確認
`-force-condition`	`cond` の式が現在の文脈で無効でも無視して設置
`cond`	条件式を満たすときのみ停止

3.1.1.1 位置指定
位置の指定方法はいくつかあります。代表的なものは
- `[<file>:]<linenum>`: 行番号を指定
- `[<file>:]<funcname>`: 関数名を指定
- `<addr>`: アドレスを指定

詳細は https://sourceware.org/gdb/current/onlinedocs/gdb.html/Location-Specifications.html#Location-Specifications をご覧ください。

3.2 ウォッチポイント
式の値を監視して変更があった際にプログラムを停止させます。

3.2.1 ウォッチポイントの設置
ウォッチポイントを設置するには `watch` コマンドを使用します。

```
(gdb) watch [-l <location>] <expr>
```

argument name	description
`location`	`expr` が指すアドレスのメモリを監視
`expr`	監視する式

3.3 キャッチポイント

例外やシステムコールなどの際に停止します。

3.3.1 キャッチポイントの設置

キャッチポイントを設置するには `catch` コマンドを使用します。

```
(gdb) catch <subcommand> [<args>]
```

subcommand name	description
`catch`	例外のキャッチ
`exec`	`exec` の呼び出し
`fork`	`fork` の呼び出し
`load`	共有ライブラリロード
`rethrow`	例外の再スロー
`signal`	シグナルキャッチ
`syscall`	システムコール発行
`throw`	例外スロー
`unload`	共有ライブラリアンロード
`vfork`	`vfork` 呼び出し

3.4 ブレークポイントの削除

ブレークポイントを削除するには `delete` コマンドを使用します。

```
(gdb) delete [breakpoint-num]
```

argument name	description
`breakpoint-num`	削除するブレークポイントを指定。ブレークポイント番号は `info breakpoints` で確認

3.5 ブレークポイントの無効化

ブレークポイントを無効化するには `disable breakpoints` コマンドを使用します。

```
(gdb) disable breakpoints [breakpoint-num]
```

argument name	description
`breakpoint-num`	無効化するブレークポイントを指定。ブレークポイント番号は `info breakpoints` で確認

3.6 ブレークポイントの有効化
ブレークポイントを有効化するには `enable breakpoints` コマンドを使用します。

```
(gdb) enable breakpoints [breakpoint-num]
```

argument name	description
`breakpoint-num`	有効化するブレークポイントを指定。ブレークポイント番号は `info breakpoints` で確認

3.7 ブレークポイント到達時実行コマンド
ブレークポイントに到達した際に実行するコマンドを `commands` コマンドで設定できます。

```
(gdb) commands <command> [<breakpoint-num> ...]
```

argument name	description
`command`	実行するコマンド
`breakpoint-num`	設定するブレークポイントを指定。ブレークポイント番号は `info breakpoints` で確認。指定しない場合、一番最後のブレークポイントが指定される

3.8 ブレークポイントの保存
`save breakpoints` コマンドで現在設定されているブレークポイントをファイルに保存できます。

```
(gdb) save breakpoints
```

3.9 ブレークポイントの読み込み
保存したブレークポイントは `source` コマンドで読み込めます。

```
(gdb) source <file>
```

4 プログラムの再開
ブレークポイント等で停止したプログラムを様々な方法で再開できます。再開方法は主にステップ実行と継続実行に分かれ、ステップ実行では一定分だけ進んで停止し、継続実行では次のブレークポイントまで進みます。

4.1.1 継続実行
継続実行をするには `continue` コマンドを使用します。

```
(gdb) continue [<count>]
```

argument name	description
`count`	ブレークポイントを無視する回数

4.1.2 関数から返る

関数から返るまで実行するには `finish` コマンドを使用します。 現在選択中のフレームが帰るまで実行します。フレームの選択方法は Section 5.2 を参照してください。

```
(gdb) finish
```

4.1.3 ジャンプ

特定の位置まで継続実行をするには `jump` コマンドを使用します。

```
(gdb) jump <loc>
```

argument name	description
`loc`	停止位置。行番号かアドレスで指定。

4.1.4 ソース行単位ステップオーバー

ソースコード行単位でステップし、関数呼び出しで中に入らないような実行(ステップオーバー)は `next` コマンドを使用します。

```
(gdb) next [<count>]
```

argument name	description
`count`	ステップ回数

4.1.5 命令単位ステップオーバー

機械語命令単位でのステップオーバーは `nexti` コマンドを使用します。

```
(gdb) nexti [<count>]
```

argument name	description
`count`	ステップ回数

4.1.6 ソース行単位ステップイン

ソースコード行単位でステップし、関数呼び出しで中に入るような実行(ステップイン)は `next` コマンドを使用します。

```
(gdb) step [<count>]
```

argument name	description
`count`	ステップ回数

4.1.7 命令単位ステップイン

機械語命令単位でのステップインは `nexti` コマンドを使用します。

```
```

```
(gdb) stepi [<count>]
```

argument name	description
count	ステップ回数

4.1.8 until

ソース行の上で実際に現在の次の行まで、または指定の場所まで実行するには until コマンドを使用します。

```
(gdb) until [<loc>]
```

argument name	description
loc	停止場所。指定方法は Section 3.1.1.1 を参照。指定しない場合、現在の行の次の行が指定される

5 スタックフレーム

5.1 バックトレース

コールスタックを表示するには backtrace コマンドを使用します。

```
(gdb) backtrace [<options>] [<count>]
```

option name	description
-entry-values <v>	関数の引数の表示方法を設定
-frame-arguments <v>	スカラ値でない引数の表示設定
-raw-frame-argument on|off	raw フォームで引数を表示
-frame-info <v>	フレーム情報の表示設定
-past-main on|off	main 関数以前のフレームを表示するか
-past-entry on|off	エントリポイント以前のフレームを表示するか
-full	局所変数を全て表示

5.2 フレームの選択

フレームを選択するには frame コマンドを使用します。

```
(gdb) frame [<num>]
```

argument name	description
num	選択するフレーム番号。フレーム番号は backtrace で確認。Section 5.1 を参照

6 変数の表示

6.1 プリント

変数や式の値を確認するには `print` コマンドを使用します。

```
(gdb) print [<options>...] [/<fmt>] [<expr>]
```

argument name	description
`options`	オプションを指定
`fmt`	フォーマットを指定
`expr`	表示する式

指定できるオプションおよびフォーマットは `help print` で確認できます。

6.2 ディスプレイ

ブレークポイント等で停止した際に自動的に変数や式の値を表示するには `display` コマンドを使用します。

```
(gdb) display[/<fmt>] <expr>
```

argument name	description
`fmt`	フォーマットを指定
`expr`	表示する式