

第 10 回 Unix ゼミ

C プログラム(デバッグ編)

川島研 B4 高木 空

2024 年 07 月 01 日

お品書き

お品書き	0
デバッガ	1
デバッガとは	1
デバッグの手法	2
デバッガの具体例	4
GDB の使い方	5

デバツガ

- デバッグ (debug)
 - ▶ バグ(bug)を取り除く (de-) こと
- デバッグの手法
 - ▶ print デバッグ
 - ▶ コードを読む
 - ▶ デバツガを使う
 - 今回の主題

デバッグの手法

- print デバッグ
 - ▶ ソースコードに print を埋め込む
 - ▶ 利点
 - 気軽に実行できる
 - 欲しい出力を欲しい形式で得られる
 - ▶ 欠点
 - ソースコードを改変する必要がある
 - バグの箇所を検討してからしかできない
 - 得られる情報が少ない

デバッグの手法

- デバッガを使う
 - ▶ デバッガ - デバッグを補助するツール
 - ▶ 利点
 - プログラム全体を観察できる
 - プログラムの変更が(一般には)不要
 - スタックやメモリの監視もできる
 - ▶ 欠点
 - 使い方を知っている必要がある

C 言語のデバッグ

- C 言語プログラムのデバッグ
- GDB
 - ▶ Gnu Project のデバッグ
 - ▶ gcc を使うならコレ
 - ▶ Linux に標準搭載されている
- LLDB
 - ▶ LLVM のデバッグ
 - ▶ clang を使うならコレ

GDB の使い方

1. 起動

```
> gdb [options] [<file> [<core-file>|<pid>]]
```

- 実行ファイルとコアファイルまたはプロセス ID を指定

例:

```
> gdb ./a.out
```

- よく使う options
 - ▶ `--args`: 実行ファイルに渡す引数を指定
 - ▶ `--help`: 簡単な使用方法を表示

GDB の使い方

2. 停止

```
(gdb) quit [<expression>]
```

で GDB を終了する

- `expression`: GDB の終了コード

GDB の使い方

3. コマンド

```
(gdb) <command> [args...]
```

の形式でコマンドを使用

- ヘルプコマンド

```
(gdb) help [<class>]
```

- コマンドのヘルプを表示
- 引数無しで実行すると指定できる class を表示

GDB の使い方

4. プログラムの開始

```
(gdb) start [<args>...]  
(gdb) run [<args>...]
```

でプログラムを開始

- start : 開始直後で停止
- run : 停止させるかプログラム終了まで継続実行
- args... の引数はプログラムに渡される

GDB の使い方

5. プログラムの停止

- ブレークポイント
 - ▶ プログラムが到達すると停止する場所

```
(gdb) break [<loc>] [if <cond>]
```

でブレークポイントを設置

- loc: ブレークポイントを設置する場所
 - ▶ [<filename>:]<linenum>: 行番号
 - ▶ [<filename>:]<funname>: 関数名
- cond: 条件式。満たすときだけ停止

GDB の使い方

5. プログラムの停止

- ウォッチポイント
 - 式の値が変更したら停止

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

でウォッチポイントを設置

- -l, -location: exp が指すアドレスのメモリを監視
- exp: 監視対象の式

GDB の使い方

5. プログラムの停止

- キャッチポイント
 - ▶ 例外などを検出して停止

```
(gdb) catch <event>
```

でキャッチポイントを設置。event に指定できるもの:

- catch [<regex>]: 例外キャッチ
- rethrow [<regex>]: 例外再スロー
- throw [<regex>]: 例外スロー
- syscall [<name>|<num>|g:<group>]...: システムコール発行
- load [<regex>]: 共有ライブラリロード
- unload [<regex>]: 共有ライブラリアンロード
- signal [<sig>|all]: シグナル配信

GDB の使い方
