第20回

標準的デバッガGDBのすべて

岸 哲夫

シンボル・テーブルの調査と実行のコントロール

今回はGNUデバッガgdbを使ってソース・コードのシンボル・テーブルを調べる方法と，シグナルやジャンプ機能を使ってデバッグ中のプログラムの実行を制御する方法について解説する．（編集部）

今回はプログラムの動的解析ツールvalgrindの説明の続きを行う予定でしたが，予定を変更してソース・コードのシンボル・テーブルを調べるコマンドの説明とgdbからプログラムを制御する方法について解説します．

gdbを使ってプログラム内部のデータを解析するためには，変数名と関数名，そして型などの情報が重要です．また，プログラムの動作を制御することにより，変数の値を変化させて通常は呼び出されない関数を通すことも可能です．

● infoコマンドによるシンボル・テーブルの調査

コマンドの詳細については次回以降で触れます．ここでは，機能と使い方の概略の説明にとどめておきます．

リスト1に，シンボル・テーブルの調査を行うために使うサンプル・プログラムを示します．

関数の外部で定義した変数，および関数の一覧は，次のgdbコマンドで表示できます．`info`はgdbが持っているさまざまな情報を表示するためのコマンドです．`variables`はすべてのグローバル変数およびスタティック変数を表示します．`function`はすべての関数名を表示します．

```
(gdb)info variables
(gdb)info functions
```

図1に，**リスト1**のプログラムで`info variables`と`info functions`を実行した結果を示します．`info variables`では関数外部で定義した変数`wk`が，`info functions`ではソース中で定義されている関数`main()`，`test()`，`wkfree()`が表示されています．

リスト1　シンボル・テーブルの調査用サンプル・プログラム

```
#include <stdlib.h>
static char *wk;          ← 関数外部で定義した変数
void test(void);
void wkfree(void);

int main(void)
{
    test();
    wkfree();
    test();
    wkfree();
    test();
    wkfree();
}

void test(void)
{
    wk = (char*)malloc(10);
    wk[0] = '0';
}

void wkfree(void)
{
    free(wk);
}
```

● そのほかのコマンド

gdbには，`info`以外にも下記のようなデータの調査に便利なコマンドが用意されています．

- `whatis　変数名`

```
(gdb)whatis wk
type = char *
(gdb)
```

この場合このコマンドは変数`wk`のタイプを表示します．

- `whatis`

```
(gdb)whatis
The history is empty.
(gdb)print wk
$1 = 0x0
(gdb)whatis
type = char *
(gdb)
```

`whatis`の引き数を省略すると，変数履歴の最後のデータ型を表示します．

- `ptype　タイプ名`

このコマンドは，構造体の詳細を表示します．**リスト2**のサ

リスト2　ptype実行用サンプル・プログラム

```
#include <stdio.h>
struct kouzou {
        int wk01;
        char wk02[10];
        double wk03;
        int wk04;
};

struct kouzou k01 = { 5, "xxxxxxxxxx", 83.5,1 };
int main(void)
{
        return 0;
}
```


KEYWORD —— GDB，デバッガ，シグナル，シンボル・テーブル