教科書には載っていない！

現場で役立つ プログラミングのちょい技

第5回　コンパイラの中では何が起こっているの？

邑中 雅樹

今回はGCCを例にとって，コンパイラの中で何が起こっているのかをのぞき見する．コンパイラの途中経過を知ることは，単にしくみが分かって楽しいというだけでなく，デバッグの役にも立つ．多くの開発経験を持つ筆者にだからこそ知っている“ちょい技”を教えてもらおう．

（編集部）

1. 準備運動 ―― コンパイルしてみる

前回は，少し細かいところに話題が行きすぎて，紙幅が尽きてしまいました．今回こそ手を動かしましょう．

お好きなエディタで，**リスト1**のソース・コードを入力してください．ファイル名は何でもかまいませんが，とりあえずtest.cとでもしておきましょう．test.cをセーブしてコマンド・プロンプトに戻って，これをバイナリにコンパイルします．

```
gcc test.c
```

コンパイルは一瞬で終わります．同じディレクトリに，a.outというファイルができているはずです．Microsoft Windows系のgccの場合は，生成されるファイルがa.outではなくて，a.exeとなる場合もあります．

セルフ・コンパイラならば，これを実行すると次のような出力が得られます．

```
Hello World.
```

図1　test.c→a.outの間を見てみよう

● コンパイラとコンパイラ・フロントエンド

test.cのコンパイルで使用したコマンドはgccのみです．しかしながら実際には，コマンドとしてのgccはフロントエンドと呼ばれる機能しか提供していません．裏側（バックエンド）では，複数のコマンドが動作しています．

これらのコマンド群を総称して，ツールチェイン（toolchain）と呼ぶことがあります．道具（tool）の鎖（chain）の意味どおり，鎖のように連携し，流れ作業を分担してソース・コードをバイナリに変換します．

バックエンドのコマンドが動いている様子の見やすさは，ツールチェインに依存します．内部処理の隠しようがないオープン・ソース製品では，オープン・ソースではない製品よりもより多くの情報を得られるように配慮されています．GCCの場合は，豊富なコマンドライン・オプションで，内部の各処理の過程を取得できます．

● 処理の痕跡を残してみる

通常，gccはツールチェインのバックエンドが行った処理の痕跡（こんせき）を残しませんが，コマンドライン・オプションで変更できます．次のようにgccを実行してみましょう．

```
gcc -save-temps test.c
```

a.out（a.exe）が出力されますが，同時にtest.i，

リスト1　メッセージが表示される基本プログラム（test.c）

```
#include <stdio.h>

#define MESSAGE "Hello World.\n"

int
main(void)
{
    printf(MESSAGE);
}
```