第5章 実機テストでも見つからないようなメモリ操作ミスやデータ競合の検出もOK!

高機能で簡単! 最新コンパイラのデバッグ機能

東 工太郎

● 進化中! GCCやLLVMのデバッグ機能

最新GCCやLLVMを使う理由の一つとして，デバッグ機能が豊富であることが挙げられます．

GCCやLLVMでは，GDBとLLDBというインタラクティブなシンボリック・デバッガを利用できます．シンボリック・デバッガとは，人間が理解しやすい形式（シンボル）で対象プログラムの実行状態を観測/制御できるようにしたデバッガです．

これに加え，最新GCCやLLVMは実行時プログラム解析に基づくSanitizerと呼ばれるバグ検出ツールを備えています．

さらに，発展途上/研究段階のさまざまなデバッグ機能/ツールが，GCCやLLVMをベースに実装されており，利用可能になっています．

本稿では，これらGCC&LLVMの豊富なデバッグ機能と，それらを実現するしくみを解説します．

基本デバッガ・ソフトGDBやLLDB

● 特徴1：人間が見やすい形式でデバッグできる

GCCとLLVMは，標準でGDBとLLDBというインタラクティブなシンボリック・デバッガを利用できます．

表1に示すように，GDBとLLDBはともに，多くの場面で頻繁に使用されるデバッグ用コマンドをサポートしています．ユーザはこれらのコマンドを介して，インタラクティブに対象プログラムの実行を制御し，状態を観測することで，デバッグ作業を効率的に行えます．

● 特徴2：実行状態を言語レベルで観測できる

シンボリック・デバッガの実現には，GCCやLLVMのデバッグ情報生成機能が利用されます．GCCやLLVMにおいては，プログラムのコンパイル時にソースコードの構成要素（関数/文や変数など）と目的コードの構成要素（命令やレジスタなど）との間の対応関係を，デバッグ情報として実行可能ファイルに埋め込めます．

GDBやLLDBは，**図1**に示すように，デバッグ情報を参照することにより，ユーザが発行したソースコード言語レベルのコマンドを解釈して，目的コードの実行を命令レベルで制御します．また，ユーザがプログラムの実行状態を言語レベルで観測できるようにします．

表1　最新コンパイラGCC&LLVMに対応する基本デバッグ・ソフトGDB&LLDBで使えるコマンド

デバッグ・コマンド	機　能
break	ブレーク・ポイントを設定
watch watchpoint set	ウォッチポイントを設定（GDBはwatch，LLDBはwatchpoint set）
run	デバッグ対象プログラムを実行
step	現在の観測対象スレッドをソースコード・レベルで1行実行（関数呼び出し文のstep実行は呼び出される関数の先頭文に制御を移す）
next	現在の観測対象スレッドをソースコード・レベルで1行実行（関数呼び出し文のnext実行は関数呼び出し文の次の文まで制御を移す）
continue	停止中の観測対象スレッドの実行を継続
print expr print	コマンド引き数として与えられた式を観測対象スレッドの現在のフレーム内で評価した結果を表示（GDBはprint，LLDBはexpr print）
bt	現在の観測対象スレッドのバックトレース（スタック・フレームの列）を表示
up	現在の観測対象スレッドの現在のスタック・フレームの関数を呼び出した関数のフレームに戻る
down	upの逆の動作を行う