無償評価版ツールで試せる

メモリ・リーク/変数破壊…見つけにくいバグよサヨウナラ～！

実行ログから抽出&発見！ C言語の不正なメモリ・アクセス

柴下 哲

「まったく動かない」バグであれば止まった場所にバグがあります．しかし「なんとなく動くのだけど正常な値を返さない」という性質の悪いバグもあります．この種のバグは，プログラマが気付かない場所でメモリの内容が破壊されているのが多くの原因で，発生場所の特定が困難です．

ここでは，まずはバグを発見するためのソース・コード解析の手法について解説します．そしてメモリ破壊のバグについて，メモリ・アクセスに注目してバグの発生個所を特定し，デバッグする方法について解説します．（編集部）

ソース・コード解析の二つの手法

バグのないプログラムを目指して，ソース・コードを解析するための方法として，静的解析と動的解析があります．

静的解析は，ソース・コードをツールなどで解析する方法です．簡単な文法チェッカから，ソース・コードが特定の規格に合致しているかをチェックするツールまで，さまざまな物があります．静的解析はプログラムを動かさないでチェックする手法です．

これに対し動的解析は，プログラムをコンパイルし動作させて，動作中の挙動（プログラム・カウンタの動きや変数の値など）を採取し，適切な動作をしているかチェックする方法です．デバッガやprintfデバッグなどで変数の値を見ながらデバッグする手法が該当します．

迅速なデバッグを行うには，静的解析と動的解析の両方を適材適所で使うことが重要です．

■ 静的解析…コードの論理的な問題を発見！

● コンパイラ

静的解析のもっとも基本的な物はコンパイラです．コンパイラは高級言語からターゲットの命令セットを生成する際に，コードの静的な解析をしてバグを発見しています．

さらにコード生成はせずに，静的解析に特化した各種の専用ツールが各社から提供されています．

静的解析ツールの最大の長所は，テスト・データが不要なことです．テスト・データなしで論理的な問題の検出や，バグを生みやすいコーディング・スタイルを指摘してくれます．

■ 動的解析…動かしながらチェック

● 文法が正しい≠正しく動く…動かさないとわからないバグがある

実際にコードを実行してその結果を解析する動的解析の場合は，入力データに依存して，実行されるコードと実行されないコードがあります．動的解析は「動いたコードをデバッグ対象とする」ため，そもそも通らなかったコードは解析できません．その点では静的解析の方が入力依存性がないので優れていますが，静的解析では入力データに依存する問題が全て分かるわけではありません．例えば静的解析ツールでは，そのコードがどのように動作するべきかが分かっていません．コードの記述に文法的に問題が無く，静的解析を全てパスしても，それは実際にコードを実行した際に，設計者の意図どおり正しく動作することを意味しません．

● 動的解析はコード・カバレッジとの併用が必須

よって実際にコードを実行した結果を解析する動的解析は必須です．しかし動的解析では，場合によって実行されないコードがあり，チェックできないという可能性があります．まず現在の入力データで動作するまでを解析し，修正を行います．その後でコード・カバレッジ（プログラムがすべての個所を通ったかを網羅する）のチェックを行います．これをしないと検証されていない条件分岐（テストでは

KEYWORD――不正メモリ・アクセス，メモリ破壊，バグ検出，静的解析，動的解析