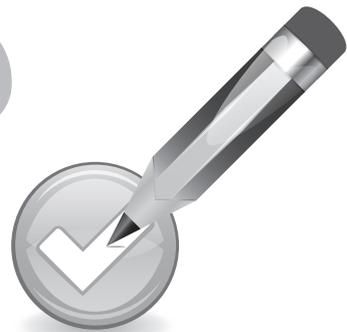


デバッグ/仕様変更/バージョンアップ...
オープン・ソースの大規模開発用環境で
不毛な繰り返しテストとはオサラバ

マイコンでトライ!

C/C++自動テスト・ツールで 高品質プログラムをつくる



第2回 準備…自動テストで試す静的解析の機能

橘田 和仁

ソース・コードのビルドやテストを自動実行するCI (Continuous Integration, 継続的インテグレーション) ツールに静的解析 (Static Code Analysis) を組み込むと、機械的にソース・コードをチェックできます。今回は、この静的解析の機能をおさらいします。(編集部)

静的解析とは

静的解析を利用すると、

- (1) パターン・マッチングでコーディング・ルールをチェック
- (2) コード内のパスを総当たりで調べる
- (3) プログラムの複雑度を数値化

して、不具合を見つけやすくなります。

● 静的解析は機械的に実行できるから見落としが少ない

人間の作業にはどうしても間違いが入り込みます。目を皿のようにしてコードのレビューを実施しても、レビュアー

リスト1 コンパイラはC言語のlintコマンドによる静的解析と同等のチェックができる

```
int foo(int sum) {  
    if (sum >= 100) {  
        return sum;  
    }  
    return; // 戻り値がない  
}
```

(a) lintコマンドで警告の出るソース・コード

```
$ lint foo.c  
foo.c:  
foo.c(5): warning: function foo expects to return value [214]
```

(b) lintコマンドでの警告メッセージ

```
$ gcc -Wall -c -o foo.o foo.c  
foo.c: In function 'foo':  
foo.c:5:3: 警告: 'return' with no value, in function returning non-void
```

(c) コンパイラgccの警告メッセージ

がバグを見逃す可能性があります。コーディング規約チェックのような、機械的に実施できるものはCIツールに組み込んだ静的解析ツールに任せてしまうのが確実です。

● 検証用のコードやデータを用意せずに実施できる

静的解析とは、ソフトウェアを実行せずに行うソース・コードの解析です。解析によってバグを見付けたり、正常に動作することを検証したりします。

一般的な静的解析ツールでは、パーサと呼ばれる構文解析器を使いソース・コードを細かく分解します。そして分解したコードが特定のパターンにマッチするかどうかをチェックします。特定のパターンとはコーディング規約ルールです。このルールに違反しているパターンを見つけ出して検出します。

静的解析は実施するための準備が少なく、手軽に実施できます。単体テストのように検証用のコードやテスト用のデータを準備する必要はありません。ヘッダ・ファイルとソース・ファイルさえあれば解析できます。

▶ コンパイラにも静的解析機能がある

静的解析は、C言語用のlintコマンドとしてかなり古くから利用され、特定のコーディング規約への準拠や、書式の統一などのチェックに使われてきました。最近では、このような単純な記述ミスの発見は、リスト1に示すように、コンパイラが行っています。

● 今どきの静的解析のメリット

今どきの静的解析ツールはコーディング規約へ準拠しているかどうかのチェックやソース・コードの複雑度の測定なども行えます。

今どきの静的解析の機能とメリットは、

- (1) コーディング規約への準拠を自動的にチェックすることで、バグの起こりやすいコードを作らせない
- (2) 問題の起こりそうな部分を総当たりチェックし、バグを見つける