UNIXでのテスト支援環境の構築

-ツール概要-

4G-4

渡邉俊雄　飛山哲幸　明智憲三郎

(三菱電機(株))

1. はじめに

UNIXオペレーティングシステムの「良さ」を十分に生かしたテスト支援環境を構築するには、次のコンセプトが重要なポイントとなる。

(1)開放的なシステム(UNIXとの親和性、テスト/デバッグ/カバレッジ測定の分離)

(2)テスト方式を制限しないシステム(テストの柔軟性)

(3)被テストモジュールのソース修正不要なシステム

当社では32ビット・スーパー・ミニコンM70MX/3000のOS60/UMX(UNIXオペレーティングシステムをもとにしたもの)上で動作するテスト支援システムを開発し充実しつつある。本論では、このテスト支援システムの各ツールについて機能とその特長について述べる。

2. 本テスト支援システムの概要

本テスト支援システムは以下のツールおよびUNIXのコマンド、ツールにより構成されている。

(1)データジェネレータ

(2)ドライバジェネレータ

(3)スタブジェネレータ

(4)カバレッジ測定

(5)テスト自動再実行

(6)テスト結果自動判定

本システムを使用してテストを行う場合の一般的な手順は以下のようになる。

①被テストモジュール(群)とドライバジェネレータ、スタブジェネレータにより作成したモジュール(群)とを自由に組み合わせ、コンパイラ、リンカによりテスト用のロードモジュールを作成する。

②データジェネレータやvi等のエディタにより、必要なテストデータファイルを作成する。

③シェルプロシジャーにより、全テストケースに対するテスト実行を制御し、被テストモジュールに対するデータは必要に応じて会話型またはファイルから入力する。すべての入力データを予めファイルに準備しておくことにより、最初から自動的に行うことも可能である。

④テスト結果をUNIXのパイプやリダイレクション機能により収集し、diffコマンドやテスト結果自動判定ツールにより判定する。異常があれば、sdb等のデバッガによりデバッグを行った後、cシェルのヒストリ機能やテスト自動再実行ツールが保存したデータを用いて全自動で再テストの実行を行う。

⑤全テストケースの結果がすべてよければ、全自動再実行を行いカバレッジ測定ツールを用いてテストの評価と性能解析を行う。

⑥カバレッジ測定の結果、必要に応じてテストケースの追加やプログラムの改造を行う。

本論では、本システムのうちドライバジェネレータ、スタブジェネレータ、カバレッジ測定ツールについて詳細に説明する。

3. 各ツールの詳細説明

以下に各ツールの機能とその特長を述べる。

3.1 ドライバジェネレータ

ドライバジェネレータは、フルスクリーン画面の穴埋め形式により、会話型でターゲット言語(C,FORTRAN)のドライバソースモジュールを生成するツールであり、以下の特長を持つ。

●会話型であり、また型指定の記号化やヘルプ機能などにより、操作性が優れている。

●一般的には図3.1のように、被テストモジュールの名前と実引数を入力するだけでドライバが生成できる。

●C言語の場合、ポインタ型の引数に対するドライバ内での領域確保のための宣言文は自動的に生成される。

●引数定義画面で入力した名前や型を、他の定義画面に転記する機能があるため、同じものを何度も入力する必要がない。

●生成されるソースはターゲット言語であるため読解性がよく、ユーザが容易にソースを修正し必要なロジックを追加することができる。

●穴埋めで入力したドライバ生成情報をエディット機能で修正していくことにより、類似ドライバの作成が効率的に行える。

●一度生成したドライバソースモジュールを逆変換することにより、会話型での修正作業が行える。(生成するソース中に穴埋めで入力したデータをコメント形式で挿入するようにしたため、ターゲット言語の構文解析が不要になり、逆変換部が高速化、軽量化できた。)

Software Test System on the Unix —Tools—
Toshio WATANABE, Tetsuyuki TOBIYAMA, Kenzaburo AKECHI
MITSUBISHI ELECTRIC CORPORATION

3.2 スタブジェネレータ

スタブジェネレータは、スタブソースモジュールを生成するツールであり、ドライバジェネレータと同様の特長に加え、以下の特長を持つ。

●生成するスタブの機能別に次の標準タイプを備えており、ユーザがタイプを選択するとツール側がそれぞれのタイプに応じた入力画面や入力項目を誘導するため、効率的にスタブを生成することができる。

①NOPスタブ

引数や外部領域への出力は行わずNOPのままリターンする。

②固定値スタブ

オペレータへの問合せをせず、予め決められた項目に固定の値(式)を無条件に出力する。

③問合せスタブ

スタブ作成時に出力項目だけを決めておき、出力値は実行時にオペレータに問い合わす。

④条件スタブ

②~③のスタブの機能を包括し、さらに実行時の入力の条件により異なった値を出力できる。

図3.1 ドライバ/スタブジェネレータの画面例

```
<<< Module definition >>>

Function type [     ]
Module   name [sankaku]

Comment   [SANKAKUKEI NO HANTEI        ]
          [                            ]
          [                            ]
          [                            ]
          [                            ]

<<< Argument definition >>>

     Name          Type   Dimension  Area
1.  [a          ] [I    ] [        ] [    ]
2.  [b          ] [I    ] [        ] [    ]
3.  [c          ] [I    ] [        ] [    ]
4.  [r          ] [I*   ] [        ] [1   ]
5.  [           ] [     ] [        ] [    ]

<<< Set/Snap definition >>>

     Item          Type  Set.Snap
1.  [a          ] [I   ] [Y] [N]
2.  [b          ] [I   ] [Y] [N]
3.  [c          ] [I   ] [Y] [N]
4.  [*r         ] [I   ] [N] [Y]
5.  [           ] [    ] [ ] [ ]
```

3.3 カバレッジ測定ツール

カバレッジ測定ツールは、C,FORTRAN言語で記述された被テストモジュールのC0カバレッジ(命令網羅度)、C1カバレッジ(分岐網羅度)及び各実行文毎の実行回数(プロファイル)を測定するためのツールで、以下の特長を持つ。

●テスト方式を問わず、また任意のモジュール(群)に対する測定を行うことが可能。

●集計結果はソースリストと共に出力されるため、未実行文や、未実行パスの確認が容易であり、また実行回数も出力されるため、性能解析も行える。

●UNIXのgrepコマンド等と組み合わせることによりノットヒットレポート等をユーザが容易に作成できる。

●測定用のデータの収集方法を改良することにより、プログラムの実行回数によらずわずかなデータエリアで、カバレッジが測定できる。

●被測定プログラムへのカバレッジ測定用の「仕掛け」の挿入によりプログラムの論理が変わることはない。(従来のツールでは測定のための 仕掛け により被測定プログラムの論理が変わってしまうことがあった。)

図3.2 カバレッジ測定結果例

```
*** ALL RUNNING TOTAL ***  ( RUNNING COUNT=    3)
LN-SN COUNT C1 LINE:----------SOURCE LIST ---------
                    1 #include <stdio.h>
                    2 sankaku(a,b,c,r)
                    3 int a,b,c,*r;
                    4 {
 5- 1      3 TF     5       if( a >= b + c || b >= c +
 6- 1      1        6           *r = 0;
 7- 1      2 TF     7       else if( a == b && b == c )
 8- 1      1        8           *r = 3;
 9- 1      1 T-     9        else if( a == b || b == c |
10- 1      1       10           *r = 2;
                   11       else
12- 1   ####       12           *r = 1;
13- 1      3       13       return;
                   14 }
-----< REPORT >-------------------------------------
C0 COVERAGE PROPORTION  =   87.5 %  (     7 /     8 )
C1 COVERAGE PROPORTION  =   83.3 %  (     5 /     6 )
```

4. おわりに

本論ではUNIXオペレーティングシステム上に構築したテスト支援システムの各ツールについて、機能とその特長について述べた。本システムの開発における成果としては以下の点があげられる。

●UNIXとの親和性を損なうことなく、従来のUNIXには見られなかったフルスクリーン入力によるマンマシンインタフェイスを持つ、操作性に優れたツールが開発できた。なお、フルスクリーン入力の実現にあたってはUNIXのcursesを使用したため、移植性にも優れている。

●テスト/デバッグ/カバレッジ測定を分離したため、複雑な構文解析を必要とせず、かつ被テストプログラムに対する制約がほとんどないツールを開発できた。

今後も冒頭に述べたコンセプトに従ったツールの開発を推進していきたい。

参考文献

松本ほか 「パーソナルコンピュータによるテスト環境の構築」 情報処理学会第32回全国大会

飛山ほか 「UNIXでのテスト支援環境の構築ー全体構成ー」 情報処理学会第33回全国大会