仮想マシンを用いたOSのテスト環境

an Operating System Test Environment using Virtual Machine

吉田　幸二†　　早川　栄一‡
Kouji Yoshida　　Eiichi Hayakawa

１．はじめに

近年、ソフトウェアの開発において、製品をリリースするまでの期間の短縮が求められている。このような要求はオペレーティングシステム(以下、OS)の開発においても同じである。OS の開発において、最も時間のかかる工程の一つにテスト, デバッグの工程が挙げられる。シミュレータを用いたデバッグの環境としては[1]があるが、OS のテストにおいて、筆者が問題と考える割込みや入出力によって起こる挙動の変化に対するサポートがない。また OS 以外のソフトウェアのテストを目的としたツールをそのまま適用することはできない。

そこで、本研究では、仮想マシンに着目し、それを用いたOSのテスト環境の開発を目的とする。

２．問題分析

OSをテストする時の問題点について述べる。

(1)　割込みや入出力の問題

割込みや入出力は、プログラムの実行中において、いつ発生するかわからない。そのため、OS の挙動が一定ではなく実行するたびにその挙動が変化してしまい、同じ条件下でのテストが行えないという問題がある。そこで、割込みや入出力の発生パターンを再現し、同じ条件下でテストできる環境が必要である。

(2)　テストの結果判断の問題

テスト結果を人が見て判断するようなテスト方法では、テストに時間がかかる上に、何度も繰り返しテストを実行するのが面倒である。そこで、テスト結果の判定を自動化することで、テストを容易に行える環境が必要である。

３．設計方針

本システムの設計方針を次に示す。

(1)　OSの挙動を再現可能な環境の提供

OS は、割込みや入出力がいつ発生するかわからないので、その挙動に再現性がない。そこで、テスト対象の OS を仮想マシン上で動作させ、仮想マシン上で発生するすべての割込みや入出力の発生をログとして取得する。そのログを基に割込みや入出力を再現することで、OS の挙動を再現する環境を提供する。

(2)　対象となるOSのソースコードを変更しない

テストを行うために特別なコードをソースコードに加えると、そのコードの影響で挙動が変化する可能性がある。そのため、本システムでは、対象となる OS のソースコードに特別なコードを追加することなくテストを行える環境を提供する。

(3)　テストの自動化

テストの自動化を行うことで、繰り返しテストを行う場合でも、容易に実行できるようにする。テストの自動化には、テスト結果の判定条件とテストケースを記述するスクリプト言語を提供する。これにより、テスト結果の判定の自動化をすると共に、テストケースの作成を容易にする。

４．システムの設計

4.1　システムの全体構成

本システムは、問題分析で挙げた問題に対して仮想マシンを用いた環境を提供することで解決する。

テスト対象の OS を動作させる仮想マシンと仮想マシンの制御、テスト結果の解析などを行うテストツールから構成されている。図１に本システムの全体構成を示す。

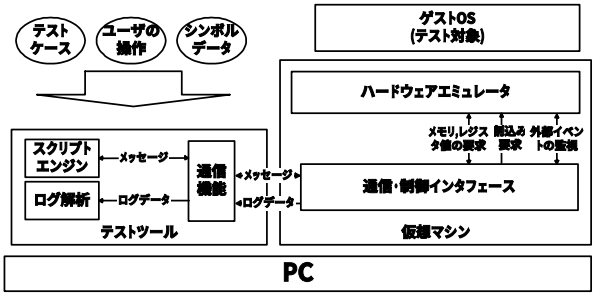

図１：システムの全体構成図

4.2　仮想マシンの設計

本システムで使用する仮想マシンは、大きく分けて次の二つの機能を提供する。

(1)ハードウェアエミュレーション機能

ゲスト OS が動作するハードウェア環境のエミュレーションを行う。提供する機能としては、機械語命令、例外、割込み、MMU、レジスタ、タイマ、入出力のエミュレーションを行う。

(2)通信・制御インタフェース

テストツールとの間でメッセージを送受信する通信インタフェースと、そのメッセージを基にハードウェアエミュレータを制御するインタフェースを提供する。制御可能な機能としては、テスト対象 OS の実行の制御、割込み、入出力の制御、メモリやレジスタの値の取得がある。

4.3　テストツールの設計

本システムで提供するテストツールは次に挙げる機能を提供する。

(1)通信機能

仮想マシンが提供する通信インタフェースを用いて、仮想マシンに接続し、仮想割込みと仮想入出力の発生要求や

†拓殖大学大学院工学研究科電子情報工学専攻
Graduate school of Engineering,Takushoku University
‡拓殖大学工学部情報工学科
Takushoku University

メモリの値、レジスタの値、ログデータの送信要求とログデータの受信を行う。

(2)テストケース定義スクリプト

テストケースを記述するスクリプト言語を提供することで、ユーザのテストケースの作成を容易にする。このスクリプトで、メッセージとテスト結果の判定条件を記述する。次にスクリプトで記述する情報を示す。

・発生させる例外，割込みの種類

発生させる例外または割込みの種類を指定する。割込みを発生させないときは省略できる。

・例外，割込みを発生させる位置

例外または割込みを発生させる位置をソースコードの行番号またはアドレスから指定する。割込みを発生させないときは省略できる。

・仮想割込み，仮想入出力のログデータ

ログデータを指定することで、ログに記録されているアドレス,時間で、仮想割込み、仮想入出力を発生させる。

・テスト結果の判定条件

テスト結果の判定条件を変数、メモリ、レジスタの値などを用いた条件式で指定する。

(3)シンボル情報の取得

OS は一般的に、C 言語のような高水準言語で記述される部分とアセンブリ言語で記述される部分がある。デバッグやテストを行う時には、C 言語で書かれた部分についてはソースコードのシンボルレベルの情報が使いたい。また、アセンブリ言語で書かれた部分については、メモリのアドレスやレジスタ値などのレベルでの情報を必要としている。そこで、ソースコードの関数名や変数名などといった、シンボル情報を、テストケース定義スクリプトから使用できるようにすることで、各レベルに対応した抽象度の情報を扱えるようにする。

5.プロトタイプの設計

5.1 プロトタイプの構成

本システムの検証を行なうためにプロトタイプを作成した。プロトタイプでは、仮想マシンのハードウェアエミュレータ部分にはオープンソースソフトウェアとして提供されている、IA-32 PC エミュレータの Bochs[2]、通信・制御インタフェースとして GDB スタブ[3]、テストケース定義スクリプトのスクリプトエンジンとして Ruby のスクリプトエンジンを用いた。

割込みや入出力の制御は Bochs や GDB スタブに拡張する形で実装を行った。また、テストに必要な機能は Ruby のライブラリとして実装し、テストケースの定義を Ruby で記述するようにした。これによりテストケースの記述を柔軟に行えるようになる。

5.2 仮想マシンの拡張

Bochs では、ハードウェアエミュレータとデバッグ用インタフェースとして、GDB スタブが提供されている。しかし、そのままでは本システムで提供する環境を実現できないので、ハードウェアエミュレータと GDB スタブに拡張を行った。

(1)割込み，入出力の制御

ハードウェアエミュレータに拡張を行い割込み,入出力を任意の時間またはアドレスで発生させる。また、割込み、入出力の発生を検出し、その時のアドレスと時間の情報をログとしてテストツールに送信できるようにした。

(2)通信プロトコルの拡張

テストツールと仮想マシンとの通信プロトコルには GDB のリモートプロトコルを用いる。テストツールは、このプロトコルを使用して、仮想マシンの制御やメモリ、レジスタ、ログなどのデータを取得する。GDB スタブで提供されていない仮想割込みや仮想入出力の制御を行えるコマンドを追加する。表１に本システムで追加するコマンドを示す。

表 1：拡張する機能

コマンド	説明
vintpoint xx,*addr,clock*	仮想割込みの挿入
Vintpoint *addr,clock*	仮想割込みの削除
vio xx,*addr,clock*	仮想入出力の挿入
Vio *addr,clock*	仮想入出力の削除

※xx:割込みまたはデバイスの種類を表す番号,addr:アドレス,clock:時間

5.3 テストツールの設計

(1)テストケース定義スクリプト

テストケース定義スクリプトの機能は、Ruby のライブラリとして実装される。次に提供する機能を示す。

表 2：提供するライブラリ

関数名	説明
run()	ゲスト OS を実行
stop()	テストの終了
set_breakpoint(addr)	ブレイクポイントの設定
step(start_addr,end_addr)	ステップ実行
set_interrupt(addr)	割込みの設定
set_memory_value(addr,value)	メモリの値を設定
get_memory_value(addr)	メモリの値を取得
set_register_value(reg_name,value)	レジスタの値を設定
get_register_value(reg_name)	レジスタの値を取得
assert(condition)	判定条件の指定
start_intlog(host_name,port_num, file_nema)	割込みログの取得開始
stop_intlog()	割込みログの取得停止

(2)仮想割込み， 仮想入出力イベントログ

仮想割込み,仮想入出力が発生した時のログを取る。ログは、テキスト形式で一つのイベントが一行に記述され、イベントの種類、発生した時のアドレス、時間、仮想入出力の場合にはその値の順番で記録される。

6．おわりに

本報告では、仮想マシンを用いた OS のテスト環境の設計とプロトタイプの設計について述べた。現在までに、割込み部分に関する Bochs の拡張、GDB スタブの拡張、割り込みのログ、Ruby のライブラリの実装が完了した。これにより、割込みの発生の再現や割込みに関する部分のテストを行なえるようになった。今後は、入出力部分の実装を行い、システムの評価をする。

参考文献

[1]Lars Albartsson,Peter S Magnusson,「Using Complete System Simulation for Temporal Debugging of General Pur-pose Operating Systems and Workloads」, MASCOTS 2000

[2]Bochs, http://bochs.sourceforge.net/

[3]Debugging with GDB, http://www.asahi-net.or.jp/~wg5k-ickw/html/online/gdb-5.0/gdb-ja_toc.html