クロス・デバッグ方式

1R−1

安田 剛、 三原 幸博
株式会社東芝 システム・ソフトウェア技術研究所

1．はじめに

クロス環境上でアプリケーションプログラムのデバッグのためにターゲットＯＳ（Operating System）のシミュレートが行えることは非常に有効である。しかし、現状では、ＯＳレベルでのシミュレータはあまりみられず、バイナリレベル（コンピュータ・ハードウェアを機械語のレベルでシミュレートする方法）のシミュレータが一般的である。

本稿では、ターゲットＯＳのシステムコールを使用している組み込み用アプリケーション・プログラムをクロス環境上でデバッグできるような方式を提案し、その構成法について記述する。

なお、これはソフトウェア生産工業化システムＩＭＡＰ（Integrated software MAnagement and Production support system）[1]の一環として行っているものである。

2．システムコールのシミュレーション

組み込み用のアプリケーション・プログラム中で、ＯＳのシステムコールを使っている場合、クロス環境上でのデバッグは困難なものとなる。

システムコールのシミュレーションでの問題は、異なる環境上におけるシステムコール入口以後のインプリメントにある。インプリメントに際しては仮想ＯＳインターフェースを設定し、対象ＯＳを直接実現する方法[2]や、実ＯＳインターフェースとの整合をとることによって行う方法[3]などがある。しかし、ハードウェア資源とくにハードウェアを構成するコンポーネントレベルまでのシミュレーションについては言及されていない。これは、各コンポーネントのモデル化の困難、実時間性を忠実にシミュレートできないなどといった問題によるものと考えられる。

また、デバッグの観点からみると、現在の資源の状態を表示するといった機能はバイナリレベルでのデバッガではおこなえずクロス環境上でターゲットＯＳのシミュレートをしアプリケーションプログラムのデバッグを行うということに関しての開発効率は決して満足のいくものではない。

A Cross debugging Method
Takeshi YASUDA and Yukihiro MIHARA
Toshiba Corporation

ここでの目的は、ターゲットＯＳ用アプリケーションプログラムのデバッグ環境の構築をより汎用的に行えるような枠組みを与えることであり、より多くのクロス環境上で容易に実現可能な方式の提案である。

本稿では上の目的に沿って、以下の点を中心にクロス環境上でのデバッグ方式を検討した。

i)システムコール呼出し時の情報提供
ii)対話形式による資源状態情報の変更容易性
iii)シミュレート部の実現容易性

i)についてのアプローチはシステムコール呼び出し時にインターフェース解析を行いそのデータをデータベース化することで解決する。ii)については、後述するモデルを適用することにより、テストを行う場合にも対処できるようにする。iii)については、シミュレート部のハードウェアに関する部分を独立にさせ、システムコールシミュレート部との交信によって全体のシミュレートを行わせる。このハードウェアシミュレート部が実現されていない場合でも、ユーザとの会話で必要情報を提供してもらうことにより疑似的にシミュレートをおこなう。

3．基本モデル

基本モデルを考えることで汎用的な方式の基盤を与える。本方式では、オブジェクト指向的な考え方を導入し、各資源の内部状態を記述した資源状態管理表をオブジェクトとみなし、それとの交信によってシミュレーションを行い、デバッグ機能を実現する。つまり、シミュレート対象のＯＳが管理する資源の状態を表形式で管理し、各資源の状態がシミュレートによって変化した場合、該当する表を変更していく。

また、デバッグ機能はこの表に対して参照、変更の依頼をすることによって実現される（図１）。

次章では、デバッガ・オブジェクトを実行制御部として実現している。

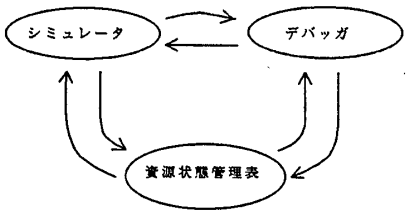

図１ 本方式の基本モデル

4．構成

図２は基本モデルに沿った本方式の構成図である。アプリケーション・プログラムからシステムコールを通して本システムとのやりとりが行われる。インターフェース解析部、システムコール・シミュレート部、資源状態管理表群、ハードウェア・シミュレート部、実行制御部に分かれている。こうすることにより各部の独立性が上がり、実現が行い易くなる。

（１）インターフェース解析部

ここでは、アプリケーション・プログラムから渡されるパラメータを解析し、目的のシステムコールを判別する。また、呼びだし回数をカウントしたりといった実行制御部で必要となるデータを取得し、実行制御部あるいはシステムコール・シミュレート部へ制御を渡す。

（２）システムコール・シミュレート部

ここは、各システムコールをシミュレートするモジュールで構成されるが、必要に応じて資源管理表群の該当する管理表にデータを書き込み、逆にそこに書いてあるデータを読んでシミュレーションに役立てる。

（３）資源状態管理表群

ここでは、ＯＳ内で使用される管理表がはいる。例えば、メモリ管理表、タスク管理表、ＩＯデバイスの使用状態、ハードウェアコンポーネントの内部状態などが含まれる。これらの表は、システムコール・シミュレート部、ハードウェア・シミュレート部、実行制御部から直接アクセス（読み書き）される。ユーザからは、実行制御部のヒューマン・インターフェースを通してアクセスされることになる。

（４）ハードウェア・シミュレート部

ここでは、ハードウェア環境のシミュレーションを行う。ここは、システムコール・シミュレーション部と関係しており、各ハードウェア・コンポーネントの状態遷移やＩＯデバイスのシミュレートなどを行う。シミュレートを行った結果は、資源状態管理表群の該当する表に書き込まれる。

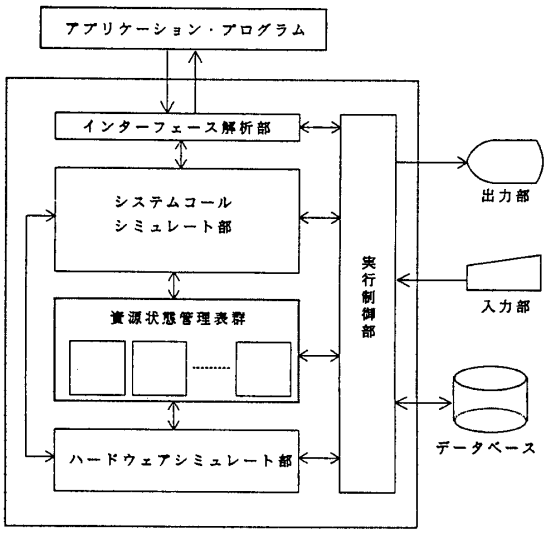

図２　本方式の構成

ハードウェア構成要素の各コンポーネントはオブジェクトとしてモデル化される。このオブジェクトへの問い合わせによってシミュレートが行える。また、実時間性の問題に対してはシミュレートする時間を予めデータベースとして登録しておきブレークポイント間の実行時間を算出するか、ユーザとの会話によって設定できるようにする。なお、このオブジェクト部分が実現されない場合でもユーザとの会話により疑似的にシミュレートが行える。

（５）実行制御部

ここは、本システム内での、実行を制御する部分であり、ユーザとのヒューマン・インターフェースをも含んでいる。ユーザからのコマンドによりブレークポイントまでの実行や各資源の状態の表示なども行えデバッガの役割も果たす。

5．応用例

本方式によると、インターフェース解析部と実行制御部のみでもユーザとの会話やパラメータ登録によって疑似実行が行える。さらに、テストを行う場合には、テストデータをデータベースに蓄積しておくことで自動的に行えるようにすることも可能である。

6．おわりに

クロス環境上における組み込み用アプリケーションプログラムのデバッグを支援する汎用的な方式を提案し、その構成法について述べた。ここでは、シミュレート部、デバッグ部、状態管理表部のモデルに従って実現された。本方式によって、システムコールを用いているアプリケーションプログラムをクロス環境上でデバッグできるようになる。また、従来では困難となっていたハードウェアのシミュレートについても実現できない部分はユーザとの会話により疑似的にシミュレートする事ができ、デバッグの中断を減らすことができる。

【参考文献】

［１］大筆他：ＩＭＡＰシステム(1)～(10)，情報処理学会第３１回全国大会講演論文集(I)，4F-1～10，1985.9.

［２］遠城：異種ＯＳ上におけるＵＮＩＸインターフェースの実現法，情報処理学会第３３回全国大会講演論文集(I)，2V-2，1986.9.

［３］松本，金田：ＯＳインタフェース整合のための仮想化実現方式の一検討，情報処理学会第３９回全国大会講演論文集(II)，6M-6，1989.9.