OSの開発環境について

2R-6

今井 徹　岡本 利夫　前田 賢一　斎藤 光男
(株) 東芝 総合研究所

1．はじめに

一般に新しいアーキテクチャをもつ計算機のOSを開発するのにはかなりの労力を必要とする。これはOSがハードウェア資源を効率よく使用し、かつユーザからハードウェア依存部を隠蔽するためのソフトウェアであるため、本質的にハードウェアに依存せざるを得ない性格をもつことや、インタラプト、トラップ等の非同期処理の再現が難しく、デバッグが困難であることが一因であると考えられる。

このため、例えばトレースして得たログを解析するといった方式で開発が行なわれることが多かったが、効率のよい開発には、より強力な開発環境による支援が必要であると思われる。

本稿では、当社で開発したAIP(AI処理強化プロセッサ)[2][3]にUNIX*(4.2BSD)を移植した際の手法について報告し、OSの開発環境について考察する。

2．必要とされる開発環境

(1)カーネルの開発に必要な機能をもったデバッガを擁すること

カーネル開発のデバッガには、以下のような機能が必要である。

a．MMU、キャッシュ、割込みレジスタなどのハードウェアリソースの制御ができること

b．任意の場所でのカーネル動作の停止、再実行が容易にできること

c．複数のプロセスをデバッグ対象とできること

d．デバッグ対象のOSの暴走によりデバッガが破壊されないこと

e．ソースレベルでのデバッグができること

a,b は、カーネルがこれらのリソースを扱うため必要な機能であり、仮想記憶、割込みなどのデバッグで使用される。

c は、OSが複数のプロセスをサービスするため必要な機能である。

d は、開発中のカーネルは暴走することがあるからである。

e は、開発効率を高めるために必要である。

(2)開発環境を提供するマシンが、十分なソフトウェア環境をもっていること

従来カーネルの開発には簡単なモニタを搭載して行なうことが多かったが、開発環境を提供するマシンが、十分なソフトウェア環境をもっており、エディタなどのユーティリティと統合的に結合していなければ、ふさわしい環境であるとは言えない。

3．AIP上のOSの開発環境

以上の点を踏まえ、つぎのような開発環境を構築した。

まず、AIPを VMEバスで AS3000 と結合し、バックエンドプロセッサとして動作するようにした。そして、ホストである AS3000 からAIP上のカーネルのデバッグを行う、クロスデバッガを開発した。これは、セルフデバッグでは、デバッガの破壊の恐れがあったり、OSの動作していないマシンでは十分な開発環境が得られないためである。

このデバッガは UNIX 上のシンボリックデバッガである dbx を改造して作成されており[1]、アプリケーションと全く同じインタフェースでカーネルのデバッグができる。

また、バックエンドプロセッサのI/O要求はホストを介して実現されるが、この要求を処理するための、ホスト上のI/Oプロセスの機能をデバッガが代行することにした。これにより、AIP上カーネルのデバイスドライバのデバッグが容易になり、またI/O空間の割当ての前のデバッグ用メッセージの出力を容易に実現することができた。

Development Environment for OS
Toru IMAI,Toshio OKAMOTO,Ken-ichi MAEDA, Mitsuo SAITO
TOSHIBA Corporation

* UNIXはATTが開発しライセンスしているOSです

しかし dbx には、カーネルしか制御のできないハードウェアリソースを扱う機能が備わっていないため、ハードウェアデバッガ[4]を開発し、機能の補強を行った。

これにより、MMU、キャッシュやシステムレジスタのモニタリングや、マイクロプログラムレベルのデバッグ、割込みの制御、物理アドレスによるメモリ参照などが可能となった。

このようなデバッグ方式を採ることにより、例えば次のようにカーネルの各機能のテストを行うことができた。

・仮想記憶

MMUをモニタすることにより、ページの割り当て、解放のデバッグをすることができる。特にページアウトやスワップは、通常の稼動時にはメモリの使用率が上がらないと起こらないが、ハードウェアデバッガによりMMUを操作して、多くのページを使用している状態にすることにより、人為的にページアウトやスワップを起こすことができる。

・割込み

AIPでは、割込みが起こると、マイクロプログラムによりレジスタ等のリソースの退避が行われる。従って、マイクロプログラムの割込み処理部にブレークポイントをかけることにより、非同期におこる割込みを捕捉してデバッグすることができる。

また、特にユーザプロセスのデバッグ時のような、低い割込みレベルでのデバッグをしている時は、dbxでブレークポイントを設け、止まった後の再実行の際にステップ実行をすると、タイマ割込みがかかっているため、割込み処理ルーチンに飛んでしまう。このような場合に、割込み許可レジスタを操作することにより、一時的に、また特定の割込みのみを禁じることができる。

・キャッシュの無効化

AIPはライトバック方式のキャッシュを採用しているため、ページアウトや、I/Oの際には、該当アドレスのキャッシュをライトバックして無効化しなければならない。

このデバッグのためには、キャッシュにあるデータの属性を保持しているタグの値を見る機能や、メモリを物理アドレスで見る機能が有効である。

これら2つのデバッガはいずれもホスト上で動作するため、 AS3000のソフトウェア環境を生かして開発を行うことができる。

4．おわりに

本稿ではOS開発の一手法として、クロスデバッグ方式によるカーネルの開発環境について述べた。

今回開発したデバッガでは、カーネルはソースレベルでデバッグできるが、その他のプロセスは機械語レベルのデバッグとなっている。これは、dbxはデバッグ対象のプログラムのシンボルテーブルからデバッグの情報を得て、子プロセスとして実行するのに対し、カーネルはホスト上のdbxに無関係にユーザプロセスを実行するため、dbxからはシンボルテーブルが読めないからである。

dbxの枠組みを変えずにホストからユーザプロセスをシンボリックデバッグするためには、カーネルがプロセスをexecする時にシンボルテーブルを読み、ホストのdbxに伝達する必要があるが、この方法ではカーネルのexecの動作を変更するため、実際のカーネルの動作をデバッグしているとは言えない。

クロスデバッグでのシンボリックデバッグの方式が今後の課題である。

参考文献

[1]今井他：AIP－dbxの実現法 情報処理学会第37回全国大会(1988)

[2]斎藤他：AIワークステーション(WINE)の開発1－7 情報処理学会第35回全国大会(1987)

[3]斎藤他：AIワークステーションの開発思想 人工知能学会第1回全国大会(1987)

[4]前田他：新しいプロセッサのためのクロス統合デバッガ 情報処理学会第39回全国大会(1989)