学習支援のためのOSの可視化ツールの設計と実現

2F－7

伊藤能康、早川栄一、並木美太郎、高橋延匡

（東京農工大学　工学部）

1. はじめに

OSの動作を理解することは容易なことでない。なぜなら、OSは非同期な割込みによって動作しているために動作を追跡しにくいからである。また現行のOSは多機能化しソースコードレベルで概略をつかむことはほとんど不可能である。そこで我々はOSの動作の理解を助けるために、より理解しやすい簡単なOSを可視化し、OSの学習を支援することにした。

本報告では、OSの学習を支援するための可視化ツールの設計と実現について述べる。

2. 設計方針

学生にOSを理解させるための可視化ツールの設計方針について次に述べる。

(1) 利用方針

本ツールの対象者はOSの基本的な概念をすでに学んだ学生とする。また、学習効果について、頭だけでOSの動作を追跡することは困難であり、実際に学習者が理解しようとするOSを利用しながら学習させる。

(2) 可視化方針

表示項目として、OSの管理するハードウェアリソース、プログラムモジュールとOSの管理構造を示す。表示はOSとアプリケーション（AP）の動作原因と結果をアニメーション処理を用いて示す。表示は必要最小限のものにとどめ、学習者がOS内のソースコードとの対応を理解できるようにする。ユーザインタフェース（UI）は、理解を助けるために、必要な時点での動作停止、動作速度の変更、見たい項目だけの表示選択などの可視化動作の変更を取り入れる。

(3) 実装方針

可視化する方法として参考文献[1]のようなシミュレータを用いたり、可視化対象OS自体を書き換える方法もあるが、我々は対象OSとは独立したツールを作成し、可視化する方法をとることにした。これは対象OS内に可視化用のコードが埋め込まれて、理解の妨げになることを防ぐためである。また、OSが変更された時のツールの変更を最小にする。

3. 可視化対象OS

可視化対象OSとして、本研究室内で開発されたOS「礎石」を可視化対象OSとした。なぜなら次の条件を満たしているからである。

(a) 多機能でなく、動作が複雑ではないが、OSの機能としてタスク管理、排他制御、タスク間通信、コンソール入出力など、OSの学習において必要十分な機能が実装されている。

(b) ソースリストのほとんどの部分が日本語識別子を用いた言語Cで記述されており、量的にも1000行以内で学生でも解読可能である。また、AP開発環境も整えられている。

4. 設計

4.1 可視化の設計

(1) リソースの表示

OSによって管理され、ユーザに見えている資源を図示する。なお、これは実際には図Aの①である。

(2) プログラムモジュールの表示

管理機構との対応を取るためにOSとロード中のモジュール名称を図示する。これは図Aの②に示されている。

(3) リストの表示

リスト構造は矩形とポインタを示す矢印に抽象化して示す。また、プログラムモジュールとタスクリストとの対応がわかるように関係を示す。この他、OS内の管理機構は図Aの③の領域に示されている。

(4) OS動作のアニメーション表示

今回の可視化対象OSである礎石のタスク管理リストとセマフォリスト、メッセージキューの更新の様子をアニメーション処理を利用して示す。アニメーションは要因と結果の対応が取れるように残像を残す。

(5) ソースコードとの対応

子タスクはAPのどの関数によっているのかをモジュールのシンボル名を用いて示す。

(6) OSの変更への対応

SVCの発生する前後の状態を比較して、どこが、どのように更新されたのかを認識させて、OSが変更されても情報取得のメカニズムさえ動作可能であればツール側を変更する必要をなくす。またソースコードレベルでOSに依存する部分と依存しない部分を分離する。

(7) 可視化動作の変更

任意時にアニメーション速度を変更したり、必要最低限の情報だけを表示する表示選択インタフェースを

Design and Implementation of OS Visualizer for Learning Support

ITOH Yoshiyasu, HAYAKAWA Eiichi, NAMIKI Mitarou and TAKAHASHI Nobumasa

Faculty of Technology, Tokyo University of Agriculture and Technology

組み込む。

4.2 実現に必要な機構

ツールの実現には次のような機構が必要である。

(1) 割込み・例外・特権命令のトラップと通知

(2) 可視化対象 OS の存在するメモリ領域の参照

これは OS の管理リストやキューを取得するために必要である。

(3) 対象 OS の実行コンテキストの参照 / 変更

SVC などの引数の取得や可視化動作変更時に必要である。

(4) 可視化ツールの処理落ちの回避

(5) 時間の整合性の保持

ツール側で OS 上の SVC や割込みを検知し、OS が変更した管理テーブルを取得できる必要がある。また可視化中に OS 上の割込みを取りこぼしたり、ツールの実行によって OS の動作が変わってはいけない。

そこで我々は本研究室で開発された、仮想機械を提供するハイパ OS「忍」[2] を用いて実現を行った。実現環境を次図に示す。

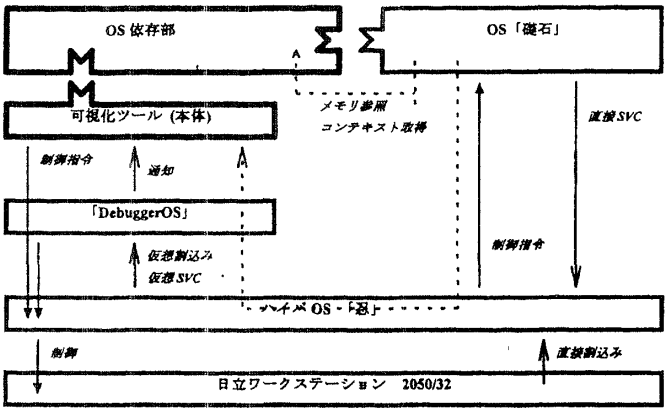

図. 全体構成

5. 実現

5.1 実現規模

前述した環境下で可視化ツールを実現した。実現規模は OS 非依存部が 3250 行、OS 依存部が 5660 行である。

5.2 実行画面

実現した可視化ツールの実行画面を示す。

(A) タスク生成 SVC 発行時には、この後生成パケットが待ちリストに繋がれるアニメーションが行われる。

(B) セマフォ初期化 SVC 発行を示すアニメーション後に、新規セマフォが示される。

(C) メッセージ受信 SVC の発行後、OS 内のメッセージキューから該当メッセージが消える。

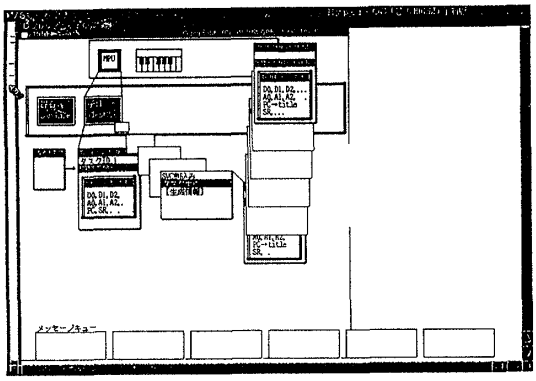

図 A. タスク管理

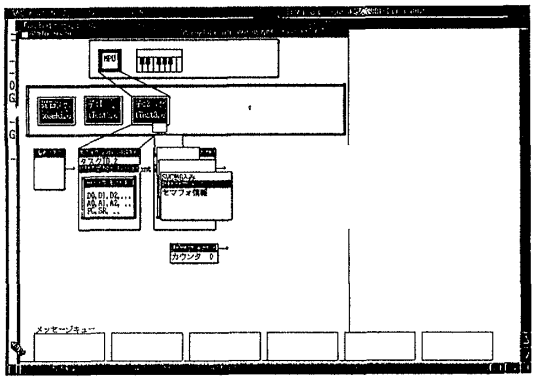

図 B. セマフォ管理

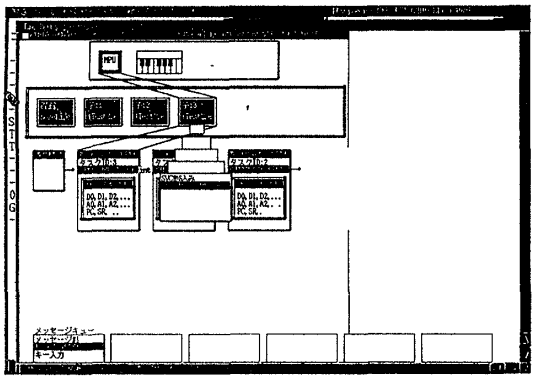

図 C. タスク間通信管理

6. おわりに

本報告では、OS の理解を助けるために OS の動作と管理機構を可視化するツールの設計と、仮想機械を用いた実現について述べた。今後の課題は学生に本ツールを実際に利用してもらい、OS の学習効果や UI などについての評価することである。

参考文献

[1] 下山他:並列プログラミングの実行の可視化，情報処理学会第 46 回全国大会 講演論文集，7J-06,1993

[2] 山本他 : OS デバッグ支援のためのハイパ OS「忍」第 2 版の設計と実現，情報処理学会第 50 回全国大会 講演論文集，3H-7,1995