システムソフトウェアと オペレーティング・システム 74－30
マルチメディア通信と分散処理 81－30
（1997. 2. 28）

学習支援のためのOSの可視化ツールの設計と実現

伊藤能康 早川栄一 並木美太郎 高橋延匡

東京農工大学大学院

本報告では、OS の動作・機能・実装などの学習を支援するための、OS の可視化ツールについて述べる。オペレーティングシステム（OS）はコンピュータサイエンスを学ぶ学生が理解すべき重要なソフトウェアであるが、OS はその動作の追跡が困難であるため、理解することは簡単ではない。本ツールは、OS の動作とアプリケーション（AP）の動作の対応、OS と AP のシンボル情報の活用、OS 動作の原因・結果のアニメーション表示、可視化目的に応じた表示の変更と OS コードの変更への対応などを考慮に入れて設計を行い、実行環境として可視化対象 OS とツールの環境を分離するために仮想機械を提供するハイパ OS「忍」を用いて実現を行った。また本ツールを実際に授業に用いて簡単な評価を行い、OS の理解に有効であることを確認できた。

A Design and Implementation of OS Visualizer for Learning Support

ITOH Yoshiyasu, HAYAKAWA Eiichi, NAMIKI Mirarou, TAKAHASHI Nobumasa

Tokyo University of Agriculure and Technology

In this paper we describe Operating System (OS) visualizer to support understanding OS activities, facilities and implementations for studying computer science. But it's very difficult to understand them. This tool's characteristics are following, relationship between OS's and application (AP)' s activities, applying to symbol information, animations and capacity for OS module modifying. We used the Hyper-OS "SHINOBI" providing virtual machines for separating out one target OS and our tool. We have evaluated this tool in our OS course and confirmed effect of OS understanding.

1. はじめに

オペレーティングシステム（OS）はコンピュータサイエンスを学ぶ我々が、その動作や仕組み、機構などを理解していなければならない基本ソフトウェアの一つである。しかしながら、OS の動作など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ことと、特権状態や割込みレベルなどの CPU の状態などに直接影響を受けるために動作を追跡しにくく、アプリケーションの影に隠れているために、いつ、どこで、何をやっているのかが見えないためである。したがって、学生は OS の動作の概要をつかむことさえ難しい。さらに現行の OS は多機能化し、OS の実現上の機構や動作をソースコード上で追跡することは非常な労力を要するようになっている。そこで我々は OS の動作や構造の理解を助けるために、より理解しやすい簡単な OS について可視化を行うことにした。

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. 利用方針

これから我々の実現した可視化ツールについて述べていくが、可視化ツールをどのような形態で利用するのかをはじめに述べる。

(1) 対象者

可視化を行い、OS の学習を支援すべき対象者は、OS の基本的な概念や動作原理、基本的な機能などをすでに学んだ学生とする。可視化ツールは OS やアプリケーション（AP）のソースコードが与えられたときに、ソースコードの記述が実際の動作にどのように影響するかについての理解を深めるために利用することを考える。具体的には、東京農工大学電子情報工学科コンピュータサイエンスコース３年次の学生を対象とする。

(2) 利用形態

まず OS を可視化した際の学習効果について考えると、可視化された表示だけを見ていたのでは、頭だけで OS の動作を理解しようとしていることになり、学習の効果が上がらない。そこで我々は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ことにする。また可視化は実 OS を対象とする方針とする。これは実 OS であればソースコードを学習者が書き換えて、その実際の動作を確認することができるからである。

3. 全体方針

3.1 実装方法

OS の可視化を実装する方法としては次のような方法が考えられる。

(1) OS シミュレータの作成

(2) 対象 OS の書換え

(3) OS とは独立したツールの作成

(1) は既存の OS 上に可視化したい OS のシミュレータを構築し、その動作を別画面に表示させる形態であるが、これは OS を擬似的に動作させることになり、2.（2）で述べたように、学習者が OS 自体を変更してそれを可視化する、といった用途には利用できない。(2) の方法は OS 自体の可視化したい部分に可視化用の手続き呼出しを埋め込むなどする [1] ことになるが、これでは OS のソースコードが可視化用に書き換えられてしまい、学習者の OS の書換えに対応できないだけでなく、可視化のために埋め込まれたコードが理解を妨げることが考えられる。そこで我々は (3) の学習するべき OS とはまったく独立した可視化ツールを作成する方法を採ることにした。

ツールの実装上の方針は次のとおりである。

(a) OS が変更されたときのツールの変更を最小にする

可視化対象 OS を自由に変更して動作がどのように変わるかを見たいが、そのときツールの変更はなるべくしないで済むようにする。

(b) 可視化対象 OS 内の時間は正しくする

可視化対象 OS がタイムスライス型に変更しても問題がなるべく生じることがないようにする。

3.2 可視化対象 OS

OS の学習させる時に、どのような OS を学生に対して示せば OS の理解が深まるかについて考察し、次に示す条件を満たしていれば、OS の学習に利用できると考えた。

(1) 多機能でなく、動作が複雑でないこと

(2) OS の持つべき基本的な機能が実装されて

いること
(3) ソースリストを自由に利用 / 変更できること
(4) AP を自由に記述し、利用できること

そこで条件を満たす OS として、筆者の所属する東京農工大学工学部電子情報工学科高橋・並木研究室内で開発された OS「礎石」を可視化対象 OS をすることとした。礎石は上記の条件を次のように満たしている。

(a) 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の、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 行程度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. 可視化ツールの設計方針

4.1 可視化すべきこと

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。管理構造は礎石の場合、タスク管理リスト・セマフォリスト・メッセージキューである。

4.2 可視化方針

本可視化ツールの設計方針は OS の動作や構造の理解のしやすさを考慮して、次のようにした。

(1) OS と AP の動作原因と結果を示す

OS は AP 内のスーパバイザコール (SVC) やハードウェアからの非同期な割込みによって動作を開始し、その結果が OS や AP に返される。その様子を可視化して見せる。

(2) 理解に必要な情報以外はあえて見せないで、わかりやすい形式にする

例えば CPU コンテキストの格納は、格納されていることがわかればよく、その値までは必要としない。またアドレスは即値を見せずに関数名などを見せる。また割込み禁止中なども視覚的に示す。

(3) OS の動作をアニメーションを利用して可視化し、理解を助ける

ただ紙芝居のような動きのないものよりも、動きのあるものを用いた方が、どこがどのような過程を経て変更されたのかが把握しやすいため、理解しやすく学習効果が上がる。

(4) 学習対象者が利用しやすいユーザインタフェース (UI) を提供する

可視化動作の速度が速すぎたり、余計な情報を表示しないようにして、学習時に混乱しないようにする。

5. 可視化ツールの設計

前述の設計方針にしたがって可視化ツールの設計を行った。

5.1 表示の設計

(1) ハードウェアリスト

OS の監視するハードウェアとしてキーボード・タイマと、動作中のタスクなどを示す時に利用する CPU を示す。また割込み禁止中であることを示すアイコンを表示する。

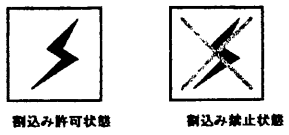


図 5.1(1) 割込み状態表示

(2) プログラムモジュール

プログラムモジュールと OS のモジュール、さらにタスク内で生成された子タスクの存在を示す。これは現在実行中のプロセスを示すために利用する。

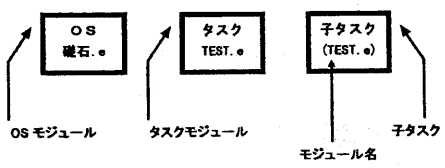


図 5.1(2) モジュール表示

(3) リスト構造

リスト構造は、リストの先頭とリストに繋がれた内容をポインタを示す矢印を用いて示す。

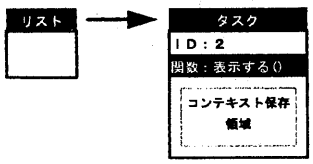


図 5.1(3) リスト表示

(4) 実行中タスク

実行中のタスクは CPU とプログラムモジュールと、それが管理されている OS 内のリスト構造を結ぶことで示す。これによって OS 内に隠れている管理機構と目に見えるモジュールとの関連がわかる。

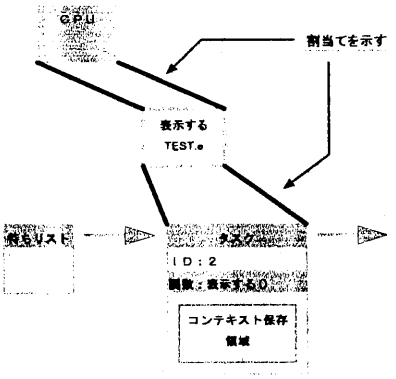


図 5.1(4) 実行中タスク

(5) 関数名の表示

関数名はタスクの開始番地をわかりやすくするために利用し、これは各モジュールのシンボル情報を利用して表示を行う。

5.2 動作表示の設計

(1) 割込みの発生

割込みの発行は発行元からズームイン表示を行うことで、原因とそれが何であるかを示す（図5.2 (1.1)）。SVC の場合は、SVC に付随する情報があれば、さらにその内容を示す（図5.2 (1.2)）。

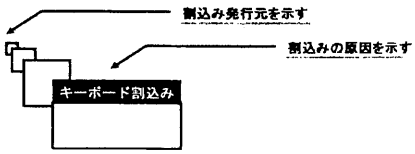


図 5.2(1.1) 割込み発生

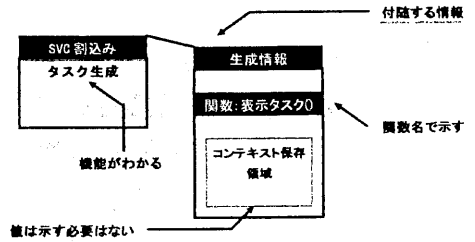


図 5.2(1.2) SVC と付随する情報

(2) リストの更新

リストの更新は移動アニメーションによって示される。アニメーションはどれが移動中であるのかを見失うことがないように、アニメーション対象の内容を表示しつつ移動させる。（図5.2 (2)）

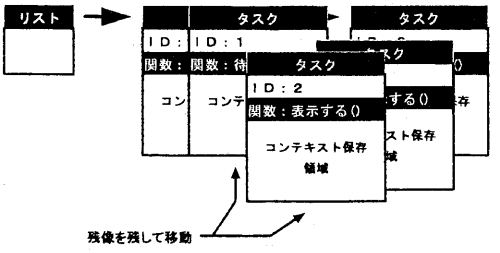


図 5.2 リンク更新

5.3 ユーザインタフェースの設計

可視化ツールのインタフェースとして、学生が理解する時間を残すためと表示させる情報を減らしてわかりやすくするために次の機能をいれる。

(1) 可視化対象 OS の停止
(2) 可視化スピードの調節
(3) OS の動作の選択表示

(3) はタスク管理だけに着目したり、ハードウェア割込みの表示は行わないなどの選択を行うことができるようにする。これらの機能は直ちに利用できるように、ファンクションキーやテンキーに割り当てる。

6. 可視化ツールの実現

6.1 実装上の問題点

(a) 磯石は子タスクもスーパバイザ (SV) モードで動作している
(b) 管理リストの一部が OS 内に定義された変数から参照できない

これらの問題点を実現によって回避した。

6.2 実現機構

可視化ツールの実現には、複数の仮想機械を提供する OS で、デバッグ用のいくつかの機能を持ったハイパ OS「忍」[2] [3] を用いた。

忍の機能は次のようなものがある。

(1) 割込み・特権命令のトラップと通知
(2) 仮想機械間のメモリ領域の参照
(3) 割込み・特権命令のキューイング

(4) 時間の整合性の保持

可視化ツールでは、SVC やキー割込みなどの検出のために (1) を利用した。特に SVC の場合は、それを検知した後、(2) を利用して OS が変更した OS 内の管理テーブルを参照して、表示を更新する。特権命令のトラップは礎石の実装からくる制限によるもので、これによって OS 内の処理からユーザタスクに処理が移行したことを検出する。(3) は可視化中に対象 OS 上での他の割込みを取りこぼさないようにするために必要な機能である。(4) がないと、対象 OS の時間を可視化ツールの処理によってずらしてしまい、例えば OS がタイムスライス型のタスク・スケジューリングを行った場合にタイマ割込みの間隔が保てなくなる。

実現時の可視化ツールと可視化対象 OS の情報のやりとりの概要を図 6.2 に示す。

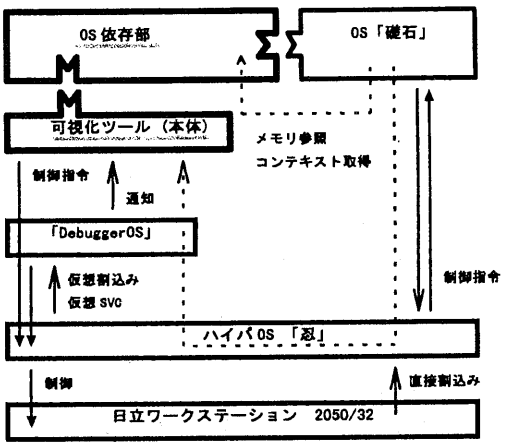


図 6.2 全体構成

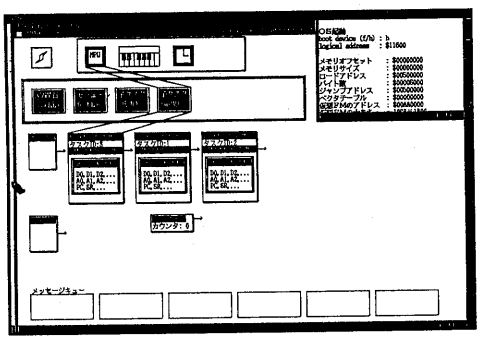
6.3 実現規模

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。

6.4 実行画面

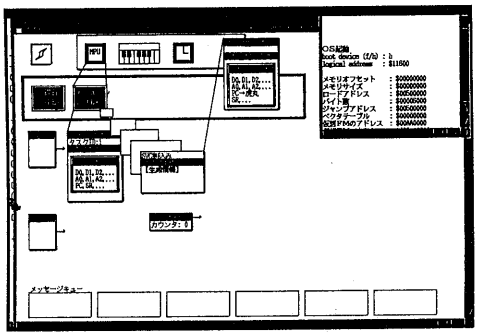
実現した可視化ツールの実行画面を示す。①はハードウェアリストを示し、②はプログラムモジュール、③・④はそれぞれ待ちリスト・停止リスト、⑤はセマフォ、⑥はメッセージキューを示している。

(A) 三つのタスクがあり、その一つのタスクが実行中で、OS 内の待ちリストの先頭に繋がれている様子が表示されている。



タスク実行中の表示

(B) タスク生成 SVC が親タスクで発行された瞬間を表示しており、SVC の内容とそれに付随する情報が示されている。この後、付随情報は移動アニメーションによって待ちリストに繋がれる表示がなされる。



タスク生成の瞬間

7. 可視化ツールの評価

本ツールのデモンストレーションを OS 演習の講義中に行い、そのアンケート結果などから本ツールの簡単な評価を行った。OS 演習では OS の概念・礎石などの講義が行われ、礎石のソースリストが配布されて、それに機能を追加する課題が与えられた。

7.1 アンケート内容

アンケートの内容は次の二種類に大別できる。

(1) 被験者の経験の調査

まず、被験者がどの程度の能力を持っているのかを調査するために次のような質問を行った。

(A) 言語 C によるプログラミングの経験はどの程度か
(B) 礎石のソースリストの解読率
(C) 礎石の自己理解度

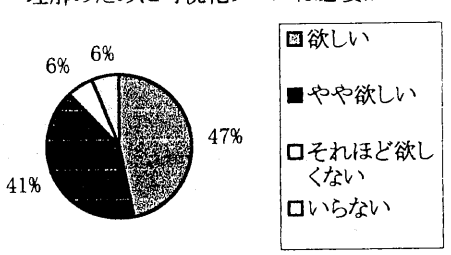
(2) ツールの有用性の調査

次に、被験者がツールを用いて可視化された OS の動作をみて、ツールの有用性があるかどうかを調査した。

(D) ツールによって理解できた点はなにか
(E) ツールによって理解できなかった点はなにか
(F) 本ツールに欲しい機能はなにか
(G) 理解のためにツールが必要であると思うか

7.2 アンケート結果と考察

本可視化ツールを必要と感じるかどうかは次のグラフで示すとおりである。20 人中で 9 割近くの学生が本可視化ツールがあったほうが理解しやすいと考えていることがわかる。また理解できた点とできなかった点はつぎのようなものがあった。



(A) 理解できた点
- 礎石の動き
- 礎石の処理順序
- タスク切替えの動き他

(B) 理解できなかった点
- 今の動きが礎石のソース上のどこが担当しているのか
- タスクコントロールブロック (TCB) への退避方法他

可視化ツールに欲しい機能としては次があった。
- ソースリストのどこにいるのかを知る
- 今までの動きを保存しておく

今回の評価は非常に簡単なものであったが、上記 (A) (B) を見ると礎石の動作を理解することができ、逆にソースリストとの対応が理解できなかったということが言える。全般的な学生からの反応として、自分の目には見えない OS の動作が見ることができることは、OS の理解に有効であるということが確認できた。

8. おわりに

本報告では、コンピュータサイエンスを学ぶ学生が OS の動作や管理機構を理解するための可視化ツールの設計、仮想機械を用いた実現と、可視化ツールの簡単な評価について述べた。この結果、学生が OS を理解するときに利用できた方がわかりやすいという結果を得、本ツールの有用性を確認できた。今後の課題は、学生からの要望のあった詳細な動作追跡の実現と、OS の学習効果や UI についてのより詳細な評価である。これらの課題を解決し、学生の OS の理解に本ツールを役立てていきたい。

参考文献

[1] 野口他：OS 動作の可視化機能の設計，情報処理学会コンピュータシステム・シンポジウム論文集 Vol.96, No.7, 1996
[2] 清水他：OS デバッグ環境の考察と OS デバッグ用ハイパ OS の実現，情報処理学会第 34 回プログラミングシンポジウム報告集，1G-7, 1993
[3] 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
[4] 日本モトローラ株式会社編集：MC68030 ユーザーズ・マニュアル，日本モトローラ株式会社 半導体事業部，1990