システムソフトウェアと
オペレーティング・システム 84−24
（２０００．５．２５）

可視化によるOS基本機能の学習支援システムの開発

西野洋介，早川栄一，高橋延匡
拓殖大学大学院 工学研究科 電子情報工学専攻

計算機科学を学ぶ学生にとってOSの理解は必須である。しかし従来のテキスト、講義による学習では、OSの動作が見えず、非同期で実行が行われるので、学習者はOSの動作と構造の対応がイメージしにくい。これに対して、シミュレータをベースとしたOSの可視化システムによって、OSの機能、構造を表示し、学習支援を行う環境を提供することで、学習者の理解を促進させる。本システムでは、OSの概念から実装段階の各段階に応じて支援を行い、また学習者が実際にOSのソースコードを書換えと可視化を行うことで、学習者のレベルに応じた学習を行うことを可能にする。我々は、プロセススケジューリングについての可視化機構を試作し、その有効性を確認した。

Development of a Visualization Environment to Learn Fundamental Operating System Facilities

Yosuke Nishino, Eiichi Hayakawa and Nobumasa Takahashi
Takushoku University

This paper describes an operating system (OS) visualization system that supports learning fundamental OS facilities. Learning operating system is required in computer science student. However, it is hard for student to understand the execution of operating system is invisible and invoked by interrupt asynchronously. For this problem OS visualization system based on simulator that is shown OS facilities and structure is presented. The system supports the student with each learning phase that is from concept learning to implementation learning depending on student's level and is rewritten OS and visualization code executing on the system by student. The prototype of process scheduling that is part of the system is implemented and is confirmed its availability.

1．はじめに

オペレーティングシステム（以下OS）はハードウェア、ソフトウェアを管理し、仮想化する最も重要なシステムである。ゆえに計算機科学を学ぶ学生にとってOSの学習、理解は最重要課題の一つである。

しかし、計算機科学を学ぶ学生がOSの概念、動作、機構などを理解するのは難しい。その大きな理由としてOSの動作が実体として見えないことにある。この問題は、OSを学習する側だけではなく、教える側にとっても、教材の作成を難しくしている。

このような問題に対して、我々は可視化を中心としたOS学習を支援する環境を提案した。この環境では、従来の計算機環境の上にCPUシミュレータを作成し、その上で学習用OS, アプリケーション(AP)を動かすことで、OSの動作を制御する。また、シミュレータを通して可視化機構を提供し、OS自体の動作や、APとOSの協調動作を可視化することを可能にする。

本稿ではまず計算機科学を学ぶ学生を対象として、OS学習におけるの教育環境と本研究の目的を述べ、OS学習の支援環境の開発について述べる。

2．本研究の目的

本研究ではOSの動作を、アニメーションを用いた具体化を行うことで、従来のテキストによるOS学習に比べ、視覚的に理解を促す事ができる学習支援環境を学生に提供することを目的とする。なおここでいう学生とは計算機科学を学ぶ学生であり、特に本学習支援環境ではOSについての知識がないOS学習の初期段階の学生から、ソースコードをある程度読め、実装レベルを学習しようとする段階の学生を対象とする。また学習項目を学習者の理解度に応じてレベルの差を設け、段階的な学習を行うことで学習者の混乱を防ぎ、理解を促進させる。

3．問題分析

ここでなぜOSの学習が難しいか、OSの学習、理解を妨げる原因である要素をあげる。

(1)OSの構造からくる問題

ここではOSそのものの構造からなる、OS学習における問題点と理解を妨げる原因をいくつか示す。

- OSそのものが巨大である
- OSが非同期で動作するので把握しにくい。
- 実行性能を上げるためにわかりにくいコードがある。
- ハードウェアとソフトウェアの知識が必要になる。

等の問題点がある。これらは学習する際にOSの構造上の問題である。特にOS学習においてOSの複雑さ、コードの汚さ、非同期な動作などがOSの理解を妨げていると考えられる。

(2)現在の計算機科学における教育環境の問題

従来のテキストによる学習や、講義による学習では、概念や実装コード例を示すことはできるが、実際にOSを操作して実験しながら学習することは難しい。

これに対して、実装が簡易なOSを用いてユーザに実装のコードを読ませたり、変更させたりするアプローチがあるが、OSは非同期な割込みによって動作するので、テキストベースによる学習では、ユーザはその動作概念がイメージできず、結果としてOSの理解を妨げる原因になってしまう。

例えばプロセス管理におけるセマフォによる排他制御機構やハードウェアに依存した処理などはテキストによる学習では状態遷移などがイメージしにくい。またOSのソースコードとOSの実行との動作対応とれず、OSを概念的には理解したが実装レベルとしての理解が得られにくい。そこで実際にOSを動かしながら、動作とOSのコードとの対応づけがなされた学習環境のニーズが生まれてくる。

また、教材を用意し、教える側にとっても、説明と実験とを関連づけるためには、制御可能でなおかつ、内部の動きをイメージづけられる可視化機構が必要となってくる。

4. 本 OS 学習支援環境のコースウェア

本学習支援環境の利用者は先にも延べたように OS について初期の学習段階にある学生から実装を学ぶ学生、また学生に OS を教育する教育者を対象としているので、OS の基本的機能からソースコードレベルの構造、解析を主な学習対象とする（図１）。

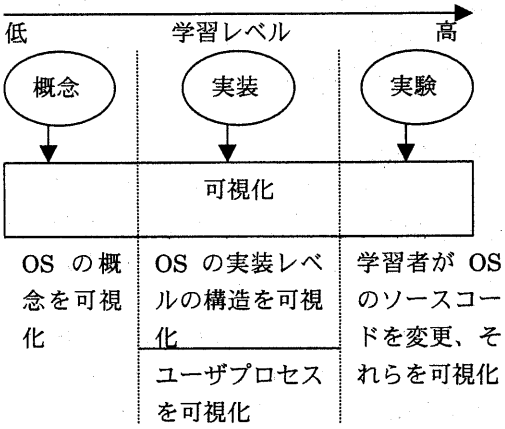

図１．本学習環境におけるコースウェア

(1)OS 学習初期段階の学生

OS 学習初期段階の学生はまず OS の概念、基本的な構造を理解する必要がある。OS の機能において最も重要な概念は仮想化による資源管理の概念である。よって学習項目も、余計な混乱を招く可能性を排除するため、複雑な内部構造や実装レベル、ソースコードレベルの要素は表示せずに、機能を抽象化した図などを用いた学習環境を提供することが望ましい。

例えば、プロセス管理については、

・CPU を仮想化したのもがプロセスの概念
・効率的にスケジュールすることであたかも複数の処理を同時に行っているように見える
・各プロセスの状態
・状態遷移の図
・セマフォの概念
・セマフォによる同期、排他制御

などを可視化する。

(2)OS の基本的概念を理解した学生

OS の基本的な機能の概念を理解した学生に対しては複雑な OS の内部構造や実装的な機構などより具体的な学習をすることになる。特に OS の学習においてはモデルの理解と同時に実現方式の理解が必須である。よって OS のソースコードを表示し、ソースコードと OS の動作の対応を可視化する。これにより学習者は直感的にソースコードの意味と実践的な OS の構造を理解できる。

またこれらの学習者に対しては OS そのものの構造を可視化したものと、OS がアプリケーション、ユーザプロセスを管理している様子を可視化したのを表示することで、OS とアプリケーションの関連についての学習環境を提供する。

プロセス管理を例にとれば、次のものが挙げられる。

- ソースコードの表示によるスケジューリングアルゴリズムの解析
- スケジューリングアルゴリズムの書換え
- コンテキストスイッチの様子
- リスト構造、チェーンのつなぎ替え

(3)実装を理解した学生

OS の内部構造、実装的な機構を理解した学生に対しては、実際に自らの手で OS を動かす実験学習、演習の段階に入る。教育者は OS に関する課題、問題を提示し、学習者は本環境を用いて試行しながら演習学習を行う。これらの学習者にはソースコードを書換えさせ、変更した後の動作を可視化し、動作の違いを直感的に理解させる。

また非同期割込みの制御のテストなどを行う際に実機でタイミングを再現することができず、テストを行うのは難しい。シミュレータを用いることで、タイミングを変えつつ、割込みに関するテストが容易になる。

(4)教材作成者への支援

OSについて説明を行う側にとっては、可視化環境をテキスト,資料とともに利用することでテキストだけの解説に比べより具体的な講義にすることができる。また、OSの実験演習においての課題作りや実験を簡単にシミュレートすることができるようになる。

5．設計方針

学生にOSの機能、概念を理解させるための可視化システムについての設計方針について述べる。

5．1 設計方針

前述の現行のOS学習における学習環境についての考察よりつぎのような設計、実装方針を立てた。実装する方針として既存のOS（Windows、Unix等）上に仮想化したプロセッサ、つまりプロセッサのシミュレータを実装し、さらにこのプロセッサのシミュレータ上で動作するOSを実装し、これらを仮想マシン（以下VM）として動作させる方法を採用した。

さらにこれらのVMとは独立してVMの動作を可視化する可視化モジュールを既存のOS上に実装し、VMと可視化モジュール間で情報の通信を行う。ここでVM、可視化モジュールを一体化し、ひとつのシステムとして構成しなかった理由は次のとおりである。

- 学習対象 OS 内に可視化に関する部分を取り除き、学習者の混乱を避ける
- 学習対象 OS が変更されたときにシステム全体の変更を最小に抑える

これらの理由により VM 部と可視化モジュール部を分離して実装する方法を採用した。

5．2 OSの可視化方針

OS を可視化する方針として、OS の構造、OS によるアプリケーションの実行の様子、各リソースの動作状況を表示する。また動作時に状態が遷移するものにおいてはアニメーションを用いて可視化する。

アニメーションを用いる利点は、直感的に動作の意味を捉えることができるまた学習効率の面を考慮して可視化された表示を見せるのではなく、実際に動作しているOSのソースコードを表示し、現在の状況をトレースしながら動作結果と照らし合わせることで学習を進める。これにより学習者はソースコードと動作の対応付けができ、ソースコードの動作の意味を直感的に理解できる。

6．全体設計

先にも延べたように学習対象 OS には独立したOSを用いる。学習対象OSには仮想ハードウェアに対するデバイスドライバがあり、プロセッサ、メモリなどハードウェアをシミュレータで代用する。それらの動作状況やOSのリソース管理状況を可視化モジュールによって表示する。

仮想プロセッサと可視化対象OS（学習対象OS）をシミュレータモジュールとし、既存のOS上でVMとして実装する。またシミュレータモジュールとは分離して可視化モジュールを既存のOS上に実装する。ユーザは可視化モジュールをインターフェースとし動作の表示やシミュレータモジュールのコントロールを行う。またOSの概念学習を終えた学生はOSのソースコードを書換え、実装することで動作の変化を可視化モジュールで可視化し、ソースコードの解析を行う。全体の構成図を(図2)に示す。

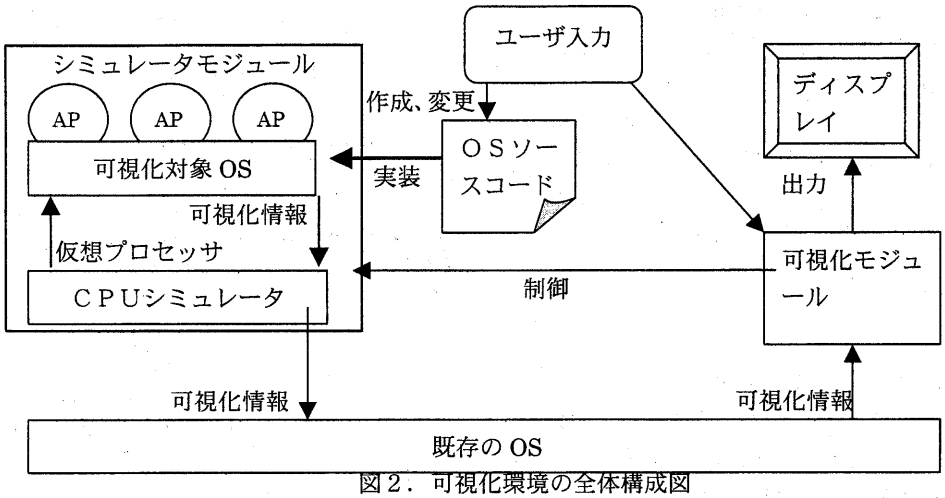

図２．可視化環境の全体構成図

６．１ プロセッサのシミュレータ設計

プロセッサには既存の RISC プロセッサ Alpha AXP アーキテクチャをベースに機能を追加したものを採用する。RISC アーキテクチャは処理を高速化するためにキャッシュやパイプラインなど複雑な処理を行っているが本可視化環境では高速な処理を求めているわけではなく、学習者が理解しやすい簡潔なアーキテクチャを求めているため実装しない。また OS の実装段階の学習になると OS の性質上必ずアセンブラ、機械語レベルの理解が必要となってくる。これらはコーディングにおいてもあまりきれいで、簡潔なものとはいえない。そこでアーキテクチャを簡潔にすることで、学習者がアセンブラ，機械語レベルにおいて余計な混乱を招くことを防ぐ。

さらにアーキテクチャの複雑な機能を簡略化することで、学習者が余計な混乱を招く可能性を除くことができること、また実際に存在するアーキテクチャなのでクロスコンパイラを使用することで新しいコンパイラ作成する手間を省くことができるという利点がある。

６．２ 学習対象 OS の設計

問題分析の結果より学習対象 OS の設計方針として次の方針を立てる。

・OS の巨大化、構造の複雑さを排除した小型かつ簡潔な構造
・効率の良い動作を求めているわけではないのでソースコードの読みやすさを重視

これらの特徴を兼ね備える。特に本学習支援環境ではユーザが自ら OS のソースコードを理解し、書換え、機能を付け加えていくことで学習者に OS の構造を理解させる学習スタイルであるので、初期状態での学習対象 OS の機能はごく最小限な機能しか実装しない。

可視化対象 OS を設計するにあたり、複雑な OS の機能の中から次の機能を実装する。

・プロセス管理、特に同期排他制御やプロセス間通信機能を実装したもの
・入出力管理
・スケジューリングなどを制御するタイマ割り込み

これら多機能ではなく、動作が複雑ではない OS を可視化対象として実現する。基本的な機能だけを備えた OS にすることで学習対象の的を絞ることができ、またソースコードの解析も容易に行うことができるからである。

6．3可視化モジュールの設計

可視化モジュールを設計するにあたりまずOSの機能からプロセス管理についての可視化部分について試作を行った。可視化機構は、Windows上でC++-Builderを用いて実装している。

実際にこのシステムを、OSの講義において用いて説明を行い、学生からアンケートを取った。

6．3．1 プロセス管理の試作

このシステムでは、ラウンドロビンで複数のプロセスを切り替えた場合の、APとOSとの関係を図示する。ユーザは新しくプロセスを作ることが可能になっている。

プロセス管理の可視化については各プロセスのスケジューリングを行う様子を、アニメーションを用いて表示している。CPUやメモリといったハードウェアを直接見せず、抽象化した箱として見せることで、プロセススケジューリングと、プロセスの状態の変化を概念的に見せるようにした。

また各プロセスの存在状態の遷移やコンテキストスイッチの表示を色分けして表示することで直感的にその状態を理解できるようにしている(図3、図4、図5)。

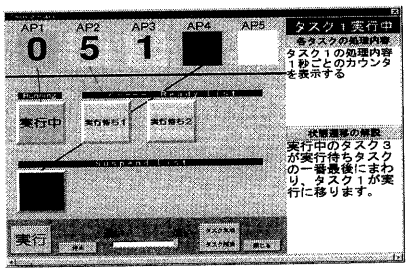

図3．プロセス管理可視化の試作画面

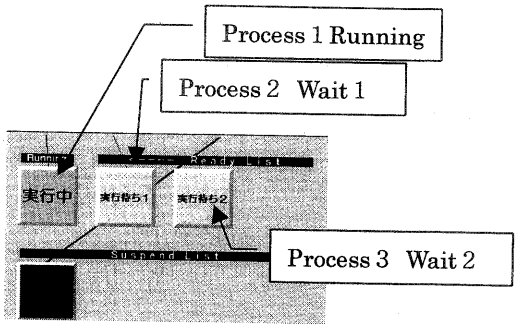

図4 プロセスの実行例

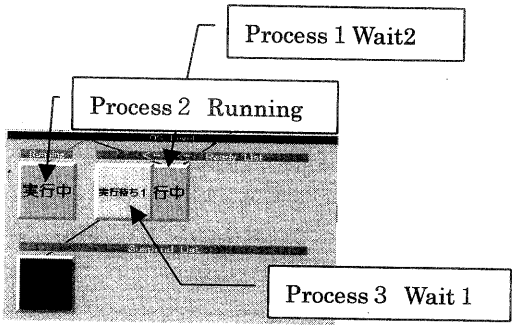

図5. ラウンドロビンスケジューリングによるプロセス管理機構の可視化画面

ラウンドロビンスケジューリングでプロセスがスケジューリングされる。

6．3．2 試作における考察

試作した可視化モジュールをOSの講義において用い、アンケートをとった結果、可視化について次の知見を得た。

1. 配色を多彩にすることで直感的な理解を得られる
2. ユーザプログラムとOSコードの動作を対応した表示にすることでユーザプロセスとOSの構造を理解できる
3. ハードウェアの状態があればOSとハードウェアの関連がわかる

6. 3. 3可視化モジュールの設計

可視化モジュールの試作における考察より可視化モジュールの設計を示す。

可視化対象 OS の動作状況を表示する可視化モジュール部について次の特徴を持たせる。ユーザインタフェースについては理解の促進を図るため必要な時点での動作停止、動作速度変更、表示項目の選択などの機能を取り入れる。

(1)OS 動作のアニメーション表示

プロセス管理におけるコンテキストスイッチ、セマフォによる排他制御の動作をアニメーションを用いて表示する。

(2)ソースコードの表示

ソースコードを表示し、動作状況と対応させてどの関数を実行しているのかをトレース表示する。

(3)ハードウェアリソースの表示

ハードウェアをアイコンもしくは抽象化した形で表示することでハードウェアの動作状況を可視化、表示する。

(4)可視化表示の操作、変更

アニメーション速度の変更や任意に情報の表示、非表示を行う。

7. 考察

ここではいくつかの既存の教育用 OS を用いて OS を学習する方法とそれらについてのいくつかの問題点を挙げる。

(1)MINIX[3]

MINIX は A.S.Tanenbaum 氏が開発した UNIX ライクな、主に OS 教育を目的とした OS である。また MINIX は、学習に適した OS と目的とするためソースコードに教科書とも言えるコメントが大に書かれている。

これらの特徴から OS の構造的問題において述べた、見やすさについては問題点をクリアしているが、学習者にとっては、それでもまだサイズが大きいと思われる。

またこの OS は実際のプロセッサ、メモリなどハードウェア上で直接動作する OS であるため、ハードウェアに依存した部分があり、ソースコードを書き換える場合にハードウェアの知識も必要となってくる。

これらの問題点があるため、これらか OS を学習しようとしている学生、または OS の基本的な概念を理解した学生には OS が巨大である、ハードウェアの知識が必要といった問題点がある。

(2) Nachos[4]

Nachos は UCB で開発された教育用の OS である。OS そのものは小型であり、ソースコードにもコメントが丁寧に書いてあるため OS 学習に向いている。Nachos 述の MINX のような実際のハードウェアを制御する OS ではなく、既存の OS（Linux, Windows など）上で動作する OS である。ユーザプログラムは、ハードウェアシミュレータを用いて動作する。Nachos では MIPS　R2/3000 のシミュレータを採用している。

Nachos 習用 OS として設計されているため、OS の機能は必要最低限の機能しか実装されていない。学習者はソースコードを解読し、OS の機能の改良，追加をしながら OS の学習を行う。しかしながらソースコードを変更する以前と変更した後の変化の様子は目に見えるものではなく、果たして改良があっているのかどうか確認する方法はない。これは MINIX にも共通する問題である。これではたとえ実践的な OS 学習が行えるとしても、動作のイメージがつきにくく、OS 学習初期段階の学生には扱いにくい。

また、OS 自体はネイティブコードで動作するので、OS と AP とが協調して動作するものについては、可視化が行いにくいという問題もある。

これに対して、本学習支援環境では、OS の動作と同時に動作の可視化を表示する特徴をもたせた。ソースコードを変更するという実践

的な学習とともに、OSの動作を可視化し、ひと目でソースコードの変更による動作の違いを理解できる学習環境を提供することが可能である。

8．おわりに

本報告では計算機科学、特にOSのプロセス管理機構を学ぶ学生が、OSの概念や動作、管理機構の学習時において可視化によって学習を支援する環境の提案について述べた。このシステムによって、OSの基本的な機能を容易に学習でき、より深く理解することが可能となる。

現在、シミュレータの実装および、OS部分の設計を行っている。今後は、これらの設計および実装を行い、実際に用いて評価を行っていく。

参考文献

[1]伊藤能康他：学習支援のための OS の可視化ツールの設計と実現：平成７年、東京農工大学

[2]日本ディジタルイクイップメント株式会社：Alpha AXP アーキテクチャ概要、１９９３年、共立出版株式会社

[3]A.S.Tanenbaum：MINIX オペレーティングシステム、１９８９年、アスキー出版局

[4]W.Chiristopher, et al : The Nachos Instructional Operating System, http://http.cs.berkeley.edu/~tea/nachos/nachos.ps