システムソフトウェアと
オペレーティング・システム 72−10
(1996. 5. 15)

OS動作の可視化機能

野口 裕介　後藤 真孝　谷口 秀夫　牛島 和夫

九州大学工学部

〒812 福岡県福岡市東区箱崎 6-10-1

TEL: 092-642-3867

E-mail: noguchi,gotoma,tani,ushijima@csce.kyushu-u.ac.jp

あらまし　本稿では、オペレーティングシステムの処理内容を可視化する機能について述べる。この機能で得られる情報を用いることで、オペレーティングシステムの改版や部分開発を効率良く行なうことが可能となる。可視化の実現には、可視化内容、効果的な可視化のための時計機能、可視化に有効なプログラム構造という3つの課題がある。可視化内容の種類として、動作の可視化と状態の可視化という2つの分類を示す。次に、効果的な可視化のために必要な時計の機能を提案する。最後に、オペレーティングシステムのプログラムを処理単位毎に独立したモジュールに分割し、プログラムポインタ表と各モジュールに与える識別子を用いた呼出を行なうことが、可視化に有利であることを示す。

キーワード　オペレーティングシステム, 可視化, 時計, 識別子

Facility for Visualizing Operating System

Yusuke Noguchi, Masataka Goto, Hideo Taniguchi and Kazuo Ushijima

Faculty of Engineering, Kyushu University

〒812 6-10-1 Hakozaki, Higashi-ku, Fukuoka, Japan

TEL: 092-642-3867

E-mail: noguchi,gotoma,tani,ushijima@csce.kyushu-u.ac.jp

Abstract　In this paper, we describe facility for visualizing operating system structure and behaviors. By using this facility, we can get information to revise an operating system and to make a partial development of an operating system efficiently. To realize the visualization, we have three issues; contents of visualization, clock facility for effective visualization and program structure. First, we explain two contents of visualization, behavior and state. Second, for effective visualization we propose what are needed as clock facility. Finally, we describe the way to divide operating system program into independent parts per processing and to call those parts by using a program-pointer table and their identifiers. And we show this way is effective for visualization of operating system.

key words　operating system, visualization, clock, identifier

1 はじめに

ハードウェア性能の向上は処理負荷の大きなアプリケーション(以降、APと略す)の実現を可能とした一方、これに伴いオペレーティングシステム(以降、OSと略す)に要求される機能は多様化・複雑化している。このため、OSは度々改版がなされている。また、新しいOSの開発も行なわれているが、開発コストの軽減のために、既存OSを部分的に利用している。このようにOSの改版や部分開発を行なう場合、OSの各機能間の関係や、プログラムの呼出やデータ参照関係といった動作内容を把握できれば非常に有効である。

しかしながら、既存OSのほとんどでは、ユーザーがその内部処理やデータ構造を容易に知ることはできない。この原因には大きく二つある。一つは、OS設計の時点で内部処理やデータ構造を外部に見せるということが考られていないためである。もう一つは、要求される機能の実現や開発期間の短縮に重点を置いたOSの改版が繰り返された結果、OSの処理やデータ構造の明瞭さが失われてしまったためである。

今後、OSに対して求められる機能はますます増大すると考えられる。これに応えるためにOSの改版や開発を行なう際に、OSはそのプログラム構造やデータ構造が明瞭で動作内容の把握が容易であることが望まれる。さらに、利用者がより容易に動作を把握するためには、動作を視覚的に掴めるように可視化することが有効である。

本稿では、可視化の内容、可視化に必要な時計機能、および可視化情報の容易な取得を支援するOSプログラム構造について述べる。

2 課題

OSの可視化を実現するためには、以下の課題がある。

(1) 可視化内容
  (a) 可視化内容の種類
  (b) 可視化のレベル
  (c) 取得情報項目
(2) 時計機能
(3) プログラム構造

各課題について、以降に説明する。

2.1 可視化内容

OSプログラム構造や機能を明らかにするためには“どの様な内容を可視化するか”、“どの程度詳しい可視化を行なうか”、“必要な情報は何か”を明確にする必要がある。

(a) **可視化内容の種類**

可視化する内容の種類を明らかにする。種類としては、プログラムの動作や管理表の内容などが考えられる。

(b) **可視化のレベル**

可視化する内容の程度(レベル)を明らかにする。可視化のレベルは、可視化対象の大きさとそれらの間の操作内容の詳しさから決定される。

(c) **取得情報項目**

可視化のために取得すべき情報を明らかにする。

2.2 時計機能

可視化では、処理の行なわれた時刻も重要な項目である。何を可視化するかによってそれぞれの意識する時刻は異なる。

種々の可視化を可能とするためには、可視化の種類や利用法に合わせて時刻や時間の進み方を変えることのできる時計機能が必要である。

2.3 プログラム構造

可視化に必要な情報は漏れなく取得しなければならないが、情報取得の負荷が大きくなりすぎてはならない。そのため、情報取得を完全かつ容易に行なえるプログラム構造を明らかにする。

3 対処

3.1 可視化内容

3.1.1 可視化内容の種類

プログラムは、各機能を実現している処理が、資源を利用し、相互に関連しながらサービスを提供している。ここで資源とは、プロセッサやメモリあるいはファイルなどのことである。したがって、プログラムの処理を可視化するためには、相互関係の変化を表す**動作**に関する情報と、資源の管理情報の**状態**に関する情報の二種類の情報を可視化する必要がある。つまり、可視化の種類には、以下の二種類が考えられる。

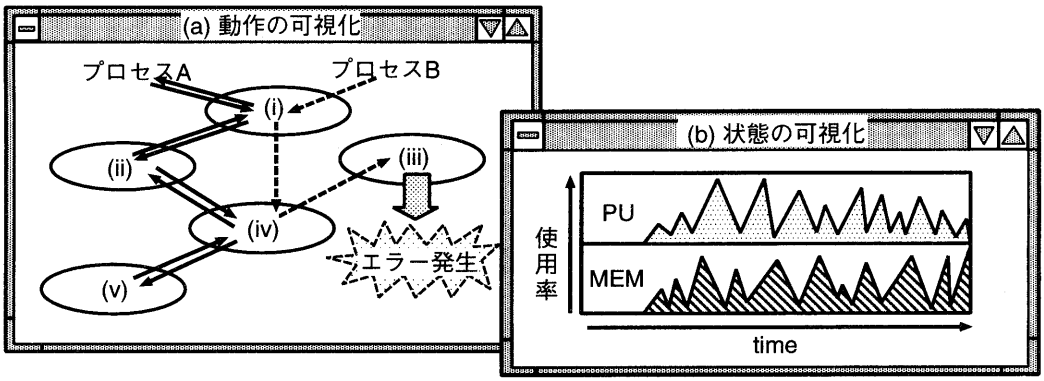

図 1 可視化のイメージ

(1) 動作の可視化
(2) 状態の可視化

それぞれの可視化について、図1に示す。

動作の可視化は、OS の内部処理の呼出関係や操作内容に関する情報の可視化である。図 1(a) 動作の可視化では、二つのプロセスからのシステムコールをうけた時の、各機能を実現している処理部分の参照関係を可視化している。ここでは、プロセス B からの要求による処理が、何らかの理由により処理部分 (iii) でエラーとなり失敗している。状態の可視化とは、呼び出された処理が実行された結果として変化した資源の状態に関する可視化である。図 1(b) 状態の可視化では、OS の内部処理の実行によるメモリとプロセッサの使用率変化を可視化している。

3.1.2 可視化のレベル

プログラムの可視化を行なう際、可視化によって得られる情報の詳しさは、可視化の “対象の大きさ” と “操作内容の詳しさ” という二つの項目から決定される。

(i) 対象の大きさ

対象とは、一つの機能の実行や他の呼出を行なう処理単位であり、「分散環境上の計算機」のように大きなものから、「一つの変数」のような細かなものまで様々な単位が考えられる。

OS プログラムの可視化の場合、対象として、OS の各機能を実現している「機能処理部分」と、さらに細かい単位の「関数」の二種類が考えられる。

(ii) 操作内容の詳しさ

操作内容とは、対象の間の呼出に関する情報のことである。これには大きく、「呼出関係」だけを取得するレベルと、「呼出関係」に加え「操作内容」も取得するレベルがある。

(iii) 対象と操作内容の組合せ

対象と操作内容に関する情報を組み合わせると、OS の可視化のレベルは以下の 4 通りが考えられる。

(1-a) **機能処理部分, 呼出関係**
(1-b) **機能処理部分, 呼出関係と操作内容**
(2-a) **関数, 呼出関係**
(2-b) **関数, 呼出関係と操作内容**

対象と操作の組み合わせを、図 2に示す。

対象の大きさは、動作記録処理の頻度と関係する。対象をより細かい**関数**として記録すると、全ての関数呼出が記録されるため、非常に詳しい動作履歴を記録することができる。しかし、動作履歴記録処理の頻度が高くなるため、記録のオーバーヘッドが増加し、それにともない処理速度が低下する。また、関数名はプログラム作成者に左右されるため、関数名からその機能を一意に決定することは困難である。さらに、動作履歴記録の処理負荷を軽減するためには、可視化の対象となるレベルでのインタフェースを統一する必要があるが、関数のレベルで統一するとプログラム構造やデータ構造の柔軟性を失わせてしまう。

操作内容の詳しさは、一回の動作記録処理の処理負荷と関係する。操作内容を**呼出関係**として記録すると、「要求があった」ということのみを記録

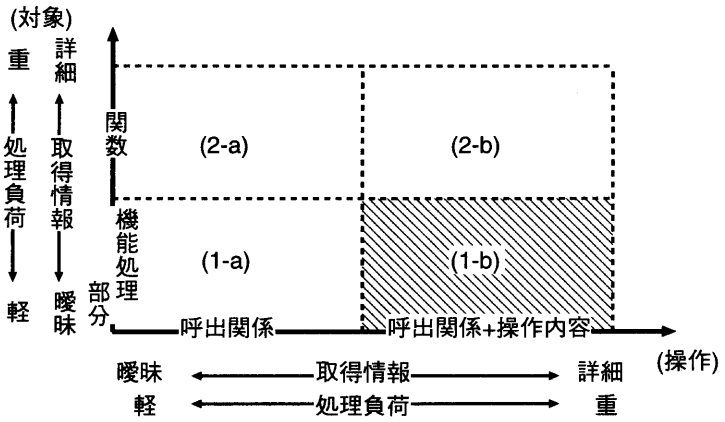

図 2 操作と対象の組み合わせ

するだけなので、一回の動作履歴記録処理の負荷は軽い。しかし、得られる情報は“対象間の呼出関係”程度であるためOS動作の可視化のために十分な情報が得られるとはいえない。一方、操作内容を**呼出関係と操作内容**として記録すると、より詳しい操作内容を把握することができるが、一回の動作履歴記録処理負荷は重い。さらに、呼出内容を効率よく記録するためには呼出の際の引数の型を一致させる必要がある。

可視化では、“何がどの様な処理を行なったか”という情報を表現する。この時、対象間の呼出関係だけでは十分に動作を表せない。そのため、「操作」としては呼出関係と操作内容までの情報を取得する。また、動作履歴記録時の処理速度と柔軟なプログラム構造を重視し、「対象」としては機能処理部分を単位とする。

3.1.3 取得情報項目

OSの可視化のために必要な情報は、可視化内容の種類と可視化のレベルから表1の様になる。

動作の可視化には、「処理部Aが処理部Bに対してCの呼出を行ない、その結果はDであった」という内容である。この時「どのプロセスに要求された処理であるのか」という表示も有効である。したがって、OS動作の可視化には以下の情報が必要となる。

(a) 操作要求元
(b) 操作要求先
(c) 操作内容
(d) 操作結果
(e) プロセス id

状態の可視化は、「資源Xは状態Yとなっている」という内容である。したがって、状態の可視化を行なうためには以下の情報が必要となる。

(i) 対象
(ii) 状態

さらに、可視化では、処理が発生した時刻も重要である。OSの動作を様々な立場から見ようとすると、単一の時刻では不都合である。具体的には、ある特定のプロセスの動作に関する動作を可視化しようとする場合、プロセスの動作速度を調節[1]できることが好ましい。これにより、可視化した時のユーザインタフェースが優れたものとなる。このような場合、そのプロセスが意識する時刻は、カーネルや他のプロセスが意識している時刻とは異なる。プロセス内で、可視化のために時刻を調節することも可能であるが、APプログラムに変更が必要であり好ましくない。したがって、可視化を実現するためには以下の二種類の時間軸に基づく時刻を取得しなければならない。

(1) カーネルの意識する時刻(カーネル時刻)
(2) プロセスの意識する時刻(プロセス時刻)

これら複数の時刻を取得するためには、カーネルやプロセス毎に独立して持たせる必要がある。

3.2 時計機能

可視化のための取得情報項目としてカーネルやそれぞれのプロセスが意識する独立した時刻が要

表 1 取得情報項目

動作の可視化	状態の可視化
(a) 操作要求元 (b) 操作要求先 (c) 操作内容 (d) 操作結果 (e) プロセス id	(i) 対象 (ii) 状態
(f) カーネル時刻 (iii) (g) プロセス時刻 (iv)	

求される。これらの時刻を取得するためには、以下の機能が時計には必要である[2]。

(a) **独立した時刻**
(b) **可変な時刻時刻進度**
(c) **時刻逆行操作の禁止**

それぞれの機能について以降に述べる。

(a) **独立した時刻**

カーネルやプロセス毎に、それぞれに独立した時刻を持たせるために、OS は、各々独立した時刻を持った複数個の時計を提供する。カーネルやプロセスは独立に時計を保持することで、関連の無いプロセスによる時刻の操作の影響を防止し、それぞれの処理が行なわれた時刻を独自に取得することが可能となる。各々の時計は、時刻の読み出しや設定、以下に述べる時刻の進み具合の設定を独立して行なうことができる。

(b) **可変な時刻進度**

実際に可視化によって得られる情報を利用する人間の立場からすると、その利用の仕方にも様々ある。処理が高速で通常では目で追うのが困難な場合は、その部分の処理速度を落とす必要がある。逆に、現在表示している部分よりももっと先の動作内容を見たい時は、その部分までの処理速度を早めることが求められる。

このような様々な利用法に対して柔軟に対応するために、可視化の処理速度に合わせて時刻の進み具合 (以降、時刻進度と呼ぶ) を自由に変化させる機能を時計に持たせる。時刻進度を変更させる機能を持たせることで、様々な利用に対して柔軟な可視化を提供することができる。

(c) **時刻逆行操作の禁止**

使用中の時計の時刻が逆行すると、得られる履歴情報に矛盾が生じ、得られる情報の信頼性が失われてしまう。信頼性のある可視化を行なうためには、時計の時刻が逆行することがあってはならない。そこで、時刻を逆行させる操作を禁止する機能を時計に持たせ、履歴情報の矛盾を防止する。

3.3 プログラム構造

可視化に必要な情報を取得し、かつ情報取得の処理負荷を軽減するためには、OS のプログラム構造に以下の性質が要求される。

(1) プログラム部品単位に分割されていること
(2) 各プログラム部品の呼出が把握できること
(3) プログラム部品の識別ができること

これらの性質を実現するために、“プログラムの独立化”、“プログラム呼出表の利用”、“識別子の付与” という特徴を持つプログラム構造を提案する。それぞれの性質について、以下に述べる。

3.3.1 プログラムの独立化

OS プログラムを “資源” と “操作内容” から決まる処理を行なうプログラム群 (プログラム部品) に部品化する。プログラム部品はその中で処理が完結し、プログラム部品間には共有関数や大域変数を持たない。プログラムを機能内容ごとに部品化することで、処理に独立性を持たせる。

さらに、扱う資源が同じプログラム部品を一つのグループ (以降、プログラム部品群と呼ぶ) にまとめる。プログラム部品群は、“資源” の管理のために必要な管理情報を共有する。プログラム部品群は、プログラム部品が要求を受けるためのインタフェースを外部に提供する。

3.3.2 プログラム呼出表の利用

プログラム部品群が提供するインタフェースを他のプログラム部品が直接呼び出すと、呼出側は特定のインタフェースの存在を常に意識することになる。そこで、インタフェースの呼出には呼出表と呼ぶプログラムポインタ表を用い、かつ、その呼出は一つの共通な呼出関数を用いる方式とする。

呼出表の構造を図 3に示す。呼出表は、“資源の

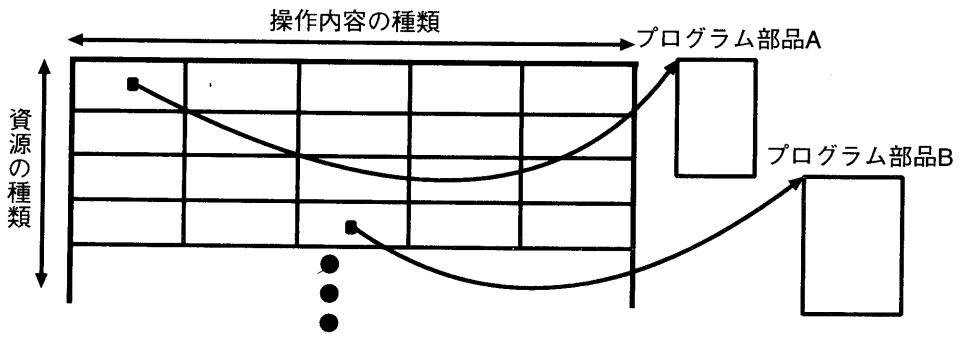

図 3 呼出表の構成

種類" と "操作内容の種類" から構成され、表の各項目は、該当するプログラム部品へのポインタを持つ。呼出元は "資源の種類" と "操作内容の種類" を指定した呼出要求を行なうことで、より抽象化された処理の呼出を行ない、プログラム部品間の独立性を促進できる。

共通な呼出関数では、呼出の記録機能を持ち、可視化に必要な履歴情報を漏れ無く取得する。

3.3.3 識別子の付与

共通な呼出関数での、各資源の種類を意識した処理を無くすため、資源に識別子を与える。さらに、各プログラム部品群の呼出には、この識別子を用いる。

これにより、呼出元の処理や呼出表の操作は、全てのプログラム部品の操作をその種類によらない統一的な操作と管理を行なうことが可能となる。また、識別子を用いることで、履歴情報の統一的な記録が可能となるため、記録のための資源の節約、および処理負荷の軽減を図ることが可能となる。

4 おわりに

稼働中の OS について、その動作内容を可視化する機能を述べた。これにより、OS の改版や部分開発を効率良く行なえるようになる。可視化の実現のために、可視化内容と要求される時計機能について述べた。また、可視化に必要な情報を取得し、かつ情報取得の処理負荷を軽減する OS のプログラム構造を提案した。

可視化内容の種類としては、各機能の相互関係の変化を表す動作の可視化と、管理する資源の状態の可視化の二種類がある。可視化のレベルとしては、OS の各機能を実現している機能処理部分を単位とし、これらの呼出関係と操作内容についての可視化を行なう。

各プロセス毎の個別な動作の可視化を実現するために、時計機能としては、(1) 独立した時刻、(2) 可変な時刻進度、(3) 時刻逆行操作の禁止、の三つの機能が必要となる。

また、(1) プログラム部品の独立化、(2) プログラム呼出表の利用、(3) 識別子の付与、の三つの特徴を持つプログラム構造を提案する。

今後は、我々の開発しているオペレーティングシステム ***Tender***[3][4] 上での実現を進めていく予定である。

参考文献

[1] 谷口 秀夫 : "プロセッサ性能を調整する制御方式の実現と評価" 第 37 回プログラムシンポジウム報告書, pp.177-188(1996).

[2] 野口裕介, 後藤真孝, 谷口秀夫, 牛島和夫 : "***Tender*** における資源「時計」の実現", 情報処理学会第 52 回全国大会予稿集, 4M-8 (1996).

[3] 谷口秀夫 : "分散指向永続オペレーティングシステム ***Tender***", 情報処理学会シンポジウム論文集 Vol.95, No.7, pp.47-54(1995).

[4] 後藤真孝, 谷口秀夫, 牛島和夫 : "Tender における資源管理方式", 情報処理学会第 51 回全国大会予稿集, 5L-7 (1995).