システムソフトウェアと
オペレーティング・システム 70－8
（１９９５．８．２２）

マイクロカーネルLavenderにおける階層化インタフェース

毛利 公一† 山田 博士† 斎藤 彰一† 中村 素典†† 大久保 英嗣††

†立命館大学大学院理工学研究科
††立命館大学理工学部情報学科

マイクロカーネルLavenderは，ポリシとメカニズムの分離，ユーザカスタマイズ可能なカーネル，クロスアドレススペースコールのオーバヘッドの軽減などを目標に設計されている．本論文では，ポリシとメカニズムの分離を達成するための，Lavenderにおける階層化インタフェースについて述べる．階層化インタフェースは，ニュークリアス層，カーネル層，システム層の3層によって構成される．ユーザは，個々のアプリケーションの要求に応じて，使用するインタフェースを選択することができる．本論文では，特に，メモリ管理部とプロセス管理部のインタフェースについて述べる．

Layered Interface in Lavender Micro Kernel

Koichi Mouri† Hiroshi Yamada† Shoichi Saito†
Motonori Nakamura†† Eiji Okubo††

†Graduate School of Science and Engineering, Ritsumeikan University
1916 Noji, Kusatsu, Shiga 525, Japan

††Department of Computer Science,
Faculty of Science and Engineering, Ritsumeikan University
1916 Noji, Kusatsu, Shiga 525, Japan

Lavender micro kernel is now being designed to achieve the following objectives: separation of policy and mechanism, user-customizable kernel and decrease of overhead in cross-address-space call. In this paper, a layered interface in Lavender micro kernel for achieving the separation of policy and mechanism is described. The layered interface consists of nucleus layer, kernel layer and system layer. Users can selectively use each layer according to each application's requirements. In particular, in this paper, the interfaces in memory management and process management are described.

1 はじめに

現在, マイクロカーネルの技術は多くのオペレーティングシステム (以下, OS と記す) で採用され, その有効性が認められている. この理由として, 以下のものがある.

- OS として必要最低限の機能をカーネルとして実現しているため, システムのサイズを小さくすることができる.
- サーバの機能を追加·変更することによって, ユーザは OS の動作を制御・変更することが可能である. 従って, 拡張性や柔軟性が高い.
- ユーザに対してハードウェア独立のインタフェースを提供することによって, 移植性の高いソフトウェアを構築することができる.

このように, マイクロカーネルには多くの利点が存在する. しかし, そのマイクロカーネルにもまだ多くの課題が残されている [1]. 特に我々が注目した課題には以下のものがある.

- マイクロカーネルはその構成上, モノリシックな OS と比較した場合に, プロセスの切替えによるオーバヘッドが大きい.
- マイクロカーネルは, 一般に必要最低限の機能をユーザに提供するものであるが, その機能がユーザにとって必要以上に高機能であったり, 融通がきかなかったりする場合がある. それらの機能のために, 逆にユーザが柔軟な操作の実行を阻害されたり, 必要とする情報の参照や変更ができないことがある.

我々は, これらの課題を解決するために, ユーザによって柔軟にカーネル内の機能の変更, 及び状態の参照·変更を可能にするインタフェースを提供するマイクロカーネル Lavender を PC/AT 互換機上に構築している.

本論文では, Lavender の階層化インタフェースについて, 特にメモリ管理部とプロセス管理部を中心に述べる. 以下, 2 章で Lavender の特徴について, 3 章で Lavender の構成について述べる. 4 章でメモリ管理部の階層化インタフェース, 5 章でプロセス管理部の階層化インタフェースについて述べる.

2 Lavender の特徴

Lavender は, メモリ, プロセス, スレッド, プロセス間通信, 割り込みに関する機能のみを持ち, これら以外の機能はすべてユーザプロセスとして実現される. Lavender は以下の特徴を有している.

(1) 階層化インタフェース

Lavender では, OS の各機能をハードウェアに依存する最下位層から, より抽象化, 高度化した最上位層までの 3 階層で実現している. 階層化インタフェースは, OS の機能を各層毎のインタフェースとしてユーザに提供するものである. ユーザは, 下位の層のインタフェースを使用することで, よりきめ細かな処理を行うことができる. また, 従来 OS に隠蔽されていたカーネル内部の状態や情報を参照・変更することもできる. これらにより, システムサーバをより柔軟に構築することが可能となる.

(2) プロセスグループ機能

一つの仮想アドレス空間内に複数のユーザプロセスを同時に存在させることを可能にする機能である. この機能により, グループ内のプロセス間で, 同一アドレス空間内のジャンプを用いたスレッドの切り替えが容易に実現できる. また, ユーザプロセス間で共有メモリを確保することでプロセス間の協調作業も容易に実現できる.

(3) レジデントアドレス空間

Lavender では, 仮想アドレス空間をユーザアドレス空間, カーネルアドレス空間, レジデントアドレス空間の 3 つに分類している. レジデントアドレス空間はユーザアドレス空間の特殊な形態である. 異なるグループに属するプロセス間でプロセスの切り替えが発生した場合, ユーザアドレス空間は切り替わる. しかし, レジデントアドレス空間はこの場合でも切り替わらない. すなわち, ユーザプロセスから見ると, レジデントアドレス空間内にあるプロセス (以下, レジデントプロセスと記す) は, 同じグループに属しているように見える. これによって, レジデントプロセスとユーザプロセスの間でも, 同一アドレス空間内のジャンプを用いたスレッドの切り替えや, 共有メモリによる協調作業も容易に実現できる.

3 Lavender の構成

Lavender の開発に際しては, “ポリシとメカニズムの分離” を設計目標の中心に据えている. Lavender

におけるポリシとメカニズムの分離は, 動作アルゴリズムと, 実際の処理を行う機構を分離することを言う. さらに, ユーザがポリシをサーバプロセスの形で構成し, カーネルがメカニズムを提供する.

3.1 Lavender の全体構成

マイクロカーネル Lavender と各サーバプロセス, デバイスドライバを含めた全体構成を図 1に示す. カーネルアドレス空間にメカニズムを提供する Lavender マイクロカーネルを配置し, ユーザアドレス空間にポリシを決定するサーバプロセス, そしてデバイスドライバを配置している.

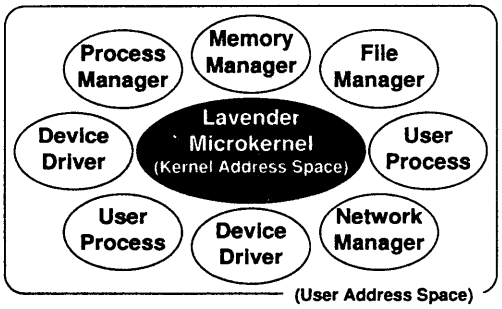

図 1 Lavender の全体構成

ユーザアドレス空間に配置される各サーバは, カーネルのインタフェースが提供するメモリ, プロセス, スレッド, プロセス間通信, 割り込みに関する機能を利用して, 以下の例に示すようなポリシを構成する.

- メモリマネージャ … ページやセグメントなどのメモリの管理単位を決定するポリシ , ページングポリシ
- プロセスマネージャ … スケジューリングアルゴリズム, プロセス及びスレッドの生成方法
- ネットワークマネージャ … エラー訂正の方法, パケットの構成方法
- ファイルマネージャ … 遅延書き込みのアルゴリズム, バッファリングアルゴリズム.

また, Lavender ではデバイスドライバもユーザアドレス空間にプロセスとして配置される. デバイスドライバをプロセスとして実現することで, 新しいデバイスドライバの追加や, 不要なデバイスドライバの削除を, カーネルに影響を与えることなく行うことが可能となる.

3.2 Lavender マイクロカーネル

マイクロカーネルを使用するユーザには, システムプログラマ, アプリケーションプログラマ, エンドユーザなど, さまざまなユーザが存在する. 従って, ユーザがサーバプログラムを作成する際に, カーネルに要求するサービスの内容は一元的には定まらない. そこで, 我々はカーネル内部の機能を, ハードウェアに直接関係したニュークリアス層から, より高度化, 抽象化されたシステム層までの 3 階層に階層化した.

Lavender は, メモリ, プロセス, スレッド, プロセス間通信, 割り込みに関する機能を持っており, 各機能は図 2に示すように階層化されて実現されている.

この階層化インタフェースは, メカニズムに相当する. ユーザは目的に応じて, 以下に示す Lavender の提供する 3 つのインタフェースから, 適切な機能を持つインタフェースを選択し, そのインタフェースを用いて, サーバとしてポリシを記述する.

(1) ニュークリアス層

この層は, ユーザに対してハードウェアを直接操作するためのインタフェースを提供する. また, ユーザがハードウェアに依存した形式でカーネル内の情報を参照, 変更することを可能にしている.

(2) カーネル層

ニュークリアス層で提供されるインタフェースを組み合わせて構成されている. カーネル層が提供する機能は, システム層の処理内容を, 意味をなすひとかたまりの処理を行う単位に分割したものである. Lavender ではこの単位をユニットファンクションと呼んでいる. そして, このユニットファンクションをハードウェアに依存しないように抽象化している. また, カーネル内の情報や状態もハードウェアに依存しない形式に抽象化されており, ユーザはハードウェアの違いを意識することなく直接参照・変更することができる.

(3) システム層

カーネル層で提供されるインタフェースを組み合わせて構成されている. ユーザに対して従来のマイクロカーネルのシステムコールに相当する高機能なインタフェースを提供する. ユーザは, このインタフェース層を通じてカーネルの内部情報を直接参照・変更することはできない.

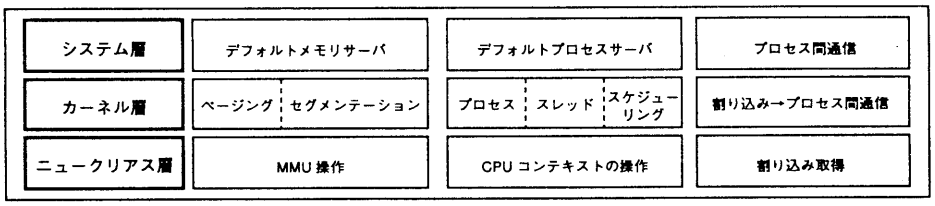

図 2　Lavender の構造

ニュークリアス層は, 主にシステムプログラマが, ハードウェアを直接操作しなければならないようなプログラムを作成する場合に使用する層である. 通常, ハードウェアを直接操作するにはアセンブリ言語を使わなければならない. しかし, アセンブリ言語によるプログラミングはプログラマに負担がかかる. ニュークリアス層を用意することにより, C 言語を用いてハードウェアを操作することを可能としている.

カーネル層は, 主にシステムプログラマやアプリケーションプログラマが, 以下に示す場合に使用する層である.

- システム層で提供されている機能よりも粒度の小さいインタフェースを必要とする.
- カーネル内の状態や情報を参照, 変更したい.
- ハードウェア独立のプログラムを作成する.

カーネル層を用意することで, 従来のマイクロカーネルのシステムコールではユーザが触れることのできなかった, カーネル内部で自動的に行われていた処理を制御することを可能としている.

また, ハードウェアを直接操作するような操作が必要な場合を除き, マイクロカーネル上にソフトウェアを構築する際には, そのソフトウェアの移植性も課題となる. カーネル層では, ハードウェアに依存する部分を抽象化しており, プログラマが移植性の高いプログラムを作成することを可能としている.

システム層は, 主にアプリケーションプログラマやエンドユーザが利用する層である. これらのユーザは, 通常カーネル内部の状態や情報を必要とせず, かつ高機能なインタフェースを用いてプログラミングを行う. システム層は, ハードウェア独立で, かつ高機能なインタフェースを提供し, 粒度の小さいインタフェースを必要としないユーザのプログラミングの負担を軽減することができる.

4　メモリ管理部

4.1　仮想アドレス空間

Lavender における仮想アドレス空間は, 1 つのカーネル空間, 1 つのレジデント空間, 複数のユーザ空間からなる. 仮想アドレス空間の構成を図 3に示す. カーネル空間には保護されたカーネルのコードとデータが置かれる. ユーザ空間にはプロセスが置かれる.

プロセスは複数のセグメントから構成される. セグメントは, プロセスのレジデント空間への再配置や, プロセスグループを行う時の単位となる. セグメントを使うことで, ポインタを含むようなデータ構造に対して一意なアドレスを割り当てることができ, 再配置にかかるコストを軽減できる. また, セグメントはプロセス間で共有することが可能である. 例えばプロセス間でデータセグメントを共有することで, 共有メモリの利用が可能となる.

(1) プロセスグループ

Lavender では, 同じユーザ空間に異なるプロセスの複数のセグメントを配置することを可能とする. プロセスグループは, この機能によって実現される. グループ化されたプロセスは同一ユーザ空間に存在するため, 従来のアドレス空間をまたがるプロセス間手続き呼出しを, 同一のアドレス空間内での手続きの呼出しに変更することができ, そのオーバヘッドを軽減することができる. またグループ化されたプロセス間で共有メモリを取得した場合, そのアドレスの一元化を可能とするため, プロセス間の協調作業を行う際のデータ参照の効率化が図れる.

(2) レジデント空間

レジデント空間はユーザ空間の一種であり, プロセスが配置される. しかし, レジデント空間はユーザ空間のスイッチに影響されず, 常に仮想アドレス空間

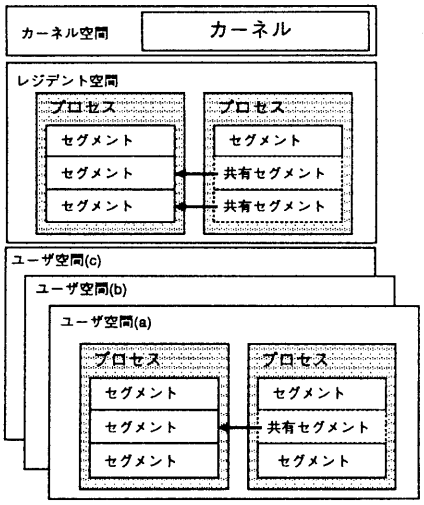

図 3　仮想アドレス空間

上に存在する．レジデント空間に存在するプロセスは，図1のようにすべてのプロセスとグループ化されている．従来のマイクロカーネルでは，頻繁なプロセス間通信によるオーバヘッドが指摘されているが，レジデント空間にシステムサーバを配置し，共有メモリ及び同一空間内の手続き呼出しを用いることで，そのオーバヘッドを軽減することができる．

4.2　メモリ管理インタフェース

メモリ管理部は階層化インタフェースとして，システム層，カーネル層，ニュークリアス層を提供する．以下に各階層について説明する．

(1) システム層

システム層では，下位層のカーネル層の機能を利用し，デフォルトのメモリサーバ（後述）としての機能を提供する．主な機能としてユーザ空間やセグメントの生成・破棄などがある．また，カーネルがメモリサーバと交信する際のインタフェースとしての機能も提供する．

(2) カーネル層

カーネル層は，仮想アドレス空間とセグメントを管理するための内部テーブルの生成，破棄，拡張，参照，変更を行う機能を提供する．カーネル層は，メモリサーバと交信し，メモリを管理するための必要な通知を行う．通知を受けたメモリサーバはカーネル層を利用し，内部テーブルの操作によりメモリ管理を行う．

・ 仮想アドレス

仮想アドレス空間はページを単位として管理される．各ページはそれぞれ，所有プロセス，メモリサーバ，保護などの属性を持つ．カーネル層は，これらの属性の変更を可能とするインタフェースを提供する．またカーネル層では，ページ管理のために新たな属性を追加する機能を持つ．

・ セグメント

セグメントは，各仮想アドレス空間に1つ存在するセグメントテーブルで管理される．カーネル層ではこのセグメントテーブルの操作を可能とする．セグメントテーブルは，それぞれのセグメントの開始アドレスとその領域の大きさを持っている．この開始アドレスと領域の大きさの変更により，仮想アドレス空間上でセグメントの再配置を行うことができる．例えば，プロセスのレジデント空間への移動はセグメントの再配置により行う．

・ 物理メモリ

物理メモリを管理する機能もカーネル層で提供される．物理メモリの管理はページ毎に行われる．空きページの残数が設定値を下回った時や空きページがなくなった時，カーネル層はメモリサーバにその通知を行う．ページ置換などの必要な処理は，通知を受けたメモリサーバが行うことができる．

・ 仮想ページテーブル

カーネル層の最下層で，仮想的なページテーブルを提供することにより，ページング機構などのハードウェアの抽象化を行う．

(3) ニュークリアス層

ニュークリアス層では直接ハードウェアを操作するインタフェースを提供する．この階層では，ページングに関するレジスタの設定，セグメンテーションに関するレジスタの設定，ページテーブルの操作，セグメントディスクリプタテーブルの操作を実現する．

4.3　メモリサーバ

メモリ管理は，メモリサーバと呼ばれるプロセスによって，その機能拡張が可能である．メモリサーバは複数個の生成が可能であり，複数のメモリ管理ポリ

シを実現することができる．メモリサーバにおけるメモリ管理の単位はページである．

メモリサーバは，階層化インタフェースの利用により，カーネルの内部状態を取得，変更し，メモリ管理を行うことができる．階層化インタフェースのカーネル層を利用することにより，ページ置換アルゴリズム，キャッシュアルゴリズム，ディスクバッファアルゴリズムなどのポリシの変更を可能とする．これによって，例えば連続メディアを扱うようなマルチメディアアプリケーションの場合，ディスクのデータ読み出し時におけるプリフェッチや，そのバッファ配置の効率化が可能となる．メモリサーバは，機種非依存を意識してカーネル層を利用することが望ましいが，ニュークリアス層の利用によって，ハードウェア固有の機能を利用することができ，それを活かしたメモリ管理が可能となる．

5 プロセス管理部

5.1 プロセス・スレッドモデル

Lavenderにおけるプロセスは静的な概念であり，1つの仮想アドレス空間に属し，1つ以上のセグメントに分割された実行環境である．ただし，プロセスが属するアドレス空間は，プロセスグループ機能によって他の複数のプロセスと共有されていてもよい．

スレッドは動的な概念であり，レジスタセットとスタックを持った実行実体である．スレッドはプロセスが提供する環境の中で動作する．そして，スレッドは1つのプロセス内に複数あってもよい．

5.2 プロセス管理インタフェース

プロセス管理部の処理は，割り込みハンドラからの要求受け付け，スケジューリング，ディスパッチ，その他のシステムコール処理に分けることができる．

システム層を用いてプロセスマネージャを構築する場合，ユーザはスケジューラのポリシ部分を記述する．システム層ではこれをサポートするためのメカニズムとして，プロセスとスレッドの状態を取得・変更するための機構や，プロセスとスレッドの構造体を取得する機構などを用意している．ユーザはこれらの機構を用いることで，スケジューリングの計算によって得られた結果をカーネルに伝えるだけでよい．

カーネル層では，ユーザがプロセスマネージャを構築する場合，割り込みハンドラからの要求受け付け，スケジューリング，ディスパッチ，システムコールに対するポリシを決定することができる．カーネル層ではメカニズムとして，プロセス管理に関する割り込みをカーネルが拾った場合に制御を移す関数を登録する機構や，プロセス構造体の作成の機構，プロセスのアドレス空間の取得，プロセスのセグメント設定の機構などを提供している．これによって，柔軟なプロセス，スレッドの操作を実現できる．さらにプロセス管理のすべてをユーザが制御することが可能となる．

5.3 プロセス管理機構

プロセス管理部に伝えられる割り込みは，プロセス管理に関する，ハードウェア割り込みとシステムコール割り込みである[2]. Lavenderにおける，割り込みによるプロセス管理部の処理の流れについて述べる(図4参照).

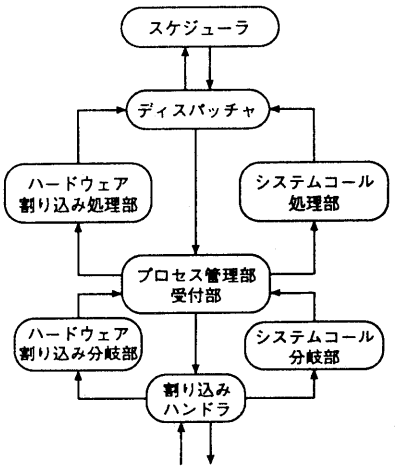

図4 プロセス管理の流れ

Lavenderでは，割り込みが発生すると，制御は割り込みハンドラに移る．割り込みハンドラでは，割り込みがシステムコールのものか，ハードウェアによるものかを調べる．システムコールによる割り込みの場合，システムコール分岐部を呼び出す．また，ハードウェアの場合は，ハードウェア割り込み分岐部を呼び出す．両方とも分岐部で，どの管理部に対する割り込み要求であるかを調べる．

割り込みがプロセス管理部に対するものであった場合，プロセス管理部の受付部に要求を送る．受付部では，割り込みの具体的な要求を解析し，システムコー

ルであればシステムコール処理部を呼び出して該当するサービスを行う．ハードウェア割り込みであれば，ハードウェア割り込み処理部を呼び出して処理を行う．

それぞれの処理が終了するとディスパッチャに制御を移す．ディスパッチャ内ではスケジューリングのためにスケジューラを呼び出す．そして，スケジューリングの結果をディスパッチャに伝え，処理を継続する．

図4のプロセス管理部の受付部に，プロセスに関するすべての割り込み要求を集めることによって，ユーザがプロセス管理の割り込みを処理することが可能となる．すなわち，プロセス管理のすべてをユーザが行うことができる．

5.4 スケジューリングサーバ

Lavenderの階層化インタフェースのシステム層を用いて，特定のプロセスのスケジューリングアルゴリズムを変更可能にするスケジューリングサーバを実現することができる (図5参照)．

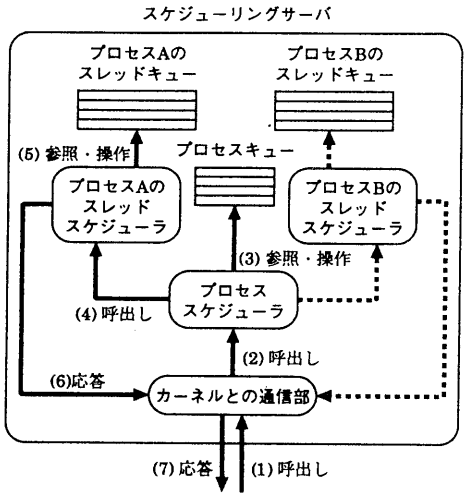

図5 階層化インタフェースの応用例

図5では，プロセスA及びBに対するスケジューリングが行われるときの処理の流れを示している．これは，図4のスケジューラに相当する．図5の (1) は，このプロセスマネージャへの要求であり，図4のスケジューラの呼び出しに相当する．

要求を受けた通信部は，(2) でプロセススケジューラを呼び出す．プロセススケジューラは (3) でプロセスキューを参照・操作し，プロセスに対するスケジューリングを行う．さらに，スケジューリングされるプロセスにスレッドスケジューラが定義されているかどうかを調べる．

ここでは，プロセスAがスケジューリングされたとする．プロセススケジューラは，プロセスAのスレッドスケジューラが定義してあるため，(4) でプロセスAに対するスレッドスケジューラを呼び出す．

スレッドスケジューラは，(5) でプロセスAのスレッドキューを参照・操作して，プロセスAに属するスレッドのスケジューリングを行い，次に実行すべきスレッドを決定する．

決定したスレッドを (6) で通信部に伝え，(7) でディスパッチャにスケジューリングの結果を伝える．

6 おわりに

本論文では，マイクロカーネルLavenderにおける，メモリ管理部とプロセス管理部の階層化インタフェースを中心に，その特徴と構成を述べた．

Lavenderは，ポリシとメカニズムの分離の概念をカーネルの構成方法に適用し，階層化インタフェースの形で構築している．カーネルは階層化インタフェースでメカニズムを提供し，ユーザはポリシをサーバプロセスとして記述できる．さらに，ユーザが個々のアプリケーションの目的に応じて，インタフェースを選択することで，プログラミングの負担を軽減するとともに，柔軟な処理を行うことができる．また，レジデントアドレス空間，プロセスグループ機能によって，クロスアドレススペースコールのオーバヘッドを軽減し，さらにプロセス間の協調作業を容易に実現することができる．

現在は，単一ノードのシングルプロセッサ上で構築を行なっている．今後，継続して構築するとともに，分散・並列環境を考慮した設計も進めていく予定である．

参考文献

[1] B. N. Bershad et al.: *"SPIN - An Extensible Microkernel for Application-specific Operating System Service"* ,Technical Report UW-CSE-94-03-03, Department of Computer Science and Engineering, University of Washington (1994).

[2] S. J. Leffler 他著, 中村 明 他訳: *"UNIX 4.3 BSD* の設計と実装" 丸善 (1991).