システムソフトウェアと
オペレーティング・システム 83－5
（２０００．２．２５）

マイクロカーネルLavender上へのUNIXサーバの構築

外山 正勝 † 毛利 公一 †† 大久保 英嗣 †††

†立命館大学大学院理工学研究科
††東京農工大学工学部
†††立命館大学理工学部

マイクロカーネルベースのオペレーティングシステム (OS) は, その柔軟性や拡張性が注目されている. 我々が構築している Lavender は, ユーザカスタマイズ可能なマイクロカーネル方式の OS である. 現在, 我々は, Lavender の性能評価およびマイクロカーネルベースの OS に関する考察を目的とし, UNIX サーバを構築している. Lavender における UNIX サーバは, UNIX 機能を適度な粒度のシステムサーバに分割して実現するマルチサーバ方式を採用している. また, エミュレータを使用することにより, 柔軟なシステムを実現している. 本稿では, Lavender における UNIX サーバの構成について述べる.

An Implementation of UNIX Server on Lavender Micro Kernel

Masakatsu Toyama † Koichi Mouri †† Eiji Okubo †††

†Graduate School of Science and Engineering, Ritsumeikan University
††Faculty of Technology, Tokyo University of Agriculture and Technology
†††Faculty of Science and Engineering, Ritsumeikan University

An operating system based on micro kernel architecture is appreciated with its flexibility. We have been developing Lavender micro kernel which provides a mechanism for customizing operating system services at user's level. Now, we have been also developing UNIX server on Lavender in order to confirm the performance of Lavender's functions and to study an effective architecture based on micro kernel. We have employed the multi-server structure and the emulator module for implementing UNIX server. In this paper, an implementation of UNIX server on Lavender micro kernel is described.

1 はじめに

現在のオペレーティングシステム (OS) では, 計算機能力の向上に伴い, 高度で多様な機能が求められている. しかし, 従来の OS では, これらの要求の実現のために OS の機能をカスタマイズすることは困難である. そこで, 我々は, マイクロカーネルの技術に注目し, システムサーバによる機能拡張性の向上を目的としたマイクロカーネル Lavender を構築している [1].

Lavender では, OS として最低限必要とされる基本機能のみをカーネルで実現し, システムの機能の多くは, ユーザプロセスであるシステムサーバにより実現する. したがって, OS としての振舞い (パーソナリティ) は, システムサーバによって決定される. このような構成にすることにより, 各機能の独立性が向上し, システムサーバを入れ換えることで OS の機能を変更・拡張することが可能となる.

現在, 我々は, Lavender 上で UNIX パーソナリティを実現するため, UNIX サーバを構築している. UNIX サーバの構築には, マイクロカーネルベース環境の研究, カーネルの性能および適用性の評価, 開発環境の整備などの目的がある. また, Lavender 上で異なる OS パーソナリティを実現するために必要なフレームワークの抽出も目的の 1 つである.

以下, 本稿では, 2 章で Lavender の概要, 3 章で UNIX サーバの概要, 4 章で UNIX サーバの構成について述べる. また, 5 章で UNIX プロセスの構成, 6 章でファイルシステムについて触れ, 7 章でまとめを述べる.

2 Lavender の概要

Lavender は, ポリシとメカニズムの分割の概念に基づき, マイクロカーネルとシステムサーバ群から構成される. Lavender の構成を図 1 に示す.

Lavender では, カーネル内部の機能を階層化し, 各層において粒度の異なる機能を提供するための階層化インタフェースを備えている. カーネル層ではメカニズムを, システム層ではポリシを提供している. また, ニュークリアス層ではハードウェア依存のインタフェースを提供している. ユーザは, 処理の内容に合わせて適切な粒度のインタフェースを選択することが可能である. この機能により, カーネル層を直接呼び出し, システム層の提供するポリシを置き換えることが可能であり, OS の機能の変更および拡張に柔軟に対応できる.

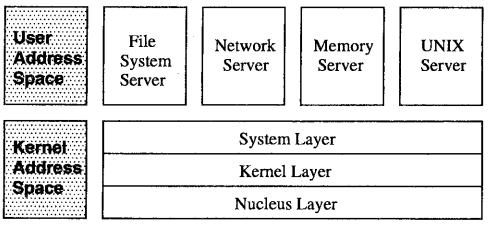

図 1 Lavender の構成

ユーザアドレス空間に配置されるシステムサーバは, カーネルの提供するメカニズムを利用し, システムのポリシを実現する. システムサーバは, ユーザ権限で動作するプロセスであり, 動的な追加・削除が可能である. Lavender では, ネットワーク機構, ファイルシステム, メモリ管理機構などはシステムサーバによって実現される.

システムサーバによる OS 機能の実現により, システムの柔軟性を向上させることができる. しかし, システムの機能を複数のプロセスに分割するため, プロセス間の協調作業が頻繁に発生し, そのオーバヘッドが問題となる. Lavender では, 協調作業時のオーバヘッドを, モジュール機構およびレジデントアドレス空間により解決する. モジュール機構は, 各プロセスが持つデータや手続きから構成されるモジュールを単位とした, プロセス間でのメモリの共有および同期のための機能を提供する. また, レジデントアドレス空間は, ユーザアドレス空間の一部であり, ユーザアドレス空間の切替えに影響されず常に仮想アドレス空間上に存在する領域である. レジデントアドレス空間上のプロセスは, プロセス間協調作業時に, 処理コストのかかるアドレス空間の切替えを必要としない. これらの機構を用いることにより, 効率のよいプロセス間協調作業が可能である.

3 UNIX サーバの概要

UNIX サーバは, マイクロカーネル上に UNIX パーソナリティを実現するためのシステムサーバである. UNIX サーバは, クライアントである UNIX プロセスに対し, UNIX のセマンティクスによる実行環境を提供する.

3.1 構成方式

UNIX サーバの構成方式として, 以下に示すものが考えられる.

(1) シングルサーバ方式
すべての機能を単一のシステムサーバにおいて実現する方式であり, 従来のモノリシックカーネルと構造がよく似ている. プロセス間での処理の移行が最少限で済むため, 効率が良い. しかし, 巨大で扱い難いモノリシックサーバとなり, マイクロカーネルの利点である柔軟性が損なわれてしまう. 実装例として, Mach 上で動作する LITES [2] などがある.

(2) マルチサーバ方式
UNIX の機能を適度な粒度のシステムサーバ群に分割して実現する方式である. この機能分割により, 個々のシステムサーバの構造が単純になり, 開発時のデバッグが容易になる. また, 機能の変更および拡張が容易に実現できる. この方式では, 1 つのサーバで実現する機能が細かくなるほど, マイクロカーネルが抱えているオーバヘッドの問題が顕著に現われる. 実装例として, Mach 上で動作する GNU HURD [3] などがある.

Lavender 上における UNIX サーバは, マルチサーバ方式を採用し, ユーザカスタマイズ可能なシステムを目指す. プロセス管理, ファイルシステム, ネットワーク, TTY などは, 独立したシステムサーバとして実装され, これらが協調して動作することにより UNIX パーソナリティを実現する. したがって, Lavender における UNIX サーバとは, 各種機能を備えた複数のシステムサーバ群の総称である.

3.2 設計目標

現在構築中の UNIX サーバは, BSD 互換システムを目指している. 特に, 我々が開発環境として使用している FreeBSD とのバイナリの互換性を考慮している. Lavender と FreeBSD とのプロセスの差異は, エミュレータを用いることによって吸収する. バイナリ互換性を実現することにより, 既存のアプリケーションをそのまま利用することが可能となり, システムの実用性を高めることができる.

UNIX サーバで実現される機能は, RPC (Remote Procedure Call) を用いて他のプロセスから利用される. しかし, RPC は, オーバヘッドの大きな処理である. これは, RPC は IPC (Inter-Process Communication) によって実現されているためである. また, RPC では呼び出すプロセスが異なるアドレス空間に存在するため, 処理が他のプロセスへ移行する際に, 処理コストのかかるアドレス空間の切替えが発生する. マルチサーバ方式では, 複数のプロセス間において協調作業が頻繁に発生するため, このオーバヘッドは無視できない. UNIX サーバでは, Lavender が提供している以下に示すような機構を利用することにより, RPC におけるオーバヘッドを軽減する.

(1) モジュールの共有によるプロセス間手続き呼出し
Lavender のモジュール機構により, 協調するプロセスが同一計算機上にある場合, 他プロセスが持つコードおよびデータをモジュールとして自プロセスのアドレス空間に取り込み, コンテクストの切替えを伴わずに他プロセスの手続きを呼び出すことが可能となる. これにより, 協調プロセス間での制御の移行におけるオーバヘッドを軽減できる.
さらに, 異なる計算機上のプロセス間においてもモジュールの共有が可能であり, 同一インタフェースによる分散環境への対応が容易に実現できる. したがって, RPC をモジュール機構を用いて実現することにより, 位置透過なプロセス間協調作業を行うことができる.

(2) レジデントアドレス空間
レジデントアドレス空間上に配置されたシステムサーバは, すべてのユーザプロセスから, 自分と同じアドレス空間上に存在するように見える. これにより, レジデントアドレス空間内のシステムサーバへの RPC は, プロセスの切替えが発生するが, アドレス空間の切替えを必要としないため, 効率よく実行できる.

4 UNIX サーバの構成

Lavender の UNIX サーバでは, UNIX のシステム機能を, 複数のシステムサーバ群で実現する. 以下, 本章では, UNIX の持ついくつかの概念を実現するために必要となる, 主なシステムサーバについて述べる.

4.1 UNIX プロセスサーバ

UNIX プロセスサーバでは, Lavender のプロセスに関するメカニズムを用いて, UNIX におけるプロセス管理ポリシを提供する. UNIX プロセスサーバ

は, UNIX プロセスの状態, 確保している資源の識別子, 受信したシグナルなどを管理するためのプロセス構造体を持つ. また, UNIX プロセス構造体を参照・操作する機能を提供する.

Lavender では, 多段階スケジューリング機構によるスケジューラの定義が可能となっている [4]. この機構を利用し, UNIX プロセスのスケジューリングポリシを新たに定義し, 優先度クラスやシステム時間などを考慮したスケジューリングを実現する. しかし, このスケジューリングが行われるためには, UNIX プロセスサーバが管理しているプロセス構造体が, いつでも参照可能でなくてはならない. そこで, UNIX プロセスサーバをレジデントアドレス空間に配置し, スケジューラから UNIX プロセスサーバの持つプロセス構造体に直接アクセスする手法を用いることにより, UNIX プロセスのスケジューリングを可能にしている.

クライアントである UNIX プロセスは, エミュレータを通して UNIX サーバの持つシステム機能を利用する. UNIX プロセスの構成については, 5 章で述べる.

4.2 コミュニケーションサーバ

コミュニケーションサーバは, ネットワークおよびプロセス間通信のポリシを提供するシステムサーバである. 本サーバでは, UNIX ドメインおよび INET ドメインのソケットの管理, パイプ機構の実現などが行われる. これらの機能は, コミュニケーションサーバとしてまとめて定義されるが, それぞれ独立したシステムサーバにより実現される.

Lavender では, システム機能の一部としてネットワーク機能およびプロセス間通信機構を実現している. パイプ機構は, Lavender の IPC により実現する. また, その他の通信機能は, Lavender のプロトコルサーバ [5] を利用することにより実現する.

4.3 ファイルサーバ

ファイルサーバでは, UNIX におけるファイルシステムを実現する. ファイルサーバでは, ディスクなどのメディアデバイスに対し, ファイルを単位とした操作機能を実現する. また, デバイスを高度に抽象化し, 通常ファイルの操作インタフェースを用いてデバイスにアクセスする機能を実現する. ファイルサーバの構成は, 6 章で詳しく述べる.

5 UNIX プロセスの構成

5.1 UNIX プロセスとエミュレータ

Lavender のプロセスは, 1 つの仮想アドレス空間に属し, 1 つ以上のスレッドを持つものである. UNIX プロセスは, 1 つのスレッドを持つ Lavender プロセスで実現される. その際, プロセスにおけるポリシの違いが問題となるが, UNIX プロセスサーバおよびスケジューラのポリシモジュールによりこれを解決する.

UNIX プロセスは, UNIX サーバが持つシステム機能を利用するために, それぞれのシステムサーバと協調する. この協調作業を支援するための機構として, Lavender では, UNIX プロセスと同じ仮想アドレス空間上にエミュレータを配置している. エミュレータは, UNIX サーバと直接交渉を行う部分であり, プロセスに対しシステムサービスインタフェースを提供する. エミュレータは, UNIX プロセスのコンテクストとは別に, システムサービス処理のためのスレッドおよび例外処理のためのスレッドを備えている.

エミュレータを使用することにより, ユーザレベルでシステムサービス要求の変換が可能になり, カーネルレベルでのカスタマイズを抑えることができる.

5.2 システムコール

UNIX システムコールは, UNIX サーバへの RPC に変換される. エミュレートされた UNIX プロセスは, 本来の UNIX カーネル上で動作していることを想定してシステムコール要求を発行する. したがって, この要求を Lavender カーネルが受け取り, RPC によるサービス要求へと変換する仕組みが必要になる. システムコール発行時の処理の流れを, 図 2 に示す.

(1) ソフトウェア割込みにより, UNIX プロセスからシステムコール要求が発行され, カーネルに伝えられる.
(2) カーネルは, 受信したシステムコール要求の内容を, エミュレータに転送する.
(3) エミュレータは, 要求内容を解釈し, 適切なサーバを選択して処理を依頼し, 結果を受信する.
(4) エミュレータは, 受信した結果を適切な形式に変換して UNIX プロセスに通知し, 次の処理要求を待つ. カーネルは, UNIX プロセスのスレッドの実行を再開する.

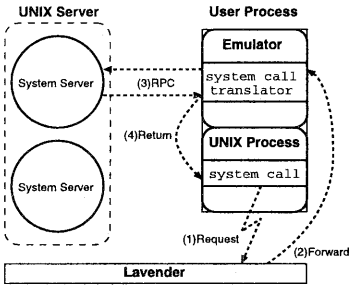

図 2 UNIX システムコール処理の流れ

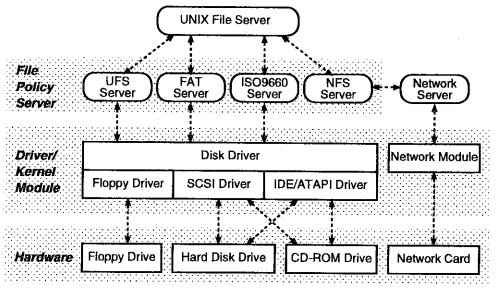

図 3 UNIX ファイルシステム部の構成

6 ファイルシステム

UNIX は, システムの持つ資源をファイルに抽象化する OS であるため, あらゆる場面においてファイルシステムへのアクセスが頻繁に発生する. このため, ファイルシステム部の性能がシステム全体の性能に大きな影響を与える. Lavender における UNIX サーバは, 複数のサーバにより UNIX のファイルシステムを実現するが, Lavender の持つ効率化の機構を取り入れ, オーバヘッドを抑えた構成となっている. 本章では, ファイルシステム部の構成を述べる.

6.1 ファイルシステム部の構成

現在, メディアデバイスの特性や用途に合わせて, 多くのファイルシステムが考案されている. いくつかの UNIX システムでは, UNIX ファイルシステム (UFS) の他に, MS-DOS で採用された FAT, CD-ROM のための iso9660, ネットワークファイルシステム (NFS) などを扱うことができる. UFS における i-node では, これらのファイルシステムをすべて扱うことはできないため, ポリシの異なるファイルシステムを統一的に扱うための仮想ファイルシステム (VFS) という層を用意し, v-node でインタフェースを定義する手法がとられている [6].

UNIX サーバにおけるファイルシステム部は, ファイルの概念を実現し, ファイル操作のための統一的なインタフェースを提供する UNIX ファイルサーバと, メディアデバイスの特性に合わせた操作機能を提供するファイルポリシサーバによって構成される. UNIX ファイルサーバはシステムに 1 つ存在するが, ファイルポリシサーバは複数存在することが可能である. ユーザからのファイルへのアクセス要求は, UNIX ファイルサーバに伝えられ, 必要に応じて各種ファイルポリシサーバが呼び出される. 図 3 に, UNIX ファイルシステム部の構成を示す.

(1) ファイルポリシサーバ
ファイルポリシサーバは, 各種メディアデバイスに対応し, その特性に合わせたファイル操作インタフェースを提供するシステムサーバである. ここでは, デバイスドライバによる入出力を, ファイル単位で行えるように抽象化する. また, ファイルシステムに関するデータ構造を管理する. 例えば, UFS サーバでは, ディスクのデータブロックから i-node テーブルを作成し, その操作および管理を行う.

(2) UNIX ファイルサーバ
UNIX ファイルサーバでは, VFS で定義されているファイルにアクセスするためのインタフェースと, 各ファイルシステム用の管理情報を, v-node によって保持する. VFS のインタフェースに対応する手続きはファイルポリシサーバが持っており, UNIX ファイルサーバは RPC によりこの手続きを呼び出す.

UNIX ファイルサーバをレジデントアドレス空間に配置することにより, アドレス空間の切替えを軽減し, 高速なサービスが実現できる. また, UNIX ファイルサーバからファイルポリシサーバへの RPC は, モジュール機構を用いて実現する. ファイルポリシサーバで定義されるデータ構造および手続きは, Lavender のモジュールとして実装され, UNIX ファイルサーバ内で直接参照される. これにより, 従来の OS では 4 回のアドレス空間切替えを伴う処理を, Lavender ではアドレス空間の切替えを行わずに実現できる. このときの処理の様子を図 4 に示す.

以上のように, 2 階層モデルを採用することにより, 新規ファイルシステムへの対応や, 既存のファイルシステムのカスタマイズが容易になる. また, 多くのシステムとの互換性を高めることができる.

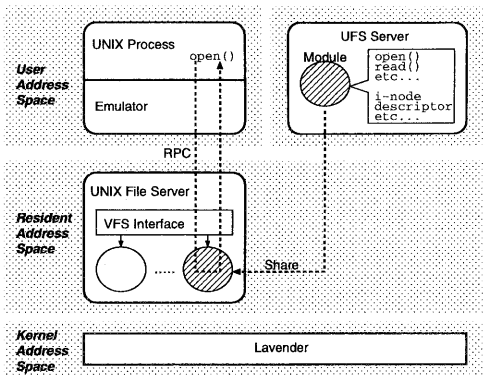

図 4 メモリ配置と処理方式

さらに, ファイルシステムの動的な置き換えも可能となる.

6.2 デバイスインタフェース

UNIX では, 計算機の持つデバイスはデバイスファイルに抽象化され, 通常ファイルと同様のインタフェースによるアクセスが可能である. デバイスファイルは, デバイスクラス, メジャー番号, マイナー番号の識別子を持っている. デバイスファイルに対する操作が行われた場合, これらの識別子に対応するデバイスドライバが呼び出され, 入出力処理が行われる.

Lavender のデバイスドライバは, システムサーバによるユーザモードドライバおよびLKM(Loadable Kernel Module) によるカーネルモードドライバで構成される [7]. Lavender のデバイスドライバでは, ポリシの記述は行われず, 低レベルな入出力処理のみが提供されている. UNIX サーバでは, ユーザモードドライバの提供するインタフェースを用いてデバイスにアクセスし, より高度な抽象化を実現する. この高度な抽象化は, デバイスファイルサーバで行われる.

デバイスファイルサーバは, ファイルポリシサーバの 1 つとして実現される. デバイスファイルサーバでは, デバイスクラス, メジャー番号, マイナー番号から対象となるデバイスを識別する. また, そのデバイスに対応するユーザモードドライバの提供する機能を用いてポリシを記述し, 基本操作インタフェースを提供する. これにより, UNIX プロセスは, デバイスを通常ファイルと同様のインタフェースで扱うことが可能となる.

7 おわりに

本稿では, Lavender における UNIX サーバの概要について述べた. Lavender の UNIX サーバは, マルチサーバ方式を採用している. これにより, マイクロカーネルの利点を活かした, 機能の変更および拡張が容易なシステムを目指す. 特に, ファイルシステムでは, UNIX ファイルサーバとファイルポリシサーバによる 2 階層モデルの採用により, 適用性を高めている.

UNIX サーバの持つシステム機能は, RPC により呼び出される. Lavender では, モジュール機構およびレジデントアドレス空間を用いることにより RPC を効率化し, 複数のサーバによるスムーズな協調作業を実現する. また, エミュレータを用いることにより, カーネルのカスタマイズを最少限に抑え, システムサービスインタフェースの変換をユーザレベルで可能にしている.

参考文献

[1] 毛利公一, 大久保英嗣: マイクロカーネル Lavender の設計と開発, 電子情報通信学会論文誌 (D-I), Vol. J82-D-I, No. 6, pp. 730–739 (1999).

[2] Helander, J.: Unix under Mach – The LITES Server –, Master's thesis, Faculty of Information Technology, Helsinki University of Technology (1994).

[3] GNU Hurd information: http://www.gnu.org/software/hurd/hurd.html.

[4] 毛利公一, 大久保英嗣: マイクロカーネル Lavender における 2 レベルスケジューラの構成方式, 情報処理学会論文誌, Vol. 40, No. 6, pp. 2534–2542 (1999).

[5] 森田健司, 佐脇秀登, 芝公仁, 豊岡明, 毛利公一, 大久保英嗣: マイクロカーネル Lavender 上へのネットワークサーバの構築, 情報処理学会研究報告 98-OS-79, Vol. 98, No. 71, pp. 31–36 (1998).

[6] 砂原秀樹, 石井秀治, 植原啓介, 林周志: プロフェッショナル BSD, 株式会社アスキー (1994).

[7] 豊岡明, 佐脇秀登, 芝公仁, 毛利公一, 大久保英嗣: マイクロカーネル Lavender における IPC 機構とデバイスドライバの構成, 情報処理学会研究報告 97-OS-76, Vol. 97, No. 77, pp. 49–54 (1997).