オペレーティング・システム　47－2
（１９９０．６．８）

OS/omicron第３版ファイルシステム内部アーキテクチャの設計と実現

横関　隆†，岡野　裕之‡，並木　美太郎，高橋　延匡

東京農工大学　工学部　電子情報工学科　情報工学講座
†ソニー　　‡日本IBM

本報告では，我々が研究・開発を行っているOS/omicron第３版のファイルシステム内部アーキテクチャ「ソフトウェアバス」について述べる．ソフトウェアバスはハードウェアのコモンバスシステムを参考に考案したもので，OS内部をモジュール化し各モジュールの呼び出しを，バスと呼ばれる仲介手続きを通して行うものである．この結果，OSを構成する各モジュールの独立性を高め，保守・拡張を容易に行うことができる。

ソフトウェアバスを導入し，OS/omicronファイルシステムでは，モジュールの交換・単体デバッグが容易に行え，OSのプロトタイピング環境の基礎を実現した．また複数のファイルシステムが共存できるマルチファイルシステムの環境を整えた．

Design and Implementation of "Software Bus" Architecture — A Module Approach for OS Construction and its Implementation on File System of OS/omicron Version 3 —

Takashi Yokozeki, Hiroyuki Okano, Mitarou Namiki and Nobumasa Takahashi

Department of Computer Science,
Tokyo University of Agriculture and Technology

2-24-16 Nakamachi, Koganei-Shi, Tokyo, 184 Japan

This paper describes a "Software Bus" architecture on which OS kernel is constructed with independent modules. The idea of this architecture is based on common bus systems hardware architectures. Each module is accessed through a common bus by internal procedures and it manages many resources uniformly.

We have implemented this software bus architecture on a file system of OS/omicron V3 , which is designed for super personal computing and Japanese information processing.

1．はじめに

我々は日本語情報処理を指向したＯＳである OS/omicron（以下ＯＳ／ｏ）の研究・開発を行っている．現在ＯＳ／ｏは第２版が稼動中で，我々の研究室で行っている手書き文字認識，和欧混合組版などを実現した卓上出版，日本語文章処理に関する研究をはじめ，学科の学生実験にも利用されている．

上記の研究だけでなく，ＯＳの研究にもこのＯＳ／ｏ第２版を使用してきたが，２年ほど使用するうちにこの版はＯＳ自体の保守・拡張が困難であるという問題点が明らかになった[1]．ＯＳ／ｏのプロジェクトでは「著作権に制限されることなく自由に改造できるＯＳ」が必要であるという見地から，システム記述言語（言語Ｃコンパイラ）からＯＳまで，すべてを独自開発で築き上げてきた[2]．この点に関してはＯＳ／ｏ第２版も手作りのＯＳであり，自由に改造できることは確かである．しかし実際には仕様書の不足という面で，ＯＳ／ｏ第２版の保守・拡張は困難であった．これは大学の研究という性質上，卒業に伴う人員の交代があること，一研究室における比較的大規模なソフトウェア開発であるため，企業並に徹底した仕様書作りは困難であることが主な理由である．

さらに我々はＯＳ自体が改造されることを前提にされていなかった点にも問題があり，ＯＳのアーキテクチャ（ここで言うアーキテクチャとはユーザーアプリケーションに対するＯＳのアーキテクチャではなく，ＯＳ研究者にとっての内部アーキテクチャを表す）からの見直しが必要であると考えた．

そこで本研究では，ＯＳの研究を指向したＯＳを実現することを目標とし，次に示す２点を目的とした．

（１）改造することを前提としたＯＳを実現する

ＯＳの研究を行うためのＯＳといった見地から，改造は常に行われるということを前提としたＯＳを実現する．

（２）ＯＳのプロトタイプ構築環境を実現する

ＯＳのアプリケーションインターフェースに関する研究のために，構成の異なるＯＳを自由に構築できる環境を実現する．またこれを利用して個人が専用ＯＳを自由に実現できる環境を構築する．

本研究では，上記２点の目的を踏まえて実現したＯＳ／ｏ第３版（以下 OMICRON V3）ファイルシステムの設計と実現について，「ソフトウェアバス[3]」と呼ぶ内部アーキテクチャを中心に述べる．

2．OMICRON V3 ファイルシステムの設計方針

本研究では OMICRON V3 ファイルシステムを実現するにあたり，次に挙げる設計方針をとった．

（１）ＯＳ内部における計算機資源の表現形式を統一する

資源の表現形式を統一することにより，図１に示すようにＯＳ内部を二つに分け，アプリケーションインターフェースと資源の抽象化に関する問題を分離する．すなわち，図１中のアプリケーションインターフェース部分は，統一された形式で表現された資源をアプリケーションにどのように見せるかだけを考えれば良く，各資源がどのような過程を通して抽象化されているかに関知する必要がない．また資源の抽象化を行う部分は，各計算機資

源をいかに統一された表現形式に変換するかだけを考えれば良く，その上位層であるアプリケーションインターフェースがどのようになっているかは関知する必要がない．

（２）ＯＳを交換可能なモジュールで構成する

ＯＳ内部をモジュール化し，モジュール間のインターフェースを統一する．これによりモジュール単位の交換が可能な構成とし，ＯＳの保守・改造を容易にする．またモジュールに関する仕様書記述の簡略化を図る．

３．資源の抽象化

２．で述べたとおり，OMICRON V3 ファイルシステムではＯＳ内部における計算機資源の表現形式を統一する方針をとった．本章ではファイルシステム内部における資源の扱い（表現やアクセスなどのモデル）について述べる．

３．１　資源の表現

我々は stored program 方式計算機のすべての資源はアドレス空間で表現できるのではないかと考え，ファイルシステムで扱うすべての資源を架空のアドレス空間（以後単に空間）に割り付ける方式をとった．例えばフロッピーディスクやハードディスクは，セクタアドレス（またはブロックアドレス）と空間のアドレスを１対１に割り付ける．より抽象度の高いファイル（ディスク上に格納されたデータ）はバイトごと（レコードごと）にアドレス付けされる．またシリアルプリンタなどのように，シーケンシャルなデータを扱う資源の場合には，空間の，ある特定番地を次々にアクセスする方法などが考えられ，プロセッサデバイスの場合には，レジスタ群を空間に割り付け，命令レジスタをアクセスすることにより，命令が実行されるなどの方法が考えられる．

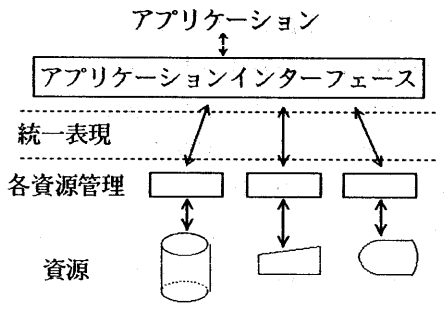

図１　ＯＳ内部の二分化

OMICRON V3 では資源の表現形式を図２に示すように属性，子資源名表を伴ったものとした．属性は各資源の大きさや，入出力の最小単位などを保持するもので，子資源名表は各資源の論理的な親子関係（階層関係）を表現するために利用するものである．OMICRON V3 ではこの子資源名表を用いて，ファイルモデルとして入れ子構造を実現している（後述）．

３．２　資源のアクセス

資源のアクセスは，一種のメッセージであり，一つ一つの資源はオブジェクトであると考えることができる．また資源は一つ一つが完全に独立したものである必要はなく，資源アクセスの副作用として他資源のアクセスを

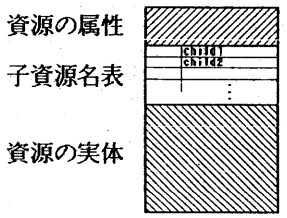

図２　OMICRON V3の資源

伴う場合もあり得る．例えば資源A，B，Cの間に図3に示すような因果関係があったとすると，資源A，Bへのアクセスは，資源Cへのアクセスという副作用を伴う．

資源のアクセスについては，読み書きを中心として表1に示すものを定義した．このなかで LINK は一つの資源を複数の親資源から共有させる（子資源名表に登録する）コマンドで，DUPLICATE は資源の複製を作成するものである．OMICRON V3 ファイルシステムはファイル単位の版管理を指向しており，これら二つのコマンドは共に資源を単位とした版管理を実現するために使用する．ここでは詳細な説明は省略する．

各コマンドはアクセスの対象となる資源（オブジェクト）に対して発行され，各資源が解釈・実行を行う．

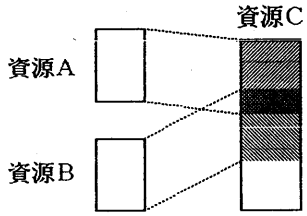

図3　資源空間の透過

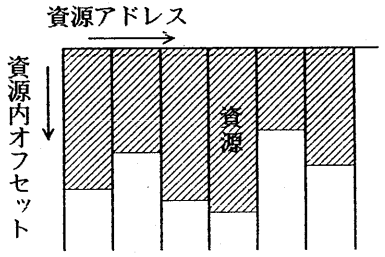

図4　資源の2次元空間

3.3　資源のアドレッシング

次に資源の特定方法である．OMICRON V3 ファイルシステムでは資源が割り付けられる空間を図4に示すように2次元にし，個々の資源（オブジェクト）に対してアドレス付けを行い，資源アドレスとした．資源のアクセスを行う場合，アクセスする資源のアドレスと資源内の位置（資源内オフセットアドレス）をコマンドの引数とする．

OMICRON V3 ではソフトウェアバス上のアドレス空間として，資源アドレスと資源内オフセットアドレスをそれぞれ31ビットで表した．

表1　資源アクセスのコマンド

コマンド名	機能
INITIALIZE	モジュールの初期化
TERMINATE	モジュールの無効化
OPEN	資源の使用開始宣言
CLOSE	資源の使用終了宣言
CREATE	子資源の作成
DELETE	子資源の削除
DUPLICATE	実体を共有した資源のコピー
LINK	子資源の共有
READ_DATA	資源実体の読出し
WRITE_DATA	資源実体の書込み
READ_STATE	資源属性の読出し
WRITE_STATE	資源属性の書込み
READ_NAME	子資源名表の読出し
WRITE_NAME	子資源名表の書込み
LOCK	資源の排他
UNLOCK	資源の排他解除
FLUSH	保持データの吐出し
CANCEL	保持データの破棄
ABORT	資源操作の中断

4．ソフトウェアバス

これまで述べてきた資源管理の機構を実現するため，ハードウェアのコモンバスシステムを参考にしたアーキテクチャ「ソフトウェアバス」を実現した．ソフトウェアバスは図5に示すように，資源のアクセスを管理する手続きを「モジュール」として実現し，「バス」が資源アドレスからモジュールの割出し，呼出しを行う．

各モジュールの呼出しインターフェースはコマンドパケットにコマンド（読み書きの別など）とそれに伴う引数（資源のアクセス位置や大きさ）を格納し，引数として渡す形に統一した．またモジュールの入口は一つの関数（入口関数）に限定し，モジュール間の結合を疎なものにした．

これによって各モジュールは，動作仕様が同じものであれば互いに交換可能になる．また，あるモジュールを呼び出す際に，呼び出し先のモジュールとして何が組み込まれているかは関知する必要がなくなる．例えば図6(a)に示すようにファイルの入出力を管理するモジュールがディスク入出力のモジュールを呼び出している場合に，図6(b)に示すようにそれまでディスク入出力モジュールがあった位置（アドレス）にディスクキャッシュメモリモジュール（外部的な動作仕様はディスク入出力モジュールと同じとする）を組み込むと，ファイル入出力モジュールの変更を行うことなくシステムの改良が行える．

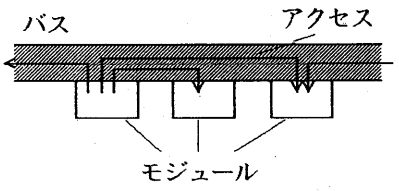

図5　ソフトウェアバス

またＯＳ内部の動作を評価するために，評価データ測定ルーチンを組み込んだモジュールを用意したり，モジュール単体のデバッグも容易に行うことができる．

5．OMICRON V3 ファイルシステムの実現

5.1　資源のアクセス

各資源をＤＡＴ（Dynamic Address Translator）などのメカニズムによりＣＰＵの仮想アドレス空間として表現すれば，ＣＰＵによる空間へのアクセスが即資源へのアクセスとなるのであるが，OMICRON V3 は実記憶方式のＯＳであるため次に示す方式をとった．

基本的には仮想記憶のメカニズムを実記憶上のソフトウェアで模擬したものと考えることができる．資源に対するアクセス操作は，手続きの呼び出しに変換される．システム内にこの変換を行うための仲介手続きを一つ用意し，資源のアクセスは必ず仲介手続きを通す形態をとるのである．すなわちこの仲介手続きが前述のバスとなる．

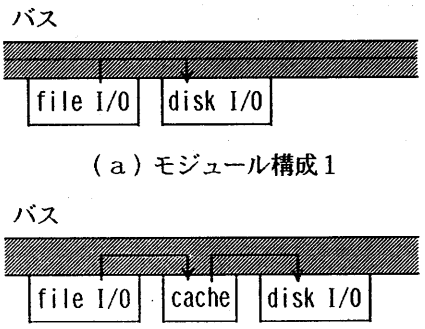

図6　モジュール構成の変更例

OMICRON V3 では３１ビットの資源アドレスをモジュールアドレスの１１ビットとモジュール内でのローカル資源アドレス２０ビットとに分けた．したがってシステム内には最大 2,048 個のモジュールが実現でき，一つのモジュールは最大 1,048,576 個の資源を管理することができる．また一つの資源は２Ｇ(ギガ)のアドレス空間を持つ．

OMICRON V3（言語Ｃで記述）の仲介手続きは関数の表を持っており，表へのインデックスがモジュールアドレスとなっている．

５．２　モジュール構成

OMICRON V3 ファイルシステム内での主なモジュール構成は図７のようになっており，ディスク入出力装置からファイルまでの抽象化を段階的に行っている．

（１）ディスク入出力モジュール

ハードディスクやフロッピーディスクの物理的な入出力を行う．

（２）キャッシュメモリモジュール

ディスク上にあるデータのキャッシングとセクタからページ（１ｋバイト）へとデータの抽象度を上げる．

（３）ＦＣＢ（File Control Block）管理モジュール

ディスク上でのＦＣＢ領域のアクセスを管理する．

（４）ページ番号管理モジュール

ＦＣＢが管理しているページ番号列の入出力を担当する．

（５）ファイルプリミティブ入出力モジュール

ページ番号列とキャッシュメモリを使用してファイル内容の入出力を行う．

（６）ファイル入出力モジュール

最終的にファイルとしての体裁を整える．

これら一連のモジュールは，ディスク装置１台につき一つづつ別々のアドレスに用意されており，モジュールアドレスによってディスクドライブを特定する形態になっている．

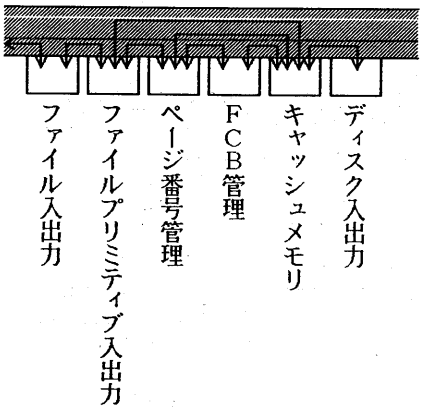

図７　OMICRON V3のモジュール構成

５．３　アプリケーションインターフェース

OMICRON V3 ではファイル（資源）のモデルとして，入れ子構造を考案し導入した．これはＯＳ／ｏ第２版ファイルシステムの２重階層構造を拡張・一般化したもので，ファイルが図８に示すように一つの実体と複数の子ファイルを持ち，子ファイルも親ファイルと同

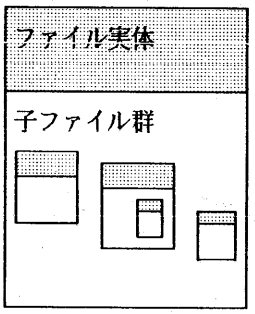

図８　入れ子構造

じ構造を持つものである．実現の際の構造としては，従来のＯＳで呼ぶディレクトリとファイルが一つずつペアになったものが OMICRON V3 のファイルであると考えればよい．

この構造における特徴の一つとしてファイルオープンの方法がある．OMICRON V3 ではファイルのオープンを，

子fd = open(親fd, パス名, ...);

のように必ずすでにオープンされている親ファイルを基点にして行う．これによってアプリケーションが起動された時に，少なくとも一つのファイルがオープンされている必要があるという制約がつくが，一方ＯＳ内部からカレントディレクトリという概念がなくなった．なぜならばオープンされているファイルすべてがカレントディレクトリ（OMICRON V3 ではカレントファイル）となり得るためである．

ファイルモデルとして入れ子構造を導入したことにより，カレントディレクトリを複数持つ環境や，プロセス（OMICRON V3 ではタスクフォースと呼ぶ）ごとに異なるルートディレクトリを持つ環境，ファイルのサーチパスをオープンしておくことによりファイル検索の高速化を図る，といった応用が自在にアプリケーションインターフェースとして実現できる環境となった．

また OMICRON V3 では，ファイルシステムはタスク管理部が管理するタスクフォースとして実現した．ファイル操作ＳＶＣ（Super-Visor Call）は表１に挙げたソフトウェアバスのコマンドをタスク管理を通して見せたものが基本となっている．ただし資源の指定は資源アドレスではなくファイル名で行う．ファイルシステム内の図１で示すアプリケーションインターフェースの部分に，ファイル名から資源アドレスを求める手続きが組み込まれている．

５．４　マルチファイルシステム

OMICRON V3 のファイルシステムは図１に示したとおり，資源の統一表現形式を境界にアプリケーションインターフェースと資源の抽象化を行う部分とが切り離されている．これを利用して OMICRON V3 では異なる複数のファイルシステムが共存する環境を構築した．

すなわちどのようなファイル（ＯＳ／ｏ第２版形式のファイルや MS-DOS 形式のファイルなど）であっても，統一された表現形式に変換すればアプリケーションインターフェースによって区別されずに扱われる．これを利用して OMICRON V3 ではそれ自身のファイルとＯＳ／ｏ第２版形式のファイルが共存する環境を構築した．

６．おわりに

本報告では OMICRON V3 ファイルシステムの内部アーキテクチャであるソフトウェアバスについてと，ソフトウェアバスを基本にして実現したファイルシテムについて述べた．

現在 OMICRON V3 ファイルシステムは内部モジュールのコーディングが終了し実用化に向けての実現が行われている．プログラムの規模はファイルシステムだけで約 20,000 行（言語Ｃ）であり，モジュールあたり平均約 1,000 行である．今後は，モジュールを管理するためのモジュールデータベースの構築や，モジュールを組み合わせて手軽にＯＳのプロトタイプを作成できる環境の実現を行っていくと同時に，ＯＳ内部アーキテクチャとしてのソフトウェアバス自体の評価も行っていく

予定である．

参考文献

[1] 横関，並木，高橋：“OS/omicron 第２版におけるＯＳの開発環境”，情報処理学会第３８回全国大会，pp.791-792，1989.3.

[2] 高橋：“研究プロジェクト総説：OS/omicron の開発”，情報処理学会オペレーティング・システム研究会，39-5，1988.6.17.

[3] 横関，岡野，並木，高橋：“OS/omicron 第３版ファイルシステムの設計と実現”，情報処理学会第４０回全国大会，pp.726-727，1990.3.