システムソフトウェアと
オペレーティング・システム 75－3
（１９９７．６．５）

OS/omicron 第４版におけるデータ管理機構の設計と実現

高野了成，佐藤元信，早川栄一，並木美太郎，高橋延匡
東京農工大学　工学部

〒184　東京都小金井市中町 2-24-16
E-mail : takano@omicron.ei.tuat.ac.jp

本稿では，OS/omicron 第 4 版における，多様性，多義性を持つデータを管理するサーバの設計，実現について述べる．パターン情報など多様性，多義性を持つデータを効率的に管理するには，データの構造，データと操作の対応を管理する必要があるが，従来のファイルシステムではユーザが管理する必要があった．そこで，筆者はパターン情報を扱うためのデータモデルとして，「意紙」を提案した．「意紙」の実体はワンレベルストアで実現されるメモリオブジェクトであり，各メモリオブジェクトは属性を持つ．「意紙」は多義性を，メモリオブジェクトの属性を切替えによって表現するため，属性変換機構を用意する．また，「意紙」間の柔軟な構造を管理するためのリンク構造を管理する．

Design and Implementation of Data Management Mechanism on OS/omicron Version 4

TAKANO Ryousei, SATO Motonobu, HAYAKAWA Eiichi,
NAMIKI Mitarou and TAKAHASHI Nobumasa
Department of Computer Science, Tokyo University of Agriculture and Technology

2-24-26 Nakacho Koganei, Tokyo, 184 Japan
E-mail : takano@omicron.ei.tuat.ac.jp

This paper describes a design and implementation of data management server on OS/omicron Version 4. The charactaristics of pattern data are polysemy and diversity. An effective pattern data processing is required to manage data structure and relation between data and procedures. However, filesystems such as in UNIX don't support it. Thus, we propose a data model called "ISHI" as an object of pattern data management. "ISHI" has memory objects with an attribute of data type. The polysemy is represented by the attribute conversion mechanism. The flexiblity of data structure is represented by "ISHI"' s link mechanism.

1. はじめに

近年，計算機の処理能力の向上とともに，計算機で扱うことができる情報が多種多様化してきた．パターン情報もその一例であり，PDA（Personal Digital Assistant）をはじめとした携帯型の計算機などで，ペンインタフェースが採用される機会が多くなってきている．パターン情報の特徴として多様性，多義性が挙げられるが，このような情報を扱うには，対応する手続きも含め，オブジェクトとして抽象化し，管理することが有効である．

筆者らの所属する東京農工大学工学部電子情報工学科高橋・並木研究室では，パターン処理を指向したOS/omicron第4版（以下，V4）[1]の研究・開発を行っている．当研究室では，ウィンドウシステムや手書き入力によるグループウェア[2]の研究が行われており，大量のパターン情報を管理する必要がある．

筆者は，このようなパターン情報を扱うためのデータモデルとして「意紙」[3]を提案した．本稿では，V4における意紙管理機構の設計と実現について述べる．

2. データ管理機構に対する要求

2.1 OS/omicron第4版の概要

V4はパターン処理を指向したOSであり，紙を仮想化した「電紙」[1]をデータモデルとして提供する．

V4は柔軟な資源管理を可能にするためマイクロカーネル[4]構成を採用し，資源と手続きを動的に関連付けるためのダイナミックリンク，データ間の関係をポインタで記述するためのワンレベルストア，データの保護を実現するための単一2次元アドレス空間を特徴としている．図1にV4の全体構成を示す．

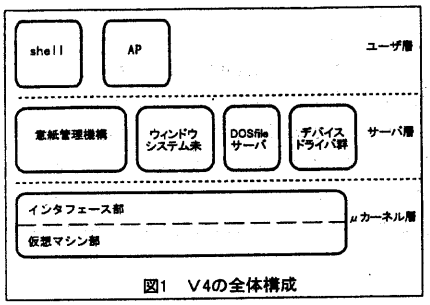

図1　V4の全体構成

2.2 パターン情報の特徴

手書き情報など，パターン情報の特徴はその多様性と多義性である．紙の特徴として文字や図など，書く対象が制限されないことが挙げられる．このように，多様性を持つということは，一つの情報に対して複数の情報に翻訳可能であるという多義性を持つ可能性も考えられる．たとえば，手書き情報では同一のストロークを入力した場合，それが図形としても，文字コードとしても認識することができる．

さらに，紙は自由に切り貼りすることができるが，このような操作をオブジェクト間の構造の操作として表す．

2.3 従来のファイルシステムの問題点

パターン情報を，従来のUNIX型のファイルシステムにおいて管理する場合の問題点を次に述べる．

(1) リンク機構による名前空間の拡張の整合性

紙を自由に切り貼りする，といった操作をシステムレベルで実現する場合，システムでオブジェクト間の構造を管理する必要がある．したがって，高い自由度，変更に対する整合性が要求される．

従来のファイルシステムは木構造の名前空間を持っているが，リンク機構[5]を用いることで名前空間を拡張することができる．このようなリンク機構を用いた場合，ファイルの移動，削除などの操作によって，リンク先の実体が存在しないなど，リンク間の整合性が破綻する可能性がある．

(2) 多義性を持つオブジェクトの管理

一般的にデータ型は拡張子やファイル内に埋めこまれたマジックナンバなどの識別子で表している．しかし，識別子では多義性を持つオブジェクト，つまり一つのオブジェクトに対して複数の型が存在するものを表すことはできない．したがって，このようなオブジェクトに対して個々のアプリケーションで対応することは困難である．

(3) データ型に特化した管理方法

従来のファイルシステムではファイルをデータ型を規定せず，ストリームとして抽象化して提供している．また，デバイスをメモリにマッピングする方法として，mmapを提供している．これらはすべてのファイルに対して平等な抽象化を提供するが，データ型のアクセスパターンに応じたページングポリシ，2次記憶への配置ポリシなどをアプリケーションレベルから指定することは困難である．したがって，データ型を規定する有効性がある．

2.4 目的

このように，従来のファイルシステムではデータの操作に対してデータ間の構造の整合性をとるのが困難である．そこで，データとそのデータに対する操作を，オブジェクトとして抽象化する．そして，オブジェクトは一つのデータ型に固定せず，場合によって適切なデータ型に切り替え可能にする．このようにオブジェクトの特徴を示すデータ型のことを属性と呼ぶ．

V4 におけるデータ管理機構の目的を次に述べる．

(1) 複数の属性を持ったオブジェクトの管理
(2) 構造を持ったオブジェクトの管理
(3) ユーザからの動的な拡張方法の提供

3.「意紙」の概念

「意紙」は電紙の特徴の中でも，電紙間の関連性に注目したデータモデルである．「意紙」間は柔軟なリンク構造を持つ．そして，個々の「意紙」は複数の属性を持ち，「意紙」の実体は各属性のインスタンスの集合と考えることができる．「意紙」の特徴を次に述べる．

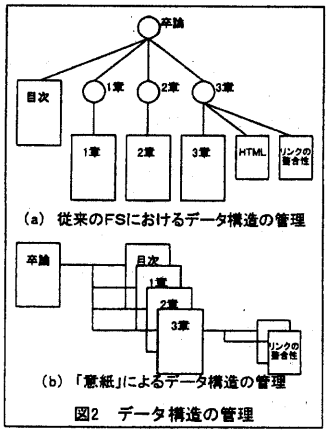

図2 データ構造の管理

(1) リンク構造

紙は文章の書かれた紙の上に図表を書いた紙を貼り付けることが可能である．このような「意紙」間の相互の関係をリンク構造として管理する．リンクのトポロジはネットワーク構造とし，UNIX のように特別なリンク機構を使わなくても，すべての「意紙」間のリンクをすべて等価に管理する．

また，図 2 のように，データ構造はファイルとディレクトリという二つの構造ではなく，「意紙」という一つの構造で表現される．

(2) 属性

「意紙」はデータ型を表す属性を持っている．また，手書き情報が文字認識によって文字コードに変換できるように，オブジェクトは複数の属性のインスタンスである可能性がある．「意紙」はこのように属性が異なる複数のオブジェクトを一つの論理的単位として扱うことができる．そして，各属性間で相互に変換できる属性変換機構を用意することで，「意紙」の操作に対する多態性を確保する．したがって，「意紙」をウィンドウに表示する場合，属性に関係なく，「表示する」という操作を実行すると，システムが「意紙」の属性を調べ，属性に応じた関数を呼び出し，実行することが可能になる．

(3) 柔軟な拡張性

属性を OS が規定してしまうと，多様なデータ型に対してユーザが自由に拡張することができない．そこで，ユーザに，新しい属性を継承の利用によって追加するインタフェースを提供し，新しい属性を追加する手間を極力軽減する．

また，属性に対する操作は多態性を持つ．このような多態性を持った操作をシステムに登録するインタフェースを提供する．

4. 意紙管理機構の設計

4.1 全体構成

「意紙」の管理機構を V4 上でサーバとして実現する．意紙管理機構は意紙サーバとファイルサーバから構成される．意紙サーバは「意紙」のリンク構造，名前，属性を管理し，インタフェース部，属性管理部，リンク管理部から構成される．ファイルサーバは「意紙」を格納するための 2 次記憶管理を提供し，ディスクのジオメトリ，「意紙」の実体となるメモリオブジェクトの操作を管理する．意紙管理機構の全体構成を図 3 に示す．

ファイルシステムを複数の層に分割する構成法として Stackable File System Layers [6] がある．Stackable File System Layers では，ファイルの属性，操作ごとに，階層を追加する．ユーザがファイル操作を実行するとき，ファイルの属性によって，カーネルが動的に実行するファイルシステムを選択する．一方，意紙管理機構では，属性の管理は意紙サーバが

行い，操作関数の追加によって拡張性を持たせる．

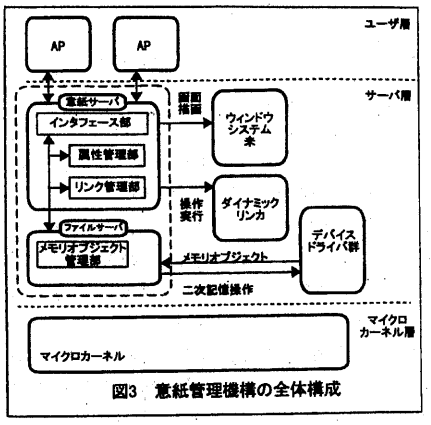

図3　意紙管理機構の全体構成

4.2 メモリインタフェース

V4 ではセグメントマッピング方式 [7] により，すべてのデバイスをメモリのインタフェースで扱うことができる．セグメントマッピング方式でメモリにマップした資源をメモリオブジェクトと呼ぶ．メモリオブジェクトのページングポリシはページャが管理する．ただし，ディスクの管理情報など頻繁にアクセスされるものは，ページイン・アウトが多発すると，システム全体の性能が低下につながるため，ページアウトが発生しないようにレジデントに指定する．

4.3 動作の流れ

実際に意紙をメモリ上にマップする要求がなされた場合，意紙サーバが意紙名から，その実体を一意に決定し，ファイルサーバがデバイスドライバにメモリオブジェクトの構成を通知する．デバイスドライバは物理ディスクの内容をメモリオブジェクトにマップし，そのメモリオブジェクトへのポインタを返す．意紙サーバはこのポインタを得ると，必要な場合には属性変換を行う．「意紙」に対する操作関数はこのメモリオブジェクトへのポインタを利用することで実行する．

4.4 属性変換機構

2.3 で述べたように，多義性を持つオブジェクトの属性を識別子で表すことは困難である．そこで，属性はすべてリンクテーブルに登録し，オブジェクトの属性はリンクテーブルへのポインタで表す．属性の追加，削除はリンクテーブルへの操作で実現する．

「意紙」の実体は属性を持ったメモリオブジェクトであり，各属性のメモリオブジェクトの集合である．図 4 で示すように，「意紙」へのアクセス時に，要求があった属性のメモリオブジェクトに対してアクセスするように，アクセスをリダイレクトするのが属性変換機構である．もし，要求があった属性のメモリオブジェクトが存在しない場合は，メモリオブジェクトの内容を変換し，新規のメモリオブジェクトとして追加する．

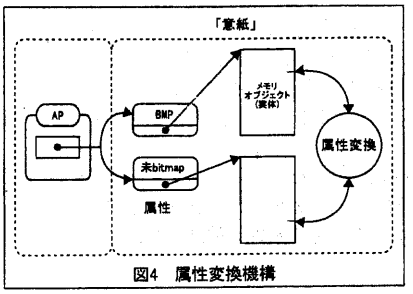

図4　属性変換機構

4.5 アプリケーションインタフェース

まず，すべての「意紙」のプリミティブとなる，メタオブジェクトを定義する．メタオブジェクトはデータに対するもっともプリミティブなアクセス法であるポインタ操作を提供している．ユーザは通常，ある属性のインスタンスとして「意紙」を生成したり，「意紙」に対する操作を実行することで，メタオブジェクトの機能を意識せずに使用している．しかし，メタオブジェクトを操作して新しい属性や，操作を定義できるインタフェースを用意する．また，「意紙」に対する操作は，多態性を持つ．ユーザは属性固有の操作を記述し，意紙管理機構に登録できる．

表1　「意紙」の基本API

・意紙の生成	・意紙のフラッシュ
・意紙の削除	・意紙のリンク
・意紙のマップ	・意紙のアンリンク
・意紙のリマップ	・意紙のリンク関係を得る
・意紙のアンマップ	・意紙の属性を得る

```
UBYTE   **ポインタ;
INT     i;

/** 意紙をメモリ上にマップする **/
ポインタ = 意紙のマップ( "テスト" );

/** 属性変換する **/
ポインタ = 意紙のリマップ( ポインタ, フル2バイトJIS );

/** 意紙の実体の中身を表示する **/
for( i = 0; i < 意紙のサイズを得る( ポインタ ); i++ ) {
    printf( "%c", (*ポインタ)[i] );
}

/** 二次記憶に書き込む **/
意紙のフラッシュ( ポインタ, 意紙のサイズを得る( ポインタ ) );

/** 意紙をアンマップする **/
意紙のアンマップ( ポインタ );
```
図5　プログラム例

基本的なアプリケーションインタフェースを表 1 に示す．これらの関数は「意紙」に対して汎用的に使用できる．たとえば，従来のシステムとのアプリケーションレベルでの互換性を保つために，open-close/read-write モデルの操作を実現する場合は，「意紙」の実体をメタオブジェクトとしてマップ

し，表 1 のアプリケーションインタフェースとポインタ操作によって実装する．メタオブジェクトに対する実際のプログラム例を図 5 に示す．まず，「意紙」のマップを要求すると，その「意紙」に対するメモリオブジェクトのポインタを得ることができる．このポインタにアクセスすることで，「意紙」の実体を得ることができる．

4.6 名前空間

「意紙」の名前空間も 2 次記憶上に格納されており，システム起動時に読み込む．名前空間もメモリオブジェクトの一つであるので，リンクをポインタで表すことができる．

4.7 永続性

「意紙」には，プロセスの生存時間に関係なく，半永久的に利用したいものがある．また，不慮のシステムクラッシュなどに対する信頼性の面からも，「意紙」を 2 次記憶上に永続化する必要がある．

ファイルサーバでは，メモリオブジェクトをページ単位に分割し，メモリオブジェクトのページ構成を i ノード [5] として管理する．また，使用可能な空き領域を調べるためにビットマップを利用している．

5. 意紙管理機構の実現

5.1 実現環境と規模

これらの設計をもとに，PC/AT 機上で動作する V4 上で意紙管理機構を実現した．実現環境を表 2 に示す．また，記述言語には当研究室で開発された言語 C 処理系，CAT386 を用いた．実現規模を表 3 に示す．

表2　実現環境

機種	Dell OptiPlex 4100/MXV
CUP	Intel 486DX4 100MHz
主記憶	48MB
HDD	200MB + 540MB

表3　実現規模

意紙サーバ	2600行
(リクエスト処理部) (属性変換部) (リンク管理部)	(730行) (1220行) (650行)
ファイルサーバ	2100行
	4700行
(ペイントツール)	(440行)

5.2 「意紙」の内部構造

V4 はワンレベルストアを採用しており，「意紙」の実体はページャによって，メモリオブジェクトにマッピングされる．そして図 6 で示すようにすべての「意紙」は管理テーブルを持っており，実体，属性へのポインタ，リンク構造を示す双方向リスト，「意紙」に対する操作関数の名前から構成される．

「意紙」のリンク関係は管理テーブル内の双方向リストで表される．意紙間のリンクはポインタで表されており，i ノード方式と比較して，柔軟な構造を持つことができる．また，「意紙」の操作に対して，管理テーブルを更新することで，リンクの整合性を動的に保つことができる．

操作関数とは「意紙」に対する操作であり，意紙サーバは，管理テーブルに格納された操作関数名をダイナミックリンカに渡し，実行時に操作関数名に操作関数の実体をバインディングする．

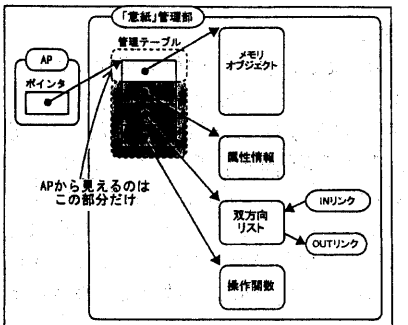

図6　「意紙」の内部構造

5.3 リンク構造の整合

5.2 で述べたように，「意紙」間リンクは双方向リストで表され，リストには「意紙」の管理テーブルへのポインタが格納されている．したがって，「意紙」の移動，削除などによって，リンク構造が変化した場合には，リストをたどることで，変更を有効にできる．また，双方向のリストなので，従来のリンク構造のように，実体が削除されても，リンクが残っているといった不整合は起こらない．

5.4 属性変換機構

属性変換は属性ごとの操作関数で実現する．属性としてビットマップとテキストを用意し，属性間での相互変換を実現した．ビットマップとして Windows 標準の BMP, V4bitmap，テキストとしてフル 2 バイト JIS，シフト JIS といった属性を用意した．属性変換を利用したアプリケーションの実行結果を図 7 に示す．図 7 では，ビットマップのフォーマットの変換に属性変換を用いたペイントツール，文字コードの変換に属性変換を用いたテキストビューアを実行している．たとえば，ペイントツールは 2 次記憶上は BMP で格納

されており，ウィンドウに表示する際に未 bitmap へ属性変換し，編集作業を行う．編集結果を保存するときは，BMP へ再び属性変換する．

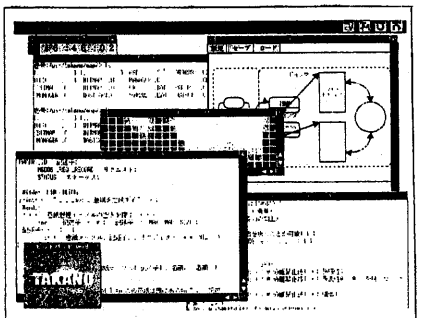

図7　実行画面

6. おわりに

本稿では，パターン情報を管理するためのデータモデルとして，「意紙」を提案し，OS/omicron 第 4 版における「意紙」を用いたデータ管理機構である意紙管理機構の設計と実現について述べた．本研究の成果として次の点が挙げられる．

・一つのオブジェクトに対して複数の属性を持たせることが可能な「意紙」による，属性に依存しない，オブジェクト単位で一貫性のある操作の実現
・「意紙」を格納するための 2 次記憶管理機構の実現

また，今後の課題として，次の点が挙げられる．

・ダイナミックリンクを用いた，属性，操作の動的な拡張インタフェースの実現
・意紙管理機構をはじめ，現在，V4 上で実装されている MS-DOS ファイルサーバ，今後，移植が予定されている OS/omicron 第 3 版のファイルシステム [8] など，複数のファイルシステムの共存を可能にする機構の実現

参考文献

[1] 早川栄一，並木美太郎，高橋延匡："手書きインタフェースを支援する OS OS/omicron 第 4 版の構成"，情報処理学会第 4 回コンピュータシステムシンポジウム論文集，pp.35-42,1992

[2] 中島一彰，早川栄一，並木美太郎，高橋延匡："コンピュータネットワーク上での発想支援のための手書きによる協調作業システム"，インタラクション'97 論文集，pp.111-118,1997

[3] 高野了成，早川栄一，並木美太郎，高橋延匡："OS/omicron 第 4 版におけるデータ管理機構の設計と実現"，情報処理学会第 54 回全国大会，1F-7,1997

[4] 森永智之，早川栄一，並木美太郎，高橋延匡："OS/omicronV4 のためのマイクロカーネルの開発"，情報処理学会第 37 回プログラミングシンポジウム報告書，pp.165-176,1996

[5] M.K.McKusick,K.Bostic,M.J.Karels, J.S.Quarterman："The Design and Inplementation of the 4.4BSD UNIX Operating System"，Addison-Wesley,1996

[6] Uresh Vahalia："UNIX Internals The New Frontiers"，Prentice Hall,1996

[7] 佐藤元信，森永智之，早川栄一，並木美太郎，高橋延匡："OS/omicron 第 4 版におけるセグメントマッピングによるデバイス管理機構の実現と評価"，電子情報通信学会技術研究報告，pp.49-54,1996

[8] 横田大輔，早川栄一，並木美太郎，高橋延匡："ファイル開発におけるファイルの履歴管理システムの実装方式の比較評価"，情報処理学会第 54 回全国大会，2F-6,1997