OS/omicron第4版におけるオブジェクト指向環境の設計

6Z－3

早川栄一，加藤泰志，佐藤元信，森永智之，並木美太郎，高橋延匡*
東京農工大学　工学部　*拓殖大学　工学部

1．はじめに

計算機で扱えるデータ型の多様化は，システム構造および，その表現方式の改善を必要とする．特に，ビットマップとそこから生成されるコード，図形表現，シンボルのような構造化されたデータを扱う場合には，これらをいかに融合するかが，システムの記述のしやすさと効率化の鍵となる．

そこで，本システムでは，計算機資源およびデータを管理するＯＳ，データ形式および操作を表現する言語処理系，データを生み出すアプリケーションの三つを融合し，オブジェクト指向の枠組みを用いて，データの表現と操作を容易にするシステムを開発する．

本報告では，パターン指向のＯＳであるOS/omicron上での，オブジェクト指向システムの設計について述べる．

2．オブジェクト指向への要求

これまで我々は，OS/omicron第4版と呼ぶ，パターン指向のOSを開発してきたが，この設計・実現過程で次の要求が生じた．

・パターンなどの多様なデータ型に対する統一インタフェースの提供
・データに対する多態性の提供．一つのデータ名に対して複数の表現および複数の処理モジュールを提供する．
・マイクロカーネルからミドルレイヤまで，レイヤの位置に依存しない一貫したインタフェースの提供

さらに，これらを実現する上で，次の2点が必要となった．

・多様なデータ型に対応したシステムの動的な拡張
・プログラム間でのデータ交換の容易さ

Design of an Object-oriented Environment on the OS/omicron V4
Eiichi Hayakawa, Yasushi Kato, Motonobu Sato, Tomoyuki Morinaga, Mitarou Namiki and Nobumasa Takahashi

これに対して我々は，Ｃ言語を用いて，パターンを統合して扱うインタフェースとして「電紙」を提案してきた．これをユーザが容易に表現でき，システムで効率よく管理できる枠組みを提供する．

3．オブジェクト指向環境

3．1　オブジェクト指向の機能

パターンデータ自体の多様性から，パターンをどのように解釈するかが重要となる．パターン自体は，多様な型を持つ．そのために，パターンを定義する型および，それを処理するメソッドが必要である．これをクラスとして定義する．そして，電紙で定義した「紙」に相当する部分をそこから生成したオブジェクトとする．

システムの各層は次の機能を提供する．

(1)言語処理系

クラスなどのオブジェクト指向プログラミングを支援すること．さらに，データ型の動的な拡張や，データの多態性を保証するレイトバインディングを提供する．

本システムは，C++およびJavaの二つの言語処理系を中心に構築する．C++はマイクロカーネルおよびミドルウェアの中で性能面での要求がある部分を担う．また，Javaは，ミドルウェアやアプリケーションの一部を記述する．生成されるオブジェクトはこの両方の言語から共有できる．

(2)OSおよびミドルウェア

OS自体もスレッドや基本的な型のようなシステム定義型を含む．効率などの観点から，システム型自体も再定義できることや，動的にＯＳ型の追加や削除が容易にできることが必要となる．また，デバイスや，ファイルなどの資源をオブジェクトとして抽象化可能な機構を提供する．

(3)アプリケーション

アプリケーション自体も，新しいデータ型を定義し，用いる．他のアプリケーションでデータ型を利用可能にするには，データ型を定義す

るクラスをシステム全体で共有する必要がある．

３．２　システムの全体構成

システムの全体構成を図１に示す．

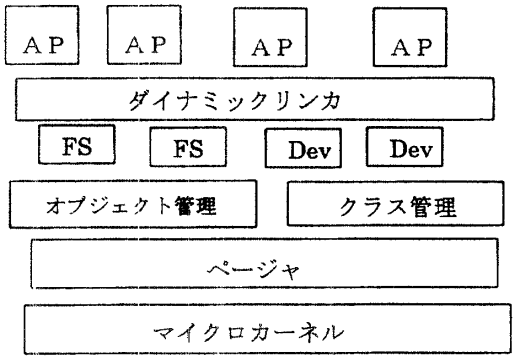

図１　システムの全体構成

システムは，資源管理，タスク管理および記憶保護機構を提供するマイクロカーネル，ミドルウェア群，およびアプリケーションから構成する．システムで提供する名前空間の管理と実体との束縛を，ダイナミックリンカが行う．これにより，すべてのクラス，オブジェクトは名前と実体とを動的に対応づけることができる．

３．３　オブジェクト構造

クラスの名前構造を，ＯＳの持つ名前構造（名前空間）に対応させる．名前空間は，ネットワーク名，マシン名，ファイル名（資源名），クラス名，識別子名を含んでいる．名前はリンケージテーブルと同等の構造を持った木構造として実装される．クラスは，クラス変数やメソッドを持つ．

クラスと，そのクラスを構成するメソッドは，コンパイル単位の集合であるモジュールから構成される．

ミドルウェアおよびAPで定義されたオブジェクトおよびクラスは，システム全体で共有される．これは，従来のＣ＋＋のように独立した複数のアプリケーションにはならないが，データの管理，通信を考えるとこの方が有効である．

また，マイクロカーネルレベルのクラスは起動時に定義可能である．

３．４　オブジェクトの表現

V4 のアドレスは，単一の二次元アドレス空間を持ち，さらに単一階層記憶を実現している．

このとき，

アドレス
＝ポインタ（32bit セグメント：32bit オフセット）
＝オブジェクト ID

となる．セグメントごとに記憶保護ポリシーを変更できるので，オブジェクト単位の保護が可能である．ページング機構は OS 本体から独立し，拡張可能な複数のページャを実現することができる．これを用いて，拡張可能な記憶管理を提供する．言語処理系が提供する new，delete などのメモリ確保演算子と統合する．

C++では，リンケージテーブルを介した間接アクセスによるメソッド呼出しである．また，表の中に型のシグネチャやアクセス属性を入れることで，スタブの自動生成や，動的な型チェックを可能にする．

３．４　言語処理系の実装

我々は，システムの基本機能である，マルチセグメントとダイナミックリンクに対応したC++コンパイラを開発した．Java については，Java VM[1]を OS レベルで提供する．また，リンケージテーブルを用いて Java と他言語とをリンクすることで，メソッドを統合する．さらに，オブジェクトは，C++および Java の間で統一した形にする．

３．５　例外処理

マイクロカーネルが提供する非同期手続き呼出しインタフェース throw[2]により引き起こされる．これは，Ｃ＋＋や Java の持つ例外処理と統合することで，言語処理系と統合した利害処理が可能になる．

４．おわりに

本報告では，一貫したオブジェクトおよびクラス構造を用いて，データおよびメソッドを共有し，オブジェクトの性質に適した環境の設計について述べた．現時点では，ネイティブ環境でのコンパイラ初版および，JavaVM が完成している．今後は，これを用いてシステム全体を実現していく予定である．

謝辞

本研究は文部省科学研究費補助金（奨励研究(A) 09780255 ）により行われた．

参考文献

[1]T.Lindholm et al., The Java Virtual Machine Specification, Addison-Wesley, 1997.

[2]森永他：単一２次元アドレス空間を提供する拡張可能なマイクロカーネルの開発，情処学論文誌，38-5，pp.1016-1024，1997.