5G－7

オペレーティングシステムにおけるオブジェクト管理の一方式

岡野裕之，横関隆，並木美太郎，高橋延匡
(東京農工大学　工学部　情報工学講座)

1．はじめに

パーソナルコンピュータ用のOSを設計する場合，デバイス（周辺装置）をいかに扱うかが重要となる．管理すべきデバイスは，レーザビームプリンタ（LBP），タブレット，イメージスキャナなど数多くあり，今後さらに増加するはずである．これらのデバイスを管理する場合，場当り的にOSを拡張して対処し，それぞれ別々のSVC（スーパバイザコール）体系とするのが従来の方式であった．この方式では次のような問題が生じた．

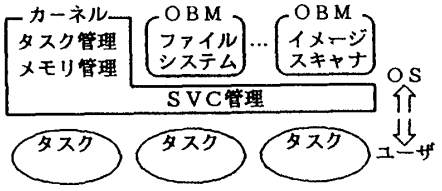
(1) SVCの数が増加し，統一性がなくなった．
(2) 資源を管理するモジュールがOS内に分散するため，資源解放などの後処理が困難となった．

筆者らは，上記(2)が特に深刻な問題であると考える．そこで本報告では，OSを複数の資源（オブジェクト）管理モジュールに分割して構築する一方式を提案する．この方式は，オブジェクトのオープン，オブジェクトへのアクセス，オブジェクトのクローズを，統一したインタフェースにするというものである．オブジェクトのオープンの仕様は，ネットワークシステムに対応可能なように設計した．また，この方式を用いたOSとして，OS/omicron第3版（以下**OMICRON V3**）[1]を実現した．

2．OSの構成

本方式では，OSは図1のように構成されているとする．オブジェクトマネージャ（以下**OBM**）群は，デバイスを管理したり他の**OBM**のインタフェースを加工したりする（図2）．タスクが**OBM**内のオブジェクトをオープンすると，そのタスクにはオブジェクトID（以下ObjID）が返され，以後このObjIDを通じてオブジェクトにアクセスできるものとする．このとき，ObjIDの有効範囲は一つのタスクだけで閉じる設計にしてもよいし，タスクフォースなどのマルチスレッドの形態をとるならば，タスク群でObjIDを共有してもよい．

タスクはSVCを通じて**OBM**を利用でき，そのSVCは次の



OBM：OBject Manager

図1　　OSの構成

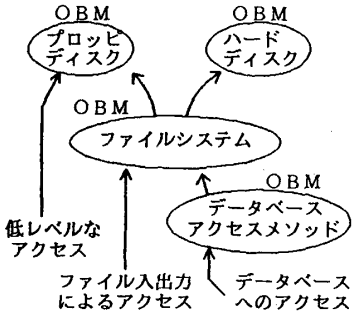


図2　オブジェクトマネージャの利用形態

3つに分類される．

(1) オブジェクトのオープン（ObjIDを得る）．
(2) あるObjIDに対応するオブジェクトをアクセスする．
(3) あるObjIDに対応するオブジェクトをクローズする．

各**OBM**にもタスクと同様に，上記の機能が提供され，他の**OBM**を利用することが可能とする．以下の章では，特に上記(1)の処理方式を述べ，**OMICRON V3**による実現例を示す．

3．オブジェクトのオープン

3．1 オブジェクトを指定するパス名

各**OBM**には名前が付けられていて，それぞれその名前を管理する**OBM**が存在するとする．例えば，**OBM** “A”は“B”，“C”を管理していて，さらに“B”が“D”，“E”を管理しているというように，木構造の階層を持つ．また，各**OBM**が管理するオブジェクトにも，“x”，“y”のような名前が付けられているとする．このとき，次のような名前（パス名）を使って，システム内に存在する任意のオブジェクトを指定する（図3）．

〈パス名〉　：＝　{〈**OBM**名〉@}…〈オブジェクト名〉

現実のシステムと照らし合わせると，図3のA（ルート）の位置にくる**OBM**がファイルシステムで，その下のB，Cが，実際はファイルを指定するパス名と考えることができる．

3．2 オープン処理の手順

各**OBM**には名前の他に，システムで一意に決まる番号が付けられていて，これを**OBM**番号と呼ぶことにする．**OBM**にとってオブジェクトをオープンする手続きとは，システムで一意に決まるObjIDと〈オブジェクト名〉を受け取り，それらの対応をとることであるとする．また，カーネルにとってオブジェクトをオープンする手続きとは，ObjIDと**OBM**番号の対応をとることであるとする．このとき，次のような手順でオブジェクトのオープンを行う．

(1) カーネルが〈パス名〉を受け取る．
(2) カーネルがObjIDを生成する．
(3) カーネルが〈パス名〉の‘@’までを切り出し，対応する**OBM**番号を求める．
(4) 対応する**OBM**に，ObjIDと‘@’以下の〈パス名〉を渡す．

A Object Management Method for Operating System
Hiroyuki OKANO, Takashi YOKOZEKI, Mitarou NAMIKI and Nobumasa TAKAHASHI, Department of Computer Science, Faculty of Tech., Tokyo Univ. of Agri. and Tech.

(5) OBMが与えられた文字列を〈オブジェクト名〉と判断すれば (6) へとぶ．下位のOBMがあれば (3) へ戻る．
(6) オープン処理完了（カーネルではObjIDとOBM番号が，OBM内ではObjIDとオブジェクトがバインドされる）．

上記のような処理を行う場合カーネルは，OBM番号とその下位OBMの名前の組を管理する必要がある．ただし，ルートのOBMの扱いは特別となる．

3．3 本方式の利点

前述のようなオープン処理を採用すると，別マシン上にあるオブジェクトも，一つのパス名で統一的に表現することができる．ここで，ルートのOBMがファイルシステムでその名前を“fs”とし，ファイルシステムが管理しているOBMに“ns”があるとする．このとき，

fs@ns@マシンｘ；fs@ファイルＡ

のような〈パス名〉のオープン処理の際，“ns”が

マシンｘ；fs@ファイルＡ

を受け取った時点でマシンｘ上のOBMと交信し，マシンｘ上のOBMが

fs@ファイルＡ

をオープンする（図4）．つまり，次のような仕様の〈オブジェクト名〉を受け取って，別マシン上のObjIDと自分のマシン上のObjIDのバインドをとるOBM“ns”を作れば，別マシン上のオブジェクトがオープン可能となる．

〈オブジェクト名〉 ：＝ 〈マシン名〉；〈パス名〉

このような，別マシン上のオブジェクトをオープンする機構さえあれば，これの上位の機能としてマシン名を隠す機構を作るなどして，ネットワーク透過なファイルシステムとすることも可能である．

4．オブジェクトへのアクセス

本方式では，オブジェクトへのアクセスはObjIDへの操作である．タスクがObjIDへの操作を行う場合，カーネルがそのObjIDに対応するOBMに対して，ObjIDと操作方法を渡す．前章で述べたOBM“ns”の場合，ObjIDに関する操作を別のマシン上に通信で渡し，そのマシン上の対応するObjIDに対して操作を代行してもらう．このとき，呼び出しを行ったタスク側のバッファに，オブジェクトからデータを送る処理（ファイルの読込みなど），あるいはこの逆の処理が必要な場合は，“ns”が中継してデータ転送を行わなければならない．

5．オブジェクトのクローズ

本方式では，どのタスクがどのObjIDを使用しているかをカーネルが把握しており，さらにObjIDにはOBMが対応している．タスクが明示的にオブジェクトをクローズする場合，カーネルがOBMに対してObjIDを渡し，クローズを指示する．タスクが異常終了した場合は，そのタスクがオープンしていたすべてのオブジェクトに関して，カーネルが同様なクローズ処理を行う．

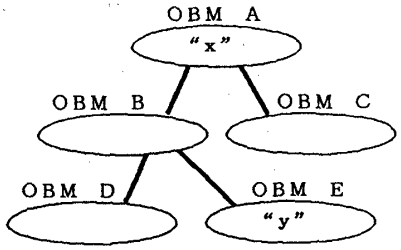


図3 オブジェクトマネージャの階層構造と資源を一意に指定するパス名

6．本方式によるOSの実現

本方式を用いたOSとして，OMICRON V3を実現した．OMICRON V3はタスクフォースと呼ばれるマルチスレッド機能を有しており，タスク群がプログラムを共有できる．ユーザにはオブジェクトにバインドされるIDとして，fd（ファイル記述子）が与えられる．fdの有効範囲はタスクフォースである．OS内で用いるObjIDは，システムで一意に決まるタスクフォースIDとfdの組で表現される．

OMICRON V3におけるオブジェクトのインタフェースは，ランダムアクセス可能なファイルとして統一した．その一例を次に示す．

write（fd，オフセット値，バッファへのポインタ，サイズ）；

このSVCでオブジェクトの任意のオフセットへの書込みができる．デバイスの場合は，このファイルイメージをメモリマップドI/O領域として捕らえ，それぞれのデバイスに合ったインタフェースにする．

カーネルとOBMとの通信にはメッセージ通信を用い，各OBMごとにメッセージキューを設けることで実現した．オープン，アクセス，クローズのメッセージ群の他に，親子タスクフォース間でオブジェクトを引き継ぐための操作メッセージを用意した．

OMICRON V3では，タスクフォースもオブジェクトとして扱われる．よって，子タスクフォースの生成はオブジェクトのオープンと等価な意味となる．これによって，ネットワークシステムを介して別マシン上に子タスクフォースを生成することも可能になる．

7．おわりに

OSにおいて，ファイル，デバイスなどの資源をオブジェクトとして統一的に管理する一方式を示し，特にそのオープンの処理について述べた．この方式は，ネットワークシステムを構築する際の枠組みとして利用できる．

参考文献

[1] 岡野，横関，並木，高橋：OS / omicron 第3版の設計と実現，情報処理学会OS研究会資料，45 - 11 (1989)．

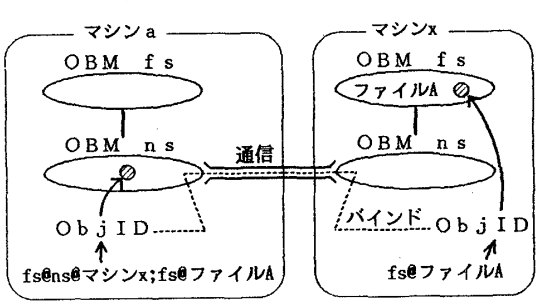


図4 他のマシン上のオブジェクトのオープン