システムソフトウェアと
オペレーティング・システム 69−19
（１９９５．６．２３）

オブジェクトの堆積モデルに基づく高機能ディレクトリ・オブジェクト

北島 研二
<kita@ocean.ie.u-ryukyu.ac.jp>

新城 靖
<yas@ie.u-ryukyu.ac.jp>

翁長 健治
<onaga@ie.u-ryukyu.ac.jp>

琉球大学　情報工学科
〒903-01 沖縄県西原町千原１番地
電話：098-895-2221　内線3266
Ｆａｘ：098-895-2688

概要　オブジェクトの堆積（object-stacking）とは、object-basedシステムを構造化するためのモデルである。オブジェクトの堆積では、オブジェクトの層をつくることで、オブジェクトの機能を組み合わせることができる。この論文では、オブジェクトの堆積に基づき、融合、半透明、フィルタ機能を提供するディレクトリ・オブジェクトについて述べる。また、これらの高機能ディレクトリ・オブジェクトの実現において、キャッシングの利用について述べる。

Directory objects providing high-level functions based on the Object-Stacking Model

Kenji Kitajima
<kita@ocean.ie.u-ryukyu.ac.jp>

Yasushi Shinjo
<yas@ie.u-ryukyu.ac.jp>

Kenji Onaga
<onaga@ie.u-ryukyu.ac.jp>

Department of Information Engineering
University of the Ryukyus
Nishihara, Okinawa 903-01, Japan
Phone: +81 98 895 2221 Ext. 3266
Fax: +81 98 895 2688

Abstract　Object-stacking is a model for structuring object-based systems. In object-stacking, by creating layers of objects, the functions of the object can be used together. This paper presents directory objects that provide high-level functions based on the object-stacking model. High-level functions mean merge, translucent and filtering. To implement these directory objects, caching is used.

1 はじめに

オブジェクトの堆積（object-stacking）は、object-basedシステムを構造化するモデルである[1]。このモデルの特徴は、一様なインタフェースを持つオブジェクトの層を形成することで、それらのオブジェクトが持つ機能を統合して利用する点にある。オブジェクトの堆積において個々のオブジェクトは、単純な機能を提供する。それらのオブジェクトは、オブジェクト識別子（OID:object identifier）と遠隔手続き呼出し（RPC:Remote Procedure Call）により結合され、それらの機能を合わせ持つようなオブジェクトを形成する。オブジェクトの堆積は、プロセス間通信におけるクライアント・サーバ・モデルが成り立っているobject-basedシステムで広く利用可能である。例えば、分散オペレーティング・システムのファイル・システムや、インターネット上の情報提供システムであるThe World-Wide Web（W3）などで広く利用可能である[2]。

ディレクトリ・オブジェクト(directory object)は、ファイル・システムにおいてオブジェクトに名前をつける役目を持っている。オブジェクトの堆積に基づくファイル・システムにおいて、単純なディレクトリ・オブジェクトは既に実現されている。

多くのアプリケーションは、複数のディレクトリにわたって検索する。例えば、環境変数PATH、MANPATH、FONTPATHなどにより指定されたディレクトリを検索する。アプリケーションは、自分で複数のディレクトリにわたって検索しなければならない。

本論文では、はじめに、融合ディレクトリ・オブジェクト（merge directory object）について述べる。融合ディレクトリ・オブジェクトは、複数のディレクトリ・オブジェクトの内容をすべて含むような1つのディレクトリ・オブジェクトをつくる機能を提供する。このオブジェクトの機能により、それぞれのアプリケーションが実現している、複数のディレクトリを検索する機能が不必要となる。

次に、半透明ディレクトリ・オブジェクト（translucent directory object）について述べる。半透明ディレクトリ・オブジェクトは、あるディレクトリ・オブジェクトのリストの内容とそれ自身が持つリストの内容を合わせ持った1つのディレクトリ・オブジェクトをつくる機能を提供する。もとのディレクトリ・オブジェクトの内容を変えずに、新たなエントリを追加したり、削除したりすることが可能になる。

さらにフィルタ・ディレクトリ・オブジェクト（filter directory object）について述べる。フィルタ・ディレクトリ・オブジェクトは、あるディレクトリ・オブジェクトから、あるパタンにマッチした名前だけをもつディレクトリ・オブジェクトをつくる機能をもっている。これらの高機能ディレクトリ・オブジェクトの実現において、キャッシングを利用している。

最後に、関連した研究との比較について述べる。

2 オブジェクトの堆積

オブジェクトの堆積とは、一様なインタフェースを持つオブジェクトを積み重ねることである。インタフェースとは、オブジェクトが受け付けることのできる手続きの集合である。オブジェクトを積み重ねるとは、上位層のオブジェクトが下位層のオブジェクトのOIDを持ち、かつ、上位層のオブジェクトが、その機能を実現するために下位層のオブジェクトの機能を呼び出すことである。一様なインタフェースを持つオブジェクトを利用することにより、オブジェクトを積み重ねる順番を自由に変えることができる。

オブジェクトの堆積において、自分自身のインタフェースと下位層のオブジェクトのインタフェースが同一であるようなオブジェクトを堆積可能オブジェクト（stackable object）という。堆積可能オブジェクトは、下位層のオブジェクトのOIDを保持している。

図1では、次の3つの機能を持つオブジェクトが積み重ねられている。

(a1) データを保持するもの
(b1) 大文字を小文字に変換するもの
(c1) 行単位のソートを行うもの

最上位層のファイル・オブジェクト（c1）をア

クセスすることにより、これらの機能が統合される。すなわち、小文字に変換されソートされたデータを保持しているファイル・オブジェクトが作られている。（b1）と（c1）は、下位層のオブジェクトのOIDを保持しているので、堆積可能オブジェクトである。（a1）は、基底オブジェクトである。

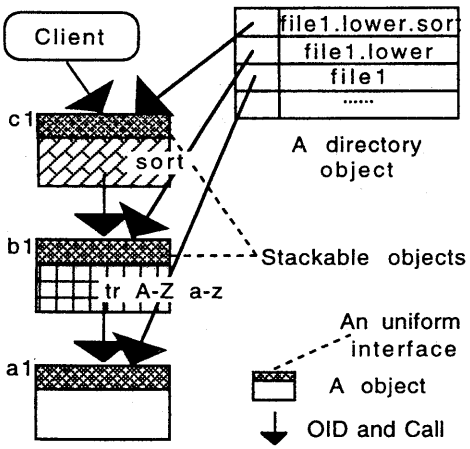

図１：オブジェクトの堆積による機能の統合

２．１　オブジェクトの堆積に基づくファイル・システムにおけるディレクトリ・オブジェクト

図１では、ディレクトリ・オブジェクトは、各ファイル・オブジェクトを文字列の名前で参照することがを可能にしている。ディレクトリ・オブジェクトは、オブジェクトの文字列の名前とOIDを結び付けている。

ディレクトリ・オブジェクトは、以下のようなインタフェースを通して、自らの機能を提供している。

create：ディレクトリ・オブジェクトを生成する
destroy：ディレクトリ・オブジェクトを削除する
dir_lookup：オブジェクトの名前で検索しOIDに変換する
dir_register：リストにオブジェクトの名前とOIDの組を登録する
dir_remove：リストからオブジェクトの名前とOIDの組を削除 する
dir_readdir：リストを全て返す

これらの手続きのうち、生成、削除する手続きは、すべてのオブジェクトが持っている。残りの手続きは、ディレクトリ・オブジェクト固有のものである。

3　融合ディレクトリ・オブジェクト

融合ディレクトリ・オブジェクトは、複数のディレクトリ・オブジェクトの内容をすべて含むような１つのディレクトリ・オブジェクトをつくる機能を提供する。利用者は、検索したい複数のディレクトリ・オブジェクトを下位層のオブジェクトとして、融合ディレクトリ・オブジェクトを生成する。この融合したディレクトリ・オブジェクトを検索することにより、従来のアプリケーションが独自に実現している複数のディレクトリを検索する機能は不必要となる。

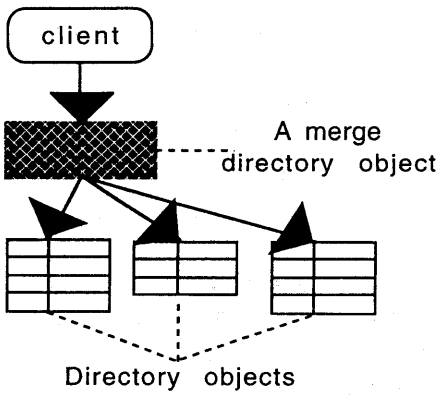

図２：融合ディレクトリ・オブジェクトによる複数のディレクトリ・オブジェクトの融合

３．１　インタフェース

融合ディレクトリ・オブジェクトに、名前を登録、削除する手続きを持たせるか、持たせないかという議論がある。オブジェクトの名前を登録、削除する機能を持たせることにより、普通のディレクトリ・オブジェクトと区別なく利用できるという利点がある。一方、融合という性質から、オブジェクトの名前を登録、削除する機能は本来必要がないとも考えられる。

本研究では、融合ディレクトリ・オブジェクト

にオブジェクトの名前を登録、削除する機能を持たせないことにする。すなわち、融合ディレクトリ・オブジェクトは、読み出し専用である。この場合は、オブジェクトの名前を登録、削除する機能については、4章で述べる半透明ディレクトリ・オブジェクトにより、付加することができる。

3．2　実現方法

融合ディレクトリ・オブジェクトの実現方法として、キャッシングを利用する方法と、利用しない方法がある。両方の比較を dir_lookup 手続きをつかって述べる。キャッシングを利用する方法では、全ての下位層のオブジェクトに、dir_readdir 要求を送り、メモリ中に融合したデータをつくる。このとき、同じ名前のオブジェクトがある場合は、先に指定されたディレクトリ・オブジェクトに含まれているものを選択する。融合ディレクトリ・オブジェクトは、メモリ中のデータを検索し、結果を利用者に返す。キャッシングを利用しない方法では、全ての下位層のディレクトリ・オブジェクトに dir_lookup 要求を送り検索する。マッチした場合、融合ディレクトリ・オブジェクトは、下位層オブジェクトから受け取った結果を利用者に返す。マッチしなかった場合、利用者にエラーを返す。

本研究では、融合ディレクトリ・オブジェクトの実現にキャッシングを利用する。これにより、実現が容易になる。さらに、キャッシングを利用することにより、検索が高速化される。

融合ディレクトリ・オブジェクトのデータを安定に保存するために、永続構造体ライブラリを使用する[3]。永続構造体ライブラリでは、揮発データ、永続固定長データ、永続可変長データの3つデータに分けて考える。

融合ディレクトリ・オブジェクトは、以下の構造体 mdir により実現されている。

```
struct mdir {
   struct mdir_volatile {
      rwlock_t          md_lock;
      mdir_stat_t       md_stat;
      dir_entry_t       cache<>;
      int          md_dfix_updated;
      int          md_dvar_updated;
   } md_volatile ;
   struct mdir_dfix {
      oid_t             mddf_oid;
      attr_t            mddf_attr;
      resc_time_t       mddf_created;
   } md_dfix ;
   struct mdir_dvar {
      oid_t             mddv_lowers<>;
   } md_dvar ;
}
```

この記述は、XDR(eXternal Data Representation)に定義されたデータ型に基づいている。XDRとは、Sun RPCで用いられているデータの記述とコード化のための標準である。

構造体mdirのメンバである構造体md_volatileは、揮発データである。ロック、状態、キャッシュ、永続固定長、可変長の更新フラグで構成される。構造体 md_dfix は、永続固定長データである。自分のOID、オブジェクトの属性、オブジェクトの生成時間から構成されている。構造体 md_dvar は、永続可変長データである。下位層のディレクトリ・オブジェクトのOIDを要素とする配列である。

4　半透明ディレクトリ・オブジェクト

半透明ディレクトリ・オブジェクトは、登録リスト、削除リストと、下位層に1つのディレクトリ・オブジェクトを持つ。そして、下位層のディレクトリ・オブジェクトの内容に、登録リスト、削除リストの内容を反映した内容をもつ、1つのディレクトリ・オブジェクトをつくる機能を提供する。利用者がオブジェクトの名前を登録した場合、半透明ディレクトリ・オブジェクトの登録リストに追加される。利用者がオブジェクトの名前を削除した場合、半透明ディレクトリ・オブジェクトの削除リストに追加される。どちらの場合でも、下位層のディレクトリ・オブジェクトの内容は変更されない。

半透明ディレクトリ・オブジェクトは、読み出し専用のディレクトリ・オブジェクトから、読み書き可能なディレクトリ・オブジェクトをつくり出すことができる。たとえば、3章で述べた融合ディレクトリ・オブジェクトは読み出し専用であ

った。融合ディレクトリ・オブジェクトに、書き込みの機能を付加するためには、この半透明ディレクトリ・オブジェクトを上に堆積させればよい。また、半透明ディレクトリ・オブジェクトを利用することにより、複数のマシン・アーキテクチャ用のバイナリ・プログラムをつくることが容易になる。1つのソース・プログラムを含むディレクトリの上に、マシン・アーキテクチャごとに半透明ディレクトリ・オブジェクトを堆積させればよい。

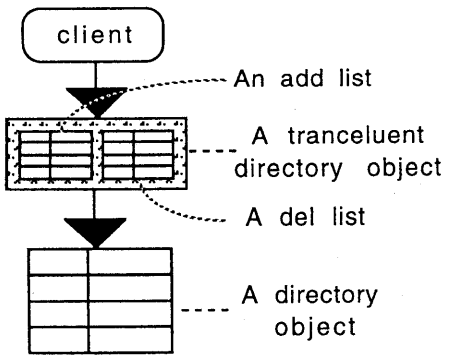

図3：半透明ディレクトリ・オブジェクト

4．1　実現方法

半透明ディレクトリ・オブジェクトの実現方法もまた、キャッシングを利用する方法と、使用しない方法がある。それらの方法を dir_lookup をつかって述べる。キャッシングを利用する方法では、はじめに dir_readdir 要求を下位層オブジェクトに送り、メモリ中に下位層のディレクトリ・オブジェクトの内容を読み込む。このデータに、それ自身が保持する登録リストのデータを追加し、削除リストのデータを削除する。あとは、融合ディレクトリ・オブジェクトと同じである。キャッシングを利用しない方法は、はじめに削除リストを検索する。削除リストにエントリがあった場合、利用者にエラーを返す。削除リストになかった場合は、登録リストを検索する。登録リストにあった場合、利用者にその名前に対応するOIDを返す。登録リストにもなかった場合、下位層のディレクトリ・オブジェクトに、dir_lookup を送る。その検索結果は、そのまま、利用者に返す。

本研究では、半透明ディレクトリ・オブジェクトの実現にキャッシングを利用することにする。半透明ディレクトリ・オブジェクトの構造体tdirは、融合ディレクトリ・オブジェクトと同様に、永続構造体ライブラリを利用して実現を行う。揮発データは、融合ディレクトリ・オブジェクトと同じである。永続固定長データは、自分のOID、オブジェクトの属性、オブジェクトの生成時刻、下位層のディレクトリ・オブジェクトのOIDで構成される。永続可変長データは、登録リスト、削除リストで構成されている。

5　フィルタ・ディレクトリ・オブジェクト

フィルタ・ディレクトリ・オブジェクトは、下位層のディレクトリ・オブジェクトから、パタンにマッチした名前のみを持つディレクトリ・オブジェクトをつくる機能を提供する。パタンは、オブジェクトを生成するときに与えられる。

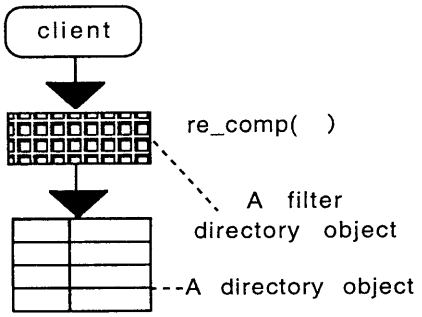

図4：フィルタ・ディレクトリ・オブジェクトとディレクトリ・オブジェクトが堆積した様子

5．1　実現方法

フィルタ・ディレクトリ・オブジェクトの実現方法もまた、キャッシングを利用する方法と、キャッシングを利用しない方法がある。それらの方法を、dir_lookup をつかって述べる。キャッシングを利用する方法では、下位層のディレクトリ・オブジェクトの内容をメモリ中に読み込む。この時、パタン・マッチした名前のみキャッシュされる。あとは、融合ディレクトリ・オブジェクトと同じである。キャッシングを利用しない方法では、利用者から検索要求が出された名前が、フィル

タ・ディレクトリ・オブジェクトが持っているパタンと比較する。この名前がマッチした場合、下位層のディレクトリ・オブジェクトにdir_lookupを送る。マッチしなかった場合、利用者にエラーを返す。

本研究では、フィルタ・ディレクトリ・オブジェクトの実現においてもまた、キャッシングを利用することにする。フィルタ・ディレクトリ・オブジェクトは、パタン・マッチングを行うのに、C言語のライブラリ関数re_comp、re_exec関数を用いる。

フィルタ・ディレクトリ・オブジェクトの構造体の実現もまた、永続構造体ライブラリを利用している。揮発性データと永続固定長データは、半透明ディレクトリ・オブジェクトと同じである。永続可変長データは、パタンで構成されている。

6 他の研究との比較

6．1 多重名前空間を導入した分散共有格納庫システムとの比較

文献[4]では、分散共有格納庫（DSR:Distributed Shared Repository）と呼ばれる永続処理とプロセス間通信を実現するための仕組みに、多重名前空間を導入する方法が提案されている。この論文で述べられている多重名前空間は、半透明ディレクトリ・オブジェクトにより、実現することができる。

本研究の特徴は、第1に、融合機能とフィルタ機能があるという点である。DSRの多重名前空間は、融合、フィルタ機能を持っていない。第2に、ディレクトリ単位で管理する点である。DSRの多重名前空間は、ファイル・システム単位で管理する。ファイル・システム単位で、半透明機能を利用したい時には、DSRの多重名前空間の方が簡単である。同様の機能を半透明ディレクトリ・オブジェクトでは、コマンド・レベルにおいて、再帰的に堆積させることで実現することができる。

6．2 TFSとの比較

Sun Microsystems社は、NFS（Network File System）の拡張機能として、半透明ファイル・サービス（TFS：Translucent File Service）を提供している[5]。TFSは、与えられたファイル・システムの内容と、それ自身が持つ内容を合わせ持つようなファイル・システムをつくる機能を提供する。もとになるファイル・システムとして、複数のファイル・システムを指定することができる。

本研究の特徴は、融合、半透明機能を別々に提供する点である。TFSは、融合、半透明機能を一緒に提供している。また、本研究では、フィルタ機能を提供することができる。

7 おわりに

本論文では、オブジェクトの堆積に基づく高機能ディレクトリ・オブジェクトについて述べた。高機能ディレクトリ・オブジェクトとしては、融合、半透明、フィルタディレクトリ・オブジェクトについて述べた。これらの高機能ディレクトリ・オブジェクトの実現においては、キャッシングすることで、実現を容易にし、手続きの実行を高速化することができる。

この論文で述べた高機能ディレクトリ：オブジェクトの内部設計を完了した。これらは、現在開発中である。今後の課題として、キャッシュの一貫制御を効率的に実現する。

参考文献

[1] Y.Shinjo and Y.Kiyoki: "The Object-Stacking Model for structuring Object-Based Systems", IEEE Proc 2nd International Workshop on Object Orientation in Operating System (I-WOOOS'92), pp.328-340 (1992).

[2] Y.Shinjo and S.Zakimi and S.Kyan: "Object-Stacking in the World-Wide Web",IEEE Proc 4th International Workshop on Object Orientation in Operating System (I-WOOOS'95) (To appear)

[3]島袋、新城、翁長: "オブジェクトの堆積モデルに基づく間接オブジェクトの実現"、情報処理学会研究報告94-OS-65、Vol.94、No.64、pp.113-120 (1994).

[4] 加藤、坂田、益田: "分散共有格納庫への多重名前空間の導入について"、情報処理学会第5回コンピュータシステム・シンポジウム、情処シンポジウム論文集、Vol.93、No.7、pp.115-122 (1993).

[5] "SunOS Reference Manual", Sun Microsystems, Inc. (1988).