# 分散ファイルシステム ALeFs での属性による連想アクセス機能

稲村 浩† 盛合 敏†

ファイルなどの永続的なオブジェクトの名前の解釈について述べる．従来より階層的な名前付けと属性による名前付けは独立に提案され用いられてきた．本研究ではこれらを統合し，利用者に対して自由度が高く連想的なアクセス手段を提供するインタフェースについて述べ，これらを実現する分散ファイルシステム ALeFs の設計について議論する．また，動的分散環境への応用も可能になったことも述べる．

# Associative Access Capability in ALeFs Distributed File System

Hiroshi Inamura† and Satoshi Moriai†

This paper discusses the interpretation of the naming scheme of persistent objects, such as files. Two kinds of name resolution systems, namely hierarchical and attribute-based, have been proposed and are now used independently. We must integrate these name resolution systems in order to achieve more flexible and associative naming schemes. We explain the design of our distributed file system named "ALeFs". Our integration makes it possible to cope with the dynamically changing environment.

## 1. はじめに

進んだ分散環境においては高速なアクセスが可能なファイル，複数の CPU やビットマップディスプレイなど様々な資源が利用可能になる．分散オペレーティングシステムにおいて，資源はオブジェクトとして抽象化されている．こういったシステムでは，ファイルシステムはファイルを管理するのみならず，一般のオブジェクトすなわち資源を利用者にアクセス可能にするものである．利用者に使いやすいオブジェクト名の表現を利用可能にすることは，利用者インタフェースの面で重要である．また，ファイルシステムに一貫した構成を与えることは，利用者インタフェース面の改良にとどまるのみならず，分散システム自体の骨格を与えることである．

本論文では分散システムにおける永続的なオブジェクトの名前とその解釈について考察する．こういったオブジェクトの代表としてファイルを考え，ファイルシステムを考察の対象とする．

従来のファイル名解釈システムは大きく以下の2つにまとめることができる．

**階層的な名前解釈システム**

階層的な名前解釈システムではディレクトリの持つ構造の階層性をもとに名前を決定し，各々のディレクトリに順に問い合わせることによって名前を解釈する．

階層的なファイル名解釈システムの一例として UNIX*[12] のファイルシステムがある．UNIX におけるファイル名はパスコンポーネントを "/" でつなげることで構成される．例えば "/usr/bin" ならば，"usr"，"bin" がパスコンポーネントである．名前を構成する個々のパスコンポーネントを単位として，1つ前までのパスコンポーネントの解釈結果に基づいて与えられたパスコンポーネントを解釈してゆく．

また，階層名前空間で利用者によるカスタマイズを提供するためにリンクが用いられている．Comer ら[2] は1つのパスコンポーネント *nameString* を解釈するために必要な情報として

〈*nameString*, *context*, *environment*〉

という3つ組を挙げている．これは階層名前空間の解釈のモデルであると同時にリンクで保持すべき中間結果の形の1つの形式を示した例でもある．

**問い合わせ的な名前解釈システム**

問い合わせ的な名前解釈システムでは名前を構成するそれぞれの語が独立して意味を持ち，前後関係によらず一定の意味を持つ．

† NTT 情報通信網研究所
NTT Network Information Systems Laboratories

* UNIX is a trademark of AT&T Bell Laboratories, in USA.

こういったシステムには記述名表現[13]や属性による名前システムがある.

例えば記述名では意味ネットワークをベースにしており,{属性型=層性値} という形式で要求する資源にあるべき特徴を述べることで対象を特定する方式である.

属性による名前システムもいくつかの提案[1),7),8)]がなされている.なお,これらは階層的で属性の意味づけも固定されて運用されているものもあるが,基本となるモデルにはそういった制限は本質的にはない.例えば Grapevine[1)] では2連構成,Clearing House[7)]では3連構成である.ここで言う連とは属性記述の単位である.また,バージョンコントロールを特に念頭に置いて属性名を導入したもの[6)]もある.

**統合の必要性**

階層的,問い合わせ的な名前解釈システムはそれぞれ独立に提案,運用されているが,より自由度の高いファイル名解釈システムの構築のためにこれらを統合することが必要である[3),9),4)].

この統合を外部のデータベース管理システムではなくファイルシステム自体で行うことで,変化するファイルの状況を即座に反映できる,通常のユーティリティがそのまま利用できるといったメリットがある.

本論文では,以下2章で従来の名前付けの問題点を指摘し,3章で階層型と属性型の名前付けの統合について提案し,4章で分散ファイルシステム ALeFs の設計について議論し,5章で性能について評価し,6章で関連研究と比較し,7章でまとめる.

## 2. 従来の名前付けの問題点

従来の階層的なファイルシステムは広く使われ成功している.しかし,システムの規模が大きくなるにつれ,問題も明らかになっている.ここでは,望まれる名前付けの自由度と,その必要性から見た従来の名前解釈システムの問題点について考える.今後より広く分散システムが利用されていくことを考えると,分散ファイルシステムの性能面での向上は当然要求されるところであるが,機能面での不満も表面化しつつある.本論文では,より自由度の高い名前付けを特徴付ける以下の要求を満足する設計について述べる.

### 2.1 複数の観点からの分類

情報の共有や再利用のためには,多くの観点から行われた分類を用いて,情報に確実にアクセスできることが重要である.通常の階層的な名前空間では,複数の分類の観点が無秩序に混在するためにわかりにくさを生んでいると考えられる.例えば UNIX オペレーティングシステムは,以下の観点から分類を行う.

- 由来
  (/usr/ucb,/usr/5bin,/usr/local など)
  そのファイルが何故そこにあるか.
- 物理的な性質
  (/tmp,/dev など)
  特別な構造のファイルシステムであることを示すこともある.例えば/tmp はメモリベースのファイルシステムであることもある.
- 管理上の区分
  (/etc,/usr など)
  利用者向けか,管理者向けか.
- 内容,目的による区分
  (/usr/man,/usr/include など)
  明示的に内容を示している.

これらの分類の観点をサポートする上で問題なのは,これらの階層の様々なレベルにちらばっていることである.利用者は分類の観点が何であるかを把握するのに労力を必要とする.すなわち,**階層のあらゆる場所に分類の軸がちらばっているため,ある1つの観点から情報を捕えることが難しくなっているのである.** 例えば,デバイスファイルについてその由来や性質を名前から直接判断することはできない.

また,分類上適当だと思われる場所でも利用者に書き込み許可がないとその場所に記録することはできない.これは管理上の性質が,分類の観点とミスマッチを起こしているためである.

### 2.2 動的な分散環境への対応

分散システムでは提供される資源の数,種類ともに日々増加,更新されてゆく.1ノードでは大した数ではない更新も,システム全体では大きな変化として表れる.

動的に変化する環境では,状況に応じて利用する資源の切り替えが起こる.そういった際には固有の名前に依存した記述が障害になる.

利用者がそのたびに名前に依存した記述を変更するのは煩瑣であり,その数が多ければ人手で行うのはほとんど不可能になる.従って自動化のためのメカニズムが必要である.

### 2.3 多数のファイルへの直接的なアクセス

分散システムの規模が大きくなるにつれて,扱う資源の数は急速に増大する.ファイルを例にすれば,扱

うファイルの数が増えるとファイルの置き場所を忘れることさえある．また，複数の利用者による共同作業においては，データの共有，相互のアクセスが必要になる．個人のディレクトリにあるデータを集める場合，そのメンバの名前付けの慣習（naming convention）を知らない場合，アクセスが難しい．共有ディレクトリにおいてもその配置を知らない利用者には何らかのアクセスの補助手段を提供する必要がある．例えば，特定個数のキーワードで連想的にアクセスできることが望ましい．

## 3. 階層型と属性型の名前付けの統合

2章で述べた問題を解決するために，属性に基づいたアクセスのためのメカニズムを導入する．しかし既存の環境との適合性を鑑みると，階層的な名前空間と共存できることが望ましい．すなわち，「ファイルシステムにちらばっているファイルの中から要求に応じた属性に基づく問い合わせによって必要なものを抜き出し，それに名前をつけて階層ファイルシステムに保存し利用する」といった一連のサイクルをサポートすることが必要である．これを階層的な名前付けと問い合わせ的な名前付けを統合することにより実現する．

本論文で提案する階層ベースと属性ベースの名前解決システムの統合環境の概念図を**図1**に示す．図中で実線矢印は通常のアクセスを，破線矢印は問い合わせによるアクセスを意味している．

ここでは利用者はアプリケーションを使うためのコマンドサーチパスを，固定的に設定するのではなく環境に問い合わせる（図中破線矢印1）ことで得ている．

属性を用いて問い合わせを行うことで，ファイルの実体がファイルシステムのどこに格納されているかを知らずにアクセスすることが可能になる．

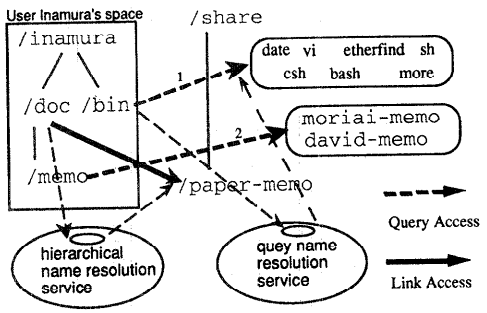



**図1** 階層ベースと属性ベースの名前付けが統合された環境の概念図

Fig. 1 Conceptual overview of the environment that integrates hierarchical and attribute-based naming schemes.

例えばあるファイルに特定の属性を付加すれば，その属性をキーにアクセスすることで，そのファイルが階層空間中のどこにあってもただちに利用者からアクセス可能（図中破線矢印2）になる．しかもその属性が付加されている限り名前や格納位置の変更にも影響を受けない．

問い合わせ的なファイル名解決システムへの問い合わせの条件を保持して再利用，共有するためにはシンボリックリンクを利用する．

これらのメカニズムによって最初に述べた2つの名前解釈の方式を1つに統合する．

## 4. ALeFs の設計

提案した統合環境を実現する分散ファイルシステムALeFs について述べる．ALeFs では階層的ファイルシステムに属性を導入し，それに基づくアクセス方法を加えた．

利用形態としては例えば2.1節で述べた分類の観点それぞれを1つの属性に対応させる．利用者はいくつかのキーワードを知っていればよく，すべてのファイルの置き場所を知る必要はない．

属性によるアクセスを実現することで，階層的オブジェクト名解釈システムと問い合わせ的オブジェクト名解釈システムを統合している．

### 4.1 属性によるファイル操作

属性でのファイル操作は通常のファイル名の部分に属性を記述することで行う．属性操作のための拡張された記法を**表1**にまとめた．また，UNIX 環境における利用の形態を**表2**にまとめた．ここで問い合わせに用いる文法は**表3**のとおりである．

#### 4.1.1 属性の付与

“agenda. txt” というファイルを例にして，ALeFs の機能を説明する．以下では “agenda. txt” の属性を調べている．“agenda. txt@ :” というのは，“agenda. txt” ファイルの属性を表現している特殊な隠れディレクトリであり，引き数なしの ls コマンド等では見えないが名前を陽に指定されることで機能する．これを調べることで，そのファイルの属性を知ることができる．

```
ls-R agenda.txt@ :
type owner

agenda.txt@ :/type :
normal
```

```
agenda.txt@ :/owner :
inamura
```

これで“agenda.txt”が属性“type”を持ち，その値“normal”から通常のファイルであることが分かる．

次の例は“agenda. txt”に“contents”属性を定義して．“memo”という値を付加する．

```
$mv agenda.txt agenda. txt
        @contents : memo
```

#### 4.1.2 何が属性として定義されているか

現在の環境で何が属性として定義されているかを知らないと，検索には不便である．ALeFs では特別なディレクトリとして“…attrib”を提供している．これも引き数なしの ls コマンドでは見えない隠れディレクトリである．この下には ALeFs で定義されている属性がディレクトリの形で現れ，すべての属性値がその下にファイルの形で現れる．ここのファイルの大きさは 0 である．なお，例中の“(etc..)”は省略を示す．

```
$ls…attrib
contents name type package
```

また，その属性値も含めると，

```
$ls-R…attrib
contents name type package owner

…attrib/contents :
meeting memo manual (etc..)

…attrib/type :
normal executable (etc..)

…attrib/package :
emacs gnu info (etc..)

…attrib/owner :
inamura moriai miura (etc..)
(etc..)
```

この中でシステムが自動的に付加する“owner”，“type”属性への利用者による変更はできない．これらの属性は別の（例えば chmod などの）操作コマンドの結果を反映する．

#### 4.1.3 属性値を用いた問い合わせ

表 1 の記述に則して説明する．ALeFs では問い合わせのために特別な文字として“%”を使う．通常の階層的なファイル操作では“/”をパスコンポーネントの区切りに使うが，ALeFs の問い合わせ空間での属性操作には“%”を区切りに使う．

ALeFs ではすべてのディレクトリの下に“…”という名前の問い合わせのためのディレクトリがある．“…”は隠れディレクトリであり，明示的に指定された場合のみ応答する．この下に検索条件をファイル名で指定することで問い合わせが可能になる．

例えば，以下の最初の例では実行形式で所有者が“inamura”または“moriai”のものを検索している．次の例ではコマンド“date”を実行している．

```
$ls…/type :executable%owner^inamura, moriai
adb(etc..)

$…/type :executable/date
Fri Mar 6 15 :01 :28 JST 1992
```

例えば，環境変数“PATH”は以下のように設定できる．

```
PATH=
…/type :executable%owner^inamura, moriai
```

同様に検索条件をリンクで固定して検索することもできる．以下の例では“memos”というディレクトリをアクセスすることで問い合わせが起こり，最近の変

**表 1** 属性の記述法
Table 1 Notation for an attribute-based access in ALeFs.

| 記述法 | 意味 |
|---|---|
| 〈ファイル名〉@: | 指定ファイルの属性リストを表すディレクトリ |
| 〈ファイル名〉@〈属性〉: | 指定ファイルの属性を表すディレクトリ |
| …/〈検索条件式〉 | 検索条件を満たすファイルのリストを表すディレクトリ |
| …attrib | 全ファイルの属性リストを表すディレクトリ |

**表 2** UNIX 環境での利用
Table 2 Usage patterns in the UNIX environment.

| 意味 | 記述法 |
|---|---|
| 属性の付与 | mv 〈パス名〉 〈パス名〉@〈属性〉: 〈属性値〉 |
| 属性の一覧 | ls …attrib |
| 属性と属性値の一覧 | ls -R …attrib |
| 属性による問い合わせ | ls …/〈検索条件式〉 |
| 検索条件の保存 | ln -s …/〈検索条件式〉 〈パス名〉 |

**表 3** ALeFs における検索条件の文法
Table 3 Syntax for query expression in ALeFs.

```
〈検索条件式〉⇒〈検索条件式〉
〈検索条件式〉⇒〈単純式〉|〈検索条件式〉〈単純式〉
〈単純式〉    ⇒〈属性〉〈演算子〉〈値〉
〈演算子〉    ⇒':' | '/\'
```

化を反映することが可能である．

```
ln -s
…/owner^inamura, moriai % contents : memo
~/memos
```

### 4.2 ALeFs の実現について

ALeFs では前項までで説明した機能を UNIX システム上でファイルサーバの形で実現している．**図 2** の通り実現には NFS プロトコル[11]を用いている．クライアントからの NFS 要求は，まずその中身を ALeFs で判断する．属性操作や問い合わせに関するものは ALeFs で処理されるが通常の階層名による操作は適切なシステムコールや NFS プロトコルにマッピングしている．この対応関係は**表 4** に示す．表中で空欄は特別な処理をしていないことを示す．

なお，UNIX システム自体に変更を加えずに機能の追加を行うために，ALeFs は利用者プロセスで実現されている．

現状では属性操作のために文字 “@” を用いており，これは通常のファイルの名前の中に含むことができないという制限がある．

ALeFs は現在筆者らの環境で Sun SparcStation1＋上で稼働している．

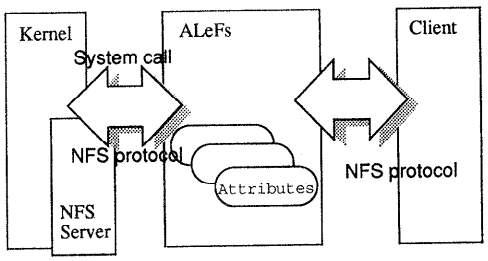



**図 2** ALeFs の構成図
Fig. 2 Diagram of the ALeFs system.

**表 4** NFS プロトコルの ALeFs での対応
Table 4 Interpretation of the NFS protocol in ALeFs.

| NFS プロトコル手続き名 | ALeFs での処理 |
|---|---|
| NFSPROC_NULL | |
| NFSPROC_GETATTR | 仮想ディレクトリの情報を生成 |
| NFSPROC_SETATTR | 属性を更新して反映 |
| NFSPROC_ROOT | |
| NFSPROC_LOOKUP | 仮想ディレクトリの情報を生成 |
| NFSPROC_READLINK | |
| NFSPROC_READ | |
| NFSPROC_WRITECACHE | |
| NFSPROC_WRITE | |
| NFSPROC_CREATE | 属性を更新して反映 |
| NFSPROC_REMOVE | 属性を更新して反映 |
| NFSPROC_RENAME | 属性の操作 |
| NFSPROC_LINK | |
| NFSPROC_SYMLINK | |
| NFSPROC_MKDIR | |
| NFSPROC_RMDIR | |
| NFSPROC_READDIR | 仮想ディレクトリの情報を生成 |
| NFSPROC_STATFS | |

## 5. 性能評価

以下に性能を測定した評価を示す．まず，NFS サーバとしての性能におけるオーバヘッドを測定した．次に問い合わせ機能の性能を測定した結果を示した．サーバ，クライアントともに SparcStation1＋ で SunOS 4.1.1 を用いて実験を行った．それぞれメモリを 40 MB 実装している．

### 5.1 オーバヘッドの測定

実現の項で述べたとおり，ALeFs では通常の NFS サーバの機能に対し，属性操作や問い合わせ機能を付加する形で実現を行っている．したがって，これらの拡張を行った部分にはオーバヘッドが生じる．NFS リクエストの応答時間を調べたグラフが**図 3** である．

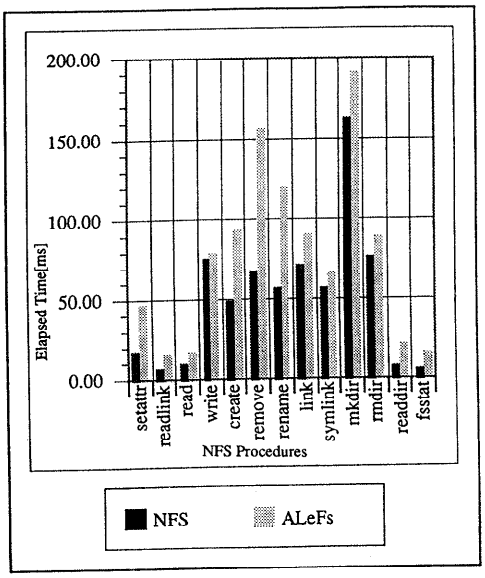



**図 3** NFS と ALeFs における各 NFS リクエストのレスポンスタイム
Fig. 3 Response time for NFS requests in native NFS and ALeFs.

**オーバヘッドの測定方法**

測定は NHFSSTONE[10] を元にしたプログラムで行った．NFS プロトコルに該当するシステムコールを複数回実行し，経過時間をそれぞれの NFS リクエスト回数で割ることで各コールの実行時間を推定したものである．これを SunOS 4.1.1 で提供される NFS サーバと ALeFs の両方に対して実行した．

この測定方法の制約から，NFS プロトコルの get_attribute, lookup に関しては独立に値が求められなかったため測定から外した．また，null, root, wchache は実際には使われていなため測定していない．グラフ中で create, remove, rename などの手続きの大きなオーバヘッドは表 4 のとおり属性操作のための余分な処理の存在を反映している．

### 5.2 問い合わせ機能の性能

ALeFs では属性に基づく問い合わせ機能を提供している．この機能の性能を測定した．属性を持つファイルを複数用意し，それらに対して条件を指定することで問い合わせを行い，その応答時間を測定する．その結果をグラフが**図 4** である．

**問い合わせ機能の性能測定方法**

ALeFs 上で生成したファイルに属性を付加する．グラフの横軸がファイル数である．これらには 3 つの属性が付加される．そしてその中の 50 個には特定の属性 (kind :art) が付加される．これらに対して ls コマンドを用いて以下の条件

ls…/kind :art

で検索を行い，その応答時間をプロットしたものである．これらは ALeFs やクライアントカーネルでのキャッシュの影響を除くためにマウント直後の状態で計測した．

なお，条件に合致する個数を 50 個に固定したのは，ls コマンドの入出力の条件を同一にするためである．

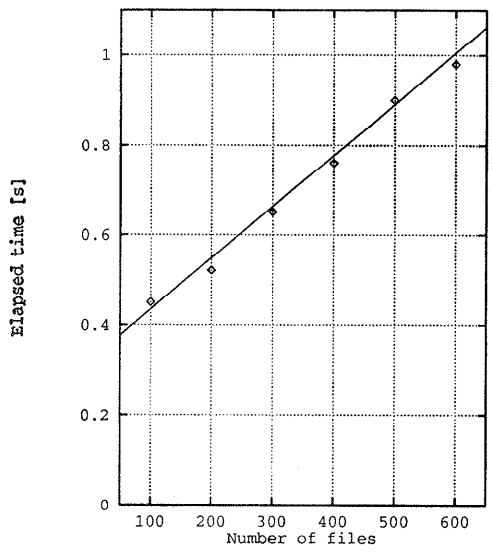



図 4 ALeFs における問い合わせのレスポンスタイムへのシステム内のファイル数の影響
Fig. 4 Influence of the number of files in the system on response time for a query.

### 5.3 ま と め

まず NFS サーバとしての性能におけるオーバヘッドを示した．属性操作を付加さた部分にはオーバヘッドが存在する．しかし通常の入出力操作には影響を与えない．

次に問い合わせ機能の性能を測定した結果を示した．ls コマンドを用いた単純な問い合わせを 600 個のファイルに対して行う場合 1 秒程度の応答時間である．したがって実用に耐えられると考える．

## 6. 関 連 研 究

現在，同様の問題意識を持ついくつかの研究が行われている．ここではそのうちの 2 つを取り上げ，ALeFs のアプローチと比較する．

### 6.1 Semantic File System

Gifford らによる Semantic File System[3] では，ALeFs と同様な属性による問い合わせで連想的なアクセスを実現している．Virtual directory と呼ばれる特殊なディレクトリを transducer というプログラムモジュールでファイルから属性を抽出することで維持し，その semantic の組み合わせによって問い合わせを実現している．この transducer というモジュールは利用者によってプログラミングされる．また，その問い合わせ構成は各条件を独立したディレクトリとして扱っている．

**ALeFs との比較**

Semantic File System では利用者によるアドホックな属性付加を実現していない．transducer からの属性付加ですべてを行うより，標準的なインタフェースを提供し，利用者とアプリケーションプログラムの両方に同じ機能を利用可能にすべきである．また，自動的なラベル付与メカニズムはサーバの外部プロセスとすべきである．ALeFs ではそのような実現を行った．

### 6.2 Multi-structured Naming Scheme

Secrest らによる Multi-structured Naming Scheme[9] では名前空間の構築／操作に必要な4つのルールを用いて，従来の階層的あるいは属性ベースの名前付けを統合している．

- Scoping rules
- Implicit value rules
- Attribute value constraint rules
- Aliasing rules

ここではパス名を構成するディレクトリが集合として扱われ，その順序に解釈結果は依存しないという提案，ならびにその評価法を柔軟に変化させる提案がなされている．現状では4つのルールのうち，implicit rule のみ実現されている．

**ALeFs との比較**

Multi-structured naming scheme では属性は論理値として扱われている．それによって属性の集合による名前付けが可能になっている．それは ALeFs での属性の定義の特別な場合に該当し，同様な機能を実現するのはたやすい．また，scoping ルールに代表されるように，名前の解釈の方法を暗黙に変化させる提案があるが，解釈ルールが利用に分からないうちにシステム側で変化させるのは不自然であると考える．したがって ALeFs では同様なアプローチは採らなかった．

## 7. おわりに

本論文では階層型と属性に基づく名前解釈システムの統合の必要性について検討し，ALeFs という分散ファイルシステムを提案した．従来の階層に基づく名前解釈システムに属性に基づく解釈機構を加えることで，問い合わせによるオブジェクトの特定を可能にし，連想的なアクセスを可能にした．

ALeFs ではファイルの属性管理を別個のインタフェースを新たに設けるのではなく，ファイルシステム自体で行うことで，変化するファイルの状況を即座に反映できる，通常のユーティリティがそのまま利用できるといったメリットが得られた．また，ファイルシステムとしての性能面においても十分実用に耐えうるものと考える．

例にはサーチパスの検索などを挙げたが，他にも指定した属性を持つファイルを拾い出してアーカイブしたり，適当なインタフェースを用意することでアーカイブやメイル，ニュース記事を整理するといった利用が可能である．

Multi-Institutional file system などの大規模なシステムにおいては，パス名は長くなる傾向がある．こういった状況には ALeFs のアプローチが有効であると考える．

今後の課題としては，以下のものがある．

- バーション管理のサブシステム
  複数のバージョンを持つファイルを管理するために現状では独立のシステムでその機能を提供している．ALeFs では問い合わせの結果を動的にディレクトリとして返すことで一種のビューを作っていると言うことができる．このビュー機能を拡張することでバージョン管理システムを実現することが可能である[5]．
- 属性の付与
  Semantic File System では属性の付与に関して独自の transducer によるアプローチを行っている．ALeFs では transducer のような機能をすべてファイルシステム側に取り込むのではなく，ファイルの属性の変化や新たなエントリの追加といった変化をトリガとして外部のプロセスに通知するメカニズムがあればよいと考えている．そうのような機構を含むことで例えば Macintosh の drag & drop のような操作性を実現できる．
- リンク機能の拡張
  マルチメディアデータは大容量になりがちであり，その管理には現在さまざまなアプローチが行われている．マルチメディアデータのブラウジングを考えるとき，データの特定区間にラベル付けし，その位置情報自体に名前を付けて保存できると都合が良い．ALeFs にリンク機能を拡張することでデータの特定区間へのアクセスを提供できる．

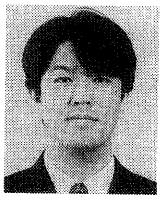
また，さらに多くの種類のオブジェクトを扱えるように拡張していつもりである．

## 参考文献

1) Birrell, A. D., Needham, R. M. and Schroeder, M. D.: Grapevine: An Exercise in Distributed Computing, *Comm. ACM*, Vol. 25, No. 4, 260-273 (1982).
2) Comer, D. E. and Peterson, L.L.: A Model of Name Resolution in Distributed Systems, *6th International Conference on Distributed Computer Systems*, pp. 523-530 (1986).
3) Gifford, D. K., Jouvelet, P., Sheldon, M. A.

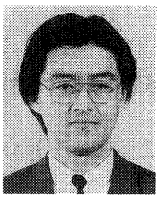
and Jr., J. W. O.: Semantic File System, *13th ACM Symposium on Operating Systems Principles*, pp. 16-25 (Oct. 1991).
4) Inamura, H. and Moriai, S.: Integration of Hierarchical and Attribute-based Naming Schemes in a Distributed File System, *7th Joint Workshop on Computer Communication*, pp. 227-236 (1992).
5) Korn, D. G. and Krell, E.: A New Dimension for the UNIX File System, *Software*, Vol. 20, S 1/20-S 1/34 (1990).
6) Lampen, A.: Advancing Files to Attributes Software Object, *USENIX Winter Proceeding*, pp. 219-229 (1991).
7) Oppen, D. C. and Dalal, Y. K.: The Clearing House: A Decentralized Agent for Locating Named Objects in a Distributed Environment, *ACM Trans. Office Information Systems*, Vol. p. 3, pp. 230-253 (1983).
8) Peterson, L. L.: The Profile Naming Service, *ACM Trans. Comput. Syst.*, Vol. 6, No. 4, pp. 341-364 (1988).
9) Sechrest, S. and McClennen, M.: Blending Hierarchical and Attribute-Based File Naming, *12th International Conference on Distributed Computer Systems*, pp. 572-580 (May 1992).
10) Stern, H.: *Managing NFS and NIS*, O'Reilly & Associates, Inc., ISBN-0-937175-75-7 (1991).
11) Sun Microsystems Inc.: *NFS: Network File System Protocol Specification*, RFC 1094 (Mar. 1989).
12) Thompson, K. and Ritchie, D.: The UNIX Time-sharing System, *Comm. ACM*, Vol. 17, No. 7, pp. 365-375 (1974).
13) 古宇田フミ子，田中英彦：記述名表現方式と意味ネットワークを利用した記述名解決法，情報処理学会論文誌，Vol. 31, No. 9, pp. 1397-1404 (1990).

(平成4年11月4日受付)
(平成5年 4 月8日採録)



**稲村 浩** (正会員)

1988 年慶応義塾大学理工学部計測工学科卒業．1990 年慶応義塾大学大学院理工学研究科計測工学専攻修了．同年日本電信電話(株)入社．分散 OS，主に分散ファイルシステムの研究に従事．現在，NTT 情報通信網研究所に勤務．日本ソフトウェア科学会会員．



**盛合 敏**

1983 年東北大学工学部電気工学科卒業．1988 年同大学大学院博士課程修了．工学博士．同年日本電信電話(株)入社．音声自動認識，言語処理，分散オペレーティングシステムの研究に従事．現在，NTT 情報通信網研究所主任研究員．日本ソフトウェア科学会，電子情報通信学会，日本音響学会各会員．