社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＱＡＩ－10　(10)
2004／1／20

分散環境における計算機群の名前環境統一機構の実現

亀井　仁志†　　中村　　豊††　　藤川　和利††　　砂原　秀樹††

† 奈良先端科学技術大学院大学 情報科学研究科　〒630-0192 奈良県生駒市高山町 8916-5
†† 奈良先端科学技術大学院大学 情報科学センター　〒630-0192 奈良県生駒市高山町 8916-5
E-mail: †hitosh-k@is.aist-nara.ac.jp, ††{yutaka-n,fujikawa,suna}@itc.aist-nara.ac.jp

あらまし　計算機の低価格化，高性能化により，個人が利用する計算機の台数が増え，複数台の計算機を利用する環境が一般的になっている．分散環境では，ファイルシステムの名前空間は個々の計算機に強く依存しており，システムに依存するファイルを利用する場合，あらかじめ個々の計算機のファイルシステムの名前空間を把握しておく必要がある．現在はNFSを利用して，ホームディレクトリ等を共通化することで統一的なアクセスができる．しかし，システムディレクトリに対してはCPUアーキテクチャやライブラリなどが同一でない場合には，共通化することが困難である．また，システム管理権限のないユーザである場合，システム依存ファイルの管理ポリシーの違いなどによる利用制限が問題となる．本研究では，ファイルシステムの名前空間をユーザ毎に管理するために，Virtual File System (VFS) のディレクトリエントリ管理に対して仮想名前空間を定義する．仮想名前空間を元にユーザ毎にファイルシステムに依存しない名前空間管理可能にするVFSの名前空間管理機構を提案する．

キーワード　分散ファイルシステム，ネットワークファイルシステム，分散コンピューティング

Design of Unified Namespace File System for Distributed Computer Networks.

Hitoshi KAMEI†, Yutaka NAKAMURA††, Kazutoshi FUJIKAWA††, and Hideki SUNAHARA††

† Graduate School of Information Science of NARA Institute of Science and Technology　Takayama-cho 8916-5, Ikoma-shi, Nara, 630-0192 Japan
†† Information Technology Center of NARA Institute of Science and Technology　Takayama-cho 8916-5, Ikoma-shi, Nara, 630-0192 Japan
E-mail: †hitosh-k@is.aist-nara.ac.jp, ††{yutaka-n,fujikawa,suna}@itc.aist-nara.ac.jp

Abstract　With the progress of computer technology, a user can use several computers at the same time via computer network. Currently, Network File System (NFS) provide a file sharing mechanism among several computers, that can use the same home directory on the different computers. However, if a user work on the different computers, he/she has to know their own name spaces due to the difference of their CPU architectures and the system directory structures. In this paper, we propose a new name space management mechanism that can provide a unique name space for each user. In the proposed mechanism, a name space layer is provided on the top of Virtual File System VFS) layer. The name space provided by our proposed mechanism is independent of the CPU architectures and OSs.

Key words　Distributed File System, Network File System, Distributed computing

1. はじめに

計算機の低価格化，高速化が進み個人で利用したり保有する計算機の台数が増加している．また，ネットワークの普及により，それらの計算機をネットワークで接続して，相互に利用することが一般的になってきた．このような計算機の環境変化によりNetwork File System (NFS) [1]や，Windowsファイル共有などの分散ファイルシステムや，Virtual Network Computing (VNC) [2]など，遠隔ログインなどの多数の計算機環境における分散環境利用が注目されるようになっている．

今までに，分散環境においてリソースを共有するために，様々な技術が提案され実装されている．その技術には「ファイル共

有」と「デバイス共有」，または「デスクトップリモートアクセス」がある．それぞれの技術に代表されるシステムは，

- ファイル共有…NFS，Windows ファイル共有
- デバイス共有…NBD，iSCSI
- デスクトップリモートアクセス…VNC

が挙げられる．

しかし，これらの技術は様々な制限がある．現在，計算機の管理上での問題とは，計算機を利用する時に，ファイルシステムの名前空間を先に調べ，名前空間構成を知らなければならないことである．名前空間構成を知らなければ，利用者は計算機に導入されているソフトウェアの利用や，設定ファイルなどを操作することができない．このような問題を解消するために，現在ではそれぞれの計算機で同じファイルを用意し，同様の環境をすべての計算機で実現している．また，利用環境統一のために，名前空間を統一するためにファイルを外部へ公開し，それぞれの計算機で共有することが行われている．しかし，この方法では，システムに依存する実行ファイルなどを統一化することや，ユーザ毎に統一的な名前空間を構築すること，管理権限に関する問題を解消することはできない．

現在，ディスクの名前空間は OS のファイルシステムと呼ばれるサブシステムに依存している．ファイルシステムは様々な用途で効率が良いように複数の実装がなされている．そのため，ファイルシステムの提供する API の違いによりプログラムが制限されないように，現在の OS では Virtual File System (VFS) と呼ばれる，ファイルシステムの抽象化を行うレイヤが存在する．VFS レイヤではシステムコールとファイルシステム専用 API とを結びつける働きをしている．

今までに述べた問題点は，ファイルシステムの名前空間がそれぞれの計算機に依存し，利用者毎に管理されていないことが問題であると考えられる．また，名前空間がファイルシステムに管理され，名前空間がファイルシステムに依存していることも問題である．本論文ではこのような問題を解決するために VFS レイヤに，ファイルシステムと名前空間を独立して管理する機構を実装し，ユーザ毎に名前空間を管理し計算機間で統一するための機構を提案する．

2. 既存の分散システム

2.1 ネットワークファイルシステム

ネットワークファイルシステムは，分散ファイルシステムを実現するための基礎となるシステムである．図 1 のように，離れた場所にある計算機のファイルシステム空間のディレクトリをある一定のポリシーで外部の計算機へ公開する．外部の計算機はローカルファイルシステムの一部としてマウントし，その計算機を使うユーザに対して利用できるようにする．有名なネットワークファイルシステムとしては Sun Microsystems 社が開発し，現在 UNIX 上で最も用いられている Network File System や Microsoft 社が自社の OS で利用する Windows ファイル共有，また，Campus ネットワークなどの中規模クラスのネットワークで用いられる The Andrew File System (AFS) [7] などがある．

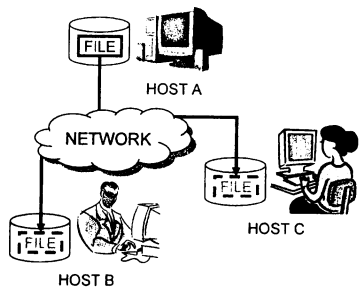

図 1 ネットワークファイルシステム

2.1.1 Netowrk File System (NFS)

NFS は Sun Microsystems 社が開発した分散ファイルシステムで，RFC1094 [3] で規定されている．クライアント／サーバ方式で実装された代表的な分散ファイルシステムである．クライアントおよびサーバは様々なシステムに移植されており，多数の OS でファイル共有が可能である．

2.1.2 Windows ファイル共有

Windows ファイル共有は，ネットワークレイヤのプロトコルに依存しないファイルシステムとして Microsoft 社が実装した．Windows のネットワーク API の一つである NetBIOS を用いて実装されており，システム自体は NetBEUI を前提に実装されているが，NetBIOS over TCP/IP (NBT) 技術により，ネットワークレイヤのプロトコルが TCP/IP であっても動作する．NBT は RFC1001 [4]，RFC1002 [5] で規格化されている．

Windows ファイル共有のプロトコルは NetBIOS 上に実装されるものとして SMB があり，それらを元にした UNIX の実装として Samba [6] がある．また，TCP/IP に直接実装し，SMB を拡張して作られたプロトコルとして CIFS がある．

2.1.3 Andrew File System (AFS)

AFS は米カーネーギーメロン大学がキャンパスネットワーク上に，クラスタ化された計算機をまとめて一つの大きな名前空間を実現するために，設計および実装を行ったシステムである (図 2)．キャンパスネットワーク上に多数存在する計算機のディスクを一つのファイルシステムとしてマウントできる．また，このシステムの問題点であった Disconnect Operation に対応したファイルシステムとして CodaFS [8] が設計され実装されている．

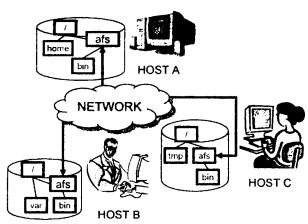

図 2 The Andrew File System

2.2 デバイスのネットワークにおける利用

先に述べたようなファイルシステムをネットワークで利用するのとは違い，ファイルシステムの下位のレイヤである，ブロックデバイスをネットワークへ公開して共有するという技術が存在する．この技術により，ディスクなどのデバイス自体を共有することができ，結果的にファイルシステムの名前空間を統一することが可能となる．

2.3 リモートデスクトップアクセス

リモートデスクトップアクセスは，これまでに述べたようなファイルシステムや，デバイスではなく画面そのものである，フレームバッファを直接クライアントへ送信し，フレームバッファをクライアントで表示することで，遠隔の計算機資源を利用する．こうすることでサーバのファイルシステムを常に使用することができ，名前空間を気にする必要がなくなる．代表的なソフトウェアとして，Virtual Network Computing (VNC) が挙げられる．

VNC を利用しサーバのファイルシステムを常に利用することで，名前空間を一つにすることが可能となる．しかし，クライアント上のファイルシステムを考慮していないため，クライアントとサーバ間の名前空間の違いに関しては解決されていない．

2.4 既存システムの問題点

この節で今までに述べたように OS における様々なレイヤで分散システムの研究がなされ，運用されている．それらのシステムの利点はと欠点は表 1 のようになる．

我々が考えている問題点とは，図 3 のように名前空間が，ローカルファイルシステムに強く依存しているため，それぞれの計算機における名前空間について，システムを利用する利用者が知っていなければならないことである．そのため，利用者はまず始めに利用する計算機の名前空間を知らなければならない．そして，どのように名前空間が構成されているかを調べる必要がある．また，システムに導入されていない実行ファイルは利用者が個別に導入する．このことで，ファイルシステムにあるファイルは冗長になる．また，計算機の名前空間の大部分が計算機依存であり，計算機を変わるたびにそのことを意識しなければならない．そして，名前空間がファイルシステム依存であるため，ユーザ毎に特別な名前空間を作ることができない．

3. Virtual File System (VFS)

3.1 ローカルファイルシステム

計算機ではデータを保存するために二次記憶装置であるディスクを用いることが多い．Linux などの OS では様々な計算機

表 1 既存システムの利点と欠点

	位置透過性	共有	スケーラビリティ	名前空間統一
NFS, SMB	○	○	×	×
AFS	○	○	○	×
NBD, iSCSI	○	△	△	×
リモートデスクトップ	○	○	×	×

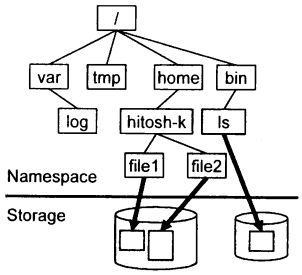

図 3 名前空間とファイルシステムの関係

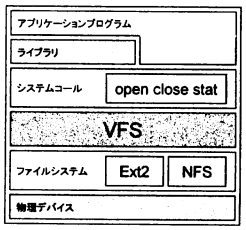

図 4 VFS レイヤ

の資源を管理しており，例えば，キーボードやマウス，ネットワークデバイスそしてディスクなどが挙げられる．ローカルファイルシステムは OS のサブシステムで，ディスクを管理し，データをディスクにファイルという単位で保存している．ファイルシステムでは，ファイルの入出力を行ったり，デバイスを管理するするプログラムで構成されている．

名前空間はファイルシステムが管理しており，実際のデータと名前空間を対応づけている．一般にディスクはハードディスクなど物理デバイスを指す．しかし，ファイルシステムでは NFS のように，ネットワークを経由して遠隔にある名前空間を扱うものもある．

3.2 VFS とローカルファイルシステム

LINUX など UNIX 系の OS では図 4 のように，VFS と呼ばれるファイルシステムを抽象化するレイヤが存在する．また，Winodws ではリダイレクタとも言う．VFS ではローカルファイルシステムと呼ばれる，実際に入出力を行うためのプログラムのインタフェースを共通化する．このため，ユーザプログラムはファイルシステムに依存したプログラミングをすることなく，ディスクに対して入出力できる．また，NFS などのネットワークファイルシステムを抽象化することによって，ネットワーク特有の問題点を隠蔽できる．

3.3 VFS と名前空間

VFS はローカルファイルシステムに対する共通のインタフェースを提供するだけでなく，利用者に対してファイルシステムの名前空間管理のインタフェースも提供する．例えば名前空間を読み出すために利用されるインタフェースとして `readdir(getdents)` システムコールがあり，また，`open` を作成モードで利用することにより，新たなファイルを名前空間に登録できる．UNIX における名前空間はディレクトリツリーと

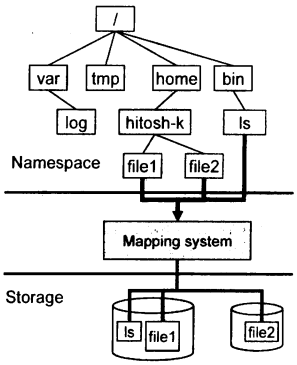

図 5　UNVFS の概念

いった root を起点としたツリー状に構成される．ローカルファイルシステムはこのツリーの一部として「つぎ木」され名前空間が構築される．このようにしてファイルシステムの名前空間が構築される．

名前空間のインタフェースは VFS が提供する．しかし，名前空間を実際に管理しているのはファイルシステムである．各ファイルシステムが管理しているデバイスに対して実際の名前空間が作成され，ファイルシステムによって管理されている．

3.4　名前空間の作成と管理

ファイルシステムはディスク上のデータをファイルとして管理している．ファイルはディレクトリと呼ばれる特殊ファイルを使って，ツリー状にアクセスできるように構成されている．ディレクトリは「/」で区切られ，再帰的にアクセスされる．例えば，ルートディレクトリの下にある `tmp` ディレクトリの `a.out` ファイルは「`/tmp/a.out`」と表される．また，名前空間は UNIX のファイルシステムセマンティクスに従い「/」(root directory) から始まっている．このようなディレクトリツリー全体を名前空間と呼ぶ．

これらのファイル名とディレクトリツリーはファイルシステムにファイル操作関数を実行させた時に作成されるもので，名前空間の管理とファイルへのアクセスはファイルシステムにすべて依存している．

4.　名前空間統一可能なファイルシステム

4.1　UNVFS の概要

本研究では 2.4 節で述べた問題点に対して VFS で解決するシステムである UNVFS(Unified Namespace VFS) を提案する．これまで名前空間は，ファイルシステムの名前空間に強く依存してきた．これを仮想的な名前空間とファイルシステムの名前空間とをマッピングすることで，図 5 のようにファイルシステムから独立させることができる．このことで，名前空間構築に自由度が生まれユーザごとに自由な名前空間が構築できる．

UNVFS では，ローカルファイルシステムと名前空間をマッピングし管理する VFS のメソッド群に変更を加え，利用者に対して統一された名前空間を提供する．メソッドを操作する部分にユーザ毎に管理するマッピング情報を挿入し，それらを元に名前空間の統一化を実現する．

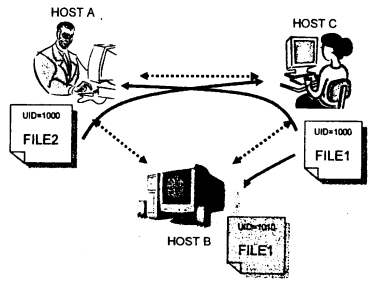

図 6　P2P 型のシステム運用

2.4 節で述べたように，計算機の名前空間がファイルシステムによって管理され，ファイルシステムと名前空間が密接に関係していることが問題となっている．また，ファイルシステムで名前空間を管理しているため，システムでただ一つの名前空間が構築され，利用者が自由に名前空間を構築することができない．このことから，「ファイルシステムに非依存な名前空間をユーザ毎に作成する」ことで，名前空間に起因する問題を解決できると考えられる．

3. 節で述べたように，VFS では名前空間を管理していない．そして，VFS はファイルシステムと直接通信するような仕組みになっている．そのため，VFS レイヤでユーザ毎の名前空間管理を行えば，アプリケーションやローカルファイルシステムに影響しない．

4.2　システムとネットワーク構成

提案システムは計算機間でのファイルシステム名前空間を統一させるためのシステムのため，ユーザが利用するすべての計算機に導入され相互に運用される．また，名前空間を統一させ，ファイルにアクセスするためにそれぞれの計算機が所有しているファイルを相互に利用できるようにしなければならない．

例えば図 6 では，各計算機でユーザ ID が 1000 であるユーザが，統一された名前空間を利用しファイルアクセスした場合 HOST A が HOST C の FILE1 を利用し，HOST C は HOST A の FILE2 を利用し，また，HOST B は HOST C の FILE1 を利用すると言った，相互にファイルを利用する形態を示いている．

提案システムは P2P 型のネットワークを形成し，それぞれのノードはユーザ毎のマッピング情報が記録されているファイルを交換し共有することで，どのノードを利用しても同じ名前空間が実現する．

4.3　変更するメソッド

2.4 節で述べたように，名前空間をユーザ毎に統一させるために VFS の名前管理に関するメソッドの変更する．現在変更しているメソッドは以下のものが挙げられる．

- open … ファイルの操作を開始するためのメソッド
- close …ファイルの操作を終了するためのメソッド
- read … ファイルを読み込むメソッド
- write … ファイルに書き込むメソッド

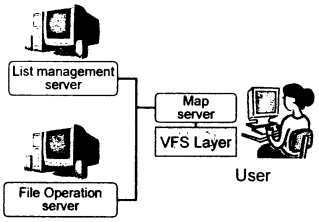

図 7　システム構成

- readdir … ls などでエントリを検索するメソッド

これらの基本的なメソッドを変更し，ユーザ毎で名前空間を定義できるように VFS を変更する．

5. UNVFS の設計

UNVFS では複数のユーザレベルデーモンと VFS の変更によって，ユーザ毎に仮想的な名前空間を用意することで，名前空間の統一を行う．図 7 に UNVFS のシステム構成を示す．

5.1 仮想名前空間

仮想名前空間とは，本システムにおいてユーザ毎に作成される名前空間のことを指す．仮想名前空間はファイルシステムに非依存の名前空間で，ファイルシステムが管理している名前空間は実名前空間と呼ぶ．本システムは仮想名前空間と実名前空間をすべてのホストで一致させ，マッピングすることで，ユーザへ統一的な名前空間を提供する．

5.2 User level daemon

VFS が提供する名前空間を管理し，ネットワークを通して離れたファイルへのアクセスを行うために，補助的な役割をするユーザレベルデーモンを設計する．ユーザレベルデーモンは「名前空間マッピング」「エントリ管理」「実ファイル管理」の 3 つの機能を別々のデーモンとして設計する．それぞれの機能のサーバは以下のように呼ぶ．

（1） Map server … 名前空間マッピング
（2） List management server … 仮想名前空間管理
（3） File operation server … 実ファイル操作

5.2.1 Map server

Map server は仮想名前空間とファイルシステム上の名前空間をマッピングする．Map server は起動時に List management server から仮想名前空間の表を取得し，その表を元に仮想名前空間を構築する．また，Map server は VFS に対して UNVFS のコードを起動させるため/proc 以下のエントリへデータを書き込む．

5.2.2 List management server

List management server は仮想名前空間の表を管理する．仮想名前空間の表が新たに更新された時のために，仮想名前空間表のバージョンナンバーを持っており，Map server が起動時に表を取得する時に使用する．仮想名前空間表は inode を模した構造となっており，表 2 のようになっている．そして，表は仮想ディレクトリエントリと対応したファイルになっている．パス検索を行う場合は inode を検索するような形をとる．ファイルの内容は表のようになっている．検索する場合は，例えば「/tmp」の場合，「/」のエントリファイルから「tmp」を検索する．名前空間表からローカルファイルシステム上のファイルへアクセスする．

表 2　仮想名前空間表（一部）

vi_mode	アクセスモード
vi_flag	アクセスフラグ
vi_ver	ファイルバージョン
vi_uid	UID
vi_url	実ファイルの URL
...	...

5.2.3 File operation server

File operation server は仮想名前空間からローカルファイルシステムの実ファイルに Map server からアクセス要求が起きた時に使用される．Map server は実ファイルを持っている対象ホストの File operation server に対してファイル操作要求を出し，Map server は仮想ファイルを操作しているように，VFS へ通知する．VFS は Map server から取得した情報から，仮想ファイルを利用しているプロセスへ結果を返す．

5.3 VFS の変更

VFS はプロセスからの要求をシステムコールの形で受ける．プロセスは OS に対してディスクなどを利用する場合，システムコールを発行してサービスを利用する．VFS はシステムコールに対応したコールバックメソッドが用意されており，システムコールに対するカーネル動作の契機となる．

UNVFS では VFS のコールバックメソッドに改良を加え，統一的な名前空間を実現する．このため，VFS の名前空間を操作するコールバックメソッドを改良しなければならないので，仮想名前空間を引数で渡してくるコールバックメソッドを対象としなければならない．名前空間を操作するメソッドは以下のようなものが考えられる．

- open
- readdir

これらのメソッドに対して変更を加えて，仮想名前空間を見せるようにし，また仮想名前空間上のファイルにアクセスできるようにする．

5.4 仮想名前空間管理

仮想名前空間の管理は List management server が行う．仮想名前空間を作成するために，現在では mkentry コマンドを使用する．mkentry コマンドは UID とファイルのフルパスなどを入力して，仮想名前空間のエントリを作成する．

5.4.1 管理データ構造

管理データは，仮想名前空間上ファイルとローカルファイルシステムの名前空間をマッピングするためのデータと，ディレクトリエントリを管理するためのデータの 2 つが必要である．仮想名前空間のファイルをマッピングする情報は先に挙げた表 2 となる．また，仮想名前空間を管理するためのデータは，仮想ディレクトリエントリとして表 3 ようなデータ構造を作成した．これらの情報を元に仮想名前空間とファイルシステム名前

空間をマッピングする．

5.5 仮想名前空間へのアクセス

仮想名前空間へアクセスする関数は，5.3 節で述べたように readdir と open である．この関数は仮想名前空間上のエントリを取得し，ファイル操作を開始するための関数で，仮想名前空間へアクセスするために最低限必要な関数である．

5.5.1 open

open は仮想名前空間上のファイルをオープンする．関数は仮想名前空間のファイル名を引数として受け取り，通常のファイルディスクリプタを生成して返す．open 手順は以下のようになる．

（1） プロセスからファイル名を取得する

（2） 表を参照して実ファイルの URL を検索する

（3） 対象ホストに接続し，ファイルをオープンする

（4） オープン情報をホストから受けとり，ディスクリプタを生成して返す

5.5.2 readdir

readdir はディレクトリをオープンしたファイルディスクリプタを受け取り，ディレクトリエントリを検索するため，ファイルディスクリプタから得られる file 構造体を元に，現在要求されている仮想ファイル名を取得しなければならない．readdir の手順は以下のようになる．

（1） ディスクリプタから file 構造体を取得する

（2） file 構造体からオープンされている仮想ディレクトリ名を調べる

（3） 仮想ディレクトリ名から表を参照してエントリ情報を得る

（4） エントリ情報をプロセスに返す

5.6 仮想ファイル操作

仮想名前空間上のファイルを操作するために，Map server は File operation server へ要求を行う．File operation server は実ファイル名を受け取り，実ファイルをオープンして，Map server へ完了を返す．open の後は，UNVFS が受け取るのはファイルディスクリプタであり，そのディスクリプタを元に仮想ファイルへアクセスする．仮想ファイルへのアクセスは read と write を用いる．そのため，それらのコールバックメソッドに対しても変更をしなければならない．

5.6.1 read

read は仮想名前空間上のファイルを読み出すために使用される．ファイルディスクリプタを受け取り，ファイルディスクリプタから URL を検索し，アクセス要求をしなければならない．read の手順は以下のようになる．

（1） ファイルディスクリプタを受け取る

（2） ファイルディスクリプタからファイルの情報を検索する

表 3 仮想ファイル管理表

vinode	管理表ファイルナンバー
namelen	名前サイズ
name	仮想ファイル名

表 4 /tmp の管理表

vinode	5
namelen	4
name	test

（3） ファイル情報から URL を検索し，File operation server へ接続する

（4） ファイル読み込みを要求し，データを受け取る

（5） プロセスへデータを返す

5.6.2 write

write は read とは逆に仮想名前空間上のファイルへ書き込みを行なう．write も read と同様にファイルディスクリプタを受け取るため，ファイルディスクリプタからファイル情報を検索して，実ファイルへアクセスしなければならない．write の手順は以下のようになる．

（1） ファイルディスクリプタを受け取る

（2） ファイルディスクリプタからファイル情報を検索する

（3） ファイル情報から URL を検索して，File operation server へ接続する

（4） ファイル書き込みを要求し，データを送信して書き込む

（5） プロセスへ書き込み完了通知を返す

5.7 動 作 例

UNVFS を利用した場合，例えば表のような仮想名前空間が構築されていた場合は，「/tmp」の下にローカルファイルシステム上にはないファイル名である「test」が readdir によって取得され，open によって仮想ファイルがオープンできる．そして，read および write でファイルを読み書きする．

6. ま と め

本稿では，UNVFS のシステム構成，名前空間を統一させるために必要な情報，変更が必要な関数群，および動手作順について述べた．ファイルシステムにおける名前空間の問題を示し，名前空間をファイルシステムから独立させるために VFS へ変更を行う UNVFS を提案した．現在，UNVFS は実装中であり，実装完了後は，システム全体の評価を行っていきたい．

文 献

[1] 笠野英松 監修，マルチメディア通信学会 編：
インターネット RFC 事典，ASCII，1998．

[2] VNC:
http://www.realvnc.com/

[3] RFC 1094:
http://www.ietf.org/rfc/rfc1094.txt?number=1094

[4] RFC 1001:
http://www.ietf.org/rfc/rfc1001.txt?number=1001

[5] RFC 1002:
http://www.ietf.org/rfc/rfc1002.txt?number=1002

[6] Samba:
http://www.samba.org/

[7] The Andrew File System (AFS):
http://www.psc.edu/general/filesys/afs/afs.html

[8] Coda File System:
http://www.coda.cs.cmu.edu/