オペレーティング・システム 43 － 7
（１９８９．６．９）

異種ＯＳを結ぶ分散ファイルシステム

谷口　秀夫　　遠城　秀和　　箱守　聰
ＮＴＴデータ通信（株）

異なったＯＳが走行する計算機間をＬＡＮで結合した分散処理環境として、ＵＮＩＸ-likeなＯＳが走行する計算機と分散型リアルタイムＯＳ（ＤＩＲＯＳ：DIstributed Real-time Operating System）が走行する計算機間に構築した分散ファイルシステムの機能・処理方式を述べる。ファイルアクセス系システムコールの内、ＵＮＩＸシステムコールの大半とＤＩＲＯＳシステムコールの基本機能について、リモートファイルアクセスを可能にした。その処理方式は、ＤＩＲＯＳが持つ非完了システムコール機能を利用して効率的に実現され、ローカル処理とリモート処理の応用プログラム統一を可能にしている。

The Distributed File System on the Heterogeneous Computing Environment

Hideo TANIGUCHI, Hidekazu ENJO, Satoshi HAKOMORI
NTT DATA COMMUNICATIONS SYSTEMS CORPORATION
NTT Data Yokohama Nishi Bldg., 2-11-6 Kita Saiwai,
Nishi-ku, Yokohama-shi, Kanagawa 220 Japan

This paper describes the distributed file system on the heterogeneous computing environment. There are two kinds of computer in the environment and computers are connected by LAN. One is the machine supported by the extended UNIX, and the other is the machine supported by DIROS(DIstributed Real-time Operating System). The characters of this distributed file system are as follows : (1) networking all of UNIX's system-calls and basic DIROS's system-calls for file-access, (2) supporting the application program interface for remote file-access as same as local.

１．はじめに

マイクロプロセッサを搭載した計算機をＬＡＮなどの高速通信路で結合した分散処理システムの構築が進んでいる。このようなシステムをサポートする分散処理ＯＳには、計算機間の資源共用を可能にする分散ファイルシステムが必要である。ＵＮＩＸ*-like なＯＳ間をサポートする分散ファイルシステムとしては、ＮＦＳ[1] やＲＦＳ[2] が代表的である。しかし、分散処理システム内を同種ＯＳで統一することは難しく、今後は異種ＯＳ間における分散ファイルシステムが必要になると思われる。

本稿では、分散ファイルシステムを実現するリモートファイルアクセス機能[3] を持つＵＮＩＸ-likeなＯＳを搭載したマシン（以降、ＷＳ：Work Stationと略す）とトランザクション処理用の分散型リアルタイムオペレーティングシステム[4]（ DIROS : DIstributed Real-time Operating System ）を搭載したマシン（以降、ＣＳ：Control Station と略す）をＬＡＮで結合し、分散ファイルシステムを構築する方式について、主にＣＳの機能・処理方式を報告する。

２．分散ファイルシステム

２．１　グローバルトリー

ＷＳがサポートしている分散ファイルシステム[3] は、以下の特徴を持つ。

（１）ＬＡＮで結ばれたすべてのＷＳ内資源が１つの木構造（以降、グローバルトリーと呼ぶ）で管理されている。

（２）１対１の単一リモートファイルアクセス機能と１対ｎのグループリモートファイルアクセス機能を持つ。

（３）すべての機能をＯＳ核内に実現している。

＊：UNIXはAT&Tのﾍﾞﾙ研究所が開発したOSです。

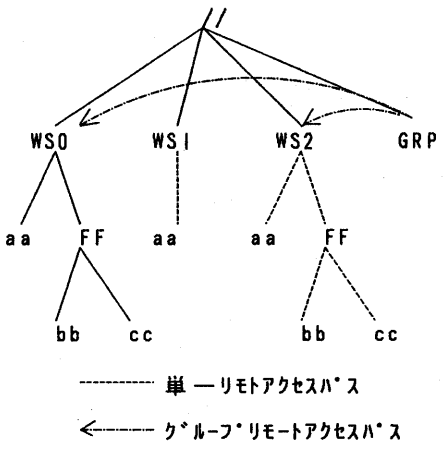


図１　グローバルトリーの例
WS0からのview

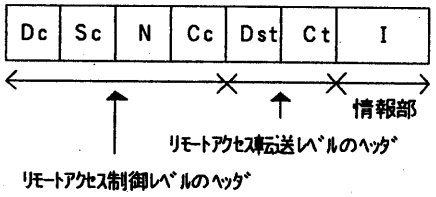
図１にグローバルトリーの例を示す。図１において、例えば、"／／ｗｓ１／ａａ"のパス名指定による単一リモートファイルアクセスの場合、ｗｓ１上のファイル"／ａａ"にアクセスできる。また、例えば、"／／ＧＲＰ／ａａ"のパス名指定によるグループリモートファイルアクセスの場合、ｗｓ０とｗｓ２上にあるファイル"／ａａ"にアクセスできる。

２．２　通信手順

リモートファイルアクセスの通信手順は、システムコール転送を基本に設計されており、リモートアクセス制御とリモートアクセス転送の２レベルから成る。パケット形式を図２に示すとともに、各レベルのコマンドを表１に示す。

３．ＤＩＲＯＳの概要

ＤＩＲＯＳは、マルチマイクロプロセッサをサポートするトランザクション処理用ＯＳである。異ＯＳ間の分散ファイルシステムを構築する際に関連するＤＩＲＯＳの特徴を以下に示す。



Dc, Sc, Dst :通信パス番号
N :シーケンス番号
Cc, Ct :コマンド

図2 パケット形式

（１）磁気ディスクのファイルは文字列で識別し、通常ファイルと呼ぶ。
（２）入出力装置や通信回線は通常ファイルと同じように文字列で識別し、特殊ファイルと呼ぶ。
（３）通常／特殊ファイルは、ディレクトリにより木構造で管理する。
（４）ファイル管理機能には、オープン・クローズ・データ入出力の基本アクセス機能以外に、アクセス権変更などの拡張制御機能がある。
（５）トランザクション処理用に、処理の要求と処理の結果取得を別システムコールとする非完了システムコール機能を持つ。
（６）ＵＮＩＸのようなユーザフレンドリなインタフェースは持たない。

４．要求条件

ＷＳとＣＳ間の分散ファイルシステムでは、ＷＳとＣＳ間の両方向についてリモートファイルアクセスできる必要がある。走行しているＯＳに機能差があるものの、分散ファイル

表１　リモートファイルアクセス通信手順のコマンド一覧

レベル	コマンド	内　　容
リモートアクセス制御	ＭＰ	リモートアクセス制御レベルの通信パスを設定する。
	ＤＰ	リモートアクセス制御レベルの通信パスを解放する。
	ＦＰ	リモートアクセス制御レベルの通信パスを分離する。
	ＩＦ	情報を転送する。
	ＳＮ	グループアクセスにおける入力ＷＳを設定する。
	ＲＴ	リモートアクセス機能の処理終了を通知する。
リモートアクセス転送	本レベルのコマンドは、ＵＮＩＸシステムコールに対応しており、リモートアクセス転送レベルの通信パスの設定・継続・解放・単独処理により、以下に分類されている。 （設定）　ｃｒｅａｔ，ｏｐｅｎ （継続）　ｒｅａｄ，ｗｒｉｔｅ，ｌｓｅｅｋ，ｉｏｃｔｌ，ｆｓｔａｔ （解放）　ｃｌｏｓｅ （単独）　ａｃｃｅｓｓ，ｃｈｍｏｄ，ｃｈｏｗｎ，ｌｉｎｋ，ｍｋｎｏｄ，ｍｏｕｎｔ，ｓｔａｔ，ｕｍｏｕｎｔ，ｕｎｌｉｎｋ，ｄｕｐ，ｆｃｎｔｌ	

システムに対して以下の要求条件がある。

［条件１］ＷＳからＣＳへのリモートファイルアクセス機能は、豊富な機能にしたい。

［条件２］ＣＳからＷＳへのリモートファイルアクセス機能は、ファイルのコピーによるファイル転送やＦＩＦＯファイルを利用したメッセージ転送、ができる程度でよい。

［条件３］ＣＳ上の応用プログラム（以降、ＡＰ：Application Programと略す）に提供するインタフェースは、ローカル処理とリモート処理を同形式にしたい。

５．実現機能

５．１　ＵＮＩＸマシンからの被アクセス

ＷＳからの被アクセス機能として、「ＷＳがリモートファイルアクセス機能として実現しているすべてのシステムコールをサポートする。但し、グループリモートアクセス機能に関するシステムコールは除く。つまり、１８種類(creat, open, read, write, lseek, ioctl, fstat, close, access, chmod, chown, link, mount, stat, umount, unlink, dup, fcntl)のシステムコールをサポートする」とした。また、ＣＳ側にはグループリモートファイルアクセス機能を用いた有効なＡＰがないため、単一リモートファイルアクセス機能のみとした。

上記機能により、［条件１］を満足する。これにより、ＷＳ上のＡＰは、アクセスするファイルがＷＳ上かＣＳ上かにかかわらず同じ形式でアクセスできる。つまり、同じＡＰでＷＳやＣＳにアクセスできる。また、ファイル操作関連システムコールの大半をサポートすることにより、ＵＮＩＸのファイル操作コマンドのほとんどをＣＳ上のファイルに対して利用できる。

５．２　ＵＮＩＸマシンへのアクセス

ＤＩＲＯＳのファイル管理機能は、ＵＮＩＸのファイル管理機能に比べ高機能であり、例えば４０個以上のシステムコールを提供している。そのため、すべての機能をリモート化することは難しい。そこで、基本アクセス機能である８個のシステムコールをリモート化した。その中のいくつかのシステムコールについては、その機能に制限を加えた。一覧を表２に示す。

表２　ＣＳからＷＳへリモートファイルアクセスできるシステムコール一覧

ｼｽﾃﾑｺｰﾙ名	機　能	制限された機能
CONNECT	ﾌｧｲﾙとｱｸｾｽ識別子の連結	１つのﾌｧｲﾙに対する未ｵｰﾌﾟﾝ状態での連結は１つ
DISCONNECT	ﾌｧｲﾙとｱｸｾｽ識別子の切離し	なし
OPEN	ﾌｧｲﾙのｵｰﾌﾟﾝ	１ﾊﾞｲﾄを最小ｱｸｾｽ単位とするｽﾄﾘｰﾑｱｸｾｽ法のみ
CLOSE	ﾌｧｲﾙのｸﾛｰｽﾞ	なし
READ	ﾌｧｲﾙからのﾃﾞｰﾀ読み込み	現ｱｸｾｽ位置からの読み込み機能のみ
WRITE	ﾌｧｲﾙへのﾃﾞｰﾀ書き出し	現ｱｸｾｽ位置への書き出し機能のみ
CONTROL	ﾌｧｲﾙの制御	tty装置に関するｺﾏﾝﾄﾞのみ
SEEK	ﾌｧｲﾙのｱｸｾｽ位置変更	なし

上記機能は、［条件２］を満足するものである。また、ＵＮＩＸは入出力装置なども外部二次記憶装置のファイルと同様に木構造で管理しているため、表２に示した機能だけでも基本的なアクセス処理のリモート化が可能である。

６．処理方式

分散ファイルシステムをＣＳ上に実現する際には、以下の処理方式が問題になる。

（１）ＷＳからの被アクセス時に発生するＵＮＩＸシステムコール-likeな通信手順コマンドからＤＩＲＯＳシステムコールへの変換方式

（２）ＷＳへのアクセス時に発生する、①ローカル処理とリモート処理を判別する方式、②ＤＩＲＯＳシステムコールからＵＮＩＸシステムコール-likeな通信手順コマンドへの変換方式

以降、６．１節で項目（１）、６．２節で項目（２）の①、６．３節で項目（２）の②、について述べる。また、その処理構成を６．４節に述べる。

６．１　ＵＮＩＸシミュレータ

ＵＮＩＸシステムコール-likeな通信手順コマンドからＤＩＲＯＳシステムコールへの変換は、リモートファイルアクセス機能提供とともにＣＳ上でＵＮＩＸコマンドが使えることによるＤＩＲＯＳのマンマシンインタフェース向上を目的に、ＤＩＲＯＳ上のプロセスがＵＮＩＸシステムコールを利用できる形で実現した。

ＵＮＩＸのシステムコールインタフェースを実現する方式として、以下の３案がある。

｛案１｝ライブラリとして実現し、同一プロセス上でシミュレートする方式

｛案２｝別プロセスでシミュレートする方式

｛案３｝ＯＳ核でシミュレートする方式

ＵＮＩＸコマンドをロードモジュールのまま利用でき、かつシミュレート速度が速い｛案３｝を採用した[5]。

６．２　ローカル／リモート解析

［条件３］を満足するため、アクセス先がローカルかリモートかの判断は以下のようにし、ローカル処理とリモート処理のファイルパス名形式を統一した。

（１）リモートに関する情報をネットワークディレクトリに持つ。

（２）ファイルのパス名途中にネットワークディレクトリがあるとリモート処理と判断する。

６．３　ＤＩＲＯＳシステムコールから通信手順コマンドへの変換

ＤＩＲＯＳシステムコールからＵＮＩＸシステムコール-likeな通信手順コマンドへの変換は、基本アクセス機能のみでよいため、①CONNECT&OPENをopen、②CLOSE&DISCONNECTをclose、③READをread、 WRITEをwrite、④CONTROLをioctl、⑤SEEKをlseekに各々対応づけた。変換は、表２に示したような機能制限チェックを行なうルーチンで実現した。

エラー値は、ローカル処理時に返却されないＤＩＲＯＳ系エラー値は返却せず、ＵＮＩＸ系エラー値を返却するようにした。また、

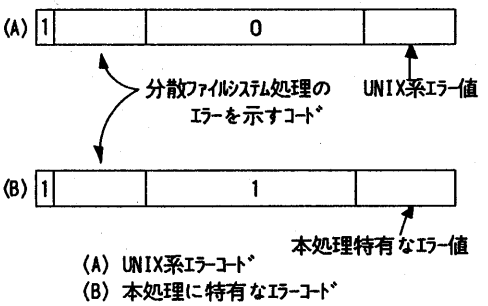


図３　エラー値の体系

その値は、①ＤＩＲＯＳの体系を基本にしながらも、②ＵＮＩＸ固有コードの識別を可能にする、ため図３のようにした。

６．４　処理構成

本機能は、通信手順に基づくコマンド処理以外に、①アクセス時にアクセス先を判別するリモート／ローカル解析処理、②ＬＡＮ制御とのデータ送受信処理、③被アクセス時に通信手順のコマンドに基づいて処理を代行する代行処理からなる。これらをＯＳ核内とＯＳ核外のどちらで実現するかにより、８通りの方式案が考えられる。以下の理由により、①はＯＳ核内、②と③はＯＳ核外に実現する方式を採用した。概要を図４に示す。

（１）［条件３］を満足するため、リモート／ローカル解析処理はＯＳ核内で実現する必要がある。

（２）ＬＡＮ以外のいろいろな通信路サポートを容易に実現できるようにするため、ＬＡＮ制御とのデータ送受信処理はＯＳ核外で実現したい。

（３）複数のリモートファイルアクセス方式を同時にサポートし、かつ性能が許せば、本機能実現時のデバグを容易にするため、できるだけＯＳ核外で実現したい。

図４からわかるように、①代行プロセスは処理の要求と結果取得が１つになった完了型のＵＮＩＸシステムコールを発行するため依頼元のユーザプロセス対応に存在するが、②データ送受信プロセスはＲＯＳの非完了システムコール機能を利用することにより１つのプロセスで実現した。これにより、多数のアクセス／被アクセスを処理する場合でもプロセスの増加を最小限に押えて処理の効率化を図った。

以下に、アクセス処理と被アクセス処理時のＣＳ側の処理フロー例を説明する。

〈アクセス処理〉

①ＡＰが発行したシステムコールについてリモート／ローカルの処理を判別する。

②システムコールの内容をメールボックスに送る。

③メールボックスからシステムコールの内容を受取り、通信手順に従ってパケット化する。

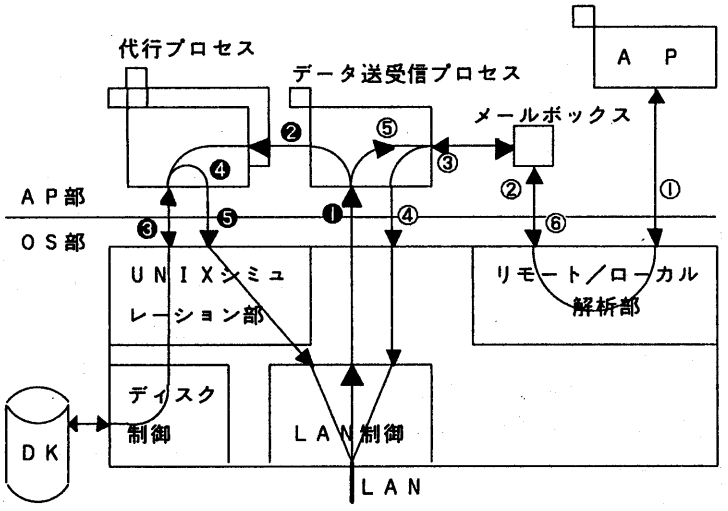


図４　処理構成の概要

④パケットをＬＡＮ制御を介して送る。

⑤パケットをＬＡＮ制御から受信し処理結果をメールボックスに送る。

⑥メールボックスから処理結果を受取りシステムコールの返却値としてＡＰに渡す。

〈被アクセス処理〉

❶パケットをＬＡＮ制御から受信し通信手順に従って解析する。

❷処理内容を対応する代行プロセスに送る。

❸代行プロセスは処理内容に対応するＵＮＩＸシステムコールを発行して処理結果を得る。

❹処理結果を通信手順に従ってパケット化する。

❺パケットをＬＡＮ制御を介して送る。

７．まとめ

異ＯＳ間の分散ファイルシステム構築例として、ＵＮＩＸ-likeなＯＳが走行するマシンとトランザクション処理用ＯＳであるＤＩＲＯＳが走行するマシンをＬＡＮで結合した場合について報告した。

リモートファイルアクセス機能は、そのアクセス方向により機能差があるものの、①ＵＮＩＸ系マシンからは大半のファイルアクセス機能をリモート化でき、②ＤＩＲＯＳマシンからは基本的なファイルアクセス機能をリモート化できた。また、その処理方式においては、ＤＩＲＯＳが提供している非完了システムコール機能を利用することにより、効率的に本機能を実現できることを示した。

本機能により、ＵＮＩＸ系マシンから、ｌｓやｃｐなどのＵＮＩＸコマンドを用いてＤＩＲＯＳが走行するマシン上のファイルを操作できる。特に、スクリーンエディタが利用できるため大変便利である。また、ＤＩＲＯＳマシン上でもＵＮＩＸコマンドを利用できるようになった。

〈参考文献〉

[1] Standberg,R.,et al.: " Design and Implementation of the Sun Network Filesystem " , Proceedings of USENIX Summer Conference , USENIX Association , 1985

[2]Rifkin,A.P.,et al.:" RFS Architectual Overview " ,Proceedings of USENIX Summer Conference , USENIX Association , 1986

[3]谷口、鈴木、瀬々：ﾌｧｲﾙ管理のﾈｯﾄﾜｰｸ化による分散処理OSの構成法 , 情処論文誌第27巻第1号 , pp56-63 , 1986

[4]谷口：分散型ﾘｱﾙﾀｲﾑOSの一構成法 , 情処第34回全大 , 4B-4 , 1987

[5]遠城：異種OS上におけるUNIXｲﾝﾀﾌｪｰｽの実現法 , 情処第33回全大 , 2V-2 , 1986