オペレーティング・システム 37 － 3
（１９８７．１２．４）

高セキュリティ化を目的とした多重仮想ファイルシステム管理方式

西門 隆　　近藤 恵

(株) 日立製作所システム開発研究所

各ノード毎に独立に管理されたファイルをリモートノードから共用する分散ファイル共用の方式として、多重仮想ファイルシステム方式を考案した。この方式は、メモリ管理の多重仮想技術をファイル管理に応用し、ファイルの名称空間を多重化して、ユーザ、グループ、さらには処理毎に別々のファイルシステムを定義することを可能にした。その結果、以下の効果が得られる。

（１） 空間の多重化により部外者のアクセスを防止でき、セキュリティが向上する。
（２） 管理者ではなく、一般ユーザが独自にリモートファイルをアクセスするための環境を設定できる。
（３） 処理固有のファイルシステムにより、ファイルデータのデータアブストラクションができる。

Multi-virtual file system for secure remote file sharing

Takashi Nishikado, Megumu Kondo

Systems Development Laboratory, Hitachi, Ltd., 1099 Ohzenji, Asao, Kawasaki, Kanagawa, 215 Japan

"Multi-virtual file system" is a new file management concept suitable for sharing remote files independently managed on each node. This concept is analogous to multi-virtual technology in memory management, and enables to build individual file name space for each user, group of users and group of programs. With this concept,

(1) file system is made more secure by the separation of file name spaces,
(2) each user, not a system manager, can select remote files to share, and build own environment to access them,
(3) data abstraction of file data is realized by building a file system which is visible only to specific group of programs.

1．緒言

業務の拡大と通信手段の進歩により、各ノード毎に独立に管理されているファイルを別のノードから共用して使用するケースが多くなってきた。こうした分散ファイル共用では、ネットワークを介すため不正なユーザからのアクセスも予想しておかなければならず、ファイルセキュリティに関しては十分な配慮が必要である。

これまでは、リモートのファイルを共用すると言っても、共用データベースのようにネットワーク内の全ユーザや、あるノードの全ユーザに共通で、しかも固定的に必要なファイルに限られていた。そのため、管理者だけが共用するファイルを決定し、アクセス環境を設定することでセキュリティを維持できた。しかし業務が多様化し、さまざまなファイルを一時的に共用したいというニーズが高まっている。すなわち、各ユーザレベルで共用するファイルの決定やアクセス環境の設定ができることを要求されている。そのためには、セキュリティを維持するための新たな手段が必要となる。

そこで、メモリ管理における多重仮想技術をファイル管理に応用し、このニーズに応える「多重仮想ファイルシステム」と呼ぶ方式を考案した。[1)]

以下本稿では、まず従来の分散ファイル管理方式を述べ、ファイル管理に要求されている課題を明確にした後、本多重仮想ファイルシステムについて述べる。

2．従来の分散ファイル管理方式と課題

2.1　従来の分散ファイル管理方式

各ノード毎に独立に管理されたファイルを共用する方式には、大きく言って次の２種類がある。[2)]

（1）　スーパルート方式

〈 方式概要 〉

このスーパルート方式は、図２．１に示すようにネットワークの各ノード上で独立管理されているファイルシステムを統合し、ネットワークで１つのファイルシステムにする方式である。

この方式によると、"/ﾉｰﾄﾞ2/ｄ21/ｆ21"のようにして、ネットワーク上の任意のファイルをどのノードのどのユーザからも指定できる。

〈 利点 〉

・新たに別のファイルを共用するのが容易

常にネットワーク内の任意のファイルを指定できる状態にあるため、新たに別のファイルを共用したい場合にも容易に対処できる。

〈 欠点 〉

・共用対象外のファイルに対する保護が困難

任意のファイルを指定できるため、リモートから見られたくないファイルには１つずつすべてプロテクトをかけなければならない。

・共用ファイルの位置変更に弱い

ファイルの指定方法がネットワーク全体を１つとしたファイルシステム上での絶対位置を指定することにより行うので、例えば共用したいファイルが別のノードに移ったとすると、それを参照していた全ユーザにその影響が及ぶ。

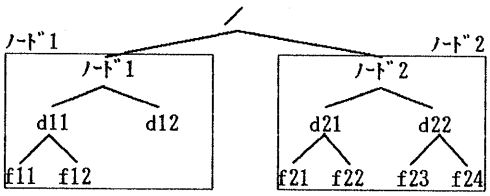


図２．１　スーパルート方式によるファイル共用

（2）ファイルシステム部分リンク方式

〈 方式概要 〉

スーパルート方式とは違い、共用したい他ノードのファイルシステムの一部分を自ノードのファイルシステムにリンクして共用する方式である。

例えば図２．２のようにノード２のｄ21 以下の部分木上にあるファイルを共用する場合、ｄ21をノード１のｄ12にリンクすることにより実現する。このリンクにより、例えばノード２のファイルｆ21は、ノード１から"/ｄ12/ｆ21"としてアクセス可能となる。

上記のリンクは、共用する側（図２．２の例ではノード１）から任意のリモートファイルに対して張れるのではなく、共用される側（図２．２の例ではノード２）がリンクを許すファイルおよびノードを予め宣言し、その許されたノードから許されたファイルに対してのみリンクを張ることができる。従って、不当なノードからのアクセスを阻止できる。

〈 利点 〉

・共用対象外のファイルに対する保護が容易

許可されたファイル以外にはリンクできないう

え、リンク中はリンクした部分木の外にあるファイルをリモートからは指定できないので、共用対象外のファイルに対する保護が容易に行える。

・共用ファイルの位置変更に強い

本方式は、必要な部分を必要な場所にリンクする方式であるため、共用する部分の木構造さえ変わっていなければ、その部分全体がどこへ移ってもリンクさえ張り直せば同一名称で参照できる。

〈 短所 〉

・一般ユーザによるリモートファイルの共用不可

この方式によるリモートファイルのリンクは、その影響がノード内の全ユーザに及ぶ。図２．２の例では、ノード１のどのユーザからもｄ12以下にはノード２のｄ21以下が見えてしまう。このことから、他のユーザへの影響を考えると、リモートファイルへのリンクは一般ユーザに任すことができず、管理者に制限せざるをえない。従って、リモートのあるファイルを自分だけが共用したいといっても事実上不可能である。

・新たに別のファイルを共用するのに操作が必要

リモートにあるファイルを共用するには、共用される側で共用を許す操作と、共用する側でリンクする操作が必要である。この操作は、セキュリティを高める上で効果を持つが、新たにファイルを共用する際必ず必要という点では短所でもある。

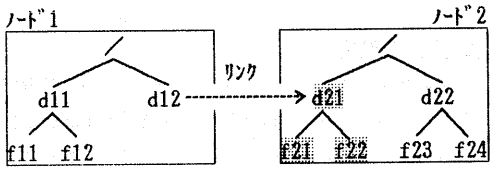


図２．２　部分リンク方式によるファイル共用

２．２　分散ファイル管理の課題

あるノードに存在するファイルを他のノードから共用する場合、他のノードの誰が共用するのかという観点から分類すると次の４つの種類に分類できる。

（１）ネットワーク内のすべてのユーザによる共用

（２）特定ノードのすべてのユーザによる共用

（３）特定のユーザまたはユーザグループによる共用

（４）特定の処理による共用

それぞれの例としては、

（１）掲示板・ニュースファイルの共用

（２）別部門データの共用

（３）あるプロジェクトや個人ファイルの共用

（４）特定のデータベースファイルを検索するデータベース処理プログラムによる共用

がある。

このそれぞれの共用に対して、分散ファイル管理システムは、

（ａ）共用するファイルに対しては、正当なユーザからのアクセスは許し、不当なユーザからのアクセスを排除する機能

（ｂ）共用を許さないファイルに対しては、他のノードやユーザからのアクセスを排除する機能

（ｃ）共用を行いたい者が、自分で共用するファイルを決定し、アクセス環境を設定する機能

を備える必要がある。

これらの観点から前節で述べた従来の管理方式を見直してみると、スーパルート方式は、(１)のタイプの共用が、ファイルシステムの部分リンク方式は、(２)のタイプの共用が可能な分散ファイル管理方式であるといえる。しかし、(３)(４)のタイプの共用については、このどちらの方式も(ａ)～(ｃ)を満足することはできない。そこで(３)(４)のタイプの共用を実現するために考案した方式が、「多重仮想ファイルシステム」である。

３．多重仮想ファイルシステム

３．１　機能概要

多重仮想ファイルシステムは以下の機能を持つ。

（１）各ノードのファイルシステムを共用したいリモートファイルを含んだ形に拡張する機能

（２）(１)の拡張において、ファイルシステムの一部を多重化し、ユーザまたはグループごとに個別のリモートファイルアクセス環境を設定する機能

（３）(２)で作られたファイルシステムの多重化された部分を不要になり次第破棄する機能

これらの機能は前章で示したリモートファイルの

リンクの概念を拡張することにより実現した。すなわち、前章で述べたリンクと同じ機能を持つグローバルリンクの他に、新たにプライベートリンクと呼ぶリモートファイルのリンク機能を設けた。この２つのリンクの違いは、そのリンクの効果の及ぶ範囲が異なることである。グローバルリンクでは、ノード内のすべてのユーザから共通に見えるファイルシステムが変化し、どのユーザから見てもリモートファイルが指定された位置にリンクされているかのように見える。それに対し、プライベートリンクでは、リンク時に指定したグループのメンバにしかそのリンクが有効とならないため、指定した位置にリモートファイルがリンクされているように見えるのは、そのグループのメンバのみである。すなわち、プライベートリンクは、リンクした位置以下のファイルシステムを多重化し、そのリンクで限定したグループメンバのみにその多重化された部分のアクセスを許す。

この機能を具体的に示した図が図３．１である。この例は、ノード１のｄ12 にノード２のｄ21 をグローバルリンクし、ノード３のｄ31及びｄ32をそれぞれ別のグループに限定してノード１のｄ11にプライベートリンクした様子を示す。その結果ノード１の各グループに属するユーザから見えるノード１のファイルシステムは、同図の下図に示す通りとなる。すなわち、ノード２のｄ21以下のファイルはｄ12にリンクされているように全ユーザから見えるが、ノード３のｄ31及びｄ32以下のファイルは、それぞれのプライベートリンク時に限定されたグループのメンバにしかｄ11にリンクされているようには見えない。

リモートファイルをリンクしてファイルシステムを拡張する機能があれば、当然そのリンクを解除する機能が必要である。本方式でも明示的にグローバルリンクやプライベートリンクを解除する機能を持っている。さらに、プライベートリンクについては、明示的に行わなくても自動的に解除する機能を持つ。プライベートリンクでリンクされたリモートファイルは、そのプライベートリンクで限定したグループのメンバにしか見えないので、同グループのメンバが使用しなくなったらそのリンクを残しておいても意味がない。逆に残しておくとファイル管理テーブルがオーバフローする結果となる。従って、プライベートリンク時に指定したグループメンバ全員がリンク先のリモートファイル対するアクセスを完全に終了したと検知した時点でそのリンクを自動的に解除する。（この自動的に解除するタイミングは、グループの取り方に依存するので、詳しくは、後述する。）

グローバルリンクは、その影響がノード内の全ユーザに及ぶため、リンクの設定・解除は、管理者にしか任すことができない。しかしプライベートリンクは、影響が指定したグループ内にとどまるため、このリンクの設定・解除を一般ユーザに任すことも可能である。この事から、プライベートリンクを使用すれば、管理者を煩わせることなく一般ユーザレベルでリモートファイルを共用することができる。

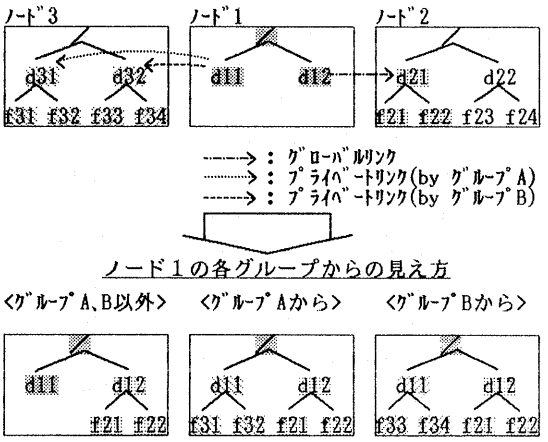


図３．１ 多重仮想ファイル管理の概念

３．２ 応用例

前節の説明では、プライベートリンクの際指定するグループについて具体的な説明を行わなかった。このグループの単純なケースとしては、ユーザの集まりのグループがある。グループとしては、他の集まりを考えることも可能であり、それに応じてプライベートリンクが実現できる効果も異なる。以下本節では、代表的なプライベートリンクの応用例を述べる。

（１）ユーザ固有ファイルシステム

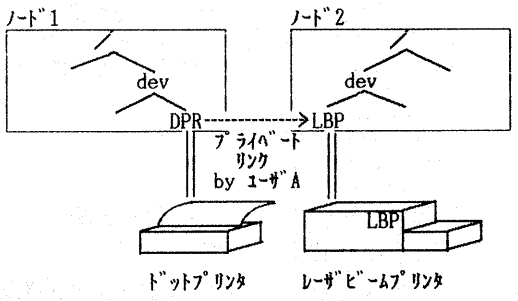
プライベートリンクのグループとして、１ユーザを１グループと考えると、ユーザ固有ファイルシス

テムが実現できる。

このユーザ固有ファイルシステムは、リモートの個人ファイルを共用するようなケースに最適であるが、全ユーザに共通な場所にリモートファイルをリンクし、個人的に同一名称で別の実体をアクセスしたいといったケースにも使用できる。

例えば、図３．２に示すような入出力デバイスの個人的な変更の例を挙げることができる。ノード１、ノード２にそれぞれドットプリンタ及びレーザビームプリンタがあり、通常の状態では、各ノードでの出力は各ノードの /dev/DPR 及び /dev/LBP というファイルを通してそれぞれのプリンタに出力されるという状況を考える。ここでノード１の DPRをノード２の LBPにユーザＡに限定したプライベートリンクで結ぶと、ユーザＡ以外の /dev/DPR への出力は、通常通りノード１のドットプリンタに出力されるが、ユーザＡの出力は、ノード２のレーザビームプリンタに出力される。

このようにユーザ限定のプライベートリンクを使うことにより、他人に迷惑をかけることなくユーザ毎に合った、個人環境を設定できる。



ノード１での/dev/DPRへのアクセス

	出力先デバイス
ユーザＡ	ノード２のレーザビームプリンタ
ユーザＡ以外	ノード１のドットプリンタ

図３．２ プライベートリンクによる出力先の変更

（２）処理固有ファイルシステム

プライベートリンクのグループとして、一連の処理を行うプログラム群を考えると、処理固有のファイルシステムとなる。

例えばデータベース処理を考えると、その処理の中でのみ必要となるファイルが存在する。そのファイルはその処理内でローカルに使用するファイルであるから、関係のないプログラムからは見える必要がない。むしろ見せないようにすべきである。処理固有のファイルシステムは、そのようなケースに最適である。

その具体例を図３．３に示す。ノード２に共用のデータベースファイルがあり、ノード１からそのデータベースを検索・更新するという状況を考える。この図は、そのために、ノード１で動作するデータベース処理プログラム群をグループとし、そのグループに限定してノード２の共用データベースファイルである "/共用DB" をノード１の "/DB" にプライベートリンクした様子を示す。このプライベートリンクは、ノード１のデータベース処理プログラムからはノード２の共用データベースファイルが見えるが、他のプログラムからはそのファイルが見えない状態を作っている。すなわち、データベース処理プログラム以外のプログラムからは直接共用データベースを検索・更新することはできず、必ずデータベース処理プログラムを介さなければならない。

このように、処理固有のファイルシステムは、ファイルデータのアブストラクションを実現している。このファイルデータのアブストラクションは、他人にデータを盗まれたり壊されたりしないというだけでなく、所有者自身も間違えてデータを壊すことがないという点からも有効な手段である。

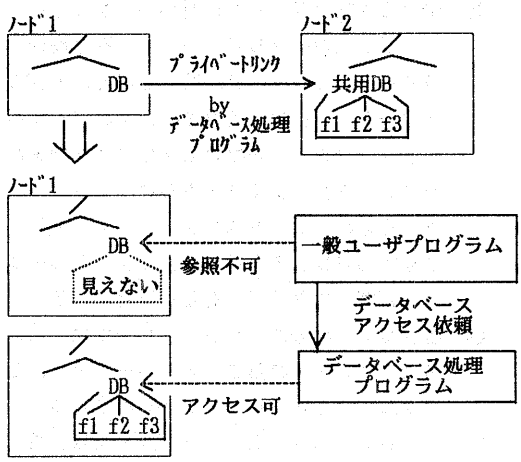


図３．３ 処理固有ファイルシステム

３．３　実現方式

ここでは、以上で述べた多重仮想ファイルシステムの機能を実現する具体的な実現方式について述べる。

（１）プライベートリンクのグループ管理

プライベートリンクを実現するに当ってまず考えなければならないことは、そのリンクの有効範囲であるグループ（以降これをリンクグループと呼ぶ）をいかに設定し、管理するかである。

まず考えられる方式は、一般のＯＳでも行われているようにユーザの集まりとしてリンクグループを静的に管理するという方式である。すなわちログイン時にそのユーザｉｄからリンクグループを決定し、これを用いてプライベートリンクに対するアクセスを制御するという方式である。この方式は、前節の(１)で述べたユーザ１人１人をグループとするユーザ固有ファイルシステムを実現することは可能であるが、前節の(２)で述べた処理固有ファイルシステムは実現できない。その理由は、処理固有ファイルシステムが前提としている一連の処理を集まりとするグループは、ログイン時に静的に定まるという性質のものではなく、その処理を要求したユーザとは無関係に実行時に動的に定まる性質のものであるからである。

そこで、リンクグループを以下のように管理する。

① プロセスに静的なリンクグループｉｄと動的なリンクグループｉｄの２種類を持たせる。

② ①の２種類のグループに対応し、プライベートリンクに静的プライベートリンクと動的プライベートリンクの２種類を設ける。各プライベートリンクは、それぞれ、そのプライベートリンクを行ったプロセスの静的及び動的なリンクグループｉｄと等しいリンクグループｉｄを持つプロセスに有効とする。但し、後述するリンクの継承時は、リンクグループｉｄが等しくなくても有効な場合がある。

③ 各プロセスの静的なリンクグループは、上記のようにユーザの集まりをグループとし、そのグループｉｄをログイン時に設定する。

④ 各プロセスの動的なリンクグループは、特別なシステムコール（genlinkgrp）を用いて設定する。このシステムコールは、新たなリンクグループを生成する機能を持ち、システム内でユニークなグループｉｄをそのプロセスの動的なリンクグループｉｄとして設定する。このシステムコールは、一般ユーザが使用可能である。

⑤ genlinkgrpにより生成された新たなリンクグループと元のリンクグループとの間には、図３．４に示すように、そのグループに対して有効なプライベートリンクに関して包含関係(リンクの継承)を設定できる。すなわち、あるグループで有効なプライベートリンクは、そのグループからgenlinkgrpにより派生したグループに対しても有効とすることができる。このリンクの継承の有無は、genlinkgrp時に指定する。

⑥ プロセスの動的なリンクグループｉｄは、ログイン時にgenlinkgrpシステムコールにより初期設定する。

⑦ あるプロセスから派生したプロセスは、もとのプロセスが持つ静的及び動的なリンクグループｉｄを継承する。

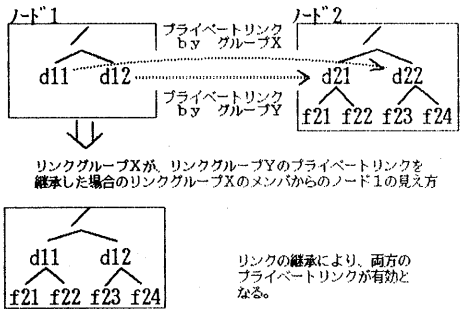


図３．４　プライベートリンクのグループ間継承

以上のような管理を行うことにより前節で述べた様々なグループ固有ファイルシステムを実現できる。

（ａ）ユーザグループ固有ファイルシステム

グループ内のあるユーザが、静的リンクグループを使った静的プライベートリンク行うプログラムを実行することにより実現できる。

（ｂ）ユーザ固有ファイルシステム

(ａ)の特殊ケースとして実現することも可能であるが、わざわざ１ユーザを１グループとするグループを定義しなくても、セッション内でユーザが動的なリンクグループを使ったプライ

ベートリンクを行えば、疑似的にユーザ固有ファイルシステムを構築するのと等しい効果を実現できる。

上述の⑥⑦のリンクグループの管理方式から、図３．５に示すように、１セッション内で派生したプロセスは、明示的にgenlinkgrpシステムコールにより新たなリンクグループを設定しない限りすべて同じリンクグループに属し、他のセッションとは異なるリンクグループとなる。従って、ユーザの要求によりセッション内のあるプロセスで行った動的プライベートリンクは、そのセッション内でのみ共通に有効であり、セッション固有のファイルシステムを作ることに等しい。ログインにより作られるセッションは１ユーザに対応するものであるから、これは疑似的にユーザ固有ファイルシステムを実現していることになる。

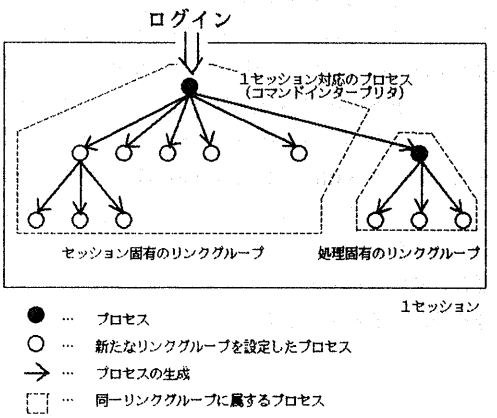


図３．５　各プロセスの動的なリンクグループ

（ｃ）処理固有ファイルシステム

処理固有ファイルシステムは、図３．５に示すように、その処理プログラムが起動された時にgenlinkgrpシステムコールを出して新たなリンクグループを生成することにより実現できる。すなわち、(ｂ)の場合と同様に、その一連の処理を行う大本のプロセスがgenlinkgrpを出すことで、そのプロセスから派生する一連のプロセスだけを新たに生成されたリンクグループにすることができるので、一連の処理に共通でかつ固有なファイルシステムを構築することが可能となる。

この処理固有ファイルシステムでは、処理の内容によって、genlinkgrp時に旧リンクグループのプライベートリンクを継承するかしないかを選択する必要がある。その処理内で参照するファイルがその処理にローカルなファイル又は全ユーザから共通に見えるファイルだけであればリンクの継承は必要ない。しかし、引数としてファイル名をもらい、そのファイルに対して何らかの処理を行うようなケースでは、そのファイルがその処理を依頼したプロセスのリンクグループにのみ有効なプライベートリンクによって結ばれたリモートファイルである可能性があり、このようなケースでは、旧リンクグループのプライベートリンクを継承する必要がある。

（２）プライベートリンクの自動解除

(1)で述べたようにプライベートリンクには、静的プライベートリンクと動的プライベートリンクの２種類がある。静的プライベートリンクについては、対応する静的リンクグループがセッションやプロセスの存続と関係ないため、ユーザからの明示的な解除要求以外の方法で解除することはできない。それに対し動的プライベートリンクの方は、そのグループに属するプロセス全部が処理を終了してしまえば、新たに同じグループに属するプロセスが現れてそのプライベートリンクを使うということはないため自動解除が可能である。すなわち、プロセスの終了を契機に他に同一のリンクグループに属するプロセスがいないことをチェックし、存在しない場合には、そのリンクグループに有効なプライベートリンクを自動的に解除する。但し、リンクの継承により別のリンクグループに対してそのプライベートリンクが有効である場合には、そのグループの消滅まで解除を延期する。

（３）プロテクション管理

グローバルリンクやプライベートリンクは、任意のリモートファイルを任意の位置にリンクできるのではない。リンクできるリモートファイル（リンク先ファイル）とそれをリンクする位置（リンク場所）に対しては、以下の制限がある。

（ａ）リンク先ファイルの制限

リンク先ファイルについては、ファイルの所有者があらかじめ、

① リンクを許すファイル

② リンクを許すノード、ユーザ、グループ

③ アクセス権限（read/write等）

を宣言する。リンク先ファイルとして許されるのは、この宣言によって許可されたファイルのみであり、しかもリンク要求を出したノード、ユーザおよびアクセスの種類が②③の条件を満たした場合のみそのリンクが許可される。

（b）リンク場所の制限

リンク要求を出したユーザの所有するファイル又は書き込みの権限を持つファイル位置のみに限定する。

プライベートリンクによりリンクされたリモートファイルに対するアクセスは、図３．６に示すように、もともとの各ファイルに対するアクセス権をチェックする以前に、リンクグループによって振い落とされる。従って、各ファイルに対して１グループに対するアクセス権しか指定できない場合でも、共用する各ファイルのグループに対するプロテクトを使わずに、リンク先ファイルに対して(a)の②のリンクを許可するグループを複数個列挙しておくことで、セキュリティを保ちながら複数のグループでファイルを共用することができる。

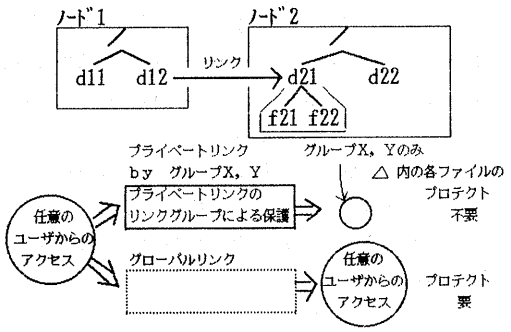


図３．６　リンクグループによるファイルの保護効果

４．結言

４．１ 多重仮想ファイルシステムの効果

以上、多重仮想ファイルシステムについて述べてきた。この多重仮想ファイルシステムの実現により以下の効果を期待できる。

（１）一般ユーザレベルでのリモートファイル共用

プライベートリンクにより、他人に迷惑をかけることなく、個人的にリモートファイルをアクセスするための環境を設定できる。

（２）ファイルセキュリティの向上

プライベートリンクを使うと、部外者にはリモートのファイルを共用していることさえ隠すことができるので、ファイルセキュリティが向上する。

（３）ファイルデータに対するデータアブストラクション

処理固有のファイルシステムを構築できるので、処理内でしか使用しないファイルを隠すことができる。

４．２　今後の課題

本稿で述べた多重仮想ファイルシステムをワークステーション上に実現し、その機能を確認した。多数のノード間でファイル共用を行う際には、

（１）ネットワーク内で共用できるリソース、ノード名等のデータベース管理

（２）アクセス権限のチェックの基盤となるネットワーク全体としてのユーザ、グループｉｄの管理

（３）障害時の処理

等を検討する必要がある。

参考文献

１）西門他：分散環境における多重仮想ファイルシステム管理方式
情報処理学会第３５回全国大会講演論文集
講演番号　３Ｕ－５

２）A.S.Tanenbaum, R.V.Renesse:
Distributed Operating Systems
ACM Computing Surveys, Vol 17, No. 4(1985)