(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4860034号
(P4860034)
(45)発行日　平成24年1月25日(2012.1.25)　(24)登録日　平成23年11月11日(2011.11.11)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 514E

請求項の数 4　(全 21 頁)

(21)出願番号　特願2000-386790 (P2000-386790)
(22)出願日　平成12年12月20日 (2000.12.20)
(65)公開番号　特開2001-229056 (P2001-229056A)
(43)公開日　平成13年8月24日 (2001.8.24)
審査請求日　平成19年12月14日 (2007.12.14)
(31)優先権主張番号　09/498798
(32)優先日　平成12年2月2日 (2000.2.2)
(33)優先権主張国　米国 (US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　３０００
(74)代理人　110000246
特許業務法人ＯＦＨ特許事務所
(74)代理人　100081721
弁理士　岡田　次生
(72)発明者　ブレーン・ディー・ゲーサー
アメリカ合衆国８０５２４コロラド州フォート・コリンズ、セラモンテ・ドライブ　１６００

最終頁に続く

(54)【発明の名称】ファイルアクセス要求処理方法

(57)【特許請求の範囲】
【請求項１】
第１のドライブに格納されていたファイルが第２のドライブに移管されたとき、前記第１のドライブに基づくアプリケーションが、前記移管を反映する修正を該アプリケーションに加えることなく、実行されることができるようにする、コンピュータにより実行される方法であって、
ファイルパスを含むファイルアクセス要求をアプリケーションから受け取ることに応じて、ファイルシステム変換手段が、前記ファイルパスに、前記第１のドライブに関連する前記アクセス要求に含まれるファイルパス（仮想ファイルパスという。）から第２のドライブの物理ファイルパスへの変換、すなわち前方の変換が必要かどうかを変換内容を記憶するレジストリを参照して判断するステップと、
前記前方の変換が必要と判断されるとき、前記ファイルシステム変換手段が、前記仮想ファイルパスを前記物理ファイルパスに変換し、前記仮想ファイルパスが前記前方の変換を必要としないとき、前記ファイルシステム変換手段が、前記仮想ファイルパスを物理ファイルパスとして規定するステップと、
ファイルシステムドライバ手段が、前記ファイルシステム変換手段が規定した物理アドレスを使用してファイルにアクセスする、ファイルアクセス要求処理方法。
【請求項２】
前記ファイルシステムドライバ手段による前記アクセスがなされた後、当該ファイルアクセス要求に関連する物理ファイルパスが前記仮想ファイルパスへの変換、すなわち後方

の変換を必要とするかどうかを、ファイルシステム変換手段が判断するステップと、
前記後方の変換が必要なとき、前記ファイルシステム変換手段が、前記物理ファイルパスを前記仮想ファイルパスに変換して前記アプリケーションに送るステップと、を含み、
前記後方の変換が必要ないとき、前記ファイルシステム変換手段が、前記仮想ファイルパスを前記アプリケーションに送る、請求項１に記載の方法。
【請求項３】
前記仮想ファイルパスに前記前方の変換が必要なとき、前記ファイルシステム変換手段が、前記仮想ファイルパスを物理ファイルパスに変換する前記ステップは、前記仮想ファイルパスおよび物理ファイルパスを係属中要求テーブルに保存することを含み、
前記後方の変換が必要なとき、前記ファイルシステム変換手段が、前記物理ファイルパスを前記仮想ファイルパスに変換して前記アプリケーションに送る前記ステップは、前記ファイルアクセス要求に関連する前記物理ファイルパスに基づいて、前記係属中要求テーブルから前記仮想ファイルパスを読み出すことを含む、請求項２に記載の方法。
【請求項４】
前記変換は、ワイルドカードの使用を含み、前記仮想ファイルパスの一部分を置換して物理ファイルパスとすることができる、請求項１に記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、オペレーティングシステムにおけるファイルアクセス処理および他の入出力処理を仮想化するための方法および装置である。より具体的には、本発明はファイルパスおよび他のリソース識別子に文字列（string）置換を実行し、処理の仮想宛先（destination）を物理宛先に変更することによって、ファイルアクセス処理と他の入出力処理を仮想化する。
【０００２】
【従来の技術】
コンピュータの分野では、コンピュータシステムのリソースへのアクセスを提供するためにオペレーティングシステムが典型的に使用される。アクセスしなければならないリソースの重要なタイプの一つはファイルシステムである。典型的に、ファイルシステムはローカルのハードディスク上のファイルへのアクセスを提供するが、これはネットワークを介してコンピュータシステムに接続するネットワークサーバ上に存在するファイルに対しても同様である。
【０００３】
コンピューティングの進化にとって重要であった一つの動向は、リソースを仮想化し、コンピュータプログラムにリソースを仮想化したものを提供して、リソースの実際の（物理的な）属性をプログラムから見えなくすることである。例えば、当該分野において既知である仮想メモリシステムは、プログラムが一つの大きく連続的なメモリ空間にアクセスできるという幻想をコンピュータプログラムに抱かせる。これは、当該分野において仮想メモリとして知られている。オペレーティングシステムは、仮想メモリと物理メモリの間の変換を管理する。仮想メモリの連続した領域を物理メモリの非連続な領域にマップすることができ、また、仮想メモリの一部をコンピュータシステムのハードディスクドライブ上のスワップファイルに記憶することができる。これによって、コンピュータシステムはコンピュータシステムで物理的に利用可能なメモリより多くのメモリを必要とするプログラムを実行でき、また、複数のプログラムを同時にアクティブにすることができる。仮想化したリソースの例には、他にマルチＣＰＵおよびウインドウ表示のオペレーティングシステムにおけるマルチウィンドウが含まれる。
【０００４】
コンピュータシステムのリソースを仮想化するこの動向にもかかわらず、ファイルシステムは非常に限られた仮想化の特徴を持つようになっている。典型的に、一旦ハードドライブの物理的位置およびそれに含まれるディレクトリ構造とファイルが定義されると、コン

ピュータプログラムはこの構造にロックされてしまう。例えば、Microsoft(商標)社の製品であるWindows NT(商標)またはWindows 98(商標)のオペレーティングシステムを使用する多くのコンピュータユーザが直面する簡単な例を考える。コンピュータユーザは、「c：\」で参照される一つのハードドライブを持つコンピュータシステムを購入する。時間が経って、ハードドライブが一杯になると、ユーザは典型的に「d：\」で参照される二台目のハードドライブを増設する。元の「c：\」ドライブ上のスペースを回復するために、ユーザは、例えばディレクトリ「c：\Program Files」の内容を「d：\Program Files」に移動したいかもしれない。しかし、そうすると多くの問題が起こる。例えば、スタートメニューからプログラムにアクセスするショートカットはすべて「c：\」ドライブを参照しているので、ユーザはもはやプログラムを起動することができない。さらに、レジストリは「c：\」ドライブ上に以前に記憶されたプログラムへの参照を多数含んでいる。典型的に、ユーザは「d：\」ドライブに移動する全てのアプリケーションを再インストールしなければならないか、または、ショートカット、レジストリ、または参照の位置している任意の他の場所にある移動したアプリケーションへのあらゆる参照を発見し、その参照の変更を試みるサードパーティのユーティリティを購入し実行しなければならない。どちらのプロセスとも厄介で、完全に効果的でないこともある。

【０００５】

ある限られた範囲でファイルシステムアクセス動作の仮想化を可能にするメカニズムが当該分野において数多く知られている。例えば、Windows NTまたはWindows 98のオペレーティングシステムでは、全てのプログラムが共通のテンポラリディレクトリにアクセスできるようにする"Temp"と呼ばれるグローバル変数が典型的に定義されている。変数に含まれるファイルパスを変更することによって、全てのプログラムにより使用されるテンポラリディレクトリの位置もまた変更することができる。

【０００６】

Unixオペレーティングシステムにおいて共通な二つの他のメカニズムは、シンボリックリンクとハードリンクである。Unixは、Open Groupの商標である。Unixオペレーティングシステムにおいて、各ファイルおよびディレクトリ位置は「ionode」番号によって定義され、ファイルシステムディレクトリはファイルおよびディレクトリ名とionode番号の間のマッピングを保持する。ハードリンクは、ファイルまたはディレクトリ名と、最初の名前に既にマップされたionode番号の間の単なる第二のマッピングである。ionode番号にマップする全てのファイルとディレクトリ名が削除されるまで、ファイルまたはディレクトリは実際には削除されない。ハードリンクは、同じファイルシステム上のファイルとディレクトリを参照しなければならず、アプリケーションに透過的であり、パーティション間または装置間での任意のリマップに使用することができない。なぜなら、ionode番号はファイルシステム内でファイルおよびディレクトリを参照するだけだからである。

【０００７】

対照的に、シンボリックリンクは自身のionode番号とは別個のファイルである。シンボリックリンクは、別のファイルまたはディレクトリを指すテキスト文字列を含む。従って、シンボリックリンクは異なるファイルシステムにあるファイルとディレクトリの参照に使用することができる。シンボリックリンクは、アプリケーションに透過的でない。なぜなら、アプリケーションはリンクをアクセスしていることは検出できるが、実際のファイルまたはディレクトリを検出することはできないからである。さらに、シンボリックリンクを削除することは、リンクを削除することであり、ファイルまたはディレクトリを削除することではない。ハードリンクと同じように、シンボリックリンクは単一のファイルまたはディレクトリをある位置から別の位置へ変換する。しかし、シンボリックリンクはあるファイルシステムから別のファイルシステムへと任意にリマップされる可能性がある。

【０００８】

マウントポイントも、ある程度までファイルシステムを仮想化することができる。マウントポイントは、単にドライブボリュームが既存のディレクトリツリー構造内のあるポイントでマウントされるのを許す。例えば、ファイルサーバ上のハードドライブがルートディ

レクトリを持ち、ルートディレクトリの下に一連のディレクトリがあり、各ディレクトリはそのディレクトリを使用するユーザのイニシャルで識別されると仮定する。ユーザがログインすると、ドライブボリュームはユーザイニシャルと対応するディレクトリでマウントされる。ユーザがそのドライブボリュームにアクセスするとき、ユーザはマウントポイントの下にあるディレクトリとファイルが見えるだけであり、他のユーザのディレクトリは見えない。マウントポイントは、アプリケーションに透過的で、任意にリマップされることができるが、ファイルレベルの細分性（granularity）を提供しておらず、マウントポイントの下にディレクトリをリマップしない。

【０００９】

最後に、企業記憶サブシステムのなかには、一つのハードドライブから別のハードドライブにユーザのファイルシステムを移動できるものもある。そのような移動は、任意に実行することができ、アプリケーションに透過的であるが、ファイルまたはディレクトリ階層の細分性を提供しない。

【００１０】

【発明が解決しようとする課題】

個別に考えると、上記のメカニズムのどれも、アプリケーションに透過的な方法によって、任意の階層で、また任意のファイルシステム間でファイルおよびディレクトリを任意にリマップする手段を提供していない。本質的に、これらのメカニズムのどれも、ファイルシステムの「仮想ビュー」をアプリケーションに提供することはできない。当該分野において、アプリケーションに透過的な方法によって、任意の階層でおよび任意のファイルシステム間でファイルおよびディレクトリをリマップできるメカニズムが必要とされている。本質的に、当該分野において、仮想メモリシステムが物理メモリシステムをアプリケーションの視点から仮想化しているように、ファイルシステムをアプリケーションの視点から完全に仮想化できるメカニズムが必要とされている。

【００１１】

【課題を解決するための手段】

本発明は、ファイルパスまたは他のリソース識別子に文字列置換を実行し入出力処理の仮想宛先を物理宛先に変換することによって、オペレーティングシステムにおけるファイルアクセス処理および他の入出力処理を仮想化する方法および装置を提供する。本発明によると、仮想ファイルシステム変換ドライバは、ファイルシステムドライバとアプリケーションおよびシステムユーティリティとの間に挿入される。仮想ファイルシステム変換ドライバは、アプリケーションおよびシステムユーティリティからファイルアクセス要求を受け取り、ファイルパスを変換してファイルシステムを仮想化する。仮想ファイルシステム変換ドライバは、ファイルシステムドライバと組み合わせて、結合仮想ファイルシステムドライバを作ることもできる。本発明は、ワークステーション、ネットワークサーバおよび他のコンピューティング装置で利用することができる。

【００１２】

発明の第一の実施形態において、ファイルシステムは部分的に仮想化され、ユーザは仮想ファイルパスと物理ファイルパスの両方を見ることができる。この実施形態において、変換が存在すれば仮想ファイルパスは物理ファイルパスに変換され、ファイルアクセス処理は物理ファイルパスを使用して処理される。変換が存在しないならば、アプリケーションまたはユーティリティによって提供されるファイルパスがファイルアクセス要求を処理するために使用される。

【００１３】

本発明の第二および第三の実施形態において、ファイルシステムはアプリケーションとシステムユーティリティの視点から、完全に仮想化される。第二の実施形態において、ユーザは物理ファイルシステムから始めることができ、また仮想ファイルシステム変換ドライバをインストールすることによってファイルシステムを仮想化することができる。ドライバが初めにインストールされると、全ての仮想ファイルパスはデフォルトで同じ名前の物理ファイルパスに変換すると考える。第三の実施形態において、ファイルとディレクトリ

が作られるとき、仮想変換は全てのファイルパスのために自動的に生成されるが、仮想ファイルパスは制限されることもあり、物理ファイルパスと類似していないこともある。

【００１４】

従来技術のファイルシステムによる不利な点の一つは、特定の記憶装置の属性がファイルシステムの全てのファイルに全体的に適用されてしまうということである。本質的に、記憶装置の属性は、記憶装置上にインストールされるファイルシステムと「腰で接続されて」いる。本発明はこのコネクションを壊す。従って、ユーザは単一のファイルシステムを見ることができ、さらにそのファイルシステム内のファイルは異なる属性を持つ記憶装置に記憶することができる。本発明は多くの他の利点を提供し、また詳細に後述するように、いくつかの方法で実行することができる。

【００１５】

【発明の実施の形態】

本発明は、ファイルパスまたは他のリソース識別子に文字列置換を実行し処理の仮想宛先を物理宛先に変換することによって、オペレーティングシステムにおけるファイルアクセス処理および他の入出力処理を仮想化する方法および装置を提供する。本発明を詳細に説明する前に、最初に、Microsoft社の製品であるWindows NTオペレーティングシステムを使用する典型的な従来技術のコンピュータシステムを考える。当業者は、本明細書中で説明される設計概念を他のオペレーティングシステム（例えばUnix）に適用する仕方について理解するであろう。Unixは、Open Groupの商標である。

【００１６】

図1は従来技術のコンピュータシステム10を示す。ファイルシステムを理解するのに必要なコンピュータシステム10の一部のみが図１に示されている。オペレーティングシステムは、ユーザモード12およびカーネルモード14において命令を実行する。典型的に、アプリケーションおよびシステムユーティリティ16はユーザモード12で実行され、入出力システムサービス18はカーネルモード14で実行される。

【００１７】

入出力システムサービス18はコンピュータシステム10の全ての入出力機能を代表する。入出力システムサービス18は入出力動作を管理する入出力マネージャ20を含むが、これについては後に詳細に説明する。CPU、アダプタ、およびコントローラハードウェアブロック24は、コンピュータシステム10の物理的な演算リソースを代表する。ハードウェア抽象化レイヤー22は、ブロック24を入出力マネージャ20に接続する。ハードウェア抽象化レイヤー22は、（典型的にコンピュータシステム間で異なる）ブロック24により代表されるリソースと通信することを目的とし、入出力マネージャ20にこれらのリソースの一様な「抽象化した」ビューを提供する。

【００１８】

図１で、ファイルシステムへのアクセスを管理する入出力マネージャ20の部分は、ファイルシステムドライバ26、ディスククラスドライバ28、ネットワークプロトコルスタック29、IDEポート30、SCSIポート32およびネットワークポート33である。ファイルシステムドライバ26は、アプリケーションおよびシステムユーティリティ16からファイルアクセス要求を受け取る。ファイルシステムドライバ26は、ファイルアクセス要求がローカルハードドライブまたはネットワークドライブのファイルにアクセスを試みているかどうか判断する。ファイルアクセス要求がローカルハードドライブにアクセスを試みているならば、要求のターゲットであるパーティション上で使用されるファイルシステムのタイプに基づいて、FAT16ブロック34またはNTFSブロック36のどちらかにより要求が処理される。FAT16とNTFSファイルシステムはWindows NTワークステーション上で使用される最も普通のファイルシステムである。しかし、もちろん、他のファイルシステムも当該分野で知られている。ファイルアクセス要求がネットワークドライブにアクセスを試みているならば、要求はネットワーク・ブロック37により処理される。

【００１９】

ファイルアクセス要求がローカルハードドライブにアクセスを試みていると仮定する。要

求は、ファイルシステムに基づく適切なハードドライブセクタにアクセスするよう変換されて、ディスククラスドライバ28に渡される。要求のターゲットがSCSIドライブであるならば、SCSIブロック38はその要求を処理する。要求のターゲットがIDEドライブであるならば、IDEブロック40はその要求を処理する。SCSIインタフェースとIDEインタフェースはハードドライブをコンピュータシステムに接続する最も普通の方法であるが、他の接続方法も当該分野において知られている。その後、ハードドライブインタフェースのタイプに基づいて要求はIDEポート30またはSCSIポート32に渡される。そして、要求はハードウェア抽象化レイヤー22、CPU、アダプタおよびコントローラブロック24により完了する。ファイルアクセス要求の結果は、アプリケーションとシステムユーティリティ16に戻される。

【００２０】

ファイルシステムドライバ26に戻って、次にファイルアクセス要求がネットワークドライブにアクセスを試みていると仮定する。上述したように、そのような要求はネットワーク・ブロック37により処理される。要求はネットワークプロトコルスタック29を介してネットワークポート33に流れ、ハードウェア抽象化レイヤー22、CPU、アダプタおよびコントローラブロック24により完了する。ファイルアクセス要求の結果は、アプリケーションとシステムユーティリティ16に戻される。ネットワークプロトコルの多くのタイプが当該分野において知られていることに注意されたい。例えば、ネットワークコンピューターシステムがイーサネット・ネットワークにおいてTCP/IPプロトコルを介して通信することは、普通のことである。

【００２１】

図２は、本発明に従った仮想ファイルシステム変換ドライバ42を有するコンピュータシステム46を示す。コンピュータシステム46の他のほとんどの構成要素は図１のコンピュータシステム10の対応する構成要素と同様であり、従ってコンピュータシステム10の対応する構成要素と同じ参照数字を持つ。

【００２２】

仮想ファイルシステム変換ドライバ42は、ファイルシステムドライバ26と、アプリケーションおよびシステムユーティリティ16との間に挿入され、Windows NTにおける別個のドライバとしてインストールすることができる。仮想ファイルシステム変換ドライバ42は、アプリケーションおよびシステムユーティリティ16からファイルアクセス要求を受け取り、ファイルパスを変換してファイルシステムを仮想化する。コンピュータシステム46はまた、点線で囲んで示す結合仮想ファイルシステムドライバ44を含むことに注意されたい。結合仮想ファイルシステムドライバ44は、仮想ファイルシステム変換ドライバ42およびファイルシステムドライバ26を含み、ドライバ42と26が結合されてひとつのドライバ44を示す。今後の説明は仮想ファイルシステム変換ドライバ42に関して行われるが、当業者は本明細書中に含まれる教示をドライバ42と26を結合するよう適用して結合仮想ファイルシステムドライバ44を作成する方法を理解するであろう。

【００２３】

本発明には多くの異なる実施形態と構成とがあるが、それらは後述する。本発明は、完全な仮想変換または部分的な仮想変換を提供するために使用することができる。本発明は、サイト要求事項、システムの要求、アプリケーション、ユーザ、プロセス、または他の任意の変数（例えば時刻）に基づいて、個々のファイルおよびディレクトリパスを仮想化することができる。仮想変換は、ローカルユーザ、ローカル管理者、または遠隔管理者によって定義することができ、または自動的に生成することができる。仮想変換は、ネットワーク管理者によって定義され、ネットワークドライブに記憶することができる。これによりワークステーションに記憶される最近受け取った変換のキャッシュコピーを用いてネットワークに接続していないときでもワークステーションは機能することができる。仮想変換は、また、好ましい変換と義務的な変換とを含む階層構造として定義することができる。本発明はまた、ネットワークサーバ上に存在することができ、これによってネットワークドライブをサーバ側で仮想化することができる。最後に、本発明は他のタイプの入出力

処理の動作にも適用することができる。例えば、後述するようにプリント要求を仮想化することができる。

【００２４】

本発明の第一の実施形態を図示するために、「従来の技術」で説明したものと同様の例を考える。コンピュータユーザは、「c：\」で参照される一つのハードドライブを持つコンピュータシステムを購入する。時間が経って、ハードドライブが一杯になると、ユーザは典型的に「d：\」で参照される二台目のハードドライブを増設する。元の「c：\」ドライブのスペースを回復するために、ユーザはディレクトリ「c：\Program Files」の内容を「d：\Program Files」に、ディレクトリ「c：\My Documents」の内容を「d：\My Documents」に移動する。上述したように、ショートカットとレジストリエントリがまだ「c：\」ドライブを指しているので、単にディレクトリを移動することはアプリケーションを「壊して」しまう。

【００２５】

この実施形態において、変換が「vftranslate.ini」と呼ばれるファイルに記憶されると仮定する。より複雑な変換データベースは後に説明され、他の実施形態において典型的に使用される。例えば、レジストリに変換を記憶することは望ましいことがある。しかし、この実施形態では、ユーザはマニュアルでファイル「vftranslate.ini」を編集するか、またはファイルを管理するユーティリティを使用してファイルが以下のエントリを含むようにすると仮定する。

【００２６】

「vftranslate.ini」の内容

c:\Program Files*　　d:\Program Files*

c:\My Documents*　　d:\My Documents*

ファイル「vftranslate.ini」のラインはそれぞれ仮想変換を含む。第一のエントリは仮想ファイル・パスであり、第二のエントリは仮想パスが変換される物理ファイル・パスである。通常のワイルドカード文字が上記の例で使用されているが、「*」は文字列の後に続くキャラクタの任意の数に合わせるために使用されていることに注意されたい。

【００２７】

図３は、発明の第一の実施形態において、仮想ファイルシステム変換ドライバ42がアプリケーションおよびシステムユーティリティ16からのファイル要求を変換する方法を示すフローチャートである。ユーザがワードプロセッサプログラムから「c：\My Documents\letters\document.wpd」と呼ばれるファイルを開こうとすると仮定する。図２において、ファイルアクセス要求は、アプリケーションおよびシステムユーティリティ16から仮想ファイルシステム変換ドライバ42に送られる。図３を参照して、ファイルアクセス要求の第一の段階では、ドライバ42は「開始」ブロック48で実行を開始し、ブロック50でアプリケーションおよびシステムユーティリティ16から仮想ファイルパス「c：\My Documents\letters\document.wpd」を受け取る。

【００２８】

決定ブロック52は、その仮想ファイルパスについて前方の（forward）変換が存在するか否か判断する。仮想変換が存在しないならば、仮想ファイルパスは物理ファイルパスと同じものであり、「NO」ブランチはブロック56へ移動し、そこで物理ファイルパスは図２のファイルシステムドライバ26に送られる。しかし、この例ではファイル「vftranslate.ini」は「c：\My Documents/*」についての変換を含むので、「YES」ブランチは54に移動する。ブロック54は、「c：\My Documents\」を「d：\My Documents\」と置き換えることによって仮想パスに文字列置換を実行する。従って、仮想ファイルパスは、「c：\My Documents\letters\document.wpd」から物理ファイルパス「d: \My Documents\letters\document.wpd」に変換される。ブロック54ではまた、要求が一意に識別できるようにする「係属中（pending）要求」テーブルに仮想ファイルパスと物理ファイルパスを記憶する。係属中要求テーブルに仮想ファイルパスと物理ファイルパスを記憶する理由は後述する。それから制御はブロック56に移り、そこで物理ファイルパスはファイルシステムドライバ26に

送られ、第一の段階の実行は「終了」ブロック58で終わる。この時点で、ファイルアクセス要求は、図２で示されるファイルシステムドライバ26と他の構成要素とによってサービスされる。

【００２９】

ファイルアクセス要求がサービスされたあと、その結果はファイルシステムドライバ26から仮想ファイルシステム変換ドライバ42に戻される。これは、図３で点線60によって表される。従って、ファイルアクセス要求の第二の段階は「開始」ブロック62から始まる。ブロック64で、仮想ファイルシステム変換ドライバ42はファイルシステムドライバ26から物理ファイルパスを受け取る。物理ファイル・パスには、「ファイルd：\My Documents\letters\document.wpdが見つかりません」といったエラーメッセージが伴うことがあり、またはファイルが開かれアクセスされている可能性があるという指示が伴うことがある点に注意されたい。図１のファイルシステムドライバ26とアプリケーションおよびシステムユーティリティ16との間の情報は、図２では仮想ファイルシステム変換ドライバ42を介して流れるが、パスは変換されることがある。

【００３０】

次に、決定ブロック66は、係属中要求テーブルにアクセスしてその物理ファイルパスについて後方の（backward）変換が存在するかどうか判断する。後方の変換がないならば、ファイルシステムドライバ26からの物理ファイルパスは仮想ファイルパスとおなじものであり、「NO」ブランチはブロック70に移動し、仮想ファイルパスはアプリケーションおよびシステムユーティリティ16に送られる。しかし、この例では後方の変換が存在し、「YES」ブランチはブロック68へ移動する。物理ファイルパスを使用して、ブロック68は係属中要求テーブルから仮想ファイルパスを検索して、物理ファイルパスを仮想ファイルパスに逆に変換する。従って、物理ファイルパスは、「d：\My Documents\letters\document.wpd」から仮想ファイルパス「c：\My Documents\letters\document.wpd」に変換される。それから制御はブロック70に移り、ファイルアクセス要求と関連する全ての他の情報と一緒に、仮想ファイルパスはアプリケーションおよびシステムユーティリティ70に送られる。最後に、実行は「終了」ブロック72で終わる。

【００３１】

本発明によって提供される利点の一つは、ディレクトリのリマッピングのプロセスがコンピュータをリブートすることなく実行中にできるということである。例えば、プログラムのディレクトリツリーを別の位置へ動かすことができ、コンピュータユーザによって更新されるファイル「vftranslate.ini」を用いて新しい位置を反映させて、プログラムを直ちに実行することができる。たとえ全てのショートカットとレジストリエントリがプログラムの旧い物理ファイルパスを指すとしても、本発明により旧いファイルパスを新しい物理ファイルパスに仮想化して、プログラムを通常に動作できる。

【００３２】

上記の例では、仮想ファイルパスと物理ファイルパスの間には１対１の対応がある。従って、単にファイル「vftranslate.ini」に含まれる情報を使用することで物理ファイルパスを仮想ファイルパスへ戻すことは可能であろうから、係属中要求テーブルは実際には必要でない。しかし、仮想ファイルパスは常に前方向に一つの物理ファイルパスに帰着されなければならない一方、物理ファイルパスは後方向に複数の仮想ファイルパスへと帰着されることがある。

【００３３】

例えば、コンピュータユーザがさまざまなバージョンのDLLファイルのトラブルを抱えていると考える。頭字語「DLL」はダイナミックリンクライブラリを表す。DLLファイルは、一般的に特定の関数または関数のセットを提供しており、実行可能なルーチンを拡張子.DLLを持つファイルとして別個に記憶することができる。Windowsファミリーのオペレーティングシステムにおける共通の問題の一つは、異なるバージョンのDLLがいろいろなディレクトリに記憶されることがあり、また、一つのプログラムをインストールすると別のプログラムによって必要なDLLファイルのバージョンを上書きすることがあるという点であ

り、これによって別のプログラムを「壊して」しまう。この例では、ユーザがファイル「msvcrt40.dll」（Microsoft Visual C++ v4.0の実行時ライブラリを含む）が問題であると思うと仮定しよう。ユーザは、ハードドライブ上に存在するファイル「msvcrt40.dll」の全てのコピーのバージョンをチェックする（こうすることは、例えば各オカレンス「msvcrt40.dll」を「msvcrt40.old」と名前を付け直し、「msvcrt40.dll」の最新バーションをファイルがあるディレクトリそれぞれにコピーする必要があり得る）代わりに、単にファイル「msvcrt40.dll」の最新バージョンをディレクトリ「c：\dll_library」へコピーし、以下のエントリをファイル「vftranslate.ini」に加えることができる。

【００３４】

「vftranslate.ini」の内容

```
*\msvcrt40.dll          c：\dll_library\msvcrt40.dll
c:\Program Files\*      d：\Program Files\*
c:\My Documents\*        d：\My Documents\*
```

それからユーザは失敗しているプログラムをテストすることができ、ターゲットディレクトリに関係なく、ファイル「msvcrt40.dll」への全ての参照は「c：\dll_library\msvcrt40.dll」に変換される。この例では、多くの仮想ファイルパスが物理ファイルパスに変換するかもしれないので、仮想ファイルシステム変換ドライバ42は仮想ファイルパスへと戻す変換のためにファイル「vftranslate.ini」を使用することができない。そこで、係属中要求テーブルが図3のブロック54とブロック68で必要となる。任意の所与の時間で、一つのファイルアクセス要求だけが係属を許されるのであれば、係属中要求テーブルは必要でない点に注意されたい。複数の要求が係属を許されても、要求が順番に完了すると保証されるならば、係属中要求テーブルは先入れ先出し行列によって実行することができ、仮想ファイルパスだけが待ち行列で記憶される必要がある。最後に、複数の要求が係属を許されて、アウトオブオーダーで完了するのを許されるならば、仮想ファイルパスを物理ファイルパスと関連付けるテーブルとして係属中要求テーブルを実施することができ、テーブルは係属中のファイルアクセス要求の最大数を表すのに十分なエントリを持つことができなければならない。

上記の例も本発明のもう一つの利点を示している。本発明は単にディレクトリを変換することに限定されない。むしろ、本発明は、仮想-物理ファイルパス変換を定義するときファイルレベルでの細分性を提供している。

【００３５】

上記の例で、ファイル「vftranslate.ini」がファイル「msvcrt40.dll」について実際には二つ以上の変換を含むことがある点に注意されたい。例えば、プログラムが「c：\Program Files\Microsoft Office\msvcrt40.dll」にアクセスを試みると仮定する。この仮想パスは、ファイル「vftranslate.ini」で二つの異なる変換を持つ。この実施形態において、ファイル「vftranslate.ini」での最初の有効な変換は使用される変換であると仮定する。従って、「c \Program Files\Microsoft Office\msvcrt40.dll」にアクセスする試みは、物理ファイルパス「c：\dll_library\msvcrt40.dll」に変換する。より洗練された変換階層構造は後述する。

【００３６】

また、具体性の低い変換より具体性の高い変換に優先順位を与えることが典型的に望ましいことに注意されたい。例えば、典型的に仮想ファイルパス「c：\Program Files\Microsoft Office\Word*」から物理ファイルパス「c：\Program Files\Microsoft Office\Word_Beta*」への変換を、仮想ファイルパス「c：\Program Files*」から物理ファイルパス「d：\Program Files*」への変換より優先させたいだろう。

【００３７】

本発明の第一の実施形態において、仮想ファイルパス名から空（null）物理ファイルパス名への変換を作成することが望ましいことがある点に注意されたい。そのような空変換を作成する理由の一つは、既存の物理ファイルパス（変換の仮想部分として）を空物理ファイルパスに変換することによって既存の物理ファイルパスをユーザから隠すためである。

空変換が作成される方法の一つは、ディスクに存在せず、ランダムなキャラクタから成る非常に長い物理パス名のような、作られそうにない変換のための物理ファイルパスを選ぶことである。代替的に、特別な空物理ファイルパスを定義することができ、図3のフローチャート42はこの特別な空物理ファイルパスをトラップし、「ファイルパスが見つかりません」のメッセージを返すように修正することができるが、その方法は、物理ファイルパス名として既に変換に存在する仮想ファイルパス名を同定するファイルアクセス作成要求が後述する第二の実施形態における図4のフローチャート42のブロック82によってトラップされるのと同様の方法である。

【００３８】

上述した発明の第一の実施形態においては、ユーザは仮想ファイルパスと物理ファイルパスの両方を見ることができるので、ファイルシステムは真には仮想化されない。例えば、ユーザがディレクトリ「c：\Program Files」およびディレクトリ「d：\Program Files」を見たならば、ディレクトリは同一に見えるだろう。ある環境で、これは問題を引き起こすことがある。ユーザが「c：\Program Files\Visio」にインストールしたVisio 5.0（Visio Corporationの製品、商標）を持っていると考える。上記の例のように、ユーザが「c：\Program Files」を「d：\Program Files」へ移動して、「c：\Program Files」への全てのファイルアクセス要求を「d \Program Files」へ変換するよう構成する。この時点では、プログラムは通常に動作する。次に、ユーザがVisio 2000にアップグレードするが、同時に前のバージョンも維持しておきたいと思うと仮定する。ユーザは、プログラムが実際には「d：\」ドライブ上に記憶されているのを忘れるかもしれない。これは、プログラムが「c：\」ドライブに存在するように見えるので、忘れるのは全く簡単だろう。Visio 5.0が「c：\」ドライブに存在すると思って、ユーザはVisio 2000インストールプログラムにディレクトリ「d \Program Files\Visio」にVisio 2000をインストールするように指示する。全てのレジストリエントリは旧バージョンが「c：\」ドライブ上にインストールされていることを示すので、インストールプログラムは、それが旧バージョンを上書きしていることを探知することができない。

【００３９】

本発明の第二および第三の実施形態は、アプリケーションおよびシステムユーティリティの視点から、ファイルシステムのビューを完全に仮想化することによって、そのような潜在的な問題について取り組んでいる。第二の実施形態において、ユーザは物理ファイルシステムから始めることができ、仮想ファイルシステム変換ドライバ42と後述する関連した変換コンフィギュレーションユーティリティをインストールすることによってファイルシステムを仮想化することができる。ドライバ42が初めにインストールされると、全ての仮想ファイルパスはデフォルトで同じ名前を付けられた物理ファイルパスに変換されると考える。第三の実施形態において、ファイルとディレクトリが作られるとき、仮想変換は全てのファイルパスについて自動的に生成されるが、仮想ファイルパスは制限されることもあるし、物理ファイルパスと類似していないこともある。

【００４０】

本発明の第二の実施形態を最初に説明する。上述したように、第二の実施形態において、ユーザは物理的なファイルシステムから始めることができる。これが以下の説明であてはまり、またドライバ42が初めにインストールされるとき、全ての仮想ファイルパスはデフォルトで同じ名前の物理ファイルパスに変換するように定義されると仮定する。上述した第一の実施形態を出発点として、ファイルシステムを完全に仮想化するためにはいくつかのルールとステップを加えなければならない。最初に、ファイル「vftranslate.ini」で定義される変換を考える。上述したように、より複雑なデータ構造をファイル「vftranslate.ini」の代わりに使用することができ、あるいは、レジストリで変換を記憶することが望ましいことがある。

【００４１】

完全な仮想化を確実にするために、いくつかの制約を新しい変換の作成の際に与えなければならない。ユーザがファイル「vftranslate.ini」に記憶される新しい変換を作ろうと

していると考える。新しい変換を作成するために、ユーザは新しい仮想ファイルパスと新しい物理ファイルパスを指定しなければならない。新しい変換が作られるとき、変換の新しい物理ファイルパスはハードドライブに存在してはならない。新しい物理ファイルパスがハードドライブに存在するなら、それは上述したようにデフォルトで最初の仮想-物理変換を表しているか、またはファイル「vftranslate.ini」にすでに記憶されている仮想変換と関連していることがある。そのような新しい変換の作成を許すと、二つまたはそれ以上の仮想化ディレクトリのファイルは混ざってしまうことになる。これは、ファイルシステムの仮想化にも言える。しかし、特定のDLLファイルへの全てのアクセスが一つのディレクトリに向け直される上述の例のように、それはこのルールを無効にするために望ましいかもしれない。

【００４２】

この制約のため、ユーザにファイル「vftranslate.ini」を編集させないで、代わりに本発明に従って、ファイルを更新するために変換コンフィギュレーションユーティリティを使用するべきである。ユーザがすでにハードドライブに存在するかまたはファイル「vftranslate.ini」に物理ファイルパスとして存在する新しい物理パスを有する新たな変換を作成しようとするとき、ユーティリティはユーザに警告をしなければならない。そして、ユーザにまだ存在しない（変換コンフィギュレーションユーティリティによって生成できる名前の）異なる新しい物理ファイルパスを選ぶオプションを与えるか、または警告を無効にするオプションを与えることができる。

【００４３】

次に、ユーザが新しい変換を作成するとき、ユーザによって指定される新しい仮想ファイル・パスを考える。新しい仮想ファイルパスが既存の変換の仮想ファイルパスとして、ファイル「vftranslate.ini」に同一のフォームですでに存在するならば、ユーザに警告がなされなければならない。そして、ユーザは、変換エントリを終了するか、または先の変換を削除するか修正するオプションを与えられるべきである。新しい仮想ファイルパスがファイル「vftranslate.ini」に存在しないが、物理パスとしてハードドライブ上に存在するならば、上述したように、新しい仮想ファイルパスとデフォルトで同じ名前を付けられた物理ファイルパスとの間にすでに先の変換が存在する。この場合、ユーザに警告がなされなければならない。変換コンフィギュレーションユーティリティは、作成されている変換の新しい物理ファイルパスへ、同じ名前を付けられた物理パスの内容を移動するためのオプションをユーザに提供することができ、または、ユーザに同じ名前を付けられた物理ファイルパスについての新しい変換を付加的に作成するオプションを与えることができ、それによって新しい仮想ファイルパスは「旧い」同じ名前を付けられた物理ファイルパスにアクセスすることができる。ユーザがディレクトリ（例えばプログラムのベータ版を含むディレクトリ）に一時的な変換を作成し、元の変換を後で復元したいと思うとき、この最後のオプションは有用となることがある。この場合、付加的な新しい変換を作成するとき、ユーザは「記入子（placeholder）」仮想ファイルパスを選択するだろう。変換コンフィギュレーションユーティリティによって提供されるべき特徴の最後の一つは、ファイル「vftranslate.ini」の変換に優先順位付けができることである。この特徴は、ユーザがファイル「vftranslate.ini」にある、または他の任意の適切なメカニズムによって変換の相対的な位置を変更できるようにすることで達成できる。

【００４４】

次に、本発明の第二の実施形態において、図２の仮想ファイルシステム変換ドライバ42がファイルアクセス要求を取り扱う方法について考える。二つのケースを考慮しなければならない。新しい仮想ファイルパスを作成しようとするファイルアクセス作成要求と、既存の仮想ファイルパスにアクセスしようとする既存ファイルアクセス要求である。最初に、既存ファイルアクセス要求を考える。変換が存在するなら、仮想ファイルパスは物理ファイルパスに変換される。変換が存在せず、仮想ファイルパスがファイル「vftranslate.ini」に物理ファイルパスとして存在しないならば、上述したように、デフォルトの変換が仮想ファイルパスと物理ファイルパスの間に存在するので、ファイルアクセス作成要求は

図２のファイルシステムドライバ26に渡される。

【００４５】

しかし、変換が存在しないが、仮想ファイルパスがファイル「vftranslate.ini」に物理ファイルパスとして存在するならば、図２のアプリケーションおよびシステムユーティリティ16に物理ファイルパスの内容を見ることを許してはならない。これは、ファイルシステムの真に仮想化されたビューを提供するために要求される。従って、仮想ファイルシステム変換ドライバ42は、「ファイルパスが見つかりません」のメッセージを返さなければならず、また要求をファイルシステムドライバ26に渡してはならない。

【００４６】

次に、ファイルアクセス作成要求を考える。変換がファイルアクセス作成要求について存在するならば、仮想ファイルパスは物理ファイルパスに変換される。変換がファイルアクセス作成要求について存在せず、仮想ファイルパスがファイル「vftranslate.ini」に物理ファイルパスとして存在しないならば、ファイルアクセス作成要求は図２のファイルシステムドライバ26に渡される。ファイルパスが作成されるとき、上述したように、デフォルトの変換は定義によって作成される。

【００４７】

しかし、ファイルアクセス作成要求が物理ファイルパスとしてファイル「vftranslate.ini」にすでに存在するファイルパスを作成しようとするとき、何が起こるかについて考える。例えば、ユーザがディレクトリ「d \My Documents」を作成するコマンドを出し、ファイル「vftranslate.ini」は、仮想ファイルパス「c：\My Documents」を物理的なパス「d：\My Documents」に変換する変換を含んでいると仮定する。そのようなファイルアクセス作成要求に、同じ名前を付けた物理ファイルパスの作成を許すことはできない。なぜなら、それはすでに存在し、そうすることは二つの仮想ファイルパスを一つの物理パスに結合することになるからである。この問題の解決法は、図２の仮想ファイルシステムドライバ42にファイル「vftranslate.ini」における新しい変換を作成させることである。

【００４８】

図４は、仮想ファイルシステムドライバ42が本発明の第二の実施形態に従って機能する仕方を図示するフローチャートである。図４のフローチャートはファイルアクセス要求の第一の段階を図示するだけである点に注意されたい。第二の実施形態でのファイルアクセス要求の第二の段階は、図３のブロック62、64、66、68、70、および72で示すように、第一の実施形態での第二の段階と同様に処理される。また、後述するように、第二の実施形態の第一の段階は、図３の第一の実施形態の第一の段階で示されるステップの一部を使用することに注意されたい。

【００４９】

まず図４を参照すると、本発明の第二の実施形態に従ったファイルアクセス要求の第一の段階において、ドライバ42は、「開始」ブロック74で実行を開始し、ブロック75でアプリケーションおよびシステムユーティリティ16から仮想ファイルパスを受け取る。次に、決定ブロック76は前方の変換が仮想ファイルパスについて存在するか否か判断する。変換が存在するならば、「YES」ブランチは図３のブロック54に移動し、仮想ファイルパスを物理ファイルパスに変換し、図3を参照して前述したように処理が続く。変換が存在しないならば、「NO」ブランチは決定ブロック78へ移動する。

【００５０】

決定ブロック78は、仮想ファイルパスが既存の変換に物理ファイルパスとしてすでに存在しているか否か判断する。仮想ファイルパスが既存の変換に物理ファイルパスとして存在しないならば、デフォルトの仮想-物理変換が存在しており、「NO」ブランチは図３のブロック56へ移動する。その後、処理は図３を参照して上述したように続く。仮想ファイルパスが既存の変換に物理ファイルパスとして存在するならば、この物理ファイルパスはアプリケーションおよびシステムユーティリティ16から隠されなければならない。従って、「YES」ブランチは、決定ブロック80へ移動する。

【００５１】

決定ブロック80は、ファイルアクセス要求がファイル作成アクセス要求であるかどうか判断する。ファイル作成アクセス要求でないならば、「NO」ブランチはブロック82へ移動する。ブロック82はアプリケーションおよびシステムユーティリティ16に「ファイルパスが見つかりません」のメッセージを返し、処理は「終了」ブロック84で止まる。この状況において、ファイルアクセス要求には、第二の段階がない。

【００５２】

しかし、ファイルアクセス要求がファイル作成アクセス要求である場合は、「YES」ブランチはブロック86へ移動する。ブロック86は、最初に新しい物理ファイルパス名を生成し、その後仮想ファイルパスを新しい物理ファイルパスに変換する変換を作成する。ユーザは新しい物理ファイル名を見ないので、任意の名前を使用することができる。例えば、上述したように、ユーザがディレクトリ「d：\My Documents」を作成するコマンドを出し、ファイル「vftranslate.ini」が仮想ファイルパス「c：\My Documents」を物理パス「d：\My Documents」に変換する変換を含むと仮定する。この状況で、ブロック86は「d：\My Documents～001」のような新しい物理ファイル名を生成することができて、「d\My Documents」を「d\My Documents～001」に変換する変換をファイル「vftranslate.ini」に入れることができる。いまや変換が存在するので、制御は図３のブロック54に移り、処理は上述したように続く。ユーザが「d：\My Documents～001」を見ようとすると、「NO」ブランチは決定ブロック80からブロック82に移動し、ユーザは「ファイルパスが見つかりません」のエラーを受け取る。ユーザがディレクトリ「d：\My Documents～001」をつくろうとすると、「YES」ブランチは決定ブロック80からブロック86に移り、新しい変換が生成される。例えば、ディレクトリ「d：\My Documents～001」を「d：\My Documents～002」に変換することができる。新しい物理ファイルパスを生成する上での唯一の制約は、新しい物理ファイルパスがファイル「vftranslate.ini」においてすでに物理ファイルパスとして存在してはならないということである。

【００５３】

本発明の第二の実施形態の利点の一つは、仮想ファイルシステム変換ドライバ42および上述の変換コンフィギュレーションユーティリティをインストールするだけで、既存の物理ファイルシステムを完全に仮想化することができる点である。第二の実施形態のもう一つの利点は、ブート・パーティションを仮想化できるという点である。ユーザのコンピュータが、そこからコンピュータがブートされる一つのパーティションを有する一つのハードドライブを持っていると仮定する。オペレーティングシステムがブートされているとき、仮想ファイルシステム変換ドライバ42がロードされるまで、ファイルシステムの仮想ビューは利用できない。しかし、コンピュータシステムをブートするのに必要なファイルがそれが位置しているはずの物理ファイルパスで見つかる限り、これは何の問題も引き起こさない。一旦ドライバ42がロードされると、定義によってデフォルトの変換が、ファイル「vftranslate.ini」に変換エントリを持たないハードドライブ上に存在しているファイルについて、仮想-物理変換を作成するので、これらのファイルを見て、処理することができる。本発明の第二の実施形態において、ブートファイルに対するデフォルト変換を変更できないようにするのが望ましいことがある点に注意されたい。言い換えると、変換コンフィギュレーションユーティリティは、コンピュータをブートするのに必要なファイルの位置を変更する変換を作成することはできない。

【００５４】

本発明の第三の実施形態においては、全てのファイルは仮想化され、仮想ファイルパスと物理ファイルパスの間のデフォルト変換は存在しない。加えて、仮想ファイルパス名と物理的なファイル名の間の対応は少ないかまたはなくてもよい。全てのファイルが仮想化されるので、コンピュータシステムを（仮想ファイルシステム変換ドライバ42がインストールされ機能する時点まで）ブートするのに必要なすべてのファイルは、同じ名前を付けられた仮想-物理変換を持っていなければならない。そうすることによって、コンピュータがブートされドライバ42が機能する後仮想的にアクセス可能になる間、コンピュータをブートするのに必要なファイルに物理的にアクセスすることができる。ユーザのファイルを

記憶する位置を制御しようとするシステム管理者によって使用されるとき、第三の実施形態は最も有用であり得る。また、ネットワークドライブを仮想化するときは特に有用であろう。

【００５５】

基本的に、本発明の第三の実施形態を実行するために必要なステップは、本発明の第二の実施形態を実行するために必要なステップのサブセットである。特に、図4において、決定ブロック78は除去され、決定ブロック76の「NO」ブランチは決定ブロック80につながる。従って、仮想ファイルパスが仮想ファイルシステム変換ドライバ42によって受け取られるとき、変換がすでに存在するならば、「YES」ブランチは図３のブロック54へ移動する。変換が存在せず、ファイルアクセス要求が既存ファイルアクセス要求であるならば、「ファイルパスが見つかりません」のメッセージがブロック82で返される。最後に、変換が存在せず、ファイルアクセス要求がファイル作成アクセス要求である場合は、新しい変換がブロック86で生成され、図３のブロック54に制御が移る。

【００５６】

ブロック86により生成された新しい変換によって作成される物理ディレクトリ構造は、仮想ディレクトリ構造に類似している必要はないことに注意されたい。実のところ、上述したように、コンピュータシステムをブートするのに必要とされるファイルを除いては、新しい変換を一つのファイルごとについて生成してもよく、結果として生じる物理ディレクトリ構造は完全に単層（flat）であってもよい。

【００５７】

上述の本発明の三実施形態の説明を容易にするために、単一のユーザがファイルシステムを仮想化することを一般的に仮定していた。しかし、ネットワーク管理者が、本発明に従って、仮想化がユーザに透過的であるように仮想ファイルシステムを構成することのほうが恐らく普通だろう。当該分野において知られられているように、Windows NTワークステーションはデフォルトの管理者アカウントを持ち、ユーザマネージャアプリケーションを使用して付加的にユーザアカウントを作成し構成できる。加えて、システム管理者はリモートでユーザアカウントを管理することができ、ユーザアクセスをリモートドメインコントローラを使用して確認することができる。本発明に従って、管理者が各ユーザについて個別の仮想ファイルシステムを確立したいことがある点に注意されたい。例えば、ワークステーションの各ユーザがディレクトリ「c：\My Documents」についての仮想ファイルパスを持つことが望ましいことがある。なぜなら、これは多くのMicrosoftのアプリケーションによって使用されるデフォルトディレクトリであり、ユーザはこのディレクトリに慣れ親しんでいるからである。しかし、全てのユーザが共通の「c：\My Documents」ディレクトリを共有することは、望ましくないだろう。従って、管理者は各ユーザについて個別の変換を作成することができる。例えばそれは次のようであり、XXXはユーザのイニシャルである。

【００５８】

c：\My Documents*　　c：\My Documents_userXXX*

第一の実施形態では物理ファイルパスは隠されないのに対して、発明の第二の実施形態においては、物理ファイルパスが隠されることに注意されたい。しかし、管理者は、発明の第一の実施形態を使用し、さらに従来技術の手法を使用して物理ファイルパスの許可と属性をセットにすることで物理ファイルパスを隠すことができる。

【００５９】

図５は、コンピュータ・ネットワーク88を示し、本発明に従ってネットワーク管理者が変換を管理する仕方について説明している。ネットワーク88はワークステーション90および変換サーバ92を含み、これらはネットワーク94によって接続される。ワークステーション90は図２の仮想ファイルシステム変換ドライバ42と変換データベース96とを含む。変換データベース96は上述のファイル「vftranslate.ini」によって提供される機能を実行する。変換データベース96は、ローカルユーザまたは管理者によって定義されるローカル/ユーザ定義変換98と、キャッシュネットワーク管理者定義変換100とに分割され、これらは

ネットワーク管理者によりリモートで定義される。変換サーバ92は、サイト、システム、アプリケーション、およびユーザについての変換を記憶するネットワーク管理者定義変換データベース102と、ユーザモードで動作しネットワーク管理者が変換を作成し管理できる変換マネージメント管理アプリケーション104とを含む。

【００６０】

ユーザがワークステーション90にログインするとき、ユーザのログイン情報は変換サーバ92に送られる。変換サーバ92は、サイト、ワークステーション90（システム）、およびユーザが使用するワークステーション90上のアプリケーションについてのネットワーク管理者によって定義された変換、およびユーザに特定の任意の変換を伝えることによって応答する。ワークステーション90は変換データベース96のキャッシュネットワーク管理者定義セクション100にこれらの変換を記憶し、変換が変換サーバ92のネットワーク管理者定義変換データベース102で変更されない限り、またはユーザがログアウトして別のユーザがログインしない限り、変換を再びワークステーション90に送る必要はない。また、ネットワーク管理者定義変換は変換データベース96に保持されるので、ワークステーション90がネットワーク94に接続されないで使用される場合も、変換が利用可能であることに注意されたい。

【００６１】

変換サーバ92の動作がWindows NTネットワークにおけるドメインサーバの動作に類似している点もあることに注意されたい。従って、発明の別の実施形態において、変換サーバ92を結合ドメイン/変換サーバに含めて、ユーザがドメインサーバにログインするとき、変換をワークステーションに提供することもできる。

【００６２】

上述したように、変換を各アプリケーションについて個別に定義することができることは本発明の範囲内である。例えば、ワードプロセッサプログラムと関連する仮想ファイルパス「c：\My Documents」を物理ファイルパス「c :\My Documents-wp」に変換し、表計算ソフトプログラムと関連する仮想ファイルパス「c：\My Documents」を物理ファイルパス「c :\My Documents-ss」に変換することは望ましいことがある。コンピュータになれていないユーザはしばしばディレクトリツリー構造に困惑するので、各アプリケーションについて個別に仮想デフォルトディレクトリを作成することによって、ユーザの混乱を減らすことができる。もちろん、アプリケーションごとに仮想ファイル変換を作成することが望ましい状況は他にも多くある。

【００６３】

義務的である変換を定義し、また好ましい変換を定義することも、本発明の範囲内である。例えば、ネットワーク管理者は、ディレクトリ「c：\My Documents」をファイルが毎日バックアップされるネットワークサーバに向け直す（redirect）好ましい変換を定義することがあり、ユーザはディレクトリ「c：\My Documents」をその本来の位置へ戻すように向け直すために義務的な変換を定義することを望むことがある。

【００６４】

本発明に従った変換を分類する全ての異なる方法を考慮すると、全ての変換を管理することは、非常に複雑になり得る。図６は、優先順位付け方式95を図示するブロック図である。優先順位付け方式95は、本発明に従って変換を管理する一つの方法を図示する。方式95で、仮想ファイルパスは図６の最下部で提示され、各変換カテゴリはマッチング変換のために探索される。変換カテゴリは、最下部から最上部に上る優先順序で順位付けされる。最新の変換カテゴリが図６の最上部で探索されたあと、見つかる最後の変換は使用された変換である（または変換が見つからない）。ユーザ名またはアプリケーションのような、カテゴリと一致するのに必要な全ての他の情報が提供されなければならないことに注意されたい。また、他のカテゴリが方式95に加えられることがある点に注意されたい。

【００６５】

例として、ネットワーク管理者が、ユーザがMicrosoft Wordを使用しているときに仮想ファイルパス「c：\My Documents」を（ネットワークドライブに位置する）物理ファイルパ

ス「g：\user1\My Documents」に変換する好ましい変換を作成したと仮定する。ネットワーク管理者は図５の変換マネージメント管理アプリケーション104を使用して変換を作成し、変換は図５のネットワーク管理者定義変換データベース102に記憶される。ユーザがログインすると、変換はネットワーク94を介してワークステーション90の変換データベース96にキャッシュされる。図６において、この変換は方式95のカテゴリ97に記憶される。次に、ユーザが「c：\My Documents」への全ての参照を「d：\My Documents」に向け直したいと思うと仮定する。ユーザは方式95のカテゴリ99にこの変換を入れることができ、ユーザの変換はネットワーク管理者の変換に優先する。方式95において、ネットワーク管理者に変換に対する最大の制御（ultimate control）を与えて、ネットワーク管理者の義務的な変換の全てがユーザの義務的な変換より高い優先順位を持つ点に注意されたい。

【００６６】

変換方式95の複雑さを考えると、変換データベースを探索するには比較的長い時間がかかることがある。本発明に従って、仮想ファイルシステム変換ドライバ42は、メモリベースのおよびファイルベースの変換キャッシュを備えることができる。図７は、キャッシュの動作を図示するブロック図である。図７で、仮想ファイルパス（およびユーザ名とアプリケーションのような全ての他の必要な情報）はブロック105に提供される。ブロック105で、メモリベースの変換キャッシュが変換または変換が存在しないという指示を含むかどうか判断するために、メモリベースの変換キャッシュが探索される。ヒットがあると、変換の物理ファイルパスが見つかったか、または、変換はないことになる。ミスであると、制御はブロック106に移る。ブロック106で、ファイルベースの変換キャッシュが変換または変換が存在しないという指示を含むかどうか判断するために、ファイルベースの変換キャッシュが探索される。ヒットがあると、変換の物理ファイルパスが見つかったか、または、変換がないことになり、メモリベースの変換キャッシュは更新される。ミスであると、制御はブロック108に移る。最後に、ブロック108で、変換を見つけるために、または変換が存在しないかどうか決定するために、変換データベース96が探索される。ブロック108でも、メモリベースの変換キャッシュおよびファイルベースの変換キャッシュが更新される。変換（または変換が存在しないという指示）が変換キャッシュに記憶される限り、変換データベース96はアクセスされる必要はない。データベース96で変換が作成されるか、変更されるか、または削除される場合、変換の一貫性を保つために変換キャッシュは更新されるかパージされなければならない。

【００６７】

これまでの説明において、本発明は主にワークステーションに関して記述された。しかし、当業者は、サーバ上に記憶されるファイルのネットワークファイルパスを仮想化するために、本発明をサーバ上にインストールすることもできると認めるであろう。サーバ上に記憶されるファイルパスにアクセスするためにサーバがワークステーションから（ネットワークコネクションを介して）要求を受け取るとき、当該分野において知られているように、要求はサーバのネットワークプロトコルスタックを通ってサーバ・モジュールに流れる。その後、サーバ・モジュールは、上述したように、仮想ファイルシステム変換ドライバ42を使用してファイルアクセス要求を処理することによって、サーバに接続する記憶装置のうちの一つからファイルを検索する。そして、結果はネットワークプロトコルスタックを介してワークステーションに送られる。

【００６８】

本発明はまた、他のタイプの入出力処理を仮想化するために使用できる点に注意されたい。例えば、上で説明した設計概念を使用して、プリント要求を一つのプリンタから別のプリンタへと向け直すことができる。これによって、ネットワーク管理者は、ユーザに透過的な方法で、プリント要求を故障しているプリンタから動作中のプリンタに向け直すことができる。同様に、ウェブアドレスおよび他のURLを仮想化することができる。

【００６９】

また、リソースの一つのタイプから別のタイプへと入出力要求を変換するのにも上述の設計概念を使用できることに注意されたい。例えば、仮想ファイルパスを実行しているコン

ピュータプログラムに変換することができる。株価のリストを含むファイルを入力として受け取る表計算ソフトをユーザが持っていると考える。インターネット上の現在の株価を検索する実行可能なプログラムに入力ファイルを変換して、ファイルアクセスの形で表計算ソフトに結果を提供することができる。このように、本発明は、ファイルと一緒でないと動作しない旧いプログラム遺産を強化し、現代のインターネットアクセスの特徴を含めるように使用することができる。そのような変換が有用であり得るもう一つの例は、乱数の生成である。大きな乱数ファイルにアクセスするために、特定のプログラムが構成されることがある。より良いランダム化を提供し、大きなファイルを除去するために、実行中に乱数を生成するプログラムにファイルパスを変換することができる。

【００７０】

従来技術のファイルシステムによる不利な点の一つは、特定の記憶装置の属性がファイルシステムにおける全てのファイルに全体として適用されてしまうことである。本質的に、記憶装置の属性は、記憶装置上にインストールされるファイルシステムと「腰で接続されて」いる。本発明はこの接続を破壊する。従って、ユーザは単一のファイルシステムを見ることができ、さらにそのファイルシステム内のファイルを異なる属性を持つ記憶装置に記憶することができる。例えば、ファイルに素早くアクセスできるローカルハードドライブ上に、テンポラリーファイルとアプリケーション実行可能ファイルを記憶することができる。テンポラリーファイルは失われても被害は生じず、アプリケーション実行可能ファイルは再インストールすることができるので、ローカルハードドライブをバックアップする必要はない。ユーザの文書ファイルは積極的なバックアップスケジュールを持つネットワークデバイスに変換することができ、これによりユーザの２日以上の作業内容が失われないことが保証される。同様に、トランザクション処理に不可欠なデータベースファイルをRAID装置に変換することができる。RAID装置は、自身に記憶されるファイルへの中断しないアクセスを提供する一方ドライブの障害に耐えることができる。従来技術では、ユーザは各タイプのファイルをその適切な記憶装置上に記憶する必要がある。本発明を使用すると、ユーザは単一のファイルシステムを見て、さらに各ファイルを要求される属性を持つ記憶装置に変換することができる。

【００７１】

本発明に従って、好ましい属性要求および義務的な属性要求を各変換に対して記憶することが可能である。従って、ユーザは、RAID記憶装置上に好ましい宛先を持ち、また積極的なバックアップスケジュールの記憶装置上に義務的な宛先を持つような重要なファイルへの変換を構成することができる。変換がリモートで作成されるとき、好ましい属性要求および義務的な属性要求が仮想-物理変換の作成時に使用される。

【００７２】

本発明によって提供されるもう一つの利点は、ファイルを一つの記憶装置から別の記憶装置に移動することができ、ファイルを移動するとき、変換を実行中に調整できるという点である。ユーザは、RAID記憶装置上に好ましい宛先を持ち、積極的なバックアップスケジュールの記憶装置上に義務的な宛先を持つような重要なファイルへの変換を構成したと仮定する。サーバは初めにファイルをRAID記憶装置に割り当てる。しかし、RAID記憶装置は一杯になる。サーバは、RAID記憶装置から積極的なバックアップスケジュールを持つ記憶装置へファイルを透過的に移動することができ、また透過的に変換を更新することができる。RAID記憶装置でスペースが再び利用可能になると、サーバはファイルを戻すことができる。

【００７３】

本発明を好ましい実施形態を参照して記述してきたが、当業者は発明の範囲から逸脱することなく変更が可能であることを認めるであろう。

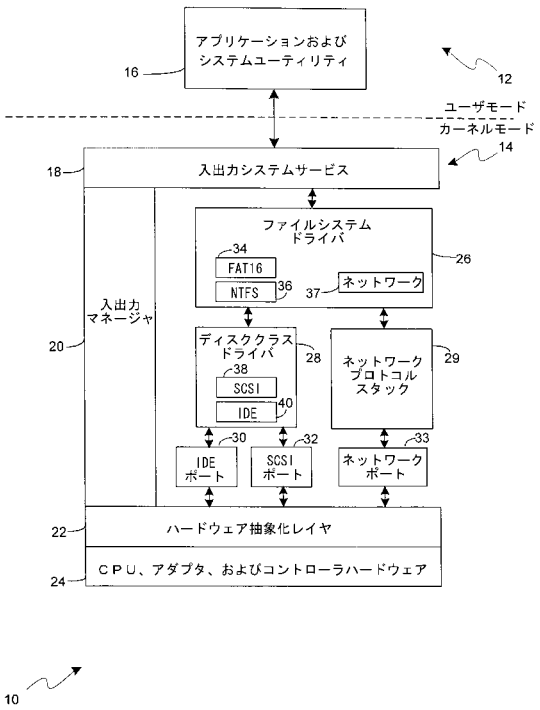
【００７４】

【発明の効果】

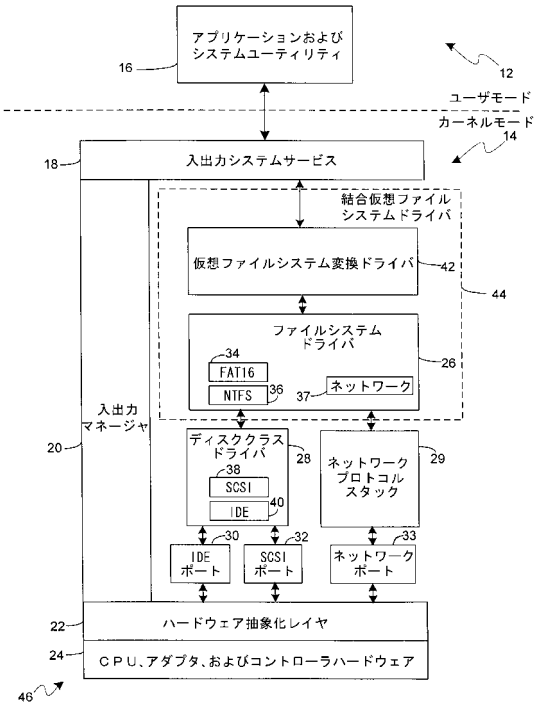
本発明によりワークステーションおよびサーバ上でのファイル管理に大きな柔軟性が提供される。

【図面の簡単な説明】
【図１】従来技術のコンピュータのファイルシステムを理解するに必要な従来技術のコンピュータシステムの一部を示す図。
【図２】本発明に従った仮想ファイルシステム変換ドライバを有するコンピュータシステムを示す図。
【図３】本発明の第一の実施形態に従って、アプリケーションおよびシステムユーティリティからのファイル要求を図2の仮想ファイルシステム変換ドライバが変換する方法を示すフローチャート。
【図４】本発明の第二の実施形態に従ってアプリケーションおよびシステムユーティリティからのファイル要求を図2の仮想ファイルシステム変換ドライバが変換する方法を示すフローチャート。
【図５】ネットワークにより接続されるワークステーションと変換サーバを示し、また本発明に従ってネットワーク管理者が変換を管理する方法を示す図。
【図６】本発明の実施形態に従って変換を管理する優先順位付け方式を図示するブロック図。
【図７】図2の仮想ファイルシステム変換ドライバとともに使用する、メモリベースおよびファイルベースの変換キャッシュを図示するブロック図。
【符号の説明】
１６　　アプリケーションおよびシステムユーティリティ
２６　　ファイルシステムドライバ
４２　　仮想ファイルシステム変換ドライバ
４６　　ファイルシステム

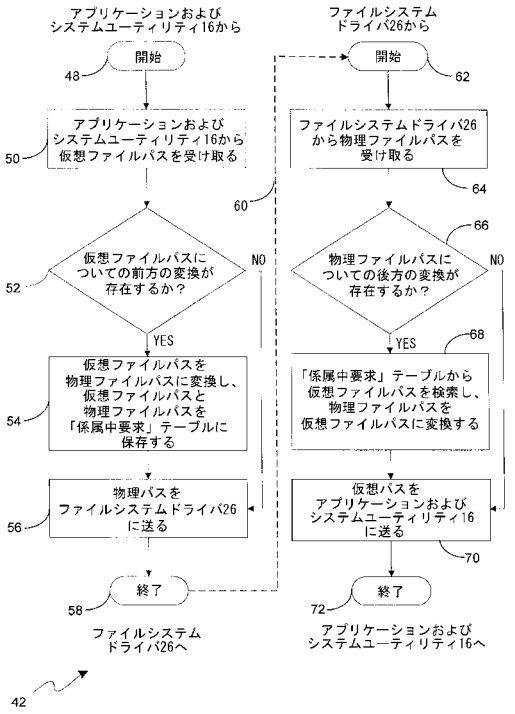
【図１】



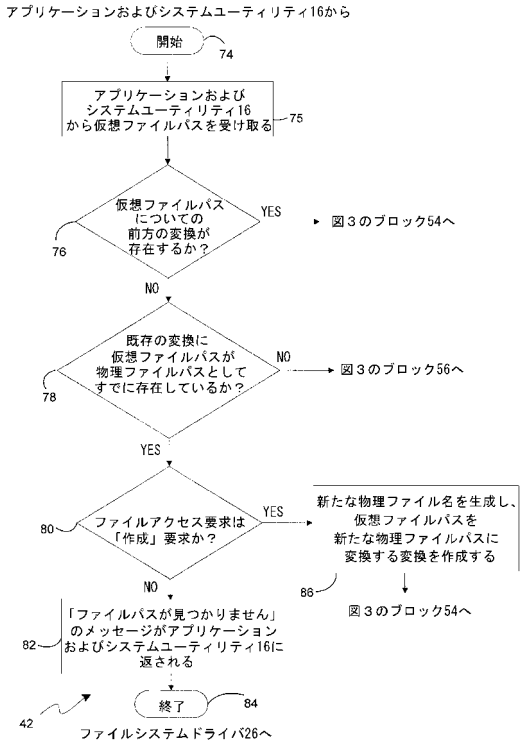
【図２】



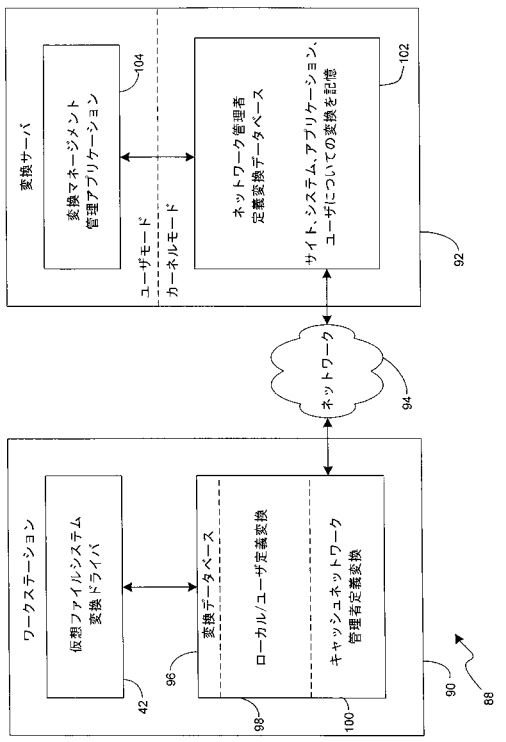
【図３】


アプリケーションおよびシステムユーティリティ16から
ファイルシステムドライバ26から
開始
48
開始
62
アプリケーションおよびシステムユーティリティ16から仮想ファイルパスを受け取る
50
ファイルシステムドライバ26から物理ファイルパスを受け取る
64
60
仮想ファイルパスについての前方の変換が存在するか？
52
NO
YES
物理ファイルパスについての後方の変換が存在するか？
66
NO
YES
68
仮想ファイルパスを物理ファイルパスに変換し、仮想ファイルパスと物理ファイルパスを「係属中要求」テーブルに保存する
54
「係属中要求」テーブルから仮想ファイルパスを検索し、物理ファイルパスを仮想ファイルパスに変換する
物理パスをファイルシステムドライバ26に送る
56
仮想パスをアプリケーションおよびシステムユーティリティ16に送る
70
終了
58
終了
72
ファイルシステムドライバ26へ
アプリケーションおよびシステムユーティリティ16へ
42

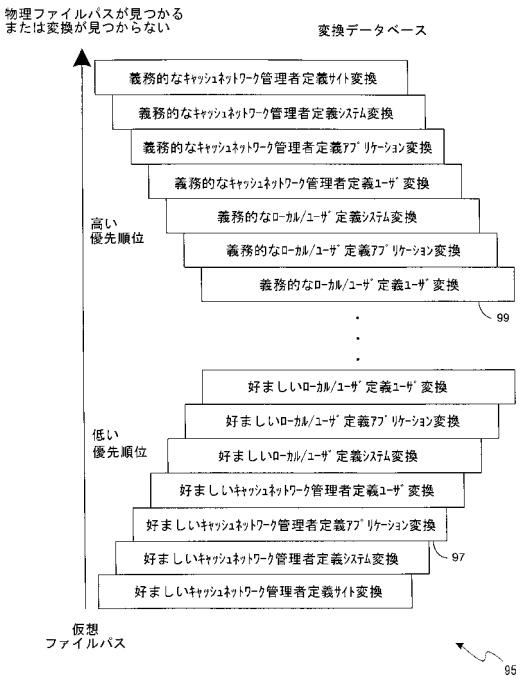
【図４】


アプリケーションおよびシステムユーティリティ16から
開始
74
アプリケーションおよびシステムユーティリティ16から仮想ファイルパスを受け取る
75
仮想ファイルパスについての前方の変換が存在するか？
76
YES
図３のブロック54へ
NO
既存の変換に仮想ファイルパスが物理ファイルパスとしてすでに存在しているか？
78
NO
図３のブロック56へ
YES
ファイルアクセス要求は「作成」要求か？
80
YES
新たな物理ファイル名を生成し、仮想ファイルパスを新たな物理ファイルパスに変換する変換を作成する
86
図３のブロック54へ
NO
「ファイルパスが見つかりません」のメッセージがアプリケーションおよびシステムユーティリティ16に返される
82
終了
84
42
ファイルシステムドライバ26へ

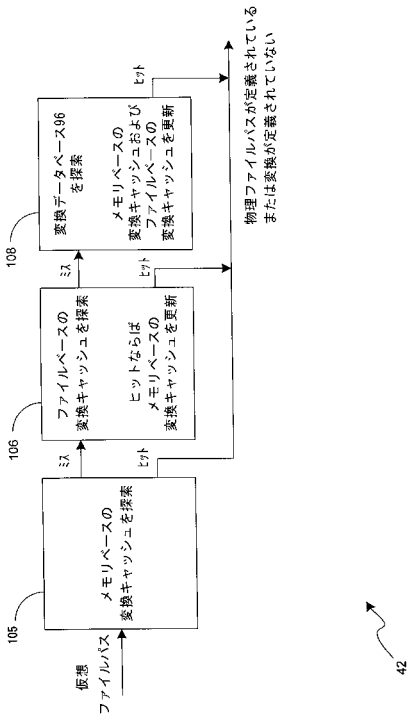
【図５】


変換サーバ
変換マネージメント管理アプリケーション
104
ネットワーク管理者定義変換データベース
サイト、システム、アプリケーション、ユーザについての変換を記憶
102
ユーザモード
カーネルモード
92
ネットワーク
94
ワークステーション
仮想ファイルシステム変換ドライバ
42
変換データベース
ローカル/ユーザ定義変換
キャッシュネットワーク管理者定義変換
96
98
100
90
88

【図６】


物理ファイルパスが見つかるまたは変換が見つからない
変換データベース
義務的なｷｬｯｼｭﾈｯﾄﾜｰｸ管理者定義ｻｲﾄ変換
義務的なｷｬｯｼｭﾈｯﾄﾜｰｸ管理者定義ｼｽﾃﾑ変換
義務的なｷｬｯｼｭﾈｯﾄﾜｰｸ管理者定義ｱﾌﾟﾘｹｰｼｮﾝ変換
義務的なｷｬｯｼｭﾈｯﾄﾜｰｸ管理者定義ﾕｰｻﾞ変換
義務的なﾛｰｶﾙ/ﾕｰｻﾞ定義ｼｽﾃﾑ変換
義務的なﾛｰｶﾙ/ﾕｰｻﾞ定義ｱﾌﾟﾘｹｰｼｮﾝ変換
義務的なﾛｰｶﾙ/ﾕｰｻﾞ定義ﾕｰｻﾞ変換
99
高い優先順位
好ましいﾛｰｶﾙ/ﾕｰｻﾞ定義ﾕｰｻﾞ変換
好ましいﾛｰｶﾙ/ﾕｰｻﾞ定義ｱﾌﾟﾘｹｰｼｮﾝ変換
好ましいﾛｰｶﾙ/ﾕｰｻﾞ定義ｼｽﾃﾑ変換
好ましいｷｬｯｼｭﾈｯﾄﾜｰｸ管理者定義ﾕｰｻﾞ変換
好ましいｷｬｯｼｭﾈｯﾄﾜｰｸ管理者定義ｱﾌﾟﾘｹｰｼｮﾝ変換
好ましいｷｬｯｼｭﾈｯﾄﾜｰｸ管理者定義ｼｽﾃﾑ変換
好ましいｷｬｯｼｭﾈｯﾄﾜｰｸ管理者定義ｻｲﾄ変換
97
低い優先順位
仮想ファイルパス
95

【図７】


仮想
ファイルパス
105
メモリベースの
変換キャッシュを探索
ミス
ヒット
106
ファイルベースの
変換キャッシュを探索
ヒットならば
メモリベースの
変換キャッシュを更新
ミス
ヒット
108
変換データベース96
を探索
メモリベースの
変換キャッシュおよび
ファイルベースの
変換キャッシュを更新
ヒット
物理ファイルパスが定義されている
または変換が定義されていない
42

フロントページの続き

(72)発明者　ブレット・エー・マッキー
アメリカ合衆国８０５２６コロラド州フォート・コリンズ、リッジウッド・ロード　１７１２
(72)発明者　グレゴリー・ダブリュー・テレン
アメリカ合衆国８０５２５コロラド州フォート・コリンズ、ティンバーウッド・ドライブ　２５０２　ナンバー５５

審査官　池田　聡史

(56)参考文献　特開平０２－０３２４３０（ＪＰ，Ａ）
特開２０００－１４８４１３（ＪＰ，Ａ）
特開平０６－００４４４６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 12/00
JSTPlus(JDreamII)