(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4931255号
(P4931255)

(45)発行日　平成24年5月16日(2012.5.16)　(24)登録日　平成24年2月24日(2012.2.24)

(51)Int.Cl.　　F I
G06F 21/24 (2006.01)　　G06F 21/24 160A
G06F 12/00 (2006.01)　　G06F 21/24 164
　　G06F 12/00 537Z

請求項の数 15　(全 19 頁)

(21)出願番号　特願2008-516269 (P2008-516269)
(86)(22)出願日　平成18年5月30日 (2006.5.30)
(65)公表番号　特表2008-547074 (P2008-547074A)
(43)公表日　平成20年12月25日 (2008.12.25)
(86)国際出願番号　PCT/EP2006/062725
(87)国際公開番号　WO2006/134023
(87)国際公開日　平成18年12月21日 (2006.12.21)
審査請求日　平成21年2月24日 (2009.2.24)
(31)優先権主張番号　11/153,846
(32)優先日　平成17年6月15日 (2005.6.15)
(33)優先権主張国　米国(US)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
INTERNATIONAL BUSINESS MASCHINES CORPORATION
アメリカ合衆国10504　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100108501
弁理士　上野　剛史
(74)代理人　100112690
弁理士　太佐　種一
(74)代理人　100091568
弁理士　市位　嘉宏

最終頁に続く

(54)【発明の名称】仮想化されたファイル・システム

(57)【特許請求の範囲】
【請求項１】
コンピュータによってファイル・システムを提供する方法であって、
ファイル・システムの複数のユーザ中の特定のユーザの各々により実行される各アプリケーションのためにファイル・システム・ビューを作成するステップであって、前記ファイル・システム・ビューの各々が、前記アプリケーションが前記特定のユーザのために実行するときに、前記アプリケーションにより作成されまたは変更された全てのファイルを含み、前記特定のユーザのために作成された前記ファイル・システム・ビュー中の前記全てのファイルは、他のユーザからのアクセスおよび前記特定のユーザのために実行される他のアプリケーションからのアクセスから隔離される、ステップを含み、
前記実行されるアプリケーションの各々が完了したならば、当該アプリケーションのために作成された作成済みファイル・システム・ビューは、前記特定のユーザのための前記アプリケーションの次の実行のために、前記変更されまたは作成されたファイルの各々が使用可能であるように、存続される、方法。
【請求項２】
前記特定のユーザのために作成された前記ファイル・システム・ビューのうちの少なくとも１つ中の少なくも１つのファイルは、オペレーティング・システムのファイルであり、前記ファイル・システム・ビューが作成された前記アプリケーションは、当該オペレーティング・システムの制御下で実行される、請求項１に記載の方法。
【請求項３】

複数の起動可能なアプリケーション・プログラミング・インターフェース・ルーチンを含み、当該ルーチンの各々は１つのタイプのファイル・アクセスを提供し、実行するアプリケーションは、前記の作成済みファイル・システム・ビューの中のファイルにアクセスするために、当該アプリケーションが所望するファイル・アクセスのタイプに従って前記ルーチンのうちの１つを起動することを特徴とする請求項１または２に記載の方法。

【請求項４】

特定のファイルからデータを読み出すリクエストが受け取られたところのアプリケーション及びユーザのために作成された前記ファイル・システム・ビューの中の前記特定のファイルのコピーから、もし当該コピーが当該ファイル・システム・ビューにおいて使用可能ならば、前記データを読み出すステップと、

さもなければ、前記特定のファイルのベース・バージョンから前記データを読み出すステップと、を更に含むことを特徴とする請求項３に記載の方法。

【請求項５】

特定のファイルにデータを書き込むリクエストが受け取られたところのアプリケーション及びユーザのために作成された前記ファイル・システム・ビューの中の前記特定のファイルのコピーに、もし当該コピーが当該ファイル・システム・ビューにおいて使用可能ならば、前記データを書き込むステップと、

さもなければ、前記特定のファイルのベース・バージョンから作られたコピーを前記ファイル・システム・ビューに格納し、格納されたコピーに前記データを書き込むステップと、を更に含むことを特徴とする請求項３に記載の方法。

【請求項６】

特定のファイルを削除するリクエストが受け取られたところのアプリケーション及びユーザのために作成された前記ファイル・システム・ビューから前記ファイルのコピーを、もし前記コピーが当該ファイル・システム・ビューにおいて使用可能ならば、削除するステップと、を更に含むことを特徴とする請求項３に記載の方法。

【請求項７】

前記アプリケーションのうちの選択された１つのために作成されたファイル・システム・ビューに特定のファイルのコピーを、前記選択されたアプリケーションの実行中に前記コピーに対してデータの作成又は変更を行えるように、前記特定のファイルのデータを作成し又は変更するために前記ルーチンのうちの１つが起動されると判定したときに、付け加えるステップを更に含むことを特徴とする請求項３に記載の方法。

【請求項８】

前記アプリケーションのうちの１つ以上の選択されたアプリケーションのために作成された前記ファイル・システム・ビューにアクセスするための許可を、前記特定のユーザが１つ以上の他のアプリケーション及び１つ以上の他のユーザのうちの少なくとも一方に与えることを可能にするステップを更に含むことを特徴とする、請求項１～７のいずれか１項に記載の方法。

【請求項９】

前記アプリケーションのうちの１つ以上の選択されたアプリケーションを実行するときに前記特定のユーザのための前記ファイル・システムに対してなされた変更は、前記選択されたアプリケーションとのために夫々作成された前記ファイル・システム・ビューを除去することによって前記ファイル・システムから除去され得ることを特徴とする、請求項１～８のいずれか１項に記載の方法。

【請求項１０】

前記特定のユーザにより実行される各アプリケーションの実行中に、データが作成又は変更されるべきところのファイルのコピーが前記アプリケーションに対応するファイル・システム・ビューの中に既に存在するか否かを判定するステップと、

前記判定ステップが否定の結果を持ったならば前記ファイルのコピーを前記対応するファイル・システム・ビューに付け加えるステップと、

前記対応するファイル・システム・ビューの中の前記ファイルのコピーにおいて前記デ

ータを作成又は変更するステップと、を更に含むことを特徴とする、請求項１～９のいずれか１項に記載の方法。
【請求項１１】
前記ファイルのコピーは前記ファイルのベース・バージョン及び前記ファイルのスタックされたコピーのうちの少なくとも１つから作成され、前記スタックされたコピーはスタックされたファイル・システム・ビューに対応し、前記スタックされたファイル・システム・ビューは、当該スタックされたファイル・システム・ビューの上に前記対応するファイル・システム・ビューが論理的にスタックされているところのファイル・システム・ビューであることを特徴とする請求項１０に記載の方法。
【請求項１２】
ファイル・システムを提供するためのシステムであって、
ファイル・システムの複数のユーザ中の特定のユーザの各々により実行される各アプリケーションのためにファイル・システム・ビューを作成するための手段であって、前記ファイル・システム・ビューの各々が、前記アプリケーションが前記特定のユーザのために実行するときに前記アプリケーションにより作成されまたは変更されたデータのうちの少なくとも１つを含み、前記特定のユーザのために作成された前記ファイル・システム・ビュー中の前記全てのファイルは、他のユーザからのアクセスおよび前記特定のユーザのために実行される他のアプリケーションからのアクセスから隔離される、手段を備え、
前記実行されるアプリケーションの各々が完了したならば、当該アプリケーションのために作成された作成済みファイル・システム・ビューは、前記特定のユーザのための前記アプリケーションの次の実行のために、前記変更されまたは作成されたファイルの各々が使用可能であるように、存続されることを特徴とするシステム。
【請求項１３】
前記特定のユーザのために作成された前記ファイル・システム・ビューのうちの少なくとも１つ中の少なくも１つのファイルは、オペレーティング・システムのファイルであり、前記ファイル・システム・ビューが作成された前記アプリケーションは、当該オペレーティング・システムの制御下で実行されることを特徴とする請求項１２に記載のシステム。
【請求項１４】
複数の起動可能なアプリケーション・プログラミング・インターフェース・ルーチンを含み、当該ルーチンの各々は１つのタイプのファイル・アクセスを提供し、実行するアプリケーションは、前記の作成済みファイル・システム・ビューの中のファイルにアクセスするために、当該アプリケーションが所望するファイル・アクセスのタイプに従って前記ルーチンのうちの１つを起動することを特徴とする請求項１２または１３に記載のシステム。
【請求項１５】
特定のファイルからデータを読み出すリクエストを受け取るための手段と、
前記リクエストが受け取られたところのアプリケーション及びユーザのために作成された前記ファイル・システム・ビューの中の前記特定のファイルのコピーから、もし前記コピーがそのファイル・システム・ビューにおいて使用可能ならば、前記データを読み出すための手段と、
さもなければ、前記特定のファイルのベース・バージョンから前記データを読み出すための手段と、
を更に含むことを特徴とする請求項１２～１４のいずれか１項に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、計算装置に関し、特に、計算装置上で実行するアプリケーションのためのファイル・システム使用の管理に関する。
【背景技術】

【０００２】
現在のファイル・システムはユーザ毎に（ｏｎ　ａ　ｐｅｒ－ｕｓｅｒ　ｂａｓｉｓ）保護を提供する。すなわち、特定のユーザにより実行されるプログラムが、それについて対応する許可をユーザが持っている任意のファイルを読み、書き、削除し、且つ（又は）実行し得るように、ファイルについてユーザ許可が確立される。これは、今日のコンピュータ・ユーザを悩ませる多くのセキュリティ、信頼性、及びメンテナンスの問題につながる。
【発明の開示】
【発明が解決しようとする課題】
【０００３】
セキュリティ問題の例として、ユーザのコンピュータは、ユーザが無意識のうちに実行したウィルスによる安易な襲撃に対して弱い。所与のユーザのどのファイルをも読んで、ユーザに無断でそれらのファイルからの情報を報告する可能性のあるトロイのプログラムが解き放たれることもある。いわゆる“スパイウェア”が偶然にインストールされることがある。
【０００４】
たとえばプログラム開発者によりなされたプログラミング・エラーを通して、他のファイル・システム問題が余り悪意なく引き起こされることがある。例えば、プログラムが他のプログラムの重要なファイルをそれ自身の（場合によってはコンパチブルでない）バージョンで上書きすることがあり、これは予測できない結果につながることがある。プログラムを完全にアンインストールしないアンインストーラ・プログラムが使用されて、最早使えないアプリケーションの部分を後に残すことがある。また、或る場合には欠陥のあるアンインストーラが提供され、プログラムのアンインストールを困難にする。最早不要なアプリケーションがアンインストールされないときには、システムのリソースが無駄に使用されるかもしれない。
【課題を解決するための手段】
【０００５】
ユーザにより実行される各アプリケーションのためにファイル・システム・ビューが作成され、この特定ユーザ向けビューは、そのアプリケーションを動作させるために必要なオペレーティング・システム・ファイルとこのユーザのためにこのアプリケーションでなされたファイル・システム変更とを含む。実行中にファイル・システムに対してなされた変更は（ユーザの許可に従って）許されるけれども、デフォルトでは、それらの変更は他のアプリケーションにも他のユーザにも見えない。オプションで、ユーザ又はシステム・アドミニストレータは、特定のアプリケーションが他のビューへのアクセスを有すると明示的に定めることができ、そのアクセスはリード・オンリー・アクセス、又はコピー・オン・ライト（ｃｏｐｙ－ｏｎ－ｗｒｉｔｅ）・アクセスなどであり得る。他のビューへのアクセスは、同じユーザにより実行される複数のアプリケーション間で許され得、且つ（又は）このユーザのファイル・システム・ビューのうちの１つ以上が一人以上の他のユーザにより実行されるアプリケーションによりアクセスされることを許すことを含み得る。
【０００６】
１つの代わりのアプローチでは、個々のユーザを顧慮せずにアプリケーションについてファイル・システム・ビューを作成することができ、この場合、そのアプリケーションのビューは、そのアプリケーションを実行する全てのユーザのために使用される。
【０００７】
上記は、要約であるので、やむを得ずに単純化、一般化、及び細部の省略を含んでいる。従って、要約が単なる説明であって、限定をすることを決して意図していないことを当業者は認めるであろう。本発明の他の側面、独創的特徴、及び利点は、以下の非限定的で詳細な説明で明らかとなろう。
【０００８】
本発明は、以下の図面と関連して記述され、図面全体において同様の参照番号は同じ要

素を示す。

【発明を実施するための最良の形態】

【０００９】

本発明の実施態様は、ファイル・システムの領域に仮想メモリの利点をもたらす。検討してみると、仮想メモリは、複数のアプリケーションが他の実行中のアプリケーションに知らせずに実行することを可能にし、その間に、実行中のアプリケーションの集合は同じ物理的メモリを良く管理された仕方で共有する。仮想メモリ使用は、従来の共有メモリ・スペース技術を用いるときには不可能であったセキュリティと信頼性とを提供する。本書においてはファイル・システムを共有する複数のアプリケーションに言及するために“仮想化されたファイル・システム”という用語が使用され、各アプリケーションは、そのファイル・システムからの１つ以上のファイルの、自分自身のビューを有する。

【００１０】

本発明の１つ以上の実施態様において、各アプリケーションは、そのアプリケーションを実行する全てのユーザについて、ユーザ毎に別々のエンティティーとして扱われる。アプリケーションが初めに特定のユーザにより実行されるとき、そのアプリケーションにはベース・ファイル・システムの“新鮮な”ビューが与えられる（“ベース”ファイル・システムのこのビューは、好ましくは、そのアプリケーションを実行するために必要とされるオペレーティング・システム・ファイルを含む）。そのアプリケーションがユーザのために実行されているとき、ファイル・システムに対して行われたどの様な変更も（ユーザの許可に従って）許されるけれども、デフォルトでは、それらの変更は他のアプリケーションにも他のユーザにも見えない。このアプリケーションが同じユーザによって後に実行されるとき、この特定ユーザ向けのビューは、このユーザのためにこのアプリケーションでなされたファイル・システム変更だけを示し、デフォルトでは、他のユーザによりなされた変更、またこのユーザにより他のアプリケーションの実行時になされた変更は、見えない。特に、ユーザの変更のこの隔離は、マイクロソフト・ウィンドウズ・オペレーティング・システムを使用しているときにレジストリに対してなされた変更などの、オペレーティング・システムに対してなされることのあるシステム・ファイル及びメタデータ改変にも適用される。（マイクロソフト、ウィンドウズ、ウィンドウズＮＴ、及びウィンドウズ・ロゴは米国、他の国、又はその両方におけるマイクロソフト社の商標である。

【００１１】

好ましくは、本発明の１つ以上の実施態様は、特定のアプリケーションに関して作成されたユーザのファイル・システム・ビューへの他のアプリケーションによるアクセスをユーザが許すことを可能にする。所望の場合には、ファイル・システム・ビューにアクセスする各々のその様な他のアプリケーションは、そのビュー内のファイルを使用するための別々の許可を与えられ得る。ユーザは、ファイル・システム・ビューにアクセスするための許可を他のユーザに与えることもできる。いずれにせよ、ユーザ毎のセキュリティはなお維持される。（例えば、ユーザは、リード・オンリーのアクセスを他のユーザに認め、又はおそらくコピー・オン・ライト許可を与えることにより、自分のビューのうちの特定の１つへのアクセスを許すことができる。しかし、ユーザのファイルの完全性は維持され、オリジナルのユーザのビューの中のファイルを他のユーザが改変する危険はない。）

【００１２】

これらの実施態様に存するアプリケーションは、ユーザにより明示的に許されなければ他のアプリケーションのデータにはアクセスできないので、本書において開示される技術は、たとえトロイのプログラムがユーザによって無意識のうちに実行されたとしてもトロイのプログラムがユーザの明示的許可なしにユーザのファイルにアクセスすることを防止する。その代わりとして、トロイのプログラムのファイル・システムの変更は、トロイのプログラムのために作成されたビューの中に隔離される。それらの変更は、このビューを除去することによってシステムから除去され得る。同様に、ウィルスはユーザの基本オペレーティング・システムに感染することはできず、また他のアプリケーションに影響を及ぼしたり感染したりすることもできない。なぜならば、ウィルスが実行するときにウィル

スがファイル・システムに対して行った変更はそれ自身の独特のビューに限定される（従って、ウィルスが実行したラン・タイム環境により使用されるファイルのコピーに制限される）からである。

【００１３】

本書において開示されている技術は、インストールされたアプリケーションが、同じオペレーティング・システムを使用するシステム間で移行可能であることを可能にする。アプリケーションのファイル・システム・ビューを第１システムから第２システムへ移すことにより（例えば、システム・ファイルとアプリケーションにより使用／作成されたデータ・ファイルとの集合を含むパッケージを移すことにより）、第２システムはそのアプリケーションを、あたかもそれが依然として第１システム上で実行しているかのように、また同様にあたかもそのアプリケーションの前の実行が第２システム上で行われたかのように、実行することができる。（従来技術では、インストールされているアプリケーションを第１システムから第２システムへ移すことは、普通、ファイルの適切なセットの位置を突き止めて第２システムに移せるように、第１システム上でそのアプリケーションにより使用される各々のファイルを知っていることを必要とする。）

【００１４】

本書において開示される技術を用いて、アプリケーションを、そのファイル・システム・ビューを除去することにより、完全にアンインストールすることができ、もし複数のユーザがそのアプリケーションのビューを持っているならばアドミニストレータはそれらのビューを全て削除することができる。すなわち、アプリケーションによりなされた全ての変更は、それらの変更のレコードを含むビューを除去することにより削除され得る。更に、ユーザは、どのファイルがアプリケーションにより変更されたかを、そのアプリケーションのファイル・システム・ビューを調べることによって、判定することができる。

【００１５】

（例えば、“ｄｌｌ”又は“ｓｏ”ファイルなど）ライブラリの別々のバージョンを使用するアプリケーションは、本書において開示される技術を用いて、他と衝突すること無く夫々のライブラリ・バージョンを各々インストールすることができる。これは、アプリケーションの信頼性と開発の容易さを高めることができる。不規則なプログラム（ａｎ　ｅｒｒａｎｔ　ｐｒｏｇｒａｍ）は、それが作成するか又は改変するものを除いて、ユーザのファイルを破壊することができず、これにより破壊の範囲を限定する。重要なオペレーティング・システム・ファイルは即時回復のために常に無傷である、すなわち、その様なファイルは、一般に、改変され得ない。（例えば１つのアプリケーションがオペレーティング・システム・カーネルを削除しても、その削除はその特定のアプリケーションのファイル・システム・ビューにだけ適用され、カーネルは他のアプリケーションにとっては利用可能なままである。

【００１６】

図１－２は、サンプルのファイル・システム使用シナリオを示す。（各々の場合に、シナリオは単一のユーザの観点から描かれている。）初めに図１を参照して、３つのアプリケーションが特定のオペレーティング・システム（“ＯＳ”）の制御下のシステムで実行していると想定する。ここで、それらのアプリケーションは“アプリケーション１”、“アプリケーション２”、及び“アプリケーション３”と夫々称される。システムのファイルは、図１において参照番号１１０のところに示されている或る数のコア・オペレーティング・システム・ファイルを通例含む。そのオペレーティング・システムへの或る数のパッチも存在することがある。これらのファイルは参照番号１１１のところに示されている。オペレーティング・システムのファイル・アクセス許可がコラム１２０において示されている。図１において参照番号１２１の図形により示されているように、オペレーティング・システム自体がコアＯＳファイル及びパッチへのフル・アクセスを有する。（キー１６０を参照。ここに、図１及び２で用いられる図形が記述されている。図１及び２は２つの許可の使用を示しているけれども、１６０のところに示されているもの以外の、リード・オンリー・アクセス或いは共有ファイル・アクセスなどの、他の許可が指定され得る。

）コア・オペレーティング・システム・ファイル１１０とパッチ１１１とのために１つのファイル・システム・ビューが作成され得、或いは、希望に応じて１１０及び１１１のために別々のファイル・システム・ビューが作成され得る。

【００１７】

この第１シナリオにおいて、図１の参照番号１１２，１１３、及び１１４は、３つのアプリケーションの各々がファイルのセットを使用することを示す。１つ以上の実施態様では、上記のように、ユーザにより実行される各アプリケーションは自分のファイルのための独自のビューを作成する。従って、好ましくは１１２，１１３、及び１１４の各々のために個々のファイル・システム・ビューが作成される（各ビューは好ましくは夫々のアプリケーションにより使用されるオペレーティング・システム・ファイル／パッチを含む）。このシナリオについて、ファイルのセット１１２－１１４はオーバーラップしないと想定する（すなわち、該アプリケーションは、自分たちがコアＯＳファイル１１０及びパッチ１１１を使用する以外にはファイルを共有しない）。実例として、アプリケーション１は企業の従業員について使用される人的資源アプリケーションであり、アプリケーション２はその企業のサプライヤに支払いをするために使われる支払勘定アプリケーションであり、アプリケーション３はオンライン・ショッピング・アプリケーションであると想定する。おそらく、オペレーティング・システムには各アプリケーションにより使用されるファイル（アプリケーションにより新たに作成された任意のファイルを含む）へのフル・アクセスが与えられる。従って、参照番号１２２の図形は、ＯＳ１２０がアプリケーション１のファイル・システム・ビューの中のファイルへのフル・アクセスを有することを示し、同様に、参照番号１２３及び１２４の図形は、ＯＳ１２０が夫々アプリケーション２及びアプリケーション３のためのファイル・システム・ビューの中のファイルへのフル・アクセスを有することを示す。

【００１８】

アプリケーション１のファイル・アクセス許可はコラム１３０において示されている。このシナリオについては、アプリケーション１はオペレーティング・システムのコア・ファイル１１０及びパッチ１１１へのコピー・オン・ライト許可を有する。１３１の図形を参照。アプリケーション１はそれ自身のファイルへのフル・アクセスを有する。１３２の図形を参照。“コピー・オン・ライト（ｃｏｐｙ－ｏｎ－ｗｒｉｔｅ）”アクセス、すなわち“ＣＯＷ”は、この用語が本書で使われるとき、更新が必要であると判定されるまではファイルのベース・バージョンを使用し、その後にそのファイルのコピーを作ってその更新をそのコピーに適用することを指す。そのコピーは、そのアプリケーションの更新されたファイル・システム・ビューにおいて表される。デフォルトでは、その様な変更は、そのビューの外では見えない。

【００１９】

アプリケーション２及び３のファイル・アクセス許可はコラム１４０及び１５０に夫々示されている。図１のシナリオについては、アプリケーション２及び３は、両方共に、オペレーティング・システムのコア・ファイル及びパッチへのコピー・オン・ライト許可を有する。１４１及び１５１の図形を参照。アプリケーション２及びアプリケーション３は、両方共、自分自身のファイルへのフル・アクセスを有する。図形１４２及び１５２を参照。しかし、これらのアプリケーションのいずれも、他のアプリケーションのファイル・システム・ビューからのファイルにアクセスしない。

【００２０】

ここで図２を参照すると、第２ファイル・システム使用シナリオが示されている。図１の場合と同じく、オペレーティング・システム１２０は、図２の１２１で示されているように、コアＯＳファイル１１０及びパッチ１１１へのフル・アクセスを有する。オペレーティング・システムは、この場合には参照番号１１２，２１３、及び１１４により表されるこれらのアプリケーションのファイルへの（すなわち、これらのアプリケーションのために作成されたファイル・システム・ビューへの）フル・アクセスをも有する。図２の参照番号１２２，２２３、及び１２４の図形を参照。

【００２１】
この第２シナリオにおいてアプリケーション１及びアプリケーション３は、上記の同じアプリケーション、すなわち企業の従業員について使用される人的資源アプリケーション及びオンライン・ショッピング・アプリケーション、であると想定する。しかし、”アプリケーション４”と称されるアプリケーションがアプリケーション２に取って代わっており、ここでアプリケーション４は企業の従業員が使うことのできる電子メール・アプリケーションであると想定する。更に、アプリケーション１はその企業の従業員の現行のリストを維持することに関して責任を負い、このリストは（例えば、公認された従業員だけがその企業の電子メール・アドレスにアクセスできることを保証するために）アプリケーション４により調べられると想定する。従って、この第２シナリオでは、アプリケーション１及び４により使用されるファイルにはオーバーラップがある。このシナリオについては、図２は、アプリケーション４がコア・オペレーティング・システム・ファイル１１０及びパッチ１１１のためのファイル・システム・ビュー（１つ又は複数の）へのコピー・オン・ライト・アクセスを有し（参照番号２４１を参照）、また自分自身のファイル２１３を含むファイル・システム・ビューへのフル・アクセス（参照番号２４３を参照）及びアプリケーション１のファイル（の少なくとも幾つか）へのコピー・オン・ライト・アクセスを有する（参照番号２４２を参照）ことを示している。好ましくは、アプリケーション４は、アプリケーション１のファイルのために作成されたファイル・システム・ビューからのファイルのうちの該当するもののコピーを作り、そのコピーをアプリケーション４のためのローカル・ファイル・システム・ビューに格納し、そのローカル・コピーに対する更新を行う。特に、アプリケーション４がそのローカル・コピーに対して行う変更は（ユーザ又は管理者がその様なアクセスを明示的に許さなければ）アプリケーション１には見えない。

【００２２】
本書において開示される仮想化されたファイル・システムを使用するオペレーティング・システムの制御下で実行するアプリケーションは、自分自身の独自のラン・タイム環境を有するものと考えられ得る。アプリケーションのファイル・システム・ビューは、“その上に”該アプリケーションがインストールされたファイル・システム・ビューと、該アプリケーションにより行われたファイル・システム改変とを一般に含む“スタックされた（ｓｔａｃｋｅｄ）”ビューと見なされ得る。この観点からは、他のアプリケーションのためのファイル・システム・ビューは、特定のアプリケーションのファイル・システム・ビューの“下に”スタックされ得る。このスタッキングは、今すぐに記述されるように、図３を用いて説明される。

【００２３】
図３の例において、ベース・オペレーティング・システムがインストールされ、ビュー“Ｖ（０）”を作成する。図３の参照番号３１０を参照。ワード・プロセッサもインストールされ、このオペレーティング・システムの制御下で実行している。従ってワード・プロセッサはオペレーティング・システムの上にインストールされていると考えられて良く、従って、ワード・プロセッサのファイル・システム・ビューはオペレーティング・システムのファイル・システム・ビューの“上に”ある。図３において、ワード・プロセッサのファイル・システム・ビューは、３２０のところに示されて、“Ｖ（１）”と称されている。電子メール・プログラムもオペレーティング・システムの上にインストールされており、従ってそのファイル・システム・ビューはオペレーティング・システムのファイル・システム・ビューの上にスタックされている。電子メール・プログラムのファイル・システム・ビューは“Ｖ（２）”として示されている。参照番号３３０を参照。スプレッドシート・プログラムが実行される予定でワード・プロセッサの上にインストールされていると想定する。図３は、ワード・プロセッサのファイル・システム・ビューＶ（１）の上に積み重ねられているものとして、スプレッドシート・プログラムのファイル・システム・ビュー“Ｖ（３）”を示している。参照番号３４０を参照。（このスタッキングは、例えば、１つ以上のアプリケーションのインストールの後にユーザ又はアドミニストレータ

によって行われ得る。）

【００２４】

この例では、スプレッドシート・プログラムはワード・プロセッサによりインストールされ又は作成された全てのファイルに（適切な許可を用いて）アクセスすることができるが、その逆は不可である。なぜならば、ワード・プロセッサのファイル・システム・ビューＶ（１）はスプレッドシート・プログラムのファイル・システム・ビューＶ（３）の、下にあるスタックの一部分だからである。また、これらのアプリケーションの全ては、その下にあるファイル・システム・ビューＶ（０）に存するオペレーティング・システムによりインストールされたファイルにアクセスすることができる。もし、例えばセキュリティ脆弱性を処理するために、オペレーティング・システムのファイルがパッチされなければならず、そのパッチをファイル・システム・ビューＶ（０）に対して行うことができるならば－－これらのファイルを更新するために適切な認証が確立されると想定し－－また（ビューにアクセスするための明示的なユーザ／アドミニストレータ許可を想定すれば）ビューＶ（０）の上にインストールされた全てのアプリケーションがそのセキュリティ・パッチから利益を受けるであろう。同様に、もしワード・プロセッサが、新しい文書を作成するなどにより、そのファイル・システム・ビューＶ（１）において変更を行うならば、スプレッドシート・プログラムはそれらの変更をそのスタックされたファイル・システム・ビューＶ（３）を通して見ることができる。（この例は幾つかのビューの間でのファイル・アクセスを説明しているが、それは限定をするものと解されるべきではない。ビュー間での許可は、図３のスタックされたビューに関連して論じられた方法以外の方法で確立され得る。例えば、ユーザ又はアドミニストレータは、ワード・プロセッサと電子メール・アプリケーションとの両方に他方のビューの中のファイルへのフル・アクセス許可が与えられると定めることができる。）

【００２５】

本発明の仮想化されたファイル・システムの中のファイルは、作成され、上書きされ、改変されるなどすることができる。しかし、１つ以上の実施態様では、全ての変更はファイル・システム処理層により捕捉され、この層はそれらの変更を、それらの変更を行うユーザ及びアプリケーションと関連させる。その関連は、リレーショナル・データベースのような永続的記憶装置に記録される。格納されるべきデータの集合は本書では“メタデータ”と称される。その後の、ユーザによる特定のアプリケーションの実行のために、前の実行中にファイル・システムに対してなされた変更は見えるけれども、一般に（例えば、ファイル・システム・ビューへの明示的アクセス許可がなければ）他のアプリケーションはそれらの変更を見ないであろう。その代わりに、他のアプリケーションは、ファイル・システム内のファイルの自分自身のビューを見るであろう。

【００２６】

もしユーザにより実行される２つのアプリケーション同士が相互に作用を及ぼし他方のファイルにアクセスする必要があるならば、一方のアプリケーションによりなされた変更が他方のアプリケーションによって見られるように、それらのファイル・システム・ビューは合併され得る。或いは、１つのアプリケーションが他のアプリケーションのファイル・システム・ビューのファイルにアクセスすることを許すように許可が与えられ得る。例えば、図３の電子メール・プログラムに、ワード・プロセッサのファイル・システム・ビューＶ（１）へのリード・オンリー許可が与えられ得る。これは、例えば、ワード・プロセッサにより作成された文書を電子メール・アプリケーションを用いて送ることを可能にする。

【００２７】

図４は、サンプルのＧＵＩ表示４００を示し、これでファイル・システム・ビューにアクセスするための許可が与えられ得る。このＧＵＩは２ユーザのための特定ユーザ向けビューを利用し得ることを示している（“Ｂｙ　Ｕｓｅｒ（ユーザによる）”と述べている、参照番号４１０を参照）。参照番号４１１及び４１９を参照、ここでユーザは“ｐｉｅｒｃｅｊｕ”及び“ｐｋｄｏ”として特定されている。

【００２８】
図４は、本発明の１つ以上の実施態様によりサポートされ得る代わりのアプローチも示しており、その場合にはファイル・システム・ビューは（前記のように）ユーザ毎に、アプリケーションごとに提供されるだけではない。この、代わりのアプローチでは、ファイル・システム・ビューは、どのユーザがアプリケーションを実行しているかに関わらず、アプリケーションごとにも確立され得る。この後者のアプローチは参照番号４２０のところに示されており、“Ｂｙ　Ａｐｐｌｉｃａｔｉｏｎ（アプリケーションにより）”作成されたファイル・システム・ビューを指示している。この、代わりのアプローチは、全てのユーザが特定のファイル・システム・ビューにおいてファイルの同じバージョンを共有するべきである場合に特に有益であろう。ユーザ毎、アプリケーション毎のビューも、またアプリケーション毎のビューも許す実施態様は、例えば、それを通してこれらのアプローチを選択し得るところの設定インターフェースを提供することができる。
【００２９】
再びユーザ毎、アプリケーション毎のファイル・システム・ビューと関連して、この図ではユーザ“ｐｉｅｒｃｅｊｕ”のためのビューが展開されている。一般的に、参照番号４１２を参照。ここに示されているように、このユーザは“Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ　１（オペレーティング・システム１）”、“Ｐａｔｃｈｅｓ（パッチ）”、“Ｎｏｔｅｐａｄ（ノートパッド）”、及び“Ｅ－ｍａｉｌ　Ｃｌｉｅｎｔ（電子メール・クライアント）”と称されるアプリケーション・ファイル・システム・ビューを有し、この例ではこれらのビューの各々は該ビューの利用可能な論理的コンポーネントを示すように更に展開され得る。例えば、４１３のところの電子メール・クライアント・ビューは、２つのコンポーネント、“Ｃｏｒｅ　Ｆｉｌｅｓ（コア・ファイル）”４１４と“Ｃｒｅａｔｅｄ　Ｆｉｌｅｓ（作成されたファイル）”４１５と、を示すように展開されている。これはオプションの拡張を示しており、これにより１つ以上の実施態様は、それらが当初インストールするファイル（例えば、コア・ファイル（Ｃｏｒｅ　Ｆｉｌｅｓ）コンポーネント４１４）と、それらがインストール後に作成するファイル（例えば、作成されたファイル（Ｃｒｅａｔｅｄ　Ｆｉｌｅｓ）コンポーネント４１５）とについての別々のファイル・システム許可を使用することができる。
【００３０】
４１３のところのテキスト“Ｅ－ｍａｉｌ　Ｃｌｉｅｎｔ（電子メール・クライアント）”の周りに、このアプリケーションのファイル・システム・ビューが（例えば、マウス・ポインタを用いて）選択されていることを示すために、図形ハイライト表示が与えられている。４０１も参照すること。ここに、ウィンドウ４００のための記述情報の形で、選択されたアプリケーションの名称が提示されている。ペイン４４０，４５０、及び４６０は、ユーザ４１１による電子メール・クライアント・アプリケーションの実行のために作成されたビューの中のファイル・システム・アクセス許可を示す。ペイン４４０の中に示されているように、この例では、電子メール・クライアント・アプリケーション・ビューの“Ｃｒｅａｔｅｄ　Ｆｉｌｅｓ（作成されたファイル）”及び“Ｃｏｒｅ　Ｆｉｌｅｓ（コア・ファイル）”コンポーネントの中のファイルに対して直接制御（フル・アクセス）が提供される。従って、電子メール・クライアント・アプリケーションは、これらのファイルに対して読み出し、書き込み、削除などを含む動作を実行することができる。ペイン４５０は、“Ｐａｔｃｈｅｓ（パッチ）”ビュー、“Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ　１（オペレーティング・システム１）”ビュー、及び“Ｃｏｒｅ　Ｓｙｓｔｅｍ（コア・システム）”ビューの中のファイルについてコピー・オン・ライト・アクセスが指定されていることを示す。ペイン４６０は、“Ｎｏｔｅｐａｄ（ノートパッド）”アプリケーション・ビューの“Ｃｒｅａｔｅｄ　Ｆｉｌｅｓ（作成されたファイル）”コンポーネントの中のファイルについてリード・オンリー・アクセスが指定されていることを示す。
【００３１】
ユーザ（或いはおそらくアドミニストレータ）は、例えばファイル・システム・コンポーネントの図形表示をペイン４３０からペイン４４０－４６０へドラッグしてドロップす

ることにより、或いはペイン４４０－４６０間で図形表示をドラッグしてドロップすることにより、図４に示されているもののようなＧＵＩを通してファイル・アクセス許可を指定することができる。

【００３２】

図５は、ファイル・システム・ビューにおけるファイル・コピーの使用の図を提供し、ここで第１ファイル“ＡＢＣ”５００と第２ファイル“ＤＥＦ”５４０との各々についてベース・バージョンが存在する。この例では、ファイルＡＢＣのコピーが３つの異なるファイル・システム・ビューのために作られている。ユーザ“ｐｉｅｒｃｅｊｕ”は、ワード・プロセッサ（“ＷＰ”）アプリケーションについてのビューにおいてコピー５１０を有する。ユーザ“ｐｋｄｏ”は、ワード・プロセッサについてのビューにおいて第１コピー５２０を有し、スプレッドシート・アプリケーションについてのビューにおいて第２コピー５３０を有する。前記のように、これらのコピー５１０－５３０に対してなされた変更は、アドミニストレータ、又はそのコピーがそのために作られたところのユーザが、そのコピーを含むビューにアクセスするための許可を与えなければ、ユーザ間でもアプリケーション間でも見えない。ファイルＤＥＦ５４０のコピーも１つ以上のアプリケーション・ビューにおいて作られ得るけれども、それは図示されていない。例えば、ユーザ“ｐｉｅｒｃｅｊｕ”により実行されるワード・プロセッサがファイルＤＥＦに対して更新を行うことができるように、コピー５１０を含むビューはファイルＤＥＦのコピーを含むこともできる。

【００３３】

図６は、図５からのファイル・コピーと種々のファイル・システム・ビューにおけるその使用とに関する情報を格納するためにメタデータを組み立て得る方法の一例を示す。表の形が図６に示されているけれども、他のデータ構造が使用され得る。この例におけるメタデータ６００は、各ファイルについての項目６１０，６２０を含み、そして、各ファイルについて、そのファイルの特定のビューの“所有者”のアイデンティフィケーション（ＩＤ）６３０（例えば、これがベース・バージョンであることを明示し、或いはコピーがそのために作成されたところのユーザ及びアプリケーションを特定する）と、そのビューで使用されるコピーの場所６４０と、該当する場合そのビューについてのアクセス許可６５０とを提供する。例えば、項目６６０がファイルＡＢＣのベース・バージョン（図５において参照番号５００のところに示されている）のために設けられて、ベース・バージョンがアドレス“００１１２２３３”に置かれていることを明示している。（或いは、各ベース・バージョンの場所を明示するために別々のデータ構造が使用され得、その場合には図６に示されているデータ構造は特定ビュー向けの情報だけを含む。）一般に６７０のところに示されている他の項目は、図５と関連して上で論じられたビューについてファイルＡＢＣのコピーが何処に置かれているかを示す。（本書において好ましい実施態様は“ファイル”のコピーを作るものとして論じられているが、これは限定をするものと解されるべきでない。ファイル全体をコピーすること、或いはその部分をコピーすること、が望ましいことがある。ファイル自体と関連付けられてはいるがファイル自体とは別に格納されるブロック又はセクタを用いて変更を表すことができる。）許可コラム６５０は、この例では、アプリケーション“ＷＰ”を使うときユーザ“ｐｉｅｒｃｅｊｕ”のために作成されたビューにコピー・オン・ライト許可が適用されることを示す。

【００３４】

本発明の範囲から逸脱せずに他の情報をメタデータ６００の中に設けることができるが、それは図示されていない。更に、メタデータのために代わりの編成を使用することができる。例えば、図６に示されているようにファイルによってメタデータを編成する代わりに、使用され得る１つの代わりのアプローチは、特定のユーザのために作成された全てのファイル・コピーが、それらのコピーがそのために作成されたところのアプリケーションに従ってグループ分けされて一緒に保存されるように、ユーザの中でアプリケーションによりメタデータを編成することである。

【００３５】

１つ以上の実施態様では、オペレーティング・システム・アプリケーション・プログラミング・インターフェース（“ＡＰＩ”）が設けられ、これを通してファイル・アクセス・リクエストが行われ、このＡＰＩのルーチンに対する呼出しは、本書に開示されている仮想化されたファイル・システムを提供するファイル・システム層の動作を引き起こす。ＡＰＩ呼出しに応じて、呼出しをしているアプリケーションとユーザとのためのメタデータが調べられて、インボケーションはそれに応じて処理される（図７－９と関連してより詳しく論じられるように）。好ましくは、仮想化されたファイル・システムの使用はアプリケーション開発者にとってはトランスペアレントであって、アプリケーション開発者は、単に、下にあるファイル・システムがどのように実装されるかを顧慮せずにＡＰＩを起動するようにアプリケーションをコード化する。或いは、しかし、或るシナリオでは、仮想化されたファイル・システムに気づいているアプリケーションを書くことが望ましいかもしれない。その様なアプリケーションは、例えば、このアプリケーションのファイル・システム・ビューが他の選択されたアプリケーション（共通のベンダによって作成されたアプリケーションのセットなど）にとってアクセス可能であることをユーザが望むか否かユーザにたずねるコードを含むことができる。

【００３６】

ここで図７－９を参照すると、本発明の実施態様を実施するときに使用され得る論理を描いたフローチャートが提供されている。これらの図の各々が今論じられる。

【００３７】

初めに図７を参照すると、特定のファイルについて書き込みリクエストが受け取られると、リクエストしているユーザ及びアプリケーションが判定される（ブロック７００）。リクエストしているユーザ及びアプリケーションに対応するビューにおいてコピーが作成されたか否かを判定するために、このファイルについてのメタデータが調べられる（ブロック７１０）。（上で論じられたように、図６に示されているようにメタデータをファイルにより編成せずに、代わりの編成を使用することができる。ブロック７１０を、残りのフローチャートの対応する論理と同じく、メタデータ編成のための他のアプローチに合わせて改変することができる。例えば、ファイル名をキーとして用いてリレーショナル・データベースにアクセスする代わりに、図６に示されている構造に従って、キーは、リクエストしているユーザ及びアプリケーションの識別子から構成されても良い。）

【００３８】

ブロック７２０のテストは、メタデータに従って、リクエストしているユーザ及びアプリケーションのためにこのファイルのコピーが既に存在するか否かを調べる。もし否であれば、処理はブロック７３０で続行し、ここでそのファイルのコピーが作られて、このアプリケーションを実行しているこのユーザに対応するファイル・システム・ビューに付け加えられる。コピーは該ファイルのベース・バージョンから作られ得、その場所は（例えば）メタデータから判定され得る。或いは、複数のファイル・システム・ビューがスタックされるとき（図３と関連して論じられたように）、コピーは、現在のビューの下にあって且つこの特定のファイルのビューを含む一番上のビューから作られ得る。その後、ブロック７４０は、この新しく作成されたコピーについての項目を含むようにメタデータを更新する。

【００３９】

ブロック７２０のテストが肯定的結果を持った（すなわち、このアプリケーションを実行しているこのユーザのためのビューに該ファイルのコピーが既に存在する）とき、及びブロック７４０の実行後に、コントロールはブロック７５０に到着する。ブロック７５０で、リクエストしているユーザ及びアプリケーションのためのビューの中のコピーを用いて書き込みリクエストが実行される。その後に処理は図７から出る。

【００４０】

図７はコピー・オン・ライト処理の詳細を示していない。例えば、ファイル中のデータに変更を生じさせるために書き込みリクエストが実際に行われると判定するまでブロック７３０のコピー作成を遅らせるために、図示されている論理をどの様に改変し得るかは当

業者にとっては明白であろう。（更に、本発明の実施態様は、コピー・オン・ライト最適化を含まない書き込み処理を使用することができる。）更に、例えば、リード・オンリー許可を有するファイルについて書き込みリクエストが試みられているのか否か判定する許可検査の詳細を図７は示していない。本発明の教示内容が分かれば、その様な検査のために図７（及び図８及び９）に描かれている論理をどの様に強化し得るかは明白であろう。

【００４１】

図８は、特定のファイルについて受信された読み出しリクエストのための処理を示す。リクエストをしているユーザ及びアプリケーションが判定される（ブロック８００）。リクエストをしているユーザ及びアプリケーションに対応するファイル・システム・ビューにおいてコピーが作成されているか否か判定するために、このファイルについてのメタデータが調べられる（ブロック８１０）。

【００４２】

ブロック８２０のテストは、そのメタデータに従って、リクエストをしているユーザ及びアプリケーションのためにこのファイルのコピーが既に存在するか否かを調べる。もし否であれば、処理はブロック８３０で続行され、ここで読み出しリクエストがファイルのベース・バージョンを用いて実行される。（複数のビューがスタックされたときには、図３に関して論じられたように、“優先権（ｐｒｅｃｅｄｅｎｃｅ）”アプローチが使用され得、その場合には読み出しリクエストは、ベース・バージョンを用いるのではなくて、現在のビューの下にあって且つこの特定のファイルのビューを含む一番上のビューからファイル・コピーを読む。）或いは、ブロック８２０のテストが肯定結果を持ったならば、読み出しは、現在のファイル・システム・ビュー（すなわち、リクエストをしているユーザ及びアプリケーションのためのファイル・システム・ビュー）のファイルのコピーを用いてブロック８４０で実行される。その後、処理は図８から出る。

【００４３】

図９は、特定のファイルについて受信されたファイル削除リクエストのための処理を示す。リクエストをしているユーザ及びアプリケーションが判定される（ブロック９００）。リクエストをしているユーザ及びアプリケーションに対応するファイル・システム・ビューにおいてコピーが作成されているか否かを判定するために、このファイルについてのメタデータが調べられる（ブロック９１０）。

【００４４】

ブロック９２０のテストは、そのメタデータに従って、リクエストをしているユーザ及びアプリケーションのために現在のビューの中にこのファイルのコピーが既に存在するか否かを調べる。もし否であれば、ファイル削除動作は実行されず、処理は図９から出る。（所望ならば、リクエストが実行されないことを示すメッセージが作られ得る。）さもなければ、ブロック９３０はそのファイルのコピーを現在のファイル・システム・ビューから除去する。その後、処理は図９から出る。

【００４５】

ファイルから個々のデータ・アイテムを削除する処理リクエストは、図８に示されている書き込み動作と同様に実行され得る。すなわち、そのファイルのコピーがリクエストをしているユーザ及びアプリケーションのためのファイル・システム・ビューにまだ存在しなければ、そのビューのためにコピーが作られ、その後にそのファイル・コピーを用いてデータの削除が実行される。さもなければ、ファイル・システム・ビューに既に存在するコピーに対して削除が行われる。（もし現在のビューがそのファイルのコピーを持っていないけれども、下にあるスタックされているビューにコピーが存在するならば、現在のビューのためにコピーを作成するときにそのスタックされているコピーが好ましくは使用される。）

【００４６】

ファイルを開き及び閉じるリクエストは、“読み出す”の代わりに“開く”又は“閉じる”を夫々用いて図８のそれに類似する論理を用いて処理され得る（これらの詳細を示す別のフローチャートは、本発明を充分に理解するために必要であるとは思われない）。具

体的には、現在のファイル・システム・ビューにコピーが存在するか否かを判定するためにメタデータが好ましくは調べられる。もし否であれば、ベース・バージョン（又は、該当する場合には、下にあるスタックされているコピー）が、リクエストされている動作により好ましくは開かれ又は閉じられ、さもなければ、現在のファイル・システム・ビューの中のコピーが開かれ又は閉じられる。

【００４７】

ファイル・システム・ビューをアプリケーションごとに、そのアプリケーションをどのユーザが実行しているかを顧慮せずに確立するアプローチを用いるときには、図７－９に示されている論理は適宜改変され得る。例えば、ブロック７００は、リクエストをしているアプリケーションの識別子を得るように改変され得、ブロック７１０はこのアプリケーションに関連付けられたメタデータを調べるように改変され得る。好ましくは、別々のＡＰＩ呼出しがこのアプローチについて提供され、これらの別々のＡＰＩ呼出しをサポートするために、図７－９に示されている例から改変された論理が使用される。

【００４８】

本書に開示されている技術を用いて、ユーザは、自分が使用するアプリケーションをかなり管理することができるようになる。特定のユーザによってファイル・システムになされた変更は、全て、そのユーザによって作成されたファイル・システム・ビューを除去することによって除去され得る。同様に、特定のアプリケーションのどのユーザによってなされた変更も、全て、そのアプリケーションについて作成されたファイル・システム・ビューの各々を除去することによって除去され得る。或いは、特定のアプリケーションを実行する一人のユーザによりなされた変更は、全て、そのユーザ及びそのアプリケーションについて作成されたファイル・システム・ビューを除去することによって除去され得る。ユーザ間でアプリケーション・アクセスを制御するために、好ましい実施態様は特定のファイル・システム・ビューへのアクセスを許可し或いは拒否する。同様に、アプリケーション間での相互作用を制御するために、許可は、好ましくは、アプリケーションにより使用されるファイル・システム・ビューに関して指定される。（従来技術では、ユーザは、どのアプリケーションがどのファイルを使用しているかを、従ってアプリケーション同士がどの様に相互作用しているかを判定し得ないことがある。）また、１つ以上のビューのファイル・コンテンツが損傷したとき、コンピュータ・システムの他の部分は、損傷したビューによる感染無しで作動し続けることができ、従って一種の保証付きシステム可用性を提供する。

【００４９】

単一の計算システムへの複数の異種オペレーティング・システムのインストールを含む、特定の計算システムへの２つ以上のオペレーティング・システムのインストールをサポートする本発明の１つ以上の実施態様が提供され得る。各オペレーティング・システムは、好ましくは、本書に開示されている技術を用いて、それ自身のファイル・システム・ビューを備える。（例えば、図４の参照番号４２１及び４２２を参照。そこには、ＧＵＩ４００において夫々“Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ　１（オペレーティング・システム１）”と“Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ　２（オペレーティング・システム２）”とが表されている。）ユーザは、それらのインストールされているオペレーティング・システムのうちの特定の１つの下で実行することを選択することを許され得る。各オペレーティング・システムのファイルは別々のファイル・システム・ビューに格納されるので、それらの間での衝突及び非両立性は回避される。所望の場合、オペレーティング・システムのファイル・システム・ビュー間で他のビューのファイルにアクセスするための許可が可能にされ得る。

【００５０】

当業者に認められるであろうように、本発明の選択された構成要素は、方法、システム、及び、コンピュータ可読プログラム・コードを含むコンピュータ・プログラム製品のうちの少なくとも１つとして提供され得る。従って、本発明は完全にハードウェア／ファームウェア実施態様の形をとることができる。代わりに、ソフトウェア及びハードウェア／

ファームウェア・アスペクトを組み合わせた実施態様が使用され得る。或いは、本発明の構成要素はソフトウェア実施態様において提供され得る。

【００５１】

更に、本発明の構成要素は、コンピュータ又は任意の命令実行システムにより、又はこれらと関連して、使用されるプログラム・コードを提供するコンピュータ使用可能な又はコンピュータ可読の媒体からアクセスし得るコンピュータ・プログラム製品の形をとることができる。この明細書の目的上、コンピュータ使用可能な又はコンピュータ可読の媒体は、命令実行システム、装置、又はデバイスにより、又はこれらと関連して、使用されるプログラムを包含し、記憶し、伝達し、伝播させ、或いは運ぶことのできる任意の装置であり得る。

【００５２】

媒体は、電子、磁気、光学、電磁、赤外線、又は半導体システム（或いは装置又はデバイス）又は伝播媒体であり得る。コンピュータ可読媒体の例は、半導体又は固体メモリ、磁気テープ、取り外し可能なコンピュータ・ディスケット、ランダム・アクセス・メモリ（“ＲＡＭ”）、読み出し専用メモリ（“ＲＯＭ”）、リジッド磁気ディスク、及び光ディスクを含む。光ディスクの現在の例は、コンパクト・ディスク読み出し専用メモリ（“ＣＤ－ＲＯＭ”）、読み書きコンパクト・ディスク（“ＣＤ－Ｒ／Ｗ”）及びＤＶＤを含む。

【００５３】

ここで図１０を参照すると、プログラム・コードを格納し又は実行し或いはその両方を行うのに適するデータ処理システム１０００は、システム・バス１０１４を通してメモリ・エレメントに直接に又は間接的に結合された少なくとも１つのプロセッサ１０１２を含む。メモリ・エレメントは、プログラム・コードの実際の実行の間に使用されるローカル・メモリ１０２８と、大容量記憶装置１０３０と、実行中に大容量記憶装置からコードを検索しなければならない回数を減らすために少なくとも或る程度のプログラム・コードの一時記憶域を提供するキャッシュ・メモリ（図示されていない）とを含むことができる。

【００５４】

（キーボード１０１８、表示装置１０２４、ポインティング・デバイス１０２０、他のインターフェース・デバイス１０２２、などを含むが、これらに限定はされない）入出力（Ｉ／Ｏ）デバイスは、直接に、或いは介在するＩ／Ｏコントローラ又はアダプタ（１０１６，１０２６）を通して、システムに結合され得る。

【００５５】

オプションとして、データ処理システムが、介在する構内ネットワーク又は公衆ネットワークを通して他のデータ処理システム又はリモート・プリンタ若しくは記憶装置に結合され得るように、ネットワーク・アダプタ（図１０には示されていない）をシステムに結合させることができる。モデム、ケーブル・モデム・アタッチメント、無線アダプタ、及びイーサネット（登録商標）・カードは、現在入手し得るタイプのネットワーク・アダプタの本の幾つかである。

【００５６】

本発明の好ましい実施態様が記載されたけれども、当業者は、基本発明思想を知ったならば、それらの実施態様の追加的な変更及び改変に想到し得る。従って、添付されている請求項は、好ましい実施態様と、本発明の範囲に含まれる全てのその様な変更及び改変を含むと解されなければならない。更に、請求項において「１つの（“ａ”又は“ａｎ”）」という言葉は本発明の実施態様をその様にして導入された単数のエレメントに限定することを意図していないことが理解されるべきである。

【図面の簡単な説明】

【００５７】

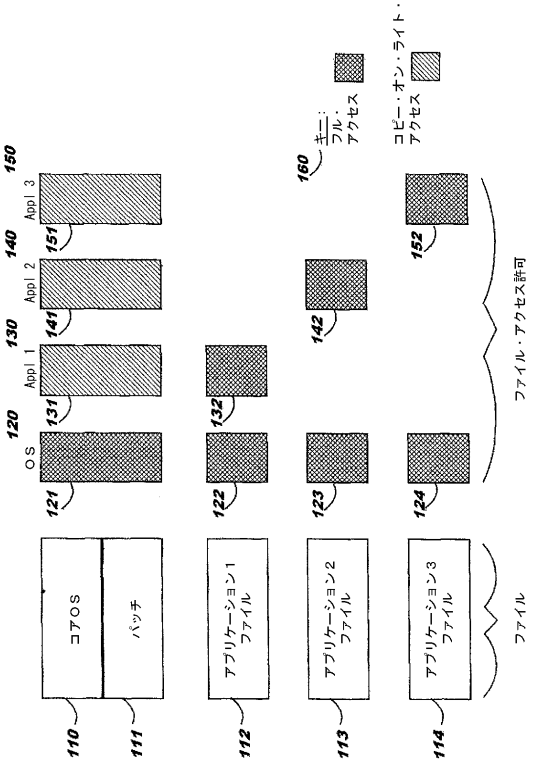
【図１】サンプルのファイル・システム使用シナリオを示す。

【図２】サンプルのファイル・システム使用シナリオを示す。

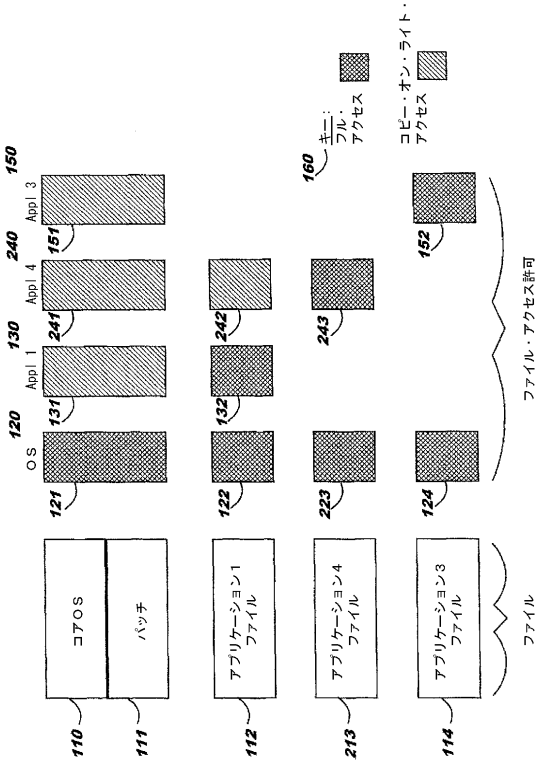
【図３】アプリケーション・ファイル・システム・ビューのスタッキングを示す。

【図４】サンプルのグラフィカル・ユーザ・インターフェース（“ＧＵＩ”）表示を示し、これで、ファイル・システム・ビューにアクセスするための許可が与えられ得る。
【図５】ユーザ毎の、特定アプリケーション向けのファイル・システム・ビューにおけるファイル・コピーの使用を示す。
【図６】図５からのファイル・コピーと種々のファイル・システム・ビューにおけるその使用とに関する情報を格納するためにメタデータを組み立てることのできる１つの方法を示す。
【図７】本発明の１つ以上の実施態様を実装するときに使用され得る論理を描いたフローチャートを提供する。
【図８】本発明の１つ以上の実施態様を実装するときに使用され得る論理を描いたフローチャートを提供する。
【図９】本発明の１つ以上の実施態様を実装するときに使用され得る論理を描いたフローチャートを提供する。
【図１０】プログラム・コードを格納し且つ（又は）実行するために適するデータ処理システムを描いている。

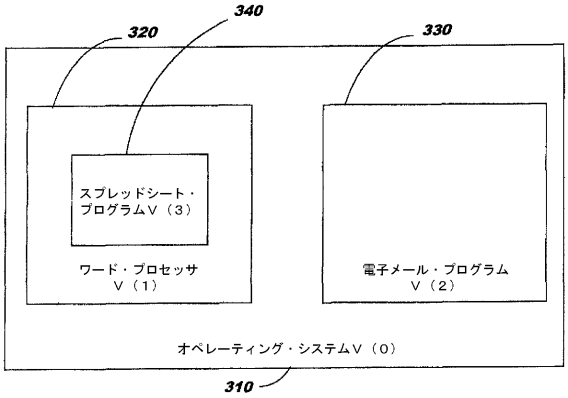
【図１】



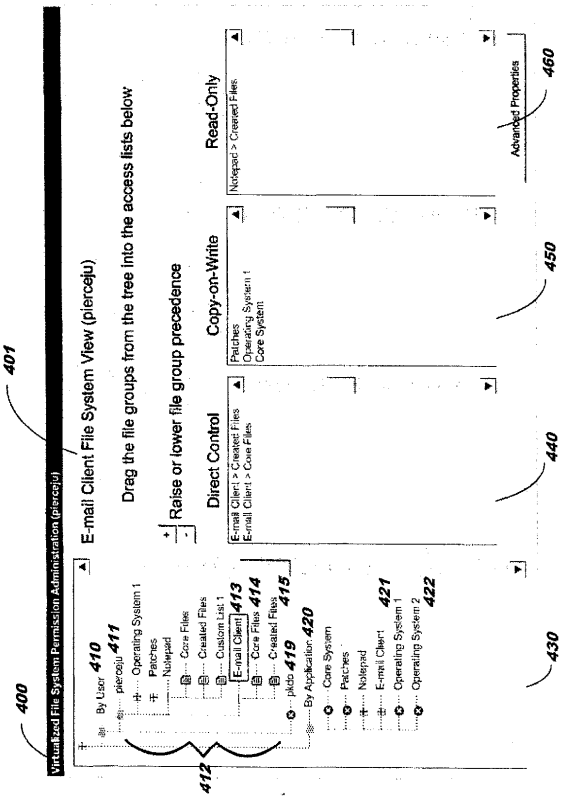
【図２】



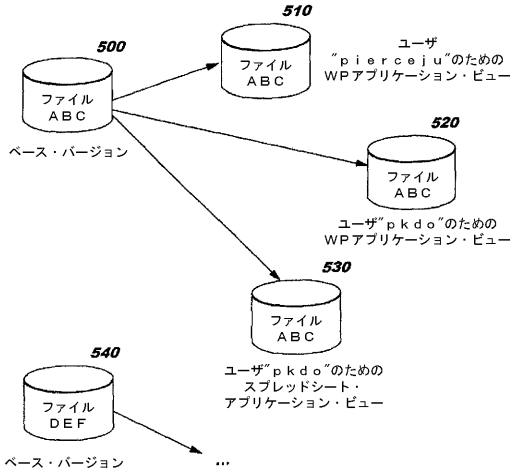
【図３】



【図４】



【図５】



【図６】

600

610

ファイルABC:

ビューの所有者：	場所：	許可：
ベース pierceju:WP pkdo:WP pkdo:spr.sh. ...	00112233 00113355 00114488 00115678	コピー・オン・ライト

630 640 650

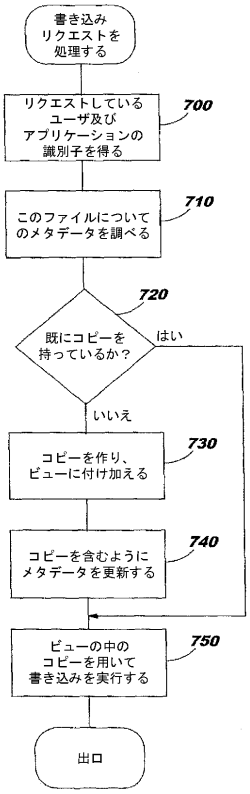
660 670

620

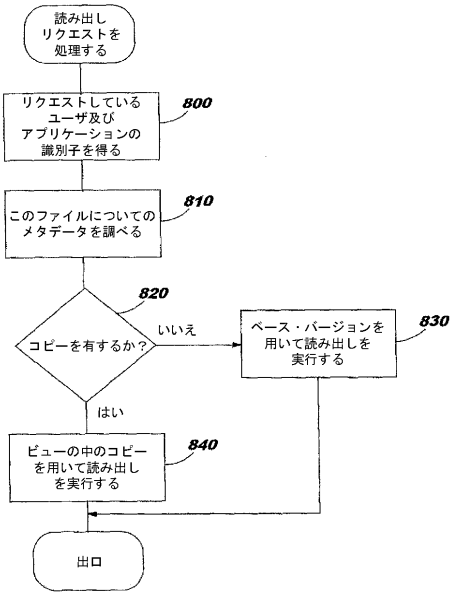
ファイルDEF:

バージョンの所有者：	場所：	許可：
...	...	

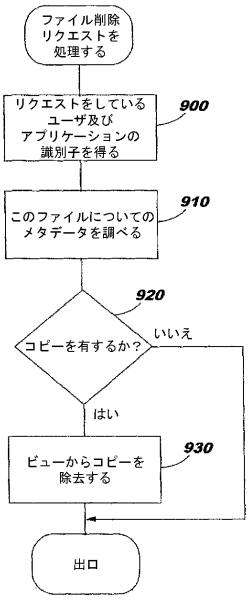
【図７】


書き込み
リクエストを
処理する
リクエストしている
ユーザ及び
アプリケーションの
識別子を得る
700
このファイルについて
のメタデータを調べる
710
720
既にコピーを
持っているか？
はい
いいえ
コピーを作り、
ビューに付け加える
730
コピーを含むように
メタデータを更新する
740
ビューの中の
コピーを用いて
書き込みを実行する
750
出口

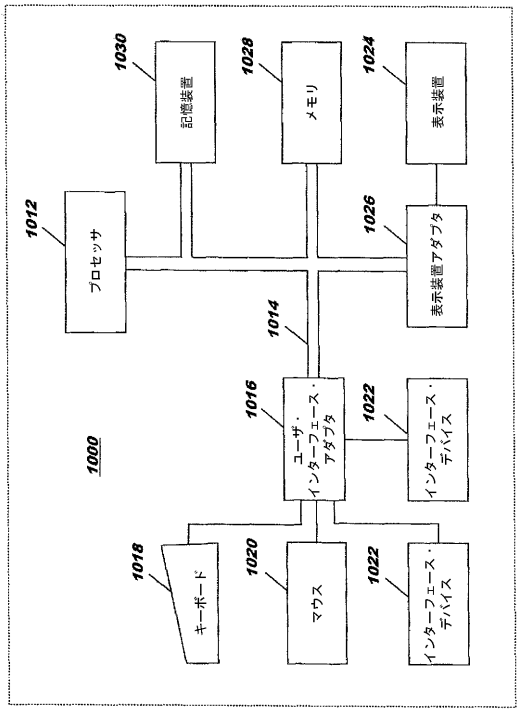
【図８】


読み出し
リクエストを
処理する
リクエストしている
ユーザ及び
アプリケーションの
識別子を得る
800
このファイルについての
メタデータを調べる
810
820
コピーを有するか？
いいえ
ベース・バージョンを
用いて読み出しを
実行する
830
はい
ビューの中のコピー
を用いて読み出し
を実行する
840
出口

【図９】


ファイル削除
リクエストを
処理する
リクエストをしている
ユーザ及び
アプリケーションの
識別子を得る
900
このファイルについての
メタデータを調べる
910
920
コピーを有するか？
いいえ
はい
ビューからコピーを
除去する
930
出口

【図１０】


1000
1012
プロセッサ
1014
1016
ユーザ・
インターフェース・
アダプタ
1018
キーボード
1020
マウス
1022
インターフェース・
デバイス
1022
インターフェース・
デバイス
1024
表示装置
1026
表示装置アダプタ
1028
メモリ
1030
記憶装置

フロントページの続き

(74)代理人 100086243
弁理士 坂口 博
(72)発明者 ピアース、ジャスティン、モンロー
アメリカ合衆国２７５１９ ノースキャロライナ州 ケリー ケリー・グレン・ブールバード ５５０８
(72)発明者 ドゥー、フック、キ
アメリカ合衆国２７５６０ ノースキャロライナ州 モリスヴィル ウェストン・エステーツ・ウェイ ３１４

審査官 市川 武宜

(56)参考文献 米国特許出願公開第２００２／００９５４７９（ＵＳ，Ａ１）
米国特許出願公開第２００３／０１８７８２２（ＵＳ，Ａ１）
米国特許第０６０２６４０２（ＵＳ，Ａ）
米国特許第０５７０６５１０（ＵＳ，Ａ）
国際公開第２００３／０９００７４（WO，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 21/24
G06F 12/00