(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-526257
(P2004-526257A)
(43)公表日　平成16年8月26日(2004.8.26)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 9/445	G06F	9/06　610A	5B076

審査請求 有　予備審査請求 有　（全 63 頁）

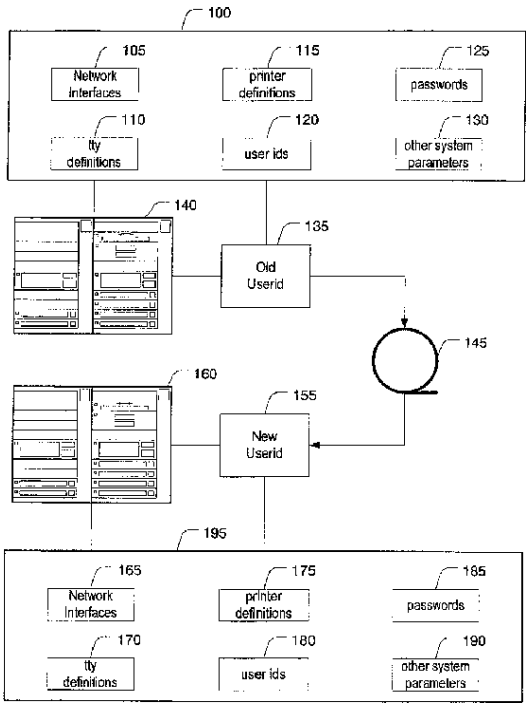
(21)出願番号	特願2002-580166(P2002-580166)
(86)(22)出願日	平成14年3月25日(2002.3.25)
(85)翻訳文提出日	平成15年10月3日(2003.10.3)
(86)国際出願番号	PCT/GB2002/001415
(87)国際公開番号	WO2002/082266
(87)国際公開日	平成14年10月17日(2002.10.17)
(31)優先権主張番号	09/826,608
(32)優先日	平成13年4月5日(2001.4.5)
(33)優先権主張国	米国(US)

(71)出願人　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　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏
(74)代理人　100108501
弁理士　上野　剛史

最終頁に続く

(54)【発明の名称】コンピュータ・システムにおいてユーザ環境データを復元・復旧するシステムと方法

(57)【要約】
【課題】ワークステーションに関するデータを取り込み、かつ物理的に分離された媒体に該データを複製する手段を提供することにより、あるワークステーションから別のワークステーションへの移動を円滑にするとともに、ユーザが新ワークステーションを構成する際に経験するダウン時間を低減する。
【解決手段】データ収集プログラムはユーザのワークステーションからデータを収集し、ユーザ設定とアプリケーション・プログラム・データを含むユーザ環境データを取得する。このユーザ環境データは着脱可能な不揮発性記憶媒体１４５に格納し複製処理に備える。格納したユーザ環境データは、複製プロセスによって処理し旧ワークステーションから新ワークステーションにユーザ環境データを複製するのに用いたり、破滅的なシステム故障からの復旧の備えに用いたりする。本発明では、既存のバックアップ・ソフトウェアが取得・復元しなかった様々なユーザ環境設定を取得・復元している。たとえば、ホスト名、ＩＰアドレスなどのユーザ固有の情報に加え、ライセンシング情報やアプリケーション・パーソナ



【特許請求の範囲】
【請求項１】
第１のコンピュータ・システムにおいてユーザ環境を複製する方法であって、
前記第１のコンピュータ・システムにおいて自動化データ収集ツールを実行して前記第１のコンピュータ・システムからユーザ環境データを収集するステップと、
着脱可能な不揮発性媒体に前記ユーザ環境データを格納するステップと
を備えた
方法。
【請求項２】
前記収集するステップが、
前記ユーザ環境データに含めるべき属性を特定するステップ
を備えている、
請求項１に記載の方法。
【請求項３】
前記方法がさらに、
前記第１のコンピュータ・システムによって格納された前記ユーザ環境データを第２のコンピュータ・システムに復元するステップ
を備えた、
請求項１または２に記載の方法。
【請求項４】
前記第１のコンピュータ・システムがＵＮＩＸ(R)オペレーティング・システムを備えている、
請求項１～３のうちの１項に記載の方法。
【請求項５】
前記収集ステップを複数のユーザについて実行し、
前記複数のユーザの各々が前記第１のコンピュータ・システムに少なくとも１つのアカウントを持っている、
請求項１～４のうちの１項に記載の方法。
【請求項６】
前記ユーザ環境データはプリンタ定義、ｔｔｙ定義、ネットワーク・インタフェース、ユーザ・パスワード、およびライセンス情報のうちの少なくとも１つを含んでいる、
請求項１～５のうちの１項に記載の方法。
【請求項７】
前記方法がさらに、
前記着脱可能な不揮発性媒体を前記第１のコンピュータ・システムから第２のコンピュータ・システムに輸送するステップと、
前記着脱可能な不揮発性媒体を該媒体を読み取り可能な装置にロードするステップと、
前記ユーザ環境データを前記着脱可能な不揮発性媒体から前記第２のコンピュータ・システムに復元するステップと
を備えている、
請求項１～６のうちの１項に記載の方法。
【請求項８】
少なくとも１つのプロセッサと、
前記プロセッサによって操作可能なオペレーティング・システムと、
前記プロセッサによってアクセス可能なメモリと、
前記プロセッサによってアクセス可能な着脱可能な不揮発性記憶装置と、
第１のコンピュータ・システムにおいてユーザ環境データを複製するユーザ環境複製ツールと
を備え、
前記ツールが、

前記第１のコンピュータ・システムからユーザ環境データを収集する手段であって、前記収集はコンピュータ・プログラムが実行する、手段と、
前記ユーザ環境データを着脱可能な不揮発性媒体に格納する手段と
を備えている
情報処理システム。
【請求項９】
前記収集する手段が、
前記ユーザ環境データに含めるべき属性を特定する手段
を備えている、
請求項８に記載の情報処理システム。
【請求項１０】
さらに、
前記第１のコンピュータ・システムによって格納した前記ユーザ環境データを第２のコンピュータ・システムに復元する手段
を備えた、
請求項８または９に記載の情報処理システム。
【請求項１１】
前記第１のコンピュータ・システムがＵＮＩＸ(R)オペレーティング・システムを備えている、
請求項８～１０のうちの１項に記載の情報処理システム。
【請求項１２】
前記収集する手段は複数のユーザについて実行し、前記複数のユーザの各々は前記第１のコンピュータ・システムに少なくとも１つのアカウントを持っている、
請求項８～１０のうちの１項に記載の情報処理システム。
【請求項１３】
前記ユーザ環境データはプリンタ定義、ｔｔｙ定義、ネットワーク・インタフェース、ユーザ・パスワード、およびライセンス情報のうちの少なくとも１つを含んでいる、
請求項８～１０のうちの１項に記載の情報処理システム。
【請求項１４】
第１のコンピュータ・システムにおいてユーザ環境を複製するようにプログラムされ、コンピュータ操作可能な媒体に格納されたコンピュータ・プログラムであって、
前記第１のコンピュータ・システムからユーザ環境データを収集するプログラム・コードであって、前記収集をコンピュータ・プログラムによって実行する、プログラム・コードと、
前記ユーザ環境データを着脱可能な不揮発性媒体に格納するプログラム・コードと
を備えた
コンピュータ・プログラム。
【請求項１５】
前記収集するプログラム・コードが、
前記ユーザ環境データに含めるべき属性を特定する手段
を備えている、
請求項１４に記載のコンピュータ・プログラム。
【請求項１６】
さらに、
前記第１のコンピュータ・システムによって格納された前記ユーザ環境データを第２のコンピュータ・システムに復元する手段
を備えた、
請求項１４または１５に記載のコンピュータ・プログラム。
【請求項１７】
前記収集する手段は複数のユーザについて実行し、前記複数のユーザの各々は前記第１の

コンピュータ・システムに少なくとも１つのアカウントを持っている、
請求項１４～１６のうちの１項に記載のコンピュータ・プログラム。
【請求項１８】
前記ユーザ環境データはプリンタ定義、ｔｔｙ定義、ネットワーク・インタフェース、ユーザ・パスワード、およびライセンス情報のうちの少なくとも１つを含んでいる、
請求項１４～１６のうちの１項に記載のコンピュータ・プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は情報処理技術に関する。特に、本発明はコンピュータ・システムにおいてユーザ環境データを復元・復旧するシステムと方法に関する。
【背景技術】
【０００２】
ＵＮＩＸ(R)オペレーティング・システムは１９６９年に発明された対話型のタイム・シェアリング・オペレーティング・システムである。ＵＮＩＸ(R)オペレーティング・システムは複数のユーザ用の直列接続およびネットワーク接続された端末をサポートするマルチユーザ・オペレーティング・システムである。ＵＮＩＸ(R)は同じシステムを複数のユーザが同時に使用することのできるマルチタスク・オペレーティング・システムである。ＵＮＩＸ(R)オペレーティング・システムはカーネル、シェル、およびユーティリティを備えている。ＵＮＩＸ(R)は移植可能なオペレーティング・システムであり、アセンブラで書く必要があるのはカーネルだけである。また、広範囲のサポート・ツール、たとえば開発、デバッガー、コンパイラなどをサポートしている。
【０００３】
マルチユーザ・オペレーティング・システムとしてのＵＮＩＸ(R)に使用すれば、複数の人々が同じコンピュータ・システムを同時に共用するすることが可能になる。これを実行するのにＵＮＩＸ(R)では、コンピュータの中央処理装置（すなわち「ＣＰＵ」）を時間間隔にタイムスライスしている。各ユーザはシステムが要求された命令を実行する所定量の時間を取得する。あるユーザに割り当てられた時間が満了すると、オペレーティング・システムはＣＰＵに割込みをかけることにより割り込み、当該ユーザのプログラム状態（プログラム・コードとデータ）を保管し、次のユーザのプログラム状態を復元し、当該次のユーザのプログラムを（当該次のユーザの時間だけ）実行することを開始する。このプロセスは際限なく続きシステムを使用しているすべてのユーザに行き渡るまで反復する。そして、最後のユーザのタイム・スライスが満了したら、最初のユーザに制御を戻し別のサイクルを開始する。
【０００４】
ＵＮＩＸ(R)オペレーティング・システムはマルチユーザ・オペレーティング・システムであるとともにマルチタスク・オペレーティング・システムでもある。その名称が示唆しているように、ＵＮＩＸ(R)のマルチユーザの側面によれば、同じシステムを複数のユーザが同時に使用することができる。マルチタスク・オペレーティング・システムとしてのＵＮＩＸ(R)によれば、複数のプログラム（または実行スレッドと呼ばれるプログラムの部分）を同時に実行することができる。このオペレーティング・システムはプログラムまたはスレッドの各々を実行するために、様々なプログラム（または実行スレッド）の間でプロセッサを切り換える。ＩＢＭのＯＳ／２（商標）やマイクロソフトのＷｉｎｄｏｗｓ(R)９５／９８／ＮＴがシングルユーザ・マルチタスク・オペレーティング・システムの例であるのに対して、ＵＮＩＸ(R)はマルチユーザ・マルチタスク・オペレーティング・システムの例である。マルチタスク・オペレーティング・システムはフォアグラウンド・タスクとバックグラウンド・タスクの双方をサポートしている。フォアグラウンド・タスクとは入力装置と画面を用いてユーザと直接にインタフェースするタスクのことである。バックグラウンド・タスクはバックグラウンドで実行され、入力装置（たとえばキーボード、マウス、タッチパッドなど）にアクセスせず、画面にもアクセスしない。バックグラ

ウンド・タスクには後刻の実行用にスプールすることのできる印刷のようなオペレーションがある。

【０００５】

ＵＮＩＸ(R)オペレーティング・システムはシステムで実行されているプログラムをすべて追跡しており、必要に応じてディスク、メモリ、プリンタ・キューなどのリソースを割り当てる。ＵＮＩＸ(R)は理想的には各プログラムが適切に実行されるのに相当な分量のリソースを受け取れるようにリソースを割り当てる。ＵＮＩＸ(R)は２つの方法、すなわちスケジューリング優先順位とシステム・セマフォアを用いてリソースを配分する。各プログラムには優先順位が割り当てられている。優先順位の高いタスクは（ディスクに対する読み書きと同様に）より頻繁に実行される。ユーザ・プログラムはそのアクティビティと利用可能なシステム・リソースとに応じて自分の優先順位を上方または下方に動的に調整してもらう。システム・セマフォアはオペレーティング・システムがシステム・リソースを管理するのに使用する。プログラムはオペレーティング・システムにシステム・コールを行ってセマフォアを取得することにより、リソースを割り当ててもらうことができる。リソースが必要でなくなったら、セマフォアをオペレーティング・システムに返す。これにより、オペレーティング・システムはそのセマフォアを別のプログラムに割り当てることができる。

【０００６】

ディスク駆動装置とプリンタはその性質上シリアルである。このことは一度に１つの要求しか実行できないということを意味している。複数のユーザがこれらのリソースを一度に利用できるようにするために、オペレーティング・システムはそれらをキューを使って管理している。各シリアル装置はキューに関連付けられている。あるプログラムが装置（たとえばディスク駆動装置）にアクセスする必要がある場合、そのプログラムは当該装置に関連付けられたキューに要求を送る。ＵＮＩＸ(R)オペレーティング・システムは（デーモンと呼ばれる）バックグラウンド・タスクを実行しており、そのデーモンが上記キューをモニタするとともにそのキュー用の要求を処理する。上記要求はデーモンが処理し、その結果はユーザ・プログラムに返される。

【０００７】

マルチタスク・システムはプロセスを管理する１組のユーティリティを備えている。ＵＮＩＸ(R)では、これらはｐｓ（list processes: プロセスを列挙）、ｋｉｌｌ（kill a process: プロセスを削除）、およびコマンド・ライン末の＆（プロセスをバックグラウンドで実行）である。ＵＮＩＸ(R)では、すべてのユーザ・プログラムとアプリケーション・ソフトウェアはシステム・コール・インタフェースを用いてシステム・リソース、たとえばディスク、プリンタ、メモリなどにアクセスする。ＵＮＩＸ（R)におけるシステム・コール・インタフェースは１組のシステム・コール（Ｃ言語の関数）を備えている。システム・コール・インタフェースの目的は低レベルのハードウェア・アクセスをすべてＵＮＩＸ(R)オペレーティング・システムの管理下に置き、ユーザ作成プログラムに管理させないことにより、システムの保全性を実現することである。これにより、プログラムによってシステムが破壊されるのを防止することができる。

【０００８】

システム・コールを受信すると、オペレーティング・システムはそのアクセス許可を検査し、要求元プログラムの代わりに要求を実行し、その結果を要求元プログラムに返す。要求が無効すなわちユーザにアクセス許可がない場合には、オペレーティング・システムはその要求を実行せず、要求元プログラムにエラーを返す。ＵＮＩＸ(R)の大部分がＣ言語で書かれているから、システム・コールは１組のＣ言語関数として理解することができる。典型的なシステム・コールには次のものがある。すなわち、ディスクから読み取る＿ｒｅａｄ、ディスクに書き込む＿ｗｒｉｔｅ、端末からキャラクタを読み取る＿ｇｅｔｃｈ、端末にキャラクタを書き込む＿ｐｕｔｃｈ、装置パラメータを制御し設定する＿ｉｏｃｔｌである。

【０００９】

その名称が示唆しているように、カーネルはＵＮＩＸ(R)オペレーティング・システムの核に存在する。システムを開始するときにはいつも、このカーネルをロードする（システム「ブート」とも呼ばれている）。カーネルはシステムのリソースを管理し、それらをユーザに一貫性のあるシステムとして提示する。ＵＮＩＸ(R)システムを使用するために、ユーザはカーネルについて（もし理解するとしても）あまり理解している必要はない。カーネルはＵＮＩＸ(R)環境で必要な様々な機能を備えている。カーネルはシステムのメモリを管理し、それを各プロセスに割り当てる。カーネルがプログラムの状態を保管・復元し、あるプログラムから次のプログラムに切り替える（ディスパッチと呼ばれる）には時間がかかる。この動作は迅速に行う必要がある。なぜなら、プログラム間の切り替えに要する時間の分だけユーザのプログラムを実際に実行するのに使われる時間が減少するからである。ユーザ・プログラム間の切り替えのようなタスクをカーネルが実行している「システム状態」における時間消費はシステム・オーバーヘッドであるから、できるかぎり少なくする必要がある。典型的なＵＮＩＸ(R)システムでは、システム・オーバーヘッドは全時間の１０％未満であろう。

【００１０】

また、カーネルは各ユーザの仕事が効率的に実行されるように、中央処理装置（すなわち「ＣＰＵ」）が実行する仕事のスケジューリングも行う。カーネルはシステムのある部分から別の部分へデータを転送する。メインメモリ上のユーザ・プログラム間の切り替えを行うのもカーネルである。メイン・システム・メモリはオペレーティング・システム用の部分とユーザ・プログラム用の部分とに分割されている。カーネル用のメモリ空間はユーザ・プログラムから分離されている。あるプログラムを実行するのに十分なメインメモリが存在しない場合、別のプログラムをディスクに書き出し（すなわちスワップし）て当該プログラムを実行するのに十分なメインメモリを解放する。カーネルは様々な要因に基づいてどのプログラムがディスクにスワップアウトするのに一番よい候補であるかを決める。システムで同時に実行されているプログラムが多すぎると、システムは過負荷になり、オペレーティング・システムはファイルをディスクにスワップアウトするのに多くの時間を使うから、プログラムを実行する時間が少なくなる。この結果、性能が劣化する。また、カーネルは「シェル」から命令を受け取り、それらの実行も行う。さらに、カーネルはシステムで即使用可能なアクセス権の付与も行う。アクセス権はシステム中の各ファイルおよび各ディレクトリごとに存在し、他のユーザが所定のファイルまたはディレクトリに対してアクセスし、実行し、または変更しうるか否かを決めるものである。

【００１１】

ファイルの取り扱いについては、ＵＮＩＸ(R)はファイルを編成しそれらを保守するのに階層ディレクトリ構造を使用している。アクセス権はファイルとディレクトリに対応している。上述したように、ＵＮＩＸ(R)オペレーティング・システムでは、ファイルをディレクトリに編成する。ディレクトリは階層型ツリー構造に格納されている。このツリーの最上部にはルート・ディレクトリがある。ルート・ディレクトリはスラッシュ（／）キャラクタで表わす。ルート・ディレクトリは少なくとも１つのディレクトリを備えている。これらのディレクトリはさらに、ユーザ・ファイルおよび他のシステム・ファイルを格納したディレクトリを備えている。多くのＵＮＩＸ(R)中に存在する標準的なディレクトリを次に示す。

／ｂｉｎ　このディレクトリは基本システム・コマンドを格納している。

／ｅｔｃ　このディレクトリはシステム構成ファイルおよびシステムを管理するのに使用するプログラムを格納している。

／ｌｉｂ　このディレクトリはシステム・ライブラリを格納している。

／ｔｅｍｐ　このディレクトリはテンポラリ（一時）ファイルを格納するのに使用する。

／ｕｓｒ／ｂｉｎ　このディレクトリは／ｂｉｎに格納されていないコマンドを格納している。

／ｕｓｒ／ｍａｎ　このディレクトリはマニュアル用のページを格納している。

／ｕｓｒ／ｌｏｃａｌ　このディレクトリは元のシステムには含まれておらずシステム管

理者（ｓｙｓａｄｍｉｎ）がインストールしたローカル・プログラムを格納している。特に、／ｕｓｒ／ｌｏｃａｌ／ｂｉｎはローカル・コマンド・ファイル（バイナリー）を格納しており、／ｕｓｒ／ｌｏｃａｌ／ｍａｎはローカル・マニュアル・ページを格納している。
／ｈｏｍｅ　実際のディレクトリの場所はシステムによって変化するが、システム中のどこかを、すべてのユーザのホーム（ｈｏｍｅ）ディレクトリを配置する場所にすることができる。
【００１２】
ＵＮＩＸ(R)オペレーティング・システムが情報を格納するのに使う基礎構造はファイルである。ファイルは一連のバイトである。ＵＮＩＸ(R)は各ファイルに一意の識別番号を割り当てることにより、ファイルを内部で追跡している。これらの番号はｉノード番号と呼ばれるが、ＵＮＩＸ(R)のカーネル自体の内部でのみ使用されている。ＵＮＩＸ(R)はファイルを指すのにｉノード番号を使用しているが、ユーザがユーザ指定の名前で各ファイルを識別することを可能にしている。ファイル名は一連のキャラクタでよく、最大１４キャラクタ長にすることができる。
【００１３】
ＵＮＩＸ(R)のファイル・システムには次に示す３種類のファイルがある。（１）通常ファイル。これには実行可能ファイル、テキスト、または入力として使用される、もしくは何らかの操作の出力として生成される種類のデータがある。（２）ディレクトリ・ファイル。これには上で概説したディレクトリ中のファイルのリストがある。（３）特別ファイル。これは入力装置／出力装置にアクセスする標準的な方法を提供するものである。
【００１４】
内部的には、ディレクトリは１つのファイルであり、通常ファイルの名前、他のディレクトリ、および通常ファイルの対応するｉノード番号を格納している。ｉノード番号を用いることにより、ＵＮＩＸ(R)は別の内部テーブルを調べてファイルを格納すべき場所を決めるとともに当該ファイルをユーザに対してアクセス可能にすることができる。ＵＮＩＸ(R)のディレクトリはそれ自体が名前を備えている。この名前の例は上述したが、最大１４キャラクタ長にすることができる。
【００１５】
ＵＮＩＸ(R)は自身が管理しているファイルに関する情報をきわめて大量に保守している。各ファイルごとに、ファイル・システムはファイル・サイズ、場所、所有者、セキュリティ、種類、作成時刻、変更時刻、およびアクセス時刻を追跡している。これらの情報はすべてファイルの作成時および使用時にファイル・システムが自動的に保守している。ＵＮＩＸ(R)のファイル・システムは大容量記憶装置たとえばディスク駆動装置やディスク・アレイなどに常駐している。ＵＮＩＸ(R)はディスクを一連のブロックに編成している。これらのブロックは普通、５１２バイト長または２０４８バイト長である。ファイルの内容は少なくとも１つのブロックに格納されている。ブロックはディスク上に広く分散している。
【００１６】
通常ファイルにはｉノード構造によってアドレスする。各ｉノードにはｉリストに含まれているインデックスによってアドレスする。ｉリストはファイル・システムのサイズに基づいて生成される。一般に、ファイル・システムが大きいほどより多くのファイルを保持しているものと考えられるから、ｉリストは大きくなる。各ｉノードには１３個の４バイト・ディスク・アドレス要素が含まれている。直接ｉノードには最大１０個のブロック・アドレスを含めることができる。ファイルがこれより大きい場合、第１レベルの間接ブロックを指す１１番目のアドレスを使用する。アドレス１２とアドレス１３はそれぞれ第２レベルの間接ブロックおよび第３レベルの間接ブロック用に使用する。この場合、直接ｉノードにおいて新たなアドレス・スロットが必要になったら、第１のデータ・ブロックより前の間接アドレッシング・チェーンを１レベルだけ増やす。
【００１７】

入力と出力（Ｉ／Ｏ）はすべてファイルに対する読み書きによって行う。なぜなら、ファイル・システムではすべての周辺装置（したがって端末でさえ）ファイルとして扱うからである。最も一般的な場合では、ファイルに対して読み書きする前にそうするという意図をファイルをオープン（open）することによりシステムに通知する必要がある。ファイルに書き込むためには、ファイルを作成（create）する必要もある。（「ｏｐｅｎ」システム・コールまたは「ｃｒｅａｔｅ」システム・コールによって）ファイルをオープンまたは作成すると、システムはそうする権利があるか否かを検査し、ユーザにそうする権利があれば、ファイル・ディスクリプタと呼ばれる非負の整数を返す。以後、このファイルについてＩ／Ｏを行うときには、ファイル名の代わりにこのファイル・ディスクリプタを用いて当該ファイルを識別する。このオープン・ファイル・ディスクリプタは当該ファイルをオープンしたユーザの「プロセス」空間に保持されているファイル・テーブル・エントリに関連付けられている。ＵＮＩＸ(R)の用語法では、「プロセス」なる用語は「実行中のプログラム」と可逆的に使用される。ファイル・テーブル・エントリにはオープン・ファイルに関する情報、たとえば当該ファイルに対するｉノード・ポインタおよび当該ファイルに対するファイル・ポインタなどが含まれている。ファイル・ポインタは当該ファイル中の読み取るべきまたは書き込むべき現在の位置を規定している。オープン・ファイルに関する情報はすべてシステムが保守する。

【００１８】

既存のＵＮＩＸ(R)システムでは、すべての入力と出力は２つのシステム・コール「ｒｅａｄ」と「ｗｒｉｔｅ」によって行う。これらのシステム・コールは同名の機能を有するプログラムからアクセスする。両方のシステム・コールでは、第１の引数はファイル・ディスクリプタであり、第２の引数はデータのソース（送信元）またはデスティネーション（送信先）として機能するバッファへのポインタであり、第３の引数は転送するバイト数である。「ｒｅａｄ」システム・コールまたは「ｗｒｉｔｅ」システム・コールの各々は転送するバイト数をカウントする。読み取りの場合、返されるバイト数は要求したバイト数より少ない。なぜなら、実際に読み取られるバイト数は要求したバイト数より少ないからである。「０」なる戻りコードはファイルの終端（end-of-file)に到達したことを意味し、「－１」なる戻りコードはエラーが発生したことを意味している。書き込みの場合、戻りコードは実際に書き込んだバイト数である。したがって、この数値が書き込んだと考えられるバイト数と一致しない場合にはエラーが発生している。

【００１９】

ＵＮＩＸ(R)はｇｅｔｔｙと呼ばれるシステム・プロセスによって、システムに接続された各端末の入力ラインの状態をモニタしている。ｇｅｔｔｙはユーザが端末の電源を入れたのを検出すると、ログオン・プロンプトを提示する。そして、ユーザＩＤとパスワードが有効なら、ＵＮＩＸ(R)システムは当該端末にシェル・プログラム（たとえばｓｈ）を関連付ける。この結果、当該ユーザはシェル・プログラム中にに配置される。シェル・プログラムは通常、どのシェル・プログラムが実行中であるのかを示すプロンプトを提示する。ユーザはそのプロンプトに続けてコマンドをタイプ入力する。シェル・プログラムはコマンド・インタープリタとして動作し、各コマンドを取得しそれをカーネルに渡して処理させる。次いで、シェルは処理結果を画面に表示する。ユーザはシェルを用いて当該ユーザの必要に適合し個人化した環境を生成することができる。ユーザは当該ユーザの環境を制御している環境変数を変更することができる。

【００２０】

ＥＤＩＴＯＲ環境変数は他のプログラムたとえばメール・プログラムなどが使うことになるエディタを設定する。ＰＡＧＥＲ環境変数はマニュアル・ページを表示するｍａｎなどのプログラムが使うことになるページャを設定する。ＰＡＴＨ環境変数はシェルがコマンドを見つけるのに目を通すはずのディレクトリを特定する。これらのディレクトリは出現順に探索される。ＰＲＩＮＴＥＲ環境変数はｌｐｒコマンドによってすべての出力が送られるプリンタを設定する。ＳＨＥＬＬ変数はユーザが使用するデフォルトのシェルを設定する。ＴＥＲＭ変数はエディタやページャなどのプログラム用の端末の種類を設定する。

ＴＺ環境変数はユーザが位置しているタイム・ゾーンを設定する。

【００２１】

ＵＮＩＸ(R)ユーザには利用可能なシェルがいくつかある。各シェルは他のシェルとは異なる特徴と機能を有している。最も一般的なＵＮＩＸ(R)シェル・プログラムには「Ｂｏｕｒｎｅ」シェル、「Ｃ」シェル、「ＴＣ」シェル、「Ｋｏｒｎ」シェル、および「ＢＡＳＨ」シェルがある。コマンドを実行するために使用するのに加え、これらのシェルは各々、組み込みプログラミング言語を備えており、ユーザはそれを用いて自身のコマンドまたはプログラムを書くことができる。また、ユーザはコマンドをファイルに入れて（シェル・スクリプトと呼ばれる）、そのファイルをコマンドまたはプログラムのように実行することができる。シェルは次に示す２種類のコマンドを呼び出す。すなわち、シェル・プログラムが処理する内部コマンド（たとえばｓｅｔ、ｕｎｓｅｔなど）、およびプログラムとして呼び出される外部コマンド（たとえばｌｓ、ｇｒｅｐ、ｓｏｒｔ、ｐｓなど）である。

【００２２】

ＵＮＩＸ(R)オペレーティング・システムの１つの利点はユーザが自分の必要に適合するように自分の作業環境をカスタマイズできる点である。たとえば、ユーザは次に示すものを選定することができる。すなわち、デフォルトのエディタ、マニュアルのページを表示するページャ、コマンドを求めて探索されるディレクトリを特定するパス、デフォルトのプリンタ、エディタとページャが使用する端末の種類、正確な時刻を表示するタイム・ゾーン、およびシステムにログオンするときにユーザの端末に関連付けられるシェル・プログラムである。

【００２３】

現在の複雑なコンピューティング環境における１つの課題は、システムの変更またはあるシステムから別のシステムへのユーザの再配置に起因してユーザをあるシステムから別のシステムに移動させることである。コンピュータが複雑であることおよびユーザ環境を作り上げているカスタマイズの量が多いことにより、ユーザのコンピューティング環境を複製することは以前より挑戦する価値のあることになった。特に、ＵＮＩＸ(R)イメージを再作成するには、次に示す大量のシステム・パラメータを必要とする。すなわち、プリンタ定義、ｔｔｙ定義（端末定義、所定のジョブを制御する特定の端末の名前、またはシリアル・ポート定義。ＵＮＩＸ(R)ではこのような装置の名前はｔｔｙ＊という形をしている）、ネットワーク・インタフェース、ユーザＩＤ、およびパスワードである。このようなパラメータの複製にすべて失敗すると、ユーザは主要なアプリケーションを実行できなくなったり、そのようなシステムの複製に起因して重要なリソースにアクセスできなくなったりする。複製を大部分人手に頼っている現在の複製方式の課題は、ユーザおよび／またはシステム管理者が行う人手仕事自体、ならびにそのような人手仕事は間違いを生じやすいということである。（「Ａおよび／またはＢ」は「ＡおよびＢ、Ａ、またはＢ」を表わす。）

【００２４】

あるワークステーションから別のワークステーションにユーザ環境データを半自動態様で複製しうるということは分かっている。まず、自動データ収集プロセスがユーザ環境データを旧コンピュータ・システムから収集する。ユーザ環境データには、アプリケーション・プログラム・データ、ライセンス情報、ｔｔｙ設定、カスタマイズしたディレクトリ、その他ユーザがカスタマイズしたワークステーション環境変数がある。

【００２５】

ワークステーションのリストを用いて多数のワークステーションを複製することもできる。そのような場合とは、たとえば一群の個人が自分のワークステーションをアップグレードするような場合である。この場合、後刻、新ワークステーションへ複製することに備えてユーザ環境データを格納する。ユーザ環境データは着脱可能かつコンピュータ操作可能な媒体、たとえばディスケットまたは磁気テープなどに格納し、搬送、新ワークステーションへの入力、システム故障発生時の旧コンピュータへの復元での使用に備える。新ワー

クステーションへユーザ環境設定を複製するには、コンピュータ操作可能な媒体に格納されているユーザ環境データを読み取り、当該ユーザ環境データを新ワークステーションに入力する。

【発明の開示】

【発明が解決しようとする課題】

【００２６】

本発明の目的は、ワークステーションに関するデータを取り込み、かつ物理的に分離された媒体に該データを複製する手段を提供することにより、あるワークステーションから別のワークステーションへの移動を円滑にするとともに、ユーザが新ワークステーションを構成する際に経験するダウン時間を低減することである。

【課題を解決するための手段】

【００２７】

本発明は以下のように構成する。

（１）

第１のコンピュータ・システムにおいてユーザ環境を複製する方法であって、

前記第１のコンピュータ・システムにおいて自動化データ収集ツールを実行して前記第１のコンピュータ・システムからユーザ環境データを収集するステップと、

着脱可能な不揮発性媒体に前記ユーザ環境データを格納するステップと

を備えた

方法。

（２）

前記収集するステップが、

前記ユーザ環境データに含めるべき属性を特定するステップ

を備えている、

上記（１）に記載の方法。

（３）

前記方法がさらに、

前記第１のコンピュータ・システムによって格納された前記ユーザ環境データを第２のコンピュータ・システムに復元するステップ

を備えた、

上記（１）または（２）に記載の方法。

（４）

前記第１のコンピュータ・システムがＵＮＩＸ(R)オペレーティング・システムを備えている、

上記（１）～（３）のうちの１つに記載の方法。

（５）

前記収集ステップを複数のユーザについて実行し、

前記複数のユーザの各々が前記第１のコンピュータ・システムに少なくとも１つのアカウントを持っている、

上記（１）～（４）のうちの１つに記載の方法。

（６）

前記ユーザ環境データはプリンタ定義、ｔｔｙ定義、ネットワーク・インタフェース、ユーザ・パスワード、およびライセンス情報のうちの少なくとも１つを含んでいる、

上記（１）～（５）のうちの１つに記載の方法。

（７）

前記方法がさらに、

前記着脱可能な不揮発性媒体を前記第１のコンピュータ・システムから第２のコンピュータ・システムに輸送するステップと、

前記着脱可能な不揮発性媒体を該媒体を読み取り可能な装置にロードするステップと、

前記ユーザ環境データを前記着脱可能な不揮発性媒体から前記第２のコンピュータ・シス

テムに復元するステップと
を備えている、
上記（１）～（６）のうちの１つに記載の方法。
（８）
少なくとも１つのプロセッサと、
前記プロセッサによって操作可能なオペレーティング・システムと、
前記プロセッサによってアクセス可能なメモリと、
前記プロセッサによってアクセス可能な着脱可能な不揮発性記憶装置と、
第１のコンピュータ・システムにおいてユーザ環境データを複製するユーザ環境複製ツールと
を備え、
前記ツールが、
前記第１のコンピュータ・システムからユーザ環境データを収集する手段であって、前記収集はコンピュータ・プログラムが実行する、手段と、
前記ユーザ環境データを着脱可能な不揮発性媒体に格納する手段と
を備えている
情報処理システム。
（９）
前記収集する手段が、
前記ユーザ環境データに含めるべき属性を特定する手段
を備えている、
上記（８）に記載の情報処理システム。
（１０）　さらに、
前記第１のコンピュータ・システムによって格納した前記ユーザ環境データを第２のコンピュータ・システムに復元する手段
を備えた、
上記（８）または（９）に記載の情報処理システム。
（１１）
前記第１のコンピュータ・システムがＵＮＩＸ(R)オペレーティング・システムを備えている、
上記（８）～（１０）のうちの１つに記載の情報処理システム。
（１２）
前記収集する手段は複数のユーザについて実行し、前記複数のユーザの各々は前記第１のコンピュータ・システムに少なくとも１つのアカウントを持っている、
上記（８）～（１０）のうちの１つに記載の情報処理システム。
（１３）
前記ユーザ環境データはプリンタ定義、ｔｔｙ定義、ネットワーク・インタフェース、ユーザ・パスワード、およびライセンス情報のうちの少なくとも１つを含んでいる、
上記（８）～（１０）のうちの１つに記載の情報処理システム。
（１４）
第１のコンピュータ・システムにおいてユーザ環境を複製するようにプログラムされ、コンピュータ操作可能な媒体に格納されたコンピュータ・プログラムであって、
前記第１のコンピュータ・システムからユーザ環境データを収集するプログラム・コードであって、前記収集をコンピュータ・プログラムによって実行する、プログラム・コードと、
前記ユーザ環境データを着脱可能な不揮発性媒体に格納するプログラム・コードと
を備えた
コンピュータ・プログラム。
（１５）
前記収集するプログラム・コードが、

前記ユーザ環境データに含めるべき属性を特定する手段
を備えている、
上記（１４）に記載のコンピュータ・プログラム。
（１６）　さらに、
前記第１のコンピュータ・システムによって格納された前記ユーザ環境データを第２のコンピュータ・システムに復元する手段
を備えた、
上記（１４）または（１５）に記載のコンピュータ・プログラム。
（１７）
前記収集する手段は複数のユーザについて実行し、前記複数のユーザの各々は前記第１のコンピュータ・システムに少なくとも１つのアカウントを持っている、
上記（１４）～（１６）のうちの１つに記載のコンピュータ・プログラム。
（１８）
前記ユーザ環境データはプリンタ定義、ｔｔｙ定義、ネットワーク・インタフェース、ユーザ・パスワード、およびライセンス情報のうちの少なくとも１つを含んでいる、
上記（１４）～（１６）のうちの１つに記載のコンピュータ・プログラム。
【発明を実施するための最良の形態】
【００２８】
図１はあるコンピュータ・システムから別のコンピュータ・システムにユーザ環境データをコピーするとともに送る様子を示すシステム図である。ユーザの旧ユーザ環境データ１００にはネットワーク・インタフェース１０５、ｔｔｙ定義１１０、プリンタ定義１１５、ユーザＩＤ１２０、パスワード１２５、およびアプリケーション固有情報など他のシステム・パラメータ１３０がある。要するに、旧ユーザ環境データ１００はユーザが自分の仕事関連のタスクをよりよく実行しうるように行ったユーザ設定の結果としてユーザ・アカウントになされたカスタム化と変更である。この例では、ユーザは旧コンピュータ・システム１４０の旧ユーザＩＤ１３５から新コンピュータ・システム１６０に結び付けられた新ユーザＩＤ１５５に移動しつつある。旧ユーザ環境データ１００は旧コンピュータ・システム１４０から収集したら、不揮発性コンピュータ媒体１４５を用いて新コンピュータ・システム１６０に転送する。不揮発性コンピュータ媒体１４５には、ディスケット、磁気テープ、ＺＩＰディスク、ＪＡＺディスク、ＣＤ－Ｒ（再符号化可能なＣＤ－ＲＯＭ）、ハード・ディスク駆動装置、光ディスク、またはあるコンピュータ・システムから別のコンピュータ・システムに輸送することによりデータを転送しうる任意の媒体を用いることができる。
【００２９】
新コンピュータ・システム１６０で旧ユーザ環境データ１００を受け取ったら、新ユーザ環境データ１９５を作成して旧ユーザ環境データ１００を新コンピュータ・システム１６０に複製するプロセスを呼び出す。旧ユーザ環境データ１００と同様に、新ユーザ環境データ１９５にはネットワーク・インタフェース１６５、ｔｔｙ定義１７０、プリンタ定義１７５、ユーザＩＤ１８０、パスワード１８５、およびアプリケーション固有情報など他のシステム・パラメータ１９０がある。旧ユーザ環境データ１００が新コンピュータ・システム１６０に複製されると、そのカスタム化設定は結局、ユーザが旧コンピュータ・システム１４０で使用するのに慣れていた設定と同じになる。
【００３０】
あるシステムから別のシステムに移動する場合に加え、環境データ１００はシステムで発生する様々なイベントに応答して取り込まれる。たとえば、所定のスケジュールで、ユーザの環境データは取り込まれ、旧コンピュータ・システム１４０または新コンピュータ・システム１６０に格納される。このように、環境データ１００は周期的に取り込まれるから、旧コンピュータ・システム１４０でシステムの故障が生じても、新ユーザＩＤ１５５を迅速に作成し、新環境データ１９５を作成することにより環境データを復元することができる。これにより、ユーザは最小限の時間内にまったく同じ動作中のシステムを手に入

れることができる。また、ユーザが環境データ１００を変更するたびごとに、他のイベント（すなわちトリガー）が環境データ１００を取り込むはずである。したがって、旧環境データ１００には最新のコピーが存在する。また、次々に取り込み操作が行われたあとには、旧環境データ１００のコピーが複数個あると好都合である。旧環境データ１００のコピーを複数個保持していれば、ユーザが環境データに新たに適用した更新が不所望であることが明らかになっても、ユーザは以前の環境データの組のうちの所望のものに迅速に復帰することができる。他のイベント（すなわちトリガー）とは数人のユーザの環境データを当該ユーザ達の介在を必要とすることなく、周期的に取り込む中央処理領域が出すコマンドのことである。このように、システム管理者は自分がサポートしているユーザ達のうちの少なくとも一人のユーザにシステムの故障が発生しても、自信をもってユーザ環境を復元することができる。

【００３１】

図２はデータ収集とデータ複製を行う高レベルのフローチャートを示す図である。プロセスは開始２００で開始し、判断２１０でデータ収集の時期であるか否かを判断する。判断２１０は様々な要因に基づいて行いうる。たとえば、データ収集はカスタム化のデータをバックアップする様々な時間間隔で繰り返すようにスケジュールを組むことができる。判断２１０の別のトリガーはユーザの環境データが変化したときに常にフラグ（すなわち標識）をセットするモニタである。この場合、このフラグがセットされたら、データ収集プロセスを呼び出す。また、判断２１０はユーザがあるシステムから別のシステムに移動したときにのみ行う人手による判断であってもよい。さらに、判断２１０はシステム管理者が行い、多数のユーザのデータを収集する時期を判断するようにしてもよい。判断２１０が偽である（すなわちデータ収集の時期ではない）場合、「Ｎｏ」分岐をとる。判断２１０が真である（すなわちデータ収集の時期である）場合、「Ｙｅｓ」分岐をとり、収集プロセス２２０（データ収集プロセスの詳細なフローチャートは図３を参照）を実行する。データ収集を行う意図は新ワークステーションにパーソナリティ情報を複製する点にある。特に、図３に詳細を示すように、データ収集の対象には様々なシステム・パラメータがあるが、たとえばプリンタ定義、ｔｔｙ定義、ネットワーク・インタフェース、ユーザＩＤ、パスワードなどが挙げられる。システム・パラメータ値はシステムの「パーソナリティ」を規定するものである。パーソナリティ情報とはユーザおよび／またはグループが選定できるパラメータ、設定、および／または、コンピュータ・システム、ソフトウェア、もしくはファームウェアの属性をカスタマイズするのに用いるオプションのことである。パーソナリティ・パラメータはメニューのカラー・スキームほど単純ではないが、特定のアプリケーション・プログラムに関する情報を処理するのに必要な好適なアルゴリズムの仕様と同程度には複雑である。収集プロセス２２０の処理または「Ｎｏ」分岐２１２に続いて、判断２４０で複製処理の時期であるか否かに関する第２の判断を行う。判断２４０が偽である（すなわち複製処理の時期ではない）場合、「Ｎｏ」分岐２４２をとり、開始２００と判断２１０にループバックする。他方、判断２４０が真である（すなわち複製処理の時期である）場合、「Ｙｅｓ」分岐２４４をとり、複製プロセス２５０（複製プロセス２５０の詳細は図４を参照）を実行する。複製プロセス２５０に続いて、プロセスは終了２７０で終了する。

【００３２】

図３は図２に示した収集プロセス２２０の詳細を含む中レベルのフローチャートを示す図である。プロセスは開始３００で開始し、不揮発性媒体を検査して図２に示した収集プロセス２２０が収集したデータを保持するのに十分な空き容量があるか否かを判断する（ステップ３０５）。不揮発性媒体に十分な空き容量がない場合、エラー・メッセージを表示してプロセスを終了する。不揮発性媒体に十分な空き容量がある場合、ライセンス・ファイルを特定し収集する必要のあるライセンス・ファイル情報を特定する（ステップ３１０）。一実施形態では、ライセンス・ファイルを特定するステップは人手によるプロセスであり、ユーザはライセンス・ファイルを特定するスクリプト・ファイルを変更する。ここで使用している「スクリプト・ファイル」なる用語はＵＮＩＸ(R)オペレーティング・シ

ステム内のシェル・プログラムとともに使用するシェル・スクリプト・ファイルを指している。これの例は付録に収められている。このようなスクリプト・ファイルは他のプログラミング態様、たとえばＲＥＸＸやＢＡＳＩＣのようなインタプリタ言語に加え、ＣやＰａｓｃａｌのようなコンパイル言語で実装することもできる（この場合、プログラム・ファイルを再コンパイルするのではなく、ライセンス・ファイルなどを特定するのにデータ・ファイルを使用する）。したがって、「スクリプト」なる用語はここでは、本発明の機能を実装するのに使うことのできるすべてのプログラミング言語を表わすのに使用している。

【００３３】

このスクリプト・ファイルは後刻、コンピュータ・システムがデータ収集処理を行うために実行する。ライセンス・ファイルの特定（ステップ３１０）が完了したら、アプリケーション・パーソナリティ・データを特定し（ステップ３１５）、インストールされているアプリケーションを特定するとともにそのようなアプリケーションに対応するカスタマイズされた設定も特定する。ここでも一実施形態では、そのような特定は人手で行い、アプリケーション・パーソナリティ・データに対応するスクリプト・ファイルを変更し引き続いて行われるコンピュータ・システムによる実行に備える。次いで、ライセンス・データとパーソナリティ・データを、収集すべき情報を特定するデータ収集プログラム（データ収集スクリプト）に付加する（ステップ３２０）。本発明の一実施形態によると、複数のワークステーションに対してデータ収集とデータ複製を行うことができる。ワークステーションを特定する際には（ステップ３２５）、収集したデータを保持することになるワークステーションをアドレスによって特定する。次いで、これらのアドレスを収集リストに含める（ステップ３３０）。引き続いて呼び出すデータ収集スクリプトはワークステーション収集リストを読み取り、そのリストで特定されている各ワークステーションごとにデータ収集を行う。

【００３４】

収集リストに列挙された各ワークステーションごとに、データ収集プロセス（事前定義プロセス３４０、詳細は図５参照）、アプリケーション情報収集プロセス（事前定義プロセス３５０、詳細は図６参照）、および品質チェックリスト作成プロセス（事前定義プロセス３６０、詳細は図７参照）を実行し、データを不揮発性記憶媒体に書き込む（ステップ３７０）。データを不揮発性記憶媒体に書き込んだら、データ収集プロセスはステップ３９０で終了する。

【００３５】

図４は図２に示した複製準備・実行プロセス２５０が行う処理の中レベルのフローチャートを示す図である。プロセスは開始４００で開始し、複製プログラムを更新する（ステップ４１０）。複製プログラム更新ステップ（ステップ４１０）では、不揮発性記憶媒体に書き込んだ情報（図３のステップ３７０参照）を用いてワークステーションで実行されることになる複製スクリプトを更新する。次いで、ホスト名ＩＰアドレス・プロセス（ステップ４２０）の間に、ユーザの新ワークステーションに永続的なホスト名とＩＰアドレスを付与する。次いで、複製プログラムを新ワークステーションに転送する（ステップ４３０）。複製プログラムは新ワークステーションにおいてルート・ユーザ・セッションから実行されることになる。次いで、複製プログラム（事前定義プロセス４４０）を実行して旧ワークステーションから以前に収集した設定を新ワークステーションに複製する。複製処理（事前定義プロセス４４０）の詳細は図８に示す。次いで、複製プログラムは（ＵＮＩＸ(R)環境において）Ｘウインドウのカスタマイズを行いうるように、Ｘウインドウ設定プログラム（事前定義プロセス４５０）を呼び出す。Ｘウインドウ設定プログラム（事前定義プロセス４５０）の詳細は図９に示す。次いで、カスタム許可リスト（ｐｅｒｍ．ｌｉｓｔ）を用いてファイル許可を復元する（ステップ４６０）。ファイル許可を復元したら、旧ワークステーションから以前に収集した情報から一意のファイルシステムを生成する（ステップ４７０）。一意のファイルシステムを生成したら、一意のファイルシステムから以前に収集したデータをステップ４７０で生成したファイルシステムに復元する（

ステップ４８０）。データを復元したら、複製プロセスは終了ステップ４９０で終了する。

【００３６】

完了したら、ユーザの複製済みワークステーションには、ユーザの旧ワークステーションに存在したパーソナリティ情報と一致するパーソナリティ情報が備えられている。したがって、ユーザは新ワークステーションを再カスタマイズすることに付随する退屈な仕事から解放される。また、パーソナリティ情報を収集しそのデータを用いて一群のワークステーションをカスタマイズする日常的な仕事は自動化して中央集中した場所から実行することが可能になる。他方、システム管理者はユーザがあるワークステーションから別のワークステーションに移動したことに起因してユーザの新ワークステーションを手動で再構成する責任から解放される。ワークステーションに関するデータを取り込み、かつ物理的に分離された媒体に該データを複製する手段を提供しているので、本発明によれば、あるワークステーションから次のワークステーションへの移動がより滑らかになるとともに、ユーザが新ワークステーションを構成する際に経験するダウン時間が低減する。本発明によれば、ＵＮＩＸ(R)ベースのオペレーションにさらなる利点が得られる。ＩＢＭのＡＩＸオペレーティング・システムで開発したけれども、ここに開示する原理は他のＵＮＩＸ(R)環境、たとえばＬＩＮＵＳやＳｏｌａｒｉｓなどに容易に適用することができる。また、ここに開示する原理は非ＵＮＩＸ(R)ベースのオペレーティング・システム、たとえば他のマルチユーザ・オペレーティング・システムや他のマルチタスク・オペレーティング・システム（たとえばＩＢＭのＯＳ／２やマイクロソフトのＷｉｎｄｏｗｓ(R)９５／９８／ＮＴなど）にも適用することができる。多数のコンピュータ・システム間でユーザ環境設定を複製するというコンセプトは新規かつ独自であり、あるワークステーションから別のワークステーションに移転させる必要のあるカスタマイズしたユーザ設定を有するあらゆる人にとって有益なものである。

【００３７】

図５はデータ収集プロセス（図３に示した事前定義プロセス３４０）の低レベルのフローチャートを示す図である。データ収集プロセスはステップ５００で開始する。まず、データを格納するのに使用した不揮発性記憶装置（たとえばテープやディスク駆動装置など）の場所を特定する試験を行う（ステップ５０５）。次いで、以後の処理に備えて環境変数を設定する（ステップ５１０）。環境変数を設定したら、クライアント・ワークステーションと不揮発性記憶装置との間の接続性を確認したのち、ホスト名の作業リストを作成する（ステップ５１５）。接続性を確認したら、ワークステーションの／ｈｏｍｅ／ｌｏｃａｌディレクトリから情報を取り込んだのち、ワークステーション固有の他の情報を収集する（ステップ５２０）。次いで、ネットワーク情報を取り込み（ステップ５２５）（これには主要なネットワーク構成ファイルの取り込みが含まれる）、ワークステーションのイーサネット(R)・アダプタの状態を取得し（ステップ５３０）、イーサネット(R)・アダプタに対応するＩＰアドレスを取得し（ステップ５３５）、ネットワークへのデフォルトのゲートウェイ・アドレスを取得し（ステップ５４０）、デフォルトのネットマスクを取得し（ステップ５４５）、そして、（ドメイン名とネーム・サーバ・アドレスを含む）ドメイン情報を取得する（ステップ５５０）。ネットワーク情報を取り込んだら、システム情報を取り込む（ステップ５５５）。次いで、旧ワークステーションからｔｔｙ情報を取り込む（ステップ５６０）。ｔｔｙ情報を取り込んだら、プリンタ・キュー情報を取り込む（ステップ５６５）。次いで、（図３に示したステップ３１５で先に特定した）アプリケーション固有データを取り込む（ステップ５７０）。アプリケーション固有データを取り込んだら、（図３に示した事前定義プロセス３１０で先に特定した）ライセンス情報を取り込む。最後に、ファイルシステム情報を取り込む（ステップ５８０）。図５に示す収集プロセスの間に取り込んだ情報は着脱可能かつ不揮発性のコンピュータ操作可能媒体、たとえばディスケット、テープ、ＣＤ－Ｒなどに格納する。

【００３８】

図６はアプリケーション情報収集（図３の事前定義プロセス３５０を参照）に含まれる詳

細を示す低レベルのフローチャートを示す図である。プロセスを開始したのち（ステップ６００）、着脱可能な不揮発性記憶装置から個別のワークステーションにプログラムをコピーする（ステップ６１０）。次いで、ワークステーションからアプリケーション情報を収集する（ステップ６２０）。次いで、特定した各アプリケーションごとにループ構造に入り、アプリケーション情報を収集する。もしデータを取り込む対象となるアプリケーションが存在しなければ（判断６３０）、「Ｎｏ」分岐６３５を実行し６９０でプロセスを終了する。他方、アプリケーションが存在すれば、「Ｙｅｓ」分岐６４０を実行しアプリケーション情報を取得する。

【００３９】

まず、アプリケーション論理ボリューム名を取り込む（ステップ６５０）。次いで、アプリケーション・マウント・ポイントとファイルシステム名を取り込む（ステップ６６０）。最後に、アプリケーション・ファイルシステムのサイズを取り込み（ステップ６７０）、プロセスをループバックし（ループ６８０）、ステップ６３０で取り込むべきデータを有するアプリケーションが他にもあるか否かを調べる。このループは取り込むべきアプリケーションがなくなるまで続ける。それがなくなった時点で、「Ｎｏ」分岐６３５を実行し６９０でプロセスを終了する。

【００４０】

図７は品質チェックリスト（図３の事前定義プロセス３６０を参照）作成の詳細を示す低レベルのフローチャートを示す図である。プロセスを開始したら（ステップ７００）、クライアントのホスト名を取り込んだことを確認するチェックを行う（ステップ７０５）。次いで、ワークステーションと着脱可能な不揮発性記憶装置との間が接続されていることを確認するチェックを行う（ステップ７１０）。次いで、ネットワーキング・ファイルをチェックし（ステップ７１５）それらが正しいことを確認する。ネットワーキング・ファイルをチェックしたら、／ｈｏｍｅ／ｌｏｃａｌディレクトリの復元をチェックする（ステップ７２０）。次いで、パスワードをチェックしそれらが適切に収集されていることを確認する（ステップ７２５）。パスワードをチェックしたら、ファイルシステムをチェックする（ステップ７３０）。次いで、／ｕｓｒ／ｌｏｃａｌディレクトリの復元をチェックする（ステップ７３５）。上記ディレクトリをチェックしたら、ｔｔｙをチェックし確認する（ステップ７４０）。次いで、すべての包含ファイル／排除ファイルについてＡＤＳＭ（ＩＢＭのADSTAR Distributed Storage Manager）をチェックする（ステップ７４５）。次いで、以前に特定し収集したアプリケーション情報をチェックする（ステップ７５０）。最後に、ワークステーション用のイーサネット(R)とＩＰアドレスをチェックする（ステップ７５５）。すべてのチェックを行ったら、７９０でプロセスを終了する。以上、直列プロセスとして説明したが、上述したプロセスは異なった順序で行うことがことができる。また、あるプロセスは他のプロセスと同時に（並行して）行うことができる。図７のステップ群は命令シートに従って操作者が人手で行うことができるし、あるいは上述したチェック群を実行するように設計したプログラムによって自動的に行うこともできる。

【００４１】

図８は収集したワークステーションのデータを別のワークステーションに複製する手順（図４の事前定義プロセス４４０を参照）を示す低レベルのフローチャートを示す図である。まず、着脱可能な不揮発性記憶装置が導入され設定されていない場合、着脱可能な不揮発性記憶装置を導入し、および／または、ワークステーションとともに動作しうるように設定する（ステップ８１０）。着脱可能な不揮発性記憶装置に、取得したデータの保存元である着脱可能な不揮発性コンピュータ媒体をロードして、以前に取得したユーザ環境データをコピーする（ステップ８２５）。次いで、ネットワーキング・ファイルを復元する（ステップ８３０）。次いで、以前に取得した一意のシステム情報を新ワークステーションに複製する（ステップ８３５）。一意のシステム情報を複製したら、ｈｏｍｅ／ｌｏｃａｌのデータを復元し（ステップ８４０）、ＡＤＳＭファイルをすべて復元する（ステップ８４５）。次いで、以前に特定し取得したアプリケーション・データを新ワークステー

ションに複製する（ステップ８５０）。
【００４２】
アプリケーションを復元したら、旧ワークステーションから取得したｔｔｙ情報を新ワークステーションに複製する（ステップ８５５）。次いで、旧ワークステーションから取得したリモート・プリンタ定義を新ワークステーションに複製する（ステップ８６０）。リモート・プリンタ定義を複製したら、ワークステーションのＩＰアドレスとワークステーション名（ホスト名）を、ユーザが付与した永続的なホスト名とＩＰアドレスに割り当てる（ステップ８６５）。ホスト名とＩＰアドレスを変更したら、イーサネット(R)・インタフェースをチェックし確認する（ステップ８７０）。次いで、新ワークステーションに複製された主要なファイルのファイル許可を旧ワークステーションから新ワークステーションへの移動を反映するように変更する（ステップ８７５）。次いで、モニタのタイムアウト無効にしたのち（ステップ８８０）、ライセンス・ファイルのデータを旧ワークステーションから新ワークステーションへの被ライセンス・データ（たとえばソフトウェア製品など）の複製を反映させるように変更する（ステップ８８５）。最後に、テンポラリ・ファイルをクリーンアップしたのち（ステップ８９０）、８９５でプロセスを終了する。
【００４３】
上で概説したステップ群はＵＮＩＸ(R)環境用のものである。別の実施形態では、オペレーティング・システムの相違に起因してプロセス・ステップがいくぶん異なる。また、上で概説したステップ群はいくぶん異なる順序で実行することもできるが、ＵＮＩＸ(R)環境用には上で概説した順序が好適である。
【００４４】
図９は新ワークステーションにおいてＸウインドウを変更する手順（図４のステップ４５０参照）の詳細を示す低レベルのフローチャートを示す図である。まず、新ウインドウを生成する（ステップ９１０）。次いで、Ｘウインドウ・セッションを終了し（ステップ９２０）、新ワークステーションのＸウインドウのランドスケープ（横長）設定（ステップ９３０）およびポートレイト（縦長）設定（ステップ９４０）を設定する。設定を変更したら、９９０でＸウインドウ設定のプロセスを終了する。
【００４５】
図１０はここで説明した取得と複製を実行しうるコンピュータ・システムの簡易な例であるコンピュータ・システム１００１を示す図である。コンピュータ・システム１００１はホスト・バス１００５に接続されたプロセッサ１０００を備えている。ホスト・バス１００５にはレベル２（Ｌ２）キャッシュ・メモリ１０１０も接続されている。メインメモリ１０２０に接続されたホスト－ＰＣＩブリッジ１０１５はキャッシュ・メモリ制御機能とメインメモリ制御機能を備えるとともに、ＰＣＩバス１０２５、プロセッサ１０００、Ｌ２キャッシュ１０１０、メインメモリ１０２０、およびホスト・バス１００５の間の転送を処理するバス制御機能を備えている。ＰＣＩバス１０２５はたとえばＬＡＮカード１０３０を含む様々な装置用のインタフェースを提供している。ＰＣＩ－ＩＳＡブリッジ１０３５はＰＣＩバス１０２５とＩＳＡバス１０４０と間の転送を処理するバス制御機能、ＵＳＢ（universal serial bus）機能１０４５、ＩＤＥ装置機能１０５０、電力管理機能１０５５を備えているが、さらにリアル・タイム・クロック（ＲＴＣ）、ＤＭＡ制御、割込みサポート、およびシステム管理バス・サポートなど図示しない他の機能要素を備えていてもよい。周辺装置および入力／出力（Ｉ／Ｏ）装置はＩＳＡバス１０４０に接続された様々なインタフェース１０６０（たとえばパラレル・インタフェース１０６２、シリアル・インタフェース１０６４、赤外線（ＩＲ）インタフェース１０６６、キーボード・インタフェース１０６８、マウス・インタフェース１０７０、および固定ディスク（ＦＤＤ）１０７２など）に取り付けることができる。あるいは、ＩＳＡバス１０４０にスーパーＩ／Ｏコントローラ（図示せず）を取り付け、それにより多くのＩ／Ｏ装置を制御するようにしてもよい。
【００４６】
ＢＩＯＳ１０８０はＩＳＡバス１０４０に接続されており、様々な低レベルのシステム機能

とシステム・ブート機能用のプロセッサ実行可能なコードが組み込まれている。ＢＩＯＳ１０８０は任意のコンピュータ読み取り可能な媒体、たとえば磁気記憶装置、光記憶媒体、フラッシュ・メモリ、ランダム・アクセス・メモリ、リード・オンリー・メモリ、および命令を符号化する信号（たとえばネットワークからの信号）を搬送する通信媒体などにも格納することができる。コンピュータ・システム１００１をローカル・エリア・ネットワークを介してＮＩＭサーバに接続するために、ＬＡＮカード１０３０がＰＣＩ－ＩＳＡブリッジ１０３５に接続されている。同様に、電話線接続を用いてＮＩＭサーバに接続するために、シリアル・ポート１０６４とＰＣＩ－ＩＳＡブリッジ１０３５にモデム１０７５が接続されている。

【００４７】

図１０に記載したコンピュータ・システムはここで説明した取得プロセスと複製プロセスを実行しうるけれども、このコンピュータ・システムはコンピュータ・システム一般の単なる一例にすぎない。当業者が認識しうるように、他の多くのコンピュータ・システム構成でＵＮＩＸ(R)オペレーティング・システム（またはマイクロソフト・コーポレーションがライセンスしたＷｉｎｄｏｗｓ(R)９５／９８／ＮＴやＩＢＭがライセンスしたＡＩＸもしくはＯＳ／２など任意のオペレーティング・システム）を実行しうるとともに、ここで説明した処理を実行することができる。

【００４８】

図１１はワークステーション環境の収集と複製に含まれるプロセスの階層を示す図である。この階層図ではシステム複製１１００が最高レベルにある。システム複製１１００は２つの一般的なプロセス・カテゴリ、収集１１１０と複製１１２０に分解することができる。収集１１１０は旧ワークステーションからユーザ環境データを収集するプロセスである。複製１１２０は収集１１１０によって収集した情報を新ワークステーションに複製するプロセスである。収集１１１０は次に示す４つのプロセスに分解することができる。データ収集１１３０は旧ワークステーションからユーザ環境データを収集する（図５に示したフローチャートを参照）。ワークステーション・リスト１１４０にはユーザ環境データを収集させ複製させたワークステーションのリストが含まれている。アプリケーション・データの収集１１５０には旧ワークステーションからアプリケーション・データを収集する処理が含まれている（図６に示したフローチャートを参照）。最後に、品質チェックリスト１１６０にはユーザ環境データの収集に成功したか否かをチェックする処理が含まれている（図７に示したフローチャートを参照）。

【００４９】

階層図の複製１１２０側では、複製１１２０から分解された２つのプロセスが示されている。第１に、複製１１７０は旧ワークステーションから収集したユーザ環境データを新ワークステーションに複製するプロセスである（図８に示したフローチャートを参照）。第２に、Ｘウインドウ設定１１８０は新ワークステーションにおいてＸウインドウ設定を設定するプロセスである（図９に示したフローチャートを参照）。

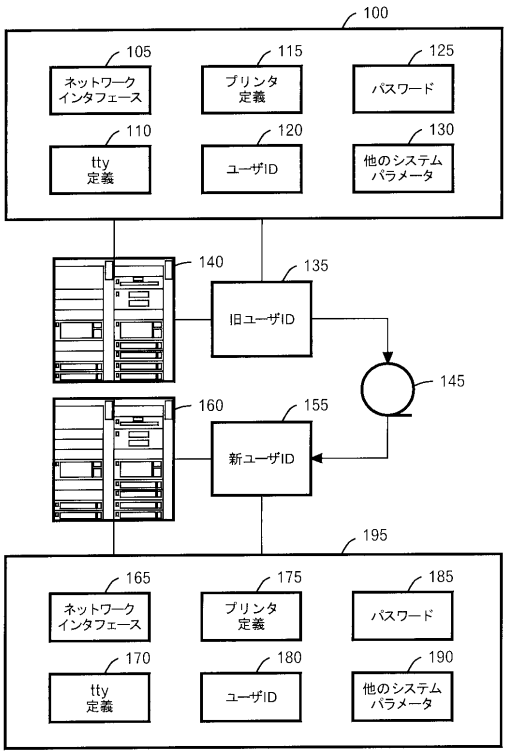
【００５０】

本発明の好適な実装の１つはクライアント・アプリケーション、すなわちたとえばコンピュータのランダム・アクセス・メモリに常駐するコード・モジュール中の１組の命令（プログラム・コード）である。コンピュータが必要とするまで、この１組の命令は別のコンピュータ・メモリ、たとえばハード・ディスク駆動装置、または着脱可能メモリ、たとえば光ディスク（終局的にはＣＤ－ＲＯＭ駆動装置で使用する）もしくはフロッピー(R)・ディスク（終局的にはフロッピー(R)・ディスク駆動装置で使用する）に格納しておく、あるいは、インターネットその他のコンピュータ・ネットワークを介してダウンロードする。したがって、本発明はコンピュータで使用するコンピュータ・プログラム製品として実装することができる。また、上述した様々な方法は便宜上、ソフトウェアによって選択的に駆動または再構成される汎用コンピュータに実装したが、当業者が認識しうるように、そのような方法は必要な方法ステップを実行しうるように構築されたハードウェア、ファームウェア、または専用装置で実行することができる。

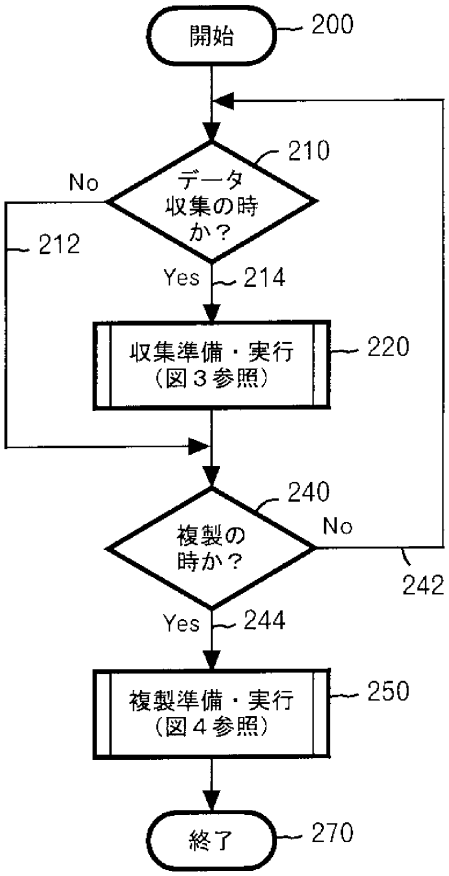
【００５１】
本発明の特定の実施形態を示しかつ説明したが、当業者にとって明らかなように、ここに開示した教示に基づいて、本発明およびそのより広い側面の範囲内で変形と変更をなすことができる。したがって、そのような変形と変更は本発明の真の範囲内のものである。
【図面の簡単な説明】
【００５２】
【図１】コンピュータ操作可能媒体で複製中のユーザ環境データを示すシステム図である。
【図２】本発明の一実施形態を示す高レベルのフローチャートを示す図である。
【図３】収集相を示す中レベルのフローチャートを示す図である。
【図４】複製相を示す中レベルのフローチャートを示す図である。
【図５】データ収集ステップ群を示す低レベルのフローチャートを示す図である。
【図６】アプリケーション情報収集を示す低レベルのフローチャートを示す図である。
【図７】品質チェックリストの作成を示す低レベルのフローチャートを示す図である。
【図８】複製ステップ群を示す低レベルのフローチャートを示す図である。
【図９】ディスプレイ設定の複製を示す低レベルのフローチャートを示す図である。
【図１０】情報処理システムを示すブロック図である。
【図１１】本発明の一実施形態で使用するスクリプト・ファイルを示す階層図である。
【符号の説明】
【００５３】
１００　ユーザの旧ユーザ環境データ
１０５　ネットワーク・インタフェース
１１０　ｔｔｙ定義
１１５　プリンタ定義
１２０　ユーザＩＤ
１２５　パスワード
１３０　他のシステム・パラメータ
１３５　旧ユーザＩＤ
１４０　旧コンピュータ・システム
１５５　新ユーザＩＤ
１６０　新コンピュータ・システム
１６５　ネットワーク・インタフェース
１７０　ｔｔｙ定義
１７５　プリンタ定義
１８０　ユーザＩＤ
１８５　パスワード
１９０　他のシステム・パラメータ
１９５　新ユーザ環境データ
１０００　プロセッサ
１００１　コンピュータ・システム
１００５　ホスト・バス
１０１０　レベル２キャッシュ
１０１５　ホスト－ＰＣＩブリッジ
１０２０　メインメモリ
１０２５　ＰＣＩバス
１０３０　ＬＡＮカード
１０３５　ＰＣＩ－ＩＳＡブリッジ
１０４０　ＩＳＡバス
１０４５　ＵＳＢインタフェース
１０５０　ＩＤＥインタフェース

１０５５　電力管理機能
１０６０　インタフェース
１０６２　パラレル・インタフェース
１０６４　シリアル・インタフェース
１０６６　赤外線（ＩＲ）インタフェース
１０６８　キーボード・インタフェース
１０７０　マウス・インタフェース
１０７２　固定ディスク（ＦＤＤ）
１０８０　ＢＩＯＳ
１１００　システム複製
１１１０　収集
１１２０　複製
１１３０　データ収集
１１４０　ワークステーション・リスト
１１５０　アプリケーション情報の収集
１１６０　品質チェックリスト
１１７０　複製
１１８０　Ｘウインドウ設定

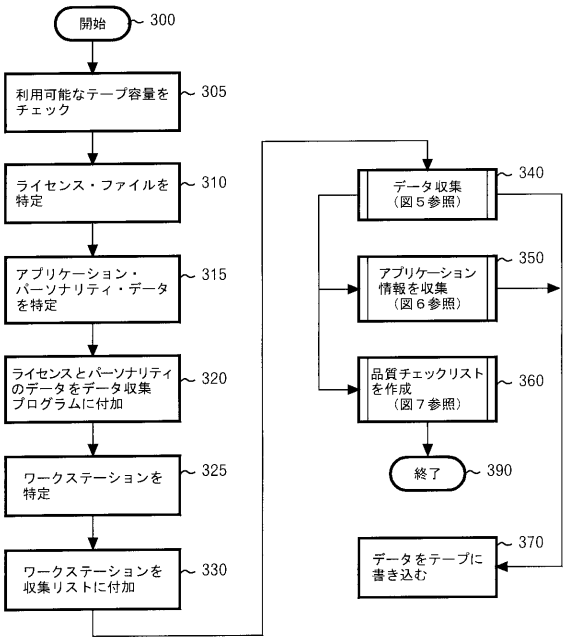
【図１】



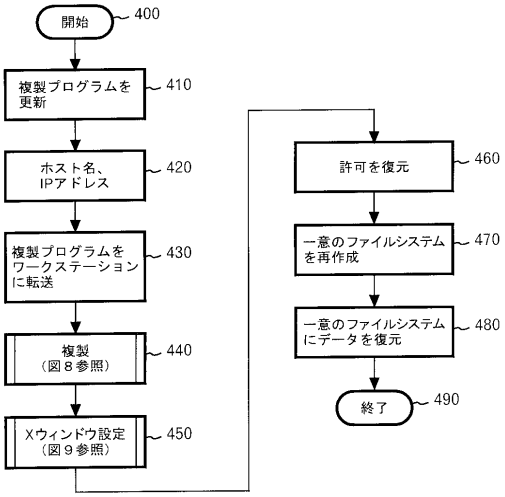
【図２】



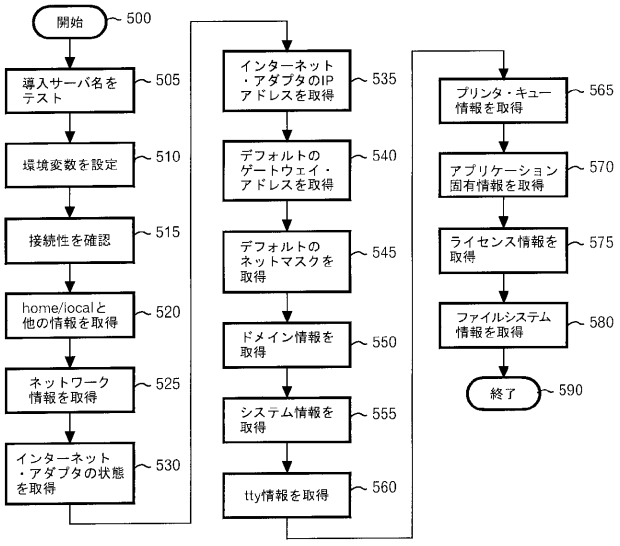
【図３】


開始
300
利用可能なテープ容量をチェック
305
ライセンス・ファイルを特定
310
アプリケーション・パーソナリティ・データを特定
315
ライセンスとパーソナリティのデータをデータ収集プログラムに付加
320
ワークステーションを特定
325
ワークステーションを収集リストに付加
330
データ収集（図５参照）
340
アプリケーション情報を収集（図６参照）
350
品質チェックリストを作成（図７参照）
360
終了
390
データをテープに書き込む
370

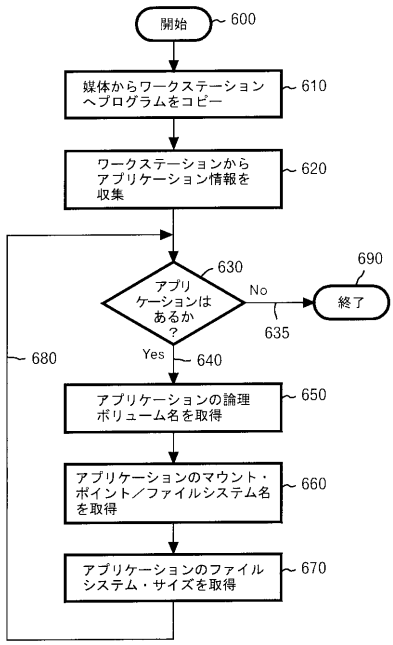
【図４】


開始
400
複製プログラムを更新
410
ホスト名、IPアドレス
420
複製プログラムをワークステーションに転送
430
複製（図８参照）
440
Xウィンドウ設定（図９参照）
450
許可を復元
460
一意のファイルシステムを再作成
470
一意のファイルシステムにデータを復元
480
終了
490

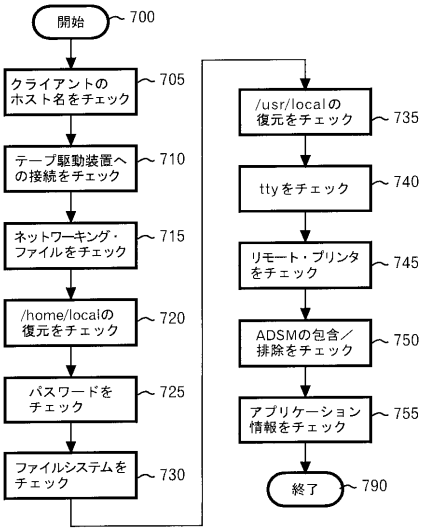
【図５】


開始
500
導入サーバ名をテスト
505
環境変数を設定
510
接続性を確認
515
home/iocalと他の情報を取得
520
ネットワーク情報を取得
525
インターネット・アダプタの状態を取得
530
インターネット・アダプタのIPアドレスを取得
535
デフォルトのゲートウェイ・アドレスを取得
540
デフォルトのネットマスクを取得
545
ドメイン情報を取得
550
システム情報を取得
555
tty情報を取得
560
プリンタ・キュー情報を取得
565
アプリケーション固有情報を取得
570
ライセンス情報を取得
575
ファイルシステム情報を取得
580
終了
590

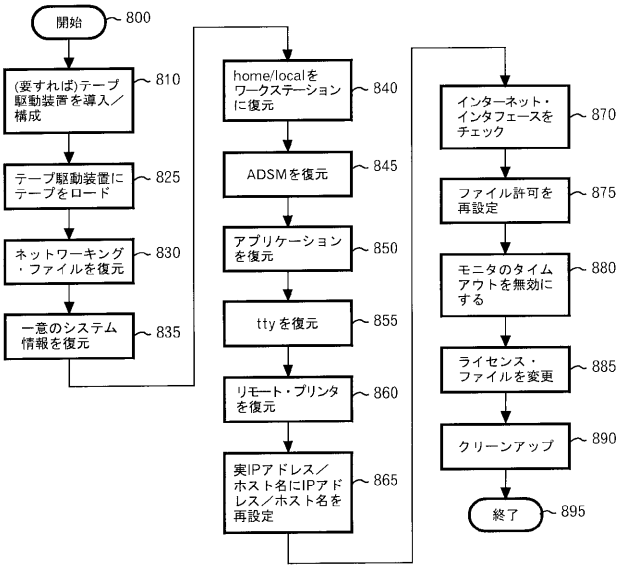
【図６】


開始
600
媒体からワークステーションへプログラムをコピー
610
ワークステーションからアプリケーション情報を収集
620
アプリケーションはあるか？
630
No
635
終了
690
Yes
640
680
アプリケーションの論理ボリューム名を取得
650
アプリケーションのマウント・ポイント／ファイルシステム名を取得
660
アプリケーションのファイルシステム・サイズを取得
670

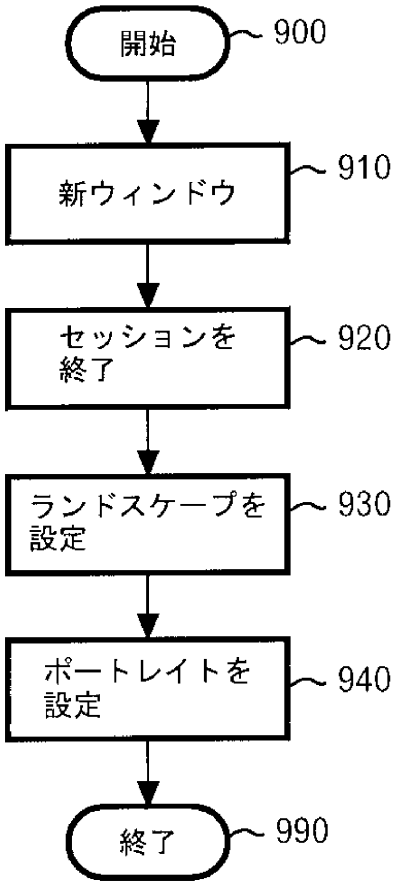
【図７】


開始
700
クライアントのホスト名をチェック
705
テープ駆動装置への接続をチェック
710
ネットワーキング・ファイルをチェック
715
/home/localの復元をチェック
720
パスワードをチェック
725
ファイルシステムをチェック
730
/usr/localの復元をチェック
735
ttyをチェック
740
リモート・プリンタをチェック
745
ADSMの包含／排除をチェック
750
アプリケーション情報をチェック
755
終了
790

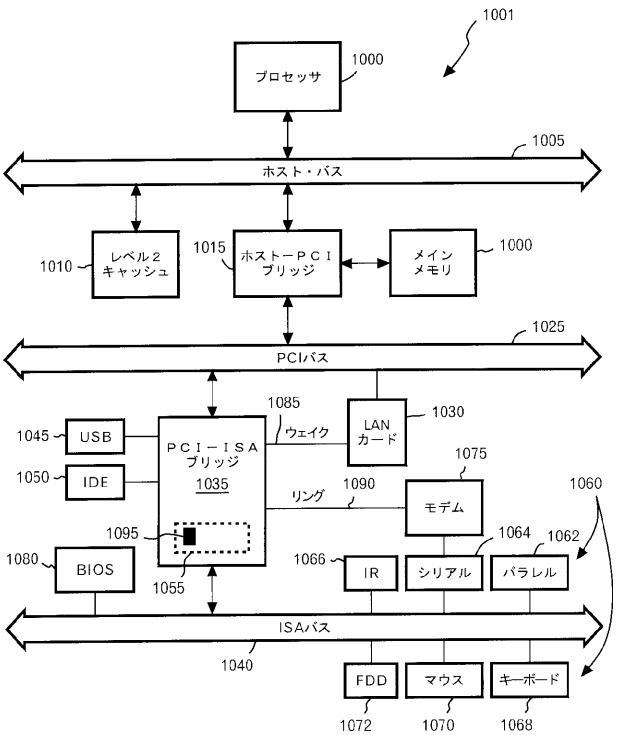
【図８】


開始
800
(要すれば)テープ駆動装置を導入／構成
810
テープ駆動装置にテープをロード
825
ネットワーキング・ファイルを復元
830
一意のシステム情報を復元
835
home/localをワークステーションに復元
840
ADSMを復元
845
アプリケーションを復元
850
ttyを復元
855
リモート・プリンタを復元
860
実IPアドレス／ホスト名にIPアドレス／ホスト名を再設定
865
インターネット・インタフェースをチェック
870
ファイル許可を再設定
875
モニタのタイムアウトを無効にする
880
ライセンス・ファイルを変更
885
クリーンアップ
890
終了
895

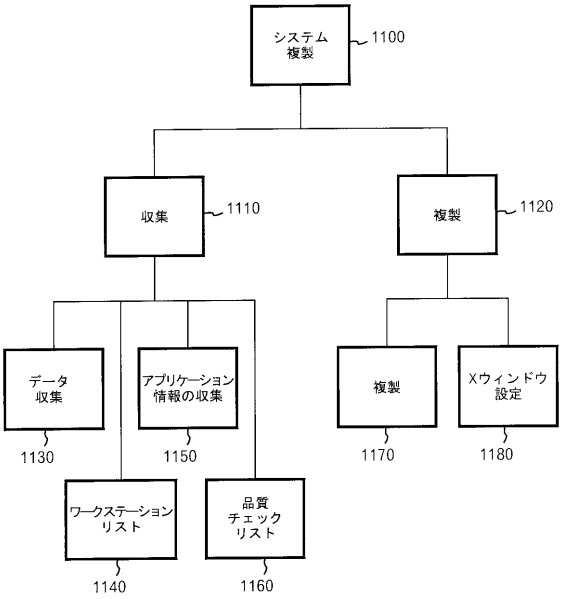
【図９】


開始
900
新ウィンドウ
910
セッションを終了
920
ランドスケープを設定
930
ポートレイトを設定
940
終了
990

【図１０】


1001
プロセッサ
1000
1005
ホスト・バス
1010
レベル２キャッシュ
1015
ホストーＰＣＩブリッジ
メインメモリ
1000
1025
PCIバス
1085
ウェイク
LANカード
1030
1045
USB
1050
IDE
ＰＣＩ－ＩＳＡブリッジ
1035
リング
1090
1075
モデム
1060
1095
1080
BIOS
1055
1066
IR
1064
シリアル
1062
パラレル
ISAバス
1040
FDD
マウス
キーボード
1072
1070
1068

【図１１】


システム
複製
1100
収集
1110
複製
1120
データ
収集
1130
アプリケーション
情報の収集
1150
複製
1170
Xウィンドウ
設定
1180
ワークステーション
リスト
1140
品質
チェック
リスト
1160

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 October 2002 (17.10.2002)

PCT

(10) International Publication Number
WO 02/082266 A2

(51) International Patent Classification[7]: G06F 9/46

(21) International Application Number: PCT/GB02/01415

(22) International Filing Date: 25 March 2002 (25.03.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/826,608 5 April 2001 (05.04.2001) US

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, NY 10504 (US).

(71) Applicant *(for MG only)*: **IBM UNITED KINGDOM LIMITED** [GB/GB]; PO Box 41, North Harbour, Portsmouth, Hampshire PO6 3AU (GB).

(72) Inventors: **HAMILTON, Rick**; 1532 Dairy Road, Charlottesville, VA 22903 (US). **LIPTON, Steven**; 2609 Maywood Court, Flower Mound, TX 75028 (US).

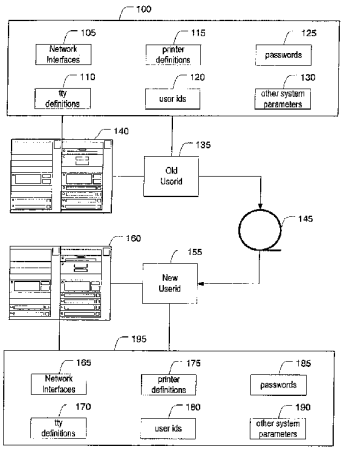
(74) Agent: **LITHERLAND, David, Peter**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester, Hampshire SO21 2JN (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR,

[Continued on next page]

(54) Title: COLLECTING AND RESTORING USER ENVIRONMENT DATA USING REMOVABLE STORAGE



(57) Abstract: A data collection program collects data from a user's workstation and captures the user environment data, including user settings and program application data. The user environment data is stored on a removable nonvolatile storage media for duplication processing. The stored user environment data is processed by a duplication process to duplicate the user environment data from the old workstation onto a new workstation or for recovery from a catastrophic system failure. A variety of user environment settings, not traditionally captured and restored by traditional backup software, are captured and restored. For example, licensing information and application personality data is identified, stored, and recovered along with other user-specific information such as hostnames, IP addresses, and the like.

WO 02/082266 A2

WO 02/082266 A2

GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Published:

— *without international search report and to be republished upon receipt of that report*

COLLECTING AND RESTORING USER ENVIRONMENT DATA USING REMOVABLE STORAGE

This application is related to the following co-pending U.S. Patent Applications filed on March 31, 2000 and having the same inventors and assignee: "System and Method for Event-Based Computer System Duplication" (Application No. 09/540,914, filed March 31, 2000), "System and Method for Computer System Duplication" (Application No. 09/540,344, filed March 31, 2000), and "Method and System for Implementing Network Filesystem-Based Customized Computer System Automated Rebuild Tool" (Application No. 09/422,361, filed October 21, 1999) each by Hamilton and Lipton and each assigned to the IBM Corporation.

The present invention relates to information processing technology. More particularly, the present invention relates to a system and method for restoration and recovery of user environment data in a computer system.

The UNIX operating system is an interactive time-sharing operating system invented in 1969. The UNIX operating system is a multi-user operating system supporting serial and network connected terminals for multiple users. UNIX is a multitasking operating system allowing multiple users to use the same system simultaneously. The UNIX operating system includes a kernel, shell, and utilities. UNIX is a portable operating system, requiring only the kernel to be written in assembler, and supports a wide range of support tools including development, debuggers, and compilers.

As a multi-user operating system, UNIX allows multiple people to share the same computer system simultaneously. UNIX accomplishes this by time-slicing the computer's central processing unit, or "CPU," into intervals. Each user gets a certain amount of time for the system to execute requested instructions. After the user's allotted time has expired, the operating system intervenes by interrupting the CPU, saving the user's program state (program code and data), restores the next user's program state and begins executing the next user's program (for the next user's amount of time). This process continues indefinitely cycling through all users using the system. When the last user's time-slice has expired, control is transferred back to the first user again and another cycle commences.

The UNIX operating system is both a multi-user operating system and a multi-tasking operating system. As the name implies, the multi-user aspect of UNIX allows multiple users to use the same system at the same time. As a multi-tasking operating system, UNIX permits multiple programs (or portions of programs called threads of execution) to execute at the same time. The operating system rapidly switches the processor between the various programs (or threads of execution) in order to execute each of the programs or threads. IBM's OS/2 and Microsoft's Windows 95/98/NT are examples of single-user multi-tasking operating systems while UNIX is an example of a multi-user multi-tasking operating system. Multi-tasking operating systems support both foreground and background tasks. A foreground task is a task that directly interfaces with the user using an input device and the screen. A background task runs in the background and does not access the input device(s) (such as the keyboard, a mouse, or a touch-pad) and does not access the screen. Background tasks include operations like printing which can be spooled for later execution.

The UNIX operating system keeps track of all programs running in the system and allocates resources, such as disks, memory, and printer queues, as required. UNIX allocates resources so that, ideally, each program receives a fair share of resources to execute properly. UNIX doles out resources using two methods: scheduling priority and system semaphores. Each program is assigned a priority level. Higher priority tasks (like reading and writing to the disk) are performed more regularly. User programs may have their priority adjusted dynamically, upwards or downwards, depending on their activity and the available system resources. System semaphores are used by the operating system to control system resources. A program can be assigned a resource by getting a semaphore by making a system call to the operating system. When the resource is no longer needed, the semaphore is returned to the operating system, which can then allocate it to another program.

Disk drives and printers are serial in nature. This means that only one request can be performed at any one time. In order for more than one user to use these resources at once, the operating system manages them using queues. Each serial device is associated with a queue. When a programs wants access to the device (i.e., a disk drive) it sends a request to the queue associated with the device. The UNIX operating system runs background tasks (called daemons), which monitor the queues and service requests for them. The requests are performed by the daemon process and the results are returned to the user's program.

Multi-tasking systems provide a set of utilities for managing processes. In UNIX, these are ps (list processes), kill (kill a process), and & at the end of a command line (run a process in the background). In UNIX, all user programs and application software use the system call interface to access system resources such as disks, printers, and memory. The system call interface in UNIX provides a set of system calls (C language functions). The purpose of the system call interface is to provide system integrity, as all low-level hardware access is under the control of the UNIX operating system and not the user-written programs. This prevents a program from corrupting the system.

Upon receiving a system call, the operating system validates its access permission, executes the request on behalf of the requesting program, and returns the results to the requesting program. If the request is invalid or the user does not have access permission, the operating system does not perform the request and an error is returned to the requesting program. The system call is accessible as a set of C language functions, as the majority of UNIX is written in the C language. Typical system calls are: _read - for reading from the disk; _write - for writing to the disk; _getch - for reading a character from a terminal; _putch - for writing a character to the terminal; and _ioctl - for controlling and setting device parameters.

As the name implies, the kernel is at the core of the UNIX operating system and is loaded each time the system is started, also referred to as a system "boot." The kernel manages the resources of the system, presenting them to the users as a coherent system. The user does not have to understand much, if anything, about the kernel in order to use a UNIX system. The kernel provides various necessary functions in the UNIX environment. The kernel manages the system's memory and allocates it to each process. It takes time for the kernel to save and restore the program's state and switch from one program to the next (called dispatching). This action needs to execute quickly because time spent switching between programs takes away from the time available to actually run the users' programs. The time spent in the "system state" where the kernel performs tasks like switching between user programs is the system overhead and should be kept as low as possible. In a typical UNIX system, system overhead should be less than 10% of the overall time.

The kernel also schedules the work to be done by the central processing unit, or "CPU," so that the work of each user is carried out efficiently. The kernel transfers data from one part of the system to

another. Switching between user programs in main memory is also done by the kernel. Main system memory is divided into portions for the operating system and user programs. Kernel memory space is kept separate from user programs. When insufficient main memory exists to run a program, another program is written out to disk (swapped) to free enough main memory to run the first program. The kernel determines which program is the best candidate to swap out to disk based on various factors. When too many programs are being executed on the system at the same time, the system gets overloaded and the operating system spends more time swapping files out to disk and less time executing programs causing performance degradation. The kernel also accepts instructions from the "shell" and carries them out. Furthermore, the kernel enforces access permissions that are in place in the system. Access permissions exist for each file and directory in the system and determine whether other users can access, execute, or modify the given file or directory.

For file handling, UNIX uses a hierarchical directory structure for organizing and maintaining files. Access permissions correspond to files and directories. As previously stated, the UNIX operating system organizes files into directories which are stored in a hierarchical tree-type configuration. At the top of the tree is the *root* directory which is represented by a slash (/) character. The root directory contains one or more directories. These directories, in turn, may contain further directories containing user files and other system files. A few standard directories that will be found in many UNIX are as follows:

/bin	This directory contains the basic system commands.
/etc	This directory contains system configuration files and programs used for administrating the system.
/lib	This directory contains the system libraries.
/tmp	This directory is used to store temporary files.
/usr/bin	This directory contains the commands that are not stored in /bin.
/usr/man	This directory contains manual pages for programs
/usr/local	This directory contains local programs that were installed by the system administrator (sysadmin) and were not included with the original system. In particular, /usr/local/bin contains local command files (binaries), and /usr/local/man contains local manual pages.
/home	The actual directory location varies from system to system, but somewhere on the system will be a location where all of the users' home directories are located.

The fundamental structure that the UNIX operating system uses to store information is the file. A file is a sequence of bytes. UNIX keeps track of files internally by assigning each file a unique identification number. These numbers, called i-node numbers, are used only within the UNIX kernel itself. While UNIX uses i-node numbers to refer to files, it allows users to identify each file by a user-assigned name. A file name can be any sequence of characters and can be up to fourteen characters long.

There are three types of files in the UNIX file system: (1) ordinary files, which may be executable programs, text, or other types of data used as input or produced as output from some operation; (2) directory files, which contain lists of files in directories outlined above; and (3) special files, which provide a standard method of accessing input/output devices.

Internally, a directory is a file that contains the names of ordinary files and other directories and the corresponding i-node numbers for the files. With the i-node number, UNIX can examine other internal tables to determine where the file is stored and make it accessible to the user. UNIX directories themselves have names, examples of which were provided above, and can be up to fourteen characters long.

UNIX maintains a great deal of information about the files that it manages. For each file, the file system keeps track of the file's size, location, ownership, security, type, creation time, modification time, and access time. All of this information is maintained automatically by the file system as the files are created and used. UNIX file systems reside on mass storage devices such as disk drives and disk arrays. UNIX organizes a disk into a sequence of blocks. These blocks are usually either 512 or 2048 bytes long. The contents of a file are stored in one or more blocks which may be widely scattered on the disk.

An ordinary file is addressed through the i-node structure. Each i-node is addressed by an index contained in an i-list. The i-list is generated based on the size of the file system, with larger file systems generally implying more files and, thus, larger i-lists. Each i-node contains thirteen 4-byte disk address elements. The direct i-node can contain up to ten block addresses. If the file is larger than this, then the eleventh address points to the first level indirect block. Addresses 12 and 13 are used for second level and third level indirect blocks, respectively, with the indirect addressing chain before the first data

block growing by one level as each new address slot in the direct i-node is required.

All input and output (I/O) is done by reading and writing files, because all peripheral devices, even terminals, are treated as files in the file system. In a most general case, before reading and writing a file, it is necessary to inform the system of the intention to do so by opening the file. In order to write to a file, it may also be necessary to create it. When a file is opened or created (by way of the "open" or "create" system calls), the system checks for the right to do so and, if the user has the right to do so, the system returns a non-negative integer called a file descriptor. Whenever I/O is to be done on this file, the file descriptor is used, instead of the file name, to identify the file. The open file descriptor has associated with it a file table entry kept in the "process" space of the user who has opened the file. In UNIX terminology, the term "process" is used interchangeably with a program that is being executed. The file table entry contains information about an open file, including an i-node pointer for the file and the file pointer for the file, which defines the current position to be read or written in the file. All information about an open file is maintained by the system.

In conventional UNIX systems, all input and output is done by two system calls - "read" and "write" - which are accessed from programs having functions of the same name. For both system calls, the first argument is a file descriptor, the second argument is a pointer to a buffer that serves as the data source or destination, and the third argument is the number of bytes to be transferred. Each "read" or "write" system call counts the number of bytes transferred. On reading, the number of bytes returned may be less than the number requested because fewer bytes than the number requested remain to be read. A return code of zero means that the end-of-file has been reached, a return code of -1 means that an error occurred. For writing, the return code is the number of bytes actually written. An error has occurred if this number does not match the number of bytes which were supposed to be written.

UNIX monitors the state of each terminal input line connected to the system with a system process called getty. When getty detects that a user has turned on a terminal, it presents the logon prompt, and when the userid and password are validated, the UNIX system associates a shell program (such as sh) with that terminal placing the user in the shell program. The shell program provides a prompt that typically signifies

which shell program is being executed. The user types commands at the prompt. The shell program acts as a command interpreter taking each command and passing them to the kernel to be acted upon. The shell then displays the results of the operation on the screen. Users use the shell to create a personalized environment that suits the needs of the user. The user can change environment variables that control the user's environment.

The EDITOR environment variable sets the editor that will be used by other programs such as the mail program. The PAGER environment variable sets the pager that will be used by programs such as man to display manual pages. The PATH environment variable specifies the directories that the shell is to look through to find a command. These directories are searched in the order in which they appear. The PRINTER environment variable sets the printer to which all output is sent by the lpr command. The SHELL variable sets the default shell that is used by the user. The TERM variable sets the terminal type for programs such as the editor and pager. The TZ environment variable sets the time zone where the user is located.

There are several shells that are available to UNIX users. Each shell provides different features and functionality than other shells. The most common UNIX shell programs are the "Bourne" shell, the "C" shell, the "TC" shell, the "Korn" shell, and the "BASH" shell. As well as using the shell to run commands, each of these shell programs have a built-in programming language that a user can use to write their own commands or programs. A user can put commands into a file - known as a *shell script* - and execute the file like a command or program. Shells invoke two types of commands: internal commands (such as set and unset) which are handled by the shell program and external commands (such as ls, grep, sort, and ps) which are invoked as programs.

One advantage of the UNIX operating system is that users can customize their working environment to suit their needs. For example, users can choose a default editor, a pager to display manual pages, a path to specify directories that are searched for commands, a default printer, a terminal type for use by the editor and the pager, a time-zone for displaying the correct time, and the shell program that is associated with the user's terminal upon logging on to the system.

One challenge in today's complex computing environment is moving a user from one system to another due to system changes or user relocation from one system to another system. Because of computer complexity and the

amount of customizing a user may make to his or her environment, duplicating a user's computing environment has become even more challenging. Recreating UNIX images, in particular, requires that numerous system parameters, including printer definitions, tty definitions (terminal definitions or the name of a particular terminal controlling a given job or even serial port definitions - in UNIX such devices have names of the form tty*), network interfaces, user Ids, and passwords. Failure to duplicate all such parameters may result in the inability of the user to run key applications or access critical resources following such a system duplication. Challenges in present duplication schemes wherein the duplication is largely a manual effort include time consuming manual tasks performed by the user and/or system administrator and the fact that such manual tasks are prone to errors.

It has been discovered that user environment data can be duplicated from a workstation to another workstation in a semi-automated fashion. Initially, an automated data collection process collects user environment data from the old computer system. User environment data may include application program data, license information, tty settings, customized directories, and other user customized workstation environment variables.

A list of workstations can also be used to duplicate a number of workstations, for example when a group of individuals are upgrading their workstations. The user environment data is stored for later duplication onto a new workstation. The user environment data is stored onto a removable computer operable medium, such as a diskette or a magnetic tape, for transporting and inputting into the new workstation or for use in restoring the old computer when a system failure occurs. Duplicating the user environment settings on a new workstation involves reading the user environment data stored on the computer operable medium and applying the user environment settings to the workstation.

An exemplary embodiment of the invention will now be described with reference to the accompanying drawings in which:

Figure 1 is a system diagram showing user environment data being duplicated through a computer operable medium;

Figure 2 is a high-level flowchart showing of one embodiment of the present invention;

Figure 3 is a mid-level flowchart showing the collection phase;

Figure 4 is a mid-level flowchart showing the duplication phase;

Figure 5 is a lower-level flowchart showing the data collection steps;

Figure 6 is a lower-level flowchart showing the application information collection;

Figure 7 is a lower-level flowchart showing the creation of a quality checklist;

Figure 8 is a low-level flowchart showing duplication steps;

Figure 9 is a lower-level flowchart showing display settings duplication;

Figure 10 is a block diagram showing an information handling system; and

Figure 11 is a hierarchy chart showing the script files used in one embodiment of the present invention.

Figure 1 is a system diagram illustrating user environment data being copied and sent from one computer system to another computer system. The user's old user environment data **100** includes network interfaces **105**, tty definitions **110**, printer definitions **115**, userids **120**, passwords **125**, and other system parameters **130**, such as application specific information. In essence, old user environment data **100** includes the customizations and modifications made to the user's account as a result of the user's preferences or made so the user could better perform his or her job related tasks. In this example, the user is moving from old userid **135** in old computer system **140** to new userid **155** connected to new computer system **160**. Once old user environment data **100** is gathered from old computer system **140**, they are transferred to new computer system **160** using nonvolatile computer media **145**. Nonvolatile computer media **145** may be may be a diskette, magnetic tape, ZIP disk, JAZ disk, CD-R (recordable CD-ROM), hard drive, optical disk, or any medium that can transfer data by being transported from one computer system to another.

After old user environment data **100** has been received at new computer system **160**, a process is invoked that duplicates old user

environment data **100** onto new computer system **160** creating new user environment data **195**. Similarly to old user environment data **100**, new user environment data **195** includes network interfaces **165**, tty definitions **170**, printer definitions **175**, userids **180**, passwords **185**, and other system parameters **190**, such as application specific information. After old user environment data **100** has been duplicated onto new computer system **160**, the customization settings are ultimately the same as the settings the user was accustomed to using on old computer system **140**.

In addition to moving from one system to another system, environment data **100** may be captured in response to various events that occur in the system. For example, on a given schedule, the user's environment data may be captured and stored on old computer system **140** or new computer system **160**. In this manner, the environment data **100** is periodically captured so that if a system failure occurs on old computer system **140**, new userid **155** can quickly be created and the environment data restored creating new environment data **195** and the user can have a substantially similar system in operation in a minimum amount of time. Other events, or triggers, may be to capture environment data **100** each time environment data **100** is modified by the user. In this manner, the old environment data **100** would have an up to date copy. It may also be advantageous to keep multiple copies of old environment data **100** after subsequent capture operations have taken place. By keeping multiple copies of old environment data **100**, the user can quickly regress to a former set of environment data should newly applied updates to the environment data prove to be undesirable by the user. Another event, or trigger, may be a command from a centralized operations area that periodically captures several users' environment data without need of such users' intervention. In this way, systems management personnel can have confidence in restoring user environment data should a system failure occur for one or more of the users they support.

Figure 2 shows a high-level flowchart for collecting data and duplicating data. The process is commenced at start **200** whereupon a decision is made to determine whether it is time for data collection at decision **210**. Decision **210** can be based upon various factors. For example, data collection could be scheduled to repeat at various time intervals to backup the customization data. Another trigger for decision **210** could a monitor that sets a flag or indicator any time user environment data is changed. If the flag is set, the data collection process is invoked. In addition, decision **210** could be a manual decision that is only made when the user is moving from one system to another.

Decision **210** could also be made by systems management personnel that decide when to collect data for a number of users. If decision **210** is false (i.e., not time for data collection), "no" branch **212** is taken. If decision **210** is true (i.e., time for data collection), "yes" branch **214** is taken to execute collection process **220** (see **Figure 3** for a detailed flowchart of the data collection process). Data collection is performed with the intention of duplicating personality information on a new workstation. In particular, and as described in further detail in **Figure 3**, data collection includes an awareness of many system parameters, including printer definitions, tty definitions, network interfaces, user Ids, and passwords. The system parameter values define a system's "personality." Personality information can be thought of as any user and/or group selectable parameters, settings, and/or options used for customizing either a computer system, software, or firmware attributes. Personality parameters might be as uncomplicated as menu color schemes or as complex as the specification of preferred algorithms needed for processing information with a particular application program. Following either processing of collection process **220** or "no" branch **212**, a second decision is made as to whether it is time for duplication processing at decision **240**. If decision **240** is false (i.e., not time for duplication processing), "no" branch **242** is taken looping back to start **200** and decision **210**. On the other hand, if decision **240** is true (i.e., time for duplication processing), "yes" branch **244** is taken to execute duplication process **250** (see **Figure 4** for details about duplication process **250**). Following duplication process **250**, processing terminates at termination **270**.

Figure 3 shows a mid-level flowchart containing details of collection process **220** shown in **Figure 2**. The process is commenced at start **300** whereupon the nonvolatile media is checked (step **305**) to determine whether there is sufficient space to hold the data collected by collection process **220** shown in **Figure 2**. If insufficient room exists on the nonvolatile media, an error message is displayed and processing is terminated. If sufficient room exists on the nonvolatile media, license files are identified (step **310**) to identify license file information that needs to be collected. In one embodiment, identifying license files is a manual process whereby the user modifies script files to identify the license files. The term "script file" as used herein refers to shell script files used with shells programs within the UNIX operating system, examples of which are included in the appendices. The functionality of such script files could be implemented in other programmatic fashions, such as interpreted languages such as REXX and BASIC, as well as compiled

languages, such as C and Pascal (with data files used to identify license files and the like rather than recompiling the program files). Consequently, the term "script" is used herein to represent any programming language that could be used to implement the functionality of the present invention.

The script files will later be executed by the computer system to perform data collection processing. After identifying license files (step **310**) completes, application personality data is identified (step **315**) to identify applications installed and also identify customized settings corresponding to such applications. Again, in one embodiment such identification is done manually and script files are modified respective to the application personality data for subsequent execution by the computer system. Next, license and personality data are added to the data collection program (step **320**) to identify the information to be collected to the data collection script. One embodiment of the present invention allows for data collection and duplication to be performed for multiple workstations. In identifying workstations (step **325**), the workstations that will have data collected are identified by address. These addresses are then included in a collection list (step **330**). The data collection scripts that will be invoked subsequently read the workstation collection list and perform data collection for each workstation identified in the list.

For each workstation listed in the collection list, data collection process (predefined process **340**, see **Figure 5** for further details), collect application information process (predefined process **350**, see **Figure 6** for further details), and create quality checklist process (predefined process **360**, see **Figure 7** for further details) are performed and the data is written to the nonvolatile storage media (step **370**). After the data is written to the nonvolatile storage media, data collection processing ends at **390**).

Figure 4 is a mid-level flowchart of the processing performed by duplication preparation and execution process **250** shown in **Figure 2**. The process is commenced at start **400** whereupon duplication program is updated (step **410**). Update duplication program (step **410**) uses the information that was written to the nonvolatile storage media (see **Figure 3**, step **370**) to update the duplication script that will be executed on the workstation. Next, during hostname IP address process (step **420**) a permanent hostname and IP address is provided for the user's new workstation. The duplication program is then transferred to the workstation (step **430**)

where it will be executed from a root user session on the new workstation. The duplication program (predefined process **440**) is then executed to duplicate the settings previously collected from the old workstation onto the new workstation. Details of the duplication (predefined process **440**) processing are shown in **Figure 8**. The duplication program subsequently calls the X-Windows settings program (predefined process **450**) so that customization of X-Windows (in a UNIX environment) can occur. Details of the X-Windows settings program (predefined process **450**) are shown in **Figure 9**. Next, file permissions are restored (step **460**) using a custom permission list (perm.list). After file permissions are restored, unique filesystems are created (step **470**) from the information previously collected from the old workstation. After the unique filesystems are created, data previously collected from the unique filesystems is restored (step **480**) to the filesystems created during step **470**. After the data has been restored, the duplication process ends at termination step **490**.

Upon completion, a user's duplicated workstation includes personality information matching the information that was present in the user's previous workstation. The user is thus freed from the tedious tasks associated with re-customizing the new workstation. The mundane tasks of gathering personality information and using that data for customizing a group of workstations is automated and may be performed from a centralized location. The system administrator, on the other hand, is freed from the responsibility of manually re-configuring a user's new workstation following the user's move from one workstation to another. By providing the means, both to capture data about workstations and duplicate it on physically separate media, the present invention ensures a smoother transition from one workstation to the next and reduces the amount of down-time a user experiences in configuring a new workstation. The present invention provides additional benefits to UNIX-based operation. Although developed on IBM's AIX operating system, the principles here are easily extendable to other UNIX environments, such as LINUS, Solaris, and others. The principles here are also extendable to non-UNIX-based operating systems, such as other multi-user operating systems and other multitasking operating systems (such as IBM's OS/2 and Microsoft's Windows 95/98/NT). The concept of duplicating user environment settings across a wide number of computer systems is both new and unique and will benefit anyone with customized user settings that needs to migrate from one workstation to another.

Figure 5 shows a low-level flowchart for the data collection process (predefined process **340**, shown in **Figure 3**). Data collection processing

commences at step **500**. First, a test is made to determine the location of the nonvolatile media device, such as a tape or disk drive, used to store the data (step **505**). Next, environment variables are set (step **510**) in preparation for further processing. After environment variables have been set, the connectivity between the client workstation and the nonvolatile storage device is verified and a working list of hostnames is created (step **515**). After connectivity has been verified, information is retrieved from the workstations /home/local directory and other workstation specific information is gathered (step **520**). Next, network information is retrieved (step **525**) which includes retrieving key network configuration files, getting the status of the workstations' Ethernet adapters (step **530**), getting the IP addresses corresponding to the Ethernet adapters (step **535**), getting the default gateway addresses to the network (step **540**), getting the default netmasks (step **545**), and getting the domain information (step **550**) which includes the domain name and name server addresses. After the network information is retrieved, system information is retrieved (step **555**). Then tty information is retrieved from the old workstation (step **560**). After tty information has been retrieved, print queue information is retrieved (step **565**). Next, application specific data (previously identified in step **315** shown in **Figure 3**) is retrieved (step **570**). After application specific data has been retrieved, license information (previously identified in predefined process **310** shown in **Figure 3**) is retrieved. Finally, filesystem information is retrieved (step **580**). The information retrieved during the collection process shown in **Figure 5** is stored in a removable nonvolatile computer operable media, such as a diskette, tape, or CD-R.

Figure 6 is a low-level flowchart showing the detail involved in collecting application information (see predefined process **350** in **Figure 3**). After commencing the process (step **600**), the program is copied from the removable nonvolatile storage device to the individual workstation (step **610**). Application information is then collected from the workstation (step **620**). A loop construct is entered to collect the application information for each of the applications identified. If no more applications exist for which data is to be retrieved (decision **630**), "no" branch **635** is executed and processing is terminated at **690**. On the other hand, if more applications exist, "yes" branch **640** is executed to get the application information.

First, the application logical volume name is retrieved (step **650**). Next, the application mount point and filesystem name are retrieved (step

660). Finally, the application filesystem size is retrieved (step 670) before processing loops back (loop 680) to check if more applications have data to be retrieved at step 630. This loop continues until no more applications need to be retrieved, at which point "no" branch 635 is executed and processing is terminated at 690.

Figure 7 shows a low-level flowchart detailing the creation of a quality checklist (see predefined process 360 on Figure 3). Processing commences (step 700) whereupon a check is made to make sure a client hostname has been retrieved (step 705). Next, a check is made to ensure that a connection has been made between the workstation and the removable nonvolatile storage device (step 710). Networking files are then checked (step 715) to make sure they are correct. After networking files are checked, the restoration of the /home/local directory is checked (step 720). Next, passwords are checked to make sure they have been collected properly (step 725). After the passwords have been checked, the filesystems are checked (step 730). Next, the restoration of the /usr/local directory is checked (step 735). After the directory is checked, the ttys are checked and verified (step 740). Next, ADSM (IBM's ADSTAR Distributed Storage Manager) is checked for any included/excluded files (step 745). The application information that was previously identified and collected is then checked (step 750). Finally, the Ethernet and IP addresses for the workstation are checked (step 755). After all checks have been performed, the process ends at 790. While described as a sequential process, the processing described above could take place in a different order and some processing can be done at the same time (in parallel with) other processing. The steps shown in Figure 7 can be performed manually by an operator following an instruction sheet or may be performed automatically by a program designed to perform the above-described checks.

Figure 8 shows a low-level flowchart for duplicating the collected workstation data onto a different workstation (see predefined process 440 in Figure 4). First, a removable nonvolatile storage device is installed and/or configured to work with the workstation if such device is not already installed and configured (step 810). The removable nonvolatile computer medium onto which the captured data was saved is loaded in the installed and configured removable nonvolatile storage device in order to copy the previously captured user environment data (step 825). Next, the networking files are restored (step 830). Next, the unique system information that was previously captured is duplicated to the new

workstation (step 835). After the unique system information is duplicated, the home/local data is restored (step 840) and any ADSM files are restored (step 845). Next, the application data that was previously identified and captured is duplicated to the new workstation (step 850).

After application data is restored, the tty information that was captured from the old workstation is duplicated to the new workstation (step 855). Then the remote printer definitions that were captured from the old workstation are duplicated to the new workstation (step 860). After the printer definitions are duplicated, the workstation's IP address and workstation name (hostname) are assigned to the permanent hostname and IP address which is provided by the user (step 865). After the hostname and IP address changes have taken place, the Ethernet interfaces are checked and verified (step 870). Next, the file permissions of key files that have been duplicated onto the new workstation are modified to reflect the move from the old workstation to the new workstation (step 875). After the monitor timeout has been disabled (step 880), the license file data is modified (step 885) to reflect the duplication of licensed data, such as software products, from the old workstation to the new workstation. Finally, temporary files are cleaned up (step 890) before the processing ends at 895.

The above-outlined steps are for a UNIX environment. Alternative embodiments may have somewhat different processing steps due to differences in operating systems. In addition, the steps outlined above could be performed in a somewhat different order, however the order outlined above is preferred for a UNIX environment.

Figure 9 shows a low-level flowchart of details of modifying X-windows settings on the new workstation (see step **450** in **Figure 4**). First, a new window is created (step 910). Then the X-windows session is terminated (step 920), followed by setting the landscape (step 930) and portrait (step 940) settings of the new workstation's X-windows. After the settings have been changed, processing of the X-windows settings ends at 990.

Figure 10 illustrates a computer system **1001** which is a simplified example of a computer system capable performing the capturing and duplicating processing described herein. Computer system **1001** includes processor **1000** which is coupled to host bus **1005**. A level two (L2) cache memory **1010** is also coupled to the host bus **1005**. Host-to-PCI bridge **1015**

is coupled to main memory **1020**, includes cache memory and main memory control functions, and provides bus control to handle transfers among PCI bus **1025**, processor **1000**, L2 cache **1010**, main memory **1020**, and host bus **1005**. PCI bus **1025** provides an interface for a variety of devices including, for example, LAN card **1030**. PCI-to-ISA bridge **1035** provides bus control to handle transfers between PCI bus **1025** and ISA bus **1040**, universal serial bus (USB) functionality **1045**, IDE device functionality **1050**, power management functionality **1055**, and can include other functional elements not shown, such as a real-time clock (RTC), DMA control, interrupt support, and system management bus support. Peripheral devices and input/output (I/O) devices can be attached to various interfaces **1060** (e.g., parallel interface **1062**, serial interface **1064**, infrared (IR) interface **1066**, keyboard interface **1068**, mouse interface **1070**, and fixed disk (FDD) **1072**) coupled to ISA bus **1040**. Alternatively, many I/O devices can be accommodated by a super I/O controller (not shown) attached to ISA bus **1040**.

The BIOS **1080** is coupled to ISA bus **1040**, and incorporates the necessary processor executable code for a variety of low-level system functions and system boot functions. BIOS **1080** can be stored in any computer readable medium, including magnetic storage media, optical storage media, flash memory, random access memory, read only memory, and communications media conveying signals encoding the instructions (e.g., signals from a network). In order to attach computer system 1001 to a NIM server over a local area network, LAN card **1030** is coupled to PCI-to-ISA bridge 1035. Similarly, to connect to a NIM server using a telephone line connection, modem **1075** is connected to serial port **1064** and PCI-to-ISA Bridge **1035**.

While the computer system described in **Figure 10** is capable of executing the capturing and duplicating processes described herein, this computer system is simply one example of a computer system. Those skilled in the art will appreciate that many other computer system designs are capable of running the UNIX operating system (or any operating system, such as Windows 95/98/NT licensed by Microsoft Corporation or AIX or OS/2 licensed by IBM) and performing the processing described herein.

Figure 11 shows a hierarchy chart for processes involved in collecting and duplicating a workstation environment. System duplication **1100** is the highest level in the hierarchy chart. System duplication **1100** breaks down into two general processing categories: collection **1110**, which

collects the user environment data from the old workstation, and duplication **1120** which duplicates the information collected by collection **1110** onto a new workstation. Collection **1110** breaks down into four processes. Data collection **1130** collects the user environment data from the old workstation (for a flowchart depiction, see **Figure 5**). Workstation list **1140** includes a list of workstations that will have user environment data collected and duplicated. Collection of application data **1150** includes processing to collect application data from the workstation (for a flowchart depiction, see **Figure 6**). Finally, quality checklist **1160** includes processing to check whether the user environment data has been successfully collected (for a flowchart depiction, see **Figure 7**).

On the duplication **1120** side of the hierarchy chart, two processes are shown breaking down from duplication **1120**. First, duplication **1170** duplicates the user environment data collected from the old workstation onto the new workstation (for a flowchart depiction, see **Figure 8**). Second, X-Windows Settings **1180** sets the X-Windows settings in the new workstation (for a flowchart depiction, see **Figure 9**).

One of the preferred implementations of the invention is a client application, namely, a set of instructions (program code) in a code module which may, for example, be resident in the random access memory of the computer. Until required by the computer, the set of instructions may be stored in another computer memory, for example, in a hard disk drive, or in a removable memory such as an optical disk (for eventual use in a CD ROM) or floppy disk (for eventual use in a floppy disk drive), or downloaded via the Internet or other computer network. Thus, the present invention may be implemented as a computer program product for use in a computer. In addition, although the various methods described are conveniently implemented in a general purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would also recognize that such methods may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps.

While particular embodiments of the present invention have been shown and described, it will be obvious to those skilled in the art that, based upon the teachings herein, changes and modifications may be made without departing from this invention and its broader aspects and, therefore, the appended claims are to encompass within their scope all

such changes and modifications as are within the true scope of this invention.

CLAIMS

1. A method for duplicating a user environment in a first computer system, said method comprising:

executing an automated data collection tool on the first computer system to collect user environment data from the first computer system; and

storing the user environment data on a removable nonvolatile media.

2. The method of claim 1, wherein the collecting includes:

identifying attributes to include in the user environment data.

3. The method of claim 1 or claim 2, said method further comprising:

restoring the user environment data stored by the first computer system onto a second computer system.

4. The method of any preceding claim, wherein the first computer system includes a UNIX operating system.

5. The method of any preceding claim, wherein the collecting is performed for a plurality of users, each of the plurality of users having one or more accounts on the first computer system.

6. The method of any preceding claim, wherein the user environment data includes at least one of printer definitions, tty definitions, network interfaces, user passwords, and license information.

7. The method of any preceding claim, said method further comprising:

transporting the removable nonvolatile media from the first computer system to a second computer system;

loading the removable nonvolatile media in a device capable of reading the media; and

restoring the user environment data from the removable nonvolatile media to the second computer system.

8. An information handling system comprising:

one or more processors;

an operating system operable by the processors;

a memory accessible by the processors;

a removable nonvolatile storage device accessible by the processors;

a user environment duplication tool for duplicating user environment data in a first computer system, the tool including:

means for collecting user environment data from the first computer system, the collecting performed by a computer program; and

means for storing the user environment data on a removable nonvolatile media.

9. The information handling system of claim 8, wherein the collecting includes:

means for identifying attributes to include in the user environment data.

10. The information handling system of claim 8 or claim 9, further comprising:

means for restoring the user environment data stored by the first computer system onto a second computer system.

11. The information handling system of any of claims 8 to 10, wherein the first computer system includes a UNIX operating system.

12. The information handling system of any of claims 8 to 10, wherein the means for collecting is performed for a plurality of users, each of the plurality of users having one or more accounts on the first computer system.

13. The information handling system of any of claims 8 to 10, wherein the user environment data includes at least one of printer definitions, tty definitions, network interfaces, user passwords, and license information.

14. A computer program product stored on a computer operable medium, the computer program product programmed to duplicate a user environment in a first computer system, said computer program product comprising:

program code for collecting user environment data from the first computer system, the collecting performed by a computer program; and

program code for storing the user environment data on a removable nonvolatile media.

15. The computer program product of claim 14, wherein the collecting includes:

means for identifying attributes to include in the user environment data.

16. The computer program product of claim 14 or claim 15, further comprising:

means for restoring the user environment data stored by the first computer system onto a second computer system.

17. The computer program product of any of claims 14 to 16, wherein the means for collecting is performed for a plurality of users, each of the plurality of users having one or more accounts on the first computer system.

18. The computer program product of any of claims 14 to 16, wherein the user environment data includes at least one of printer definitions, tty definitions, network interfaces, user passwords, and license information.

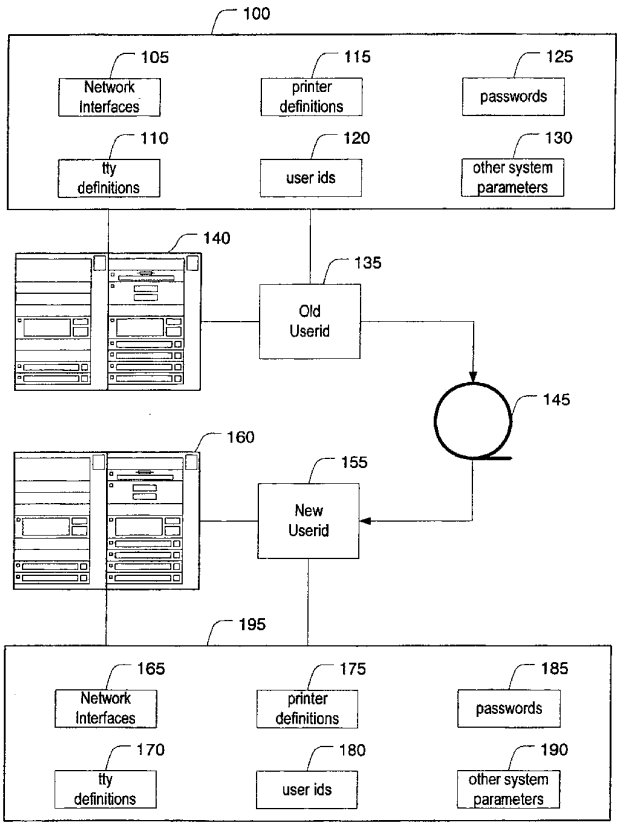

100
105
Network Interfaces
115
printer definitions
125
passwords
110
tty definitions
120
user ids
130
other system parameters
140
135
Old Userid
145
160
155
New Userid
195
165
Network Interfaces
175
printer definitions
185
passwords
170
tty definitions
180
user ids
190
other system parameters

Figure 1

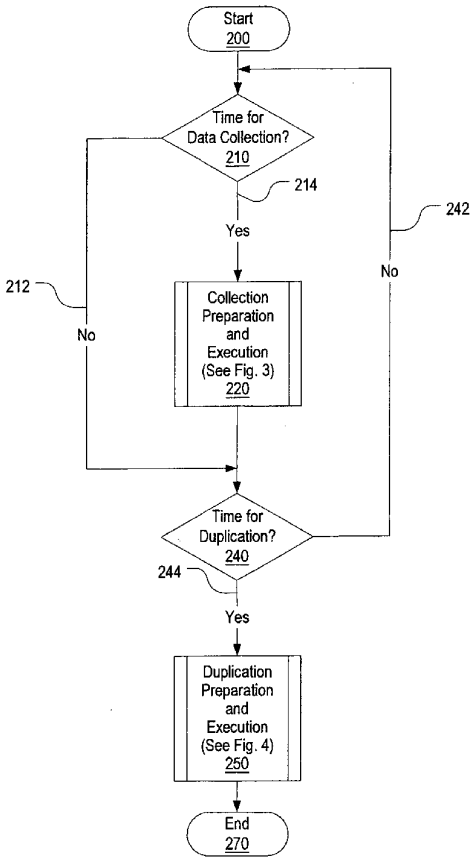

Start
200
Time for
Data Collection?
210
214
Yes
242
No
212
No
Collection
Preparation
and
Execution
(See Fig. 3)
220
Time for
Duplication?
240
244
Yes
Duplication
Preparation
and
Execution
(See Fig. 4)
250
End
270

Figure 2

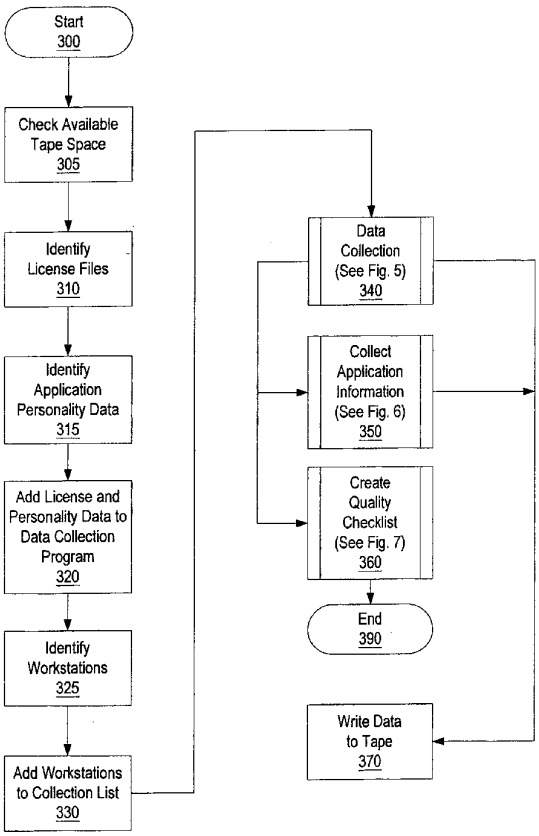

Start
300
Check Available Tape Space
305
Identify License Files
310
Identify Application Personality Data
315
Add License and Personality Data to Data Collection Program
320
Identify Workstations
325
Add Workstations to Collection List
330
Data Collection (See Fig. 5)
340
Collect Application Information (See Fig. 6)
350
Create Quality Checklist (See Fig. 7)
360
End
390
Write Data to Tape
370

Figure 3

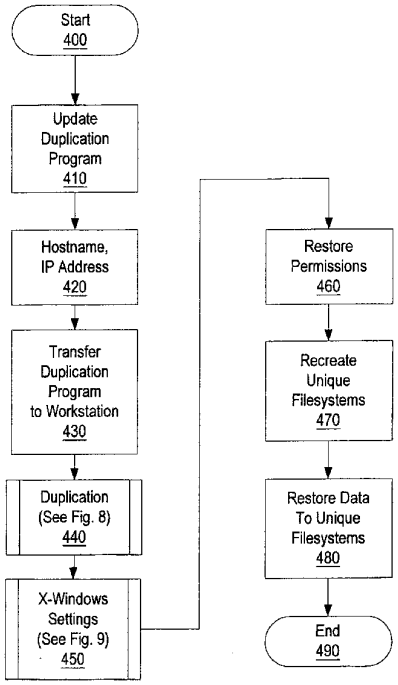

Start
400
Update Duplication Program
410
Hostname, IP Address
420
Transfer Duplication Program to Workstation
430
Duplication (See Fig. 8)
440
X-Windows Settings (See Fig. 9)
450
Restore Permissions
460
Recreate Unique Filesystems
470
Restore Data To Unique Filesystems
480
End
490

Figure 4

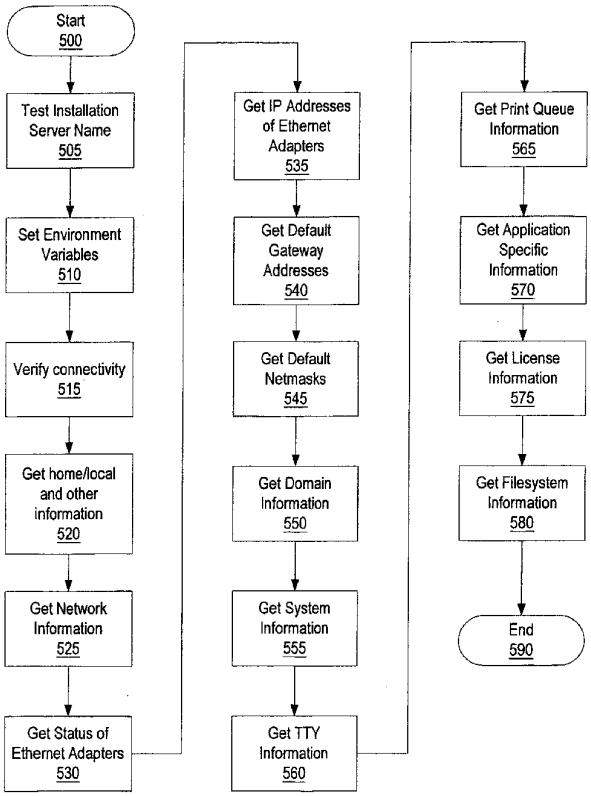

Start
500
Test Installation Server Name
505
Set Environment Variables
510
Verify connectivity
515
Get home/local and other information
520
Get Network Information
525
Get Status of Ethernet Adapters
530
Get IP Addresses of Ethernet Adapters
535
Get Default Gateway Addresses
540
Get Default Netmasks
545
Get Domain Information
550
Get System Information
555
Get TTY Information
560
Get Print Queue Information
565
Get Application Specific Information
570
Get License Information
575
Get Filesystem Information
580
End
590

Figure 5

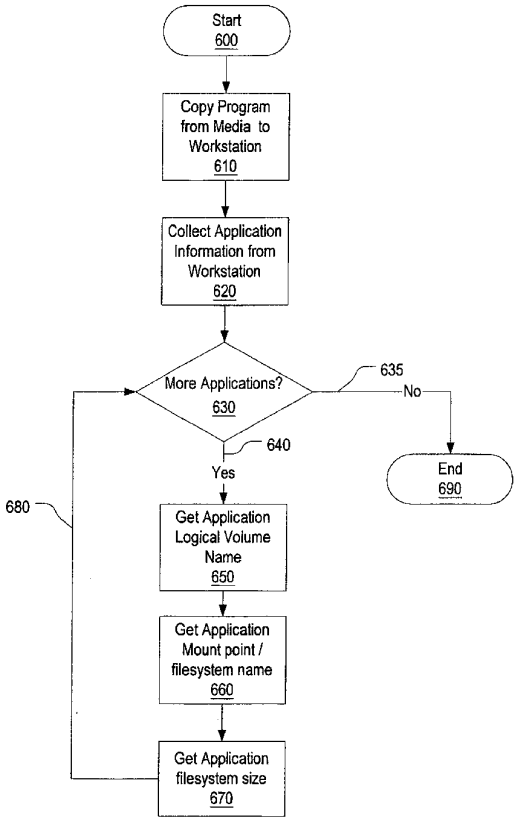

Start
600
Copy Program from Media to Workstation
610
Collect Application Information from Workstation
620
More Applications?
630
635
No
End
690
640
Yes
680
Get Application Logical Volume Name
650
Get Application Mount point / filesystem name
660
Get Application filesystem size
670

Figure 6

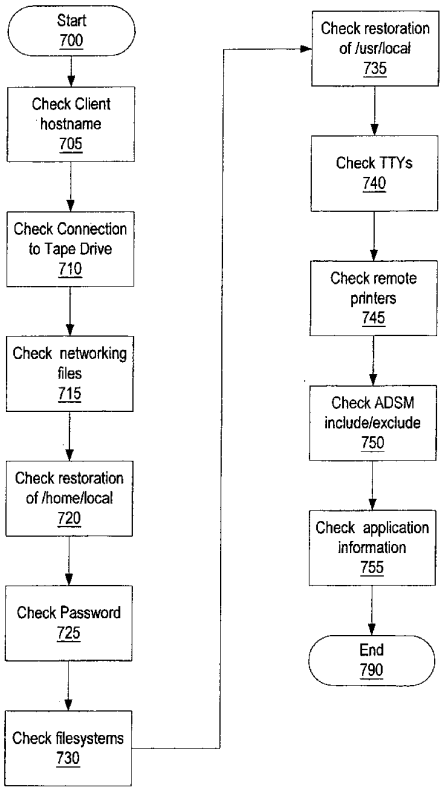

Start
700
Check Client hostname
705
Check Connection to Tape Drive
710
Check networking files
715
Check restoration of /home/local
720
Check Password
725
Check filesystems
730
Check restoration of /usr/local
735
Check TTYs
740
Check remote printers
745
Check ADSM include/exclude
750
Check application information
755
End
790

Figure 7

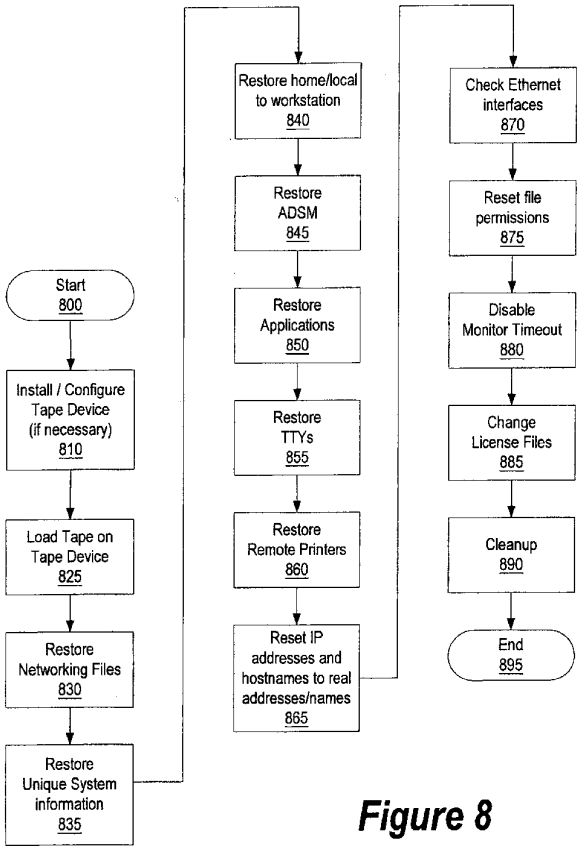

Start
800
Install / Configure Tape Device (if necessary)
810
Load Tape on Tape Device
825
Restore Networking Files
830
Restore Unique System information
835
Restore home/local to workstation
840
Restore ADSM
845
Restore Applications
850
Restore TTYs
855
Restore Remote Printers
860
Reset IP addresses and hostnames to real addresses/names
865
Check Ethernet interfaces
870
Reset file permissions
875
Disable Monitor Timeout
880
Change License Files
885
Cleanup
890
End
895

Figure 8

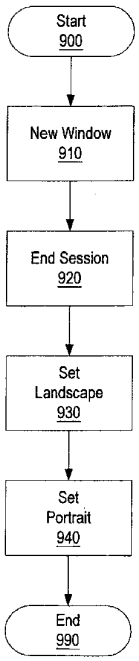

Start
900
New Window
910
End Session
920
Set
Landscape
930
Set
Portrait
940
End
990

Figure 9

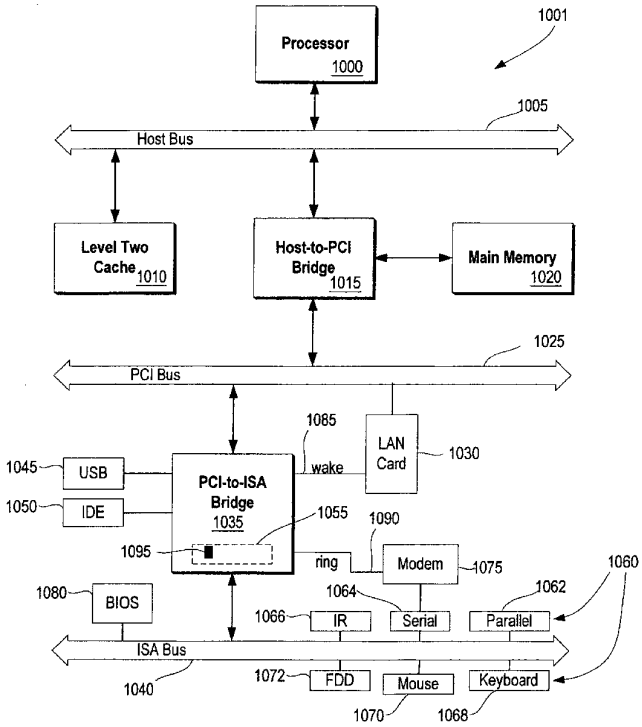

Processor
1000
1001
1005
Host Bus
Level Two
Cache 1010
Host-to-PCI
Bridge
1015
Main Memory
1020
1025
PCI Bus
1085
LAN
Card
1030
wake
1045
USB
1050
IDE
PCI-to-ISA
Bridge
1035
1055
1090
1095
ring
Modem
1075
1060
1080
BIOS
1064
1062
1066
IR
Serial
Parallel
ISA Bus
1040
1072
FDD
Mouse
Keyboard
1070
1068

Figure 10

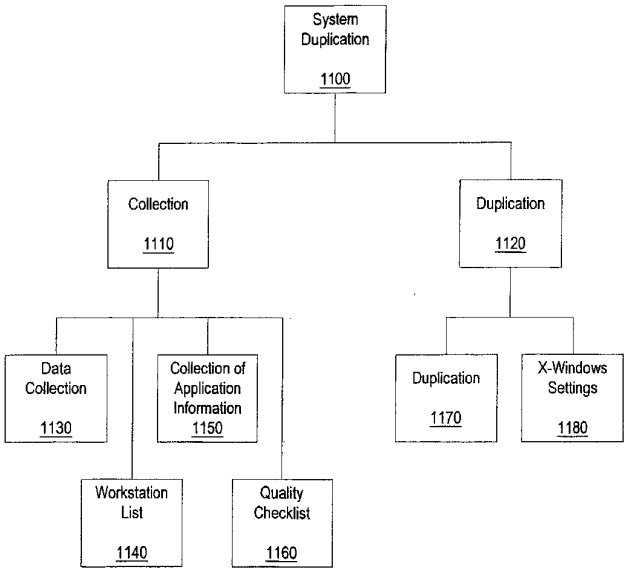

System Duplication
1100
Collection
1110
Duplication
1120
Data Collection
1130
Collection of Application Information
1150
Duplication
1170
X-Windows Settings
1180
Workstation List
1140
Quality Checklist
1160

Figure 11

【国際公開パンフレット（コレクトバージョン）】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 October 2002 (17.10.2002)

PCT

(10) International Publication Number
WO 02/082266 A3

(51) International Patent Classification[7]: G06F 9/46, 11/14, 11/20

(21) International Application Number: PCT/GB02/01415

(22) International Filing Date: 25 March 2002 (25.03.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/826,608 5 April 2001 (05.04.2001) US

(71) Applicant: INTERNATIONAL BUSINESS MACHINES CORPORATION [US/US]; New Orchard Road, Armonk, NY 10504 (US).

(71) Applicant (for MG only): IBM UNITED KINGDOM LIMITED [GB/GB]; PO Box 41, North Harbour, Portsmouth, Hampshire PO6 3AU (GB).

(72) Inventors: HAMILTON, Rick; 1532 Dairy Road, Charlottesville, VA 22903 (US). LIPTON, Steven; 2609 Maywood Court, Flower Mound, TX 75028 (US).

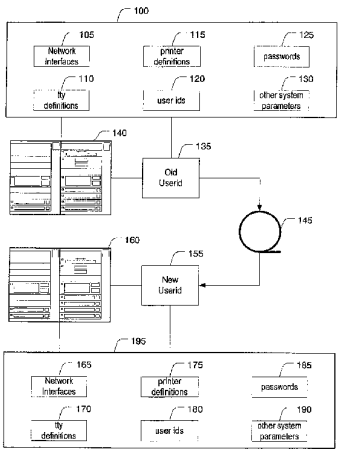
(74) Agent: LITHERLAND, David, Peter; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester, Hampshire SO21 2JN (GB).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR,

[Continued on next page]

(54) Title: COLLECTING AND RESTORING USER ENVIRONMENT DATA USING REMOVABLE STORAGE



(57) Abstract: A data collection program collects data from a user's workstation and captures the user environment data, including user settings and program application data. The user environment data is stored on a removable nonvolatile storage media for duplication processing. The stored user environment data is processed by a duplication process to duplicate the user environment data from the old workstation onto a new workstation or for recovery from a catastrophic system failure. A variety of user environment settings, not traditionally captured and restored by traditional backup software, are captured and restored. For example, licensing information and application personality data is identified, stored, and recovered along with other user-specific information such as hostnames, IP addresses, and the like.

WO 02/082266 A3

WO 02/082266 A3

GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*
before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

(88) Date of publication of the international search report:
24 July 2003

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Internati Application No
PCT/GB 02/01415

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/46 G06F11/14 G06F11/20

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6 182 212 B1 (CHALLENER DAVID CARROLL ET AL) 30 January 2001 (2001-01-30) abstract; figures 2,3 column 3, line 1 -column 10, line 46 ---	1-18
X	HASSEL D: "Please Back-Up DNA Before Cloning" STORAGESOFT JOINT WHITE PAPER, XX, XX, December 1999 (1999-12), pages 1-8, XP002202161 page 4, paragraph 5 -page 7, paragraph 5 ---	1-18
A	EP 0 957 616 A (SUN MICROSYSTEMS INC) 17 November 1999 (1999-11-17) the whole document -----	1-18

☐ Further documents are listed in the continuation of box C. ☒ Patent family members are listed in annex.

* Special categories of cited documents :

'A' document defining the general state of the art which is not considered to be of particular relevance
'E' earlier document but published on or after the international filing date
'L' document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
'O' document referring to an oral disclosure, use, exhibition or other means
'P' document published prior to the international filing date but later than the priority date claimed

'T' later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
'X' document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
'Y' document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
'&' document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
28 May 2003	10/06/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Bozas, I

INTERNATIONAL SEARCH REPORT

ation on patent family members

Interne Application No
PCT/GB 02/01415

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 6182212	B1	30-01-2001	NONE	
EP 0957616	A	17-11-1999	US 6119157 A	12-09-2000
			EP 0957616 A2	17-11-1999
			JP 2000067022 A	03-03-2000

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,OM,PH,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TN,TR,TT,TZ,UA,UG,UZ,VN,YU,ZA,ZM,ZW

(72)発明者 ハミルトン、リック
アメリカ合衆国 バージニア州 ２２９０３、シャーロッツビル、デイリー ロード １５３２
(72)発明者 リプトン、スティーブン
アメリカ合衆国 テキサス州 ７５０２８、フラワー マウンド、メイウッド コート ２６０９

Ｆターム(参考) 5B076 AA02

【要約の続き】
リティ・データをも特定し、格納し、復元している。
【選択図】 図１