(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2005-44360
(P2005-44360A)
(43)公開日　平成17年2月17日(2005.2.17)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 9/44	G06F	9/06	620Z	5B076
G06F 12/00	G06F	12/00	511A	5B082

審査請求 未請求　請求項の数 34　OL　（全 27 頁）

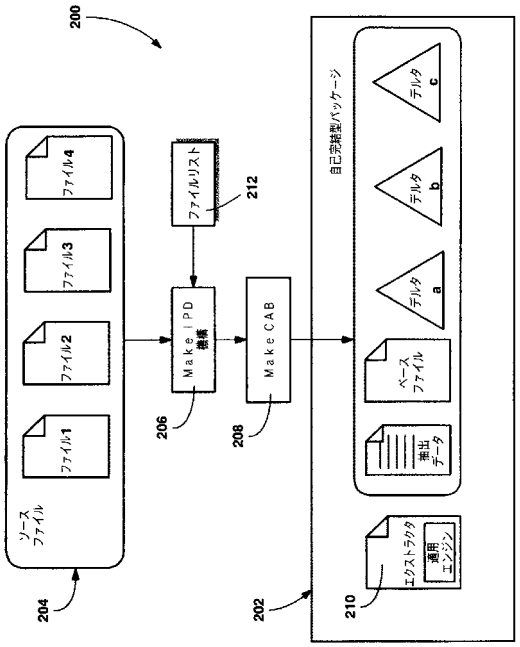
(21)出願番号　特願2004-213492(P2004-213492)
(22)出願日　平成16年7月21日(2004.7.21)
(31)優先権主張番号　10/633,375
(32)優先日　平成15年7月21日(2003.7.21)
(33)優先権主張国　米国(US)

(71)出願人　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　アレクサンダー　ブイ．ペトロフ
アメリカ合衆国　９８００５　ワシントン州　ベルビュー　１１８　アベニュー　サウスイースト　３２４　ナンバー３３

最終頁に続く

(54)【発明の名称】データのパッケージ内デルタ圧縮（ｉｎｔｒａ－ｐａｃｋｅｔｄｅｌｔａｃｏｍｐｒｅｓｓｉｏｎ）のためのシステムおよび方法

(57)【要約】
【課題】　コンピュータシステムを更新するファイルの組のファイルデータを自己完結型パッケージで提供しデルタ圧縮を介してパッケージサイズを大幅に低減する。
【解決手段】　構築機構は、配布するファイルを検査しファイルおよびデルタを含む自己完結型パッケージを生成。可能な様々なベースファイルおよびデルタファイルのサイズから有向グラフが構築され最小スパニングツリー計算によって最も小さいパッケージをもたらすファイルを選択。ベースファイルは複数のデルタを適用させて複数のファイルを合成できる。任意選択でパッケージとともに提供されるクライアント抽出機構はマニフェストによって指示されたようにパッケージの内容を処理してベースファイルおよび含まれるデルタから対象ファイルを合成する。
【選択図】　図２



【特許請求の範囲】
【請求項１】
コンピューティング環境において
複数のソースファイルに対応する情報を受信するステップと、
第１のソースファイルをベースファイルとして選択するステップと、
デルタを前記第１のソースファイルおよび第２のソースファイルから生成するステップと、
前記ベースファイルおよび前記デルタを自己完結型パッケージにひとまとめにするステップと
を含むことを特徴とする方法。
【請求項２】
前記第２のソースファイルに対応する対象ファイルを前記ベースファイルおよび前記デルタから合成するようクライアントエクストラクタに指示するデータをひとまとめにするステップをさらに含むことを特徴とする請求項１に記載の方法。
【請求項３】
それによってクライアントエクストラクタが前記第２のソースファイルに対応する対象ファイルを前記ベースファイルおよび前記デルタから合成することができる少なくとも１つのファイル名を設定するステップをさらに含むことを特徴とする請求項１に記載の方法。
【請求項４】
前記第１のソースファイルおよび前記第２のソースファイルは同じファイルの異なるバージョンのものではないことを特徴とする請求項１に記載の方法。
【請求項５】
前記第１のソースファイルおよび前記第２のソースファイルは同じファイルの異なる言語の翻訳版ではないことを特徴とする請求項１に記載の方法。
【請求項６】
前記第１のソースファイルおよび前記第２のソースファイルは同じファイルの異なる言語の翻訳版であることを特徴とする請求項１に記載の方法。
【請求項７】
前記第１のソースファイルを前記ベースファイルとして選択するステップは、パッケージサイズの考慮に基づいて前記ソースファイルを選択するステップを含むことを特徴とする請求項１に記載の方法。
【請求項８】
ソースファイルの複数の可能な組合せに基づいてファイルサイズの有向グラフを構築するステップと、前記有向グラフでの情報に基づいて前記第１のソースファイルを選択するステップとをさらに含むことを特徴とする請求項７に記載の方法。
【請求項９】
前記第１のソースファイルを前記ベースファイルとして選択するステップは、最小スパニングツリーなどのアルゴリズムを前記有向グラフに適用するステップを含むことを特徴とする請求項８に記載の方法。
【請求項１０】
前記第１のソースファイルを前記ベースファイルとして選択するステップは、可能なデルタのサイズを計算するステップと、前記サイズに基づいて前記第１のソースファイルを選択するステップとを含むことを特徴とする請求項１に記載の方法。
【請求項１１】
前記パッケージを受信者に提供し、前記受信者は前記デルタを前記第１のソースファイルに適用して前記第２のソースファイルを合成するステップをさらに含むことを特徴とする請求項１に記載の方法。
【請求項１２】
請求項１に記載の方法を実行するコンピュータ実行可能命令を有することを特徴とする

コンピュータ可読媒体。
【請求項１３】
コンピューティング環境において
少なくとも１つのベースファイルおよび複数のデルタを含むパッケージを受信するステップと、
前記パッケージ内のデルタをベースファイルに適用して対象ファイルを合成するステップと
を含むことを特徴とする方法。
【請求項１４】
前記デルタを前記ベースファイルに適用するステップは、前記デルタを前記パッケージに含まれるベースファイルに適用するステップを含むことを特徴とする請求項１３に記載の方法。
【請求項１５】
前記デルタを前記ベースファイルに適用するステップは、前記デルタを別のデルタおよび別のベースファイルから合成されたベースファイルに適用するステップを含むことを特徴とする請求項１３に記載の方法。
【請求項１６】
データファイルを解釈して各デルタをどのベースファイルに適用するかを決定するステップをさらに含むことを特徴とする請求項１３に記載の方法。
【請求項１７】
前記データファイルは、特定のデルタファイルが適用される特定のベースファイルを識別する指示を含む１組の指示を含むことを特徴とする請求項１４に記載の方法。
【請求項１８】
セットアッププログラムを実行するステップをさらに含むことを特徴とする請求項１３に記載の方法。
【請求項１９】
各デルタを対応するベースファイルに適用した後、前記セットアッププログラムを実行することを特徴とする請求項１８に記載の方法。
【請求項２０】
前記デルタを一時ディレクトリから削除するステップをさらに含むことを特徴とする請求項１３に記載の方法。
【請求項２１】
別のデルタを前記合成された対象ファイルに適用して別の対象ファイルを合成するステップをさらに含むことを特徴とする請求項１３に記載の方法。
【請求項２２】
少なくとも２つのデルタを共通のベースファイルに適用して少なくとも２つの対象ファイルを合成するステップをさらに含むことを特徴とする請求項１３に記載の方法。
【請求項２３】
請求項１３に記載の方法を実行するコンピュータ実行可能命令を有することを特徴とするコンピュータ可読媒体。
【請求項２４】
ベースファイルを含む第１のデータの組と、
前記ベースファイルとともにひとまとめにされ、前記ベースファイルに適用されるとき、対象ファイルを合成するように構成されたデルタファイルを含む第２のデータの組と
を含むことを特徴とするデータ構造を格納するコンピュータ可読媒体。
【請求項２５】
別のデルタファイルを含む第３のデータの組をさらに含むことを特徴とする請求項２４に記載のデータ構造。
【請求項２６】
前記他のデルタは、前記ベースファイルに適用されると別の対象ファイルを合成するよ

うに構成されていることを特徴とする請求項２４に記載のデータ構造。
【請求項２７】
前記他のデルタは、前記対象ファイルに適用されると別の対象ファイルを合成するように構成されていることを特徴とする請求項２４に記載のデータ構造。
【請求項２８】
前記データ構造をソースからクライアントの受信者に送信する手段をさらに含むことを特徴とする請求項２４に記載のデータ構造。
【請求項２９】
抽出プログラムに指示するデータを含む第３のデータの組をさらに含むことを特徴とする請求項２４に記載のデータ構造。
【請求項３０】
抽出プログラムを含む第３のデータの組をさらに含むことを特徴とする請求項２４に記載のデータ構造。
【請求項３１】
前記抽出プログラムに指示するデータを含む第４のデータの組をさらに含むことを特徴とする請求項３０に記載のデータ構造。
【請求項３２】
ベースファイルでもデルタでもないファイルを含む第３のデータの組をさらに含むことを特徴とする請求項２４に記載のデータ構造。
【請求項３３】
ベースファイルでもデルタでもない前記ファイルが圧縮されることを特徴とする請求項３２に記載のデータ構造。
【請求項３４】
コンピューティング環境において
デルタを適用することによって第２のソースファイルをそこから導出することができるベースファイルとして第１のソースファイルを選択する手段と、
前記ベースファイルおよび前記データを自己完結型パッケージにひとまとめにする手段と
を含むことを特徴とするシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は一般にコンピュータシステムに関し、より詳細にはパッケージ化されたコンピュータファイルに関する。
【背景技術】
【０００２】
ソフトウェアベンダーが新しい製品リリースや比較的大きいアップグレードなど１つまたは複数のファイルの組をその顧客に提供したい場合、ファイルをアーカイブにマージして、関連の内容の単一のパッケージを作成することができる。パッケージは一般に、セットとして使用するデータファイルの何らかの集まりである。実行されるとパッケージの内容を抽出して以前マージされた１組のファイルに戻す実行可能コードを追加することによって、アーカイブを自己抽出型アーカイブ（ｓｅｌｆ－ｅｘｔｒａｃｔｉｎｇ　ａｒｃｈｉｖｅ）にすることが多い。また、一般にちょうど抽出されたファイルのうちの１つを実行することによって、自己抽出型コードはセットアップ手順を開始することもでき、次に顧客のコンピュータの適切な場所にファイルがコピーされる。セットアップ手順が終了すると、自己抽出型コードは、抽出されたファイルを削除し、次いで終了する。ほとんどの場合、これによって製品の特徴または最新版全体を単一のファイルオブジェクトとして取得することができ、これを直接実行して製品の内容にアクセスしたり、それをインストールしたりできるようになる。
【０００３】

アーカイブプロセスは、通常、ある種のデータ圧縮を使用してアーカイブのサイズを低減する。これによって、特に大きいアーカイブの場合には、配布および取得のコストが低減する。こうした圧縮技術の１つでは、ファイルを別々に圧縮し、必要に応じて任意の個々のファイルへのアクセス権が顧客に提供される。こうしたパッケージのサイズは一般に、含まれている各ファイルの圧縮されたサイズに抽出コードのサイズを加えた合計である。実行されると、パッケージは圧縮ファイルのそれぞれを一時場所に抽出し、ユーザは各ファイルをそこからシステムのディレクトリ内の適切な場所にコピーすることができる。

【０００４】

セットアップ手順が自動的に実行されて抽出されたファイルをインストールする場合など、個々のファイルアクセスが必要ではないパッケージでは、パッケージ圧縮は、キャビネット（またはＣＡＢ）ファイルの使用によってさらに向上する。ＣＡＢファイルでは、圧縮前にファイルが本質的に互いに結び付けられ（連結され）ている。これによって、ＬＺベースのエンコーダ（ＬｅｍｐｅｌおよびＺｉｖによる研究を発案した後に名付けられたよく知られているタイプのディクショナリエンコーダである）による符号化の効率が向上する。というのは、ＬＺ符号化では、入力データストリームの圧縮は、履歴として知られている入力データストリームの前の部分に依存し、ファイルの連結によって、使用可能な履歴データの量が増加するからである。圧縮ファイルを使用すると、抽出中に圧縮データが解凍され、したがってファイルは、セットアップ手順が起動してそれらのファイルに作用する前に元の形になることに留意されたい。

【０００５】

圧縮技術を使用しても、パッケージは、例えばネットワークを介して便利に送信できるデータ量に対して大きい可能性がある。広域帯ネットワークアクセスを有していない顧客の場合、大きいサイズのパッケージでは、こうしたパッケージをダウンロードするのは非現実的、あるいは少なくとも非常に不便となる。データをダウンロードするのに、長距離または長時間接続使用料を支払わなければならない顧客もいれば、ダウンロードできるデータ量の制限、および／またはセッションの接続時間の制限があるユーザもいる。他の顧客は、モデムを介して大きいファイルをダウンロードすることを苦にしないにすぎない。大きいファイルのダウンロードはさらに、セッションを終了させるネットワーク接続の問題を受けやすい。こうした顧客にとっては、大きいパッケージの配布が問題となっている。

【０００６】

また、パッケージベンダーは、提供するダウンロードのサイズに応じてコストがかかる。例えば、大きいファイルを配布することによって、費用のかかる、かなりの量のネットワークサーバ設備が必要となる。ＣＤ－ＲＯＭは、多くの場合、ベンダーの負担で一部の顧客に提供される。インターネットを介した配布でさえ、より大きいパッケージを送信すると増加する変動費がかかる。

【０００７】

送信する必要のあるデータの量を低減する、最新版を提供する改良された方法が特許文献１に記載されている。この手法では、クライアント（顧客）コンピュータは、まず、セットアッププログラムを含む初期セットアップパッケージ、およびソフトウェア製品のインストールに必要なファイルのリストをセットアップサーバから取得する。次いでクライアントコンピュータ上のセットアッププログラムは、インストールに必要な現在または以前の何らかのバージョンのこうしたファイルがクライアントコンピュータ上にすでに存在しているかどうかを決定し、クライアントコンピュータの更新に必要なファイルの要求リストをコンパイルする。クライアントコンピュータは、要求リストをダウンロードサーバに送信する。ダウンロードサーバは、更新ファイルおよびパッチの集まりを維持し、更新に必要な適切な１組のファイルをクライアントに送信することによって要求リストに応答する。１つまたは複数のファイルは、パッチの形とすることができる。パッチは、以前のバージョンのファイルまたはより新しいバージョンのそのファイルから導出される小さいデータファイルである。パッチは、すでにクライアントコンピュータにある以前のファイ

ルバージョンのコピーに適用して新しいバージョンを生成することができ、これによって新しいバージョンを全部ダウンロードする必要がなくなる。

【０００８】

こうしたデータ圧縮はクライアントがダウンロードしなければならないデータの量を大幅に低減することはできるが、この技術にはいくつかの欠点もある。１つには、こうしたバイナリパッチは、デルタ圧縮とも呼ばれ、所与のクライアントコンピュータでファイルのどの表現がすでに使用可能であるかをベンダーが知っている（または間違いなく推定できる）ときにのみ動作する。これは、例えばＣＤ－ＲＯＭや他の一定の配布方式で常に可能であるとは限らない。複数のファイルを異なるバージョンごとにアーカイブに含め、そのうちの１つを特定のクライアントが有する所与のバージョンのファイルに適用できるようにすることによって、ベンダーの顧客が使用している可能性のある様々なバージョンのファイルを単一の一般的なアーカイブが更新するようにすることは可能であることに留意されたい。しかし、これもまた非効率で、パッケージを介して更新する必要のある（およそ何百、あるいは何千もの）大量のファイルが存在するいくつかの状況においては非現実的であり、または管理不能である。デルタ圧縮を介して達成された節約のほとんどは、多数のファイルの複数のバージョンを扱う必要があることによって失われることになる。

【０００９】

【特許文献１】米国特許第６，４９３，８７１号明細書
【特許文献２】米国特許第６，４９６，９７４号明細書
【特許文献３】米国特許第６，４９３，８７１号明細書
【特許文献４】米国特許第６，４６６，９９９号明細書
【特許文献５】米国特許第６，４４９，７６４号明細書
【特許文献６】米国特許第６，２４３，７６６号明細書
【特許文献７】米国特許第６，２１６，１７５号明細書

【発明の開示】

【発明が解決しようとする課題】

【００１０】

要約すれば、従来の圧縮では、結果として生じる圧縮パッケージのサイズは、簡単に配布するには依然として大きすぎるため、費用がかかり、かつ／または多くのユーザおよびベンダーには不向きである。同時に、各顧客のサーバで動的カスタマイズを必要としない自己完結型パッケージを使用する必要がある、または使用したい顧客および／またはベンダーにとって、これまでデルタ圧縮は十分には役立たなかった。高効率で、実質的にパッケージに内蔵されるソフトウェア製品データを提供する方法が必要である。

【課題を解決するための手段】

【００１１】

簡単に言えば、本発明は、自己完結型パッケージでデータを提供し、デルタ圧縮を介してデータの量が大幅に低減されるシステムおよび方法を提供する。このために、（関連データの任意の組とすることができる）ファイルの１つのグループを通常の（圧縮された）形式でパッケージ化し、第２のグループを第１または第２のグループの出力から導出されたデルタとして表す。したがってパッケージは、同じパッケージ内の他のファイルに、または同じパッケージから以前導出された他のファイルに適用されるデルタの集まりを含む。

【００１２】

本発明のパッケージ内デルタシステムおよび方法は、構築機構およびクライアント機構という２つの主要な機構を含んでいる。一般に、構築機構は、配布する１組のファイル（対象ファイル）を検査して、最適化された自己完結型パッケージ内デルタパッケージを生成し、クライアント構成要素は、パッケージの内容に対する処理を行い、含まれているデルタから対象ファイルを合成する。

【００１３】

パッケージ内デルタ機構は、ベンダー側では、パッケージ内のファイル間の類似点を利

用して総パケットサイズを低減する。これは、最新版の場合には特にうまく働く。というのは、最新版は、通常複数のバイナリファイルを含んでおり、これらのバイナリファイルは、しばしば共有されている共通のソースコードまたはライブラリをいくつか有することによって互いに関連付けられるからである。最新版では、異なる言語ごとに同等の様々なファイルを提供するなど、異なるシナリオごとに同等のファイルが提供されることも多い。

【００１４】

一実装形態では、自己完結型デルタ圧縮パッケージは、ファイルのほとんどがベースファイルおよびデルタから合成されるように構成されており、単一のベースファイルは複数のデルタをそれに適用させて複数のファイルを合成することができ、かつ／または任意のベースファイル自体は別のベースファイルおよびデルタから前もって合成されていてもよい。したがって、パッケージ内デルタ機構を介して構築されたパッケージは、それ自体何らかの方法で圧縮され得る単一のベースファイルに加えて、そのベースファイルのコピーを変換して他のファイルを合成する任意の数のデルタを含み得る。デルタは、前の合成から出力されたファイルに適用することもでき、それによって対象ファイルごとに最適なソース選択が可能になる。受信側ですでに使用可能なファイルについて想定できる場合、既存のファイルのコピー、またはコピーに適用されるデルタを１組の対象ファイルに追加して完全なパッケージを再作成し、それによってさらにパッケージサイズを低減することができる。

【００１５】

一実装形態では、パッケージ内デルタ機構は、パッケージ内デルタ圧縮パッケージのデルタを自動的に生成する。このために、パッケージを介して提供する必要のある対象ファイルのリストが与えられている場合、機構は、各ファイルを合成する様々な可能性を探り、可能なデルタを作成し、結果として生じるファイルサイズを検査してベースファイルおよびデルタのどれが最も小さいパッケージサイズをもたらすかを決定すると同時に、クライアントで後に抽出されると対象ファイルが完全に再作成されるようにする。抽出を行うのに必要な指示（例えば適切な順序）はマニフェストファイルに保存され、パッケージングの終了に必要な情報は命令(directive)ファイルなどに保存され、したがって、例えばキャビネットファイルを生成して、ベースファイル、デルタ、およびマニフェストファイルを、抽出を行うためにクライアントが実行する実行可能抽出ツールなど他の任意の必要なファイルとともに含むようにすることができる。

【００１６】

クライアント側で、パッケージ内デルタセルフエクストラクタフレームワーク（intra-package delta self-extractor framework）は、一時ディレクトリを作成し、従来のパッケージのようにパッケージの内容を展開し、しかしセットアッププログラムを開始する前に、本発明の一態様に従って追加の処理を行う実行可能抽出ツールを含む。このために、抽出ツールは、マニフェストを解釈してデルタを適用して、パッケージ内のベースファイルから、またはそれ自体が以前合成されたベースファイルから対象ファイル（の一部）を合成する。セットアッププログラムには、解凍され、かつ／または合成された対象ファイルの完全な組しか見えないように、抽出ツールは、セットアッププログラムを開始する前にデルタファイルを破棄することができる。圧縮されたベースファイルに基づいてデルタとして含まれる他のファイルとともに、いくつかのファイルを（圧縮された）ベースファイルとしてパッケージ内に含めることによって、パッケージのサイズを大幅に低減できることは容易に理解できる。パッケージ内デルタを使用するパッケージの形式およびアプリケーションは、顧客がセットアッププロセスを開始することなくパッケージの内容を抽出するだけの場合でさえも、既存の構成要素の観点からすると、従来の自己完結型パッケージと同じであることに留意されたい。これは、パッケージ内デルタの内容のデルタ処理が自己抽出型実行可能コードによって透過的に行われるためである。

【００１７】

他の利点は、以下の詳細な説明を図面と併せ読めば明らかになる。

【発明を実施するための最良の形態】
【００１８】
（例示的オペレーティング環境）
図１は、本発明を実施するのに適したコンピューティングシステム環境１００の例を示している。コンピューティングシステム環境１００は、適したコンピューティング環境の一例にすぎず、本発明の使用または機能の範囲に関する限定を示唆するものではない。また、コンピューティング環境１００を、動作環境１００の例に示した構成要素のいずれか１つ、またはその組合せに関連する依存性または必要条件を有しているものと解釈すべきではない。
【００１９】
本発明は、他の多くの汎用または専用コンピューティングシステム環境または構成で動作可能である。本発明との使用に適したよく知られているコンピューティングシステム、環境、および／または構成の例には、それだけには限定されないが、パーソナルコンピュータ、サーバーコンピュータ、ハンドヘルドまたはラップトップ装置、タブレット装置、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、ビデオゲーム、セル式または他の電話製品、プログラム可能家庭用電化製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、上記の任意のシステムまたは装置を含む分散コンピューティング環境などがある。
【００２０】
本発明は、コンピュータによって実行されるプログラムモジュールなどのコンピュータ実行可能命令の一般的な文脈で説明することができる。一般にプログラムモジュールは、特定のタスクを実行する、または特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、構成要素、データ構造などを含む。また、本発明は、タスクが通信ネットワークによってリンクされているリモート処理装置によって実行される分散コンピューティング環境でも実施することができる。分散コンピューティング環境では、プログラムモジュールを、メモリ記憶装置を含むローカルおよび／またはリモートのコンピュータ記憶媒体に置くことができる。
【００２１】
図１を参照すると、本発明を実施するシステムの例は、汎用コンピューティング装置をコンピュータ１１０の形で含んでいる。コンピュータ１１０の構成要素は、それだけには限定されないが、処理ユニット１２０、システムメモリ１３０、およびシステムメモリを含む様々なシステム構成要素を処理ユニット１２０に結合するシステムバス１２１を含む。システムバス１２１は、様々なバスアーキテクチャのうちの任意のものを使用するメモリバスまたはメモリコントローラ、周辺バス、およびローカルバスを含むいくつかのタイプのバス構造のうちどんなものでもよい。こうしたアーキテクチャには、それだけには限定されないが一例として、業界標準アーキテクチャ（ＩＳＡ）バス、マイクロチャネルアーキテクチャ（ＭＣＡ）バス、拡張ＩＳＡ（ＥＩＳＡ）バス、ビデオ電子装置規格化協会（ＶＥＳＡ）ローカルバス、およびメザニンバスとしても知られている周辺部品相互接続（ＰＣＩ）バスなどがある。
【００２２】
コンピュータ１１０は、一般に様々なコンピュータ可読媒体を含む。コンピュータ可読媒体は、コンピュータ１１０からアクセスできる使用可能な任意の媒体とすることができ、揮発性および不揮発性媒体、リムーバブルおよび非リムーバブル媒体を含む。コンピュータ可読媒体は、それだけには限定されないが一例として、コンピュータ記憶媒体および通信媒体を含み得る。コンピュータ記憶媒体には、コンピュータ可読命令、データ構造、プログラムモジュール、他のデータなど、情報を記憶するための任意の方法または技術で実施される揮発性および不揮発性のリムーバブルおよび非リムーバブル媒体がある。コンピュータ記憶媒体には、それだけには限定されないが、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）または他の光ディスク記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置

または他の磁気記憶装置、または所望の情報の格納に使用でき、コンピュータ１１０からアクセスできる他の任意の媒体などがある。通信媒体は一般に、コンピュータ可読命令、データ構造、プログラムモジュール、または他のデータを搬送波または他の移送機構などの変調されたデータ信号に組み込む。これには任意の情報配送媒体がある。「変調されたデータ信号」という用語は、信号内の情報を符号化するように設定または変更された１つまたは複数のその特徴を有する信号を意味する。通信媒体には、それだけには限定されないが一例として、有線ネットワーク、直接配線された接続などの有線媒体、および音響、ＲＦ、赤外線、その他の無線媒体などの無線媒体がある。また、上記のどんな組合せでもコンピュータ可読媒体の範囲内に含まれるものとする。

【００２３】

システムメモリ１３０は、読取り専用メモリ（ＲＯＭ）１３１やランダムアクセスメモリ（ＲＡＭ）１３２など、揮発性および／または不揮発性メモリの形のコンピュータ記憶媒体を含む。基本入出力システム１３３（ＢＩＯＳ）は、例えば起動中など、コンピュータ１１０内の要素間での情報の転送を助ける基本ルーチンを含み、一般にＲＯＭ１３１に格納されている。ＲＡＭ１３２は一般に、処理ユニット１２０から直接アクセス可能な、かつ／または処理ユニット１２０が現在処理しているデータおよび／またはプログラムモジュールを含む。図１は、それだけには限定されないが一例として、オペレーティングシステム１３４、アプリケーションプログラム１３５、他のプログラムモジュール１３６、およびプログラムデータ１３７を示している。

【００２４】

コンピュータ１１０は、他のリムーバブル／非リムーバブル、揮発性／不揮発性コンピュータ記憶媒体を含むこともできる。一例にすぎないが、図１は、非リムーバブル不揮発性磁気媒体から読み取り、あるいはそこに書き込むハードディスクドライブ１４１、リムーバブル不揮発性磁気ディスク１５２から読み取り、あるいはそこに書き込む磁気ディスクドライブ１５１、およびＣＤ－ＲＯＭや他の光媒体など、リムーバブル不揮発性光ディスク１５６から読み取り、あるいはそこに書き込む光ディスクドライブ１５５を示している。動作環境の例で使用できる他のリムーバブル／非リムーバブル、揮発性／不揮発性コンピュータ記憶媒体には、それだけには限定されないが、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、半導体ＲＡＭ、半導体ＲＯＭなどがある。ハードディスクドライブ１４１は一般に、インターフェース１４０などの非リムーバブルメモリインターフェースを介してシステムバス１２１に接続され、磁気ディスクドライブ１５１および光ディスクドライブ１５５は一般に、インターフェース１５０などのリムーバブルメモリインターフェースによってシステムバス１２１に接続される。

【００２５】

上述し、図１に示したドライブおよびその関連のコンピュータ記憶媒体は、コンピュータ可読命令、データ構造、プログラムモジュール、およびコンピュータ１１０の他のデータの記憶を提供する。図１では例えば、ハードディスクドライブ１４１は、オペレーティングシステム１４４、アプリケーションプログラム１４５、他のプログラムモジュール１４６、およびプログラムデータ１４７を記憶するものとして示されている。これらの構成要素は、オペレーティングシステム１３４、アプリケーションプログラム１３５、他のプログラムモジュール１３６、およびプログラムデータ１３７と同じであっても、異なっていてもよいことに留意されたい。オペレーティングシステム１４４、アプリケーションプログラム１４５、他のプログラムモジュール１４６、およびプログラムデータ１４７は少なくとも異なるコピーであることを示すために、本明細書ではそれらに異なる番号を付している。ユーザは、タブレットまたは電子デジタイザ１６４、マイクロフォン１６３、キーボード１６２、および一般にマウス、トラックボール、またはタッチパッドと呼ばれるポインティング装置１６１などの入力装置を介してコマンドおよび情報をコンピュータ１１０に入力することができる。図１には示していないが、他の入力装置には、ジョイスティック、ゲームパッド、衛星パラボラアンテナ、スキャナなどがある。これらおよび他の

入力装置は、しばしばシステムバスに結合されているユーザ入力インターフェース１６０を介して処理ユニット１２０に接続されるが、パラレルポート、ゲームポート、ユニバーサルシリアルバス（ＵＳＢ）など他のインターフェースおよびバス構造で接続してもよい。モニタ１９１または他のタイプの表示装置もまた、ビデオインターフェース１９０などのインターフェースを介してシステムバス１２１に接続される。モニタ１９１は、タッチ画面パネルなどに一体化することもできる。モニタおよび／またはタッチ画面パネルは、タブレット型パーソナルコンピュータなど、コンピューティング装置１１０が組み込まれるハウジングに物理的に結合することができることに留意されたい。さらに、コンピューティング装置１１０などのコンピュータは、出力周辺インターフェース１９４などを介して接続できるスピーカ１９５、プリンタ１９６などの他の周辺出力装置を含むこともできる。

【００２６】

コンピュータ１１０は、リモートコンピュータ１８０など１つまたは複数のリモートコンピュータへの論理接続を使用してネットワーク式環境で動作することができる。リモートコンピュータ１８０は、パーソナルコンピュータ、サーバ、ルータ、ネットワークＰＣ、ピア装置、または他の一般のネットワークノードでよく、一般にコンピュータ１１０に関連して上述した多くまたはすべての要素を含むが、図１にはメモリ記憶装置１８１のみを示している。図１に示した論理接続は、ローカルエリアネットワーク（ＬＡＮ）１７１および広域エリアネットワーク（ＷＡＮ）１７３を含むが、他のネットワークを含んでいてもよい。こうしたネットワーキング環境は、オフィス、全社規模のコンピュータネットワーク、イントラネット、およびインターネットではごく一般的である。ＬＡＮネットワーキング環境で使用する場合、コンピュータ１１０は、ネットワークインターフェースまたはアダプタ１７０を介してＬＡＮ１７１に接続される。ＷＡＮネットワーキング環境で使用する場合、コンピュータ１１０は一般に、モデム１７２、またはインターネットなどＷＡＮ１７３を介して通信を確立する他の手段を含む。モデム１７２は、内蔵のものでも外付けのものでもよく、ユーザ入力インターフェース１６０または他の適切な機構を介してシステムバス１２１に接続することができる。ネットワーク式環境では、コンピュータ１１０に関連して示したプログラムモジュール、またはその一部をリモートメモリ記憶装置に格納することができる。図１は、それだけには限定されないが一例として、リモートアプリケーションプログラム１８５をメモリ装置１８１上に存在するものとして示している。図示したネットワーク接続は例であり、コンピュータ間の通信リンクを確立する他の手段を使用してもよいことは理解されよう。

【００２７】

（パッケージ内デルタ圧縮）

本発明は一般に、一部には、抽出されるとインストーラなどがコンピュータシステムを更新するために必要とするファイルを生成する、ファイルおよびデルタ圧縮ファイル（以下デルタと呼ぶ）の自己完結型パッケージを提供する方法およびシステムを対象とする。したがって、本明細書の例の多くは、一般に最新版のパッケージの提供を対象とする。しかし、こうした製品の用途は最新版以外にも非常にたくさんあることは理解されよう。例えば、ソフトウェアアプリケーション一式などのまったく新しいインストールを、本発明の一態様に従ってファイルおよびデルタ圧縮ファイルの自己完結型パッケージとして提供することができる。本発明は、シナリオによって変わるものなど、多くの実行可能ファイルおよび／または大部分は同等のファイルを含むパッケージのサイズを低減するという点で非常にうまく働く傾向があるが、他のデータファイルも同様に本発明のシステムおよび方法の恩恵を受ける可能性がある。さらに、本明細書で使用する場合、「ファイル」という用語は、従来ファイルと考えられているものを含み得るが、事実上、必ずしも従来のファイルシステムのファイルとして構成されているとは限らない株価やバイトストリームなどのデータの任意の集まりをさらに含むことができる。

【００２８】

さらに、本明細書に記載したパッケージは、自己完結型と呼ぶが、パッケージは、本発

明から利益を得るのに完全に自己完結型である必要はないことは容易に理解されよう。本発明は、デルタの従来の使用と組み合わせることができる。既存のファイルをベースファイルとして使用するデルタを含む混成物を構成することができる。このデルタから合成されたファイルを、次いでパッケージ内の別のデルタのベースとして使用することができる。例えば、ユーザが所与のコンピュータ上にどのファイルを有しているかについて何らかの知識を有することは可能であり、例えば、パッケージの内容を読み取る自己抽出型プログラムを、通常ユーザのコンピュータシステム上に存在するオペレーティングシステムの構成要素とすることができ、それによってそのプログラムがパッケージの一部として含まれている必要はなくなる。また、更新が現時点で唯一のバージョンのファイルに対するものである場合など、所与のファイルバージョンが顧客のコンピュータ上に存在することがわかっている可能性がある。こうした知識に基づいて、パッケージ生成手順では、あるデータを含む必要性をなくし、パッケージサイズをさらに低減することができる。

【００２９】

一般に、図２に示すように、本発明の一態様は、パッケージ生成環境２００において動作し、パッケージが含む必要があるデータのサイズを（少なくとも妥当な程度まで）最小限に抑えることを試みる方法で、ファイルおよびデルタを含む自己完結型パッケージ内デルタ圧縮パッケージ２０２を構築する。このために、パッケージに含める必要のある新しいファイルバージョンを含み得る（しかし必要に応じてデルタを生成するために以前のファイルバージョンを含む可能性がある）１組のソースファイル２０４が提供される。これらのソースファイル２０４は一般に、クライアントが有している必要がある対象ファイルに対応する。一般に、パッケージ内デルタ（ＩＰＤ）作成機構（make intra-package deltas (IPD) mechanism）２０６は、後述するように、まず（例えばそのリストを読み取ることによって）ソースファイル２０４を処理して本発明の一態様によるベースファイルおよびデルタにする。次いで従来のプロセスであるＣＡＢ（キャビネットファイル）作成機構がベースファイルおよび／またはデルタのデータを自己完結型パッケージ２０２に圧縮する。他の圧縮技術を使用することもできる。

【００３０】

従来技術では、デルタを生成するために以前のバージョンのファイルを必要とすることに留意されたい。例えば、デルタ圧縮の既知の技術では、元のファイル（またはその後のあるバージョン）を新しいバージョンとともに、デルタファイルを生成するデルタ作成エンジンに入力する。デルタは後にクライアント側でその元のバージョンに適用されて新しいバージョンが再作成される。これらの技術、およびよりよい圧縮をもたらすこれらの技術に対する改良については、「File update performing comparison and compression as single process」という名称の特許文献２、「Method and system for downloading updates for software installation」という名称の特許文献３、「Preprocessing a reference data stream for patch generation and compression」という名称の特許文献４参照、「File update by pre-initializing compressor/decompressor with other than decompression aid data」という名称の特許文献５、「Method and system for updating software with smaller patch files」という名称の特許文献６、および「Method for upgrading copies of an original file with same update data after normalizing differences between copies created during respective original installations」という名称の特許文献７に記載されている。

【００３１】

本発明の一態様によれば、後述するように、一般に、デルタの生成に使用するベースファイルの識別について想定する必要はなく、本明細書で使用する場合、ベースファイルとは、別のファイルを生成するためにデルタを後に適用する任意のファイルである。したがって、従来技術とは異なり、同じファイルの以前のバージョンだけではなく、任意のファイルを、デルタを適用することによって別のファイルをそこから合成することができるベースファイルとして使用することができる。例えば、以前のファイルバージョンおよび新しいファイルからデルタを生成する代わりに、人間の観察者にとって本質的に無関係なフ

ァイルに思われるものからデルタを生成することができる。例えば、１組のファイルが与えられている場合、適用されると文書処理構成要素ファイルを合成するデルタのベースファイルとしてスプレッドシート構成要素ファイルを使用することができることがわかる。さらに、単一のベースファイルを複数のデルタとともに再利用して、結果として生じる複数のファイルを合成することができる。

【００３２】

したがって、一実装形態では、本発明のシステムおよび方法は、（１つまたは複数の）新しいファイルバージョンからのみ導出されたデルタを使用することができるので、ソースファイル２０４は、以前のファイルバージョンを含んでいる必要はない。例えば図２では、ファイル１は、パッケージ２０４の一部として最新版として含まれる新しいバージョンとすることができ、また、これを使用してファイル２からデルタａ、ファイル３からデルタｂ、およびファイル４からデルタｃを生成することができる。次いで、これらのデルタをファイル１に適用することによりクライアント側の顧客によって後に抽出されると（図６に関連して後述）、４つの対象ファイル、ファイル１～ファイル４を、クライアントマシンでセットアッププログラムなどによって使用することができるようになる。後述するように、他のファイルの合成に使用され、しかし次いで破棄され、実際には最終的なファイルの組の一部にはならず、例えばセットアップ手順では使用されない１つまたは複数のベースファイルをパッケージが含むようにすることができることに留意されたい。

【００３３】

本発明の別の態様によれば、図７に関連してさらに後述するように、ファイルを合成するためにデルタを適用するベースファイル自体を、以前のデルタ圧縮操作から合成しておくことができる。したがって例えば図７では、ファイル２は、ファイル１およびデルタａから作成される。次いでファイル２を、デルタｂを適用してファイル３を生成するベースファイルとして使用する。したがって本発明では、パッケージにおいて必要なデータ量を低減するようにそれぞれ働くいくつかの新しい概念が提供される。

【００３４】

さらに、あるファイルバージョンを別のファイルバージョンのデルタを生成するベースファイルとして使用し、そのベースファイルが削除され、セットアップ手順で使用されない場合でさえも、そのベースファイルを必要に応じて抽出の目的でパッケージに含めることは可能である。例えば図７で、ファイル１は以前のファイルバージョンでもよく、これをデルタの作成に使用し、パッケージに入れ、他のファイルを作成するためにエクストラクタ２１０が使用し、しかし次いでセットアップ手順の前に削除することができる。

【００３５】

図面のうちの図３を参照すると、パッケージ生成環境２００（図２）の一実装形態における構成要素を示している。一般にこの環境２００は、ソフトウェアベンダー、またはパッケージの生成を必要とするベンダーに関連する何らかのサードパーティで、１つまたは複数の構成要素の組を介して実装される。この生成環境２００は、通常、クライアントの要求に応答して動的に動作するものではなく、むしろ自己完結型パッケージ２０２を生成するのにおよそ何時間または何日もの比較的長い時間がかかり得る、計算上費用がかかるプロセスであることに留意されたい。

【００３６】

一般に、ＭａｋｅＩＰＤ機構２０６は、提供されたファイルリスト２１２を読み取り、パッケージに入っているファイル、およびそれらのファイルを見つけることができる場所を決定する。例えば、ファイルリストは、各エントリがパッケージ内のファイルの名前を指定している［Ｆｉｌｅ］セクション、およびファイルへのフルパスを含む。ファイルリストは、このパッケージ内には必要なく、しかし上述した混成物の構築に使用することができる、ユーザのコンピュータ上に存在することがわかっているいくつかの参照ファイルを指定することもできる。これらのファイルは、パッケージ内の他の任意のファイルの潜在的なベースとしてみなすことができる。

【００３７】

ファイルリスト２１２は、例えば［Ｏｐｔｉｏｎ］セクションでいくつかの処理オプションを指定することもできる。例えば、存在するとしたら、抽出後にパッケージ内のどのファイルを実行すべきかを指定する「Ｒｕｎ」命令を（例えばマニフェストファイルへのパススルーとして）提供することができる。Ｖｅｒｉｆｙ命令を設定して、ＭａｋｅＩＰＤ機構にすべてのファイルの［Ｖｅｒｉｆｙ］セクションを生成させるようにすることができる。ＰａｔｃｈＤＬＬ命令は、マニフェストファイルにわたされるファイルを識別し、ＭａｋｅＩＰＤ機構２０６がこのファイルを、キャビネットファイルの作成時に使用するスクリプトに含めるようにするとともに、このファイルをパッケージ内の他のファイルの潜在的なベースとしてみなすようにする。

【００３８】

図３に示すように、一実装形態では、生成プロセス全体をＧＵＩまたはコマンドライン３０４から開始する。これは、パーサー３０６によって解釈されて操作パラメータが決定される。操作パラメータは、ファイルリスト２１２の識別を含み、これによって、上述したように、例えばパッケージ内に必要なファイルの名前を含むファイル名のテキストファイルリスト内で使用するソースファイル３０８が識別される。後述するように、必要なシンボルファイルを含み得る１つまたは複数のディレクトリ（例えばセミコロンで区切られる）を指定するパス情報も提供される。指定されていない場合、使用されるディレクトリは、ソースファイルのディレクトリである。

【００３９】

また、別の操作パラメータとして、パッケージ構築プロセス中に使用する様々な中間ファイルとともに使用するディレクトリを指定することができる。より詳細には、ＭａｋｅＩＰＤ機構２０６は、処理中、いくつかの中間ファイルを作成し、これらを指定された作業ディレクトリ内で維持する。こうしたファイルは、記号一覧（ｓｙｍｂｏｌ ｌｉｓｔｉｎｇｓ）、デルタファイル、およびソリューションを１組のデルタ解凍指示として記載するマニフェストファイルを含む。ＭａｋｅＩＰＤ機構２０６が続いて実行され、同じ作業ディレクトリが指定された場合、既存の任意のファイルを使用して分析を助けることができることに留意されたい。例えば、必要なすべてのファイルが依然として使用可能である場合、ＭａｋｅＩＰＤ機構２０６は、例えばおよそ数秒など迅速にその操作を完了させる。前の構築以降パッケージファイルのうちの一部しか変わっていない場合、影響を受けていないデルタを再利用することによって、かなりの処理時間が短縮される。

【００４０】

別のオプションでは、例えばＣＡＢ圧縮済み形式などの最終的なパッケージの作成に使用するスクリプトの名前を指定することができる。他の操作パラメータは、後述するように、前の何らかの実行からソリューションをコピーするためにインポートされる既存のマニフェストの場所を指定することができる。また、ユーザは、処理中の（例えばユーザやテキストファイルへの）異なるタイプの出力を指定して、明確なまたは詳細な出力を選択したり、出力を止めたりすることができる。

【００４１】

ファイルリストが解析され、各エントリによって、パッケージ内のソースファイルの場所、およびそのソースファイルの名前が提供される。任意の入力ファイルオプションが解析され、これには、抽出後に実行するファイルの名前、エクストラクタとして使用するデルタ適用エンジン（ＤＬＬなど）の名前、および抽出後自己解凍機構がファイルの署名（ＭＤ５ハッシュ生成ものなど）を確認すべきかどうかが含まれ得る（ＭＤ５はＲＳＡ Ｄａｔａ Ｓｅｃｕｒｉｔｙ，Ｉｎｃ．Ｍｅｓｓａｇｅ Ｄｉｇｅｓｔ Ａｌｇｏｒｉｔｈｍ ５、別名Ｉｎｔｅｒｎｅｔ ＲＦＣ１３２１を指す。ＣＲＣ、ＭＲ５、ＳＨＡ１などを含む適切な任意のエラー検出または整合性検証ハッシュを使用することができる）。

【００４２】

前処理の別の一部として、デルタ適用エンジンのエントリだけでなく、後述するように、マニフェストファイルのファイルリストにエントリが追加される。また、任意の複製を原本(original occurrence)のコピーとして識別するために、ソースファイル３０８につ

いてＭＤ５署名が計算される。ファイルが実行可能ファイルである場合、それ自体のシンボルファイルについての詳細が抽出される。

【００４３】

より詳細には、上記の特許文献４は、デルタの使用時により最適なサイズ低減を得るための実行可能ファイルの記号（ＥＸＥ、ＤＬＬ、ＯＣＸ、ＳＹＳなど）の使用について記載している。ＭａｋｅＩＰＤ機構２０６は、この技術を利用して、ソースファイルが提供されている同じディレクトリ内で記号を探す。追加の複数のシンボルディレクトリは、各ディレクトリをセミコロンで区切るオプションを使用して、明示的に提供することができる。指定された各ディレクトリは、分析に利益を提供し得る任意の記号について再帰的に検索される。シンボルパスがスキャンされ、識別された任意のシンボルファイルが検索される。

【００４４】

ＭａｋｅＩＰＤ機構２０６は、一意のソースファイルごとに、パッケージ内の他の一意の各ソースファイルのエントリを含む、予想されるデルタ入力のリストを生成する。これらのリストには、合計Ｎ・（Ｎ－１）個、またはほぼN^2個の合計エントリがある。

【００４５】

本発明の一態様によれば、ＭａｋｅＩＰＤ機構２０６の繰り返し構成要素３１２は、ファイルおよびそのファイルのリストの各ファイルをデルタ作成エンジン３１４に入力することによって、各ファイルのリストの予想されるデルタごとにデルタを作成する。デルタを作成する入力がいずれも実行可能ファイルである場合、上記の特許文献４に記載されているように、使用可能な記号情報を使用してデルタのサイズを最適化する（一般に適したデルタ作成エンジン３１４については、上記の複数の米国特許に記載されていることに留意されたい）。これらのデルタ３１６は、作業ディレクトリに格納され、各デルタのサイズは、次の計算のリストエントリに追加される。

【００４６】

本発明の別の態様によれば、このサイズおよびファイル識別情報から有向グラフ３１６が生成される。より詳細には、各ソースファイルが頂点として有向グラフ３１６に追加され、予想される各デルタが辺として追加され、重みはそのデルタのサイズに等しい。また、「ＮＵＬＬ」頂点からソースファイルの頂点のそれぞれへの辺も含まれ、その重みは、そのソースファイルの圧縮されたサイズに等しい。

【００４７】

一例として、図４に示すように、４つのソースファイルＡ、Ｂ、Ｃ，およびＤに対する繰り返しから生成される有向グラフ４１６について考察する。この情報の代替のテーブル表現４１７との関連でわかるように、各ファイルは頂点、予想される各デルタはサイズ値であり、例えば、ファイルＡをベースファイルとして使用し、ファイルＢを合成ファイルとして使用するデルタのサイズｂａ、および逆のサイズａｂがある。また、例えば「ＮＵＬＬ」頂点を辺として使用することによってＢを単に圧縮するサイズｂもある。

【００４８】

この情報から、最小スパニングツリー（ｍｉｎｉｍｕｍ　ｓｐａｎｎｉｎｇ　ｔｒｅｅ）３２０を有向グラフ３１６上で計算することができる。このために、一部はほぼ線形の実行時間による、よく知られている様々な最小スパニングツリー計算３１８のうちの１つを使用することができる。こうした最小スパニングツリー計算は、情報工学の文献に記載されており、本明細書ではごく簡単に述べるに留める。

【００４９】

概念上、各ファイルは、使用可能な最も小さいデルタから導出される。しかし、それ自体を合成するのにファイルを使用できないため、循環参照は形成されないことが重要である。いくつかのデルタを除外して、これらの循環を阻止する。プロセスは、他のデルタを無くして他のより有望なデルタの使用が可能になるように全体的に最適化されて、合計サイズが最小限に抑えられるようになる。この問題は、「有向最小スパニングツリー（ｄｉｒｅｃｔｅｄ　ｍｉｎｉｍｕｍ　ｓｐａｎｎｉｎｇ　ｔｒｅｅ）」として知られる問題に

対応する。辺の重みにデルタの節約を使用し、有向最大スパニングツリーを探す同等のソリューションを構築することができる。

【００５０】

次いでルートとして「ＮＵＬＬ」頂点を使用したスパニングツリーが列挙される。ルート頂点を離れる辺は、パッケージ内に単に圧縮されるファイルに対応する。他の頂点を離れる辺は、生成されたデルタを使用するものに対応する。引き続き図４の例では、最小スパニングツリー４２０は結果として得られるソリューションに列挙され、Ａは単に圧縮され、Ｂは圧縮され、ファイルＤを適切なデルタから合成するためのベースファイルとして使用されることがわかる。次にファイルＣは、合成されたベースファイルＤ、およびＤからＣを生成するのに適切なデルタから合成される。理解できるように、重みのようにサイズに基づく最小スパニングツリーは、このように使用したときに可能な最も小さいパッケージを提供する。

【００５１】

図３に戻ると、列挙３２２では、本質的に各ソースファイルのソリューション（圧縮済みまたはデルタ）がマーク付けされ、ツリー内で見つけられた順序でソースファイルのリンクリスト３２４が作成される。後述するように、マニフェスト生成プロセス３２６は、このリンクリスト３２４（および上記の他の一部の情報）からマニフェストファイル３２８を形成する。本質的に、マニフェストファイル３３２は、クライアントコンピュータ上で実行されると、エクストラクタ２１０の操作を指示する。デルタを適用するベースファイルとして働くようにファイルを合成する必要があるが、ファイルがまだ存在していないという状況にエクストラクタが陥らないようにするために、操作は、特定の順序でマニフェスト３２６に列挙され、この順序で実行されることに留意されたい。例えば図４では、ファイルＤは、ファイルＣをファイルＤから合成する前にファイルＢから合成しておく必要がある。

【００５２】

さらに、ＭａｋｅＩＰＤ機構２０６は、圧縮パッケージがそこから作成される命令ファイル３３２を生成する命令ファイル生成構成要素３３０を含む。命令ファイルは、一時ディレクトリ内のソースファイルおよびデルタファイルの場所、およびパッケージ内のこれらのファイルの名前を含む。従来のＭａｋｅＣＡＢ機構２０８（図２）などのコンプレッサ／パッケージャ３３４構成要素は、命令ファイルを使用して自己完結型のデルタ圧縮パッケージ２０２を生成する。パラメータオプションは、作成する命令ファイルのパスおよび名前を指定し、この命令ファイルを使用して、同じパスで同じベースファイル名を有するＣＡＢファイルを指定する。

【００５３】

図４の簡略化された例では４つのファイルのみを使用しているが、ＭａｋｅＩＰＤ機構２０６は、パッケージの内容を分析するのに比較的長い時間がかかる可能性がある。したがって、可能な場合、プロセスの繰り返し／分析部分を回避することが望ましい。例えば、ほぼ同じ内容をそれぞれ含む複数のパッケージを生成する必要がある場合、前の実行からのソリューション（例えばマニフェストで維持される）を使用して、現在のパッケージにソリューションをより直接的に指定することができる。これは、第１のパッケージが最適化された後、追加言語のパッケージを生成するときに行うことができる。これは、内容のわずかな変更によるパッケージの再構築の際にも使用可能である。

【００５４】

パッケージを構築する既存のソリューションを利用するために、ＭａｋｅＩＰＤ機構２０６は、ソリューションをインポートすることができる。例えばパラメータオプションとして指定されると、パラメータは、マニフェストファイルを見つけることができるディレクトリ、または使用するファイルの完全ファイル名を指定することができる。

【００５５】

一般に、使用時に、ＭａｋｅＩＰＤ機構２０６のインポートプロセスは、以前作成された何らかのマニフェストファイルから［Ｄｅｌｔａｓ］セクションを読み取ってどのデル

タが選択されたかを調べる。より詳細には、ソリューションをインポートするときに、インポートされたマニフェストが読み取られて、予想されるデルタ入力のリストが生成される。このパッケージ内の複製ファイルを参照する任意のエントリが代わりに元のファイルに関連するものであると推測される。複製を識別するインポートされたマニフェスト内の任意のエントリは、予想されるデルタ入力と推測される。またはその逆も推測される。これらのリストは一般に、ファイルごとにエントリを１つだけ有している、すなわちほぼＮ個の合計エントリを有している。

【００５６】

容易にわかるように、インポートオプションを使用してソリューションファイルを指定する場合、ほとんどのＭａｋｅＩＰＤ分析が省略されて、直前に選択されたデルタのみが使用される。構築する必要があるのは、N^2 個ではなく、約Ｎ個だけでよいことに留意されたい。新しいパッケージが新しい任意の内容を含んでいる場合、これらのファイルはデルタとしては考えられない。元のパッケージが特定のファイルをデルタ参照として選択しており、そのファイルが新しいパッケージ内で見つからない場合、これらのデルタは考慮に入れられない。しかし、インポートオプションは、パッケージが非常に似た内容を有する状態で最もよく働くことを理解されたい。

【００５７】

図５に示した自己抽出プロセスの説明を参照すると、顧客環境５００で、ネットワーク送信を介する、またはコンパクトディスクまたはＤＶＤ－ＲＯＭディスクなどの物理媒体によるなど何らかの方法で、自己完結型パッケージ５０２のコピーが受信される。留意すべき点として、図５ではパッケージプロデューサを、パッケージを提供するものとして示しているが、サードパーティディストリビュータや、パッケージをそのマシンから使用できるようにする企業ネットワークなど１つまたは複数の仲介物が存在していてもよいことを理解されたい。

【００５８】

一般に、実行可能抽出ファイル（自己完結型パッケージ内などにあるが、すでに顧客のマシン上にある可能性もある）がＧＵＩ、ネットワークスクリプト、コマンドラインなど何らかの方法で実行されると、抽出機構５０４が開始する。指定されたパラメータがある場合もあり、必要に応じてこれらを解析してもよい。

【００５９】

第１の前処理操作として、実行可能プログラムは、無作為に名前を付けた一時ディレクトリ５０６を、最大の空き容量を有するハードドライブなどのローカルハードドライブ上に作成することができる。あるいは、任意選択のパラメータなどを介して一時作業ディレクトリを指定することもできる。

【００６０】

ファイルのそれぞれは、キャビネットファイルから一時ディレクトリ５０６に抽出される。この抽出の一部として、キャビネットが作成されたときに圧縮されたファイルが解凍される。キャビネットファイルが作成される前にすでに圧縮されており、抽出のこの第１の部分では解凍されない他のファイルがあり得ることに留意されたい。

【００６１】

この時点で、ディレクトリ５０６は、１つまたは複数のベースファイル５０８、１つまたは複数のデルタ５１０、およびデルタ圧縮を使用するより圧縮することがより効率的であることがわかっている場合に単に圧縮されたファイル（図４の例のファイルＡおよびＢなど）など、他の任意のファイル５１２を含む。まだ圧縮されている任意のファイルを必要に応じて解凍することができる。この例では、マニフェスト５１４、適用エンジン５１６（パッケージに付属している可能性のあるＤＬＬなど）、およびセットアッププログラム５１８（それ自体合成されていない場合）もディレクトリ５０６内で使用可能である。マニフェストファイルは存在しないことがあり、これはパッケージ内デルタ圧縮のない、ファイルの従来のキャビネット圧縮の状態であることに留意されたい。

【００６２】

図５の例のように、マニフェスト５１４が存在する場合、抽出機構５０４は、マニフェスト５１４に列挙された各指示を処理する。そこに列挙されているデルタ５１０ごとに、指定されたデルタ５１０が指定された入力ベースファイル（例えば複数のベースファイル５０８のうちの１つまたは前に合成されたファイル５２０など）に適用されて、図５の合成ファイル５２０で示した新しいファイルが生成される。適用エンジンは、本質的にデルタ作成エンジンの逆であり、ベースファイルを１つの入力とみなし、デルタをもう１つの入力としてみなし、対象ファイルを出力として合成することに留意されたい。デルタ作成エンジンのように、適した適用エンジンについては、上記の複数の米国特許出願に記載されている。また、後述するように、複製ファイルごとに、指定したファイルのコピーが新しい名前で作成され、マニフェスト５１４で指定された削除パターンごとに、抽出機構５０４は、そのパターンに一致する一時ディレクトリ５０６内の任意のファイルを削除する。一般にこれを使用して、すべてのファイルが合成された後にデルタが破棄される。

【００６３】

一例として、図６は、３つのファイル、ファイル２、ファイル３、およびファイル４を合成するために３つのデルタａ、ｂ、ｃが適用される単一のベースファイルを有する自己完結型パッケージを介して取得されたファイルを示している。上述したようにこれは必須ではないが、ベースファイル、ファイル１を最新版として使用することができる。一般に、ラベル表示の順序で矢印をたどることによって、エクストラクタが抽出データ（マニフェストなど）を読み取り、必要に応じてセットアッププログラムがファイルのインストールに使用するディレクトリにベースファイルを解凍することがわかる。次いでベースファイルを各デルタａ、ｂ、ｃとともに使用して、それぞれファイル２、ファイル３、およびファイル４のファイルが合成される。

【００６４】

図７は、ファイル２がファイル１およびデルタａから合成され、次いでファイル２をベースファイルとして使用して、デルタｂとともにファイル３を合成するわずかに異なる例を示している。ファイル４は、ベースファイルおよびデルタｄから合成される。

【００６５】

図５に戻ると、対象ファイルが完全に使用可能になった後、プログラムがＲＵＮ命令を介してセットアッププログラム５１８としてマークされている場合、そのプログラムが自動的に実行される。セットアッププログラムが完了すると、抽出機構は、その戻りコードを取り出し、保存する。次いで抽出機構５０４は、任意のディレクトリエントリを含めて、一時ディレクトリ内において作成された残りの任意のファイルを削除し、セットアッププログラムの保存された戻りコードで終了する。抽出機構５０４も必要に応じて削除することができる。

【００６６】

（マニフェストファイル情報）

一実装形態では、パッケージ内デルタ機能は、マニフェストファイルによって実行可能になる。しかし、抽出機構を指示する他の方法が可能であることは容易に理解できよう。例えば、適した他の抽出データを使用することができる。あるいは、デルタを何らかの方法で順序付け（キャビネット解凍後一時ディレクトリ内に配置されている順序によるなど）、その順序でベースファイルに適用して、例えばファイル名がデルタから導出される、または１組の名前変更操作を介して変更されるようにすることができる。しかし、後述するように、マニフェストファイルは、抽出機構を案内するために、端的で効率的な方法を提供する。

【００６７】

この特定の実装形態では、ファイルは、組み込みキャビネットファイルにおいて「＿ｓｆｘ＿ｍａｎｉｆｅｓｔ＿」と名付けられている。キャビネットを一時対象ディレクトリに抽出する間、抽出機構は、このファイル名に注意し、その内容を使用して、抽出が完了した後に実行する様々な処理を指定する。＿ｓｆｘ＿ｍａｎｉｆｅｓｔ＿ファイルは、セットアッププログラムを混乱させないように対象ディレクトリには追加されない。この追

加処理が完了した後、セットアッププログラムが開始する。

【００６８】

一実装形態では、マニフェストは、角括弧で示したセクション内にまとめられるテキストファイルであるが、マークアップ言語形式など、他の形式も同等であることは容易に理解できよう。各セクションは、操作の詳細を記載した行を含む。［Ｄｅｌｔａｓ］、［Ｃｏｐｙ］、［Ｖｅｒｉｆｙ］、および［Ｄｅｌｅｔｅ］を含めて、いくつかのセクション名が定義される。各セクションは、セクション固有のエントリを含み、それらは、それらが属するセクションに従って解釈される。あるセクション内の一部のエントリは、事前定義のキーワードから始めることができる。これを命令と呼ぶ。

【００６９】

マニフェスト内のセクションは、任意の順序で指定できるが、［Ｄｅｌｔａｓ］、［Ｃｏｐｙ］、［Ｖｅｒｉｆｙ］、次いで［Ｄｅｌｅｔｅ］の順序で処理される。各セクション内で、エントリは連続して処理される。したがってサポートされているセクションは、ファイル内の場所ではなく、名前で検索される。

【００７０】

この形式では、ファイル内の各セクションは、括弧で囲まれたセクション名から始まり、新しいセクションの先頭、またはファイルの末尾で終了する。複数のセクションが同じ名前を有している場合、それらは単一のセクションに論理的にマージされる。セクション名、エントリ、および命令は、ケースインセンシティブ(case-insensitive)である。他の規則は、セクション内の各エントリおよび命令は改行文字（ｈｅｘ　０ｘ０Ａ）またはファイルの末尾で終了することである。コメントは、セミコロン（；）文字で始まり、改行文字またはファイルの末尾で終了する。コメントは、単独で、またはエントリの後に１行に表示することができる。カンマは、セクションエントリおよび命令内に提供された複数の値を区切るために使用し、等号は、「キー」値を他のパラメータと区切るのに必要となり得る。

【００７１】

ファイル名エントリは、パッケージルートに対して相対的であるが、代替実装形態では、絶対パスをサポートすることが可能である。パッケージ内のサブディレクトリ内のファイルは、相対パスで表され、例えば「ｕｐｄａｔｅ」サブディレクトリに存在する「ｕｐｄａｔｅ．ｅｘｅ」という名前のファイルは、ｕｐｄａｔｅ＼ｕｐｄａｔｅ．ｅｘｅと呼ばれることになる。空白行、行の先頭の空白（スペース、タブなど）は無視され、空白は引用符の外に置く。空白文字、カンマ、または等号が文字列の一部となる場合、その文字列は引用符内に含まれる。引用符内に含まれないセミコロン文字をコメントに使用し、その行の末尾までのすべてが無視される。

【００７２】

スペースまたは他のブレーク文字（ｂｒｅａｋｉｎｇ　ｃｈａｒａｃｔｅｒ）を含む必要のあるファイル名または相対パスは、二重引用符で囲まれる。セクション名は、引用符で囲んではならない。以下は、上記の規則に従うマニフェストの例である。

【００７３】

【表１】

```
; Sample _sfx_manifest_ for Q326863
; This package contains eight closely-related files

[Options]
      run = xpsp1hfm.exe

[Deltas]
      sp2\ntkrnlpa.exe = sp2_ntkrnlpa_exe._p, sp2\ntoskrnl.exe
      sp2\ntkrpamp.exe = sp2_ntkrpamp_exe._p, sp2\ntkrnlpa.exe
      sp2\ntkrnlmp.exe = sp2_ntkrnlmp_exe._p, sp2\ntkrpamp.exe
      sp1\ntoskrnl.exe = sp1_ntoskrnl_exe._p, sp2\ntoskrnl.exe
      sp1\ntkrpamp.exe = sp1_ntkrpamp_exe._p, sp2\ntkrpamp.exe
      sp1\ntkrnlpa.exe = sp1_ntkrnlpa_exe._p, sp2\ntkrnlpa.exe
      sp1\ntkrnlmp.exe = sp1_ntkrnlmp_exe._p, sp2\ntkrnlmp.exe

[Delete]
      *._p

[Verify]
      sp1\ntkrnlmp.exe = 1D575A38471CB066CC23925AEFCD9A49
      sp1\ntkrnlpa.exe = 89A0875AEA13E021C9E63F2EB6446327
      sp1\ntkrpamp.exe = 934AAC402BA1F8D1C9319AA0DB849E6F
      sp1\ntoskrnl.exe = C78CA71C81A051DF25A79102C867BB10
      sp2\ntkrnlmp.exe = E62EA04019BC4AE785855DA0EE36D231
      sp2\ntkrnlpa.exe = 6C1BD8121224A83DC0FD9E36BFCF2AD9
      sp2\ntkrpamp.exe = 64F50291904454883478204DF6A53A6C
      sp2\ntoskrnl.exe = A7379A2180D3AA4F64D804D8B5CDD659
```

【００７４】

［Ｄｅｌｔａｓ］セクションは、パッケージ内デルタの中核となる特徴、つまりパッケージ内の他のファイルからのセットアップに必要なファイルの一部をどのように合成するかを記載している。［Ｄｅｌｔａｓ］セクション内のエントリの構文は

[Deltas]
{targetfilename}={deltafilename}[,{reference}]

を含む。この場合、｛ｔａｒｇｅｔｆｉｌｅｎａｍｅ｝は生成するファイルの名前、｛ｄｅｌｔａｆｉｌｅｎａｍｅ｝はパッケージ内のデルタファイルの名前、および｛ｒｅｆｅｒｅｎｃｅ｝は、存在する場合はベースファイルとして使用する既存のファイルの名前である。エクストラクタは、デルタファイルを参照ファイルのコピーに適用して対象ファイルを作成する。

【００７５】

この構文の一例は、

[Deltas]
file2=file1_to_file2.delta,file1

などである。

【００７６】

この例で、ｆｉｌｅ１は、実際のパッケージ内に含まれている、または以前の何らかの［Ｄｅｌｔａｓ］エントリから合成する必要がある。明らかなように、ｆｉｌｅ１＿ｔｏ＿ｆｉｌｅ２．ｄｅｌｔａは、ｆｉｌｅ１のコピーに適用され、ｆｉｌｅ２を作成し、ｆｉｌｅ１は変更されない。一般に、ｆｉｌｅ１＿ｔｏ＿ｆｉｌｅ２．ｄｅｌｔａは［Ｄｅｌｅｔｅ］セクションにも含まれており、セットアッププログラムが開始する前に削除されることになる。

【００７７】
参照ファイルなしに「デルタ」を有することができることに留意されたい。これは、ゼロ長参照ファイルに基づくデルタに等しい。こうしたデルタのエントリは、単に｛ｒｅｆｅｒｅｎｃｅ｝を省略するだけである。
【００７８】
［Ｃｏｐｙ］セクションによって、ファイルをパッケージ内に複写することができる。例えば、セットアッププロセスが異なる名前の同一の３つのファイルを必要とする場合、ファイルの１つのコピーをパッケージに含め、他の２つをセットアップ前に複製することができる。［Ｃｏｐｙ］セクション内のエントリの構文は、

```
[Copy]
  {targetfilename}={sourcefilename}
```

を含む。この場合、｛ｔａｒｇｅｔｆｉｌｅｎａｍｅ｝は生成されるファイルの名前、｛ｓｏｕｒｃｅｆｉｌｅｎａｍｅ｝は同じ内容を有する既存ファイルの名前である。
【００７９】
一例には、

```
[COPY]
  file3=file2
```

などがある。
【００８０】
この例では、ｆｉｌｅ２は、パッケージ内に含まれている、または以前の［Ｄｅｌｔａｓ］または［Ｃｏｐｙ］エントリから合成される。明らかなように、ｆｉｌｅ２はｆｉｌｅ３にコピーされ、ｆｉｌｅ２は変更されない。
【００８１】
［Ｄｅｌｅｔｅ］セクションによって、セットアップに必要ないファイルを、セットアッププログラムが開始する前に削除することができる。一般的な使用は、必要なファイルの合成に使用する任意のデルタファイルを削除することである。［Ｄｅｌｅｔｅ］セクション内のエントリの構文は

```
[Delete]
  {targetfilename}
```

である。この場合、｛ｔａｒｇｅｔｆｉｌｅｎａｍｅ｝は、削除されるファイルの名前である。指定された｛ｔａｒｇｅｔｆｉｌｅｎａｍｅ｝はワイルドカードを含んでいる可能性があり、この場合、そのパターンに一致する任意のファイルが削除される。所与の名前またはパターンに一致するファイルがない場合、エラーは報告されない。ファイルの削除は再帰的ではない。サブディレクトリを明示的に指名する必要がある。
【００８２】
一例には、

```
[Delete]
  *.delta
```

などがある。
この例では、「＊．ｄｅｌｔａ」に一致するファイルが削除される。
【００８３】
［Ｖｅｒｉｆｙ］セクションは、破損の有無をチェックするいくつかのファイルを指定する。このセクション内の各エントリは、確認する単一のファイル、およびそのファイルの予想されるＭＤ５署名を指名する。セクション内の任意のファイルを確認できない場合、インストールが失敗する。［Ｖｅｒｉｆｙ］セクション内のエントリの構文は、

```
[Verify]
  {targetffilename} = {md5signature}
```

を含む。この場合、｛ｔａｒｇｅｔｆｆｉｌｅｎａｍｅ｝はパッケージ内にあることが予想されるファイルの名前、｛ｍｄ５ｓｉｇｎａｔｕｒｅ｝はそのファイルのＭＤ５署名の１６進法表現である。

【００８４】
一例には、次のものなどがある。
【００８５】
【表２】

```
[Verify]
     file1 = 3D2EDAF98C77086F18925193E471C1C8
     file2 = CCF3719A65DB9637864A4340A74575DE
     file3 = 7BEB665C45858982E58D496C3A474CB2
```

【００８６】
この例では、ファイルｆｉｌｅ１、ｆｉｌｅ２、ｆｉｌｅ３のそれぞれの署名が計算され、指定された値と比較される。署名のいずれかが一致していない場合、またはファイルのうちのいずれかがない場合、インストールが失敗する。
【００８７】
［Ｏｐｔｉｏｎｓ］セクション内の特定の命令を使用して制御できるＩＰＤオプションがいくつかある。未定義の命令は無視される。
【００８８】
【表３】

```
[Options]
     Run = update\update.exe    ; program to execute
     PatchDLL = update\mspatcha.dll    ; updated delta core
```

【００８９】
Ｒｕｎ＝命令は、実行するセットアッププログラムの名前を定義する。エクストラクタによって、キャビネットファイルの作成時に命令ファイル内の／ＲＵＮオプションを使用して実行するようにプログラムをすでにマーク付けしておくことができる。しかし実行するプログラムは、キャビネット内のファイルのうちの１つではなく、しかし他のファイルのうちの１つに基づくデルタとしてパッケージされる可能性がある。この場合、キャビネット内でマーク付けされるファイルは存在しない。Ｒｕｎ＝命令は、実行するプログラムを識別するものであり、キャビネット内のそのファイルをマーク付けするのと機能的に等しい。Ｒｕｎ＝命令は、存在する場合、キャビネット内のマーク付けされたファイルを無効にする。
【００９０】
ＰａｔｃｈＤＬＬ＝命令は、デルタの適用に使用するＤＬＬの名前を定義する。デフォルトでは、エクストラクタは、Ｗｉｎｄｏｗｓ（登録商標）システムディレクトリのｍｓｐａｔｃｈａ．ｄｌｌを使用するが、この命令は、パッケージの代替ファイルを明示的に指名することができる。マニフェストで指定されたファイルは、対象ディレクトリに対して相対的であるため、このＤＬＬは、生パッケージ内のファイルのうちの１つである。［Ｄｅｌｅｔｅ］セクションは一般に、セットアッププログラムが開始する前にこのＤＬＬを破棄するエントリを含む。
【００９１】
従来より、キャビネットパッケージは、その内容が展開され、セットアッププログラムが実行され、内容がクリーンアップされるという点で、ただ単に実行されるだけである。様々な理由から、パッケージの内容を展開し、しかしセットアップまたはクリーンアップ操作は実行しないことも望ましい可能性がある。例えば、ベンダーは、パッケージ内にある１つまたは複数のファイルを取得する必要はあるが、このコンピュータ上に実際にインストールされているパッケージは必要ないかもしれない。パラメータオプションは、使用されると、宛先ディレクトリの入力をユーザに要求し（または／Ｘ：ｔａｒｇｅｔｄｉｒを使用して明示的に与えられ）、内容がそこで単に展開される機能を提供する。
【００９２】

本発明のパッケージ内デルタシステムおよび方法は、この概念もサポートしている。このパラメータオプション（／Ｘ：ｔａｒｇｅｔｄｉｒ）を使用するとき、エクストラクタは、セットアッププログラムを実行することなく内容を抽出する。しかし、内容が任意のデルタファイルを含んでいるとき、これらのデルタは一般に、この形式での使用のものではないため、デフォルトではパッケージ内デルタ処理が依然として実行されて、内容が固有の形式に戻される。

【００９３】

一部のオペレーティングシステムでは、ファイルの関連付けを行うことができ、特定のプログラムをあるタイプのファイルと関連付けることができる。ファイル名サフィックス（「拡張子」）により関連付けが可能になるものもあれば、ファイルの他の属性を使用するものもある。本明細書に記載した自己抽出型パッケージは、関連付けに依存してファイルが起動されると自己解凍プロセスを開始する、キャビネットファイルなどのファイルの集まりのみを使用して実施することができることがわかる。したがって、自己抽出型機能の実行可能コードは、パッケージの一部である必要はない。

【００９４】

上記の詳細な説明からわかるように、自己完結型パッケージでデータを提供するために使用できるデルタ圧縮による方法およびシステムが提供されている。パッケージサイズは大幅に低減され、しかしベンダーおよびクライアントの顧客は自己完結型パッケージの恩恵を受けることができる。したがってこの方法およびシステムによって、現代のコンピュータ処理で必要なかなりの利点および利益が提供される。

【００９５】

本発明は、様々な変更形態および代替構成の余地があるが、そのいくつかの実施形態の例を図面に示し、上記で詳細に説明してきた。しかし、本発明を開示した特定の形態に限定する意図はなく、逆に本発明は、本発明の意図および範囲内に含まれるすべての変更形態、代替構成、および均等物をカバーすることを意図する。

【図面の簡単な説明】

【００９６】

【図１】本発明を組み込むことができるコンピュータシステムを表す概略ブロック図である。

【図２】本発明の一態様によるパッケージ内デルタ圧縮を使用した自己完結型パッケージの生成を表す概略ブロック図である。

【図３】本発明の一態様によるパッケージ内デルタ圧縮を使用して自己完結型パッケージを生成する、ソフトウェアベンダーなどのパッケージ生成ソース環境における構成要素を表す概略ブロック図である。

【図４】本発明の一態様による、パッケージ内デルタ圧縮パッケージに含めるベースファイルおよび／またはデルタをどのように選択するかを表す概略ブロック図である。

【図５】本発明の一態様によるパッケージ内デルタ圧縮パッケージからファイルを抽出する、クライアントコンピュータなどの顧客対象環境における構成要素を表す概略ブロック図である。

【図６】本発明の一態様による単一のベースファイルへの複数のデルタの適用による複数のファイルの抽出を表す概略ブロック図である。

【図７】本発明の一態様による、ベースファイルへのデルタの適用、およびそれ自体デルタ圧縮を介して合成されたベースファイルへのデルタの適用による複数のファイルの抽出を表す概略ブロック図である。

【符号の説明】

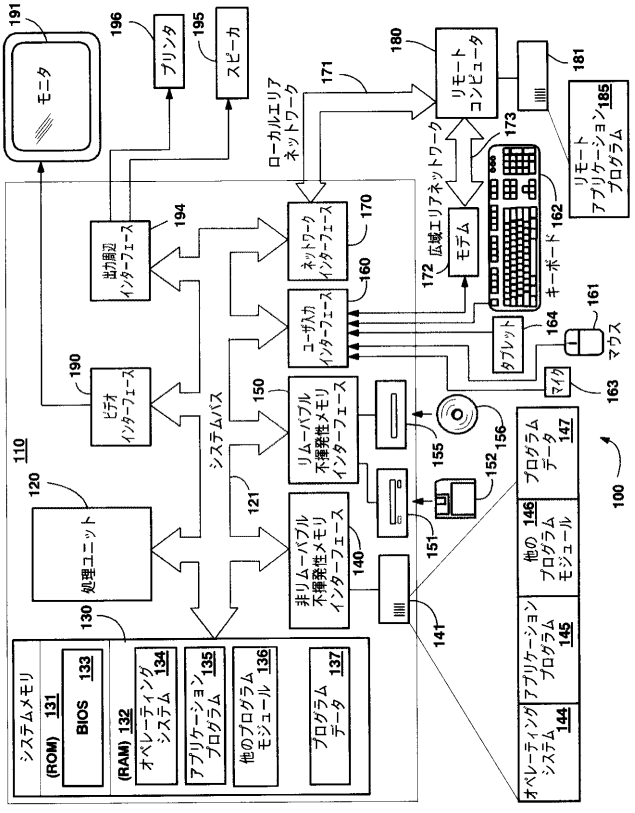
【００９７】

１２０　処理ユニット
１２１　システムバス
１３０　システムメモリ
１３４　オペレーティングシステム

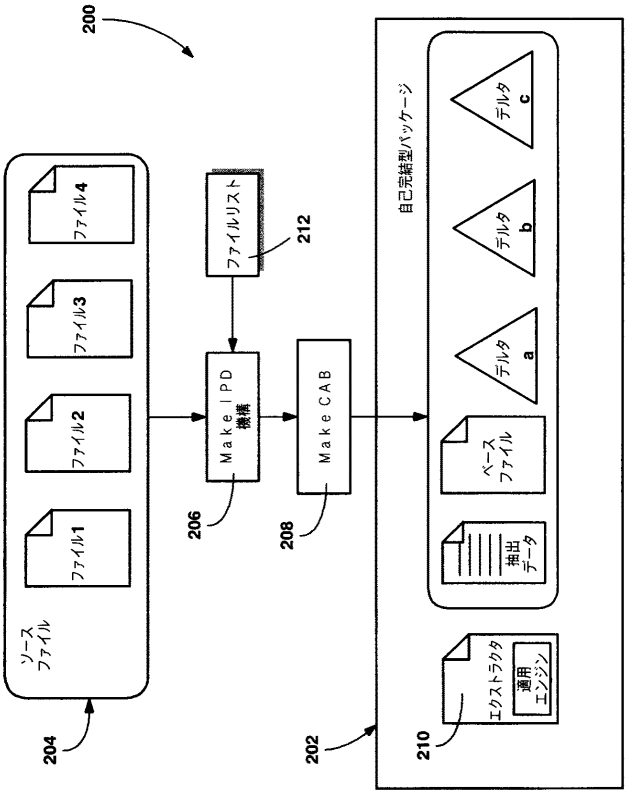
１３５　アプリケーションプログラム
１３６　他のプログラムモジュール
１３７　プログラムデータ
１４０　非リムーバブル不揮発性メモリインターフェース
１４４　オペレーティングシステム
１４５　アプリケーションプログラム
１４６　他のプログラムモジュール
１４７　プログラムデータ
１５０　リムーバブル不揮発性メモリインターフェース
１６０　ユーザ入力インターフェース
１６１　マウス
１６２　キーボード
１６３　マイク
１６４　タブレット
１７０　ネットワークインターフェース
１７１　ローカルエリアネットワーク
１７２　モデム
１７３　広域エリアネットワーク
１８０　リモートコンピュータ
１８５　リモートアプリケーションプログラム
１９０　ビデオインターフェース
１９１　モニタ
１９４　出力周辺インターフェース
１９５　スピーカ
１９６　プリンタ
２０２　自己完結型パッケージ
２０４　ソースファイル
２０６　ＭａｋｅＩＰＤ機構
２０８　ＭａｋｅＣＡＢ
２１０　エクストラクタ
２１２　ファイルリスト
３０４　ＧＵＩ／コマンドライン
３０６　パーサー
３０８　ソースファイル
３１２　繰り返し構成要素
３１４　デルタ作成エンジン
３１６　デルタ
３１６　サイズの有向グラフ
３１８　最小スパニングツリー計算
３２０　最小スパニングツリー
３２２　最小スパニングツリー列挙
３２４　リンクリスト
３２６　マニフェスト生成
３２８　マニフェスト
３３０　命令ファイル生成
３３２　命令ファイル
３３４　コンプレッサ／パッケージャ
４２０　最小スパニングツリー
５０２　自己完結型パッケージ
５０４　抽出機構

５０６　一時ディレクトリ
５０８　ベースファイル
５１０　デルタ
５１２　解凍（非ベース）ファイル
５１４　マニフェスト
５１６　適用エンジン
５１８　セットアッププログラム
５２０　合成ファイル
５２２　インストール済みファイル

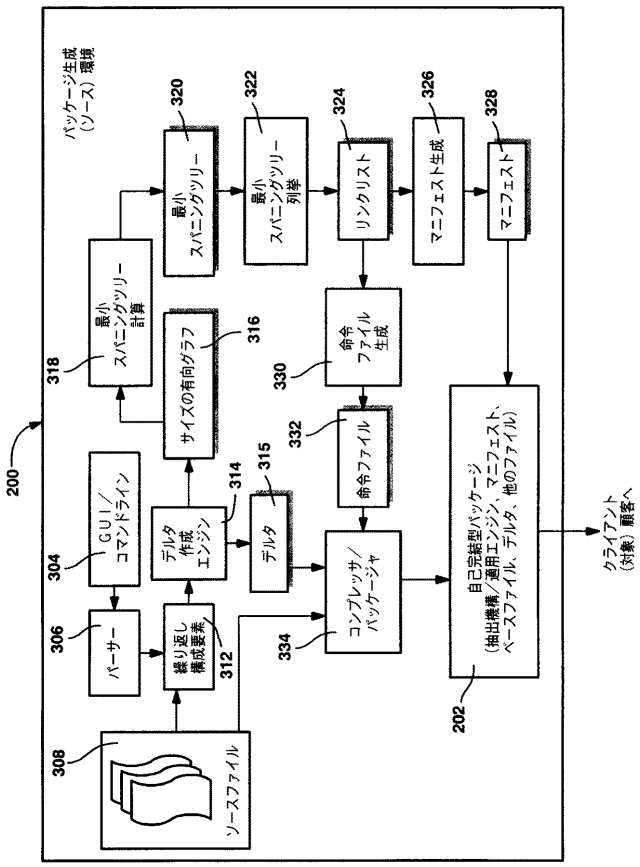
【図１】



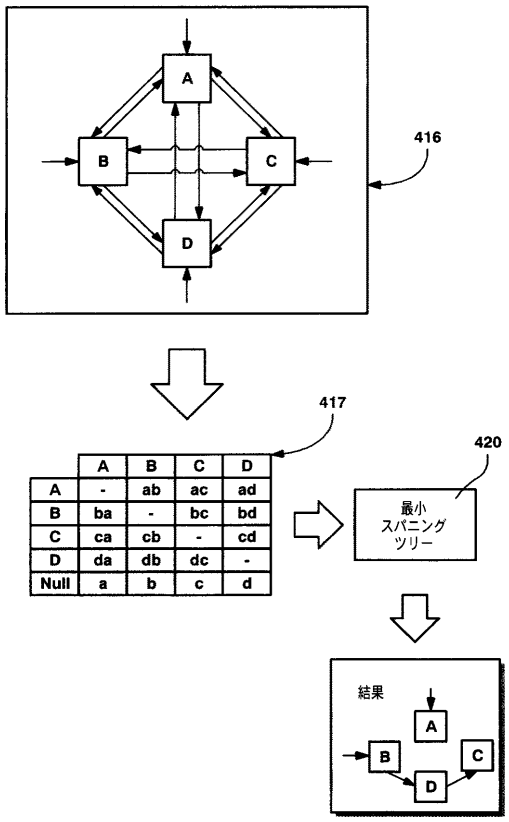
【図２】



【図３】

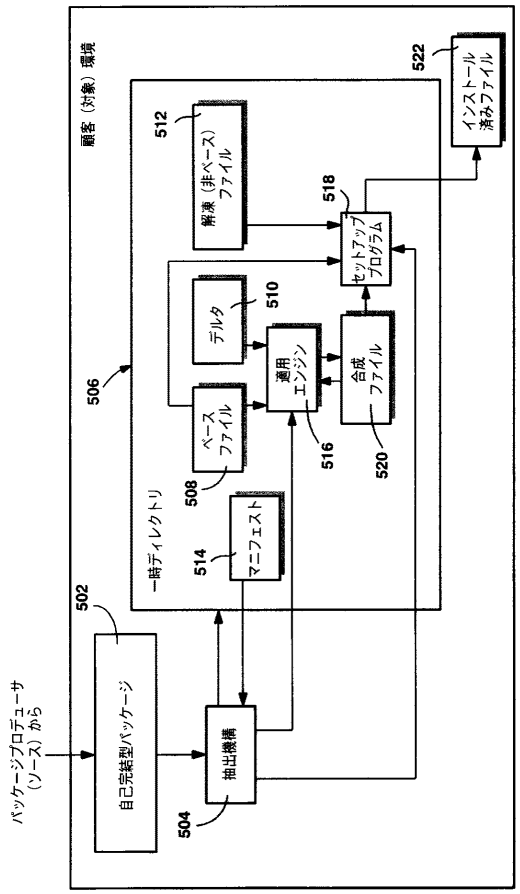


【図４】

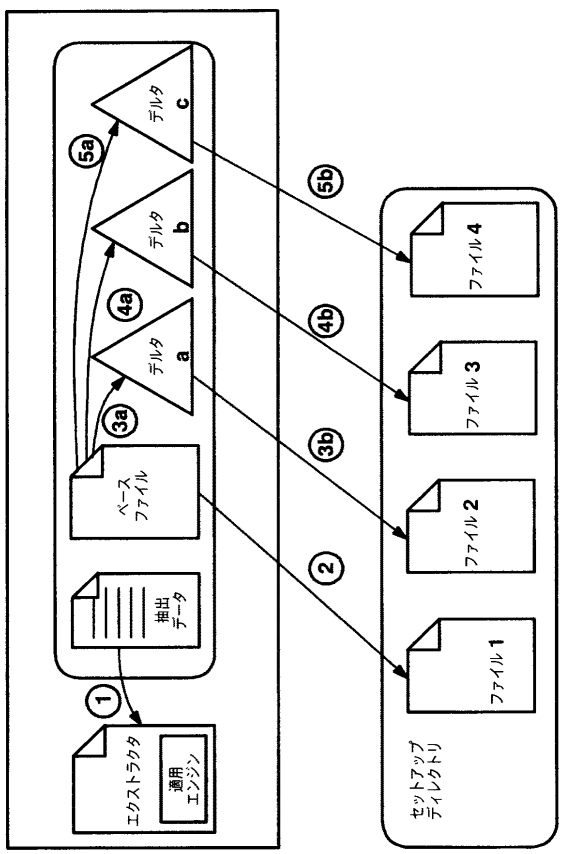


	A	B	C	D
A	-	ab	ac	ad
B	ba	-	bc	bd
C	ca	cb	-	cd
D	da	db	dc	-
Null	a	b	c	d

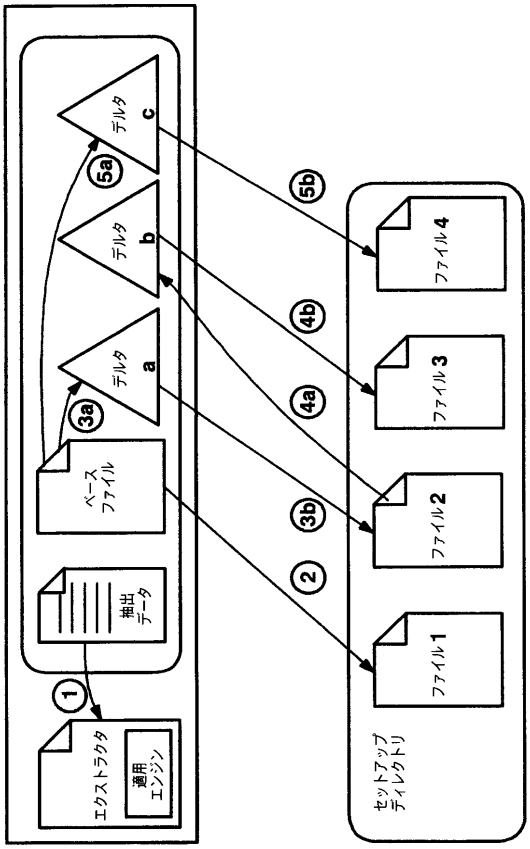
【図５】



【図６】



【図７】


エクストラクタ
適用エンジン
抽出データ
ベースファイル
デルタa
デルタb
デルタc
セットアップディレクトリ
ファイル1
ファイル2
ファイル3
ファイル4
①
②
3a
3b
4a
4b
5a
5b

フロントページの続き

(72)発明者　マノクマー　エイチ．シェンデ
アメリカ合衆国　９８０５２　ワシントン州　レッドモンド　１６８　コート　ノースイースト　４６２９
(72)発明者　マイケル　ブイ．スリガー
アメリカ合衆国　９８０７５　ワシントン州　サマミッシュ　サウスイースト　２３　ストリート　１９７１６
(72)発明者　トーマス　マクガイアー
アメリカ合衆国　７８６２８　テキサス州　ジョージタウン　ハリー　コート　１０２

Ｆターム(参考) 5B076 AB09
5B082 GA01 GA04