(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6316694号
(P6316694)
(45)発行日　平成30年4月25日(2018.4.25)　(24)登録日　平成30年4月6日(2018.4.6)

(51)Int.Cl.　F I
G06F 9/54 (2006.01)　G06F 9/46 480F

請求項の数 16 (全 16 頁)

(21)出願番号　特願2014-152967(P2014-152967)
(22)出願日　平成26年7月28日(2014.7.28)
(65)公開番号　特開2015-43202(P2015-43202A)
(43)公開日　平成27年3月5日(2015.3.5)
審査請求日　平成29年7月13日(2017.7.13)
(31)優先権主張番号　14/010,070
(32)優先日　平成25年8月26日(2013.8.26)
(33)優先権主張国　米国(US)

早期審査対象出願

(73)特許権者　510149482
ヴイエムウェア　インコーポレイテッド
ＶＭｗａｒｅ，Ｉｎｃ．
アメリカ合衆国　９４３０４　カリフォルニア州　パロ　アルト　ヒルビュー　アベニュー　３４０１
(74)代理人　100105957
弁理士　恩田　誠
(74)代理人　100068755
弁理士　恩田　博宣
(74)代理人　100142907
弁理士　本田　淳

最終頁に続く

(54)【発明の名称】クラウドスケールの異種データセンタ管理インフラストラクチャ

(57)【特許請求の範囲】
【請求項１】
コンピュータシステムにおいて、オペレーティング・システム・カーネルがユーザスペース・ゲスト・アプリケーションと通信するための方法であって、
複数の登録ハンドルを複数のＡＰＩ署名にマッピングするためのデータを含むデータストアを提供することであって、前記複数のＡＰＩ署名の各々は、少なくとも１つのコールバックを有し、前記データストアは、前記コンピュータシステムのカーネルスペース内のカーネルスペーストランスポートに存在する、前記データストアを提供すること、
前記データストアで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に任意の登録ハンドルを対応させて、前記データストアから前記任意の登録ハンドルに対応させた前記関連付けられたＡＰＩ署名を取得すること、
前記データストアから取得されたＡＰＩ署名内の前記コールバックに対応する記述子を前記オペレーティング・システム・カーネルから取得すること、
前記記述子を前記カーネルスペーストランスポートに記憶すること、
前記コンピュータシステムのユーザスペースに存在するユーザスペーストランスポートに前記記述子を送ること、
前記ユーザスペース・ゲスト・アプリケーションに対する前記記述子を、前記ユーザスペーストランスポートを介して処理することを備える、方法。
【請求項２】
前記ユーザスペース・ゲスト・アプリケーションが前記コールバックを呼び出す、請求

項１に記載の方法。
【請求項３】
前記ＡＰＩ署名が、名前と、バージョンと、前記少なくとも１つのコールバックの各々に対応するコールバックパラメータのリストと、パラメータセマンティクスとを含む、請求項１に記載の方法。
【請求項４】
前記オペレーティング・システム・カーネルおよび前記ユーザスペース・ゲスト・アプリケーションが仮想マシン上で動作する、請求項１に記載の方法。
【請求項５】
前記カーネルスペーストランスポートおよび前記ユーザスペーストランスポートがキャラクタ装置ノードで実行される、請求項１に記載の方法。
【請求項６】
前記記述子を前記ユーザスペーストランスポートに送ること、
前記ＡＰＩ署名と関連付けられている対応するキャラクタ装置ノードでファイルを開くためにスリープ解除リクエストを送ること、
前記キャラクタ装置ノードのポーリング状態を調べること、
前記ユーザスペーストランスポートを介して前記記述子を取得すること、
をさらに備える、請求項５に記載の方法。
【請求項７】
ホストコンピュータのオペレーティング・システム・カーネルにあるカーネルモジュールとユーザスペース・ゲスト・アプリケーションとの間での通信方法であって、
ユーザスペースに存在するユーザスペーストランスポート内のライブラリで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に任意の登録ハンドルを対応させること、
前記ＡＰＩ署名内の前記コールバックに対応する記述子を前記ユーザスペース・ゲスト・アプリケーションから取得すること、
前記ユーザスペーストランスポートに前記記述子を記憶すること、
カーネルスペースに存在するカーネルスペーストランスポートに前記記述子を送ること、
前記記述子と、前記カーネルモジュールに対するパラメータとを、前記カーネルスペーストランスポートを介して処理することを備える、方法。
【請求項８】
前記カーネルモジュールが前記コールバックを呼び出す、請求項７に記載の方法。
【請求項９】
前記ユーザスペーストランスポートおよび前記カーネルスペーストランスポートがキャラクタ装置ノードで実行される、請求項７に記載の方法。
【請求項１０】
コンピュータシステムであって、
プロセッサと、
メモリと、
カーネルスペースにオペレーティング・システム・カーネルを有するオペレーティング・システムと、
前記カーネルスペースに存在するカーネルスペーストランスポートと、
前記カーネルスペーストランスポートに存在するデータストアであって、登録ハンドルをＡＰＩ署名にマッピングするためのデータを含む前記データストアと、
ユーザスペース内のゲストアプリケーションと、
前記ユーザスペースに存在するユーザスペーストランスポートと、
前記プロセッサで実行された場合に、前記オペレーティング・システム・カーネルが前記ゲストアプリケーションと通信するための処理を実行する前記メモリ内のアプリケーションとを備える前記コンピュータシステムにおいて、前記処理は、

前記データストアで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に任意の登録ハンドルを対応させること、
前記ＡＰＩ署名内の前記コールバックに対応する記述子を前記オペレーティング・システム・カーネルから取得すること、
前記記述子を前記カーネルスペーストランスポートに記憶すること、
前記ユーザスペースに存在する前記ユーザスペーストランスポートに前記記述子を送ること、
前記ゲストアプリケーションに対する前記記述子を、前記ユーザスペーストランスポートを介して処理することを備える、コンピュータシステム。
【請求項１１】
前記ゲストアプリケーションが前記コールバックを呼び出す、請求項１０に記載のコンピュータシステム。
【請求項１２】
前記ＡＰＩ署名が、名前と、バージョンと、前記少なくとも１つのコールバックの各々に対応するコールバックパラメータのリストと、パラメータセマンティクスとを含む、請求項１０に記載のコンピュータシステム。
【請求項１３】
前記オペレーティング・システム・カーネルおよび前記ユーザスペースのゲストアプリケーションが仮想マシンで動作する、請求項１０に記載のコンピュータシステム。
【請求項１４】
前記カーネルスペーストランスポートおよび前記ユーザスペーストランスポートがキャラクタ装置ノードで実行される、請求項１０に記載のコンピュータシステム。
【請求項１５】
前記記述子を前記ユーザスペーストランスポートに送ることは、
前記ＡＰＩ署名と関連付けられている対応するキャラクタ装置ノードでファイルを開くためにスリープ解除リクエストを送ること、
前記キャラクタ装置ノードのポーリング状態を調べること、
前記ユーザスペーストランスポートを介して前記記述子を取得することをさらに含む、請求項１４に記載のコンピュータシステム。
【請求項１６】
プロセッサによって実行された場合に、請求項１～９のいずれか一項に記載の方法を実行するコードを記憶している非一時的なコンピュータ可読記憶媒体。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、概してコンピュータおよびコンピュータ装置の分野に関するものであり、特に、データセンタにおけるコンピュータの管理およびコンピュータ装置内での通信の提供の分野に関するものである。
【背景技術】
【０００２】
オペレーティング・システムのユーザスペースおよびカーネルスペースにおけるコンポーネント間通信は、データセンタ管理における既知の問題である。この問題は特に、スケーラブルなコンピューティングプラットフォーム向けの管理ソフトウェア、ドライバ、モジュール、ネットワークインターフェイスなどを開発するサードパーティのソフトウェアベンダにとっての課題である。
【０００３】
例えば、クラウド・コンピューティングの分野では、クラウド展開のスケーラビリティが高まるにつれて、ホスト管理施設が重要なアーキテクチャコンポーネントになりつつある。そのため、ホスト管理施設もスケーラブルでなければならない。ホスト－インベント

リサービスと統合された共通情報モデル（CIM : Common Information Model）アーキテクチャを用いて、管理者は、単一ステーションからホストに管理処理を発行し、その処理を、各ホストでベンダ固有のＣＩＭプロバイダソフトウェアにサービスさせることができる。ただしＣＩＭアーキテクチャは、ＣＩＭプロバイダがサポートするホスト－クライアント間インターフェイスをサードパーティの開発者が正式に規定することを許可してはいるものの、プロバイダ処理をサービスするのに必要なプロバイダ－カーネル間インターフェイスを規定するための案内を提供していない。

【０００４】

さらに、仮想化プラットフォームでは、サードパーティの開発者がユーザスペース－カーネルスペース間通信を実装するため主たる方法が、キャラクタ装置（character devices）を使用するものである。キャラクタ装置は、ファイルシステムにファイルノードとして表示され、従来のファイル操作（例えば、ｏｐｅｎ、ｒｅａｄ、ｗｒｉｔｅ、ｐｏｌｌ、ｉｏｃｔｌ、ｃｌｏｓｅなど）をサポートする。キャラクタ装置インターフェイスをエクスポートするカーネルモジュールが、一連のファイル操作ハンドラを登録する。こうして、ファイルシステム内のノードに対応するファイルに対するファイル操作を実行するユーザスペースアプリケーションが、キャラクタ装置インターフェイスをトリガする。

【発明の概要】

【発明が解決しようとする課題】

【０００５】

キャラクタ装置を直接操作する際に生じる１つの問題が、低次のセマンティクス（semantics）を処理する際に生じる構造および一貫性の欠如である。つまり、多くの開発者は、自身専用の（そして機能的に同等または同様であることの多い）インターフェイスを即興で記述し、共通の低次トランスポート処理およびプロトコル（例えば、リクエストの復号、パラメータの整理、イベントの送出）を実装して、カーネルスペース内の関数を簡潔に呼び出す（あるいは逆に、カーネルスペースからユーザスペース内の関数を簡潔に呼び出す）。一般に、かかるインターフェイスは、低次のビット操作を必要とし、カーネルスペースとユーザスペースとの間での非常に限られた通信方法しか提供しない。加えて、低次の処理をプログラムすることが困難であることから、でき上がったアプリケーションは、エラーを起こしやすい場合がある。ファイルインターフェイスおよびソケットインターフェイスを通じて通信を実装することは可能だが、両手法とも、構造化された意味データおよび処理を、ユーザとカーネルの境界を超えて送ることができないという根本的な制約を有する。キャラクタ装置の場合と同様、ファイルインターフェイスとソケットインターフェイスとを使用するには、両側のソフトウェアがデータをパックおよびアンパックする必要がある。

【０００６】

さらに、仮想化環境では、カーネルへのアクセスがホストコンピュータのファイルシステムを経由し得るため、開発者がカーネルと対話するために記述するアプリケーションがファイルシステムに晒される。これにより、すでに制約の多い仮想化環境の外部への管理アプリケーションの移植性が制限され、リッチなユーザアプリケーション（rich user applications）をほとんどサポートしない可能性がある。また、複雑なアプリケーションがリソースを使用するので、ホストコンピュータが仮想マシンを効率良く実行および管理できなくなり得る。

【０００７】

加えて、サードパーティの開発者がキャラクタ装置を構築する際の基盤とするアプリケーション・プログラミング・インターフェイス（API : application programming interfaces）（またはファイルインターフェイスおよびソケットインターフェイス）は、バージョニング（versioning）を生来的にサポートしていないため、時間が経過して後継のＡＰＩバージョンが実装されたときに、ユーザコンポーネントとカーネルコンポーネントとの間で互換性の問題が生じる。例えば、特定のサードパーティベンダによるＣＩＭプロバイダであれば、複数のドライバをサポートする必要があり得るが、ドライバ自体が、ファー

ムウェアのアップグレード後など、時間の経過とともに進化して新機能を追加し得る。このような場合に、そのＣＩＭプロバイダには、低次のインターフェイスに基づいて構築されたどの高次の処理がサポートされているかを検出する方法がない。データ構造の定義が少しでも変わると、現行のソリューションが扱いにくくなり、破損につながる。現在は、破損を防ぎ、互換性を維持するのに、カーネルスペースアプリケーションとユーザスペースアプリケーションが厳格に統合および一致している必要がある。

【課題を解決するための手段】

【０００８】

本明細書に提示された一実施形態は、オペレーティング・システム・カーネルがユーザスペース・ゲスト・アプリケーションと通信するための方法を含んでいる。この方法は、登録ハンドルをＡＰＩ署名にマッピングするためのデータを含むデータストアで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に所与の登録ハンドルを一致させることを概して含み得る。データストアは、カーネルスペースに存在する第１のトランスポートに存在する。この方法は、ＡＰＩ署名内の少なくとも１つのコールバックのうちの１つに対応する記述子をカーネルから取得すること、第１のトランスポートに記述子を記憶することも含み得る。この方法は、ユーザスペースに存在する第２のトランスポートに記述子を送ることを含み得る。この方法は、ゲストアプリケーションに対する記述子を第２のトランスポートを介して処理することを含み得る。

【０００９】

別の実施形態は、第１のホストコンピュータのオペレーティング・システム・カーネル内のアプリケーションとユーザスペース・ゲスト・アプリケーションとの間での抽象通信の方法を含んでいる。この方法は、ユーザスペースに存在する第１のトランスポート内のライブラリで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に所与の登録ハンドルを一致させることを概して含み得る。この方法は、ＡＰＩ署名内の少なくとも１つのコールバックのうちの１つに対応する記述子をゲストアプリケーションから取得することも含み得る。この方法は、第１のトランスポートに記述子を記憶すること、第２のトランスポートに記述子を送ることも含み得る。第２のトランスポートは、カーネルスペースに存在する。この方法は、記述子と、カーネルに対するパラメータとを、第２のトランスポートを介して処理することも含み得る。

【００１０】

一実施例では、オペレーティング・システム・カーネルがユーザスペース・ゲスト・アプリケーションと通信するための方法が提供されており、この方法は、登録ハンドルをＡＰＩ署名にマッピングするためのデータを含むデータストアであって、カーネルスペースにある第１のトランスポートに存在する前記データストアで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に所与の登録ハンドルを一致させること、ＡＰＩ署名内の少なくとも１つのコールバックのうちの１つに対応する記述子をカーネルから取得すること、第１のトランスポートに記述子を記憶すること、ユーザスペースに存在する第２のトランスポートに記述子を送ること、ゲストアプリケーションに対する記述子を第２のトランスポートを介して処理することを含む。

【００１１】

一実施例では、プロセッサによって実行されるコードを記憶している非一時的なコンピュータ可読記憶媒体が存在し、このコードは、プロセッサによって実行されると、オペレーティング・システム・カーネルがユーザスペース・ゲスト・アプリケーションと通信するための処理であって、登録ハンドルをＡＰＩ署名にマッピングするためのデータを含むデータストアで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に所与の登録ハンドルを一致させることであって、データストアがカーネルスペースにある第１のトランスポートに存在すること、ＡＰＩ署名内の少なくとも１つのコールバックのうちの１つに対応する記述子をカーネルから取得すること、記述子に第１のトランスポートを記憶すること、ユーザスペースに存在する第２のトランスポートに記述子を送ること、ゲストアプリケーションに対する記述子を第２のトランスポートを介して処理することを

含む処理を実行する。
【００１２】
一実施例では、プロセッサと、プロセッサによって実行されると、オペレーティング・システム・カーネルがユーザスペース・ゲスト・アプリケーションと通信するための処理であって、登録ハンドルをＡＰＩ署名にマッピングするためのデータを含むデータストアであって、カーネルスペースにある第１のトランスポートに存在する前記データストアで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に所与の登録ハンドルを一致させること、ＡＰＩ署名内の少なくとも１つのコールバックのうちの１つに対応する記述子をカーネルから取得すること、第１のトランスポートに記述子を記憶すること、ユーザスペースに存在する第２のトランスポートに記述子を送ること、ゲストアプリケーションに対する記述子を第２のトランスポートを介して処理することを含む処理を実行するアプリケーションをホストしているメモリとを備えるシステムが提供される。
【００１３】
一実施例では、ゲストアプリケーションが少なくとも１つのコールバックのうちの１つを呼び出す。
一実施例では、ＡＰＩ署名が、名前と、バージョンと、少なくとも１つのコールバックの各々に対応するコールバックパラメータのリストと、パラメータセマンティクスと、を含む。
【００１４】
一実施例では、オペレーティング・システム・カーネルおよびゲストアプリケーションが仮想マシン上で動作する。
一実施例では、第１および第２のトランスポートがキャラクタ装置で実行される。
【００１５】
一実施例では、記述子を第２のトランスポートに送ることが、ＡＰＩ署名と関連付けられている対応するキャラクタ装置でファイルを開くためにスリープ解除リクエストを送ること、キャラクタ装置のポーリング状態を調べること、第２のトランスポートを介して記述子を取得することをさらに含む。
【００１６】
一実施例では、ホストコンピュータのオペレーティング・システム・カーネルに存在するアプリケーションとユーザスペース・ゲスト・アプリケーションとの間での抽象通信の方法が提供され、この方法は、ユーザスペースに存在する第１のトランスポート内のライブラリで、少なくとも１つのコールバックを有する関連付けられたＡＰＩ署名に所与の登録ハンドルを一致させること、ＡＰＩ署名内の少なくとも１つのコールバックの１つに対応する記述子をゲストアプリケーションから取得すること、第１のトランスポートに記述子を記憶すること、カーネルスペースに存在する第２のトランスポートに記述子を送ること、記述子と、カーネルに対するパラメータとを第２のトランスポートを介して処理することを含む。
【００１７】
一実施例では、カーネルアプリケーションが少なくとも１つのコールバックの１つを呼び出す。
一実施例では、第１および第２のトランスポートがキャラクタ装置で実行される。
【図面の簡単な説明】
【００１８】
【図１】一実施形態にかかる、カーネル－ユーザ間通信メカニズムを用いて構成された仮想化環境におけるホスト・コンピュータ・システムのブロック図例を表す。
【図２】一実施形態にかかる、ホスト計算システムのユーザスペースおよびカーネルスペースにおける通信メカニズムのレイアウト例を表す。
【図３】一実施形態にかかる、通信メカニズムを使用してカーネルスペースからユーザスペース内のコールバックを呼び出すための方法図を表す。
【図４】一実施形態にかかる、通信メカニズムを使用してユーザスペースからカーネルス

ペース内のコールバックを呼び出すための方法図を表す。

【発明を実施するための形態】

【００１９】

本明細書に開示された実施形態は、カーネルモジュールとユーザスペースアプリケーション（user-space applications）との間で通信を行うための高次のプリミティブ（high-level primitives）を提供する。一実施形態では、通信メカニズムが、コールバックパラメータをコピーし、非同期配信処理（handles asynchronous delivery）して、ユーザスペースおよびカーネルスペースコンポーネントが、互換性のあるアプリケーション・プログラミング・インターフェイス（ＡＰＩ）のバージョンを確実に使用するようにする。ユーザスペースおよびカーネルスペースに通信メカニズムのトランスポートコンポーネントが存在し、（例えばコールバックを呼び出す際に）ユーザスペースアプリケーションおよびカーネルスペースモジュールが表示され、互いに直接通信できるようにする。さらに、各種実施形態は、開発者がイベントを送信し、コールバックをディスパッチするためのヘッダーファイルとマクロとを自動的に発する目的で使用する定義基準（definition standard）を提供する。

【００２０】

例えば、仮想マシンを起動および実行するホスト・コンピュータ・ノードのクラスタネットワークを含む仮想化されたコンピューティング環境では、サードパーティの開発者が、ハイパーバイザカーネルと通信する必要のあるアプリケーションを記述する。概して、これらのアプリケーションを実装するには、開発者が、カーネルから情報を取得し、カーネルに情報を提供するための共通トランスポート処理を実装する必要がある（例えば、デバッグまたはパラメータ化を実行する多くのユーザスペースアプリケーションは、カーネルと通信する必要がある）。

【００２１】

提供された通信メカニズムは、ユーザスペースアプリケーションとオペレーティング・システム・カーネルとの間での通信を簡便化する。このメカニズムを使用して、開発者は、低次のビット演算を必要とする共通トランスポート処理をプログラミングすることなく、（カーネルまたはアプリケーションのどちらでも）実行時に実行できる管理処理を定義し得る。さらに、このメカニズムは複数のＡＰＩバージョンをサポートできるため、サードパーティのユーザスペースアプリケーションの開発者およびカーネルモジュールの開発者は、要件を結合することによって制約を減らし、将来の実装形態を一方向的に追求することができる。

【００２２】

一実施形態では、互換性を持つユーザスペースコンポーネントと通信するカーネルが、トランスポートモジュールに管理ＡＰＩを登録する。ユーザスペースコールバックを呼び出すために、カーネルモジュール内のコードがトランスポートＡＰＩを呼び出し、登録されたＡＰＩハンドルと、呼び出されるコールバック識別子と、コールバックに対するパラメータとを、パラメータとして渡す。モジュールは、アンロードされると、各々の登録済みハンドルを用いてトランスポートＡＰＩを呼び出し、トランスポートからＡＰＩインスタンスの登録を解除する。同様に、カーネルモジュールと通信するユーザスペースアプリケーションが、ユーザスペーストランスポートにＡＰＩ署名を登録する。成功すると、不透過ハンドル（opaque handle）が戻される。アプリケーションは、カーネルスペースでコールバックを呼び出す際にこのハンドルを使用する。アプリケーションが終了すると、カーネルスペーストランスポートは終了を検出し、カーネル内の残りの状態をクリーンアップする。

【００２３】

一実施形態では、通信メカニズムが、複数のＡＰＩバージョンに対するサポートも提供する。つまり、提供されたフレームワークにより、アプリケーションがバージョンを特定のＡＰＩ署名と関連付けることができる。例えば、カーネルドライバからネットワークに関する統計情報を定期的に取得するゲストアプリケーションの場合を検討する。開発者が

ドライバまたはアプリケーションを更新すると、ドライバとアプリケーションは時間とともに進化し得る。例えば、ドライバの新しい繰り返しにより、サポートされる処理が増加したり、さまざまな処理セマンティクスが必要となったりし得る。これらの新機能により、新バージョンのアプリケーションのリリース前に旧バージョンのユーザスペースアプリケーションが陳腐化する（その逆もあり得る）が、新バージョンのアプリケーションのリリースは、時間がかかる場合や、アプリケーションとドライバとの間での密な結合が必要となる場合がある。しかし、通信メカニズムは、かかる懸念を軽減する。

【００２４】

引き続き先述の例に関し、バージョン１．０およびバージョン２．０という２つのバージョンのカーネルドライバが存在するものと仮定する。新バージョンである２．０は、追加処理に対するサポートを提供する。一般に、このドライバと通信するユーザスペースアプリケーションの開発者は、ドライバのバージョン２．０と通信できるようにするために、アプリケーションを更新する必要がある。ただし、開発者がバージョン１．０と通信するソフトウェアを引き続きサポートしたい状況が存在し得る。一実施形態では、カーネルモジュールが、バージョン１．０のハンドルを有するバリアント（variant）と、バージョン２．０のバンドルを有する別のバリアントというＡＰＩ署名の２つのバリアントを同時に登録することができる。その後、実行時に、どのハンドルおよびセマンティクスがどのソフトウェアで関連付けたられたかを、システムが自動的に解決する。この手法の下では、ユーザスペースアプリケーションの異なるバージョンのセマンティクスをサポートするカーネルにおいて異なる機能をもたらす。実行時には、カーネルスペーストランスポートが、特定バージョンに対する問い合わせに関係なくリクエストを処理する。

【００２５】

次に、いくつかの実施形態を詳しく参照する。これらの実施形態の具体例が添付の図に表されている。なお、該当する場合には、類似または同様の参照符号が図内で使用され、類似または同様の機能を表し得る。これらの図は、各種実施形態を説明目的で表しているに過ぎない。当業者であれば、以降の説明から、本明細書に例示された構造および方法の代替実施形態が、本明細書に記載された原則から逸脱しない範囲で用いられ得るということを容易に認識するであろう。

【００２６】

以下、仮想化環境で実施されている実施形態について言及する。かかる言及は、本明細書に記載された実施形態に対する理解を徹底するために提供されるものである。しかし、当業者にとっては、これらの実施形態が他のコンピューティング環境で適用可能であることが明らかであろう。同様に、これらの実施形態に対する理解を徹底するために、多数の具体的な詳細情報が提供されている。当業者であれば、これらの具体的な詳細情報の一部がなくてもこれらの実施形態が実施され得ることを認めるであろう。他の事例では、新規性のある本開示の態様を無用にわかりにくくするのを避けるために、周知の処理および実施に関する詳細については説明していない。

【００２７】

図１は、一実施形態にかかる、カーネル－ユーザ間通信メカニズムで構成されている仮想化環境におけるホスト・コンピュータ・システム１００のブロック図例を表す。ホスト・コンピュータ・システム１００は、デスクトップ、ラップトップ、またはｘ８６アーキテクチャプラットフォームなどサーバグレードのハードウェアプラットフォーム１０２に構築され得る。ハードウェアプラットフォーム１０２は、１つ以上の中央演算処理装置（ＣＰＵ）１０３と、ホスト物理メモリ１０４と、ホスト・コンピュータ・システム１００をネットワークに接続するネットワーク・インターフェイス・コントローラおよびホスト・コンピュータ・システム１００を永続的記憶装置に接続する１つ以上のホスト・バス・アダプタなど他の標準的なハードウェアコンポーネントと、を備える。一実施形態では、ホスト・コンピュータ・システム１００がＶＭＷａｒｅ　ＥＳＸｉホストである。

【００２８】

ハードウェアプラットフォーム１０２は、ハイパーバイザ１１４を備える。ハイパーバ

イザ１１４は、複数の仮想マシン実行スペース１１６をサポートしており、ＶＭプロセスが、対応するＶＭ１２０のインスタンスを生成するように実行され得る。各ＶＭ１２０について、ハイパーバイザ１１４が、仮想ＣＰＵとゲスト物理メモリなどエミュレートされたハードウェアを含む対応仮想ハードウェアプラットフォーム（すなわち仮想ハードウェアプラットフォーム１２２）を管理する。各仮想ハードウェアプラットフォーム１２２は、ゲスト・オペレーティング・システム（ＯＳ）（例えばゲストＯＳ１３２）のインストールをサポートする。各インスタンスで、ゲストＯＳ１３２は、例えば、仮想マシンの仮想ハードウェアプラットフォームに対するインターフェイスであるアプリ１１３など、仮想マシンで稼働するユーザレベルアプリケーションを提供する。

【００２９】

図１で仮想化コンポーネントについて説明する目的で使用されている各種用語、層、および分類は、それらの機能から逸脱しない範囲で別の言葉で言及され得るものと認識すべきである。例えば、仮想ハードウェアプラットフォーム１２２は、ハイパーバイザ１１４とそれぞれのＶＭとの間での処理を調整するのに必要な仮想システムサポートを実装する仮想マシンモニタ（VMM:virtual machine monitors）１４０の一部とみなされ得る。あるいは、仮想ハードウェアプラットフォーム１２２は、ＶＭＭ１４０とは別であると（例えば、かかるプラットフォームが仮想マシンのハードウェア・エミュレーション・コンポーネントを備えることから、対応する仮想マシンのコンポーネントと）もみなされ得る。そしてＶＭＭ１４０はハイパーバイザ１１４とは別であるとみなされ得る。使用され得るハイパーバイザ１１４の一例が、ＶＭｗａｒｅのｖＳｐｈｅｒｅ製品のコンポーネントとして含まれている。このコンポーネントは、カリフォルニア州パロアルトにあるＶＭｗａｒｅ社より市販されている。さらには、ホストされた仮想マシンシステムなど、他の仮想化されたコンピュータシステムが想定され、ハイパーバイザはホスト・オペレーティング・システムとともに実装されるということも認識すべきである。

【００３０】

図示のとおり、ゲストＯＳ１３２は、通信メカニズム１１８も備える。通信メカニズム１１８は、コールバック呼び出しなどでユーザスペースアプリケーションがカーネルと通信できるように、そして逆にカーネルがユーザスペースアプリケーションと通信できるようにする高次のインフラストラクチャである。メカニズム１１８は、ユーザスペースアプリケーションとカーネルとの間で送受信されるデータをインターセプトし、コールバックを実行するため、あるいはユーザスペースまたはカーネルをまたがってイベントを送信するために必要な低次キャラクタ装置処理を実行する。

【００３１】

別の実施形態では、通信メカニズム１１８がハイパーバイザ１１４（例えばＶＭＭ１４０内）に位置し得る。この場合には、通信メカニズムによって提供される抽象化により、ユーザスペースアプリケーション（例えば管理ソフトウェア）が、ハイパーバイザ１１４内にある仮想マシンのゲストＯＳ１３２で、管理プラットフォームの一部としてリモートで稼働することができる。それでも、通信メカニズム１１８は、場所に関係なく、まるでアプリケーションがカーネルモジュールでホストと通信しているかのように同じセマンティクスを提供する。この手法により、データセンタ用のスケーラブル管理アプリケーションを記述することに対する障壁が下がる。加えて、ハイパーバイザ１１４内にメカニズム１１８を設けるのではなく、ゲストＯＳ１３２で実装することにより、カーネルへのアクセスを必要とする管理アプリケーションが、そのためにホスト・コンピュータ・システム１００のファイルシステムにアクセスする必要が一切なくなる。さらに、この手法により、ホスト・コンピュータ・サーバ１１４がＶＭ１１２を管理するためのリソースが解放される。

【００３２】

図２は、一実施形態にかかる、ホスト計算システム２００のユーザスペース２２５およびカーネルスペース２３０における通信メカニズムのレイアウト例を表す。図示のとおり、アプリケーション２０５はユーザスペース２２５に存在し、カーネルモジュール２２０

はカーネルスペース２３０に存在する。通信メカニズムは、ユーザスペース２２５内のアプリケーション２０５とカーネルスペース２３０内のカーネルモジュール２２０との間での通信を簡便化するためのユーザスペーストランスポート２１０とカーネルスペーストランスポート２１５とを含み得る。この通信メカニズムを使用して、ユーザスペースアプリケーション２０５およびカーネルモジュール２２０が、まるでコールバックを直接呼び出しているかのように通信することができる。具体的には、トランスポート２１０およびトランスポート２１５が、パラメータをコピーし、非同期イベントの配信を処理し、ユーザスペース２２５およびカーネルスペース２３０内のコンポーネントが使用するＡＰＩバージョンの互換性を維持し得る。

【００３３】

通信メカニズムを使用するアプリケーションを記述するサードパーティの開発者が、トランスポートコンポーネント２１０，２１５にとって認識可能なＡＰＩ署名を最初に定義する。このＡＰＩ署名により、ユーザスペースまたはカーネルスペースアプリケーションがこのメカニズムを使うことができる。一実施形態では、このＡＰＩ署名が、名前、バージョン、呼び出し可能なコールバックのリスト、コールバックがユーザスペースに存在するか、あるいはカーネルスペースに存在するか、各コールバックが同期型か非同期型か、コールバックごとのパラメータの量、パラメータが入力パラメータか、出力パラメータか、あるいは入力／出力パラメータか、そして各々のパラメータのサイズ、を含み得る。加えて、このＡＰＩは、互換性のあるユーザスペースＡＰＩのバージョンをカーネルモジュールが指定できるようにするバージョン適合カーネルコールバックも含み得る。

【００３４】

初期化時に、カーネルモジュール２２０がカーネルスペーストランスポート２１５に管理ＡＰＩを登録する。一実施形態では、通信メカニズムが、異なるタイプの複数のＡＰＩを同時にサポートし得る。例えば、カーネルモジュール２２０は、ベンダ固有のＡＰＩに加え、構成およびレポートＡＰＩの登録およびサービス提供もし得る。別例として、１つのドライバが、（例えば、新バージョンのＡＰＩの使用する後発バージョンのユーザスペースアプリケーションに対するサポートを追加するために）２つのバージョンの同一ＡＰＩをサポートし得る。同様に、ユーザスペースアプリケーション２０５がユーザスペーストランスポート２１０にＡＰＩを登録する。どちらの場合でも、アプリケーション２０５またはカーネルモジュール２２０がＡＰＩ署名を登録すると、トランスポート２１０またはトランスポート２１５がそれぞれ不透過登録ハンドル（opaque registration handle）を返す。通信メカニズムを使用するために、アプリケーションは、コールバック呼び出し中にハンドルをパラメータとして渡す。

【００３５】

一実施形態では、通信メカニズムが、Ｍユーザスペースアプリケーションとカーネルモジュールとの間でのＭ－Ｎ間通信をサポートする。例えば、特定のカーネルモジュールにリクエストを送信する際に、アプリケーションが、ＡＰＩ署名のパラメータとしてコールバック識別子を渡す。Ｍ－Ｎ間通信により、（例えば、ドライバがイベントをブロードキャストする必要がある状況で）カーネルからの非同期イベントの配信が簡便化される。

【００３６】

一実施形態では、通信メカニズムが、ホストコンピュータのファイルシステムで、キャラクタ装置ノードを使用して実装され得る。そのために、カーネルスペーストランスポート２１５は、インスタンスごとにキャラクタ装置ノードを登録する。例えば、３つのＡＰＩを登録する１つのドライバであれば、３つの装置ノードを登録し得る。それを受けて、ユーザスペーストランスポート２１０が、そのＡＰＩの名前および主要バージョンに一致するファイルシステムでキャラクタ装置ノードを特定する。一実施形態では、ユーザスペーストランスポート２１０が、カーネルスペーストランスポート２１５に位置しかつ利用可能なＡＰＩ名、ベンダ、バージョン、および対応ファイルを提供する辞書（dictionary）２１２に問い合わせることにより、部分一致を判断し得る。所与のファイルが開かれると、そのバージョン情報がリクエストされたものであることをユーザスペース内のライブ

ラリ２０８がカーネルに確認（すなわちハンドシェイク）する。一致を判断すると、ユーザスペーストランスポート２１０は、対応するキャラクタ装置に対するファイル記述子（file descriptor）を開いた状態に保ち、スレッドを作成してカーネルからの入来イベント（incoming event）を監視し、そのファイル記述子について説明するハンドルを返す。代替実施形態では、通信メカニズムが、カーネルおよびユーザソケットなど他のカーネルーユーザ間プロトコルを使用して実装され得る。

【００３７】

一実施形態では、ユーザスペーストランスポート２１０がライブラリ２０８を備える。アプリケーション２０５がライブラリ２０８と通信し、それを受けてライブラリ２０８がカーネルスペース２３０と通信する。ライブラリ２０８は、アプリケーション２０５によって開かれたキャラクタ装置ノードのリスナ（listeners for character device nodes）を維持する。ライブラリ２０８は、ユーザスペースアプリケーション２０５向けのデータをカーネルスペーストランスポート２１５から検出すると、そのデータの読み出しおよび検証を司り、適切な形態であることを確認した上で、ユーザスペースアプリケーション２０５向けのデータをアンロードする。加えて、ライブラリ２０８は、キャラクタ装置ノードに対する登録ハンドルのマッピングを維持する。別の実施形態では、登録ハンドルが、カーネルマッピングやＶＭｗａｒｅのＶＭＫｅｒｎｅｌシステム情報（VSI：VMKernel System Information）ノードなど、他の手段によってカーネルにマッピングし得る。

【００３８】

カーネルスペース２３０にアクセスするために、ユーザスペースアプリケーション２０５は、ライブラリ２０８にリンクし、カーネルスペーストランスポート２１５に存在する辞書２１２を通じて通信する。一実施形態では、辞書２１２が、ＡＰＩにより提供される処理を指定する登録済みＡＰＩ定義のリストを維持するデータストアである。このリストを維持するために、辞書２１２は、登録ハンドルを対応するＡＰＩ署名にマッピングする。ユーザスペーストランスポート２１０は、ライブラリ２０８を辞書２１２にリンクすることによって辞書２１２と通信して、サポートしているＡＰＩ署名を特定する。同様に、カーネルスペースアプリケーションも、辞書２１２によって内部的に提供されるハンドルを使用して、ハンドルと関連付けられたアプリケーションを特定し得る。

【００３９】

図３は、一実施形態にかかる、ホスト計算システムのカーネルスペースからユーザスペース内のコールバックを呼び出すための方法図を表す。コールバック構造は、コールバック情報記述子およびパラメータを含んでいる。コールバック情報記述子は、パラメータのセマンティクスと、コールバックによって動作する機能と、について記述し得る。このセマンティクスは、パラメータの数と、各パラメータのサイズと、パラメータを渡す目的で使用されるメモリの割り当て方を表すインジケータと、を含み得る。このセマンティクスは、これらのパラメータが入力パラメータか、入力／出力パラメータか、あるいは出力パラメータか、という情報も含み得る。例えば、セマンティクスがパラメータを入力パラメータと説明していれば、呼ばれた側（すなわちユーザスペースアプリケーション）は、コールバックが呼び出された後にデータを受信すべきである。別例として、セマンティクスが入力／出力パラメータと説明していれば、呼ばれた側は、データを受信した後にデータを修正し、その後そのデータを呼び出し側に返送する。あるいは、セマンティクスが出力パラメータと説明していれば、呼ばれた側は、データを生成し、そのデータを呼び出し側に出力する必要がある。

【００４０】

この方法はステップ３０５から始まる。ステップ３０５では、カーネルスペーストランスポートが、コールバック情報記述子およびパラメータ用の一時記憶域を割り当てる。カーネルスペースモジュールは、そのコールバック情報記述子およびパラメータを一時記憶域にコピーし、これらのパラメータをポインタとして呼ばれた側に提示する。ステップ３１０で、記述子およびパラメータがカーネルスペーストランスポートにコピーされた後、トランスポートは、ＡＰＩインスタンスのカーネル側用として使用されているキャラクタ

装置と関連付けられたすべてのオープンファイルに関するスリープ解除リクエストを送信する。これにより、イベントを待機しているユーザ側トランスポートでスリープ状態にあるすべてのスレッドがスリープ状態から復帰する。代替実施形態では、コールバックパラメータが可変長であることがあり、パラメータを呼び出すことにより、各パラメータ内の要素の数が符号化される。この場合、カーネルスペースモジュールは、データにポインタを渡すのではなく、要素の数を表すインジケータを含む記述子およびデータの始点にポインタを渡す。

【００４１】

ステップ３１５で、ユーザスペーストランスポートは、キャラクタ装置のポーリング状態を調べて、キャラクタ装置が読み出し可能か（すなわちイベントが利用可能か）どうかを判断する。ステップ３２０で、ユーザスペーストランスポートは、コールバックリクエストを読み出し、そのリクエストを処理して、コールバックに対応するパラメータを読み出す。その後ステップ３２５で、ユーザスペーストランスポートは、コールバックをディスパッチして、監視スレッド内のイベントを効果的に配信し、コールバックを呼び出す。

【００４２】

図４は、一実施形態にかかる、ホスト計算システムのユーザスペースからカーネルスペース内のコールバックを呼び出すための方法図を表す。既述のとおり、ユーザスペースアプリケーションがＡＰＩ署名を登録すると、カーネルスペーストランスポートは、ＡＰＩ定義および処理を維持している辞書を使用して、ユーザスペースおよびカーネルスペースでＡＰＩ署名にマッピングされている不透過ハンドルを返す。このライブラリによってカーネルへの接続が保証されるとともに、リクエストされたドライバまたはモジュールが存在することが確認される。ステップ４０５で、ユーザスペーストランスポートは、パラメータの妥当性を検証すると、一時記憶域をコールバック情報記述子とパラメータとに割り当てる。一実施形態では、トランスポートが、対応するＡＰＩ署名の定義に従い、パラメータの指定サイズに基づいて記憶装置を割り当てる。

【００４３】

ステップ４１０で、ユーザスペーストランスポートは、一時記憶域を割り当てた後、基底のキャラクタ装置（underlying character device）でシステム呼び出しを実行する。システム呼び出しの種類は、コールバック呼び出しが非同期型か同期型かに依存し得る。一実施形態では、非同期型のコールバック呼び出しの場合、ユーザスペーストランスポートは、キャラクタ装置で書き込みコマンドを実行し得る。この場合、ユーザスペーストランスポートは、書き込みコマンドを実行する前に、コールバック・記述子・ヘッダを作成する。一実施形態では、このヘッダが、呼び出されるコールバック識別子とパラメータセマンティクス（例えば、ヘッダに続くパラメータカウントおよびサイズ）とを符号化する。同期型コールバック呼び出しの場合、ユーザスペーストランスポートは、キャラクタ装置でｉｏｃｔｌコマンドを実行し得る（この場合、コールバック記述子はｉｏｃｔｌリクエストで符号化される）。これにより、カーネルがコールバックをディスパッチし、実行すると、ユーザスペーストランスポートが出力パラメータおよび入力／出力パラメータをカーネルからコピーできるようになる。ステップ４１５で、カーネルスペーストランスポートは、基底のキャラクタ装置がプロセスコールバック記述子を受信し、処理するためのシステム・コール・ハンドラを実装する。ステップ４２０で、カーネルスペーストランスポートは、データをアンパックし、コールバックを呼び出す。

【００４４】

先述のとおり、本明細書に開示された実施形態は、カーネルスペースまたはユーザスペースで実行時に実行される所望の管理処理をサードパーティの開発者が定義できるようにするインフラストラクチャを提供する。通信メカニズムは、カーネルスペースおよびユーザスペースに存在するトランスポートコンポーネントを実行時に接続し、処理を実行するための高次のコールバックスタイルセマンティクスを提供する。この手法は、関数コールのように関数を呼び出すという見た目および抽象化（appearance and abstraction of invoking）を提供する。さらには、インターフェイス定義の徹底により、関数コールの抽象

化のどちら側でも、コンポーネントソフトウェア（例えば、アプリケーションおよびカーネルモジュール）がビットまたはデータ型を操作する必要性が低減される。加えて、インフラストラクチャのバージョニング機能により、互換性確保のためにユーザスペースアプリケーションとカーネルスペースアプリケーションとを密に結合する必要性が低減される。

【００４５】

概して言えば、本明細書に記載された各種実施形態では、コンピュータシステムに記憶されたデータを伴う各種コンピュータ実装された処理が用いられ得る。例えば、これらの処理は、必ずというわけではないものの、通常は物理的な量の物理的操作を必要とし得る。その量は、電気信号または磁気信号という形を取り得る。これらの信号あるいは信号表示は、記憶、移送、結合、比較、あるいは別の形での操作が可能である。さらに、かかる操作は、作成、識別、判断、比較などの言葉で言及されることが多い。本明細書に記載された１つ以上の実施形態の一部を成す処理は、有用な機械処理であり得る。加えて、１つ以上の実施形態は、これらの処理を実行するための装置または機器に関するものである。その機器は、特定の必須目的で特別に作られたものであり得るか、あるいはコンピュータに記憶されたコンピュータプログラムによって選択的に起動または構成された汎用コンピュータであり得る。特に、本明細書の教示に従って記述されたコンピュータプログラムと共に各種汎用機械が使用され得るか、あるいは必須の処理を実行するために、より専門的な機器を作成した方が便利であり得る。

【００４６】

本明細書に記載された各種実施形態は、ハンドヘルド装置、マイクロプロセッサシステム、マイクロプロセッサベースまたはプログラミング可能な家庭用電化製品、ミニコンピュータ、メインフレームコンピュータなど、他のコンピュータシステム構成で実施され得る。

【００４７】

１つ以上の実施形態が、１つ以上のコンピュータプログラムとして、あるいは１つ以上のコンピュータ可読媒体に組み込まれた１つ以上のコンピュータ・プログラム・モジュールとして実装され得る。コンピュータ可読媒体という用語は、後でコンピュータシステムに入力可能なデータを記憶できる任意のデータ記憶装置を表し、コンピュータ可読媒体は、コンピュータによって読み出せるようにコンピュータプログラムを具現化するための任意の既存技術あるいはその後開発された技術に基づき得る。コンピュータ可読媒体の例としては、ハードドライブ、ネットワーク接続型記憶装置（NAS : network attached storage）、読み出し専用メモリ、ランダムアクセスメモリ（例えばフラッシュメモリ装置）、ＣＤ（コンパクトディスク）、ＣＤ－ＲＯＭ、ＣＤ－Ｒ、またはＣＤ－ＲＷ、ＤＶＤ（デジタル多用途ディスク）、磁気テープ、ならびに他の光学および非光学データ記憶装置などがある。コンピュータ可読媒体は、コンピュータ可読コードが分散して記憶および実行されるように、ネットワークに連結されたコンピュータシステムを介して配信することもできる。

【００４８】

１つ以上の実施形態について、明確に理解してもらえるよう、ある程度詳しく説明してきたが、請求項の範囲内で一定の変更および改変が可能であることは明らかであろう。したがって、記載された実施形態は例示的なものとみなすべきであり、制限的なものとみなすべきでない。そして、請求項の範囲は本明細書に記載された詳細に制限されるべきでなく、請求項の範囲および均等物内で修正され得る。請求項では、明示的に述べられていない限り、要素および／またはステップが任意の特定の処理順を示唆することはない。

【００４９】

加えて、上記仮想化方法では、仮想マシンが特定のハードウェアシステムに整合するインターフェイスを提示すると概して想定してきたが、上記方法は、任意の特定ハードウェアシステムに直接対応しない仮想化と併用され得る。ホスト側の実施形態として、非ホスト側の実施形態として、あるいはその両者の区別を曖昧にする傾向がある実施形態として

実装された各種実施形態にかかるすべての仮想化システムが想定される。さらに、各種仮想化処理は、全体または一部がハードウェアで実装され得る。例えば、あるハードウェア実装形態では、非ディスクデータのセキュリティを確保するための記憶装置アクセスリクエストを修正するのにルックアップテーブルが使用され得る。

【００５０】

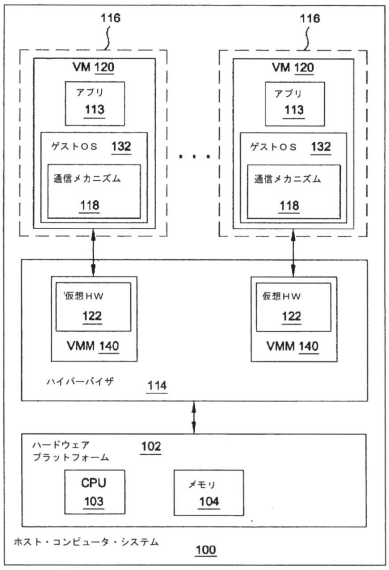
仮想化の度合いに関係なく、多くの変形例、修正例、追加例、および改善例が存在し得る。そのため、仮想化ソフトウェアは、仮想化機能を実行するホスト、コンソール、またはゲスト・オペレーティング・システムのコンポーネントを含むことができる。本明細書に記載されたコンポーネント、処理、または構造について複数の事例が提供され得る。最後に、各種コンポーネント、処理、およびデータストア間の境界はやや恣意的なものであり、特定の処理は具体的な構成例という文脈の中で例示されるものである。他の機能割り当ても想定され、１つ以上の実施形態の範囲内に属し得る。概して、構成例の中で別々のコンポーネントとして表された構造および機能が組み合わさった構造またはコンポーネントとして実装されても良く、同様に、単一のコンポーネントとして表された構造および機能が別々のコンポーネントとして実装されても良い。これらの、そして他の変形例、修正例、追加例、および改善例は、添付の請求項の範囲内に属し得る。

【符号の説明】

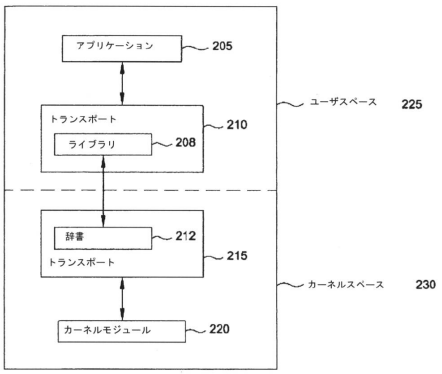
【００５１】

１００　ホスト・コンピュータ・システム
２０５　アプリケーション
２１０　トランスポート
２０８　ライブラリ
２２０　カーネルモジュール
２２５　ユーザスペース
２３０　カーネルスペース

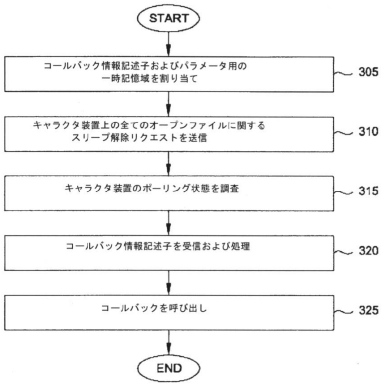
【図１】



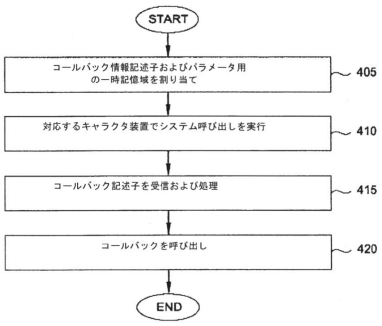
【図２】



【図３】



【図４】


START
コールバック情報記述子およびパラメータ用の一時記憶域を割り当て
405
対応するキャラクタ装置でシステム呼び出しを実行
410
コールバック記述子を受信および処理
415
コールバックを呼び出し
420
END

フロントページの続き

(72)発明者　ポール　ウィルマン
アメリカ合衆国　０２１８９　マサチューセッツ州　ウェーマス　ショウマット　ストリート　７３

審査官　田中　幸雄

(56)参考文献　特開２００４－９８６５８（ＪＰ，Ａ）
米国特許出願公開第２００４／０１４８６０９（ＵＳ，Ａ１）
米国特許第８２０９７０４（ＵＳ，Ｂ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／５４