(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4006337号
(P4006337)
(45)発行日　平成19年11月14日(2007.11.14)　(24)登録日　平成19年8月31日(2007.8.31)

| (51)Int.Cl. | | F I | | |
|---|---|---|---|---|
| G06F 9/54 | (2006.01) | G06F | 9/06 | 640B |
| G06F 11/00 | (2006.01) | G06F | 9/06 | 630K |
| G06F 9/45 | (2006.01) | G06F | 9/44 | 322K |

請求項の数 4　(全 13 頁)

| | |
|---|---|
| (21)出願番号 | 特願2002-561683 (P2002-561683) |
| (86)(22)出願日 | 平成14年2月1日(2002.2.1) |
| (65)公表番号 | 特表2004-530184 (P2004-530184A) |
| (43)公表日 | 平成16年9月30日(2004.9.30) |
| (86)国際出願番号 | PCT/FR2002/000398 |
| (87)国際公開番号 | WO2002/061579 |
| (87)国際公開日 | 平成14年8月8日(2002.8.8) |
| 審査請求日 | 平成17年1月28日(2005.1.28) |
| (31)優先権主張番号 | 01/01378 |
| (32)優先日 | 平成13年2月1日(2001.2.1) |
| (33)優先権主張国 | フランス(FR) |

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　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】共有ライブラリを使用して実行可能ファイルの実行を修正する方法

(57)【特許請求の範囲】

【請求項１】

コンピュータ内の共有ライブラリ中およびサービスファイル中に位置する関数を参照する複数の記号を含む実行可能ファイルの実行を修正する方法であって、

前記実行可能ファイルを起動する際、前記複数の記号のうち少なくとも１つの記号と前記複数の関数のうち少なくとも１つの関数に対する参照との関連づけが定義された少なくとも１つのエントリから構成されている参照ファイルをロードするステップと、

動的リンク編集段階においては、前記複数の記号のうち少なくとも１つの記号が前記参照ファイル中で定義されている場合は、当該記号と、当該記号を含む少なくとも１つのエントリとをリンク編集し、前記複数の記号のうち少なくとも１つの記号が前記参照ファイル中で定義されていない場合は、当該記号と、当該記号に対応する前記共有ライブラリ中の少なくとも１つの関数とをリンク編集するステップと、

前記実行可能ファイルの実行中においては、前記複数の記号のうち少なくとも１つの記号により関数呼び出しがあると、当該記号が前記参照ファイル中で定義されている場合は、当該記号とリンク編集されている少なくとも１つのエントリに含まれ、当該記号に関連づけられている少なくとも１つの参照が指し示す、前記サービスファイル中の少なくとも１つの関数を実行し、当該記号が前記参照ファイル中で定義されていない場合は、当該記号とリンク編集された前記共有ライブラリ中の少なくとも１つの関数を実行するステップとを前記コンピュータが自動的に実行することを特徴とする方法。

【請求項２】

前記実行可能ファイルの実行中においては、前記参照ファイル中で定義されている各記号と当該記号に関連づけられている前記少なくとも１つの参照が指し示す、前記サービスファイル中の前記少なくとも１つの関数との間のリンクが動的に編集されることを特徴とする、請求項１に記載の方法。

【請求項３】

前記動的リンク編集が共有ライブラリ操作ルーチンを使用して実行されることを特徴とする、請求項２に記載の方法。

【請求項４】

前記参照ファイルと前記サービスファイルとが共有ライブラリの形態で管理されることを特徴とする、請求項１乃至３の何れか１項に記載の方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、共有ライブラリを使用して実行可能ファイルの実行を修正する方法に関する。

【背景技術】

【０００２】

実行可能ファイルが大容量になるのを避けるため、実行可能ファイルに共通の関数は全て、一般に別個のファイルにプールされる。この別個のファイルすなわち共有ライブラリは普通オペレーティングシステム中に統合される。例えば、関数を定義するオブジェクトコードはもはや実行可能ファイル中に統合されず、その代わりこの関数への参照だけが実行可能ファイル中に記述される。

【発明の開示】

【発明が解決しようとする課題】

【０００３】

従って、少なくとも１つの共有ライブラリを使用する実行可能ファイル、すなわち動的実行可能ファイルは、共有ライブラリから共通して使用される関数を検索する。実行可能ファイルの小さな部分であっても、修正が望ましい場合には、実行可能ファイルのソースコード全体を修正する必要があり、その後でこのソースコードに対するコンパイル段階を再実行しなければならない。しかし、実行可能ファイルのソースコードは必ずしも利用可能ではない。

【０００４】

本発明の目的は、動的かつ侵入を伴わずに、すなわち、実行可能ファイルおよびそれが共有しうるライブラリに何ら物理的な修正を行うことなく、実行可能ファイルの挙動を修正できるようにすることである。

【０００５】

本発明の別の目的は、既存の実行可能ファイルの実行中にその容量を、動的かつ侵入を伴わずに拡張することである。

【課題を解決するための手段】

【０００６】

上記の目的は、コンピュータ内の少なくとも１つの共有ライブラリ中に位置する関数（Ｆｕｎｃｔｉｏｎ）への参照を含む実行可能ファイルを侵入を伴わずに修正する方法によって達成される。本発明の目的のため、共有ライブラリの所定の関数を呼び出す、実行可能ファイル中の参照ファイルについて、参照ファイルがアクセスされて実行可能ファイルと前記所定の関数が実際に位置するロケーションとの間のリンクとして動作し、この所定の関数のアドレスがこの参照ファイルに含まれ、このアドレスが他の関数を指す新しいアドレスによって置換されるので、前記関数を呼び出すと、実行可能ファイルの実行中にこの他の関数が実行できるようになる。

【０００７】

本発明は、関数のアドレスを修正できるようにそうしたアドレスを含む参照ファイルに

作用することによって機能し、それによって、実行可能ファイルを変更せずに実行可能ファイルに新しいオブジェクトコードを提供する。

【０００８】

本発明を実施する第１の形態では、

－実行可能ファイルを起動する際、サービスファイル中に位置する新しい関数への複数の参照を含む、参照ファイルが事前ロードされ、

－実行可能ファイルの起動中に行われる動的リンク編集段階で、実行可能ファイル中に含まれる各未解決の参照について、前記参照が、この参照が参照ファイル中で定義される限り優先順位の問題として前記参照ファイルに割り当てられ、さもなければ前記参照が共有ライブラリに割り当てられ、

－実行可能ファイルの実行中、参照ファイル中に参照が存在する関数を呼び出すことで、実際に呼び出された関数を実行する代わりに、サービスファイル中に位置する新しい関数を実行することができ、この新しい関数へのアクセス経路が参照ファイル中に定義される。

【０００９】

参照先の関数のオブジェクトコードが実行可能ファイル中に含まれていない時、参照は未解決であると考えられる。

【００１０】

サービスファイルは、実行可能ファイルにアタッチすべき新しいサービスまたは関数を含むファイルである。参照ファイルは実行可能ファイル中に存在する参照の一部を含む。通常、こうした参照の目的は共有ライブラリ中に含まれる関数を実行することである。しかし、本発明が提案する方法によれば、実行可能ファイルの各参照と共有ライブラリ中の対応する関数との間のリンクを確立する前に、まずその関数が参照ファイル中に記述されていることを確認する検査が実行される。その場合、参照ファイルはいわばこの対応する関数を遮断し、この参照と参照ファイルとの間のリンクが確立される。そうでない場合、すなわち、関数が参照ファイルに記述されていない場合、従来の方法で実行可能ファイルと共有ライブラリ中の対応する関数との間に動的リンクが確立される。

【００１１】

関数が遮断される時はいつでも、参照ファイルはこの関数への参照を含み、１つかそれ以上の新しい関数がこの参照に割り当てられる。別言すれば、参照ファイル中の各参照は新しい関数へのアクセス経路に関連付けられる。この新しい関数はサービスファイル中に含まれる関数（またはサービス）でもよい。さらに、参照ファイル中で定義されるアクセス経路によって共有ライブラリからの関数を実行できるようにしてもよい。すなわち、関数の遮断という事実によって、この元の関数は、新しい関数か、または共有ライブラリ中に格納される元の関数を含むことも含まないこともある関数の組み合わせかによって置換される。

【００１２】

１つの特定実施形態では、参照される関数が参照ファイルから最初に呼び出される時、新しい段階が実行され、参照ファイル中に含まれる参照とその参照にリンクされる関数との間のリンクが動的に編集される。動的リンクを編集するこの新しい段階は共有ライブラリ操作ルーチンを使用して実行してもよい。

【００１３】

従って、参照ファイルとサービスファイルとは、共有ライブラリの形態で管理する事が好ましい。

【００１４】

本発明を実施する１つの形態では、実行可能ファイルの実行中、参照ファイル中のアクセス経路が制御チャネルによって制御され修正される。

【００１５】

サービスファイル中に含まれる関数を制御するため、制御チャネルを使用してもよい。

【００１６】

次に、実行可能ファイルは再生成の必要なしに変更できる。実行中にサービスを挿入、削除または修正してもよい。

【００１７】

参照ファイルはコンピュータのオペレーティングシステムの動的リンク編集関数を使用して実行可能ファイルを修正しサービスを起動または停止する。

【００１８】

既存の技術では、動的リンク編集処理は普通、実行可能ファイルがロードされると、それが起動される前に排他的に実行される。本発明が提案する方法によれば、この処理は実行可能ファイルの実行中いつでもアクセスできる。

【００１９】

同じ関数がいくつかのサービスによって変更でき、そうしたサービスに対する実行指令はこの参照ファイルによって定義されるため、参照ファイルは、いわば再使用可能である。

【００２０】

本発明の別の態様によれば、コンピュータ内の少なくとも１つの共有ライブラリ中に位置する関数への参照を含む実行可能ファイルを侵入を伴わずに修正できるシステムが提案される。有利にも、本システムは、

－この実行可能ファイルの実行段階の間に、実行可能ファイルに動的に挿入されることを目的とする複数の関数を含むサービスファイルと、

－前記サービスファイル中および共有ライブラリ中に位置する関数への複数の参照を含む参照ファイルと、

－サービスファイルと参照ファイルとを監視および制御する制御手段とを備える。

【００２１】

本発明を実施する第２の形態では、共有ライブラリ中の関数の存在をシミュレートできるローカル疑似関数を有し、実際にはこの疑似関数が再ルーティングテーブル中に位置するアドレスを指す共有ライブラリを管理することを目的とする。この方法のいくつかのステップは管理モジュールによって実現される。このため、参照ファイルは再ルーティングテーブルであり、管理モジュールがロードされると、

－動的オブジェクトとそれに対応するローディングアドレスのリストと、各動的オブジェクトについて、インポート外部関数のリストとエクスポートされた関数のリストとが実行可能ファイル内で決定され、

－実行可能ファイルの実行中に、再ルーティングテーブル中の所定の関数のアドレスが修正されるので、前記所定の関数が次に呼び出される時、新しいアドレスに対応する関数が実行できるようになる。

【００２２】

この第２の形態によって先在するアドレステーブルによる動作が可能になる。実際には、エクスポートされた関数は再ルーティングテーブルを有し、先行技術ではこうしたテーブルは静的である。再ルーティングテーブルを編集すると、呼び出しグラフが修正される。

【００２３】

再ルーティングテーブルに対してなされる修正は全て有利にはメモリ中に格納できるので、編集処理は可逆的となる。

【００２４】

管理モジュールは指令言語を有し、再ルーティングテーブルは動的に編集できることが好ましい。

【００２５】

管理モジュールを実行可能ファイルにロードするため、管理モジュールは、実行可能ファイルが起動される時共有ライブラリの形態で事前ロードできる。また、実行可能ファイルは実行中に監視モジュールによって割り込みしてもよく、この監視モジュールは実行可能ファイルの外部にあり管理モジュールを制御するようプログラムされている。割り込み

は最新の命令で実行可能ファイル中に挿入される。管理モジュールは共有ライブラリと同じ方法で処理される。

【００２６】

本発明の１つの有利な特徴の結果、再ルーティングテーブルを編集する指令は、管理モジュールを呼び出す手段を有する監視モジュールによって管理モジュールに伝送される。

【００２７】

この第２の形態によって、新しい関数を事前定義する必要はない。新しい関数は事前定義によってサービスファイル中に配置してもよく、また実行可能ファイルの実行中に作成してもよい。

【００２８】

本発明の目的のため、監視モジュール中に含まれる解放手段の助けによって、管理モジュールを実行中に実行可能ファイルから除去してもよい。

【００２９】

第３の実施形態では、コンピュータのオペレーティングシステムは、実行中エクスポートされた関数を含む新しい共有ライブラリを即座にロードし、こうしたエクスポートされた関数のアドレスが実行可能ファイルの任意のローカル変数中に格納され、参照ファイルがそのローカル変数と、実行可能ファイルによって呼び出される見込みのエクスポートされた関数との間に挿入され、参照ファイルは標準設定でこうしたエクスポートされた関数のアドレスを含む。参照ファイルは、管理モジュールによって修正できるように設定される。すなわち、この参照ファイルはローカル変数ではなく管理モジュールによってアクセスできるような位置に配置される。

【００３０】

本発明の他の利点と特徴は、実施形態の詳細な説明（ただしこれはいかなる意味でも制限的であることを意図したものではない）と添付の図面から明らかになるだろう。

【発明を実施するための最良の形態】

【００３１】

図１は、先行技術に基づいて共有ライブラリＢを使用する動的実行可能ファイルＥをどのように編成するかを例示する。実行可能ファイルＥはオブジェクトコードの形態のプログラムであり、そこでは記号Ｆ１、Ｆ２およびＦ３が指定される。これらの記号は共通して使用される関数Ｆ１、Ｆ２およびＦ３への参照であり、それらの関数のオブジェクトコードｃｏＦ１、ｃｏＦ２およびｃｏＦ３は共有ライブラリＢ中に格納されている。このライブラリは別個のファイルでありプログラムＥとは連結されない。プログラムＥ中にコード化された初期命令が実行される前に、プログラムＥが起動プログラム（またはブートローダ）によってメモリ中にロードされる時はいつでも、記号への参照を解決する段階が実行される。この参照解決段階は、プログラムＥの各参照Ｆ１を、ライブラリＢ中に格納されたそのオブジェクトコードにリンクすることを目的とするリンク編集段階の一部を形成する。

【００３２】

図２は、図１に示される要素を組み込んだ、本発明が提案するシステムを例示する。図３は、本発明が提案する方法の種々のステップを説明する。

【００３３】

ここで、図２および図３を参照して本発明の実施形態を説明する。

【００３４】

図２に示すように、図１に示されていない２つの新しいファイルが存在する。すなわち、参照Ｆ２およびＦ３を含む特定の共有ライブラリであるＭＥＴＡ（メタ）参照ファイルが存在する。また、新しい関数またはサービスＸ１、Ｘ２およびＸ３のオブジェクトコードｃｏＸ１、ｃｏＸ２およびｃｏＸ３を含む特定の共有ライブラリであるサービスファイルＳが存在する。その目的は、実行可能ファイルＥとライブラリＢとを物理的に修正しないのと同時にいくつかの新しい関数Ｘｉを統合することによって、プログラムＥが実行される方法を修正することである。

【００３５】
ＭＥＴＡファイルでは、各参照は次式のように割り当てられる。
Ｆ２＝Ｓ：Ｘ２，Ｂ：Ｆ２
および
Ｆ３＝Ｓ：Ｘ３
【００３６】
Ｆ２に関連する数式が意味するのは、関数Ｆ２を実行する代わりに、ファイルＳ中に格納された関数Ｘ２を実行し、その後共有ライブラリＢ中に格納された関数Ｆ２を実行するというものである。
【００３７】
Ｆ３に関連する数式が意味するのは、関数Ｆ３の実行は、ファイルＳ中に格納された関数Ｘ３の実行によって置換されるということである。
【００３８】
ステップ１でプログラムＥを起動する際、内部で本発明が運用されるオペレーティングシステムは、Ｅ中でコード化された第１の命令を実行する前にステップ２でＭＥＴＡファイルを事前ロードするよう命令される。例えば、リナックス（ＬＩＮＵＸ）システムでは、これは環境変数“ＬＤ＿ＰＲＥＬＯＡＤ”を配置することによって得られる。
【００３９】
次に、起動プログラム（ブートローダ）によって実行される動的リンク編集段階がステップ３で開始される。ステップ４は、ファイルＥがオブジェクトコードを含まない、すなわち、ファイルＥがオブジェクトコードの参照または記号だけを含み、その場合記号はいわば解決されないような全ての関数を示す。図２では、こうした参照はＦ１、Ｆ２およびＦ３である。優先順位の問題として、それらはＭＥＴＡファイルに割り当てられ、ステップ５では、それらがＭＥＴＡファイル中に記述されているかを確認するチェックが最初に実行される。参照Ｆ２およびＦ３だけがＭＥＴＡファイル中に記述されている。この場合、ステップ７で実行可能ファイルＥの参照Ｆ２およびＦ３と、ＭＥＴＡ参照ファイルの参照Ｆ２およびＦ３との間のリンクが編集される。他方、関数Ｆ１の場合のように、関数がＭＥＴＡファイル中に記述されていない場合、ステップ６で実行可能ファイルＥの参照Ｆ１と関数Ｆ１のオブジェクトコードｃｏＦ１との間のリンクが編集され、このオブジェクトコードｃｏＦ１は共有ライブラリＢ中に格納される。
【００４０】
ステップ８でリンク編集段階が終了すると、ステップ９でファイルＥの実際の実行が開始される。Ｅが実行されると、始めにＭＥＴＡファイルの初期化が起動され、遮断された関数の１つ、この特定の事例ではＦ２がステップ１０で呼び出される。この初期化はステップ３とステップ８との間で行われるリンク編集段階を完了することにある。この目的で共有ライブラリを操作するために使用されるルーチンは、始動プログラムによって使用されるものと同じである。２つの関数Ｆ２およびＦ３について、以下が解決される。
【００４１】
－サービスが挿入されることを示すＭＥＴＡ中の各関数の参照。図２では、ＭＥＴＡファイル中のＦ２はサービスファイルＳ中のＸ２を指さなければならず、ＭＥＴＡファイル中のＦ３はサービスファイルＳ中のＸ３を指さなければならず、挿入されるサービスがない場合、関数は元のライブラリＢにリダイレクトされる。
【００４２】
－必要な場合、サービスファイルＳ中の、前提関数を示す参照（単数または複数）。例えば、図２では、Ｓ中の関数ｘ２はＢ中の関数Ｆ２を逆参照する。この動作によってサービスのリンク連鎖が確立され、インクリメントサービスを開発できるようになるので、これは非常に重要である。
【００４３】
この初期化段階はステップ１１およびステップ１４で実行される。このＭＥＴＡの初期化に続いて、関数間の呼び出しグラフが確立され、その後ステップ１２で関数Ｘ２が実行

され、それに続いてステップ１３で関数Ｆ２が実行される。

【００４４】

共有ライブラリの形態で挿入される関数呼び出しグラフ可能化サービスを再定義する能力はプログラムＥが起動される時利用可能であるが、このサービスはプログラムＥの実行中もいつでも、特にＭＥＴＡファイル中の制御チャネルによって、起動できる。関数をＭＥＴＡ中で遮断する時、信号による誘発、等といったいくつかの戦略を使用してこのチャネルを開始すればよい。

【００４５】

一例として、ＭＥＴＡファイルを使用して次のように呼び出しシステムを指定すればよい。

”

```
extern int open(const char *pathname, int
flags, mode_t mode)
{
if(traceon&&tracefile)
  fprintf(tracefile, ”trace%s:%d:open(\”%
s\”, %o, %o)\n”,
              INDEX, pid, pathname, flags, mode)
;
Return(*meta_open)(pathname, flags, mode)
);}
```

”

【００４６】

上記の例では、下線で示した行は後続の“ｏｐｅｎ”関数の呼び出しである。変数“ｍｅｔａ＿ｏｐｅｎ”は、同一の方法で別のサービスを実現する元の関数またはルーチンへのポインタである。この関数、例えばＳ中で定義されるＸ２は、複数の実行可能ファイルについてそれらを修正する必要なしにＭＥＴＡによってロードしてもよい。

【００４７】

ＭＥＴＡファイルは編集されるプログラムおよび実行されるサービスと無関係である。各サービスは、共有ライブラリの形態で、ＭＥＴＡファイルに以下のデータを供給する。

【００４８】

－呼び出される内部関数へのリンクを伴う、ＭＥＴＡファイルが遮断しなければならない関数のリスト。例えば図２では、Ｅ中の関数Ｆ２の呼び出しはＳ中の関数Ｘ２を起動しなければならない。

【００４９】

－サービスを実現するための（すなわち遮断されうる）前提外部関数のリスト。この情報によってＭＥＴＡファイルはサービスと関数との間のリンクを設定できるようになる。

【００５０】

－様々な新しい関数と共に、遮断の際に起動される各代替関数の定義。

【００５１】

－後続のサービスが使用しうる、サービスによってエクスポートされる新しい関数のリスト。例えば、ＭＥＴＡは制御チャネルを開く関数をエクスポートするので、他のサービスはそれを再実装する必要なしに使用できる。

【００５２】

本発明は、アプリケーションによる耐障害性（フォールトトレランス）、アプリケーションの移行、および入出力の外部制御といった一般的な問題に対処する簡単な手段を提供する代替方法を提案する。これによって実行可能ファイルは侵入を伴わずに変異させられるようになる。例えば、本発明が提案するシステム中の実行可能ファイルにおいて、変更可能な関数は、呼び出しシステムまたは他の種類の関数（数学的、等々）である、動的ロード可能な共有ライブラリ中で定義されるものである。

【００５３】
本発明が提案する方法はオペレーティングシステムの動的リンク編集関数を使用するので、実行可能サービスを侵入を伴わずに操作し、それ自体共有ライブラリの形態であるサービスを起動／停止することができる。動的リンク編集関数は有利には、実行可能ファイルの実行中いつでもアクセスできる。最後に、ＭＥＴＡ参照ファイルは、同じ関数をいくつかのサービスが変更できるため再使用可能である。また、ＭＥＴＡ型のいくつかの参照ファイルといくつかのサービスファイルＳとを設定することも考えられる。
【００５４】
実行可能ファイルＥは図４にも見られる。ライブラリＢ２は、関数Ｆ３に対するオブジェクトコードを含まず、その代わり、実行可能ファイルＥに応答して関数Ｆ３の存在のシミュレートを可能にする、「スタブ」と呼ばれるアプリケーションまたは疑似関数を含むように設定される。実際には、「スタブ」アプリケーションは再ルーティングテーブルＴＲ中に位置するアドレスＴＲ（２）を指し、一方それは、関数Ｆ３に対するオブジェクトコードｃｏＦ３が実際に位置するロケーションのアドレスＭ（２）を指す。注目すべき点は、実行可能ファイルの外部にある任意のモジュールによって再ルーティングテーブルにアクセスできるということである。本発明は管理モジュールＭＧを使用するが、これは再ルーティングテーブル中に含まれるアドレスを修正する能力を有する知的エージェントまたはアプリケーションである。図５に例示される管理モジュールＭＧによって、再ルーティングテーブル中のアドレスＭ（２）はアドレスＭ（３）によって置換される。一方、Ｍ（３）は新しい関数Ｘ４に対するオブジェクトコードｃｏＸ４を含む。従って、実行可能ファイルが関数Ｆ３を呼び出す時、処理されるのは関数Ｘ４のオブジェクトコードｃｏＸ４である。管理モジュールはこの置換を記憶する手段を有するので、初期状態、すなわち、再ルーティングテーブルＴＲ中のＭ（２）の回復が可能である。
【００５５】
管理モジュールは監視モジュールと共に動作するが、監視モジュールは、ユーザからの指令を受信して管理モジュールに理解可能な言語で解釈することが可能であり、その後でそうした指令が起動される。
【００５６】
別言すれば、本発明を実施する第２の形態は、その中で関数が呼び出され、その定義が外部共有ライブラリ中に格納される実行可能ファイルまたはプログラムを包含する。動的オブジェクト（プログラムまたは共有ライブラリ）の外部にあり、動的オブジェクトによって明示的に使用される全ての関数について、リンク編集プログラム（リナックスの場合ＬＤ）によって実行される事前リンク編集段階の間に生成されるオブジェクトに対してローカルな擬似関数（「スタブ」と呼ばれる）が存在する。共有ライブラリに関する限り、「スタブ」の目的は、関数のオブジェクトコードが実際にはライブラリに対してローカルでないという事実を隠すことである。さらに、このオブジェクトコード共有ライブラリが生成された時点で未知のアドレスに位置してもよい。プログラムが実行される時、この疑似関数は、このオブジェクトコードへのアクセス経路を確立することによって真のオブジェクトコードを検索する。このアクセス経路は、動的リンクを編集する時に、動的オブジェクトのＤＡＴＡ（データ）セグメント中に位置する、再ルーティングテーブルと呼ばれるゾーン中にオブジェクトコードのアドレスを格納することによって確立される。有利には、このＤＡＴＡセグメントは修正できる。
【００５７】
本方法の第２の実施形態は、動的リンク編集の後の段階において、プログラムの実行中に任意の方法で、共有ライブラリの再ルーティングテーブル中の外部関数のアドレスを編集できるということにある。このため、ある関数の定義を別のものによって置換する。本方法の興味深い特徴は、プログラムが実行中に割り込みされた場合でも、関数への参照が可能になるということである。実際には、関数自体は修正されず、修正されるのはそのロケーションアドレスだけなので、直前に発生した関数はその動作を正常に終了し、次回その関数が呼び出されるまで修正は有効にならない。

【００５８】
本発明が提案する管理モジュールは、特に、以下の関数を含む。
【００５９】
－内省関数、これはプログラム中にロードされると以下のことを決定できる。
・このプログラムの動的オブジェクトのリスト（一部実行可能および共有ライブラリ）およびそれらがロードされたそれぞれのアドレス。
・各動的オブジェクトに対する、インポート外部関数、すなわち、それらに対するエントリが再ルーティングテーブル中に存在し、その後修正されると思われるもののリスト。
・各動的オブジェクトに対する、エクスポートされた関数、すなわちその定義アドレスが同じプログラムの別の動的オブジェクトの再ルーティングテーブル中に出現しうるもののリスト。
【００６０】
－再ルーティングテーブルのための編集関数、これは、第１に、ライブラリの外部の関数のアドレスを修正し、第２に、必要な場合更新を削除し、システムの完全な可逆性を確保するため、連続的に更新を記憶することができる。
【００６１】
－モジュールロード関数、これは、以下の方法に基づいて、監視モジュールからのロードと起動を可能にする制御チャネルを挿入することにある。
・共有ライブラリの形態で、管理モジュール自体を事前ロードすることによってプログラムをブートするか、
・または、プログラムの動作中に、プログラムに割り込み、オペレーティングシステムを介してプログラム中に最新の命令で割り込みを挿入し、管理モジュールをロードする。
【００６２】
－モジュール解放関数、これは動作中にプログラムからモジュールを除去できるようにする。
【００６３】
また、動的プログラムのための外部関数を呼び出す別の可能性が存在する。すなわち、プログラムは、実行中に、新しい共有ライブラリをオンザフライで（実行中に）ロードできるようにする、オペレーティングシステムによって供給される関数（リナックスの場合“ＤＬＯＰＥＮ”）にアクセスし、その後エクスポートされた関数のアドレスを得ることによってこのライブラリを通じてエクスポートされた関数にアクセスすることができ（リナックスの場合“ＤＬＳＹＭ”関数）、それらは、上記で説明されたように再ルーティングテーブル中に格納するのではなく、任意のローカル変数中に直接格納する。先験的に、こうしたローカル変数は外部モジュールにとって未知でありアクセス不能である。
【００６４】
図６は、共有ライブラリＢ３を示すが、これはオペレーティングシステムによってロードされるライブラリである。先行技術では、エクスポートされた関数のアドレスは普通ローカル変数ＶＬ中に位置し、そのローカル変数は利用できない。
【００６５】
本発明の場合、オペレーティングシステムのアプリケーション（リナックスの場合“ＤＬＳＹＭ”関数）を例に取ると、このアプリケーションは普通、共有ライブラリＢ３からのエクスポートされた関数Ｆ３に対するオブジェクトコードのアドレスＭ（２）へのアクセスを可能にする。次に、このアプリケーションは、関数Ｆ３に対するオブジェクトコードのアドレスＭ（２）を返すのではなく、管理テーブルＭＧによって生成される補間テーブルのアドレスを返すように編集される。この補間テーブル（Ｆ３に対応するアドレスに挿入される）は標準設定で、Ｍ中のエクスポートされた関数Ｆ３のオブジェクトコードｃｏＦ３を指す。その利点は、補間テーブルが管理テーブルＭＧを介して修正できるということである。

【００６６】
本発明が提案する方法は、動作を動的に修正できることが望ましいことがありうる対象アプリケーションに適用される。第２および第３の形態では、本方法は特に、以下の要素を使用して実現される。
【００６７】
－共有ライブラリの形態の管理モジュール。これは関数に対する呼び出しグラフの動的な編集を可能にする指令言語を使用する。例えば、その指令は、「ライブラリ１中で定義される関数１を、ライブラリ２中で定義される関数２によって置換する」、「置換を削除する」、「置換された関数を記述する」、「新しい関数を記述する」等でよい。例えば、置換指令の後、関数１に対する呼び出しは全て、実際には、関数１の代わりに、それに当たる部分で関数２を呼び出すことになる。こうした変更は、プログラムが終了するまで適用され続ける。それは永続的ではないので、プログラムは標準設定で再起動される。
【００６８】
－外部プログラムの形態の監視モジュール。これは以下のことを可能にする、
・割り込みを挿入し、共有ライブラリの形態の管理モジュールのロード関数を呼び出すことによって、実行中に管理モジュールをプログラム中にロードすること。リナックスの場合（“ｄｌｏｐｅｎ（“ｍｇ．ｓｏ”））”。実現する技術はデバッガによって使用されるものと同様である。
・このプログラムの呼び出しグラフを操作するための、制御チャネルを介したプログラム（実行可能ファイル）との対話。指令はグラフィカルインタフェースによってか、指令文字列によってか、または何らかの他のプログラム中に含まれる関数によって起動すればよい。
【００６９】
－呼び出しグラフを編集する能力を別にすれば、外部プログラムはアプリケーションの円滑な実行にとって絶対的に必要ではない。
【００７０】
これらの例は制限的であることを意図したものではないが、本発明は以下のアプリケーション中で実現してもよい。
【００７１】
試験作業の際測定プローブを挿入することによって性能を測定するために使用される情報システムにおける可逆的な動的計測。原理的に、こうした操作は不可逆的な既存の計測技術では不可能であり、操作の費用はこの種の操作配置とは比較にならない。
【００７２】
関数が、バグを訂正したり新しい関数を導入したりする新しいバージョンによって動的に置換される、動作中の更新および訂正によるプログラム保守。現在使用されている技術では、プログラムを完全に停止して置換し、再起動する必要がある。
【００７３】
例えば、障害の際に処理を別のマシンにリダイレクトできるシステム関数を代用することによる、アプリケーションおよびオペレーティングシステム両方についてのシステム関数の侵入を伴わない実現。現在の技術では、システムの中核に介入するか、個々のアプリケーションを書き換えることなしには、こうした変更はできない。
【００７４】
コネクタおよび変換フィルタを開発することによる異質のアプリケーションの侵入を伴わない統合。これはアプリケーションを書き換えずに代用することによって既存のインタフェース上で適切に挿入できる。
【００７５】
明らかに、本発明は上記で説明された例に制限されるものではなく、本発明の範囲から離れることなくこれらの例に任意の数の変更がなされることがある。
【図面の簡単な説明】
【００７６】

【図１】先行技術から周知の、共有ライブラリを使用する実行可能ファイルの物理編成を例示する単純化された図である。
【図２】本発明が提案するようないくつかの共有ライブラリを使用する実行可能ファイルの物理編成を例示する単純化された図である。
【図３】本発明が提案する方法の種々のステップを例示する流れ図である。
【図４】エクスポートされた関数へのリダイレクト手段を伴う、共有ライブラリを使用する実行可能ファイルの物理編成を例示する単純化された図である。
【図５】本発明が提案する第２の実施形態を使用してエクスポートされた関数が置換された状況で図４の要素を例示する。
【図６】実行中に共有ライブラリがコンピュータのオペレーティングシステムによってロードされた状況で本発明を実施する第３のアプローチを例示する単純化された図である。

【図１】

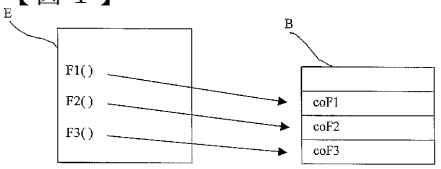



FIGURE 1

【図２】

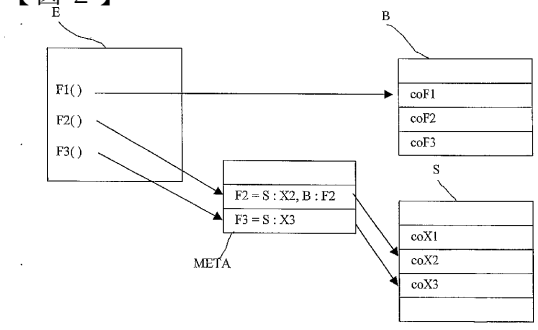
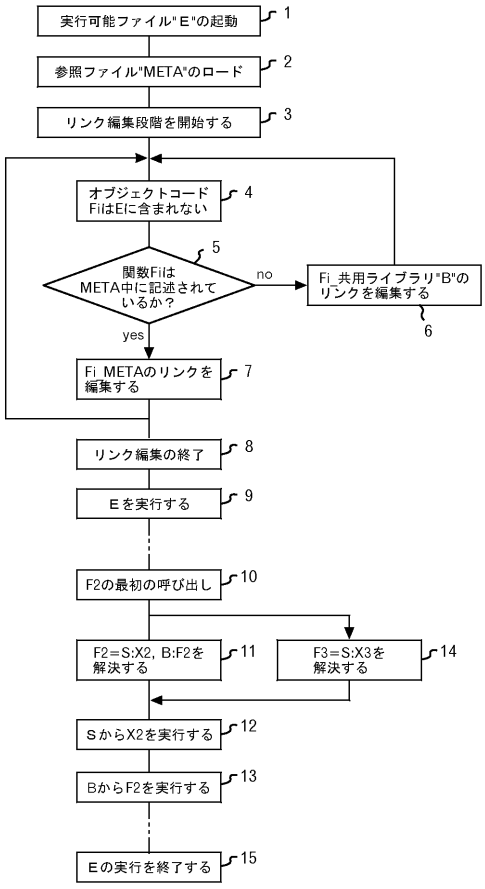



FIGURE 2

【図３】



【図４】

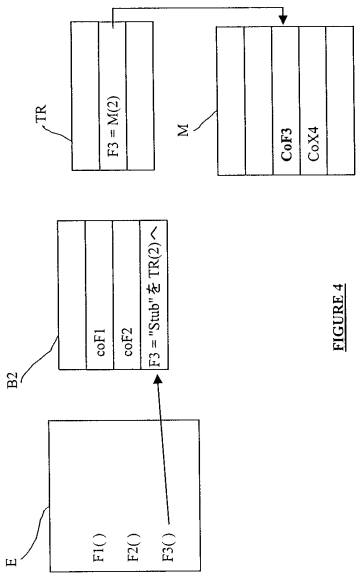



FIGURE 4

【図５】

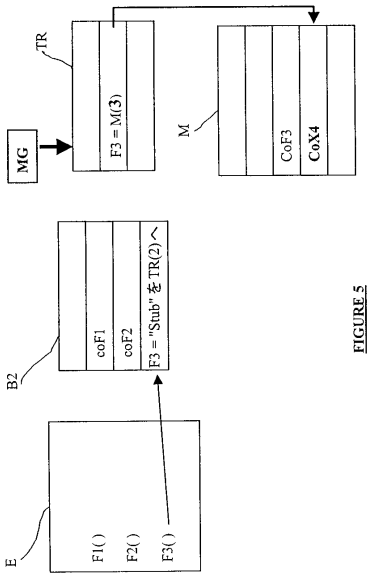



FIGURE 5

【図６】

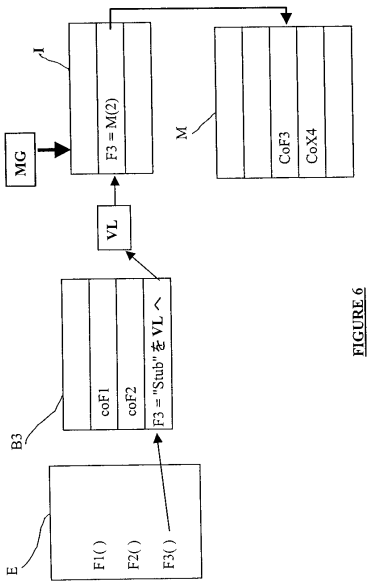



FIGURE 6

フロントページの続き

(74)代理人 100112690
弁理士 太佐 種一
(72)発明者 ベルト，マルク フィリップ
フランス国，エフ－３１０００ トゥールーズ，アレー デュ ニジェール 2，レジダンス ジャコモ，アパルトマン ２０１３

審査官 林 毅

(56)参考文献 特開平０４－２４３４２４（ＪＰ，Ａ）
米国特許第０６１５４８７８（ＵＳ，Ａ）
特開平０１－３０７８２５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 9/54
G06F 9/45
G06F 11/00