



(19)
Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 602 26 019 T2** 2009.06.25

(12)

Übersetzung der europäischen Patentschrift

(97) **EP 1 386 230 B1**

(51) Int Cl.⁸: **G06F 9/445** (2006.01)

(21) Deutsches Aktenzeichen: **602 26 019.1**

(86) PCT-Aktenzeichen: **PCT/FR02/00398**

(96) Europäisches Aktenzeichen: **02 701 384.6**

(87) PCT-Veröffentlichungs-Nr.: **WO 2002/061579**

(86) PCT-Anmeldetag: **01.02.2002**

(87) Veröffentlichungstag

der PCT-Anmeldung: **08.08.2002**

(97) Erstveröffentlichung durch das EPA: **04.02.2004**

(97) Veröffentlichungstag

der Patenterteilung beim EPA: **09.04.2008**

(47) Veröffentlichungstag im Patentblatt: **25.06.2009**

(30) Unionspriorität:

0101378 01.02.2001 FR

(84) Benannte Vertragsstaaten:

**AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT,
LI, LU, MC, NL, PT, SE, TR**

(73) Patentinhaber:

**International Business Machines Corp., Armonk,
N.Y., US**

(72) Erfinder:

**VERTES, Marc Philippe, 2 Allee du Niger, F-31000
Toulouse, FR**

(74) Vertreter:

**Duscher, R., Dipl.-Phys. Dr.rer.nat., Pat.-Anw.,
70176 Stuttgart**

(54) Bezeichnung: **VERFAHREN UND SYSTEM ZUM STEuern VON AUSFÜHRbaren DATEIEN MIT GETEILTEN BI-
BLIOTHEKEN**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

[0001] Die vorliegende Erfindung bezieht sich auf ein Verfahren und System zum Verwalten von ausführbaren Dateien mit gemeinsam genutzten Bibliotheken.

[0002] Im Allgemeinen wird die Gesamtheit der Funktionen der ausführbaren Dateien in einer individuellen Datei zusammengefasst, um umfangreiche ausführbare Dateien zu vermeiden. Diese individuelle Datei oder gemeinsam genutzte Bibliothek ist gewöhnlich in Betriebssystemen integriert. Objektcodes, die eine Funktion bestimmen, werden also nicht mehr in die ausführbaren Dateien integriert, sondern nur der Bezug auf diese Funktion ist in den ausführbaren Dateien angegeben.

[0003] Eine ausführbare Datei, die zumindest eine gemeinsam genutzte Bibliothek verwendet, bzw. eine dynamische ausführbare Datei ruft also gemeinsam verwendete Funktionen auf, die in der gemeinsam genutzten Bibliothek enthalten sind. Sogar bei Änderung eines zweitrangigen Bestandteils der ausführbaren Datei muss der gesamte Quellcode der ausführbaren Datei geändert werden, gefolgt von einer erneuten Kompilierungsphase des Quellcodes. Allerdings ist der Quellcode einer ausführbaren Datei nicht immer verfügbar.

[0004] Die Erstellung eines dynamischen Link nach dem Stand der Technik umfasst eine erste Stufe der Link-Erstellung zwischen einer aufgerufenen Funktion und einer Tabelle mit einem Zeiger auf einen Code aus der Code-Bibliothek sowie eine weitere Stufe des Aufrufs der Link-Ladefunktion aus dieser Bibliothek nur im Augenblick der Ausführung. Nach dem gleichen Prinzip ermöglicht die in der US-Patentschrift Nr. 5 546 586 beschriebene Erfindung der Firma Apple Computer die Beibehaltung des Funktionscodes durch Speicherung dieses Codes in einer Bibliothek, die sich nur im Schreib-Lesespeicher (RAM) befindet, ohne den anwendbaren Code ändern zu müssen, wenn der für die Ausführung anwendbare Code in den Festwertspeicher (ROM) geschrieben ist. Allerdings ist es wünschenswert, den Funktionscode nicht nur beim Laden des Programms, sondern auch während jeder Programmausführung ändern zu können.

[0005] Die US-Patentschrift Nr. 6 154 878 der Firma Hewlett-Packard beschreibt ein System zur Änderung der Version einer durch ein anwendbares Programm verwendeten Bibliothek, und zwar während seiner Ausführung. Die Lösung besteht darin, die Erstellung eines Link mit einem Proxy-Programm zu kompilieren, das den Aufruf einer Funktion in einer Funktionsbibliothek in einen Aufruf in einer Schnittstellen-Bibliothek abändert, die dann selbst eine Version der Funktionsbibliothek aufruft. Ein Steuerungsprogramm ermöglicht in der Schnittstellen-Bibliothek die Definition der Version der Funktionsbibliothek, die während der Ausführung des Anwendungsprogramms aufgerufen werden muss. Diese Erfindung ermöglicht es, eine Link-Erstellung durch Definieren einer Bibliothek als diejenige Bibliothek zu umgehen, aus der der Objektcode der Funktionen bei Ausführung abgerufen wird. Diese Lösung, die eine Änderung der Bibliothek ermöglicht, lässt keine Änderung des Funktionscodes zu; außerdem macht diese Lösung eine Kompilierung und Proxy-Link-Erstellung mit dem anwendbaren Programm erforderlich.

[0006] Das Ziel der Erfindung besteht darin, eine dynamische Verhaltensänderung einer dynamisch ausführbaren Datei ohne Eingriff in das System zu ermöglichen, das heißt, ohne physische Änderung der ausführbaren Datei und eventueller gemeinsam genutzter Bibliotheken.

[0007] Die vorliegende Erfindung hat ebenfalls zum Ziel, die Größe der vorhandenen ausführbaren Dateien während ihrer Ausführung dynamisch und ohne Eingriff in das System zu erhöhen.

[0008] Man erreicht die vorgenannten Ziele mit einem Verfahren nach einem der Ansprüche 1 bis 11, die die Änderung der Ausführung einer ausführbaren Datei ohne Eingriff in das System ermöglichen.

[0009] Ein Bezug wird als nicht aufgelöst angesehen, wenn der Objektcode der Funktion, auf die sie sich bezieht, in der ausführbaren Datei nicht enthalten ist.

[0010] Eine Dienstdatei ist eine Datei, die neue Leistungsmerkmale oder Funktionen enthält, die man der ausführbaren Datei hinzufügen möchte. Die Relais-Datei oder Bezugsdatei umfasst einen Teil der in der ausführbaren Datei vorhandenen Bezüge. Diese Bezüge dienen nach der herkömmlichen Art zur Ausführung von Funktionen in der gemeinsam genutzten Bibliothek. Aber mit dem Verfahren nach der Erfindung prüft man vor Erstellung eines Link zwischen jedem Bezug der ausführbaren Datei und einer entsprechenden Funktion in der gemeinsam genutzten Bibliothek zunächst, ob ein Bezug auf diese Funktion in der Bezugsdatei besteht. Wenn das der Fall ist, sagt man, dass die Bezugsdatei diese entsprechende Funktion abgefangen hat, und dann wird ein Link zwischen diesem Bezug und der Bezugsdatei erstellt. Im gegenteiligen Fall, das heißt, wenn die Funk-

tion nicht in der Bezugsdatei abgefangen ist, wird auf herkömmliche Art und Weise ein dynamischer Link zwischen der ausführbaren Datei und der entsprechenden Funktion in der gemeinsam genutzten Bibliothek erstellt.

[0011] Wenn eine Funktion abgefangen wurde, umfasst die Bezugsdatei einen Bezug auf diese Funktion, und man ordnet eine oder mehrere neue Funktionen diesen Bezug zu. In anderen Worten: Jedem Bezug der Bezugsdatei wird ein Zugangspfad auf eine neue Funktion hin zugeordnet. Diese neue Funktion kann ein(e) in der Dienstdatei vorhandene(r) Funktion (Dienst) sein. Übrigens kann der in der Bezugsdatei definierte Zugangspfad auch die Ausführung einer Funktion der gemeinsam genutzten Bibliothek möglich machen. Durch das Abfangen einer Funktion kann also das Ersetzen dieser Originalfunktion entweder durch neue Funktionen oder durch eine Funktionskombination, die die in der gemeinsam genutzten Bibliothek gespeicherte Originalfunktion gegebenenfalls umfasst, ermöglicht werden.

[0012] Von da an sind die ausführbaren Dateien veränderlich, ohne dass sie regeneriert werden müssen. Die Dienste können eingefügt, aufgehoben oder während der Ausführung geändert werden.

[0013] Die Bezugsdatei verwendet Funktionen der dynamischen Link-Erstellung aus dem Betriebssystem des Computers, um ausführbare Dateien zu ändern und Dienste zu aktivieren oder zu deaktivieren.

[0014] Nach dem Stand der Technik wird der dynamische Link-Erstellungsprozess normalerweise nur beim Laden der ausführbaren Datei vor ihrem Start aktiviert. Mit dem Verfahren nach dieser Erfindung ist dieser Prozess während der Dauer der Ausführung einer ausführbaren Datei zugänglich.

[0015] Man sagt, die Bezugsdatei ist ablaufinvariant, weil ein und dieselbe Funktion durch mehrere Dienste verändert werden kann, wobei die Ausführungsreihenfolge dieser Dienste durch diese Bezugsdatei definiert wird.

[0016] Entsprechend eines anderen Gesichtspunkts der Erfindung wird nach Anspruch 13 ein System zur Änderung der Ausführung einer ausführbaren Datei ohne Eingriff in das System vorgeschlagen.

[0017] Weitere Vorteile und Kennzeichen der Erfindung werden bei der Prüfung der detaillierten Beschreibung einer in keiner Weise einschränkenden Ausführungsform sowie der beigefügten Zeichnungen erkennbar:

[0018] [Fig. 1](#) stellt vereinfacht die physische Organisation einer ausführbaren Datei bei Verwendung einer gemeinsam genutzten Bibliothek nach dem Stand der Technik dar;

[0019] [Fig. 2](#) stellt vereinfacht die physische Organisation einer ausführbaren Datei bei Verwendung mehrerer gemeinsam genutzter Bibliotheken nach der vorliegenden Erfindung dar;

[0020] [Fig. 3](#) zeigt ein Organigramm, das verschiedene Stufen eines Verfahrens nach dieser Erfindung umfasst;

[0021] [Fig. 4](#) stellt vereinfacht die physische Organisation einer ausführbaren Datei bei Verwendung einer gemeinsam genutzten Bibliothek mit der Möglichkeit der Umleitung auf exportierte Funktionen;

[0022] [Fig. 5](#) nimmt die Elemente der [Fig. 4](#) für den Fall auf, dass ein Ersetzen einer exportierten Funktion gemäß des zweiten Modus der Umsetzung der vorliegenden Erfindung stattgefunden hat; und

[0023] [Fig. 6](#) stellt vereinfacht einen dritten Modus der Umsetzung vorliegender Erfindung im Falle einer gemeinsam genutzten Bibliothek dar, die während der Ausführung durch das Betriebssystem des Computers geladen wird.

[0024] In [Fig. 1](#) ist eine physische Organisation einer ausführbaren dynamischen Datei E bei Verwendung einer gemeinsam genutzten Bibliothek B nach dem Stand der Technik dargestellt. Die ausführbare Datei E ist ein Programm in Form eines Objektcodes, in dem Symbole F1, F2 und F3 angegeben sind. Diese Symbole sind Bezüge auf die gemeinsam genutzten Funktionen F1, F2 und F3, deren Objektcodes coF1, coF2 und coF3 in der gemeinsam genutzten Bibliothek B gespeichert sind. Diese Bibliothek ist eine getrennte, nicht mit dem Programm E verknüpfte Datei. Vor Ausführung der ersten codierten Anweisungen in Programm E erfolgt eine Phase der Auflösung der Bezugnahme zu den Symbolen bei jedem Laden des Programms E in den Speicher durch ein Urladeprogrammprogramm (oder Bootloader auf Englisch). Diese Phase der Bezugsauflösung

ist Teil einer Link-Erstellungsphase, deren Zweck es ist, jeden Bezug Fi des Programms E mit seinem in der Bibliothek B gespeicherten Objektcode zu verknüpfen.

[0025] [Fig. 2](#) zeigt ein System nach der Erfindung, in dem man die Elemente der [Fig. 1](#) wiederfindet. [Fig. 3](#) beschreibt verschiedene Stufen eines Verfahrens nach der Erfindung.

[0026] Jetzt wird eine Ausführungsform der Erfindung in Bezug auf die [Fig. 2](#) und [Fig. 3](#) beschrieben.

[0027] In [Fig. 2](#) sind im Vergleich zu [Fig. 1](#) zwei neue Dateien zu sehen. Man erkennt eine Bezugsdatei META, bei der es sich um eine spezifische gemeinsam genutzte Bibliothek handelt und die Bezüge F2 und F3 enthält. Außerdem sieht man die Dienstdatei S, eine spezifische gemeinsam genutzte Bibliothek mit den Objektcodes coX1, coX2 und coX3 der neuen Funktionen oder Dienste X1, X2 und X3. Man möchte die Ausführung des Programms E durch Einfügen einiger neuer Funktionen Xi ändern, ohne dafür die ausführbare Datei E und die Bibliothek B physisch zu verändern.

[0028] In der Datei META ist jedem Bezug eine Gleichung zugeordnet, z. B.:

$F2 = S:X2, B:F2$

und

$F3 = S:X3$

[0029] Die F2 zugeordnete Gleichung bedeutet, dass man die Ausführung der Funktion F2 durch die Ausführung der in der Datei S gespeicherten Funktion X2 und dann durch Ausführung der in der gemeinsam genutzten Bibliothek B gespeicherten Funktion F2 ersetzt.

[0030] Die F3 zugeordnete Gleichung bedeutet, dass man die Ausführung der Funktion F3 durch die Ausführung der in der Datei S gespeicherten Funktion X3 ersetzt.

[0031] Beim Starten des Programms E in der Stufe 1 wird das Betriebssystem, in dem die Erfindung ausgeführt wird, vor Ausführung der ersten in E codierten Anweisungen in Stufe 2 zum Vorab-Laden der Datei META angewiesen. Zum Beispiel wird das im System LINUX durch Setzen der Umgebungsvariablen „LD_PRELOAD“ erreicht.

[0032] Dann beginnt in Stufe 3 die Phase der dynamischen Link-Erstellung, die durch das Umladeprogramm (bootloader) ausgeführt wird. Man berücksichtigt in Stufe 4 alle Funktionen, deren Objektcode nicht in der Datei E enthalten ist, das heißt, dass die Datei E nur ihre Bezüge oder Symbole umfasst; dann spricht man davon, dass die Symbole nicht aufgelöst sind. In [Fig. 2](#) sind dies die Bezüge F1, F2 und F3. Sie werden mit Priorität META so zugeordnet, dass in Stufe 5 zuerst überprüft wird, ob sie in die META-Datei aufgenommen worden sind. Nur die Bezüge F2 und F3 werden in die META-Datei aufgenommen. In diesem Fall erstellt man in Stufe 7 einen Link zwischen den Referenzen F1 und F2 der ausführbaren Datei E und den Bezügen F1 und F2 der META-Bezugsdatei. Wenn hingegen eine Funktion in der Meta-Datei nicht in Bezug gesetzt worden ist wie z. B. die Funktion F1, erstellt man in Stufe 6 einen Link zwischen dem Bezug F1 der ausführbaren Datei E und dem Objektcode coF1 der Funktion F1, wobei dieser Objektcode coF1 in der gemeinsam genutzten Bibliothek B gespeichert wird.

[0033] Wenn die Link-Erstellungsphase in Stufe 8 beendet ist, startet man konkret die Ausführung der Datei E in Stufe 9. Bei Ausführung von E wird die Initialisierung der META-Datei durch das erstmalige Aufrufen einer der abgefangenen Funktionen ausgelöst, in diesem Fall der Funktion F2 in Stufe 10. Diese Initialisierung besteht aus der Vervollständigung der Link-Erstellungsphase der Stufen 3 bis 8. Dafür verwendet man die gleichen Routinen zur Bearbeitung von gemeinsam genutzten Bibliotheken, wie sie auch vom Umladeprogramm eingesetzt werden. Für die beiden Funktionen F1 und F2 wird Folgendes aufgelöst:

- der Bezug jeder Funktion in META auf den Einfügedienst; in [Fig. 2](#) muss F2 in der META-Datei auf X2 in der Dienstdatei S zeigen; F3 in der META-Datei muss auf X3 in der Dienstdatei S zeigen; wenn der Einfügedienst fehlt, wird die Funktion an seine ursprüngliche Bibliothek B zurückgeleitet;
- eventuell der Bezug oder die Bezüge auf die zuvor erforderlichen Funktionen in der Servicedatei S; in [Fig. 2](#) zum Beispiel reaktiviert die Funktion x2 in S die Funktion F2 in B; dieses Verhalten ist sehr wichtig, weil so eine Verkettung der Dienste erstellt und auch inkrementelle Dienste entwickelt werden können.

[0034] Diese Initialisierungsphase findet in den Stufen **11** und **14** statt. In Folge dieser META-Initialisierung ist der Aufruf-Graph zwischen den Funktionen von da an erstellt, so dass man die Funktion X2 in der Stufe **12** und anschließend die Funktion F2 in der Stufe **13** ausführen kann.

[0035] Die Speicherkapazität für die Neufestlegung des Graphs zum Funktionsabruf wird beim Starten des Programms E bereitgestellt, um Dienste in Form der gemeinsam genutzten Bibliothek einzufügen; man kann diese Dienste aber auch während der Ausführung des Programms, insbesondere über einen Befehlskanal in der META-Datei, aktivieren. Dieser Befehlskanal kann auf verschiedene Arten geöffnet werden: durch Abfangen von Funktionen in META, über ein Signal usw.

[0036] Man kann zum Beispiel die META-Datei verwenden, um ein Aufrufsystem folgender Art festzulegen:

```
"
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));}
"
```

[0037] Im obigen Beispiel bestimmt die fettgedruckte Zeile den Aufruf der untergeordneten Funktion "open". Die Variable „meta_open“ ist ein Zeiger auf die Originalfunktion, oder ein Routinevorgang führt einen anderen identischen Dienst aus. Diese Funktion, z. B. X2 definiert in S, kann durch META für eine Vielzahl von ausführbaren Dateien geladen werden, ohne dass diese Dateien geändert werden müssen.

[0038] Die META-Datei ist weder von geänderten Programmen noch von eingerichteten Diensten abhängig. Jeder Dienst aus einer gemeinsam genutzten Bibliothek liefert folgende Daten an die Meta-Datei:

- die Liste der Funktionen, die die META-Datei in Übereinstimmung mit der intern aufzurufenden Funktion abfangen muss, z. B. nach [Fig. 2](#) muss der Aufruf der Funktion F2 in E die Funktion X2 in S auslösen;
- die Liste der externen Funktionen (also derer, die abgefangen werden können), die für die Einsetzung des Dienstes Vorbedingung sind; das sind die Daten, die es der META-Datei ermöglichen, die Verknüpfung zwischen Diensten und Funktionen vorzunehmen;
- die Definition aller während des Abfangens zu aktivierender Ersatzfunktionen sowie die Definition neuer verschiedenartiger Funktionen; und
- die Liste der neuen durch den Dienst exportierten Funktionen, die die untergeordneten Dienste nutzen können; zum Beispiel exportiert META die Funktionen zur Öffnung des Befehlskanals so, dass die anderen Dienste sie verwenden können, ohne dass sie erneut erstellt zu müssen.

[0039] Die vorliegende Erfindung bringt ein alternatives Verfahren ein, um allgemeine Probleme einfach anzugehen, z. B. die Toleranz bei Anwendungsfehlern, Umstellung von Anwendungen, die externe Überwachung von Ein-/Ausgaben. Sie macht die ausführbaren Dateien ohne Eingriff in das System veränderbar. Daher sind die in einer ausführbaren Datei in dem System nach der Erfindung veränderbaren Funktionen diejenigen, die in gemeinsam genutzten dynamisch ladbaren Bibliotheken definiert sind, also Systemaufrufe oder andere (mathematische, ...) Funktionsarten.

[0040] Das Verfahren nach der Erfindung verwendet Funktionen der dynamischen Link-Erstellung des Betriebssystems, um ausführbare Dateien ohne Eingriff in das System zu bearbeiten und Dienste zu aktivieren/deaktivieren, die ihrerseits die Form von gemeinsam genutzten Bibliotheken aufweisen. Die Funktionen der dynamischen Link-Erstellung sind vorteilhafterweise während jeder Ausführung einer ausführbaren Datei zugänglich. Letztendlich ist die Bezugsdatei META ablaufinvariant, weil ein und dieselbe Funktion durch mehrere Dienste verändert werden kann. Im Übrigen ist eine Ausführung mit mehreren Bezugsdateien des META-Typs und mehreren Dienstdateien S vorstellbar.

[0041] In [Fig. 4](#) findet man die ausführbare Datei E wieder. Die Bibliothek B2 ist so gestaltet, dass sie den Objektcode der Funktion F3 nicht einschließt; sie schließt vielmehr eine Anwendung oder Pseudo-Funktion ein, „Stub“ genannt, die es ermöglicht, das Vorhandensein dieser Funktion F3 gegenüber der ausführbaren Datei E zu simulieren. In Wirklichkeit zeigt die Anwendung „Stub“ auf eine Adresse TR(2), die sich in einer Umsetzungstabelle TR befindet, welche wiederum auf die Adresse M(2) im Speicherbereich M zeigt, und zwar an der Stelle, wo sich der Objektcode coF3 der Funktion F3 tatsächlich befindet. Bemerkenswert daran ist, dass

die Umsetzungstabelle TR für die ausführbare Datei durch jedes externe Modul zugänglich ist. Die Erfindung verwendet ein Verwaltungsmodul MG, bei dem es sich um ein Systemteil oder eine Anwendung mit einer Intelligenz zur Änderung der in der Umsetzungstabelle enthaltenen Adressen handelt. In [Fig. 5](#) wird über das Verwaltungsmodul MG in der Umsetzungstabelle die Adresse M(2) durch die Adresse M(3) ersetzt. In der Zwischenzeit hat man in M(3) einen Objektcode coX4 der neuen Funktion X4 angeordnet. Wenn also die ausführbare Datei die Funktion F3 aufruft, ist es der Objektcode coX4 der Funktion X4, der verarbeitet wird. Das Verwaltungsmodul umfasst auch Mittel zur Speicherung dieser Ersetzung, um zum Anfangszustand zurückkehren zu können, also zu M(2) in der Umsetzungstabelle TR.

[0042] Zu dem Verwaltungsmodul gehört ein Überwachungsmodul, das es ermöglicht, Befehle von einem Benutzer zu empfangen, sie in eine für das verknüpft verständliche Sprache zu übertragen und diese Befehle dann zu aktivieren.

[0043] Anders gesagt, in der zweiten Ausführungsform der Erfindung wird eine ausführbare Datei oder ein Programm berücksichtigt, in dem eine Funktion aufgerufen wird, deren Definition in einer gemeinsam genutzten externen Bibliothek gespeichert ist. Für jede externe Funktion zu einem dynamischen Objekt (Programm oder gemeinsam genutzte Bibliothek), für deren Verwendung das dynamische Objekt ausdrücklich erforderlich ist, gibt es eine lokale Pseudo-Funktion („stub“ genannt) zum Objekt, die durch das Link-Erstellungsprogramm (LD unter LINUX) während der einleitenden Link-Erstellungsphase erstellt wurde. Für die gemeinsam genutzte Bibliothek ist der Zweck des „stub“ zu verbergen, dass der Objektcode der Funktion nicht wirklich in der Bibliothek lokalisiert ist. Außerdem kann dieser Objektcode während der Erstellung der gemeinsam genutzten Bibliothek unter einer unbekannten Adresse abgelegt werden. Bei Ausführung des Programms ruft diese Pseudo-Funktion den echten Objektcode durch Aufbau eines Zugangspfades zu diesem Objektcode auf. Dieser Zugangspfad wird während der Erstellung dynamischer Links erstellt und besteht darin, die Adresse des Objektcodes in einem von der Umsetzungstabelle aufgerufenen Bereich zu speichern, der sich im DATA-Segment des dynamischen Objekts befindet. Von Vorteil ist, dass dieses DATA-Segment veränderbar ist.

[0044] Die zweite Ausführungsform des Verfahrens besteht darin, die Adresse einer externen Funktion in der Umsetzungstabelle der gemeinsam genutzten Bibliothek in einer weiteren Phase der dynamischen Link-Erstellung abändern zu können, und zwar willkürlich im Laufe der Programmausführung. Dafür ersetzt man die Definition einer Funktion durch eine andere. Eine interessante Eigenschaft dieses Verfahrens ist, dass es die Änderung des Bezugs auf eine Funktion sogar dann ermöglicht, wenn das Programm während der Ausführung unterbrochen worden ist. Die Funktionen selbst werden nicht wirklich geändert, sondern nur ihre indirekte Adresse, die vorhergehende Zuordnungsmarkierung der Funktion beendet ihren Ablauf normal, die Änderung wird nur beim nächsten Aufruf der Funktion berücksichtigt.

[0045] Das Verwaltungsmodul nach der Erfindung weist insbesondere die folgenden Funktionen auf:

- eine Selbstfindungsfunktion, die folgende bereits einmal in das Programm geladene Listen bestimmen kann:
 - die Liste der dynamischen Objekte dieses Programms (teilweise ausführbar und gemeinsam genutzte Bibliotheken) sowie ihre jeweiligen Ladeadressen,
 - für jedes dynamische Objekt die Liste der externen exportierten Funktionen, das heißt, für die eine nachträglich veränderbare Eingabe in die Umsetzungstabelle vorhanden ist,
 - für jedes dynamische Objekt die Liste der exportierten Funktionen, das heißt für diejenigen, deren Auflösungsadresse möglicherweise in einer Umsetzungstabelle eines anderen dynamischen Objektes des gleichen Programms erscheint,
- eine Funktion zur Änderung der Umsetzungstabelle, die einerseits ermöglicht, die Adressänderung einer externen Funktion in der Bibliothek vorzunehmen und andererseits die nachfolgenden Aktualisierungen zu speichern, um sie möglicherweise annullieren zu können und eine vollständige Umkehrbarkeit des Systems zu erreichen,
- eine Modullade-Funktion, die aus dem Einfügen eines Befehlskanals besteht, das Laden dieses Kanals von einem Überwachungsmodul aus und anschließend seine Aktivierung nach den folgenden Methoden ermöglicht:
 - entweder bei Programmstart durch Vorab-Laden des Verwaltungsmoduls selbst in Form der gemeinsam genutzten Bibliothek
 - oder während der Ausführung des Programms durch Programmunterbrechung, Einfügen eines Haltepunktes in das Programm bei der aktuellen Anweisung über das Betriebssystem und Laden des Verwaltungsmoduls.
- eine Funktion der Modulfreigabe, die es dem Modul ermöglicht, sich während der Ausführung aus dem Programm zurückzuziehen.

[0046] Andererseits besteht für die dynamischen Programme eine andere Möglichkeit, externe Funktionen aufzurufen: das Programm kann bei laufender Ausführung auf vom Betriebssystem gelieferte Funktionen zugreifen, die es ermöglichen, eine neue gemeinsam genutzte Bibliothek während der Ausführung zu laden (Funktion „DLOPEN“ unter LINUX), anschließend auf die durch diese Bibliothek exportierten Funktionen zuzugreifen, da ihre Adressen (Funktion „DLSYM“ unter LINUX) vorliegen und diese direkt in willkürlichen lokalen Variablen gespeichert werden, also nicht mehr in einer Umsetzungstabelle, wie vorstehend beschrieben. Diese Variablen aus einem externen Modul sind von vornherein unbekannt und unzugänglich.

[0047] In [Fig. 6](#) sieht man eine gemeinsam genutzte Bibliothek B3, bei der es sich um eine vom Betriebssystem geladene gemeinsam genutzte Bibliothek handelt. Nach dem Stand der Technik sind die Adressen exportierter Funktionen normalerweise in lokalen Variablen (VL) angeordnet, und diese lokalen Variablen sind nicht verfügbar.

[0048] Nach der Erfindung betrachtet man eine Anwendung (Funktion „DLSYM“ unter LINUX) des Betriebssystems; diese Anwendung ermöglicht im Standardfall den Zugriff auf die Adresse M(2) des Objektcodes der aus der gemeinsam genutzten Bibliothek B3 exportierten Funktion F3. Man ändert dann diese Anwendung in der Art, dass sie nicht mehr die Adresse M(2) des Objektcodes der Funktion F3, sondern die Adresse einer durch das Verwaltungsmodul MG erstellten Einfügetabelle I zurückgibt.

[0049] Diese Einfügetabelle (mit der F3 entsprechenden Adresse) zeigt standardmäßig den auf Objektcode coF3 der exportierten Funktion F3 in M. Der Vorteil besteht darin, dass die Einfügetabelle über das Verwaltungsmodul MG geändert werden kann.

[0050] Das Verfahren nach der Erfindung bezieht sich auf eine Zielanwendung, für die es gilt, die Operation dynamisch zu ändern. In diesem zweiten und dritten Modus lässt dieses Verfahren insbesondere folgende Elemente für seine Einführung eingreifen:

- einen Verwaltungsmodul in Form einer gemeinsam genutzten Bibliothek mit einer Befehlssprache, die die dynamische Änderung des Funktionsaufrufgraphen zulässt. Diese Befehle können zum Beispiel sein: „Ersetzen der in Bibliothek 1 definierten Funktion1 durch die in Bibliothek 2 definierten Funktion2“, „Ersetzen annullieren“, „Auflisten der ersetzten Funktionen“, „Auflisten der neuen Funktionen“, ... Im Anschluss an einen Ersetzen-Befehl zum Beispiel wird jedes Aufrufen der Funktion1 den Aufruf der Funktion2 anstelle von Funktion1 bewirken. Diese Änderungen bleiben bis zum Ende der Programmausführung bestehen. Sie sind nicht endgültig, so dass ein Programm in seiner standardmäßigen Konfiguration neu starten kann.
- Ein Überwachungsmodul in Form eines externen Programms, das es ermöglicht:
 - das Verwaltungsmodul in einem Programm durch Einfügen eines Haltepunktes und Aufrufen der Funktion Laden des Verwaltungsmodul in Form einer gemeinsam genutzten Bibliothek während der Ausführung zu laden: „dlopen(„mg.so“)“ unter LINUX; die eingesetzte Technik hat Ähnlichkeit mit der von Fehlersuchprogrammen;
 - Dialogführung mit dem Programm (ausführbare Datei) über einen Befehlskanal, um den Aufrufgraph dieses Programms zu bearbeiten; die Befehle können über eine grafische Schnittstelle, eine Kontrollleitung oder auch durch in allen anderen Programmen enthaltene Funktionen aktiviert werden;
- abgesehen von der Änderungskapazität des Aufrufgraphen ist das externe Programm für eine gute Anwendungsleistung nicht erforderlich.

[0051] Die vorliegende Erfindung kann beispielsweise, aber nicht ausschließlich, in die folgenden Anwendungen eingeführt werden:

Dynamische und von der Art her umkehrbare Instrumentierung verstärkt eingesetzter und betriebsbereiter Informationssysteme durch Einfügen von Leistungsmessfühlern bei Testaktionen. Im Prinzip sind diese Operationen mit den jetzigen Instrumentierungstechniken, die irreversibel sind und deren Anwendungskosten mit dem Einsatzzweck nicht kompatibel sind, nicht möglich.

[0052] Aufrechterhaltung der Weiterentwicklungs- und Korrekturfähigkeit der Programme während des Betriebs durch dynamisches Ersetzen einer Funktion durch eine neue Version zur Korrektur von Programmfehlern oder durch Einführen neuer Funktionalitäten. Die gegenwärtigen Techniken machen den vollständigen Abbruch, das Ersetzen und das Wiederanlaufen der Programme erforderlich.

[0053] Einrichtung von Systemprozess-Definitionen ohne Eingriff in die Anwendungen und in das Betriebssystem zum Beispiel durch Ersetzen von Systemfunktionen, die die Verarbeitung auf ein anderes Gerät bei Störung umleiten kann. Die gegenwärtigen Techniken erlauben diese Erweiterungen nicht ohne Eingriff in das System selbst oder ohne Umschreiben der spezifischen Anwendungen.

[0054] Integration heterogener Anwendungen ohne Eingriff in das System durch Erweitern der Stecker und Konvertierungsfilter, wodurch sie zweckmäßig durch Ersetzen auf den vorhandenen Schnittstellen eingefügt werden können, ohne die Anwendung umschreiben zu müssen.

[0055] Die Erfindung ist selbstverständlich nicht auf die Beispiele beschränkt, die beschrieben worden sind, und zahlreiche Veränderungen können an diesen Beispielen vorgenommen werden, ohne den Rahmen der Erfindung zu verlassen.

Patentansprüche

1. Verfahren zur Änderung der Ausführung einer ausführbaren Datei ohne Eingriff in das System, die Symbole zur Bezugnahme auf in gemeinsam genutzten Bibliotheken innerhalb des Computers angeordneten Funktionen umfasst, **dadurch gekennzeichnet**, dass das Verfahren Folgendes umfasst:

- beim Starten der ausführbaren Datei Vorab-Laden (2) einer Relais-Datei, die einigen der Symbole der ausführbaren Datei jeweilige Zugangspfade zu mindestens einer Ersatzfunktion hin zuordnet.
- Ausführen einer dynamischen Link-Erstellungsphase (3), die für jedes in der ausführbaren Datei enthaltene Symbol die Zuordnung des Symbols zur Relais-Datei (5, 7) umfasst, wenn das Symbol in der Relais-Datei definiert ist, und wenn nicht, d. h., falls das Symbol einen Bezug auf eine Funktion darstellt, die in einer gemeinsam genutzten Bibliothek angeordnet ist, Zuordnen des Symbols zur gemeinsam genutzten Bibliothek (5, 6) und
- während der Ausführung der ausführbaren Datei (9), in Reaktion auf einen Aufruf einer Datenfunktion, auf die durch eines der Symbole in der ausführbaren Datei ein Bezug besteht, Ausführen der Ersatzfunktionen, die mit dem Symbol in der Relais-Datei (12, 13) verbunden sind, wenn das Symbol der Relais-Datei zugeordnet wurde, und andernfalls die Datenfunktion, wenn das Symbol der gemeinsam genutzten Bibliothek zugeordnet wurde.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die Relais-Datei (META) als gemeinsam genutzte Bibliothek bearbeitet wird.

3. Verfahren nach einem der Ansprüche 1 und 2, dadurch gekennzeichnet, dass die Zuordnungsphase des Symbols zur Relais-Datei die Erstellung eines Link zwischen der ausführbaren Datei und der Relais-Datei für das Symbol (7) umfasst.

4. Verfahren nach einem der vorstehenden Ansprüche, dadurch gekennzeichnet, dass jeder mit einem Symbol in der Relais-Datei verbundene Zugangspfad eine vorgeschriebene Kombination der Ersatzfunktionen definiert und dadurch, dass die Ausführungsphase der Ersatzfunktionen (12, 13) entsprechend der vorgeschriebenen Kombination, die durch den verbundenen Zugangspfad definiert ist, eingeleitet wird.

5. Verfahren nach Anspruch 4, dadurch gekennzeichnet, dass bestimmte Ersatzfunktionen der Kombination in den gemeinsam genutzten Bibliotheken (13) lokalisiert werden.

6. Verfahren nach einem der vorstehenden Ansprüche, dadurch gekennzeichnet, dass zumindest einige Ersatzfunktionen in Dienstdateien angeordnet sind.

7. Verfahren nach Anspruch 6, dadurch gekennzeichnet, dass die Dienstdateien als gemeinsam genutzte Bibliotheken bearbeitet werden.

8. Verfahren nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, dass es im Verlauf der Ausführung einer ausführbaren Datei die Vorab-Initialisierung der Relais-Datei (11, 14) umfasst, und zwar beim ersten Aufruf einer Funktion, in Bezug gesetzt durch ein Symbol, das der Relais-Datei zugewiesen wurde.

9. Verfahren nach Anspruch 8, dadurch gekennzeichnet, dass die Initialisierungsphase eine dynamische Erstellungsphase der Links (11, 14) umfasst, die die Erstellung von Links zwischen jedem Symbol umfasst, das der Relais-Datei zugewiesen wurde, und der mit dem Symbol in der Relais-Datei verbundenen Ersatzfunktionen.

10. Verfahren nach Anspruch 9, dadurch gekennzeichnet, dass die dynamische Phase der Erstellung der Links (11, 14) Routinen zur Bearbeitung der gemeinsam genutzten Bibliotheken einleitet.

11. Verfahren nach irgendeinem der vorangegangenen Ansprüche, dadurch gekennzeichnet, dass die in der

Relais-Datei definierten Zugangspfade über einen Befehlskanal steuerbar und veränderbar sind.

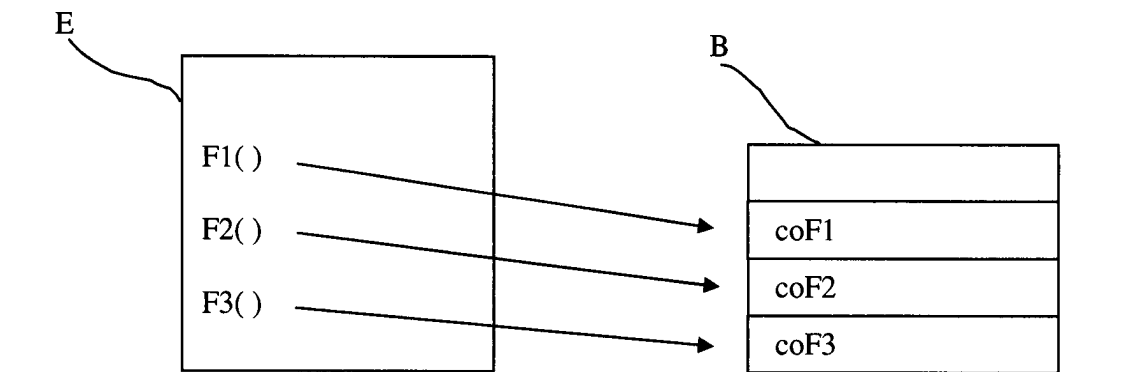
12. Verfahren nach Anspruch 11, dadurch gekennzeichnet, dass der Befehlskanal außerdem für die Steuerung der Ersatzfunktionen verwendet wird, die mit den Symbolen in der Relais-Datei verbunden sind.

13. System zur Änderung der Ausführung einer ausführbaren Datei (E) ohne Eingriff in das System, die Symbole (F1, F2, F3) umfasst, welche einen Bezug zu den Funktionen darstellen, die in gemeinsam genutzten Bibliotheken innerhalb des Computers angeordnet sind, dadurch gekennzeichnet, dass das Verfahren Folgendes beinhaltet:

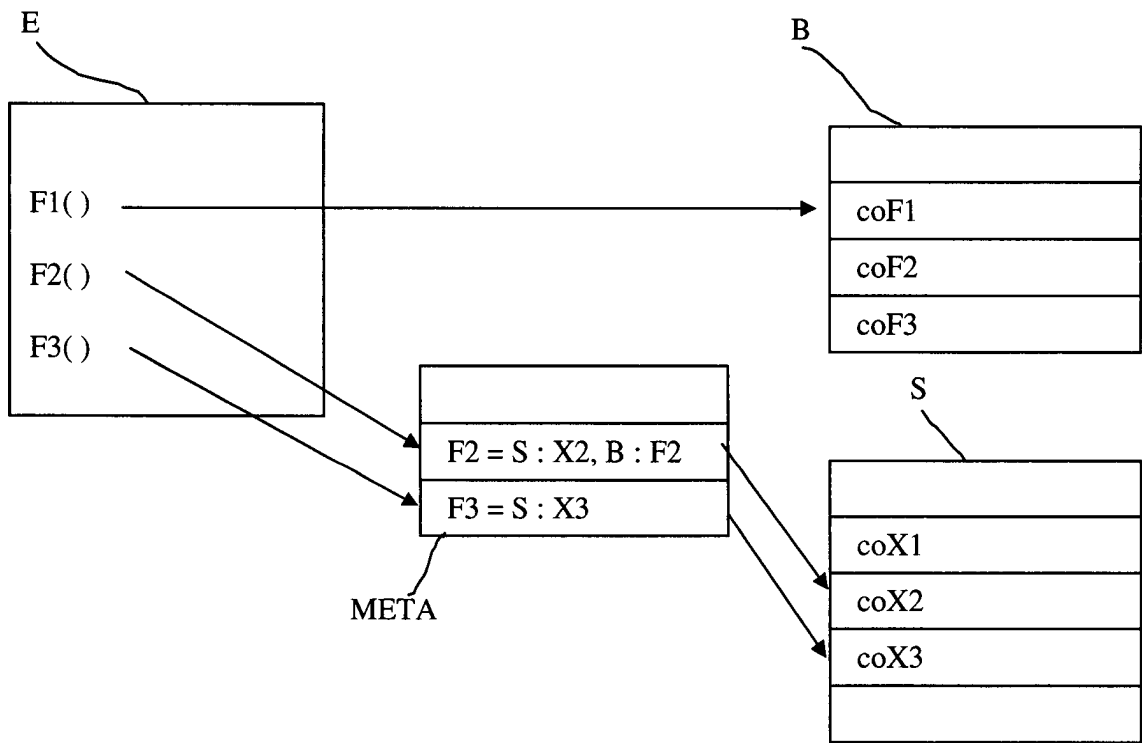
- eine Relais-Datei (META), die bereits beim Starten der ausführbaren Datei vorab geladen wird, wobei die Relais-Datei entsprechende Zugangspfade zu wenigstens einer Ersatzfunktion (X1, X2 X3) hin mit einigen der Symbole der ausführbaren Datei verbindet,
- einen dynamischen Link-Editor, der jedem in der ausführbaren Datei enthaltenen Symbol das Symbol der Relais-Datei (META) zuordnen kann, wenn das Symbol in der Relais-Datei definiert ist, und wenn dem nicht so ist, wenn also das Symbol einen Bezug zu einer in einer gemeinsam genutzten Bibliothek angeordneten Funktion (B) darstellt, der Link-Editor das Symbol der gemeinsam genutzten Bibliothek (B) zuordnen kann, wobei das System im Verlauf der Ausführung der ausführbaren Datei (E) auf den Aufruf einer Datenfunktion hin reagiert, auf die durch eines der Symbole in der ausführbaren Datei Bezug genommen wird, indem es die mit dem Symbol in der Relais-Datei (META) verbundenen Ersatzfunktionen ausführt, wenn das Symbol der Relais-Datei zugewiesen worden ist, und andernfalls die Datenfunktion, wenn das Symbol der gemeinsam genutzten Bibliothek zugewiesen worden ist (B).

Es folgen 4 Blatt Zeichnungen

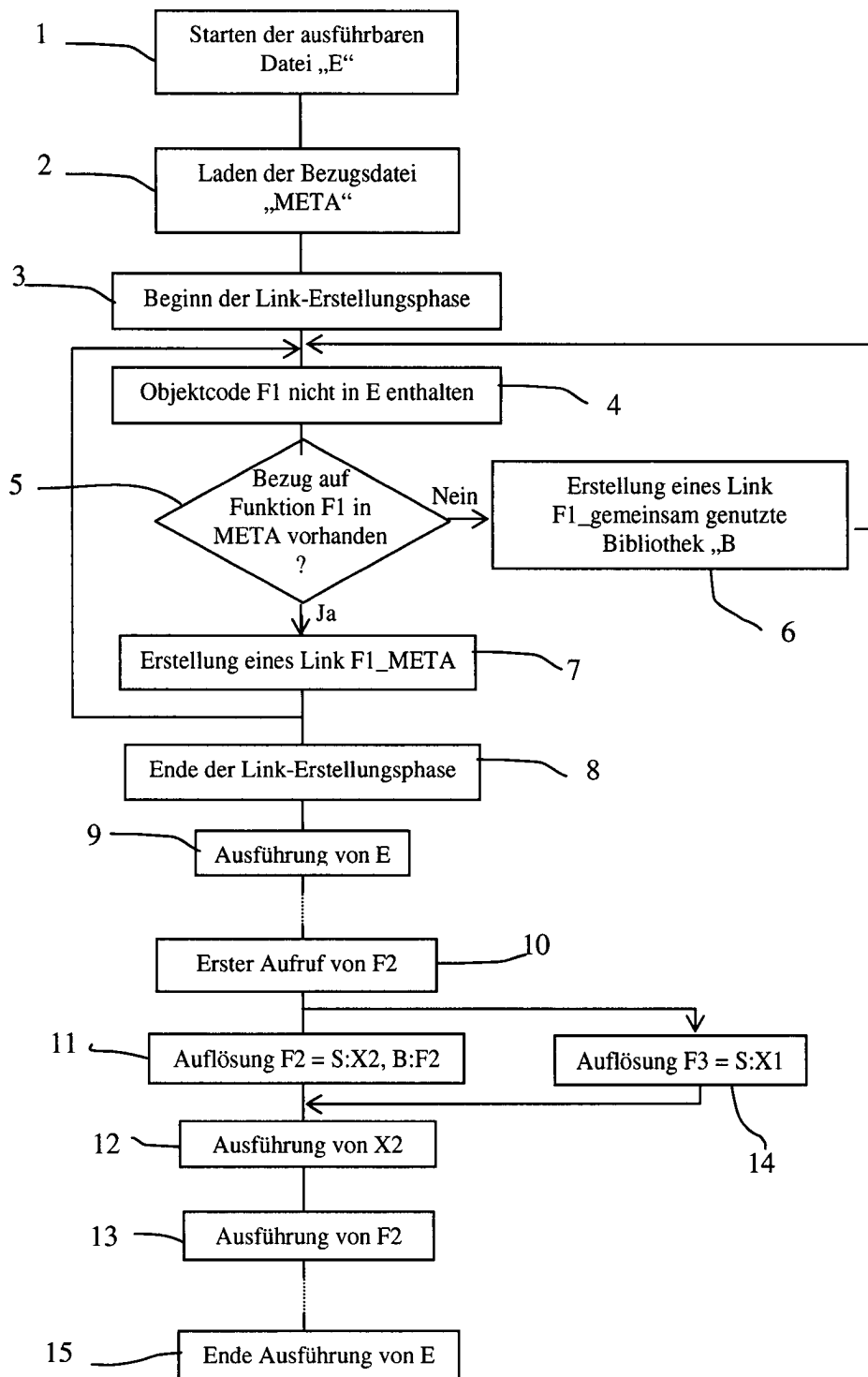
Anhängende Zeichnungen

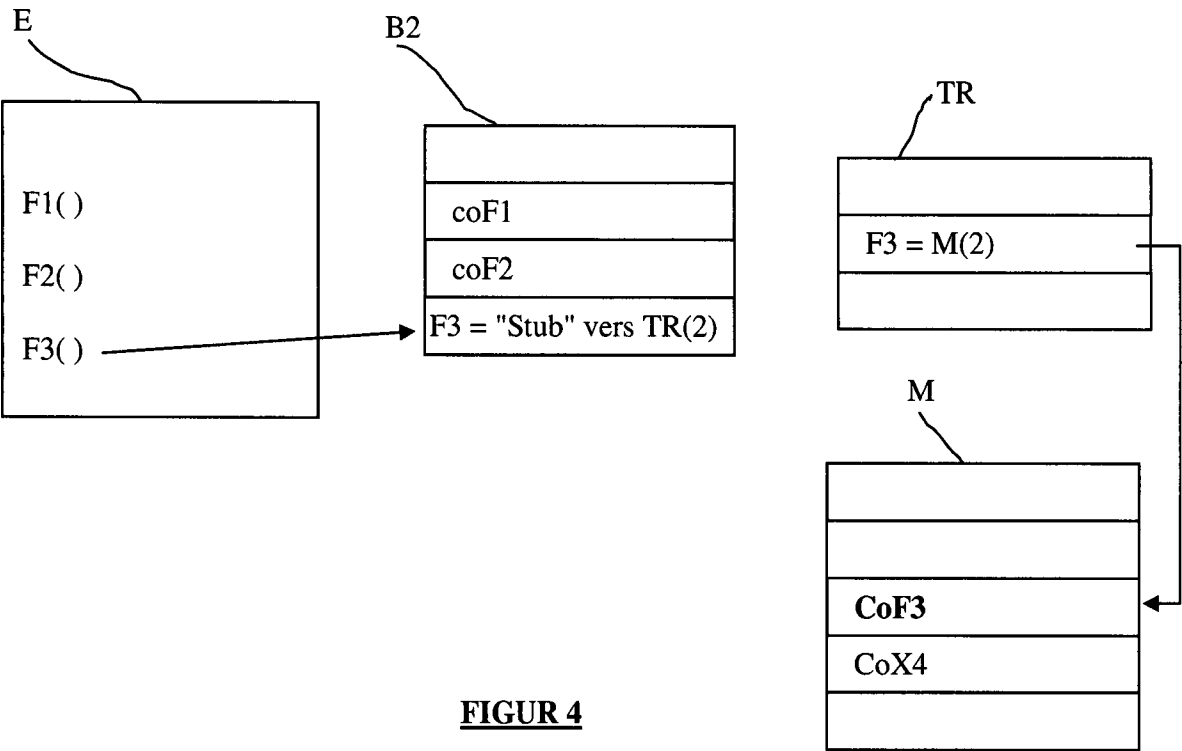


FIGUR 1

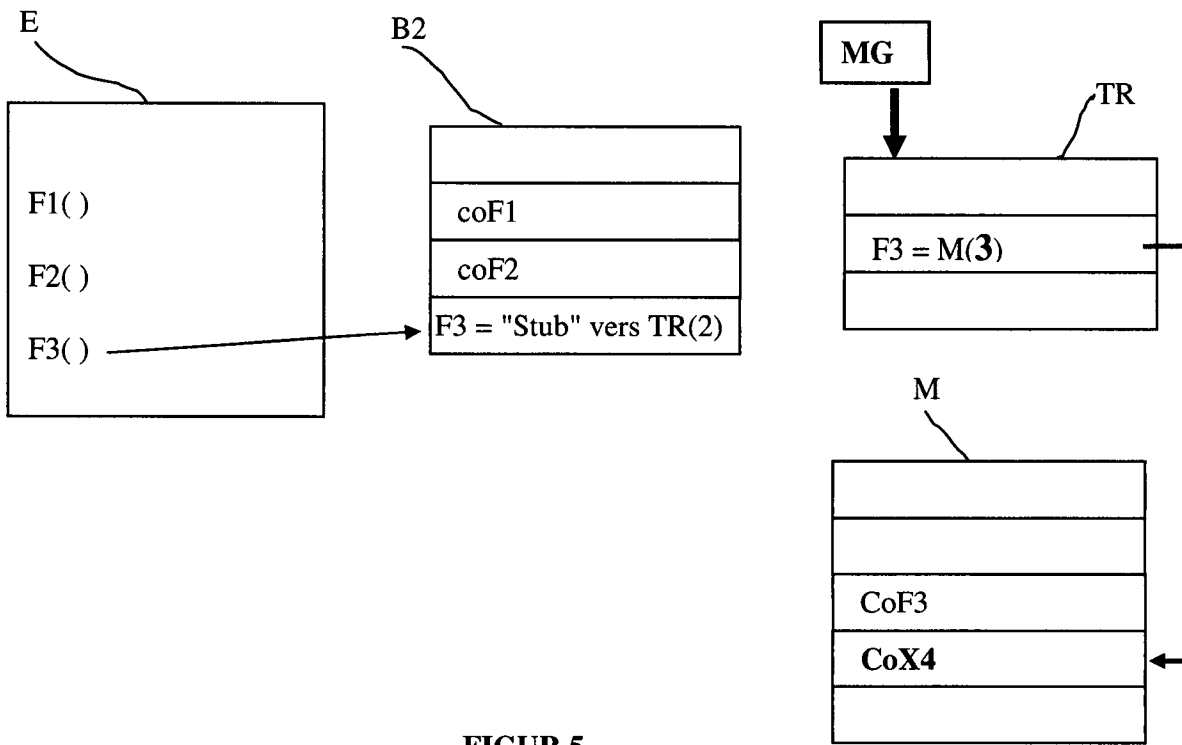


FIGUR 2

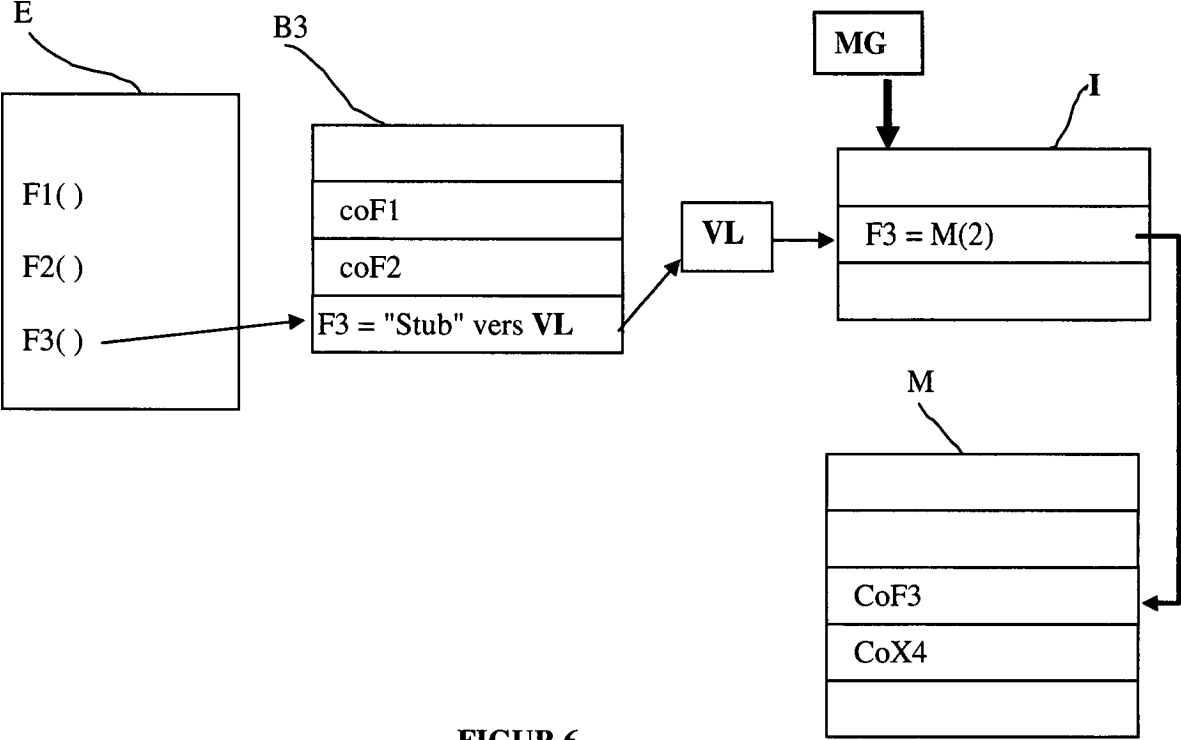
**FIGUR 3**



FIGUR 4



FIGUR 5



FIGUR 6