[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/45

[12] 发明专利说明书

专利号 ZL 98124159.X

[45] 授权公告日 2005 年 10 月 12 日

[11] 授权公告号 CN 1222874C

[22] 申请日 1998.11.12 [21] 申请号 98124159.X

[71] 专利权人 英业达股份有限公司
地址 台湾省台北市
[72] 发明人 林光信 陈玄同 侯震宇
审查员 杨 蕊

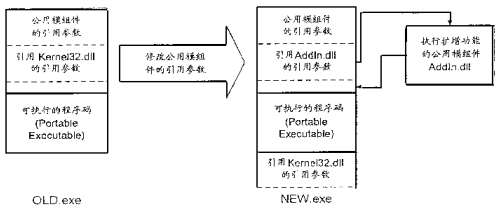
[74] 专利代理机构 北京市柳沈律师事务所
代理人 马 莹

权利要求书 2 页 说明书 7 页 附图 5 页

[54] 发明名称 动态链接的电脑系统中扩增程序功能的方法

[57] 摘要

一种动态链接的电脑系统中扩增程序功能的方法，利用动态链接的运作方式，将扩增功能以公用模组件的形式完成，并在一程序中以公用模组件取代其原先引用的另一公用模组件，同时存储被取代公用模组件的相关引用数据，使该程序执行前先执行扩增功能的公用模组件，随后扩增功能的公用模组件按所存储相关引用数据，将被其取代的公用模组件还原至程序中，使其正常执行程序并以扩增功能公用模组件的执行结果，决定其执行方式。



ISSN1008-4274

知识产权出版社出版

1. 一种动态链接的电脑系统中扩增程序功能的方法，其特征在于，该方法包括:

以一新增公用模组件取代一可执行程序中引用的一原有公用模组件，并将该原有公用模组件的相关引用参数存储于该可执行程序中的步骤，该步骤包括:

首先读取磁盘中要修改的可执行程序文件;

判断其是否为 32 比特可执行程序，如果不是则结束修改操作，如果是则对该可执行程序文件进行如下的修改;

将修改项中包含的信号保存下来附加于可执行程序文件尾部;

将该项目的引入文件名更改为自行编写的动态链接库的文件名，并将其改为到自行编写的动态链接库的文件名引入函数的名称或序号的表，且将该引入函数改为自行编写的动态链接库的文件名的函数; 和

将修改过的可执行程序写回磁盘，以保存修改结果，

以及在该新增公用模组件执行完后，使该可执行程序引用该原有公用模组件，随后执行该可执行程序的步骤，该步骤包括:

操作系统执行可执行程序时，根据其引入库表，将自行编写的动态链接库的文件名载入存储器，并执行其入口点函数;

在其入口点函数中，执行任何程序码以进行所希望的新增功能;

判断是否继续执行可执行程序文件，若否，则退出执行，若是，则继续执行;

将保存在可执行文件的文件尾部的文件名，与其引入函数的名称或序号的表，填写到可执行程序引入库表中的原来位置;

由磁盘中将动态链接库载入至存储器，并对照动态链接库中的引入项，将存储器中动态链接库引入函数的正确地址填写到引入库表中;

动态链接库将控制权交回至可执行文件; 和

执行可执行文件中原可执行程序的功能。

2. 如权利要求 1 所述的一种动态链接的电脑系统中扩增程序功能的方法，其特征在于，该方法以多个新增公用模组件取代一可执行程序中引用的多个原有公用模组件，并将该各原有公用模组件的相关引用参数存储于

该可执行程序中。

3. 如权利要求 1 所述的一种动态链接的电脑系统中扩增程序功能的方法，其特征在于，该新增公用模组件，包括将存储单元中，一可执行程序已修改为引用该新增公用模组件的相关引用参数，以可执行程序文件的尾端所存储的该原有公用模组件的相关引用参数值取代。

4. 如权利要求 1 所述的一种动态链接的电脑系统中扩增程序功能的方法，其特征在于，进一步包括建立至少一新增公用模组件，该新增公用模组件可供电脑系统中的一可执行程序相关引用参数值引用。

5. 如权利要求 4 所述的一种动态链接的电脑系统中扩增程序功能的方法，其特征在于，新增公用模组件由输出屏幕输出至少一显示画面。

6. 如权利要求 5 所述的一种动态链接的电脑系统中扩增程序功能的方法，其特征在于，该显示画面显示出该可执行程序的使用期限。

7. 如权利要求 1 所述的一种动态链接的电脑系统中扩增程序功能的方法，该新增公用模组件根据该输入数据，将存储单元中，该可执行程序已修改为引用该新增公用模组件的相关引用参数，以可执行程序文件的尾端所存储的该原有公用模组件的相关引用参数值取代。

动态链接的电脑系统中扩增程序功能的方法

技术领域

本发明涉及一种动态链接的电脑系统中扩增程序功能的方法，特别是涉及一种在不对原程序码重新编译、链接的情况下，加入新编程序码以扩增新功能，且使不同电脑系统都可共用所增加新编程序码的方法。

背景技术

在一电脑系统中，要发展出一应用程序，使电脑系统可按设计者在应用程序中的规划，进行一连串的处理过程，必须经过程序源代码(source code)撰写、将程序源代码编译(Compile)成电脑系统可执行的格式(Portable Executable以下简称PE格式)、再将该PE格式码与电脑系统提供的公用模组件(Utility Module)链接(Link)等步骤，之后电脑系统才能正确地按设计者的规划进行处理。

程序源代码通常是以适合人阅读的高级语言(High Level Language)来编写，用方便程序撰写者进行思考、规划；而PE格式码则是一种适合电脑系统读取的机器码(machine code)，该PE格式码因涉及电脑系统的硬件结构，及其低级运作〔通常与寄存器(Register)、指针(Pointer)、地址(Address)、堆栈(Stack)等有关〕，极为复杂而难懂；公用模组件则是电脑系统中，将大多数程序经常进行运作，按计算机系统的特定格式，写成可被其他程序重复引用的模组件，例如：微软公司(Microsoft)视窗(Windows)操作系统(Operating System)下的动态链接库(Dynamic Link Library)。

而现今的电脑系统为了减少众多程序中，每一个别程序都包括其本身引用的公用模组件，造成系统资源的大量浪费(例如一公用模组件被1000个程序所引用，则该各程序中便存在着1000份重复的该公用模组件，占用大量存储空间)，多采用动态链接(Dynamic Linking)模式，在该模式下，程序中仅存有其所引用公用模组件的相关信号(如公用模组件的名称及其输入参数、输出参数等)，电脑系统在程序执行时，才动态地将程序本身与

其所引用的公用模组件进行链接，之后程序才能正常地执行。

当设计者依照其构想、规划、撰写一应用程序(Application program)使电脑系统按其意思进行处理后，通常该应用程序的内容就不再变动，此时当使用者有了新构思，欲在该应用程序中加入新功能，例如，在一程序执行以前加入使用期限检查等功能时，现有技术通常采用下列两种方法实施:

1. 应用程序重整: 即先编写好一段程序源代码(source code)，此段程序源代码用以执行所希望添加的新功能，再将这段程序源代码所执行的功能加入可执行程序中，即将这段新编写的程序源代码，与原可执行程序的程序源代码结合，成为一包括扩增功能的新程序源代码，再重新编译(Compile)该新程序源代码，得到一个新的 PE 格式码 - -PE，再将该 PE 格式码与电脑系统的公用模组件链接(Link)，之后电脑系统才能正确进行处理。

如此新的可执行程序 PE 即可完成扩增的功能，并且不影响原可执行程序应有的功能，其缺点是必须取得欲扩增功能 PE 格式文件的程序源代码，且需要在相关的电脑系统环境下进行编译(Compile)和链接(Link)，例如: 微软公司的 Windows 操作系统下的程序码，只能在 Windows 操作系统环境下进行编译(Compile)和链接(Link)，而微软公司的 NT 操作系统下的程序码只能在 NT 操作系统环境下进行编译(Compile)和链接(Link)，故应用程序重整的方法受操作系统的限制，费时费力，如图 1A 所示。

2. 用汇编语言(Assembly Language)编写一段程序码，这段汇编语言程序码可以实现附加功能，再修改原 PE 格式码的入口点(Entry Point，即程序开始执行的起点)，使之与自编程序码相呼应，指向自编程序码的开始执行位置，此种方法也可在原程序码中实现附加的功能，但必须修改原来 PE 格式码的内容，其缺点在(a)复杂，因 PE 格式码是电脑所执行的机器码(Machine Code)，较为艰涩难懂; (b)由于汇编语言与硬件结构密切相关，不同的硬件结构有不同的汇编语言，因此该方法虽优于前一种技术方法，但对于每一种不同的硬件平台结构(CPU 类型)，都要重新编写一对应的汇编语言程序码，因而在应用上有很大的限制，如图 1B。

发明内容

本发明所采用的方法，克服上述两种现有技术的缺点，无须重新编译或链接，不受操作系统与硬件平台的限制，即可达到扩充功能的目的，如图 1C 所示。

本发明为一种动态链接的电脑系统中扩增程序功能的方法，该方法主要是利用该电脑系统的动态链接特性，在其可执行程序码(PE 格式码)中，将对公用模组件的引用，改以对新编程序码的引用取代，使原程序码在执行以前，必先执行新编程序码，以在不对原程序码重新编译、链接的情况下，加入新编程序码以扩增功能。

本发明的目的在于由修改可执行程序中的动态链接信号，以在不重新编译、链接的情况下，在一可执行程序中加入新编程序码以扩增功能，且使不同电脑系统都可共用所增加的新编程序码。

本发明的目的是这样实现的，即提供一种动态链接的电脑系统中扩增程序功能的方法，该方法包括：以一新增公用模组件取代一可执行程序中引用的一原有公用模组件，并将该原有公用模组件的相关引用参数存储于该可执行程序中的步骤，该步骤包括：首先读取磁盘中要修改的可执行程序文件；判断其是否为 32 比特可执行程序，如果不是则结束修改操作，如果是则对该可执行程序文件进行如下的修改；将修改项中包含的信号保存下来附加于可执行程序文件尾部；将该项目的引入文件名更改为自行编写的动态链接库的文件名，并将其改为到自行编写的动态链接库的文件名引入函数的名称或序号的表，且将该引入函数改为自行编写的动态链接库的文件名的函数；和将修改过的可执行程序写回磁盘，以保存修改结果，以及在该新增公用模组件执行完后，使该可执行程序引用该原有公用模组件，随后执行该可执行程序的步骤，该步骤包括：操作系统执行可执行程序时，根据其引入库表，将自行编写的动态链接库的文件名载入存储器，并执行其入口点函数；在其入口点函数中，执行任何程序码以进行所希望的新增功能；判断是否继续执行可执行程序文件，若否，则退出执行，若是，则继续执行；将保存在可执行文件的文件尾部的文件名，与其引入函数的名称或序号的表，填写到可执行程序引入库表中的原来位置；由磁盘中将动态链接库载入至存储器，并对照动态链接库中的引入项，将存储器中动态链接库引入函数的正确地址填写到引入库表中；动态链接库将控制权交回至可执行文件；和执行可执行文件中原可执行程序的功能。

附图说明

下面结合附图，详细说明本发明的实施例，其中：

图 1A，图 1B 和图 1C 为现有技术与本发明的原理对照图；

图 2 为本发明一实施例对一可执行程序实施前后的对照图；

图 3 为本发明一实施例的部分流程图；

图 4 为本发明一实施例的部分流程图；

图 5 为本发明一实施例的屏幕显示图。

具体实施方式

本发明是一种动态链接的电脑系统中扩增程序功能的方法，尤指一种一公用模组件，可使自行编写的程序码在一可执行程序码之前执行，使扩充原可执行程序码的功能，且在此扩充功能的过程，不需对原可执行程序码重新编译、链接，而在执行完自行编写的程序码以后，动态地将原可执行程序码还原为未加入自行编写的程序码以前的内容，使执行完自行编写的程序码之后，可正常执行原可执行程序码的功能。

为能清楚揭露本发明的技术特点，以下就本发明在微软公司 Windows 操作系统下的一实施例说明，在该实施例中，原可执行程序为 OLD.exe，该可执行程序中引用了公用模组件，文件名为 Kernel32.dll 的动态链接库 (Dynamic Link Library—DLL)，而自行编写的程序码则存于文件名为 AddIn.dll 的动态链接库中，而 NEW.exe 为引用了 AddIn.dll 新的可执行程序，本实施例的实施步骤逐一说明如下：

在动态链接的电脑系统中，PE 格式文件都包括有一引入库表(Import Library List)，其结构如下：

.....
引入 1# DLL
引入 2# DLL
.....

此引入库表(Import Library List)中的每一项表示了对一个 32 比特动态链接库的引入，其中最主要的信号是要引入的动态链接库的文件名，及要由此动态链接库中引入的函数的函数名称或序号表(引入函数表)；在

每一动态链接库中，也都包括一与引入函数表对应的表(输出函数表)，其中列出了一动态链接库文件中的所有输出函数。

当 Windows 操作系统执行可执行程序 OLD.exe 时,会先行搜寻(Search)其引入库表(Import Library List)，将该表中包括有其文件名的每一动态链接库(本实施例中为 Kernel32.dll)，由磁盘载入存储器内，然后对照引入库表中的信号，将存储器内该动态链接库 Kernel32.dll 中，每一引入函数(Import Function)的地址(address)，填写到引入函数表中，之后才执行动态链接库 Kernel32.dll 的入口点函数(Entry Function)。

当 Windows 操作系统完成此搜寻工作后，可执行程序 OLD.exe 所使用到的所有动态链接库都已载入存储器内，也正确填写了所有要引用的引入函数(Import Function)的地址，此时 Windows 操作系统才执行可执行程序 OLD.exe 的入口点函数(Entry Function)，并开始执行可执行程序 OLD.exe 中的程序码。

由此可知，引入库表(Import Library List)中所指定的动态链接库 Kernel32.dll，其入口点函数会优先于可执行程序 OLD.exe 的入口点函数执行，即动态链接库的入口点函数必先于可执行程序的入口点函数执行。

故只要自行编写一个动态链接库 AddIn.dll,在该动态链接库的入口点函数中完成欲新加入的功能，再修改可执行程序 OLD.exe，使其引入库表中包括有自行编写的动态链接库 AddIn.dll，即可达成在可执行程序 OLD.exe 执行前，先执行自编程序码 AddIn.dll 的目的。

而由于 32 比特可执行程序格式上的限制，在其引入库表中增加一项相当困难，因此较可行的做法是修改引入库表中的一项，使其所包括的动态链接库的文件名、要由此动态链接库中引入函数的名称或序号，变更为自行编写的动态链接库的文件名，及由该动态链接库中引入函数的名称或序号的表(引入函数表)。

因此，被修改的可执行程序 OLD.exe 中至少需引入一动态链接库，而实际上，Windows 操作系统下的 32 比特可执行程序都满足此条件，为便于以下的描述，在此以 Kernel32.dll 为引入库表中的待修改项所包括的动态链接库，详述本发明修改可执行程序 OLD.exe 的步骤。

为能在一可执行程序 OLD.exe 执行之前，先执行包括在 AddIn.dll 中自行编写的程序码，首先要对磁盘中存储的该可执行程序码文件 OLD.exe

进行修改，其修改的流程如图 3 中所示。

首先读取磁盘中要修改的可执行程序文件 OLD.exe(步骤 3－10)，并判断其是否为 32 比特(Bit)可执行程序(步骤 3－11)，否则结束修改动作；若 OLD.exe 为 32 比特可执行程序，对该可执行程序文件进行修改，其修改方法与 Windows 操作系统 32 比特可执行程序的结构密切相关。

若 OLD.exe 为 32 比特可执行程序，由其引入库表中，找出对 Kernel32.dll 引入的项目，而为避免该引入库表中，原来对 Kernel32.dll 及其对应的引入函数表的相关信号被覆盖掉，故于修改前，须先将待修改项中所包括的信号(即文件名 Kernel32.dll 与该文件中所包括引入函数的名称或序号的表)，保存下来附加于 NEW.exe 文件尾部(步骤 3－12)，以便在执行完自编程序码 AddIn.dll 之后，由 NEW.exe 文件尾部所保留的该各信号，使可执行程序 NEW.exe 的引入库表中，被修改项目可还原成 Kernel32.dll 的文件名，及该文件中所包括引入函数的名称或序号的表。

进行修改时，将该项目的引入文件名 Kernel32.dll 更改为自行编写的动态链接库的文件名 AddIn.dll，并将该项目中的引入函数表改为由 AddIn.dll 引入函数的名称或序号的表(引入函数表)，且该表只引入 AddIn.dll 中的一个函数(步骤 3－13)，如此，原可执行程序 OLD.exe 的引入库表中就包括一引入 AddIn.dll 的项目；之后将存储器中修改过的可执行程序 NEW.exe 写回磁盘，以保存修改结果(步骤 3－14)。

本发明该实施例，将原可执行程序 OLD.exe 修改为可执行程序 NEW.exe，就两者文件间的内容作一比较，如图 2 所示。

请参阅图 4 所示的流程图，依据本发明的方法，在可执行程序 NEW.exe 中加入自行编写的动态链接库 AddIn.dll 后，其处理流程如下：

Windows 操作系统执行可执行程序 NEW.exe 时，根据其引入库表，将 AddIn.dll 载入存储器，并执行其入口点函数(Entry Function)(步骤 4－11)，在 AddIn.dll 的入口点函数中，可执行任何程序码以进行所希望的新增功能(步骤 4－12)，之后若不继续执行程序 NEW.exe，则直接结束(步骤 4－18)。

执行完 AddIn.dll 的入口点函数中的新增功能后，判断是否要继续执行程序 NEW.exe(步骤 4－13)，如要继续执行程序 NEW.exe，则修改存储器中的可执行程序 NEW.exe，使其引入 Kernel32.dll 动态链接库，使其可正

确执行，具体步骤如下：将引入库表中被覆盖掉，而保存在文件尾部的Kernel32.dll 文件名，与其引入函数的名称或序号的表(引入函数表)，填写到可执行程序 NEW.exe 引入库表中的原来位置(步骤 4－14)，接着依Windows 操作系统的方法，由磁盘中将 Kernel32.dll 载入至存储器，并对照引入库表中 Kernel32.dll 的引入项，将存储器内 Kernel32.dll 中引入函数的正确地址填写到引入库表中(步骤 4－15)，接着由 AddIn.dll 的入口点函数返回(步骤 4－16)，此时存储器中的可执行程序 NEW.exe 与未在引入库表中加入 AddIn.dll 前的 OLD.exe 相同，因此 Windows 操作系统可正确执行该可执行程序 NEW.exe，进行原有功能(步骤 4－17)，直至其完全结束而退出执行(步骤 4－18)。

本发明可于 AddIn.dll 的入口点函数中，执行任何程序码以进行所希望的新增功能(步骤 4－12)，以下举两具体范例说明：

在一试用版程序中加入试用期的保护，其目的是在原可执行程序中加入试用信息的处理，以于其执行前先显示试用期限信息(请参阅图 5 所示)，并限制用户使用该试用版程序期间(如 30 天)，同时显示购买正式版程序的方法；若用户选择继续执行，而当时尚未超出试用期限，则动态恢复原可执行程序使其正常执行，而若当时已超出试用期限，则结束该试用版程序的执行。

另一范例应用在限制特定使用者执行应用程序上，如将一密码检查功能加入解压缩程序前，则未被授权的使用者(未持有密码或密码错误)因无合格密码以执行解压缩程序，故可使未被授权的使用者不能解压缩文件档案，以防止其查看该文件档案的内容。

本发明的方法因使自行编写的程序码在现有的应用程序之前执行，且不需对现有应用程序再进行编译、链接，以及修改其可执行程序文件，故依本发明的方法可不受开发环境限制，适用于各种硬件平台，实施简易且具有移植性。

以上所述，仅为本发明的实施例，但是，本发明所保护的权利范围，并不局限于此，凡熟悉该项技术人员，依据本发明所揭露的技术内容，可作些改变，均应属于本发明的保护范围。

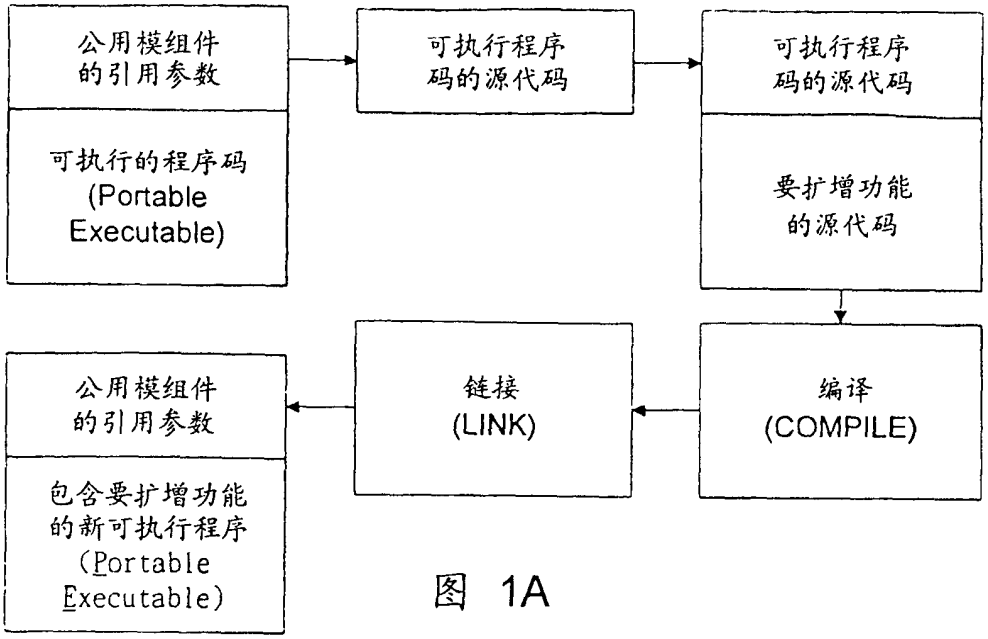

公用模组件的引用参数
可执行的程序码(Portable Executable)
可执行程序码的源代码
可执行程序码的源代码
要扩增功能的源代码
编译(COMPILE)
链接(LINK)
公用模组件的引用参数
包含要扩增功能的新可执行程序(Portable Executable)

图 1A

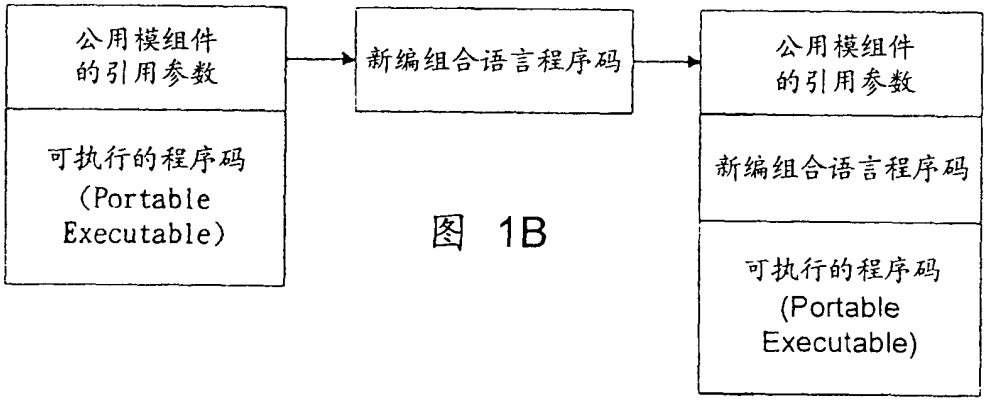

公用模组件的引用参数
可执行的程序码(Portable Executable)
新编组合语言程序码
公用模组件的引用参数
新编组合语言程序码
可执行的程序码(Portable Executable)

图 1B

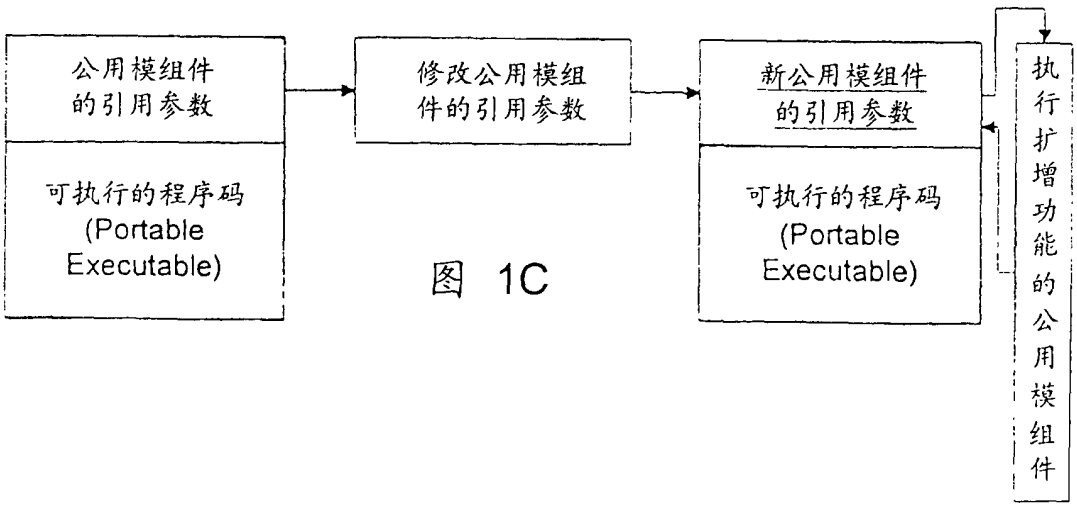

公用模组件的引用参数
可执行的程序码(Portable Executable)
修改公用模组件的引用参数
新公用模组件的引用参数
可执行的程序码(Portable Executable)
执行扩增功能的公用模组件

图 1C

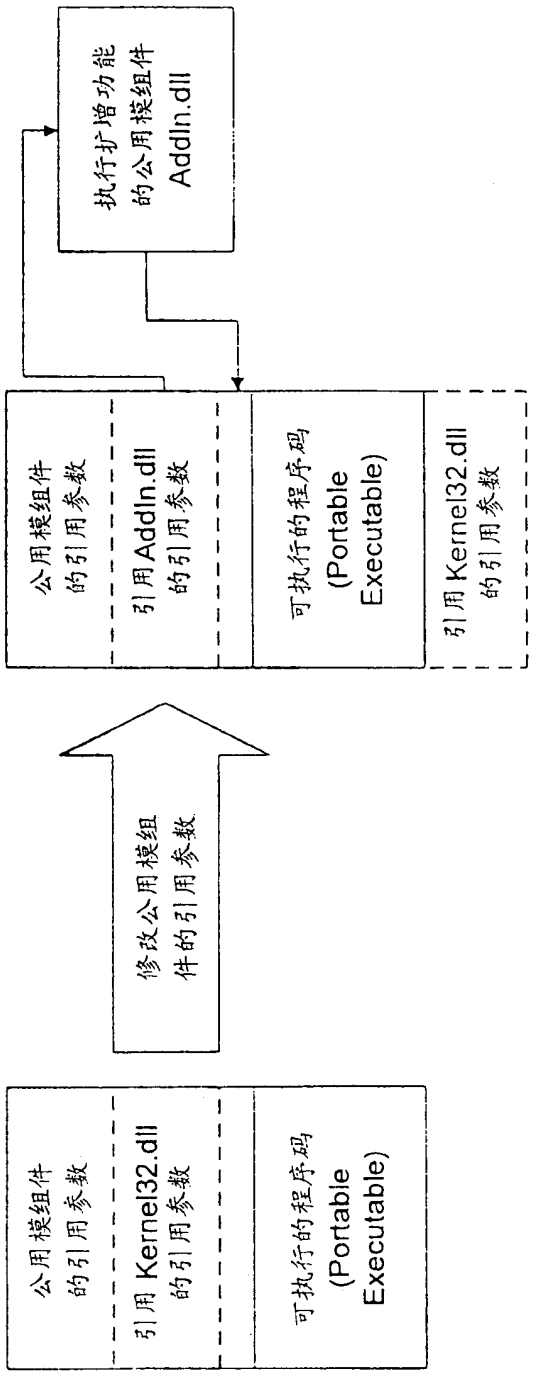

公用模组件
的引用参数
引用 Kernel32.dll
的引用参数
可执行的程序码
(Portable
Executable)
OLD.exe
修改公用模组
件的引用参数
公用模组件
的引用参数
引用 AddIn.dll
的引用参数
可执行的程序码
(Portable
Executable)
引用 Kernel32.dll
的引用参数
NEW.exe
执行扩增功能
的公用模组件
AddIn.dll

图 2

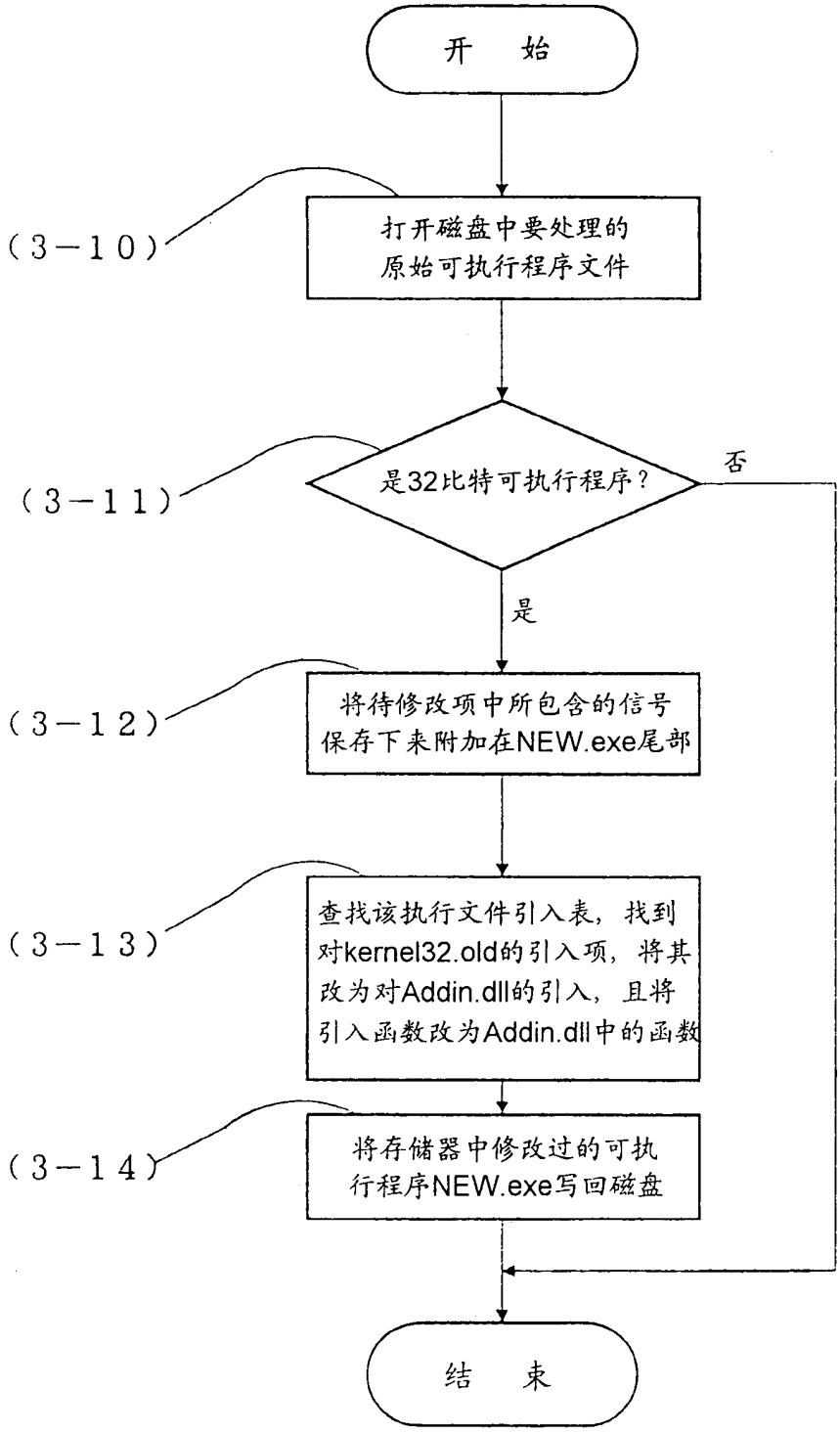

开 始
(3－10)
打开磁盘中要处理的
原始可执行程序文件
(3－11)
是32比特可执行程序?
否
是
(3－12)
将待修改项中所包含的信号
保存下来附加在NEW.exe尾部
(3－13)
查找该执行文件引入表，找到
对kernel32.old的引入项，将其
改为对Addin.dll的引入，且将
引入函数改为Addin.dll中的函数
(3－14)
将存储器中修改过的可执
行程序NEW.exe写回磁盘
结 束

图 3

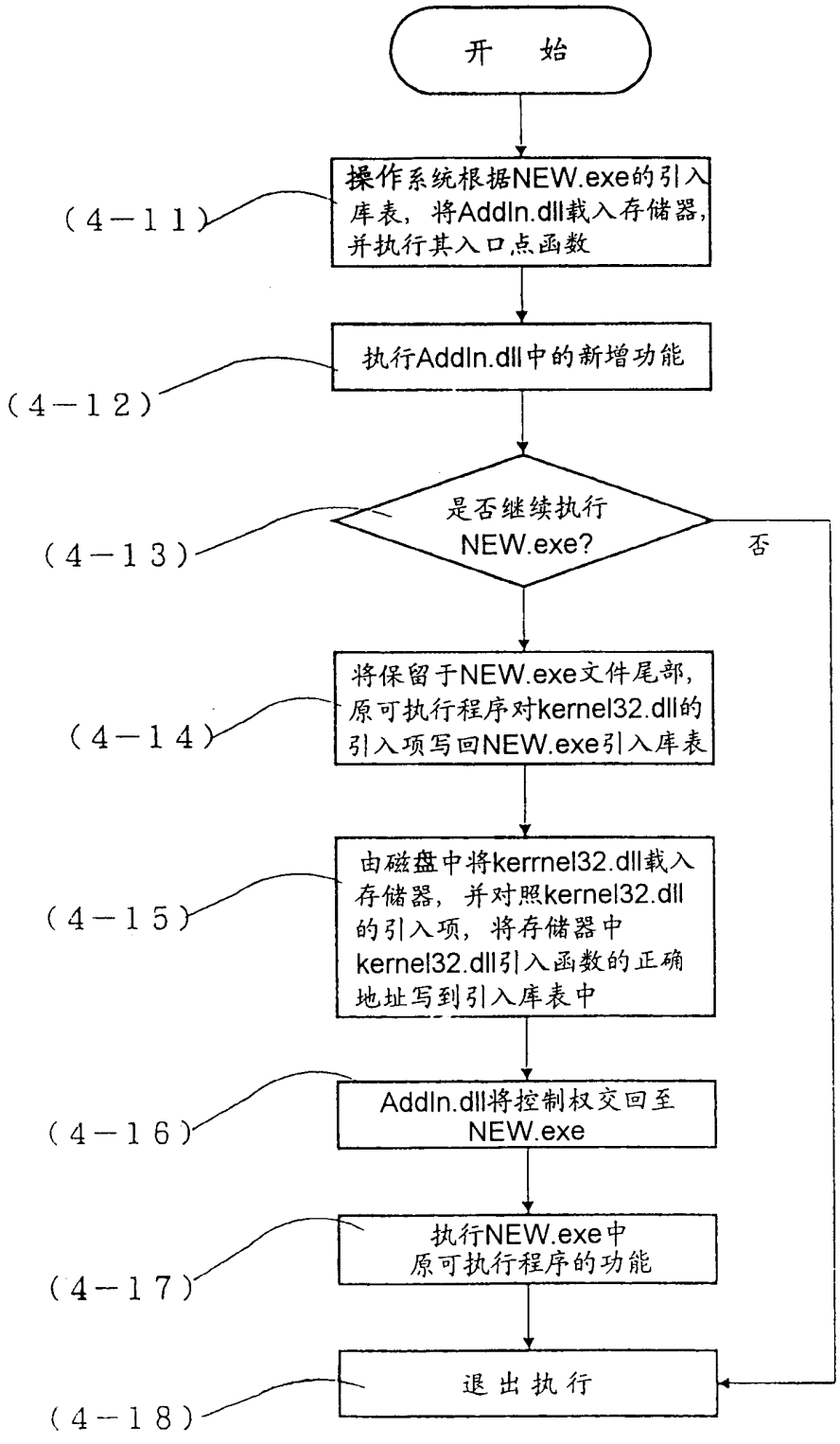

开 始
操作系统根据NEW.exe的引入库表，将AddIn.dll载入存储器，并执行其入口点函数
(4－11)
执行AddIn.dll中的新增功能
(4－12)
是否继续执行NEW.exe?
(4－13)
否
将保留于NEW.exe文件尾部，原可执行程序对kernel32.dll的引入项写回NEW.exe引入库表
(4－14)
由磁盘中将kernel32.dll载入存储器，并对照kernel32.dll的引入项，将存储器中kernel32.dll引入函数的正确地址写到引入库表中
(4－15)
AddIn.dll将控制权交回至NEW.exe
(4－16)
执行NEW.exe中原可执行程序的功能
(4－17)
退 出 执 行
(4－18)

图 4

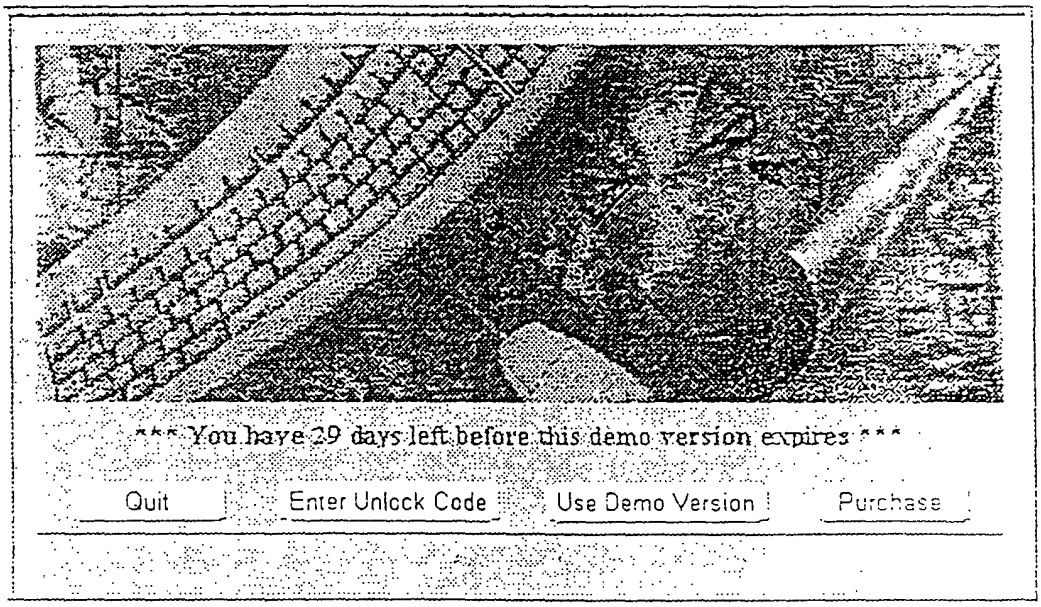

*** You have 29 days left before this demo version expires ***
Quit
Enter Unlock Code
Use Demo Version
Purchase

图 5