[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/46
G06F 9/445
G06F 17/30

[12] 发明专利申请公开说明书

[21] 申请号 200410088288.9

[43] 公开日 2005 年 5 月 18 日

[11] 公开号 CN 1617101A

[22] 申请日 2004.10.21
[21] 申请号 200410088288.9
[30] 优先权
[32] 2003.10.24 [33] US [31] 60/513,941
[32] 2004. 6.15 [33] US [31] 10/868,182
[71] 申请人 微软公司
地址 美国华盛顿州
[72] 发明人 E·J·普莱蒂斯 F·L·阿容
J·C·刘 J·J·卡瓦兰
R·费茨西蒙斯 T·D·诺南
V·V·苏里克 D·B·普罗伯特
D·C·杉伯丁 G·费南德斯
E·李 J·A·莱克托

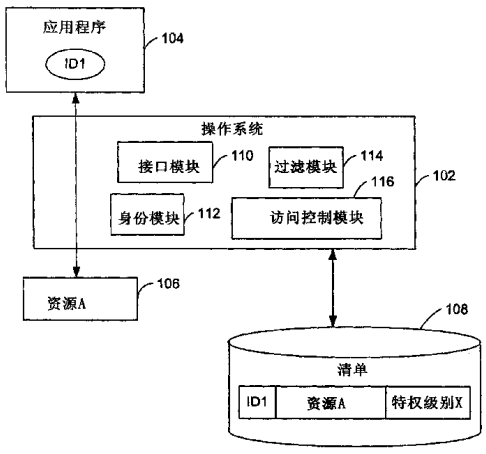
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 谢喜堂

权利要求书 3 页 说明书 22 页 附图 7 页

[54] 发明名称 操作系统资源保护

[57] 摘要

向应用程序授予对资源的访问权限，作为与该应用程序相关联的特权的函数。本发明的实施例使用与一个应用程序或一组应用程序的组件相关联的持久、个别的身份，以允许操作系统标识并区分在计算系统上安装的不同应用程序或应用程序组。与应用程序的每一组件相关联的身份启用应用程序的标识和移除或卸载。身份还启用应用程序资源的隔离和操作系统资源的保护。



ISSN1008-4274

1．一种向应用程序授予对计算系统上资源的访问权限的计算机实现的方法，其特征在于，所述计算机实现的方法包括：

接收来自所述应用程序的请求以访问在所述请求中标识的资源；

确定所述应用程序的应用程序标识符；

将来自清单的特权标识为所确定的应用程序标识符和所标识的资源的函数，所述清单指明应用程序所具有的用于访问所标识的资源的特权；以及

根据所标识的特权向所述应用程序授予对所标识的资源的访问权限。

2．如权利要求1所述的计算机实现的方法，其特征在于，标识来自清单的特权包括标识来自清单的特权，所述清单与所述应用程序相关联。

3．如权利要求1所述的计算机实现的方法，其特征在于，标识来自清单的特权包括标识来自清单的特权，所述清单与在所述计算系统上执行的操作系统相关联。

4．如权利要求1所述的计算机实现的方法，其特征在于，确定所述应用程序的应用程序标识符包括确定与多个文件和系统设置的每一个相关联的应用程序标识符，所述多个文件和系统设置表示所述应用程序。

5．如权利要求1所述的计算机实现的方法，其特征在于，向所述应用程序授予对所标识的资源的访问权限包括向所述应用程序授予对所标识的资源的只读访问权限。

6．如权利要求1所述的计算机实现的方法，其特征在于，向所述应用程序授予对所标识的资源的访问权限包括拒绝所述应用程序访问所标识的资源。

7．如权利要求1所述的计算机实现的方法，其特征在于，向所述应用程序授予对所标识的资源的访问权限包括生成由所述应用程序使用的资源副本。

8．如权利要求7所述的计算机实现的方法，其特征在于，所述应用程序将被安装在计算系统上，它还包括利用生成的所述资源副本在所述计算系统上安装所述应用程序。

9．如权利要求1所述的计算机实现的方法，其特征在于，所述应用程序将被安装在计算系统上，它还包括在所述计算系统上安装所述应用程序，包括向所述计算系统应用系统设置。

10．如权利要求 1 所述的计算机实现的方法，其特征在于，接收来自所述应用程序的请求以访问所述请求中标识的资源包括接收来自所述应用程序的请求以访问以下的一个或多个：文件、目录、命名对象以及系统设置。

11．如权利要求 1 所述的计算机实现的方法，其特征在于，它还包括在安装所述应用程序之后，基于所述应用程序的一个或多个行动生成所述清单。

12．一种从计算系统卸载特定应用程序以及关联的系统设置和对象的计算机实现的方法，所述特定应用程序具有与其关联的至少一个文件，所述特定应用程序是安装在所述计算系统上的多个应用程序中的一个，所述计算机实现的方法包括：

接收卸载所述特定应用程序的请求；

确定与所述特定应用程序相关联的标识符；

通过所确定的标识符，标识仅与所述多个应用程序中的特定应用文件相关联的文件，所标识的文件有所确定的标识符与其关联；以及

删除所标识的文件。

13．如权利要求 12 所述的计算机实现的方法，其特征在于，它还包括：

通过所确定的标识符标识响应于安装所述特定应用程序而应用的一个或多个资源变化；以及

还原所标识的资源变化。

14．如权利要求 13 所述的计算机实现的方法，其特征在于，它还包括维护所应用的资源变化的日志。

15．如权利要求 14 所述的计算机实现的方法，其特征在于，标识所述一个或多个资源变化包括标识文件类型关联，并且其中，还原所标识的资源变化包括将所述文件类型关联还原成所述日志中维护的前一关联。

16．如权利要求 12 所述的计算机实现的方法，其特征在于，标识仅与多个所述应用程序中的特定应用程序相关联的所述文件包括标识仅与所述多个应用程序中的特定应用程序相关联的一个或多个对象。

17．具有用于执行上述权利要求中任一项所述的计算机实现的方法的计算机可执行组件的一个或多个计算机可读媒质。

18．一种用于执行上述权利要求中任一项所述的计算机实现方法的系统。

19．一种在其上存储了表示指定应用程序访问多个资源的访问权限的清单的数据结构的计算机可读媒质，其特征在于，所述数据结构包括：

存储表示对应于所述应用程序的身份的值的第一字段；

存储与所述应用程序相关联的资源列表的第二字段；

存储与来自所述第一字段的身份和存储在所述第二字段中的所述资源列表相关联的特权的第三字段，所述特权定义了所述应用程序访问所述资源列表中每一资源的访问权限。

20．如权利要求19所述的计算机可读媒质，其特征在于，所述第一字段基于下列一项或多项存储值：版本、中央处理单元及公钥。

21．如权利要求19所述的计算机可读媒质，其特征在于，所述第二字段存储所述资源列表，其包括以下列的一项或多项：文件、目录、命名对象以及系统设置。

22．如权利要求19所述的计算机可读媒质，其特征在于，所述第三字段存储表示意图声明的特权。

操作系统资源保护

相关申请的参照

本申请要求编号为 60/513,941，2003 年 10 月 24 日提交的美国临时申请的优先权。与其同时提交的是题为“软件产品的应用程序身份（Application Identity for Software Products）”，委托摘要编号为 MS#307048.01(5102)的美国非临时专利申请（其同样要求编号为 60/513,941，2003 年 10 月 24 日提交的美国临时申请的优先权），其整体揭示通过引用结合于此。

技术领域

本发明的实施例涉及计算机操作系统领域，尤其涉及由操作系统管理应用程序的安装、执行以及移除。

背景技术

尽管操作系统在改善其可用性和可靠性上进展迅速，涉及应用程序的安装、关联以及移除（即，卸载）的用户体验仍然需要改善。例如，应用程序在安装期间可能错误地配置了系统设置，或重写了另一应用程序所需的文件。用户也很难卸载不合期望的应用程序，诸如广告软件(ad-ware)和间谍软件(spy-ware)。许多系统崩溃以及性能的降低（例如，缓慢的启动时间）也可归咎于应用程序问题。例如，下列情况可能导致应用程序故障，并有可能导致基本的操作系统故障：应用程序的不完全卸载、卸载应用程序时的重复删除、以及不正确地存储的文件。

在某些当前的操作系统中，新安装的应用程序会用新安装的应用程序所需的一个更旧或更新版本重写动态链接库（DLL）文件。若该更旧或更新的文件与重写的文件不兼容，则在试图访问重写文件时，依赖于重写文件的当前安装的应用程序就会崩溃。

因此，期望一种用于管理应用程序影响的改进系统和方法来解决一个或多个这些和其他缺点。

发明内容

本发明实施例包括一种允许操作系统保护其资源的方法。在一个实施例中，本发明包括使用一种与一个应用程序或一组应用程序相关联的持久、单独的身份，以允许操作系统在不同的应用程序或应用程序组和其组件之间进行标识和区分。

操作系统或其他程序通过与每一应用程序相关联的身份操纵应用程序。例如，操作系统使用身份以：（1）确保干净的卸载、（2）防止应用程序访问或执行该应用程序未被授权的服务或行动、（3）虚拟化资源，以更好地将每一应用程序彼此隔离、（4）启用应用程序影响回退（rollback）（例如，将文件类型关联还原至应用程序安装之前的状态）、以及（5）启用文件和注册表所有权跟踪。保护机制包括，但不限于，提供只读访问、记录变化以启用回退、以及对每一应用程序和每一用户虚拟化资源。例如，操作系统为请求对写保护文件的写访问的应用程序生成写保护文件的副本。

依照本发明的一个方面，一种方法向计算系统上的资源授予应用程序访问权限。该方法包括接收来自应用程序的请求，用于访问在请求中标识的资源。该方法还包括为应用程序确定应用程序标识符。该方法包括将来自清单的特权标识为确定的应用程序标识符和已识别资源的函数。该清单指明应用程序所具有的用于访问已标识资源的特权。该方法也包括依照已标识的特权向已标识资源授予应用程序根据访问权限。

依照本发明的另一方面，一个或多个计算机可读媒质具有用于准许向资源授予应用程序访问权限的计算机可执行组件。该组件包括一接口模块，用于接收来自应用程序的请求以访问请求中标识的资源。该组件还包括一身份模块，用于为应用程序确定应用程序标识符以将应用程序及其组件与其他应用程序中区别开来。该组件还包括一过滤模块，用于将来自清单的特权标识由身份模块所确定的应用程序标识符和已识别资源的函数。该清单指明应用程序所具有的用于访问已识别资源的特权。该组件还包括一访问控制模块，用以依照过滤模块标识的特权向已识别的资源授予应用程序访问权限。

根据本发明的又一方面，一种计算机可读媒质存储了表示指定应用程序访问多个资源的访问权限的清单的数据结构。该数据结构包括储存表示对应于该应用程序的身份的值的第一字段。该数据结构也包括储存与该应用程序相关联的资源列表的第二字段。该数据结构还包括存储与第一字段的身份和第二字段中储存的资源列

表相关联的特权的第三字段。该特权定义了应用程序访问资源列表中每一资源的访问权限。

[0011] 根据本发明的再一方面，一种系统向系统资源授予应用程序访问权限。该系统包括储存清单的存储区。该清单将应用程序标识符和资源映射到特权。应用程序标识符与应用程序相关联。该系统还包括一处理器，它被配置成响应于应用程序对资源的请求，执行计算机可执行指令将确定来自存储在存储区内清单的特权确定为应用程序标识符和资源的函数。该处理器还被配置成执行计算机指令，以依照确定的特权向资源授予应用程序访问权限。

依照本发明的另一方面，一种方法从计算系统卸载特定的应用程序。该特定的应用程序具有与其关联的至少一个文件。该特定的应用程序是安装在计算系统上的多个应用程序中的一个。该方法包括接收卸载特定应用程序的请求。该方法也包括确定与特定应用程序相关联的标识符。该方法还包括通过所确定的标识符，标识仅与多个应用程序中的特定的应用程序相关联的文件。该标识的文件具有与其相关联的所确定的标识符。该方法也包括删除标识的文件。

可选地，本发明可包括各种其它方法和装置。

后文将部分地指出其它特征，并通过阅读能够部分地清楚这些特征。。

附图说明

图 1 是向应用程序提供对资源的访问的操作系统的示例性实施例。

图 2 所示是一种访问控制方法的操作的示例性流程图。

图 3 所示是用于保护各种资源的缓和（mitigation）体系结构的示例性流程图。

图 4 所示是提供文件、系统设置以及扩展的访问控制的方法的操作的示例性流程图。

图 5 所示是提供系统设置的访问控制的方法的操作的示例性流程图。

图 6 所示是从计算系统上移除一个已安装的应用程序的操作的示例性流程图。

图 7 所示是可在其中实现本发明的合适的计算系统环境的一个示例的框图。

在整个附图中，对应的标号表示对应的部件。

具体实施方式

在一个实施例中，本发明提供一种保护资源的方法。具体地，操作系统的功

能性启用对文件和系统设置的保护的声明。所声明的保护是持久的，且由操作系统或其他应用程序通过一组行动来实施，操作系统在整个应用程序生命周期中能使用该组行动来管理、跟踪、预测和缓和应用程序安装、运行、服务以及移除。资源保护提供重要系统数据（例如，文件关联）的参考完整性，解决了应用程序脆弱性的问题，以通过跟踪和隔离每一应用程序对资源的访问提高可靠性和一致性，并管理系统和具有保护资源的应用程序的交互的影响。例如，本发明的实施例可用于提供安全，避免应用程序受病毒或蠕虫感染。本发明的实施例可用任一操作系统模型操作，以提供可扩充性并启用集成。资源保护策略和本发明实施例的实现也防止应用程序安装程序意外或恶意地修改或替换重要的系统资源。本发明的实施例可与用于保护系统资源的其他策略相组合。例如，计算系统可以实现包括锁定（lock down）、隔离、虚拟化、交易以及沙箱（sandboxing）的组合的策略。

首先参考图 1，操作系统 102 的示例性实施例向应用程序 104 提供经由每个清单 108 访问资源。资源包括，但不限于，文件、文件夹、进程、线程、系统设置、命名对象、应用编程接口（API）、特定代码路径、可执行例程库、操作系统属性值以及操作系统资源。命名对象包括由字母、数字、字母数字混合编制或非人类可读的（例如，全球唯一的标识符）数据标识的任一对象。例如，可由用户身份保护的任一对象可由应用程序身份（如，网络套接字）保护。例如，许多 API 和代码路径提供发送邮件的能力，而对这些 API 和代码路径的访问会受限制。在另一示例中，重启系统的能力受到限制。资源也包括系统的名字空间（如，“名字”本身），而不仅是具体的命名对象。例如，在用名字对象创建之前在名字上保留或“占据”会同时产生脆弱性和安全问题。

在一个实施例中，清单，如由应用程序（如，应用程序 104）呈现的清单 108，指明了应用程序将具有的特权。在操作中，操作系统可以授予或拒绝导致操作系统为应用程序维护的计算或有效的清单的某些请求的特权。

每一应用程序，如应用程序 104，都被分配一个应用程序标识符，以将应用程序和其他应用程序相区别。在一个实施例中，应用程序标识符被分配到一组应用程序上，以使该组应用程序中的每一应用程序能具有与该组中的其他应用程序对资源的相同的访问或特权。在图 1 中，应用程序 104 具有与其关联的标识符 ID1 。操作系统 102 截取应用程序 104 访问诸如资源 A 106 等资源的尝试。操作系统 102 参考清单 108 以确定对应用程序 104 允许的对资源 A 106 的特权或访问。在本示例

中，存储区存储清单 108。清单 108 将应用程序标识符和资源映射到特权。清单 108 将特权（如特权 X）储存为应用程序标识符（如，ID1）和资源（如，资源 A 106）的函数 。例如，特权 X 可以对应于只读访问。操作系统 102 根据确定的特权提供具有对资源 A 106 的访问的应用程序标识符。

在一个实施例中，操作系统 102 存储或访问计算机可读媒质上的一个或多个计算机可执行组件。与操作系统 102 相关联的处理器被配置成执行计算机可执行组件或其他计算机可执行指令，以响应于来自应用程序 104 的对资源的请求，将来自储存在存储器中的清单 108 的特权确定为应用程序标识符和资源的函数 。该处理器还被配置成执行计算机可执行指令，以根据确定的特权向应用程序 104 授予对资源或其副本的访问权限。

具体地，计算机可执行组件向应用程序 104 授予对资源的访问权限。在图 1 的具体实施例中，组件包括接口模块 110、身份模块 112、过滤模块 114 以及访问控制模块 116。图 1 中的模块可从操作系统 102 中分离并对其独立地存在。此外，本发明的实施例的功能和结构可以被组织成任意数量的模块、组件等等。例如，模块可以分布。

接口模块 110 接收来自应用程序 104 的请求以访问在请求中标识的资源。在一个实施例中，接口模块 110 接收来自应用程序 104 的请求以访问以下的一个或多个：文件、目录以及系统设置（例如，注册表条目）。身份模块 112 确定应用程序 104 的应用程序识别符，以将应用程序 104 及其组件与其他应用程序区别开来。在一个实施例中，身份模块 112 确定一组应用程序的应用程序识别符（例如，隔离标识符）。由于应用程序 104 可包括多个文件和系统设置，因此身份模块 112 确定与表示应用程序 104 的多个文件和系统设置相关联的应用程序标识符。过滤模块 114 将来自清单 108 的特权标识为由身份模块 112 确定的应用程序标识符和已识别资源的函数。该清单 108 指明了应用程 104 所具有的用于访问已识别资源的特权。访问控制模块 116 根据由过滤模块 114 标识的特权向应用程序 104 授予对已识别资源的访问权限。在一个实施例中，配置模块从与应用程序 104 相关联的安装媒质接收应用程序清单。该应用程序清单表示与应用程序 104 相关联的文件和资源变化（例如，系统设置）列表。配置模块可以用应用程序清单内包含的数据更新操作系统清单。可选地，配置模块能够为每一安装的应用程序维护每一应用程序清单。

清单

清单 108 包括与应用程序 104 或诸如操作系统 102 等操作系统相关联的项目（如，文件和资源变化）或对象的列表 。可选地，与应用程序 104 相关联的项目列表能存储在每一资源提供商的配置文件或存储中。在另一实施例中，对象的创建者直接在对象上指定访问特权。

清单 108 也可包括对与应用程序 104 或操作系统 102 相关联的资源的特权列表。例如，应用程序 104 的作者可在清单中指定对操作系统 102 的资源和/或对应用程序 104 可创建的资源的特权。可选地，清单可以仅存储与应用程序 104 相关联的身份信息。在另一示例中，存储要安装的应用程序 104 的安装媒质也可存储列出与应用程序 104 相关联的项目以及与应用程序专用资源相关联的特权的应用程序清单。负责应用程序 104 的展开（deployment）的第三方应用程序销售商或人员可以创建应用程序清单。在另一示例中，操作系统清单存储与用操作系统 102 安装的应用程序相关联的项目列表。该操作系统清单还可储存与操作系统 102 相关联的组件列表。在一个实施例中，操作系统清单表示对操作系统组件或已安装应用程序的每一个的保护行为的集合。该集合清单定义了对每一文件、目录以及系统设置准许的交互的类型。

操作系统 102 是自描述的，它指定了希望如何受保护以及操作系统组件以及其他组件如何交互和扩展系统。在本发明的一个实施例中，可声明应当由操作系统 102 对作为操作系统 102 的一部分的每一项目或资源（如，文件、目录、注册表项和值、驱动程序等等）实施的保护行为的类型。

清单 108 在计算机可读媒质上被储存为数据结构。清单 108 指定诸如应用程序 104 等应用程序 访问多个资源的访问权限。图 1 中的示例性数据结构包括存储表示对应应于用程序 104 的身份的值（例如，ID1）的第一字段。例如，第一字段可以存储基于以下的一个或多个的值：版本、中央处理单元以及公钥。数据结构也包括储存与应用程序 104 相关联的的资源（例如，资源 A 106）的列表 的第二字段。例如，第二字段可以存储资源列表如：文件、目录以及系统设置。数据结构也包括存储 与第一字段中的身份和存储在第二字段中的资源列表相关联的特权（如，特权 X）或其它意图声明的第三字段。该特权定义了应用程序 104 访问资源列表中每一资源的访问权限。

应用程序的作者可用可信信息部分创建诸如清单 108 的清单。应用程序作者

也可向应用程序分配强名（strong name）并签署应用程序的清单（例如，用数字签名或证书）。当安装应用程序后，操作系统 102 可被配置成核对一个或多个证书存储，以确认应用程序清单的证书和签名。在一个实施例中，仅安装已签署的驱动程序包。例如，企业可具有其自己的证书存储。类似地，特定系统也具有可对照其来确认应用程序清单的证书存储。一旦经过确认，操作系统 102 可被配置成基于清单数据和预配置的默认策略来管理可信行动。

可以若干种方式签署诸如清单 108 等清单。例如，可使用在存储中保存了用于确认的证书的验证代码（authenticode）进程来签署清单。域管理员也可为其特定的企业或域签署清单。例如，展开清单可用于指定为特定安装签署了哪一应用程序。本地管理员也可以签署清单。也可用经签署的密钥来配置每一个别的机器。

在一个实施例中，清单 108 可同时包括弱名（weak name）和强名。弱名可对应于传统的应用程序文件名，而强名可对应于文件名、版本号、文化（culture）以及公钥。在另一实施例中，强名可以是模块签名的公钥的散列。在再一实施例中，强名可以是公钥令牌。

例如，以下 XML 可代表清单 108 的一个强名。

```
<assemblyIdentity
    version="1.0.0.0"
    processorArchitecture="x86"
    name="SampleApp"
    publicKeyToken="0123456789abcdef"
    type="typeA"/>
```

以下是清单 108 的一个实施例的示例可信信息部分。

```
<trustInfo>
  <security>
    <requestedPrivileges>
      <requestedExecutionLevel
        leastPrivileged="true"
        adminPrivileged="true"
        requireDefaultDesktop="false"/>
    </requestedPrivileges>
```

</security>

</trustInfo>

对于不具有清单 108 的应用程序，操作系统 102 可被配置成根据预定的默认值生成具有请求的特权设置的清单 108。例如，清单 108 可被配置成请求用户访问的最小特权级别。

可选地或另外，操作系统也可观测应用程序的行动并定制清单以提供应用程序实际使用的唯一特权。在应用程序的多次执行之后，清单被锁定，且需要显式用户输入或管理员政策来扩展清单授予的特权。在某些实施例中，与应用程序稍后被泄密的可能性相比，易受攻击的应用程序在安装之后很快被泄密的可能性相对较低。若应用程序在清单被锁定之后被泄密，则被泄密的应用程序的行为会被限制在清单允许的行为上，该行为由应用程序未被泄密的行为确定。

接下来将描述利用应用程序身份来保护资源的方法。

提供访问控制

接下来参看图 2，示例性流程图示出了访问控制方法的操作。在一个实施例中，本发明向应用程序授予对计算系统上的资源的访问权限。该方法包括在 202 接收来自应用程序的请求以访问在请求中标识的资源、在 204 确定应用程序的应用程序标识符、在 206 将来自清单（例如，操作系统清单和/或应用程序清单）的特权标识为已确定的应用程序标识符和已识别资源的函数、以及在 208 根据所识别的特权向应用程序授予对已识别资源的访问权限。在一个实施例中，在 204 确定应用程序的应用程序标识符包括用唯一、一致、持久、可重复的标识符标记每一文件、文件夹、系统设置变化（例如，注册表键和数值）或资源。

在一个实施例中，操作系统执行图 2 所示的方法。在另一实施例中，与操作系统分离的应用程序或服务执行图 2 所示的方法。一个或多个计算机可读媒质具有用于执行图 2 所示的方法的计算机可执行指令。

接下来将参考用于保护资源的示例缓和（mitigation）体系结构来描述各种示例性特权或其他形式的访问。

示例性缓和体系结构

接着参看图 3，示例性流程图示出了用于保护各种资源的缓和体系结构。在一

个实施例中，图 3 所示的方法基于试图访问资源的应用程序的应用程序标识符实施清单中所描述的资源特权。尽管本发明描述了的某些特权，但是本发明未描述的各种特权、特权级别或访问也落入本发明的范畴之内。同样，尽管本发明描述了的某些资源，但是本发明未描述的各种资源落入本发明的范畴之内。

图 3 中，应用程序，诸如具有识别符 ID1 的应用程序，请求访问各种资源。本发明的一个实施例接收请求并根据为应用程序指定的对资源的特权或访问的处理该请求。在图 3 的示例中，应用程序能以只读特权访问某些操作系统资源（例如，文件和设置）。若应用程序向这些资源之一发送读访问请求，则本发明的一个实施例向应用程序授予对一个资源的只读访问权限。若在 302 应用程序 发送修改这些只读资源之一的请求，则本发明的一个实施例拒绝该应用程序对一个资源的访问。在 304，该请求悄然（例如，没有向应用程序返回响应）或明确地（例如，向应用程序返回否定响应）失败。该应用程序也能以读写特权访问应用程序专用资源（例如，与应用程序相关联的那些文件和设置）。由于这些资源与应用程序相关联，因此这些资源的修改通常不引发到操作系统脆弱性问题。操作系统对这些资源的语义知之甚少或毫无兴趣。

应用程序能以受保护特权访问其他操作系统资源。若在 306 应用程序资源发送修改这些受保护操作系统资源（例如，设置或文件）之一的请求，则在 308，本发明的一个实施例 为应用程序返回受保护资源的虚拟视图。具体地，就受保护特权而言，本发明的一个实施例为应用程序读写访问生成一个请求资源的副本（如果不存在的话）。在副本尚未存在的一个实施例中，若来自应用程序的请求仅用于只读访问，则不生成副本。资源的副本仅由具有相同应用程序标识符的应用程序或应用程序组使用。应用程序标识符允许本发明的一个实施例向具有不同应用程序标识符的应用程序提供他们自己的虚拟视图或一个或多个资源的副本。例如，操作系统维护系统设置的其自己的副本，而向系统设置写入值的应用程序接收系统设置的其自己的副本。在某些示例性实施例中，不同的应用程序可接收不同的系统设置虚拟视图（例如，注册表条目）。根据（例如，用户）所期望的系统保护的类型，可对每一用户和/或每一应用程序虚拟化资源。具有特定应用程序标识符的应用程序对虚拟化的资源的改变对具有其他应用程序标识符的应用程序没有影响（如，不可见）。通过向个别的应用程序或应用程序组提供所选的系统资源的其自己的视图，操作系统可避免一个应用程序重写或破坏其他应用程序所需的资源。

在一个实施例中，在计算系统上安装应用程序的过程中，应用程序使用资源的虚拟化副本。例如，应用程序可以利用生成的存储系统设置的文件副本向计算系统应用系统设置。

应用程序能够访问应用程序专用资源。应用程序专用资源包括对应用程序专用的资源。操作系统和其他应用程序一般不受应用程序专用资源的影响。若在 301，应用程序发送修改应用程序专用资源的请求，则在 312，本发明的一个实施例 允许并处理该请求。

在 314，应用程序可以 发送改变系统可扩充性（例如，向操作系统添加功能）的请求。在一个实施例中，在 312，本发明的一个实施例允许 请求的改变。

在 312，对系统可扩充性和应用程序专用资源（例如，文件和系统设置）的变化在 318 将被记入日志或被记录。通常，系统可扩充性变化向操作系统提供额外的功能，而不修改操作系统。记录系统可扩充性变化以及对应用程序专用资源的变化启用变化回退以及与该变化相关联的应用程序的完全移除或卸载。

示例性缓和策略

接下来参看图 4，示例性流程图示出了为文件、系统设置以及扩展提供访问控制的方法的操作。在图 4 的示例中，操作系统实现该方法。尽管如此，不与该操作系统相关联的应用程序或服务也可以实现该方法。在图 4 中，创建进程以通过诸如 CreateProcess()的函数执行应用程序（例如，xxxx.exe）。在 402，操作系统 确定是否有与应用程序相关联的应用程序标识符。如果没有，则在 404，操作系统 确定应用程序标识符并持久保存该信息（例如，在清单中存储确定的应用程序标识符）。在 406，应用程序 执行并实行该操作。操作系统分析该操作。例如，在一个实施例中，在受保护区域内仅可添加、修改或删除以特殊特权执行的授权的可信安装进程。在受保护区域中，阻止应用程序创建或修改数据。

在图 4 的实施例中，若在 408 ，操作是文件操作，则在 410，操作系统 确定文件操作是否对文件有影响（例如，文件操作修改了该文件）。若文件操作对文件没有影响，则在 414，操作系统 允许在文件系统上执行文件操作。若文件操作对文件有影响，则在 412，操作系统 根据诸如图 3 所示的缓和策略执行缓和文件操作。

若在 416，操作是系统设置，则在 418，操作系统 确定系统设置操作是否对系统设置有影响（例如，系统设置操作修改了该系统设置）。若系统设置操作对系统

设置没有影响，则在422，操作系统允许在系统设置上执行系统设置操作。若系统设置操作对系统设置有影响，则在420，操作系统根据诸如图3 所示的缓和策略执行缓和系统设置操作。在415，将对系统设置的变化（如果有的话）记入日志。

若在424，该操作表示向操作系统加载扩展的请求，则在426，操作系统确定应用程序（如，xxxx.exe）是否需要保护（例如，启用“撤销”）。例如，应用程序会明确向操作系统通知保护期望。若应用程序不希望被保护，则在428，操作系统允许加载扩展。若应用程序指示期望保护，则在430，操作系统确定该扩展是否为外来扩展（例如，由第三方提供）。若该扩展不是外来的，则在428，操作系统允许加载扩展。若该扩展是外来的，则在432，操作系统根据诸如图3 所示的缓和策略 执行缓和的扩展加载。扩展加载将被记入日志。例如，记录可由执行操作系统的计算系统用户配置。

使用虚拟化，应用程序在其局部名字空间创建并修改对象，而操作系统在全局名字空间创建并修改对象。有一个全局名字空间，并可能有多个局部名字空间。对于创建操作，应用程序在其局部名字空间中创建对象。当应用程序试图修改对象时，操作系统检查对象是否驻留在应用程序的局部名字空间内。若局部对象存在，则应用程序在其局部名字空间内打开对象。若应用程序试图修改全局名字空间内的对象，则操作系统将该对象复制到应用程序的局部名字空间内，并允许在该局部名字空间上出现操作。若资源不存在于局部或全局名字空间内，则打开操作失败。

接下来参看图5，示例性流程图示出了为系统设置提供访问控制方法的操作。即使图5 示出了与系统设置有关的示例，但是本发明虚拟化的方面可用于其他对象（例如，命名对象）以及名字空间。例如，在图5中，诸如操作系统等本发明的一个实施例分析应用程序请求系统设置上的操作。具体地，在502，操作系统确定请求的操作是否会写或删除系统设置。若请求的操作不会写或删除系统设置（例如，请求了只读访问），则在 504，操作系统确定当前是否存在系统设置的虚拟副本。若虚拟副本存在，则操作系统在506标识虚拟副本，并在508在系统设置的虚拟副本上执行所请求的操作。若虚拟副本不存在，则在508，操作系统 在系统设置上执行所请求的操作。

若请求操作会写或删除系统设置，则在 510，操作系统 确定请求应用程序是否与只读密钥相关联（例如，请求应用程序不是可信安装程序）。若请求应用程序与只读访问（例如，通过由操作系统维护的访问控制列表）相关联，则在512，操

作系统将 使请求的操作失败或拒绝请求的操作。若请求应用程序不与只读访问相关联，则在 514，操作系统确定请求的操作是否会写或删除系统限制的设置。若请求的操作会写或删除系统限制的设置，则在 516，操作系统确定是否批准请求应用程序执行操作。例如，操作系统将确定请求应用程序在计算系统上是否具有管理员特权。若批准请求应用程序执行操作，则在 508，操作系统将 执行请求的操作。若不批准请求的应用程序执行操作，则在 512，操作系统将 使请求的操作失败或拒绝请求的操作。

若请求的操作不会写或删除系统限制的设置，则在 516，操作系统确定请求的操作是否用于受保护的设置（例如，与请求应用程序相关联的系统设置的副本）。若操作系统确定请求的操作用于受保护的设置，则在 520，操作系统 按照请求应用程序的应用程序标识符来虚拟化受保护设置。即，操作系统标识系统设置的虚拟副本，并在 508 在系统设置的已标识、虚拟副本上执行请求的操作。若操作系统确定请求的操作不用于受保护的设置，则在 522，操作系统 确定请求的操作是否用于专用设置（例如，与请求应用程序相关联的系统设置）。若操作系统确定请求的操作用于专用设置，则在 508，操作系统 在专用设置上执行请求的操作。若操作系统确定请求的操作不用于专用设置，则操作系统结束处理并悄然地或明确地使请求失败。

当应用程序试图从局部名字空间删除一个对象，且存在有同样名字的全局对象时，系统将局部对象标记为被删除，但在名字空间内保留该对象。因此，系统能够检测到对该对象的应用程序的查询不应当见到该对象的名字。当应用程序试图删除在局部名字空间存在但在全局名字空间不存在的对象时，系统删除该局部对象。根据操作系统配置，删除全局对象可导致删除所有对应的局部对象。同样，添加全局对象可能导致删除被标记为从所有局部名字空间内删除的所有对应的对象。

采用这一设计，应用程序认为它在全局名字空间内运作，但事实上，它在其自己的名字空间内运作。系统处理全路径查询、枚举以及其他操作，以使应用程序认为它在全局名字空间内运作。例如，名字空间枚举包括列出特定目录下的所有文件。系统查询特定名字空间内（例如，首先从局部名字空间开始，然后是全局名字空间）的所有对象。系统忽略具有在局部名字空间内找到的全局名字空间枚举的重复对象。枚举同样忽略被标记为从局部名字空间内删除的对象及其对应的全局名字空间对象。

对于希望共享资源的应用程序而言，操作系统会将那些应用程序放置在同一个虚拟化应用程序组内（例如，同样的隔离身份）。可选地，操作系统可指定名字空间的特定部分不应当被虚拟化。在另一替换方案中，应用程序指定其他应源程序能够访问的其虚拟化的名字空间的一部分。当客户机应用程序访问共享的虚拟化名字空间时，操作系统搜索目标应用程序的对应的导出名字空间。

在某些环境中，操作系统希望具有多个虚拟化层 。每一用户可有一个虚拟化层，且每一应用程序组有一个虚拟化层。多个虚拟化层的各种排序处于本发明的范畴之内。在本示例中，用户虚拟化层优先于应用程序虚拟化层。因此，对于对象的查询请求和打开请求首先检查当前用户的虚拟化层，然后检查当前应用程序组的虚拟化层，最后检查全局名字空间。系统返回找到的第一个对象，或者若在任一虚拟化层或全局名字空间内不存在对象，则不返回对象。同样，对于写操作，操作系统首先打开对象。若对象在最高优先层内存在，则在该对象上出现写操作。若对象在最高优先层内不存在，则对象被复制到最高优先层内，并在该复制的对象上出现写操作。创建操作在最高优先层中出现，尽管某些实施例中的操作系统允许代码将特定虚拟化层指定为优选。类似地，当删除对象时，在最高优先虚拟化层内出现操作，尽管某些实施例中的操作系统允许代码将特定虚拟化层指定为优选。一旦找到了确切的对象，操作系统检查对象是否存在于任何可用的更低优先级名字空间内。若对象存在于更低优先级的名字空间内，则预期的删除的对象被标记为“删除”，并保留在其名字空间内。若对象不存在于更低优先级的名字空间内，则删除该对象，并从更高优先级的名字空间内移除。在某些配置中，操作系统可从更高优先级的名字空间内删除对应的对象。尽管如此，更高优先级的对象的创建者可以指定在该情况下不删除对象。

当向更低优先级的名字空间添加对象时，操作系统移除被标记为从更高优先级删除的所有的对应对象。搜索和移除从目标名字空间开始向上到下一可用的更高优先层，直至搜索找到没有被标记为删除的对应对象，或搜索完所有可用层。

枚举操作考虑上下文和全局名字空间的所有可用虚拟化层。该枚举从最高优先级的可用名字空间开始，并下移至全局名字空间。当枚举遇到被标记为删除的对象时，在更低优先级的名字空间中忽略对该对象的枚举。

操作系统的内部对象保护

操作系统创建各种对象。某些对象预期供应用程序访问，而其它对象（如，内部对象）仅仅可由操作系统组件访问。。操作系统定义对这些对象的访问权限（例如，打开以及读访问）。

在一个实施例中，内部操作系统对象应当仅能由内部操作系统组件访问。为防止外部代码访问内部对象，操作系统标记内部对象，以仅由内部操作系统组件访问。作为内部操作系统代码运行的运行时对象与内部操作系统身份相关联。因此，当运行时对象尝试访问内部对象时，操作系统检查该运行时对象是否与内部操作系统身份相关联。若运行时对象具有内部操作系统身份，则操作系统允许访问。否则，操作系统执行适当的行动。适当的行动可包括拒绝访问、将访问尝试记入日志等等。

当内部操作系统组件创建对象时，标记该对象以仅由内部操作系统组件访问，除非该创建者专门将对象标记为对外部访问可用。操作系统可以利用来自存储、清单、配置文件、数字签名等资源信息离线标记内部对象。

某些操作系统组件被归类为中间件（middleware）组件，这意味着即使它们是操作系统的一部分，他们也不应当访问内部对象，除了也允许外部应用程序访问的某些特殊的例外之外。一个实施例中的操作系统期望中间件组件停止使用特殊例外的内部对象，并迁移到外部对象。为解决该问题，操作系统将中间件应用程序身份与中间件组件相关联。用反对属性来另外标记特殊例外的内部对象。当中间件组件访问该反对对象时，系统用诸如审核访问和/或阻止访问等适当的行动来响应。中间件反对资源检测可更一般地用于反对外部对象或其他外部对象或其他内部对象。

应用程序的移除

接下来参看图 6，示例性流程图示出了用于利用应用程序身份执行应用程序撤销的方法。具体地，该流程图示出了通过与应用程序及其组件（例如，文件和资源）相关联的应用程序标识符从计算系统上完全移除已安装的应用程序的方法。尽管现有应用程序当前可支持卸载功能，但是当前的操作系统缺少一种确保与特定应用程序相关联的组件在卸载过程中被移除的机制。本发明的一个实施例维护跟踪哪些文件与特定应用程序相关联的数据存储（例如，日志）。因此，当卸载应用程序时，操作系统标识并删除卸载过程遗留的任何文件（例如，包括在应用程序名字空间内已经虚拟化的那些文件）。这通过卸载应用程序的所有元素，包括通过扩展或修改操作系统的行为或在其他（例如，更低）级别的虚拟化中创建的那些元素，，而提

供了应用程序的完全卸载。本方法也有助于在从计算系统上卸载之后应用程序移除间谍软件、广告软件或常伴随已安装应用程序的其它不需要的应用程序。

在一个实施例中，将要移除的特定应用程序是安装在计算系统上的多个应用程序中的一个。在 602，本发明的一个实施例 接收卸载特定应用程序的请求。例如，该请求可起源于计算系统的用户。可选地，可在安装操作系统的当前版本之前由卸载应用程序的先前版本的升级实用程序生成请求。在 604，本发明的一个实施例 确定与特定应用程序相关联的标识符。例如，该标识符可以是应用程序的一部分或单独存储在存储区内。在 606，本发明的一个实施例通过确定的标识符 标识仅与特定应用程序相关联的一个或多个文件。即，标识的文件不与安装在计算系统上的其他任何应用程序相关联。由于系统上的每一程序有与其关联的至少一个应用程序标识符，因此仅与特定应用程序相关联的文件的标识可从执行对确定的标识符的搜索来获得。。在 608，本发明的实施例 删除标识的文件。在一个实施例中，本发明避免删除任何用户文件（例如，文字处理文档、电子表格文档）。

另外，在 610，标识响应于安装特定应用程序而应用的系统设置或资源变化，并在 612 还原。例如，在应用程序安装过程中，应用到计算系统的任何系统设置都由本发明的一个实施例记入日志并维护。标记应用程序作出的对文件和系统的改变，用于所有权跟踪。日志将每一个变化与已安装应用程序的应用程序标识符相关联。在一个实施例中，维护该日志以允许一个或多个变化的回退。例如，用户希望撤消对系统作出的最近的改变。在另一实施例中，操作系统通过回退与特定应用程序相关联的变化来执行特定应用程序的完全卸载。在应用程序的移除或卸载过程中，本发明的实施例利用确定的标识符来标识并还原，或移除与要卸载的应用程序的应用程序标识符相关联的已应用的设置或变化。例如，可将对文件类型关联的改变记入日志，以使卸载应用程序不会留下没有关联的应用程序的特定文件类型。即，如果应用程序安装期间作出了文件类型关联，则当卸载应用程序时还原文件类型关联。

示例性操作环境

图 7 以计算机 130 的形式示出了通用计算装置的一个示例。在本发明的一个实施例中，诸如计算机 130 等计算机或其他计算系统适用于本发明所示出并描述的其他附图。计算机 130 具有一个或多个处理器或处理单元 132，以及系统存储器

134。在所示的实施例中，系统总线 136 将包括系统存储器 134 的各种系统组件 耦合至处理器 132。总线 136 代表任一多种总线结构类型中的一种或多种，包括存储器总线或存储控制器、外围总线、加速图形端口以及使用各种总线体系结构的任一种的处理器或局部总线。作为示例而非局限，这类体系结构包括工业标准体系结构（ISA）总线、视频电子技术标准协会（VESA）局部总线以及 外围部件互连（PCI）总线，又称为 Mezzanine 总线。

计算机 130 通常至少具有某一形式的计算机可读媒质。计算机可读媒质可以是计算机 130 能够访问的任何可用媒质，包括易失和非易失的媒质、可移动和不可移动的媒质。作为示例而非局限，计算机可读媒质包括计算机存储媒质以及通信媒质。计算机存储媒质包括以存储诸如计算机可读指令、数据结构、程序模块或其他数据等信息的任何方法或技术实现的易失和非易失、可移动和不可移动媒质。例如，计算机可读媒质包括 RAM、ROM、EEPROM、闪存或其他存储器技术、CD-ROM、数字多功能盘（DVD）或其他光盘存储、磁盒、磁带、磁盘存储器或其他磁存储器设备、或可用于存储所期望的信息并可由计算机 130 访问的任一其它媒质。通信媒质通常在诸如载波或其他传输机制等已调制数据信号中包含计算机可读指令、数据结构、程序模块或其他数据，，且包括任一信息传送媒质。本领域的技术人员熟悉已调制数据信号，它以对信号中的信息进行编码的方式设置或改变其一个或多个特性。诸如有线网络或直线连接等有线媒质、以及诸如声学、RF、红外以及其他无线媒质等无线媒质都是通信媒质的示例。以上任一组合也包括在计算机可读媒质的范畴之内。

系统存储器 134 包括可移动和/或不可移动，易失和/或非易失存储器形式的计算机存储媒质。在所示的实施例中，系统存储器 134 包括只读存储器（ROM）138 和随机存取存储器(RAM)140。基本输入/输出系统 142(BIOS)通常存储在 ROM138 中，它包含如在启动期间有助于在计算机 130 内的元件之间传输信息的基本例程。RAM 140 通常包含处理单元 132 立即访问和/或当前正在运行的数据和/或程序模块。作为示例而非局限，图 7 示出了操作系统 144、应用程序 146、其他程序模块 148 以及程序数据 150。

计算机 130 也可包括其他可移动/不可移动，易失/非易失计算机存储媒质。例如，图 7 示出了读取或写入不可移动、非易失磁性媒质的硬盘驱动器 154。图 7 也示出了读取或写入可移动、非易失磁盘 158 的磁盘驱动器 156、以及读取或写入

可移动、非易失光盘 162，如 CD-ROM 或其他光媒质的光盘驱动器 160。可以在示例性操作环境中使用的其它可移动/不可移动、易失/非易失计算机存储媒质包括，但不限于，磁带盒、闪存卡、数字多功能盘、数字录像带、固态 ROM、固态 ROM 等等。硬盘驱动器 154、磁盘驱动器 156 和光盘驱动器 160 常通过非易失存储器接口，如接口 166 连接到系统总线 136。

以上讨论并在图 7 中示出的驱动器或其他大容量存储设备及其关联的计算机存储媒质为计算机 130 提供了计算机可读指令、数据结构、程序模块和其他数据的存储。例如，图 7 中，示出硬盘驱动器 154 存储操作系统 170、应用程序 172、其他程序模块 174 以及程序数据 176。注意，这些组件可以与操作系统 144、应用程序 146、其他程序模块 148 以及程序数据 150 相同或者不同。此处对操作系统 170、应用程序 172、其他程序模块 174 以及程序数据 176 给予不同的标号以说明至少他们是不同的副本。

用户可通过输入设备或用户接口选择设备，如键盘 180 和定位设备 182（例如，鼠标、跟踪球、输入笔或触摸垫）向计算机 130 输入命令以及信息。其他输入设备（未示出）可包括麦克风、操纵杆、游戏垫、圆盘式卫星电视天线、扫描仪等等。这些和其他输入设备通过耦合至系统总线 136 的用户输入接口 184 接连至处理单元 132，但也可通过其他接口和总线结构连接，如并行端口、游戏端口或通用串行总线（USB）。监视器 188 或其他类型的显示设备也通过接口，诸如视频接口 190 连接至系统总线 136。除监视器 188 之外，计算机通常包括可通过输出外围接口（未示出）连接的其他外围输出设备（未示出），如打印机和扬声器。

计算机 130 可以在使用到一个或多个远程计算机，诸如远程计算机 194 的逻辑连接的网络化环境中操作。远程计算机 194 可以是个人计算机，服务器、路由器、网络 PC、对等设备或其他公用网络节点，且通常包括多个或所有上述与计算机 130 相关的元件。图 7 所示的逻辑连接包括局域网（LAN）196 以及广域网（WAN）198，但也可包括其他网络。LAN 136 和/或 WAN 138 可以是有线网络、无线网络、它们的组合等等。这类网络环境常见于办公室、企业范围计算机网络、内联网以及全球计算机网络（例如，因特网）。

当在局域网环境中使用时，计算机 130 通过网络接口或适配器 186 连接至 LAN 196。当在广域网环境中使用时，计算机 130 通常包括调制解调器 178 或其他装置，用于通过 WAN198，如因特网建立通信。调制解调器 178 可以是内置的，也可以

是外置的，它通过用户输入接口 184 或其他适当的机制连接至系统总线 136。在网络化环境中，描述的有关计算机 130 的程序模块或其部分可以储存在远程储存器存储设备（未示出）中。作为示例而非局限，图 7 示出了远程应用程序 192 驻留在存储器设备上。所示的网络连接是示例性的，且可以使用在计算机之间建立通信链路的其他装置。

一般地，计算机 130 的数据处理器通过在不同时刻储存在计算机的各种计算机可读存储媒质上指令来编程。例如，程序和操作系统通常分布在软驱或 CD-ROM 上。他们可以从那里安装或加载到计算机的次级存储器上。在执行时，他们被至少部分地加载到计算机初级电子存储器中。当此类媒质包含用于下文结合微处理器或其它数据处理器描述的步骤的指令或程序时，此处描述的本发明包括这些和其他各种类型的计算机可读媒质。当根据此处描述的方法和技术来编程时，本发明也包括计算机本身。

为说明目的，此处示出程序和其他可执行程序组件，如操作系统，为离散块。然而，应认识到，这类程序和组件在不同时刻驻留在计算机的不同存储组件中，并由计算机数据处理器执行。

尽管结合包括计算机 130 在内的示例性计算系统环境来描述，本发明可用许多其他通用或专用计算系统环境或配置来操作。计算系统环境并非建议对本发明的使用或功能的范围的限制。此外，计算系统环境不应被认为对示例性操作环境中示出的任一组件或其组合具有任何依赖和需求。适合与使用本发明的众所周知的计算系统、环境、和/或配置的示例包括，但不限于，个人计算机、服务器计算机、手提式或膝上型设备、多处理器系统、基于微处理器的系统、机顶盒、可编程消费者电子设备、移动电话、网络 PC、小型机、大型机、包括上述系统或设备的分布式计算环境等等。

本发明可在诸如由一个或多个计算机或其他设备执行的程序模块等计算机可执行指令的一般上下文中描述。一般地，程序模块包括但不限于，例程、程序、对象、组件和数据结构，执行特定任务或实现特定的抽象数据类型。本发明也可在分布式计算环境中执行，其中，任务由通过通信网络链接的远程处理设备来执行。在分布式计算环境中，程序模块可位于本地和远程计算机存储媒质中，包括存储器存储设备。

软件体系结构上下文中的接口包括软件模块、组件、代码部分或计算机可执

行指令的其它序列。例如，接口包括，访问第二模块以代表第一模块执行计算任务的第一模块。在一个示例中，第一以及第二模块包括诸如由操作系统提供的应用编程接口（API）、组件对象模型(COM)接口（例如，用于对等应用通信）以及可充标记语言元数据互换格式（XMI）接口（例如，用于在网络服务之间通信）。

接口可以是一个紧密耦合的、同步实现，如在 Java 2 平台企业版本（J2EE）、COM、或分布式 COM（DCOM）示例中的实现。可选地或另外，接口可以是松散耦合的、非同步实现，如在 web 服务中（例如，利用简单对象访问协议）的实现。一般而言，接口可包括下列特性的任一组合：紧密耦合、松散耦合、同步以及非同步。此外，接口可符合标准协议，专有协议或标准和专有协议的任一组合。

此处描述的接口都可以是单个接口的部分，或被实现为单独的接口或其中的任一组合。该接口本地或远程执行以提供功能。此外，该接口可比此处示出或描述的功能更多或更少的功能。

在操作中，计算机 130 执行诸如在附图中示出的计算机可执行指令，以根据与应用程序和资源相关联的特权向应用程序授予对资源的访问。附图所示并在此处描述的系统和方法可使用本领域中已知的技术以软件或硬件或其两者来实现。

清单示例

以下示例进一步说明了本发明。尽管以下的某些示例包括注册表的引用，但是本发明的实施例并不限于注册表。本发明的实施例可用用于储存系统设置的任一机制来操作。以某些机制来继承属性，然而继承不以其他机制作保证。以下的表 1 列出了清单中的示例性特权，并描述了与每一级别相关联的资源保护类型。

特权	保护类型
readOnlyIgnoreWrites	只读—与本特权相关联的文件或设置仅能由操作系统在安装或服务（例如，升级）时修改。其他写入本文件或设置的尝试被未作记述地忽略（例如，即使没有写入发生，仍返回成功响应）。
readOnlyFailWrites	只读—与本特权相关联的文件或设置仅能由操作系统在安装或服务（例如，升

	级）时修改。其他写入本文件或设置的尝试被明确忽略(例如，返回失败响应)。
OSOnlyIgnoreWrites	与本特权相关联的文件或设置仅能由操作系统组件修改。写入本文件或设置的其它尝试被未作记述地忽略（例如，即使没有写入发生，仍返回成功响应）。
OSOnlyFailWrites	与本特权相关联的文件或设置仅能由操作系统组件修改。写入本文件或设置的其它尝试被明确忽略（例如，返回失败响应）。
Change recording	为与本特权相关联的设置储存的不同值将在每一应用程序基准上保留，但在全局基准上可见（写的最后一个应用程序）。若全局值属于被卸载的应用程序，则使用最近的应用程序算法回退当前全局值。
ApplicationVirtualized	对与本特权相关联的文件或设置的变化响应于来自应用程序写请求，对每一应用程序虚拟化。
UserVirtualized	对与本特权相关联的文件或设置的变化响应于来自用户的写请求，对每一个用户虚拟化。
applicationAndUserVirtualized	对与本特权相关联的文件或设置的变化响应于来自执行应用程序的用户的写请求，对每一用户和每一应用程序虚拟化。
notProtected	与本特权相关联的文件或设置没有与其关联的保护或缓和。任何具有适当许可的第三方应用程序或管理员可以修改这些文件和设置。

表 1－示例性特权

在另一个示例中，示例操作系统组件（例如，“Comp Name”）期望对与该组件相关联的资源的以下保护行为。

目录名	保护行为
C:\Comp Name\	基于身份的访问特权
C:\Comp Name\Sub\	应用程序虚拟
C:\Common Files\Shared\Comp Name\	基于身份的访问特权（使写入失败）

表 2－示例目录以及期望的保护行为

目录名	文件名	保护行为
C:\	CompName.dll	基于身份的访问特权
C:\Comp Name\	Sample.sys	基于身份的访问特权
C:\Comp Name\Sub\	CompName.dat	应用程序虚拟
C:\Common Files\Shared\Comp Name\	Common.dll	基于身份的访问特权（使写入失败）
C:\Common Files\Shared\	Base.dll	应用程序虚拟

表 3－示例文件以及期望的保护行为

键名	保护行为
HKLM\Software\Comp Name\	基于身份的访问特权
HKLM\Software\Comp Name\SubKey\	基于身份的访问特权
HKLM\Software\Comp Name\Setting\	应用程序虚拟
HKCR\.comp\	基于身份的访问特权

表 4－示例注册表键以及期望的保护行为

键名	值名	保护行为
HKLM\Software\Comp Name\	Version	基于身份的访问特权
HKLM\Software\Comp Name\SubKey\	SubVulue	基于身份的访问特权
HKLM\Software\Comp Name\Setting\	（默认）	应用程序虚拟
HKCR\.comp\	MyEnv	基于身份的访问特权

表5－示例注册表值以及期望的保护行为

此处示出并描述的方法的执行或实行的顺序并非必要的，除非另外指明。即，该方法的元素可以任何顺序执行，除非另外指明，且该方法可包括比此处所揭示的方法更多或更少的元素。

当介绍本发明的元素或其实施例时，冠词“一”、“一个”、“该”以及“所述”意味着有一个或多个元素。术语“包括”、“包含”以及“具有”是包括性的，并意味着可以有除列出的元素之外有其他元素。

鉴于以上内容，可以看到，实现了本发明的若干对象，并获得了其他的有利结果。

由于可以不脱离本发明的范围的情况下在以上结构、产品以及方法中作出各种变化，因此包含在以上描述中并在附图中示出的所有内容都应当被认为是说明性的而非限制的。

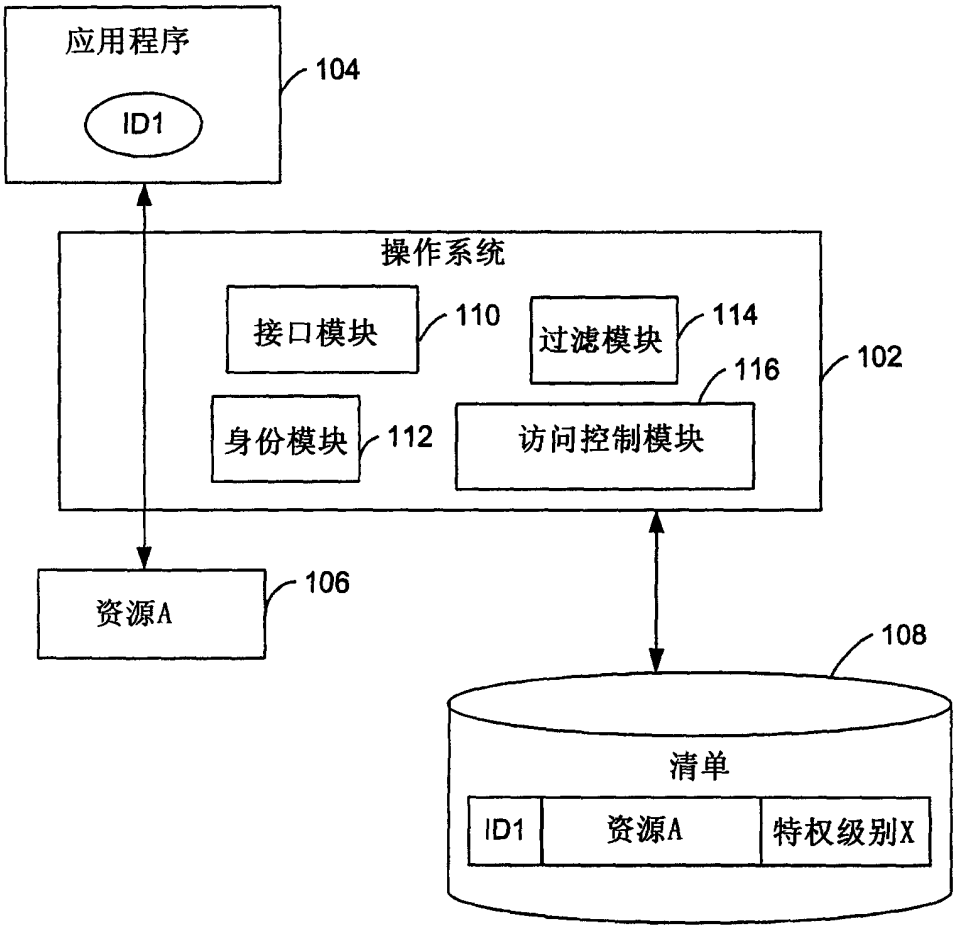

应用程序
ID1
104
操作系统
接口模块
110
过滤模块
114
116
102
身份模块
112
访问控制模块
资源A
106
108
清单
ID1
资源A
特权级别X

图 1

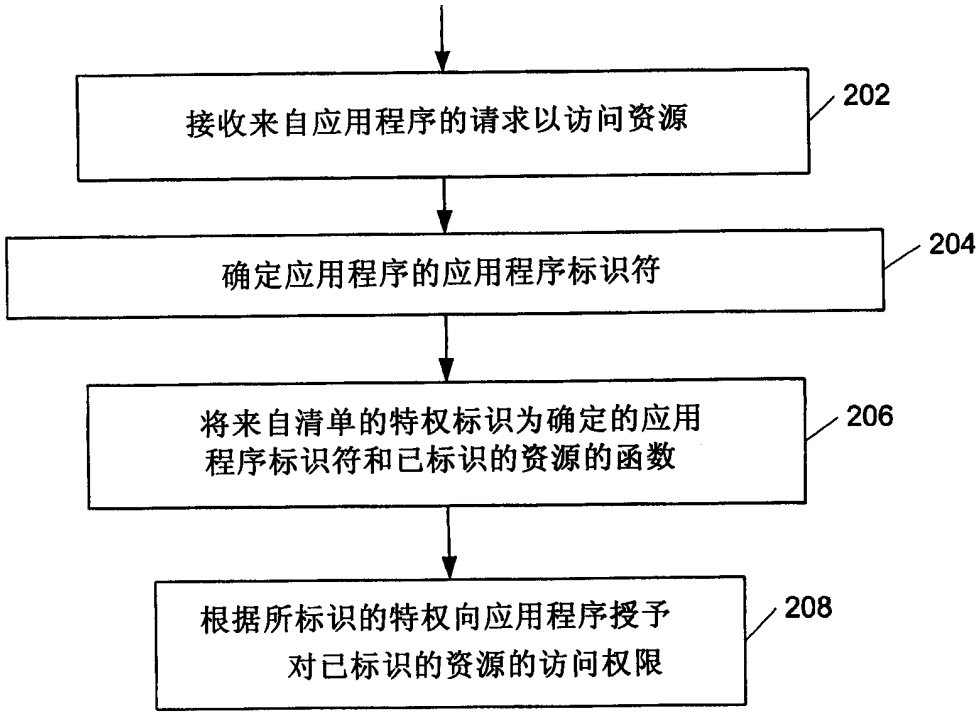

接收来自应用程序的请求以访问资源
202
确定应用程序的应用程序标识符
204
将来自清单的特权标识为确定的应用程序标识符和已标识的资源的函数
206
根据所标识的特权向应用程序授予对已标识的资源的访问权限
208

图 2

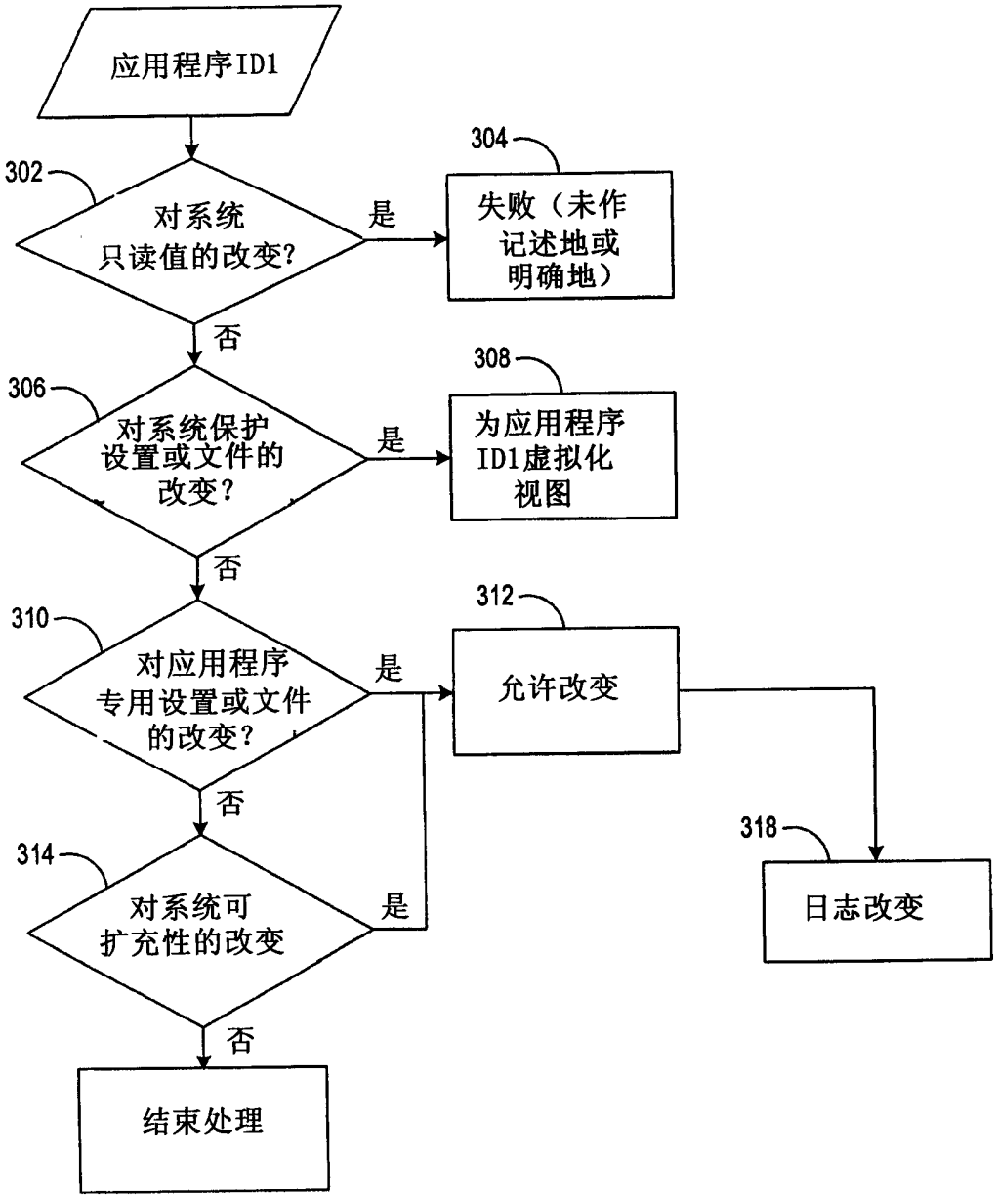

应用程序ID1
302
对系统
只读值的改变?
是
304
失败（未作
记述地或
明确地）
否
306
对系统保护
设置或文件的
改变?
是
308
为应用程序
ID1虚拟化
视图
否
310
对应用程序
专用设置或文件
的改变?
是
312
允许改变
否
314
对系统可
扩充性的改变
是
318
日志改变
否
结束处理

图 3

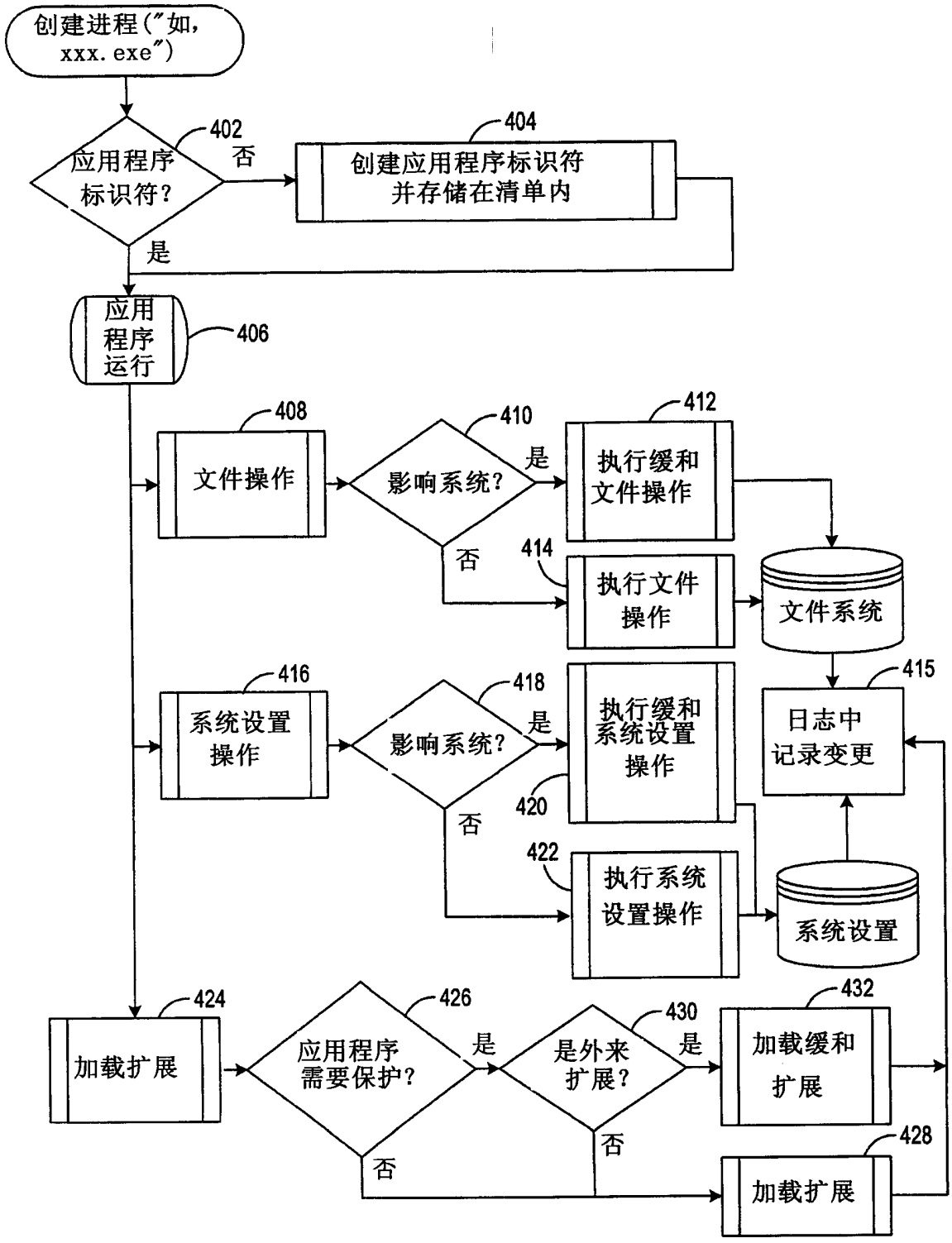

创建进程("如, xxx.exe")
402
应用程序标识符?
否
404
创建应用程序标识符并存储在清单内
是
应用程序运行
406
408
文件操作
410
影响系统?
是
412
执行缓和文件操作
否
414
执行文件操作
文件系统
416
系统设置操作
418
影响系统?
是
执行缓和系统设置操作
415
日志中记录变更
420
否
422
执行系统设置操作
系统设置
424
加载扩展
426
应用程序需要保护?
是
430
是外来扩展?
是
432
加载缓和扩展
否
否
428
加载扩展

图 4

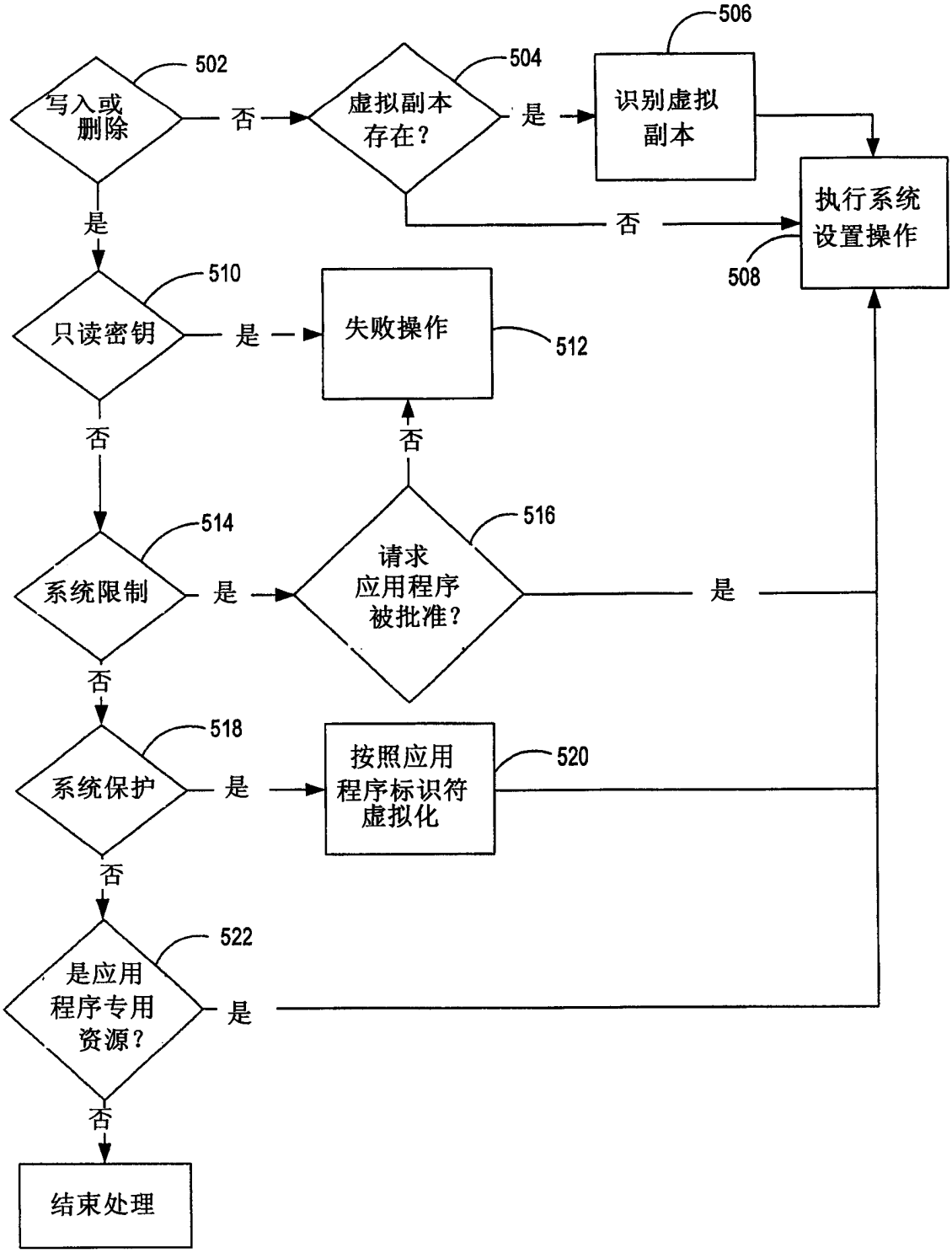

写入或删除
502
否
虚拟副本存在?
504
是
识别虚拟副本
506
否
执行系统设置操作
508
是
只读密钥
510
是
失败操作
512
否
否
系统限制
514
是
请求应用程序被批准?
516
是
否
系统保护
518
是
按照应用程序标识符虚拟化
520
否
是应用程序专用资源?
522
是
否
结束处理

图 5

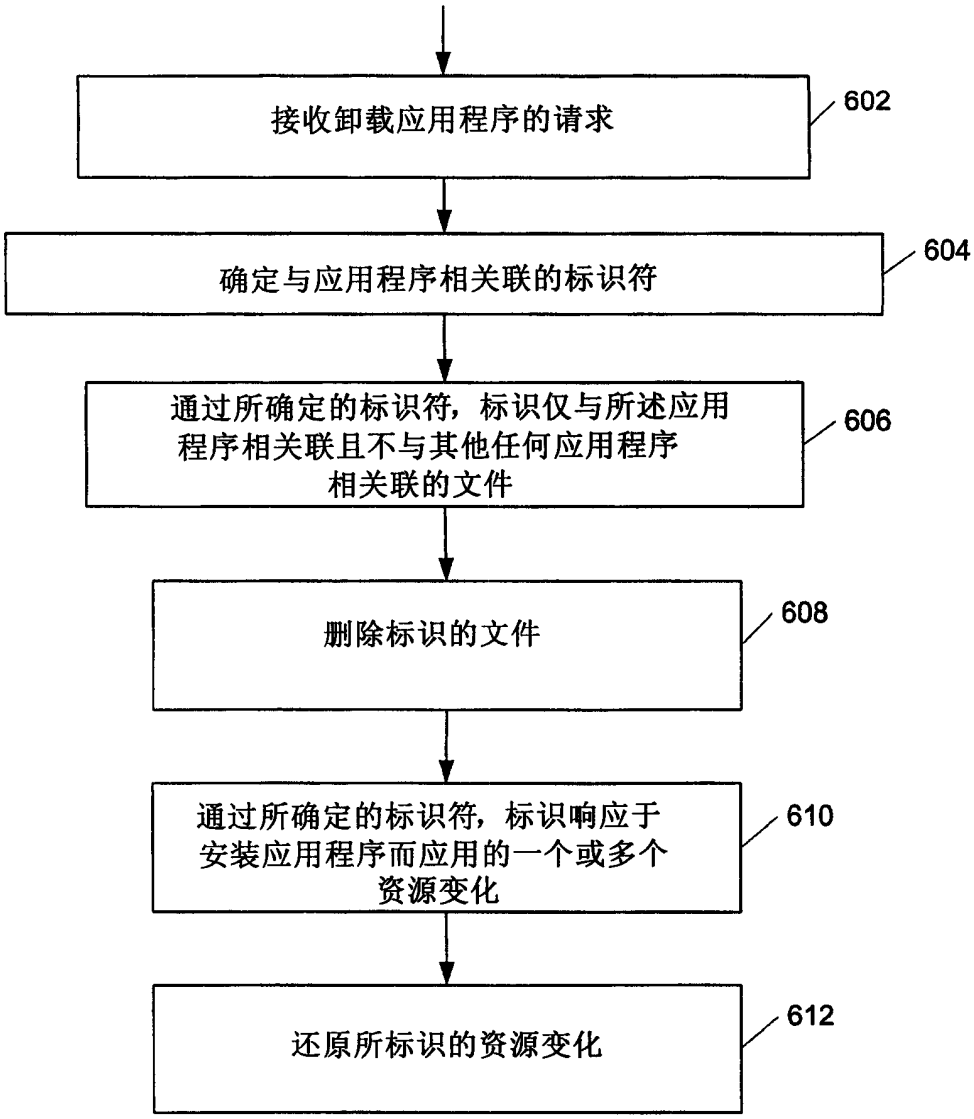

接收卸载应用程序的请求
602
确定与应用程序相关联的标识符
604
通过所确定的标识符，标识仅与所述应用程序相关联且不与其他任何应用程序相关联的文件
606
删除标识的文件
608
通过所确定的标识符，标识响应于安装应用程序而应用的一个或多个资源变化
610
还原所标识的资源变化
612

图 6

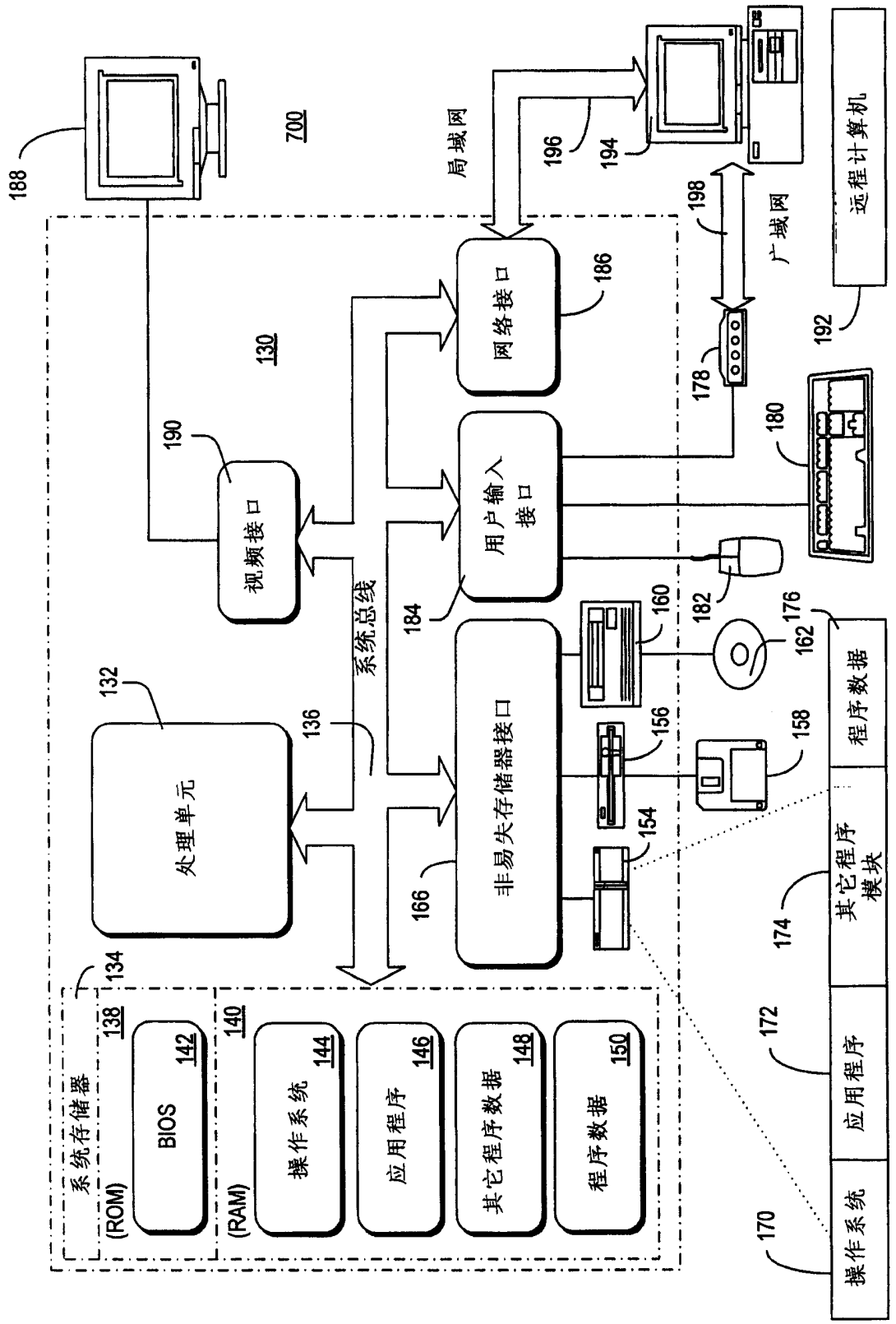

188
700
局域网
196
194
198
广域网
远程计算机
192
178
180
190
视频接口
130
网络接口
186
用户输入接口
184
系统总线
132
处理单元
136
非易失存储器接口
160
182
176
162
156
158
154
166
程序数据
其它程序模块
174
应用程序
172
操作系统
170
系统存储器
134
138
(ROM)
BIOS
142
140
(RAM)
操作系统
144
应用程序
146
其它程序数据
148
程序数据
150

图 7