(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 582 164**

(51) Int. CI.:

G06F 9/54 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **15.05.2002 E 02736858 (8)**

(97) Fecha y número de publicación de la concesión europea: **08.06.2016 EP 1419437**

(54) Título: **Capa de abstracción y protección del sistema operativo**

(30) Prioridad:

16.05.2001 US 859209

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
09.09.2016

(73) Titular/es:

MICROSOFT TECHNOLOGY LICENSING, LLC (100.0%)
One Microsoft Way
Redmond, WA 98052, US

(72) Inventor/es:

SCHAEFER, STUART

(74) Agente/Representante:

CARPINTERO LÓPEZ, Mario

DESCRIPCIÓN

Capa de abstracción y protección del sistema operativo

La presente invención se refiere a un software informático, y más específicamente a un software de sistema operativo.

Antecedentes de la invención

En muchos entornos, pero sobre todo en entornos en los que una aplicación se presenta a través de una red, la característica más importante es la capacidad de ejecutar aplicaciones sobre la marcha, sin una instalación compleja. Por lo general, en ciertos sistemas de la técnica anterior, se han hecho grandes esfuerzos para modificar un sistema cliente para aparecer como si un programa estuviese instalado, o para instalar realmente el software en sí mismo, y a continuación volver atrás estas modificaciones para restaurar la configuración original. Al hacer esto, se presentan múltiples problemas: conflictos entre una aplicación y la configuración actual del equipo, múltiples instancias de las mismas o diferentes aplicaciones, la complejidad del procedimiento de volver atrás requiere que una aplicación se ponga a través de un procedimiento riguroso para garantizar que todas sus modificaciones puedan tenerse en cuenta, y el uso de archivos compartidos y componentes del sistema por múltiples aplicaciones complica la vuelta atrás y el procedimiento de instalación.

Sumario de la invención

Es el objeto de la presente invención mantener la integridad del sistema. Este objeto se resuelve mediante el objeto de la reivindicación independiente. Las realizaciones preferidas están definidas por las reivindicaciones dependientes.

La presente invención proporciona un sistema para crear un entorno de software de aplicación sin cambiar un sistema operativo de un equipo cliente, comprendiendo el sistema una capa de abstracción y protección de sistema operativo, en el que dicha capa de abstracción y protección está interpuesta entre una aplicación de software en ejecución y dicho sistema operativo, por lo que se proporciona un entorno virtual en el que una aplicación puede ejecutarse y se eliminan sustancialmente las interacciones a nivel de aplicación. Preferentemente, cualquier cambio directamente en el sistema operativo se realiza de manera selectiva dentro del contexto de la aplicación en ejecución y la capa de abstracción y protección cambia de manera dinámica el entorno virtual de acuerdo con la configuración administrativa. Además, en ciertas realizaciones, el sistema monitoriza continuamente el uso de los recursos y los actos del sistema compartidos como un servicio de aplicar y quitar los cambios a los componentes del sistema.

Así, por ejemplo, en las realizaciones dentro de los sistemas operativos basados en Windows, y en los que todas las operaciones en el registro de Windows son a través de la API de Win32, el sistema proporciona preferentemente un medio para las funciones de interceptación, por lo que cada vez que se invocan dichas funciones otra función o aplicación intercepta la llamada, y el sistema intercepta más preferentemente cada función API adecuada para dar servicio a una solicitud si se hace por una aplicación ejecutada desde un servidor o si se hace por una aplicación contra una clave de configuración que se administra activamente.

En otras realizaciones preferidas de la presente invención, se proporciona una funcionalidad adicional, tal como estas realizaciones en las que la capa de abstracción y protección de sistema operativo administra la integración de múltiples instancias de una aplicación reconociendo cuántas instancias de una aplicación están en ejecución, y en tales realizaciones más preferentemente se evita hacer cambios en el arranque y el apagado a menos que solo haya una instancia de la aplicación en ejecución. En esta realización es posible también soportar los sistemas operativos multiusuario en los que pueden estar en ejecución múltiples instancias de una aplicación en nombre de los diferentes usuarios.

Por lo tanto, la capa de abstracción y protección de sistema operativo presenta al entorno a una aplicación que parece ser un entorno de instalación sin realizar una instalación, por lo que se crea una “pseudo instalación” en la que todos las configuración se ponen en un entorno virtual en el momento de ejecutar la aplicación. O en el caso de una aplicación instalada, actúa para modificar dinámicamente el comportamiento de la aplicación en el tiempo de ejecución. Las realizaciones preferidas proporcionan un medio para evitar que la información en el equipo cliente interfiera o modifique el comportamiento de una aplicación, y lo más preferentemente proporciona un medio para cambiar dinámicamente el entorno virtual de acuerdo con las configuraciones administrativas. Como se ha mencionado anteriormente, en ciertas realizaciones será posible tener más de una instancia de una única aplicación de software en ejecución en el mismo equipo cliente, incluso si originalmente no se autorizó a hacerlo. En tales realizaciones, se proporcionan contextos compartidos y controlados en los que al menos dos de dichas instancias de una única aplicación comparten una o más configuraciones virtuales.

El documento US 5.752.005 describe un equipo que incluye un sistema operativo que tiene un administrador de sistema y un sistema de archivos nativo. Un sistema de archivos instalable instala unas interceptaciones en una interfaz central entre el administrador de sistema y el sistema de archivos nativo. Las interceptaciones se instalan en sólo una parte de las muchas rutinas de función nativas proporcionadas por el sistema de archivos nativo. Un

controlador virtual se comunica con el sistema de archivos nativo y simula un dispositivo de almacenamiento de archivos que no tiene substancialmente ningún archivo almacenado en el mismo. El dispositivo de sistema de archivo externo solicita que las rutinas de función que no se ha interceptado se manejen por el sistema de archivos nativo y el dispositivo virtual. El dispositivo de sistema de archivo externo solicita que las rutinas de función interceptadas se manejen por un sistema de archivo externo.

El documento US 6.026.402 describe un procedimiento y un aparato para restringir un procedimiento o jerarquía de procedimientos para un subconjunto del sistema(s) de archivos de un equipo central en un entorno en el que todos los sistemas de archivos están disponibles al mismo tiempo para una aplicación. Una persona que llama está provista de la capacidad de organizar la restricción de una jerarquía de procedimientos (que consiste en un procedimiento y todos los procedimientos que pueden crearse posteriormente) para un conjunto de subjerarquías de sistemas de archivos. Cuando se ejecuta un procedimiento dentro de un dominio de restricción en el que se ha especificado una subjerarquía para cada uno de los sistemas de archivos disponibles, una realización modificará el funcionamiento habitual de la interfaz de sistema operativo del equipo central de tal manera que cualquier intento de acceso al sistema de archivos por el procedimiento afectado está obligado a producirse lógicamente dentro del dominio de restricción.

Breve descripción de los dibujos

La figura 1 es un diagrama de bloques esquemático que muestra la relación relativa de la presente invención, un sistema operativo y una aplicación de software;

La figura 2 es un diagrama de bloques esquemático que muestra

La figura 3 es un diagrama de bloques esquemático que muestra

La figura 4 es un diagrama de bloques esquemático que muestra

Descripción detallada de las realizaciones preferidas

Haciendo referencia ahora a la figura 1, se ilustra un diagrama de bloques esquemático que muestra la relación relativa de la presente invención, un sistema operativo y una aplicación de software. Las realizaciones preferidas de la presente invención proporcionan una capa 100 de abstracción y protección de sistema operativo denominada “restricción de sistema operativo”. Internamente, muchos sistemas 10 operativos proporcionan dominios de fallo para proteger las aplicaciones 50 de afectarse entre sí cuando se ejecutan. Sin embargo, los recursos de sistema compartidos y muchas otras características de sistema operativo permiten que este dominio de protección se vea comprometido. Una capa 100 de abstracción y protección de sistema operativo proporcionará una barrera adicional controlada mediante programación entre las aplicaciones 50 para eliminar la mayoría de las interacciones a nivel de aplicación. Dispuesta entre la aplicación 50 y el sistema 10 operativo, la capa 100 de abstracción y protección de sistema operativo permite selectivamente cambios directamente en el sistema 10 operativo, frente a contener el cambio dentro del contexto de la aplicación en ejecución. Como un ejemplo, en los sistemas basados en Windows, todas las operaciones en el registro de Windows se realizan normalmente a través de la API de Win32. Como se explica a continuación, las funciones de sistema como QueryRegEx y GetProfileString pueden interceptarse de tal manera que cada vez que se invocan, otra función o aplicación intercepta la llamada. La restricción 100 de sistema operativo de la presente invención interceptará cada función API adecuada para dar servicio a la solicitud, si se hace por una aplicación que se administra activamente o si se hace por una aplicación contra un elemento de configuración que se administra activamente. De esta manera, a menos que se configure explícitamente para hacerlo así, la presente invención puede crear el entorno de aplicación sin realizar cambios reales en el sistema del usuario final. Por lo tanto, cualquier modificación que se realice en tiempo de ejecución por la aplicación puede persistirse o eliminarse fácilmente.

Como se usa en el presente documento, la expresión “restricción de sistema operativo”, define una capa entre una aplicación en ejecución y el sistema operativo de un equipo de destino o equipo cliente que proporciona un entorno virtual en el que la aplicación puede ejecutarse. Este entorno virtual tiene diversos fines. En primer lugar, evita que una aplicación en ejecución realice cambios en el equipo cliente. Si una aplicación intenta cambiar la configuración de sistema operativo subyacente de un equipo cliente, tal configuración está protegida y solo se “realiza” en el entorno virtual. Por ejemplo, si una aplicación intenta cambiar la versión de un objeto compartido como MSVCRT.DLL, este cambio se localiza en la aplicación y el código residente en el equipo cliente se deja intacto.

En segundo lugar, la invención presenta un entorno para una aplicación en ejecución que parece ser un entorno de instalación sin realizar una instalación, y es, por lo tanto, una “pseudo instalación” o “como una instalación”. Todas las configuraciones se ponen en un entorno virtual en el momento en que la aplicación se sirve en ejecución, o justo en el momento cuando la aplicación necesita una configuración específica. Por ejemplo, si un programa de ordenador tal como Adobe Photoshop® espera ver un conjunto de entradas del registro de Windows bajo HKEY_LOCAL_MACHINE\Software\Adobe y no están allí en el equipo cliente ya que Photoshop nunca se ha instalado, un sistema realizado de acuerdo con este aspecto de la presente invención “mostrará” esas entradas de registro en el código de programación Photoshop exactamente como si estuvieran residentes en el equipo cliente.

A continuación, la invención evita que la información que puede existir en la máquina cliente/usuarios interfiera con o modifique el comportamiento de una aplicación. Por ejemplo, si el usuario tiene ya las entradas del registro que existen bajo:

HKEY_LOCAL_MACHINE\Software\Adobe

para una versión anterior de Photoshop, pero ahora desea hacer funcionar una versión más reciente, esas entradas pueden ocultarse a la nueva aplicación para evitar conflictos.

Finalmente, la presente invención desbloquea el comportamiento de aplicación que puede no existir cuando la aplicación se escribe actualmente. Esto se hace a través de la capacidad para cambiar dinámicamente el entorno virtual de acuerdo con las configuraciones administrativas. Por ejemplo, en una instancia típica de una aplicación de software empresarial, una aplicación cliente puede esperar para leer una configuración de la dirección de la base de datos a la que el usuario debería conectarse a partir de una configuración en el registro. Debido a que esta clave de registro se almacena a menudo en HKEY_LOCAL_MACHINE, la configuración es global para todo el equipo cliente. Un usuario solo puede conectarse a una base de datos sin reinstalar el cliente, o sabiendo cómo modificar esta clave de registro, y hacerlo cada vez que se desee ejecutar la aplicación. Sin embargo, implementando la presente invención, ahora pueden ejecutarse dos instancias de la aplicación en el mismo equipo cliente, cada una conectada a una base de datos diferente.

CONTEXTOS

Al proporcionar esta funcionalidad, cada aplicación es capaz de ejecutarse en un contexto privado dentro del sistema. Para la aplicación, esto tiene su propia vista privada de lo que parecen el sistema y su comportamiento. La presente invención proporciona esto por su naturaleza inherente. Haciendo referencia a la figura 2, pueden proporcionarse dos aplicaciones separadas 52, 54, o dos instancias de la misma aplicación (50 ilustradas en la figura 1), en contextos privados en los que parecerán tener copias separadas o diferentes de los servicios, la configuración y los datos del sistema. En la realización preferida, este es el comportamiento por defecto del sistema.

Con la extensión de este concepto, la restricción 100 de sistema operativo de la presente invención puede proporcionar también unos contextos compartidos y controlados en los que dos o más aplicaciones 52, 54 pueden compartir alguna o la totalidad de sus configuraciones virtuales. Esto es importante para los conjuntos de aplicaciones tales como Microsoft Office, o para las aplicaciones que se comportan de manera diferente en la presencia de otras aplicaciones. Por ejemplo, muchas aplicaciones utilizan Microsoft Word como el motor para hacer la combinación de correspondencia o la funcionalidad de creación de documentos. La aplicación debe conocer la instalación o la presencia de Word y ser capaz de acceder a sus funciones. En la realización preferida, dos instancias de la misma aplicación compartirán un único contexto por defecto, mientras que dos aplicaciones individuales mantendrán contextos privados. Haciendo referencia a la figura 3, las dos aplicaciones 52, 54 pueden ejecutarse mientras que la restricción 100 de sistema operativo proporciona una visión compartida de los recursos disponibles del sistema.

DISEÑO

Como se ilustra en la figura 4, la restricción de sistema operativo está compuesta por los siguientes subsistemas: un núcleo 102, un administrador 104 de configuración, un administrador 106 de archivos, un administrador 108 de objetos compartidos, un administrador 110 de dispositivos, un administrador 112 de fuentes, un administrador 120 de procedimientos, un administrador 114 de entorno de procedimientos, un cargador 116, y un administrador 118 de recuperación. Con la excepción del núcleo 102, el administrador 120 de procedimientos, y el cargador 116, todos los demás subsistemas son elementos del sistema de virtualización descrito con más detalle a continuación. El núcleo 102 es el principal responsable de la administración de las aplicaciones y su contexto como se define en los archivos de configuración.

El administrador 120 de procedimientos proporcionado por la restricción de sistema operativo permite que se informe al núcleo 102 de cualquier procedimiento o evento de subprocedimiento que pueda ser de interés. Esto también proporciona una capa de abstracción a las implementaciones dependientes del sistema operativo para administrar un espacio de procedimiento y manejar el procesamiento de subprocedimientos. Los procedimientos pueden agruparse entre sí en paquetes de aplicaciones. Un paquete de aplicaciones es un grupo de procedimientos en el que todos comparten sus recursos virtuales entre sí. Por ejemplo, Microsoft Word y Microsoft Excel pueden querer compartir el registro virtual y el sistema de archivos virtual para poder trabajar juntos como un conjunto de aplicaciones. El administrador 120 de procedimientos llama a esos paquetes de aplicaciones “aplicaciones”. La información acerca de una aplicación existe hasta que se le dice al administrador 120 de procedimientos que libere la aplicación. Si otro procedimiento necesita cargarse en el paquete de aplicación, puede hacerlo siempre y cuando no se haya liberado la aplicación.

El subsistema 116 cargador de la presente invención se usa para permitir que los entornos virtuales se transfieran dentro y fuera del sistema en ejecución. Cada uno de los subsistemas de virtualización es capaz de serializar su configuración para el cargador 116, y recuperarla a través del procedimiento inverso. Además, el cargador 116 es capaz de realizar fases de carga/descarga y combinar los resultados de las fases individuales en una única

descripción de entorno.

REGISTRO Y CONFIGURACIÓN

Las aplicaciones requieren cantidades variables de información de configuración para funcionar correctamente. En cualquier lugar existen de cero a miles de registros de configuración para los que una aplicación puede leer su configuración. En Windows, hay dos lugares comunes para la información de configuración, el registro de Windows y los archivos de inicialización a nivel de sistema win.ini y system.ini. Además, el directorio \WINDOWS\SYSTEM es un lugar común para que las aplicaciones escriban la configuración específica de aplicación o los archivos de inicialización. Las aplicaciones usarán también los archivos de configuración o de datos en sus directorios de aplicación locales para almacenar la información de configuración adicional. A menudo, esta información es difícil de tratar, ya que está en un formato propietario. En las plataformas que no sean Windows, no existe un equivalente del registro, pero existen directorios comunes para la información de configuración. X Windows tiene un directorio de aplicaciones por defecto (app-defaults). Macintosh tiene la carpeta de sistema (System Folder), y otros sistemas operativos tendrán los elementos correspondientes. Es importante observar que en la mayoría de los sistemas UNIX, cada aplicación 52, 54 individual almacenará lo más a menudo su propia configuración 152, 154 localmente, como se ve en la figura 2.

La presente invención, en una realización, incluye un componente virtual de registro de Windows, que proporcionará un registro de función completo a una aplicación, pero evita la modificación del registro de sistema subyacente. Todas las claves que una aplicación espera acceder estarán presentes, pero solo pueden existir en el registro virtual. De esta manera, la restricción 100 de sistema operativo de la presente invención y el registro de Windows forman un procedimiento de dos fases para acceder al registro. Si una aplicación necesita acceder a una clave, consultará al registro. La restricción de sistema operativo responderá con la clave y su valor si la conoce. De lo contrario, permitirá que la solicitud pase a través del registro de Windows. Si se hace un intento de modificar el valor, la restricción de sistema operativo permitirá que se produzca la modificación solamente sobre sí misma. La próxima vez que la aplicación acceda a la clave, estará presente en la restricción de sistema operativo y la solicitud no fluirá a través del registro real, dejándolo intacto.

Las claves que usan la restricción de sistema operativo se especifican en tres secciones individuales. Estas claves de restricción de sistema operativo se especifican como comandos en esas secciones para modificar una clave existente, eliminar la presencia de una clave, o añadir una nueva clave al registro. De esta manera, el registro virtual puede aparecer exactamente como las intenciones de sistema. Esto es importante cuando la presencia o la ausencia de una clave pueden ser tan importantes como el valor real de la clave.

En la realización preferida, la restricción de sistema operativo carga primero un archivo de datos que contiene unas entradas de registro básicas para la aplicación. A continuación, se carga un segundo archivo de datos que contiene las preferencias del usuario. Por último, la restricción de sistema operativo puede cargar opcionalmente, un conjunto de claves que incluye elementos de política que el usuario no está autorizado a anular. Los tres archivos se cargan uno encima de otro con elementos duplicados en cada archivo anulando los elementos del archivo anterior al mismo. La primera vez que un usuario ejecuta la aplicación, el segundo archivo de datos no existe, porque no habrá información específica del usuario, solo las predeterminadas de aplicación. Después de cada sesión, sin embargo, la restricción de sistema operativo guardará los cambios del usuario, generando este segundo archivo de datos para su uso en sesiones futuras.

Los archivos de configuración pueden modificarse de dos maneras. En primer lugar, el archivo puede editarse directamente por una aplicación. En este escenario, el subsistema de archivos de restricción de sistema operativo descrito a continuación abordará la modificación realizada en el archivo. En segundo lugar, en la realización preferida, un aplicación puede llamar a la familia de llamadas API de Windows GetProfileString, WriteProfileString, u otras para modificar estos archivos. En este caso, la restricción de sistema operativo de la presente invención se realiza exactamente como se ha descrito anteriormente interceptando esas llamadas y dándolas servicio desde dentro.

OBJETOS COMPARTIDOS

Muchos componentes usados por los sistemas operativos y las aplicaciones en ejecución se comparten entre varias aplicaciones o instancias. En general, esta es una muy buena idea. Se ahorra espacio en disco, no requiriendo muchas copias del mismo archivo. También proporciona la capacidad para los proveedores de sistemas operativos y de terceros de crear y distribuir bibliotecas de código usado comúnmente. En la plataforma Windows, las bibliotecas de enlaces dinámicos, DLL, se comparten a menudo dentro y a través de las aplicaciones. En otras plataformas, el problema es el mismo. En Macintosh, los INIT y otros componentes del sistema se cargan para las aplicaciones. Estos componentes pueden tener muchas versiones, de las cuales solo se usa una a la vez. En los sistemas UNIX, los objetos compartidos dinámicos, por ejemplo, los archivos de biblioteca “.so”, se usan por las aplicaciones para acelerar el tiempo de carga, ahorrar espacio en disco, y por otras razones. Muchos programas usan el “libc.so” por defecto. Sin embargo, este archivo de biblioteca es normalmente un enlace simbólico a alguna versión de sí mismo: como libC.so.3. En la práctica, esta característica ha hecho estragos. Estos componentes compartidos han ido a menudo a través de una revisión, con muchas versiones del mismo componente disponible para instalarse. Los

autores de aplicaciones han descubierto que su software trabaja potencialmente con solo una o algunas de las versiones del componente compartido. Por lo tanto, en la práctica, las aplicaciones instalan normalmente la versión que ellas desean, sobrescribiendo otras versiones actuales. Potencialmente, esto hace que los valores por defecto en otras aplicaciones se ejecuten en un sistema.

En Windows 98, Windows 2000, Microsoft ha creado el sistema de archivos protegidos de Windows (WPFS) para permitir a los administradores de sistemas crear un archivo llamado XXXX.LOCAL en el directorio raíz de una aplicación, en el que XXXX es el nombre del archivo ejecutable sin la extensión. Esto hace que el cargador de Windows altere su procedimiento de resolver las referencias de ruta durante las ejecuciones de LoadLibrary. Esto, sin embargo, no es suficiente para resolver por completo el problema. En primer lugar, la configuración del archivo XXXX se deja al conocimiento del administrador del sistema, lo que es muy variable. En segundo lugar, una versión de componente debe someterse a un retroceso de nuevo al original, a continuación, instalar este componente en el directorio local y, a continuación, crear el archivo ".LOCAL". Este no es un procedimiento sencillo para cualquiera de los componentes más básicos localizados en WINDOWS\SYSTEM. También, esta solución no cubre todas las funcionalidades necesarias. Durante LoadLibrary, Windows usa una semántica diferente de resolución de ruta en función de si el componente se resuelve como resultado de una LoadLibrary explícita o implícita, y también si existe una clave de registro que indique que es una DLL nombrada o bien conocida. En este caso, la llamada de LoadLibrary siempre se resolverá en el directorio WINDOWS\SYSTEM.

Las DLL y otros componentes compartidos conservan también la semántica de recuento de referencia para garantizar que un componente no se toca a menos que no haya aplicaciones en ejecución que se refieran al mismo. En la práctica, solo las aplicaciones del proveedor de sistema operativo y el propio sistema operativo han hecho un buen trabajo de obedecer este protocolo.

Como regla general, se desea tener un objeto compartido que siempre se resuelva para el componente correcto. Para proporcionar esta funcionalidad, se requiere entender la versión de un componente, o intervalo de versiones, con el que es posible hacer funcionar la aplicación. A continuación, cuando la aplicación se va a ejecutar, la presente invención garantizaría que el componente se resuelve correctamente. Es aceptable, en la presente invención, para automatizar el uso de WPFS u otro sistema operativo que proporciona la capacidad, si se desea. En este caso, es necesario detectar los componentes necesarios y colocarlos en el sistema de archivos local. Esto es más complejo que la simple observación de la instalación, puesto que un programa de instalación a menudo no instalará un componente si el necesario ya está ahí.

Esto se desea para identificar un procedimiento para garantizar que los objetos nombrados se cargan correctamente. En la plataforma Windows, MSVCRT.DLL es un responsable significativo dentro de esta área de problemas. Si se mantienen múltiples versiones de este objeto, la clave de registro mencionada anteriormente puede cambiarse dinámicamente, lo que permite que la función LoadLibrary resuelva la versión correcta del componente. Otro procedimiento razonable de garantizar la carga correcta del componente es la edición dinámica de un entorno de procedimiento para usar una ruta de búsqueda válida. Esta ruta de búsqueda garantizará que un componente local se resuelve antes que un componente amplio de sistema. Otro posible procedimiento para la resolución del objeto compartido correcto es a través del uso de los enlaces simbólicos. Puede hacerse un enlace simbólico para un componente compartido, que se resuelve en tiempo de ejecución por el sistema de archivos del equipo para el componente necesario. Por último, las solicitudes abiertas/leídas/cerradas reales de un archivo del objeto compartido pueden interceptarse por la presente invención y responderse dinámicamente para la versión correcta del archivo que pueda existir en el sistema local o dentro de los subsistemas de la invención.

Existen varias formas especiales. En la plataforma Windows, OLE, ODBC, MDAC,… así como una serie de otros componentes específicos de fabricante, están escritos para compartirse globalmente entre varios o todos los procedimientos en ejecución. En el caso de OLE, yendo tan lejos como compartir datos y espacio de memoria entre procedimientos individuales. OLE evita más de una copia de sí mismo en ejecución al mismo tiempo, al igual que muchos de estos componentes. OLE tiene también muchos errores y características que requieren una versión específica a cargarse para una aplicación específica. En la presente invención, una aplicación es capaz de cargar cualquier versión de OLE que se requiera, aun permitiendo la semántica compartida con otros componentes que usan la misma versión de OLE.

En general, a menos que se configure específicamente como tal, los objetos compartidos deberían cargarse privadamente para garantizar la prevención de conflictos. Nada evitaría, acerca del procedimiento usado para permitir que un componente se cargue privadamente, que se descarguen limpiamente o se carguen correctamente para otra aplicación de software, si se administra o no activamente por la restricción de sistema operativo. Además, si el sistema se bloquea, se requiere para recuperarse de este bloqueo un estado limpio, sin haber sobrescrito o modificado el sistema operativo subyacente.

ARCHIVOS

Muchas aplicaciones usan archivos de datos dentro de la aplicación para almacenar entradas de configuración u otros datos de aplicación. La presente invención proporciona un sistema de archivos virtual al igual que el registro virtual descrito anteriormente. Antes del comienzo de la aplicación, la presente invención puede cargar una lista de

cambios de sistema de archivos, que incluye archivos para ocultar y archivos para agregar al entorno virtual o archivos para redirigir a otro dentro del entorno virtual. Siempre que la aplicación accede o modifica cualquier archivo, la restricción de sistema operativo comprueba si el archivo debe redirigirse, y si es así, en la realización preferida redirige la solicitud a una localización especificada en la configuración de la restricción de sistema operativo.

Si una aplicación intenta crear un nuevo archivo o abrir un archivo existente para escribir en una unidad local del usuario, la restricción de sistema operativo debe garantizar que el archivo se crea o se modifica realmente en la localización redirigida. Si la aplicación se vuelve a cargar en un momento posterior, este mapeo de archivo debe recargarse en el entorno virtual de la restricción de sistema operativo. Cuando la solicitud es para modificar un archivo existente, que reside en una unidad local del usuario, la restricción de sistema operativo debe copiar el archivo en cuestión al punto de redirección antes de continuar con la solicitud. Los archivos redirigidos pueden no ser del mismo nombre que el archivo original para garantizar el mapeo seguro de las rutas de archivos. En la realización preferida, los archivos INI se manejan de esta manera para ofrecer la máxima seguridad del sistema al tiempo que permiten la máxima compatibilidad de aplicaciones.

La presente invención es específicamente útil para las aplicaciones distribuidas a través de una red. En tales implementaciones es importante entender que las aplicaciones de software están hechas de varios tipos de datos, en las que el grueso de los archivos de aplicaciones que usan una aplicación de software están montados lo más preferentemente en una unidad lógica independiente. La configuración, que incluye tanto la basada en archivo como la basada en el registro, puede ser específica del usuario y de todo el sistema. El sistema de distribución de aplicaciones usado marcaría cada archivo para el que de estos tipos hay algún archivo. Esta información proporciona consejos al sistema de restricción de sistema operativo para actuar de manera apropiada.

CONTROLADORES DE DISPOSITIVOS

Muchas aplicaciones usan controladores de dispositivos u otro software a nivel de sistema operativo para implementar algunas de sus funciones tales como el soporte de hardware o las interacciones de bajo nivel directamente con el sistema operativo. En la presente invención, la restricción de sistema operativo proporcionará la capacidad de dinámicamente, y tan privadamente como sea posible, agregar y eliminar estos componentes en el entorno virtual de la aplicación.

Muchos controladores de dispositivos están diseñados para poder cargarse dinámicamente. Si es posible, es la realización preferida cargar todos los controladores de dispositivo dinámicamente. Si un controlador de dispositivo requiere una carga estática en el momento del arranque, debe presentarse este conocimiento al usuario antes de ejecutar la aplicación. Una vez que el sistema ha reiniciado la aplicación debería continuar desde donde lo dejó. Sin embargo, un gran porcentaje de los controladores de dispositivos no puede descargarse dinámicamente. Aunque se prefiere descargar de manera dinámica el controlador, si esto no puede realizarse el controlador será marcado para su eliminación en el siguiente reinicio, y el usuario debería estar al tanto de esto. Si la aplicación se ejecuta una segunda vez antes del siguiente reinicio, el sistema debería ser consciente de la presencia del controlador y no intentar una segunda instalación, esperando a la terminación para remarcar el componente eliminable en el siguiente reinicio.

Es importante caracterizar las similitudes y las diferencias de base, tal como existen para cada clase de controlador de dispositivo, para garantizar que la presente invención pueda funcionar correctamente. No se desea realmente cargar y descargar los controladores de dispositivos para el hardware del sistema que está constantemente presente. Debería entenderse que aunque esto no es una realización preferida en términos de facilidad de programación, está dentro del ámbito de la presente invención y puede requerirse por razones específicas, tales como la restricción en los acuerdos de concesión de licencias para las aplicaciones que se entregan y ejecutan usando la presente invención.

En las plataformas que no son de Microsoft, los controladores de dispositivos se manejan normalmente de manera muy diferente. Los sistemas Macintosh soportan tanto los controladores estáticos como dinámicos, pero todos ellos se instalan y se eliminan a través del mismo procedimiento. Enlazar con la carpeta de sistema de Macintosh proporcionará el soporte necesario. Para los sistemas UNIX, los controladores de dispositivos requieren más normalmente una modificación para el núcleo de UNIX en ejecución, seguido de un reinicio. Este procedimiento puede ser muy complejo. En la realización preferida, este procedimiento está automatizado; incluyendo restablecer el núcleo una vez que la aplicación está completa. Los parámetros generales del procedimiento son los mismos que se han descrito anteriormente para las aplicaciones de Windows, las etapas de procedimiento reales de la compilación y las personas familiarizadas con tales sistemas operativos pueden realizar el reinicio.

Finalmente, los expertos en la materia entenderán que es deseable ser capaz de recuperar y eliminar los controladores a través de los fallos del sistema. Por lo tanto, cualesquiera que sean los datos o procedimientos necesarios para mantener la integridad del sistema es una realización preferida de la presente invención. Los expertos en la materia apreciarán también que todo tipo de controladores de dispositivos podrían no proporcionarse conveniente o eficientemente a través de la presente invención, más específicamente los asociados con los dispositivos de hardware conectados permanentemente.

OTROS ARTÍCULOS

En la presente invención, se reconoce que hay varios componentes de la invención, el comportamiento o la presencia de lo que es diferente en los sistemas operativos alternativos. Estos componentes incluyen fuentes, procedimientos, variables de entorno, y otros.

Algunas aplicaciones requieren fuentes a instalarse con el fin de funcionar correctamente. Cualquier fuente requerida será especificada en el archivo de configuración de la restricción de sistema operativo. La restricción de sistema operativo habilitará esas fuentes antes de ejecutar la aplicación y si es necesario las eliminará posteriormente. La mayoría de los sistemas tienen un área común para el almacenamiento de fuentes, además de un procedimiento de registro de las mismas o para hacer que el sistema sea consciente de su presencia, la restricción de sistema operativo utilizará estos procedimientos disponibles.

En Windows, una fuente se copia en el directorio WINDOWS\FONTS. Sin embargo, esto no garantiza que la fuente esté disponible para el programa en ejecución. En la realización preferida, si el programa usa la API de Windows para acceder a las fuentes, la fuente necesitará registrarse con una API de Win32 llamada tal como CreateScalableFontResource/AddFontResource. Esto insertará la fuente en la tabla de fuentes del sistema. Una vez completo, la restricción de sistema operativo puede eliminar la fuente con otra API apropiada llamada como RemoveFontResource, a continuación, elimina el archivo del sistema. Como una realización alternativa, la restricción de sistema operativo podría interceptar las funciones API como se describe en el procedimiento de registro virtual. Además, la restricción de sistema operativo puede usar su subsistema de archivos para evitar colocar el fichero de fuente real en el sistema en ejecución.

En Macintosh, el procedimiento es extremadamente similar y basado en los archivos de la carpeta de sistema y el registro de activación de Macintosh. En UNIX, sin embargo, el procedimiento depende de la aplicación. Lo más normal, los recursos de fuentes se agregan al sistema como archivos regulares resueltos en la localización adecuada, por lo que pueden accederse por el nombre. Con muchos sistemas Motif, necesita colocarse una descripción de la fuente en un archivo de recursos de fuente, lo que permitirá que la fuente se resuelva. La aplicación Motif o X puede invocar la fuente o a través del subsistema de resolución o mediante una llamada directa. Recientemente, muchos sistemas basados en Motif y en CDE utilizan fuentes postscript escalables de Adobe. Estas fuentes necesitan administrarse a través del sistema de administración de tipo Adobe. Sin embargo, hay excepciones, y como se ha indicado anteriormente, hay alternativas a los sistemas de administración de fuentes por defecto de Windows u otro sistema operativo. El administrador tipo Adobe proporciona algunas interfaces alternativas para este procedimiento, como hacen otros tipos de sistemas de administración de terceras partes. En la mayoría de los casos, debería decidirse si soportar la interfaz o ignorarla. El fin de la restricción de sistema operativo no es proporcionar una capa universal para todos estos sistemas, solo hacerlo así para el propio subsistema del sistema operativo.

Muchas aplicaciones requieren que se configuren las variables de entorno. Esto es muy común en los sistemas UNIX, pero también se usa mucho por el software, que fue escrito originalmente en UNIX y portado a los sistemas operativos Windows. Las aplicaciones en los sistemas operativos Windows dependen en gran medida de la variable de entorno DOS PATH y a menudo configurar sus propias entradas específicas de aplicación. En los entornos de Windows 9x/Me, hay muchas configuraciones de entorno, que pueden aplicarse cuando en su núcleo está el subsistema DOS. Si una aplicación requiere la presencia de unas variables específicas, o valores a configurarse en las variables de entorno existentes, las variables de entorno necesarias se especificarán en el archivo de configuración de la restricción de sistema operativo. La restricción de sistema operativo configurará esas variables para el procedimiento principal de la aplicación cuando se ponga en marcha. Puesto que las aplicaciones no cambian normalmente las configuraciones de entorno cuando están en funcionamiento, el entorno virtual no atrapará esas llamadas, ni las proporcionará el complemento completo de la funcionalidad que hacen el registro y el subsistema de configuración.

RECUPERACIÓN

En algunos casos mostrados en las secciones anteriores, las modificaciones reales deben hacerse al sistema operativo. Esto es frecuente con los controladores de dispositivos y las fuentes. Además, pueden realizarse cambios al entorno virtual que se necesite para persistirse y estar disponible la próxima vez que se ejecute la aplicación. Se requiere que el sistema de restricción de sistema operativo sea capaz de recuperarse de cambios en el sistema, eliminando el cambio del sistema en la primera oportunidad posible. Como alternativa, si el sistema se bloquea durante una ejecución de la aplicación, la restricción de sistema operativo debería realizar un seguimiento de la suficiente información para eliminar cualquier cambio en el sistema si se reinicia o no, y debería realizar un seguimiento de los cambios hechos en el entorno virtual. En la realización preferida, esto se implementa como un registro de transacciones, pero en otras realizaciones puede hacerse como cualquier otro componente similar, que puede leerse al iniciar el sistema de manera que pueden anularse los cambios.

CONTROL DE VIRTUALIZACIÓN

Un aspecto importante de la invención se refiere al control de las muchas facetas de la virtualización que es capaz de hacer la restricción de sistema operativo. En la realización preferida existe una instrumentación programable para determinar los aspectos correctos de un sistema de software a controlar. También se incluye un procedimiento para permitir a los administradores y a los usuarios finales para ver y modificar los elementos a virtualizarse por el sistema.

En el programa automatizado, se observa la aplicación a controlarse con el fin de evaluar los aspectos de control. El programa automatizado es capaz de realizar esta tarea durante el procedimiento de instalación de la aplicación, durante el tiempo de ejecución de la aplicación, o una combinación de ambos. En la realización preferida, la restricción de sistema operativo está integrada en una aplicación contenedora. Tras la instalación, o después de uno o muchos usos del software, la aplicación contenedora consultará a la restricción de sistema operativo para obtener una lista detallada de todas sus acciones. A partir de esta lista de acciones, la aplicación contenedora creará los archivos de configuración necesarios para cargar y hacer funcionar la restricción de sistema operativo en usos posteriores.

Si se usa como parte del procedimiento de instalación, la restricción de sistema operativo, en la realización preferida, actuará como una capa virtual que permite la instalación que se introducirá solo en su entorno. Después de la instalación, todos los archivos, configuraciones, y col. pueden volcarse para una recarga posterior. De esta manera, la instalación dejará el sistema original intacto y habrá creado automáticamente los archivos de configuración necesarios. Cuando se usa durante el uso de la aplicación, la restricción de sistema operativo es capaz de grabar o las modificaciones diferenciales del entorno, o recodificar los archivos de configuración.

La restricción de sistema operativo pasará su información a la aplicación contenedora para un procesamiento posterior. En la realización preferida, además de las entradas automáticas que puede crear el sistema, la aplicación contenedora está programada con un sistema operativo específico y el conocimiento específico de la aplicación o del dominio. Este conocimiento se usa para alterar la salida del procedimiento para reflejar los usos conocidos de los elementos de configuración o de otras entradas. En la realización preferida, se emplea un sistema basado en reglas para comparar los comportamientos observados con los escenarios conocidos con el fin de efectuar cambios en la codificación.

La aplicación contenedora se usa como un visor y/o un editor para la salida de configuración del procedimiento. Este editor, en la realización preferida, permite a un administrador de sistema añadir, editar o eliminar elementos o grupos de elementos de la configuración. Al observar la configuración a través del editor, el administrador puede hacer también réplicas de la configuración, cambiar elementos específicos cuando sea necesario para efectuar los cambios a nivel de aplicación o personalizados por el usuario.

Haciendo referencia ahora a la figura 1, se ilustra funcionalmente una realización de la presente invención. En esta realización, se ilustran dos conjuntos de datos de aplicación/usuario 60. La restricción 100 de sistema operativo mantiene las dos instancias de la aplicación 50 de que interfieran entre sí. Además, como se ha explicado anteriormente, la restricción 100 de sistema operativo sirve como una capa de abstracción y como tal recoge los comandos y las comunicaciones entre el software 50 de aplicación y el sistema 10 operativo real del equipo cliente. Como se ilustra gráficamente por las flechas, ciertos comandos están entre la restricción de sistema operativo y la aplicación de software, esto es a diferencia de las instalaciones típicas en las que estos comandos serían ejecutados en su lugar por el propio sistema operativo, lo que resulta en cambios en el equipo cliente que podrían no ser necesariamente los que el operador ha previsto. Por otro lado, otros comandos pasan a través de la restricción de sistema operativo y a continuación se transfieren al propio sistema operativo.

REIVINDICACIONES

1. Un sistema para crear un entorno de software de aplicación sin cambiar un sistema (10) operativo de un equipo cliente, comprendiendo el sistema una capa (100) de abstracción y protección de sistema operativo,

en el que dicha capa de abstracción y protección está interpuesta entre una aplicación (50, 52, 54) de software en ejecución y dicho sistema operativo, por lo que se proporciona un entorno virtual en el que pueden ejecutarse una aplicación y por lo que dicho sistema operativo es un sistema operativo Windows que comprende un registro de Windows,

en el que la capa de abstracción y protección de sistema operativo responde con una clave y su valor si dicha clave y el valor se almacenan dentro de la capa de abstracción y protección de sistema operativo, si no se almacenan, la capa de abstracción y protección de sistema operativo permite la solicitud para pasar a través del registro de Windows.

2. El sistema de la reivindicación 1, en el que todas las operaciones en el registro de Windows y los archivos .ini son a través de la API de Win32, comprendiendo además el sistema un medio para funciones de interceptación, por lo que cada vez que se invocan dichas funciones otra función o aplicación intercepta la llamada.

3. El sistema de la reivindicación 2, en el que el sistema intercepta cada función API adecuada para dar servicio a una solicitud si se hace por una aplicación ejecutada desde un servidor o si se hace por una aplicación contra una clave de configuración que se administra activamente.

4. El sistema de la reivindicación 1, que comprende además un componente de registro de Windows virtual para proporcionar un registro de función completo a una aplicación, pero evitar la modificación del registro de sistema subyacente.

5. El sistema de la reivindicación 1, en el que si se hace un intento de modificar el valor de dicha clave, la capa de abstracción y protección de sistema operativo permite que la modificación se produzca solamente sobre sí misma.

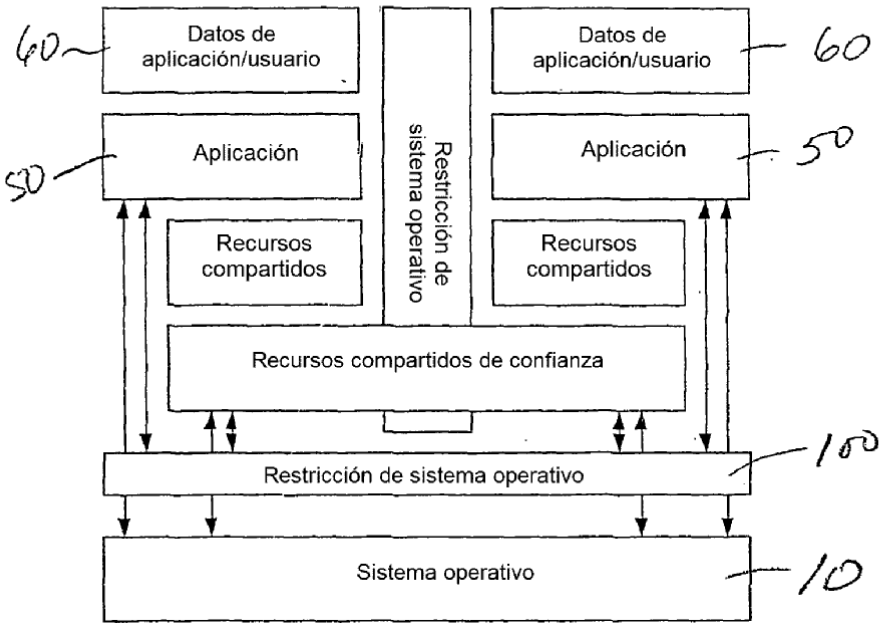
60
Datos de aplicación/usuario
Datos de aplicación/usuario
60
50
Aplicación
Aplicación
50
Restricción de sistema operativo
Recursos compartidos
Recursos compartidos
Recursos compartidos de confianza
100
Restricción de sistema operativo
10
Sistema operativo

FIG. 1

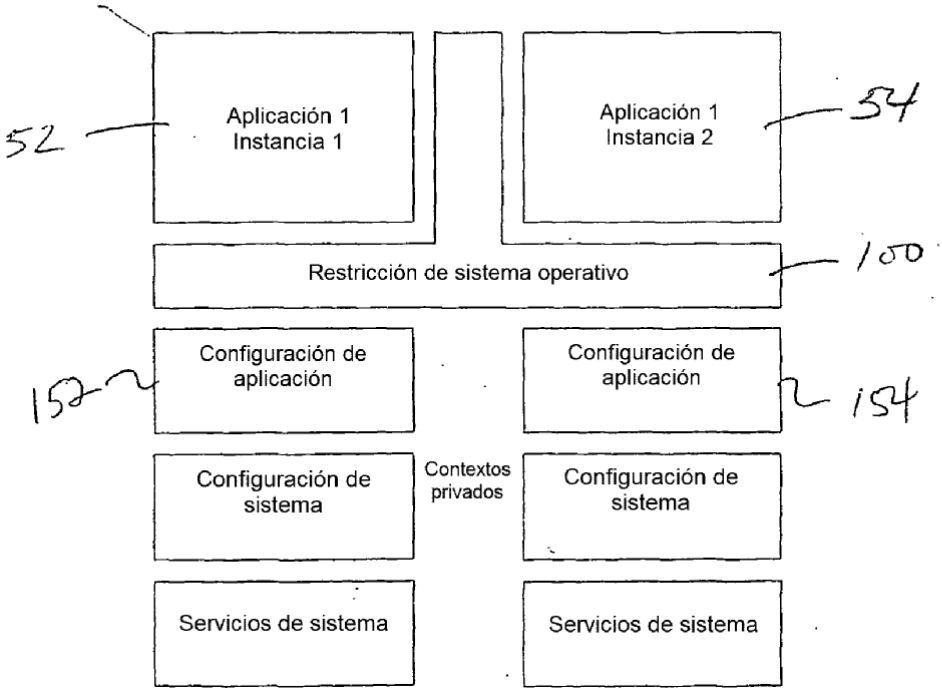
52
Aplicación 1 Instancia 1
Aplicación 1 Instancia 2
54
150
Restricción de sistema operativo
152
Configuración de aplicación
Configuración de aplicación
154
Configuración de sistema
Contextos privados
Configuración de sistema
Servicios de sistema
Servicios de sistema

FIG. 2

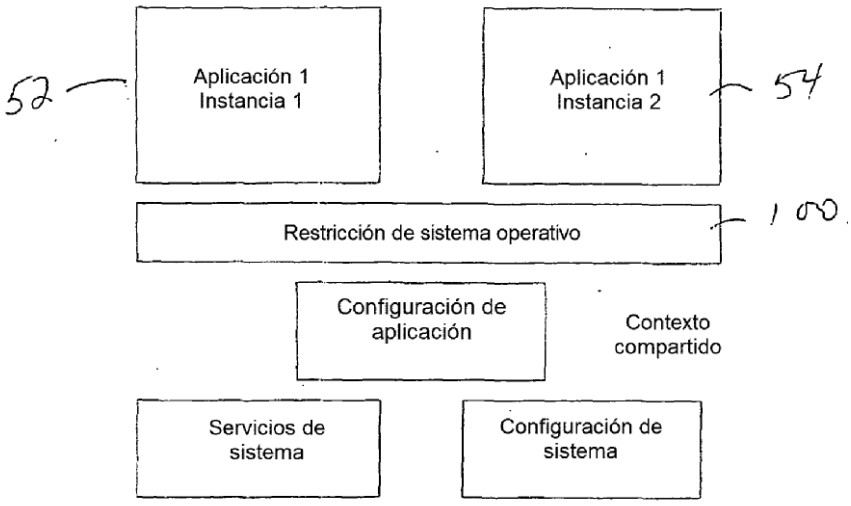
52
Aplicación 1
Instancia 1
Aplicación 1
Instancia 2
54
100
Restricción de sistema operativo
Configuración de aplicación
Contexto compartido
Servicios de sistema
Configuración de sistema

FIG. 3

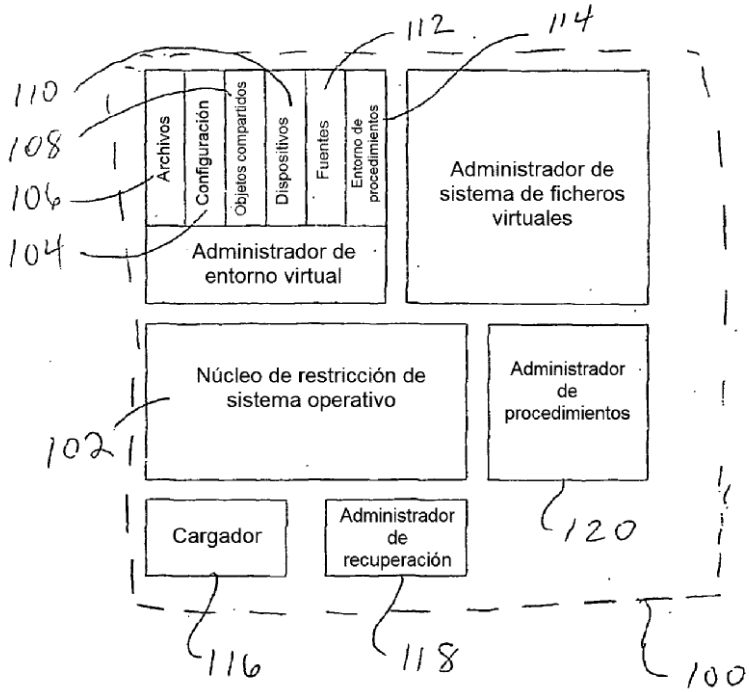
110
108
106
104
112
114
Archivos
Configuración
Objetos compartidos
Dispositivos
Fuentes
Entorno de procedimientos
Administrador de entorno virtual
Administrador de sistema de ficheros virtuales
Núcleo de restricción de sistema operativo
102
Administrador de procedimientos
120
Cargador
Administrador de recuperación
116
118
100

FIG. 4