(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 404 051**

(51) Int. CI.:

G06F 12/02 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **03.08.2009 E 09009988 (8)**

(97) Fecha y número de publicación de la concesión europea: **20.03.2013 EP 2151762**

(54) Título: **Gestión de almacenamiento en un soporte de almacenamiento de datos portátil**

(30) Prioridad:

07.08.2008 DE 102008036873

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
23.05.2013

(73) Titular/es:

GIESECKE & DEVRIENT GMBH (100.0%)
PRINZREGENTENSTRASSE 159
81677 MÜNCHEN, DE

(72) Inventor/es:

KRAMPOSTHUBER, GEORG

(74) Agente/Representante:

ARPE FERNÁNDEZ, Manuel

DESCRIPCIÓN

Gestión de almacenamiento en un soporte de almacenamiento de datos portátil

La presente invención se refiere a un procedimiento para gestionar recursos de memoria en un soporte de datos portátil y a un soporte de datos con una gestión de memoria de este tipo,

Los recursos de memoria de una memoria volátil de un soporte de datos portátil son asignados por una gestión de memoria del soporte de datos a los procesos ejecutados de forma concurrente por un procesador o "hilos de ejecución (threads)" (denominados "procesos de poco peso"), pudiendo consistir un proceso en varios *threads* ejecutables a su vez de forma concurrente.

Para gestionar los recursos de memoria requeridos por un proceso o *thread* se conocen en particular las dos estructuras de memoria abstractas de "pila (stack)" y "montículo (heap)". La "stack", designada en adelante como memoria de pila, representa un área de memoria estructurada de cada proceso o *thread* ejecutado, asignada individualmente a dicho proceso o hilo (*thread)*. Sirve para almacenar variables locales, resultados intermedios o similares, y se gestiona mediante las, así llamadas, operaciones "push" (inserción) y "pop" (extracción). Por regla general, la necesidad de memoria para la memoria de pila de procesos o *threads* ya se conoce durante la compilación del programa correspondiente. En cambio, el montículo o montón (*heap)* representa un área de memoria a la que se le pueden solicitar recursos de memoria durante el tiempo de ejecución de un proceso o *thread*, cuyo tamaño y momento de solicitud todavía no eran conocidos durante la compilación del programa. En principio, un único montículo (*heap)* es suficiente para gestionar los recursos de memoria solicitados por todos los procesos o *threads*. En este escenario, la gestión de memoria de un sistema operativo tiene la misión de proporcionar suficientes recursos de memoria físicos para las estructuras lógicas de la memoria de pila, por ejemplo en la memoria RAM volátil del soporte de datos.

En este contexto resulta ventajoso que la memoria física prevista para una memoria de pila sea esencialmente coherente, ya que de lo contrario no es posible aprovechar las ventajas de la estructura lógica simple de la memoria de pila. Sin embargo, en caso de un sistema operativo *multithread (hilos de ejecución múltiples)*, que puede ejecutar varios procesos/*threads* de forma concurrente, se plantea el problema de que a cada proceso/*thread* se le ha de asignar previamente un área de memoria constante para la memoria de pila correspondiente. Sobre todo en caso de aparatos con recursos de memoria limitados, como por ejemplo tarjetas chip o similares, esto conduce a un consumo de memoria innecesariamente alto, ya que en muy pocos casos todas las memorias de pila utilizan realmente la totalidad del área de memoria asignada.

Por ello, los recursos de memoria para las memorias de pila necesarias se gestionan de forma dinámica (es decir, las áreas de memoria se asignan y liberan sobre demanda, sin que se conozca previamente el momento de la solicitud y/o el tamaño solicitado) asignando a un proceso/*thread* un área de memoria para su memoria de pila únicamente cuando se genera el proceso/*thread*. Sin embargo esto implica una fragmentación progresiva de la memoria, ya que una memoria libre que en principio es suficientemente grande se va dividiendo con el paso del tiempo en numerosas secciones de memoria pequeñas, lo que impide asignar después grandes recursos de memoria coherentes.

Esta fragmentación de la memoria se puede eliminar mediante una "garbage collection" (recuperación de espacio inutilizado), en la que por un lado primero se liberan las áreas de memoria asignadas que ya no son requeridas por los procesos/*threads* correspondientes y por otro lado se compactan al menos partes de la memoria, es decir, los recursos de memoria asignados se "aproximan entre sí" de tal modo que se forman de nuevo áreas de memoria coherentes más grandes. Sin embargo, por regla general la gestión de memoria del sistema operativo realiza la *garbage collection* de forma autónoma sin que puedan influir en ello los procesos/*threads* ejecutados. A causa de ello, un proceso/*thread* puede quedar interrumpido un tiempo excepcionalmente largo durante su ejecución, lo que resulta particularmente desventajoso en relación con los procesos/*threads* que han de satisfacer requisitos en tiempo real.

Por consiguiente, el objetivo de la invención consiste en proponer un procedimiento para la gestión dinámica de recursos de memoria que evite las desventajas anteriormente descritas.

El documento US 4.445.170 da a conocer un procedimiento para la gestión dinámica de recursos de memoria, en el que una primera gestión de memoria gestiona dinámicamente primeras porciones de los recursos de memoria y una segunda gestión de memoria, independiente de la primera gestión de memoria, gestiona dinámicamente segundas porciones de los recursos de memoria diferentes de las primeras porciones, de acuerdo con el preámbulo de la reivindicación 1. La primera gestión de memoria gestiona datos. La segunda gestión de memoria gestiona una memoria de pila (*stack*).

El documento US 5.727.185 da a conocer un procedimiento para la gestión dinámica de la memoria libre de un sistema informático, en el que la memoria libe se divide en dos áreas de memoria lógica con diferentes condiciones de acceso. En las áreas de memoria están almacenados por ejemplo datos de usuario.

Este objetivo se resuelve según la invención mediante un procedimiento y un soporte de datos portátil con las características indicadas en las reivindicaciones independientes. Las reivindicaciones subordinadas a éstas describen configuraciones ventajosas y perfeccionamientos de la invención.

La presente invención se basa en la idea fundamental de gestionar determinadas porciones de los recursos de memoria mediante una segunda gestión de memoria, diferente de la gestión de memoria usual, de tal modo que dichas porciones gestionadas de otro modo ya no están sometidas a la usual *garbage collection*. Por consiguiente, un procedimiento según la invención para la gestión dinámica de recursos de memoria de un soporte de datos se caracteriza porque una primera gestión de memoria gestiona dinámicamente primeras porciones de los recursos de memoria del soporte de datos y una segunda gestión de memoria, independiente de la primera gestión de memoria, gestiona dinámicamente segundas porciones de los recursos de memoria diferentes de las primeras porciones.

De este modo, por un lado se puede poner a disposición una segunda gestión de memoria que está adaptada especialmente a los requisitos de las segundas porciones de memoria o de los procesos/*threads* que las solicitan y que asegura siempre la disposición de recursos de memoria prácticamente en tiempo real. Por otro lado se descarga la *garbage collection*, dado que ésta ya no se ha de encargar de la depuración de toda la memoria, y se mejora el comportamiento del tiempo de ejecución de los procesos/*threads* ejecutados en el procesador del soporte de datos. Dado que tanto la primera como la segunda gestión de memoria gestionan las primeras y las segundas porciones, respectivamente, de forma dinámica, los recursos de memoria del soporte de datos se aprovechan eficazmente y se evita de forma fiable un derroche innecesario de recursos de memoria.

Para una mayor simplicidad y con el fin de facilitar la comprensión de la invención, se supone que la memoria lógica y la memoria física tienen el mismo espacio de dirección y por ello no se ha de hacer ninguna distinción entre memoria lógica y memoria física. Evidentemente, la presente invención también puede ser aplicada cuando los espacios de dirección de la memoria lógica y la memoria física son diferentes, por ejemplo porque la memoria lógica es mayor que la memoria física. Además, en adelante un proceso o un *thread* designa respectivamente un programa ejecutado o una parte de programa ejecutada, que ha sido cargado al menos parcialmente en la memoria física y que es ejecutado por el procesador del soporte de datos. Por lo tanto, siempre que no se indique otra cosa, en el contexto de la presente invención no se ha de hacer ninguna distinción entre procesos y *threads*.

El soporte de datos portátil según la invención incluye recursos de memoria, por ejemplo en forma de una memoria RAM volátil o similar, un procesador, un sistema operativo con un primer dispositivo de gestión de memoria para la gestión dinámica de los recursos de memoria, y al menos una aplicación ejecutable en el procesador como un proceso o como uno o más *threads*. De acuerdo con la invención, el primer dispositivo de memoria está preparado para gestionar dinámicamente primeras porciones de los recursos de memoria. El sistema operativo también incluye un segundo dispositivo de gestión de memoria que es diferente del primer dispositivo de gestión de memoria y que está preparado a su vez para gestionar dinámicamente segundas porciones de los recursos de memoria diferentes de las primeras porciones.

De acuerdo con la invención, la primera y la segunda gestión de memoria se realizan mediante dos dispositivos de gestión de memoria del sistema operativo del soporte de datos separados, que funcionan del modo más independiente posible entre sí. Las primeras y las segundas porciones de los recursos de memoria son gestionadas exclusivamente de manera respectiva por uno de los dos dispositivos de gestión de memoria. En este contexto, cada uno de los dispositivos de gestión de memoria reconoce los recursos de memoria gestionados por el otro dispositivo de gestión de memoria como tales. Los recursos de memoria no gestionados por ninguno de los dos dispositivos de gestión de memoria están a disposición de estos dos dispositivos para nuevas asignaciones a procesos/*threads*. Por consiguiente, en principio cada uno de los dispositivos de gestión de memoria puede acceder a toda la memoria (libre), sin que sea necesario dividir previamente los recursos de memoria en áreas de memoria accesibles exclusivamente para uno de los dispositivos de gestión de memoria respectivamente. De este modo se logra un aprovechamiento óptimo de los recursos de memoria existentes.

Preferentemente, el soporte de datos está configurado como tarjeta chip, tarjeta multimedia segura, tarjeta de telefonía móvil, soporte de memoria USB o similares. El procedimiento según la invención se puede realizar de forma especialmente ventajosa en un soporte de datos de este tipo, que debido a su tipo de construcción en la mayoría de los casos solo dispone de recursos de memoria relativamente pequeños y con frecuencia ha de ejecutar procesos que han de satisfacer requisitos en tiempo real.

Una gestión dinámica de recursos de memoria mediante las dos gestiones de memoria incluye principalmente una asignación de los recursos de memoria libres (al principio accesible para las dos gestiones de memoria) a un proceso ejecutado a demanda de dicho proceso, así como una liberación de los recursos de memoria asignados a este proceso por la gestión de memoria correspondiente. En este contexto, las porciones de los recursos de memoria gestionadas por una de las gestiones de memoria corresponden a los recursos de memoria que dicha gestión de memoria ha asignado y no ha vuelto a liberar.

La gestión dinámica de recursos de memoria mediante una gestión de memoria según la invención puede incluir en particular las siguientes etapas:

- recepción de una solicitud de recursos de memoria con un tamaño definido durante la generación de un proceso/*thread* o emitida por el proceso/*thread* durante su tiempo de ejecución;

- determinación por el dispositivo de gestión de memoria de las porciones libres de los recursos de memoria con el tamaño solicitado;

- asignación de las porciones libres determinadas de los recursos de memoria al proceso/*thread* que las solicitan y marcado de las porciones asignadas como “asignada” o “no libre”;

- recepción de un mensaje de liberación de memoria de un proceso/*thread* en relación con porciones de los recursos de memoria asignadas a dicho proceso/*thread* o reconocimiento por el dispositivo de gestión de memoria de que un proceso/*thread* ya no requiere las porciones de recursos de memoria asignadas al mismo;

- liberación de las porciones correspondientes de los recursos de memoria mediante el marcado de dichas porciones como “asignable” o “libre” por el dispositivo de gestión de memoria.

La gestión dinámica de recursos de memoria mediante uno de los dispositivos de gestión de memoria, puede incluir además una compactación de las porciones de los recursos de memoria gestionadas (fragmentadas) por dicho dispositivo de gestión de memoria, que preferentemente es completamente independiente de la gestión de las otras porciones de los recursos de memoria por el otro dispositivo de gestión de memoria. Preferentemente, las gestiones de memoria reconocen, independientemente entre sí, cuándo las porciones de los recursos de memoria asignadas a un proceso/*thread* ya no son requeridas por este proceso/*thread* y pueden ser liberadas. Después, las porciones de los recursos de memoria liberadas de este modo están de nuevo a disposición de las dos gestiones de memoria para una nueva asignación a procesos/*threads* que las soliciten.

De este modo, los dos dispositivos de gestión de memoria pueden gestionar las porciones de memoria gestionadas respectivamente por los mismos de forma individual y adaptada por ejemplo a los procesos que las solicitan. Mientras que, por ejemplo, para la primera gestión de memoria puede resultar útil y ventajoso depurar los recursos de memoria gestionados solo de forma ocasional mediante una *garbage collection* usual, es decir, liberar los recursos de memoria que ya no son necesarios para los procesos y compactar en caso dado las áreas de memoria fragmentadas, para la segunda gestión de memoria puede resultar ventajoso ejecutar etapas de liberación de la memoria que ya no es necesaria o de compactación de las respectivas áreas de memoria correspondientes directamente mediante una operación de liberación explícita o en la primera ocasión posible.

De acuerdo con la invención, la primera gestión de memoria asigna y gestiona dinámicamente los recursos de memoria solicitados por un proceso/*thread* durante su tiempo de ejecución como primeras porciones (del *heap*). En cambio, la segunda gestión de memoria asigna y gestiona dinámicamente las áreas de memoria solicitadas para una memoria de pila de un proceso/*thread* como segundas porciones. De este modo se logra en particular que las porciones de memoria previstas por la segunda gestión de memoria para una memoria de pila sean independientes de una *garbage collection* de la primera gestión de memoria y, por lo tanto, que puedan ser asignadas siempre por la segunda gestión de memoria de forma esencialmente directa y en particular sin interrupción por una *garbage collection*. De esta forma se puede evitar eficazmente un retardo no deseado de procesos/*threads* que durante su tiempo de ejecución requieren muy poca o ninguna memoria (del *heap*), es decir, que esencialmente solo necesitan la memoria de pila.

De acuerdo con una forma de realización de la presente invención, el segundo dispositivo de gestión de memoria gestiona los recursos de memoria para la memoria de pila en forma de porciones de memoria coherentes de tamaño constante. De este modo se puede evitar una fragmentación externa en la memoria, ya que las áreas de memoria libre siempre presentan el mismo tamaño constante y como las porciones de memoria asignadas por la segunda gestión de memoria. Preferentemente, este tamaño constante presenta una proporción conveniente con respecto al tamaño de los marcos de método archivados (“pushed”) en la memoria de pila, de modo que en un área de memoria de este tipo se puedan archivar respectivamente al menos algunos marcos de método. De este modo se evita en gran medida una fragmentación interna, ya que las áreas de memoria libres coherentes de tamaño constante se llenan, esencialmente por completo, con datos coherentes funcionales de una memoria de pila y no se perturba la ejecución del proceso correspondiente.

De acuerdo con otra forma de realización, la segunda gestión de memoria puede gestionar los recursos de memoria asignados a las memorias de pila en forma de porciones de memoria coherentes con un tamaño que corresponde exactamente al tamaño de un marco de método a archivar en la memoria de pila correspondiente. De este modo se evita por completo una fragmentación interna de las porciones de memoria gestionadas, ya que el tamaño de cada porción de memoria corresponde exactamente al tamaño del marco de método para el que la segunda gestión de memoria pone a disposición porciones de memoria. Sin embargo, dado que estas porciones de memoria presentan por regla general diferentes tamaños, no es posible prescindir por completo de una fragmentación externa. No obstante, como la segunda gestión de memoria preferentemente libera directamente las porciones de los recursos de memoria que ya no son necesarias y realiza directamente una compactación eventual, la fragmentación externa se evita en gran medida.

De acuerdo con la invención, el sistema operativo incluye una máquina virtual *multithread*, por ejemplo una “Máquina Virtual Java”, y el primer y el segundo dispositivo de gestión de memoria están integrados en una gestión de memoria libre de la máquina virtual.

Cuando una máquina virtual preparada según la invención ejecuta al menos un *thread*, el primer dispositivo de gestión de memoria gestiona recursos de memoria para objetos del *thread* referenciados por otros objetos y/o que por su parte referencian otros objetos. En cambio, la segunda gestión de memoria gestiona recursos de memoria para la memoria de pila de los *threads* ejecutados en la máquina virtual.

Además de los recursos de memoria para la memoria de pila, la segunda gestión de memoria también gestiona dinámicamente los recursos de memoria para aquellos objetos del *thread* que no son referenciados o que solo son referenciados desde una memoria de pila y que por su parte no referencian ningún otro objeto. Estos recursos de memoria asignados a objetos se pueden liberar directamente después de la ejecución de sus métodos.

Cuando un objeto de este tipo (no referenciado o referenciado de una memoria de pila) gestionado adicionalmente por la segunda gestión de memoria es referenciado por otro objeto, el segundo dispositivo de gestión de memoria transfiere a la primera gestión de memoria la gestión de los recursos de memoria asignados a dicho objeto. En caso dado, la porción de memoria correspondiente se desplaza en la memoria con fines de compactación.

Otras características y ventajas de la invención se desprenden de la siguiente descripción de ejemplos de realización según la invención y de otras alternativas de realización en relación con los dibujos adjuntos, en los que:

- la figura 1 muestra una forma de realización preferente de un soporte de datos según la invención; y

- la figura 2 muestra una representación esquemática de la ocupación de una memoria RAM de un soporte de datos según la figura 1.

Con referencia a la figura 1, un soporte de datos 10, que en este caso está configurado como una tarjeta chip, incluye una interfaz de comunicación de datos 20, un procesador (CPU) 30 y diferentes memorias 40, 50 y 60. El soporte de datos 10 también puede estar configurado de otro modo, por ejemplo como una ficha USB, una tarjeta de telefonía móvil, una tarjeta multimedia segura o similares.

La interfaz de comunicación de datos 20 está configurada como un campo de contactos según ISO 7816 y sirve para la comunicación de datos con un dispositivo de procesamiento de datos externo (no representado), por ejemplo un terminal de tarjetas, y como fuente de alimentación para el soporte de datos 10. Alternativa o adicionalmente, el soporte de datos 10 también puede disponer de una fuente de alimentación propia (no representada), por ejemplo a través de una batería. También pueden estar previstas otras interfaces por contacto. Alternativa o adicionalmente, también puede estar prevista una interfaz de comunicación de datos sin contacto (no representada) para una comunicación de datos sin contacto, por ejemplo una antena.

En la memoria ROM permanente no regrabable 40 está grabado un sistema operativo *multithread* 42 que controla el soporte de datos 10. El sistema operativo 42 incluye un primer dispositivo de gestión de memoria 44 y un segundo dispositivo de gestión de memoria 46 para la gestión dinámica de la memoria RAM 50 que sirve como memoria volátil principal para el soporte de datos 10. En este contexto, la primera y la segunda gestión de memoria 44, 46 están preparadas en particular de tal modo que

- en la gestión dinámica de recursos de memoria 50 por una de las gestiones de memoria 44, 46 se asignan recursos de memoria libres a un proceso ejecutado en el soporte de datos 10 a demanda del proceso y se liberan recursos de memoria asignados a un proceso, representando las porciones A1, A2, ..., A*n*, B1, B2, ..., B*n* de los recursos de memoria 50 gestionadas por una de las gestiones de memoria 44, 46 los recursos de memoria asignados y todavía no liberados por dicha gestión de memoria 44, 46;

- la gestión dinámica de recursos de memoria 50 por una de las gestiones de memoria 44 incluye una compactación de las porciones A1, A2, ..., A*n* de los recursos de memoria 50 gestionadas por dicha gestión de memoria, siendo dicha compactación independiente de la otra gestión de memoria 46 respectivamente; y

- la primera 44 y la segunda gestión de memoria 46 reconocen, independientemente entre sí de manera respectiva, las porciones A1, A2, ..., A*n*, B1, B2, ..., B*n* de los recursos de memoria 50 asignadas a un proceso y ya no requeridas por éste, y liberan las porciones A1, A2, ..., A*n*, B1, B2, ..., B*n* de los recursos de memoria 50 reconocidas como ya no necesarias.

A continuación se describen más detalladamente dispositivos de gestión de memoria 44, 46 correspondientes con referencia a la figura 2. Alternativamente, el sistema operativo 42, o al menos partes del mismo, como por ejemplo el primer y/o el segundo dispositivo de gestión de memoria 44, 46, pueden estar almacenados en la memoria FLASH regrabable 60, que alternativamente también puede estar configurada como memoria EEPROM o similares. En la memoria FLASH 60 está almacenada al menos una aplicación 64 ejecutable en el procesador 30 como un proceso con varios *threads*.

Durante la ejecución de la aplicación 64 en el procesador 30, las gestiones de memoria 44, 46 asignan al proceso correspondiente recursos de la memoria RAM 50. En este contexto, el primer dispositivo de gestión de memoria 44 gestiona dinámicamente primeras porciones A1, A2, ..., A*n* de los recursos de memoria (que constituyen el área de memoria 54) asignadas a procesos, mientras que el segundo dispositivo de gestión de memoria 46 gestiona dinámicamente, de forma independiente con respecto al primer dispositivo de gestión de memoria 44, segundas porciones B1, B2, ..., B*n* de los recursos de memoria (que constituyen el área de memoria 56) diferentes de las primeras porciones A1, A2, ..., A*n*. El sistema operativo 42 o el núcleo del sistema operativo ocupa el área de memoria 52.

El área de memoria 54 con las porciones A1, A2, ..., A*n* es gestionada exclusivamente por el primer dispositivo de gestión de memoria 44, mientras que el área de memoria 56 con las porciones B1, B2, ..., B*n* es gestionada exclusivamente por el segundo dispositivo de gestión de memoria 46. La gestión dinámica de recursos de memoria en la memoria RAM 50 incluye la recepción de solicitudes de memoria de procesos, la localización de recursos de memoria libres y asignables en la memoria RAM 50 (es decir, de los recursos de memoria que no se encuentran en el área de memoria 52 ni en el área de memoria 54), la asignación de una porción de estos recursos de memoria libres al proceso que la solicita, y el reconocimiento de porciones de memoria que ya no son necesarias y la liberación de dichas porciones. Las porciones de memoria liberadas se pueden asignar después de nuevo adaptadas individualmente.

Tanto el primer dispositivo de gestión de memoria 44 como el segundo 46 tienen acceso a toda la memoria 50, excepto el área de memoria 52 reservada para el sistema operativo 42, es decir, los recursos de memoria restantes no ocupados por el sistema operativo 42 no están divididos previamente o de forma estática entre los dispositivos de gestión de memoria 44, 46. Los dispositivos de gestión de memoria 44, 46 funcionan de forma esencialmente independiente entre sí, es decir, una porción A*i* asignada por el primer dispositivo de gestión de memoria 44 permanece bajo la gestión del primer dispositivo de gestión de memoria 44 hasta que la porción A*i* asignada ya no es necesaria para el proceso correspondiente y es liberada por la gestión de memoria 44. En este contexto, cada uno de los dos dispositivos de gestión de memoria 44, 46 reconoce las porciones de memoria B*i*, A*i* ya asignadas por el otro dispositivo de gestión de memoria 46, 44 correspondiente, es decir, las porciones de memoria ocupadas que actualmente no pueden ser asignadas.

Además, cada dispositivo de gestión de memoria 44, 46 reconoce las porciones de memoria asignadas por el mismo y ya no requeridas por los *threads* del proceso. Después, estas porciones A1, A2, ..., A*n*, B1, B2, ..., B*n* de los recursos de memoria son liberados de nuevo por la gestión de memoria 44, 46 correspondiente, con lo que a partir de ese momento también están disponibles para la otra gestión de memoria 44, 46.

Por otro lado, cada una de las dos gestiones de memoria 44, 46 está preparada para compactar recursos de memoria que han sido asignados por la misma y que están distribuidos de forma no coherente en la memoria 50, para contrarrestar una fragmentación de las áreas de memoria 54, 56. En este contexto, por ejemplo las porciones de memoria A1, A2, ..., A*n,* gestionadas por el primer dispositivo de gestión de memoria 33 y situadas en posiciones dispersas se desplazan en la memoria 50 de tal modo que se forma una o al menos pocas secciones coherentes relativamente grandes de recursos de memoria asignados. Como resultado de ello, la memoria libre asignable también está presente de nuevo de forma esencialmente coherente, con lo que el proceso de gestión de memoria se simplifica y acelera en general y además se posibilita la asignación de grandes áreas de memoria coherentes. La figura 2 muestra una disposición ideal de la memoria con la memoria libre y las áreas de memoria 54, 56 coherentes con el tamaño máximo posible.

Las dos gestiones de memoria 44, 46 también se diferencian por el modo en que reconocen las porciones de los recursos de memoria asignadas y ya no requeridas y por cuándo liberan estas porciones, y también por el modo en que compactan las porciones que no han de ser liberadas.

Al comenzar la aplicación 64 se asignan al proceso correspondiente porciones de memoria de dos áreas de memoria diferentes 54, 56 de la memoria 50. El área de memoria 54 gestionada dinámicamente por el primer dispositivo de gestión de memoria 44 actúa como *heap* para disponer porciones A1, A2, ..., A*n* de los recursos de memoria solicitadas durante el tiempo de ejecución del proceso. El dispositivo de gestión de memoria 44 asigna porciones de memoria A1, A2, ..., A*n* a los procesos y *threads* durante todo el tiempo posible en vista del tamaño de la memoria libre en la memoria 50. En primer lugar solo libera las porciones A1, A2, ..., A*n,* de los recursos de memoria señalados explícitamente por un proceso o *thread* como recursos a liberar.

Si ya no es posible una asignación de porciones de memoria solicitadas, por ejemplo porque el tamaño de los recursos de memoria solicitados sobrepasa el tamaño del área de memoria libre coherente más grande, el dispositivo de gestión de memoria 44 ejecuta una *garbage collection*. En este proceso se detectan los objetos dispuestos en el *heap* 54 que no son referenciados y que, correspondientemente, pueden ser borrados. Existen diferentes procedimientos conocidos para detectar dichos objetos, por ejemplo el procedimiento *mark-and-sweep* (marcado y barrido). Después, el dispositivo de gestión de memoria 44 libera las porciones asignadas a estos objetos en el área de memoria 54 y a continuación puede tener lugar una compactación. Por regla general, el momento exacto de la *garbage collection* no es previsible y depende de los requisitos de memoria concretos de los *threads* del proceso.

El área de memoria 54 gestionada por el primer dispositivo de gestión de memoria 44 comienza en un extremo del espacio de dirección de la memoria 50 disponible para los dos dispositivos de gestión de memoria 44, 46. En cambio, el dispositivo de gestión de memoria 46 con la asignación de áreas de memoria B1, B2, ..., B*n,* comienza en el otro extremo del espacio de dirección en el área de memoria 56 (véanse las flechas en la figura 2). Al principio ninguna de las áreas de memoria 54, 56 está limitada, sino que éstas aumentan o disminuyen en función de las asignaciones y liberaciones de memoria de los dos dispositivos de gestión de memoria 44, 46.

El segundo dispositivo de gestión de memoria 46 gestiona las áreas de memoria B1, B2, ..., B*n,* en el área de memoria 56 que han sido respectivamente asignadas a los *threads* del proceso para la disposición de sus memorias de pila correspondientes, teniendo cada *thread* su propia memoria de pila. Durante la ejecución del *thread*, una memoria de pila se gestiona mediante operaciones “push” y “pop”, estando dispuesto “arriba”, respectivamente sobre la pila correspondiente, el marco de método para el método ejecutado actualmente, que incluye variables locales, resultados intermedios del método y similares. Una vez procesado el método, el marco de método se extrae de la pila (“pop”) y el marco del método subyacente, que ha activado el método recién terminado, pasa a ser el marco de método actual. Si desde este método se inicia otro método, el marco de este otro método se dispone sobre la pila (“push”), con lo que constituye el marco de método actual.

El segundo dispositivo de gestión de memoria 46 asigna las porciones B1, B2, ..., B*n,* de los recursos de memoria para la disposición de las memorias de pila de tal modo que el tamaño de la porción en cuestión corresponde exactamente de manera respectiva al tamaño del marco de método a disponer en dicha porción B1, B2, ..., B*n,* en la memoria de pila correspondiente. De este modo se evita una fragmentación interna de la memoria 50 en el área de memoria 56. Una vez procesado el método al que se le había asignado una porción de los recursos de memoria, la porción de memoria se libera directamente sin que sea necesaria una *garbage collection* independiente. Este proceso puede estar adaptado a las características específicas de la memoria RAM 50 (por ejemplo mediante la utilización de propiedades de *hardware* especiales, códigos de operación de máquina especiales, DMA y similares). Alternativamente, una liberación de memoria de la porción de un método procesado también se puede aplazar mientras haya suficientes recursos de memoria libres, hasta que se puedan liberar simultáneamente porciones de memoria de varios marcos de método ejecutados. Los “huecos” que eventualmente se forman en la memoria de nuevo asignable se cierran inmediatamente mediante una compactación, con lo que prácticamente apenas se producen fragmentaciones externas del área de memoria 56.

Tal como se ha descrito, el tipo de gestión del área de memoria 56 mediante el segundo dispositivo de gestión de memoria 46 se diferencia claramente de la *garbage collection* del primer dispositivo de gestión de memoria 44, pero es descargada por la primera gestión de memoria 44 diferente a ella. Gracias a la gestión de las porciones B1, B2, ..., B*n,* para la memoria de pila, que no está sometida a *garbage collection*, la segunda gestión de memoria 46 siempre puede poner a disposición otras porciones de memoria B1, B2, ..., B*n,* para una memoria de pila prácticamente en tiempo real.

Alternativamente, el segundo dispositivo de gestión de memoria 46 también puede asignar las porciones de memoria B1, B2, ..., B*n,* para la disposición de las memorias de pila para los *threads* de un proceso en forma de porciones de memoria de tamaño constante (denominadas “chunks”). De este modo se puede evitar una fragmentación externa, dado que los huecos formados por la memoria liberada pueden ser asignados de nuevo con el mismo tamaño. En esta forma de realización, la liberación de porciones de memoria que ya no son necesarias se puede llevar a cabo directamente para cada “chunk” o se puede aplazar para realizarla simultáneamente para varios “chunks”. Sin embargo, para la gestión de estas porciones B1, B2, ..., B*n* de los recursos de memoria se requiere información de gestión adicional, por ejemplo en forma de listas concatenadas para seguir los “chunks” adyacentes o similares. Además es importante elegir el tamaño (constante) de los “chunks” de tal modo que en cada caso en un ”chunk” se pueda almacenar al menos un marco de método completo y preferentemente varios marcos de método completos, para no influir negativamente en la ejecución de los *thread* y evitar en gran medida la fragmentación interna.

El sistema operativo 42 incluye una máquina virtual *multithread*, por ejemplo una máquina virtual Java, y los dos dispositivos de gestión de memoria 44, 46 están integrados en la gestión de memoria libre de la máquina virtual. El segundo dispositivo de gestión de memoria 46 gestiona dinámicamente las porciones de memoria B1, B2, ..., B*n,* de los objetos que no son referenciados por otros objetos y que por su parte no referencian ningún otro objeto. Estos objetos temporales son referenciados en todo caso por un marco de método dispuesto sobre una memoria de pila. Por consiguiente, las porciones de memoria B1, B2, ..., B*n,* asignadas a estos objetos se pueden liberar cuando el marco de método referenciado respectivo es retirado (“pop”) de la memoria de pila una vez finalizado el método. Sin embargo, cuando un objeto inicialmente temporal es referenciado por otro objeto, el segundo dispositivo de gestión de memoria 46 transfiere al primer dispositivo de gestión de memoria 44 la gestión de la porción de memoria correspondiente para este objeto.

Exactamente del mismo modo, los dos dispositivos de gestión de memoria 44, 46 pueden realizar una gestión de los recursos de memoria 50 cuando además del proceso para la aplicación 64 también se ejecutan otros procesos de forma concurrente. Las porciones de memoria A1, A2, ..., A*n,* son gestionadas dinámicamente por el primer dispositivo de gestión de memoria 44 en el área de memoria 54, y las porciones de memoria B1, B2, ..., B*n,* para las memorias de pila de los procesos son gestionadas por el dispositivo de gestión de memoria 46 en el área de memoria 56.

Los dispositivos de gestión de memoria 44, 46 se pueden ampliar de tal modo que, además de la gestión de recursos de memoria volátiles, también controlen la eliminación de datos en áreas de memoria permanente, por ejemplo en la memoria FLASH 60. Esto es particularmente importante cuando los recursos de memoria volátil son escasos y/o cuando la memoria principal lógica de un proceso es mayor que la memoria principal física disponible. En este caso, además del área de memoria 54 (el *heap*) en la memoria volátil 50 se dispone un *heap* en la memoria permanente 60 y se establece una diferencia entre objetos almacenados de forma volátil y objetos almacenados de forma permanente. Si se dispone de un *heap* volátil rápido (no representado), en el *heap* volátil rápido se gestionan objetos temporales, mientras que los objetos volátiles y permanentes usuales se gestionan tal como se describe más arriba. En este contexto, un objeto volátil se convierte en permanente cuando es referenciado por un objeto permanente, y un objeto permanente únicamente se puede convertir de nuevo en volátil mediante *garbage collection*. Ni un objeto volátil ni un objeto permanente se pueden convertir de nuevo en un objeto volátil rápido.

Por último, el segundo dispositivo de gestión de memoria 46 también puede gestionar dinámicamente porciones de memoria para elementos nativos, por ejemplo un *buffer* (memoria tampón) de octetos. Estos elementos nativos se pueden referenciar por ejemplo mediante inscripciones en listas previstas para ello. Si falta una inscripción de este tipo, la porción de memoria correspondiente se puede liberar directamente.

REIVINDICACIONES

1. Procedimiento para gestión dinámica de recursos de memoria (50), en el que una primera gestión de memoria (44) gestiona dinámicamente primeras porciones (A1, A2, ..., A*n*) de los recursos de memoria (50) y una segunda gestión de memoria (46) prevista para memorias de pila, independiente de la primera gestión de memoria (44), gestiona dinámicamente segundas porciones (B1, B2, ..., B*n*) de los recursos de memoria (50) diferentes de las primeras porciones (A1, A2, ..., A*n*), **caracterizado porque**

- los recursos de memoria (50) son los de un soporte de datos (10) portátil,

- dos dispositivos de gestión de memoria independientes (44, 46) de un sistema operativo (42) del soporte de datos (10) realizan la primera y la segunda gestión de memoria de tal modo que las primeras porciones (A1, A2, ..., A*n*) y las segundas porciones (B1, B2, ..., B*n*) de los recursos de memoria volátiles (50) del soporte de datos (10) son gestionadas dinámicamente de forma exclusiva por el primer (44) o el segundo dispositivo de gestión de memoria (46), respectivamente,

- el primer dispositivo de gestión de memoria (44) y el segundo dispositivo de gestión de memoria (46) están integrados en una máquina virtual de hilo de ejecución múltiple (*multithread)* del sistema operativo (42) y gestionan dinámicamente los recursos de memoria (50) como componentes de una gestión de memoria libre de la máquina virtual,

- la máquina virtual ejecuta al menos un hilo de ejecución (*thread)* y el primer dispositivo de gestión de memoria (44) gestiona dinámicamente primeras porciones (A1, A2, ..., A*n*) de los recursos de memoria (50) para aquellos objetos del hilo o hilos de ejecución (*thread*/*threads)* que son referenciados por otros objetos y/o que por su parte referencian otros objetos, y el segundo dispositivo de gestión de memoria (46) gestiona dinámicamente segundas porciones (B1, B2, ..., B*n*) de los recursos de memoria (50) para memorias de pila de los objetos del hilo o hilos de ejecución (*thread*/*threads)*, y

- el segundo dispositivo de memoria (46) también gestiona dinámicamente segundas porciones (B1, B2, ..., B*n*) de los recursos de memoria (50) para aquellos objetos del hilo o hilos de ejecución (*thread/threads)* que no son referenciados o que solo son referenciados desde una memoria de pila.

2. Procedimiento según la reivindicación 1, **caracterizado porque** los recursos de memoria (50) solicitados por un proceso durante su tiempo de ejecución son asignados por la primera gestión de memoria (44) como primeras porciones (A1, A2, ..., A*n*) de los recursos de memoria (50), y porque los recursos de memoria (50) solicitados para una memoria de pila del proceso son asignados por la segunda gestión de memoria (46) como segundas porciones (B1, B2, ..., B*n*).

3. Procedimiento según la reivindicación 2, **caracterizado porque** la segunda gestión de memoria (46) gestiona recursos de memoria (50) para la memoria de pila en forma de porciones (B1, B2, ..., B*n*) de los recursos de memoria (50) de tamaño constante.

4. Procedimiento según la reivindicación 3, **caracterizado porque** la segunda gestión de memoria (46) gestiona recursos de memoria (50) para la memoria de pila en forma de porciones (B1, B2, ..., B*n*) de los recursos de memoria (50), cuyo tamaño corresponde al respectivo tamaño de un marco de método almacenado en la memoria de pila.

5. Procedimiento según una de las reivindicaciones 1 a 4, **caracterizado porque** el segundo dispositivo de gestión de memoria (46) transfiere al primer dispositivo de gestión de memoria (44) la gestión de las segundas porciones (B1, B2, ..., B*n*) de los recursos de memoria (50) de un objeto no referenciado o referenciado únicamente desde una memoria de pila, cuando dicho objeto es referenciado por otro objeto.

6. Procedimiento según una de las reivindicaciones 1 a 5, **caracterizado porque** el segundo dispositivo de gestión de memoria (46) gestiona dinámicamente segundas porciones (B1, B2, ..., B*n*) de los recursos de memoria (50) de un elemento nativo.

7. Soporte de datos portátil (10), que incluye recursos de memoria (50), un procesador (30), un sistema operativo (42) con un primer dispositivo de gestión de memoria (44) para la gestión dinámica de recursos de memoria (50), y al menos una aplicación (64) ejecutable en el procesador (30) como un proceso, y un segundo dispositivo de gestión de memoria (46) del sistema operativo (42) independiente del primer dispositivo de gestión de memoria (44), estando preparado el primer dispositivo de gestión de memoria (44) para gestionar dinámicamente primeras porciones (A1, A2, ..., A*n*) de los recursos de memoria, y estando preparado el segundo dispositivo de gestión de memoria (46) para gestionar dinámicamente segundas porciones (B1, B2, ..., B*n*) de los recursos de memoria (50) diferentes de las primeras porciones (A1, A2, ..., A*n*), **caracterizado porque** el primer (44) y el segundo dispositivo de gestión de memoria (46) están preparados para gestionar dinámicamente recursos de memoria (50) de acuerdo con un procedimiento según una de las reivindicaciones 1 a 6.

8. Soporte de datos (10) según la reivindicación 7, **caracterizado porque** está configurado como tarjeta chip, tarjeta multimedia segura, tarjeta de telefonía móvil o soporte de memoria USB.

FIG 1

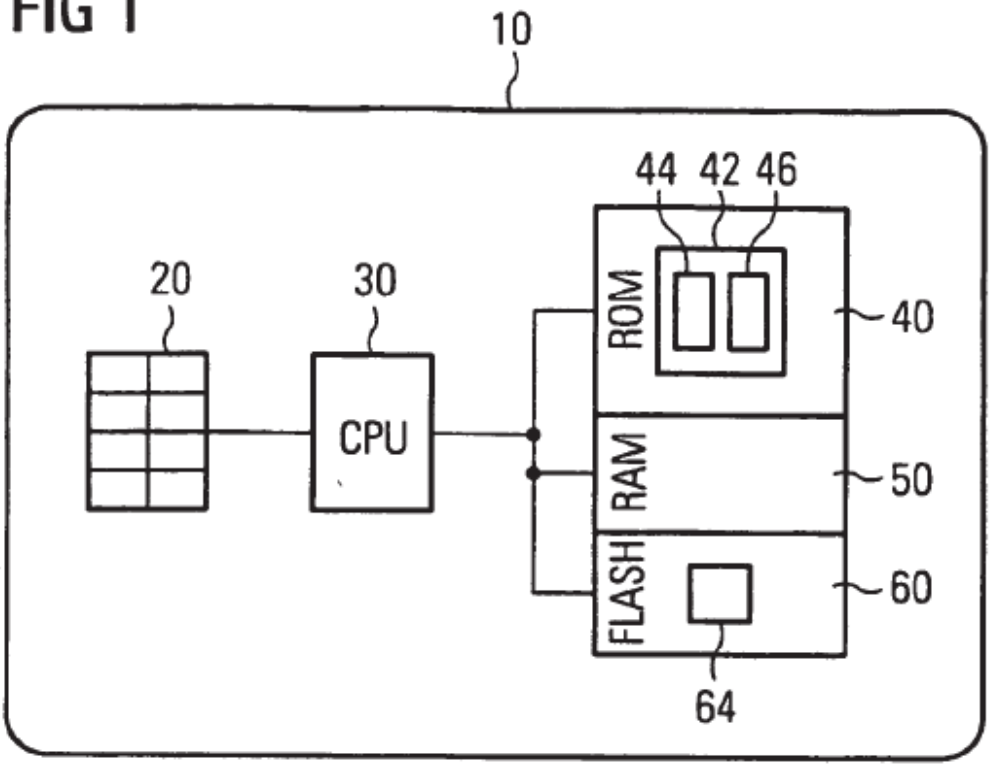
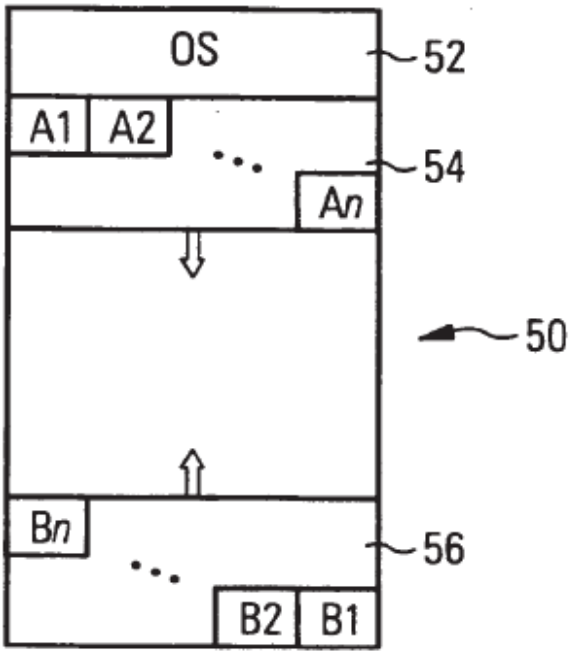

10
44
42
46
20
30
ROM
40
CPU
RAM
50
FLASH
60
64

FIG 2


OS
52
A1
A2
54
An
50
Bn
56
B2
B1

REFERENCIAS CITADAS EN LA DESCRIPCIÓN

La lista de referencias citada por el solicitante lo es solamente para utilidad del lector, no formando parte de los documentos de patente europeos. Aún cuando las referencias han sido cuidadosamente recopiladas, no pueden excluirse errores u omisiones y la OEP rechaza toda responsabilidad a este respecto.

Documentos de patente citados en la descripción

- US 4445170 A **[0008]**
- US 5727185 A **[0009]**