Sistemas y Aplicaciones Informáticas

Tema 17. Sistemas Operativos: Gestión de Memoria.

1. Ámbito de docencia.

- Sistemas informáticos monousuario y multiusuario (ASI 1).
- Sistemas informáticos multiusuario y en red (DAI 1).
- Sistemas operativos en entornos monousuario y multiusuario (ESI 1).

2. Gestión de memoria.

2.1. Concepto de memoria. Localidad de referencias. Jerarquía de memoria.

- La memoria es el componente del ordenador que se encarga de almacenar información, instrucciones y datos en bloques a los cuales se puede hacer referencia a través de una dirección. Existe la posibilidad de realizar dos tipos de operaciones con ella: lectura y escritura.
- Según el principio de localidad de referencias, existe una alta probabilidad de que dos referencias consecutivas en el tiempo se encuentren alojadas en posiciones relativamente próximas en memoria. Además, una referencia reciente tiene cierta probabilidad de ser referenciada de nuevo. Por tanto, es posible organizar los datos en una serie de niveles para que el porcentaje de accesos a niveles inmediatamente inferiores sea menor que el del nivel superior.
- La memoria de un sistema se distribuye jerárquicamente para optimizar la relación que existe entre el coste, tiempo de acceso y tamaño de la memoria. A medida que se desciende en la jerarquía se produce una disminución del coste por bit, un aumento de la capacidad, un aumento del tiempo de acceso y una disminución de frecuencia de acceso por parte del procesador.
- Según esta jerarquía, se pueden distinguir en orden descendente varios tipos de memoria:
  - *Registro*. Es una memoria volátil integrada en la CPU de reducida capacidad y alta velocidad. Permite almacenar y acceder rápidamente a la información más utilizada por el procesador, como direcciones de memoria y los resultados de operaciones matemáticas.
  - *Memoria caché*. Es una memoria volátil situada entre el procesador y la memoria principal de reducida capacidad y alta velocidad. Almacena la información de la memoria principal más frecuentemente utilizada por la CPU para agilizar la transmisión de datos entre ambos. Existe memoria caché integrada en el procesador (L1) y externa al procesador (L2).
  - *Memoria principal*. Es una memoria volátil externa al procesador que contiene la información correspondiente a los procesos existentes en el sistema. Para que un proceso pueda ejecutarse debe estar ubicado en la memoria principal del ordenador.
  - *Caché de disco*. Es una parte de la memoria principal implementada por software, o puede estar dentro del soporte de almacenamiento, que guarda la información más recientemente leída de la memoria secundaria para acelerar la velocidad de acceso a los datos de la misma.
  - *Memoria secundaria*. Formada por un conjunto de dispositivos externos al procesador que permiten almacenar de manera no volátil una gran cantidad de información. Están vinculados a la memoria principal, y el más importante de ellos es el disco duro.

2.2. Imagen de un proceso. Direcciones lógicas y físicas. Relocalización.

- La imagen de un proceso en memoria está formada por las siguientes zonas:
  - *Código*. Área de tamaño fijo que guarda las instrucciones en código máquina del proceso.
  - *Datos*. Espacio de tamaño fijo que almacena las variables y constantes globales del proceso.

    - *Pila*. Área de tamaño variable que guarda los parámetros y las variables locales de los procedimientos y las funciones, y la información de control de retorno. Si el proceso tiene más de un hilo, habrá una pila por cada uno de ellos.
    - *Heap*. Espacio de tamaño variable que almacena las variables creadas dinámicamente. Normalmente la suma del espacio ocupado por la pila y el heap es constante, es decir, cuando la pila crece el heap disminuye y viceversa.
- Internamente los programas contienen referencias a las direcciones de memoria de instrucciones, rutinas y variables. Estas referencias forman el espacio de direcciones lógicas del programa, siendo la más baja la dirección cero. Cuando un programa se ejecuta, el sistema operativo le asigna una zona de memoria que forma el espacio de direcciones físicas del programa.
- La correspondencia entre direcciones lógicas y físicas se denomina relocalización, y puede ser:
    - *Estática*. El proceso se sitúa en una zona fija y concreta de memoria. Puede hacerse:
        - **En tiempo de compilación**. Se genera código absoluto con direcciones físicas.
        - **En tiempo de carga**. El compilador genera código relocalizable con direcciones lógicas, y el cargador del sistema operativo le asigna direcciones físicas.
    - *Dinámica*. Permite que un proceso o parte de él pueda asignarse a distintas zonas de memoria. El compilador genera código relocalizable con direcciones lógicas, y durante la ejecución se traducen a direcciones físicas en el momento en que se referencian.

2.3. Multiprogramación. Protección de la memoria. Registros base y límite.

- La multiprogramación consiste en mantener varios procesos independientes de manera simultánea en la memoria principal. El procesador divide su tiempo entre ellos para que siempre haya un proceso en ejecución, evitando la espera por operaciones de entrada/salida.
- Al existir varios procesos en ejecución, en los sistemas multiprogramados es necesario evitar que entre ellos puedan modificar los espacios de memoria que tienen reservados, incluyendo el sistema operativo. Para ello, el procesador tiene los siguientes registros:
    - *Registro base*. Guarda la dirección física más baja asignada a un determinado proceso.
    - *Registro límite*. Almacena el tamaño máximo permitido a un determinado proceso.
- El sistema operativo se encarga de escribir en estos registros y en el bloque de control de proceso los valores adecuados antes de transferir el control al proceso, con el fin de mantener la integridad del propio sistema operativo y la independencia en la ejecución de los procesos.
- En los sistemas multiprogramados con relocalización dinámica, cuando el programa emplea una dirección lógica, el hardware comprueba que sea inferior al límite. Si es así, a dicha dirección lógica le añade el valor del registro base para obtener la dirección física que se emplea para acceder a memoria. En caso contrario se genera una excepción.

2.4. Intercambio. Archivo de intercambio. Intercambiador.

- El intercambio consiste en retirar de la memoria principal los procesos suspendidos y llevarlos al disco. También consiste en cargar del disco a la memoria principal procesos para su ejecución.
- Para ello se dispone de un archivo de intercambio en el que se almacenan las imágenes dinámicas de los procesos. Puede haber un archivo de intercambio global del sistema, y también archivos de intercambio dedicados, uno por cada proceso.

- El intercambiador es la parte del sistema operativo que gestiona las operaciones de intercambio. Trabaja con el planificador a medio plazo suspendiendo y reanudando procesos. El planificador establece la política y el intercambiador implementa el mecanismo para llevarla a cabo.

2.5. Gestor de memoria. Funciones. Tipos de asignación de memoria.

- El gestor de memoria es la parte del sistema operativo que administra la memoria utilizada por los procesos. Se encarga de lo siguiente:
  - Reservar y liberar zonas de memoria según se solicite.
  - Determinar las zonas de memoria libres y ocupadas.
  - Controlar el acceso y proteger la memoria frente a otros procesos.
  - Utilizar el disco como una extensión de la memoria principal (memoria virtual).
- Existen dos tipos de asignación de memoria:
  - *Contigua*. Los procesos se ubican en su totalidad en posiciones consecutivas.
  - *No contigua*. Los procesos se dividen en bloques situados en posiciones no consecutivas.

3. Gestión de memoria real.

3.1. Asignación contigua.

3.1.1. Sistemas monoprogramados.

- En estos sistemas sólo existe un proceso de usuario en cada momento, que disfruta de todos los recursos del ordenador, y que comparte la memoria con los procesos del sistema operativo. Las zonas de memoria están perfectamente definidas:
  - *Área de monitor residente*. Asignada permanentemente a la parte del sistema operativo que debe estar residente en memoria, y situada habitualmente en la parte inferior.
  - *Área de procesos de usuario*. Contiene un proceso de usuario, que es ejecutado y eliminado al finalizar, para a continuación cargar y ejecutar el proceso siguiente.
- Cuando se carga un proceso en memoria, el sistema operativo se asegura de que sus direcciones estén dentro del área de procesos de usuario utilizando un registro límite integrado en la CPU, que contiene la dirección de inicio de carga del sistema operativo.

3.1.2. Sistemas multiprogramados.

3.1.2.1. Particiones estáticas. Asignación de procesos. Fragmentación interna.

- Consiste en dividir la memoria en varias zonas fijas de tamaño diferente antes de ejecutar los programas de usuario. En cada partición existe un único proceso con tamaño menor o igual al de la partición. Por tanto, el número de particiones determina el grado de multiprogramación.
- La información de cada partición se recoge en una estructura de datos denominada tabla de descripción de particiones. Cada una está descrita por su dirección base, su tamaño y su estado. La dirección base y el tamaño son constantes, y el estado puede ser libre o asignada.
- Existen dos maneras de asignar los procesos a las particiones:
  - *Carga absoluta*. Cada partición tiene asignada una cola de procesos, y cada proceso se envía a la cola de la partición correspondiente a sus exigencias de memoria. Los procesos en ejecución permanecen en la partición hasta su finalización. Esto significa que si un proceso tiene asignada una partición ocupada, debe esperar aunque estén libres otras particiones.

    - *Carga relocalizable*. Los procesos pueden ejecutarse en cualquier partición disponible de tamaño suficiente. Si no existe ninguna el proceso debe esperar. Existen varias estrategias:
        - **Primer ajuste**. En la primera partición que se encuentre en la que quepa el proceso.
        - **Mejor ajuste**. En la partición más pequeña que exista en la que quepa el proceso.
        - **Peor ajuste**. En la partición más grande que exista.
- Las particiones estáticas tienen el inconveniente de la fragmentación interna, que se produce porque dentro de cada partición queda una zona de memoria no aprovechable, debido a que el proceso es más pequeño que la partición y no se puede asignar a ningún otro proceso.

3.1.2.2. Particiones dinámicas. Asignación de procesos. Fragmentación externa y compactación.

- Consiste en asignar a cada uno de los procesos exactamente la memoria contigua que necesita. Para ello se intenta localizar un área libre de memoria que sea igual o mayor que el tamaño del proceso, si existe se crea una partición del mismo tamaño. Si no existe, el proceso debe esperar.
- Cuando el proceso finaliza, su partición se libera y se une en un solo hueco con la memoria libre adyacente. Por tanto, las particiones son variables en número y tamaño. La tabla de descripción de particiones sólo almacena la información de las particiones creadas y asignadas, mientras que la información de los huecos libres se guarda en una lista enlazada.
- Los algoritmos más habituales para la selección de una zona libre de memoria son los siguientes:
    - *Primer ajuste*. Se selecciona el primer hueco de la lista lo suficientemente grande como para alojar el proceso. Suele ser el que mejores resultados ofrece.
    - *Siguiente ajuste*. Como el anterior, comenzando la búsqueda en el lugar donde se dejó.
    - *Mejor ajuste*. Se selecciona el hueco que tiene el tamaño más parecido al del proceso.
    - *Peor ajuste*. Se selecciona el hueco más grande independientemente del tamaño del proceso.
- El problema de las particiones dinámicas es la fragmentación externa por el no aprovechamiento de los huecos libres de memoria entre particiones asignadas, debido a que estos huecos libres no pueden utilizarse porque son muy pequeños para cualquiera de los procesos que esperan.
- Una posible solución a la fragmentación externa es la compactación de la memoria, que consiste en mover todos los procesos hacia la parte inferior o superior de la memoria de forma que es posible combinar todos los huecos libres pequeños en uno grande. Hay que tener en cuenta que el sistema debe detener todas sus actividades mientras realiza la compactación.

3.2. Asignación no contigua.

3.2.1. Paginación.

3.2.1.1. Descripción. Fragmentación interna. Estructuras de datos utilizadas.

- La paginación es una técnica que permite dividir a los procesos en partes, ocupando varias zonas dispersas de la memoria. Consiste en lo siguiente:
    - Se divide el espacio de direcciones lógicas de los procesos en bloques idénticos de **tamaño fijo** denominados páginas. Las direcciones lógicas de los procesos están formadas por el número de página y el desplazamiento dentro de la página.
    - A su vez, la memoria principal se divide en bloques del mismo tamaño que las páginas denominados marcos de página. Las páginas se forman **al cargar un proceso en memoria**, asignándolas a los marcos de página libres, los cuales no tienen por qué ser consecutivos.

- La fragmentación externa se elimina. Sin embargo presenta fragmentación interna ya que la unidad mínima de asignación es la página, y por tanto se desperdicia como media la mitad de una página por asignación. El tamaño de página suele estar impuesto por el hardware.
- Las estructuras de datos utilizadas en la paginación son las siguientes:
  - *Tabla de páginas*. Relaciona las páginas de un proceso con los marcos de página donde se almacenan. Existe una tabla de páginas por cada proceso y una entrada en la tabla por cada página del proceso. Cada entrada de la tabla contiene el número de página y el número de marco de página donde se almacena, además de los **permisos de acceso (rwx)**. Normalmente estas tablas se almacenan en memoria principal. El procesador dispone de los siguientes registros, cuyo contenido se guarda para cada proceso en su respectivo PCB:
    - Registro base de la tabla de páginas. Contiene la dirección de comienzo de la tabla.
    - Registro de longitud de la tabla de páginas. Contiene el tamaño de la tabla.
  - *Tabla de marcos*. Tiene una entrada para cada marco de página, que indica si está libre o asignado y, si está asignado, a qué página de qué proceso.

3.2.1.2. Traducción de direcciones. Memoria compartida.

- Puesto que el tamaño de las páginas y de los marcos de página es idéntico, puede establecerse una correspondencia entre los números de las páginas de un proceso y los números de los marcos de página donde se alojan. Esta relación se almacena en la tabla de páginas del proceso. Por tanto, la traducción de direcciones lógicas en físicas se realiza a partir de la dirección lógica, de la cual se obtiene el número de página. Con este número se accede a la tabla de páginas para localizar el número de marco donde está la página. Para calcular la dirección física basta concatenar el número de marco con el desplazamiento de la dirección lógica.
- La memoria compartida entre procesos se realiza con la página como unidad de compartición. Se lleva a cabo haciendo que varias páginas apunten al mismo marco de página. Debe ser controlado para evitar que un proceso modifique código y datos que otro proceso está leyendo. Para ello, los procesos se dividen en áreas separadas de código y datos, que para poder ser compartidos deben ser no modificables.

3.2.2. Segmentación.

3.2.2.1. Descripción. Fragmentación externa. Estructuras de datos utilizadas.

- La segmentación es una técnica que también permite dividir a los procesos en partes, ocupando varias zonas dispersas de la memoria. Consiste en lo siguiente:
  - Se divide el espacio de direcciones lógicas de los procesos en bloques de **tamaño variable** denominados segmentos. Las direcciones lógicas de los procesos están formadas por el número de segmento y el desplazamiento dentro del segmento.
  - Los segmentos se forman **en tiempo de traducción del programa** mediante la agrupación de elementos relacionados lógicamente como código, datos y pila. Cada uno de ellos se compila comenzando por la dirección lógica cero, y se asigna a una zona libre de memoria.
- La fragmentación interna se elimina. Sin embargo presenta el problema de la fragmentación externa provocada por el no aprovechamiento de los huecos libres de memoria entre segmentos. Si el tamaño medio de segmento es pequeño, la fragmentación externa también será pequeña.

- Las estructuras de datos utilizadas en la segmentación son las siguientes:
  - *Tabla de segmentos*. Relaciona los segmentos de un proceso con las direcciones físicas de memoria donde se almacenan. Existe una tabla de segmentos por cada proceso y una entrada en la tabla por cada segmento del proceso. Puede existir otra tabla común con los segmentos utilizados por todos los procesos. Cada entrada de la tabla contiene el número de segmento y la dirección base de la memoria donde se guarda, además de los **permisos de acceso (rwx)** y del **tamaño del segmento**. Normalmente estas tablas se almacenan en memoria principal. El procesador dispone de los siguientes registros, cuyo contenido se guarda para cada proceso en su respectivo PCB:
    - Registro base de la tabla de segmentos. Contiene la dirección de comienzo de la tabla.
    - Registro de longitud de la tabla de segmentos. Contiene el tamaño de la tabla.
  - *Lista de segmentos libres*. Se trata de una lista enlazada con información de las zonas libres.

3.2.2.2. Traducción de direcciones. Memoria compartida.

- No existe una correspondencia directa entre segmentos y memoria. Por tanto, la traducción de direcciones lógicas en físicas se lleva a cabo dinámicamente en cada referencia a memoria mediante un hardware especial. Se realiza a partir de la dirección lógica, de la cual se obtiene el número de segmento. Con este número se accede a la tabla de segmentos para obtener la dirección base y el tamaño del segmento. Para calcular la dirección física basta añadir el desplazamiento a la dirección base del segmento. Este desplazamiento no debe sobrepasar el tamaño máximo del segmento, en caso contrario se produciría una excepción.
- La segmentación permite que varios procesos compartan áreas de memoria física. Para el acceso a la memoria compartida, los procesos pueden utilizar direcciones lógicas diferentes e incluso segmentos diferentes. Los segmentos son compartidos cuando la entrada a la tabla de segmentos de dos procesos apuntan a la misma dirección base. Las referencias dentro de la zona compartida deben ser relativas al segmento actual, y el código puede compartirse en modo de sólo lectura.

3.2.3. Segmentación paginada.

- Es una técnica que divide el espacio de direcciones lógicas de los procesos en segmentos, y a su vez divide cada segmento en páginas. Aprovecha las ventajas de ambos enfoques:
  - La paginación administra mejor la memoria, debido a que elimina la fragmentación externa y a que la fragmentación interna se controla con un tamaño de página adecuado.
  - La segmentación permite un esquema de protección más flexible, debido al tamaño variable de los segmentos y a que se acopla mejor a la manera en que se estructuran los programas.
- Las direcciones lógicas están formadas por el número de segmento, el número de página dentro del segmento y el desplazamiento dentro de la página. El tamaño de cada segmento debe ser múltiplo del tamaño de una página. De esta manera se elimina la fragmentación externa, pero se introduce fragmentación interna, aunque en menor medida que en la paginación pura.
- Cada segmento tiene una tabla de páginas, y cada proceso tiene una tabla de segmentos y varias tablas de páginas. Esto significa que se aumenta el espacio de memoria ocupado por las tablas.
- La memoria compartida se implementa disponiendo entradas en las tablas de segmentos para diferentes procesos que apunten a la misma tabla de páginas.

4. Gestión de memoria virtual.

4.1. Límite de la memoria real. Memoria virtual. Características.

- Los modelos de gestión de memoria real necesitan que un proceso se encuentre en su totalidad en memoria para poder ser ejecutado. Por tanto, el tamaño de los programas y las posibilidades de multiprogramación están limitados por el tamaño de la memoria física.
- La memoria virtual es una técnica de gestión de memoria basada en el principio de localidad de referencias de los procesos. Se caracteriza por lo siguiente:
  - Permite crear espacios de direcciones lógicas de tamaño superior al de la memoria real utilizando una parte del almacenamiento secundario como una extensión de dicha memoria.
  - Sólo está limitada por el espacio que es capaz de direccionar el procesador y por el espacio disponible en el almacenamiento secundario, que normalmente es el disco.
  - El espacio lógico de direcciones pasa a ser espacio virtual de direcciones, y los procesos pueden utilizar direcciones de memoria o de disco. La traducción de direcciones lógicas a físicas se realiza de la misma manera que en los modelos de gestión de memoria real.
  - Sólo se encuentran en memoria aquellas partes del proceso que están siendo referenciados actualmente, el resto permanece en disco. Se utiliza la técnica de intercambio para mover partes de los procesos de memoria a disco y viceversa.

4.2. Implementación. Gestión de páginas.

- La implementación de la memoria virtual se realiza de la siguiente manera:
  - La técnica más utilizada es la paginación, puesto que evita el problema de la fragmentación externa del espacio de disco. Cada entrada de la tabla de páginas contiene más información:
    - **Bit de presencia**. Indica si el elemento referenciado está en memoria o en disco.
    - **Bit de acceso**. Indica si se ha accedido al elemento referenciado.
    - **Bit de modificación**. Indica si se ha modificado el elemento referenciado.
  - Cuando un proceso intenta acceder a una página que según el bit de presencia no está en memoria principal, se suspende su ejecución y se produce un fallo de página que provoca una excepción. La rutina de tratamiento de la excepción realiza lo siguiente:
    - Comprueba en la tabla de páginas si el acceso es ilegal o legal. En el primer caso finaliza el proceso, en el segundo caso se procede a buscar la página en el disco.
    - Se busca un marco de página libre en la tabla de marcos y se programa la lectura de la página del disco. Si no hay ninguno libre se reemplaza un marco ocupado.
    - Al terminar se actualiza la tabla de páginas del proceso y se planifica la ejecución del proceso para reanudarse en la instrucción que causó el fallo de página.
- El sistema de gestión de páginas de la memoria virtual debe resolver las siguientes cuestiones:
  - *Reemplazo de páginas.* Debe decidir cúal es la página a sustituir cuando se produce un fallo de página y todos los marcos están ocupados. La pagina desalojada puede ser del proceso en cuestión (sustitución local) o de cualquier otro proceso (sustitución global). Algunos de los algoritmos de sustitución a adoptar son los siguientes:
    - **Menos recientemente usada (LRU)**. Reemplaza la página que más tiempo hace que ha sido referenciada. Su implementación eficiente es difícil y requiere apoyo del hardware.

- **No recientemente usada (NRU)**. Reemplaza alguna de las páginas que no hayan sido recientemente utilizadas. Utiliza el bit de acceso de cada entrada de la tabla de páginas.
- **Primera en llegar, primera en salir (FIFO)**. Reemplaza la página que ha estado más tiempo en memoria. Es sencillo de implementar, pero su rendimiento es pobre.

* *Asignación de marcos*. Debe determinar la cantidad de marcos de página que se deben asignar a cada proceso. La hiperpaginación se produce cuando un proceso tiene pocos marcos asignados, y cada fallo de página reemplaza una página que va a ser solicitada de nuevo. Entonces el proceso está más tiempo trayendo páginas del disco que ejecutando instrucciones. Para evitarlo es necesario asignar un número mínimo de marcos a los procesos. Pero una asignación excesiva reduce el número de procesos activos que pueden coexistir simultáneamente. Hay dos tipos posibles de asignación:
  - **Asignación Fija**. El número de marcos asignados a un proceso se fija en el momento de la creación y permanece igual durante la vida del proceso.
  - **Asignación Variable**. El número de marcos asignados a un proceso varía dinámicamente en función del conjunto de páginas que en cada instante debe estar en memoria principal para que no se produzcan fallos de página (conjunto de trabajo).
* *Acceso a páginas*. Debe decidir cuándo cargar una página en la memoria principal:
  - **Acceso por demanda**. Las páginas sólo se cargan cuando son explícitamente referenciadas. De esta manera, sólo se transfieren las páginas estrictamente necesarias, pero se necesita un tiempo de espera para la resolución del fallo de página.
  - **Acceso anticipado**. Se utiliza el principio de localidad para llevar a memoria aquellas páginas que tienen una probabilidad más alta de ser referenciadas, antes de que esto se produzca. Esto hace que se reduzca el tiempo de ejecución de un proceso.

5. Ejemplos de sistemas operativos.

5.1. Gestión de memoria en Windows XP.

- La gestión de memoria la lleva a cabo el Gestor de Máquina Virtual (VMM) del Executive, y por tanto se ejecuta en modo kernel. Utiliza memoria virtual con paginación bajo demanda, siendo el tamaño de página dependiente de la arquitectura del procesador.
- En la versión de 32 bits cada proceso de usuario, independientemente de los hilos que tenga, tiene su propio espacio de direcciones virtuales de 4 GB, de los cuales los primeros 2 GB están disponibles para el código y los datos del proceso. El resto está reservado para el sistema operativo y está compartido por todos los procesos de usuario, aunque sólo accesible a través de llamadas al sistema. Se incluye el sistema operativo dentro del espacio virtual de direcciones del proceso para que las llamadas al sistemas puedan ser atendidas más rápido.
- Cuando un proceso comienza su ejecución, ninguna de sus páginas se encuentra en memoria. Según se suceden los fallos de página se llevan a memoria las páginas a demanda. El número de marcos asignados se determina dinámicamente en función del conjunto de trabajo del proceso.
- El sistema intenta mantener un numero suficiente de marcos libres en memoria para evitar en lo posible el reemplazo de páginas cuando ocurre un fallo de página. En caso de producirse una sustitución de páginas, se elige como víctima la menos recientemente usada del proceso.

5.2. Gestión de memoria en GNU/Linux.

- La gestión de memoria la lleva a cabo el núcleo. Linux utiliza memoria virtual con paginación bajo demanda, siendo el tamaño de página dependiente de la arquitectura del procesador.
- En las versiones de 32 bits cada proceso de usuario tiene su propio espacio de direcciones virtuales de 4 GB, de los cuales los primeros 3 GB están disponibles para el código y los datos del proceso. El resto está dedicado a sus propias tablas de páginas y al sistema operativo. El espacio de direcciones se crea junto con el proceso mediante la llamada al sistema fork.
- El espacio de direcciones virtuales está formado por un conjunto de áreas que a su vez contienen una serie de páginas consecutivas con las mismas propiedades. Todas las áreas que forman un proceso se incluyen en una lista enlazada ordenada por dirección virtual. Existe una estructura de datos por cada área de un proceso que contiene sus propiedades.
- Utiliza una estructura de paginación a tres niveles y cada dirección virtual tiene cuatro campos:
  - El *primero* señala una entrada del directorio global de páginas. Un proceso activo tiene un solo directorio global de páginas, que debe estar en memoria principal, donde cada entrada al mismo señala a la dirección base de un directorio intermedio de páginas.
  - El *segundo* señala una entrada del directorio intermedio de páginas apuntado por el primer campo. Cada entrada a este directorio apunta a la dirección base de una tabla de páginas. En los procesadores con dos niveles de paginación, como el Pentium, el directorio intermedio de páginas tiene una sola entrada.
  - El *tercero* señala una entrada de la tabla de páginas apuntada por el tercer campo. Cada entrada de una tabla de páginas contiene el marco de página donde se aloja la página.
  - El *cuarto* indica el desplazamiento dentro de la página seleccionada de la memoria.
- Para el reemplazo de páginas, existe un proceso demonio (*kswapd*) que se encarga de mantener suficientes marcos libres, y cada segundo comprueba si este número es demasiado bajo. Si es así, busca marcos que puedan ser sustituidos. Al asignar una página se le asocia por defecto un valor a una variable edad, que se incrementa cada vez que se accede a dicha página. Por su parte, *kswapd* realiza una rotación por las páginas de memoria y decrementa la edad de aquellas que no se utilizan. Las páginas víctimas para la sustitución se eligen de entre las páginas con edad cero.