

2. Funciones y objetivos de los sistemas operativos y manejo de la memoria

Contenido

- 2.1. El sistema operativo como interfaz usuario/computadores
- 2.2. El sistema operativo como administrador de recursos
- 2.3. Facilidad de evolución de un sistema operativo
- 2.4. Requerimientos de la gestión de memoria
- 2.5. Concepto de memoria virtual
- 2.6. Concepto de paginación
- 2.7. Incidencia de la paginación en el rendimiento del sistema
- 2.8. Descripción de la gestión de memoria en sistemas Linux, Windows y Solaris

Un sistema informático está compuesto por elementos físicos (*hardware*) y por elementos lógicos (*software*). El software es el encargado de gestionar la información de los usuarios, realizando las tareas de almacenar, procesar y recuperar la información. El software más importante del sistema informático es el sistema operativo, el cual es el encargado de controlar y gestionar todos los recursos y elementos del computador. Además, actúa de intermediario/soporte del hardware y otras aplicaciones software.

Los computadores hoy en día están compuestos por una gran cantidad de elementos hardware, como son varios procesadores, memoria, sistemas de almacenamiento, dispositivos de entrada/salida, etc. Esto hace que sea muy complicado poder desarrollar aplicaciones software que controlen todos estos elementos. Es el sistema operativo el encargado de abstraer a los desarrolladores de las dificultades hardware, permitiendo que el mismo software pueda ser ejecutado en diferentes hardwares mientras compartan la capa de sistema operativo.

En la Figura 12 se muestra la estructura de capas software que se encuentra en un sistema informático. La capa inferior es el hardware del computador y se encuentra compuesta por dos o más niveles. El primer nivel son los elementos físicos como tal, y son los elementos que serán controlados por los elementos software. Incrustado en el propio hardware se encuentra un software muy básico, que es conocido como el firmware de cada uno de los dispositivos.



Figura 12. Capas de software. En el primer nivel se encuentran los dispositivos físicos como tales (el software sería el firmware), por encima se encuentra el lenguaje ensamblador, que permite gestionar a los dispositivos físicos. El siguiente nivel es el sistema operativo, el cual actúa de máquina virtual. Las herramientas del sistema operativo se encuentran por encima de este y, finalmente, las aplicaciones o software de aplicación general.

Este software se encarga de enviar y controlar las órdenes básicas del hardware y normalmente está escrito en lenguaje ensamblador o un lenguaje propio de cada uno de los dispositivos hardware. Debido a que desarrollar aplicaciones en este lenguaje es bastante complejo, surge el sistema operativo, el cual proporciona un lenguaje de programación más usable para los desarrolladores y permite ser utilizado para diferentes dispositivos hardware siempre que se mantenga el mismo sistema operativo. El sistema operativo proporciona un conjunto de herramientas que son conocidas como las herramientas del sistema operativo, que permiten la gestión del sistema operativo y de las aplicaciones software que se usarán en el propio sistema operativo. Por ejemplo, los compiladores, intérpretes de comandos, editores etc., pertenecen a este conjunto de herramientas. Por último, se encuentran las aplicaciones finales, por ejemplo, software de ofimática, videojuegos, navegadores Web, etc.

En el presente capítulo se introduce el concepto de sistema operativo, el cual tiene dos visiones: a) como interfaz usuario/computadora; b) como administrador de recursos. Posteriormente, se detallarán las diferentes técnicas de las que dispone el sistema operativo para realizar la gestión de memoria RAM y para obtener mayor rendimiento de la misma.

2.1. El sistema operativo como interfaz usuario/computadores

La primera visión del sistema operativo es como interfaz de usuario/computador. Esta visión proviene del hecho de que la arquitectura hardware de un sistema informático (conjunto de instrucciones, estructura de niveles de memoria, interfaces de entrada/salida, etc.) conlleva una gran complejidad para los desarrolladores de aplicaciones. Imaginad que un desarrollador de videojuegos debería tener en cuenta el conjunto de instrucciones de los microprocesadores de cada uno de los procesadores donde desea que ese videojuego se pueda ejecutar. Este hecho haría imposible el desarrollo de aplicaciones tal y como hoy se conocen.

La primera capa que abstrae el hardware a los desarrolladores es el sistema operativo. Además, elimina la complejidad de la gestión de la memoria, la gestión de los sistemas de ficheros es delegada a un software específico denominado el sistema de ficheros, que se verá en el capítulo 3.

2.2. El sistema operativo como administrador de recursos

El segundo punto de vista para un sistema operativo es el de la aplicación software encargada de gestionar los elementos hardware del computador. Es decir, asignar los tiempos de procesador, el espacio de memoria RAM, el uso de buses,

la comunicación de redes para las herramientas software que están instaladas en el sistema operativo. Es decir, el sistema operativo es el encargado de repartir los recursos hardware entre las herramientas software.

2.3. Facilidad de evolución de un sistema operativo

Una de las características que debe contemplar un sistema operativo es la capacidad de evolucionar en el tiempo. Esto debe ser así debido a varias circunstancias que se producen en un sistema informático:

- **Actualización del hardware.** Los sistemas operativos deben poder adaptarse a la aparición de nuevos tipos de elementos de hardware o la actualización de los mismos. Esto se puede conseguir gracias a un completo sistema de comunicación entre sistema operativo y controladores (*drivers*).
- **Nuevos servicios.** Los sistemas operativos deben poder ampliar los servicios que proporcionan. Por ejemplo, las distribuciones linux de base incorporan pocos servicios a los usuarios, pero son fácilmente extensibles a servicios como son el Web, FTP o impresión.
- **Correcciones.** Los sistemas operativos deben poder corregir sus errores tanto de seguridad como de utilización. Es inevitable que todos los sistemas operativos tengan fallos debido a la constante evolución del software y del hardware. No obstante, deben proporcionar un mecanismo que permita implantar correcciones sin dificultad.

2.4. Requerimientos de la gestión de memoria

Una pieza fundamental en el sistema informático es la memoria, debido a que todos los procesos que se ejecutan deben estar previamente cargados en memoria para que el microprocesador pueda hacer uso de ellos. En la actualidad los sistemas operativos son multitarea, es decir, se están realizando varias tareas simultáneamente, y para ello deben compartir tanto la CPU [diferentes tiempos de procesamiento para cada uno de ellos] como la memoria RAM. La memoria RAM es un espacio limitado y no es posible cargar todas las aplicaciones software simultáneamente en la memoria y es por ello que surge la necesidad de realizar una gestión adecuada de la memoria por parte del sistema operativo.

La memoria, por tanto, puede visualizarse como una gran tabla en la que se almacenan las instrucciones y datos correspondientes a cada uno de los procesos. La parte del sistema operativo encargada de gestionar la memoria

recibe el nombre de gestor de memoria y debe tener en cuenta los huecos disponibles o libres de la memoria y los que se están utilizando, para potenciar el espacio y la eficiencia de la misma, ya que debe realizar las transferencias entre los sistemas de almacenamiento y la memoria, y de la memoria al microprocesador.

El gestor de memoria del sistema operativo debe proporcionar las siguientes características:

- **Protección.** Se debe asegurar que ningún proceso pueda modificar el espacio de posiciones de memoria de otro proceso. Lo normal es que en la memoria cohabiten varios procesos, pero cada uno con su propio espacio de memoria. Por lo tanto, el gestor de memoria debe proporcionar permisos de acceso que permitan especificar los diferentes derechos de los que dispone cada proceso sobre los otros.
- **Compartición.** Al igual que hay que evitar que los procesos accedan a posiciones de memoria de procesos no autorizados, es necesario que procesos que comparten datos puedan hacer uso de espacios de memoria compartida para ahorrar dicha memoria o poder trabajar conjuntamente sobre dichos datos.
- **Reubicación.** La reubicación es la característica que permite desplazar los procesos por las diferentes posiciones de memoria, puesto que podrán existir varios procesos simultáneamente en memoria y puede ser interesante desplazarlos a diferentes posiciones de memoria según vayan surgiendo las necesidades.
- **Organización de la memoria.** La memoria, al margen de disponer de una organización física (estudiada en el capítulo anterior), debe tener una organización lógica. La organización de la memoria corresponde a cómo se organiza internamente la memoria. *A grosso modo* los sistemas de gestión de memoria se clasifican en dos bloques:
 - **Fijos.** Aquellos que no mueven los procesos entre memoria principal y secundaria. Es decir, cuando un proceso es cargado en memoria estará ahí hasta que finalice su ejecución y sea desalojado por completo.
 - **Variables.** Aquellos que permiten mover los procesos entre memoria principal y secundaria. Estas técnicas surgen por necesidad de más espacio en memoria principal para cargar todos los procesos, circunstancia que es imposible a priori, pues el espacio disponible en los sistemas de almacenamiento secundario supera con creces el de las memorias RAM.

2.5. Concepto de memoria virtual

La memoria virtual surge debido a que la memoria RAM siempre será inferior a los soportes de almacenamiento externo. Es imposible que estén cargados en memoria todos los procesos con todos sus datos en memoria RAM, pero se quiere maximizar el número de procesos que están en ejecución simultáneamente. Por lo tanto, la memoria virtual es una técnica que permite la ejecución de procesos que no están cargados completamente en memoria principal, sino que están parcialmente. Una de las características de la memoria virtual es que se pueden cargar en memoria (por partes) procesos que son más grandes que el total de memoria del sistema informático.

Esta técnica consiste en cargar en memoria solamente la parte necesaria para la ejecución de la parte del programa que se está ejecutando en cada momento. Además, se realiza un mapeo de direcciones de memoria inexistentes (virtual) sobre el soporte de almacenamiento externo. El espacio en el soporte de almacenamiento externo actúa como memoria, con las limitaciones hardware de usar un disco en lugar de memoria real. El espacio para la memoria virtual en el soporte de almacenamiento en algunos sistemas de ficheros recibe el nombre de la partición de *swap* (zona de intercambio).

La dirección de la memoria que se debe utilizar proviene del bus de direcciones del procesador. Hasta hace poco el bus de direcciones del procesador era de 32 bits (conocidas arquitecturas de 32 bits), lo cual proporcionaba un espacio de memoria de 2^{32} posiciones, es decir, 4 GPosiciones, o lo que es lo mismo, 4 GBytes de memoria (si cada posición de memoria es de 1 Byte). Con la llegada de las arquitecturas de 64 bits se dispone de un espacio de memoria muy superior, pues el bus de direcciones se incrementa hasta 64 bits, proporcionando hasta 2^{64} posiciones de direccionamiento, lo cual permite direccionar a posiciones de memorias físicas (por ejemplo, 16 o 32 Gigas de RAM) y poder crear direccionamiento virtual en soportes de almacenamiento de gran tamaño.

En la Figura 13 se muestra cómo el espacio destinado a las direcciones virtuales es mayor que el espacio destinado a las direcciones físicas. Por lo tanto, las posiciones de memoria correspondientes a la memoria física (RAM) se encuentran relacionadas con el espacio para las direcciones virtuales. Además, se puede observar que en el espacio de memoria virtual existe una relación con posiciones que se encuentran en el soporte de almacenamiento (virtual). El sistema que relaciona las posiciones de donde se encuentra el dato, si en memoria física o virtual, es conocida como la traducción de direcciones.

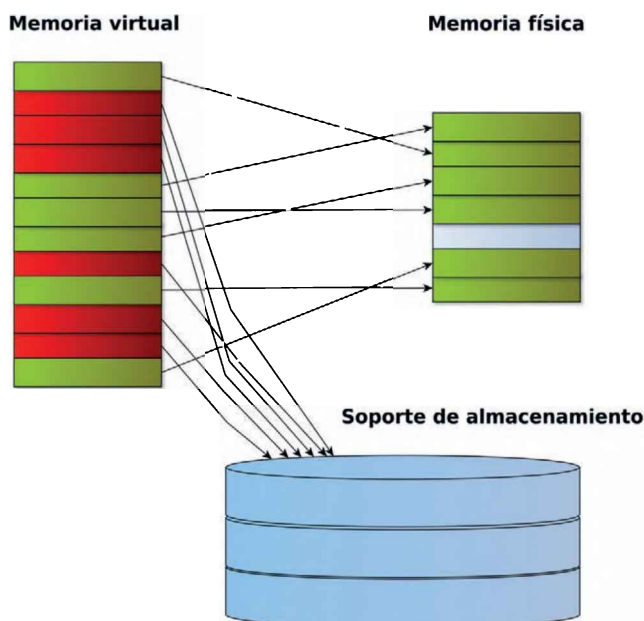


Figura 13. Direcciones virtuales y direcciones físicas con una partición de intercambio (*swap*).

2.6. Concepto de paginación

En la sección anterior se presentó el concepto de memoria virtual, en el que no es necesario cargar el proceso completo en memoria para hacer uso de este. En concreto, lo que el gestor de memoria realiza es utilizar bloques de intercambio de tamaño único y dividir la memoria principal en trozos del mismo tamaño (comprendido entre 512 bytes a 8 KB). A cada uno de estos trozos en los que se divide la memoria se le denomina marco y es donde se alojarán partes de los procesos. El lector puede deducir que los procesos se trocean en partes del mismo tamaño para poder alojarse sin desperdiciar memoria, a cada una de las partes de un proceso se la conoce como página. De este modo, la memoria tendrá un número de marcos (los cuales se encontrarán ocupados o no) y cada proceso dispondrá de un conjunto de páginas, las cuales podrán encontrarse en memoria RAM o en el soporte de almacenamiento.

En resumen, el espacio de direccionamiento físico se encuentra dividido en marcos y el espacio de direccionamiento virtual se encuentra dividido en páginas, las cuales son del mismo tamaño. De este modo, si el procesador solicita una página de un determinado proceso se revisa si dicha página se encuentra en algún marco de la memoria física. En caso de encontrarse en memoria, se accede a dicho marco, mientras que si no se encuentra la página cargada en me-

moria, se cargará desde memoria virtual a algún marco para poder hacer uso posterior de él.

La relación que existe entre las direcciones virtuales y física no es directa debido a que el número de direcciones que existen no es el mismo. Por lo tanto, hay que realizar asignaciones dinámicas de las direcciones virtuales y física, haciendo uso de una tabla intermedia que se conoce como la tabla de páginas. Esta tabla almacena la correspondencia entre páginas y marcos, y mantiene información de control que permite determinar si la página lleva mucho tiempo sin ser consultada, para que el gestor de memoria pueda tomar una decisión en caso de decidir qué página debe desalojar de la memoria.

En la Figura 14 se muestra la técnica de paginación, en la que se observa cómo es el procesador el que solicita una dirección virtual, cuya existencia y localización son comprobadas en la tabla de páginas para poder acceder a la memoria física [o virtual].

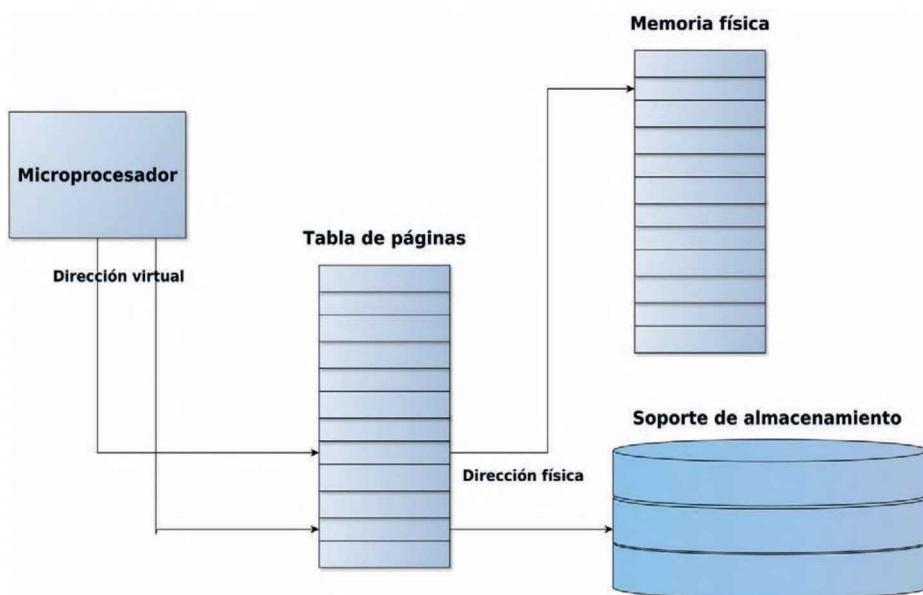


Figura 14. Técnica de paginación de memoria. Desde el microprocesador se generan direcciones virtuales, las cuales son convertidas en direcciones físicas a través de la tabla de páginas, pudiendo ir a memoria física o a un soporte de almacenamiento.

2.7. Incidencia de la paginación en el rendimiento del sistema

En el primer capítulo se presentó la jerarquía de memorias y cómo el acceso a soportes de almacenamiento externo es cientos o miles de veces más lento que

el acceso a memoria RAM; y no se debe olvidar que cada acceso a memoria virtual implica acceder a los soportes de almacenamiento. Por lo tanto, cada vez que se accede a memoria virtual hay un fuerte impacto en el rendimiento del sistema. Los mecanismos de gestión de memoria son los encargados de decidir qué páginas se cargan y cuáles no, por tanto, son una pieza fundamental en el rendimiento del sistema.

Algunos sistemas operativos tratan de reducir al máximo la memoria virtual con parámetros de configuración del gestor de memoria. Por ejemplo, en los principales sistemas se recurre a las siguientes técnicas:

- Linux permite modificar la balanza de memoria ocupada del sistema antes de realizar accesos a la memoria virtual, utilizando el parámetro de configuración que se encuentra en */proc/sys/vm/swappiness*.
- Los sistemas operativos de Microsoft permiten configurar la entrada del registro que gestiona la paginación denominada *DisablePagingExecutive*.
- Algunos sistemas operativos de UNIX permiten establecer varios dispositivos de intercambio para poder trabajar en paralelo y reducir el coste de acceder al soporte de almacenamiento.

Por lo tanto, minimizar el número de fallos de páginas (o accesos a memoria virtual) es una tarea importante que debe realizar el gestor de memoria. Por suerte, existe un algoritmo de sustitución de páginas (cuando se produce un error), conocido como el algoritmo óptimo que permite conocer el número de sustituciones de páginas óptimo. Por desgracia, este algoritmo no se puede utilizar en sistemas de tiempo real, sino que solo se puede aplicar a posteriori, cuando se conoce la secuencia de páginas solicitadas. Por lo tanto, este algoritmo permite comparar los algoritmos reales con el óptimo para saber lo alejado o cercano que se ha encontrado la secuencia de páginas con el resultado óptimo.

El algoritmo óptimo consiste en reemplazar la página que más alejada en el tiempo va a ser referenciada. En el siguiente ejemplo se muestra el uso de este algoritmo, en el cual se solicitan las páginas 3, 2, 5, 3, 1, 5, 3, 1, 2, 5, 3, 1, 5 y se producen 6 fallos de página, suponiendo que se dispone de 3 marcos de memoria RAM. En primer lugar, la carga inicial de páginas provoca que siempre haya fallo de página hasta que el algoritmo de reemplazo pueda empezar a actuar, posteriormente se sustituye aquella página que se encuentre más alejada en el tiempo a ser referenciada. En el caso de que haya varias páginas que no van a ser solicitadas nunca más en el tiempo, se suele escoger una página aleatoria entre ellas o se puede disponer de un bit de referencia para escoger cuál desempata entre las candidatas.

3	2	5	3	1	5	3	1	2	5	3	1	5
3	3	3	3	3	3	3	1	1	1	1	1	1
	2	2	2	1	2	2	2	2	2	3	3	3
		5	5	5	5	5	5	5	5	5	5	5
F	F	F		F			F			F		

En la siguiente sección se describen algunos de los algoritmos utilizados por los principales sistemas operativos para decidir qué página debe ser sustituida por otra en caso de que no resida en memoria RAM.

2.8. Descripción de la gestión de memoria en sistemas Linux, Windows y Solaris

En esta sección se va a describir cómo realizan la gestión de la memoria los principales sistemas operativos: familia UNIX, en particular Linux y Solaris, y Microsoft Windows.

El caso de Solaris es especial debido a que es un sistema operativo que deriva de los sistemas BSD, pero que en la actualidad incorpora un kernel evolucionado del sistema operativo SunOS. Los sistemas operativos SunOS y Solaris pertenecieron a la compañía Sun Microsystems hasta que fue adquirida por Oracle en el año 2010.

2.8.1. Linux

El gestor de memoria de los sistemas operativos Linux se descompone en dos partes bien diferenciadas:

- **Asignación de páginas.** Los sistemas operativos de Linux utilizan bloques de páginas contiguas que corresponden a bloques de marcos contiguos. El algoritmo utilizado para asignar la página es conocido como el sistema colega (*buddy*), donde el gestor de memoria tiene una lista de grupos de marcos de páginas contiguos.

El sistema colega (*buddy*) es un método que permite establecer el marco donde se establecerá la página. Es decir, permite mantener los bloques libres de la memoria física. En este método solo se reservan tamaños que son potencias de dos y, por tanto, solamente se dispone de dos operaciones:

—Dividir (*splitting*) un bloque en dos.

—Unir (*coalescing*) dos bloques consecutivos en uno.

Las ventajas del sistema colega son que es muy fácil de implementar y es rápido, y sus principales inconvenientes son que los tamaños solamente pueden ser potencias de dos y se provoca fragmentación interna en el procedimiento.

Inicialmente, se establece un gran bloque de memoria disponible con todo el espacio libre para su asignación, al llegar un proceso que solicita una cantidad de memoria se dividirá sucesivamente en partes de dos hasta asignar el espacio de memoria (potencia de 2) mínimo al tamaño solicitado. El método se llama *de colegas* porque cada porción de memoria se divide en dos, las cuales se dividen y unen exclusivamente con su pareja (o colega).

En la Figura 15 se muestra el procedimiento de asignar memoria a varios procesos (el procedimiento de división) y, posteriormente, la liberación de dichos procesos de la memoria. Observe que los procesos dividen los bloques de memoria en exactamente la mitad en cada iteración, hasta que se ha dividido lo mínimo para que pueda asignarse la memoria, teniendo en cuenta que siempre serán múltiplos de dos.

Bloque vacío inicialmente	2 MBytes (bloque)				
Petición de A=200 KB	A	256 KB	512 KB	1 MB	
Petición de B=510 KB	A	256 KB	B	1 MB	
Petición de C=110 KB	A	C 128 KB	B	1 MB	
Petición de D=512 KB	A	C 128 KB	B	D	512 KB
Liberar B	A	C 128 KB	512 KB	D	512 KB
Liberar A	256 KB	C 128 KB	512 KB	D	512 KB
Petición de E = 130 KB	E	C 128 KB	512 KB	D	512 KB
Liberar C	E	256 KB	512 KB	D	512 KB
Liberar E	512 KB		512 KB	D	512 KB
Liberar D	2 MBytes (bloque)				

Figura 15. Asignación y liberación de memoria de procesos utilizando el método de los colegas.

- Algoritmo de reemplazo de páginas.** El algoritmo de reemplazo de páginas utilizado es conocido como: no usada recientemente (*Not Recently Used, NRU*). En él se hace uso de un bit de uso y modificación de cada página de la memoria RAM.

El algoritmo NRU favorece las páginas que fueron utilizadas recientemente y tiene el siguiente funcionamiento: cuando una página es referenciada o modificada, fija el bit de referencia o modificación de esa página; en cada intervalo de tiempo preestablecido, el gestor de memoria resetea los bits de referencia de todas las páginas, de modo que las páginas que tienen sus bits de referencia o modificación son relevantes y, por tanto, tienen menos

prioridad para ser modificada. Es decir, el gestor de memoria clasifica las páginas en cuatro categorías:

- **Categoría 0.** No referenciada, no modificada.
- **Categoría 1.** No referenciada, modificada.
- **Categoría 2.** Referenciada, no modificada.
- **Categoría 3.** Referenciada, modificada.

Las páginas que se sustituirán serán las de la categoría 0, posteriormente las de la categoría 1 y así sucesivamente. En el siguiente ejemplo, se muestra el uso del algoritmo NRU. En este ejemplo se solicitan las páginas 3, 4, 5, 3, 2, 5, 3, 1, y se producen 5 fallos de página, suponiendo que se dispone de 3 marcos de memoria RAM. En primer lugar, la carga inicial de páginas provoca que siempre haya fallo de página hasta que el algoritmo de reemplazo pueda empezar a actuar, posteriormente se produce un acierto de página cuando se solicita la página 3 y vuelve a haber un fallo de página al solicitar la página 2. En este caso, se sustituye la página 4, dado que la 3 acaba de ser referenciada, y entre la 4 y 5, la que fue referenciada antes es la página 4. Del mismo modo, el algoritmo continúa produciendo dos aciertos al solicitar las páginas 5 y 3, que pasan a ser recientemente referenciadas. Por lo tanto, finalmente se produce un fallo al solicitar la página 1 y la página sustituida será la perteneciente al 2.

3	4	5	3	2	5	3	1
3	3	3	3	3	3	3	3
	4	4	4	2	2	2	1
		5	5	5	5	5	5
F	F	F		F			F

2.8.2. Microsoft Windows

La familia de sistemas operativos de Microsoft ha ido evolucionando a lo largo del tiempo. Es un sistema operativo privado y comercial, por lo que la información de su funcionamiento es la proporcionada por el propio fabricante o por lo descubierto por la comunidad a golpe de ingeniería inversa. No obstante, para conocer el funcionamiento de uno de estos sistemas operativos, se va a describir el gestor de memoria utilizado desde el año 2000 para la línea de sistemas operativos destinados a servidores, Windows Server.

Las partes interesantes del gestor de memoria son las siguientes:

- **Mapa de direcciones virtuales.** Cada uno de los usuarios dispone de su propio espacio de direcciones de 32 bits, lo que permite que cada usuario disponga de hasta 4 GBytes. De estos 4 GBytes el sistema operativo se reserva 2 GBytes de espacio de direcciones virtuales para su operativa.
- **Asignación de páginas.** Cada proceso de un usuario puede hacer uso de hasta 2 GBytes (todo el espacio de memoria de cada usuario). El espacio está dividido en páginas de tamaño fijo y cualquier página puede ser cargada en la memoria.
- **Algoritmo de reemplazo de páginas.** En el caso de que se produzca un fallo de página porque no se encuentre cargada en memoria, se solicita dicha página y se cargan páginas cercanas. Esta técnica se conoce como paginación por demanda con agrupación de páginas (*clustering*) y se basa en la teoría de que cuando se solicita una página de memoria hay altas probabilidades de que se soliciten pronto las páginas cercanas a esta. El algoritmo en sí utilizado como política de reemplazo es el LRU o FIFO (*First In, First Out*).

El algoritmo FIFO solamente almacena el orden de llegada de las páginas, de modo que al necesitar espacio para una nueva página solamente debe elegir la que fue cargada primero. Este método tiene el gran problema de que se elimina una página de memoria muy utilizada solo porque es antigua.

En el siguiente ejemplo se muestra el uso del algoritmo FIFO. En este ejemplo se solicitan las mismas páginas que en el ejemplo de NRU: 3, 4, 5, 3, 2, 5, 3, 1, pero se producen 6 fallos en lugar de 5, manteniendo 3 marcos de memoria RAM. En primer lugar, la carga inicial de páginas provoca que siempre haya fallo de página hasta que el algoritmo de reemplazo pueda empezar a actuar, posteriormente se produce un acierto de página cuando se solicita la página 3 y vuelve a haber un fallo de página al solicitar la página 2. Observe que en este caso se sustituye la página 3, que ha sido recientemente referenciada pero es la más antigua, la cual será referenciada posteriormente otra vez y provocará fallo al no estar disponible.

3	4	5	3	2	5	3	1
3	3	3	3	2	2	2	2
	4	4	4	4	4	3	3
		5	5	5	5	5	1
F	F	F		F		F	F

El algoritmo no usado recientemente (*Least Recently Used*, LRU) difiere del NRU en el hecho de que NRU solo tiene en cuenta los bits de referencia de las pá-

ginas desde el intervalo de tiempo desde que se pusieron en 0, mientras que el algoritmo LRU no tiene esta limitación temporal. Estadísticamente, las páginas sustituidas son las que tienen menor probabilidad de volver a ser utilizadas. Este algoritmo es costoso en cuanto a recursos y, por ello, los gestores de memoria suelen realizar modificaciones sobre esta idea.

En el siguiente ejemplo se muestra el uso del algoritmo LRU. En este ejemplo se solicitan páginas diferentes que en los anteriores ejemplos, puesto que el resultado será idéntico al mostrado para NRU (ya que no se ha tenido en cuenta el refresco del tiempo en NRU). A modo de ilustración, en este ejemplo se dispone de 4 marcos de memoria y una secuencia de páginas, las cuales se irán sustituyendo por aquellas que menos se utilizan en la secuencia. En este caso se producen un total de 8 fallos de página, pero observe como la página 3 nunca es sustituida, porque periódicamente es solicitada y, por tanto, siempre se dispone de la misma en memoria RAM.

1	2	3	4	2	3	5	1	4	3	6
1	1	1	1	1	1	5	5	5	5	6
	2	2	2	2	2	2	2	4	4	4
		3	3	3	3	3	3	3	3	3
			4	4	4	4	1	1	1	1
F	F	F	F			F	F	F		F

2.8.3. Solaris

Los sistemas Solaris hoy en día pertenecen a la compañía comercial Oracle Corporation y su funcionamiento interno, al igual que los sistemas operativos de Microsoft, es reservado. No obstante, a modo de comprensión de su funcionamiento general, se van a dar unas pautas ofrecidas por la propia compañía.

Los puntos más importantes de este gestor de memoria son los siguientes:

- **Estructuras de datos.** Para la gestión de la memoria virtual se hace uso de un conjunto de estructuras de datos lógicas:
 - **Tabla de páginas.** Cada uno de los procesos del sistema dispone de una tabla de páginas en la cual cada una de las entradas recoge la página de la memoria virtual de la que hace uso el proceso en cuestión.
 - **Tabla de marcos de página.** Existe una tabla con los marcos de la memoria que está indexada por el número de marcos para una localización rápida.

- **Descriptor de bloques de disco.** Cada página de los procesos dispone de un descriptor de bloque de disco. En la tabla de páginas existe una entrada que indica dónde se encuentra la copia en el soporte de almacenamiento de la página virtual.
- **Algoritmo de reemplazo de páginas.** El algoritmo de reemplazo utilizado en los sistemas Solaris es conocido como el algoritmo del reloj de dos agujas, el cual utiliza un bit de referencia para cada página que puede ser expulsada.

ACTIVIDADES

1. Defina un sistema operativo según las dos visiones existentes.
2. ¿Cuáles son las características que debe proporcionar un sistema operativo a la hora de permitir su evolución a lo largo del tiempo?
3. Los gestores de memoria de un sistema operativo deben proporcionar cuatro características. Describa cada una de ellas en una frase corta.
4. ¿Por qué es necesario que los programas software sean reubicables en memoria?
5. Los procesos normalmente no pueden acceder a una zona de memoria donde no están autorizados. ¿Por qué y cuándo debería un proceso ajenos acceder a una zona de memoria que no es de su propiedad?
6. ¿Qué es la memoria virtual? ¿Qué ventajas aporta disponer de memoria virtual? ¿Qué es el área de intercambio (*swapping*)?
7. ¿Qué diferencia existe entre direcciones virtuales y direcciones físicas?
8. ¿En qué consiste la paginación? ¿Qué son las páginas y los marcos?
9. ¿Cuándo ocurren fallos de página en la memoria RAM? Describa las acciones que el gestor de memoria debe llevar a cabo cuando se produce dicho fallo.
10. ¿Qué impacto tiene en el rendimiento del sistema un fallo de página de la memoria RAM?
11. Explique las diferencias entre fragmentación interna y externa. ¿Qué implicaciones negativas conlleva cada una de ellas?
12. Aplicando el sistema *colegas (buddy)*, tenga en cuenta que se dispone de una memoria RAM de 1024KBytes y se produce la siguiente secuencia de peticiones/liberación de memoria por parte de procesos:
 - a. Petición de A de 180 KB
 - b. Petición de B de 10 KB
 - c. Petición de C de 200 KB
 - d. Petición de D de 80 KB
 - e. Liberar C
 - f. Liberar A
 - g. Petición E de 150 KB

13. Para las siguientes series de referencias de páginas, contabilizar el número de fallos de memoria con los algoritmos FIFO, LRU y Óptimo, teniendo en cuenta que se dispone de entornos con dos, tres, cuatro y cinco marcos. Inicialmente, todos los marcos están vacíos.
- a. 1, 2, 3, 4, 2, 1, 1, 5, 2, 4, 2, 5, 6, 6, 6.
 - b. 1, 2, 3, 3, 2, 1, 4, 5, 2, 2, 3, 3, 4, 1, 3.
 - c. 6, 5, 4, 3, 2, 1, 2, 3, 3, 2, 3, 5, 4, 1, 4.