PAGINACIÓN DE MEMORIA EN OS X

MEMORY PAGING IN OS X

RESUMEN
Este artículo expone el esquema de manejo de memoria denominado paginación en el sistema operativo OS X. Mediante la implementación de una simulación se evidencia la administración de la memoria, la longitud de los procesos, las paginas correspondientes, los marcos (frames) referentes a las paginas, la dirección física en memoria y la tabla de páginas. Además se presenta un ejemplo detallado del manejo del esquema en la simulación propuesta.

Palabras clave: marcos, memoria, OS X, paginación.

ABSTRACT
This paper shows the memory management scheme called paging in OS X operating system. By implementing a simulation, the memory management, the length of the processes, the corresponding pages, the frameworks relating to the pages, the physical address in memory and the page table are evidenced. In addition, a detailed example of the proposed simulation scheme management is presented.

Key words: frames, memory, OS X, paging

Nancy Yaneth Gélvez García
Magister en Ingeniería de Sistemas y Computación
Docente Universidad Distrital Francisco José de Caldas
nygelvezg@udistrital.edu.co
Bogotá, Colombia

Iván Javier Ponce Fajardo
Estudiante de Ingeniería de Sistemas Universidad Distrital Francisco José de Caldas
ijponcef@correo.udistrital.edu.co
Bogotá, Colombia

Linda Sheriyn Rodríguez Castro
Estudiante de Ingeniería de Sistemas Universidad Distrital Francisco José de Caldas
lsrodriguezc@correo.udistrital.edu.co
Bogotá, Colombia

Tipo: Artículo de reflexión

Fecha de Recepción: Octubre 22 de 2013
Fecha de Aceptación: Diciembre 22 de 2013

1. INTRODUCCIÓN

La memoria es la parte encargada de los recursos que necesita el sistema operativo, por lo cual se han creado esquemas de gestión de memoria que realizan puente entre los requisitos de las aplicaciones y el hardware disponible [1]. Consiste a su vez en administrar de manera eficiente, segura y en realizar una distribución adecuada de los recursos que posee una máquina, en los diversos procesos que se ejecutan, permitiéndoles a programadores desarrollar aplicaciones sin vínculos con ejecuciones de otros procesos. La memoria hace un respectivo reparto en espacio mientras que el procesador lo hace en tiempo. Es conveniente en ciertas situaciones que dos o más procesos compartan memoria ya que permiten ahorro físico en la máquina con la respectiva compartición de código.

Un aumento de capacidad es conseguido al mantener diferentes procesos en memoria, lo cual eleva el grado de utilización y velocidad del procesador para poder realizar una buena gestión de memoria. Se debe diseñar de manera adecuada el hardware del sistema ya que cada algoritmo requiere un soporte hardware. La memoria se compone de miles de palabras o bytes en donde cada una tiene su propia dirección [2].

Es un sistema monoprogramado el cual consta de dos partes, la primera contiene procesos en ejecución y la segunda es donde el sistema operativo realiza la gestión de la memoria, lo cual consiste en subdividir la memoria. Por tanto es necesario asignar la memoria para asegurar una cantidad de procesos listos que consumen el tiempo de procesador disponible.

Cada mecanismo debe tener la capacidad de permitir a varios procesos acceder a la misma porción de memoria principal, por lo general en una memoria la organización del almacenamiento es de manera lineal o unidimensional, compuesto por una secuencia de bytes o palabras y a nivel físico, la memoria secundaria está organizada de la misma manera. La memoria tiene como función principal traer procesos para que el procesador los pueda ejecutar necesitando este una memoria virtual basadas en técnicas denominadas paginación o segmentación [3].

Los sistemas operativos como Windows, Linux, Unix y OS X manejan mecanismos para el manejo de memoria. De esta manera en el presente artículo se presenta una simulación del mecanismo de paginación en OS X.

OS X es el sistema operativo creado por Apple, se caracteriza por no tener una línea como tal de comandos, es completamente gráfico. La versión OS X está basada en el sistema Darwin, un sistema de código abierto que utiliza característica de otros sistemas UNIX [4]. Se destaca por su facilidad de uso y su multitarea cooperativa, aunque con memoria muy limitada, la falta de memoria protegida. Algunas extensiones pueden no funcionar correctamente en conjunto, o sólo funcionan cuando se cargan en un orden determinado.

El sistema VM (maquina virtual) usado en OS X es descendiente de Mach VM, que se creó en la Universidad de Carnegie Mellon en la década de 1980.

En gran medida, el diseño fundamental es el mismo, aunque algunos detalles son diferentes, sobre todo en cuanto a la mejora del sistema VM. Sin embargo, admite la posibilidad de solicitar un determinado comportamiento de búsqueda mediante el uso de listas de páginas universales. El diseño de Mach VM se centra en el concepto de la memoria física de ser un caché para la memoria virtual. En un nivel inferior, el nivel de objeto, la memoria virtual es vista como una colección de objetos de VM y objetos de memoria, cada una con un propietario particular y las protecciones. Estos objetos se pueden modificar con llamadas a objetos que están disponibles tanto para la tarea y las paginas [6].

2. PAGINACIÓN

Tanto las particiones de tamaño fijo como variable son ineficientes en el uso de la memoria ya sea por producir fragmentación interna o

externa, pero aun así si se divide la memoria principal en porciones pequeñas de tamaño fijo a igual que el proceso, y se les denomina paginas, este mecanismo se conoce como paginación [3]. La unidad básica del esquema de paginación se denomina página. La página es una zona de memoria contigua de tamaño determinado en potencia de 2, por lo general del tamaño de página más usado es de 4KB aunque lo permitió es de 2KB a 16KB [1] [3].

Tanto el mapa de memoria del proceso como la memoria principal se dividen en zonas de igual tamaño, para el proceso se denominan páginas y para la memoria se conocen como marcos. En un instante determinado un marco tendrá una página de memoria del proceso, lo que implica relacionar la página con el marco, la estructura que cumple esta funcionalidad se denomina tabla de páginas. Además la tabla de páginas es usada para la identificación de la dirección física mediante el desplazamiento y el número del marco, en la figura 1 se presenta el esquema de traducción de la paginación.

La tabla de páginas también posee la información de protección (accesos permitidos) y el indicador de página válida (que conserva una traducción asociada) [1]. Es el procesado el que debe conocer como acceder a la tabla de páginas del proceso actual, pues es el que traduce a una dirección física [3].

En caso dado de no contar con suficientes marcos libres para cargar un proceso, el sistema operativo aun así lo carga gracias al concepto de dirección lógica. A la dirección lógica se asocia un número de página y un desplazamiento. Cuando un proceso se carga en memoria todas sus páginas se asignan en los marcos disponibles [3]. La paginación permite que cada página del mapa de un proceso pueda concordar con cualquier marco de memoria, disminuyendo la traducción pero afectando el aprovechamiento de la memoria. Es decir a un tamaño pequeño de página se reduce la fragmentación y a un tamaño grande de pagina se obtiene mejor rendimiento en los accesos a disco.

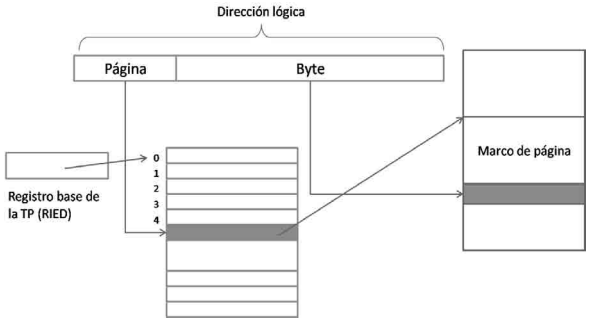

Figura 1. Esquema de traducción de la paginación [1].

La paginación descarta la obligación de almacenar de forma contigua en memoria principal el mapa de proceso, permitiendo usar cualquier marco libre.

3. SIMULACIÓN

Para realizar la simulación se identificaron las siguientes premisas:

- Cada proceso tiene asociada una tabla de páginas que crea un espacio lógico independiente.
- Cada página posee el PID del proceso y un tamaño de 16MB.
- La tabla de páginas establece la correspondencia con la parte de la memoria a la que se puede acceder.
- La memoria se divide en marcos, los cuales se identifican con un ID, poseen un valor booleano para conocer si está ocupado y la referencia a la página que lo ocupa.
- La dirección física se calcula al concatenar el número del marco y el desplazamiento, donde los dos valores son expresados en base binaria.
- La memoria principal posee un espacio de 512 MB.
- La dirección física tiene un tamaño de 16 bits.
- Al finalizar un proceso inmediatamente se asigna la memoria libre a la mayor cantidad de procesos bloqueados.
- La simulación se realizo en el lenguaje Python 2.7 con la ayuda del framework Pygame para la interfaz gráfica.

Las figuras que se presentan a continuación ilustran el uso de páginas y marcos. En un momento dado, se crea un proceso con PID=0 y una longitud requerida de 146 MB, lo que indica que se requieren 10 páginas para cubrir toda la longitud (146/16≈10). En la tabla de paginación se observa la correspondencia entre las diez páginas creadas con id de 0 a 9 y los diez marcos asociados, igualmente con id de 0 a 9. En la memoria se ocupa los primeros marcos y se presenta la dirección a la que corresponde cada uno. Para el caso del proceso PID=0, el marco 9 correspondiente en binario a 01001 y con un desplazamiento de 455 correspondiente en binario a 00111000111. Lo que indica que la dirección del marco 9 que se asocia a la página 9 del proceso PID=0 es 0100100111000111, como se observa en la figura 2.

Luego se crea el nuevo proceso con PID=1 formado por 12 páginas y el proceso PID=2 compuesto de 3 páginas. Al momento de cargar el proceso PID=1, el sistema operativo encuentra 12 marcos libres (id=10 al id=21) y carga las 12 páginas del proceso en dichos marcos. En la figura 3 se evidencia en cargue en los respectivos marcos y en la figura 4 se presenta l cargue del proceso PID=2, que ocupa los marcos id=22 al id=24.

Al crear un proceso PID=3 con una longitud de 136 MB se requiere de 9 páginas y por tanto de 9 marcos para su cargue, pero la memoria solo cuenta con 7 divisiones libres. El proceso PID=3 queda en modo bloqueado o en espera a que la memoria cuente con los recursos suficientes. En la figura 5 se observa el proceso en la pila de espera.

Con el fin de cargar el proceso PID=3, en el simulador se selecciona el proceso PID=1 (haciendo clic con el mouse sobre el proceso de la pila ejecución, ver figura 6), inmediatamente en memoria se resaltan los marcos que está usando el proceso. Luego de seleccionar el proceso, se decide finalizarlo. Para esto se oprime el botón fin del simulador. En la figura 7 y 8 se observa que los marcos del id=10 al id=21 ya no se encuentran ocupados por las páginas del proceso PID=1, sino que el marco id=10 al marco id=18 están reservados para las páginas del proceso PID=3, el cual paso a la pila de ejecución.

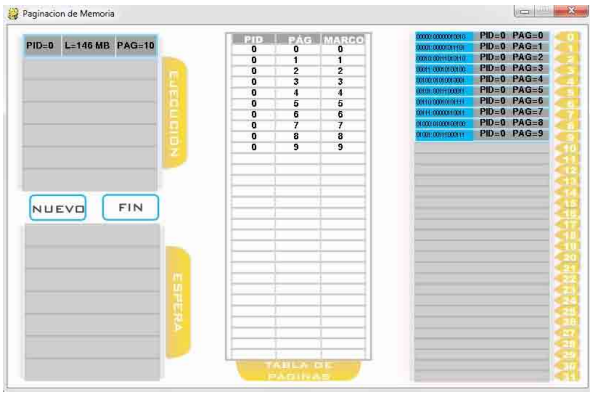

Figura 2. Creación del proceso PID=0

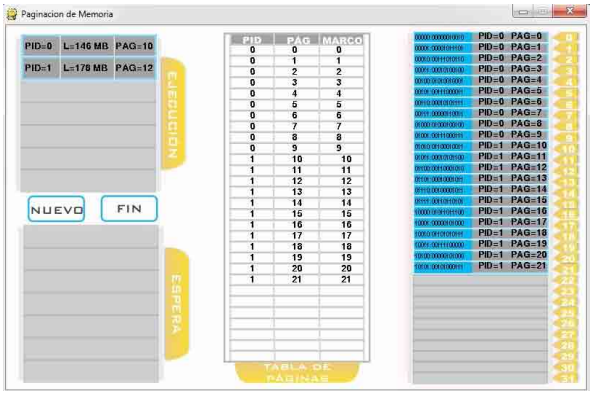

Figura 3. Creación del proceso PID=1

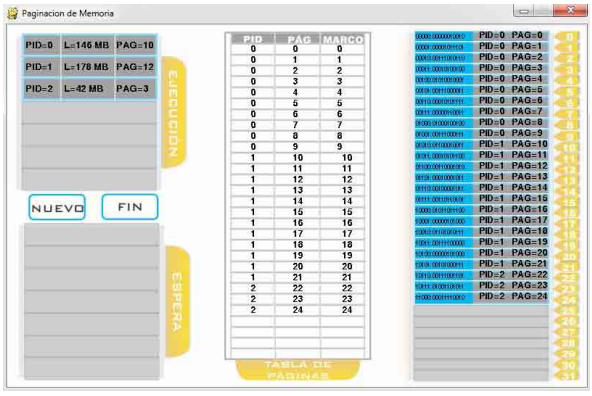

Figura 4. Creación del proceso PID=2

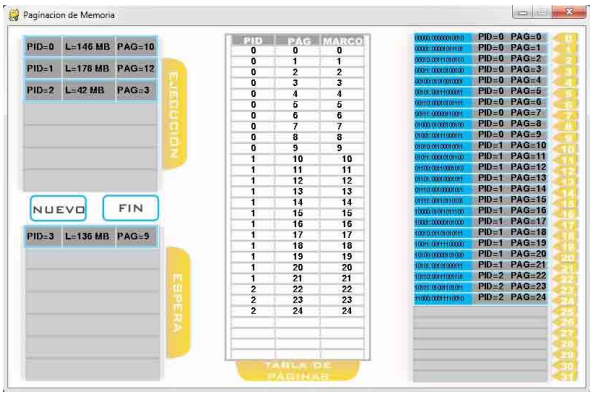

Figura 5. Creación del proceso PID=3

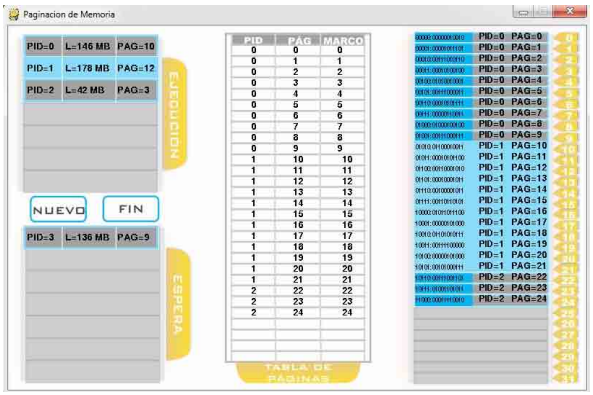

Figura 6. Selección del proceso PID=1 para finalización

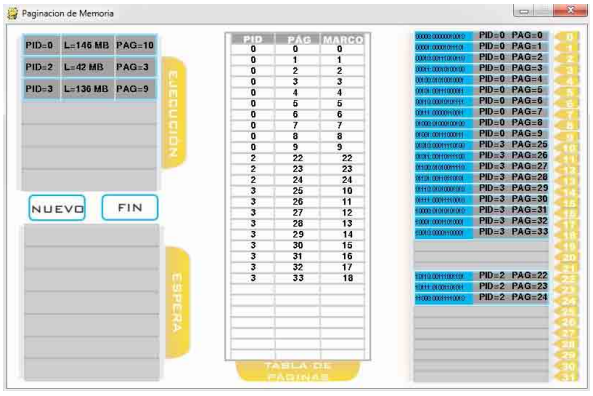

Figura 7. Carga en memoria de las paginas del proceso PID=3

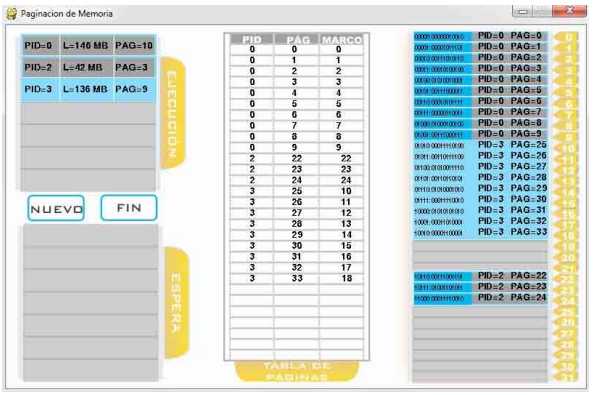

Figura 8. Visualización de los marcos correspondientes al proceso PID=3

4. CONCLUSIONES

La gestión de memoria de aplicaciones es el proceso de asignación de memoria durante la ejecución de su programa, el uso, y la liberación cuando haya terminado con él. Un programa bien escrito utiliza poca memoria.

La división de la memoria en porciones fijas no logra el manejo optimo de la misma, pero aun tamaño lo suficientemente pequeño y considerable se logra minimizar el espacio malgastado en la memoria correspondiente solo a la fragmentación de la última página. La paginación además logra un buen manejo en la asociación de la dirección lógica con la dirección física a la memoria.

5. TRABAJOS FUTUROS

Como trabajo futuro se propone la implementación de un simulador que integre los algoritmos de planificación con la administración de la memoria. Permitiendo la elección del algoritmo ya sea Short Remaining Time First (SRTF), Shortest-Job-First (SJF), apropiativo, no apropiativo, planificación con colas de multiples

niveles y retroalimentación; incluyendo la elección del mecanismo de administración de memoria sea paginación o fragmentación.

El desarrollo de esta propuesta permitirá soportar los procesos de aprendizaje, diferenciando las características de cada método de planificación y administración.

Referencias Bibliográficas

[1] J. Carretero, P. De Miguel, F. García, F. Pérez. Sistemas operativos. Una visión aplicada. Mc Graw Hill, Madrid, 2001.

[2] A. Silberschatz, P. Baer, G. Gagne. Fundamentos de sistemas operativos. Mc. Graw Hill. España. pp. 243, 2006.

[3] W. Stallings. Sistemas operativos. Pearson Educación. Madrid. p.p 308, 2005.

[4] L. Darán. El gran libro del PC interno. MARCOMBO, 2007.

[5] Apple. OSX. [En línea], consultado en Noviembre 30 de 2013, disponible en: www.apple.com.

[6] Apple Developer. Kernel Programming Guide. Apple Inc, 2013.