Capítulo 4. Memoria virtual

4.3. Algoritmo de reemplazo de páginas

En sistemas operativos que utilizan paginación para el manejo de memoria, los algoritmos de reemplazo de páginas son usados para decidir qué páginas pueden ser sacadas de memoria cuando se necesita cargar una nueva y ya no hay espacios.

Óptimo

Este algoritmo tiene como finalidad retirar la página que vaya a ser referenciada más tarde, por ejemplo si hay una página A que será usada dentro de 10000 instrucciones, y una página B que será usada dentro de 2800 instrucciones, se debería eliminar de la memoria la página A. Como se puede deducir, para esto el sistema operativo debería ver en cuánto tiempo será usada cada página en memoria y elegir la que está más distante. El problema de este método es que necesita conocimiento del futuro, por lo que es imposible su implementación. Es un algoritmo teórico. Se utiliza a los efectos comparativos con los algoritmos factibles de ser implementados para ver cuál se aproxima más a éste.

Algoritmo de reemplazo de páginas de la segunda oportunidad

Este algoritmo es una modificación del FIFO. El algoritmo hace uso del bit de referencia de la página. Cuando una página ha sido seleccionada para reemplazo, se revisa el bit de referencia. Si tiene valor de 0, se procede a reemplazar la página. Si por el contrario, el bit de referencia es 1 se le da a la página una segunda oportunidad.

Cuando esto sucede, se le cambia el bit de referencia a 0 y se actualiza su tiempo de llegada al tiempo actual para que la página se colocada al final de la cola. De esta manera, la página espera todo un ciclo completo de páginas para ser entonces reemplazada.

Si la página tiene un uso muy frecuente, el bit de referencia se mantendría constantemente en 1 y la página no sería reemplazada. En la figura 10 se puede apreciar el funcionamiento del algoritmo.

Algoritmo de reemplazo de páginas del reloj

Modificando el algoritmo de la segunda oportunidad (que a su vez es una modificación de FIFO) obtenemos el algoritmo aumentado de la segunda oportunidad o algoritmo del reloj. Usamos la misma clasificación vista en el algoritmo de uso no tan reciente.

Este algoritmo organiza las páginas en una lista circular como se muestra en la figura 11 y se usa un apuntador (o manecilla) que señala a la página más antigua.

4.4. Asignación de marcos de página

Otra opción para el manejo de la memoria es usar una forma de liberar al programador de la tarea del control de las tablas en expansión y contracción, de la misma forma que la

memoria virtual elimina la preocupación por organizar el programa en una serie de proyectos.

Esto se puede lograr dotando a la máquina de varios espacios independientes de direcciones llamados segmentos. Cada segmento tiene una serie lineal de direcciones, desde 0 hasta cierto máximo. La longitud de cada segmento puede variar de 0 hasta un máximo permitido. Los distintos segmentos pueden tener y de hecho tienen por lo general, longitudes distintas. Además, la longitud de un segmento puede variar durante la ejecución. La longitud de un segmento de la pila puede crecer si algo entra a la pila y decrecer si algo sale de ella.

Puesto que cada segmento constituye un espacio independiente de direcciones, los distintos segmentos pueden crecer o reducirse en forma independiente sin afectar a los demás.

La segmentación también facilita el uso de procedimientos o datos compartidos entre varios procesos. Un ejemplo común son las bibliotecas compartidas (Shared DLL's). Es frecuente que las estaciones de trabajo modernas que ejecutan sistemas avanzados, con ventanas, tengan bibliotecas gráficas de tamaño muy grande que se compilan casi en todos los programas. En un sistema segmentado, la biblioteca gráfica se puede colocar en un segmento y compartirse entre varios procesos, sin necesidad de tenerla en el espacio de direcciones de cada proceso.

Aunque también es posible tener bibliotecas compartidas sin los sistemas con paginación pura, es mucho más complejo. De hecho, estos sistemas simulan la segmentación.

Comparación de paginación y segmentación.

Este fenómeno de fragmentación externa o *checkboarding*, desperdicia la memoria correspondiente a los huecos, pero es fácilmente corregido mediante el uso de la compactación. De esta forma los huecos son unificados, generando así un hueco de tamaño suficiente para albergar algún otro segmento más.
Para ver el grafico, utilicé la opción "Bajar trabajo" del menú superior

Desarrollo de fragmentación externa y su corrección mediante compactación.
Segmentación con paginación: MULTICS

En el sistema MULTICS, una dirección lógica está formada por un número de segmento de 18-bit y un offset de 16 bit. Aunque este esquema crea un espacio de dirección de 34-bit, la sobrecarga en la tabla de segmentos es tolerable; solo se requiere de las suficientes localidades en la tabla de segmentos como haya segmentos, puesto que no debe haber localidades vacías.

Sin embargo, con segmentos de palabras de 64K, es donde cada una consiste de 36 bits, el tamaño promedio de segmento puede ser grande y se podría presentar un problema de fragmentación externa. Aunque no lo fuera, el tiempo de búsqueda para ubicar un segmento, usando primer ajuste o mejor ajuste, puede ser prolongado. Lo que puede causar desperdicio de memoria debido a la fragmentación externa, o desperdicio de tiempo por las largas búsquedas o ambas.

La solución al problema es paginar los segmentos. La paginación elimina la fragmentación externa y convierte el problema de ubicación en algo trivial: cualquier frame desocupado puede ser usado para una página deseada. En MULTICS, cada página consiste de palabras de 1K. En consecuencia, el offset del segmento (16 bits) es dividido en un número de página de 6 bit y un offset de página de 10 bit. El número de página se indexa en la tabla de páginas para obtener el número de frame. Finalmente, el número de frame es combinado con el offset de página para formar una dirección física.