9. Administración de Memoria: Memoria Virtual

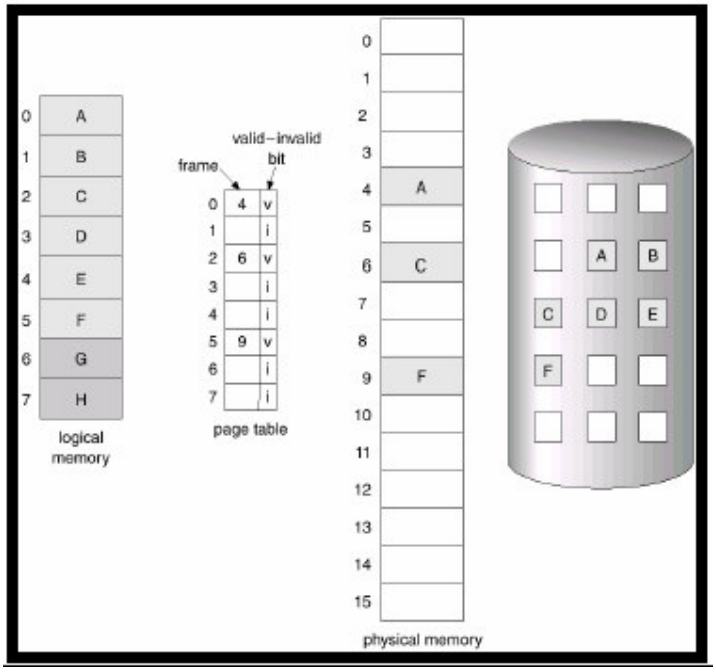
1. Introducción

La memoria virtual permite ejecutar procesos que requieren mas memoria que la disponible en el sistema, manteniendo en memoria principal solo aquella memoria que el proceso este utilizando y el resto en disco. De esta forma el usuario ya no debe preocuparse por las limitaciones de memoria física.
Cada proceso tiene su propio espacio de direccionamiento virtual (o lógico) y La MMU es la encargada de mapear las direcciones virtuales (o lógicas) a físicas.

2. Implementación

La implementación de memoria virtual es realizada a través de la técnica de paginación bajo demanda. En la paginación bajo demanda los procesos residen en un dispositivo de disco y son puestos en memoria principal cuando es necesario cargarlos para ejecutar. La carga del proceso en memoria no es total, sino que implementa un cargador "perezoso" (*lazy swapper*), que cargará las páginas según se vayan necesitando.

Utilizar un esquema de este tipo requiere el conocimiento de las páginas que están activas en memoria. Para ello se utiliza el valid-invalid bit, que consiste en agregar a la tabla de páginas un nuevo campo (bit de validez), que indique para cada entrada, si la página se encuentra o no en memoria. Al inicio, la tabla de páginas indicará que ninguna página está en memoria (todos los bits de validez se encontrarán en i (invalid)).



En este ejemplo tenemos que el proceso tiene para usar 8 páginas, de las cuales solo usa 6, y de las cuales solo 3 están en memoria principal (A, C y F). Todas las páginas estarán en el disco (incluidas aquellas que también están en memoria principal).

3. Fallo de página

La memoria cargada en memoria principal se le denomina *memoria residente*.
El acceso a memoria residente por parte de un proceso es tomado como un acceso normal, pero el acceso a memoria no residente genera un *fallo de página.*
El fallo de página genera un trap a nivel del sistema operativo, que activa una rutina de atención que carga la página en memoria principal.

4. Acceso a Memoria

El acceso a memoria genera la siguiente secuencia de pasos:

- Verificar que el proceso referencia una página correcta dentro de su espacio virtual, ya que no todas las direcciones dentro de su espacio son válidas. Por ejemplo, el acceso fuera de un arreglo puede generar un acceso a una página virtual que no fue asignada al proceso. Si el proceso referencia a una página incorrecta, se genera un error y el proceso termina.

- Si el acceso fue correcto, se busca en la tabla de páginas el frame correspondiente, verificando el bit de validez-invalidez.

- Si el bit es de validez se accede al frame correspondiente y se termina el acceso.

- Sino se genera el trap de page fault, que involucra los siguientes pasos:
    a. Se busca frame libre en memoria principal, si no hay se ejecuta el algoritmo de reemplazo.
    b. Se lee de disco la página a cargar, y se carga en el frame obtenido en el paso anterior.
    c. Se actualiza la tabla de páginas, indicando que la página está disponible en memoria principal.
    d. Se devuelve el control a la instrucción que fue interrumpida por el PF.

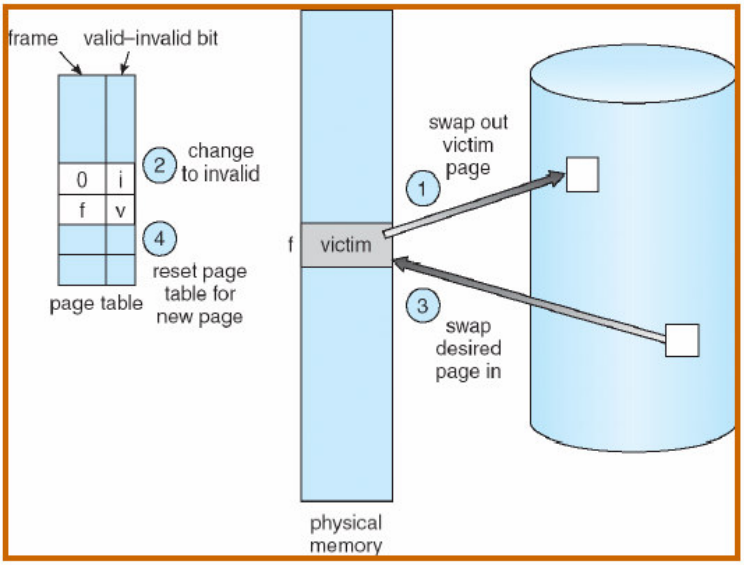
Si se aplica este método se tendrá un sistema puro de paginación bajo demanda. Tener en cuenta que para poder llevarlo a cabo se precisa una tabla de páginas y espacio swap de disco.

5. Algoritmos de remplazo

La necesidad de traer a memoria principal una página en una memoria principal llena, genera la búsqueda de un frame a remplazar, mediante un algoritmo de remplazo. El mal algoritmo de remplazo puede generar un impacto significativo de degradación del sistema, y es por esto que los estudiaremos.

Cuando se elige un frame a remplazar (la víctima), este será puesto en memoria swap, y ante un eventual uso en el futuro, volverá a memoria principal a través de un page fault. Los pasos a seguir cuando remplazamos frames son los siguientes:

- Elegir la víctima mediante algún algoritmo de remplazo
- Escribir la víctima en memoria swap (swap out) y ajustar la tabla de páginas
- Cargar la página en el frame correspondiente (swap in)
- Ajustar la tabla de página



Veamos ahora algunos algoritmos:

- **FIFO (First in First out)**

El algoritmo reemplaza la página que lleva más tiempo en memoria principal. Es un algoritmo fácil de implementar ya que requiere únicamente de una estructura tipo cola, pero remplaza las páginas sin tener en cuenta las referencias que tuvo.

- **Segunda Chance**

Este algoritmo intenta disminuir la cantidad de fallos de páginas del algoritmo FIFO, teniendo en cuenta las referencias a las páginas.
El algoritmo será igual al anterior, salvo que cada página tendrá un bit que indicará si fue o no referenciada luego de ser cargada a memoria. Al momento del remplazo, se verifica el bit de referencia; Si está encendido, a la página se le da una segunda chance y es puesta al final de la cola. Luego se continúa con la siguiente página que está al principio de la cola. Si el bit está apagado, esta página será seleccionada para ser remplazada.
Es un tanto ineficiente, pero disminuye la cantidad de fallos de páginas.

- **Óptimo**

En este algoritmo se reemplaza la página que no va a ser usada por el mayor periodo de tiempo. Es imposible de implementar porque requiere conocer a que páginas accederá el proceso.

- **LRU (Least Recently Used – recientemente menos usada)**

Este algoritmo asocia a cada página el tiempo en que fue referenciada. La página elegida por el algoritmo de remplazo será la que fue accedida hace más tiempo.
Este algoritmo es el que más se aproxima al óptimo y es bastante utilizado por los SO.

- **NRU (No Recientemente Usada)**

En este algoritmo a las páginas se les asigna un bit de referencia y otro de modificación. El bit de referencia se prende cada vez que se lee o escribe la página, mientras que el de modificación solo se prende cada vez que se escribe. Cada cierto tiempo el bit de referencia es apagado.
Al ocurrir un fallo de página, los frames son divididos en 4 clases. Se remplazará un frame al azar de la clase más baja que no esté vacía:

- Clase 0: No referenciada, no modificada
- Clase 1: No referenciada, modificada
- Clase 2: Referenciada, no modificada
- Clase 3: Referenciada, modificada

Al ejecutar el algoritmo de reemplazo, existen dos opciones de páginas a remplazar:

- Reemplazo global: Un proceso puede remplazar un frame utilizado por otro. Aunque los PF de un proceso afectan a otros, es el método más usado.
- Asignación local: Un proceso remplaza únicamente los frames que tiene asignado, es por eso que la cantidad de frames de un proceso no varía. La desventaja es que hay marcos que se pueden desperdiciar.

6. Asignación de frames a procesos e hiperpaginación

Si el SO no implementa una estrategia de asignación de memoria, un proceso que requiera mucha memoria puede hacer colapsar el sistema.

Una forma de asignar frames a procesos podría ser dividir la cantidad de frames del sistema en partes iguales para cada proceso. Este método puede ser ineficiente ya que no todos los procesos consumen la misma cantidad de memoria.

Si un proceso utiliza en forma activa una cantidad mayor de frames de los asignados por el sistema, tendrá un alto porcentaje de fallos de página, dando lugar a que el proceso esté continuamente realizando PF, pasando mas tiempo paginando que ejecutando, lo que se conoce como ***hiperpaginación***. Se degrada significativamente el rendimiento del sistema.