Curso: Sistemas Operativos
3. Administración de memoria
(Parte V)

M.C. Laura Sandoval Montaño

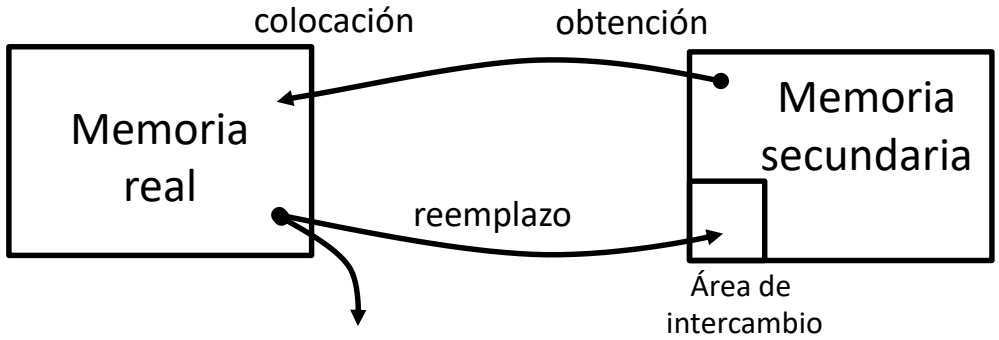
Estrategias en la administración de memoria virtual.

Una vez que ya estudiamos las diferentes organizaciones de la memoria virtual: paginación, segmentación y paginación/segmentación, veremos las estrategias de la administración de la memoria.

La memoria principal o real es un recurso caro del sistema, por lo que se debe administrar de la mejor manera para su mejor aprovechamiento.

Las estrategias de administración de la memoria se dividen en las siguientes categorías:

- *Estrategias de obtención*. Determinan cuándo debe obtenerse un bloque del programa o datos, de memoria secundaria para transferirse a memoria real
- *Estrategias de colocación*. Determinan en qué parte de la memoria principal se colocarán los bloques del programa o datos.
- *Estrategias de reemplazo*. Determinan qué parte de un proceso o datos debe desalojarse de la memoria real, para dejar espacio a los bloques entrantes.



Estrategias en la administración de memoria virtual.

Existe además, el concepto de *Conjuntos de Trabajo (Working Set),* el cual es una colección de páginas o segmentos a la que hace referencia activamente un proceso.

El Conjunto de trabajo real de un proceso es el conjunto de páginas que deben estar en la memoria real para que el proceso se ejecute con eficiencia. Es por eso que existen las *políticas de asignación*:

Tipos de algoritmos de asignación:

Asignación equitativa. Donde todos los procesos activos cuentan en memoria real con el mismo número de páginas o segmentos.

Asignación proporcional. Donde todos los procesos activos cuentan con el mismo porcentaje de páginas o segmentos en memoria real. Así, un proceso grande tiene más bloques en memoria real que uno más chico.

Asignación prioritaria. Donde, de acuerdo a la prioridad del proceso, es el número de páginas o bloques que tiene en memoria real. A más alta prioridad, mayor número de bloques tendrá en memoria real.

Ahora revisemos cada una de las estrategias:

Estrategias de Obtención:

Ya sea en un sistema de paginación o de segmentación, se tienen dos estrategias:

Por demanda.

Como el camino que toma un programa cuando se está ejecutando no es predecible, se cargan los bloques a medida que se necesitan, es decir cuando ocurra un fallo de página o segmento, según sea el caso.

Ventajas:

- Los bloques traídos son los que realmente se necesitan.
- La sobrecarga que implica la decisión de qué bloques traer al almacenamiento principal es mínima.

Paginación anticipada

Trata de evitar los retardos por fallos de página o segmento. Se cargan un cierto número de bloques con base en una predicción.

Ventajas:

- Si la predicción es buena, el tiempo de ejecución de los procesos se reduce considerablemente.
- Con la reducción de costes del hardware, las consecuencias de una mala predicción son menos graves.

Estrategias de Colocación:

En un sistema de paginación.

La estrategia de colocación de páginas, pareciera ser muy sencilla ya que la memoria real está dividida en marcos de página, por lo que no importaría en qué marco se almacenen las páginas. Sin embargo, existen casos particulares, como en Linux, donde es un poco más complejo. Lo revisaremos un poco más adelante.

En un sistema de segmentación.

La mayoría de las estrategias de colocación de segmentos, ya las hemos trabajado:

- Mejor ajuste
- Primer ajuste
- Peor ajuste

Ahora abordaremos el *algoritmo Buddy*, el cual se usa tanto para asignar memoria real como al liberar segmentos, reacomodando la memoria libre, evitando hacer tratamientos de huecos.

Estrategias de Colocación:

En un sistema de segmentación (continuación).

Funcionamiento del *algoritmo Buddy*:

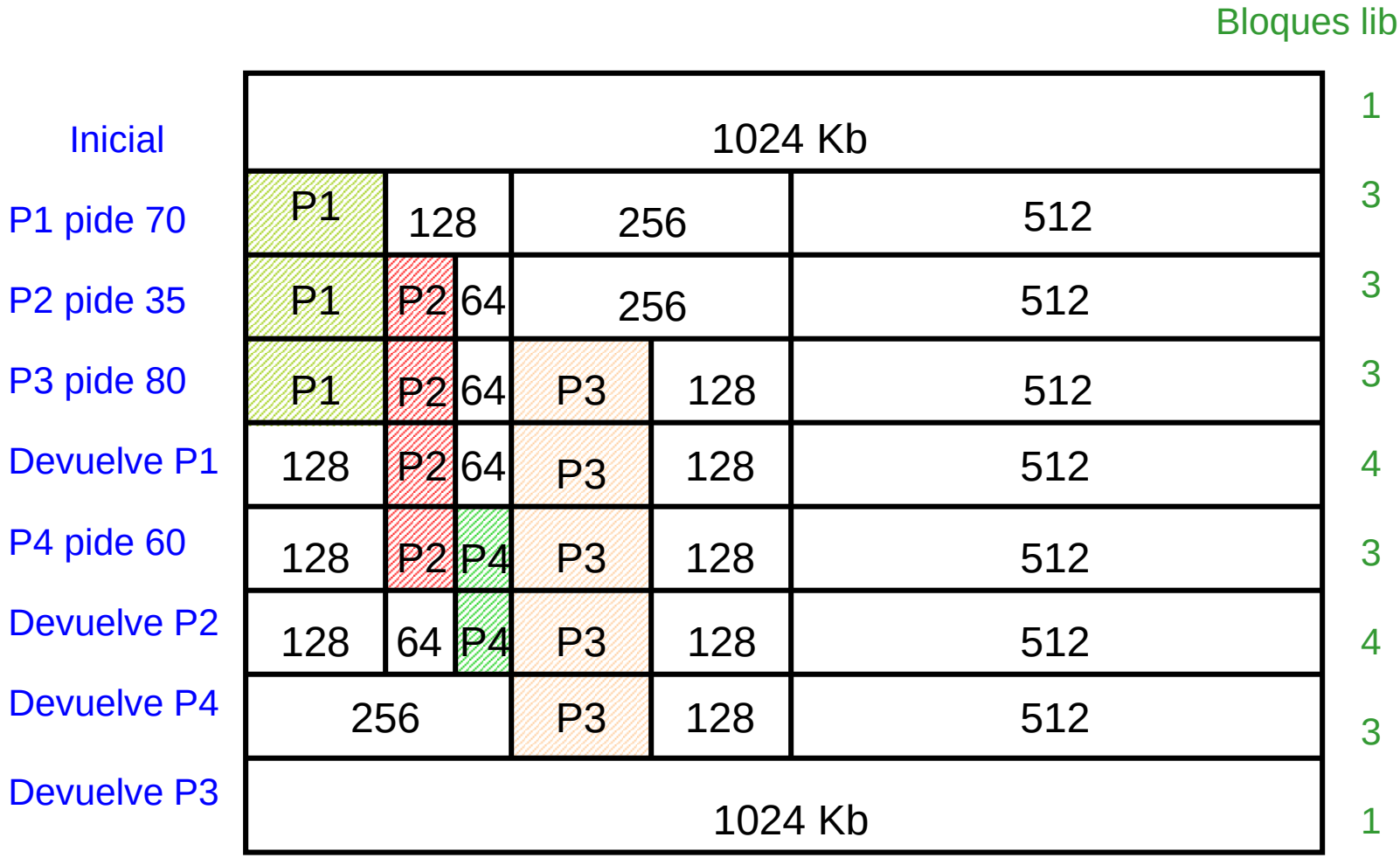
Divide la memoria sólo en bloques de 2^k KB, lo que trae la ventaja de conocer los posibles tamaños de los bloques cargados, pero tiene la desventaja de que todas las peticiones deben ser aproximadas a potencia de 2 sin pasarse. Por ejemplo: a un segmento de 35 KB se le debe asignar un bloque de 64 KB, siendo desperdiciados los restantes 29 KB (fragmento interno).

Cuando se carga un segmento en memoria, busca el primer bloque libre y:

- Si este bloque libre es del tamaño necesario para cargar al segmento (el valor 2^k próximo superior al tamaño), se carga.
- Si este bloque libre es de menor tamaño que el necesario, se busca el próximo bloque libre.
- Si este bloque libre es de tamaño superior que el necesario, se divide en dos partes iguales dicho bloque libre y se realizan todas estas verificaciones vistas.

Cuando un proceso libera el segmento porque finaliza o es descargado de memoria, las posiciones contiguas libres se unen formando un bloque de mayor tamaño. Para unirse los bloques libres, deben haber sido divididos juntos.

Ejemplo de colocación de segmentos por el algoritmo Buddy



Estrategias de Colocación:

Ejercicio tarea 1. Algoritmo Buddy:

Un cierto sistema maneja sistema de segmentación con colocación de segmentos utilizando el algoritmo Buddy.

El sistema cuenta con una memoria real de 2048 KB, inicialmente la memoria está vacía.

Indica cómo va haciendo la colocación y liberación de memoria de acuerdo a los siguientes sucesos, indicando además qué fragmentación interna se va generando.

Página	**acción**
1	Solicita 800k
2	Solicita 30k
3	Solicita 400k
4	Solicita 70k
1	Libera
5	Solicita 60k
3	Libera
2	Libera

Estrategias de Colocación:

Caso de estudio de colocación/liberación de páginas en Linux:

El algoritmo de colocación y liberación de páginas de Linux, utiliza el vector *free-area* para realizar dichas operaciones. Cada elemento de este vector apunta a una lista que indica qué bloques de marcos de página están libres. Así, el primer elemento del vector apunta a la lista de los bloques cuyo tamaño es de una página, el segundo, bloques de dos páginas, el tercero de cuatro páginas, etc., siempre bloques de tamaño potencia 2.

Teniendo esta organización de memoria, la colocación de páginas se hace a través del uso de una adaptación del algoritmo de Buddy. La diferencia está en que, en lugar de utilizar bloques de memoria en potencia de 2 KB, se forman bloques de número de páginas en potencia de 2; es decir, de 1,2,4,8,16… páginas.

Funcionamiento

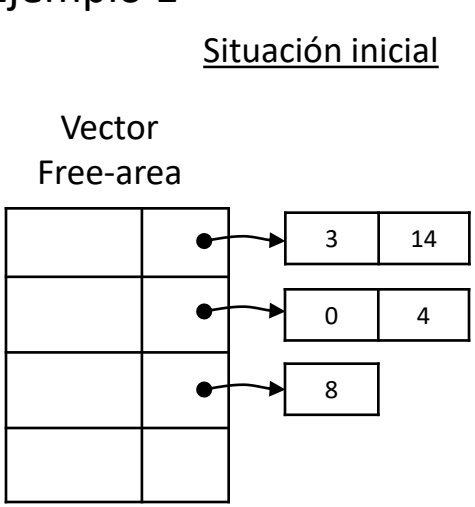
- El algoritmo de asignación primero busca bloques libres de igual tamaño al solicitado.
- Si no hay ninguno, examinará la lista de bloques libres del siguiente elemento del vector *free-area*. Así sucesivamente hasta encontrar un bloque libre.
- Si el bloque de páginas seleccionado es mayor que el solicitado, se deberá ir dividiendo en dos hasta conseguir uno de tamaño igual al deseado. Los bloques libres que se han generado, en esta división, son incluidos en las listas del vector *free-area* en función del tamaño que tengan.

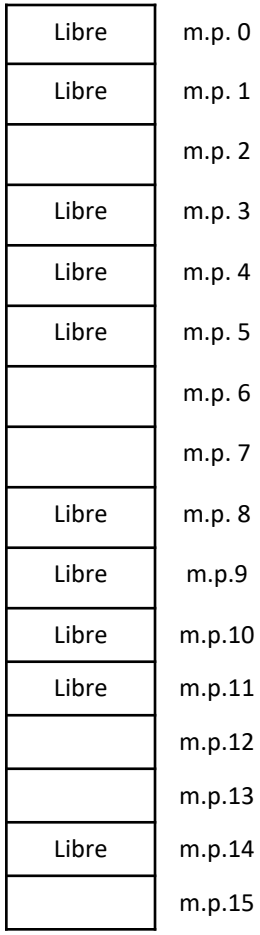
Estrategias de Colocación:

Caso de estudio de colocación/liberación de páginas en Linux:

Ejemplo 1

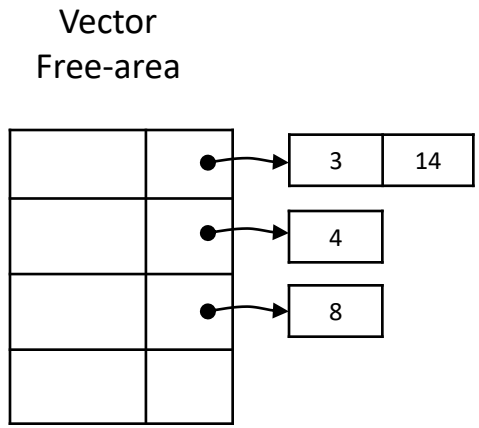
Situación inicial



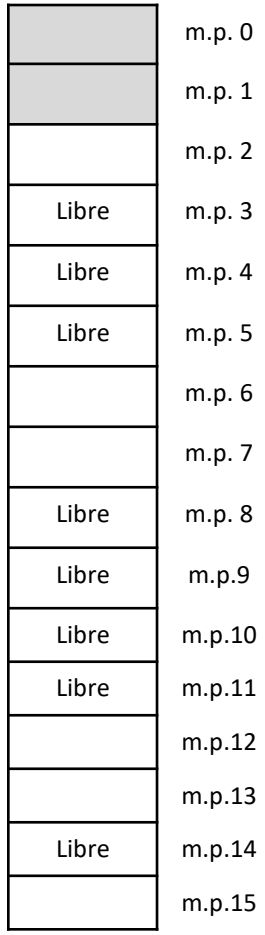


- Hay dos bloques de una página; están en los marcos 3 y 14.
- Hay dos bloques de dos páginas; inician en los marcos 0 y 4.
- Y hay un bloque de 4 páginas que inicia en el marco 8.

Se solicita colocar dos páginas contiguas



- Se asignan al bloque de dos páginas que inicia en el marco 0
- Se actualiza el vector free-area

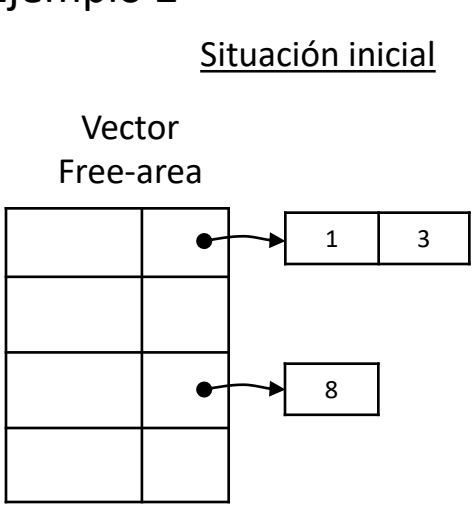


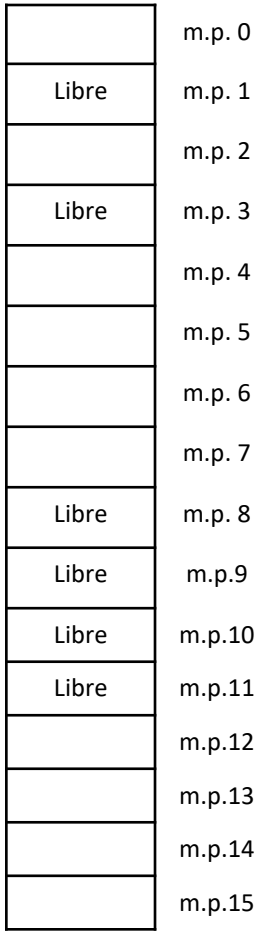
Estrategias de Colocación:

Caso de estudio de colocación/liberación de páginas en Linux:

Ejemplo 2

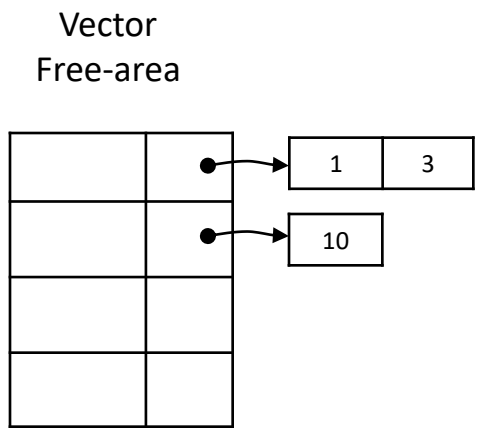
<u>Situación inicial</u>



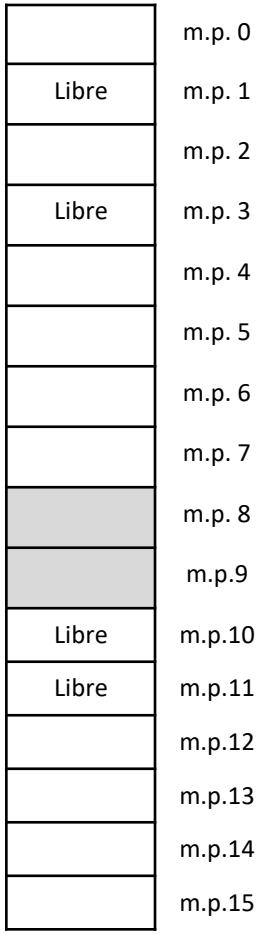


- Hay dos bloques de una página; están en los marcos 1 y 3.
- Y hay un bloque de 4 páginas que inicia en el marco 8.

Se solicita colocar dos páginas contiguas



- Como no hay bloques libres de dos páginas, se divide en dos el bloque de 4 páginas.
- Se asignan a uno de los bloques nuevos de dos páginas, por ejemplo al que inicia en el marco 8.
- Se actualiza el vector free-area

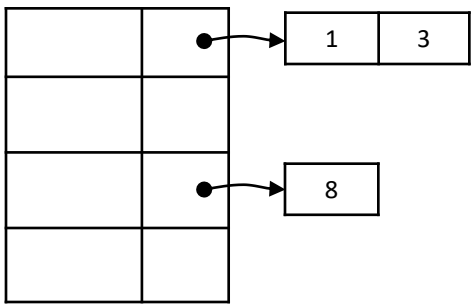


Estrategias de Colocación:

Caso de estudio de colocación/liberación de páginas en Linux:

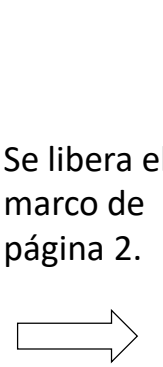
Ejemplo 3

Situación inicial

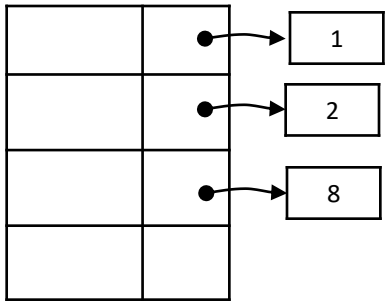


- Hay dos bloques de una página; están en los marcos 1 y 3.
- Y hay un bloque de 4 páginas que inicia en el marco 8.

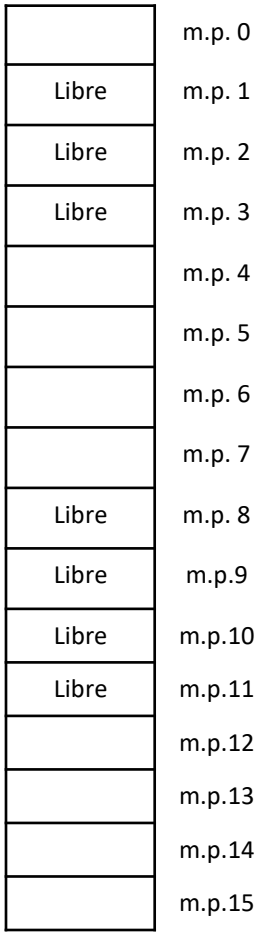
Se libera el marco de página 2.



Vector
Free-area



- Al quedar libre el m.p.2, se revisa si hay alguno libre contiguo del mismo tamaño.
- Como sí lo hay y corresponde con el que estaba unido antes de dividirse, forma un bloque de dos páginas.
- Se actualiza el vector free-area



Estrategias de Colocación:

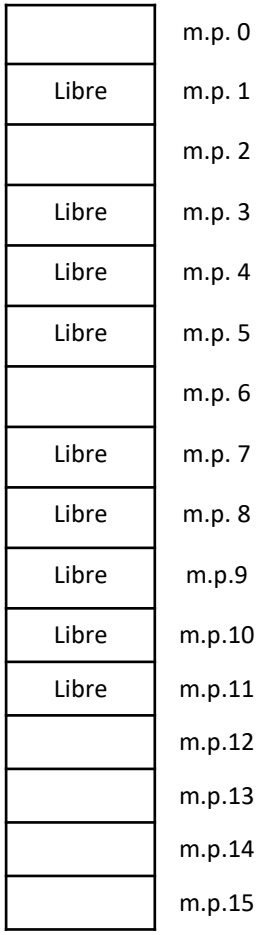
Caso de estudio de colocación/liberación de páginas en Linux:

Ejercicio tarea 2.

Con base en la situación mostrada de memoria real:

a) Muestra la situación del vector free-área
b) Cómo queda la situación tanto del free-área como de la memoria real después de realizar:
    i. Se solicita colocar 2 páginas contiguas
    ii. Se libera el marco de página 6

Nota: Recuerda que al unir bloques del mismo tamaño debieron estar juntos antes de hacer la división.



Estrategias de Reemplazo:

En un sistema de paginación.

El reemplazo de bloques radicados en memoria, se realiza cuando ésta está llena o haya sobrepasado un límite de ocupación definido por el sistema, y se requiere colocar un bloque entrante. Generalmente esto se da más en un sistema de paginación que de segmentación.

Entonces en el caso de paginación, el sistema de administración de memoria debe decidir qué pagina de memoria real se debe desplazar para dejar espacio a una página entrante, muchas ocasiones por la ocurrencia de fallos de página.

Si la página seleccionada a ser desplazada, fue modificada, deberá almacenarse en una área de intercambio, generalmente ubicada en memoria secundaria, para que al momento de ser requerida nuevamente se obtenga de esa área, ya que es la que contiene las modificaciones.

Si no se ha modificado, la página se deshecha y cuando sea requerida nuevamente, se obtiene la página virtual correspondiente de la memoria secundaria.

Lo óptimo es que la página seleccionada a ser desplazada, sea la que va a tardar más en requerirse, o bien, a ya no volver a utilizarse. A esto se le llama *Principio de optimización*

Estrategias de Reemplazo en un sistema de paginación.

Existen varias estrategias o algoritmos de reemplazo de páginas; veremos algunos.

Reemplazo de páginas aleatorio. Todas las páginas tienen la misma probabilidad de ser seleccionadas para desplazarlas. Gasto extra reducido, tanto en tiempo de procesamiento como en memoria.

Reemplazo de páginas de primeras entradas, primeras salidas (FIFO). Conforme las páginas van entrando a memoria, se van registrando en una cola. Cuando se requiera espacio en memoria real, se desplaza la página que está al inicio de la cola. Tiene la desventaja de que puede desplazar páginas muy utilizadas.

Reemplazo de páginas de la menos recientemente utilizada (LRU). Toma la página que no ha sido utilizada en mayor tiempo. Marca el instante en que se le hace referencia. Tiene la desventaja de que gasta tiempo de procesamiento en ver cuál tiene la hora más antigua de último uso.

Reemplazo de páginas de la menos frecuentemente utilizada (LFU). Se selecciona la página que ha tenido menos número de referencias en un mismo tiempo. Se utiliza un contador para cada página que indica cuántas veces se le ha hecho referencia. Cuando se desplaza una página, todos los contadores se resetean a 0, excepto a la que acaba de entrar, se le pone 1.

Estrategias de Reemplazo en un sistema de paginación.

Reemplazo de páginas de la no utilizada recientemente(NUR). Donde se evita que las páginas que han sido modificadas sean desplazadas ya que consumiría más tiempo en ser almacenadas en el área de intercambio, ubicada en memoria secundaria. Para ello, esta estrategia ocupa dos bits para cada página:

Un bit de referencia: 1 si ha sido referenciada.
Un bit de modificación: 1 si ha sido modificada.

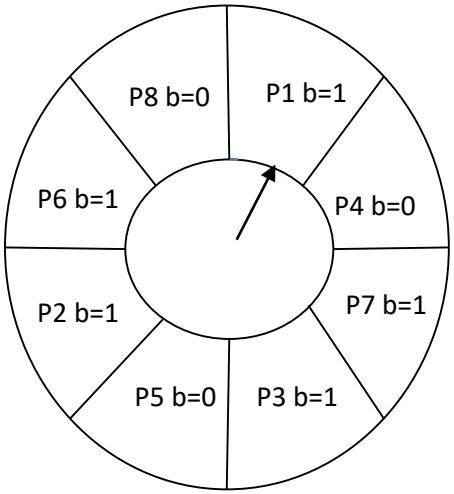
Por lo que esta estrategia selecciona primero una página que no ha sido referenciada ni modificada (bits: 00), si no hay ninguna en dicha situación, selecciona una página que ha sido referenciada pero no modificada (bits 01), si tampoco hay páginas en esa situación entonces selecciona una que no ha sido referenciada pero sí modificada (bits 10). Este último caso puede suceder que haya sido modificada cuando fue referenciada pero en algún momento se le realizó un *reset* al bit de referencia. Por último, que es muy raro que se presente esta situación, es cuando todas las páginas tienen sus dos bits en 1 (11), se opta por otra estrategia, que generalmente es la aleatoria.

El bit de referencia se puede resetear cuando se desplace una página.

Estrategias de Reemplazo en un sistema de paginación.

Reemplazo de páginas por reloj y segunda oportunidad. Estas son modificaciones a la estrategia FIFO, donde se incluye un bit de referencia que les da una segunda oportunidad a las páginas registradas en la cola. Si la página que está en la cabeza de la cola, tiene en 1 el bit de referencia, entonces se le cambia a 0 y se coloca al final de la cola; y así recorrer la cola hasta encontrar una página con el bit de referencia en 0.

El situar a una página, de la cabecera de la cola, al final de la cola, implica tiempo de procesamiento. Por lo que se opta por usar una cola circular, en donde sólo se sitúa el apuntador al siguiente elemento de la cola, dando como aspecto de un reloj, de ahí el nombre de esta estrategia.

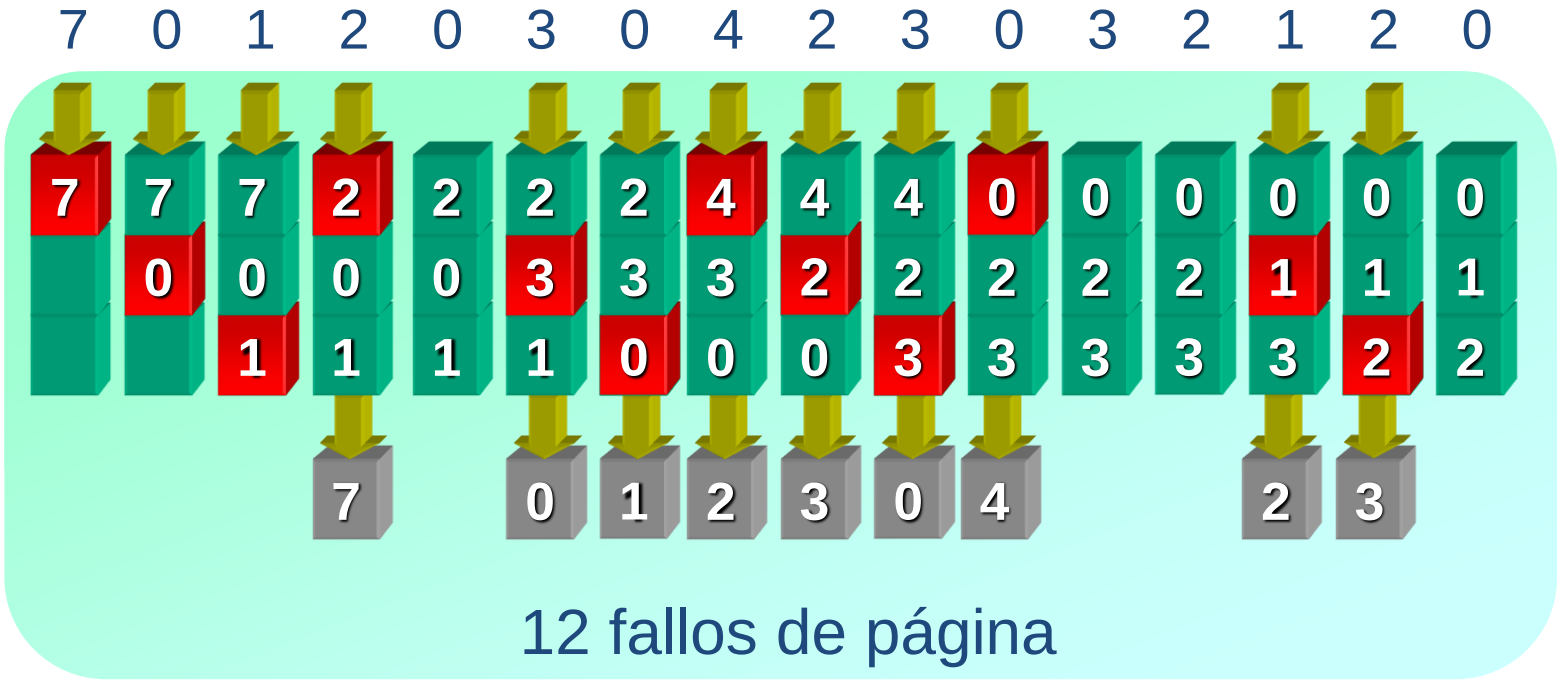


Para mostrar cómo trabajan diferentes estrategias de reemplazo de páginas, suponemos un sistema muy básico que maneja paginación, la memoria real consta de 3 marcos de página y se va requiriendo la colocación de una serie de páginas. Inicialmente la memoria real está vacía.

Ejercicio 1. Reemplazo de páginas óptimo.

Los fallos de página se muestran en rojo, indicando en qué marco se colocó la página. La estrategia consiste en reemplazar las páginas que tardarán más en ser referenciadas (indicadas en cubo gris cuando sale). No se puede implementar por no poder ver el futuro.

Ejercicio 2. Reemplazo de páginas Primeras entradas, primeras salidas (FIFO).



Los fallos de página se muestran en rojo, indicando en qué marco se colocó la página. La estrategia consiste en reemplazar las páginas conforme entraron a la cola (memoria real), indicadas en un cubo gris cuando sale.

Ejercicio 3. Reemplazo de páginas de la menos recientemente utilizada (LRU).

Los fallos de página se muestran en rojo, indicando en qué marco se colocó la página. La estrategia consiste en reemplazar las páginas que llevan mayor tiempo en no ser referenciadas (indicadas en un cubo gris cuando sale).

Ejercicio tarea 3.

Muestra cómo trabajan las estrategias de reemplazo de páginas indicadas, en un sistema muy básico que maneja paginación; la memoria real consta de 4 marcos de página y se va requiriendo la colocación de la siguiente una serie de páginas. Inicialmente la memoria real está vacía.

Serie de páginas solicitadas: 5 3 7 5 2 3 6 7 8 2 5 1 2 4

a) *Reemplazo de páginas óptimo.*

b) *Reemplazo de páginas Primeras entradas, primeras salidas (FIFO).*

c) *Reemplazo de páginas de la Menos recientemente utilizada (LRU).*