Memoria virtual:

El uso de la memoria virtual nos permite ejecutar procesos que sean de mayor tamaño al de la memoria disponible mediante la utilización de la memoria secundaria (unidades de disco). Al momento de la carga del proceso se analizará si es cargado completamente en memoria o si es almacenado en una porción del disco (llamada swap) y cargado parcialmente en memoria mediante intercambios. La traducción de direcciones virtuales a reales es manejada por la MMU (Unidad de manejo de memoria) mediante el uso de una tabla de paginación.

Cuando un proceso necesita acceder a una página que no se encuentra cargada en la memoria principal, se dispara una excepción llamada “Page fault” (fallo de página). Si hay espacio disponible en memoria, se carga la página y continua la ejecución, en caso contrario habrá que desalojar una página de la memoria. Para esto, primero se selecciona la víctima, la cual dependerá del algoritmo de reemplazo de páginas que estemos utilizando. Si la víctima fue modificada, se guardará en el disco su estado actual y será desalojada de la memoria. Luego se evalúa si la página buscada se encuentra aún en el disco, en caso contrario se dispara una excepción que aborta por no tener una dirección válida. Si se encuentra en el disco, se carga en memoria, se actualiza la tabla de direcciones, se envía la dirección física y se ejecuta.



Algoritmos de selección de víctima:

**FIFO**: Se guardan las páginas en el orden que fueron cargadas y al momento de seleccionar una víctima se elige a la que más tiempo estuvo en memoria.

EJ:

Peticiones: 2, 1, 3, 0, 2, 1, 4, 2, 1, 3, 0 y 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #1 | 2 | 2 | 2 | 0 | 0 | 0 | 4 | 4 | 4 | 4 | 4 | 4 |  |
| #2 |  | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |  |
| #3 |  |  | 3 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |  |
|  | x | x | x | x | x | x | x | - | - | x | x | - | 9 page fault |

Cada x indica cuando se produjo un page fault. Por lo que en este ejemplo hubo 9 page

**Visión de pila**:

FIFO también puede ser resuelto como pila donde las nuevas páginas van empujando a las páginas viejas y al llegar al ser empujada del último frame, la página es desalojada de la memoria.

EJ: (Mismas peticiones que el ejemplo anterior).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #1 | 2 | 1 | 3 | 0 | 2 | 1 | 4 | 4 | 4 | 3 | 0 | 0 |  |
| #2 |  | 2 | 1 | 3 | 0 | 2 | 1 | 1 | 1 | 4 | 3 | 3 |  |
| #3 |  |  | 2 | 1 | 3 | 0 | 2 | 2 | 2 | 1 | 4 | 4 |  |
|  | x | x | x | x | x | x | x | - | - | x | x | - | 9 page fault |

El algoritmo FIFO puede dar lugar a un efecto llamado “**Anomalía de Belady**” (en honor a su descubridor Laszlo Belady) en la que al aumentar el número de frames de la memoria incrementa también el número de page fault. Antes de ser descubierta esa anomalía se creía que al incrementar el número de frames se reduciría el número de page fault o, en el peor de los casos, se mantendría igual.

Este efecto ocurre si al ejemplo anterior le incrementamos el número de frames a 4:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #1 | 2 | 1 | 3 | 0 | 0 | 0 | 4 | 2 | 1 | 3 | 0 | 4 |  |
| #2 |  | 2 | 1 | 3 | 3 | 3 | 0 | 4 | 2 | 1 | 3 | 0 |  |
| #3 |  |  | 2 | 1 | 1 | 1 | 3 | 0 | 4 | 2 | 1 | 3 |  |
| #4 |  |  |  | 2 | 2 | 2 | 1 | 3 | 0 | 4 | 2 | 1 |  |
|  | x | x | x | x | - | - | x | x | x | x | x | x | 10 page fault |

**LRU**: El algoritmo LRU (siglas de Least Recently Used, que en español significa “menos recientemente usado”) es similar al FIFO, solo que en caso de que una página que se encuentra cargada vuelve a ser utilizada, pierde prioridad como víctima.

EJ:

Peticiones: 2, 1, 3, 2, 0, 1, 2, 4, 1, 3, 0 y 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #1 | 2 | 1 | 3 | 2 | 0 | 1 | 2 | 4 | 1 | 3 | 0 | 4 |  |
| #2 |  | 2 | 1 | 3 | 2 | 0 | 1 | 2 | 4 | 1 | 3 | 0 |  |
| #3 |  |  | 2 | 1 | 3 | 2 | 0 | 1 | 2 | 4 | 1 | 3 |  |
|  | x | x | x | - | x | x | - | x | - | x | x | x | 9 page fault |

Nótese como en la cuarta, séptima y novena petición se movió la página hacia arriba en lugar de dejarla en la posición que ocupaba, como es el caso de FIFO.

**Óptimo**: El algoritmo óptimo es aquel que sea capaz de seleccionar como víctima a la página más lejana en la cola. Este algoritmo es solo teórico ya que es imposible implementarlo debido a que necesitaría tener conocimiento del futuro. Ej:

Peticiones: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0 y 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #1 | 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 |  |
| #2 |  | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 4 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| #3 |  |  | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |
|  | x | x | x | x | - | x | - | x | - | - | x | - | - | x | - | - | - | x | - | - | 9 page fault |

**FIFO 2da oportunidad**: Este algoritmo es una mejora del algoritmo FIFO en el que se le asigna un bit de referencia a las páginas que se pone en 1 al memento de ser referenciada. Al momento de seleccionar una víctima, al igual que en el FIFO, se empieza evaluando por la página más vieja en el sistema, si el bit de referencia se encuentra en 1, se le perdona la vida y se pasa a evaluar la siguiente hasta encontrar alguna con el bit de referencia en 0. Al actualizar su bit a 0, la página se tomará como si hubiera sido nuevamente referenciada, por lo que debe considerarse su vejez a partir de ese momento. Cuando una página que ya se encuentra en memoria vuelve a ser utilizada, su bit de referencia se pone en 1. Ej:

Peticiones: 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5 y 2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #1 | 21 | 21 | 21 | 21 | 51 | 51 | 51 | 51 | 31 | 31 | 31 | 31 |  |
| #2 |  | 31 | 31 | 31 | 30 | 21 | 21 | 21 | 20 | 21 | 20 | 21 |  |
| #3 |  |  |  | 11 | 10 | 10 | 41 | 41 | 40 | 40 | 51 | 51 |  |
|  | x | x | - | x | x | x | x | - | x | - | x | - | 8 page fault |

**FIFO 2da oportunidad mejorado**: Es como el FIFO 2da oportunidad pero con dos bits en lugar de uno, el de referencia y el de modificación. En este caso el bit de referencia es exactamente igual al anterior y el bit de modificación empieza en 0 y se pone en 1 si la página es modificada. Al momento de seleccionar una víctima se evalúan ambos bits.

|  |  |  |
| --- | --- | --- |
| Bit referencia | Bit modificación |  |
| 0 | 0 | No referenciada, no modificada (víctima ideal). |
| 0 | 1 | No referenciada, modificada (cuando el bit se pone en 0 por la 2da oportunidad). |
| 1 | 0 | Referenciada, no modificada. |
| 1 | 1 | Referenciada, modificada. |

**Byte de referencia**: En este algoritmo no solo se tiene en cuenta cuando una página es referenciada sino también la frecuencia con la que es referenciada. Para esto hace uso de un byte, por cada tick del reloj se desplazan los bits a la derecha y se pone al principio un 1 en caso de haber sido referenciada o un 0 en caso contrario.

Por ejemplo si los bits de una página fueron 1, 0, 0, 1, 1 y 0 en los últimos 6 ticks del reloj, el contador se verá de la siguiente manera:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tick | Byte de ref. | | | | | | | |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |