**Disk Cache**

* Buffers en memoria RAM para almacenamiento temporal de bloques de disco.
* Objetivo 🡪 minimizar la frecuencia de acceso al disco.

**Observaciones**

* Cuando un proceso quiere acceder a un bloque de la cache hay dos alternativas:
  + Se copia el bloque al espacio de direcciones del usuario 🡪 no permite compartir el bloque.
  + Se trabaja el bloque como memoria compartida 🡪 permite acceso a varios procesos.
    - Dicha área compartida debe ser limitada, con lo cuál debe existir un algoritmo de reemplazo.

**Estrategia de reemplazo**

* Cuando se necesita un buffer para cargar un nuevo bloque, se elige el bloque que hace más tiempo no es referenciado.
* Consiste en una lista de bloques donde el último es el más recientemente usado (LRU).
* Cuando un bloque es referenciado o entra en la cache se queda al final de la lista.
* No se mueven los bloques en la memoria: se asocian punteros.
* Otra alternativa: LFU, se reemplaza el que tenga menor número de referencias.

**Objetivo y estructura de un Buffer Cache**

* Minimizar la frecuencia de acceso a disco.
* Estructura que se forma por buffers.
* El kernel asigna un espacio en la memoria durante la inicialización de dicha estructura.
* El buffer se compone de dos partes:
  + Header: contiene información del bloque, número del bloque, estado, relación con otros buffers, etc.
  + Buffer en sí: el lugar donde se almacena el bloque de disco traído a memoria.

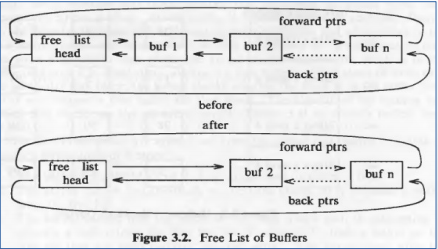
**Header**

* Nro de dispositivo, nro de bloque.
* Estado.
* Punteros:
  + 2 punteros de hash queue.
  + 2 punteros para la free list.
  + 1 puntero al bloque en memoria (donde se ubica).

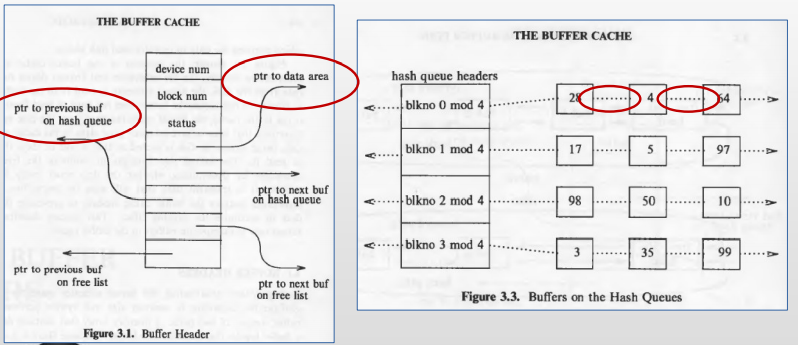
**Estados de los buffers**

* Free (disponible).
* Busy (no disponible, en uso por algún proceso).
* Escribiendo o leyendo del disco.
* Delayed Write(DW): buffers que fueron modificados en memoria, pero los cambios no se han reflejado en el bloque original de disco.

**Free list**

* Organiza los buffers disponibles para ser utilizados para cargar nuevos bloques de disco.
* No necesariamente dichos buffers disponibles están vacíos (el proceso puede que haya terminado y liberado el bloque, pero el buffer puede seguir en estado DW).
* Es ordenada según LRU.
* 

Hash queues

* Colas que buscan optimizar la búsqueda de un buffer en particular.
* Los headers de los buffers se organizan según una función de Hash (Dispositivo, #bloque).
* A esto se le aplica una función de hash que permite agrupar los buffers cuyo resultado dio igual. Esto permite la búsqueda eficiente.
* La agrupación y enganche de cada buffer con otro en cada cola se realiza mediante los punteros que se almacenan en el header
  + 

**Free list (continuando)**

* Sigue el mismo esquema de la Hash queue pero contiene los headers de los buffers de aquellos procesos que **ya han terminado**.
* El header de un buffer siempre está en la Hash Queue.
* Si el proceso que lo referenciaba terminó, va a estar en la Hash Queue y en la Free list.

**Funcionamiento del buffer cache**

* Cuando un proceso requiere el acceso a un archivo, utiliza su inodo para localizar los bloques de datos donde se halla el mismo.
* El requerimiento llega al buffer cache quien evalúa si puede satisfacer el requerimiento o si debe realizar la E/S.