Resumen Parcial 2

LCC

2021

Índice

1. Memoria compartida

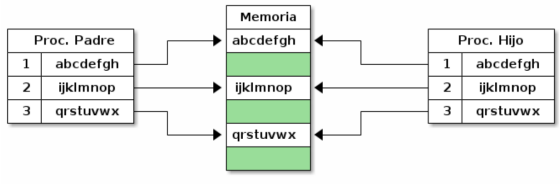
En la comunicación entre procesos (IPC) o cuando se ejecuta más de una vez un programa (sin código automodificable) no tiene sentido que las estructuras compartidas en el caso de los procesos o las páginas asociadas a cada instancia en el caso de muchas instancias de un mismo programa ocupen un marco independiente.

La idea también aplica a las bibliotecas de sistema (libc, openssl, zlib, libpng): las bilbiotecas suelen ser empleadas por una gran cantidad de programas.

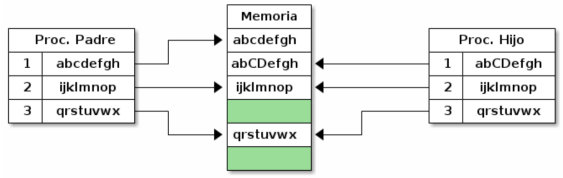
En general, cualquier programa que esté desarrollado y compilado de forma de que todo su código sea de solo lectura permite que otros procesos entren en su espacio de memoria sin tener que sincronizarse con otros procesos que lo estén empleando.

1.1. CoW

En los sistemas Unix, cuando un proceso llama a fork() se crea un proceso hijo idéntico al padre con el cual se comparten todas las páginas en memoria. Para garantizar que la memoria no se modifique si no es por los canales explícitos de comunicación entre procesos, se usa el mecanismo copiar al escribir (CoW): mientras las páginas sean de solo lectura, las páginas de padre e hijo son las mismas pero ni bien cualquiera de los procesos modifica alguna de las páginas, se genera un fallo de página. El sistema operativo, en vez de terminar el proceso debido al fallo de página que se produce al querer escribir en un bloque de solo lectura, copia la página en la que se encuentra la dirección de memoria que causó el fallo y marca esta nueva página como de lectura / escritura. Esta nueva página deja de ser compartida.



(a) Memoria luego del fork



(b) Memoria luego de la modificación de la primer página

2. Memoria virtual

En un sistema que emplea paginación, un proceso no conoce su dirección en memoria relativa a otros procesos, sino que trabajan con una idealización de la memoria, en la cual ocupan el espacio completo de direccionamiento, desde el cero hasta el límite lógico de la arquitectura, independientemente del tamaño físico de la memoria disponible.

Para ofrecer a los procesos mayor espacio en memoria del que se cuenta físicamente, el sistema emplea espacio en almacenamiento secundario (típicamente, disco duro), mediante un esquema de intercambio (*swap*) guardando y trayendo páginas enteras. La memoria es gestionada de forma **automática** y **transparente** por el sistema operativo (lxs usuarixs no saben de páginas). El encargado de esto es el *paginador*, no el intercambiador (*swapper*).

2.1. Demand Loading - Read ahead - Manejo de buffer

Para ejecutar un programa, sus páginas tienen que estar en memoria principal. Existen varias alternativas para cargar las páginas de un programa

- Carga completa: si el proceso entra completo en memoria entonces se cargan todas sus páginas. El programa tardará en comenzar ya que se estará copiando su tabla de páginas completa a memoria, pero una vez que todas sus páginas estén cargadas se generarán menos fallos de página* lo cual ahorrará en accesos al disco.

- Carga bajo demanda pura: no se carga por anticipado ninguna página en memoria. Cada vez que se quiere acceder a una página se lanza un fallo que el SO maneja. Al cargar páginas bajo demanda se minimiza la cantidad de memoria principal ocupada lo cual puede permitir un mayor grado de multiprogramación. De cualquier forma, los accesos al disco que se producen al manejar los fallos de página penalizan fuertemente la ejecución de los procesos. Ésta técnica de paginación ciertamente no es apta para procesos con requerimientos temporales.
- Carga bajo demanda parcial: se cargan ciertas páginas de un programa de antemano y por cada página que no se encuentra en memoria principal se genera un fallo que maneja el SO. Se puede combinar con la técnica read ahead, en la cual se cargan por adelantado ciertas páginas que tienen muchas chances de ser usadas próximamente. Combina las ventajas de las 2 técnicas anteriores.

* Cuando un proceso está completamente cargado en memoria se pueden seguir produciendo fallos de páginas cuando el stack crece más alla del espacio asignado originalmente o cuando se debe recuperar una página del área de intercambio.

Algunos SO buscan proactivamente tener memoria libre cuando el procesador está idle o cuando se están esperando operaciones de E/S. Esto lo hace buscando las páginas más proclives a ser llevadas al disco y transfiriéndolas en caso de que hayan sido modificadas. Cuando se produzca un fallo, se podrá cargar la nueva página en disco sin necesidad de esperar a que se lleve a disco la víctima elegida.

2.2. Asignación de marcos

2.2.1. Conjunto mínimo / activo de marcos

El **conjunto mínimo de marcos** hace referencia a la mínima cantidad de marcos que se deben cargar para ejecutar la siguiente instrucción de un proceso: muchas instrucciones requieren tener más de un marco cargado para poder ejecutarse (cada instrucción puede causar más de un fallo de página). Una suma por ejemplo requerirá que se carguen 4 páginas en memoria: una para el flujo del programa, otras dos para los operandos y una última para el resultado. Si se cargan menos de 4 páginas no va a poder avanzar con su ejecución (ver hiperpaginación).

El **conjunto activo de marcos** hace referencia al conjunto de páginas que un proceso está actualizado *actualmente*. Este conjunto debe ser necesariamente un subconjunto de los procesos que están cargado en memoria en ese mismo momento.

2.2.2. Esquemas de asignación

Los marcos se pueden asignar a los distintos procesos según diferentes esquemas:

- Asignación floja (*lazy*): se asignan marcos a medida que los procesos lo necesitan. Esto puede aumentar el grado de multiprogramación pero se penaliza la ejecución de todos los procesos al haber tantas operaciones de E/S.
- Asignación ansiosa (*eager*): al asignar marcos de antemano disminuye el grado de multiprogramación pero el rendimiento de los procesos será mejor.
    - Asignación igualitaria: se asigna la misma cantidad de páginas para cada proceso. Es un esquema de asignación deficiente ya que diferentes procesos tienen diferentes requerimientos de memoria.
    - Asignación proporcional: se asigna una cantidad proporcional al uso de memoria virtual de un proceso. Existen algunas desventajas:
        - Se debe cuidar que ningún proceso debe tener asignado menos que un mínimo de marcos definido (por tener requerimientos modestos) o más de un máximo (por tener requeriminetos desproporcionales).
        - Ignora las prioridades que normalmente tienen los procesos aunque se podría agregar al cálculo para la asignación.
        - Cada vez que cambia el nivel de multiprogramación (se agrega o finaliza un proceso a la cola de ejecución) se deben recalcular las asignaciones.
        - Desperdicia recursos. Muchos procesos trabajan con períodos inestables de actividad en los cuales por momento trabajan en un ráfaga y por momentos tienen requerimientos mínimos.

2.2.3. Tipos de reemplazo de páginas

Los algoritmos de reemplazo de páginas pueden operar en distintos ámbitos

- Reemplazo local: se reemplaza una página por otra del mismo proceso, con lo cual se preserva la cantidad de marcos asignado para el proceso. Pero es demasiado rígido como para aprovechar los períodos de inactividad de algunos procesos y así mejorar el rendimiento global.

- Reemplazo global:

  Reemplazo global simple: se elige para reemplazar una página víctima de entre todas las que están en memoria, probablemente priorizando páginas de procesos que tengan muchas páginas de memoria asignadas.

  Reemplazo global con prioridad: es un esquema mixto en el que cada proceso puede sobrepasar su asignación de marcos siempre y cuando se elijan de páginas víctima, páginas de un proceso de menor prioridad.

  Los reemplazos globales pueden mejorar el rendimiento global del sistema pero afectan la asignación de marcos de los procesos. Además, pueden producir rendimiento inconsistente: una misma sección de código puede presentar tiempos de ejecución muy diferentes dependiendo de los otros procesos que están en ejecución.

2.3. Hiperpaginación - *thrashing*

La hiperpaginación se puede producir por 2 situaciones muy distinguidas:

- bajo un esquema de reemplazo local un proceso tiene asignadas pocas páginas lo cual hace que constantemente se generen fallos y no se pueda avanzar
- bajo un esquema de reemplazo global, un proceso al inicializarse genera algunos fallos de página para cargar sus estructura iniciales, mientras que otros procesos que estaban corriendo solicitan páginas que se acaban de llevar al disco (swap) producto de la inicialización del primero.

Para solucionar este problema la solución más evidente es reducir el grado de multiprogramación: el sistema puede seleccionar uno o más procesos y suspenderlos hasta que el sistema vuelva a un estado normal. Para esta selección por lo general se tienen en cuenta procesos de baja prioridad, procesos que producen gran cantidad de fallos y/o procesos que están ocupando mucha memoria.

2.4. Algoritmos de reemplazo de páginas

Al sobre-comprometer memoria los procesos que están en ejecución eventualmente van a requerir cargar en memoria física más páginas de las que caben. Si todos los marcos están ocupados, el sistema deberá encontrar una página que pueda liberar (una página víctima) y llevarla al espacio de intercambio en el disco. Ésto da lugar a diferentes algoritmos de reemplazo de páginas.

- FIFO: se elige de víctima a la página que haya sido cargada hace más tiempo

Pagina	1	2	3	4	1	2	5	1	2	3	4	5
Marcos	1	1	1	1	1	1	5	5	5	5	4	4
		2	2	2	2	2	2	1	1	1	1	5
			3	3	3	3	3	3	2	2	2	2
				4	4	4	4	4	4	3	3	3
M/H	M	M	M	M	H	H	M	M	M	M	M	M

Tabla 1: Política FIFO en un sistema de 4 páginas físicas

Ventaja: fácil de implementar y comprender

Desventajas: 1) no tiene en cuenta el historial de solicitudes, todas las páginas tienen la misma probabilidad de ser reemplazadas, 2) es vulnerable a la anomalía de Belady

- OPT / MIN: se elige de víctima la página que no vaya a ser utilizada por el máximo tiempo. Este algoritmo es de interés teórico ya que no es posible predecir el orden en el que se van a requerir las páginas pero da una cota mínima para los otros algoritmos

Traza	1	2	3	4	1	2	5	1	2	3	4	5
P1	1	1	1	1	1	1	1	1	1	1	4	4
P2	-	2	2	2	2	2	2	2	2	2	2	2
P3	-	-	3	3	3	3	3	3	3	3	3	3
P4	-	-	-	4	4	4	5	5	5	5	5	5
M/H	M	M	M	M	H	H	M	H	H	H	M	H

Tabla 2: Política FIFO en un sistema de 4 páginas físicas

- LRU: se elige de víctima la página menos usada recientemente (la que no se usó hace más tiempo). El resultado de evaluar una cadena S empleando LRU es equivalente a emplear OPT sobre su inversa.

Traza	1	2	3	4	1	2	5	1	2	3	4	5
P1	1	1	1	1	1	1	1	1	1	1	1	5
P2	-	2	2	2	2	2	2	2	2	2	2	2
P3	-	-	3	3	3	3	5	5	5	5	4	4
P4	-	-	-	4	4	4	4	4	4	3	3	3
M/H	M	M	M	M	H	H	M	H	H	M	M	M

Tabla 3: Política FIFO en un sistema de 4 páginas físicas

Ventajas: 1) es una buena aproximación a OPT, 2) está libre de la anomalía de Belady

Desventajas: requiere apoyo de hardware (es mucho más complejo que FIFO[1])

Dada la complejidad que presenta LRU en hardware, se utilizan aproximaciones a este algoritmo.

- Bit de referencia: se utiliza un bit extra por página, que se prende cuando la página se utiliza. Periódicamente el SO apaga todos los bits. En caso de un fallo, se elige por FIFO sobre el subconjunto de marcos que no fueron referenciados en el período actual (desde el último resetteo de bits).

Traza	1	2	3	-	4	1	2	5	1	-	2	3	4	5
p1	1	1	1		1	1	1	1	1		1	3	3	3
p2	-	2	2		2	2	2	2	2		2	2	2	2
p3	-	-	3		3	3	3	5	5		5	5	5	5
p4	-	-	-		4	4	4	4	4		4	4	4	4
M/H	M	M	M		M	H	H	M	H		H	M	H	H
b1	1	1	1	0	0	1	1	1	1	0	0	1	1	1
b2	0	1	1	0	0	0	1	1	1	0	1	1	1	1
b3	0	0	1	0	0	0	0	1	1	0	0	0	0	1
b4	0	0	0	0	1	1	1	1	1	0	0	0	1	1

Tabla 4: Política Bit de referencia en un sistema de 4 páginas físicas y sus bits de referencia luego del tick

Ventaja: utiliza solamente 1 bit extra por página

Desventajas: si los resetteos se producen muy frecuentemente o bien con muy poca frecuencia degenera en un FIFO

- Columna de referencia: funciona igual que el anterior solo que utiliza más de un bit de referencia. El resetteo periódico ahora consiste en un desplazamiento de bits hacia la derecha. En caso de fallo, se elige por FIFO sobre el subconjunto de marcos con menor cadena de referencia.

Traza	1	2	3	-	4	1	2	5	1	-	2	3	4	5
p1	1	1	1		1	1	1	1	1		1	1	1	5
p2	-	2	2		2	2	2	2	2		2	2	2	2
p3	-	-	3		3	3	3	5	5		5	5	4	4
p4	-	-	-		4	4	4	4	4		4	3	3	3
M/H	M	M	M		M	H	H	M	H		H	M	M	M
c1	100	100	100	10	10	110	110	110	110	11	11	11	11	100
c2	0	100	100	10	10	10	110	110	110	11	111	111	111	111
c3	0	0	100	10	10	10	10	100	100	10	10	10	100	100
c4	0	0	0	0	100	100	100	100	100	10	10	100	100	100

Tabla 5: Política Columna de referencia en un sistema de 4 páginas físicas y sus bits de referencia luego del tick

[1]Tiene que recorrer todas las páginas si se usa un contador por página y encontrar la máxima; o actualizar reiteradas veces una lista doblemente enlazada para acceder a la víctima en tiempo constante

Ventaja: utiliza una cantidad fija de bits por página. Entre más bits, más parecido será al LRU.

Desventajas: si los resetteos se producen muy frecuentemente o bien con muy poca frecuencia degenera en un FIFO pero tendrá mejor desempeno que el algoritmo con un solo bit de referencia

- Segunda oportunidad (reloj): se mantiene un puntero a la próxima víctima. En caso de fallo, si el marco al que apunta el puntero está apagado entonces se elimina la página y en otro caso se apaga su bit de referencia y se avanza el puntero.

  En la tabla (6) el guión indica la posición del puntero.

Traza	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1-	1-	1-	5	5	5	5-	4	4
2	-	2	2	2	2	2	2-	1	1	1	1-	5
3		-	3	3	3	3	3	3-	2	2	2	2
4			-	4	4	4	4	4	4-	3	3	3
M/H	M	M	M	M	H	H	M	M	M	M	M	M
1	1	1	1	1	1	1	1	1	1	1	1	1
2		1	1	1	1	1	0	1	1	1	0	1
3			1	1	1	1	0	0	1	1	0	0
4				1	1	1	0	0	0	1	0	0

Tabla 6: Política reloj en un sistema de 4 páginas físicas y sus bits de referencia luego del tick

Ventaja: utiliza solamente 1 bit extra por página

Desventajas:en el peor caso degenera en un FIFO con una vuelta extra

- Segunda oportunidad mejorada (reloj mejorado): se amplía el algoritmo de bit de referencia con un bit de modificación que busca disminuir la cantidad de operaciones E/S.

  En la tabla (7) los * indican que la página se escribe y el guión indica la posición del puntero.

Traza	1*	2	3*	4	1	2	5*	1	2	3	4	5
1	1	1	1	1-	1-	1-	1	1	1-	1-	1-	1-
2	-	2	2	2	2	2	5	5	5	5	5	5
3		-	3	3	3	3	3-	3-	3	3	3	3
4			-	4	4	4	4	4	2	2	4	4
M/H	M	M	M	M	H	H	M	H	M	H	M	H
1	11	11	11	11	11	11	01	11	11	11	01	01
2	00	10	10	10	10	10	11	11	11	11	01	11
3	00	00	11	11	11	11	01	01	01	11	01	01
4	00	00	00	10	10	10	00	00	10	10	10	10

Tabla 7: Política reloj mejorado en un sistema de 4 páginas físicas y sus bits de referencia luego del tick

Ventaja: utiliza solamente 2 bits extra por página

Desventajas: puede requerir hasta 2 vueltas elegir una página víctima (en la práctica, con algunas simplificaciones).

3. Memoria virtual de GNULinux (apunte)

Linux implementa **memoria compartida entre procesos** e hilos, **Copy On Write** y soporta **carga por demanda**. Además implementa lectura por adelantado (***readahead***): al leer un bloque se leen los bloques siguientes y se guardan en el espacio cache de memoria. Éstas lecturas no se podrían hacer tan rápido si no existieran las cache de disco.

Mapeo de archivos en memoria (*memory mapped files*) Un proceso puede solicitar que un espacio de memoria virtual se refiera a un archivo. Escribir o leer un archivo consiste en escribir en memoria. Cuando la página se libera se sincronizan los datos en disco pero no se lleva a área de intercambio.

NUMA Linux suporta Acceso no uniforme a Memoria: determinadas porciones de memoria se asocian a un procesador brindando una mayor velocidad de acceso. Muchas de las estructuras relacionadas con la gestión de memoria deben replicarse en cada porción de memoria (nodo).

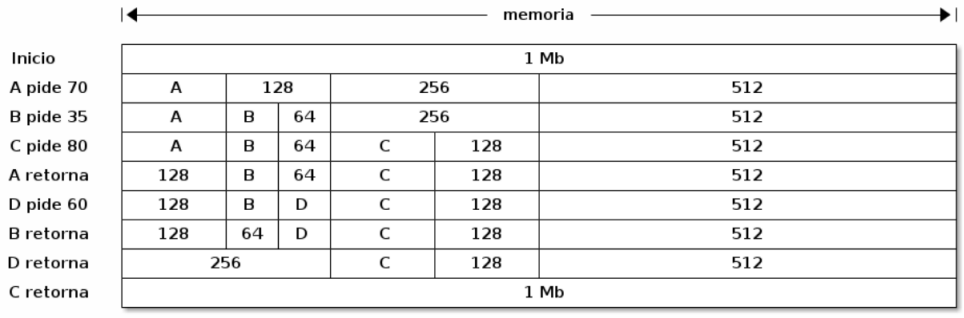
Paginación Para el i386 se utilizan páginas de 4kb y tablas de paginación de 4 niveles. Para el algoritmo de reemplazo de páginas se utiliza *reverse mapping*: es posible obtenes la/s página/s virtuales asociadas a un marco físico.

Pedido de ubicación atómica (*atomica allocation request*) Si se necesita un nuevo marco de memoria como parte del procesamiento de una interrupción o cuando se procesa código de una sección crítica del núcleo no se puede esperar la asignación de una página normal, el pedido de ubicación atómica resuelve instantáneamente la demanda o se retorna un error. Para implementar esto, se reserva un conjunto de páginas especial de antemano.

Memoria gestionada por el núcleo El segmento de memoria gestionada por el núcleo no se pagina por las siguientes razones:

- Los controladores DMA requieren memoria física contigua para poder leer o escribir masivamente. Si se pagina, puede ser necesario correr algoritmos de compactación o algún otro tipo de desfragmentación.
- La TLB se vacía cada vez que se cambia la tabla de paginación activa por lo que paginar la memoria del sistema o del núcleo penalizaría el rendimiento del sistema
- Utilizar un tamano de página más grande (4Mb) puede ser beneficioso para el núcleo pero no lo es para el resto del sistema.

Rastreo de memoria libre - Buddy system Linux emplea un sistema de colegas (*buddy system*) para hacer el seguimiento de la memoria libre.



Cuando se necesita asignar un bloque contiguo de memoria se busca el primer hueco en el que quepa el pedido (mejor ajuste) y cuando se libera una porción de memoria los bloques se funden para formar uno más grande con una técnica llamada coalescing. Este sistema se puede implementar como una lista de listas.

Reservas de memoria menores que una página El software encargado de gestionar pedidos más chicos de memoria que una página se llama *slab allocator*. Esta pieza de software permite reducir la fragmentación interna.

Asignación de memoria La memoria RAM se asigna a procesos y a caché. Si hay pocos procesos probablemente se utilice mayoritariamente para caché y si hay muchos procesos, la mayor parte se utilizará para guardar información de los procesos.

Reclamo de páginas Antes de que la memoria se llene, el algoritmo de reclamo de marcos del núcleo (PRFA *page frame reclaiming algorith*) le quita páginas a los procesos en modo usuario y los caché del núcleo. En líneas generales el algoritmo hace lo siguiente

- Libera las páginas que provocan menos daño (páginas de caché)
- Todas las páginas de un proceso (salvo las bloqueadas en memoria) son reclamables
- Antes de liberar una página compartida se ajustan las entradas de las tablas de paginación de los procesos involucrados
- Se utiliza una proximación al LRU usando el bit de acceso
- Se prioriza el reclamo de páginas limpias (sin modificaciones con respecto a las páginas en disco) lo cual ahorra operaciones de E/S

Existe un mecanismo similar para recuperar memoria de los buffers utilizados por el slab allocator.

En el caso que el escaseo de memoria sea crítico, el Out Of Memory Killer (OOM) elige un proceso para eliminar (y no enviar al swap). Por lo general se priorizan los procesos que no hayan hecho demasiado progreso y que no esté involucrado en secciones críticas, E/S o tareas del núcleo.

Área de intercambio - *Swap* Para evitar traer una página de swap que todavía está en memoria se utiliza una caché de swap en la RAM. Pero esto puede traer problemas de concurrencia

- multiple swap-in: una página compartida se carga más de una vez en memoria (y deja de ser compartida)
- swap-in y swap-out concurrentes: antes de que se efectivice una escritura en el swap, un proceso trae una versión no actualizada de la página en swap

4. Organización de archivos

Resumen de clase, no del libro.

Un archivo es simplemente un flujo de bytes. La información que guardan los archivos es persistente.

Datos asociados a un archivo

- Datos propiamente dichos
- Metadatos: información relacionada con la gestión y/o administración de los archivos. Por ejemplo: nombre, fecha de creación, tamaño, donde está guardado en el disco.

Operaciones en un sistema de archivios Alta, baja, modificación y consulta

Tipos de acceso al disco En los medios persistentes de acceso secuencial no cuestan lo mismo todas las lecturas. En las cintas magnéticas no cuesta lo mismo leer un dato del principio que del final y en un disco duro no es lo mismo leer datos del mismo cilindro en donde está parado el cabezal que leer datos de otros cilindros.

En los discos de estado sólido o medios extraíbles el acceso es aleatorio por lo que en principio todas los accesos cuestan lo mismo.

Cómo se graban los datos Los datos se suelen grabar de a bloques (512b, 1Kb, 4Kb) por lo que la información se estructurará en bloques antes de guardarse. Al querer guardar menos datos que un bloque se guardará el bloque completo posiblemente desperdiciando memoria.

Los bloques también se suelen llamar **sectores**.

Particiones Una partición es una subdivisión de un disco. No todas las arquitecturas de procesador soportan todo tipo de particiones. En cada partición se puede almacenar diferentes sistemas de archivos.

En los sistemas GNU/Linux la partición swap será la encargada de almacenar las páginas que se desalojan de memoria. En los sistemas derivados de MS-DOS, el espacio de intercambio está distruibuido por el almacenamiento secundario.

5. Sistemas de archivos

5.1. TAR

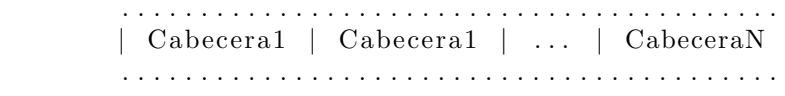
La cabecera de cada archivo contiene nombre, permisos, tamaño, fecha de creación/modificación, nombre del usuario que lo creó, grupo del usuario que lo creó, etc y ocupa un bloque de 512b. El '*' indica un relleno hasta el siguiente bloque.

Se suele utilizar para hacer backups o en combinación con compresores de archivos.

Ventajas
- Es simple: el primer byte de un archivo siempre está en el primer byte de un bloque y se puede buscar de cabecera en cabecera un archivo leyendo el tamaño de cada uno.

Desventajas
- Acceso secuencial: para buscar un archivo se deben leer todas las cabeceras hasta llegar al archivo solicitado. Además, si el medio físico que se utiliza es de acceso secuencial entonces la búsqueda necesariamente pasa por los datos de los archivos.
- Actualización compleja: si la modificación requiere más de los bloques inicialmente asignados para un archivo, se invalidan la cabecera y los datos de ese archivo y se copian al final de la lista de archivos, lo cual produce fragmentación interna.
- Los archivos están en un subbloque: en los sistemas modernos (bloques de más de 512b), los archivos comienzan en un múltiplo de 512 y no están alineados con los bloques.

5.2. Commodore 64



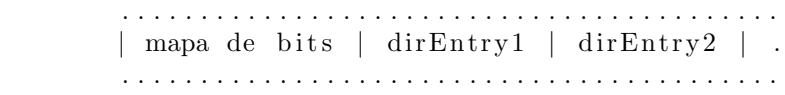
Cada cabecera incluye un campo que indica a donde se encuentra el primer bloque de un archivo.

Ventajas
- Simple: las operaciones sobre la metadada son más rápidas.
- La búsqueda o el listado de archivos por ejemplo.

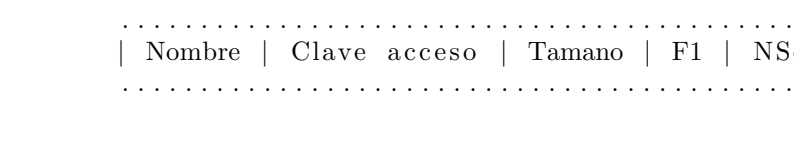
Desventajas
- La actualización tiene las mismas desventajas que el TAR
- Tamaño fijo del sector de cabeceras: se debe saber de antemano cuántos archivos se van a almacenar o estimar esa cantidad.
- Los archivos están en un subbloque (en sistemas modernos)

5.3. Questar/MFS

Introduce la **fragmentación** de archivos.



donde cada dirEntry es



El mapa de bits tiene un bit por cada bloque de 512b. Un 1 indica que dicho bloque está ocupado.

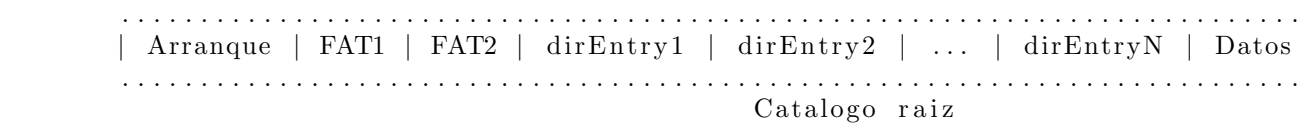
Ventajas
- Todos los metadatos están al comienzo del disco. Buscar o listar archivos y buscar sectores libres son ejemplos de operaciones eficientes

Desventajas
- La fragmentación es limitada: como las entradas de directorio tienen un tamaño fijo, la fragmentación está acotada.

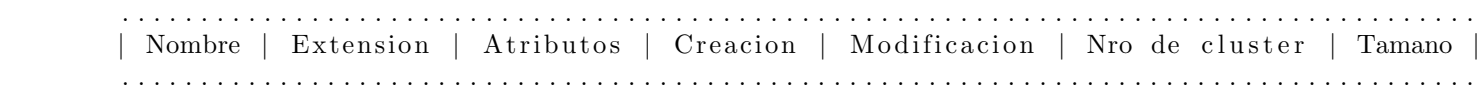
Un **directorio** (catálogo) es este sistema es un archivo con *directory entries* de fragmentos.

5.4. MSDOS FS / FAT

El disco se divide lógicamente en *clusters*, un cluster son 4 bloques. Una FAT (*file allocation table*) es un arreglo con tantas entradas como clusters hay en el disco.



donde cada dirEntry es



Una entrada en la FAT relleno con

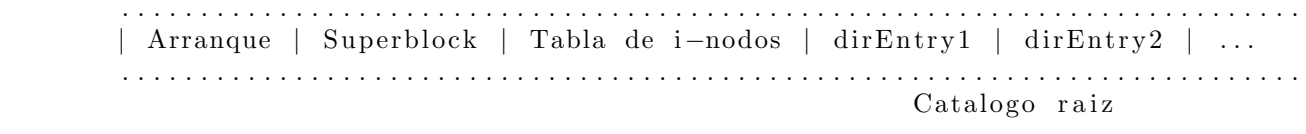
- un 0 indica que el cluster está vacío.
- un -1 indica que el cluster contiene el último fragmento de un archivo.
- un -2 indica que el cluster está dañado y se settea en el formateo del disco.
- cualquier número positivo indica el cluster en donde se encuentra el siguiente fragmento de un archivo.

donde los atributos pueden ser: lectura, oculto, sistema, etc y Nro de cluster es el número de cluster en el que inicia el archivo.

Ventajas - Permite la fragmentación arbitraria de archivos.

Desventajas - Bueno para FS chicos: la FAT entera ocupa mucho lugar. Como idealmente debería estar cargada en todo momento en RAM (leer la localización de cada segmento en la FAT cargada en disco es inaceptable), reduce significativamente el grado de multiprogramación.
- La política de actualización síncrona de la FAT ralentiza la escritura.

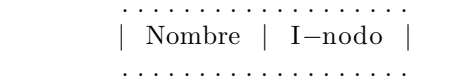
5.5. UNIX FS / UFS



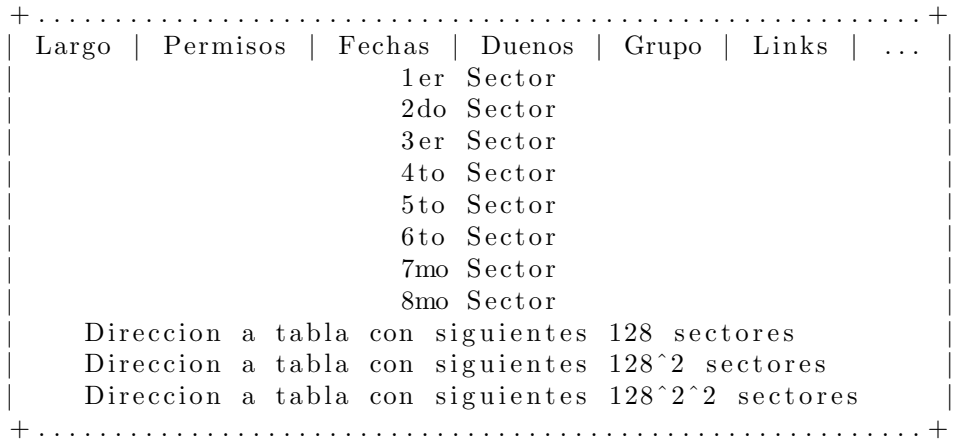
El superblock tiene

- Un mapa de bits que indica los sectores libres y ocupados
- Una lista enlazada con los i-nodos libres

Una dirEntry es



y un i-nodo es



Algunas consideraciones

- Permite tener más de un nombre para cada archivo de forma eficiente. 2 dirEntries pueden apuntar al mismo i-nodo con campo links = 2.
- Borrar un archivo significa decrementar en 1 el campo links. Si links llega a 0 se borra efectivamente el archivo (se borran todos sus bloques, su i-nodo y su dirEntry).

Ventajas	- Solo hay que cargar en memoria los i-nodos de los archivos que están abiertos - Si el archivo crece solo hay que modificar el i-nodo a diferencia de FAT en donde había que actualizar las dirEntries y la FAT - En el i-nodo están las direcciones de todos los bloques de un archivo.
Desventajas	- Cada nodo tiene espacio para una cantidad fija de direcciones de bloques

Enlaces Los sistemas Unix implementan directorios como grafos dirigidos por medio de dos mecanismos:

- Enlaces duros: nombres diferentes para un mismo i-nodo. Solo se pueden hacer links duros a archivos, no a directorios.
- Enlaces simbólico: relacion entre archivos mediante nombre (ruta)

6. Recuperación y fallas

A lo largo de la vida de un sistema de archivos, conforme los archivos se van asignando y liberando, cambian su tamaño, y conforme el sistema de archivos se monta y desmonta, pueden ir apareciendo inconsistencias en su estructura. En los sistemas tipo FAT, las principales inconsistencias son:

- Archivos cruzados: dos archivos incluyen al mismo cluster. La forma más simple de solucionar esto (pero no la que pierde menos información) es truncar uno de los archivos en el punto inmediato anterior al cruce.
- Cadenas perdidas o huérfanas: en la FAT figura que un cluster está ocupado pero ninguna entrada de directorio hace referencia al cluster. El espacio esta bloqueado y por lo tanto queda inutilizado.