Apuntes de ISO – Segundo Teórico

Memoria

La memoria principal es donde se ubican los programas y sus datos para ser ejecutados (a lo que su ejecución se denomina proceso).
Para lograr un buen manejo de ella se necesita una buena organización.
El SO debe ofrecer seguridad a los procesos de que no accedan a su sector privado y también permitir el acceso a zonas compartidas (procesos del SO).
El programador debe ser ajeno a como se organiza la mem principal.
El SO manejara los espacios de memoria de los procesos de forma aleatoria de manera que podrán re-ubicarse en caso de ser swapeados de memoria.
Cada proceso tiene su espacio de direccionamiento, por ende el SO debe asegurar que no se pueda acceder al espacio de otro proceso a menos que se conceda algún permiso.
El compartir memoria permite que no se repitan datos e instrucciones innecesariamente, de modo que se puede aprovechar mejor el espacio en RAM.
El espacio de direccionamiento es el rango de direcciones que puede direccionar un proceso (instrucciones y/o datos), este varía según la arquitectura (de 32 bits sería 2^{32}-1).
Un dirección lógica (o virtual) es aquella que simboliza una dirección del proceso durante su ejecución. En cambio una dirección física es la dirección real en la memoria, la dirección absoluta.
Para encontrar a que dirección física corresponde la dirección lógica se debe realizar una conversión.
Se puede utilizar dos registros que mantengan la dirección del comienzo del esp de direccionamiento del proceso en RAM (registro base) y el final (registro limite). Dichos valores se fijan cuando el proceso es cargado en RAM.
El mapeo en memoria de las direcciones lógicas a físicas son realizadas por el MMU (Memoria Management Unit).
El MMU es un elemento de hardware parte del procesador por lo cual re-programar a este es una operación privilegiada y debe realizarse en modo kernel.

Mecanismos de Asignación de Memoria

El primer esquema que se formó son las particiones fijas que divide la memoria en partes no necesariamente iguales y le asigna una “partición” a cada proceso.
Este espacio no cambiará por esto son fijas. Sin embargo si el proceso es demasiado grande se deberá modificar para que entre en una partición y/o repartirse.
Y si es muy chico se le cederá toda la partición provocando fragmentación interna.
El segundo esquema, una evolución del primero son las particiones dinámicas.
Están proponen particiones que varían de tamaño según lo que necesite el proceso (obviamente cada proceso tiene su partición).
El problema que presenta este esquema es que al ir cargándose los procesos de diferentes tamaños, los espacios también varían y se van dejando pequeños huecos sin

utilizar entre los procesos a medida que estos terminan, provocando así fragmentación externa.
NOTA: la fragmentación se produce cuando no se puede utilizar una locación de memoria por no encontrarse de forma contigua (ya que es tan chica que no entra todo y debe ubicarse el resto en otro lugar y esto no se puede, al menos no en este esquema).

Paginación

La paginación consta en dividir la memoria física en partes iguales denominadas "marcos". Los espacios de direccionamiento son divididos en espacios de igual tamaño que los marcos y se denominan paginas.
Cada proceso debe mantener una tabla de paginas que indica en que marco se ubica cada pagina, para conocer la dirección física de una dirección lógica se debe conocer el numero de pagina (así se busca el marco) y el desplazamiento dentro del mismo.
El tamaño de esta tabla tiene que ver directamente con el tamaño del espacio de direccionamiento del proceso.
En este esquema la fragmentación interna solo se produciría en la última pagina del proceso, sin embargo cuando el procesador se la pasa mas tiempo paginando entre las paginas de los distintos proceso que ejecutando a los mismo, entonces se genera hiperpaginación y esto baja la performance del sistema.
Las ventajas son su protección y la capacidad de compartir las paginas entre procesos.
La paginación es transparente, es decir, invisible para el programador.
Cada entrada de la tabla tiene bits que indican si la pagina fue modificada o si se encuentra en memoria.

Fallo de Paginas

El fallo de pagina se produce cuando la pagina que se quiere usar (o mejor dicho, una dirección que esta en ella) no se encuentra en la memoria principal o conjunto residente, es entonces cuando se debe buscar la pagina de la swap o disco.
Cuando sucede esto el SO pone al proceso en estado de espera hasta que se cargue la pagina que necesita. El que detecta cuando se produce un fallo de pagina es el hardware y es el que realiza un trap (una interrupción) al SO.
El SO busca un marco vacío y le encarga al E/S que traiga la pagina, mientras en el procesador se ejecuta otro proceso.
Luego de que E/S haya realizado lo que se le pidió, le avisa al SO y este, marca en la tabla de paginas, el bit de pagina en memoria.
El proceso que generó la falla de pagina ahora esta en estado ready (listo) y cuando se ejecuta empieza nuevamente con la instrucción que provocó el fallo.

Tablas de Paginas

Si bien el tamaño de la tabla tiene que ver con el tamaño del proceso, también tiene un limite del tamaño que puede adoptar, este no puede ser mayor de lo que el tamaño de direcciones permite.

Ademas las direcciones influyen en el tamaño de los marcos ya que si se tiene menos bits para el desplazamiento significa que no habrá que desplazarse tanto dentro de la pagina porque el marco es mas chico, pero por otro lado habrá mas cantidad de marcos.
La tablas de pagina de 1 nivel son aquellas con direcciones de 32 o 64 bits.
Las tablas de 2 niveles se dividen para que no ocupen demasiado espacio en RAM, es como una especie de paginación dentro de las tablas de paginas. Las direcciones tendrían una parte de bits para la pagina y otra para la tabla de pagina (?).

Tabla invertida -Hashing

Cuando el tamaño de las entradas y del espacio de direccionamiento aumenta tanto hace que la tabla de paginas se vuelva extremadamente gigante e imposible de ubicar en RAM por esto nace otra técnica.
En este caso hay una sola tabla para todo el sistema y por ende hay una entrada de tabla por cada marco de la memoria y el espacio de direccionamiento ahora es por la memoria no por cada proceso. Por esto para encontrar una pagina se utiliza una función hash para conocer el numero asociado (con mecanismo de tratamiento de colisiones el de encadenamiento).
La tabla invertida es organizada en la RAM como una tabla de hashing y solo mantiene las paginas que están cargadas en memoria, si no lo están se produce fallo de pagina.

Leer sobre tables de pagina y sus bits.

Tamaño de Página

Cuanto menor sea el tamaño menor fragmentación interna se va a producir pero por otro lado necesitará mas cantidad de paginas para almacenar el espacio de direccionamiento de un proceso por lo cual la tabla de paginas será mas grande.
Si el tamaño de la pagina es mayor entonces la fragmentación interna es mayor, pero como el sistema esta diseñado para traer bloques grandes de memoria, es mas rápido y eficiente el traslado de paginas a memoria principal.
Para traer una pagina de la memoria virtual es necesario un acceso para traer la tabla de paginas y otro para traer los datos, entonces para ello se usa una memoria cache de alta velocidad para almacenar la entradas mas recientemente usadas y que sea mas rápido encontrar las paginas en swap. Esta memoria se llama Translation Lookasider Buffer (TLB), cuando se pide una dirección entonces primero se busca la entrada en el TLB, si no esta entonces se busca la tabla de paginas del proceso y luego se fija si esta en memoria. Creado un page faul sino se encuentra allí y si se encuentra se actualiza la TLB.

Asignación de Marcos

Existen dos tipos de asignación, la dinámica, en la cual varia la cantidad de marcos asignados a un proceso y la fija donde la cantidad de marcos para cada proceso es fijo y no cambia durante la ejecución. Esta ultima puede ser equitativa, es decir, una distribución de marcos equitativos entre todos los procesos en ejecución o proporcional que es acorde al tamaño del proceso.

Reemplazo de Paginas

Cuando se produce un fallo de pagina se busca una pagina victima a la cual reemplazar, esto en caso de no haber ningún marco vació.
Lo optimo seria que esta victima sea una pagina que no se vuelva a referenciar en un futuro cercano.
Existe un alcance del reemplazo, cuando se puede reemplazar una pagina de otro proceso se dice que es de alcance global, esto por supuesto no es compatible con la asignación de marcos fija.
El alcance global permite que un proceso aumente la cantidad de marcos asignados al mismo y ademas puede tomar marcos de procesos de menor prioridad que él.
Por otro lado existe el reemplazo local, el cual solo puede reemplazar paginas del mismo proceso, es decir, si el marco pertenece a otro proceso entonces no podrá tomarlo. Esto puede ocasionar que hayan marcos ocupados por paginas que no se usen y no puedan ser ocupados por otros procesos que la necesiten.

Algoritmos de Reemplazo

Optimo-FIFO-LRU-SecondChance-NRU(No usado últimamente)

Hiperpaginación

Cuando la CPU se pasa mas tiempo paginando que ejecutando se produce lo que se llama hiperpaginación o thrashing. Esto provoca una baja de la performance del sistema.
Cuando la CPU no tiene mucho uso entonces se aumenta el grado de la multiprogramación (muchos programas funcionando a la vez). Si se observa que un proceso necesita mas paginas y es de reemplazo global, entonces le roba marcos a otros procesos, si hay demasiada paginación el bajar el grado de multiprogramación podría ayudar. Sin embargo a veces sucede que hay bajo uso de CPU no por que haya poca programación sino porque hay mucha demanda de E/S y los procesos entran en estado de espera.
El encargado de aumentar el grado de la multiprogramación es el scheduler long term.
Un modo de controlar el thrashing es utilizar el reemplazo local para que no se roben marcos entre si, por esto disminuye la paginación así como performance del sistema pero puede controlarse.
En conclusión si el proceso tiene las paginas que necesita no habrá thrashing, pero una forma de solucionarlo puede ser usando la estrategia del working set basado en el modelo de locación o el algoritmo de PFF (frecuencia de fallos por pagina).

Estrategia de Working Set

El modelo de locación se basa en ubicar los datos de un programa en direcciones cercanas, de modo que sea mas rápido buscar dichos bloques de memoria, entonces la estrategia de Working Set propone que se tengan las paginas mas usadas recientemente en memoria, de modo que no se deba buscar una y otra vez, disminuyendo la paginación y por ende la hiperpaginación.

Una ventana de working Set tiene las referencias a estas paginas usadas mas recientemente.
Cuanto mas grande sea la ventana de working set, mas referencia se obtendrán y se sabrá que paginas son las mas usadas.
Menor tamaño menos datos para procesar.
Leer sobre working set de los libros (explicado aquí muy brevemente).

Prevención de thrashing por PFF

En este algoritmo se calcula según la taza de fallos de pagina para saber que conjunto residente es mejor que se mantenga en el WS. Si la PFF(frecuencia de fallos de pagina) es muy alta entonces los procesos necesitan mas marcos, si es muy baja tiene muchos marcos asignados. Entonces según esto, se verifica quien gana mas marcos y quien no.
Hay que tener en cuenta que se debe considerar un limite de PFF, en el cual si se excede dicho limite se da marcos, también si no hay mas frames se puede llegar a suspender un proceso para darle marcos a los que tienen alto PFF.

Demonio de Paginación

El demonio de la paginación es un proceso creado por el SO al prender el sistema, este de encarga de múltiples tareas, tales como limpiar paginas modificadas, o retrasar la liberación de paginas (sacarla de memoria luego de que termine el proceso) hasta que sea realmente necesario.

Memoria Compartida

Para poder compartir una pagina, esta debe estar en la tabla de paginas de cada proceso que necesite usarla, puede pasar que el numero de la pagina asociado al marco sea diferente para cada proceso.
Leer sobre memoria compartida en los libros.

Mapeo de Archivos en Memoria

Esta técnica permite asociar el contenido de un archivo a su espacio de direcciones, este es buscado unicamente cuando se produce page fault y se trae directamente del disco no de la swap.
Cuando se termina el proceso se guardan las paginas modificadas en el archivo correspondiente, de forma que se realiza E/S sin usar operaciones o tocar el sistema de archivos. Comúnmente se usa para asociar librerías o DLLs.

Copia en Escritura

La copia en escritura permite compartir las mismas paginas para procesos padre e hijo, y solo en caso de modificación de la pagina se produce un duplicado de la misma.

Área De Intercambio

Por lo general, el área de intercambio esta separado del sistema de archivos, en otra partición.
Cada vez que se crea un proceso se reserva una porción del área de intercambio igual al tamaño de su imagen. Y se le asigna la dirección en el disco de su área. Inicialmente no se asigna ninguna pagina hasta que se realice el intercambio, es decir, el swappeo de la pagina.
Leer área de intercambio de los libros.

Segmentación

En este caso, es el programa el que se divide en partes llamados segmentos, estos segmentos corresponden a una unidad lógica del programa como por ejemplo, programa principal,procedimientos, pilas , variables locales y globales ,etc.

Estos segmentos por lo general no son de igual tamaño, y al igual que en la paginación para conocer su dirección física se necesita del segmento y el desplazamiento en él.
Se mantiene una tabla de segmentos que mantiene la dirección base del mismo y su longitud.
Esta segmentación es muy parecida a las particiones dinámicas. con la diferencia de que los segmentos no necesitan estar almacenados en lugares contiguos bajando así la fragmentación externa (no eliminándola).
En este esquema el programador es capaz de visualizar los segmentos y donde se almacenan.

Segmentación Paginada

En la segmentación paginada se divide los segmentos en paginas. Para traducir una dirección lógica se necesita el segmento, la pagina donde esta dicho segmento y luego el marco y desplazamiento.
Leer mas específicamente la traducción de dir lógicas.

Overlays/Sobrepuestos: el administrador de los sobrepuestos se encarga de intercambiar sobrepuestos del programa (partes del programa que fueron divididas en el disco), si hay espacio lo ubica allí y sino pisa al sobrepuesto que haya intercambiando así entre ellos.
Si bien el trabajo real del intercambio lo hacia el SO, era el programador el que se encargaba de dividir el programa en partes.
Este trabajo era tedioso por eso con el tiempo se ideo una forma de que lo realizará el SO, ahora se dividía el espacio de direcciones en partes, en paginas.
En este esquema no era necesario que estén todas las paginas cargadas en memoria física, por lo cual si estaba la pagina entonces el hardware lo asociaba, si no estaba el sistema mandaba una alerta para buscar la pagina faltante, aquí es donde nace el concepto de “memoria virtual” o también llamado, la swap.

Se llama conjunto residente o working set a la porción del espacio de direccionamiento que se encuentra cargado en memoria.
El hardware es unos de los que ayuda a darse cuenta de que paginas faltan. La memoria virtual ayuda a poder tener programas mas grandes que la RAM.

Entrada/Salida

Se busca que los dispositivos de entrada/salida sean silenciosos al punto de que todo lo que conozcan los procesos sean las operaciones como read, write, entre otras.
Las dispositivos de E/S son mucho mas lentos que los procesadores y memoria principal, o cache. Por esto se debe organizar de manera que se pueda usar la cpu mientras se busca un dato de disco y evitar que la computadora este ociosa por estos datos, esto se puede realizar mediante la multiprogramación.
Otra cuestión que entra en juego es el sistema de ficheros.

Unidad de transferencia:

Hay dos tipos de dispositivo, los de bloque y los de carácter.
Los dispositivos de bloque son aquellos que almacenan la información de a bloques usualmente direccionables y permiten la búsqueda y recuperación de datos de los mismo recorriendo el dispositivo (es direccionable). Los tamaños de los bloques suelen ir de 512 bytes hasta 32768 bytes.
Los dispositivos de carácter en cambio son aquellos que transfieren información constantemente o la reciben pero no permiten la búsqueda de ningún tipo de dato ni es direccionable.
Este tipo de dispositivo podría ser un mouse o una impresora.
Formas de Acceso: Secuencial o aleatoria (?)
Controlador de Dispositivos: son aquellos encargados de controlar la transferencia de datos entre la cpu y el dispositivo, consisten en una especie de chip. Estos traducen los flujos de bits de los dispositivos a bloques que pueda interpretar la cpu, y ademas corrige errores que puedan presentarse en el envió del flujo de bits.
E/S Asignada por memoria:
Los controladores mantienen un par de registros que sirven como información del estado del dispositivo para el SO y para recibir ordenes del mismo según el valor que se le setee en sus bits.
Una forma que se ideó para esto es asignarles un puerto el cual esta protegido del acceso de programas en el cual se intercomunicará con el SO.
Otra forma es asignarle una dirección única en la memoria (la cual no está asignada como memoria utilizable) a todos los registros de los controladores, esto se conoce como E/S mapeado en memoria. Este método no necesita ser protegido de los procesos ya que simplemente se quita esa porción del espacio direccionable de los procesos. Por lo general, estos registros suelen tomar la parte superior del espacio de direcciones.

Acceso Directo a Memoria (DMA)
El DMA es un dispositivo que se encarga de la transferencia de bits entre la memoria principal y los dispositivos de E/S, es independiente del procesador, por esto se puede transferir información mientras se está ejecutando un proceso.
El DMA requiere de un controlador de DMA para funcionar, el cual suele venir incluido en el controlador de dispositivos. El DMA tiene acceso directo con el bus del sistema.
Formas de Realizar E/S
-Bloqueantes: el proceso que realiza la e/s debe esperar que se termine la operación de e/s para seguir ejecutándose.
-No Bloqueantes: el proceso se sigue ejecutando a medida que le llegan los datos, por ejemplo, el reproductor de música, reproduce a medida que va tomando datos del almacenamiento. Entonces el requerimiento retorna lo antes posible para que el proceso lo utilice.

Drivers

- Interfaz entre el SO y el HW

- Forman parte del espacio de memoria del Kernel
- En general se cargan como módulos

- Los fabricantes de HW implementan el driver en función de una API especificada por el SO

- open(), close(), read(), write(), etc

- Para agregar nuevo HW sólo basta indicar el driver correspondiente sin necesidad de cambios en el Kernel

Performance
I/O es uno de los factores que mas afectan a la performance del sistema:

- Utiliza mucho la CPU para ejecutar los drivers y el código del subsistema de I/O.
- Provoca Context switches ante las interrupciones y bloqueos de los procesos.
- Utiliza el bus de mem. en copia de datos:
  -Aplicaciones (espacio usuario) – Kernel
  -Kernel (memoria física) – Controladora

Mejorar la performance

- Reducir el número de context switches
- Reducir la cantidad de copias de los datos mientras se pasan del dispositivo a la aplicación
- Reducir la frecuencia de las interrupciones, utilizando:
  – Transferencias de gran cantidad de datos

– Controladoras mas inteligentes
– Polling, si se minimiza la espera activa.

- Utilizar DMA

Sistema de Archivos-File System

Los archivos sirven para guardar grandes cantidades de información que será utilizada en un futuro, es decir, se espera la permanencia de estos datos.
Los archivos también convienen para permitir compartir datos entre distintos procesos.
Un archivo es una entidad con nombre la cual provee a los programas entrada de datos y una salida donde guardar la información. El programa mismo debe ser almacenado para poder ejecutarlo en un futuro o en el momento.
Los archivos tienen un par de operaciones comunes para su uso como crear, borrar, leer, escribir,buscar, etc.
El sistema de archivos es un software que facilita el manejo de archivos para las aplicaciones, a su vez abstrae al programador de operaciones de bajo nivel.
Los objetivos mas primordiales del SO para los sistemas de archivos es proporcionar una buena gestión de datos y asegurar su integridad o minimizar la destrucción o perdida de datos. Ademas se quiere brindar un conjunto de interfaces para la E/S para el tratamiento de archivos y cumplir también con las peticiones del usuario.

Existen diferentes tipos de archivos, los de texto plano o archivos secuenciales o los archivos binarios como ejecutables o archivos de objetos.
Los archivos tienen un par de atributos para destacarles y monitorearlos.

°Nombre
- °Identificador
- °Tipo
- °Localización
- °Tamaño
- °Protección, Seguridad y Monitoreo:
  - -Owner, Permisos, Password
  - -Momento en que el usuario lo modifico, creo, accedió por ultima vez
    -ACLs

Directorios
Los directorios son utilizados para mantener la estructura del sistema de archivos e información sobre lo que se guarda dentro de ellos, archivos o mas directorios.
También son una especie de archivo.
Los directorios también forman parte del nombre completo de un archivo.
Operaciones en directorios:
- °Buscar un archivo

 °Crear un archivo (entrada de directorio)
 °Borrar un archivo
 °Listar el contenido
 °Renombrar archivos
 °Etc.

El tener directorios ayuda la eficiencia de locación de archivos, ya que al estar organizado se vuelve mucho mas fácil su búsqueda. Esto mismo también se aplica a los agrupamientos lógicos por propiedades o funcionalidades, es decir, aplicaciones, librerías, juegos, etc.

Un archivo puede ubicarse siguiendo un path(camino) desde la raíz hasta la carpeta donde termina. Es decir, siempre tendrán como parte de su path a la raíz, y luego seguidos los directorios en donde se encuentran contenidos.

Por ende dos archivos pueden llamarse de la misma manera mientras no estén en el mismo directorio, por esto se puede decir que el full pathname (o Path absoluto)de un archivo es único.

Cuando hablamos de path absoluto nos hacemos referencia al conjunto de nombres de directorios mas el del archivo que nos muestra la ruta hacia el mismo. En un momento dado abrimos una carpeta y se dice que ese es el directorio de trabajo, es decir, el directorio en el cual estoy parado actualmente.

Si queremos entrar a un subdirectorio de nuestro directorio de trabajo, podemos hacerlo mediante su path absoluto o mediante el path relativo, es decir, la ruta desde el directorio donde estoy hasta el archivo.

En un ambiente de multiusuarios es necesario poder compartir archivos, como los del sistema por ejemplo. Por esto debe ser manejado con precaución bajo un esquema de protección y los accesos deben controlarse.

El propietario debe poder manejar quien podrá hacer cosas con un archivo dado y que cosas podrá hacer con el mismo.

Accesos y Permisos

Los directorios tienen también un sistema de permisos los cuales forman parte de la información que mantienen los mismos.

°Reading: permite leer archivos dentro de el directorio.
°Execution:permite ejecutar el archivo.
°Appending:permite agregar datos pero no modificar ni borrar.
°Updating: permite agregar, modificar y borrar archivos del directorio.
°Changing Protection: permite cambiar los permisos de acceso.
°Deletion: permite borrar el archivo.

Owner/Propietario: el propietario del archivo tiene todos los permisos sobre el mismo, y puede dar o sacar permisos a otros usuarios.

-Usuarios, Grupos de usuarios o todos

Los permisos que pueden darse son de lectura, escritura y ejecución.
Metas del sistema de Archivos
Algunas de las metas del sistema de archivos es brindar espacio en disco a los archivos del usuario o sistema. Ademas, mantiene información sobre la cantidad de espacio libre en el disco.
Algunas definiciones:

- °Sector: Unidad de almacenamiento utilizada en los discos rígidos
- °Bloque/Cluster: Conjuntos de sectores consecutivos
- °File System: Define la forma en que los datos son almacenados
- °FAT: File Allocation Table: Contiene información sobre en que lugar están alocados los distintos archivos.

Existe una etapa de pre-asignación donde se deduce cuando espacio se va a necesitar para el archivo, este suele ser definido mucho mas grande de que se necesita o en ocasiones no alcanza. Está la posibilidad de almacenar el archivos en sectores contiguos.

Formas de Asignación
La asignación se puede realizar de forma dinámica, en ese caso se asigna la memoria cuando el archivo lo requiera, el problema con esto es que los bloques no pueden quedar de forma contigua.
La forma continua pretende que todos lo bloques estén juntos o “contiguos”. Para lograr esto debe saberse el tamaño del archivo con antelación, es decir, tiene que hacerse una pre-asignacion. La FAT pasa a ser muy simple, una sola entrada que incluye principio y longitud del archivo, por esto puede leerse con una sola operación.
Los problemas que se encuentran con esta forma son:

- °Puede generarse fragmentación externa la cual se soluciona con compactación.
- °Es difícil encontrar lugares contiguos que estén vacíos.
- °Es difícil cuando el archivo incrementa de tamaño.

Otra es la forma encadenada la cual esta basada en bloques individuales, estos se enlazan a su siguiente bloque. Esta forma no permite fragmentación externa y puede crecer el archivo según lo necesite.
Evidentemente no es una buena forma si se busca acceso random, mas si para acceso secuencial.
La FAT mantiene el bloque de inicio y el tamaño del archivo.
Mientras que la forma indexada tiene también la propiedad de bloques individuales y no genera fragmentación externa. Sin embargo el acceso random es eficiente.
La FAT tiene la dirección al bloque indice.
Existen dos variantes de la forma indexada:

- °Asignación por secciones: un bloque que tiene indice a otros bloques del archivo.
- °Niveles de indireccion: organizados por I-NODOS que tienen referencias a otros bloques que a su vez tienen referencia a los bloques de datos (dependiendo la cantidad de datos habrá mas o menos bloques de referencias).

Analizar como se calcula el tamaño de un archivo.

Gestión de espacio libre

Se debe mantener un control sobre cuales bloques están libres en el disco, las alternativas para lograr esto son:

- °Tablas de bit: es una tabla de tantos bits como bloques haya, si esta en 0 el bloque esta libre, en caso contrario esta ocupado. Es fácil encontrar los bloques libre, pero depende mucho el tamaño de la tabla de los bloques que haya en sistema.
- °Bloques libres encadenados: son bloques libres encadenados por medio de punteros. Esta alternativa tiene varios inconvenientes como ser difícil conseguir los bloques (ineficiente), mas si deben ser consecutivos. Es complejo si se llega a perder un enlace.
- °Indexación: es una variante de los bloques encadenados. A diferencia del último, el primer bloque libre tiene n-1 direcciones a bloques libres y la n-ésima dirección apunta a otro bloque con mas direcciones a bloques libres.
- °Recuento: variación de la indexada, este mantiene la dirección del primer bloque libre y los bloques contiguos. Esta estrategia contempla una asignación continua, por lo cual se liberan varios bloques a la vez, y se piden bloques contiguos.

Tipos de archivos para UNIX

- °Archivo común
- °Directorio
- °Archivos especiales (dispositivos /dev/sda)
- °Named pipes (comunicación entre procesos)
- °Links (comparten el i-nodo, solo dentro del mismo file system)
- °Links simbólicos (para filesystems diferentes)

Estructura del volumen

- °Boot Block: Código para bootear el S.O.
- °Superblock: Atributos sobre el File System
  - -Bloques/Clusters libres
- °I-NODE Table: Tabla que contiene todos los I-NODOS
  - -I-NODO: Estructura de control que contiene la información clave de un archivo
- °Data Blocks: Bloques de datos de los archivos

WINDOWS

File System soportados

- °CD-ROM File System (CDFS) CD
- °Universal Disk Format (UDF) DVD, Blu-Ray
- °File Allocation Table
  - – FAT12 MS-DOS v3.3 a 4.0 (año 1980), floppy
  - – FAT16 MS-DOS 6.22, nombres cortos de archivo

– FAT32 □ MS-DOS 7.10, nombres largos pero no soportados en MS-DOS
□ °New Technology File System (NTFS)

FAT de Windows
La tabla de locación de archivos es originalmente usada por DOS y Windows 9x, Windows aun soporta este file system por su compatibilidad con otros SO en sistemas multiboot, por permitir upgrades de versiones anteriores.
Las diferentes versiones de la FAT se diferencian con un numero que indica la cantidad de bits que se necesitan para referenciar un bloque o un clúster: FAT12,FAT16, FAT32
La FAT tiene tantas entradas como bloques haya ademas de mantener un duplicado de la misma. La FAT, su duplicado y el directorio raíz se almacenan en los primeros sectores de la partición.
Se utiliza un sistema de asignación encadenada, con la diferencia de que los punteros se guardan en la FAT. Aquellos bloques libres o dañados tienen un código especial.
La FAT12 mantiene 2^16 sectores, Windows utiliza este file system para los diskets que pueden almacenar hasta 1,44MB.
La FAT16 mantiene 2^16 sectores y el tamaño de sector dependía del tamaño del volumen al formatearlo.
La FAT32 tiene 32 bits para direccionamiento, pero reserva los primeros 4 por lo cual quedan 28 bits. El modo de acceso e identificación de sectores lo hace mas eficiente que el FAT16.
La FAT32 es la mas reciente de la linea y posteriormente salio el exFAT el cual es conocido como FAT64.
Ver tamaño de sectores en el pdf 3 de filesystems.

NTFS
Este es el file system nativo desde Windows NT, utiliza 64 bits para referenciar sectores.
SE utiliza NTFS en vez de FAT ya que, por mas que FAT sea mas simple y rápido en algunas operaciones, NTFS soporta tamaños de archivo y de discos mayores, mejora performance en discos grandes, los nombres de archivos pueden ser de hasta 255 caracteres, tiene atributos de seguridad y es transaccional.

Buffer
Los buffers son porciones de memoria donde se almacenan bloques de memoria temporalmente, ya sea hasta que la RAM tome esos datos que vienen del disco o que dichos datos estén esperando para que el procesador de E/S los envíe.
Cuando un proceso quiere traer un bloque de memoria de la cache existen dos alternativas,o trae ese bloque a su espacio de direcciones, o bien la ubica en un espacio de memoria compartida sobre la cual pueden acceder varios procesos. La ultima estrategia necesita de un algoritmo de reemplazo ya que evidentemente al estar compartido por varios procesos podría llenarse fácilmente.

Estructura del buffer

°El header que mantiene toda la información respecto del bloque.

-Identificadores para el n.º de bloque y n.º de dispositivo.

-Estado

-Punteros a

→hash queue (2)

→free list (2)

→a bloque de memoria (1)

Estados del buffer:

°Free: significa que el buffer esta disponible para usarse (esta en la free list).

°Busy: significa que el buffer esta siendo utilizado por algún proceso.

°Se está escribiendo o leyendo de disco.

°Delayed Write: significa que el buffer fue modificado pero no se almaceno en disco.

Free List:

La free list es una lista que mantiene punteros a los bloques libres, estos no están necesariamente vacíos sino que fueron dejados de usar por los procesos y pueden tener estado Delayed Write. Su orden es según la estrategia LRU.

El header mantiene un puntero hacia el bloque libre previo en la lista y otro al siguiente bloque en la lista.

Hash queue:

Las listas de hash queue son aquellas que mantienen buffers organizados mediante funciones hash. Es decir, para poder buscar mas rápido un buffer en particular, se le aplica una función hash que usa como parámetros el n.º de buffer y n.º de dispositivo.

Entonces, los buffers que dieron el mismo resultado son agrupados en una hash queue particular (dependiendo el resultado le toca una hash queue diferente).

El header mantiene un puntero hacia el buffer previo en la lista de hash queue y otro hacia el siguiente en la lista.

Aclaración: hay que tener en claro que cuando hablamos de un header, hablamos de un header particular que pertenece a un buffer particular, por esta razón, cada buffer es individual y puede darse la ocasión de no formar parte de la free list y tener estos punteros vacíos. Pero es garantía que cualquiera de los buffer pertenece a una hash queue. Cuando el proceso que referenciaba al buffer termina, este último pasa a formar parte de ambas listas.

°El cuerpo del buffer donde mantiene el dato, podría decirse que es el buffer en si.

Estrategias de reemplazo

Cuando se necesita reemplazar un bloque en la cache suele utilizarse la estrategia LRU, es decir, el que fue referenciado hace mas tiempo. Para esto se mantiene una lista en la

cual el mas recientemente referenciado queda a lo ultimo de la misma. Esta lista solo mantiene punteros hacia los bloques, por lo tanto los bloques no cambian de lugar físicamente.
Otra estrategia es LFU es donde se mantiene cual es el que fue referenciado menos cantidad de veces y reemplaza este bloque.

Buffer caché
El objetivo de la cache es que haya menos accesos al disco, para minimizar el tiempo de búsqueda de un dato. Esta es una estructura formada por buffers, durante la inicializacion el kernel le asigna un espacio de memoria.

Funcionamiento
Cuando el proceso quiere un archivo del disco, busca en su i-nodo donde esta el dato, luego hace el pedido a la caché. La cache se fija si el dato esta en la misma o si debe hacerse E/S al disco.
Se pueden dar 5 escenarios:
-El bloque esta en la hash queue y su estado es free.
-El bloque esta en la hash queue y su estado es busy.
-El bloque no está en la hash queue y hay bloques libres en la free list(se usa uno de ellos).
-El bloque no está en la hash queue y el buffer libre tiene estado DW.
-El bloque no está en la hash queue y no hay buffers libres en la free list (la free lista esta vacía).

¿Que pasa en cada escenario?
-El bloque esta en la hash queue y su estado es free.
En este caso el kernel remueve el bloque de la free list, su estado pasa a ser busy y el buffer pasa a ser utilizado por el proceso. Se re-ubican los punteros de la free list.

-El bloque esta en la hash queue y su estado es busy.
El proceso pasa a estado de espera hasta que se deje de usar el buffer, cuando el proceso que lo utilizaba lo libera, se despiertan todos los procesos que necesitaban dicho buffer. El proceso busca nuevamente en la hash queue y la free list.

-El bloque no está en la hash queue y hay bloques libres en la free list.
El kernel toma el primer buffer de la lista de free list y lo saca de la misma. Pasa a utilizar dicho buffer. Le aplica la función hash y la ubica en la hash queue correspondiente (mediante punteros no físicamente).

-El bloque no está en la hash queue y el buffer libre tiene estado DW.
Cuando el kernel encuentra buffers en DW los manda a escribir y toma el siguiente buffer que este DW y lo asigna al proceso.

Cuando terminan de escribirse pasan a estar a lo ultimo de la free list.
-El bloque no está en la hash queue y no hay buffers libres en la free list
Cuando no hay buffers libres, el proceso debe esperar a que se libere uno (queda en espera), cuando el proceso despierta se debe verificar nuevamente si el bloque no está en la hash queue, ya que puede haber sido pedido por otro proceso.