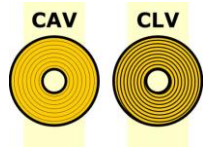
Entrada / Salida y planificación de disco

Eficiencia

- La E/S suele ser el cuello de botella
  - CPU ejecuta operaciones a 1 GHz aprox.
  - RAM: acceso de nanosegundos.
  - Dispositivos de E/S: acceso de milisegundos o más.
- Objetivo de la gestión de E/S: rendimiento

Dispositivos de E/S. Tipos.

- Según la unidad de transferencia:
  - Dispositivos **de carácter**:
    - Acceso secuencial a nivel de carácter:
      - Procesamiento canónico (*cooked*): caracteres filtrados
      - Procesamiento no canónico (*raw*): caracteres sin filtrar. Se reciben todas las teclas que se pulsan (intro, retroceso, …)
    - Ejemplo: terminales, impresoras, teclados, interfaces de red
  - Dispositivos **de bloque**:
    - Acceso secuencial o aleatorio a nivel de bloque
    - Ejemplos:
      - » Discos duros:
        - VAC (velocidad angular cte).
        - La superficie de los sectores internos es menor y determinan la densidad de grabación
      - » CDs, DVDs:
        - VLC (velocidad lineal cte).
        - La longitud de los sectores es la misma
        - Mayor capacidad, mejor aprovechamiento de la superficie.
        - Desventaja: para acceder al centro hay que girar más rápido



Dispositivos de E/S. Diferencias

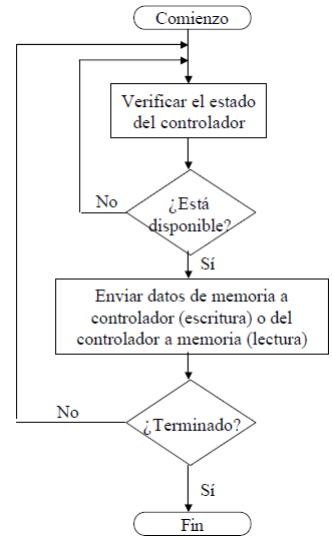
- Velocidad de transferencia de datos
  - Ej: teclado: 10 bps, pantalla gráfica: >10^8 bps
- Aplicación
  - Uso al que está destinado el dispositivo
- Complejidad de control
  - Ej sencillo en impresora, complejo en disco
- Unidad de transferencia
  - Caracteres o bloques
- Representación de datos
  - Diferentes esquemas de codificación
- Condiciones de error
  - Difieren en su naturaleza, el modo en que se notifican, sus consecuencias y el rango disponible de respuestas

Organización del sistema de E/S

- Según la interacción computadora-controlador
  - E/S programada
  - E/S dirigida por interrupciones
  - Acceso directo a memoria

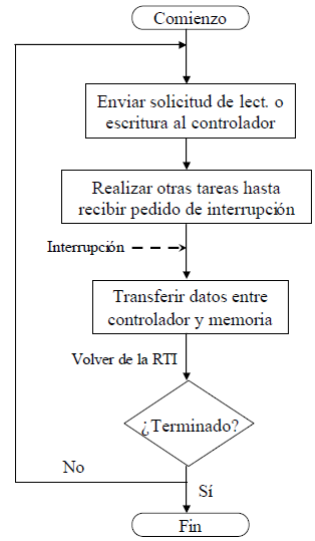
Organización de E/S. E/S programada (*polling*)

- El procesador envía un mandato de E/S a petición de un proceso a un módulo de E/S
- El proceso realiza una espera activa hasta que se complete la operación antes de continuar.
- El módulo de E/S no interrumpe al procesador sino que éste realiza una consulta periódica para detectar si el dispositivo está listo
- Desventaja: Consume CPU para dispositivos poco usados.
- Aplicación:
  - En desuso excepto para dispositivos que generan muchas interrupciones como las tarjetas ethernet



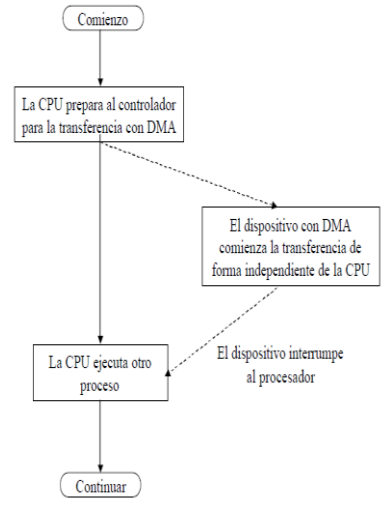
Organización de E/S. E/S dirigida por interrupciones

- El procesador emite un mandato de E/S a petición de un proceso
- Continúa ejecutando las instrucciones siguientes
- Es interrumpido por el módulo de E/S cuando éste ha completado su trabajo
- El gestor de interrupciones recibe y maneja la interrupción:
  - Escoge la rutina a ejecutar
  - La rutina no tiene acceso a las tablas del espacio de direcciones. Se emplea un buffer (memoria del núcleo) y luego ya se copia a la RAM (doble copia).
- Cada señal de interrupción tienen una determinada **prioridad**.
- Las interrupciones sólo son posibles en S.O. apropiativos



Organización de E/S. Acceso directo a memoria (DMA)

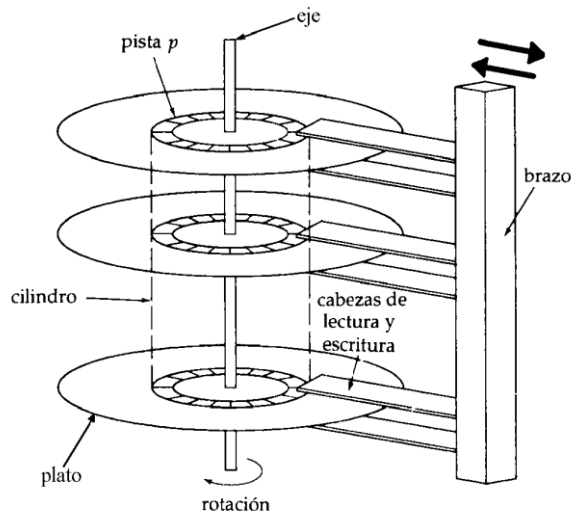
- Un módulo de DMA controla el intercambio de datos entre la memoria principal y un módulo de E/S
- El procesador manda una petición de transferencia de un bloque de datos al módulo de DMA y resulta interrumpido sólo cuando se haya transferido el bloque completo
- Evita el uso de la CPU transfiriendo los datos directamente entre los dispositivos de E/S y la memoria
  - del usuario
  - del núcleo
- Básico para aprovechar la CPU en un sistema multiprogramado, ya que libera tiempo de la CPU que se puede usar para ejecutar otros programas



Organización de E/S. Acceso directo a memoria (DMA)

1. Programación del controlador para la transferencia:
   - Operación L/E, cantidad de datos y dirección de memoria.
2. El controlador contesta aceptando la petición de E/S.
3. El controlador le ordena al dispositivo la operación (ej. lectura).
   - El dispositivo deja los datos en su propia memoria interna.
4. Según los datos están listos, el controlador los copia a la dirección de memoria que se indicó, incrementa dicha dirección y decrementa la cantidad de datos pendientes de transferir.
5. Los pasos 3 y 4 se repiten hasta que cantidad de datos es cero.
6. El controlador interrumpe a la UCP para indicar que la operación de DMA ha terminado.

Discos

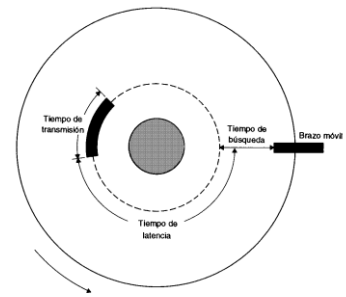


Discos. Estructura física

- Varias superficies circulares y planas recubiertas por material magnético en ambas caras.
- La información se graba en el material magnético.
- Hay un motor que lo hace girar a gran velocidad.
- Hay, normalmente, una cabeza de lectura-escritura por superficie.
- Cada superficie dividida en anillos concéntricos: pistas.
- Cilindro: conjunto de pistas situadas en la misma posición.
- Dentro de una pista la información se escribe en bloques de tamaño fijo (generalmente 512 bytes): sectores
- Capacidad = cilindros * pistas * sectores * tamaño sector

Discos. Lectura/escritura

- El tiempo de leer o escribir un sector del disco está determinado por cuatro factores:
  1. Mover el brazo hasta el cilindro: tiempo de búsqueda (T_s: *seek time*). Se mide en ms
  2. Activar la cabeza adecuada, que supone un tiempo despreciable
  3. Esperar a que el sector pase ante la cabeza: tiempo de latencia (retardo rotacional)
     - $T_r = 1/(2r)$, siendo r la velocidad de rotación del disco (revoluciones por segundo). Es un tiempo medio
  4. Leer o escribir el sector: tiempo de transferencia
     - $T_t = b/(rN)$, siendo
       b: bytes a transferir
       N: nº de bytes por pista
       r: velocidad de rotación (revoluciones por segundo)



Discos. Lectura/escritura.

- Tiempo medio de acceso total

$$T_a = T_s + 1/(2r) + b/(rN)$$

- El más costoso es el tiempo de búsqueda.
  - El objetivo es, por tanto, reducir el tiempo medio de búsqueda: planificación del disco
  - Esto tiene que ser logrado por el manejador del dispositivo

Discos. Lectura/escritura.

- Ej: calcular el tiempo total de acceso a los datos en caso secuencial y en caso aleatorio

  T_s = 10 ms

  r = 10000 rpm

  320 sectores/pista, 512 bytes/sector

  Leer fichero de 2560 sectores (1'3 MB)

  - Acceso aleatorio a los sectores

    Para leer 1 sector: 10 + 3 + 0,01875 = 13'01875 ms

$$T_r = \frac{1}{2 \times \frac{10000}{60}} = 3ms \qquad T_t = \frac{512}{\frac{10000}{60} \times 512 \times 320} = 0{,}01875ms$$

    Para leer 2500 sectores: 13'01875 * 2560= 32,33 s

Discos. Lectura/escritura.

– Acceso secuencial (fichero compacto ocupando todos los sectores de pistas adyacentes)

Total: 8 pistas (2560 sectores / 320 sectores/pista)
Para leer todos los sectores de la 1ª pista: 10+3+6= 19 ms

$$T_r = \frac{1}{2 \times \frac{10000}{60}} = 3ms \qquad T_t = \frac{512 \times 320}{\frac{10000}{60} \times 512 \times 320} = 6ms$$

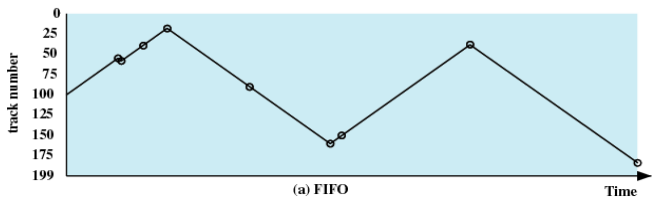
Para las 7 restantes el T_s es despreciable, por tanto: (3 + 6) * 7 = 63 ms
En total: 19 + 63 = 82 ms = 0,082 s

Planificación de disco

- El SO es responsable de usar el hardware de forma eficiente.
- Hablando de discos, esto implica accesos rápidos y mucho ancho de banda.
- El tiempo de acceso tiene dos componentes principales:
  - *búsqueda*: tiempo que tarda el brazo del disco para mover las cabezas hasta el cilindro que contiene el sector deseado.
  - *latencia*: tiempo de espera adicional para que el disco gire hasta ponerse sobre el sector deseado.
- Objetivo: **minimizar el tiempo de búsqueda**, que es proporcional a la distancia de búsqueda.
  - El orden en el que se atienden las solicitudes condiciona el tiempo de búsqueda
- Ancho de banda: *bytes* transferidos / tiempo de transferencia

Algoritmos de planificación del disco.
FIFO (Primero entrar, primero en salir)

- Procesa las peticiones en orden secuencial . Más sencillo
- Cuando atendemos una petición decidimos cual será la siguiente, de acuerdo con el orden de llegada.
  - Estrategia equitativa con todos los procesos. Más justo.
  - Si hay muchos procesos compitiendo por el disco ⇒ rendimiento similar a la planificación aleatoria
- Ejemplo:
  - Cabeza situada en el cilindro 100
  - Solicitudes de E/S a bloques situados en los cilindros: 55, 58, 39, 18, 90, 160, 150, 38, 184



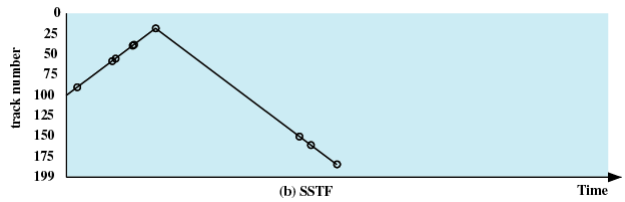
498 pistas atravesadas

Longitud media de búsqueda: 55'3

Algoritmos de planificación del disco.
SSTF (*Shortest Seek Time First)*

- Primero el de tiempo de servicio más corto
  - Selecciona la petición de E/S del disco que requiera un menor movimiento del brazo desde su posición actual
  - Siempre escoge el tiempo de búsqueda mínimo en el momento de inicio del movimiento de la cabeza.
- Idea: maximizar el ancho de banda del disco, colas pequeñas
- Problema: puede causar la inanición de peticiones periféricas.

- Ejemplo:
  - inicial: 100
  - Cadena: 55, 58, 39, 18, 90, 160, 150, 38, 184
  - SSTF: 90, 58, 55, 39, 38, 18, 150, 160, 184



248 pistas atravesadas

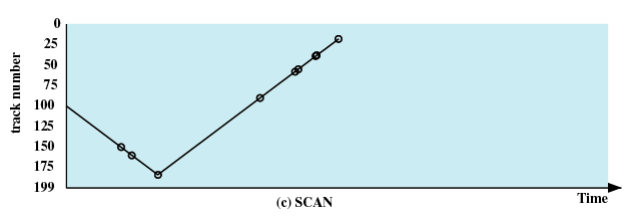
Longitud media de búsqueda: 27'5

Algoritmos de planificación del disco. SCAN (ascensor)

- El brazo se mueve tan sólo en una dirección, satisfaciendo todas las peticiones pendientes que encuentre en su camino, hasta que alcanza la última pista en esa dirección o hasta que no haya más peticiones en esa dirección (mejora denominada política LOOK)
- La dirección de servicio se invierte al llegar al final
- Idea: evitar desplazamientos atrás y adelante (agitar cabezas). Mejor distribución del servicio
- Problemas:
  - Favorece los trabajos cuyas peticiones corresponden con las pistas más cercanas, tanto a las pitas interiores como a las exteriores
  - No aprovecha tan bien la proximidad como SSTF
  - Favorece a los trabajos que han llegado más recientemente
  - Puede retrasar mucho algunas peticiones si no se insertan en el momento adecuado.

Algoritmos de planificación del disco. SCAN (ascensor)

- Ejemplo:
  - inicial: 100
  - Cadena: 55, 58, 39, 18, 90, 160, 150, 38, 184
  - Dirección: Hacia arriba
  - SCAN: 150, 160, 184, [199], 90, 58, 55, 39, 38, 18
  - Con mejora, LOOK: 150, 160, 184, 90, 58, 55, 39, 38, 18



LOOK:

250 pistas atravesadas

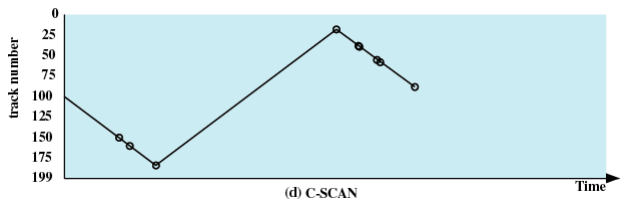
Longitud media de búsqueda: 27’8

Algoritmos de planificación del disco.
C-SCAN

- Restringe la búsqueda a una sola dirección
  - Cuando se ha visitado la última pista en una dirección, el brazo vuelve al extremo opuesto del disco y la búsqueda comienza de nuevo.
  - Con la mejora, C-LOOK, el brazo no va hasta los extremos sino que de la última petición en una dirección va directamente a la 1ª en esa misma dirección
  - No se atienden peticiones mientras las cabezas vuelven a la posición inicial
  - Se pierde muy poco en rendimiento, ya que el movimiento de volver al inicio es uno sólo y mucho más rápido que la suma de pista a pista
  - Reduce el retardo máximo que pueden experimentar nuevas peticiones
  - Gana en equilibrio. Es el más equitativo

Algoritmos de planificación del disco.
C-SCAN

- Ejemplo:
  - inicial: 100
  - cadena: 55, 58, 39, 18, 90, 160, 150, 38, 184
  - Dirección: Hacia arriba
  - C-SCAN: 150, 160, 184, [199], [0], 18, 38, 39, 55, 58, 90
  - Con mejora, C-LOOK: 150, 160, 184, 18, 38, 39, 55, 58, 90



(d) C-SCAN

C-LOOK:

322 pistas atravesadas

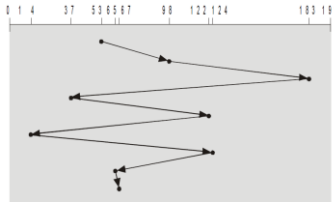
Longitud media de búsqueda: 35’8

Algoritmos de planificación del disco. Selección

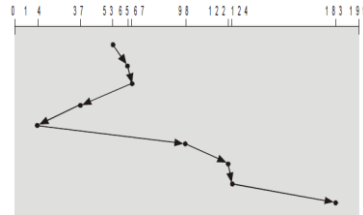
- SCAN Y C-SCAN tiene mejor rendimiento para sistemas que usan mucho el disco
- El rendimiento depende del número y el tipo de peticiones
- Las peticiones al disco pueden depender de la política de asignación de espacio a los ficheros
  - se debería intentar almacenar los bloques de un fichero de manera contigua
- El algoritmo de planificación de disco debe ser escrito como un módulo separado, para que sea fácil de reemplazar
- El algoritmo C-SCAN es el más usado actualmente.
- Se pueden limitar las latencias de algunas técnicas de planificación para operaciones de E/S a disco utilizando colas múltiples:
  - Cola activa: peticiones de procesos com mucha E/S
  - Cola inactiva: procesos con mucho cálculo
  - Se pueden agrupar las peticiones para que las operaciones de l/e sean de sectores próximos

2º ejemplo de planificación

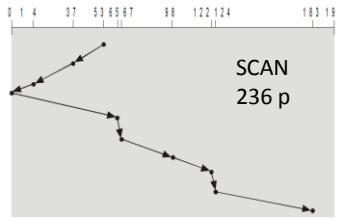
- Supongamos una cola de peticiones para los cilindros: 98, 183, 37, 122, 14, 124, 65, 67
- Se asume que las cabezas están inicialmente en el cilindro 53



FIFO
640 p,
media
80

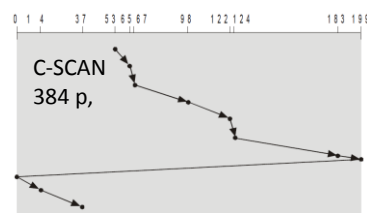


SSTF
236 p,
media
29’5



SCAN
236 p

LOOK
208 p,
media
26



C-SCAN
384 p,

C-LOOK
340 p,
media
42’5

Planificadores de Linux

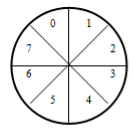
- Noop scheduler
  - Política FIFO con unificación de peticiones para mejorar la eficiencia
    - Cola única de peticiones de l/e, realizando operaciones de ordenamiento y agrupamiento sobre la cola (mismo sector del disco o uno inmediatamente adyacente)
- Deadline scheduler (planificador basado en plazos)
  - Implementa unificación de peticiones más ascensor (variación de LOOK) y caducidad de operaciones
    - Una cola para lectura y otra para escritura, ordenadas por tiempos límite.
      - Una vez que el proceso solicita una petición de escritura, no necesita esperar a que se lleve a cabo, en cambio en muchas peticiones de lectura el proceso debe esperar a que se entreguen los datos a la aplicación antes de continuar
    - Caducidad para escritura: 5s, caducidad para lectura: 0,5s

Planificadores de Linux

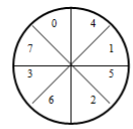
- Anticipatory scheduler (planificador de E/S previsor)
  - Implementa unificación y tiene en cuenta que generalmente tras una lectura se sucede otra en una pista cercana (principio de proximidad)
    - Espera 6ms a cambiar de pista
- Complete fair queuing scheduler ("cfq scheduler")
  - Implementa unificación, elevador más estrategia de Round-robin por proceso y dispositivo.
  - Cada proceso tiene su propia cola gestionada por FIFO con unificación de peticiones

Minimización de los tiempos de latencia y transferencia

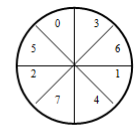
- Tras leer un sector, se transfiere la información a la memoria principal. En este T_t el disco sigue rotando y el sector contiguo queda desplazado de la cabeza lectora y por tanto hay que esperar una nueva vuelta completa del disco para leer el siguiente sector.
- Con una numeración intercalada, si el disco durante el T_t gira como mucho un sector, se podrá leer la pista completa en dos rotaciones del disco.
- En sistemas con r mayor, si el disco durante el T_t, avanza más de un sector y menos de dos, utilizando una numeración intercalada doble, se podrá leer una pista completa en tres rotaciones del disco.



NUM. NO INTERCALADA



NUM. INTERCALADA



NUM. INTERCALADA DOBLE

Discos. Particiones

- Particionar un disco duro es realizar una división en él de modo que, a efectos prácticos, el sistema operativo crea que tienes varios discos duros, cuando en realidad sólo hay un único disco físico dividido en varias partes.
  - De este modo, se pueden modificar o borrar particiones sin afectar a los demás datos del disco.
- Las particiones son como discos duros independientes, y así aparece en Windows.
  - En Linux no existe el concepto de *unidad* (C:, D:, etc.) sino que las particiones se *montan* en el árbol de carpetas.
  - En Ubuntu las particiones se suelen montar en la carpeta **/media**.
- El sector 0 guarda la tabla de particiones

Discos. Particiones

- Los sistemas de archivos indican el modo en que se gestionan los archivos dentro de las particiones.
- Hay varios tipos, normalmente ligados a sistemas operativos concretos:
  - **fat32** o **vfat:**
    - Tradicional de MS-DOS y las primeras versiones de Windows.
    - Padece de una gran fragmentación y es un poco inestable.
  - **ntfs:**
    - Es el nuevo sistema de Windows, usado a partir del 2000 y el XP.
    - Es muy estable.
    - El problema es que es privativo, con lo cual otros sistemas operativos no pueden acceder a él de manera transparente.
    - Desde Linux sólo se recomienda la lectura, siendo la escritura en estas particiones un poco arriesgada.

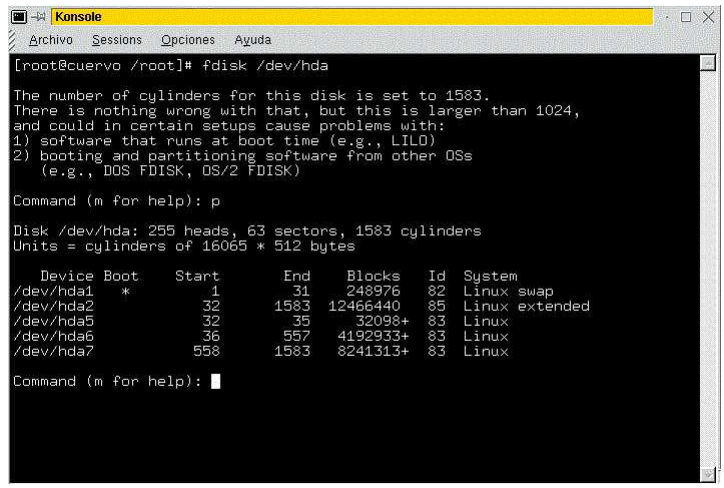
Discos. Particiones

- **ext2:**
  - Inicialmente sistema estándar de Linux.
  - Tiene una fragmentación bajísima
  - Es un poco lento manejando archivos de gran tamaño.
- **ext3:**
  - Versión mejorada de *ext2*, con previsión de pérdida de datos por fallos del disco o apagones.
  - No permite recuperar datos borrados.
  - Considerado el estándar de facto en la comunidad GNU/Linux
- **ext4:**
  - Mejora compatible de ext3, con registro por diario (*Journaling*).
  - Extent: capacidad de reservar un área contigua para un archivo; esto puede reducir y hasta eliminar completamente la fragmentación de archivos.
  - Es el sistema de archivos por defecto desde Ubuntu Jaunty.

Discos. Particiones

- **ReiserFS:**
  - Es el sistema de archivos de última generación para Linux.
  - Organiza los archivos de tal modo que se agilizan mucho las operaciones con éstos.
  - El problema de ser tan actual es que muchas herramientas (por ejemplo, para recuperar datos) no lo soportan.
- **swap:**
  - Es el sistema de archivos para la partición de intercambio de Linux.
  - Todos los sistemas Linux necesitan una partición de este tipo para cargar los programas y no saturar la memoria RAM cuando se excede su capacidad.
  - En Windows, esto se hace con el archivo pagefile.sys en la misma partición de trabajo, con los problemas que conlleva.

Discos. Particiones



Discos. Creación de la estructura lógica

- Un disco o partición puede ser accedido vía dos interfaces distintos...
  - **Dispositivo de bloques**. Todos los accesos pasan por la cache de bloques.
  - **Dispositivo de caracteres**. Acceso directo a bloques, sin pasar por cache.
    - Las peticiones deben ser múltiplo del tamaño de bloque
    - Deben estar alineadas a bloque.
- La operación de formateado lógico (*mkfs* o *format*):
  1. Construye un bloque de carga (BootBlock).
  2. Crea una lista de bloques defectuosos.
  3. Crea un sistema de archivos.

Discos. Creación de la estructura lógica

- El **bloque de carga** se carga y ejecuta al arranque del computador:
  - Realiza un bucle que carga la imagen del SO en memoria.
  - Luego salta a esa posición de memoria y se arranca el SO.
- La **lista de bloques defectuosos**, incluye los bloques en mal estado.
  - Se marcan siempre como ocupados y no se liberan nunca.
  - Un bloque es defectuoso porque alguno de los sectores que lo componen es defectuoso