Ejercicios de Sistemas Operativos. Tema 4: Sistemas de ficheros
E.T.S. Ingeniería Informática. Universidad de Málaga
Profesor: Manuel Ujaldón
Dpto. de Arquitectura de Computadores

---

1. Un disco duro gira a una velocidad angular constante de 1000 RPM (revoluciones por minuto). Calcular su latencia rotacional, es decir, el tiempo máximo que tardará en encontrar un dato en una pista determinada una vez que el cabezal se encuentre ubicado en esa pista.

   El caso más desfavorable es que el sector de la pista acabe justo de pasar tras el cabezal, en cuyo caso, tardará el tiempo necesario para dar una vuelta completa. Dado que gira a 1000 revoluciones por minuto, completa 1000 vueltas en 60 segundos, es decir, tarda 60 milisegundos en dar una vuelta.

2. Calcula la tasa de transferencia o ancho de banda máximo en Mbytes/sg. de un disco duro que gira a 7200 RPM y tiene 16 sectores de 8 Kbytes en cada pista.

   El disco recorre 7200 pistas por minuto, y cada pista alberga 16 sectores de 8 Kbytes, o sea, 128 Kbytes de datos. Por tanto, se leen 7200 × 128 Kbytes / minuto = 921.600 Kbytes / minuto = 15 Mbytes / sg.

3. Un disco duro se compone de 32 cabezales *(heads)*, 16 platos *(platters)*, 25600 cilindros *(cylinders)* y 128 sectores por pista, con un tamaño de sector de 8 Kbytes (suponer que coinciden el tamaño del sector y del *cluster*). Se pide calcular la capacidad del disco y el espacio ocupado por una FAT32.

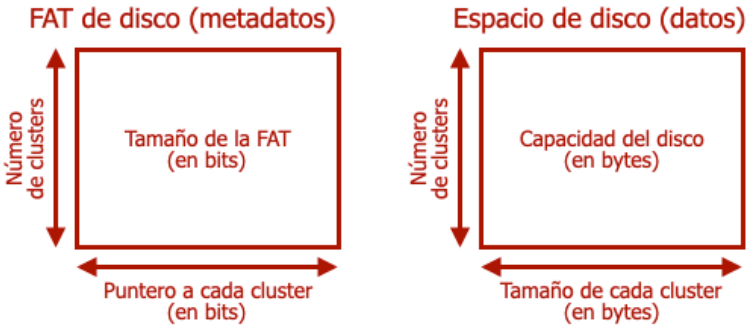
   Cada plato tiene dos superficies de lectura, el envés y el revés, por lo que para 16 platos son necesarios los 32 cabezales descritos. Por otro lado, cada cilindro consta de 32 pistas paralelas situadas a diferentes alturas (una para cada cabezal de ese cilindro). Con todo esto, tenemos:
   Capacidad de disco = 32 cabezales × 25600 pistas/cabezal × 128 sectores/pista × 8 Kbytes/sector = $2^5 \times 2^8 \times 100 \times 2^7 \times 2^{13}$ bytes = 800 Gbytes.
   Respecto al espacio ocupado por la FAT32, ésta albergará un puntero de 32 bits a cada sector. El número total de sectores es de 32 cabezales × 25600 pistas/cabezal × 128 sectores/pista = 100 Msectores. Por lo tanto, se necesitan 100 Msectores × 32 bits = 3200 Mbits = 400 Mbytes.
   Asumiendo que en el disco hay algunos metadatos de gestión adicionales a la FAT de esta partición, el espacio útil de disco para esta partición única estaría en torno a los 799.5 Gbytes.

4. Un Sistema Operativo utiliza clusters de 1 Kbyte y una FAT12 para gestionar el almacenamiento de un disco duro. ¿Qué capacidad máxima puede tener cada partición de ese disco? ¿Y si se trata de una FAT32? ¿Cuánto ocupa la FAT en cada caso?

   La capacidad máxima para una FAT12 es de 2^{12} clusters. Puesto que en este caso los clusters son de 1 Kbyte, esta capacidad sería de $2^{12}\times$ 1 Kbyte = 4 Mbytes.
   Para una FAT32, la capacidad máxima es de 2^{32} clusters, esto es, $2^{32}\times$ 1 Kbyte = 4 Tbytes.
   La FAT12 ocupará 2^{12} entradas de 12 bits, es decir, 6 Kbytes.
   La FAT32 ocupará 2^{32} entradas de 32 bits, es decir, 16 Gbytes.

5. Sea un disco de 4 Terabytes formateado en FAT32. ¿Cuál es el tamaño mínimo del *cluster*? ¿Qué espacio ocupa la FAT32?

Una FAT32 emplea 32 bits en el direccionamiento a los clusters, de ahí que 2^{32} clusters deben cubrir al menos los 4 Terabytes del espacio de disco. Esto nos conduce a la siguiente expresión: 2^{32} clusters $\times$ (Tamaño del cluster) $\geq$ 4 Terabytes. Despejando tenemos:
Tamaño del cluster $\geq \frac{4\ Terabytes}{2^{32}} = \frac{2^{42}\ bytes}{2^{32}} = 2^{10}\ bytes = 1$ Kbyte.
Para calcular el espacio ocupado por la FAT32, sabemos que ésta aloja 2^{32} entradas de 32 bits cada una, y por lo tanto, ocupa $2^{32}\ entradas \times 32\ bits/entrada = 2^{32} \times 4\ bytes = 16$ Gbytes.
Las siguientes figuras pueden ayudarnos a establecer visualmente las dos relaciones que debemos tener claras para resolver este ejercicio.



6. Sea un disco de 2 Terabytes formateado en FAT32 con clusters de 64 Kbytes. Se pide:

a) Calcular el número de clusters del disco.

Si dividimos el espacio total de 2 Terabytes entre el tamaño que ocupa cada cluster, obtendremos el número de clusters del disco:
$\frac{2\ Tbytes}{64\ Kbytes} = \frac{2^{41}\ bytes}{2^{16}\ bytes} = 2^{25}\ clusters = 32$ Mclusters tiene el disco.

b) Calcular el tamaño de la FAT32.

La FAT tiene tantas entradas como clusters tiene el disco, es decir, 2^{25} entradas. Cada entrada ocupa 32 bits (4 bytes), por lo que la FAT ocupa $2^{25}\ entradas \times 4\ bytes/entrada = 2^{27}\ bytes = 128$ Mbytes.

c) ¿Cuál es el número máximo de archivos que podemos tener en ese disco?

El disco puede alojar tantos ficheros como entradas tiene la FAT, siempre que cada fichero ocupe un cluster como máximo. Por lo tanto, serían un total de 32 Mficheros, o 32x1024x1024 ficheros. Notar que en este caso, todas las entradas de la FAT tendrían un valor `NULL`, puesto que no hay ninguna cadena de clusters que construir dentro de ella. El puntero que localiza el cluster que ocupa cada fichero no vendría dado por la FAT32, sino por la cabecera de dicho fichero junto a otros campos como el nombre del fichero y sus permisos.

d) ¿Qué directorio sufriría un mayor impacto de la fragmentación interna: uno que contiene 50 ficheros de 1 Kbyte u otro que contiene 100 ficheros de 2 Kbytes?

Cada fichero debe alojar un cluster de datos, que en este caso es de 64 Kbytes. Si necesita menos tamaño, como es el caso, el resto se pierde por fragmentación interna. Por lo tanto:

- Para el primer directorio, cada uno de los 50 ficheros desperdician 63 Kbytes, y en total se pierden $50 \times 63\ Kbytes = 3150$ Kbytes de disco.
- Para el segundo directorio, cada uno de los 100 ficheros desperdician 62 Kbytes, perdiéndose en total $100 \times 62\ Kbytes = 6200$ Kbytes de disco.

En consecuencia, el segundo directorio desperdicia más espacio por fragmentación interna.

7. Un Sistema Operativo UNIX utiliza clusters de 2 Kbytes y punteros de 32 bits para direccionarlos, gestionando el espacio de almacenamiento para cada archivo en disco mediante el típico i-nodo que registra 10 punteros directos, un puntero indirecto simple, un puntero indirecto doble y otro triple. Responder a las siguientes cuestiones:

a) ¿Cuántos punteros de clusters caben en un cluster de disco?

El *cluster* es de 2 Kbytes y los punteros son de 32 bits (4 bytes). Por lo tanto, en un *cluster* caben $\frac{2\ Kbytes}{4\ bytes} = 512$ punteros.

b) ¿Cuál es el máximo tamaño de fichero soportado?

Con los 10 punteros directos podemos direccionar hasta 10 × 2 Kbytes = 20 Kbytes.
Dado que en un *cluster* caben 512 punteros a *clusters*, con el puntero indirecto simple (PIS) podemos direccionar hasta 512 *clusters* × 2 Kbytes/cluster = 1 Mbyte.
El puntero indirecto doble (PID) direcciona 512 PIS × 1 Mbyte/PIS = 512 Mbytes.
El puntero indirecto triple (PIT) direcciona 512 PID × 512 Mbytes/PID = 256 Gbytes.
En total, el espacio máximo de disco sería 256 Gbytes + 512 Mbytes + 1 Mbyte + 20 Kbytes.

c) ¿Cuántos punteros indirectos deben habilitarse para albergar un fichero de 20 Mbytes?

Para albergar un fichero de 20 Mbytes hace falta habilitar hasta el PID, ya que con él llegamos hasta 512 Mbytes, mientras que con el PIS sólo llegamos a 1 Mbyte.

8. La organización de un Sistema Operativo Linux utiliza i-nodos con 16 punteros directos a *clusters* de 8 Kbytes, un puntero indirecto simple, otro doble y otro triple. Los punteros son de 32 bits, de los cuales 8 tienen información de control para la partición y los 24 bits restantes se utilizan para direccionar al *cluster*. Se pide:

a) ¿Cuál es el máximo tamaño de fichero soportado?

Con los 16 punteros directos podemos direccionar hasta 16 × 8 Kbytes = 128 Kbytes.
Por otro lado, en un *cluster* de 8 Kbytes caben 8 Kbytes / 4 bytes = 2048 punteros.
El puntero indirecto simple (PIS) referencia a un *cluster* lleno de punteros, es decir, contendrá 2048 punteros para direccionar hasta 2048 *clusters* × 8 Kbytes/cluster = 16 Mbytes.
El puntero indirecto doble (PID) direcciona 2048 PIS × 16 Mbyte/PIS = 32 Gbytes.
El puntero indirecto triple (PIT) direcciona 2048 PID × 32 Gbytes/PID = 64 Tbytes.
En total, el espacio máximo de disco sería 64 Tbytes + 32 Gbytes + 16 Mbytes + 128 Kbytes

b) ¿Cuál es el máximo tamaño de partición soportado?

Con 24 bits de dirección para el *cluster* se pueden gestionar hasta 2^{24} *clusters* de 8 Kbytes. Multiplicando ambas cifras, obtenemos que 128 Gbytes es el máximo tamaño de una partición.

c) ¿Puede ser el límite máximo para la partición inferior al límite máximo para cada uno de sus ficheros? ¿Por qué? ¿Cuál es el tamaño máximo de fichero que puede abarcarse al incluir en el i-nodo el puntero indirecto triple?

La partición siempre será más grande que los ficheros, puesto que alberga a todos ellos. Lo que ocurre aquí es que para una partición de 128 Gbytes no queremos limitar el máximo tamaño de archivo a 32 Gbytes, que es lo que se alcanza con el puntero indirecto doble. Habilitando el puntero indirecto triple para superar esta cota, el direccionamiento a disco

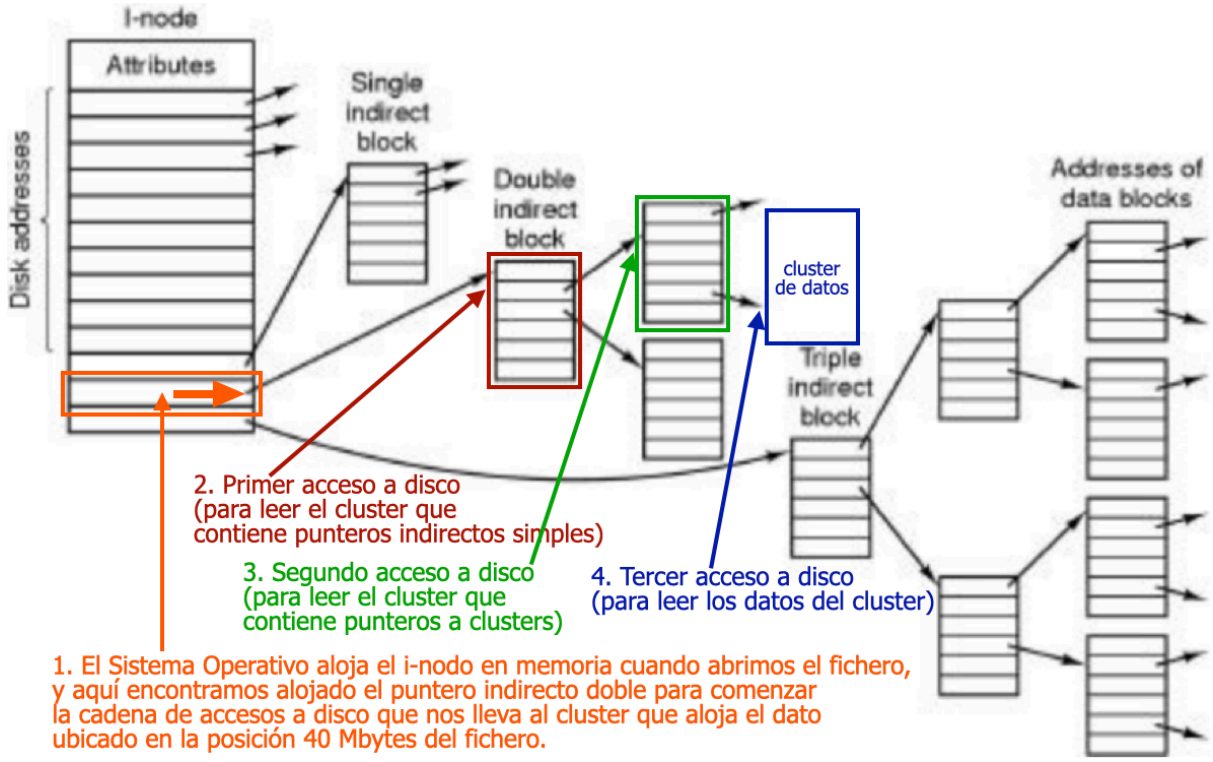
puede alcanzar hasta 64 Tbytes, pero en esta partición desaprovecharíamos buena parte de este rango, limitándonos a cubrir con el puntero indirecto triple el rango de datos que comienza en los 32 Gbytes y concluye en los 128 Gbytes, donde finaliza la partición. Para apurar todos esos 128 Gbytes, el fichero tendría que ser el único del sistema, lo que sugiere prescindir del puntero indirecto triple dentro del i-nodo y limitar el fichero a los 32 Gbytes.

d) ¿Cuántos accesos a disco son necesarios si quisiéramos acceder al byte 41.943.040 del fichero? (ese número es $40 \times 1024 \times 1024$, es decir, 40 Mbytes)

De la solución de los apartados anteriores sabemos que con los 16 punteros directos alcanzamos a direccionar los primeros 128 Kbytes del fichero, y con el puntero indirecto simple llegamos hasta los 16 Mbytes. Por lo tanto, es necesario habilitar el puntero indirecto doble para acceder al dato alojado en la posición 40 Mbytes (para que hubiera hecho falta el puntero indirecto triple tendrían que habernos solicitado un byte por encima de los 32 Gbytes + 16 Mbytes + 128 Kbytes). Si suponemos que el i-nodo de ese fichero se carga en memoria cuando se abre su fichero, consultando el i-nodo obtenemos el puntero indirecto doble, y a partir de ahí:

1) El primer acceso a disco lo necesitamos para acceder al *cluster* que alberga los punteros indirectos simples.
2) Uno de estos punteros indirectos simples corresponde al acceso en curso, y nos lleva de nuevo al disco, para obtener el *cluster* que alberga los punteros a los clusters de datos.
3) El tercer acceso a disco se necesita para acceder a los datos de dicho *cluster*.

En total, se necesitan tres accesos a disco. El siguiente diagrama muestra la secuencia de accesos narrada:



9. Un disco duro de 16 Terabytes está formateado con una única partición de i-nodos en Linux que utiliza *clusters* de 4 Kbytes.

a) ¿Cuántos bits serían necesarios para que los punteros a estos *clusters* puedan direccionar todo el espacio de datos del disco?

16 Terabytes/disco / 4 Kbytes/cluster = 2^{32} *clusters*/disco. Por lo tanto, son necesarios exactamente 32 bits para los punteros a los *clusters*.

b) Si la partición anterior alberga un fichero `miscosas.txt` de 30 Kbytes, ¿cuántos punteros o índices a *clusters* contiene el i-nodo de ese fichero? (considera que el i-nodo no guarda bits de control ni de partición en el puntero, sino únicamente los bits necesarios para direccionar a los *clusters*)

Son necesarios 8 clusters de 4 Kbytes (32 Kbytes), y por lo tanto, es suficiente con 8 punteros directos.

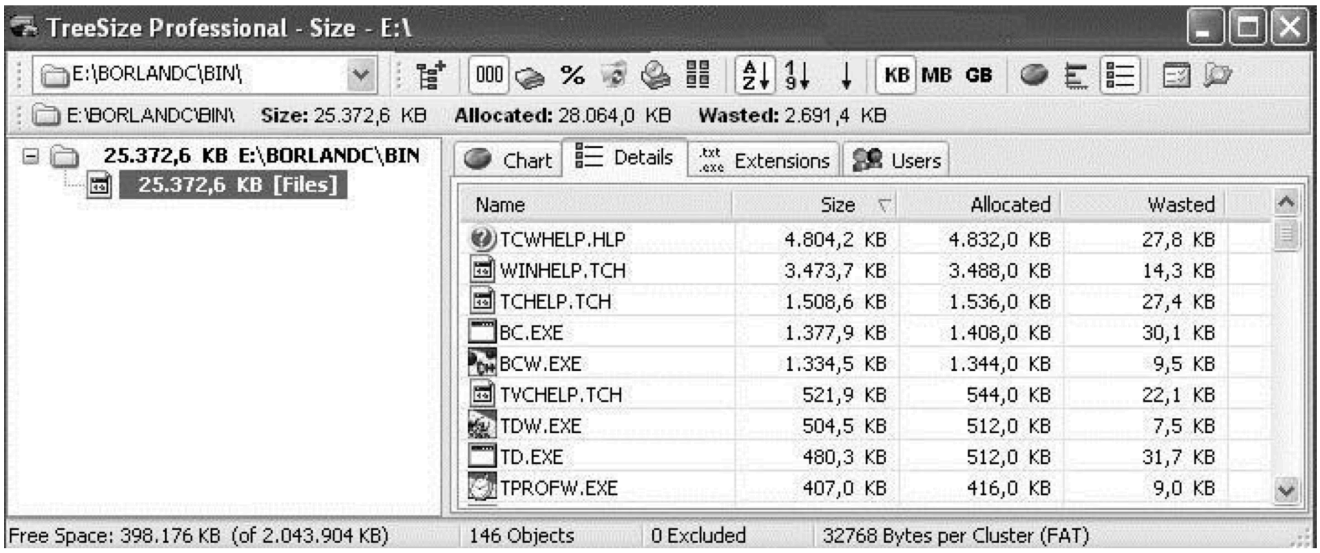
c) ¿Y si `miscosas.txt` ocupara 30 Mbytes?

El puntero indirecto simple accede a un *cluster* que alberga 1024 punteros de 32 bits que pueden alojar hasta 1024 *clusters* de datos. 1024 *clusters* × 4 Kbytes/cluster son 4 Mbytes. Por lo tanto, se nos queda corto el puntero indirecto simple y hace falta el puntero indirecto doble.

d) ¿Y si `miscosas.txt` ocupara 30 Gbytes?

El puntero indirecto simple accede a un cluster que alberga 1024 punteros de 32 bits que pueden alojar hasta 1024 clusters de datos en el disco. 1024 clusters × 4 Kbytes/cluster suponen 4 Mbytes. El puntero indirecto doble accede a un cluster que contiene 1024 punteros indirectos simples, con capacidad para direccionar 1024 × 4 Mbytes, es decir, 4 Gbytes. Por lo tanto, hace falta el puntero indirecto triple para llegar a los 30 Gbytes.

10. Un PC bajo Windows XP formatea un disco duro de 2 GBytes bajo un sistema de archivos FAT con tamaño de cluster de 32 Kbytes. A la vista de la siguiente ventana informativa:



Se pide contestar a las siguientes cuestiones:

a) La columna *Size* que informa del tamaño del archivo no coincide con la columna *Allocated* que indica el espacio que el Sistema Operativo reserva para él. ¿Por qué? ¿Cómo se denomina técnicamente a esta pérdida?

Los archivos sólo pueden reservar espacio por clusters completos, por lo que el remanente que quede del último cluster alojado se desperdicia por la fragmentación interna del disco.

b) Dicha pérdida se cuantifica en la columna *Wasted* para cada fichero. ¿Cómo podríamos reducir esta pérdida? ¿Provocaría este cambio algún efecto negativo?

Como se observa en la columna *Wasted*, la pérdida para cada fichero oscila entre 0 bytes y el tamaño del cluster, y por tanto puede minimizarse reduciendo el tamaño del cluster. Esto tiene dos efectos negativos: Primero, se aprovecha menos el ancho de banda, cobrando un mayor protagonismo la latencia del disco. Segundo, la FAT ocupa más espacio y cada fichero necesitará apropiarse de un mayor número de clusters, lo que también supone asumir un mayor riesgo de que los clusters no se encuentren alojados consecutivamente en disco y haya que posicionar varias veces los cabezales del brazo de lectura/escritura.

c) Establecer una fórmula matemática que relacione las variables *Size*, *Allocated* y *Waster*.

Allocated = ceil(size/32KB) × 32 KB Wasted = Allocated - Size

d) ¿Qué tipo de FAT sería suficiente para este disco, FAT12, FAT16 o FAT32?

El disco de 2 Gbytes (2^{31} bytes) se descompone en 2^{16} clusters de 2^{15} bytes/cluster. Por lo tanto, con punteros de 16 bits a los clusters tendríamos suficiente para direccionar todo el espacio del disco, y podría tratarse de una FAT16.

e) Calcula el espacio de disco ocupado por dicha FAT.

Esta FAT16 ocuparía una tabla de 2^{16} entradas de 16 bits/entrada, para un tamaño total de 128 Kbytes.

f) ¿Existe algún indicio en la ventana informativa anterior que indique que pueda tratarse de una FAT12, FAT16 o una FAT32?

No.

g) Si suponemos despreciable el tamaño ocupado por la FAT y el resto de metadatos de la partición del disco, ¿cuál sería el máximo número de ficheros que podría albergar esta partición?

Para maximizar el número de ficheros, tendríamos que hacerlos todos tan pequeños como un cluster, y entonces podríamos alojar tantos ficheros como clusters tiene la partición. Habría entonces $\frac{2\ GB}{32\ KB} = \frac{2^{31}\ bytes}{2^{15}\ bytes} = 2^{16}$ ficheros como máximo.

h) ¿Cuáles son las principales debilidades de un sistema de ficheros basado en FAT como éste respecto a otro basado en índices?

Los inconvenientes que presenta la FAT son básicamente tres. Listados de mayor a menor importancia, tenemos:

1) Las cadenas de enlaces compactadas en una estructura de datos común para todos los ficheros supone que si esta estructura sufre alguna anomalía o error, podría arruinar el acceso a un elevado volumen de datos.
2) Para acceder al i-ésimo cluster, hay que recorrer toda la cadena de enlaces en la FAT desde el principio, leyendo un total de i entradas de la FAT, que es un proceso lento a no ser que alojemos la FAT en memoria caché.
3) Tendríamos ocupados los 128 Kbytes que ocupa la FAT de manera fija, esto es, a pesar de que la partición pueda estar casi completamente vacía de ficheros por parte del usuario.