

1.1.1. ALGORITMOS PARA GESTIONAR LAS PETICIONES DE ACCESO A DISCO

Existen 4 algoritmos para gestionar las peticiones de acceso a disco:

1.1.1.1. Algoritmo FCFS (\cong First Come, First Served)

La planificación FCFS (First Come, First Served – Primero en llegar, primero en ser servido) es la planificación más sencilla. Como se desprende de su propio nombre se dará servicio a las solicitudes de acceso a disco de la cola según el orden de llegada de dichas solicitudes. Esta planificación hará uso de una cola tipo FIFO (First In, First Out – Primero en entrar, primero en salir).

Se puede considerar que este algoritmo es evidentemente justo. Sin embargo tiene un **inconveniente**, en promedio, puede dar lugar a tiempos bastante grandes.


2

EJEMPLO

Considerar un controlador de disco con la cabeza lectora posicionada en la pista 99 y la dirección de búsqueda creciente. La cola de peticiones es la siguiente:

Peticiones:	81	142	86	172	89	145	97	170	125
-------------	----	-----	----	-----	----	-----	----	-----	-----

Vamos atendiendo las peticiones según nos van llegando. Este algoritmo funciona bien cuando no tenemos demasiadas peticiones:

Disco:	1	81	86	89	97	99	125	142	145	170	172
Orden en el que atiende a las peticiones:		2º	4º	6º	8º		10º	3º	7º	9º	5º

1.1.1.2. Algoritmo SSF (\cong Shortest Seek First)

La planificación SSF (Shortest Seek First – Primero la búsqueda más cercana) atiende primero la solicitud de la cola de solicitudes pendientes que quiere acceder al cilindro más cercano al de la solicitud actual, que se está procesando. Es decir, atiende primero la petición que requiere el menor movimiento de la cabeza de lectura/escritura desde su posición actual.

El algoritmo SSF es un algoritmo bastante habitual. Un **inconveniente** que aparece es que pueden llegar solicitudes que impliquen cilindros próximos al actual, por lo que estas solicitudes

serán atendidas enseguida mientras que otras que llegaron antes, con cilindros más alejados, no se atenderán. Esta situación se conoce con el nombre de bloqueo indefinido.

Este algoritmo elige siempre la opción que incurre en el menor tiempo de búsqueda respecto a la posición actual. Sin embargo, éste tampoco es un algoritmo óptimo; es decir, no garantiza que la secuencia elegida sea la que menor tiempo promedio de búsqueda tenga.

3

EJEMPLO

Considerar un controlador de disco con la cabeza lectora posicionada en la pista 99 y la dirección de búsqueda creciente. La cola de peticiones es la siguiente:


Peticiones:	81	142	86	172	89	145	97	170	125
-------------	----	-----	----	-----	----	-----	----	-----	-----

Vamos atendiendo las peticiones que requieren menos tiempo del dispositivo, es decir la que se encuentra mas cerca de la petición que se está procesando:

Disco:

1	81	86	89	97	99	125	142	145	170	172
---	----	----	----	----	----	-----	-----	-----	-----	-----

Orden en el que
atiende a las
peticiones:

5° 4° 3° 2°  6° 7° 8° 9° 10°

1.1.1.3. Algoritmo SCAN o ALGORITMO DEL ASCENSOR

La planificación Scan, también llamada algoritmo del ascensor porque se comporta como tal: va dando servicio a las solicitudes que van encontrando en el sentido en el que se van desplazando las cabezas de lectura/escritura por el disco. Cuando no hay más solicitudes en ese sentido, o se llega al extremo, se invierte el sentido para hacer lo mismo otra vez pero yendo hacia el otro lado. Por tanto, en este algoritmo es necesario tener un bit que indique el sentido del movimiento.

Este algoritmo evita el bloqueo indefinido que se puede producir con la planificación SSF. Una propiedad interesante de este algoritmo es que dada cualquier colección de solicitudes, la cuota máxima del total de movimientos está fijada: es el doble del número de cilindros. En general, el algoritmo del ascensor es peor que el SSF, aunque es más apropiado para sistemas que hacen gran uso del disco.

4

EJEMPLO

Considerar un controlador de disco con la cabeza lectora posicionada en la pista 99 y la dirección de búsqueda creciente. La cola de peticiones es la siguiente:

Peticiones:

81	142	86	172	89	145	97	170	125
----	-----	----	-----	----	-----	----	-----	-----

El brazo del disco se mueve en un único sentido. Sólo se atenderá la petición más cercana en el sentido en el que estemos recorriendo el disco. Una vez alcanzada la última pista, se sigue el movimiento en sentido opuesto:

Disco:

1	81	86	89	97	99	125	142	145	170	172
---	----	----	----	----	----	-----	-----	-----	-----	-----

Orden en el que
atiende a las
peticiones:

10° 9° 8° 7°  2° 3° 4° 5° 6°

1.1.1.4. Algoritmo C-SCAN O ALGORITMO SCAN CIRCULAR

Supongamos una distribución uniforme de solicitudes de pistas. En la planificación Scan, cuando la cabeza llega a un extremo e invierte la dirección, en la zona próxima a dicho extremo habrá pocas solicitudes, ya que las solicitudes para acceder a los cilindros de dicha zona acaban de ser servidas. La mayor densidad de solicitudes se encontrará en el extremo opuesto del disco. El algoritmo C-Scan o Scan Circular, trata de evitar el problema anterior restringiendo el rastreo a una única dirección. En esta planificación la cabeza se mueve de un extremo del disco al otro, atendiendo las solicitudes que va encontrando, pero al llegar al extremo opuesto, regresa de inmediato al otro sin servir ninguna solicitud.

5

EJEMPLO

Considerar un controlador de disco con la cabeza lectora posicionada en la pista 99 y la dirección de búsqueda creciente. La cola de peticiones es la siguiente:

Peticiones:

81	142	86	172	89	145	97	170	125
----	-----	----	-----	----	-----	----	-----	-----

El brazo del disco se mueve en un único sentido, y de forma circular. Sólo se atenderá la petición más cercana en el sentido en el que estemos recorriendo el disco. Una vez alcanzada la última pista, volvemos a la primera pista.

Disco:

1	81	86	89	97	99	125	142	145	170	172
---	----	----	----	----	----	-----	-----	-----	-----	-----

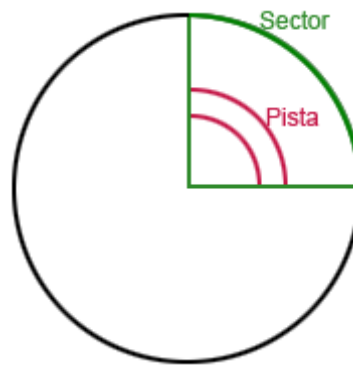
Orden en el que
atiende a las
peticiones:

7° 8° 9° 10°  2° 3° 4° 5° 6°

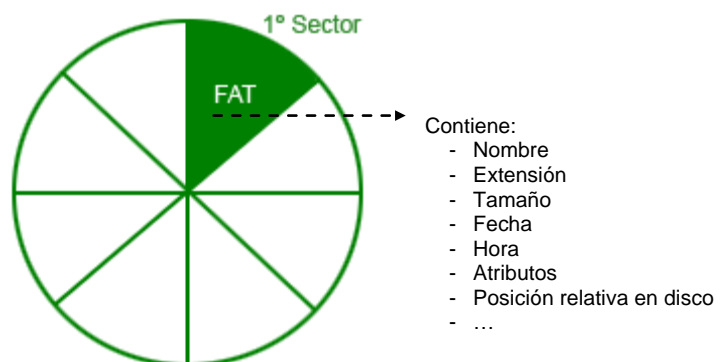
1.1.2. FAT

FAT \cong File Allocation Table \cong Tabla de Localización de Ficheros

La FAT es uno de los sistemas de archivos utilizados por los primeros ordenadores. Sabíamos que un disco está dividido en sectores y cada sector en pistas:



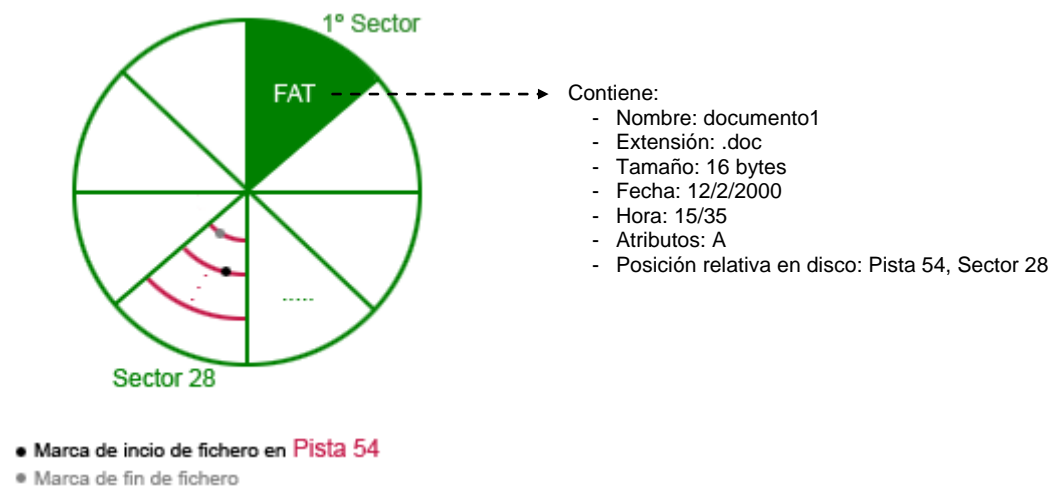
La FAT se almacena en el primer sector del disco y la FAT contiene todas las características de los ficheros: Nombre, Extensión, Tamaño, Fecha, Hora, Atributos, Posición relativa en disco, ...:



Tenemos un fichero con las siguientes características:

- Nombre: documento1
- Extensión: .doc
- Tamaño: 16 bytes
- Fecha: 12/2/2000
- Hora: 15/35
- Atributos: A (de archivo)
- Posición relativa en disco: Pista 54, Sector 28

La posición del fichero en disco y el contenido de la FAT se ve a continuación:

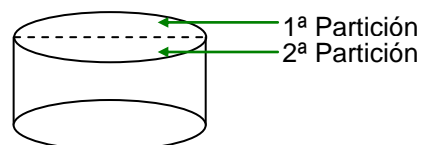


1.2. DEFINICIONES

Debemos conocer las siguientes definiciones:

1.2.1. PARTICIONES

Trozos del disco que simulan discos independientes.



1.2.2. CLUSTER

Es el mínimo tamaño que va a ocupar un archivo en el disco.

En un cluster sólo entra 1 archivo, o parte de ese archivo \Rightarrow Lo que queda por grabar de ese archivo se graba en otro cluster. Pero en un cluster no puede haber 2 archivos.

En definitiva, un cluster es la unidad más pequeña de almacenamiento en un disco duro.

Un disco con formato FAT se divide en clusters.

11 EJEMPLO

Tenemos la siguiente información:

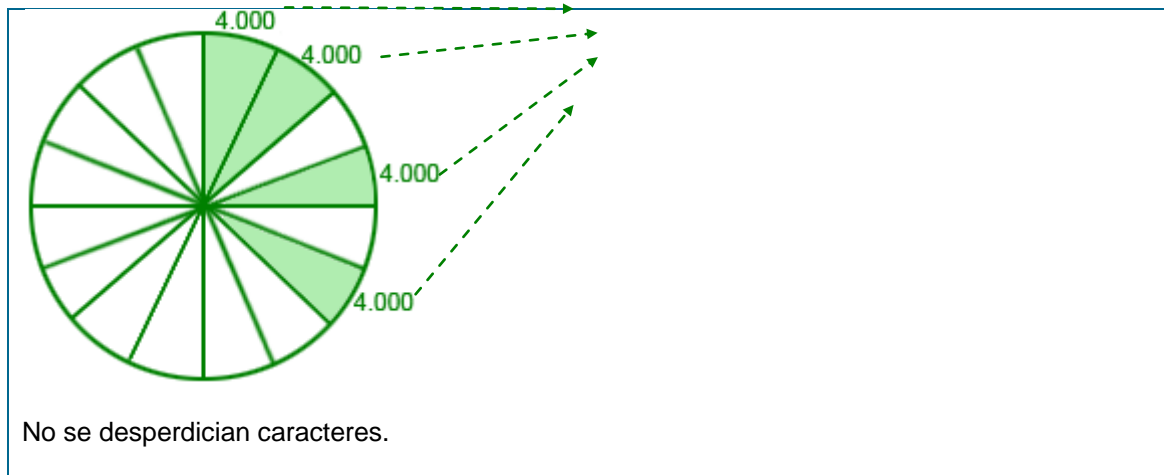
- Archivo documento1.txt de - 16.000 caracteres
- Tamaño del cluster – 10.000 caracteres

$10.000 + 6.000 = 16.000$

$10.000 - 6.000 = 4.000$ caracteres que se desperdician del cluster, pues ese trozo no se puede aprovechar para otro archivo, ya que en un cluster no puede haber 2 archivos.

Cuanto más pequeño sea el cluster menos espacio desperdiciamos, entonces vamos a ver que pasa si en este ejemplo el tamaño del cluster es de 4.000 caracteres:

$$\begin{array}{r} 4.000 \\ 4.000 \\ 4.000 \\ + 4.000 \\ \hline 16.000 \end{array}$$



1.2.3. FRAGMENTACIÓN

Según lo anterior puede interesar hacer que el tamaño del cluster sea más pequeño para no desperdiciar espacio. Pero si se hace mas pequeño, al ir guardando el archivo a trozos se tarda más y entonces aparece la fragmentación.

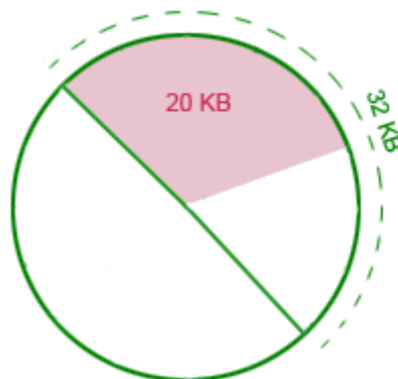
Fragmentar \cong Trocear \cong Dividir

12

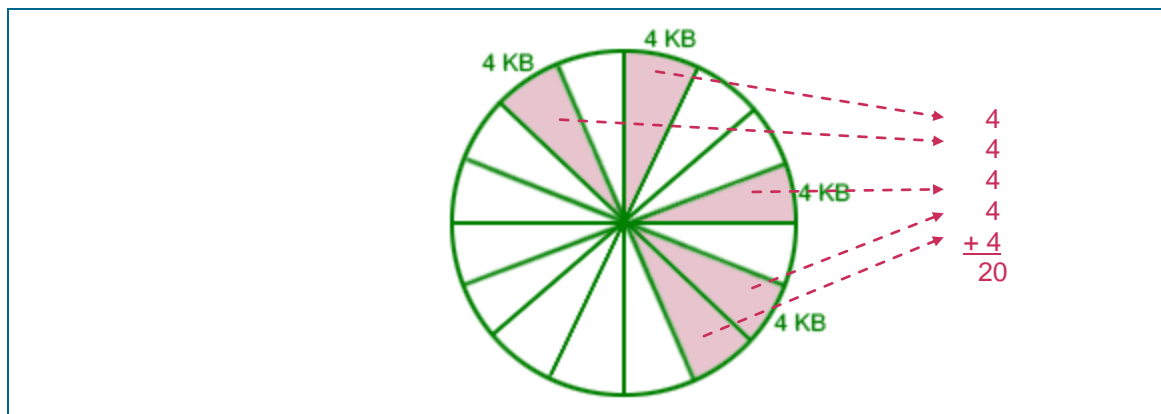
EJEMPLO

Se va a almacenar en el disco un archivo de 20 KB:

- Si se utiliza FAT 16 con clusters de 32 KB \Rightarrow El archivo no aparece fragmentado:



- Si se utiliza FAT 32 con clusters de 4 KB \Rightarrow Se necesitarán 5 clusters para formar el archivo, pero esos trozos del archivo no están seguidos \Rightarrow Hay fragmentación



2. INSTALACIÓN DE SISTEMAS OPERATIVOS EN DIFERENTES PARTICIONES

2.1. ORGANIZACIÓN DEL DISCO DURO

El disco duro (\cong HD) almacena el sistema operativo y los datos del usuario. Vamos a ver como organizamos ese disco duro en el caso de que queramos instalar varios sistemas operativos. Para ello utilizamos las particiones:

2.2. CREAR PARTICIONES

Las particiones son las divisiones del disco duro, mediante las cuales podemos organizar la información. Entonces en cada partición se puede instalar los diferentes sistemas operativos.

2.2.1. TIPOS DE PARTICIONES

2.2.1.1. Partición primaria

Contiene el sistema operativo y se encarga de arrancar el ordenador.

Ejemplo - En el caso del sistema operativo MS_DOS, la partición primaria es la partición usada para la puesta en marcha del DOS donde se almacenan los archivos: IO.SYS , MS DOS.SYS , COMMAND.COM.

2.2.1.2. Partición extendida

Sólo contiene (unidades lógicas) (\cong datos). Los datos se refiere a todo: datos, programas,...

La partición extendida se divide en unidades lógicas.

2.2.1.3. Unidades lógicas

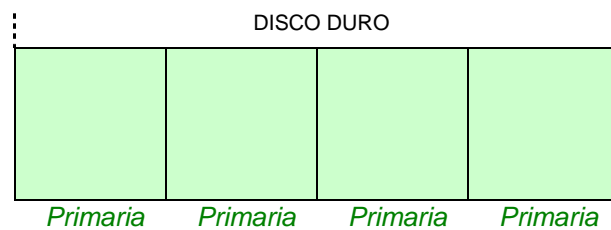
Si se crea una partición extendida, será preciso dividir esta en unidades lógicas. Cada una de estas unidades lógicas posee su propia letra identificativa, a partir de la D.

En una partición extendida se requiere al menos una unidad lógica ya que no es posible almacenar datos en la partición extendida directamente.

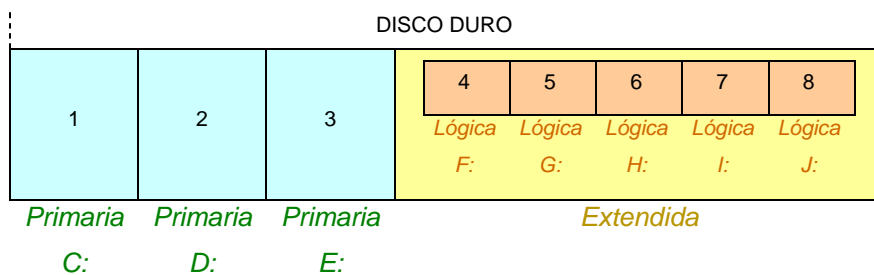
2.2.2. ORGANIZACIÓN DEL DISCO DURO (Caso práctico)

Un disco duro puede tener:

- Como máximo 4 particiones primarias. Por ese motivo se crearon las particiones extendidas ya que en ellas se pueden crear todas las unidades lógicas que se deseen.



- 3 particiones primarias y 1 partición extendida (dividida en 5 unidades lógicas)



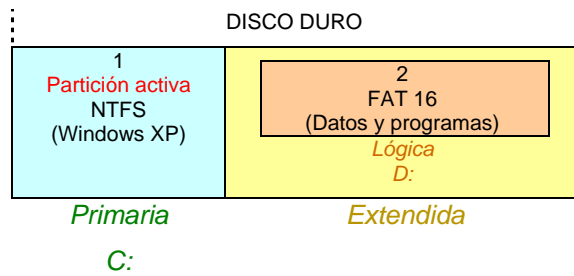
Si tuviésemos instalado el sistema operativo Windows, entonces en el Explorer veríamos las unidades (C, D, E, F, G, H, I, J), que no son otra cosa que las particiones:



13

EJEMPLO

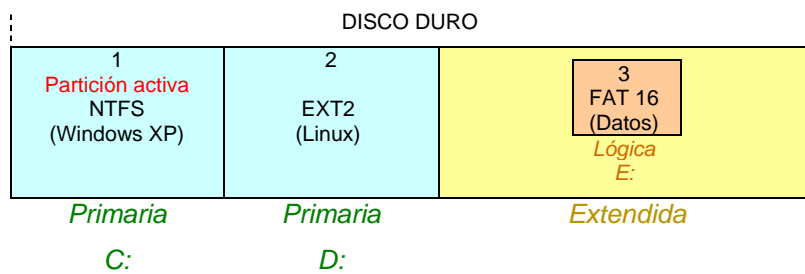
Un disco duro que contiene 1 sistema operativo:



14

EJEMPLO

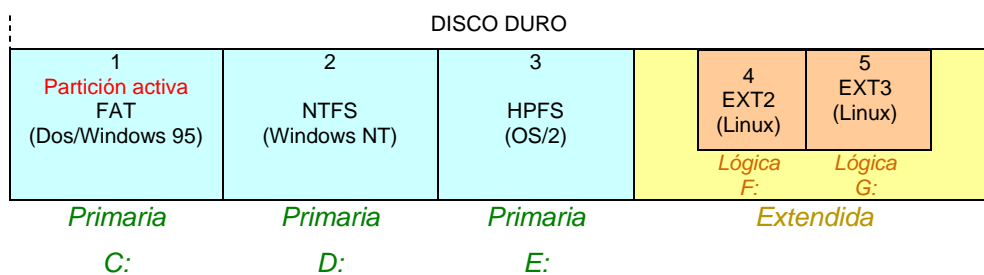
Un disco duro que contiene 2 sistemas operativos:



15

EJEMPLO

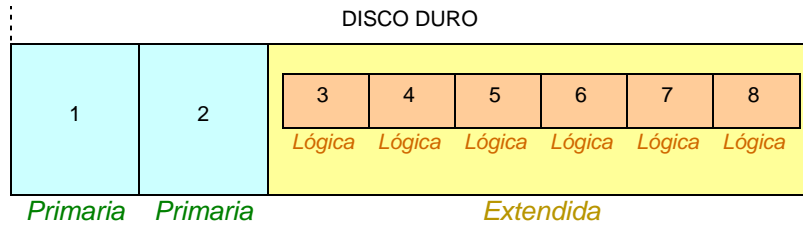
Un disco duro que contiene 5 sistemas operativos:



Si queremos crear 8 particiones, como sólo deja crear un máximo de 4

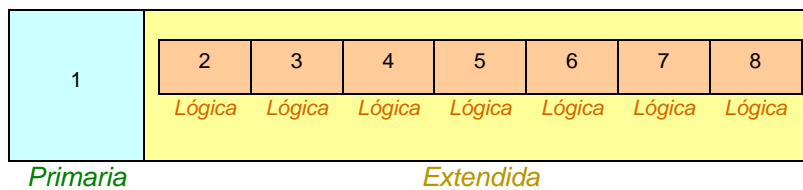
- **1ª Opción – Crear:**

- 2 Primarias
- 1 Extendida y dentro de esta 6 lógicas



- **2ª Opción – Crear:**

- 1 Primaria
- 1 Extendida y dentro de esta 7 lógicas



- Puede haber más opciones