**1) Hay distintos tipos de File System, pero todos impiden crear el Directorio Raíz de un disco lógico mediante la operación crear directorio.**

Verdadero en ningun fs se puede volver a crear el MFD o root.

In the two-level directory structure, each user has his own user file directory (UFD). The UFDs have similar structures, but each lists only the files of a single user. When a user job starts or a user logs in, the system’s master file directory (MFD) is searched. The MFD is indexed by user name or account number, and each entry points to the UFD for that user (Figure 11.10). When a user refers to a particular file, only his own UFD is searched. Thus, different users may have files with the same name, as long as all the file names within each UFD are unique. To create a file for a user, the operating system searches only that user’s UFD to ascertain whether another file of that name exists. silber 9th 543

**2)En un sistema multiprogramado, alcanza con tener en memoria una tabla global de archivos abiertos para el manejo de archivos de parte de los procesos.**

Falso, aparte cada proceso mantiene una tabla local de archivos abiertos. To perform any of the file allocation techniques described previously, it is necessary to know what blocks on the disk are available. Thus we need a disk allocation table in addition to a file allocation table.

(stallings 8th p548)

**3) Dado un SO que controla dos File Systems diferentes, el espacio libre de los mismos es manejado de igual forma para ambos.**

Falso, cada File System tiene su propia forma de manejar el espacio libre.

The operating system or file management system is responsible for allocating blocks to files. This raises two management issues. First, space on secondary storage must be allocated to files, and second, it is necessary to keep track of the space available for allocation. We will see that these two tasks are related; that is, the approach taken for file alloca- tion may influence the approach taken for free space management. (stallings 8th 543)

**4) Los archivos con organización indexada cuentan con un único índice formado por la combinación de todos los campos que maximiza la velocidad de búsqueda dentro del archivo.**

Falso. tiene multiples indices.

To achieve this flexibility, a structure is needed that employs multiple indexes, one for each type of field that may be the subject of a search. In the general indexed file, the concept of sequentiality and a single key are abandoned. Records are accessed only through their indexes.

(stallings 8th p532)

**5) La técnica ¨Bloqueo de Registro¨ en un File System no interactúa con el hardware de Memoria Virtual cuando esta es usada.**

Falso. The record-blocking technique may interact with the virtual memory hardware, if such is employed. In a virtual memory environment, it is desirable to make the page the basic unit of transfer. Pages are generally quite small, so that it is impractical to treat a page as a block for unspanned blocking. Accordingly, some systems combine multiple pages to create a larger block for file I/O purposes.(stallings 8th 543)

**6) Cuando se abre por primera vez un archivo, ¿el sistema de gestión de archivos carga en memoria el descriptor de archivos almacenado en disco, correspondiente a dicho archivo?**

Verdadero. many systems require that an open() system call be made before a file is first used. The operating system keeps a table, called the open-file table, containing information about all open files. When a file operation is requested, the file is specified via an index into this table, so no searching is required. When the file is no longer being actively used, it is closed by the process, and the operating system removes its entry from the open-file table. (silber 9th 507)

si, esta en memoria.

The OS may also maintain file tables . These tables provide information about the existence of files, their location on secondary memory, their current status, and other attributes. Much, if not all, of this information may be maintained and used by a file management system. (stallings 8th p127)

**7) Para archivos de tamaño grande es mejor, por cuestiones de desempeños, trabajar con estructuras tipo i-nodos.**

Verdadero. El uso de punteros y distintos niveles de indirección permiten manejar archivos grandes de manera eficiente.  
  
**8) La asignación de bloques en un FS elimina la fragmentación interna.**

Falso. la asignacion por bloques posee fragmentacion interna porque los bloques son porciones fijas de espacio por lo que puede ocurrir fragmentacion externa.

**9) En un filesystem la organización más compleja es la denominada pila.**

Falso. The least-complicated form of file organization may be termed the pile . Data are collected in the order in which they arrive. Each record consists of one burst of data. The purpose of the pile is simply to accumulate the mass of data and save it. Records may have different fields, or similar fields in different orders. (stalligns 8th 529)

**10) La estructura de archivo llamada Archivo Secuencial (Sequential File) soluciona una de las deficiencias de la estructura de Pila al permitir Acceso Aleatorio.**

Falso. For interactive applications that involve queries and/or updates of individual records, the sequential file provides poor performance. Access requires the sequential search of the file for a key match. (stallings 8th p530)

**11) En un filesystem basado en la técnica de Journaling, como por ejemplo NTFS o EXT3, es necesario hacer un bloqueo de la estructura de datos con el objetivo de garantizar la atomicidad de la operación de modificación.**

V. El procedimiento de journaling es básicamente el siguiente:

1. Se bloquean las estructuras de datos afectadas por la transacción para que ningún otro proceso pueda modificarlas mientras dura la transacción.

2. Se reserva un recurso para almacenar el journal. Por lo general suelen ser unos bloques de disco, de modo que si el sistema se para de forma abrupta (corte eléctrico, avería, fallo del sistema operativo...) el journal siga disponible una vez reiniciado el sistema.  
3. Se efectúan una a una las modificaciones en la estructura de datos. Para cada una: Se apunta en el journal como deshacer la modificación y se asegura de que esta información se escribe físicamente en el disco. Se realiza la modificación.  
4. Si en cualquier momento se quiere cancelar la transacción se deshacen los cambios uno a uno leyéndolos y borrándolos del journal.  
5. Si todo ha ido bien, se borra el journal y se desbloquean las estructuras de datos afectadas.

**12) Los archivos hasheados o directos, hacen uso de la capacidad del disco de acceder directamente a un bloque: no existe el concepto de orden secuencial.**

Verdadero. [Stallings, 5ª ed. Esp., Pg. 558].

El fichero de acceso directo, o *hash*, explota la capacidad encontrada en los discos para acceder directamente a cualquier bloque de una dirección conocida. Al igual que los ficheros secuenciales y secuenciales indexados, se requiere una clave para cada registro. Sin embargo, en este tipo de ficheros no existe el concepto de ordenación secuencial.

**13) La cantidad de memoria necesaria para almacenar el mapa de bits para la administración de bloques libres será mayor o menor para un mismo disco según varíe el tamaño de un bloque.**

Verdadero, con un bit se sabe si un bloque está libre o ocupado, por lo tanto se necesitará más espacio si la cantidad de bloques es mayor. Tamaño del disco en bytes / (cantidad de bloques \* 8)

Stallings. 572.

Este método utiliza un vector que está formado por un bit por cada bloque en el disco.

La cantidad de memoria (en bytes) requerida para un mapa de bits de bloques es tamaño del disco en bytes / (cantidad de bloques \* 8)

**15) La organización de archivos más sencilla que existe se denomina Pila. Una de las razones de su sencillez tiene relación con la posibilidad de acceder directamente a cualquiera de sus registros.**

Falso,Because there is no structure to the pile file, record access is by exhaustive search. That is, if we wish to find a record that contains a particular field with a particular value, it is necessary to examine each record in the pile until the desired record is found. (stallings 8th 529)

**17) Con una FAT de 10000 entradas, un archivo puede poseer más de 10010 bloques de disco.**

Falso, se puede direccionar hasta 10000 bloques.

Esto es porque la FAT guarda una entrada por cada bloque del disco y está indexada según el número de bloques. Cada entrada de directorio contiene el número de bloque del primer bloque del archivo. La entrada de la tabla indexada según ese número de bloque contiene el número de bloque del siguiente bloque de archivo. Esto continúa hasta el último bloque que tiene un fin de archivo. Los no utilizados se indican mediante un valor de tabla igual a 0

**19) Sistemas FAT necesitan el MBR, la lista enlazada de bloques del disco, la 2da lista, bit-vector, el superbloque y una estructura de directorios para funcionar.**

Falso. El superbloque está presente en los sistemas de archivos de UNIX. NO en FAT.

**Inodos**

**20) El tamaño de un file system tipo UNIX depende del tamaño del bloque, del tamaño del puntero y de la cantidad de punteros disponibles del i-nodo.**

Falso, no depende de la cantidad de punteros disponibles del i-nodo.

Tamaño máximo teórico del filesystem = http://latex.codecogs.com/png.latex?2%5E%7Bn%7D\* TB

n: tamaño de puntero en bits

TB: tamaño de bloque

**21) La lista de i-nodos de un sistema de archivos tipo UNIX crece a medida que se crean nuevos archivos.**

Falso, el espacio que ocupa la tabla de inodos es fijo.

**22) El tamaño máximo del file system de tipo UNIX está determinado únicamente por el tamaño del puntero y por la cantidad de punteros directos e indirectos que posee el I-NODO.**

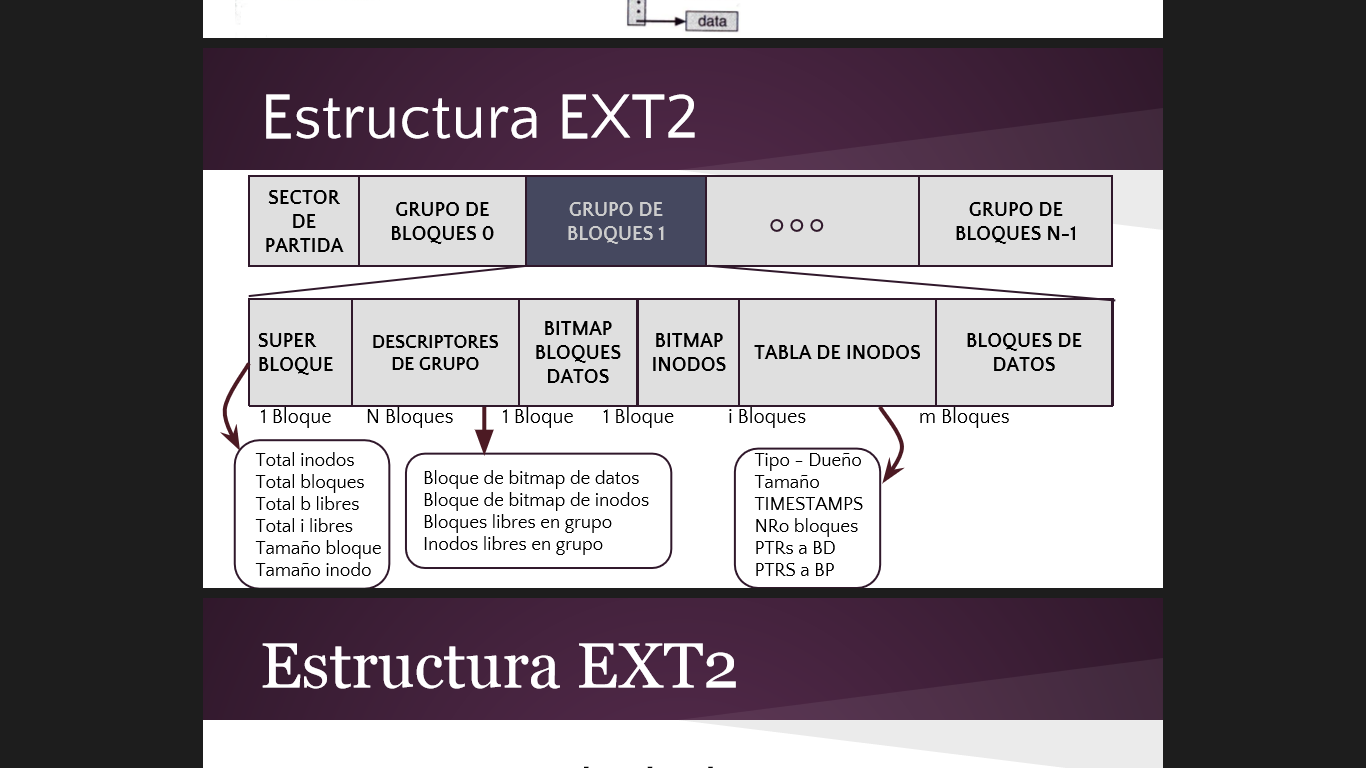
Falso, el tamaño del file system está determinado ademas por el tamaño del bloque.

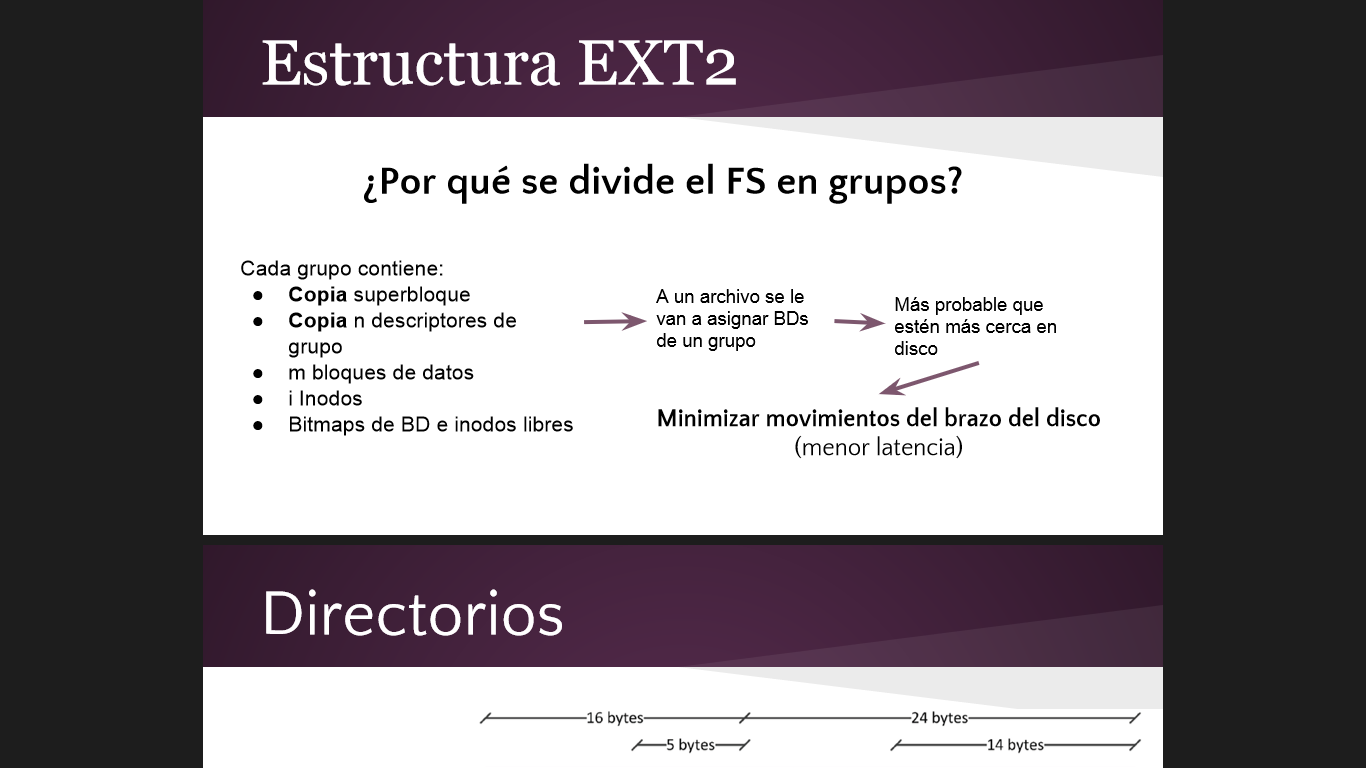
**23) Los archivos implementados bajo la estructura I-Nodos poseen fragmentación interna y externa, debido a esto deben desfragmentarse los discos esporádicamente, para obtener espacio libre contiguo.**

Falso. Los archivos implementados con I-Nodos no presentan fragmentación externa ya que no necesitan estrictamente espacio contiguo para su almacenamiento. **Y sufre poca fragmentación interna.**

**24) Como en todo FileSystem basado en i-nodos, en ext2, el superblock es único y no se repite.**

Falso, puede haber copias del superbloque en determinados Grupos de Bloques. (concepto de Sparse Superblock).





Because of the importance of the superblock and because damage to it (for example, from physical damage to the magnetic recording medium on the disk) could erase crucial data, backup copies are created automatically at intervals on the filesystem (e.g., at the beginning of each block group). For each mounted filesystem,[Linux](http://www.linfo.org/linuxdef.html) also maintains a copy of its superblock in [memory](http://www.linfo.org/memory.html).

**25) En archivos extremadamente largos es más rápido el acceso a los últimos bytes del archivo si el sistema se basa en inodos que si se basa en FAT.**

Verdadero, las indirecciones provistas por los I-Nodos permiten un acceso más rápido al bloque buscado. En fat como la asignación de espacio del archivo es enlazada se deben leer los bloques hasta el que contiene el byte.

Hard Links / Soft Links

**26) En filesystems de tipo UNIX, siempre es conveniente utilizar “hard links” por sobre “symbolic links”.**

Falso, depende el caso, una desventaja del hard link es que si quiero eliminar el archivo original para hacer espacio en el disco, lo tengo que eliminar desde todas sus ubicaciones, Los enlaces simbólicos se pueden hacer con ficheros y directorios mientras que los duros solo entre ficheros. Los enlaces simbólicos se pueden hacer entre distintos sistemas de ficheros, los duros no.

**27) Si tenemos 2 directorios que contienen enlaces duros a un mismo archivo entonces debemos tener, en los dos i-nodos, el contador de enlaces en 2(dos).**

Falso, en el inodo del archivo debería tener el contador de enlaces en 2. (verificado)

**28) Los enlaces duros permiten que un archivo de datos pueda ser accedido desde cualquier directorio del disco.**

Falso. Los enlaces duros permiten que un mismo archivo tenga más de una referencia, al mismo espacio físico de datos.

**29) Un archivo de tipo Soft-Link es un tipo de archivo especial que no posee la localización física del archivo**

Verdadero, contiene una referencia a la ruta del filesystem donde se encuentra el archivo.

**30) A diferencia de los hard links, los symbolic links pueden pertenecer a diferentes file system.**

Verdadero, Una ventaja del enlace simbólico frente a los enlaces duros es que es posible realizar enlaces simbólicos que apunten a objetos en sistemas de archivos que se hallan en otros dispositivos o particiones dentro del mismo dispositivo

**31) En el filesystem FAT, se podría simular un “hard link” similar al de UNIX si se creara una entrada de directorio apuntando al mismo número de cluster que otro archivo ya existente, sin que eso ocasione otros problemas con el manejo de atributos de los archivos.**

Falso, el problema sería que no se llevaría una cuenta de los enlaces.

**32) En el enlace simbólico de archivos UNIX, dos nombres apuntan al mismo i-nodo.**

Falso, el enlace simbólico apunta a un inodo distinto que tiene una referencia a la ruta del archivo al cual se quiere linkear.

**33) La razón por la que los i-nodos permiten que el File System posea una estructura de grafo acíclico es debido a la posibilidad de utilizar HARDLINK , cosa que no es posible con la estructura de un File System de tipo FAT.**

Falso la razon por la que se pueden utilizar hard links es porque el filesystem posee una estructura de grafo aciclico.

(silber 9th 524)  
**34) El número de enlaces de un archivo en UNIX se almacena en la entrada de directorio correspondiente.**

Falso, se almacena como atributo en el inodo.

**35) Sólo cuando se elimine el último symbolic link de un archivo, los bloques ocupados por el mismo serán considerados como disponibles.**

Falso, si se elimina un symbolic link no se elimina el archivo original, la frase sería válida si fuera hard links en vez de symbolic links.

**36) Cuando se abre un archivo se crea en memoria el FCB y los datos que este contiene provienen en parte del I-nodo o del directorio dependiendo del sistema operativo**

Verdadero. El FCB se copia en la tabla global de archivos existentes en la memoria. La tabla global de archivos abiertos almacena los inodos y otra info para archivos y directorios

**37) Cuando se abre un archivo siempre el header se copia en memoria para saber dónde comienza el archivo y cuando el proceso lo cierra esta estructura se elimina.**

falseo, si el archivo sigue abierto por otro proceso no se elimina hasta que el contador sea 0. The operating system keeps a table, called the open-file table, containing information about all open files. When a file operation is requested, the file is specified via an index into this table, so no searching is required. When the file is no longer being actively used, it is closed by the process, and the operating system removes its entry from the open-file table. (silber 9th 507)