# Unidad 3:
# Gestión de Archivos

## Tema 6, Implementación del Sistema de Archivos:

6.1 Estructura del Sistema de Archivos.
6.2 Métodos de Asignación: Continua, enlazada, indexada.
6.3 Fiabilidad del sistema de archivos.
6.4 Gestión de archivos en UNIX.
6.5 Gestión de archivos en Windows 2000: NTFS.

# 6.1 Estructura del Sistema de Archivos.

- Introducción:
  - El **sistema de archivos reside permanentemente en almacenamiento secundario**, cuyo único requisito es que debe poder contener una gran cantidad de datos de forma permanente.
  - **Gestión de almacenamiento secundario:**
    - En memoria secundaria un *archivo consta de un cjto de bloques*.
    - El SO o el sistema de gestión de archivos es responsable de la asignación de los bloques a los archivos.
    - **Para mejorar la eficiencia E/S:** Las transferencias entre la memoria y el disco se efectúan en unidades de “bloques” (uno o más sectores, de tamaño en torno a 512 bytes).

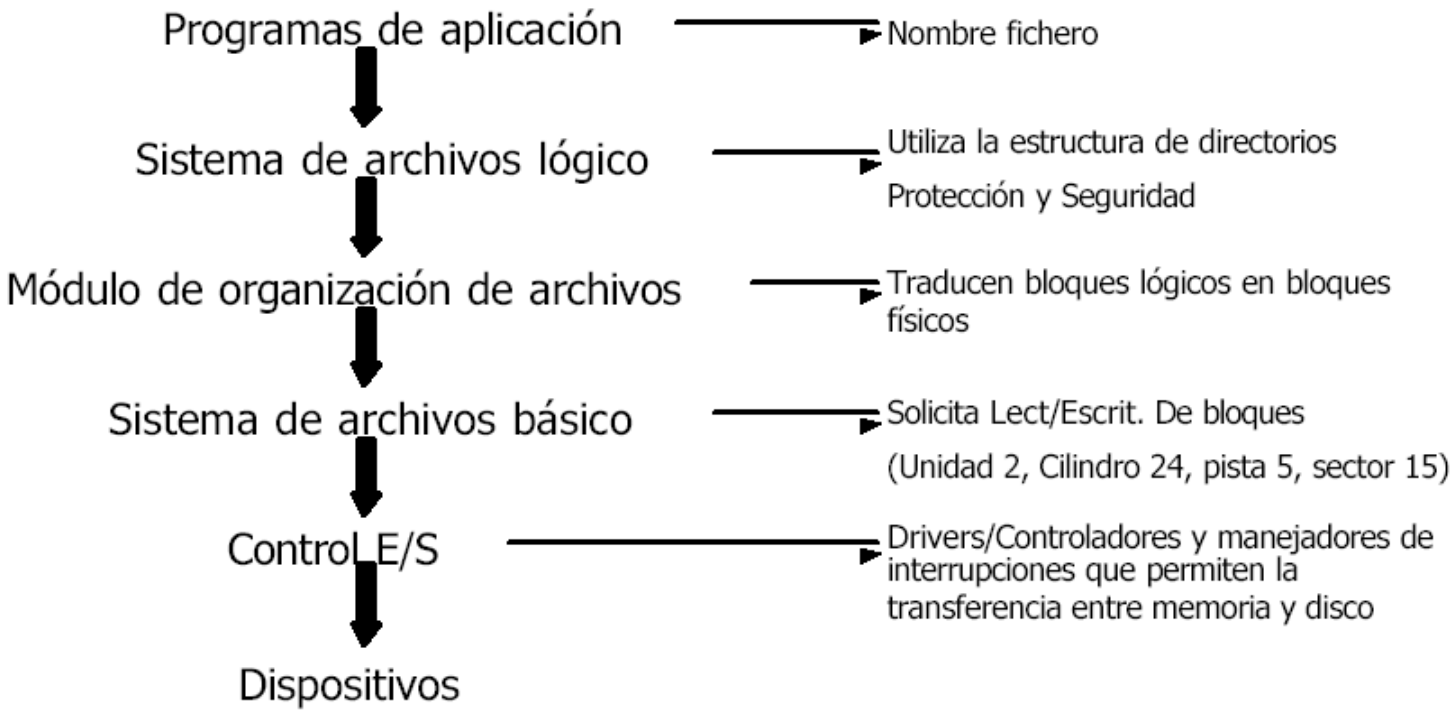
# 6.1 Estructura del Sistema de Archivos.

- **Discos:**
  - Dos características importantes los convierten en un medio cómodo para almacenar muchos archivos,
  - Se pueden reescribir en el mismo lugar.
  - **Podemos acceder directamente a cualquier bloque de información del disco.**

- Organización del sistema de archivos:
  - Un sistema de archivos presenta **dos problemas de diseño** muy distintos:
    - Definir qué aspecto debe presentar el sistema de archivos a los usuarios (atributos, operaciones, estructura de directorios, etc.).
    - Definir los algoritmos y estructuras de datos que permiten mapear el sistema de ficheros lógico sobre los equipos físicos.

# 6.1 Estructura del Sistema de Archivos.

- Organización del sistema de archivos:
  - Ejemplo de diseño por capas:



# 6.2 Métodos de asignación.

- Cuestiones surgidas:
  - 1. Cuando creamos un archivo nuevo, *¿se asigna de una sola vez el máximo espacio que necesita?.*
  - 2. Espacio que se asigna: Sección ⇒ *¿qué tamaño de sección (entre un bloque y todo un archivo) debería usarse para asignar archivos?.*
  - 3. *¿Tipo de tabla o estructura de datos que necesitamos para guardar constancia de las secciones asignadas a un archivo?* ⇒ Tabla de asignación de archivos (FAT).
- Tres métodos de asignación de espacio en disco:
  - Asignación contigua.
  - Asignación enlazada (encadenada).
  - Asignación indexada (enlazada de bloques índice).

# 6.2 Métodos de asignación.

- Asignación Contigua:
  - **Cada fichero ocupa** bloques con direcciones lógicas del dispositivo contiguas ⇒ **bloques contiguos en el disco**.
  - Número de **búsquedas y tiempo de búsqueda mínimos** para acceder a archivos contiguos.
  - Asignación contigua definida por:
    - *Dirección en disco del bloque inicial.*
    - *Longitud del área asignada al archivo (nº de bloques).*
  - Permite el acceso secuencial y directo.

# 6.2 Métodos de asignación.

- Asignación Contigua:
  - **Problemas:**
    - **Asignación dinámica de almacenamiento:**
      - Estrategias de primer ajuste y mejor ajuste.
      - Selección de un hueco libre del conjunto de los huecos disponibles.
    - **Inserción de datos difícil.**
    - **Fragmentación externa** (por los algoritmos utilizados):
      - Al borrar un fichero queda un hueco que puede no ser utilizado completamente por otro fichero.
      - Solución: compactación (creación de un único hueco suficientemente grande).

# 6.2 Métodos de asignación.

- Asignación Contigua:
  - **Resumen:**
    - Se necesita la dirección del primer bloque.
    - Todo el archivo se puede leer de una sola vez (buen rendimiento)
    - El método no es realizable, a menos que se conozca el tamaño máximo del archivo, en el momento de su creación.
    - Produce bastante fragmentación externa.
    - Fácil acceso directo a bloques.
    - Importante la gestión de espacio libre.

# 6.2 Métodos de asignación.

- Asignación Enlazada:
  - **Archivo como lista enlazada de bloques de disco.**
  - **Los bloques pueden estar dispersos** por todo el disco.
  - La entrada al directorio contiene un **puntero al primer y al último bloque del archivo.**
  - **Salto entre bloques** $\Rightarrow$ cada bloque contiene un puntero al siguiente bloque.
  - **Se solucionan los problemas de la asignación contigua:**
    - No hay fragmentación externa ya que no hay necesidad de declarar el tamaño de un archivo en el momento de crearlo.
    - Un archivo puede continuar creciendo siempre que haya bloques libres.
    - No es necesario compactar el disco.

# 6.2 Métodos de asignación.

- Asignación Enlazada:
  - **Problemas:**
    - Sólo es **eficiente para archivos de acceso secuencial**.
    - **Espacio ocupado por los punteros** (se utiliza espacio para guardar punteros perdiéndolo para guardar información) ⇒ nos queda un poco menos de espacio en cada bloque.
  - **Solución:**
    - Juntar bloques en cúmulos o **"clusters"** y asignar los clusters en vez de los bloques ⇒ **se pierde menos espacio.**

# 6.2 Métodos de asignación.

- Asignación Enlazada:
  - **Ventajas del método:**
    - La correspondencia entre bloques sigue siendo sencilla:
      - Mejora el rendimiento del disco.
      - Reduce el espacio necesario para la asignación de bloques y la administración de la lista de espacio libre.
    - Los clusters mejoran el tiempo de acceso a disco ⇒ se usan en casi todos los sistemas operativos.
  - **Inconveniente:**
    - Crece la fragmentación interna, pues se desperdicia más espacio cuando un cluster está parcialmente lleno que cuando un bloque está parcialmente lleno.

# 6.2 Métodos de asignación.

- Asignación Enlazada:
  - **FAT (File Allocation Table):**
    - **Tabla de Asignación de Ficheros.**
    - Al principio de cada partición se reserva una sección del disco para guardar una tabla en ella.
    - Tiene una entrada para cada bloque de disco.
    - Está indizada por nº de bloque (se usa de manera similar a una lista enlazada).
    - La entrada de directorio contiene el nº de bloque del primer bloque del archivo.
    - La entrada de la tabla indizada por ese nº de bloque contiene el nº del siguiente bloque del archivo.
    - La cadena continúa hasta el último bloque, que tiene un valor especial de fin de archivo como entrada de la tabla.
    - Los bloques desocupados se indican con valor cero en la tabla.

# 6.2 Métodos de asignación.

- Asignación Enlazada:
  - **FAT (Resumen):**
    - Todos los bloques están organizados en una lista enlazada. Hay dos posibilidades:
      - Los bloques están enlazados entre ellos.
      - Se guarda la lista en memoria (MS-DOS).
    - Problema:
      - Para leer o escribir en el bloque “n”, hay que recorrer los “n-1” bloques precedentes.
    - MS-DOS añade una tabla de localización de archivos duplicada. El sistema de archivos depende del tamaño de cada entrada en la FAT.

# 6.2 Métodos de asignación.

- Asignación Indexada:
  - **Indizada o enlazada de bloques índice:**
    - **Asignación enlazada resuelve:**
      - Fragmentación externa.
      - Declaración anticipada del tamaño de los archivos.
    - **Problema:**
      - Si no se usa FAT no se puede implementar un acceso directo eficiente (punteros dispersos junto con los bloques).
    - **Solución:**
      - Reunir todos los punteros en el mismo lugar ⇒ bloque índice (asignación indexada).

# 6.2 Métodos de asignación.

- Asignación Indexada:
  - *Cada archivo tiene su propio bloque índice.*
  - La i-ésima entrada del bloque índice apunta al i-ésimo bloque del archivo.
  - El directorio contiene la dirección del bloque índice.
  - Soporta el acceso directo sin sufrir fragmentación externa.
  - **Inconveniente:** Desperdicia espacio (gasto de los punteros del bloque índice).
  - *Conviene que el bloque índice sea lo más pequeño posible* (normalmente ocupa un bloque de disco).
  - Sin embargo, si es demasiado pequeño, no podrá contener suficientes punteros como requiere un archivo grande.

# 6.2 Métodos de asignación.

- Asignación Indexada:
  - **Soluciones:**
    - **Esquema enlazado:**
      - Para manejar archivos grandes podríamos enlazar varios bloques índice.
    - **Indice multinivel:**
      - Usar un bloque índice de primer nivel que apunte a un conjunto de bloques índice de segundo nivel.
    - **Esquema combinado:**
      - Guardamos los primeros punteros del bloque índice en el bloque índice del archivo. Los punteros apuntarán a bloques directos, bloques indirectos simples, bloques indirectos dobles, ...

# 6.2 Métodos de asignación.

- Comparación entre asignaciones (I):
  - **Asignación contigua:**
    - **Ventajas:**
      - Soporta acceso secuencial y directo.
      - Todo el espacio se utiliza para almacenar datos.
    - **Desventajas:**
      - Encontrar espacio para la creación de un fichero.
      - Fragmentación externa.
      - Declaración por anticipado del tamaño del archivo.
  - **Asignación enlazada:**
    - **Ventajas:**
      - No se produce fragmentación externa.
      - No es necesario declarar por anticipado el tamaño del archivo.

# 6.2 Métodos de asignación.

- Comparación entre asignaciones (II):
  - **Asignación enlazada:**
    - **Desventajas:**
      - Eficiente sólo para archivos de acceso secuencial.
      - Espacio ocupado por los punteros.
      - Confiabilidad: Pérdida de datos provocada por pérdida de punteros.
  - **Asignación indexada:**
    - **Ventajas:**
      - No se produce fragmentación externa.
      - No es necesario declarar por anticipado el tamaño del archivo.
      - Soporta acceso secuencial y directo.
    - **Desventajas:**
      - Mayor pérdida de espacio (bloque/s índice).
      - Confiabilidad: Pérdida de datos provocada por pérdida del bloque/s índice.

# 6.3 Fiabilidad del sistema de archivos.

- Supongamos el siguiente esquema:
  - 1. El usuario A solicita una asignación para añadir datos a un archivo existente.
  - 2. Se atiende la petición y se actualizan en memoria principal las tablas de asignación de discos y archivos.
  - 3. El sistema se para y a continuación se reinicia.
  - 4. El usuario B solicita una asignación y se le otorga un espacio en el disco que se solapa con la última asignación hecha al usuario A.
  - 5. El usuario A accede a la sección solapada mediante una referencia que está almacenada en el archivo A.

# 6.3 Fiabilidad del sistema de archivos.

- **Problema:**
  - **Solapamiento de secciones por doble asignación.** Debido a que el Sistema busca la máxima eficiencia y mantiene copias de la tabla de asignación de disco y de la tabla de asignación de archivos en la memoria principal.

- **Solución (Cuando se solicite una asignación):**
  - 1. Bloquear la tabla de asignación de disco.
  - 2. Buscar espacio disponible en la tabla de asignación de disco.
  - 3. Asignar el espacio, actualizar la tabla de asignación de disco y actualizar el disco.
  - 4. Actualizar la FAT y actualizar el disco.
  - 5. Desbloquear la tabla de asignación de disco.

# 6.4 Gestión de archivos en UNIX.

- **Introducción:**
  - Unix contempla todos los **archivos como flujos de bytes**.
  - Unix se ocupa de la **estructura física de los archivos.**

- **Se distinguen cuatro tipos de archivos en UNIX:**
  - **Ordinarios (Regulares):** Texto y binarios, contienen información introducida por el usuario, un programa de aplicación o un programa de utilidad del sistema.
  - **Directorios:** Contienen lista de nombres de archivos y punteros a nodos-i. Tienen estructura jerarquizada. Son como los ordinarios pero con privilegios especiales de protección (programas de usuario $\Rightarrow$ sólo lectura; sistema de archivos $\Rightarrow$ lectura y escritura).
  - **Especiales (de dispositivo):** para acceso a periféricos. Cada dispositivo de E/S está asociado a un archivo especial.
  - **Pipes (tuberías).**

# 6.4 Gestión de archivos en UNIX.

- **Características del Sistema de Archivos en UNIX:**
  - Estructura jerarquizada.
  - Tratamiento consistente de los datos.
  - Permite crear y borrar archivos.
  - Permite un crecimiento dinámico de los archivos.
  - Permite proteger los datos de los archivos.
  - Mantiene independencia de los dispositivos.

# 6.4 Gestión de archivos en UNIX.

- **Estructura del Sistema de Ficheros (nodos-i):**
  - Existe una **tabla de nodos-i para acceder a los archivos**.
  - Un *nodo-i es una estructura de control que contiene la información clave de un archivo necesaria para el SO.*
  - Cada archivo es controlado por un solo nodo-i.
  - **Tablas de control de accesos a los archivos:**
    - Tabla de ficheros abiertos.
    - Tabla de descriptores de ficheros.
  - **Tabla de nodos-i:**
    - Cuando se abre el fichero se debe cargar en memoria (si es que no está ya) su nodo-i (descriptor del fichero) en una tabla de nodos-i en memoria.
    - La tabla de nodos-i en memoria funciona de forma semejante al buffer caché (hash table)

# 6.4 Gestión de archivos en UNIX.

- El buffer caché (la caché del disco):
  - Mecanismo que utiliza Unix para **optimizar el acceso a los dispositivos de bloques.**
  - Se mantiene una zona de memoria como buffer de datos.
  - Cuando un proceso pide un acceso se trae el bloque entero pasando antes por la caché.
  - Almacena los n bloques de disco últimamente referenciados.

  - **La estructura está formada por:**
    - Lista de cabeceras y zona de datos.
  - **Las cabeceras contienen:**
    - Nº de dispositivo, nº de bloque, estado, punteros a zona de datos.
  - **Funcionamiento de la lista de cabeceras como una tabla hash:**
    - Localización de bloque = f(nº dispositivo, nº bloque).
    - Cada entrada: lista doblemente enlazada con los bloque que responden a esa entrada (aplicando la función).
  - Además, existe una **lista doblemente enlazada con los bloques libres.**

# 6.4 Gestión de archivos en UNIX.

- **Tabla de Ficheros:**
  - Tabla única con una entrada asignada cada vez que se abre un fichero.
  - Contiene el tipo de apertura y un puntero al nodo-i del fichero abierto.

- **Tabla de descriptores de ficheros:**
  - Una tabla por proceso, que contiene una entrada por cada vez que se hace una apertura. Apunta a una entrada en la tabla de ficheros.
  - Cuando se abre un fichero al usuario se le devuelve un puntero a este descriptor.
  - **Nota:** La creación de un fichero (ordinario) supone:
    - Crear una entrada en la tabla de descriptores de ficheros.
    - Crear una entrada en la tabla de ficheros.
    - Cargar (si no está ya) el nodo-i del fichero en la tabla de nodos-i.

# 6.4 Gestión de archivos en UNIX.

- **Lista de nodos-i libres en memoria:**
  - De todos los nodos-i libres guardados en disco **se guardan los n primeros en una lista en memoria.**
  - Cuando se crea un nuevo fichero se toma uno de la lista.
  - Cuando se borra un fichero se guarda en la lista si tiene < nº que el más alto de la lista.

- **Lista de bloques libres en memoria:**
  - En disco el nº de bloques libres se guarda a su vez en bloques libres.
  - En memoria se crea una lista con los bloques libres que hay en el primer bloque de la lista de bloques libres en disco.
  - Cuando se pide un bloque para un fichero se toma uno de la lista..
  - Cuando se libera un bloque se mete en la lista.

# 6.4 Gestión de archivos en UNIX.

- **Ejemplo, tabla de implantación con 13 entradas:**
  - Las 10 primeras (0-9) **apuntan a** los 10 primeros **bloques de datos del fichero.**
  - La entrada 10 **apunta a una tabla que contiene direcciones de bloques de datos (1 nivel).**
  - La entrada 11 **apunta a una tabla que apunta a tablas de implantación que apuntan a bloques de datos (2 niveles).**
  - La entrada 12 apunta a una tabla que **apunta a tablas de implantación que apuntan a tablas de implantación que apuntan a tablas de datos (3 niveles).**

# 6.4 Gestión de archivos en UNIX.

- **Ventajas de este esquema:**
  - 1. Los nodos-i son de tamaño fijo y relativamente pequeños, por lo que no pueden guardarse en la memoria principal durante periodos largos.
  - 2. Se puede acceder a los archivos más pequeños de modo directo o indirecto, reduciendo así el procesamiento y el tiempo de acceso a disco.
  - 3. El tamaño máximo teórico de un archivo es suficientemente grande como para satisfacer a casi todas las aplicaciones.

# 6.5 Gestión de archivos en Windows 2000.

- Sistema de archivos en W2K:
  - **W2k soporta varios sistemas de archivos:**
    - **FAT** (FAT16, VFAT, FAT32): que se ejecutaba sobre w95, MS-DOS, OS/2.
    - **NTFS:** W2k, WNT y XP, para incluir aplicaciones de alto nivel.
  - **Aplicaciones de alto nivel incluidas en NTFS:**
    - **Aplicaciones cliente/ servidor** tales como servidores de archivos, servidores de procesamiento y servidores de DB.
    - **Ingeniería de recursos intensivos y aplicaciones científicas.**
    - Aplicaciones de **redes para grandes sistemas corporativos.**

# 6.5 Gestión de archivos en Windows 2000.

- Características de NTFS:
  - **Recuperabilidad:**
    - *Capacidad de recuperación de caídas del sistema y de fallos de disco.*
    - *Reconstruye los volúmenes de disco* y los devuelve a un estado consistente.
    - *Recupera cada transacción en ejecución cuando surgió el fallo* (sistema de almacenamiento redundante para los datos críticos del stma de archivos).
  - **Seguridad:**
    - Archivo abierto que se implementa como un objeto archivo con un descriptor que define sus atributos de seguridad.
  - **Soporta eficientemente discos grandes y archivos grandes.**
  - **Podemos definir múltiples series de datos para un sólo archivo.**
  - **Capacidad de indexación general:**
    - El conjunto de descriptores de un archivo se organiza como una DB relacional, así los archivos se pueden indexar por cualquier atributo.

# 6.5 Gestión de archivos en Windows 2000.

- Estructura de archivos y volúmenes NTFS (I):
  - **Conceptos de almacenamiento en disco:**
    - **Sector:** Unidad de almacen. físico más pequeña sobre el disco.
    - **Cluster:** Uno o más sectores contiguos.
    - **Volumen:** Uno o más clusters que el sistema de archivos utiliza para asignar el espacio. Un volumen puede ser una parte de un disco, todo el disco o incluso varios discos (tam. máximo $2^{64}$ B).
    - **Nota:** Tamaño máximo de archivo soportado por NTFS: $2^{32}$ clusters $\approx 2^{48}$ B.
  - **Disposición de un volumen NTFS:**
    - *Cada elemento es un archivo y cada archivo consta de un conjunto de atributos.*
    - **Cuatro regiones:**
      - **1. Partición del sector de arranque:** Contiene información acerca de la disposición del volumen y de las estructuras del sistema de archivos así como la información y el código de arranque.

# 6.5 Gestión de archivos en Windows 2000.

- Estructura de archivos y volúmenes NTFS (II):
  - **Disposición de un volumen NTFS:**
    - **2. MFT (Master File Table) Tabla maestra de archivos:**
      - Contiene información acerca de todos los archivos y directorios del volumen.
      - Se organiza como un conjunto de archivos con una estructura de BD relacional.
    - **3. Archivos del Sistema:**
      - MFT2: Espejo del comienzo del MFT (duplica la MFT previniendo fallos).
      - Archivos de registro: Lista los pasos de las transacciones para la recuperabilidad.
      - Mapa de bits de agrupamientos: Representación del volumen, mostrando los clusters en uso.
      - Tabla de definición de atributos: Define los tipos de atributos de ese volumen.
    - **4. Área de archivos.**