SISTEMAS DE FICHEROS

1. ¿QUÉ ES UN SISTEMA DE FICHEROS?

Es el **software del sistema operativo** que gestiona cómo se almacenan, organizan y acceden los archivos en el disco.

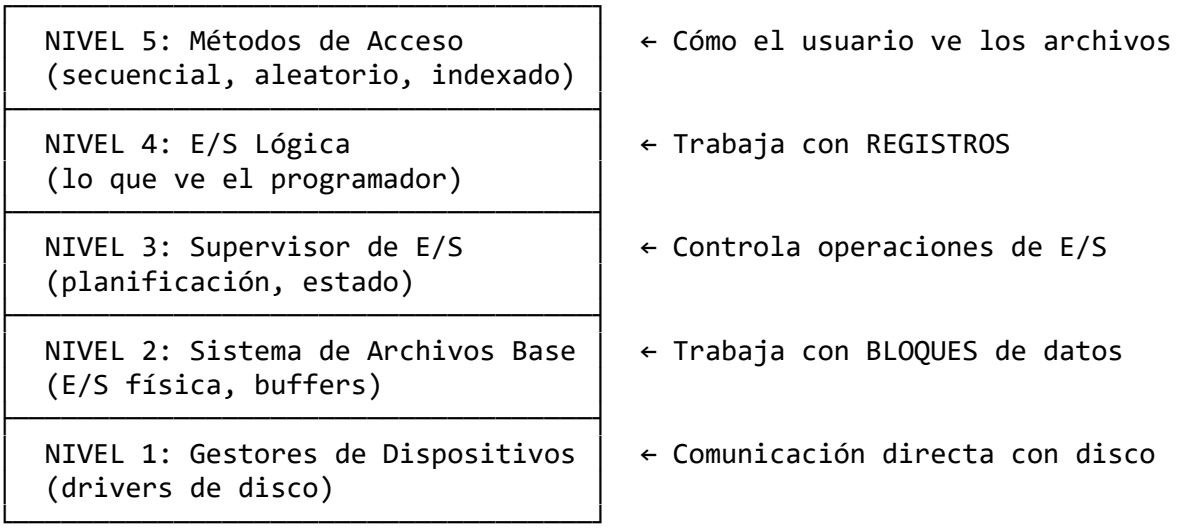
Jerarquía de datos:

- **Campo** → Dato individual (ej: nombre, DNI)
- **Registro** → Conjunto de campos relacionados (ej: ficha de empleado)
- **Archivo** → Conjunto de registros similares
- **Base de datos** → Conjunto de archivos relacionados

Objetivos principales:

1. Permitir crear, borrar y modificar archivos
2. Controlar quién accede a qué archivos
3. Optimizar el rendimiento (velocidad de acceso)
4. Evitar pérdida o destrucción de datos
5. Hacer copias de seguridad (backups)

2. ARQUITECTURA (Niveles del Sistema de Ficheros)



Regla clave: Los usuarios trabajan con **registros**, pero el disco trabaja con **bloques**.

Relación con el Kernel

Nivel	Componente	¿Kernel?
5	Métodos de acceso	No (API para aplicaciones)
4	E/S lógica	No (bibliotecas del sistema)
3	**Supervisor de E/S**	**Sí**
2	Sistema de archivos base (E/S física)	Sí
1	Gestores de dispositivos	Sí

¿Qué hace exactamente el Supervisor de E/S en el kernel?

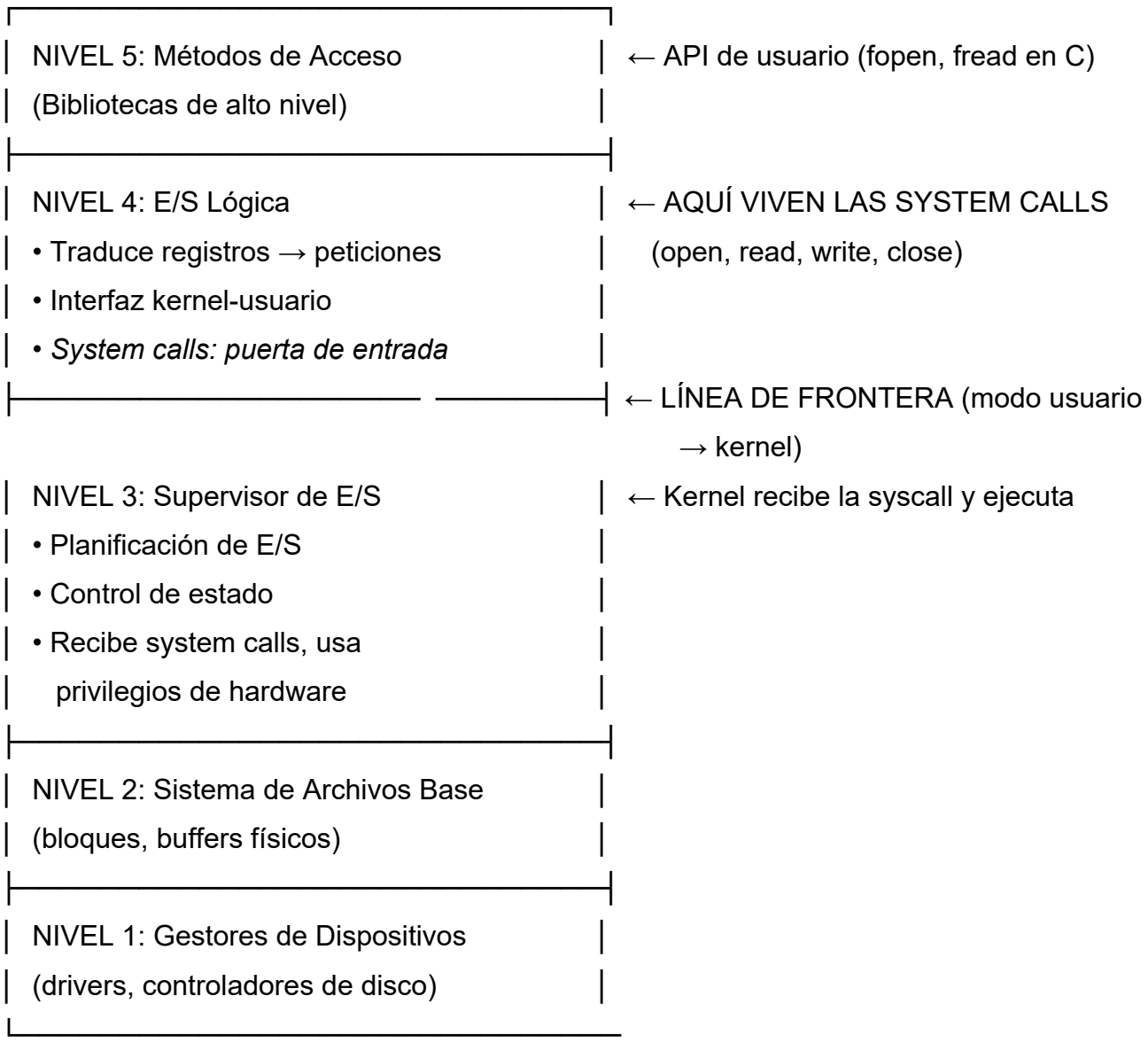
- Planifica las operaciones de lectura/escritura en disco
- Mantiene estructuras de control con el estado de cada archivo abierto
- Gestiona buffers y caché de disco
- Coordina entre el sistema de archivos base (bloques) y la E/S lógica (registros)
- Maneja interrupciones del hardware de disco

¿Por qué debe estar en el kernel?

Porque necesita privilegios de hardware:

- Acceso directo a puertos de E/S
- Manejo de interrupciones
- Control de DMA (Acceso Directo a Memoria)
- Protección de estructuras críticas (si falla, se corrompe todo el sistema de archivos)

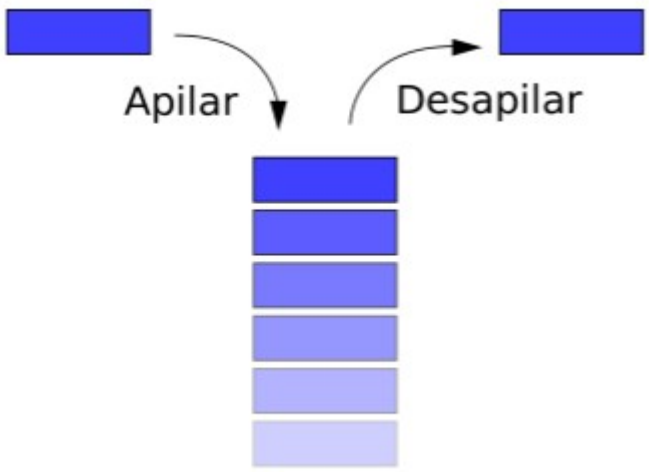
Y los System Calls?



3. ORGANIZACIÓN DE ARCHIVOS

3.1 Archivo de Pila (Heap)

Datos amontonados sin orden, como echar papeles a una caja.



- **Características:** Registros de longitud variable, campos desordenados, sin estructura fija
- **Acceso:** Búsqueda exhaustiva (lento)
- **Uso:** Acumular datos temporalmente antes de procesarlos
- **Ventaja:** Aprovecha bien el espacio con datos de tamaño variable
- **Desventaja:** Muy lento para buscar registros específicos

Ejemplo: Log de un servidor web

[TIMESTAMP] Error 404: /pagina-no-existe

[TIMESTAMP] Usuario Juan inició sesión

[TIMESTAMP] Error 500: Base de datos caída

[TIMESTAMP] Usuario María compró producto #123

[TIMESTAMP] Warning: Memoria al 90%

Características que cumple:

- Cada línea tiene campos diferentes (a veces error, a veces compra, a veces warning)
- Llegan en orden cronológico (cuando ocurrieron)
- Para encontrar "todos los errores de Juan" hay que leer TODO el archivo
- Los campos son autodescriptivos (tienen etiqueta + valor) , ejemplo JSON:
  - {
  - "nombre": "Ana",
  - "apellido": "García",
  - "DNI": "12345678",
  - "sueldo": "2500"
  - }

Otros ejemplos:

Situación	**Por qué es pila**
Historial de chat de WhatsApp	Mensajes llegan cuando llegan, sin ordenar por tema
Registro de temperatura de sensores IoT	Datos crudos que luego se procesarán
Bandeja de entrada de correo sin clasificar	Emails de todos los temas mezclados

3.2 Archivo Secuencial

Registros del mismo tamaño, ordenados por una clave, como una lista de asistencia.

- **Características:** Registros de longitud fija, mismo número de campos, ordenados por una clave
- **Acceso:** Secuencial (uno tras otro)
- **Uso:** Procesamiento por lotes (batch)
- **Ventaja:** Simple, puede guardarse en cinta o disco
- **Desventaja:** Lento para acceso aleatorio a un registro específico
- **Actualización:** Se usa un “archivo de transacciones” para añadir registros nuevos

Ejemplo: Lista de estudiantes ordenada por DNI

```
DNI        | Nombre          | Apellido      | Fecha_Nac  | Carrera
-----------|-----------------|---------------|------------|----------
10000001   | Ana             | García        | 15/03/2002 | Informática
10000002   | Bruno           | López         | 22/07/2001 | Matemáticas
10000003   | Carla           | Martínez      | 10/11/2002 | Física
10000004   | Diego           | Rodríguez     | 05/01/2001 | Informática
...
```

Características que cumple:

- Todos los registros tienen los mismos campos (DNI, Nombre, Apellido, etc.)
- Misma longitud cada registro (ej: 100 bytes exactos)
- Ordenados por clave (DNI, de menor a mayor)
- Para encontrar a "Carta" sin saber su DNI, hay que leer secuencialmente desde el principio

Analogía física:

Una cinta de casete o una cinta magnética de backup: para llegar a la canción 5, pasas por la 1, 2, 3 y 4.

3.3 Archivo Secuencial Indexado

Secuencial + índice para saltar + archivo de desbordamiento para insertar.

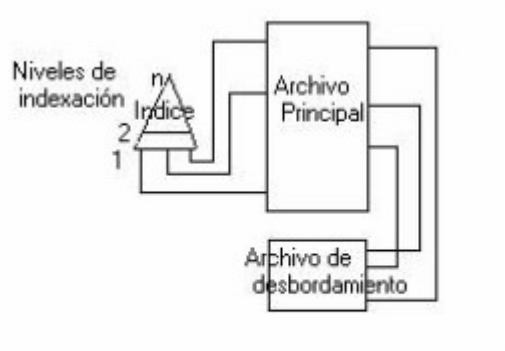


Fig. 5.4.4 Archivo secuancial indexado.

- **Características:** Secuencial + Índice + Archivo de desbordamiento
- **Acceso:** Secuencial O aleatorio (gracias al índice)
- **Funcionamiento:**
  - Índice apunta a bloques del archivo principal
  - Nuevos registros van al archivo de desbordamiento
  - Punteros conectan registros en el orden lógico
- **Ventaja:** Buen equilibrio entre secuencial y aleatorio

Ejemplo : Catálogo de una biblioteca

ARCHIVO PRINCIPAL (ordenado por ISBN):

```
ISBN        | Título                    | Autor           | Ubicación
------------|---------------------------|-----------------|----------
97800001    | Don Quijote               | Cervantes       | Estante A-1
97800005    | Cien años de soledad      | García Márquez  | Estante B-3
97800010    | El principito             | Saint-Exupéry   | Estante A-2
...
```

ÍNDICE (solo algunos ISBN clave, como "páginas de guía"):

```
ISBN_Clave  | Puntero al bloque principal
------------|---------------------------
97800001    → Bloque 1 (contiene 97800001 a 97800004)
97800005    → Bloque 2 (contiene 97800005 a 97800009)
97800010    → Bloque 3 (contiene 97800010 a 97800014)
...
```

ARCHIVO DE DESBORDAMIENTO (nuevos libros insertados):

```
ISBN        | Título                    | Autor           | Ubicación  | Puntero
------------|---------------------------|-----------------|------------|--------
97800003    | Rayuela                   | Cortázar        | Estante C-1| → null
97800007    | Ficciones                 | Borges          | Estante B-1| → null
```

Punteros de enlace:

```
Registro 97800001 en principal → apunta a 97800003 en desbordamiento
Registro 97800003 en desbordamiento → apunta a 97800005 en principal
```

¿Cómo funciona?

1. Quiero el libro con ISBN 97800003
2. Busco en el índice: 97800001 es el mayor que precede a 97800003
3. El índice me manda al Bloque 1 del principal
4. Leo secuencialmente: 97800001, 97800002... llego a 97800001 que tiene puntero
5. Sigo el puntero al desbordamiento → ¡ahí está 97800003!

Ver al final del documento que es un ***puntero de desbordamiento. (este es un ejemplo)***

3.4 Archivo Indexado

Los registros están "tirados" por el disco, pero los índices saben dónde están.

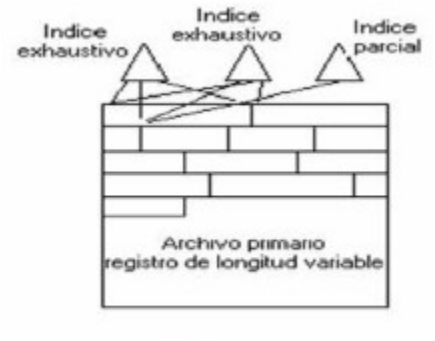


Fig. 5.4.5 Archivo indexado.

- **Características:** Acceso SOLO a través de índices, registros pueden estar en cualquier lugar
- **Tipos de índices:**
  - **Exhaustivo:** Una entrada por cada registro
  - **Parcial:** Solo para registros con ciertos campos
- **Ventaja:** Muy flexible, soporta registros de longitud variable
- **Desventaja:** Requiere actualizar todos los índices al insertar

Ejemplo: Agenda de contactos de tu teléfono

Los datos físicos (donde realmente están en memoria):

```
Posición_Memoria | Contacto
-----------------|-------------------------
0xA1F2           | {Nombre: "Ana", Tel: 555-0101}
0xB3C4           | {Nombre: "Bruno", Tel: 555-0202}
0xD5E6           | {Nombre: "Carlos", Tel: 555-0303}
0xE7F8           | {Nombre: "Ana", Tel: 555-0404}  ← ¡Otra Ana!
...
```

ÍNDICE POR NOMBRE (exhaustivo):

```
Nombre  | Puntero a posición
--------|------------------
Ana     → 0xA1F2
Ana     → 0xE7F8      ← ¡Dos entradas para "Ana"!
Bruno   → 0xB3C4
Carlos  → 0xD5E6
```

ÍNDICE POR TELÉFONO (parcial, solo algunos):

```
Teléfono   | Puntero a posición
-----------|-------------------
555-0101   → 0xA1F2
555-0202   → 0xB3C4
```

¿Cómo funciona?

- Busco "Ana" → índice me da dos direcciones → voy directo a esas posiciones
- No necesito leer a Bruno ni Carlos
- Cada contacto puede tener campos diferentes (uno tiene email, otro no)

Situación	**Índice 1**	**Índice 2**
Catálogo de productos de Amazon	Por nombre	Por categoría
Sistema de archivos de tu PC	Por nombre de archivo	Por fecha de modificación
Base de datos de una universidad	Por DNI del alumno	Por código de carrera

3.5 Archivo Directo (Hash)

Una fórmula matemática calcula dónde está el dato, como un código de barras que te dice en qué estantería está.

- **Características:** Usa una función hash sobre la clave para calcular la dirección del bloque
- **Acceso:** Directo y muy rápido
- **Uso:** Bases de datos, sistemas que necesitan acceso inmediato
- **Requisito:** Disco con capacidad de acceso directo a cualquier bloque

Ejemplo real: Sistema de cajas de un supermercado

Regla del supermercado:

"Tu caja = últimos 2 dígitos de tu DNI mod 10"

```
DNI del cliente    | Cálculo              | Caja asignada
-------------------|----------------------|---------------
45.123.456-**78**  | 78 mod 10 = 8        → Caja 8
23.987.654-**32**  | 32 mod 10 = 2        → Caja 2
12.456.789-**95**  | 95 mod 10 = 5        → Caja 5
45.123.456-**78**  | 78 mod 10 = 8        → Caja 8 (mismo cliente, misma caja)
```

En disco sería:

```
DNI del empleado   | Función hash          | Bloque del disco
-------------------|----------------------|-----------------
45.123.456-78      | hash(78) = Bloque 142 → Bloque 142
23.987.654-32      | hash(32) = Bloque 89  → Bloque 89
```

Cómo funciona?

- Quiero los datos del empleado con DNI 45.123.456-78
- Aplico la función hash a la clave → me da Bloque 142
- Voy directamente al bloque 142 del disco (sin índices, sin búsqueda)
- ¡Ahí están los datos!

4. DIRECTORIOS

4.1 ¿Qué es un Directorio?

Es un archivo especial del sistema operativo que contiene información sobre otros archivos:

- Nombre del archivo
- Ubicación física en el disco
- Tamaño
- Propietario y permisos
- Fechas de creación/modificación/último acceso

Comprobar con Inodos

4.2 Estructuras de Directorio

- **Nivel 1: Lista simple** - Todos los archivos en una sola lista - Problema: nombres deben ser únicos en todo el sistema
- **Nivel 2: Dos niveles** - Un directorio maestro + un directorio por usuario - Nombres únicos solo dentro de cada usuario - Mejor control de acceso
- **Nivel 3: Jerárquico (Árbol) ← MÁS USADO**
  - Directorio raíz → subdirectorios → archivos - Permite organizar por proyectos, tipos, etc.

4.3 Rutas

- **Ruta absoluta:** `/UsuarioB/Textos/Tema1/ABC`
- **Ruta relativa:** `Tema1/ABC` (desde el directorio actual)
- **Directorio de trabajo (working directory):** El directorio activo de cada usuario

Probarlo en WSL

5. COMPARTIMIENTO DE ARCHIVOS

5.1 Derechos de Acceso (Jerarquía)

`Ninguno → Conocimiento → Ejecución → Lectura → Adición → Actualización → Cambio de protección → Borrado`

Derecho	Descripción
Ninguno	No puede ni saber que existe el archivo
Conocimiento	Sabe que existe, puede pedir más permisos
Ejecución	Puede ejecutar el programa
Lectura	Puede leer y copiar
Adición	Puede añadir datos al final
Actualización	Puede modificar, borrar y añadir
Cambio de protección	Puede cambiar permisos de otros
Borrado	Puede eliminar el archivo

Regla: Si tienes un derecho, automáticamente tienes todos los anteriores.

5.2 Clases de Usuarios

- **Propietario:** Creador del archivo, tiene todos los derechos
- **Usuario específico:** Individual (por ID de usuario)
- **Grupo de usuarios:** Conjunto de usuarios predefinido
- **Todos:** Cualquier usuario del sistema

5.3 Accesos Simultáneos

- **Bloqueo de archivo completo:** Un usuario bloquea todo el archivo
- **Bloqueo de registros:** Solo bloquea el registro que está modificando (mejor)

- **Problema a resolver:** Exclusión mutua (evitar que dos usuarios modifiquen al mismo tiempo)

6. AGRUPACIÓN DE REGISTROS EN BLOQUES

6.1 Conceptos Clave

- **Registro:** Unidad LÓGICA de acceso (lo que ve el usuario)
- **Bloque:** Unidad FÍSICA de E/S (lo que lee/escribe el disco)
- **Problema:** Los registros no coinciden con el tamaño de los bloques del disco

6.2 Métodos de Agrupación

Bloques Fijos:

- Registros de longitud fija
- Número entero de registros por bloque
- Puede haber espacio sin usar al final (fragmentación interna) -
- **Más común** para archivos secuenciales

Bloques Variables con Tramos (Spanned):

- Registros de longitud variable
- Un registro puede ocupar dos bloques
- Se usa un puntero al siguiente bloque
- **Ventaja:** No limita tamaño de registro
- **Desventaja:** Más complejo, requiere 2 operaciones de E/S para registros divididos

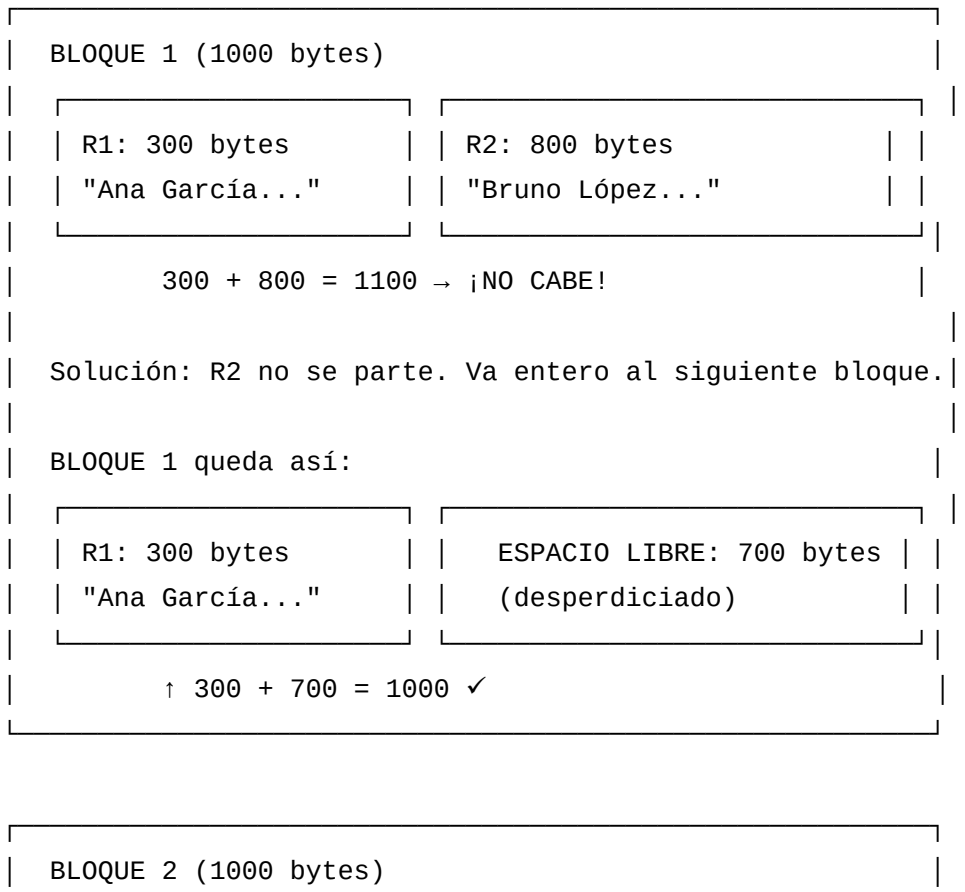
Bloques Variables sin Tramos (Unspanned):

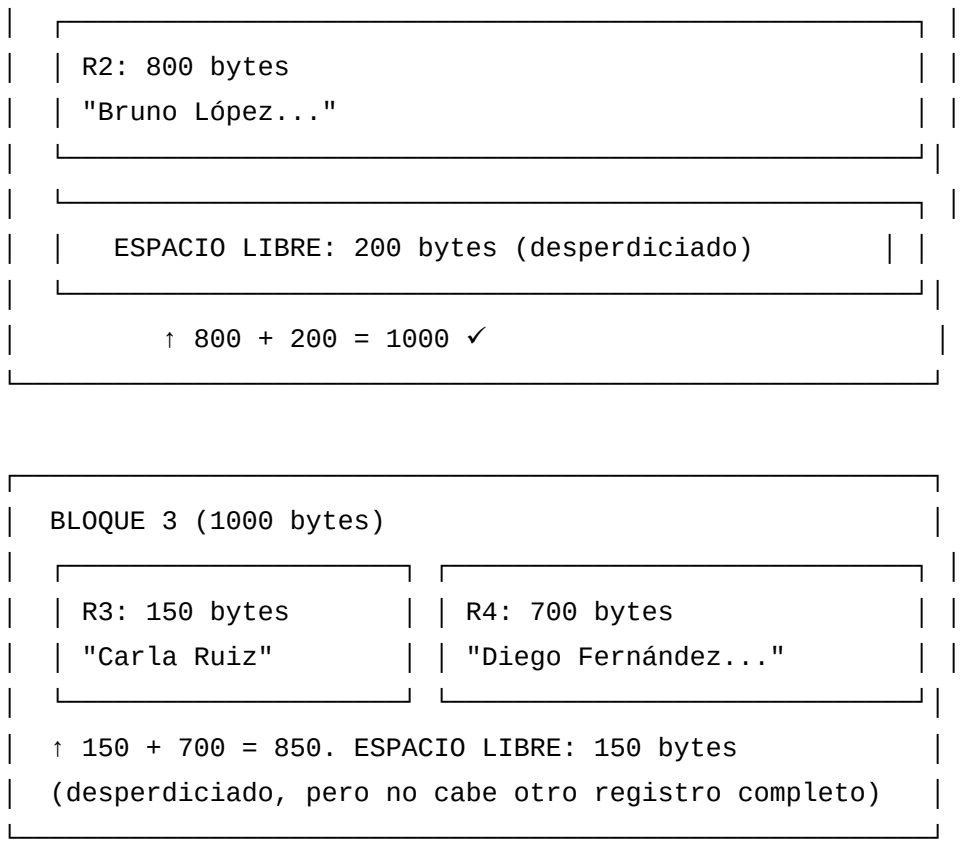
- Registros de longitud variable
- No se dividen entre bloques
- **Desventaja:** Espacio desperdiciado al final de bloques, límite de tamaño de registro

- Ejemplo : Imagina que tu disco tiene bloques de 1000 bytes (1 KB). Tienes estos registros:

Registro	Contenido	Tamaño
R1	"Ana García, DNI 12345678"	300 bytes
R2	"Bruno López Martínez del Castillo, DNI 87654321, dirección completa..."	800 bytes
R3	"Carla Ruiz"	150 bytes
R4	"Diego Fernández González con muchos apellidos y datos extra"	700 bytes

Cómo se Guardan en Bloques (sin tramos)





¿Por Qué Hay Espacio Desperdiciado?

Situación	¿Qué pasa?	Resultado
Registro pequeño sobra espacio	R1 = 300 bytes en bloque de 1000	700 bytes libres que no se pueden usar para otro registro si no cabe entero
Registro grande no cabe	R2 = 800 bytes, quedan 700 libres	Se salta al siguiente bloque, los 700 se desperdician
Suma de registros no llena exacto	R3 + R4 = 850 bytes	150 bytes sobran al final del bloque

Comparación

Método	¿Registro partido?	¿Espacio desperdiciado?	¿Límite de tamaño?
Con tramos (spanned)	✅ Sí	❌ Mínimo	❌ No hay límite
Sin tramos (unspanned)	❌ No	✅ Sí, al final de bloques	✅ Máximo = tamaño de bloque

Analogía Final

- Bloque = Maleta de mano (tamaño fijo, 1000 bytes)
- Registro = Objeto que metes en la maleta

- **Sin tramos:** No puedes partir un objeto en dos maletas. Si un jersey grande no cabe en la maleta que estás llenando (aunque queden 700 bytes libres), lo metes en una maleta nueva. Los 700 bytes de la primera maleta se quedan vacíos.

- **Con tramos:** Puedes poner la mitad del jersey en una maleta y la otra mitad en otra, con una nota que dice "continúa en maleta 5". No desperdicias espacio, pero necesitas dos maletas para vestirte.

6.3 Tamaño de Bloque

- **Bloques grandes:** Mejor para acceso secuencial (menos operaciones de E/S)
- **Bloques pequeños:** Mejor para acceso aleatorio (menos datos innecesarios transferidos)

7. GESTIÓN DEL ALMACENAMIENTO SECUNDARIO

7.1 Tipos de Memoria

Tipo	Velocidad	Capacidad	Uso
Principal (RAM)	Muy rápida	Limitada	Programas en ejecución
Secundaria (Disco)	Más lenta	Muy grande	Almacenamiento permanente

7.2 Asignación de Espacio a Archivos

Preguntas clave:

1. ¿Asignar todo el espacio al crear el archivo (asignación previa) o ir asignando según necesidad (asignación dinámica)?
2. ¿Qué tamaño de sección usar? (un bloque, varios bloques contiguos, etc.)
3. ¿Qué estructura de datos usar para saber qué bloques pertenecen a cada archivo?

Asignación Previa vs. Dinámica: - **Previa:** Se declara tamaño máximo al crear. Problema: difícil estimar, puede desperdiciar espacio. - **Dinámica:** Asigna bloques según se necesitan. **Mejor opción** en la mayoría de casos.

7.3 Métodos de Asignación de Archivos

A) ASIGNACIÓN CONTIGUA

- **Cómo funciona:** Un archivo ocupa bloques consecutivos en el disco
- **Información necesaria:** Bloque de inicio + longitud del archivo
- **Ventaja:** Excelente rendimiento para acceso secuencial (lectura rápida)
- **Desventaja:** Fragmentación externa (huecos entre archivos), difícil encontrar espacio contiguo grande

- **Solución:** Compactación periódica (mover archivos para juntar espacio libre)

B) ASIGNACIÓN ENCADENADA (Linked)

- **Cómo funciona:** Cada bloque tiene un puntero al siguiente bloque del archivo
- **Información necesaria:** Bloque de inicio + longitud
- **Ventaja:** No hay fragmentación externa, cualquier bloque libre sirve
- **Desventaja:** No mantiene cercanía física, acceso aleatorio lento (hay que seguir la cadena)

C) ASIGNACIÓN INDEXADA ← **MÁS POPULAR**

- **Cómo funciona:** Un bloque índice contiene punteros a todos los bloques del archivo
- **Información necesaria:** Puntero al bloque índice
- **Ventajas:**
  - Soporta acceso secuencial Y aleatorio
  - Elimina fragmentación externa
  - Puede usar bloques fijos o secciones variables
- **Desventaja:** Necesita espacio extra para el índice
- **Concentración:** Archivos pueden reorganizarse periódicamente para mejorar cercanía

8. GESTIÓN DEL ESPACIO LIBRE

8.1 Tabla de Bits (Bitmap)

- **Cómo funciona:** Un bit por cada bloque (0 = libre, 1 = ocupado)
- **Ventaja:** Fácil encontrar bloques libres
- **Desventaja:** Ocupa memoria. Para disco grande: tamaño del disco / (8 × tamaño de bloque)
- **Optimización:** Mantener resúmenes de subrangos para acelerar búsqueda

8.2 Secciones Libres Encadenadas

- **Cómo funciona:** Bloques libres encadenados con punteros

- **Ventaja:** No necesita tabla extra
- **Desventaja:** Con el tiempo se fragmenta en secciones de un solo bloque

8.3 Indexación del Espacio Libre

- **Cómo funciona:** Trata el espacio libre como un archivo con su propio índice
- **Ventaja:** Eficiente para todas las técnicas de asignación
- **Recomendación:** Usar secciones de tamaño variable (no bloques individuales)

8.4 Lista de Bloques Libres

- **Cómo funciona:** Lista de números de bloques libres en una sección reservada del disco
- **Espacio usado:** Menos del 1% del disco total
- **Optimizaciones:**
  - Como **pila** (stack): últimos bloques libres en memoria
  - Como **cola FIFO**: bloques al principio y final en memoria
- **Ventaja:** Solo se accede al disco cuando la memoria intermedia se llena o vacía

9. FIABILIDAD

9.1 El Problema

El sistema mantiene copias en memoria de las tablas de asignación (más rápido), pero si el sistema se cae antes de guardar en disco: - Usuario A recibe bloques asignados (solo en memoria) - Sistema se reinicia, pierde esa información - Usuario B recibe los mismos bloques (están marcados como libres en disco) - **Resultado:** ¡Dos archivos comparten los mismos bloques! (solapamiento)

9.2 Solución

1. **Bloquear** la tabla de asignación en disco (evita que otros usuarios la modifiquen)
2. Buscar espacio disponible
3. Asignar espacio y **actualizar el disco** (no solo memoria)
4. Actualizar tabla de archivos y **actualizar el disco**
5. **Desbloquear** la tabla

Optimización: Asignación por lotes (obtener varios bloques de una vez para reducir operaciones de disco)

10. SISTEMAS DE ARCHIVOS REALES

10.1 UNIX / Linux

- **Filosofía:** “Todo es un archivo” (flujo de bytes)
- **Tipos de archivos:**
  - **Ordinarios:** Datos de usuario
  - **Directorios:** Listas de nombres y punteros a nodos-i
  - **Especiales:** Dispositivos (impresora, disco, etc.)
  - **Nombrados:** Tuberías (pipes) con nombre
- **Nodos-i (inodes):**
  - Estructura de control con información del archivo (permisos, atributos, direcciones de bloques)
  - Tamaño fijo y pequeño → pueden permanecer en memoria
  - Varios nombres de archivo pueden apuntar al mismo nodo-i (enlaces duros)
- **Asignación:**
  - Bloques asignados dinámicamente (no hay asignación previa)
  - Método indexado con niveles de indirección
  - Nodo-i tiene 13 direcciones:
    - Primeras 10: apuntan directamente a bloques de datos
    - Dirección 11: indirecto simple (bloque con punteros a bloques de datos)
    - Dirección 12: indirecto doble (bloque con punteros a indirectos simples)
    - Dirección 13: indirecto triple (bloque con punteros a indirectos dobles)
  - Ejemplo: bloques de 1KB, cada puntero = 4 bytes → 256 punteros por bloque → tamaño máximo ~16 GB

10.2 Windows (NTFS)

- **Nombre:** New Technology File System

- **Características clave:**
  - **Recuperabilidad:** Registro de transacciones para reconstruir después de fallos
  - **Seguridad:** Modelo de objetos con descriptores de seguridad
  - **Discos grandes:** Soporta volúmenes de hasta 2TB
  - **Múltiples flujos de datos:** Un archivo puede tener varias “versiones” de contenido
  - **Indexación general:** Puede indexar archivos por cualquier atributo
- **Estructura del volumen:**
  - **Sector:** Unidad física mínima (típicamente 512 bytes)
  - **Agrupamiento (cluster):** 1+ sectores contiguos (unidad de asignación)
  - **Volumen:** Partición lógica del disco
- **Componentes principales:**
  1. **Sector de arranque:** Información de la partición
  2. **MFT (Master File Table):** Tabla maestra con información de TODOS los archivos y carpetas
  3. **MFT2:** Espejo de las primeras filas de la MFT (seguridad)
  4. **Archivo de registro:** Lista de transacciones para recuperación
  5. **Mapa de bits:** Muestra qué agrupamientos están en uso
- **Recuperabilidad (pasos):**
  1. NTFS escribe la transacción en el archivo de registro (en caché)
  2. NTFS modifica el volumen (en caché)
  3. Gestor de caché guarda el registro en disco
  4. Una vez seguro el registro, guarda los cambios del volumen en disco
  - Si falla antes del paso 3: la transacción se deshace
  - Si falla después del paso 3: la transacción se rehace

10.3 FAT (File Allocation Table)

- **Usado en:** MS-DOS, Windows hasta ME, disquetes, tarjetas de memoria, USB
- **Características:**

- Muy simple y compatible con casi todos los sistemas
- Tabla que indica qué bloques están libres y cuáles pertenecen a qué archivo
- Bueno para intercambio de datos entre sistemas diferentes
- Limitaciones en tamaño de archivos y seguridad

11. RESUMEN / CONCLUSIONES

Si necesitas…	**Usa…**
Procesar todo el archivo de una vez	**Archivo secuencial**
Acceso secuencial + aleatorio ocasional	**Archivo secuencial indexado**
Acceso principalmente aleatorio y rápido	**Archivo directo (hash)**
Acumular datos sin estructura fija	**Archivo de pila**
Máxima flexibilidad de acceso	**Archivo indexado**

Puntos clave a recordar:

1. **Registros vs. Bloques:** Los usuarios ven registros, el disco maneja bloques
2. **Directorios jerárquicos:** Esencial para organizar archivos y controlar acceso
3. **Asignación indexada:** El método más equilibrado (soporta secuencial y aleatorio)
4. **Fiabilidad:** Siempre actualizar en disco, no solo en memoria
5. **NTFS vs. UNIX:** Ambos usan asignación indexada, pero con diferentes implementaciones
6. **FAT:** Simple pero limitado, ideal para compatibilidad entre sistemas

Puntero de Desbordamiento

*Un **puntero de desbordamiento** es un mecanismo que permite insertar **nuevos registros** en un archivo organizado sin tener que **reordenar todo el archivo físicamente**.*

El Problema que Resuelve

*Imagina un archivo **secuencial ordenado por DNI**:*

plain

Copy

```
Archivo Principal (ordenado):
DNI 1001 → Ana
DNI 1002 → Bruno
DNI 1005 → Elena
DNI 1008 → Hugo
```

*Quiero insertar **Carlos con DNI 1003**. Si lo pongo en el archivo principal, tengo que:*

- *Mover a Elena y Hugo hacia adelante*
- *Reescribir todo el archivo desde esa posición*

***Solución:** No lo meto en el archivo principal. Lo meto en un **archivo de desbordamiento** y uso un **puntero** para enlazarlo.*

¿Cómo Funciona?

Paso 1: Archivo Principal

plain

Copy

```
Registro  DNI   Nombre   Puntero_Desbordamiento
──────────────────────────────────────────────
R1        1001  Ana      → null
```

```
R2        1002  Bruno    → null        ← ¡Este puntero va a cambiar!
R3        1005  Elena    → null
R4        1008  Hugo     → null
```

Paso 2: Insertar Carlos (DNI 1003)

*Carlos va al **Archivo de Desbordamiento** (no al principal):*

plain

Copy

```
Archivo de Desbordamiento:
_____________________________________________________

D1        1003  Carlos   → null
```

Paso 3: Actualizar el Puntero

*Bruno (DNI 1002) es el que **precede** a Carlos en orden lógico. Entonces:*

plain

Copy

```
R2 (Bruno) ahora apunta a D1 (Carlos):
_____________________________________________________

R2        1002  Bruno    → D1    ← PUNTERO DE DESBORDAMIENTO
```

Paso 4: ¿Y si inserto otro? Diana (DNI 1004)

plain

Copy

```
Archivo de Desbordamiento:
_____________________________________________________

D1        1003  Carlos   → D2    ← Ahora Carlos apunta a Diana
D2        1004  Diana    → null
```

Paso 5: ¿Qué pasa con Elena?

*Elena (DNI 1005) sigue en el **archivo principal**. El puntero de Diana apunta de vuelta:*

plain

Copy

```
D2        1004  Diana    → R3    ← Diana apunta a Elena en el principal
R3        1005  Elena    → null
```

Diagrama Visual del Recorrido Lógico

plain

Copy

```
ORDEN LÓGICO (lo que el usuario ve):   1001 → 1002 → 1003 → 1004 → 1005 → 1008

ORDEN FÍSICO EN DISCO:
┌──────────────┐     ┌──────────────────┐
│  PRINCIPAL   │     │  DESBORDAMIENTO  │
├──────────────┤     ├──────────────────┤
│ R1: 1001 Ana │────→│                  │
│ R2: 1002 Bru │──┐  │                  │
│ R3: 1005 Ele │  │  │ D1: 1003 Carlos  │
│ R4: 1008 Hug │  └──→│ D2: 1004 Diana   │
└──────────────┘     └──────────────────┘

Punteros:
R2 → D1 → D2 → R3
```

¿Por Qué se Llama "Desbordamiento"?

*Porque los nuevos registros **"desbordan"** del archivo principal — no caben en su posición ordenada, así que se almacenan en una zona aparte y se enlazan con punteros.*

Resumen con una Analogía

> ***Archivo principal*** *= Calle principal con casas numeradas en orden*
> ***Archivo de desbordamiento*** *= Calle lateral donde construyes casas nuevas*
> ***Puntero de desbordamiento*** *= Flecha en la casa 1002 que dice "la 1003 está por allá, en la calle lateral"*
