# Sistemas Operativos

## Sistema de Archivos

Dr. Wenceslao Palma M.<wenceslao.palma@ucv.cl>

La unica forma en que un usuario o aplicación puede acceder a un archivo es mediante el sistema de archivos.

Sus objetivos son:

- Cumplir con las necesidades de gestión de datos y requerimientos del usuario.
- Garantizar, en lo posible, que los datos de los archivos sean válidos.
- Optimizar el rendimiento global y desde el punto de vista del usuario.
- Ofrecer soporte de E/S considerando la variedad de tipos de dispositivos de almacenamiento.
- Minimizar o eliminar la pérdida o destrucción de datos.
- Proporcionar un conjunto estándar de rutinas de interfaz de E/S.
- Proporcionar soporte de E/S para múltiples usuario (relación con validez de los datos)

# Arquitectura de los sistemas de archivos

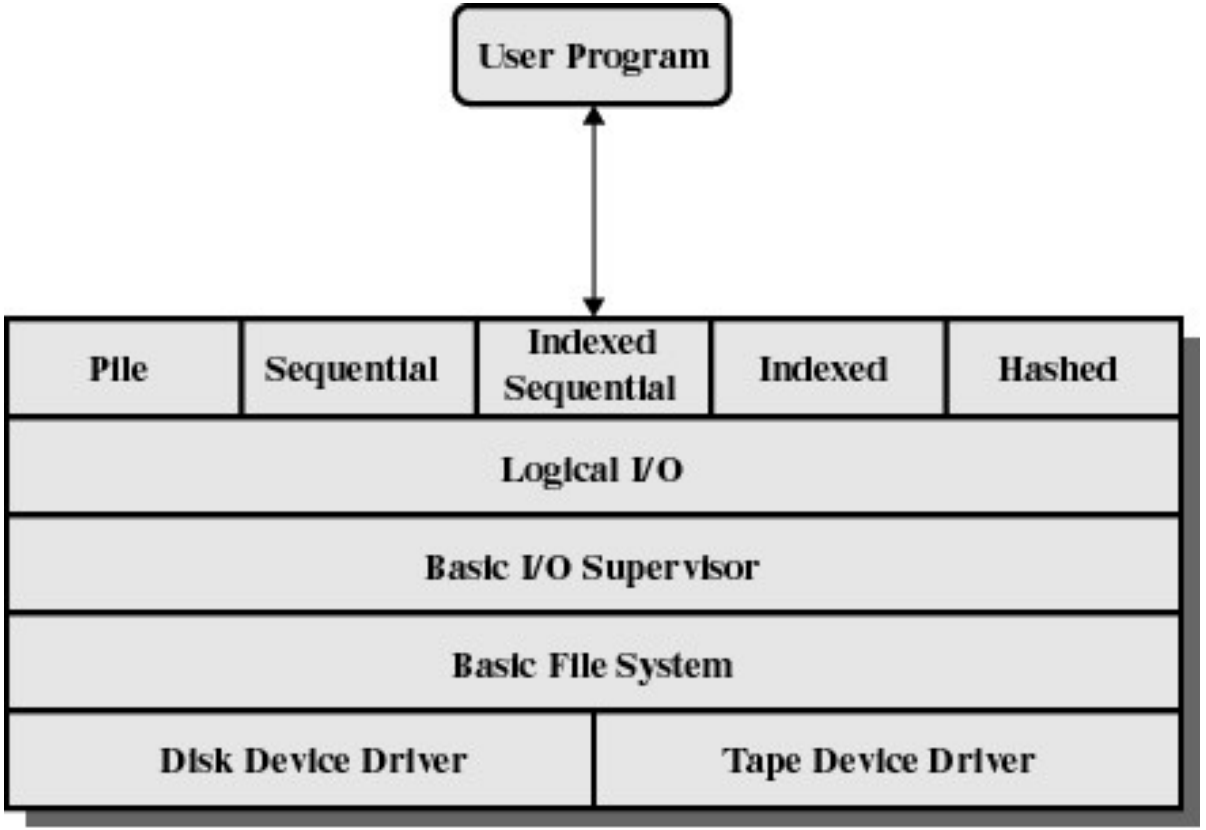



Figure 12.1 File System Software Architecture [GROS86]

# Funciones de un sistema de archivos

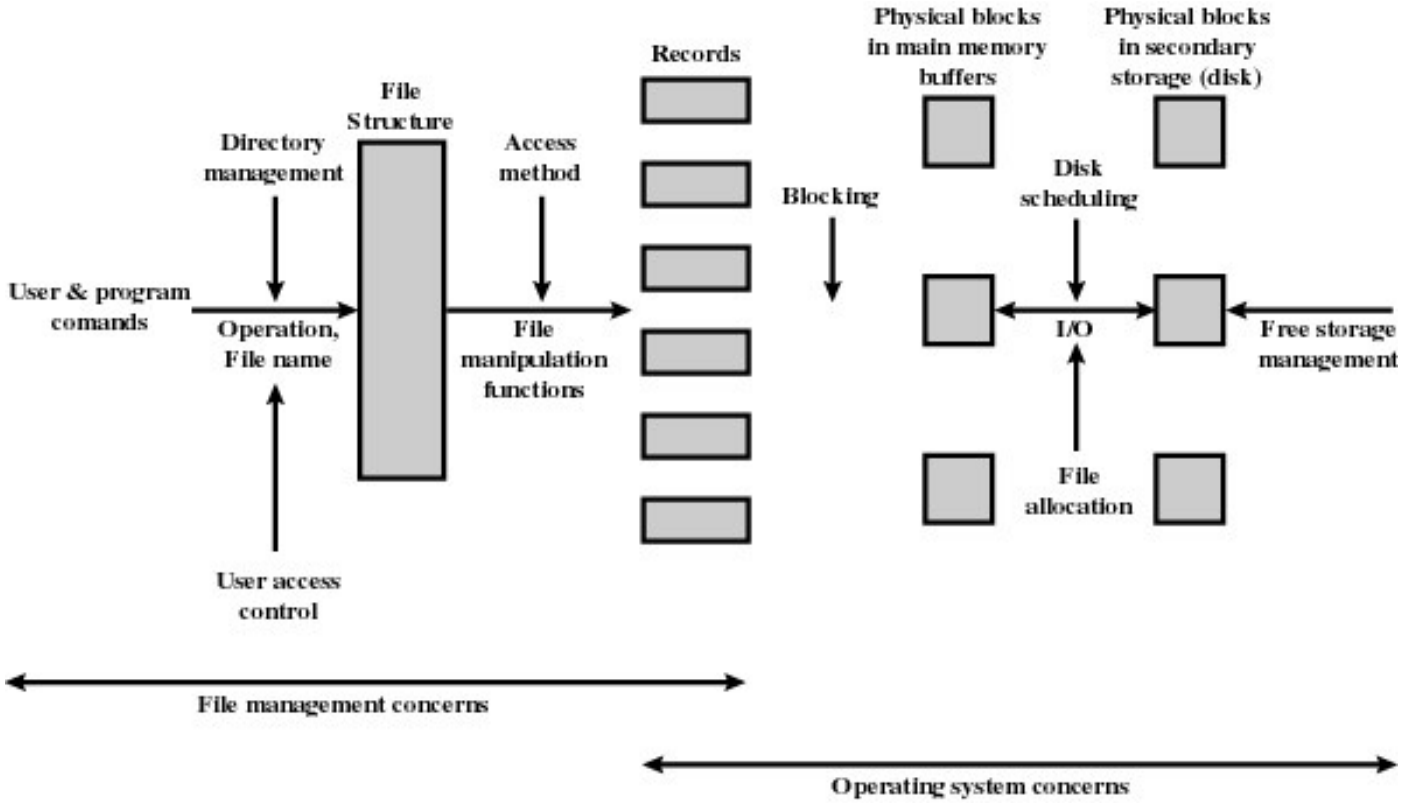



**Figure 12.2 Elements of File Management**

Considerando el esquema anterior las funciones son:

- Identificar y ubicar archivos.
- Control de acceso a los usuarios.
- Determinar cuales son los bloques que componen un archivo.
- Asignar bloques disponibles a archivos que lo requieran.

# Organizacion de directorios

Un directorio contiene información sobre archivos.

Un directorio también es un archivo.

La información de un directorio se clasifica en:

- Básica: nombre, tipo (texto, binario, módulo de carga, etc.) y organización.
- De dirección: dispositivo, dirección de inicio, tamaño usado/asignado.
- De control de acceso: propietario, operaciones permitidas
- De uso: fecha creación, fecha última lectura/escritura, identidad de quién lo crea/ultimo lector, utilización actual, etc.

La estructura de un directorio es importante. Si es lineal, el usuario tendrá muchos problemas para asignar nombres únicos. La estructura más adoptada es la de árbol.

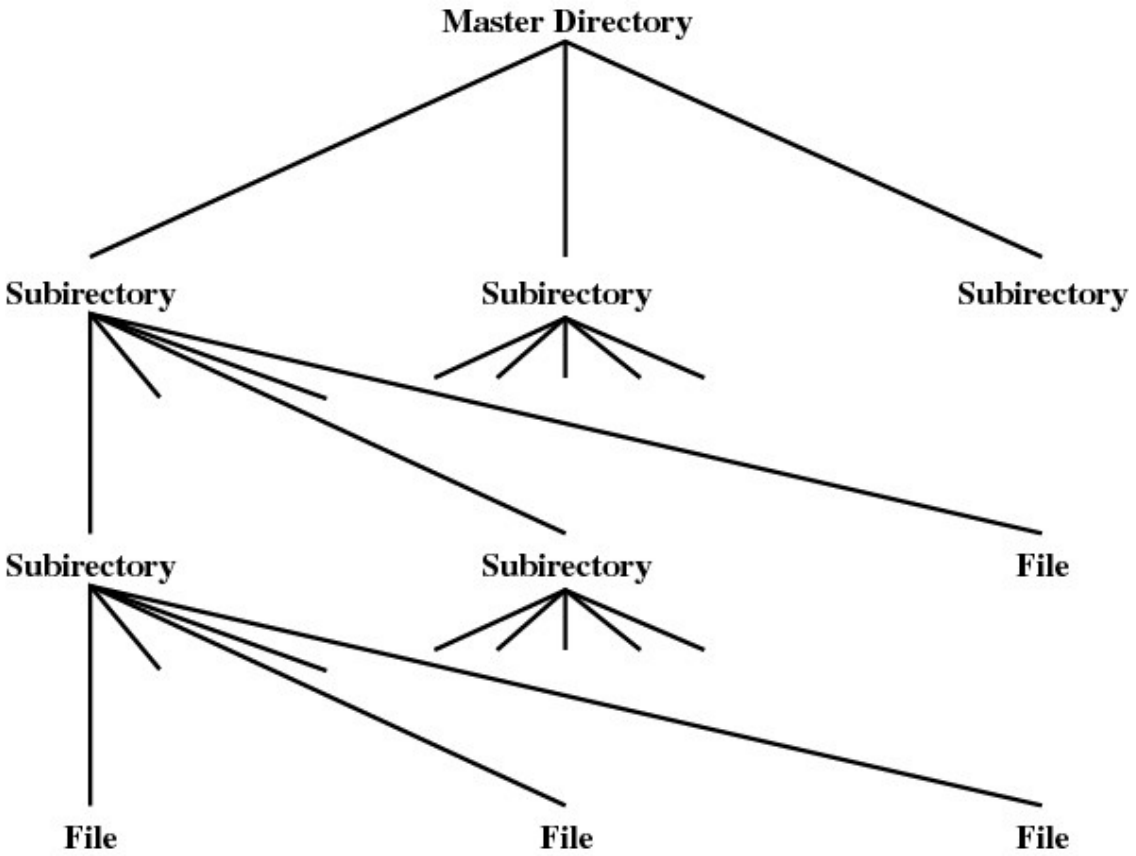



**Figure 12.4 Tree-Structured Directory**

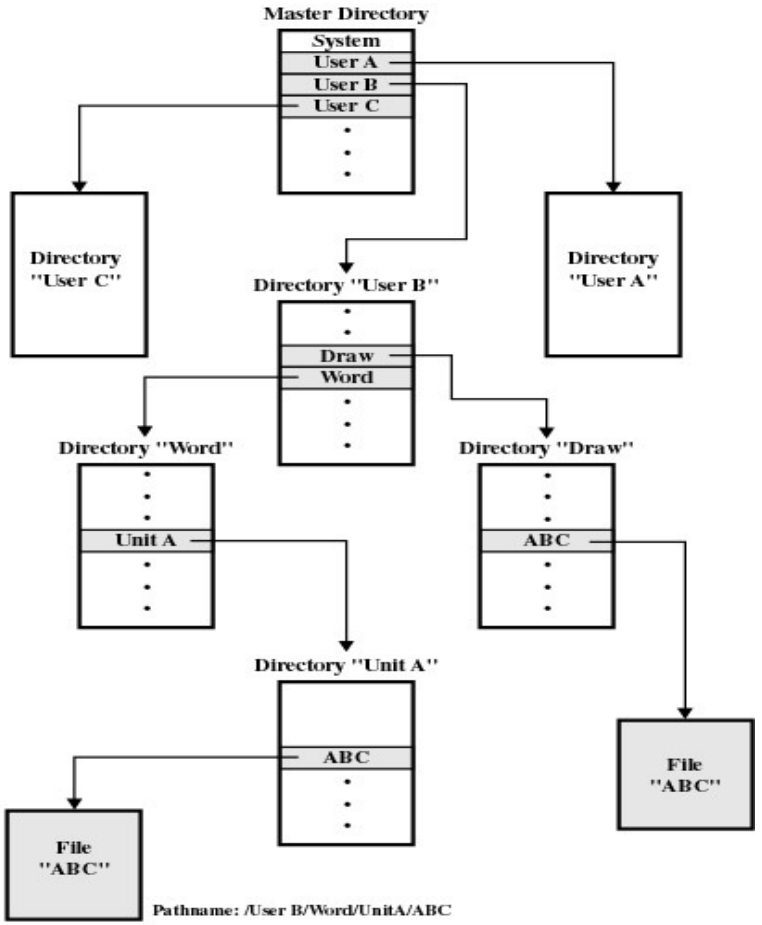



**Figure 12.5 Example of Tree-Structured Directory**

## Compartimiento de archivos

Generalmente se conceden accesos a las siguientes clases de usuarios: específico, grupo, otros.

Los acceso se relacionan con operaciones sobre los archivos. Básicamente corresponden a conocimiento, lectura, escritura, cambio de protección, borrado.

También es importante considerar como se manejan los accesos concurrentes sobre un mismo archivo.

## Administración del almacenamiento secundario

El SO es el responsable de asignar bloques a archivos.

El método usado para asignar espacio influirá en la forma en que se administra el espacio libre.

La asignación de espacio debe contemplar la forma en la que se asignan los bloques a un archivo, cuál es el tamaño del bloque y qué tipo de estructura de datos(tabla de asignación de archivos) se utilizará para llevar un registro de los bloques asignados.

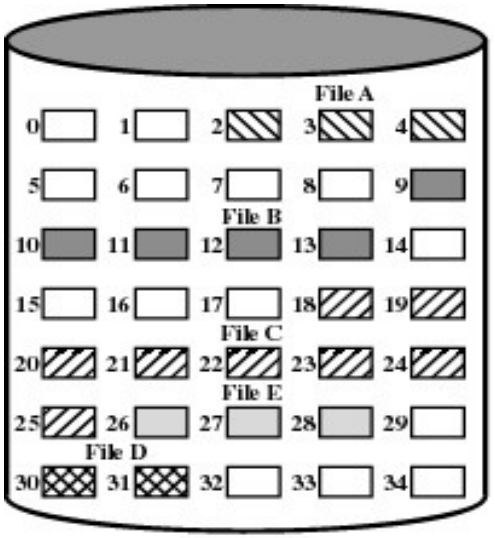
## Asignación previa v/s asignación dinámica

La asignación previa requiere conocer el tamaño máximo del archivo. Esto es muy difícil de obtener ante lo cual en la mayoría de los casos se optaría por una sobrestimación del espacio requerido lo cual es muy ineficiente.

Luego, es más ventajosa la asignación dinámica, la cual asigna espacio a los archivos en la medida que lo necesitan.

## Métodos de asignación de archivos

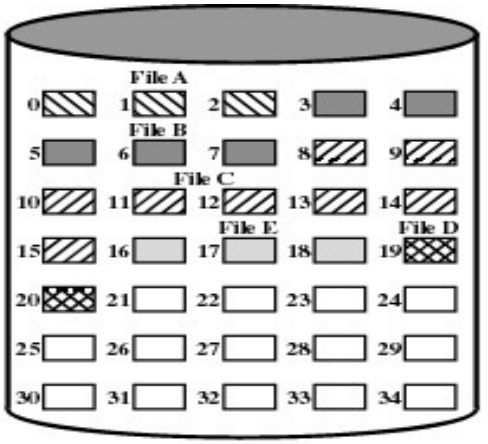
Asignación contigua: cuando un archivo es creado se le asigna un único conjunto de bloques, lo cual requiere de asignacion previa.
La tabla de asignación de archivos requiere una entrada por cada archivo que considera bloque de inicio y tamaño del archivo.
Se genera fragmentación externa que se resuelve mediante compactación.




File Allocation Table

| File Name | Start Block | Length |
|---|---|---|
| File A | 2 | 3 |
| File B | 9 | 5 |
| File C | 18 | 8 |
| File D | 30 | 2 |
| File E | 26 | 3 |

**Figure 12.7 Contiguous File Allocation**




File Allocation Table

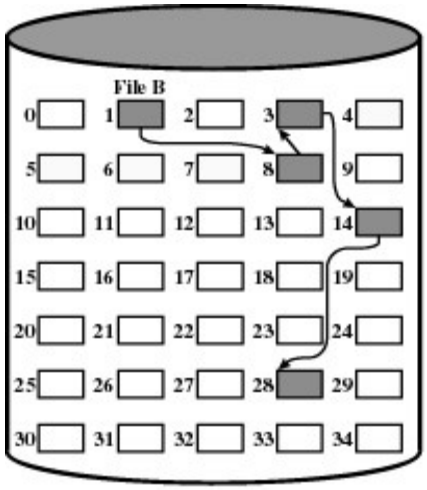
| File Name | Start Block | Length |
|---|---|---|
| File A | 0 | 3 |
| File B | 3 | 5 |
| File C | 8 | 8 |
| File D | 19 | 2 |
| File E | 16 | 3 |

**Figure 12.8 Contiguous File Allocation (After Compaction**

Asignacion encadenada

Cada bloque contiene un puntero al siguiente de la cadena. La tabla de asignación necesita solo una entrada por archivo.
La asignación de bloque es sencilla: cualquier bloque libre puede ser parte de la cadena.

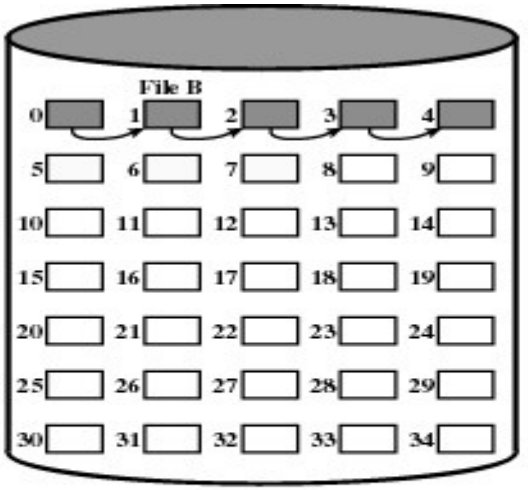
Puede generar accesos a diferentes partes del disco lo cual afecta el rendimiento.




File Allocation Table

| File Name | Start Block | Length |
|---|---|---|
| ... | ... | ... |
| File B | 1 | 5 |
| ... | ... | ... |

Figure 12.9 Chained Allocation




File Allocation Table

| File Name | Start Block | Length |
|---|---|---|
| ... | ... | ... |
| File B | 0 | 5 |
| ... | ... | ... |

Figure 12.10 Chained Allocation (after consolidation)

Asignación indexada

La tabla de asignación tiene como entrada el bloque en donde se encuentra el índice del archivo.

Cada entrada del indice contiene una referencia a cada bloque del archivo

De acuerdo a como se asigna el bloque se elimina la fragmentación externa o se mejora la cercanía.

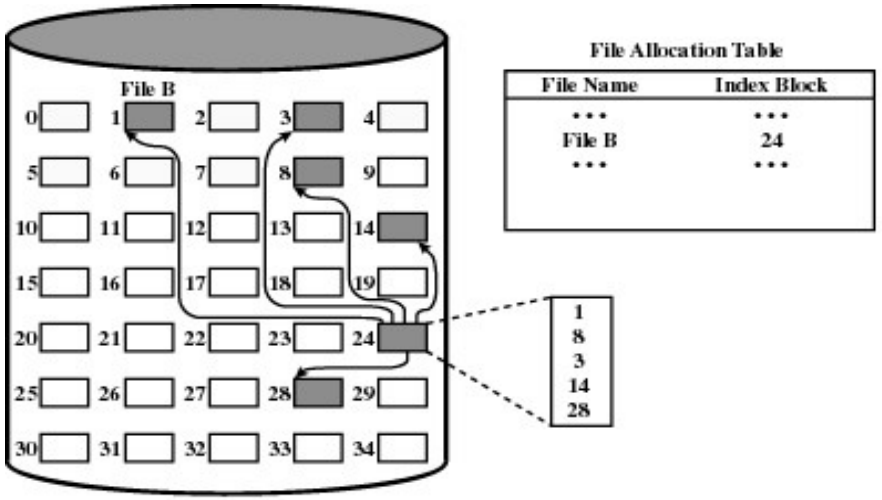



Figure 12.11 Indexed Allocation with Block Portions

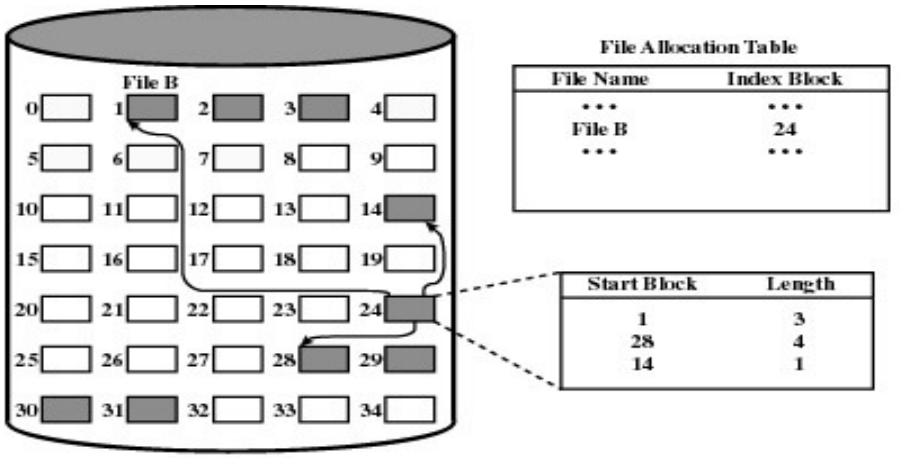



Figure 12.12 Indexed Allocation with Variable-Length Portions

# Administración de archivos en Unix

El kernel de Unix considera a todos los archivos como un flujo de bytes. Cualquier estructura interna será específica de la aplicación.

Se distinguen cuatro tipos de archivos:

Ordinarios: contienen información generada por el usuario. (código fuente, utilitario del sistema, etc.)

Directorio: contiene una lista de los archivos que contiene y punteros a i-nodos. Los directorios se organizan en forma jerárquica. Un directorio es un archivo ordinario con privilegio de protección de tal forma que solo el sistema de archivos puede escribirlo.

Especiales: usados para acceder a periféricos como terminales e impresoras. Cada dispositivo esta asociado a un archivo especial. (/dev/fd0, /dev/hda1, /dev/cdrom, etc.)

Nombrados: named pipes.

# I-Nodo

Es una estructura de control que contiene la informacion clave de un archivo.

Pueden asociarse varios nombres de archivo a un mismo i-nodo, pero un i-nodo activo se puede asociar con un único archivo y cada archivo es controlado por un solo i-nodo.

La información de un i-nodo considera:

- Modo de archivo: flag de 16 bits que almacena permisos de acceso y ejecucion asociados al archivo. Bits 12-14, tipo de archivo. Bits 9-11, indicadores de ejecucion. Bits 0-8, rwxr—r---
- Cuenta de enlaces: números de referencias al i-nodo en los directorios.
- ID del dueño
- ID del grupo
- Tamaño del archivo
- Direcciones del archivo
- Ultimo acceso

Direcciones del archivo: 39 bytes de información de dirección

Fecha último acceso

Fecha última modificación

Fecha última modificación del i-nodo

## Asignación de archivos

Los archivos se asignan en bloques en forma dinámica. Se usa un método de indexación para seguir la pista de cada archivo, parte del índice se almacena en el i-nodo.

Un i-nodo incluye 39 bytes de información de dirección, organizada en 13 direcciones de 3 bytes. Las primeras 10 direcciones apuntan a los primeros 10 bloques de datos del archivo. Si el archivo es mayor que 10 bloques se usan uno o mas niveles indirectos, de la siguiente manera:

La direccion undécima del i-nodo apunta a un bloque del disco que contiene la siguiente parte del índice. Esto se conoce como bloque indirecto simple.

Si el archivo contiene más bloques, la duodécima dirección del i-nodo apuntará a un bloque indirecto doble. Si aún asi se necesitan más bloques la decimotercera dirección apuntará a un bloque indirecto triple

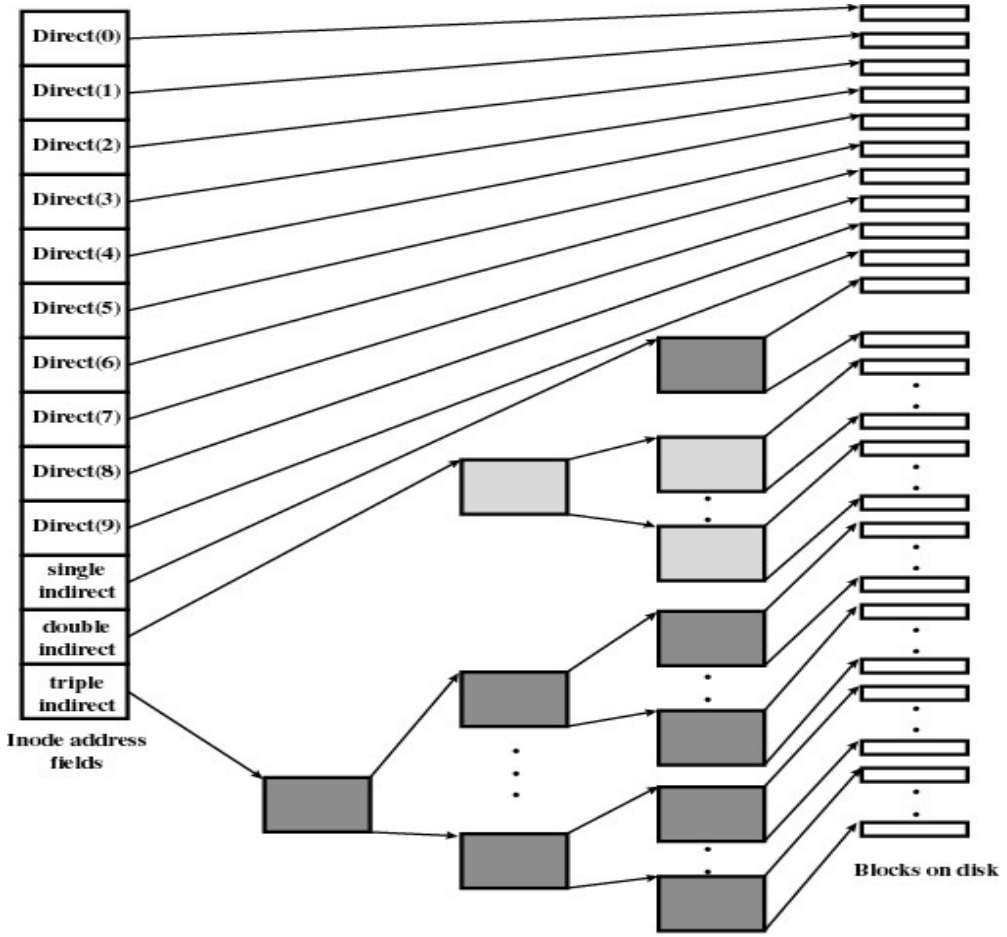



**Figure 12.13 UNIX Block Addressing Scheme**

Las ventajas del esquema está en que los i-nodo son de tamaño fijo y pueden almacenarse en memoria principal. Es posible acceder a los archivos pequeños en forma directa y el tamaño maximo teorico (16 GB) de un archivo es suficientemente grande como para satisfacer a casi todas las aplicaciones.