Operating Systems (CSE531)
Lecture # 12

Manish Shrivastava

LTRC, IIIT Hyderabad

Outline

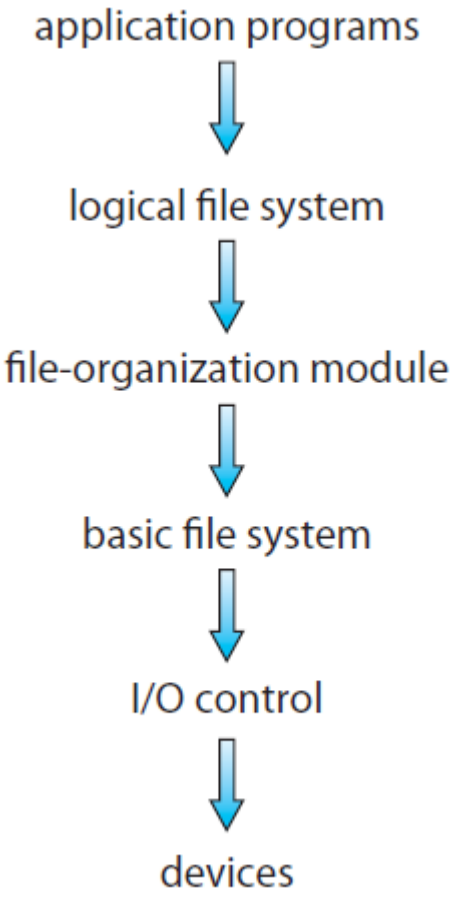
- File System Implementation
  - File System Structure
  - File System Implementation
  - Directory Implementation
  - Allocation Methods
  - Free-Space Management
  - Efficiency and Performance
  - Recovery

File-System Structure

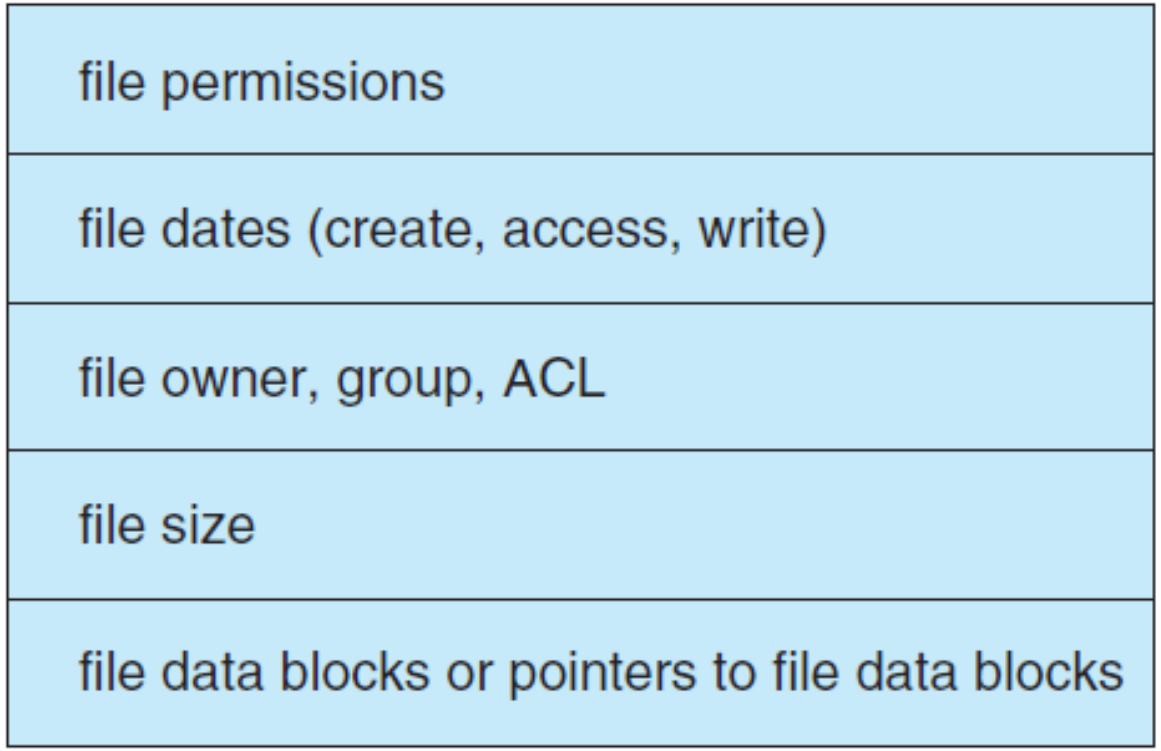
- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks).
- File system organized into layers.
- *File control block* – storage structure consisting of information about a file.

Layered file system

- Device drivers: transfer information from main memory and disk system
- Devise driver writes data into I/O controller's memory
- The basic file system issues generic commands to appropriate device drivers.
- The file organization module knows about files and logical blocks.
  - It can translate logical block addresses to physical block addresses.
- The logical file system manages meta-information.
  - File system structure (excludes contents)
  - Manages directory structure
  - It maintains file structure via file control blocks (FCBs).
    - Ownership, permissions, ACL (access control list) and location of the contents.

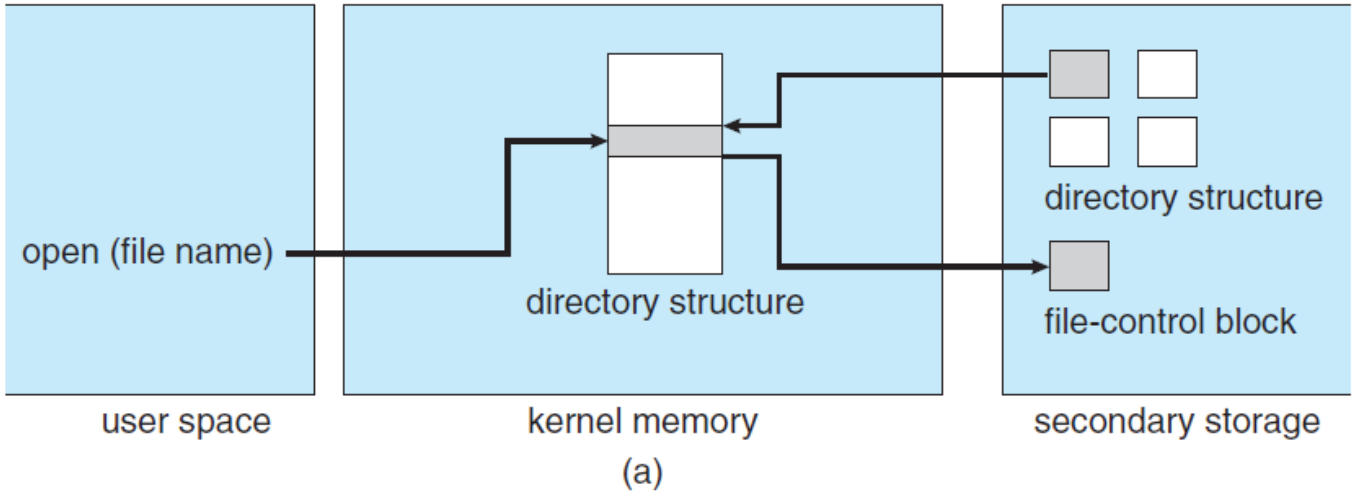


A Typical File Control Block

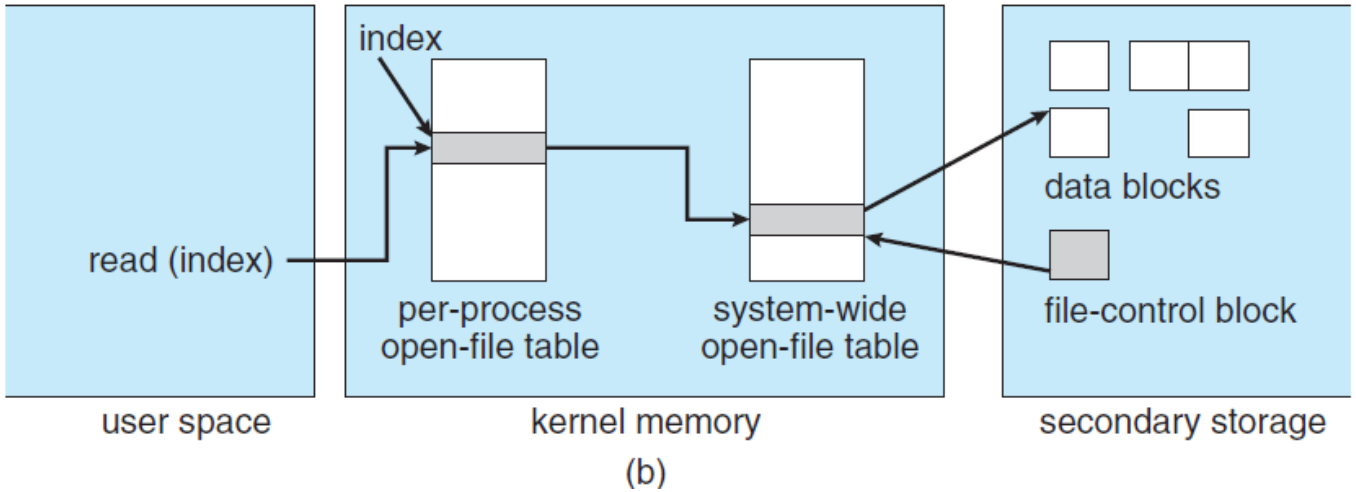


In-Memory File System Structures

- File Open



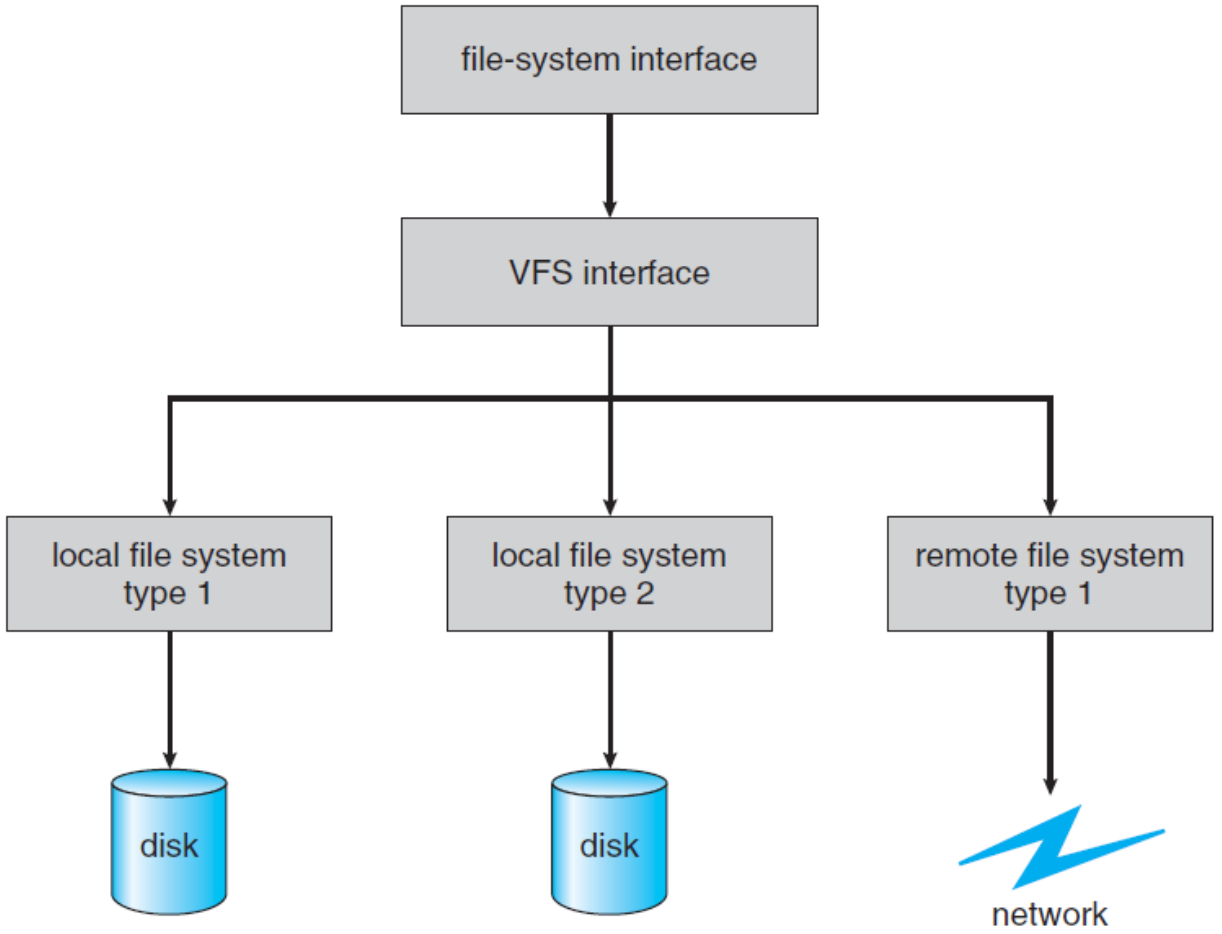
- File Read



Virtual File Systems

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.

Schematic View of Virtual File System



Directory Implementation

- Linear list of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute

- Hash Table – linear list with hash data structure.
  - decreases directory search time
  - *collisions* – situations where two file names hash to the same location
  - fixed size

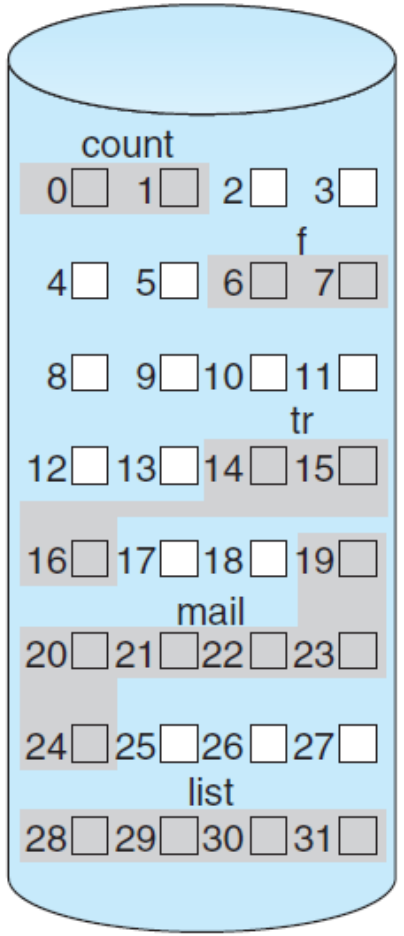
Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
- Contiguous allocation
- Linked allocation
- Indexed allocation

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Simple – only starting location (block #) and length (number of blocks) are required.
- Random access.
- Wasteful of space (dynamic storage-allocation problem).
- Files cannot grow.

Contiguous Allocation of Disk Space



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

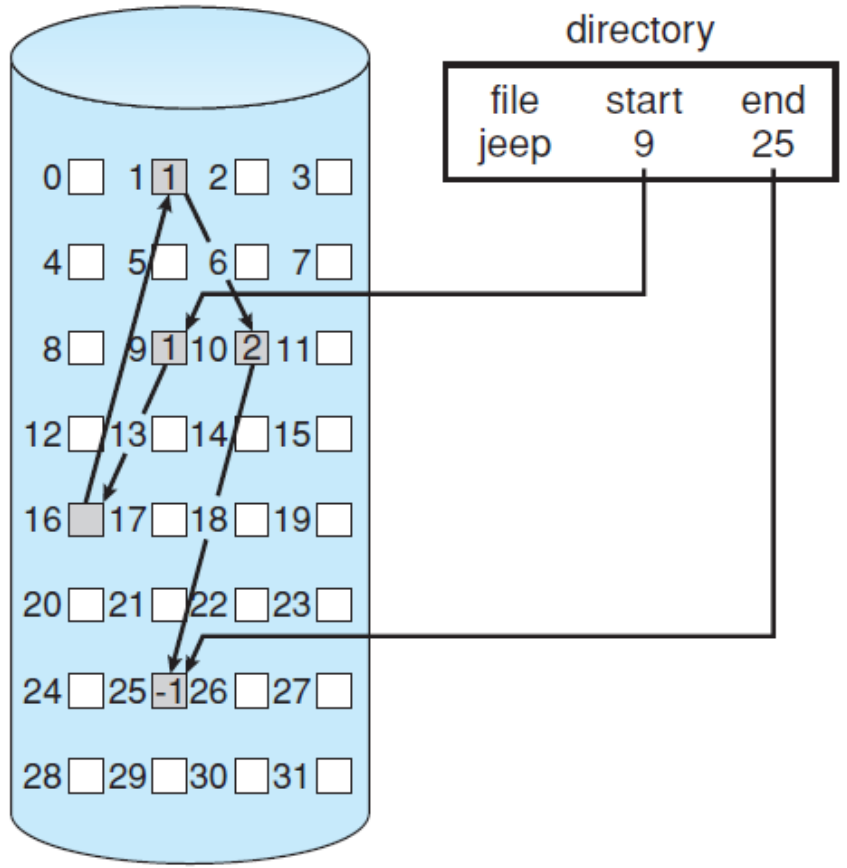
Extent-Based Systems

- Many file systems use a modified contiguous allocation scheme.
- Extent-based file systems allocate disk blocks in **extents**.
- An **extent** is a contiguous block of disks. Extents are allocated for file allocation. A file consists of one or more extents.

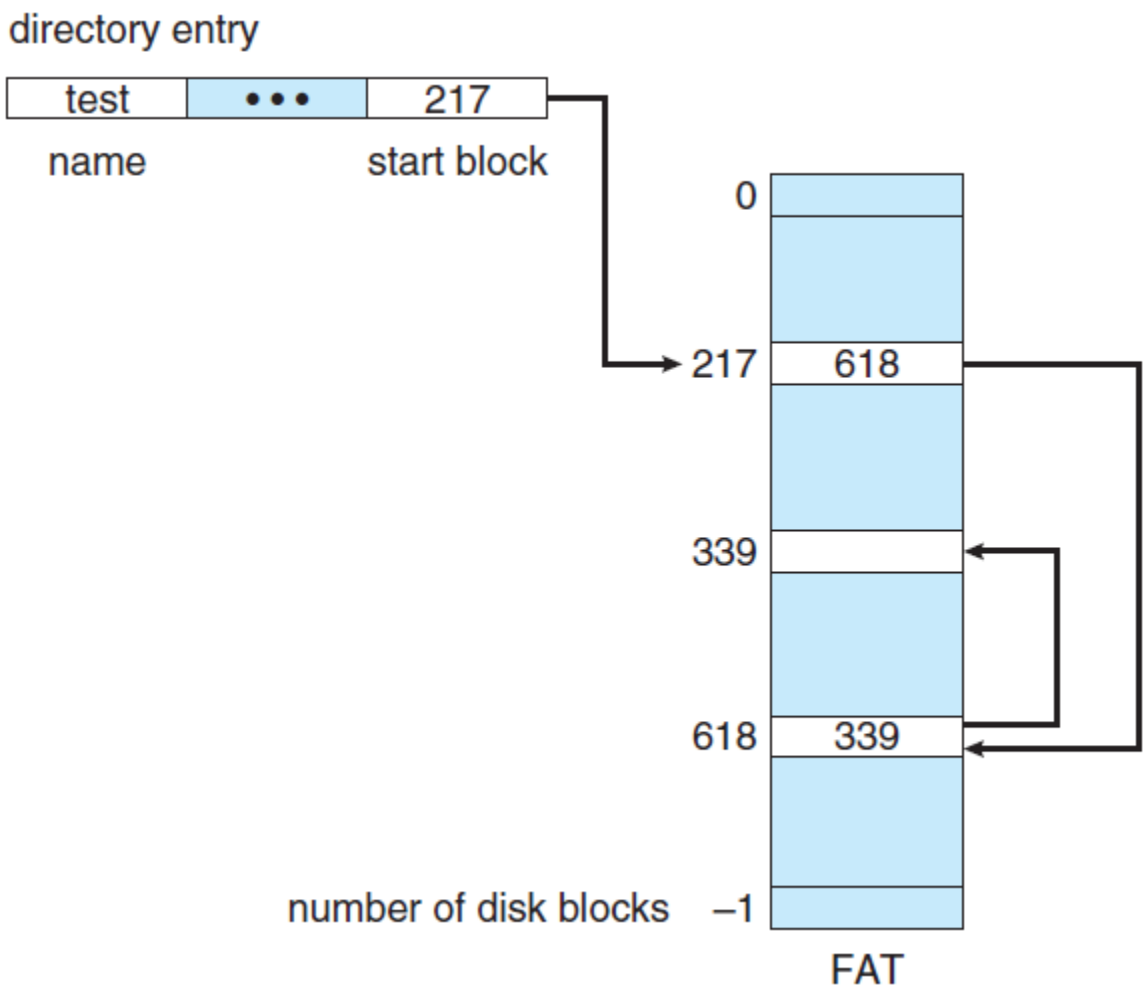
Linked Allocation

block = | pointer |

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.
  - Simple – need only starting address
  - Free-space management system – no waste of space
  - No random access
  - Mapping

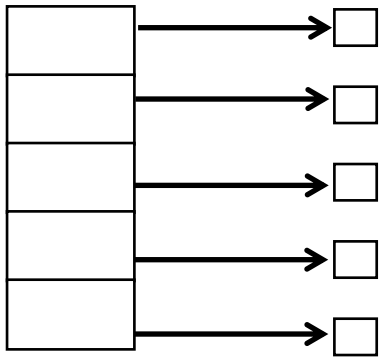


File-Allocation Table

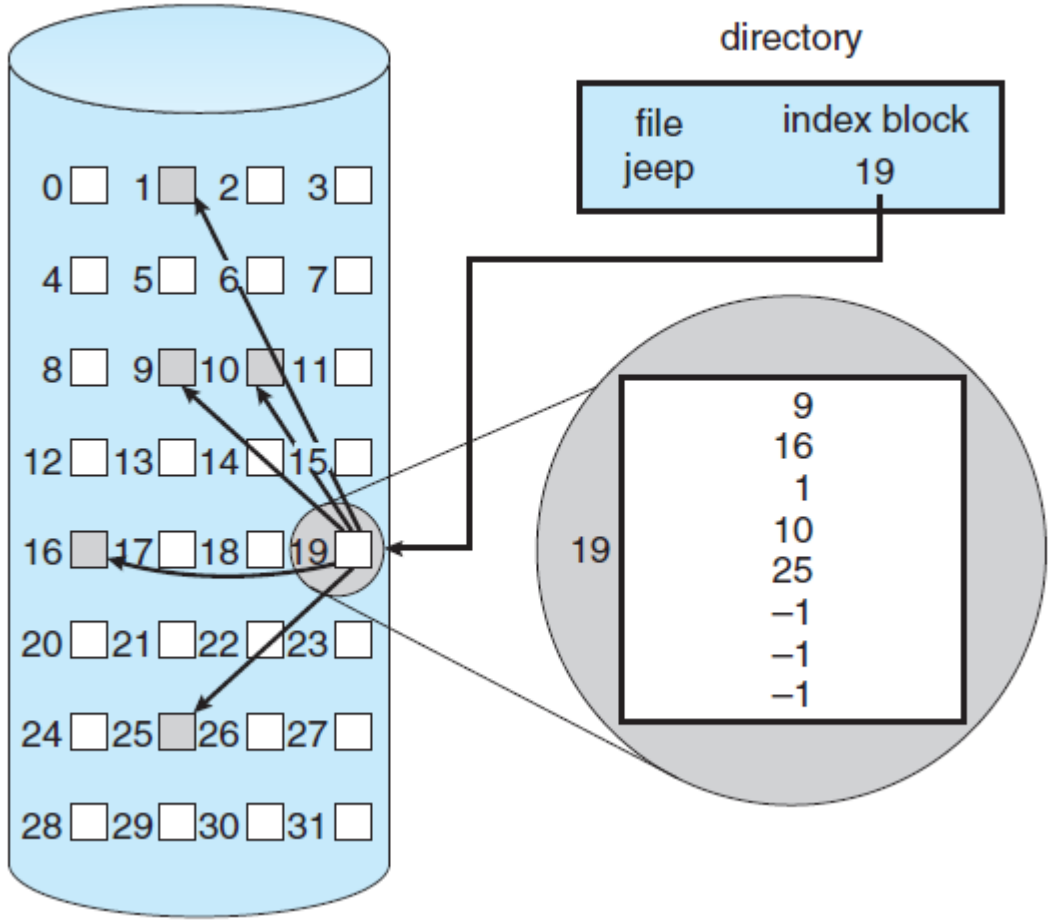


Indexed Allocation

- Brings all pointers together into the *index block.*
- Logical view.



Example of Indexed Allocation

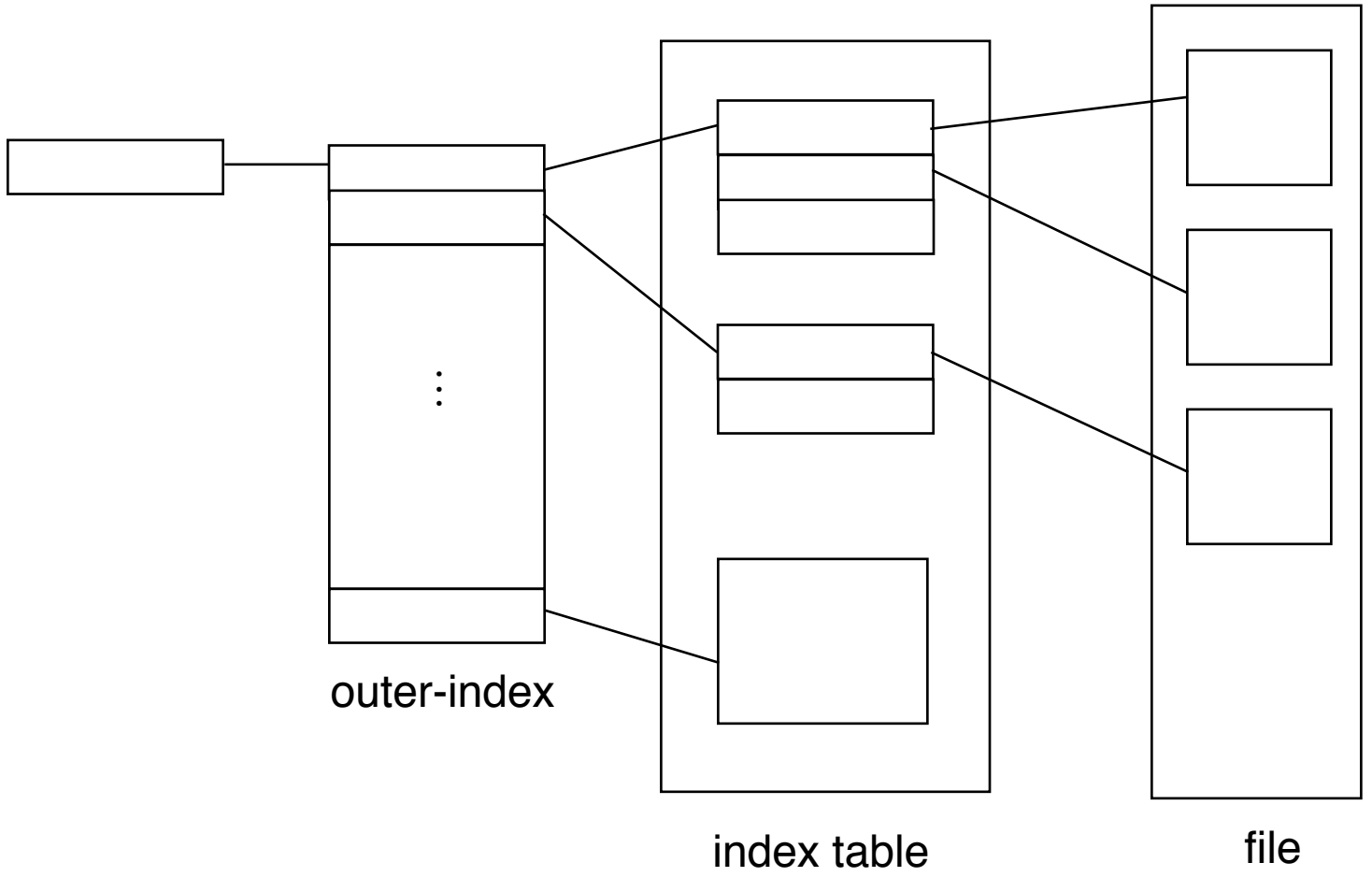


Indexed Allocation

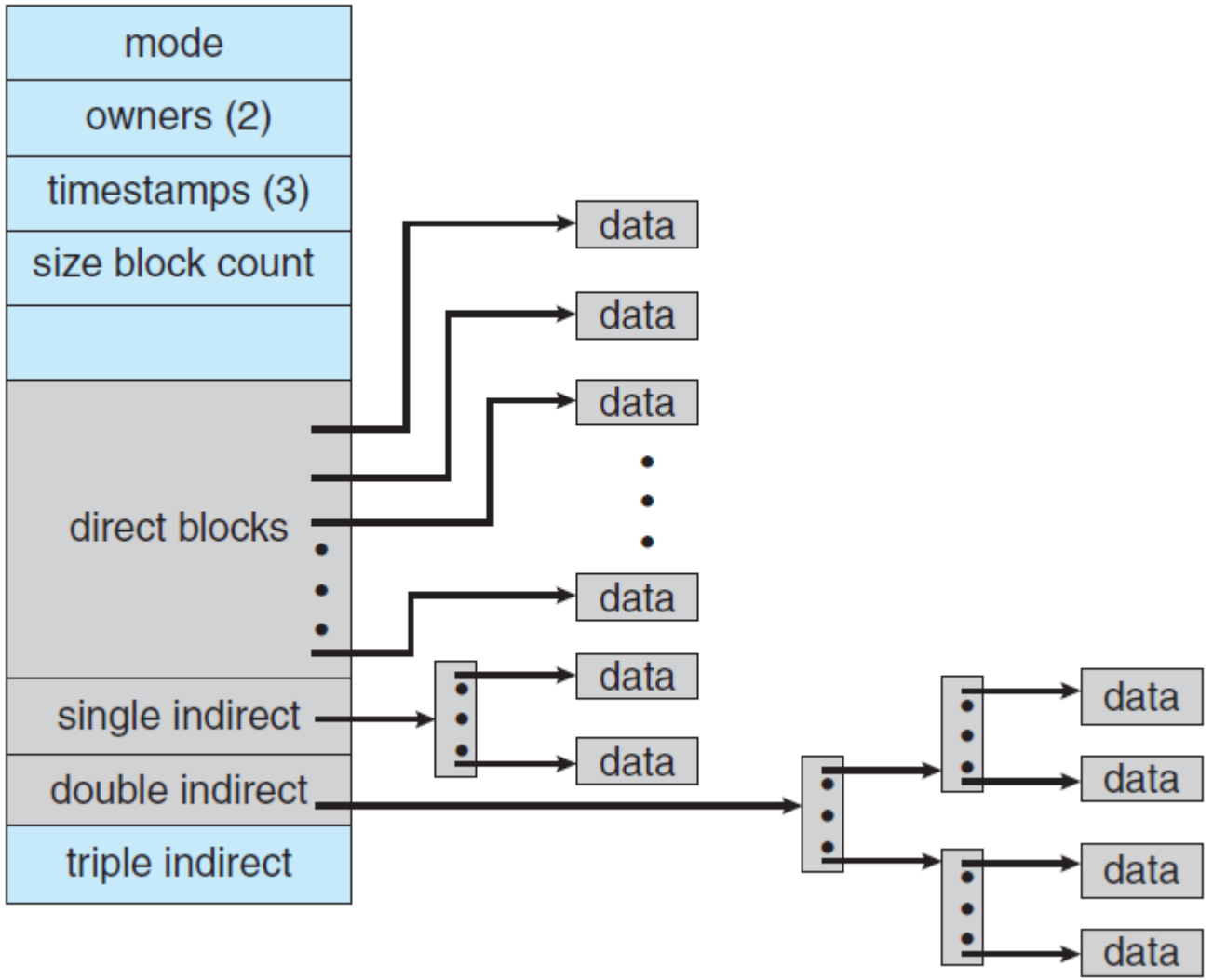
- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.
- Mapping from logical to physical in a file of maximum size of 256K words and block size of 512 words. We need only 1 block for index table.
- Mapping from logical to physical in a file of unbounded length (block size of 512 words).
- Linked scheme – Link blocks of index table (no limit on size).

Indexed Allocation

- Two-level index

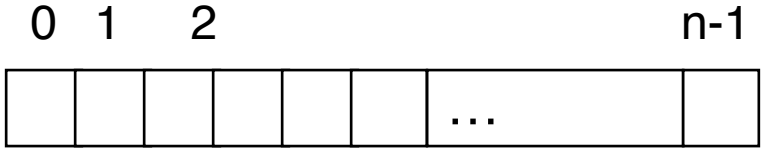


Combined Scheme: UNIX inode



Free-Space Management

- Bit vector (*n* blocks)



$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ free} \\ 1 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

Block number calculation

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

Free-Space Management

- Bit map requires extra space. Example:

  block size = 2^{12} bytes

  disk size = 2^{30} bytes (1 gigabyte)

  $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)
- Easy to get contiguous files
- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
- Grouping
- Counting

Free-Space Management

- Need to protect:
  - Pointer to free list
  - Bit map
    - Must be kept on disk
    - Copy in memory and disk may differ.
    - Cannot allow for block[*i*] to have a situation where bit[*i*] = 1 in memory and bit[*i*] = 0 on disk.
  - Solution:
    - Set bit[*i*] = 1 in disk.
    - Allocate block[*i*]
    - Set bit[*i*] = 1 in memory

Linked Free Space List on Disk