UNIT IV

STORAGE MANAGEMENT

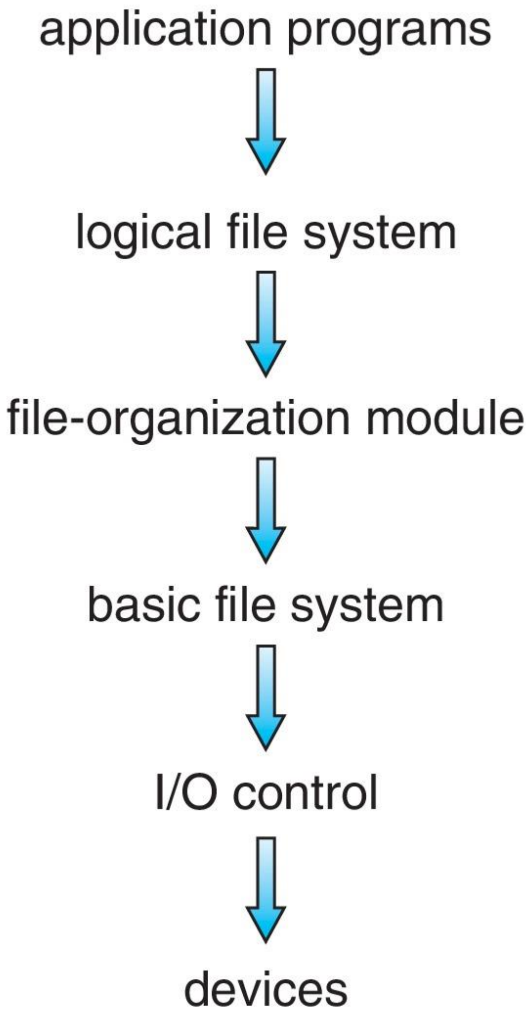
File-System Implementation

- File System Structure
- Directory implementation

File-System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- **File system** resides on secondary storage (disks)
  - Provided user interface to storage, mapping logical to physical
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved easily
- Disk provides in-place rewrite and random access
  - I/O transfers performed in **blocks** of **sectors** (usually 512 bytes)
- **File control block** (**FCB**) – storage structure consisting of information about a file
- **Device driver** controls the physical device
- File system organized into layers

Layered File System


application programs

logical file system

file-organization module

basic file system

I/O control

devices

File System Layers

- **Device drivers** manage I/O devices at the I/O control layer

  Given commands like

  read drive1, cylinder 72, track 2, sector 10, into memory location 1060

  Outputs low-level hardware specific commands to hardware controller

- **Basic file system** given command like “retrieve block 123” translates to device driver
- Also manages memory buffers and caches (allocation, freeing, replacement)
  - Buffers hold data in transit
  - Caches hold frequently used data
- **File organization module** understands files, logical address, and physical blocks
- Translates logical block # to physical block #
- Manages free space, disk allocation

File System Layers (Cont.)

- **Logical file system** manages metadata information
  - Translates file name into file number, file handle, location by maintaining file control blocks (**inodes** in UNIX)
  - Directory management
  - Protection
- Layering useful for reducing complexity and redundancy, but adds overhead and can decrease performance
- Logical layers can be implemented by any coding method according to OS designer

File System Layers (Cont.)

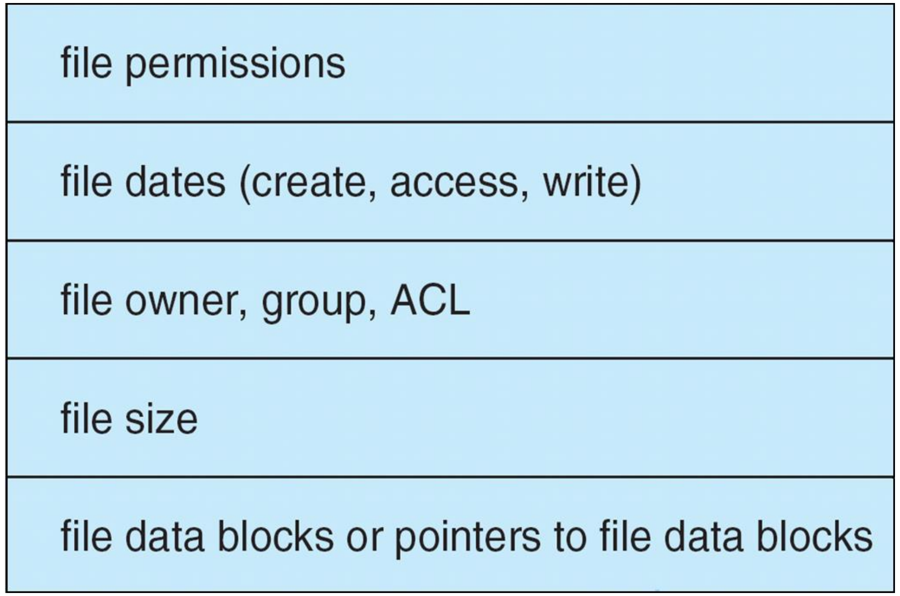
- Many file systems, sometimes many within an operating system
  - Each with its own format:
  - CD-ROM is ISO 9660;
  - Unix has **UFS**, FFS;
  - Windows has FAT, FAT32, NTFS as well as floppy, CD, DVD Blu-ray,
  - Linux has more than 130 types, with **extended file system** ext3 and ext4 leading; plus distributed file systems, etc.)
  - New ones still arriving – ZFS, GoogleFS, Oracle ASM, FUSE

File-System Operations

- We have system calls at the API level, but how do we implement their functions?
  - On-disk and in-memory structures
- **Boot control block** contains info needed by system to boot OS from that volume
  - Needed if volume contains OS, usually first block of volume
- **Volume control block (superblock, master file table)** contains volume details
  - Total # of blocks, # of free blocks, block size, free block pointers or array
- Directory structure organizes the files
  - Names and inode numbers, master file table

File Control Block (FCB)

- OS maintains **FCB** per file, which contains many details about the file
  - Typically, inode number, permissions, size, dates
  - Example



file permissions

file dates (create, access, write)

file owner, group, ACL

file size

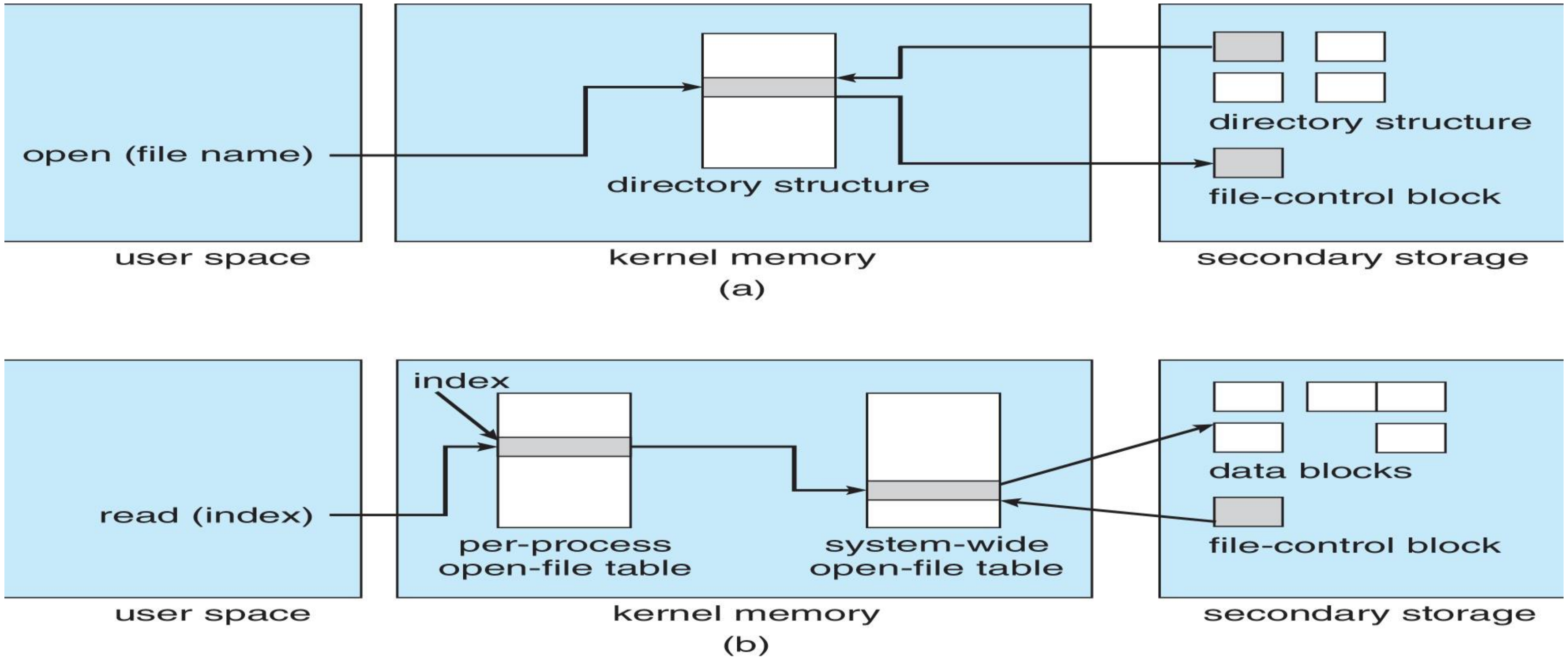
file data blocks or pointers to file data blocks

In-Memory File System Structures

- **Mount table** storing file system mounts, mount points, file system types
- **System-wide open-file table** contains a copy of the FCB of each file and other info
- **Per-process open-file table** contains pointers to appropriate entries in system-wide open-file table as well as other info

In-Memory File System Structures (Cont.)

- Figure 12-3(a) refers to opening a file
- Figure 12-3(b) refers to reading a file



Directory Implementation

- **Linear list** of file names with pointer to the data blocks
  - Simple to program
  - Time-consuming to execute
    - Linear search time
    - Could keep ordered alphabetically via linked list or use B+ tree
- **Hash Table** – linear list with hash data structure
  - Decreases directory search time
  - **Collisions** – situations where two file names hash to the same location
  - Only good if entries are fixed size, or use chained-overflow method

TEXT BOOK

1. Abraham Silberschatz, Peter B. Galvin, “Operating System Concepts”, 10th Edition, John Wiley & Sons, Inc., 2018.

2. Andrew S Tanenbaum, Herbert Bos, Modern Operating systems, Pearson, 5th Edition,2022 New Delhi.

REFERENCES

1. Ramaz Elmasri, A. Gil Carrick, David Levine, “ Operating Systems – A Spiral Approach”, Tata McGraw Hill Edition, 2010.
2. William Stallings, Operating Systems: Internals and Design Principles, 7th Edition, Prentice Hall, 2018
3. Achyut S.Godbole, Atul Kahate, “Operating Systems”, McGraw Hill Education, 2016.

THANK YOU