SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A++’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : OPERATING SYSTEMS

II YEAR/ IV SEMESTER

UNIT – IV FILE SYSTEMS
Topic: File System Implementation

Dr.B.Vinodhini
Associate Professor
Department of Computer Science and Engineering

File system Implementation

SNS COLLEGE OF TECHNOLOGY

(Autonomous)

COIMBATORE-35

File System Structure

File System Implementation

Directory Implementation

Allocation Methods

Free Space Management

Efficiency and Performance

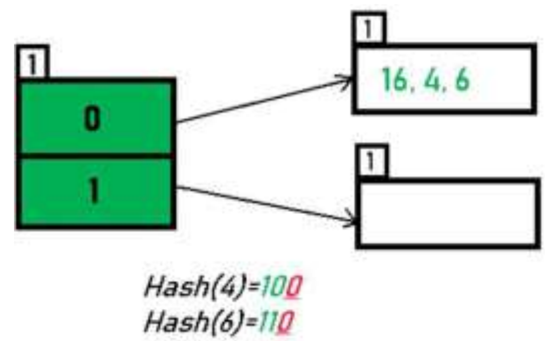
Recovery

File System Structure

File System Implementation

On-disk structures

In-memory structures



Directory Implementation

Linked List

Hash Table

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

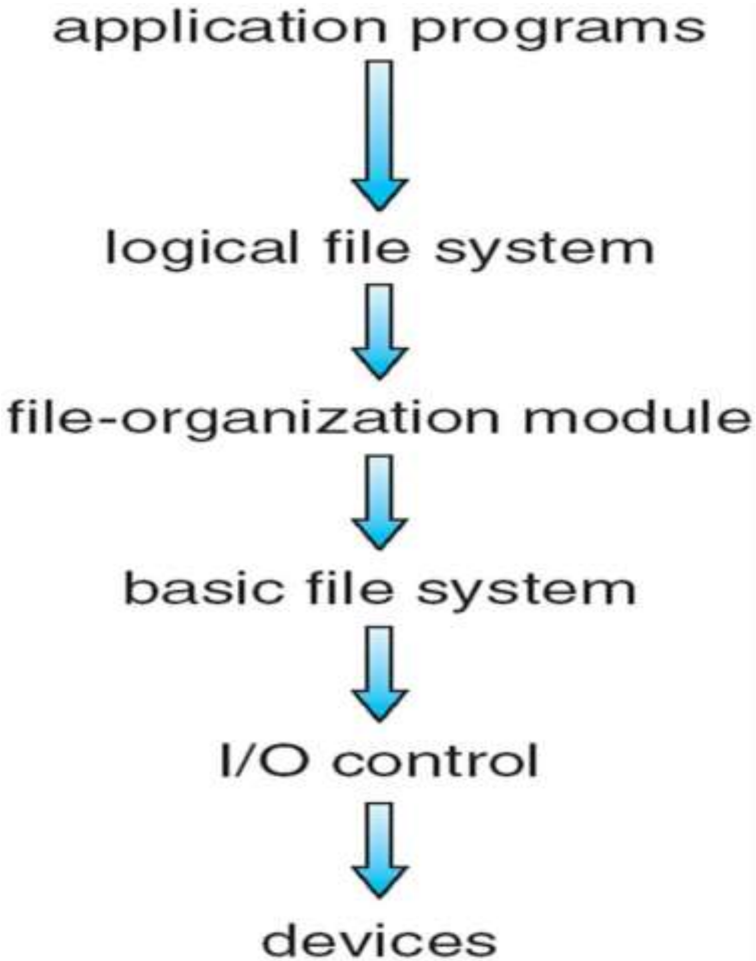
- I/O transfers performed in **blocks** of **sectors** (usually 512 bytes)

File control block – storage structure consisting of information about a file

Device driver controls the physical device

File system organized into layers

File System Structure



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

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

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 Implementation

- 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 System Implementation

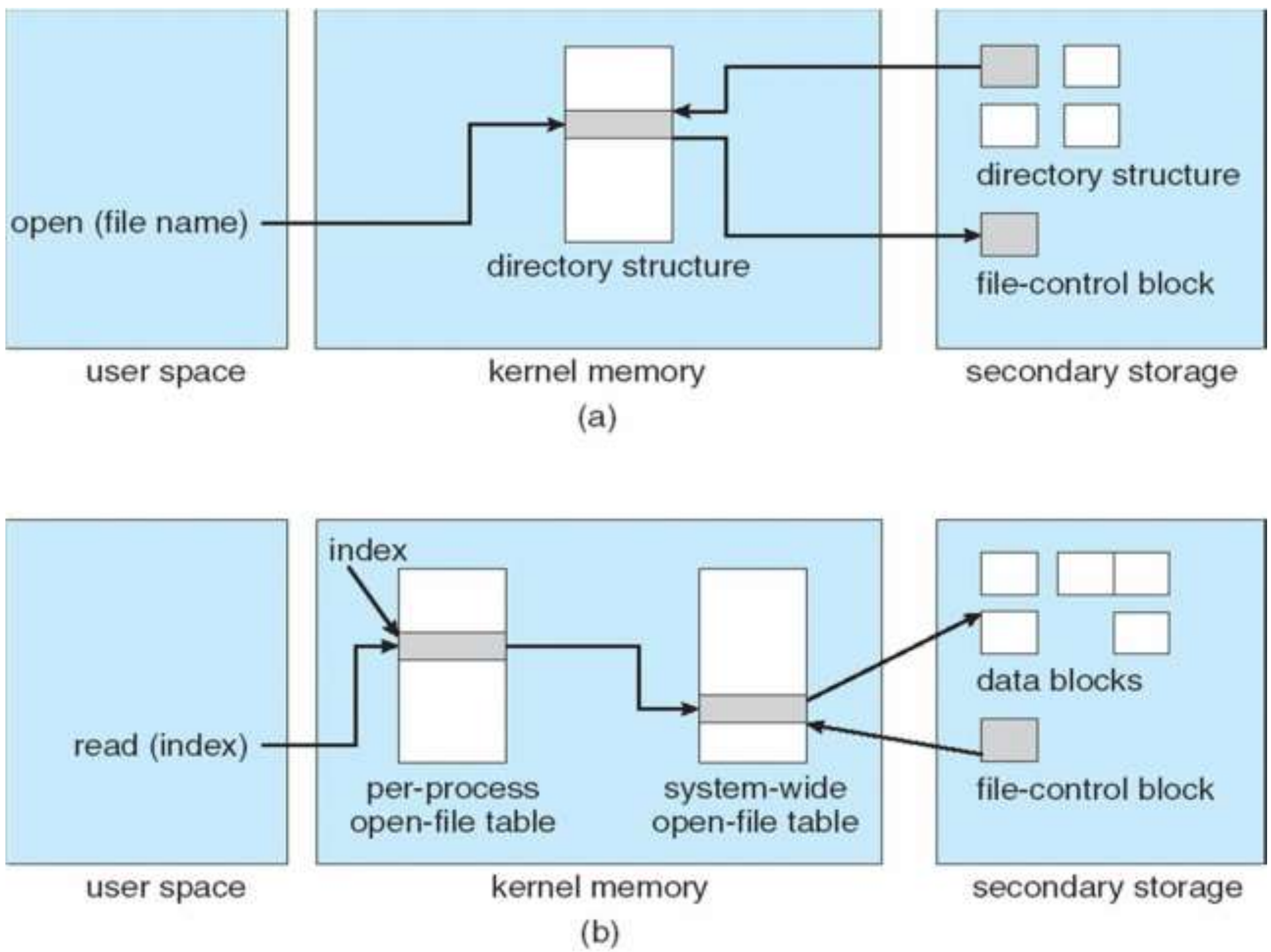
- Per-file **File Control Block** (**FCB**) contains many details about the file
  - inode number, permissions, size, dates
  - NFTS stores into in master file table using relational DB structures

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 Structure

- Mount table storing file system mounts, mount points, file system types
- The following figure illustrates the necessary file system structures provided by the operating systems
- Figure 12-3(a) refers to opening a file
- Figure 12-3(b) refers to reading a file
- Plus buffers hold data blocks from secondary storage
- Open returns a file handle for subsequent use
- Data from read eventually copied to specified user process memory address

In-Memory File System Structures



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.

Summarization

- File Structure
- File System Implementation