Operating Systems - File System Implementation

Mridul Sankar Barik

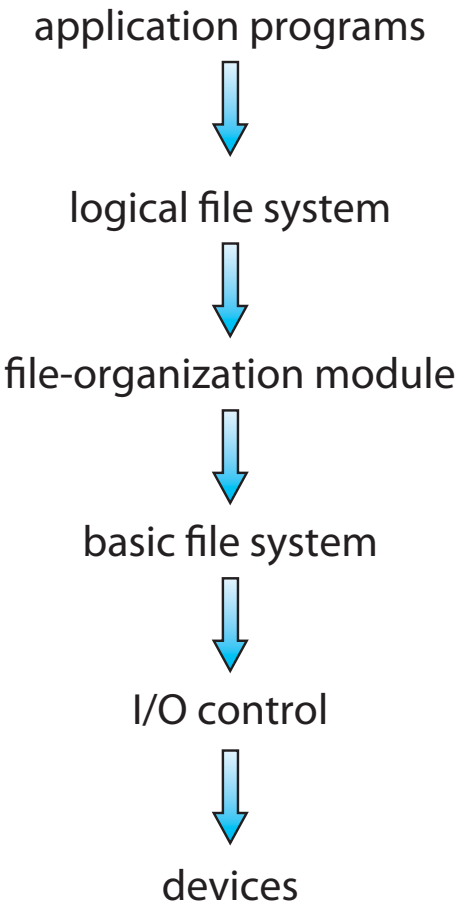
Jadavpur University

2024

File-System Structure

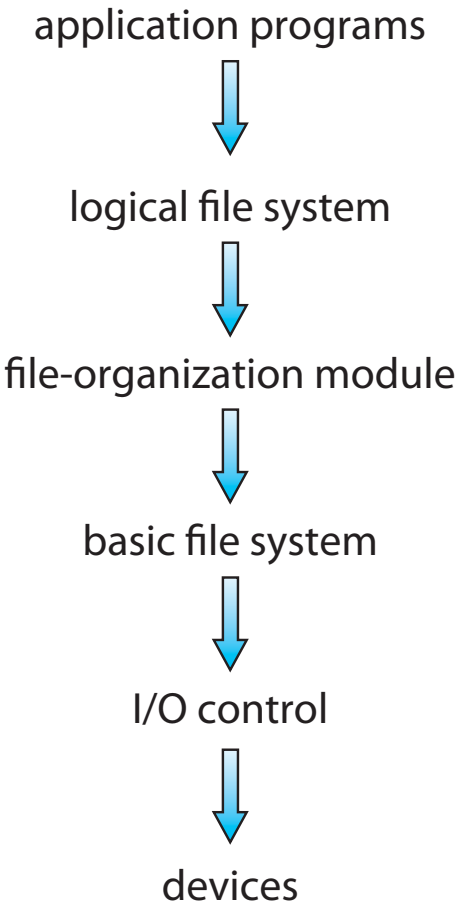
- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
  - Provides user interface to storage, mapping logical to physical
  - Provides efficient and convenient access to disk by allowing data to be stored, located retrieved 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 Layers I



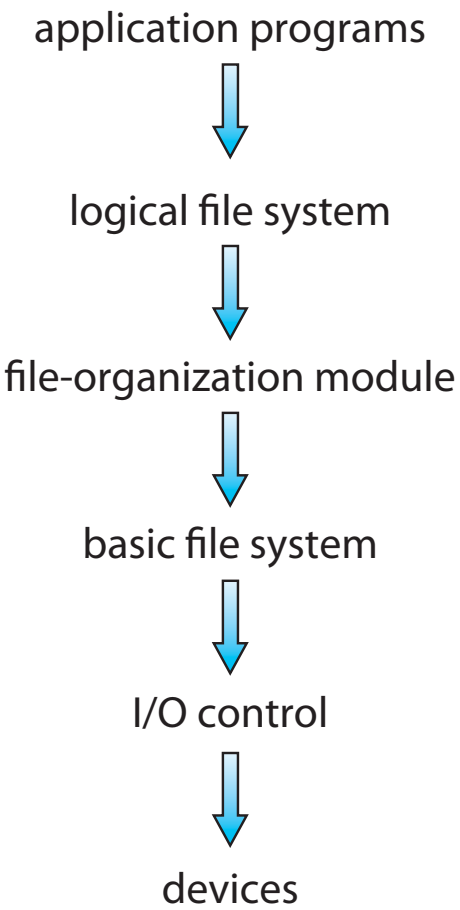
- **Device drivers** manage I/O devices at the **I/O control layer**
  - Given commands like "read drive1, cylinder 72, track 2, sector 10, into memory location 1060" outputs low-level hardware specific commands to hardware controller
- **Basic file system**: given command like "retrieve block 123" translates to device driver
  - Also manages memory buffers and caches (allocation, freeing, replacement)
    - Buffers hold data in transit
    - Caches hold frequently used data

File System Layers II



- **File organization module** understands files, logical address, and physical blocks
  - Translates logical block # to physical block #
  - Manages free space, disk allocation
- **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

File System Layers III

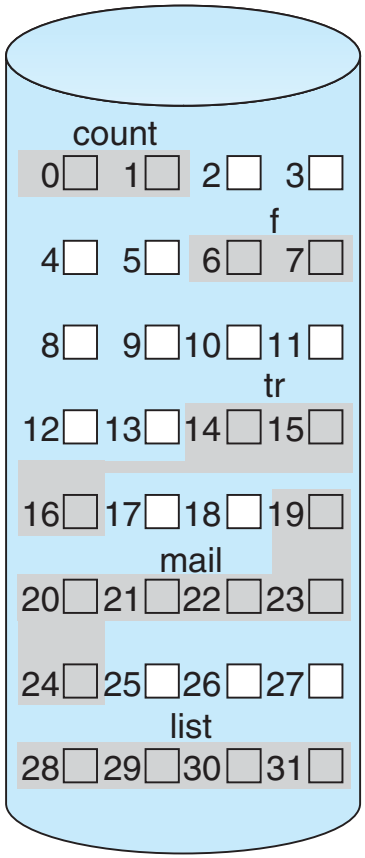


- Many file systems, sometimes many within an operating system
  - Each with its own format
    - CD-ROM is ISO 9660
    - Unix has UFS, FFS
    - Windows has FAT, FAT32, NTFS
    - Linux has more than 40 types, example: ext2 and ext3
    - New ones still arriving – ZFS, GoogleFS, Oracle ASM, FUSE

Allocation Methods

- How to allocate space to files so that disk space is used effectively and files accessed quickly
  - Contiguous Allocation
  - Linked Allocation
  - Indexed Allocation

Contiguous Allocation

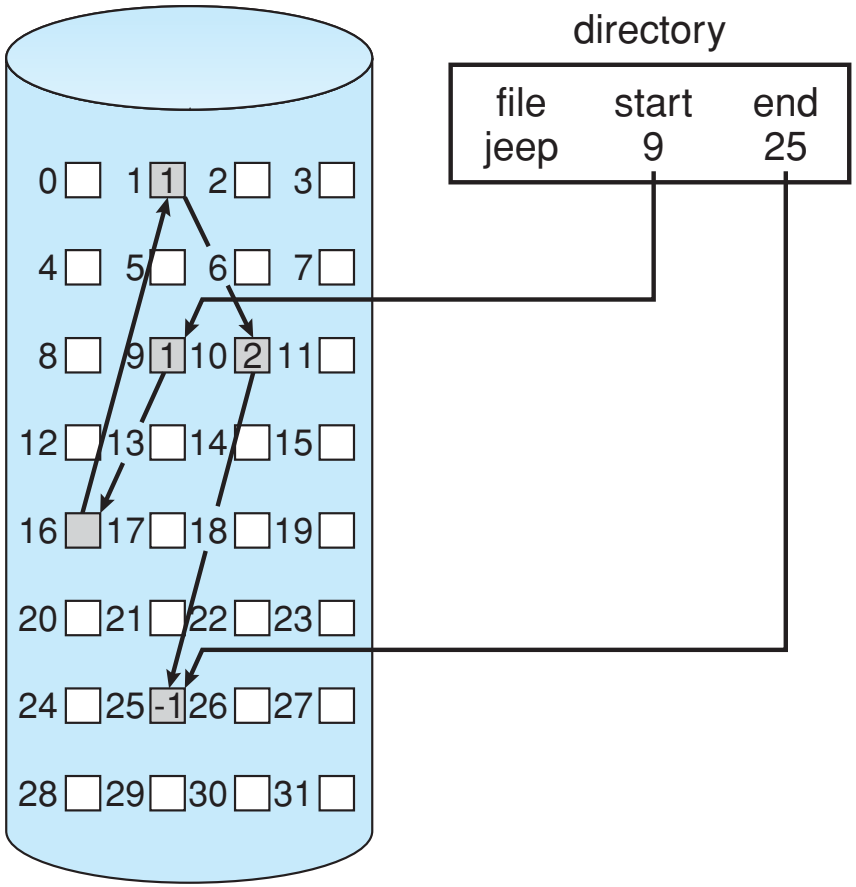


directory

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

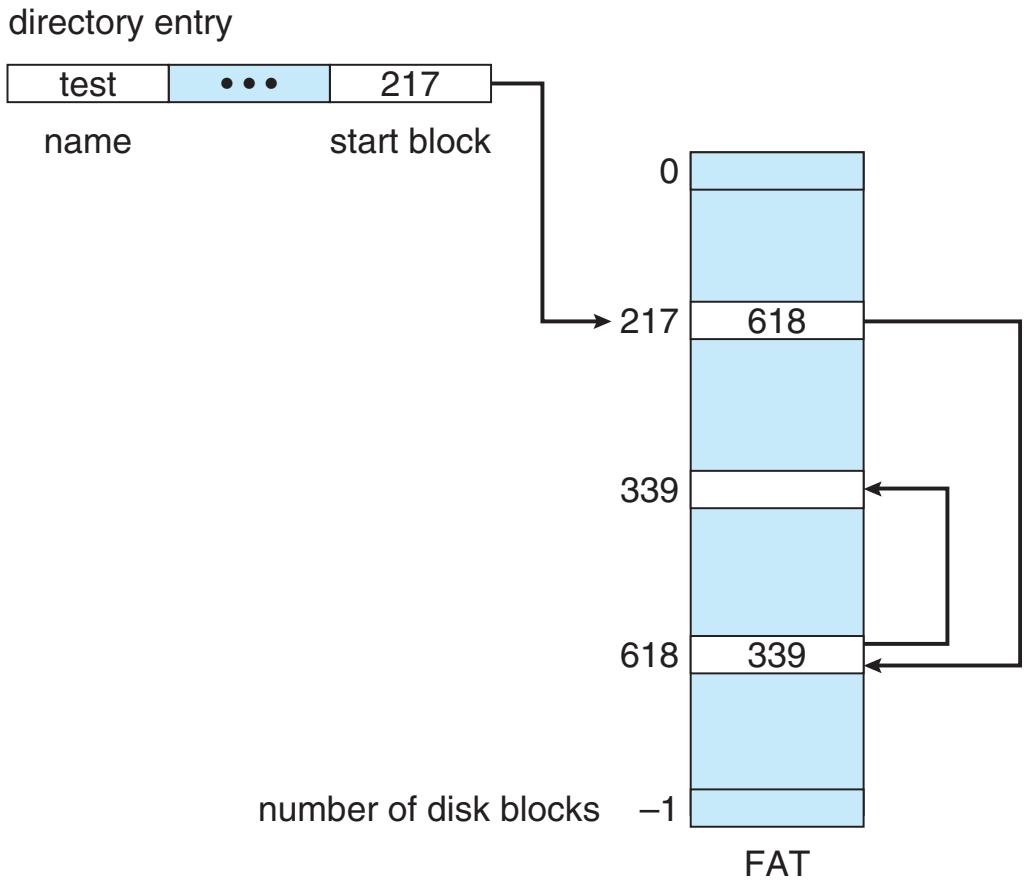
- Each file occupies a set of contiguous blocks on the disk
- Number of disk seeks required for accessing contiguously allocated files is minimal
- Directory entry contains starting location (block #) and length (number of blocks)
- Both sequential and direct access is supported
- Application of dynamic storage-allocation (how to satisfy a request of size n from a list of free holes)
- External fragmentation may occur
  - Solution: Defragmentation
- Files cannot grow
  - Solution: Allocate in contiguous chunks
  - Location of file's blocks are recorded as Location, Block count, Link to the first block of the next chunk

Linked Allocation



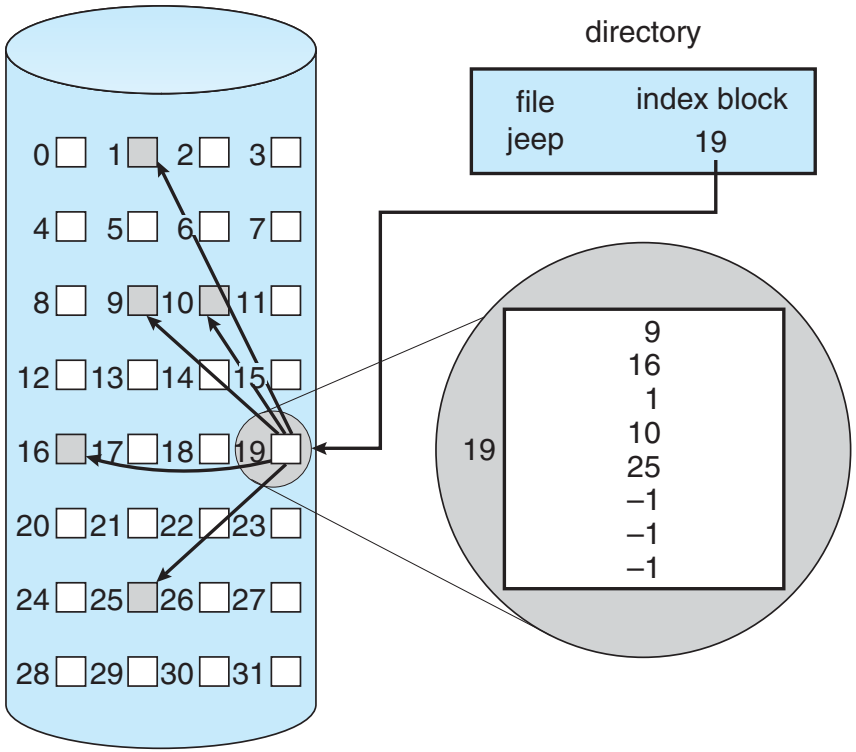
- Each file is a linked list of disk blocks, blocks may be scattered anywhere on the disk
- Directory entry contains a pointer to the first and last blocks of the file
- No external fragmentation
- File size need not be declared during creation
- Can be use only for sequential access file
- Space required for pointers are wasted
- Solutions
  - Allocate space in clusters (ex - 4 blocks)
  - Increases internal fragmentation
- Reliability
  - A bug in OS software or disk hardware failure may cause pointer to be lost or damaged
  - Use doubly linked list
  - Store name of file, relative block # in each block

File Allocation Table



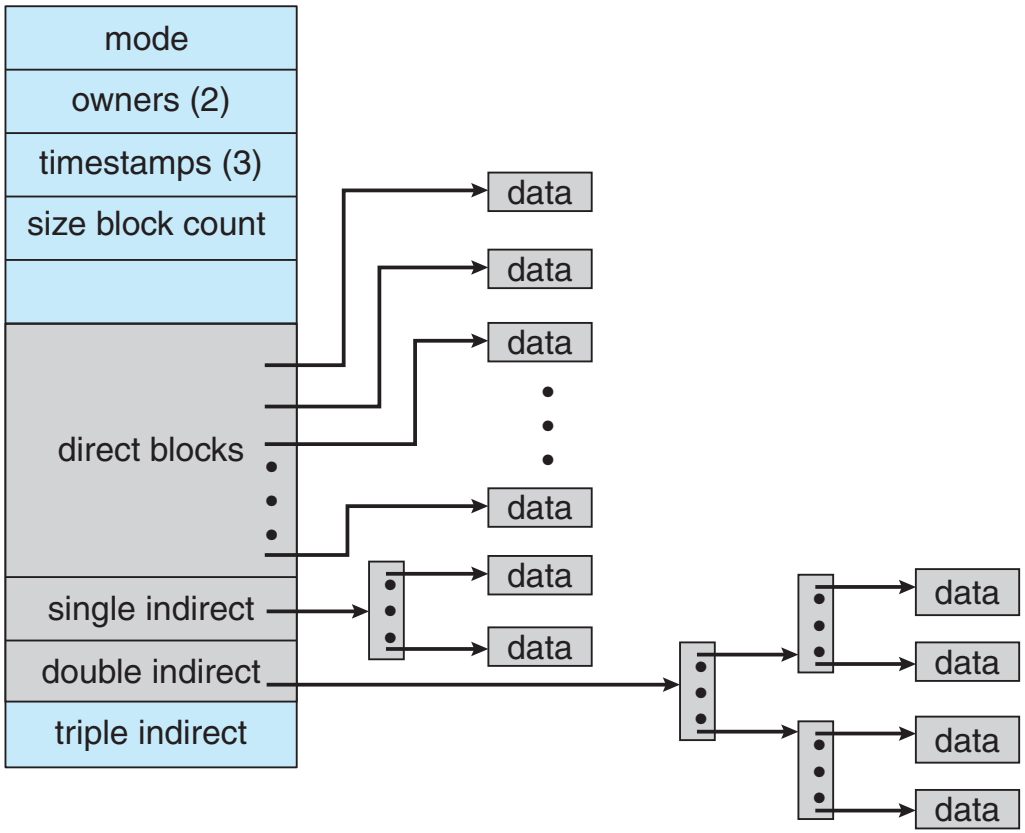
- A section of disk at the beginning of each partition is set aside to contain FAT
- FAT contain
  - One entry for each block; indexed by block number
  - Directory entry contain block number of the first block of the file
  - Last block of a file contain special end of file marker
  - Unused blocks in FAT contain 0 value

Indexed Allocation I



- Brings all pointers together into the index block
- Directory entry contains the address of the index block
- All pointers in the index block are NULL initially
- i^{th} entry in the index block points to the i^{th} block within the file
- Supports direct access
- Pointer overhead is greater than linked allocation

Indexed Allocation II



- How large the index block should be?
- Linked scheme:
  - An index block is normally one disk block
  - For large files, link together several index blocks
- Multilevel index:
  - First level index block points to a set of second level index blocks which contain pointers to the file blocks
- Combined Scheme:
  - Ex: Unix File System