File Systems (2)

Dr. Jun Zheng

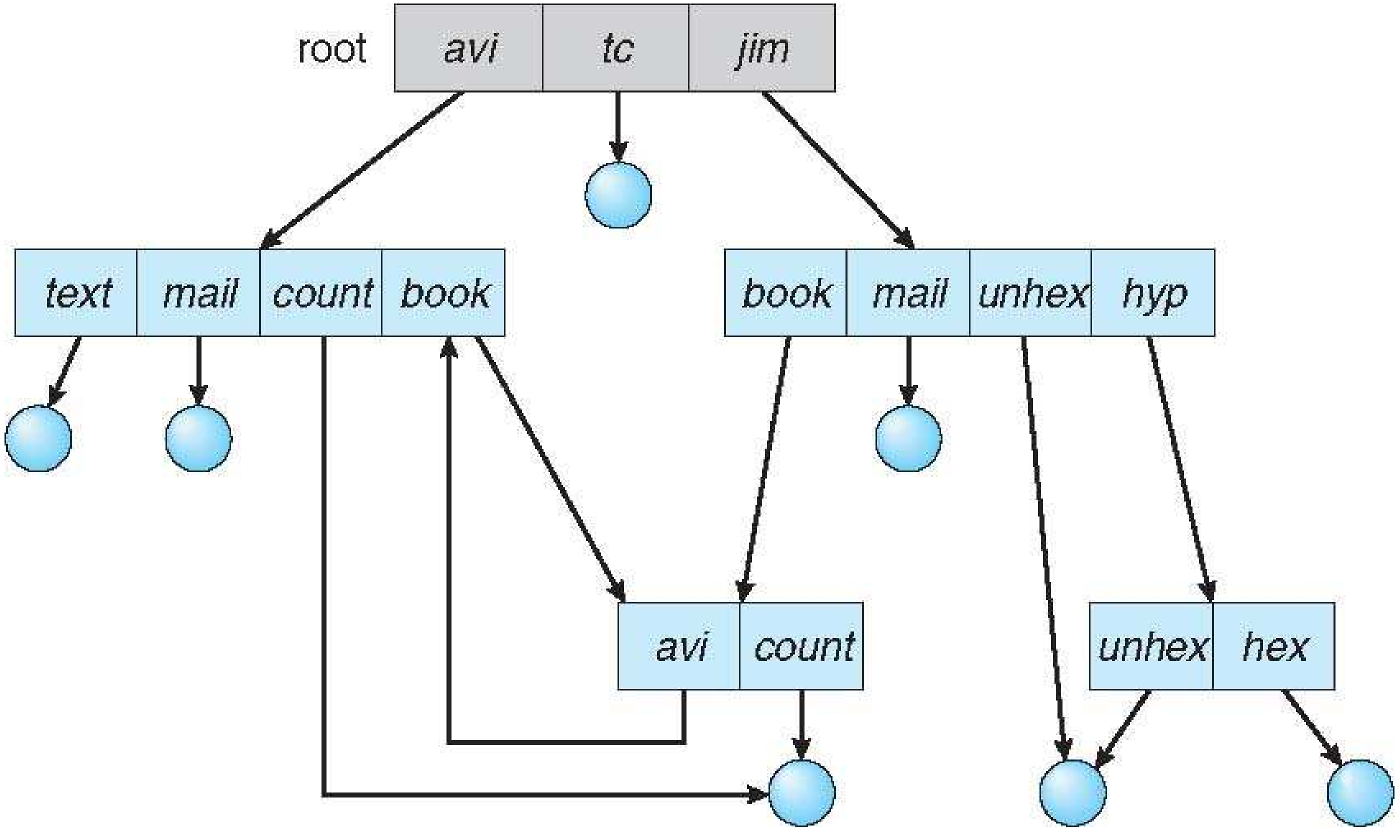
CSE325 Principles of Operating Systems

11/18/2019

NEW MEXICO TECH

SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

General Graph Directory



Cycles?

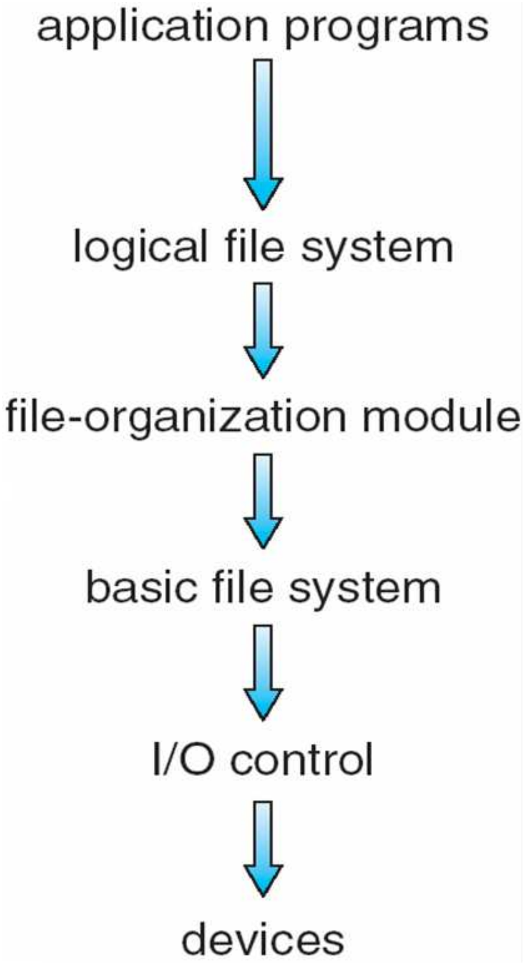
General Graph Directory

- **How do we guarantee no cycles?**
  - Allow only links to file not subdirectories
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

File-System Structure

- **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** (32B – 4,096B, usually 512B)
- **File control block** – 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

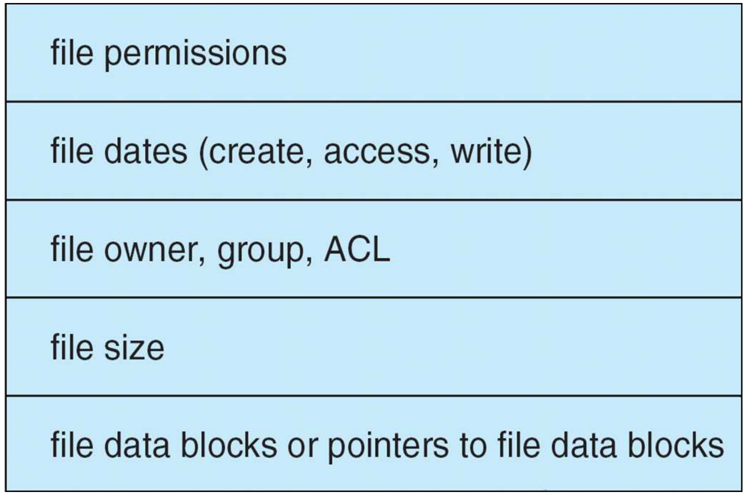
devices

File-System Implementation (1)

- 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** (**UFS: superblock, NTFS: master file table**) contains volume details
  - Total # of blocks, # of free blocks, block size, free block pointers or array
- Directory structure organizes the files
  - UFS: Names and inode numbers, NTFS: master file table

File-System Implementation (2)

- ❑ Per-file **File Control Block** (**FCB**) contains many details about the file
  - ❑ UFS: inode number, permissions, size, dates
  - ❑ NTFS: 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 Structures

- ❑ An in-memory **mount table** contains information about each mounted volume.
- ❑ An in-memory directory-structure cache holds the directory information of recently accessed directories. (For directories at which volumes are mounted, it can contain a pointer to the volume table.)
- ❑ The **system-wide open-file table** contains a copy of the FCB of each open file, as well as other information.
- ❑ The **per-process open-file table** contains a pointer to the appropriate entry in the system-wide open-file table, as well as other information.
- ❑ Buffers hold file-system blocks when they are being read from disk or written to disk.

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 the number of entries is fixed, or use chained-overflow method

Allocation Methods

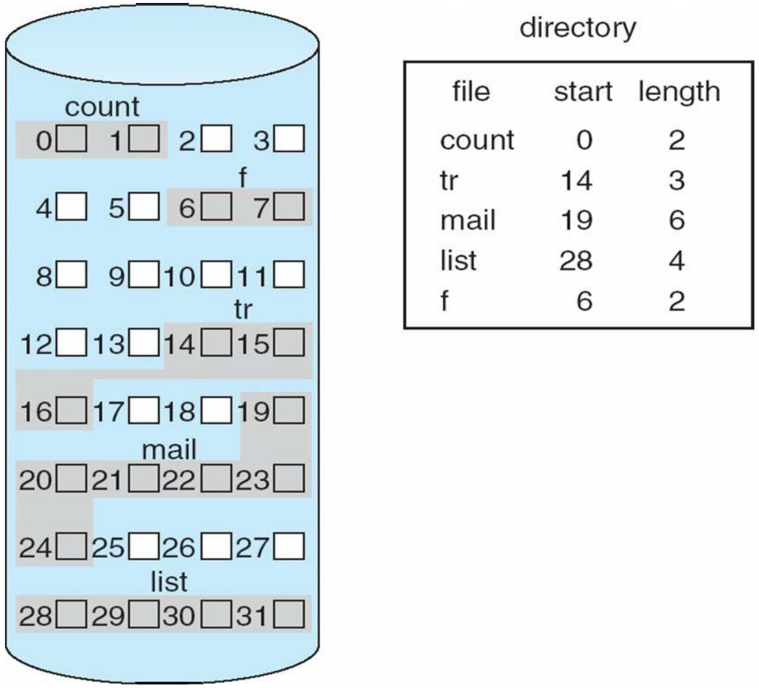
- ❑ An allocation method refers to how disk blocks are allocated for files:
- ❑ Three methods
  - ❑ Contiguous allocation
  - ❑ Linked allocation
  - ❑ Indexed allocation

Contiguous Allocation

- ❑ Each file occupies set of contiguous blocks
- ❑ Best performance in most cases
- ❑ Pros:
    - ❑ Simple – only starting location (block #) and length (number of blocks) are required
    - ❑ Direct access

Contiguous Allocation

- □ Mapping from logical address to physical address



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

Block to be accessed = Q + starting address

Displacement into block = R

Cons

- □ Waste of space
- □ Difficult to support dynamic file sizes (files cannot grow)