CSC 4103 - Operating Systems
Fall 2009

Lecture - XX
File Systems

Tevfik Koşar

Louisiana State University

November 5th, 2009

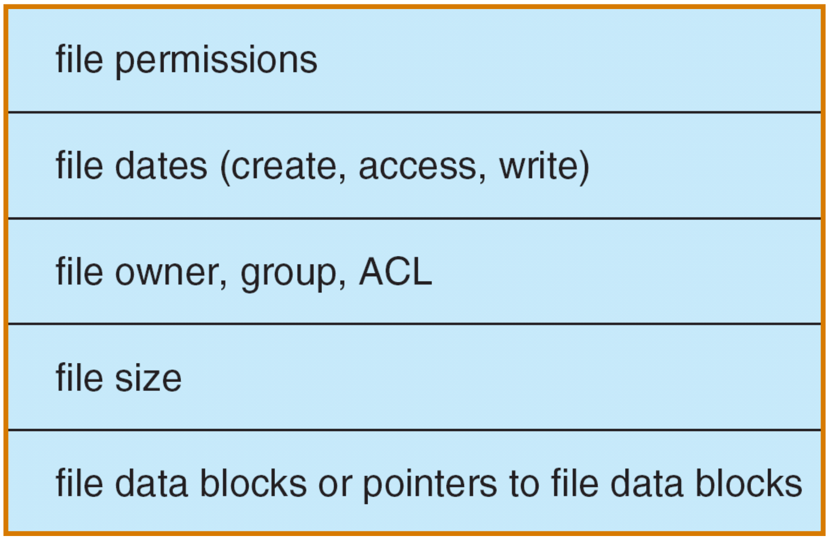
File Systems

- Provides organized and efficient access to data on secondary storage:

    1. Organizing data into files and directories and supporting primitives to manipulate them (create, delete, read, write etc)
    2. Improve I/O efficiency between disk and memory (perform I/O in units of blocks rather than bytes)
    3. Ensure confidentiality and integrity of data

    - Contains file structure via a File Control Block (FCB)
        - Ownership, permissions, location..

A Typical File Control Block



file permissions

file dates (create, access, write)

file owner, group, ACL

file size

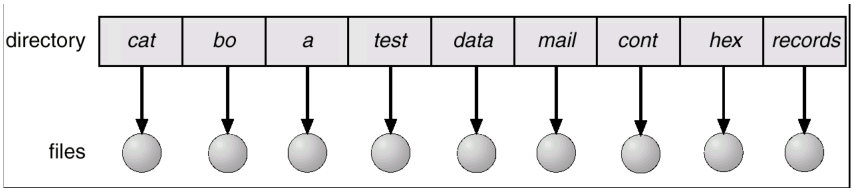
file data blocks or pointers to file data blocks

Directories

- Directories are special files that keep track of other files
  - ✓ the collection of files is systematically organized
  - ✓ first, disks are split into partitions that create logical volumes (can be thought of as “virtual disks”)
  - ✓ second, each partition contains information about the files within
  - ✓ this information is kept in entries in a **device directory** (or volume table of contents)
  - ✓ the directory is a symbol table that translates file names into their entries in the directory
    - it has a logical structure
    - it has an implementation structure (linked list, table, etc.)

Directories

- ➢ Single-level directory structure
  - ✓ simplest form of logical organization: one global or **root** directory containing all the files
  - ✓ problems
    - global namespace: unpractical in multiuser systems
    - no systematic organization, no groups or logical categories of files that belong together

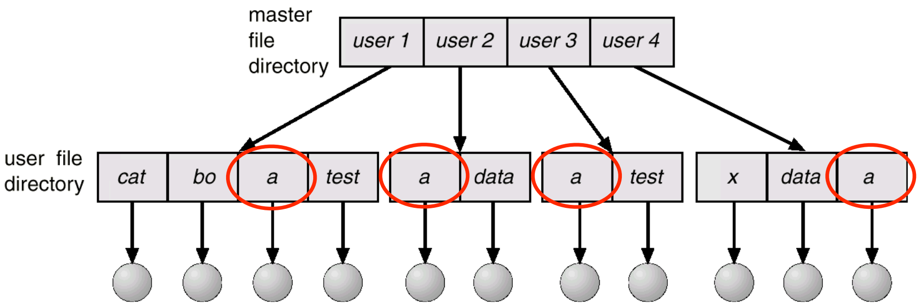


Single-level directory

Silberschatz, A., Galvin, P. B. and Gagne. G. (2003)
Operating Systems Concepts with Java (6th Edition).

Directories

- ➢ Two-level directory structure
  - ✓ in multiuser systems, the next step is to give each user their own private directory
  - ✓ avoids filename confusion
  - ✓ however, still no grouping: not satisfactory for users with many files

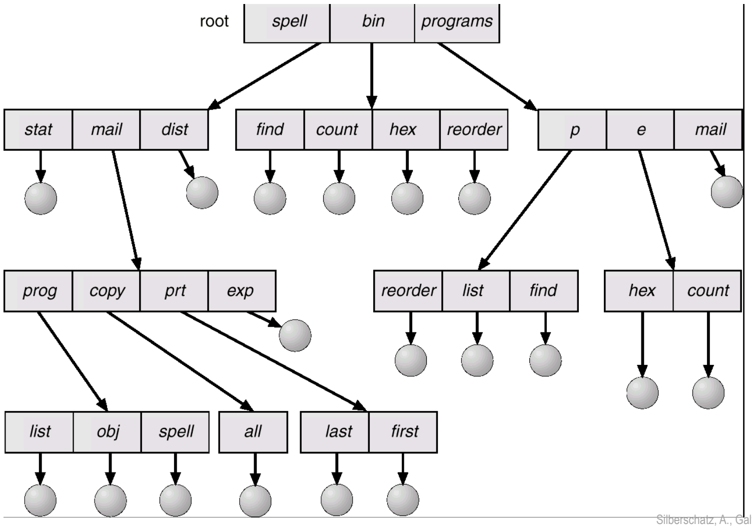


Two-level directory

Silberschatz, A., Galvin, P. B. and Gagne. G. (2003)
Operating Systems Concepts with Java (6th Edition).

Directories

- Tree-structured directory structure



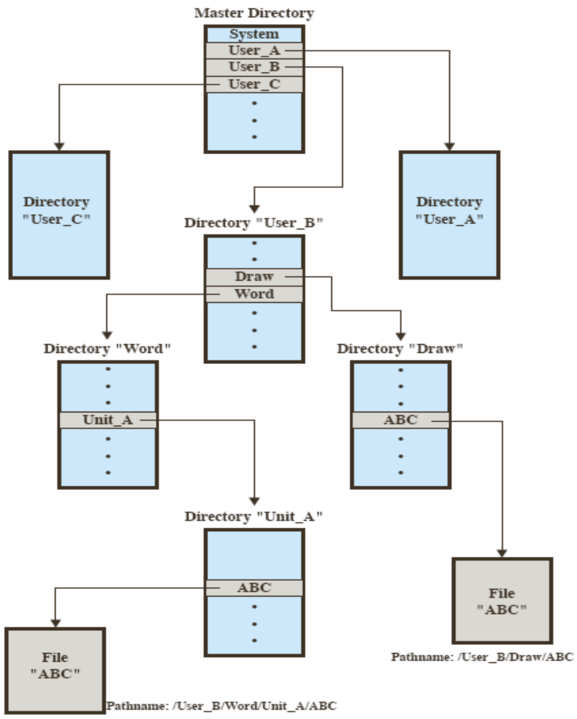
Silberschatz, A., Galvin, P. B. and Gagne. G. (2003)
Operating Systems Concepts with Java (6th Edition).

Tree-structured directory

Directories

- Tree-structured directory structure
  - ✓ natural extension of the two-level scheme
  - ✓ provides a general hierarchy, in which files can be grouped in natural ways
  - ✓ good match with human cognitive organization: tendency to categorize objects in embedded sets and subsets
  - ✓ navigation through the tree relies on **pathnames**
    - absolute pathnames start from the root, example: /jsmith/academic/teaching/cs446/assignment4/grades
    - relative pathnames start at from a current **working directory**, example: assignment4/grades
    - the current and parent directory are referred to as . and ..

Directory Implementation



Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

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

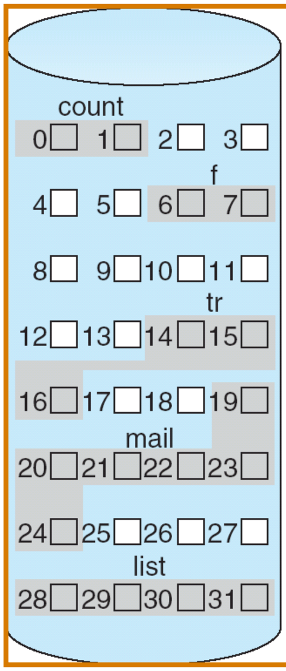
Allocation Methods

- 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
- - Wasteful of space (dynamic storage-allocation problem - fragmentation)
- - Files cannot grow

Contiguous Allocation of Disk Space

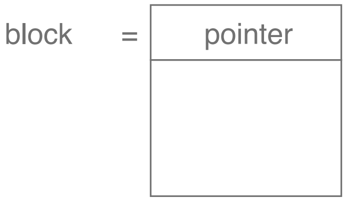


directory

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

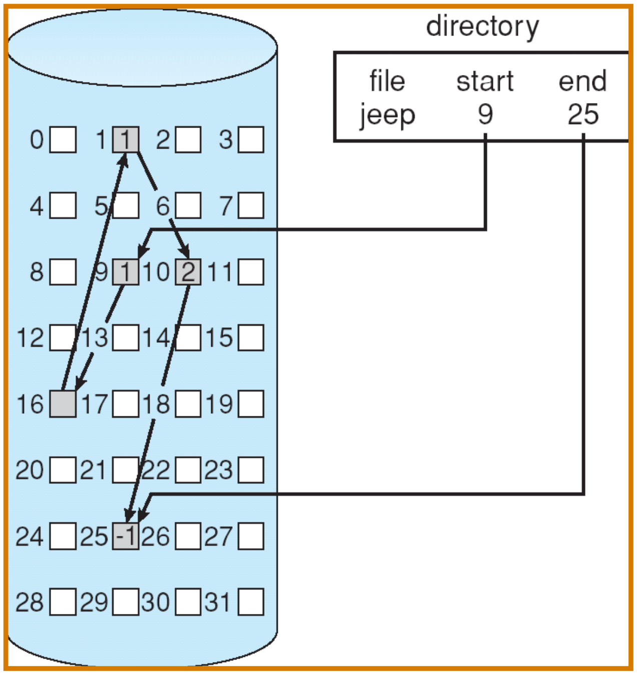
Linked Allocation

- 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
+ Defragmentation not necessary
- No random access
- Extra space required for pointers
- Reliability: what if a pointer gets corrupted?

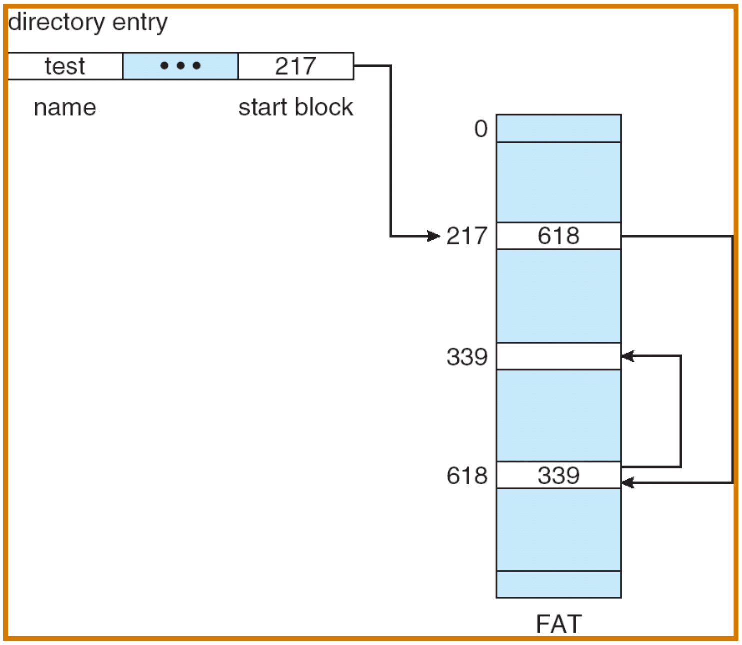
Linked Allocation



directory

file	start	end
jeep	9	25

File-Allocation Table

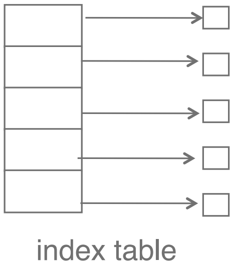


directory entry

test	• • •	217
name		start block

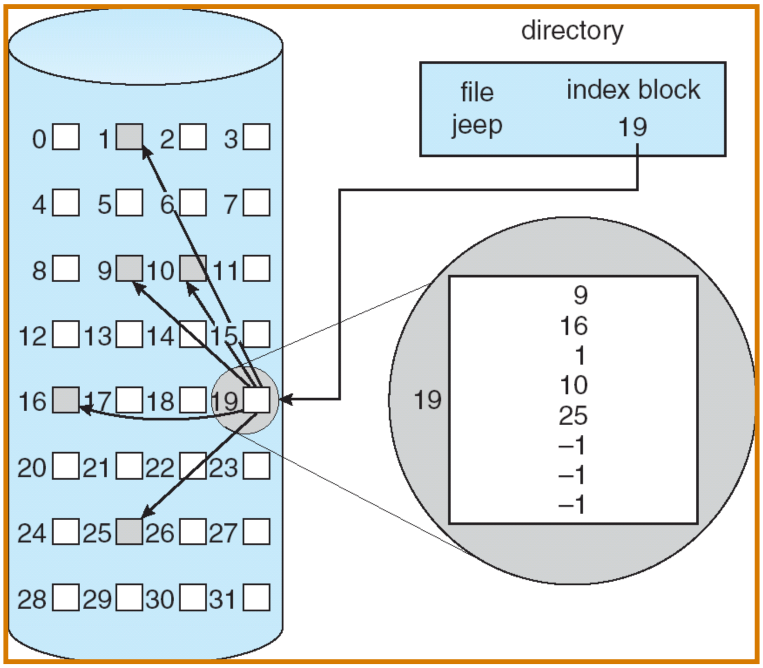
Indexed Allocation

- Brings all pointers together into the *index block*, to allow random access to file blocks.
- Logical view.



+ Supports direct access
+ Prevents external fragmentation
- High pointer overhead --> wasted space

Example of Indexed Allocation

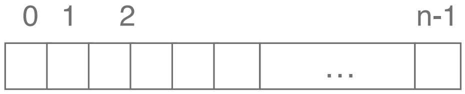


Free Space Management

- Disk space limited
- Need to re-use the space from deleted files
- To keep track of free disk space, the system maintains a free-space list
  - Records all free disk blocks
- Implemented using
  - Bit vectors
  - Linked lists

Free-Space Management (Cont.)

- Bit vector (*n* blocks)

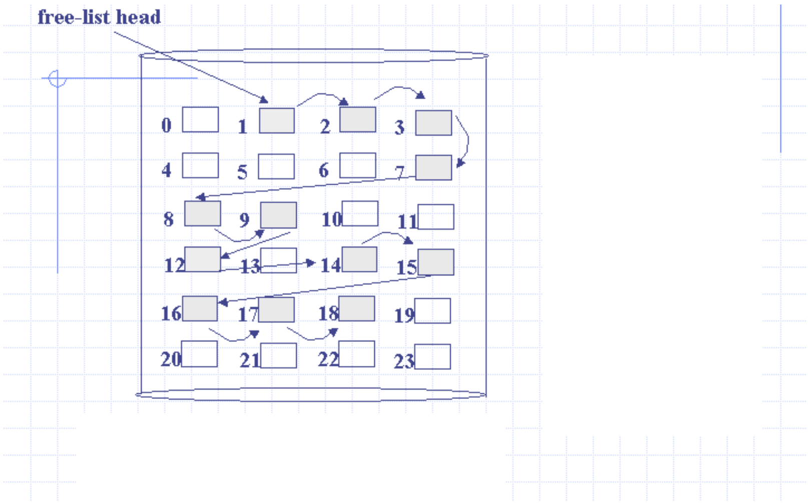


$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ free} \\ 1 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

- e.g. 000011111000100010001000

Free-Space Management (Cont.)

- Linked List



Free-Space Management (Cont.)

- Bit map requires extra space
  - Example:

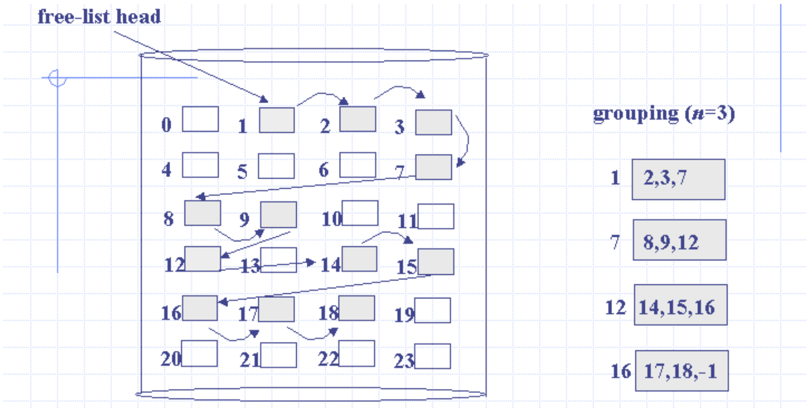
    block size = 2^{12} bytes

    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
  - requires substantial I/O
- Grouping
  - Modification of free-list
  - Store addresses of n free blocks in the first free block
- Counting
  - Rather than keeping list of n free addresses:
    - Keep the address of the first free block
    - And the number n of free contiguous blocks that follow it

Free-Space Management (Cont.)

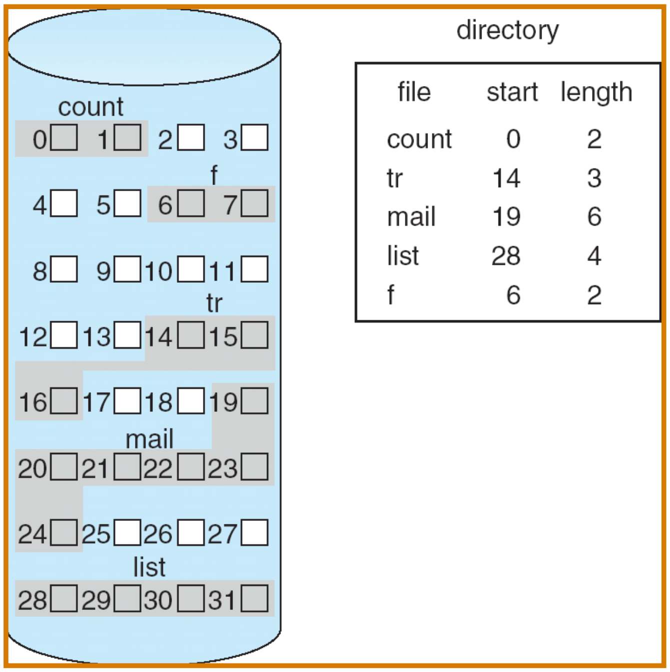
- Linked List



bit map: 011100011100101111100000

counting: (1,3), (7, 3), (12, 1), (14, 5)

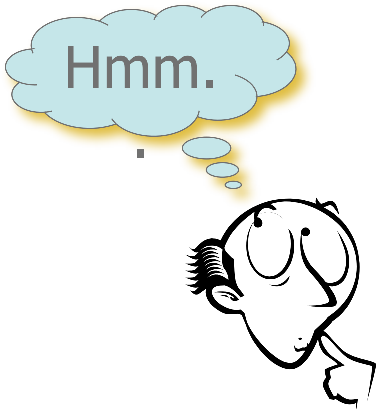
Exercise



directory

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

Any Questions?



Acknowledgements

- “Operating Systems Concepts” book and supplementary material by A. Silberschatz, P. Galvin and G. Gagne
- “Operating Systems: Internals and Design Principles” book and supplementary material by W. Stallings
- “Modern Operating Systems” book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR