CS 4410
Operating Systems

File-System: Implementation

Summer 2013

Cornell University

Today

- How is the file system implemented?
- Layered file system
- Directory implementation
- Allocation methods
- Free-space management

File Systems

- Two design problems:
  - User's interface:
    - Definition of a file, attributes, operations allowed, directory for organization.
  - Hardware interface:
    - Create algorithms and data structures.
    - Map logical file system onto the physical storage device.
- To simplify the design of a file system, several intermediate layers are implemented.

Layered file system

- Application programs
- Logical file system
  - Directory structure
  - File structure via file-control blocks (**FCB**)
  - Metadata; not actual data
- File-organization module
  - Translate logical **block** addresses to physical block addresses.
  - Free-space manager
- Basic file system
  - Issue generic commands to device driver.
  - Read and write physical blocks.
  - Manage memory buffers and caches.
- I/O control
  - Device drivers with interrupt handlers.
  - Transfer information between the main memory and the disk.
  - Write to device controller's memory the location and the action of interest.

File permitions
File dates
File owner
File size
File data blocks

FCB

Example: open()

- The **open()** call passes a **file name** to the **logical file system**.
- The **system-wide open-file table** is searched.
- If the file name is **found**, the new per-process open-file table entry points to the corresponding entry of The system-wide open-file table.
- **Else**, the **directory** is searched for the file name.
- Once it is found, its **FCB** is copied into a new entry of the **system-wide open-file table**.
- And, a new entry is created at the **per-process open-file table,** pointing to the entry above.
- In both cases, the open() call **returns the pointer** of the new entry in the per-process open-file table.

Directory Implementation

- The main function of the directory system is to **map** the **ASCII name** of the file **onto** the information needed to **locate** the data.
- Every time we open a file, which has not been opened yet in the system, we **find its directory** and **search its entry**.
- How are we going to implement the directory structure?

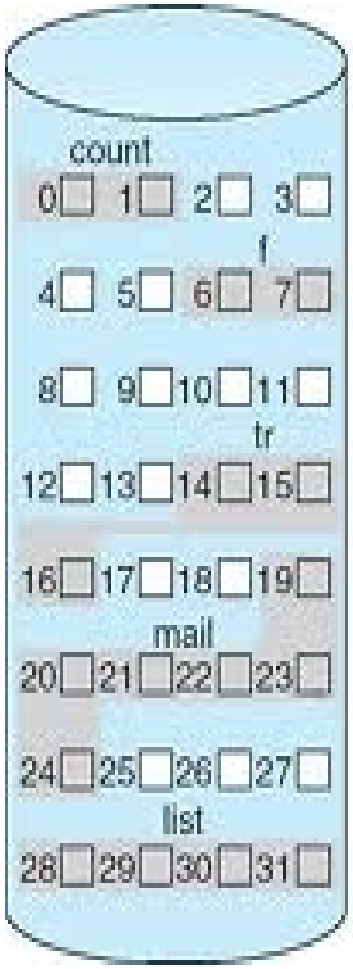
Directory Implementation

- Linear List
  - List of file names with pointers to the data blocks.
  - Simple
  - Time consuming execution:
    - Create, search, delete.
- Hash Table
  - Takes a value computed from the file name.
  - Returns a pointer to the linear list.
  - Simple insertion, deletion.
  - Attention with the collisions.

Allocation Methods

- Remember: Data is saved as **blocks** in the hard disk.
- How do we allocate space (blocks) to the files so that:
  - Disk space is utilized effectively.
  - Files are accessed quickly.

Contiguous Allocation



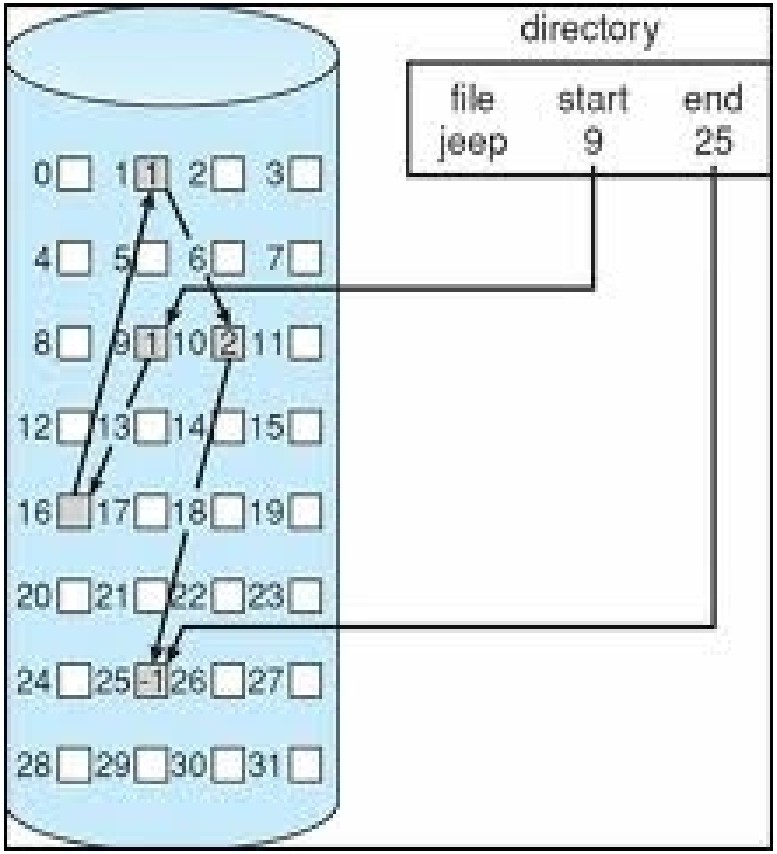
directory

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

Contiguous Allocation

- Each file occupy a set of **contiguous blocks** on the disk.
- **Minimal disk seeks** and **seek time**.
- The **directory entry** for each file indicates the address of the **starting block** and the **number of blocks used**.
- It supports both sequential and direct access.
- Difficulty in finding free space.
  - Dynamic storage-allocation problem
  - External fragmentation
    - Solution: Compaction
  - Determine space needed for a file.

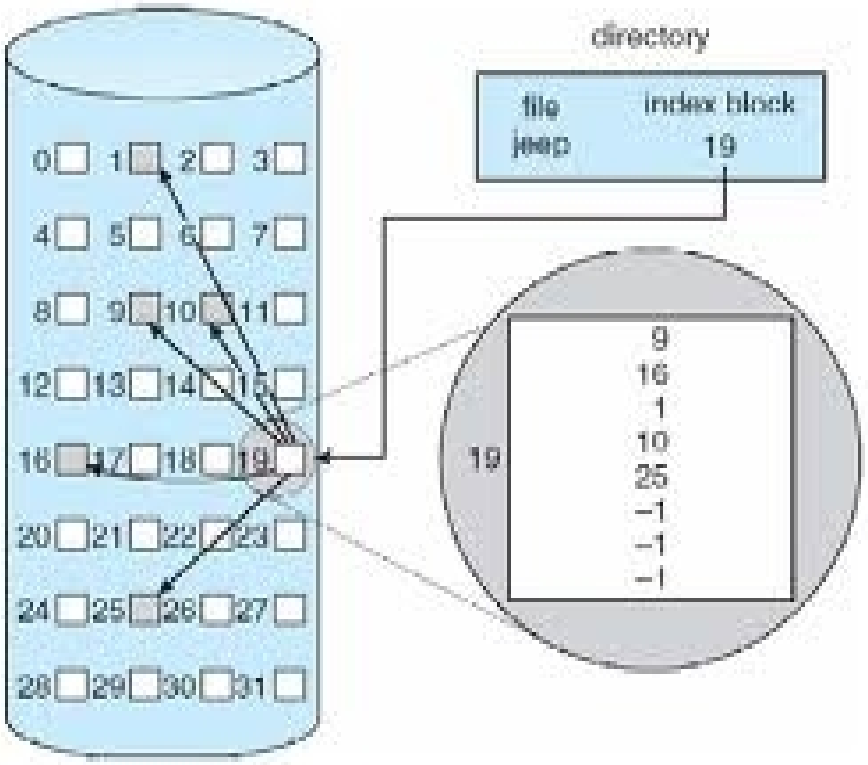
Linked Allocation



Linked Allocation

- A file is a **linked list of disk blocks**.
- The **directory entry** contains a **pointer to the first** and **last** blocks.
- Each block contains a **pointer to the next** block.
  - They consume space.
- Easy block allocation.
- Effective only for sequentially-access files.
- Solution: Allocate clusters rater than blocks.
- Another Problem: Reliability

Indexed Allocation



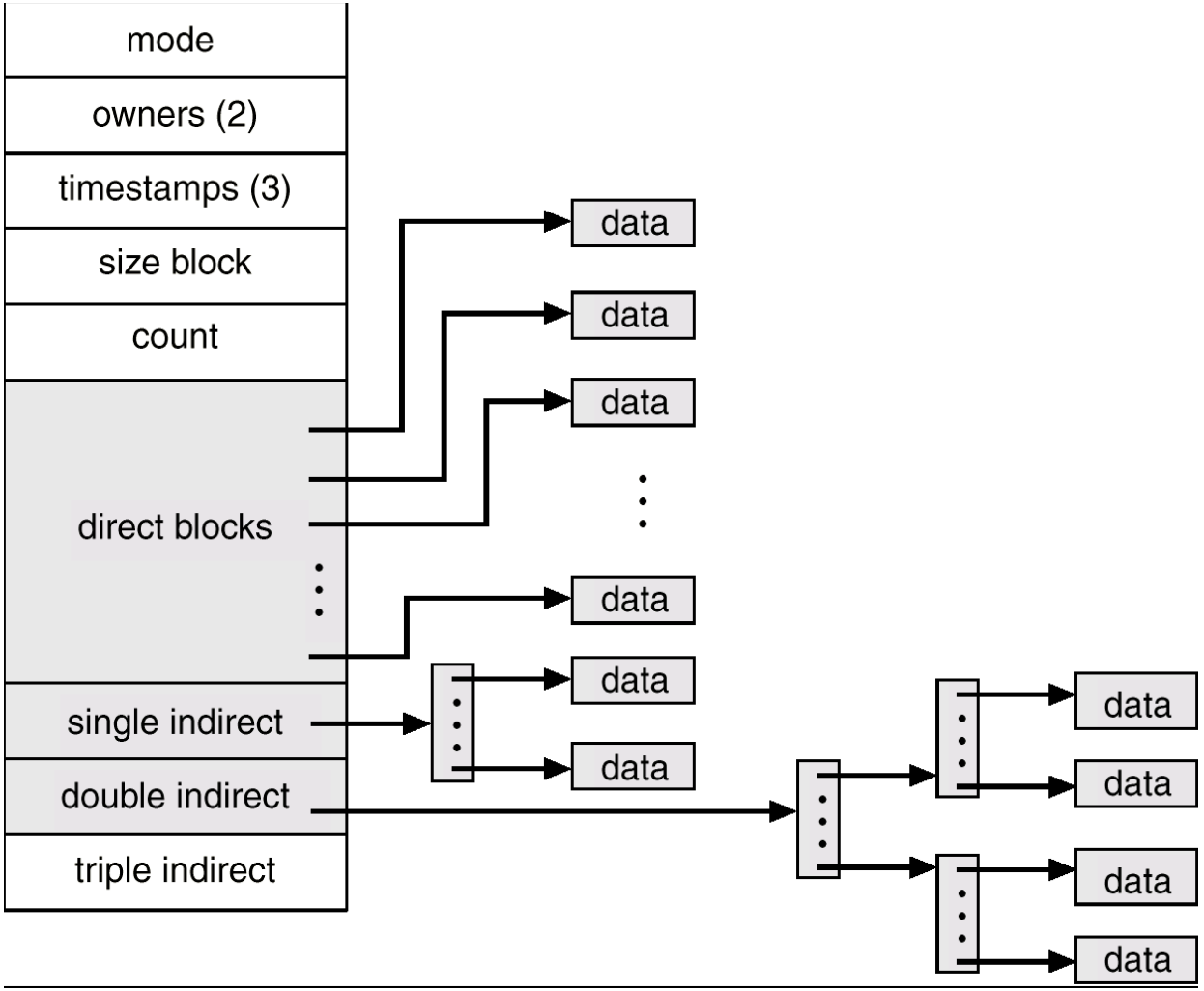
Indexed Allocation

- It keeps the advantages of the Linked Allocation (no external fragmentation,flexible size-declaration).
- It supports efficient **direct access**.
- Bring all the pointers together into the **index block**.
- The **directory entry** contains the address of the **index block**.
- It suffers from wasted space.
- How large should the index block be?
- What happens if the pointers do not fit in one block?

Indexed Allocation

- Linked scheme
- Multilevel index
- Combined scheme
  - Used in UFS
  - The directory entry has a pointer to the file's inode.
    - inode = structure that saves the FCB + 15 pointers.
  - 12 pointers point to direct blocks.
  - 1 pointer points to single indirect block.
  - 1 pointer points to double indirect block.
  - 1 pointer points to triple indirect block.

The Unix inode



Free-Space Management

- Bit Vector
- Linked List
- Grouping
- Counting

Today

- How is the file system implemented?
- Layered file system
- Directory implementation
- Allocation methods
- Free-space management