CSC 4103 - Operating Systems
Spring 2008

LECTURE - XVIII

# FILE SYSTEMS

Tevfik Koşar

Louisiana State University

April 8th, 2008

## File-System Structure

- Provides organized and efficient access to data on secondary storage, E.g.:
  - Organizing data into files and directories
  - Improve I/O efficiency between disk and memory (perform I/O in units of blocks rather than bytes)
  - Contains file structure via a File Control Block (FCB)
    - Ownership, permissions, location..

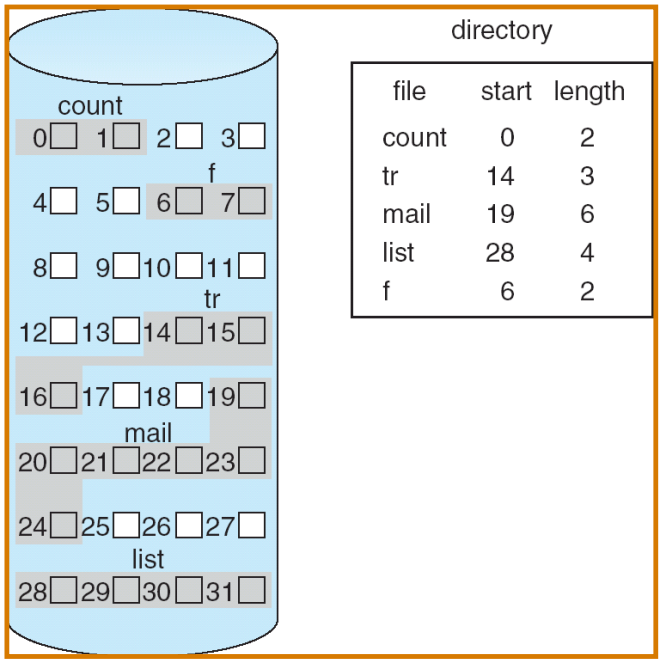
## Allocation Methods

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

## Contiguous Allocation

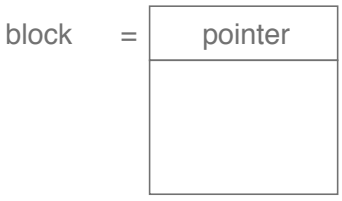
- Each file occupies a set of contiguous blocks on the disk
- Simple – only starting location (block #) and length (number of blocks) are required
- Wasteful of space (dynamic storage-allocation problem)
- Files cannot grow

# Contiguous Allocation of Disk Space



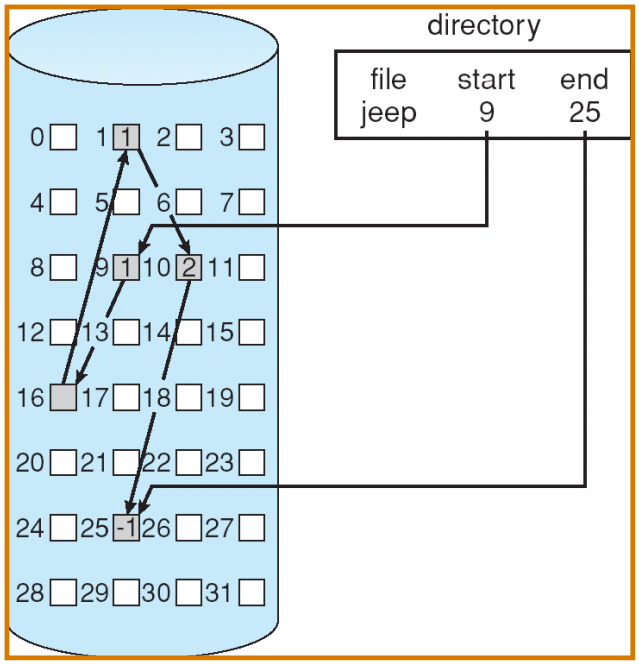

# Linked Allocation

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.

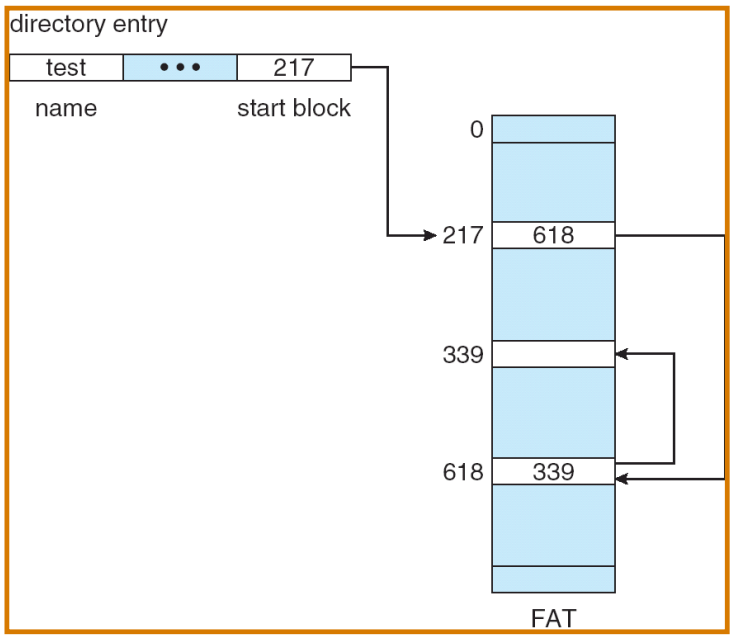



+ Simple – need only starting address
+ Free-space management system – no waste of space
+ Defragmentation not necessary
- No random access
- Extra space required for pointers
- Reliability: what if a pointer gets corrupted?

# Linked Allocation

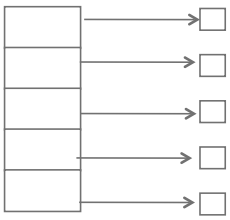



# File-Allocation Table



# Indexed Allocation

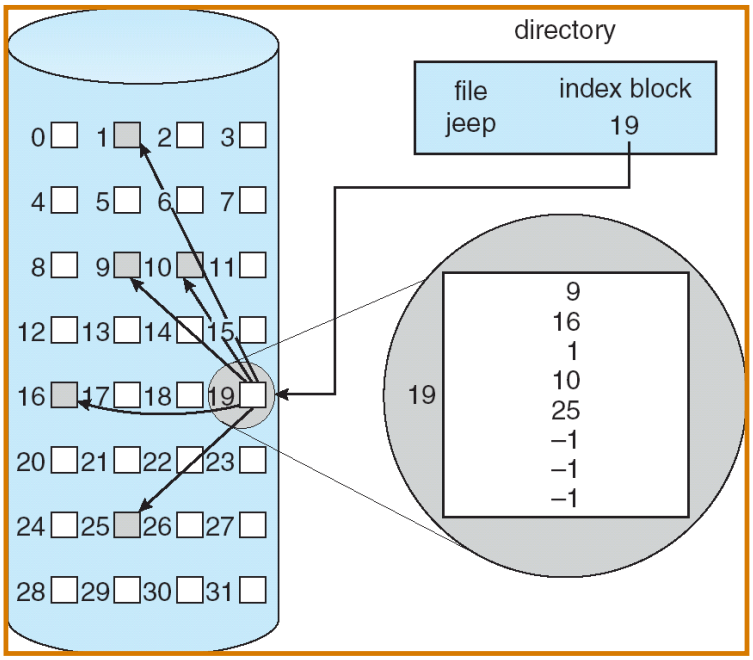
- Brings all pointers together into the *index block*, to allow random access to file blocks.
- Logical view.



index table

+ 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.)

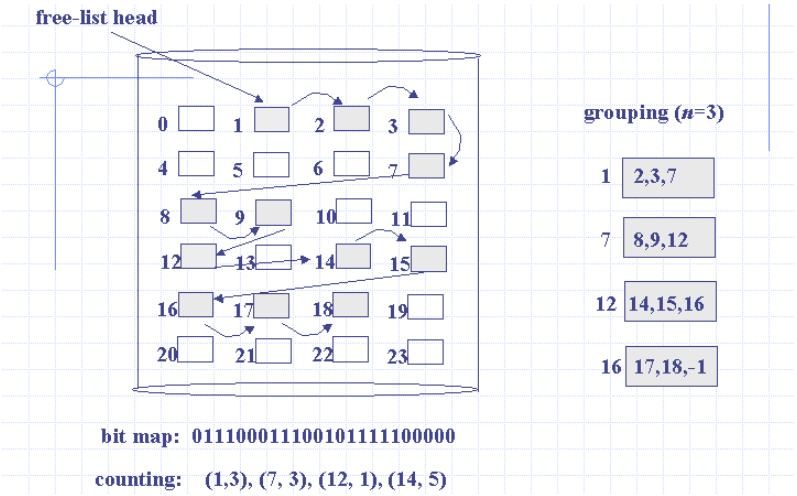
- Bit vector (*n* blocks)


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

- e.g. 000011111000100010000

# 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

# 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