Operating Systems

Lecture 9a

Dr Ronald Grau School of Engineering and Informatics Spring term 2018

Previously

Storage management

- Persistent storage
- Formats, access, operations
- File attributes & permissions

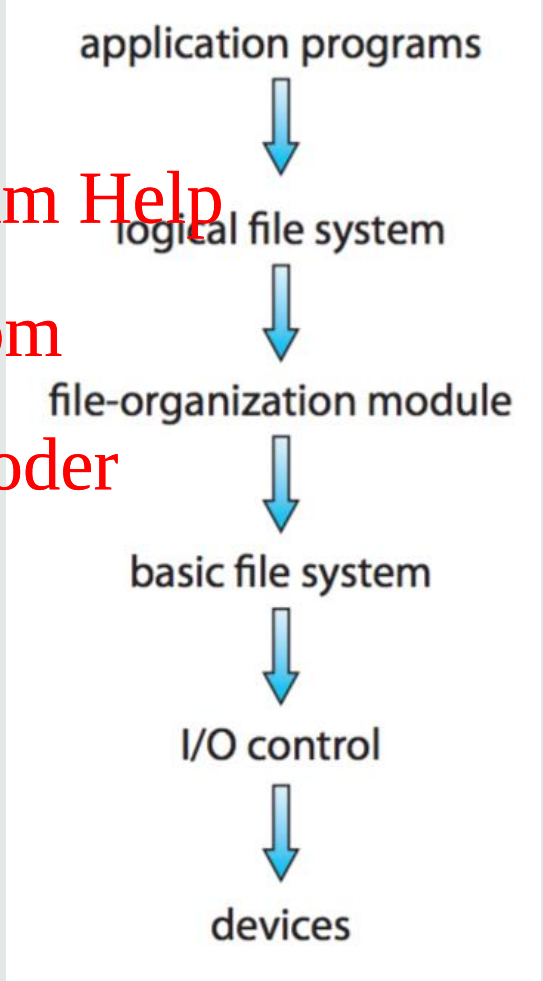
Today

Storage management

- File systems implementation
  - Implementation
  - Layout
  - Kernel data structures
- Block allocation
- Logical file system

File system implementation

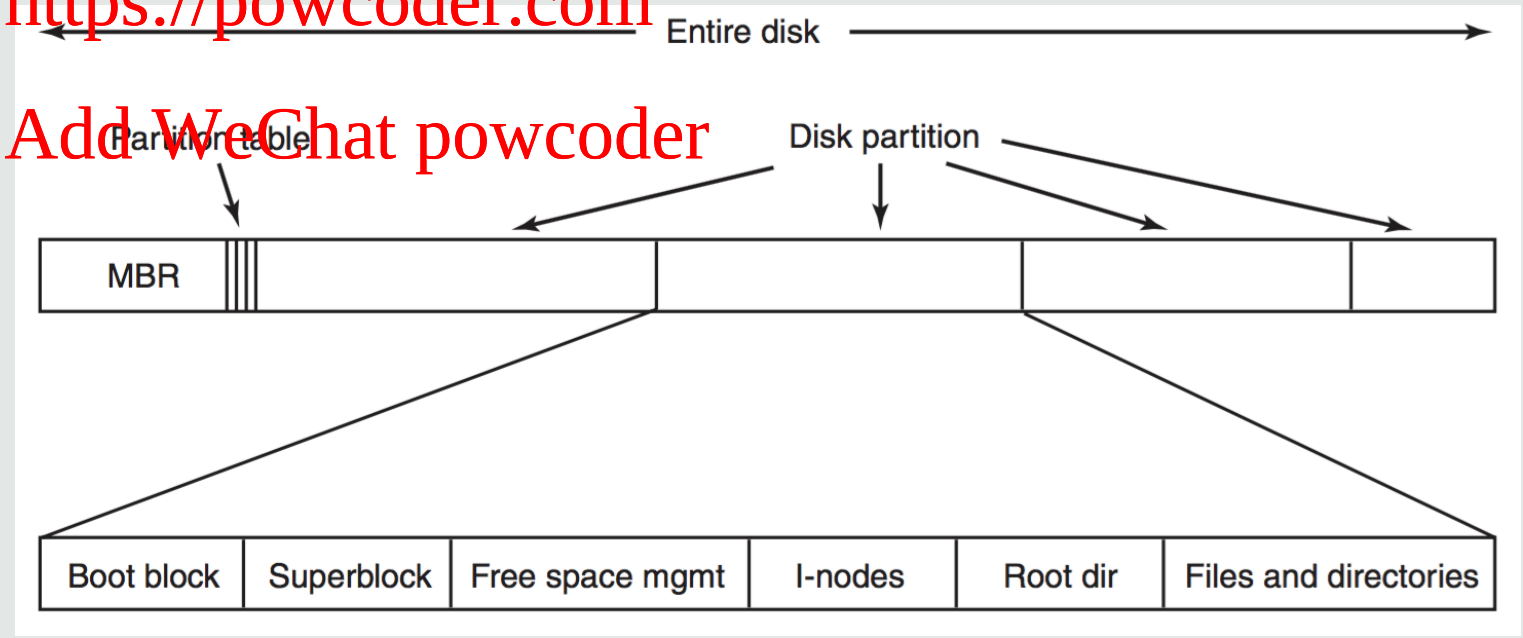
- Logical file system
  - Provides file interface (e.g., system calls)
- File organisation module
  - Manages block allocation to files
    E.g. i-nodes in UNIX
- Basic file system
  - Block-based storage format
    E.g. FAT-32, NTFS, ext4
- I/O device drivers
  - Physical access to storage media
    E.g. HD, DVD, USB



File system layout

Volume

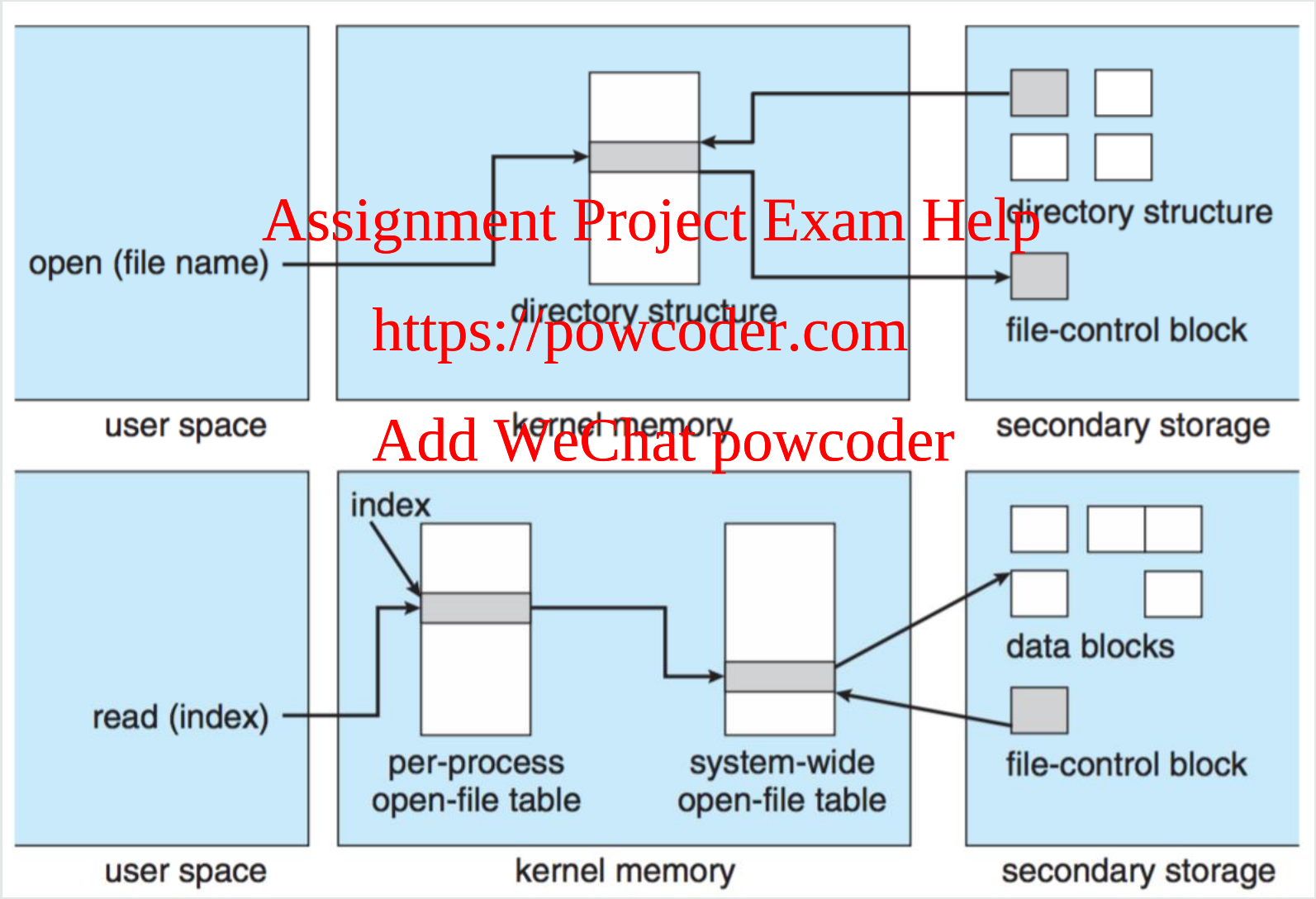
- Boot control block (e.g. boot block, partition boot sector)
- Volume control block (e.g. superblock, master file table)
- Directory structure: where to find files
- File control block for each file



Kernel data structures

- Mount table
  - Available volumes
  - Mount point (e.g. directory, drive)
- Open-file tables
  - Keeps track of files accessed by processes
  - Entry created on open(), removed on close()
  - Identified by file descriptor (or file handle)
  - Reference counting for open files
- Buffers and caches
  - Efficient locating of files
  - Efficient data transfer between storage and main memory

Kernel data structures



Block allocation

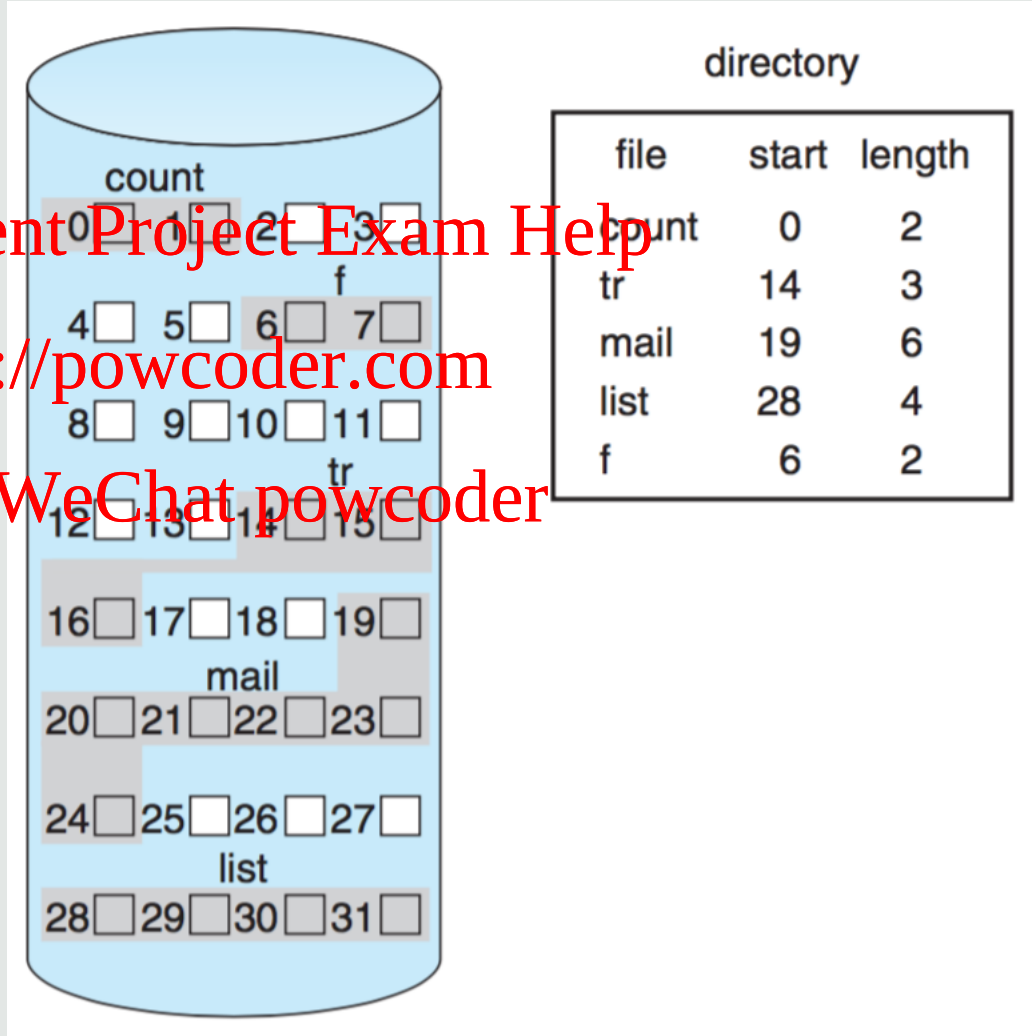
- Storage divided into blocks, e.g. 512 bytes
  - Block-wise addressing
  - File consists of one or more blocks
  - Blocks are loaded into main memory on file access
- Techniques
  - Contiguous Allocation
  - Linked-List Allocation
  - Indexed Allocation

Aspects to consider:

- Fragmentation
- Performance of finding and accessing blocks
- Size of tables for managing blocks

Block allocation

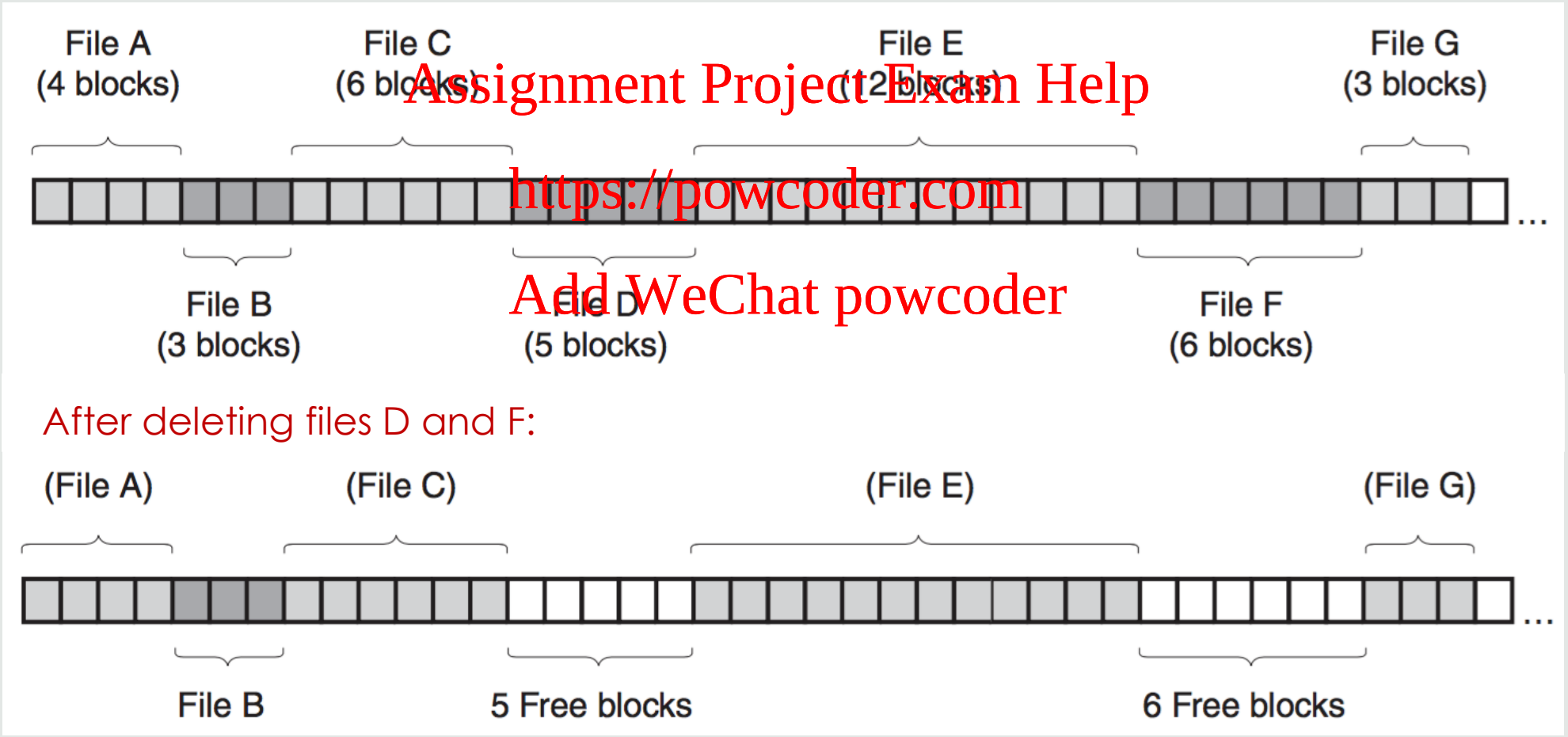
Contiguous Allocation



Block allocation

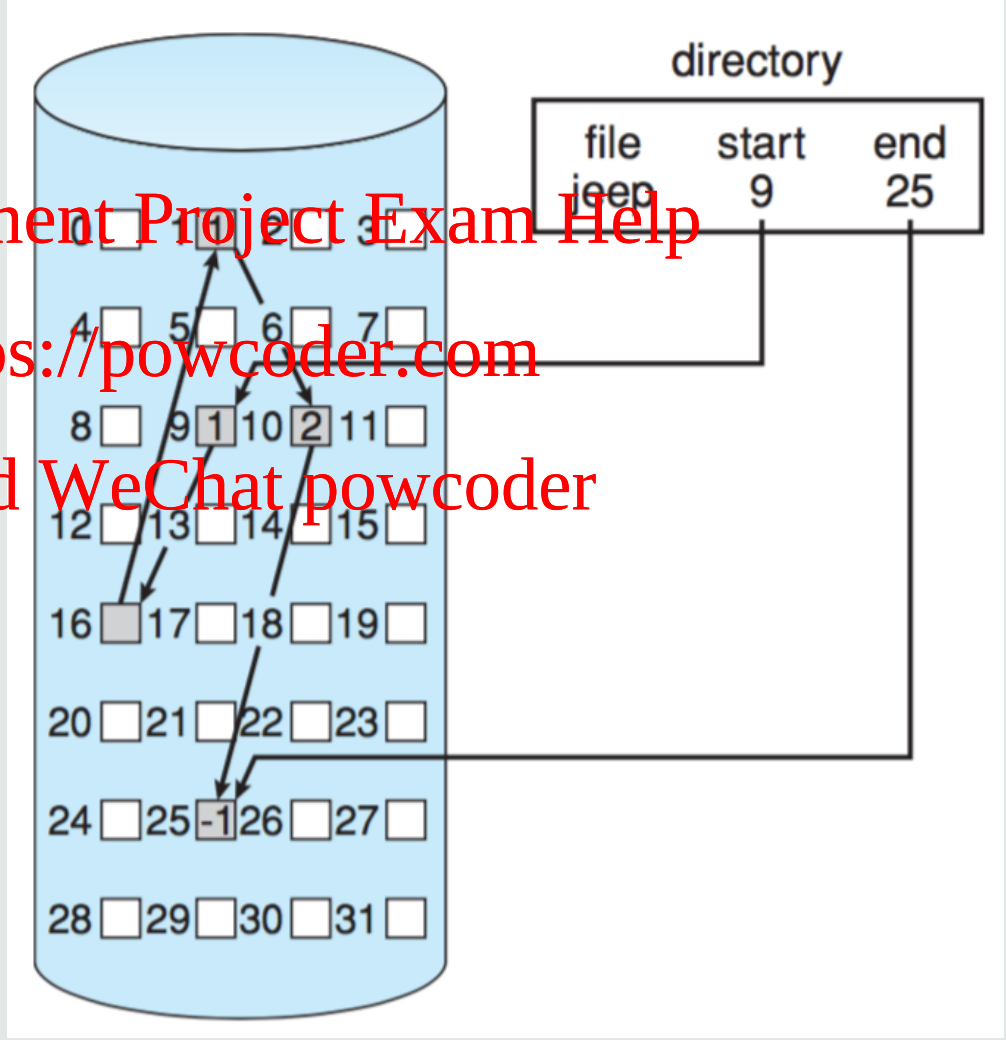
Contiguous Allocation

Problem: External fragmentation



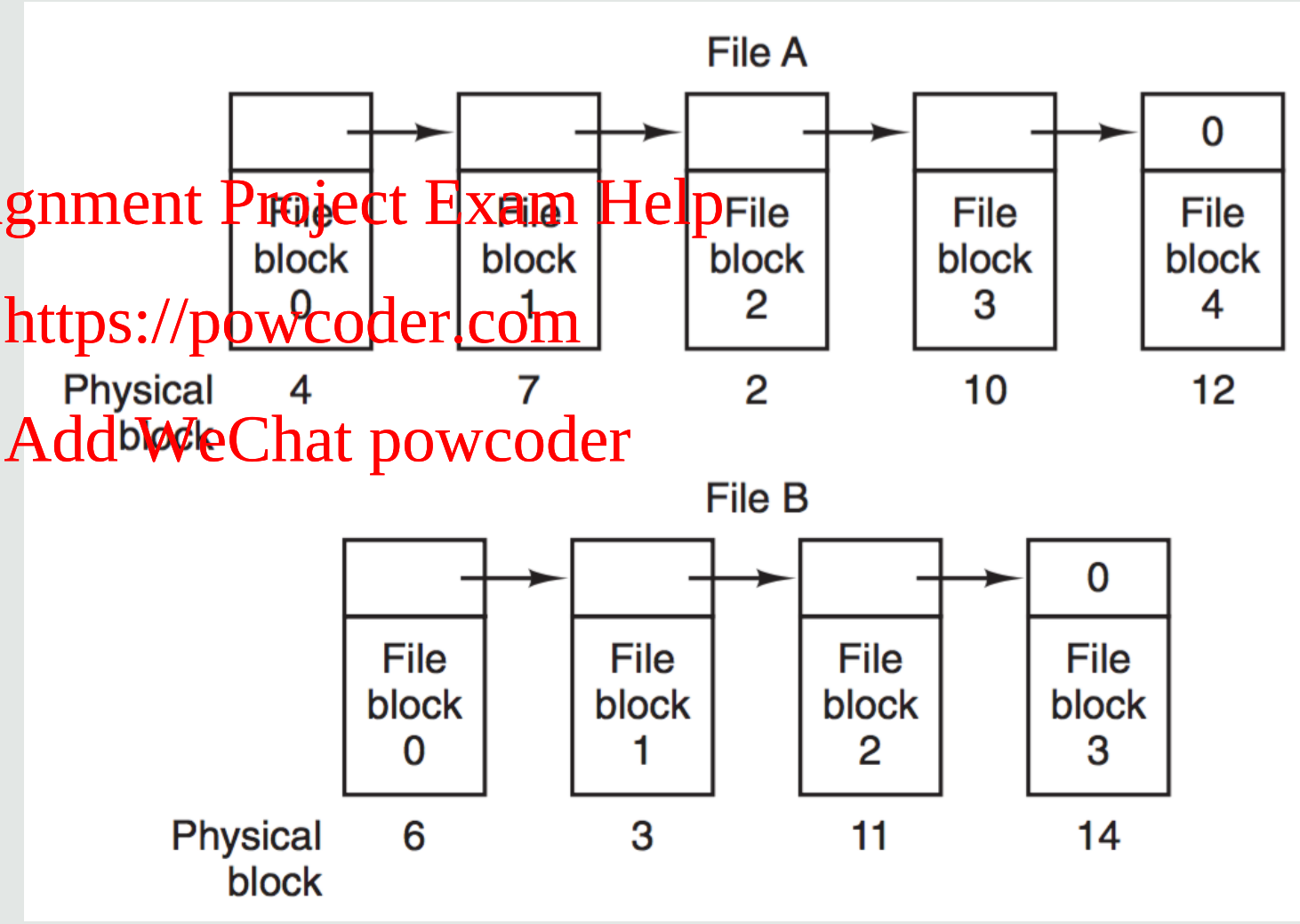
Block allocation

Linked-List Allocation



Block allocation

Linked-List Allocation

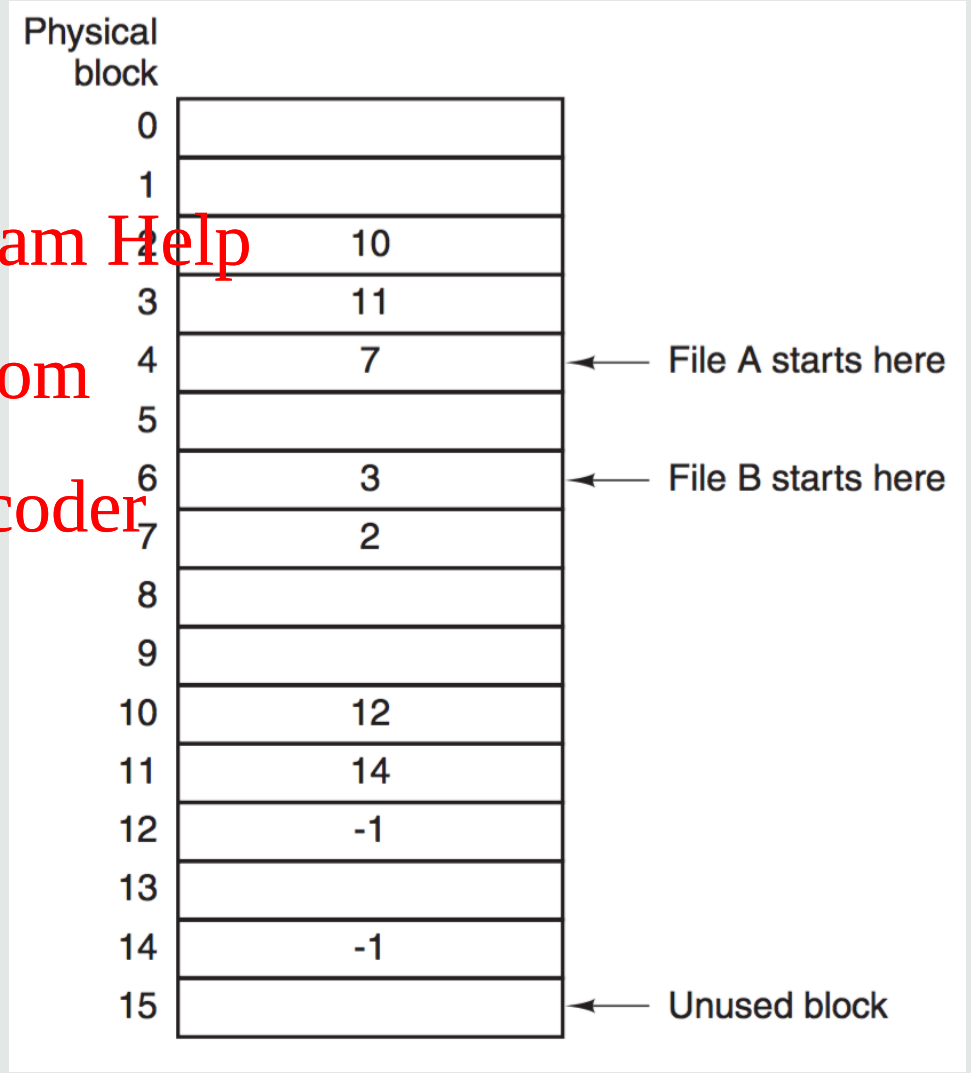


Problem:
Needs to traverse the list to find a block

Block allocation

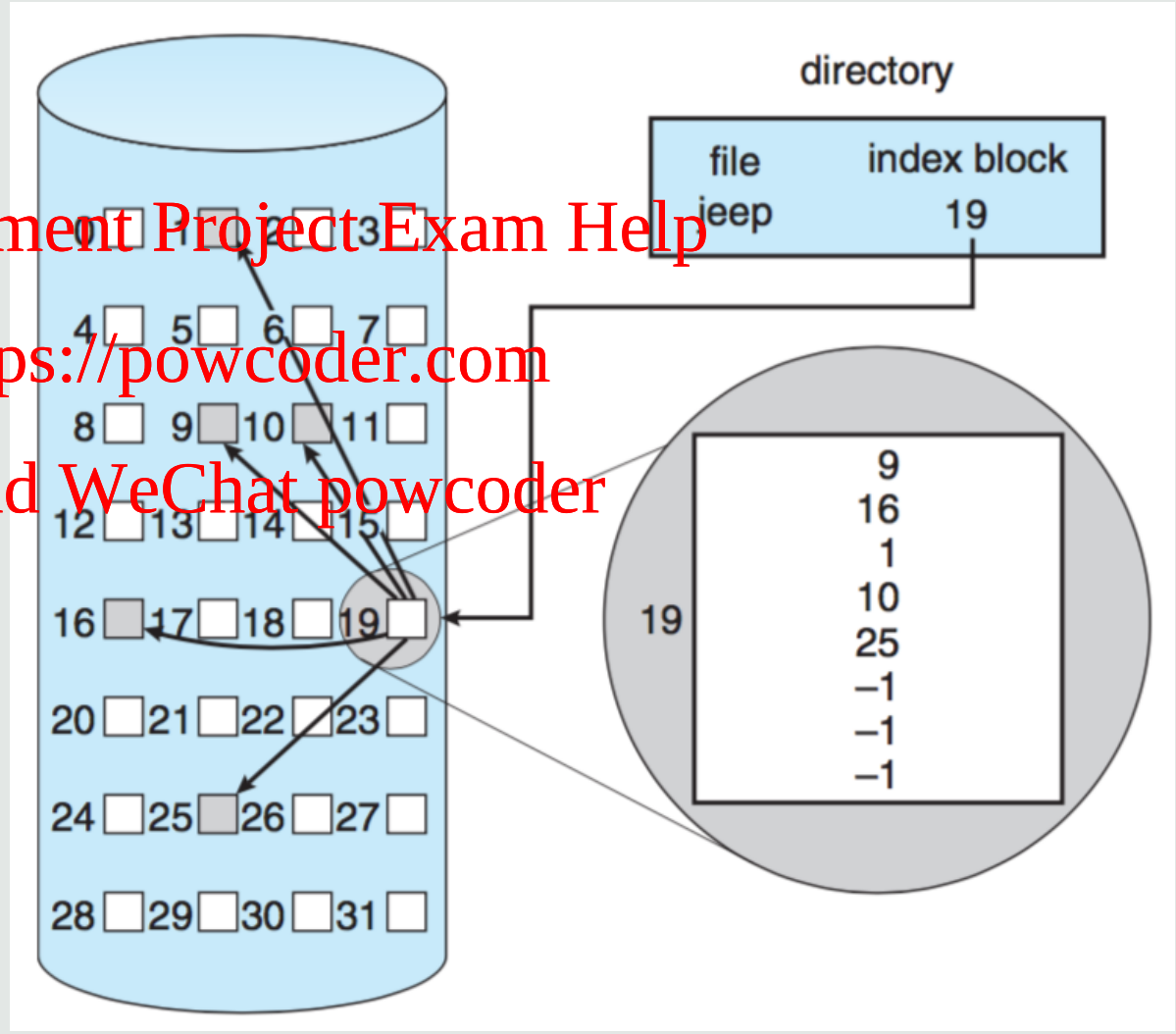
Example: File Allocation Table (FAT)

- Implementation of linked list:
  - Marker for last block of a file
  - Marker for unused blocks
- Simple to implement
- Performance optimisation by caching FAT
- E.g. FAT-32: most widely used



Block allocation

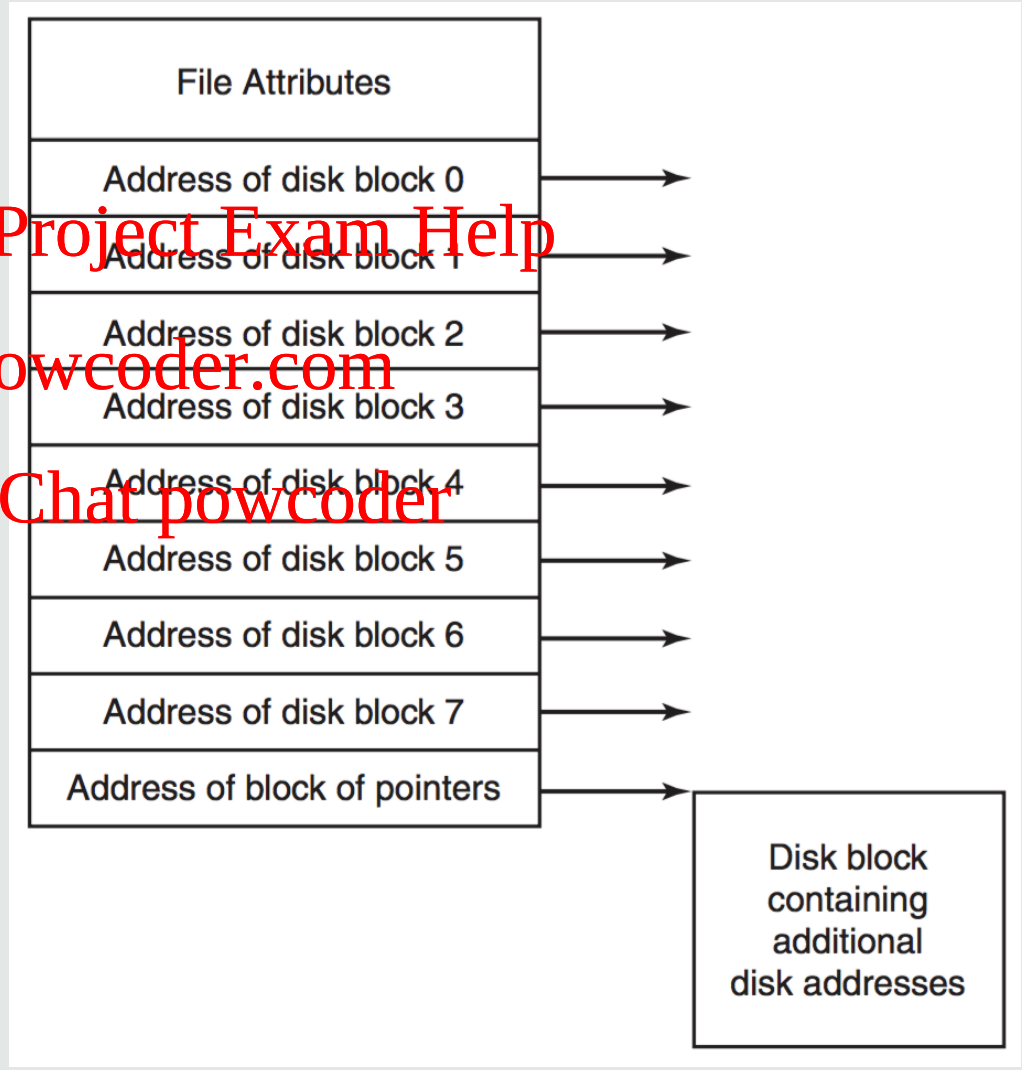
Indexed Allocation



Block allocation

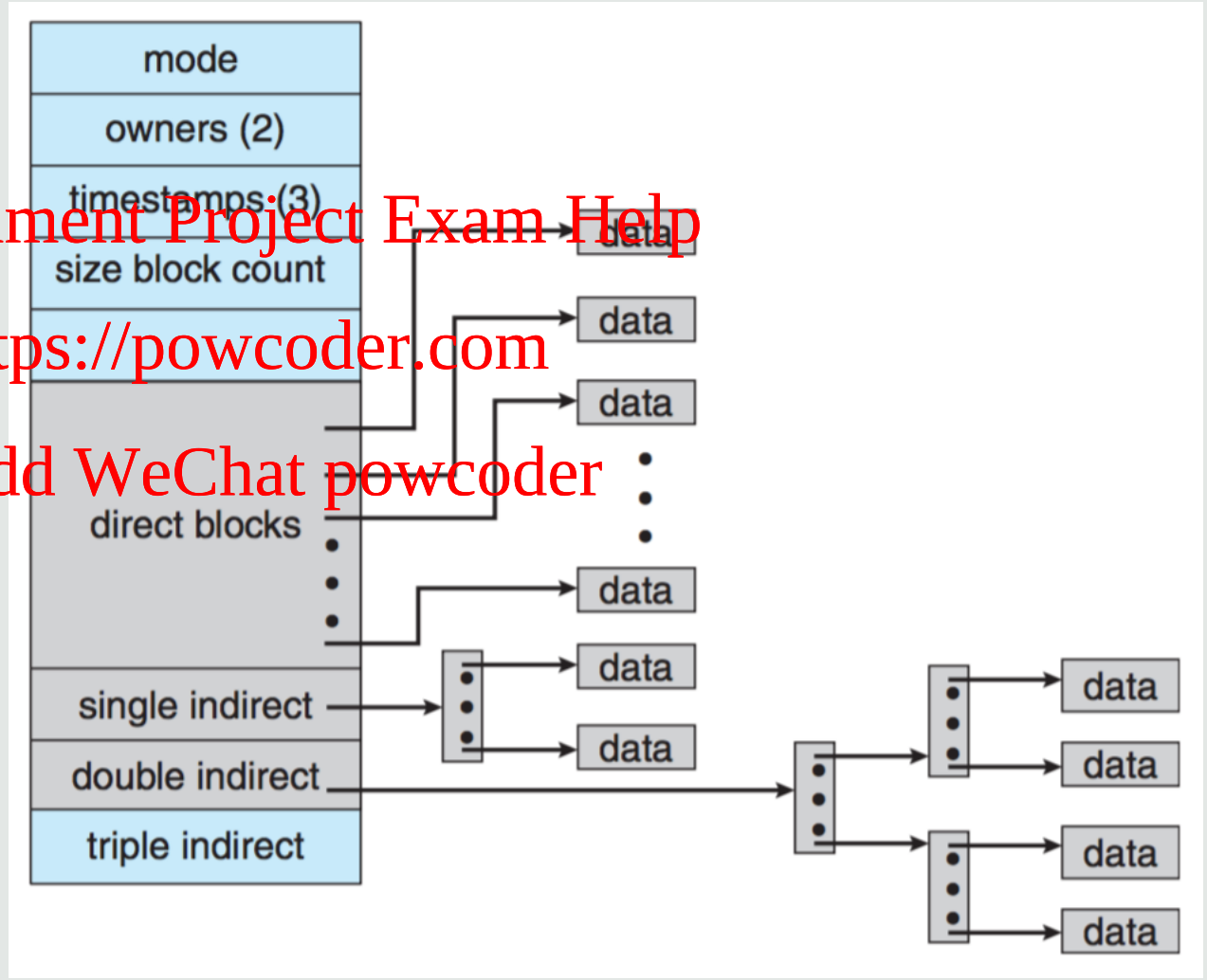
Index-Nodes (i-nodes)

- Multi-level indices



Block allocation

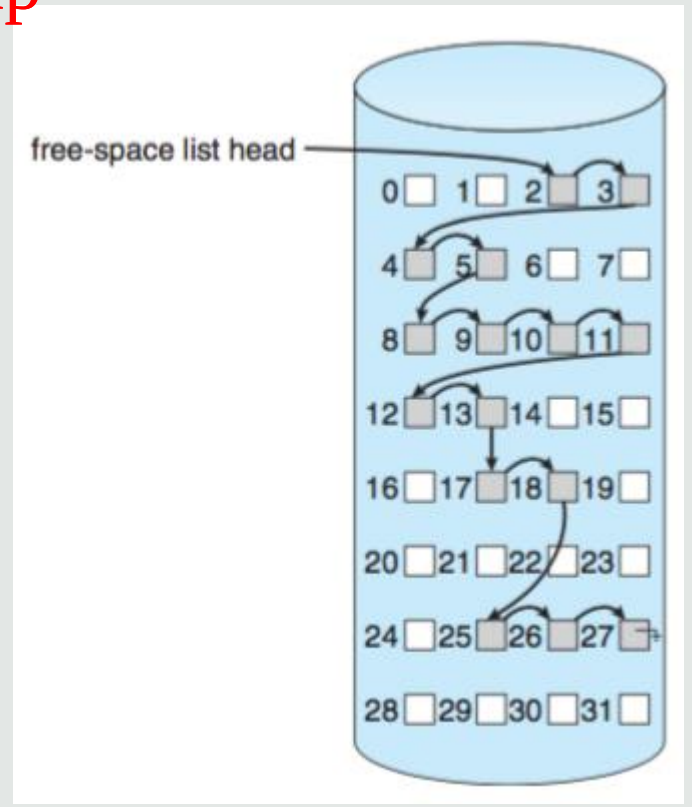
Example: UNIX i-nodes



Block allocation

Free Space Management

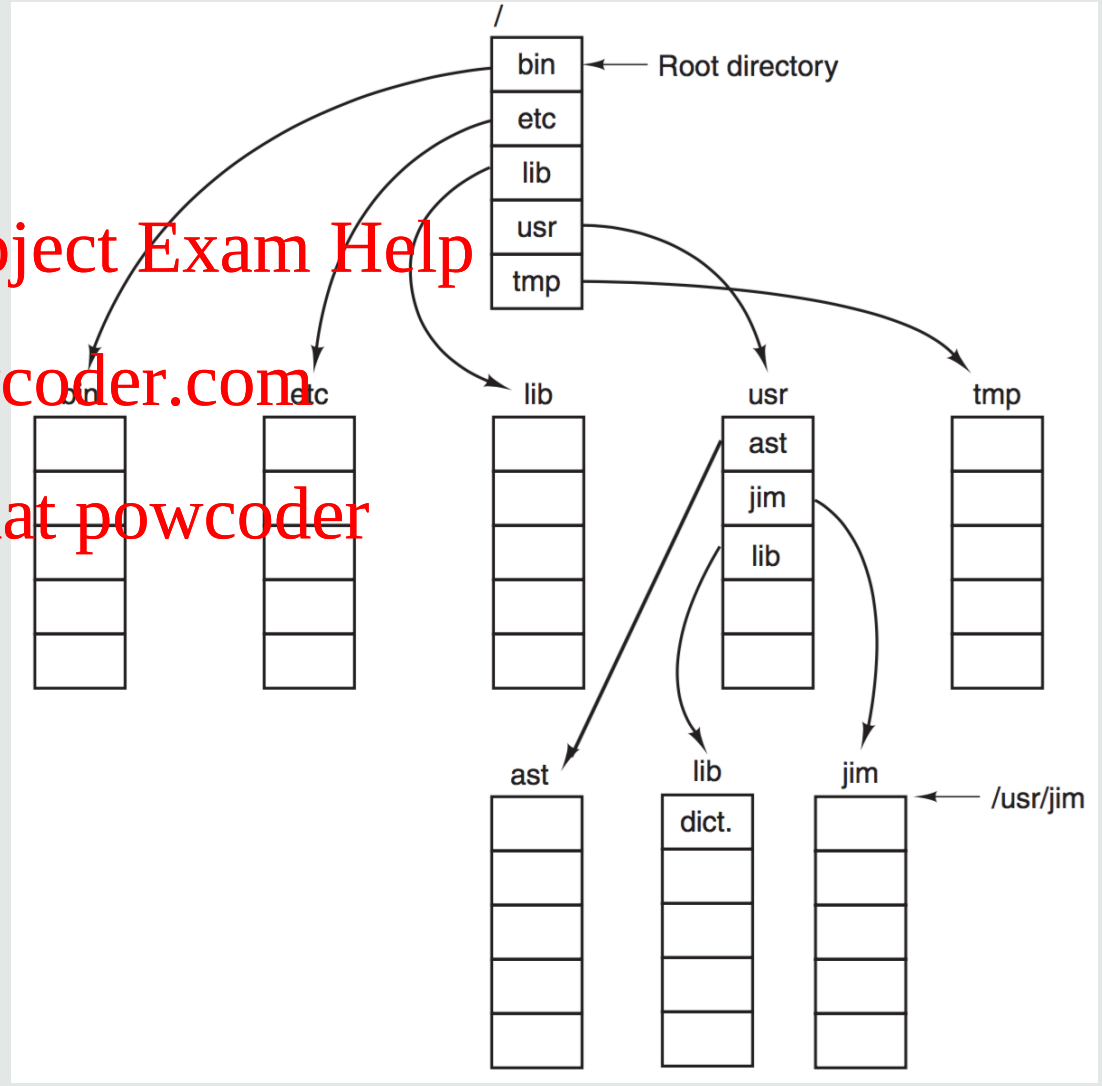
- Bitmap:
  - One bit for each block
  - E.g. `00111100111111000110000000011100000...`
- Runlength encoding:
  - Start + length
  - e.g. (2,4),(8,6),(25,3),....
- Linked list



Logical file system

User's view on the file system

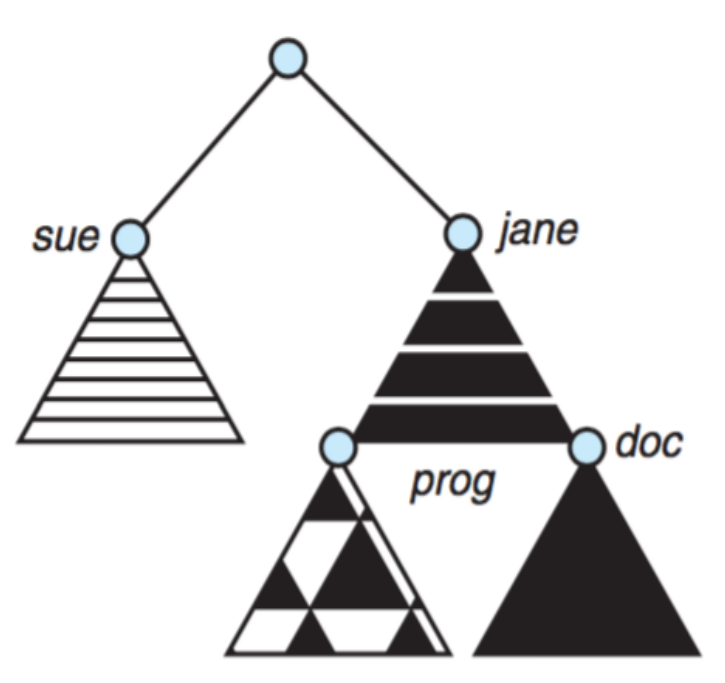
- Typically hierarchically structured



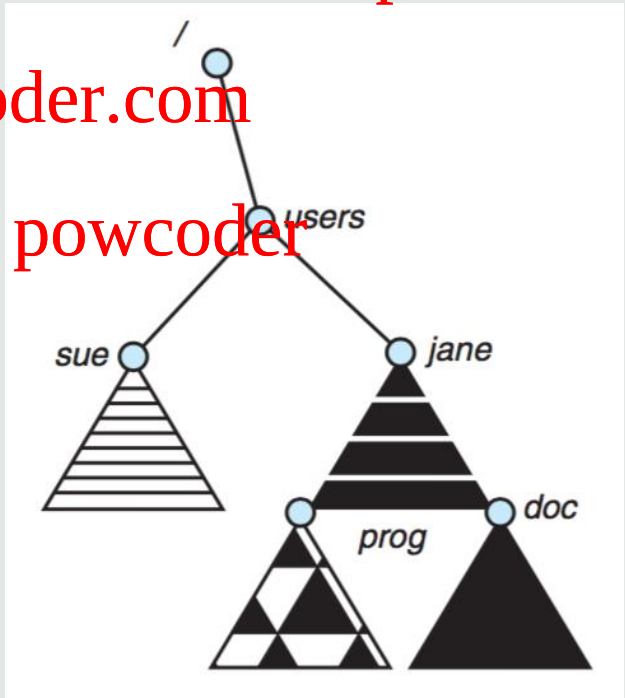
Logical file system

Mounting Volumes

Unmounted volume



Mounted at /users



Logical file system

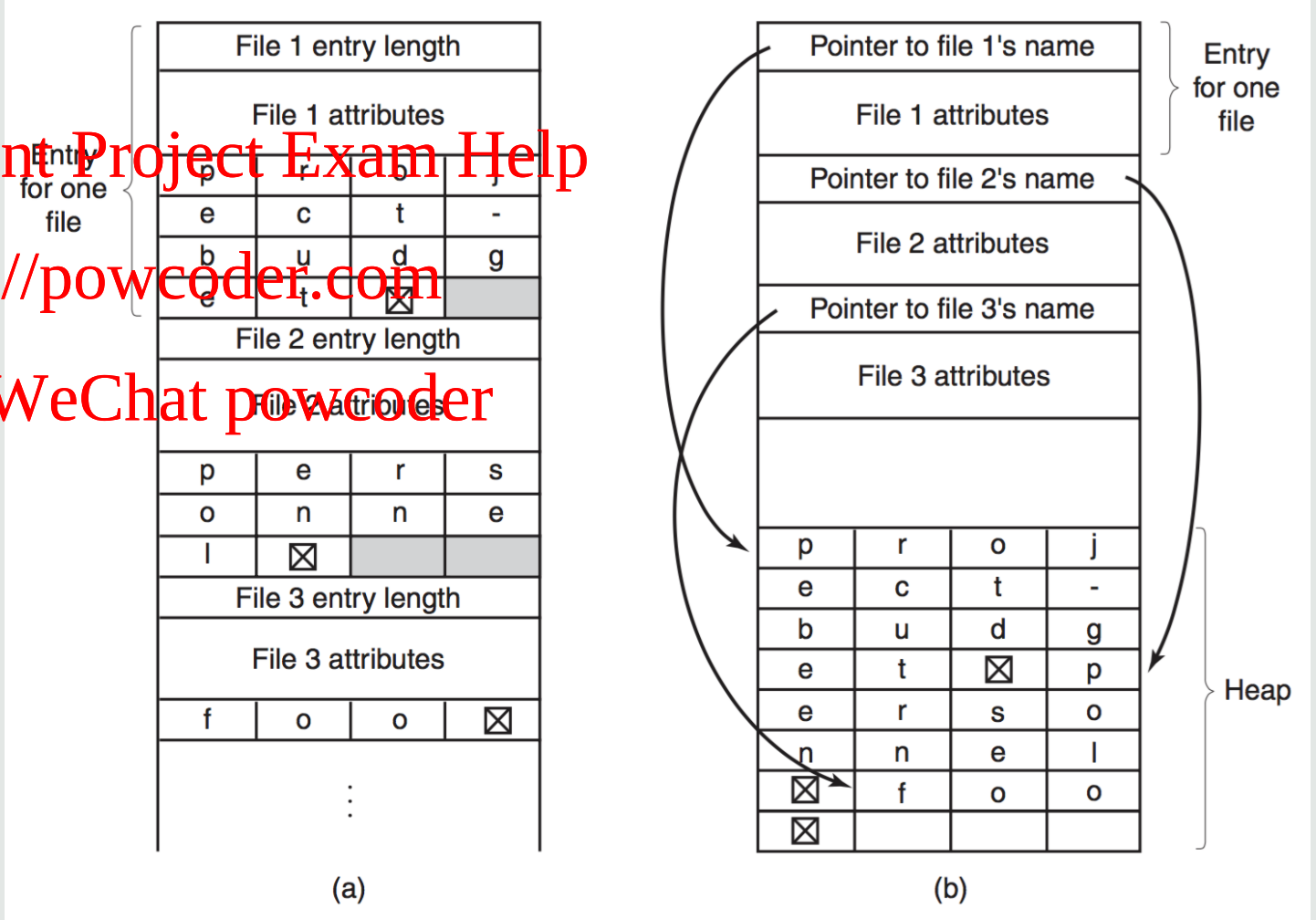
Path names

- E.g. **`/users/john/docs`**
- Uniquely identifies a file in the logical file system
  - Access path to a file
  - Elements separated by a delimiter, e.g. / , \ , >
  - Absolute: from root of the file system
  - Relative: to current working directory of the process
- Resolving a path name
  - Recursively search file system directory

Logical file system

File names implementation

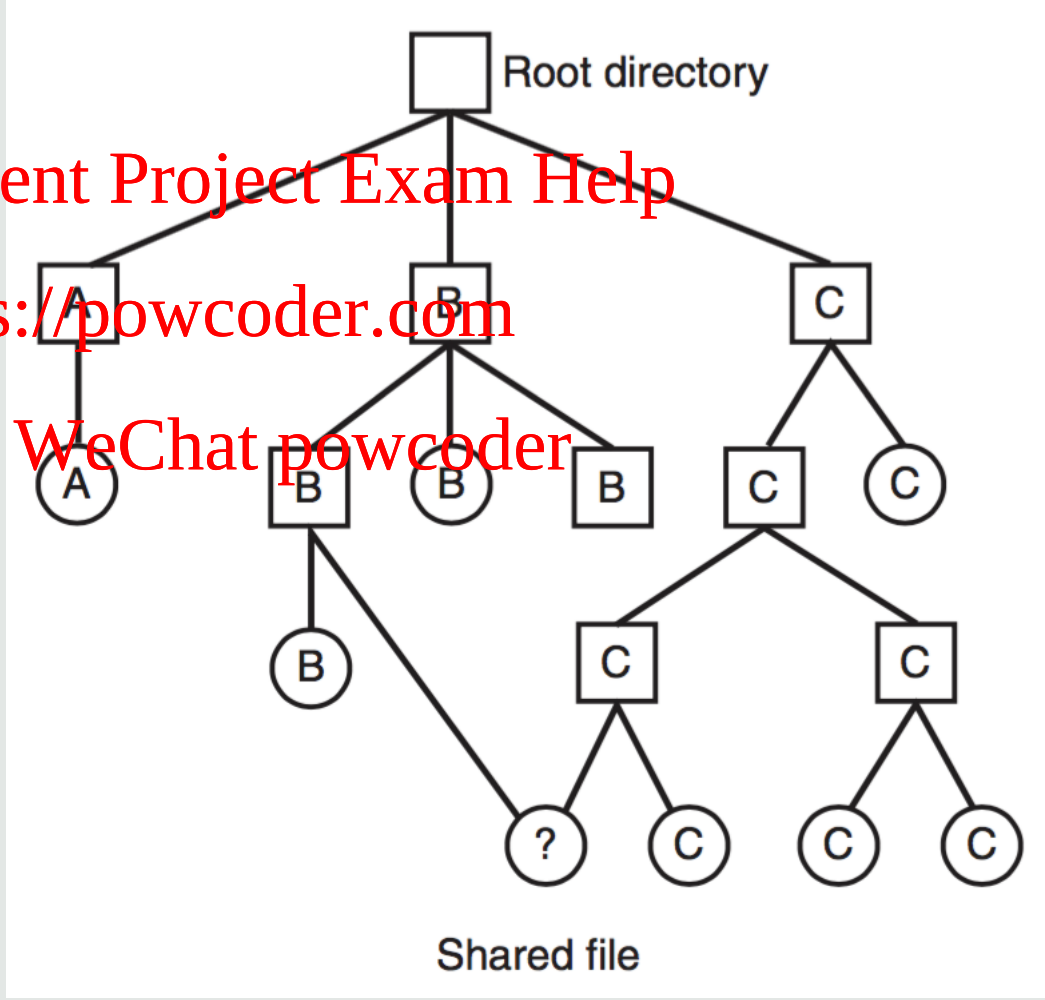
- We want to minimise management overheads



Logical file system

Shared files

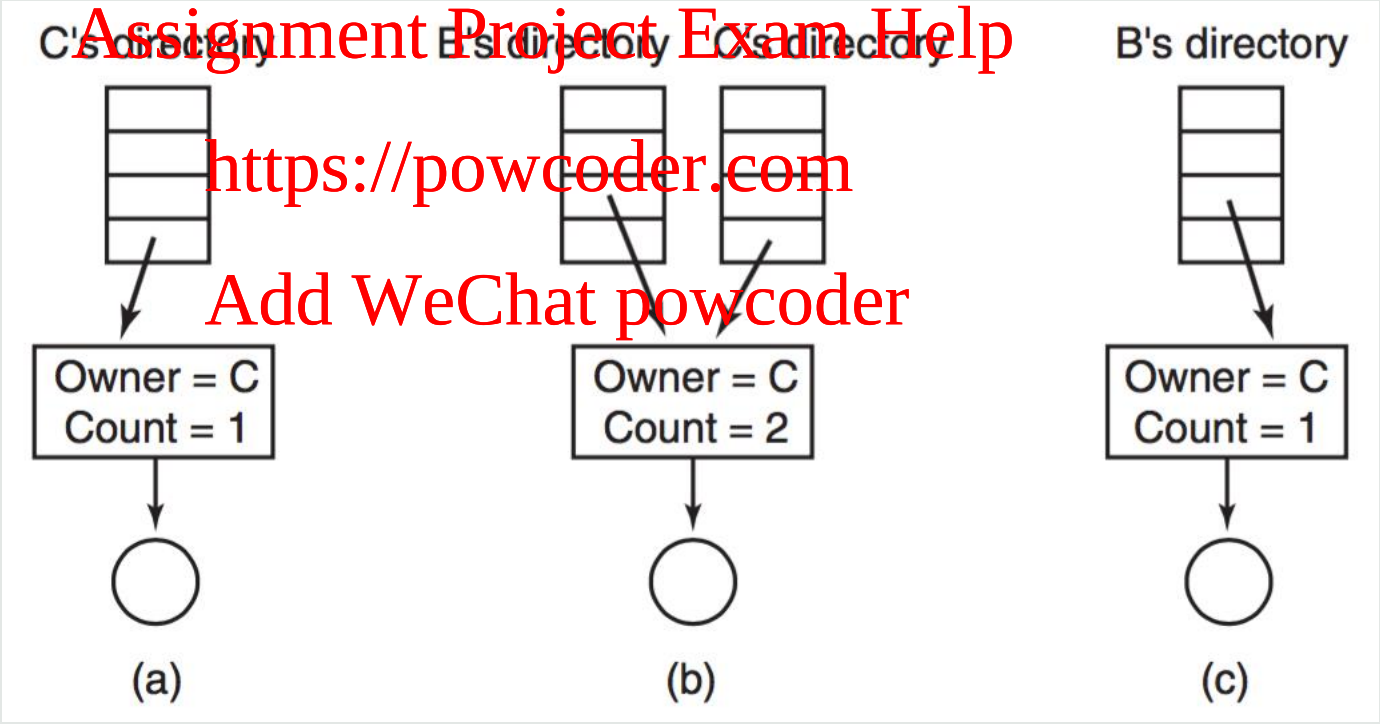
- UNIX:
  - Hard links
  - Symbolic links



Logical file system

Shared files

- Hard link:



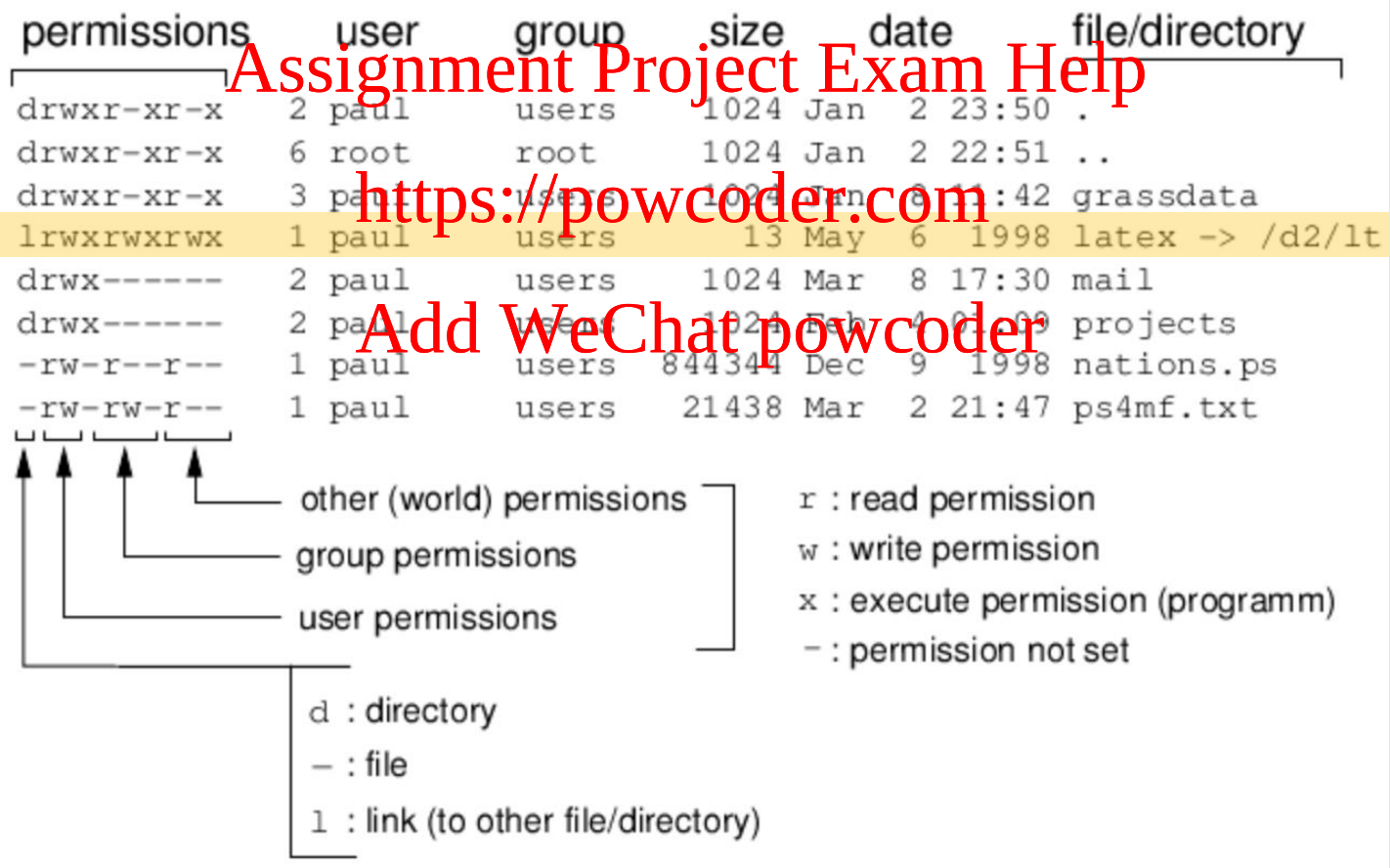
Logical file system

Shared files

- Soft (symbolic) link:
  - By pathname
  - Requires resolving the pathname to find actual file
  - Can link across volumes
  - No reference counting

Logical file system

Example: UNIX file attributes



Summary

File systems

- File system layouts
- Kernel data structures for file management
- Block allocation
  - Contiguous allocation
  - Linked list allocation
  - Indexed allocation
- Mounting, path names, hard vs symbolic links

Coursework 2 Preview

- Select two different operating systems, e.g.
  - Windows 10 and FreeBSD 11
  - Android 7 and iOS 10
  - …
- Compare two different implementation aspects of these systems in detail, e.g. from
  - Process management
  - Memory management
  - Storage management
  - …
- Write a report (~1500 words), due Friday 11 May, 4pm
- Marking criteria:
  - Depth of comparison and discussion
  - Combine information from many (primary) sources
  - Appropriate referencing of sources

Read

- Tanenbaum & Bos., Modern Operating Systems
  - Chapter 4
- Silberschatz et al., Operating System Concepts
  - Chapters 10 & 11

Next Lecture

- Introduction
- Operating System Architectures
- Processes
- Threads - Programming
- Process Scheduling - Evaluation
- Process Synchronisation
- Deadlocks
- Memory Management
- **File Systems (continued)**
- Input / Output
- Security and Virtualisation