



Operating System Practice

Che-Wei Chang

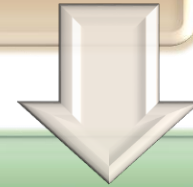
chewei@mail.cgu.edu.tw

Department of Computer Science and Information
Engineering, Chang Gung University

Course Roadmap

Advanced Operating System Concepts

- Concepts and Implementation of File System
- Storage Management and I/O Devices
- System Protection and Security



Exercises on PC and Emulators

- Understanding the Linux Kernel
- Customizing the Linux Kernel and Implementing of System Calls
- Android Programming on Android Emulator



Embedded System Exercises

- Introduction to Embedded System
- Tools and Techniques to Build Embedded Systems
- Implementation on Embedded System Evaluation Boards

Advanced Operating System Concepts



- Chapter 10: File System
- Chapter 11: Implementing File-Systems
- Chapter 12: Mass-Storage Structure
- Chapter 13: I/O Systems
- Chapter 14: System Protection
- Chapter 15: System Security





Review of Virtual- Memory Management

Virtual Memory

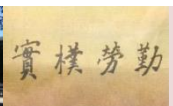
▶ Virtual Memory Technique

- A technique that allows the execution of a process that may not be completely in memory

▶ Potential Benefits

- Programs can be much larger than the amount of physical memory
- The level of multiprogramming increases because processes occupy less physical memory
- Each user program may run faster because less I/O is needed for loading or swapping user programs

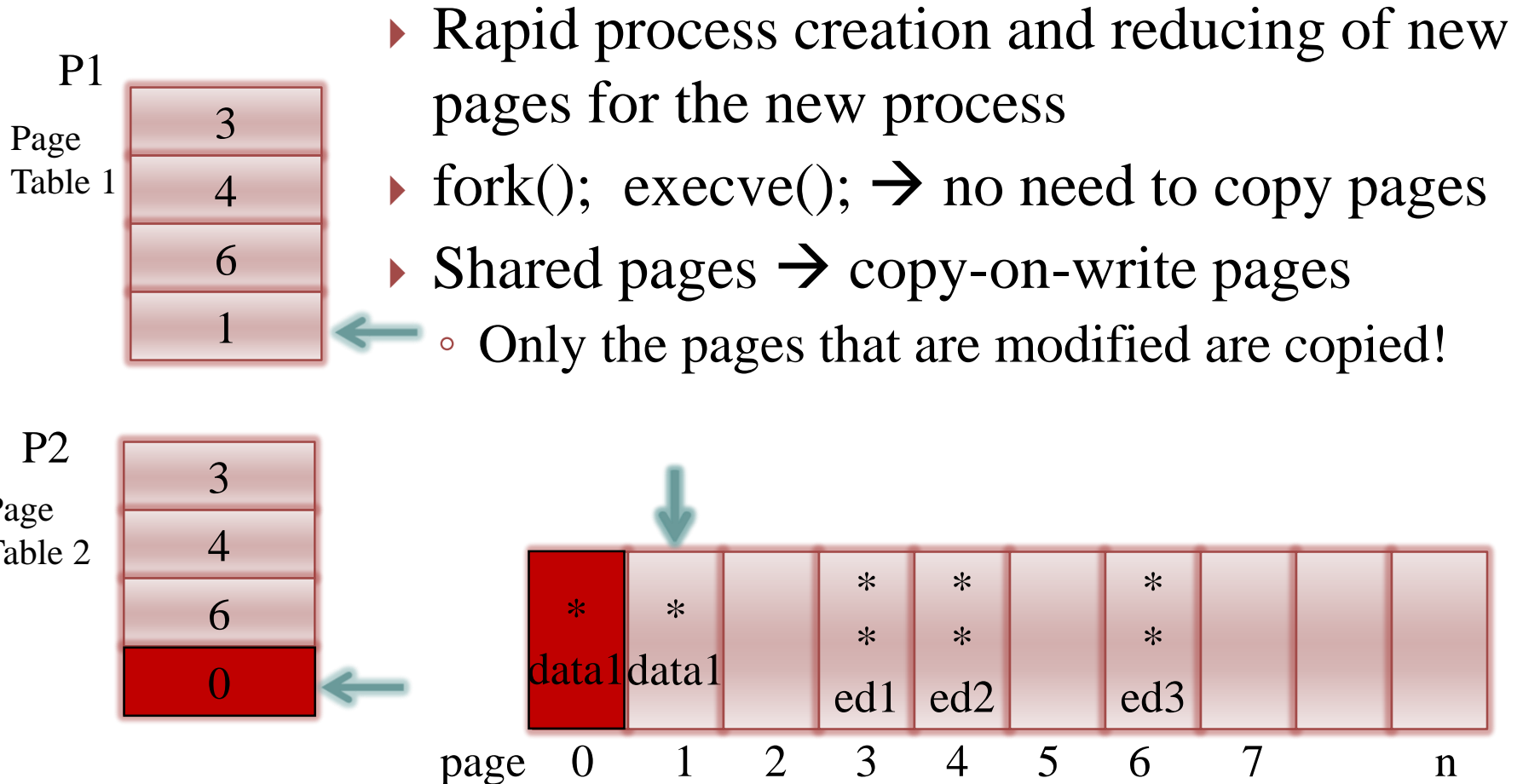
▶ Implementation: Demand Paging



Frame Allocation for Applications

- ▶ Global Allocation
 - Processes can take frames from others
- ▶ Local Allocation
 - Processes can only select frames from their own allocated frames → Fixed Allocation
 - The set of pages in memory for a process is affected by the paging behavior of only that process
- ▶ Remarks
 - Global replacement generally results in a better system throughput
 - Processes might not control their own page fault rates such that a process can affect each another easily under global replacement

Advanced Memory Management Techniques— Copy on Write



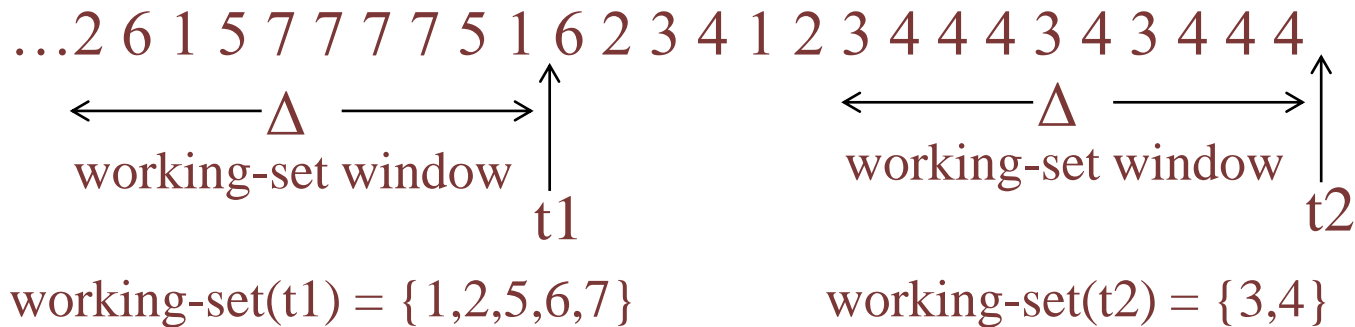
Advanced Memory Management Techniques— Working-Set Model

- ▶ Locality Model

- Spatial Locality: adjacent pages
- Temporal Locality: recently used pages

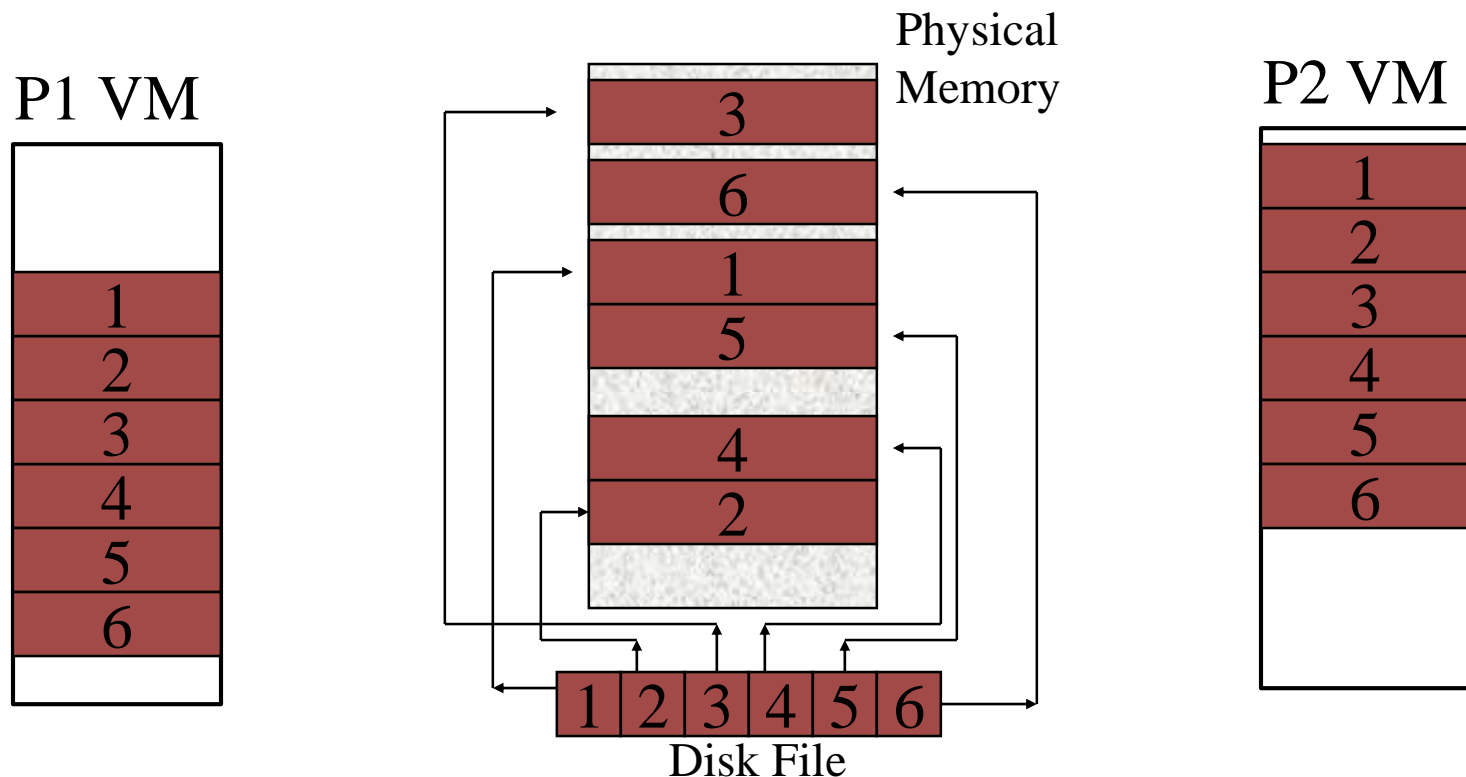
- ▶ Working Set: Approximation of a Program's Locality

Page references



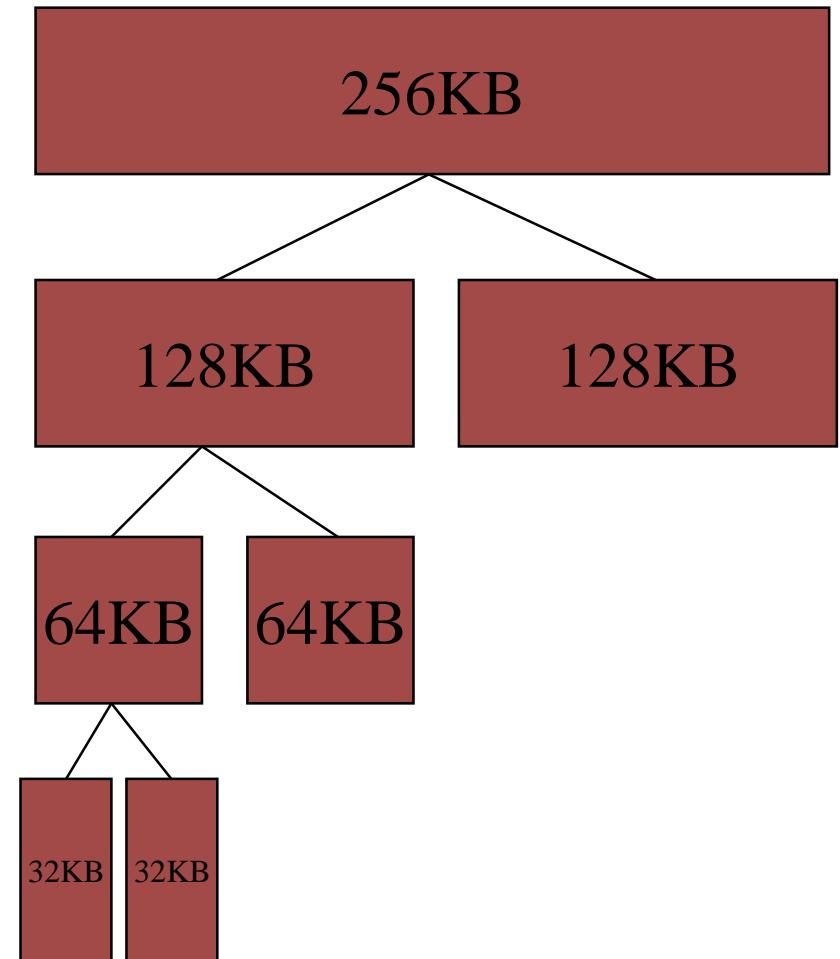
Memory Mapped Files

- ▶ File writes might not cause any disk write!
- ▶ Mapped files can be used for memory sharing!



Kernel Memory Allocation—Buddy System

- ▶ A Fixed-Size Segment of Physically Contiguous Pages
- ▶ A Power-of-2 Allocator
- ▶ Advantage: Quick Coalescing Algorithms
- ▶ Disadvantage: Internal Fragmentation

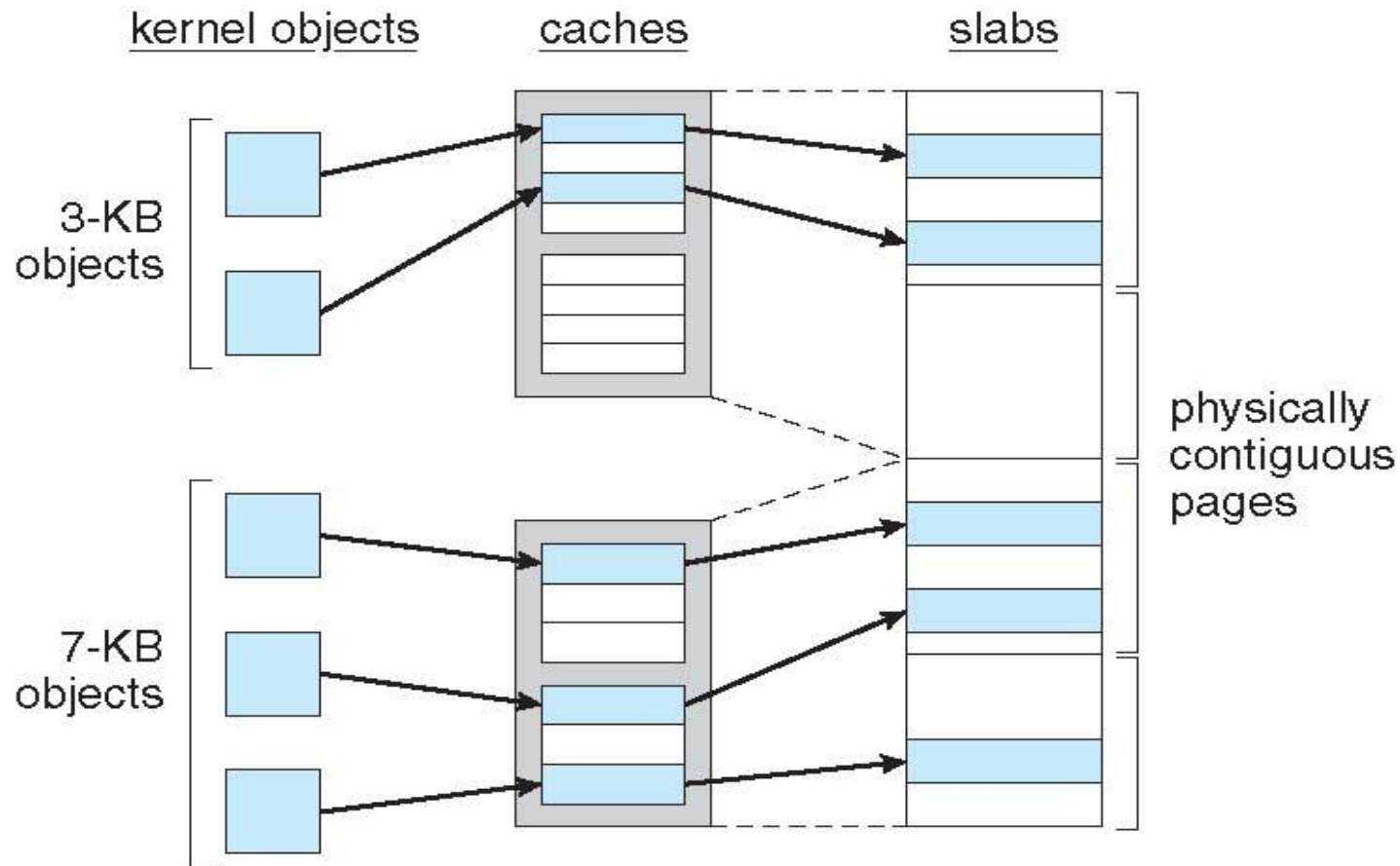


Kernel Memory Allocation— Slab Allocator (1 / 2)

- ▶ Slab: one or more physically contiguous pages
- ▶ Cache: one or more slabs
- ▶ Slab States
 - Full
 - Empty
 - Partial
- ▶ Slab Allocator
 - Look for a free object in a partial slab
 - Otherwise, allocate a new slab and assign it to a cache
- ▶ Benefits
 - No space wasted in fragmentation
 - Memory requests are satisfied quickly



Kernel Memory Allocation— Slab Allocator (2/2)





Chapter 10: File System

Why Storage Management

► Motivations

- Main memory is too small to accommodate all the data and programs permanently
→ Secondary Storage
- A mechanism is needed for on-line storage access to both programs and data residing on the secondary storage
→ File System

► Device Variety

- Speed, Dedication, Read/Write, Char/Block Transfer, Synchronous Mode, etc.



File Concepts

▶ Files

- Each is a named collection of related information
- Each is a logical unit often with its interpretation left for applications, creators, or users
 - Text, Source, Object, Executable Files

▶ A Directory Structure

- Meta Data & File Organization



File Attributes

- ▶ File attributes vary from one OS to another:
 - Name: Case-sensitive or not
 - The only information must be kept in human-readable form
 - Identifier: A unique tag
 - Type: It is only for systems that support file types
 - Location
 - Size: Current and max sizes
 - Protection: access control
 - Time, date, and user identification
- ▶ File attributes are usually kept in the directory structure



File Operations (1 / 3)

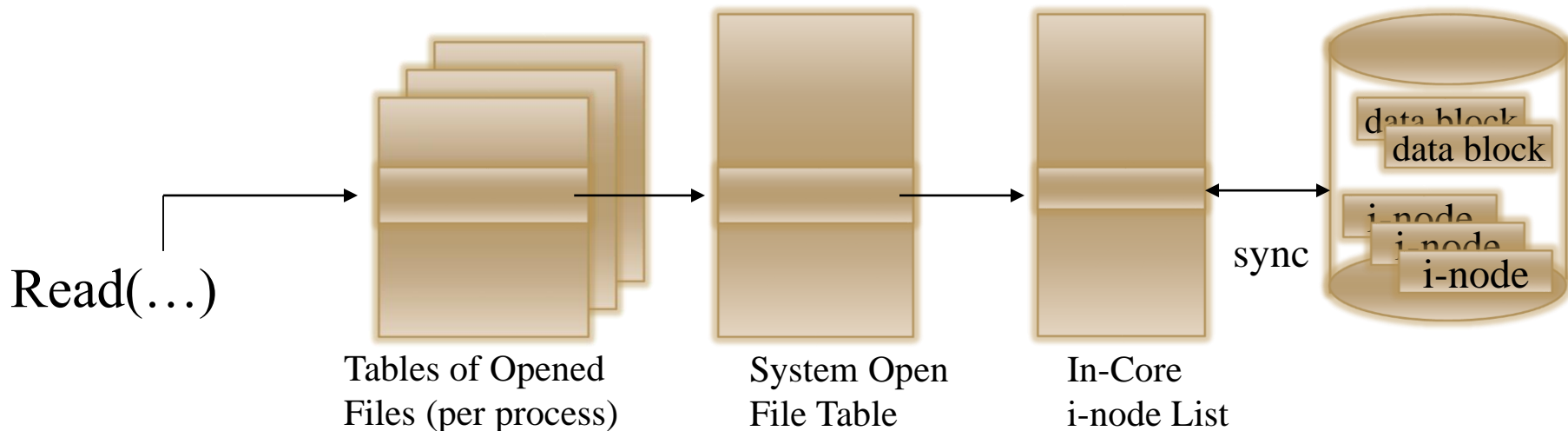
► Basic Directory Operations:

- File creation: Space allocation & directory-structure-entry creation
- File open and close
- File writing: Write pointer
- File reading: Read pointer
- File reposition: Seek-like operations
 - File-position pointer
- File deletion: Space reclaiming & directory-structure-entry deletion
- File truncating: File-length resetting



File Operations (2 / 3)

- ▶ Open, Close, Read and Write among Multiple Processes
 - File Descriptors and Tables
 - File Position Pointer, File-Open Count
 - Disk Location and Access Rights



File Operations (3 / 3)

► Extensions

- File Renaming, Appending, Copying, etc.

► Other Operations

- Attribute Retrieval and Setting
- File Locking
 - Shared or Exclusive Locks
 - Mandatory (Windows) Locks— access is denied depending on locks held and requested
 - Advisory (Unix) Locks— processes can find status of locks and decide what to do
- Search of a File
 - A File-System Traversal



File Types

▶ Key Issue

- The Recognition of File Types by OS

▶ Common Techniques

- Types as Parts of File Names
 - .doc, .txt, .rtf, .mpeg, .mp3, .avi, .pdf, .ps, .tex, .exe, .com, .bin, .c, .cc, .java, .asm, .a, .bat, .sh, .o, .obj, .lib, .dll, .zip, .tar, .arc, etc.
- A Magic Number at the Beginning of a File
 - Enforcement or Hints? → Application Duty



Access Methods (1 / 2)

▶ Sequential Access

- Read-Next and Write-Next Operations
- Reset or N-Record Skipping/Rewinding

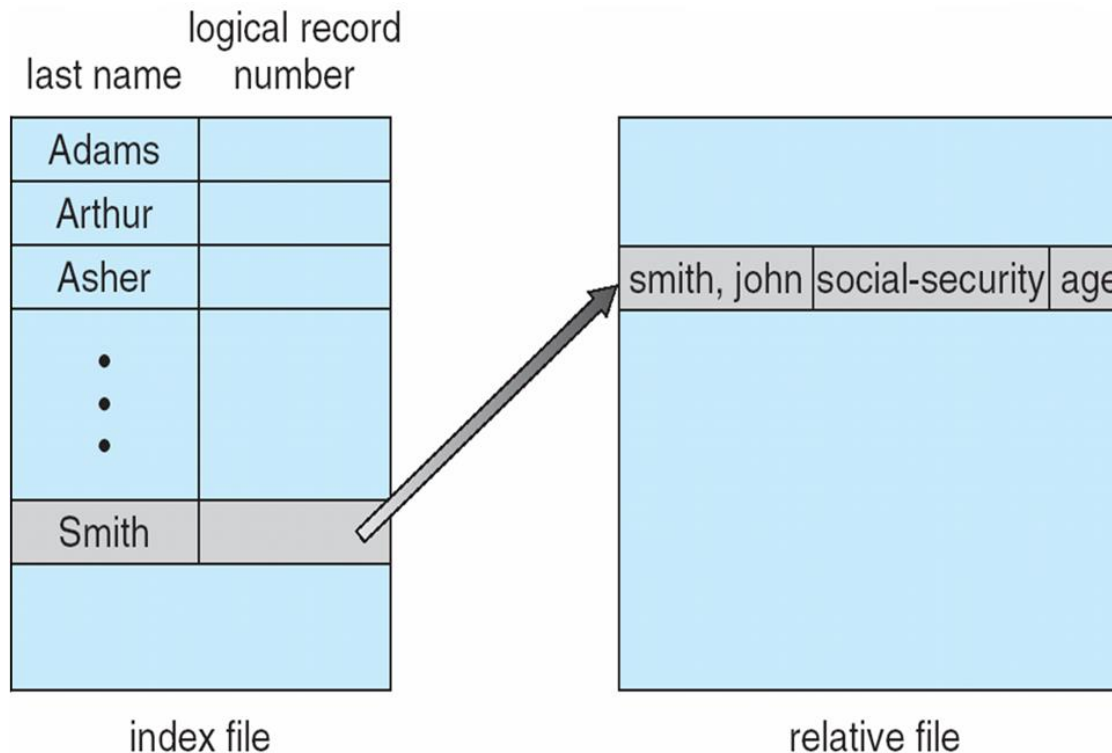
▶ Direct Access (or Relative Access)

- A file is considered as a numbered sequence of blocks or records
- Read-N, Write-N, and Position-N Operations
 - Relative Block/Record Number
- Easy Simulation of Sequential Access



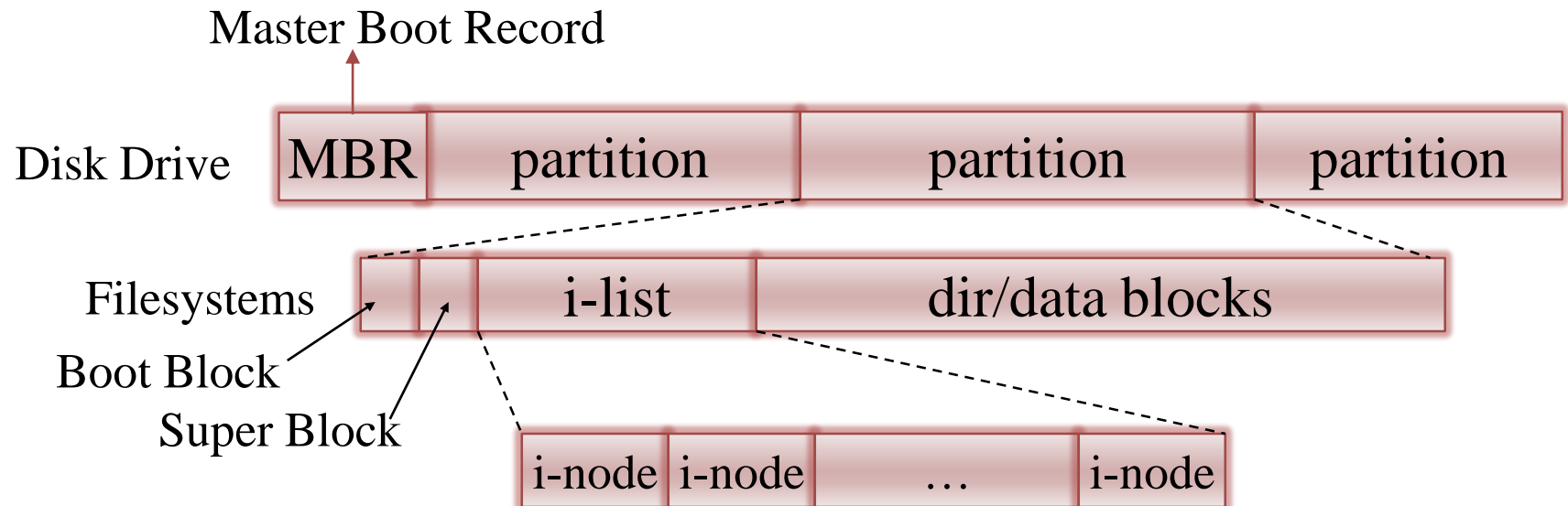
Access Methods (2 / 2)

► Index-Based Access



Directory Structure

- ▶ A hierarchical arrangement of directories and files – starting at root “/”
 - File: An abstract data type
 - Volume: A chunk of storage that holds a file system



Directory Overview

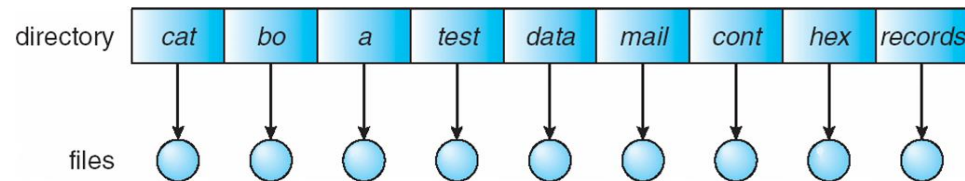
- ▶ Directory – A Symbol Table that Translate File Names into Their Directory Entries
- ▶ Operations on a Directory
 - Searching for a File
 - Create a File
 - Delete a File
 - List a Directory
 - Rename a File
 - Traverse the File System



Simple Directories

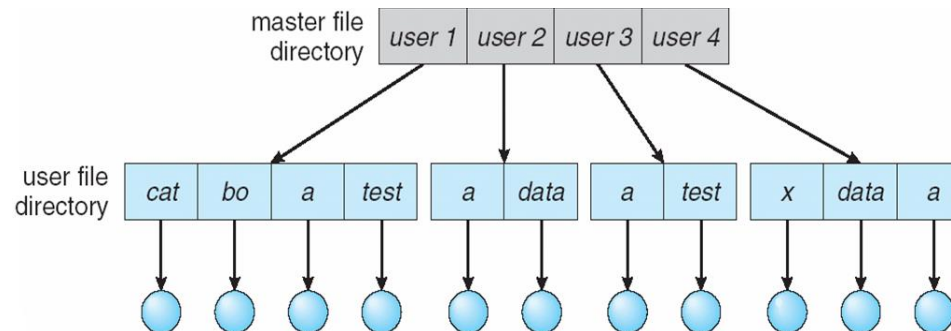
► Single-Level Directory

- All files are in the same directory
 - Problems occur when the number of files increases or when the system has more than one user



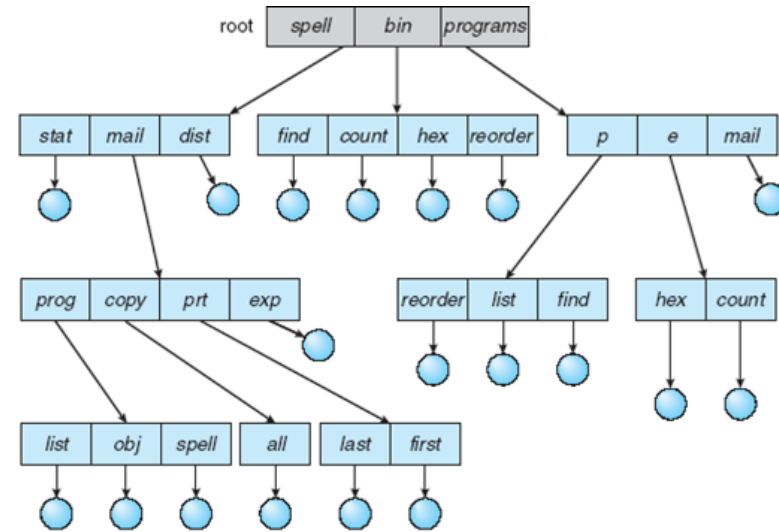
► Two-Level Directory

- The Master File Directory (MFD) → Multiple User File Directories (UFD's) → Files



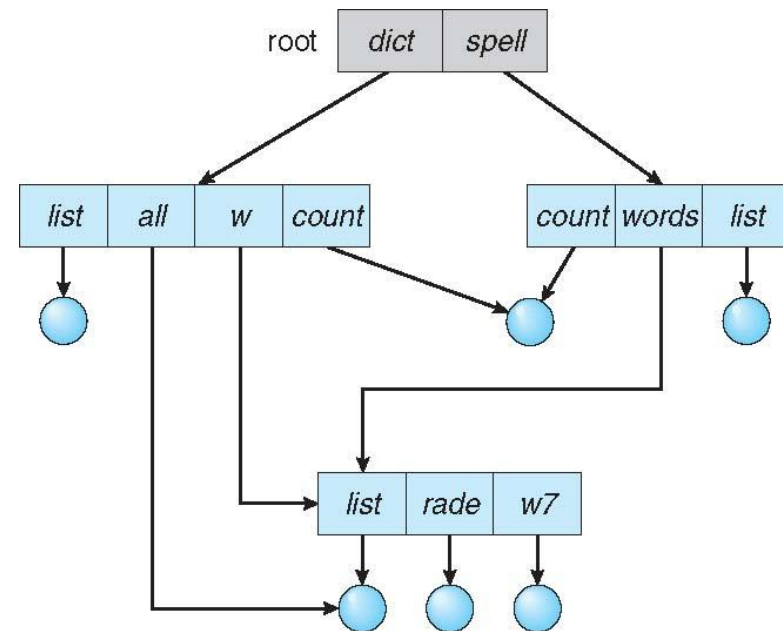
Tree-Structured Directories

- ▶ The Root Directory → Subdirectories and/or files
 - Example: MS-DOS
- ▶ Current and Home Directories
 - A child process usually inherits the current directory of its parent
- ▶ Absolute and Relative Path Names
 - Examples: /root/spell/mail and spell/mail
- ▶ Policies
 - Directory Deletion: Only Empty Directories?
 - `rm -r file-name`



Acyclic-Graph Directories

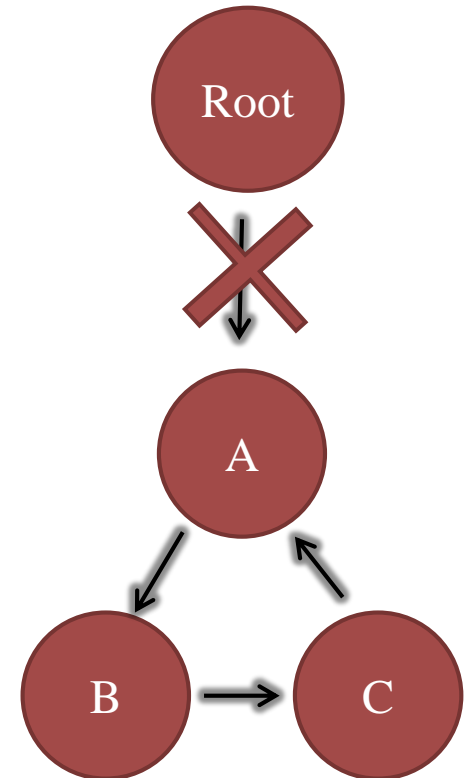
- ▶ Motivation— Allow the Sharing Compared to Tree-Structured Directories
- ▶ File-Sharing Implementations
 - Links – A pointer to another file or subdirectory
 - Hard and soft links
 - Information Duplication
 - Consistency issue
 - Potential Problems
 - Multiple path names
 - Traversal and deletion problems



General Graph Directory

► Potential Problems:

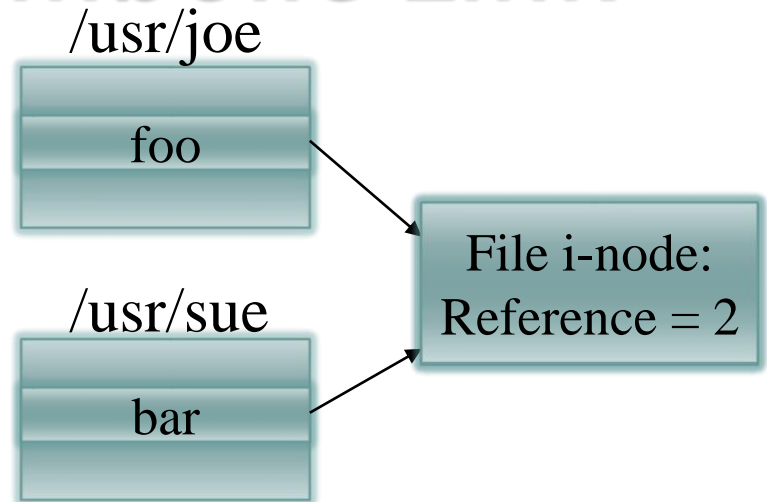
- Problems in Correctness and Performance in Searching Any Components
 - Limitation on the Number of Accessed Directories?
- Problems in File Deletion
 - Self-Referencing or a Cycle
 - Garbage Collection: Traversing, Marking and Deletion → Extremely Time-Consuming
 - Bypassing Links during Directory Traversal



Hard Link and Symbolic Link

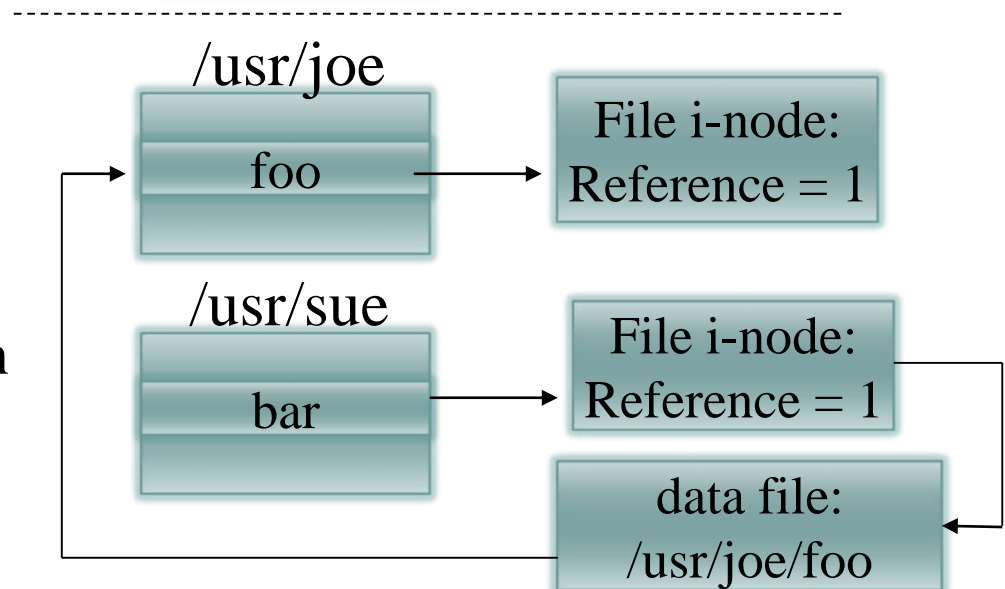
▶ Hard Link

- Each directory entry creates a link of a filename to the i-node that describes the file's contents



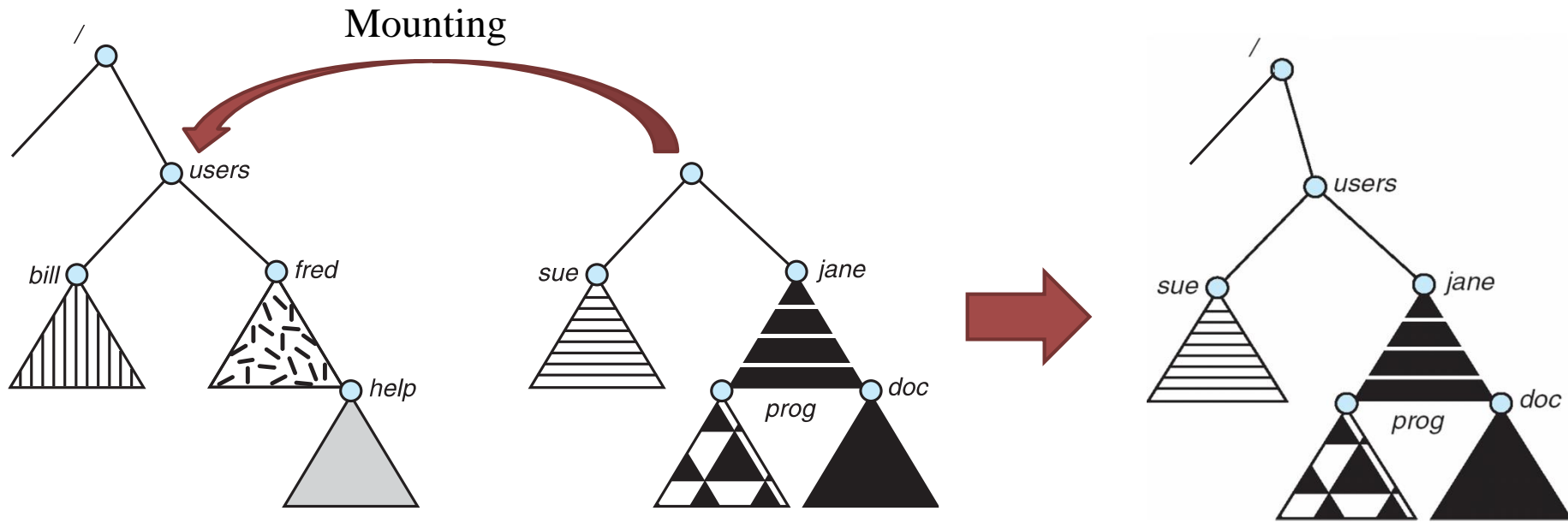
▶ Symbolic Link (Soft Link)

- It is implemented as a file that contains a pathname
- Filesize = pathname length
- Example: shortcut on Windows



File System Mounting

- ▶ A file system must be **mounted** before it can be accessed
- ▶ An unmounted file system is mounted at a **mount point**



File Sharing

- ▶ Sharing of files on multi-user systems is desirable
- ▶ Sharing may be done through a **protection** scheme
- ▶ On distributed systems, files may be shared across a network
- ▶ Network File System (NFS) is a common distributed file-sharing method
- ▶ If multi-user system
 - **User IDs** identify users, allowing permissions and protections to be per-user
 - **Group IDs** allow users to be in groups, permitting group access rights
 - Owner of a file/directory
 - Group of a file/directory



Remote File Systems

- ▶ Uses networking to allow file system access between systems
 - Manually via programs like FTP
 - Automatically, seamlessly using distributed file systems
 - Semi automatically via the world wide web
- ▶ Client-server model allows clients to mount remote file systems from servers
 - Server can serve multiple clients
 - Client and user-on-client identification is insecure or complicated
 - NFS is standard UNIX client-server file sharing protocol
 - CIFS is standard Windows protocol
 - Standard operating system file calls are translated into remote calls
- ▶ Distributed Information Systems: such as DNS (Domain Name System), NIS (Network Information Service), ... implement unified access to information needed for remote computing



File Sharing— Failure Modes

- ▶ All file systems have failure modes
- ▶ Remote file systems add new failure modes, due to network failure, server failure
 - Recovery from failure can involve **state information** about status of each remote request



File Sharing— Consistency Semantics

- ▶ Specify how multiple users access a shared file simultaneously
 - Similar to process synchronization algorithms
 - Unix File System (UFS) implements:
 - Writes to an open file visible immediately to other users of the same open file
 - Sharing a file pointer to allow multiple users to read and write concurrently
 - Andrew File System (AFS) implemented complex remote file sharing semantics
 - Writes to an open file is not visible immediately to other users
 - Writes only visible to sessions starting after the file is closed



Protection

- ▶ File owner/creator should be able to control:
 - What can be done by whom
- ▶ Types of access
 - Read
 - Write
 - Execute
 - Append
 - Delete
 - List



Protection on Unix

- ▶ Mode of access: read, write, execute
- ▶ Three classes of users on Unix / Linux

a) owner access	7	➔	RWX 1 1 1
b) group access	6	➔	RWX 1 1 0
c) public access	1	➔	RWX 0 0 1

