Unit IV: Virtual Memory Management and File Management

---

1. Demand Paging

Key Terms:

- **Page**: A fixed-length block of virtual memory.
  A page is the smallest unit of data for memory management in a virtual memory system. It's a fixed-size block that represents a portion of a program's address space and maps to a frame in physical memory.
- **Page Fault**: Happens when a page is not in memory and needs to be loaded.
  This occurs when a program tries to access data on a page that is not currently in RAM. The OS interrupts the program, loads the page from disk into memory, and resumes execution.
- **Lazy Loading**: Only load memory pages when needed.
  This technique defers loading parts of a program into memory until they are actually accessed, reducing memory usage and speeding up initial loading.

Important Points:

- Reduces memory usage and load time.
- Speeds up startup time for large programs.

20/80 Insight: Understand what a page fault is and how demand paging reduces memory usage.

Review Question: What is demand paging and how does it help optimize memory?
Answer:
Demand paging is a memory management strategy where the operating system loads pages into memory only when they are needed. When a process tries to access a page that is not currently in memory, a page fault occurs, and the OS retrieves the page from secondary storage (like a hard drive) into RAM. This technique avoids loading the entire process into memory at once, which saves space and allows more processes to run simultaneously. It leads to better memory utilization and reduced startup times for programs, making the system faster and more efficient.

---

2. Copy-on-Write (CoW)

Key Terms:

- **Shared Pages**: Initially shared among processes.
  These are memory regions accessible by multiple processes at once. They help reduce redundancy and improve efficiency when data doesn't change.
- **Copy When Written**: A page is copied only if it is modified.
  This prevents unnecessary duplication. Only when a process writes to a shared page does the OS create a private copy.

Important Points:

- Efficient memory use.
- Reduces duplication in multitasking.

20/80 Insight: Focus on the idea that pages are only copied when a change is made.

Review Question: What is Copy-on-Write and why is it used in OS?
Answer:
Copy-on-Write (CoW) is an optimization technique that delays copying memory pages until they are modified. When two or more processes share the same data, they initially point to the same memory location. As long as no process modifies the data, the memory remains shared. When a write occurs, the OS creates a separate copy of the page for the writing process. This reduces unnecessary duplication of memory, saves resources, and is particularly useful during process creation with the `fork()` system call, improving efficiency and performance.

---

3. Write (in Virtual Memory)

Key Terms:

- **Write-back**: Write to disk only when necessary.
  Changes are temporarily stored in cache. Data is written to disk only on certain triggers, like cache eviction or shutdown.
- **Write-through**: Immediate write to disk.
  Ensures data is always consistent between memory and disk by writing every change immediately.

Important Points:

- Balances speed and data safety.
- Write strategies affect performance.

Review Question: What are the differences between write-back and write-through strategies?
Answer:
In the write-back strategy, data is first written to a cache and updated to disk later, reducing the number of write operations and improving performance. However, if a system crash occurs before the cache is saved to disk, data can be lost. In contrast, the write-through strategy writes

data to both cache and disk immediately, which ensures data integrity but may slow down system performance due to frequent disk writes. Choosing between the two depends on whether performance or data safety is prioritized.

---

4. Page Replacement

Key Terms:

- **Page Replacement Algorithm**: Decides which page to remove.
  When memory is full, these algorithms decide which page to evict to load a new one. Common algorithms include FIFO, LRU, and Optimal.

Popular Algorithms:

- FIFO
- LRU
- Optimal (theoretical)

20/80 Insight: Understand LRU (Least Recently Used) clearly.

Review Question: How does the LRU page replacement algorithm work?
Answer:
The Least Recently Used (LRU) algorithm replaces the page in memory that hasn't been used for the longest period of time. It assumes that recently accessed pages will be used again soon, and older, unused pages are less likely to be needed. By keeping track of page access history, LRU minimizes page faults and improves performance. It’s more efficient than simple algorithms like FIFO and is widely implemented in operating systems.

---

5. Allocation of Frames

Key Terms:

- **Frame**: A block of physical memory.
  Frames are fixed-size blocks in RAM that hold pages from virtual memory. Each page is mapped to one frame.

Strategies:

- Equal Allocation
- Proportional Allocation
- Priority Allocation

20/80 Insight: Know the trade-off between equal and priority allocation.

Review Question: How does priority-based frame allocation work?
Answer:
Priority-based frame allocation assigns memory frames to processes based on their priority levels. High-priority processes are given more frames, ensuring they have sufficient memory to run efficiently, while lower-priority processes receive fewer frames. This helps ensure that critical tasks complete quickly and system resources are used efficiently. It balances performance and fairness, especially in multi-tasking environments.

---

6. Thrashing

Key Terms:

- **Thrashing**: Constant page swapping.
  Thrashing severely degrades performance when the system is overloaded with too many processes, leading to continuous paging and almost no useful work.

Important Points:

- System becomes slow.
- Caused by overloading RAM.

20/80 Insight: Learn how high page faults relate to thrashing.

Review Question: What is thrashing and how can it be prevented?
Answer:
Thrashing occurs when a system spends most of its time swapping pages in and out of memory rather than executing processes. It happens when too many processes are competing for limited memory, causing continuous page faults. Prevention techniques include reducing the number of running processes, using better page replacement algorithms, and applying the working set model to allocate enough memory to active processes. Proper memory management and load control help avoid thrashing and improve performance.

---

7. Memory-Mapped Files

Key Terms:

- **Mapping**: Link a file to memory space.

Uses:

- File I/O becomes faster.
- Supports IPC (Inter-process communication).

Review Question: What is the use of memory-mapped files in OS?
Answer:
Memory-mapped files allow applications to access file contents directly via memory instead of traditional I/O functions. This technique maps a file's content to the virtual memory space, enabling faster and more efficient file operations. It's particularly useful for large files and inter-process communication, as changes in memory are automatically reflected in the file. It simplifies file access and boosts system performance.

---

8. Allocating Kernel Memory

Methods:

- Buddy System
- Slab Allocation

Important Points:

- Fast and efficient.
- Avoids fragmentation.

Review Question: What are the two main methods for allocating kernel memory?
Answer:
The two main kernel memory allocation methods are:

1. **Buddy System**: Allocates memory in blocks of sizes that are powers of two. It allows quick allocation and deallocation, making it suitable for general-purpose allocation but may suffer from internal fragmentation.
2. **Slab Allocator**: Preallocates chunks (slabs) for frequently used objects. It reduces fragmentation and overhead by reusing existing memory blocks. It's widely used in Linux for kernel data structures.

---

9. Other Considerations

Key Terms:

- **Fragmentation**: Wasted memory space.
- **Swapping**: Moving processes in/out of RAM.

20/80 Insight: Learn difference between internal and external fragmentation.

Review Question: How does fragmentation affect memory usage?
Answer:
Fragmentation causes inefficient memory usage. **Internal fragmentation** happens when fixed-size memory blocks are allocated but not fully used, wasting the leftover space inside blocks. **External fragmentation** occurs when free memory is split into small chunks scattered across the system, making it hard to allocate large contiguous blocks. Both types reduce available usable memory, slow down performance, and can lead to allocation failures if not managed properly.

---

10. Storage Management

Concepts:

- Logical vs Physical Storage
- Disk Scheduling

Important Points:

- Efficient file placement improves speed.

Review Question: What is the difference between logical and physical storage?
Answer:
Logical storage refers to the way data is organized and accessed by users, like files and directories. Physical storage refers to the actual hardware (e.g., HDDs, SSDs) where the data resides. The operating system translates logical addresses (like file paths) into physical locations on disk using file systems. This abstraction allows users to interact with data without knowing how it's stored on hardware.

---

11. File Concepts

Key Terms:

- File: Named collection of data.

Attributes:

- Name, size, type, protection.

Operations:

- Create, read, write, delete.

Review Question: List common file operations and attributes.
Answer:
Common file operations include:

- **Create**: Making a new file.
- **Open**: Accessing an existing file.
- **Read**: Retrieving data from a file.
- **Write**: Adding or modifying data in a file.
- **Append**: Adding data to the end of a file.
- **Delete**: Removing a file from the system.
- **Rename**: Changing a file's name.

Key file attributes include:

- **Name**: Identifier of the file.
- **Type**: File format or extension (e.g., .txt, .jpg).
- **Location**: Path to where the file is stored.
- **Size**: Amount of data in the file.
- **Protection**: Access control settings like read/write/execute.
- **Timestamps**: Creation, modification, and access times.

These operations and attributes help in managing and organizing data effectively.

---

12. Access Methods

Types:

- Sequential Access
- Direct Access
- Indexed Access

20/80 Insight: Understand when to use sequential vs direct.

Review Question: Which access method is fastest for random file access?
Answer:
Direct Access is the fastest method for random access. It allows the system to jump directly to any location in the file without reading sequentially. This is especially useful for large files or databases where specific records need to be accessed quickly. Unlike sequential access, it avoids delay by not requiring a step-by-step read through the file.

---

13. Directory Structure

Types:

- Single-level
- Two-level
- Tree, DAG

Operations:

- Search, create, delete

Review Question: Compare single-level and tree directory structures.
Answer:

- **Single-Level Directory**: All files are stored in one directory. It is simple to implement but leads to filename conflicts and poor organization as the number of files increases.
- **Tree Directory**: Hierarchical structure where directories can have subdirectories. It supports better file organization, security, and scalability. Most modern OS use this due to its flexibility and manageability.

In summary, single-level is simple but limited, while tree structure is more advanced and preferred in practice.

---

14. File System Mounting

Key Terms:

- **Mount Point**: Where a new file system is connected.

Important Points:

- Needed to access external devices.

Review Question: What is file system mounting?
Answer:
Mounting a file system means making the files on a storage device (e.g., USB, hard drive) accessible to the operating system. This is done by attaching it to a directory (called a mount point) in the existing file system. After mounting, users can interact with the files just like any other part of the system. It is essential for using external storage or network file systems.

---

15. File Sharing

Key Points:

- Allows multi-user access.
- Needs control to prevent conflicts.

Review Question: How does OS handle file sharing between users?
Answer:
Operating systems handle file sharing by implementing access control mechanisms like:

- **File Permissions**: Define who can read, write, or execute a file.
- **Locks**: Prevent concurrent modifications that might lead to data corruption.
- **User and Group IDs**: Control access based on identity.
- **Access Control Lists (ACLs)**: Offer fine-grained permissions.

These features ensure data consistency and protect files during shared access by multiple users or processes.

---

16. Protection

Mechanisms:

- Permissions: read, write, execute
- User IDs, ACLs

Review Question: How is file protection implemented in OS?
Answer:
File protection is implemented through:

- **User Identification (UIDs)**: Ensures only authorized users can access files.
- **Permissions**: Set rules for read, write, and execute for user, group, and others.
- **Access Control Lists (ACLs)**: Define detailed access rights for specific users or groups.
- **Authentication and Authorization**: Validate user identity and permissions before granting access.
  These mechanisms help secure data from unauthorized access or accidental modifications.

---

17. Implementing File System

Structures:

- FCBs, inodes
- FAT (File Allocation Table)

Review Question: What is a file control block (FCB)?
Answer:
A File Control Block (FCB) is a data structure used by the operating system to store metadata about a file. This includes:

- File name and path
- File size
- File type
- Location of file data on disk
- Permissions and access rights
- Timestamps (created, modified, accessed)

The FCB acts like an ID card for the file, helping the OS manage file access, security, and performance.

---

18. File System Structure

Layers:

- Application > Logical FS > File Org > Basic FS > I/O

20/80 Insight: Understand separation between user-level and disk-level.

Review Question: Explain layered structure of file systems.
Answer:
File systems are structured in layers to improve modularity and simplify management:

1. **Application Layer**: User programs request file operations (e.g., open, save).
2. **Logical File System**: Handles directory structure, permissions, and file metadata.
3. **File Organization Module**: Maps files to disk blocks and manages free space.
4. **Basic File System**: Interacts with the physical storage through low-level operations.
5. **I/O Control Layer**: Includes drivers and hardware interface routines.

Each layer has a specific role and works together to manage files efficiently and securely.

---

19. File System Implementation

Structures Used:

- Boot Block, Directory Table, Allocation Table

Review Question: What data structures are used to implement file systems?
Answer:
Common data structures include:

- **Boot Block**: Contains OS boot instructions.
- **Superblock**: Stores file system configuration info.
- **Directory Table**: Maps file names to file locations.
- **File Allocation Table (FAT)/Inodes**: Track used and free blocks, and where each file's data is located.

These structures allow the file system to organize, locate, and manage files on disk efficiently.

---

20. Directory Implementation

Methods:

- Linear List
- Hash Table

Review Question: Compare linear list and hash table in directory implementation.
Answer:

- **Linear List**: Simple structure where each file entry is scanned one by one. Easy to implement but slow for large directories.
- **Hash Table**: Uses hash functions to map file names to locations, enabling fast access and updates.

Hash tables are faster but require more memory and good hash design. Linear lists are space-efficient but less performant for search-heavy tasks.

---

21. Allocation Methods

Types:

- **Contiguous**: Allocates a file in a single continuous block of memory. It offers fast access and simplicity but can suffer from external fragmentation and may struggle with growing files.
- **Linked**: Stores file blocks as a linked list scattered across the disk. It eliminates fragmentation but slows down random access since each block points to the next.

- **Indexed**: Uses a separate index block to store all addresses of the file blocks. This method supports fast random access and simplifies block management, making it suitable for large files.

Review Question: Why is indexed allocation better for random access?
Answer:
Indexed allocation keeps all block addresses of a file in an index block. This allows direct access to any block of the file without sequential traversal. It:

- Supports random access efficiently.
- Simplifies space management.
- Avoids problems of fragmentation.

This method is ideal for large files and databases where performance is critical.

---

22. Free Space Management

Methods:

- Bitmaps
- Linked Lists
- Grouping

Review Question: How does the OS track free space on disk?
Answer:
The OS uses several techniques to manage free space:

- **Bitmaps**: Each bit represents a disk block (0 = free, 1 = used).
- **Linked Lists**: Free blocks point to the next available one.
- **Grouping**: Stores addresses of free blocks in blocks themselves.
- **Counting**: Tracks the first free block and the number of contiguous free blocks.

These methods ensure efficient allocation and minimize fragmentation.

---

23. Efficiency and Performance

Factors:

- Access time
- Fragmentation
- Throughput

Review Question: What factors affect file system performance?
Answer:
Factors that influence performance include:

- **Access Time**: Speed to locate and retrieve files.
- **Throughput**: Number of file operations per second.
- **CPU Usage**: Overhead from file system operations.
- **Fragmentation**: Scattered data increases seek time.
- **Caching and Buffering**: Improve speed but consume memory.

Optimizing these factors results in faster and more reliable file systems.

---

24. Recovery

Techniques:

- **Journaling**: A technique where all file system changes are first written to a log (journal). This ensures that if the system crashes during a write operation, the system can use the journal to recover the last consistent state by redoing or undoing operations.
- **Checkpointing**: The system periodically saves its current state (a checkpoint) to a known good point. In the event of a crash, recovery can start from the last checkpoint instead of going back to the beginning, making recovery faster and less data-intensive.
- **Backup**: Creating a copy of data and system files to another storage medium (like external drives or cloud). It helps recover lost or corrupted data in case of hardware failure, accidental deletion, or software issues.

These techniques work together to protect the system against data loss and ensure smooth recovery after failures.

Review Question: How does journaling help in file system recovery?
Answer:
Journaling maintains a log (journal) of changes before committing them to the main file system. In case of a crash or power failure:

- The system checks the journal.
- Unfinished operations are either completed or rolled back.

This prevents corruption and ensures the file system remains consistent, reducing downtime and repair complexity.

Unit V: Secondary Storage Management and I/O System

---

1. Structure of Secondary Storage

Definition:
Secondary storage refers to non-volatile memory like HDDs, SSDs, and optical disks used for long-term data storage.

Key Concepts:

- Non-volatile (retains data after power off)
- Larger in size but slower than main memory

Review Question: What is secondary storage and how is it structured?
Answer:
Secondary storage holds data permanently. It uses various physical structures like platters and tracks (in HDDs) or blocks (in SSDs). The OS organizes this into sectors, tracks, and cylinders to manage files efficiently.

---

2. Overview of Mass Storage Structure

Definition:
Describes the hardware organization of large storage devices.

Components:

- Platters, heads, cylinders
- Spindle rotation for HDDs

Review Question: How is mass storage organized internally?
Answer:
Mass storage devices like HDDs are built with multiple platters and heads. Each platter has tracks divided into sectors. The controller manages read/write operations across platters, optimizing access with seek and rotational delays.

---

3. Disk Structure

Definition:
Refers to how data is physically organized on disks.

Key Terms:

- Sectors, tracks, cylinders
- Logical Block Addressing (LBA)

Review Question: What is logical block addressing in disk structure?
Answer:
LBA abstracts the disk into a linear array of blocks, simplifying addressing. It eliminates the need to refer to specific cylinder, head, or sector — enabling better compatibility and access.

---

4. Disk Attachments

Definition:
The method by which a disk is connected to the computer system.

Types:

- SATA, SCSI, NVMe, USB

Review Question: What are the common types of disk attachments?
Answer:
Disk attachments like SATA and NVMe determine speed and compatibility. SATA is common in personal computers, while NVMe offers high-speed data transfer in modern SSDs. SCSI is used in servers for reliability.

---

5. Disk Scheduling

Definition:
Algorithms used to decide the order of disk I/O requests.

Common Algorithms:

- FCFS
- SSTF
- SCAN
- LOOK

Review Question: Why is disk scheduling important?
Answer:
Disk scheduling optimizes access time by ordering requests efficiently. For example, SSTF (Shortest Seek Time First) reduces wait time by choosing the nearest request, improving system performance.

---

6. Disk Management

Definition:
Covers formatting, partitioning, and error detection of disks.

Tasks:

- Low-level formatting
- Logical formatting
- Bad block recovery

Review Question: What are the key responsibilities in disk management?
Answer:
Disk management ensures the disk is usable and reliable. It involves formatting to prepare the disk, partitioning to divide space, and detecting bad sectors to prevent data loss.

---

7. Swap Space Management

Definition:
Swap space is a portion of disk used as virtual memory.

Purpose:

- Temporarily hold inactive processes
- Extend RAM capacity

Review Question: What is swap space and how does it work?
Answer:
When RAM is full, the OS moves idle processes to swap space on disk. This keeps active processes in RAM and maintains system responsiveness. However, accessing disk is slower than RAM.

---

8. Space Management

Definition:
Techniques used to track and allocate free disk space.

Techniques:

- Bitmaps
- Linked Lists
- Grouping

Review Question: What are bitmap-based methods in space management?
Answer:
A bitmap uses 0s and 1s to represent free or used blocks. It’s compact and efficient for checking available space quickly. Each bit corresponds to a block on the disk.

---

9. RAID Structure

Definition:
RAID (Redundant Array of Independent Disks) improves performance and reliability.

Levels:

- RAID 0: Striping
- RAID 1: Mirroring
- RAID 5: Striping with parity

Review Question: What is RAID and how does RAID 5 work?
Answer:
RAID combines multiple disks into a single unit. RAID 5 distributes data and parity across disks. If one disk fails, the system can recover data using parity info, ensuring fault tolerance.

---

10. Table Storage Implementation

Definition:
Refers to managing large structured datasets in tables (used in databases).

Features:

- Efficient data access
- Support for indexing and relational queries

Review Question: How does table storage benefit data access?
Answer:
Table-based storage structures data into rows and columns, enabling fast lookups and structured queries, particularly in relational databases.

---

11. Tertiary Storage

Definition:
Refers to storage used for backup and archiving, like optical disks or tapes.

Characteristics:

- Low cost per GB
- Slow access time

Review Question: What is tertiary storage and when is it used?
Answer:
Tertiary storage is used for long-term storage of infrequently accessed data, like backups. It's cost-effective but slower compared to primary and secondary storage.

---

12. Structure of I/O Systems

Definition:
Organizes how I/O operations are handled within the OS.

Layers:

- User-level APIs
- Device drivers
- Interrupt handlers

Review Question: What are the main layers of an I/O system?
Answer:
I/O systems are layered to separate user interactions from hardware. APIs offer user-level access, drivers convert these to hardware instructions, and interrupt handlers manage I/O events.

---

13. I/O Hardware

Definition:
Devices and controllers that manage input/output operations.

Examples:

- Disk controllers
- Network cards
- USB controllers

Review Question: What is the role of I/O hardware?
Answer:
I/O hardware manages data exchange between the system and external devices. Controllers interpret OS commands and facilitate data transfer.

---

14. Application I/O Interface

Definition:
The interface through which applications request I/O operations.

Functions:

- File handling
- Device abstraction

Review Question: What is the purpose of an application I/O interface?
Answer:
It provides a consistent way for applications to perform I/O without needing to know hardware details. This abstraction improves portability and ease of use.

---

15. Kernel I/O Subsystem

Definition:
The OS component that manages I/O requests and coordinates with hardware.

Functions:

- Scheduling I/O
- Buffering
- Caching
- Spooling

Review Question: What services does the kernel I/O subsystem provide?
Answer:
It manages the lifecycle of I/O requests, improves performance via buffering and caching, and handles device-specific operations transparently to the user.

---

16. Transforming I/O Requests to Hardware Operations

Definition:
The process of converting high-level I/O calls into device-specific actions.

Steps:

- Translate file operation to block-level request
- Locate device
- Initiate controller action

Review Question: How are I/O requests transformed into hardware actions?
Answer:
The OS converts an application’s request (e.g., read a file) into a series of steps: identifying file blocks, locating them on disk, sending commands to the device controller, and returning results. This abstraction allows software to use I/O efficiently without knowing hardware details.
