Unit I: OVERVIEW AND PROCESS MANAGEMENT

Important 2-Mark Questions:

- **Define an Operating System.**1...
  - An operating system is software that manages computer hardware and provides an environment for application programs to run, acting as an intermediary between the user and hardware78. It also provides a basis for application programs8.
- **What is a process?**1...
  - A **process** is a program in execution111. It is the unit of work in a system, consisting of both operating-system processes (executing system code) and user processes (executing user code)9....
- **Differentiate between multiprogramming and time-sharing.**1...
  - **Multiprogramming** increases CPU utilization by organizing jobs (code and data) so that the CPU always has something to execute, keeping several jobs in memory simultaneously1416. **Time-sharing** is an extension of multiprogramming where CPU scheduling algorithms rapidly switch between jobs, creating the illusion that each job is running concurrently, thus providing quick response time to interactive users13....
- **List two components shared by threads belonging to the same process.**118
  - Threads within the same process share their **code section**, **data section**, and other operating-system resources like **open files** and **signals**18.
- **What is Inter-Process Communication (IPC)?**1...
  -

Inter-Process Communication (IPC) refers to mechanisms that allow cooperating processes to exchange information with each other1.... This can occur between processes on the same computer or across a network21.

Important 16-Mark Questions:

- **Elaborate on the components of a computer system and explain the fundamental role of an operating system as a resource manager and control program.**14...
  - A computer system consists of hardware, the operating system, application programs, and users22. The operating system manages computer hardware and provides a basis for application programs8. From the computer's perspective, the operating system acts as a **resource allocator**, managing CPU time, memory space, file-storage space, and I/O devices, allocating them to specific programs and users efficiently and fairly2326. It also functions as a **control program**, managing the execution of user programs to prevent errors and improper use of the computer, with particular concern for I/O devices24. Key aspects include multiprogramming for CPU utilization and time-sharing for user responsiveness14....
- **Explain the concept of a process, its various states, and the information maintained in a Process Control Block (PCB). Discuss the essential operations that an operating system performs on processes.**1...
  - A **process** is an instance of a program in execution11. As a process executes, it transitions through various states: **New**, **Running**, **Waiting**, **Ready**, and **Terminated**3134. Each process is represented in the operating system by a **Process Control Block (PCB)**, which contains information such as the process state, program counter, CPU registers, CPU-scheduling information, memory-management information, accounting information, and I/O status information28.... The operating system is responsible for critical process management activities, including **creating and deleting both user and system processes**, **suspending and resuming processes**, and **providing mechanisms for process synchronization and communication**1....
- **Discuss the various multi-threading models, such as Many-to-One, One-to-One, and Many-to-Many. Explain the common issues related to multithreaded programming, including fork() and exec() system calls, and thread cancellation.**1...
  -

A **thread** is a basic unit of CPU utilization, comprising a thread ID, program counter, register set, and stack, sharing code and data with other threads of the same process18. Multi-threading models describe the relationship between user threads and kernel threads:

▪

Many-to-One Model: Maps many user-level threads to a single kernel thread. Efficient for thread management in user space, but a blocking system call by one thread blocks the entire process, and threads cannot run in parallel on multiprocessors37.

▪

One-to-One Model: Maps each user-level thread to a kernel thread. Provides greater concurrency and allows multiple threads to run in parallel on multiprocessors. However, creating many user threads can be resource-intensive due to kernel overhead37. Windows XP, Solaris (since Solaris 9), and Linux use this model3944.

▪

Many-to-Many Model: Multiplexes many user-level threads to a smaller or equal number of kernel threads. Allows developers to create many user threads and enables parallel execution on multiprocessors. When a thread blocks, the kernel can schedule another thread38.

◦

Key **threading issues** include:

▪

fork() and exec() System Calls: When fork() is called, some UNIX systems offer two versions: one that duplicates all threads and one that duplicates only the invoking thread41. exec() typically replaces the entire process, including all threads41.

▪

Thread Cancellation: The process of terminating a thread before it has completed its task, which can be asynchronous (terminates immediately) or deferred (checks periodically if it should terminate)40.

Unit II: PROCESS SCHEDULING AND SYNCHRONIZATION

Important 2-Mark Questions:

•

List any two CPU scheduling criteria.245

◦

CPU scheduling criteria include **CPU utilization**, **throughput**, **turnaround time**, **waiting time**, and **response time**4546.

•

What is the critical-section problem?247

◦

The **critical-section problem** is to design a protocol that cooperating processes can use to ensure that when one process is executing in its critical section (where shared data is accessed), no other process is allowed to execute in its critical section247.

•

Define a semaphore.248

◦

A **semaphore** is a synchronization tool that provides a simple but effective solution to the critical-section problem. It is an integer variable accessed only through two atomic operations: wait() and signal()4849.

•

List the four necessary conditions for deadlock.250

◦

The four necessary conditions for deadlock are **mutual exclusion**, **hold and wait**, **no preemption**, and **circular wait**250.

•

What is a safe state in deadlock avoidance?5152

◦

A system is in a **safe state** if there exists a **safe sequence** of processes for the current allocation of resources. A safe sequence is a sequence of all the processes in the system such that for each process Pi, the resources that Pi can still request can be satisfied by the currently available resources plus the resources held by all processes Pj, where j < i5152.

Important 16-Mark Questions:

•

Explain various CPU scheduling algorithms, including First-Come, First-Served (FCFS), Shortest-Job-First (SJF), Priority Scheduling, Round Robin (RR), and Multilevel Feedback Queues. Provide examples and discuss their trade-offs.2...

◦

CPU scheduling determines which process in the ready queue is allocated the CPU62.

▪

FCFS: Processes are executed in the order they arrive. Simple to implement but can lead to long waiting times for short processes (convoy effect)46....

▪

SJF: Associates with each process the length of its next CPU burst and schedules the process with the shortest next CPU burst. Optimal for minimizing average waiting time but difficult to implement as burst times are hard to predict34....

▪

Priority Scheduling: Each process is assigned a priority, and the CPU is allocated to the process with the highest priority. Can suffer from starvation if low-priority processes never get to run5563.

▪

Round Robin (RR): Similar to FCFS but adds preemption using a time quantum. Each process gets a small unit of CPU time (time quantum) and if it doesn't complete within the quantum, it's preempted and added to the end of the ready queue56....

▪

Multilevel Feedback Queues: Allows processes to move between queues based on their CPU burst characteristics, favoring I/O-bound processes over CPU-bound processes. Highly configurable but complex to tune60....

◦

Trade-offs involve criteria like CPU utilization, throughput, turnaround time, waiting time, and response time45.

•

Discuss the process synchronization problem, its requirements, and explain solutions using synchronization hardware (TestAndSet, Swap) and semaphores. Illustrate with classical problems of synchronization like the Bounded-Buffer Problem or Readers-Writers Problem.2...

◦

Process Synchronization is essential when processes cooperate and share data, to ensure data consistency and avoid race conditions75.... The **critical-section problem** ensures that only one process executes its critical section at a time47. Solutions must satisfy **mutual exclusion**, **progress**, and **bounded waiting**81.

◦

Synchronization Hardware: Modern systems provide special atomic hardware instructions like **TestAndSet()** or **Swap()** that can test and modify a word or swap contents of two words as an uninterruptible unit, enabling simple mutual exclusion76....

◦

Semaphores: A synchronization tool using wait() (decrement) and signal() (increment) operations to control access to shared resources48. Binary semaphores (mutex locks) are used for mutual exclusion49.

◦

Classical Problems of Synchronization:

▪

Bounded-Buffer Problem: Producers produce items and consumers consume them using a shared buffer of fixed size, requiring synchronization to prevent overflow/underflow and concurrent access72.

▪

Readers-Writers Problem: Multiple readers can access shared data concurrently, but only one writer can access the data at a time, and no reader can access while a writer is writing73.

•

Define deadlock and elaborate on the various methods for handling deadlocks, including prevention, avoidance (Banker's Algorithm), detection, and recovery.2...

◦

A **deadlock** is a situation where two or more processes are blocked indefinitely, waiting for each other to release resources2. It occurs when four conditions hold simultaneously: **mutual exclusion**, **hold and wait**, **no preemption**, and **circular wait**50.

◦

Methods for Handling Deadlocks:90

▪

Deadlock Prevention: Ensures that at least one of the four necessary conditions for deadlock cannot hold:

-

Mutual Exclusion: Not required for sharable resources (e.g., read-only files)91.

-

Hold and Wait: Processes must request all resources before starting execution, or release all held resources before requesting new ones91.

-

No Preemption: If a process holding resources requests another that cannot be immediately allocated, it must release all held resources. Preempted resources are added to the list of resources for which the process is waiting92.

-

Circular Wait: Impose a total ordering of all resource types and require each process to request resources in an increasing order of enumeration93.

-

Deadlock Avoidance: Requires the operating system to be given in advance information about which resources a process will request107. The **Banker's Algorithm** is a common avoidance scheme that determines if granting a request leaves the system in a safe state5152.

-

Deadlock Detection: Allows the system to enter a deadlock state and then detects it, typically using a **wait-for graph**. It involves an algorithm that periodically checks for cycles in the graph101....

-

Recovery from Deadlock: Once a deadlock is detected, the system must recover, often by **process termination** (aborting one or more deadlocked processes) or **resource preemption** (taking resources from one process and giving them to another)104....

Unit III: MEMORY MANAGEMENT

Important 2-Mark Questions:

-

What is swapping?3...

◦

Swapping is a memory management technique where a process can be temporarily removed from main memory (swapped out) to a backing store (disk) and then brought back into memory (swapped in) to continue execution109.... This is often managed by a medium-term scheduler to improve process mix or free up memory110.

•

Distinguish between logical and physical address space.90

◦

A **logical address** is an address generated by the CPU90. A **physical address** is an address seen by the memory unit (i.e., the one loaded into the memory-address register)90. In systems with virtual memory, the logical address space can be much larger than the physical address space.

•

Define internal and external fragmentation.105

◦

Internal fragmentation occurs when allocated memory is slightly larger than requested memory, and the unused part within the allocated block is still assigned to the process and cannot be used by other processes105. **External fragmentation** occurs when total memory space exists to satisfy a request, but it is not contiguous; memory is fragmented into small, non-contiguous holes105112.

•

What is demand paging?3...

◦

Demand paging is a virtual memory technique that loads pages into physical memory only when they are needed or "demanded" during program execution, rather than loading an entire process at startup114115. This reduces I/O and memory requirements for processes113.

•

Explain thrashing in virtual memory.3...

◦

Thrashing occurs when a process spends more time paging (swapping pages between memory and disk) than executing instructions117. This happens when a process does not

have enough frames allocated to it to hold all the pages it is actively using (its working set), leading to a very high page-fault rate and significant performance degradation116117.

Important 16-Mark Questions:

- **Describe various memory management techniques such as contiguous memory allocation, pure paging, and pure segmentation. Compare and contrast their advantages and disadvantages, including issues like fragmentation and sharing.**3...
  - **Memory Management** is crucial for sharing memory among multiple processes to improve CPU utilization and response time84.
    - **Contiguous Memory Allocation:** Each process is loaded into a single, contiguous section of memory93. This can be implemented with fixed-sized or variable-sized partitions9596.
      - *Advantages:* Simple to implement.
      - *Disadvantages:* Suffers from **external fragmentation**, where available memory is scattered in small non-contiguous blocks, making it difficult to allocate large processes105112. Sharing code is difficult.
    - **Paging:** Divides physical memory into fixed-size blocks called **frames** and logical memory into blocks of the same size called **pages**3119. A **page table** maps logical pages to physical frames119.
      - *Advantages:* Eliminates external fragmentation, allows non-contiguous allocation, and facilitates sharing of common code (e.g., pure code)105....
      - *Disadvantages:* Can suffer from **internal fragmentation** (last page may not be full)105. Increased context-switch time due to page table reloads99.
    -

Segmentation: Supports the user's view of memory as a collection of variable-sized logical units called **segments**, each with a name and length3103. Logical addresses consist of a segment name and an offset103.

-

Advantages: Reflects the user's logical view of program, facilitates sharing of code and data between processes, and supports dynamic memory allocation.

-

Disadvantages: Suffers from **external fragmentation**105. Memory allocation is more complex due to variable segment sizes.

-

Segmentation with Paging: Combines segmentation and paging, where each segment is further divided into pages3105. This scheme typically maps segments to a paged memory space105121.

-

Advantages: Combines the benefits of both (logical view of segmentation, no external fragmentation of paging)105.

-

Disadvantages: Increased complexity and overhead for address translation.

-

Explain the concept of virtual memory and demand paging. Describe and compare different page replacement algorithms (e.g., FIFO, LRU, Optimal, Clock algorithm) and discuss how their performance is evaluated.3...

-

Virtual Memory is a technique that allows the execution of processes that are not entirely in main memory, separating the user's logical memory from physical memory3122. It enables processes to run larger than physical memory and supports multiprogramming122.

-

Demand Paging is the primary implementation of virtual memory, loading pages into memory only when they are actively referenced114. A **page fault** occurs when a referenced page is not in memory, triggering the operating system to load it from disk126.

-

Page Replacement Algorithms are needed when a page fault occurs and there are no free frames in memory3123. The goal is to select a page to replace to minimize page faults:

▪

First-In, First-Out (FIFO): Replaces the page that has been in memory the longest. Simple but may remove actively used pages135....

▪

Optimal Page Replacement (OPT): Replaces the page that will not be used for the longest period of time. Guarantees the lowest page-fault rate but is impossible to implement in practice as it requires future knowledge136....

▪

Least Recently Used (LRU): Replaces the page that has not been used for the longest period of time. A good approximation of optimal but difficult to implement precisely due to tracking usage history138....

▪

LRU-Approximation Algorithms (e.g., Clock / Second-Chance): Use a reference bit to approximate LRU, giving pages a "second chance" before replacement140.... Solaris uses a modified two-hand clock algorithm120.

◦

Performance Evaluation: Algorithms are typically evaluated by running them on a given **page reference string** and counting the number of **page faults**134145. **Effective access time** considers the memory access time and the page-fault rate146. Simulations with trace tapes or queueing models can also be used67147.

•

Discuss the allocation of frames in virtual memory. Explain the phenomenon of thrashing, its causes, and various techniques an operating system can employ to detect and eliminate this problem.3...

◦

Allocation of Frames concerns how to distribute the fixed number of available memory frames among competing processes in a demand-paged system3....

▪

Fixed Allocation: Each process is given a fixed number of frames, either equally or proportionally based on size/priority149.

▪

Dynamic Allocation: Allows processes to gain or lose frames dynamically based on their current needs. This includes **global replacement** (any page in memory is a candidate for replacement) and **local replacement** (only pages belonging to the faulting process are candidates)156.

◦

Thrashing: A situation where a system spends most of its time swapping pages in and out, rather than doing useful work3117. This occurs when the total demand for frames by all processes exceeds the total number of available physical frames (D > m)116151. Each process lacks enough memory to hold its **working set** (the set of pages actively used), leading to continuous page faults157.

◦

Detecting and Eliminating Thrashing:

▪

Page-Fault Frequency (PFF): Monitor the page-fault rate. If it's too high, the process needs more frames; if too low, it has too many frames154.

▪

Working-Set Model: Dynamically allocates enough frames to a process to accommodate its current locality of reference112.... By estimating the working-set size (WSS) for each process, the system can ensure D <= m (total demand less than or equal to available frames)112158. If D exceeds m, some processes may need to be suspended (swapped out) to reduce the degree of multiprogramming158.

Unit IV: FILE SYSTEMS

Important 2-Mark Questions:

•

Define the file concept.3...

◦

The **file concept** is a uniform, logical view of information storage provided by the operating system, abstracting from the physical properties of storage devices to define a logical storage unit140159.

•

List two common file access methods.3128

◦

Two common file access methods are **sequential access** (data is accessed in order, one record after another) and **direct access** (records are accessed directly without reading preceding records)128.

•

What is file system mounting?3...

◦

File system mounting is the process by which an operating system makes a file system on a storage device (volume) available for use, typically by attaching it to a specific directory (mount point) in the existing file system hierarchy3....

•

Distinguish between mandatory and advisory file locking.161

◦

With **mandatory file locking**, the operating system enforces the lock: if a process acquires an exclusive lock, no other process can access the locked file until it's released161. With **advisory file locking**, the operating system does not enforce the lock; it's up to software developers to explicitly acquire and release locks before accessing the file161. Windows typically uses mandatory locking, while UNIX systems use advisory locks161.

•

What is free-space management in file systems?3...

◦

Free-space management is the process by which the operating system keeps track of all the free disk blocks available for allocation to files. This is crucial for efficient disk utilization3....

Important 16-Mark Questions:

•

Explain the concept of a file, its attributes, and different access methods. Describe various directory structures used in operating systems, illustrating with diagrams.3...

◦

A **file** is an abstract data type defined and implemented by the operating system, representing a sequence of logical records140159. **File attributes** include name, identifier, type, location, size, protection, time, and user identification167.

◦

Access Methods:

▪

Sequential Access: Data is processed in order, useful for tapes and many file processing tasks128.

▪

Direct Access: Records can be read or written rapidly in any order, useful for databases128.

▪

Other methods include **indexed access** (built upon direct access, using an index to quickly locate records)143.

◦

Directory Structure: Organizes and provides information about all files in the system164. The directory acts as a symbol table mapping file names to directory entries132. Common structures include:

▪

Single-Level Directory: All files in one directory. Simple but suffers from naming conflicts and difficulty in grouping files for multiple users133.

▪

Two-Level Directory: Each user has their own User File Directory (UFD) for their files, resolving naming conflicts between users but not for files from different projects by the same user179.

▪

Tree-Structured Directory: A hierarchical structure where directories can contain files or subdirectories, allowing users to create their own subdirectories. Offers efficient searching and grouping but links may cause issues180.

▪

Acyclic-Graph Directory: Allows directories to share files and subdirectories, providing flexibility but introducing challenges like dangling pointers (when shared files are deleted)182184.

▪

General Graph Directory: Allows cycles, making directory traversal and garbage collection more complex185.

•

Discuss how file systems are implemented, focusing on their layered structure, directory implementation, different allocation methods (contiguous, linked, indexed), and free-space management techniques.3...

◦

File systems have a **layered design**117. Key layers include **I/O control** (device drivers), **basic file system** (reads/writes physical blocks), **file-organization module** (maps logical blocks to physical blocks), and **logical file system** (manages metadata like directories and FCBs)117151.

◦

On-disk structures include the boot control block, volume control block, directory structure, and File-Control Blocks (FCBs) for each file153154. **In-memory structures** include a system-wide open-file table and a per-process open-file table154.

◦

Allocation Methods (how disk blocks are allocated to files):

▪

Contiguous Allocation: Each file occupies a contiguous set of blocks on disk. Simple for sequential and direct access, but suffers from external fragmentation and difficulty in growing files198.

▪

Linked Allocation: Each block contains a pointer to the next block. No external fragmentation, but only supports sequential access and susceptible to pointer loss201. The **File-Allocation Table (FAT)** improves linked allocation by storing pointers in a table on disk202209.

▪

Indexed Allocation: Each file has an index block (inode in UNIX) containing pointers to all its data blocks. Supports direct access, no external fragmentation, but overhead of index block and potential for large index blocks for very large files187204. This can be extended with multi-level indexing (single, double, triple indirect blocks)187.

◦

Free-Space Management: Methods to keep track of available disk blocks:

▪

Bit Map: A bit vector where each bit represents a disk block (1=allocated, 0=free)163. Efficient for finding contiguous blocks but requires significant memory for large disks210.

-

Linked List: Free disk blocks are linked together, with the first free block pointing to the next, and so on206. Simple but inefficient for finding contiguous blocks.

-

Grouping/Counting: Variations to improve efficiency by storing pointers to multiple free blocks or counting consecutive free blocks206.

-

Elaborate on file sharing and protection mechanisms in multi-user operating systems. Discuss access control lists and capabilities.3...

  -

File Sharing: In multi-user systems, sharing files is highly desirable for collaboration and resource efficiency136. Issues include consistency semantics (e.g., UNIX consistency semantics for shared access)215227.

  -

File Protection: Mechanisms to control who can access files and how they can be accessed164....

-

Access Control: The system mediates file sharing, either by default allowing access or requiring explicit grants138. Most systems use a **protection domain** (a set of objects and the access rights to those objects)222.

-

Access Control List (ACL): For each object (file), a list specifies the domains and the access rights for each domain223228. It explicitly lists which users or groups have what permissions (e.g., read, write, execute)218219. UNIX uses a simplified ACL with owner, group, and other permissions220.

-

Capability List: For each domain (process or user), a list of objects that can be accessed along with the operations allowed on those objects223.... Capabilities are unforgeable tickets that permit certain operations226.

-

Lock-Key Mechanism: A compromise where each object has a list of "locks" (bit patterns) and each domain has a list of "keys." A process can access an object if its domain has a key matching one of the object's locks224.

Unit V: I/O SYSTEMS

Important 2-Mark Questions:

- **List two categories of I/O devices.**215
  - I/O devices broadly fit into categories such as **storage devices** (e.g., disks, tapes), **transmission devices** (e.g., network cards, modems), and **human-interface devices** (e.g., screen, keyboard, mouse)215.
- **What is the purpose of a device driver?**117214
  - A **device driver** is a kernel module that encapsulates the details and oddities of specific hardware devices, presenting a uniform device-access interface to the I/O subsystem214. It translates high-level commands into low-level, hardware-specific instructions117.
- **Define buffering in I/O systems.**229
  - **Buffering** is a kernel service that uses memory to temporarily store data during I/O transfers, accommodating data transfer rate mismatches, providing copy semantics for application data, and facilitating scatter-gather operations229230.
- **What is disk scheduling?**4...
  - **Disk scheduling** refers to the operating system's management of the order in which disk I/O requests are serviced, with the goal of improving performance (e.g., reducing access time and increasing bandwidth)4....
- **What is RAID?**4...
  -

RAID (Redundant Arrays of Inexpensive Disks) is a mass-storage structure that uses multiple disks to provide data redundancy (for reliability) and/or improve performance (through data striping)4....

Important 16-Mark Questions:

-

Describe the architecture of I/O hardware, including I/O ports, buses, and device controllers. Explain the role of interrupts and Direct Memory Access (DMA) in I/O operations.4...

  -

I/O Hardware consists of various components:

  -

I/O Devices: Vary widely in function and speed (e.g., mouse, disk, network card)17....

  -

I/O Port: A collection of registers used for communication between the CPU and device controllers216237.

  -

Bus: A common set of wires connecting the CPU, main memory, and I/O devices189236.

  -

Device Controller: An electronic component that operates a port, bus, or device, containing control registers, a data buffer, and special-purpose processors189.... It performs data transfer between the device and its local buffer238.

  -

I/O Operations:

  -

Polling: The host CPU repeatedly checks the busy bit of the device controller to see if it's ready for the next command239. Simple but can waste CPU cycles246.

  -

Interrupts: A hardware mechanism where a device controller asserts a signal on the interrupt request line, causing the CPU to stop its current task, save its state, and jump to an interrupt handler routine to service the device217.... This allows the CPU to perform other work while waiting for I/O248. Modern systems use interrupt priority levels218....

▪

Direct Memory Access (DMA): For large data transfers, a DMA controller is used to transfer blocks of data directly between device buffers and main memory without CPU intervention, reducing CPU overhead and improving performance189.... The CPU is only interrupted once per block or burst to acknowledge completion243249.

•

Explain the application I/O interface and the kernel I/O subsystem, including services like buffering, caching, spooling, device reservation, and error handling. Discuss I/O performance considerations.4...

◦

Application I/O Interface: Provides a standard, uniform way for applications to access I/O devices, abstracting away hardware differences191. Common categories include block I/O, character-stream I/O, memory-mapped file access, network sockets, and timers193. System calls (read(), write(), open(), close()) are the primary interface194.... Both blocking and nonblocking I/O are supported196197.

◦

Kernel I/O Subsystem: Coordinates an extensive collection of services available to applications and other parts of the kernel250267.

▪

I/O Scheduling: Determines a good order to execute I/O requests, maintaining a wait queue for each device to improve overall system efficiency and response time198199.

▪

Buffering: Using memory to hold data during transfers to cope with speed mismatches or differences in transfer unit sizes229.

▪

Caching: Storing copies of data in faster storage (e.g., main memory) for quicker access. A unified buffer cache can manage both process pages and file data using virtual memory techniques251264.

▪

Spooling: Buffering output for devices that cannot accept interleaved data streams (e.g., printers), with each application's output spooled to a separate disk file and then printed sequentially202.

▪

Device Reservation: Providing explicit mechanisms for exclusive device access for devices that cannot multiplex requests (e.g., tape drives)203.

▪

Error Handling: Detecting and compensating for transient failures (e.g., network overload, disk read errors) and handling permanent failures, often providing error codes to applications204205.

▪

Protection: Ensuring that user programs cannot directly access I/O devices or memory-mapped I/O ports without proper authorization163206.

◦

I/O Performance: A major factor in overall system performance269. It places heavy demands on the CPU (device-driver code, context switches), memory bus (data copies), and interrupt-handling mechanisms. Optimizing performance involves balancing CPU, memory, bus, and I/O performance256257.

•

Discuss Mass-Storage Structure, including various disk-scheduling algorithms (e.g., FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK), disk management (formatting, boot blocks, bad-block recovery), and RAID levels.4...

◦

Mass-Storage Structure: Disks are the primary secondary-storage devices, structured as large one-dimensional arrays of logical blocks170....

◦

Disk Scheduling Algorithms: Manage the order of disk I/O requests to optimize performance, primarily by minimizing seek time231....

▪

FCFS (First-Come, First-Served): Simplest, processes requests in the order they arrive284306.

▪

SSTF (Shortest-Seek-Time-First): Selects the request with the minimum seek time from the current head position. Can lead to starvation285306.

▪

SCAN (Elevator Algorithm): The disk arm moves from one end of the disk to the other, servicing requests along the way, then reverses direction286306.

▪

C-SCAN (Circular-SCAN): Similar to SCAN but when the arm reaches one end, it immediately returns to the other end without servicing requests on the return trip, ensuring more uniform wait times287306.

▪

LOOK/C-LOOK: Variations of SCAN/C-SCAN where the arm only travels as far as the furthest request in each direction before reversing, instead of going all the way to the end of the disk288306.

◦

Disk Management:

▪

Disk Formatting: Low-level formatting divides the disk into sectors for the controller to read/write278289. **Partitioning** divides the disk into one or more logical partitions307308.

▪

Boot Blocks: Special blocks on disk storing the **bootstrap program** (bootstrap loader) that loads the operating system kernel into memory at system startup307....

▪

Bad-Block Recovery: Mechanisms to handle sectors that are corrupted (bad sectors), often by **sector sparing** (remapping bad sectors to spare sectors) or **sector slipping**292311.

▪

Swap-Space Management: Management of disk space used as a backing store for virtual memory, typically using raw disk partitions for efficiency4....

◦

RAID (Redundant Arrays of Inexpensive Disks):4...

▪

RAID levels combine disk striping with parity bits to achieve varying cost-performance trade-offs in terms of reliability and data transfer rates234.... Examples include RAID Level 0 (striping, no redundancy), RAID Level 1 (mirroring), and RAID Level 5 (striping with distributed parity)296....

•

Case Study: Implementation of Distributed File System in Cloud OS / Mobile OS.4

◦

While the syllabus mentions "Cloud OS / Mobile OS," the provided sources don't delve into specific case studies for these. However, they discuss **Distributed File Systems (DFS)** concepts which are applicable6....

◦

Distributed File Systems (DFS): A file-service system whose users, servers, and storage devices are dispersed among the sites of a distributed system312320.

◦

Goals/Benefits: Provide access to remote resources, improve computation speed, and enhance data availability and reliability295312. They aim for **transparency**, making remote resources appear local322323.

◦

Implementation Issues:

▪

Naming and Transparency: How files are named and how location transparency (users unaware of where files are stored) or location independence (file names don't indicate physical storage) is achieved317322.

▪

Remote File Access: Strategies like transferring the entire file (data migration) or just accessing parts on demand, often involving caching at the client or server300315.

▪

Stateful vs. Stateless Service: Whether the server maintains client state information .

▪

File Replication: Maintaining multiple copies of files for availability and reliability .

▪

Consistency Semantics: Defining the consistency of shared files (e.g., UNIX consistency semantics)215.

◦

Challenges: Dealing with network failures, data consistency, concurrency control, and ensuring robustness298312.

◦

Example (AFS): The Andrew File System (AFS) is a distributed file system that focuses on client-side caching and shared name spaces for scalability and performance, addressing heterogeneity issues316.