**PART 2.2**

1. Propose a solution by using semaphores and wait(s)/signal(s) semaphore operations, to the first readerwriter problem, that is, no reader will be kept waiting unless a writer has already obtained permission to use the shared object. What is the major problem with your solution?

**Answer: In Lecture Notes Part 2.2**

metin, ekran görüntüsü, yazı tipi, sarı içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

Major problem: Starvation of writer processes, if continuous execution too many reader processes.

\*Starvation of writer processes: If there is a continuous stream of reader processes, writers may be indefinitely postponed. As long as at least one reader is active, wrt remains locked, and new readers can keep arriving, preventing writers from ever gaining access.

2. What is the most important principle to execute the procedures of a Hoare’s Monitor, compared to execution of procedures (methods) of a class/object.?

Answer: In Lecture Notes Part 2.2, procedures of Hoare’s Monitor must be executed mutually

exclusively. When a process is in the monitor, executing a monitor procedure, no other process can

execute the same or another monitor procedure. This contrasts with procedures (methods) of a class or object in general programming, which do not inherently enforce mutual exclusion unless explicitly implemented

3. Regarding deadlocks in operating systems, draw two example resource allocation graphs, one with a cycle and deadlock, and the other with a cycle but no deadlock. Then, state the relationship between cycles in resource allocation graphs and deadlock.

Answer: In Lecture Notes Part 2.2

diyagram, çizgi, ekran görüntüsü, daire içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

- For systems with **single-instance resources**, a cycle in the Resource Allocation Graph (RAG) implies a deadlock, and a deadlock implies a cycle (i.e., cycle ↔ deadlock).

- For systems with **multiple-instance resources**, a deadlock implies a cycle in the RAG, but a cycle does not necessarily imply a deadlock. The presence of additional resource instances can allow processes to proceed, avoiding deadlock despite a cycle.

4. What is the difference between a resource allocation graph and a wait for graph, used in deadlock handling in operating systems

Answer: In Lecture Notes Part 2.2 In resource allocation graph, both processes and resources are

represented, whereas in wait for graph, only processes are represented.

Key Difference: The RAG includes both processes and resources, providing a complete picture of allocation and requests, while the WFG focuses solely on process waiting relationships, abstracting away resources

5. Make a list of four conditions for deadlock to occur in an operating system

Answer: In Lecture Notes Part 2.2

\* Mutual Exclusion Condition Resources are not shared.

\* Hold and Wait Condition Process P has resource Ri, waits for Rj while holding Ri

\* No-Preemption Condition Process Pk has resource Ri, waits for Rj held by Pm. Pk cannot forcefully terminate (preempt) Pm to get Rj

\* Circular Wait Condition

6. Regarding deadlocks in operating systems, just name the two main disadvantages if processes are required to request and be allocated all their resources before they begin execution, so that deadlock problem is totally solved.

Answer: In Lecture Notes Part 2.2

1. Low resource resource utilization. A process may allocate a resource that it will use quite a long time later in its execution.
2. (ii) Starvation of a process that cannotallocate all its resources before it begins execution

7. In deadlocak avoidance, while one uses Dijktra’s Banker’s algorithm, what is the meaning of a safe OS state?

Answer: In Lecture Notes Part 2.2, A safe OS state will mean that future dynamic resource allocation requests will not lead to a deadlock, therefore resource request can be granted.

8. In deadlock avoidance, while one uses Dijktra’s Banker’s algorithm, what is the meaning of an unsafe OS state?

Answer: In Lecture Notes Part 2.2, An unsafe OS state will mean that future dynamic resource allocation requests may lead to a deadlock , therefore resource request will not be granted.

9. What is the difference between deadlock prevention and deadlock avoidance?

Answer: In Lecture Notes Part 2.2, In deadlock prevention, possibility of deadlock is totally removed by preventing dynamic resource allocations, whereas in deadlocak avoidance, dynamic resource allocations are permitted as long as they are within the declared resource needs.

10. What do we mean by “Ostrich Approach” in deadlock handling in operating systems?

Answer: In Lecture Notes Part 2.2,

The Ostrich Approach refers to ignoring the deadlock problem entirely within the operating system. No prevention, avoidance, or detection mechanisms are implemented, and it’s left to application developers to ensure their programs avoid deadlocks. This is often used in professional OSs where deadlock handling is deemed too complex or costly, and rare deadlocks are handled by manual intervention

**PART 3 Memory Management**

1. What is the main goal of paged memory management?

Answer: In Lecture Notes Part 3, The main goal of paged memory management is to solve the memory fragmentation problem, without moving allocated partitions up/down in main memory

2. What are the two main goals of segmented memory management?

Answer: In Lecture Notes Part 3, Memory Sharing and Dynamic Linking/Loading

\*Memory Sharing: Allows multiple processes to share segments (e.g., code or data), improving resource efficiency.

\*Dynamic Linking/Loading: Enables segments to be loaded and linked dynamically, facilitating modular programming and library use.

3. By drawing a diagram of the logical address to physical address translation mechanism and specifically showing memory protection components, show how paged memory management technique operates.

Answer: In Lecture Notes Part 3

metin, el yazısı, yazı tipi, mürekkep içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, el yazısı, yazı tipi, diyagram içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

4. In paged memory management, Associative Memory (or TLB: Translation Lookaside Buffer) is used to speed up the translation of a logical page number into a physical page number. (i) Explain briefly how such speed up is achieved in Associative Memory. (ii)Assume main memory access time is 40 nanoseconds and logical page number is not found in TLB. What is the actual memory access time then? And why?

Answer: In Lecture Notes Part 3

\*Speed Up with TLB: The TLB is a small, fast cache that stores recent page table entries. When a logical address is generated, the TLB is checked first. If the page number is found (a hit), the corresponding frame number is retrieved directly, bypassing the slower main memory access to the page table, thus speeding up translation

\* Memory Access Time without TLB Hit:

Given: Main memory access time = 40 ns, TLB miss.

Calculation: On a TLB miss, the system accesses the page table in main memory (40 ns). Then, it accesses the actual data using the frame number (another 40 ns).

Total time = 40 ns (page table) + 40 ns (data) = 80 ns.

5. In virtual memory management, list main operations of OS Page Fault Handler

Answer: In Lecture Notes Part 3

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

.Check Validity: Verify if the referenced page is valid (part of the process’s address space); if not, terminate the process or handle the error.

. Find Free Frame: Check for a free frame in physical memory (PAS).

. Page Replacement (if needed): If no free frame exists, run a page replacement algorithm to select a victim page.

. Write Victim Page (if dirty): If the victim page is modified, write it to the swap area on disk.

. Retrieve Page: Load the requested page from the swap area into the free or victim frame.

. Update Page Table: Modify the page table to map the logical page to the physical frame.

. Restart Instruction: Resume the process at the instruction that caused the page fault.

6. In virtual memory management, draw the Belady’s curve and explain what it means.

Answer: In Lecture Notes Part 3

metin, ekran görüntüsü, çizgi, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

\*Belady’s Curve shows that as the number of frames in PAS increases, the number of page faults decreases.

\*A big drop occurs with initial increases in frames (when PAS is small), but the drop becomes smaller as PAS grows larger.

\*When PAS equals LAS, page faults reach zero (no swapping needed).

\*Belady predicted PAS size should be half of LAS for a good balance between fault rate and memory use.

7. In virtual memory management, optimal page replacement states that OS should replace the page that will not be used in future the longest period of time. What do you think about this approach?.

Answer: In Lecture Notes Part 3. Your reasoning will be the answer

\*The optimal page replacement algorithm is theoretically the best, minimizing page faults by replacing the page that won’t be needed for the longest time.

\*However, it’s impractical because it requires knowledge of future page references, which the OS cannot predict in real-time.

\*It serves as a benchmark to evaluate other algorithms (e.g., FIFO, LRU), but cannot be implemented in practice due to its reliance on future-telling.

8. In virtual memory management, what is the assumption of LRU page replacement approach?

Answer: In Lecture Notes Part 3, If a page is not used recently, it will not be used in future..!

The assumption of the Least Recently Used (LRU) page replacement approach is that if a page has not been used recently, it will not be used in the near future. This is based on the principle of temporal locality, where recently accessed pages are likely to be accessed again soon, making the least recently used page a good candidate for replacement.

9. In paged memory management, What is the use reference bit?

Answer: In Lecture Notes Part 3

\*The reference bit in a page table entry is set to 1 when a page is accessed (read or written) and 0 otherwise. It tracks whether a page has been used recently, aiding page replacement algorithms (e.g., LRU approximations like Second Chance) in identifying active pages versus those that can be replaced.

10. In paged memory management, What is the use dirty bit?

Answer: In Lecture Notes Part 3

\*The dirty bit (or modify bit) in a page table entry is set to 1 when a page is modified (written to) and 0 if unchanged. It indicates whether the page differs from its disk copy. If a page is selected for replacement and the dirty bit is 1, it must be written back to disk; if 0, it can be discarded, reducing disk I/O

11. What is thrashing? Explain

Answer: In Lecture Notes Part 3

\*Thrashing occurs when the OS spends more time swapping pages in and out of memory than executing process instructions, leading to low CPU utilization.

\*It happens when the working set of processes exceeds available physical memory, causing frequent page faults and excessive disk I/O. This can result from too high a degree of multiprogramming, poor page replacement, or unusual reference patterns.

12. What is Spatial Locality? Explain

Answer: In Lecture Notes Part 3

\*Spatial Locality is the principle that if a memory location is accessed, nearby locations are likely to be accessed soon after. This arises because programs often access data sequentially (e.g., arrays) or instructions in order, except for jumps. It’s leveraged in memory management by prefetching or caching blocks of nearby data.

13. What is TemporalLocality? Explain

Answer: In Lecture Notes Part 3

\* Temporal Locality is the principle that if a memory location is accessed, it is likely to be accessed again soon. This occurs due to repeated use of data or instructions (e.g., in loops). It’s exploited in caching and page replacement to keep recently used items in fast memory

14. What is a page reference string?

Answer: In Lecture Notes Part 3

\*A page reference string is a sequence of page numbers accessed by a process during execution (e.g., 2, 6, 1, 5, 7). It’s used to simulate and analyze the performance of page replacement algorithms by tracking the order of page requests.

15. What does Working Set Window mean?

Answer: In Lecture Notes Part 3

\*The Working Set Window (denoted Δ) is the number of most recent page references considered to define a process’s working set. It’s a time window (e.g., Δ = 10 means the last 10 references) used to identify actively used pages, helping allocate sufficient memory to minimize page faults

16. What does Working Set WSi(tj) mean?

Answer: In Lecture Notes Part 3

\*WSi (tj )=Working Set=Set of actively referenced pages of process i at time t

- WSᵢ(tⱼ) is the working set of process i at time tⱼ, defined as the set of pages actively referenced by process i within the last Δ (Working Set Window) page references up to time tⱼ. It represents the pages currently in use, guiding memory allocation to prevent thrashing

17. What is Belady’s proposal to set the PAS size, given LAS size of a user program?

Answer: In Lecture Notes Part 3

\*Belady proposed that the Physical Address Space (PAS) size should be set to approximately half of the Logical Address Space (LAS) size of a user program. This is based on Belady’s Curve, which suggests that this allocation achieves a significant reduction in page faults while balancing memory utilization.

18. What is the goal of Working Set?

Answer: In Lecture Notes Part 3

\* Goal: Execute a program in much smaller PAS then predicted by Belady. Belady proposed PAS size should be half of the LAS size. Can we do better, for example, PAS size 5-10% LAS size

- The goal of the Working Set model is to identify and keep in memory the set of actively referenced pages (working set) of a process to minimize page faults. By ensuring these pages remain in PAS, it aims to execute programs efficiently with a smaller PAS than predicted by Belady (e.g., 5-10% of LAS), leveraging locality

19. In relation to virtual memory management, OS continuously monitors the memory demand of each process and attempts to allocate enough PAS pages to processes to match their memory demands. Assume D=total memory demand for pages by all processes and m=total number of physically available pages. Describe in detail all the OS actions to be taken for the two cases, D>m and D<m

Answer: In Lecture Notes Part 3

\*Case 1: D > m (Demand exceeds available pages): The OS detects thrashing risk due to insufficient memory.

Actions:

1. Identify processes to swap out: Select one or more processes based on criteria (e.g., priority, working set size).

2. Swap out processes: Write their pages to the swap area, freeing up PAS frames.

3. Reallocate frames: Assign freed frames to remaining processes to meet their working set needs.

4. Monitor: Continuously check D vs. m, adjusting multiprogramming as needed.

/\*This reduces the degree of multiprogramming to prevent thrashing.

\*Case 2: D < m (Demand less than available pages): Excess memory allows increased multiprogramming.

Actions:

1. Check for waiting processes: Identify swapped-out processes or new jobs ready to run.

2. Swap in or admit processes: Allocate available frames to these processes from the free pool.

3. Start/resume processes: Begin execution of newly admitted or resumed processes.

4. Monitor: Ensure D remains below m, adjusting as new demands arise.

/\*This increases CPU utilization by running more processes.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

20. Assume there exist a total of 2000K pages (frames) available in Physical Address Space (PAS) and 200K of these PAS pages (frames) are permanently allocated to OS kernel and its associated data structures. Page size is 1K bytes. There are 6 processes, p1, p2, p3, p4, p5 and p6 with virtual memory sizes 400, 600, 800, 1000, 1200 and 2000 Megabytes respectively. (i) How many pages(frames) will be allocated to p1, p2, p3, p4, p5 and p6, if equal page allocation is used? (ii) How many pages (frames), if proportional page allocation is used?

Answer: In Lecture Notes Part 3

Given:

Total PAS pages = 2000K = 2,000,000 pages (assuming K = 1000, common in context).

OS uses 200K = 200,000 pages.

Available for processes = 2,000,000 - 200,000 = 1,800,000 pages.

Page size = 1K bytes = 1024 bytes.

Virtual memory sizes: p1 = 400 MB, p2 = 600 MB, p3 = 800 MB, p4 = 1000 MB,p5 = 1200 MB, p6 = 2000 MB.

Number of processes = 6.

1. Equal Page Allocation: Each process gets an equal share: 1,800,000 / 6 = 300,000 pages.

Allocation: p1: 300,000 pages p2: 300,000 pages p3: 300,000 pages p4: 300,000 pages p5: 300,000 pages p6: 300,000 pages

1. Proportional Page Allocation:

Calculate total demand (S): Total VM = 400 + 600 + 800 + 1000 + 1200 + 2000 = 6000 MB. 1 MB = 1024 KB = 1024 pages (since page size = 1K bytes). S = 6000 × 1024 = 6,144,000 pages.

Allocation formula: aᵢ = (sᵢ / S) × m, where m = 1,800,000 pages.

Compute for each process:

p1: (400 / 6000) × 1,800,000 = (1/15) × 1,800,000 = 120,000 pages

p2: (600 / 6000) × 1,800,000 = (1/10) × 1,800,000 = 180,000 pages

p3: (800 / 6000) × 1,800,000 = (2/15) × 1,800,000 = 240,000 pages

p4: (1000 / 6000) × 1,800,000 = (1/6) × 1,800,000 = 300,000 pages

p5: (1200 / 6000) × 1,800,000 = (1/5) × 1,800,000 = 360,000 pages

p6: (2000 / 6000) × 1,800,000 = (1/3) × 1,800,000 = 600,000 pages

Verification: 120,000 + 180,000 + 240,000 + 300,000 + 360,000 + 600,000 = 1,800,000 pages.

**PART 4 I/O Management & Protection**

1. Given a hard disk drive that rotates at 3000 RPM, what is the average latency?

Answer: 3000 RPM means 3000 revolutions in 60 seconds, then 60/3000 seconds per revolution, which is full latency. Average latency is half the full latency, therefore Average latency is 60/6000 sec or 10 millisec.

2. Compute the average I/O time for a 7200 RPM disk disk drive with a 5ms average seek time, 1Gb/sec transfer rate with a 0.1ms controller overhead to transfer a 4KB block.

Answer: In Lecture Notes Part 4

Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead

Given:

Rotational speed = 7200 RPM.

Seek time = 5 ms.

Transfer rate = 1 Gb/sec = 10⁹ bits/sec = (10⁹ / 8) = 125,000,000 bytes/sec.

Controller overhead = 0.1 ms.

Block size = 4 KB = 4096 bytes.

Rotational Latency:

Time per revolution = 60 / 7200 = 0.008333 sec = 8.333 ms.

Average latency = 8.333 / 2 = 4.167 ms.

Transfer Time: Transfer time = 4096 / 125,000,000 = 0.000032768 sec = 0.032768 ms.

Total I/O Time: Seek time + Rotational latency + Transfer time + Controller overhead

= 5 ms + 4.167 ms + 0.032768 ms + 0.1 ms ≈ 9.3 ms (rounded).

3. What do we mean by the form factor of a disk drive?

Answer: In Lecture Notes Part 4

The form factor of a disk drive refers to its physical size and shape, specifically the diameter of its platters (e.g., 3.5-inch for desktops, 2.5-inch for laptops). It determines compatibility with hardware enclosures and mounting systems.

4. Modern Solid State Disks are also called nonvolatile memory or flash memory, but in reality they are not exactly nonvolatile, that is, due to semiconductor wear, they will not be able to store bits/bytes correctly. What does an operating system do about this problem?

Answer: In Lecture Notes Part 4

(SLAYT) -A built-in SSD Controller internally performs wear leveling, that is all NAND cells are read/written evenly.

\*Wear Leveling: Distributes writes evenly across cells to prevent overuse of any single cell.

\*Error Correction Codes (ECC): Detects and corrects bit errors to maintain data integrity.

\*TRIM Command: Informs the SSD which blocks are no longer in use, improving performance and longevity by managing free space.

5. What are the two main problems with MBR approach to partition disks?

Answer: In Lecture Notes Part 4

SLAYT-Problem: Max 4 partitions may not be sufficient for certain hard disk configurations.

SLAYT-Problem: MBR partition table uses 32 bit sector (each 512 bytes) numbers. This limits max partition size to 2 Terabytes (2 32 × 512 bytes).

\*Limited Partitions: Supports only up to four primary partitions.

\* Size Limitation: Maximum partition size is 2 TB due to 32-bit sector addressing.

6. Given a Disk Adress = (i,j,k) where i is cylinder number, j is track (surface, head) number and k is sector number and there are s sectors/track and t tracks/cylinder on a specific harddisk, show how we compute a logical block number b. Also Show your computation if there are m sectors per logical block.

Answer: In Lecture Notes Part 4

\*Single Sector to Logical Block Number (b):

**Formula: b = i × (t × s) + j × s + k**

Explanation: Each cylinder has t tracks, each track has s sectors.

Cylinder i starts at sector i × (t × s).

Within cylinder i, track j adds j × s sectors.

Sector k adds k to the total.

**With m Sectors per Logical Block: Logical block number = floor(b / m)**

Explanation: If each block contains m consecutive sectors, divide the sector number by m (integer division) to find the block number.

7. Given a Disk Adress = (i,j,k) where i is cylinder number, j is track (surface, head) number and k is sector number and there are s sectors/track and t tracks/cylinder on a specific harddisk, show how we compute the disk address (i,j,k) given a logical block number b. Also Show your computation if there are m sectors per logical block.

Answer: Partly in Lecture Notes Part 4 and also assigned orally as a homework during a class lecture.

**\*Single Sector (m = 1):**

Sector number = b

i = floor(b / (t × s))

Remainder = b % (t × s)

j = floor(remainder / s)

k = remainder % s

Explanation:

Total sectors per cylinder = t × s.

Cylinder i = b divided by sectors per cylinder.

Within cylinder, j = remainder divided by sectors per track.

k = final remainder.

\***With m Sectors per Logical Block**:

Starting sector = b × m

i = floor((b × m) / (t × s))

Remainder = (b × m) % (t × s)

j = floor(remainder / s)

k = remainder % s

Note: This gives the address of the block’s first sector; a block may span multiple (i, j, k) if m is large.

8. In disk drive scheduling, what is the goal to minimize?

Answer: In Lecture Notes Part 4. Disk seek time

\*The goal is to minimize average seek time, the time taken for the disk head to move between cylinders, improving overall disk access efficiency.

9. What is the main problem associated with Shortest Seek Time First disk scheduling?

Answer: In Lecture Notes Part 4. Starvation

\*The main problem is starvation, where requests far from the current head position may be indefinitely delayed if closer requests keep arriving.

10. Explain how the SCAN disk scheduling algorithm operates.

Answer: In Lecture Notes Part 4.

\*SCAN (Elevator Algorithm): The disk head moves in one direction (e.g., outward), servicing all requests in its path until it reaches the end (highest cylinder), then reverses direction and services requests on the return trip. It ensures all requests are eventually handled, reducing starvation compared to SSTF

-The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.

11. Explain how the C-SCAN disk scheduling algorithm operates.

Answer: In Lecture Notes Part 4.

\*C-SCAN (Circular SCAN): The head moves in one direction (e.g., outward), servicing requests until it reaches the end, then jumps back to the beginning (lowest cylinder) without servicing requests on the return, and repeats. It provides more uniform wait times by treating the disk as a circular list.

-Circular SCAN: Provides a more uniform wait time than SCAN

• The head moves from one end of the disk to the other, servicing requests as it goes

• When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip

• Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

12. What is the main difference between disk scheduling algorithms SCAN and C-SCAN?.

Answer: In Lecture Notes Part 4.

\*SCAN services requests in both directions (outward and inward), while C-SCAN services requests only in one direction and jumps back to the start without servicing on the return, offering fairer wait times.

-Circular SCAN: Provides a more uniform wait time than SCAN

13. Explain how the LOOK disk scheduling algorithm operates.

Answer: In Lecture Notes Part 4.

\*LOOK: Similar to SCAN, but the head only moves to the farthest request in the current direction, then reverses, rather than going to the disk’s end. This reduces unnecessary head movement when no requests exist beyond the last one.

14. Explain how the C-LOOK disk scheduling algorithm operates.

Answer: In Lecture Notes Part 4.

\*C-LOOK: Like C-SCAN, but the head moves in one direction to the last request, then jumps to the first request in the opposite direction without servicing on the return. It minimizes movement compared to C-SCAN by not going to the physical end.

15. What is the main difference between disk scheduling algorithms LOOK and C- LOOK?.

Answer: In Lecture Notes Part 4.

\*LOOK services requests in both directions up to the last request, while C-LOOK services only in one direction to the last request, then jumps to the first request, avoiding bidirectional servicing.

16. What is the main difference between disk scheduling algorithms C-SCAN and C- LOOK?.

Answer: In Lecture Notes Part 4.

\*C-SCAN goes to the end of the disk before jumping back, while C-LOOK only goes to the last request, reducing unnecessary head travel if no requests are at the end.

17. What is so special about UNIX/LINUX files, from the point of view of file types,

Answer: In Lecture Notes Part 4, in UNIX/LINUX a file is just a sequence of bytes.

\*In UNIX/Linux, a file is just a sequence of bytes with no imposed structure by the OS, unlike some systems with record-based files. The interpretation is left to applications. Additionally, UNIX supports diverse file types (regular files, directories, symbolic links, devices, sockets) uniformly within the file system.

18. What is the main difference between mandatory and advisory file locks?.

Answer: In Lecture Notes Part 4. In mandatory file lock, access is denied if a process attempts to access a locked file. In advisory file lock, access is also denied if a process attempts to access a locked file, but a process can also find the status of a file lock and decide what to do.

\*Mandatory Locks: Enforced by the OS; access to a locked file is denied to other processes, ensuring strict protection.

\*Advisory Locks: Cooperative; processes can check lock status and decide action, but the OS doesn’t enforce it, allowing access if a process ignores the lock.

19. Explain how sequential file access operates.

Answer: In Lecture Notes Part 4.

\*Sequential File Access: Data is read or written from start to end in order, one byte or record at a time. The file pointer advances sequentially, and random jumps require traversing prior data, suitable for linear processing (e.g., logs).

20. Explain how direct file access operates.

Answer: In Lecture Notes Part 4.

\*Direct File Access (Random Access): Allows reading or writing at any file position by seeking to a specific offset or record number without traversing prior data. It’s efficient for applications like databases needing quick access to specific data.

21. What are the three classes of users in UNIX/LINUX, from file protection point of view?

Answer: In Lecture Notes Part 4.

1. Owner (User): The file’s creator or assigned owner.

2. Group: Users in the file’s assigned group.

3. Others: All other system users.

22. How do we represent the read, write, execute rights in UNIX/LINUX? Explain.

Answer: In Lecture Notes Part 4.

\*Representation: Permissions are a 9-bit field in three sets of three bits:

Owner: Bits 1-3 (read, write, execute).

Group: Bits 4-6 (read, write, execute).

Others: Bits 7-9 (read, write, execute).

Values: 1 = permission granted, 0 = denied.

Octal Notation: Each set is a digit (0-7): Read = 4, Write = 2, Execute = 1 (e.g., 7 = rwx, 5 = r-x). Example: 755 = Owner: rwx (7), Group: r-x (5), Others: r-x (5).

Special bits (setuid, setgid, sticky) exist beyond these 9.

23. If a UNIX/LINUX file has access rights code 761, what does this mean? Explain in detail.

Answer: In Lecture Notes Part 4.

Code 761 in Octal:

Owner: 7 = 111 (read, write, execute).

Group: 6 = 110 (read, write).

Others: 1 = 001 (execute).

Explanation:

Owner: Full permissions to read, write, and execute.

Group: Can read and write but not execute.

Others: Can only execute, not read or write.

Detail: This is unusual—execute-only for others typically implies a script or binary where reading isn’t needed, but it’s rare without read permission for security or practical reasons.

24. What are the two main problems with contiguous file allocation.

Answer: In Lecture Notes Part 4. It is difficult to know the exact size of the file to be allocated, resulting in (i) if too much file space is allocated but file data small, then disk space is wasted, (ii) ) if too little file space is allocated but file data large, then it is very costly to have a larger new file that is contiguous.

\*1. Size Prediction Difficulty: Hard to know a file’s exact size in advance; overallocation wastes space, under-allocation requires costly relocation.

2. External Fragmentation: Free space fragments as files are created/deleted, making it difficult to find contiguous blocks for new files.

25. Explain UNIX/LINUX UFS file allocation technique.

Answer: In Lecture Notes Part 4. Slide 96

Uses a combined scheme with direct and indirect blocks in the inode:

Direct Blocks: 12 pointers to data blocks (e.g., 48 KB with 4 KB blocks).

Indirect Block: Points to a block of pointers (e.g., 1024 pointers = 4 MB).

Double Indirect: Points to a block of indirect block pointers (e.g., 1024 × 1024 × 4 KB = 4 GB).

Triple Indirect: Points to a block of double indirect pointers (e.g., 1024³ × 4 KB = 4 TB).

Operation: Small files use direct blocks for efficiency; large files use indirect levels for scalability, avoiding fragmentation but requiring more disk seeks for random access.