



Operating System — Question Bank

BTEUP CSE, 3rd Semester (Subject 3.1) · Questions with Answers

KAMALDNP kamaldnp.com

About this Question Bank — read this first

Who made it. This Question Bank is made by **KAMALDNP**, Varanasi — an IT training institute that teaches computer courses and summer training and publishes study material for diploma students. It was prepared by the institute's teachers from the official BTEUP syllabus, the companion textbook, and the board's own previous examination papers.

Why it was made. Students usually meet the board's questions for the first time inside the examination hall. The syllabus tells you what to study, but only the board's real papers tell you how it is asked — which topics return year after year, which numericals are favourites, and how much writing a 6-mark answer actually needs. This bank exists to remove that surprise: practice here should feel like a rehearsal of the real paper.

What is inside.

- **Unit-wise questions with answers** — objective (MCQ) questions with an answer key, short-answer questions with model answers, and long-answer questions with the points a full answer must contain, for all five units of the syllabus.

- **Fully worked numericals** — CPU scheduling, Banker's algorithm, memory fits, paging, page replacement and disk scheduling, solved step by step, including the solutions to the practice sets printed at the ends of the textbook's chapters.
- **Model papers (paper sets)** — complete practice papers on the board's pattern (60 marks, 3 hours), with answer hints, so you can rehearse under examination conditions.
- **Previous years' questions, marked with their year** — real questions from the board's past papers are included with the year they were asked (for example, "asked in 2024"), collected from the papers of 2020 onwards. Where the subject appeared in an earlier year under a different name or in a different semester, that is noted — the questions are the board's own, so repeated topics show themselves honestly.

Where and how to download. This Question Bank and its companion textbook, Operating System, are available on **kamaldnp.com**. To get them: open kamaldnp.com in your browser, go to the study-material / books section, find Operating System (BTEUP CSE 3rd Semester), and download the PDF — the textbook and the question bank are given as separate PDF files, free to save on your phone or computer and to print. If a link ever fails, write to **support@kamaldnp.com**.

Why you should read it. Because it is built from the board's own behaviour: the syllabus fixes the territory, the previous papers reveal the habits, and the model answers show the expected depth. A student who has honestly worked through this bank has already met most of the real paper's questions — or their close relatives — at least once.

What it is not. This bank is a companion, not a substitute. It does not replace the textbook (where every topic is explained from the beginning), your class notes, or your own practice on paper. Answers printed here are model answers — in the examination, write them in your own words

and diagrams. And if your syllabus or scheme differs from Subject 3.1 (CSE, 3rd Semester), confirm your own subject code first; question banks of other subjects are published separately.

How to use it. Finish one unit in the textbook, then attempt that unit here with the answers covered. Check, correct, and redo the numericals on paper until the method is automatic. In the last weeks before the examination, switch to the previous years' questions and the model papers, solving them timed, exactly as in the hall.

Published by KAMALDNP · kamaldnp.com · support@kamaldnp.com





Question Bank — Unit 1: Overview of Operating System

(BTEUP CSE 3rd Semester · Subject 3.1 · Unit weight in the paper: about 14%)

Section A — Objective Questions (MCQ)

Q1. An operating system is mainly:

- (a) an application program
- (b) system software between the user and the hardware
- (c) a part of the CPU
- (d) a programming language

Answer: (b)

Q2. Which of these is NOT a resource managed by the OS?

- (a) CPU
- (b) Memory
- (c) Disk
- (d) The data typed by the user in a document

Answer: (d)

Q3. In a batch system, jobs are:

- (a) executed with continuous user interaction
- (b) grouped and executed one after another without interaction
- (c) executed on many computers
- (d) executed with guaranteed deadlines

Answer: (b)

Q4. Multiprogramming mainly aims at:

- (a) better graphics
- (b) keeping the CPU busy by switching it to another job during I/O wait
- (c) running one job at a time
- (d) connecting computers in a network

Answer: (b)

Q5. Time-sharing is best described as:

- (a) one user at a time on one machine
- (b) many users sharing one machine through quick CPU switching
- (c) sharing files on the internet
- (d) a system with many CPUs

Answer: (b)

Q6. A system with several CPUs working truly in parallel is called:

- (a) batch
- (b) time-sharing
- (c) multiprocessing
- (d) embedded

Answer: (c)

Q7. A hard real-time system is one where:

- (a) responses are usually fast
- (b) missing a deadline is a failure
- (c) graphics are of high quality
- (d) many users work together

Answer: (b)

Q8. Which OS type fits a washing machine controller?

- (a) distributed
- (b) batch
- (c) real-time only
- (d) embedded

Answer: (d)

Q9. The two primary goals of an OS are:

- (a) speed and colour
- (b) convenience and efficiency
- (c) security and gaming
- (d) cost and size

Answer: (b)

Q10. Which generation first allowed many jobs to stay in memory together?

- (a) no OS
- (b) batch
- (c) multiprogramming
- (d) personal computers

Answer: (c)

Q11. A distributed OS makes several computers appear as:

- (a) separate machines
- (b) one single system
- (c) a printer network
- (d) a batch queue

Answer: (b)

Q12. Which of these is an operating system?

- (a) MS Word
- (b) Linux
- (c) Google Chrome
- (d) Tally

Answer: (b)

Q13. In a time-sharing system, each user/program receives:

- (a) the full machine for hours
- (b) small time slices in turn
- (c) a separate CPU
- (d) only printed output

Answer: (b)

Q14. Soft real-time systems are used in:

- (a) pacemakers
- (b) flight control
- (c) video streaming
- (d) missile guidance

Answer: (c)

Q15. The OS layer directly interacts with:

- (a) only the user
- (b) only applications
- (c) the hardware, on behalf of programs and users
- (d) the internet only

Answer: (c)

Section B — Short-Answer Questions (with model answers)

Q1. What is an operating system? An operating system is the system software that sits between the user/application programs and the computer hardware. It controls the whole machine, manages its resources (CPU, memory, disks, I/O devices) and provides the services and environment programs need to run. Examples: Windows, Linux, Android.

Q2. State the two goals of an operating system with one example each. Convenience — the user can run programs and save files without knowing the hardware's internals (example: double-clicking a file opens it). Efficiency — hardware should not sit idle (example: while one program waits for the printer, the CPU runs another program).

Q3. What is a batch operating system? Where is it suitable? Jobs of similar type are collected into a batch and run one after another without user interaction during execution. It suits repeated processing on large data, such as old payroll and billing work. Its limits are no interaction and slow turnaround.

Q4. How does multiprogramming reduce CPU idle time? Many jobs are kept in memory together. When the running job waits for I/O, the OS switches the CPU to another ready job instead of letting the CPU sit idle — so the CPU keeps working almost always.

Q5. Differentiate between multitasking and time-sharing (two points). Multitasking: one user runs several programs on one machine by fast CPU switching. Time-sharing: many users share one machine through terminals; it is the multi-user extension of multitasking, based on time slices and quick response for each user.

Q6. What is a real-time OS? Name its two kinds. An OS that guarantees a response within a fixed deadline; being late is failure. Hard real-time never misses a deadline (flight control). Soft real-time tolerates an occasional delay (video streaming).

Q7. What is an embedded OS? Give two examples of devices using one. A small, specialised OS built inside a device for one dedicated job. Examples: washing machine controller, ATM, smart TV, microwave oven.

Q8. Write the evolution of operating systems in four steps. (1) No OS — manual operation, machine idle between jobs. (2) Batch — automatic job sequencing. (3) Multiprogramming — CPU switched during I/O waits. (4) Time-sharing — interactive use by many users through time slices.

Section C — Long-Answer Questions (with model answers)

Q1. Define an operating system. Explain its position between user and hardware with a diagram description, and discuss its two goals in detail. Model answer (key points to develop): definition (system software; interface/manager layer) → layered picture: users → application programs → OS → hardware; requests flow down, results flow up → goal 1, convenience: hides hardware details, provides easy services (files, printing, running programs) → goal 2, efficiency: resource management so CPU/memory/devices rarely idle; multiprogramming as the classic example → conclusion: every OS function in later units serves one of these two goals.

Q2. Describe the main types of operating systems with one example of each. Model answer (key points): batch (grouped jobs, no interaction; payroll) → multiprogrammed (jobs in memory, CPU switching on I/O wait) → multitasking/time-sharing (time slices, interactive, many users; college server) → multiprocessing (several CPUs in parallel; servers) → real-time (deadline guarantee; hard — flight control, soft — streaming) → distributed (many machines as one; clusters) → embedded (inside a device; ATM) → a comparison line for each pair the examiner may join (interactive or not; single or multi CPU; deadline or not).

Q3. "A time-sharing system is a logical extension of multiprogramming." Justify this statement. Model answer (key points): multiprogramming keeps many jobs in memory and switches on I/O wait — but the switch happens only when a job waits, and users cannot interact → time-sharing switches on a timer (time slice) as well,

many times per second, whether or not a job waits → result: every user gets frequent small turns and feels continuous attention → same memory-resident multi-job base, extended switching rule, new property (interactivity) — hence a logical extension, not a different invention.

Q4. Compare batch, time-sharing and real-time systems on: interaction, response behaviour, CPU use, and typical application.

Model answer (in table form): Interaction — batch: none during execution; time-sharing: full; real-time: as designed, deadline-bound. Response — batch: hours; time-sharing: seconds or less; real-time: guaranteed fixed limit. CPU use — batch: good sequencing but idle at load/unload gaps; time-sharing: high via switching; real-time: as needed for deadlines, not maximum throughput. Applications — batch: payroll/billing; time-sharing: college server, desktops; real-time: flight control (hard), streaming (soft).





Question Bank — Unit 2: Process Management

(BTEUP CSE 3rd Semester · Subject 3.1 · Unit weight in the paper: about 25%)

Section A — Objective Questions (MCQ)

Q1. A process is:

- (a) a program stored on disk
- (b) a program in execution
- (c) a part of the kernel
- (d) a file

Answer: (b)

Q2. One program opened in five windows gives:

- (a) 5 programs, 1 process
- (b) 1 program, 5 processes
- (c) 5 programs, 5 processes
- (d) 1 program, 1 process

Answer: (b)

Q3. When I/O completes for a waiting process, it moves to:

- (a) running
- (b) terminated
- (c) ready
- (d) new

Answer: (c)

Q4. The PCB does NOT store:

- (a) process state
- (b) program counter
- (c) the program's source code
- (d) CPU registers

Answer: (c)

Q5. A context switch is:

- (a) free of cost
- (b) pure overhead
- (c) a type of system call for printing
- (d) performed by the user

Answer: (b)

Q6. The degree of multiprogramming is controlled by:

- (a) short-term scheduler
- (b) medium-term scheduler
- (c) long-term scheduler
- (d) dispatcher only

Answer: (c)

Q7. Which scheduling is non-preemptive by design?

- (a) Round Robin
- (b) SRTF
- (c) FCFS
- (d) preemptive priority

Answer: (c)

Q8. SJF gives the best possible:

- (a) response time
- (b) average waiting time
- (c) throughput always
- (d) fairness

Answer: (b)

Q9. The convoy effect belongs to:

- (a) SJF
- (b) RR
- (c) FCFS
- (d) SRTF

Answer: (c)

Q10. In Round Robin, if the quantum becomes very large, it behaves like:

- (a) SJF
- (b) FCFS
- (c) priority
- (d) LIFO

Answer: (b)

Q11. Starvation in priority scheduling is cured by:

- (a) larger quantum
- (b) aging
- (c) spooling
- (d) compaction

Answer: (b)

Q12. A race condition arises when:

- (a) two processes use different data
- (b) the result depends on the order of access to shared data
- (c) a process crashes
- (d) the CPU is slow

Answer: (b)

Q13. Which is NOT a requirement of a critical-section solution?

- (a) mutual exclusion
- (b) progress
- (c) maximum speed
- (d) bounded waiting

Answer: (c)

Q14. A binary semaphore used around a critical section is also called:

- (a) monitor
- (b) mutex
- (c) pipe
- (d) inode

Answer: (b)

Q15. Which condition is NOT necessary for deadlock?

- (a) mutual exclusion
- (b) hold and wait
- (c) preemption allowed
- (d) circular wait

Answer: (c) — the condition is no preemption.

Section B — Short-Answer Questions (with model answers)

Q1. Process vs program — two differences. A program is passive code stored on disk; a process is that code in execution with its own memory, state and resources. One program can run as many processes simultaneously; a process belongs to exactly one execution instance of a program.

Q2. List the five process states and the event that moves running → ready. New, ready, running, waiting, terminated. Running → ready happens on an interrupt — typically the time slice expiring — not on the process's own request.

Q3. What does the medium-term scheduler do? It swaps processes out of memory to disk when memory is overcrowded (and brings them back later), temporarily reducing the degree of multiprogramming to relieve pressure.

Q4. Define turnaround time and waiting time with formulas. $TAT = \text{Completion time} - \text{Arrival time}$ (total life in the system). $WT = TAT - \text{Burst time}$ (the part of that life spent waiting in the ready queue).

Q5. Why is perfect SJF impossible in a real OS? The OS cannot know a process's future burst time; it can only predict the next burst from the history of previous bursts (exponential averaging). SJF is therefore implemented approximately.

Q6. What are wait and signal on a semaphore? Wait (P): atomically decrement the semaphore; if the result is negative, block the caller. Signal (V): atomically increment; if processes are blocked, wake one. Their atomicity is what makes them safe guards.

Q7. State the four deadlock conditions in four short phrases. Mutual exclusion (one holder per resource); hold and wait (hold some, ask for more); no preemption (resources released only voluntarily); circular wait (a ring of waiting processes).

Q8. Safe, unsafe, deadlocked — one line each. Safe: an ordering exists letting every process reach its maximum and finish. Unsafe: no such ordering is guaranteed — deadlock is possible but has not happened. Deadlocked: a set of processes is already permanently blocked in a wait circle.

Section C — Long-Answer Questions (with model answers)

Q1. Draw the process state diagram and explain every transition. Also describe the contents of the PCB. Key points: five states; arrows — admission (new→ready), dispatch (ready→running), interrupt

(running→ready), I/O wait (running→waiting), I/O completion (waiting→ready, never direct to running), exit (running→terminated). PCB: PID, state, program counter, registers, scheduling info, memory info, accounting, I/O status — and its role in the context switch (save into old PCB, load from new PCB).

Q2. Compare FCFS, SJF, Round Robin and Priority scheduling on rule, preemption, strength and weakness. Which would you choose for (i) a batch payroll system (ii) an interactive lab — and why? Key points: the comparison table of Chapter 4 §4.6; choice — payroll: SJF/FCFS acceptable (no interactivity; SJF for average wait, FCFS for simplicity/predictability); interactive lab: Round Robin (response time and fairness), possibly with priority classes; starvation/aging addressed for priority; quantum trade-off stated.

Q3. Explain the producer-consumer problem and its three-semaphore solution; then show how the dining philosophers produce deadlock and how to prevent it. Key points: bounded buffer; rules (producer waits when full, consumer when empty, never both inside); semaphores mutex=1, empty=N, full=0 with the wait order (resource semaphore before mutex); philosophers: five sticks, circular wait formed when all take the left stick; prevention by breaking a condition — take both sticks atomically (break hold-and-wait), an ordering rule (break circular wait), or seat at most four (break circular wait by construction).

Q4. State the four deadlock-handling strategies. For prevention, give one method per condition with its cost. Why do general-purpose OSs mostly ignore deadlock? Key points: ignore/ostrich (rarity vs cost of prevention; recovery by restart); prevention table from Chapter 6 §6.3 (spooling vs mutual exclusion; request-all-at-once; preempt CPU/memory-like resources; resource ordering); avoidance (Banker's needs Max in advance — rarely available); detection & recovery (kill/

preempt victims). Conclusion: for general workloads the expected loss from rare deadlocks is smaller than the certain cost of prevention — an engineering trade-off, not an oversight.

Section D — Worked Numericals

N1. (Chapter 4 practice set, solved.) Processes A (AT 0, BT 6), B (AT 1, BT 3), C (AT 2, BT 2), D (AT 4, BT 4).

(a) FCFS: Order A, B, C, D. A 0–6; B 6–9; C 9–11; D 11–15. CT: 6, 9, 11, 15. TAT = CT – AT: 6, 8, 9, 11 → **average TAT 8.5**. WT = TAT – BT: 0, 5, 7, 7 → **average WT 4.75**.

(b) SJF (non-preemptive): A alone at 0 runs to 6. Ready: C (2), B (3), D (4) → C 6–8; B 8–11; D 11–15. CT: A 6, C 8, B 11, D 15. TAT: 6, 10, 6, 11 → **average 8.25**. WT: 0, 7, 4, 7 → **average 4.5**.

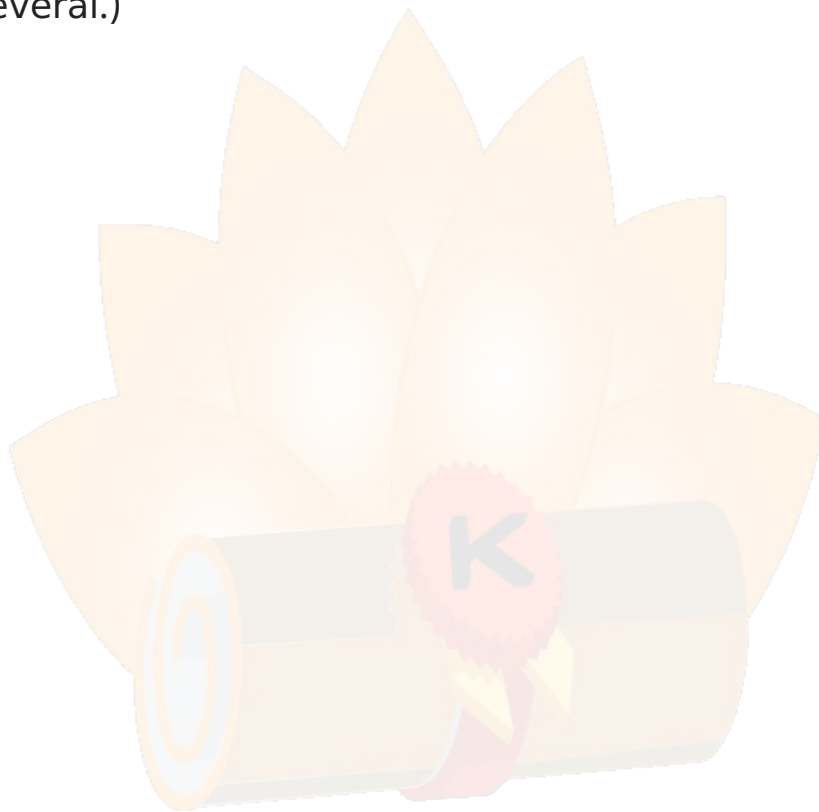
(c) SRTF: A starts; at 1, B (3) < A's remaining 5 → B runs; at 2, C (2) ties B's remaining 2 (not shorter — no switch); B finishes at 4. Ready: C (2), D (4), A (5) → C 4–6; D 6–10; A 10–15. CT: B 4, C 6, D 10, A 15. TAT: 15, 3, 4, 6 → **average 7**. WT: 9, 0, 2, 2 → **average 3.25** — the best of the three, as SRTF promises.

(d) Round Robin, quantum 2: A 0–2 → queue B, C, A → B 2–4 (D arrives at 4; queue C, A, D, B) → C 4–6 done → A 6–8 → D 8–10 → B 10–11 done → A 11–13 done → D 13–15 done. CT: A 13, B 11, C 6, D 15. TAT: 13, 10, 4, 11 → **average 9.5**. WT: 7, 7, 2, 7 → **average 5.75**. Note honestly: RR optimises response (every process ran by time 4), not average waiting — on this pattern it is the worst of the four on WT, which is exactly the trade-off Chapter 4 described.

N2. (Chapter 4 practice set, solved.) Chapter 4's Example 2 set — P1 (AT 0, BT 8), P2 (AT 1, BT 4), P3 (AT 2, BT 9), P4 (AT 3, BT 5) — under **non-preemptive priority** with priorities P1 = 2, P2 = 4, P3 = 1, P4 = 3 (smaller = higher). P1 is alone at 0 and (non-preemptive) completes at 8. Ready priorities: P3 (1), P4 (3), P2 (4). Order: P3 8–17; P4 17–22; P2 22–26. WT: P1 = 0; P3 = 17 – 2 – 9 = 6; P4 = 22 – 3 – 5 = 14; P2 = 26 – 1 – 4 = 21. **Average WT = (0**

$+ 6 + 14 + 21)/4 = 10.25$. Compare Example 4 in the chapter (priorities 3, 2, 4, 1 $\rightarrow$ average 8): identical work, different priority assignment, different suffering — priority values are policy.

N3. (Chapter 6 practice set, solved.) From Chapter 6's worked state (Available (3, 3, 2); P0 Need (7, 4, 3), Allocation (0, 1, 0)), **P0 requests (0, 2, 0)**. Check 1: $(0, 2, 0) \leq \text{Need } (7, 4, 3)$ ✓. Check 2: $\leq \text{Available } (3, 3, 2)$ ✓. Pretend: Available $\rightarrow (3, 1, 2)$; P0 Allocation $\rightarrow (0, 3, 0)$; Need $\rightarrow (7, 2, 3)$. Safety with Work (3, 1, 2): P3 Need (0, 1, 1) fits $\rightarrow$ Work (5, 2, 3); P1 (1, 2, 2) fits $\rightarrow (7, 2, 3)$; P0 (7, 2, 3) fits $\rightarrow (7, 5, 3)$; P4 (4, 3, 1) fits $\rightarrow (7, 5, 5)$; P2 (6, 0, 0) fits. All finish — **safe, request granted**. (And the chapter's other exercise: P3, P1, P4, P2, P0 is also a valid safe sequence — a safe state usually has several.)





Question Bank — Unit 3: Memory Management

(BTEUP CSE 3rd Semester · Subject 3.1 · Unit weight in the paper: about 25%)

Section A — Objective Questions (MCQ)

Q1. Address translation from logical to physical is done by:

- (a) the compiler
- (b) the MMU hardware, using OS-set values
- (c) the user
- (d) the disk controller

Answer: (b)

Q2. In fixed partitioning, the waste inside a partition is:

- (a) external fragmentation
- (b) internal fragmentation
- (c) compaction
- (d) thrashing

Answer: (b)

Q3. Best Fit chooses:

- (a) the first hole that fits
- (b) the largest hole
- (c) the smallest hole that fits
- (d) a random hole

Answer: (c)

Q4. Swapping is performed by:

- (a) long-term scheduler
- (b) short-term scheduler
- (c) medium-term scheduler
- (d) dispatcher

Answer: (c)

Q5. In paging, physical memory blocks are called:

- (a) pages
- (b) frames
- (c) segments
- (d) clusters

Answer: (b)

Q6. During paging translation, which part of the address never changes?

- (a) page number
- (b) frame number
- (c) the offset
- (d) the base

Answer: (c)

Q7. Paging eliminates:

- (a) internal fragmentation
- (b) external fragmentation
- (c) page faults
- (d) the page table

Answer: (b)

Q8. A segment table entry stores:

- (a) frame number only
- (b) base and limit
- (c) page number
- (d) priority

Answer: (b)

Q9. A page fault means:

- (a) the program has a bug
- (b) the needed page is not in memory and must be fetched
- (c) the disk failed
- (d) the process is terminated always

Answer: (b)

Q10. Optimal page replacement evicts the page:

- (a) loaded first
- (b) least recently used
- (c) whose next use is farthest in future
- (d) with the smallest number

Answer: (c)

Q11. Belady's anomaly can occur in:

- (a) LRU
- (b) Optimal
- (c) FIFO
- (d) all algorithms equally

Answer: (c)

Q12. Virtual memory is based on:

- (a) keeping all pages in RAM
- (b) demand paging — loading pages only when faulted on
- (c) tape storage
- (d) bigger disks only

Answer: (b)

Q13. Thrashing is caused by:

- (a) too few processes
- (b) combined working sets exceeding available memory
- (c) a fast disk
- (d) large page size always

Answer: (b)

Q14. Coalescing means:

- (a) splitting a hole
- (b) merging adjacent free blocks
- (c) compacting the disk
- (d) swapping out

Answer: (b)

Q15. The TLB is:

- (a) a disk
- (b) a fast cache of recent page-table translations
- (c) a type of segment
- (d) the swap file

Answer: (b)

Section B — Short-Answer Questions (with model answers)

Q1. Logical vs physical address, with the base/limit example. A

logical address is the address as the program sees it (its own space from 0); a physical address is the real location in RAM. The MMU adds the process's base to each logical address after checking it against the limit — base 12000, logical 2500 → physical 14500; logical 4100 with limit 4000 is a protection fault.

Q2. Why does variable partitioning create holes while fixed

partitioning does not? Variable partitions are cut to exact sizes and freed in arbitrary order, so free space emerges as scattered blocks between live processes. Fixed partitions are pre-cut and freed whole — the waste there stays inside partitions (internal), not between them.

Q3. What is a page table? What does one entry answer?

One table per process mapping page number → frame number (plus a valid/invalid mark). An entry answers: "if this page is in memory, in which frame; if not, a fault must fetch it."

Q4. Segmentation vs paging — division basis and fragmentation, two lines.

Paging divides into fixed sizes blindly (no external fragmentation, small internal waste). Segmentation divides by logic into variable sizes (natural meaning and protection, but external fragmentation returns).

Q5. The six steps of a page fault in brief.

Trap to OS → legality check → find/free a frame (evict a victim, writing it back if dirty) → read the page in → update the page table → restart the faulting instruction.

Q6. Why is Optimal kept in textbooks if it can never be built?

It is the provable minimum-fault benchmark: running it on real recorded reference strings measures how close a practical algorithm (LRU) comes to perfection.

Q7. What is locality, and why does virtual memory depend on it?

In any short window a program touches only a small stable set of pages (loops, hot code, nearby data). Demand paging is cheap only if faults stay rare — which locality guarantees for well-behaved programs.

Q8. Compaction — what, when, and its cost.

Sliding all processes together to merge scattered holes into one block so a large request can fit. Used when total free memory suffices but no hole does; costs copying time and a system pause, and needs all addresses re-based.

Section C — Long-Answer Questions (with model answers)

Q1. Explain the three placement strategies. On holes 100/500/200/300/600 with requests 212/417/112/426, show that only one strategy places all four, and draw the general lesson. Key points: definitions (first adequate; smallest adequate; largest); the full worked runs from Chapter 7 §7.3 (First Fit fails at 426 amid 959 KB free; Worst Fit fails identically; Best Fit preserves 600 for 426 and places all); lesson — placement is a long game about tomorrow's requests; in practice First Fit $\approx$ Best Fit overall, Worst Fit weakest, and Best Fit's slivers noted; connect to external fragmentation and compaction.

Q2. Describe address translation in paging with a diagram description and a worked example. Why is there no external fragmentation, and what does the last page waste? Key points: address = $p + d$; page table $p \rightarrow f$; physical = f with d unchanged (Figure 8.1 description); worked example (page size 100, logical 753 $\rightarrow$ physical 453); frames are interchangeable so any free frame serves any page — holes cannot form; the final page's unused tail is internal fragmentation, at most one page per process; page-table size and TLB mentioned.

Q3. Compare internal and external fragmentation: definitions, where each occurs (name the schemes), and the remedies. Why did the industry settle on paging? Key points: internal = inside an allocated block (fixed partitions, last page); external = scattered free holes (variable partitions, segmentation) with the 426-in-959 KB illustration; remedies — internal: smaller units/blocks trade-off; external: coalescing (partial), compaction (costly), or removing contiguity itself; paging removes the contiguity requirement, hence no external fragmentation — the decisive advantage, at the price of page tables and small internal waste.

Q4. What is virtual memory? Explain demand paging, its benefits, the locality assumption, and thrashing with its two remedies. Key points: per-process virtual space larger than RAM, pages on disk fetched on faults; benefits (bigger programs than RAM, more processes resident, less load I/O); locality as the load-bearing assumption; thrashing mechanism — combined working sets > RAM → fault storms → CPU collapse → scheduler admits more (fatal); remedies: guarantee working-set frames or lower the degree of multiprogramming; the CPU-utilisation curve described (rise, peak, cliff).

Section D — Worked Numericals

N1. (Chapter 7 practice set, solved.) Holes in order: 200, 600, 250, 450, 100. Requests: 380, 120, 290, 500.

- **First Fit:** 380 → 600 (first that fits), leftover 220. 120 → 200, leftover 80. 290 → 450, leftover 160. 500 → holes are 80, 220, 250, 160, 100 — **cannot be placed.**
- **Best Fit:** 380 → 450, leftover 70. 120 → 200, leftover 80. 290 → 600, leftover 310. 500 → largest hole is 310 — **cannot be placed.**
- **Worst Fit:** 380 → 600, leftover 220. 120 → 450, leftover 330. 290 → 330, leftover 40. 500 → holes 200, 220, 250, 40, 100 — **cannot be placed.**

Lesson worth marks: all three fail the 500 request although 810 KB stands free in total — the purest demonstration that external fragmentation is about the pieces, not the amount; only compaction (or paging) rescues this memory.

N2. (Chapter 8 practice set, solved.) (a) Page size 200, page table page 5 → frame 11, translate logical 1130: $p = 1130 \div 200 = 5$, $d = 1130 - 5 \times 200 = 130$; physical = $11 \times 200 + 130 = 2330$.

(b) Segmentation check from the same chapter: segment 2, base 5000, limit 800, offset 100 → $100 < 800 \checkmark$ → physical = **5100** (offset 900 would fault — the limit is the protection).

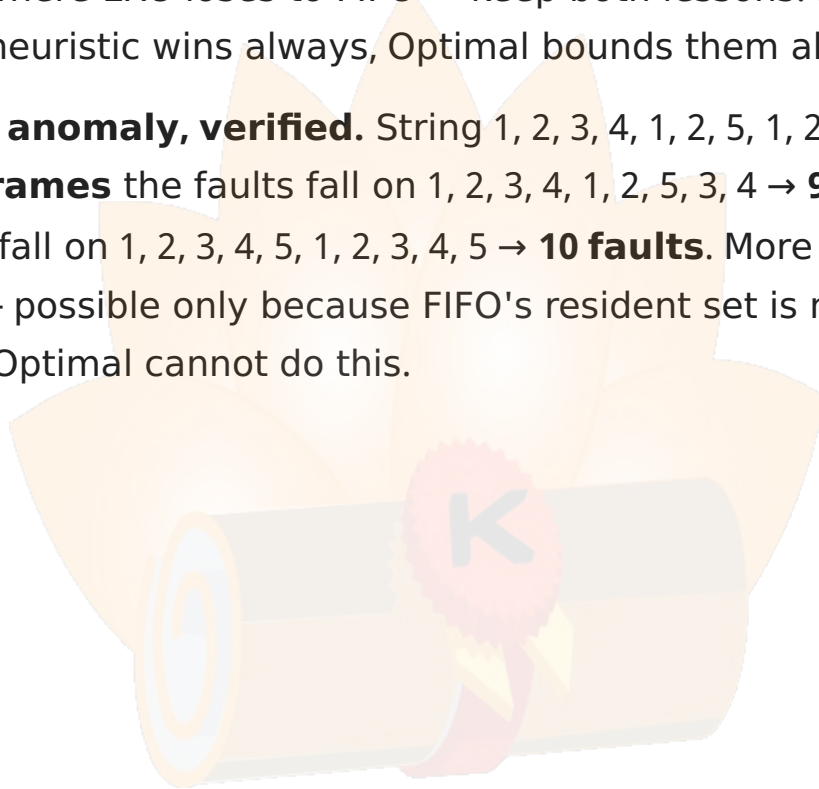
(c) A 32-bit system with 8 KB pages: offset bits = $\log_2(8192) = 13$; page-number bits = $32 - 13 = 19 \rightarrow$ a page table may need $2^{19} = 524,288$ entries.

N3. (Chapter 9 practice set, solved.) Reference string 1, 2, 3, 2, 1, 4, 3 with 3 frames.

- **FIFO:** 1 F, 2 F, 3 F, 2 hit, 1 hit, 4 F (evicts 1), 3 hit $\rightarrow$ **4 faults**.
- **LRU:** 1 F, 2 F, 3 F, 2 hit, 1 hit, 4 F (evicts 3, the least recently used), 3 F (evicts 2) $\rightarrow$ **5 faults**.
- **Optimal:** 1 F, 2 F, 3 F, 2 hit, 1 hit, 4 F (evicts a page never used again — 1 or 2), 3 hit $\rightarrow$ **4 faults**.

A string where LRU loses to FIFO — keep both lessons: LRU usually wins, no heuristic wins always, Optimal bounds them all.

N4. Belady's anomaly, verified. String 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5 under FIFO: with **3 frames** the faults fall on 1, 2, 3, 4, 1, 2, 5, 3, 4 $\rightarrow$ **9 faults**; with **4 frames** they fall on 1, 2, 3, 4, 5, 1, 2, 3, 4, 5 $\rightarrow$ **10 faults**. More memory, more faults — possible only because FIFO's resident set is not a stack set; LRU and Optimal cannot do this.





Question Bank — Unit 4: I/O System

(BTEUP CSE 3rd Semester · Subject 3.1 · Unit weight in the paper: about 18%)

Section A — Objective Questions (MCQ)

Q1. Which is a property of mass storage that RAM lacks?

- (a) speed
- (b) non-volatility
- (c) byte addressing by CPU
- (d) small size

Answer: (b)

Q2. The smallest physical storage unit on a disk platter is the:

- (a) track
- (b) cylinder
- (c) sector
- (d) block

Answer: (c)

Q3. A cylinder is:

- (a) one track
- (b) the tracks at the same arm position on all surfaces
- (c) a sector group
- (d) the disk's cache

Answer: (b)

Q4. Average rotational latency at 6000 RPM is:

- (a) 10 ms
- (b) 2.5 ms
- (c) 5 ms
- (d) 20 ms

Answer: (c) — one rotation takes 10 ms; the average wait is half a rotation.

Q5. NAS exports ___ over the network; SAN exports ___.

- (a) blocks; files
- (b) files; blocks
- (c) sectors; files
- (d) files; sectors

Answer: (b)

Q6. Disk scheduling mainly reduces:

- (a) transfer time
- (b) rotational latency
- (c) seek time / total head movement
- (d) disk size

Answer: (c)

Q7. Which disk algorithm can starve far-out requests?

- (a) FCFS
- (b) SSTF
- (c) SCAN
- (d) C-SCAN

Answer: (b)

Q8. C-SCAN differs from SCAN in that it:

- (a) serves both directions
- (b) serves in one direction and returns to the start
- (c) uses no queue
- (d) is only for SSDs

Answer: (b)

Q9. Swap space is managed:

- (a) as ordinary files for sharing
- (b) by the OS directly, bypassing the file system
- (c) by the user
- (d) on tape

Answer: (b)

Q10. RAID 0 provides:

- (a) mirroring
- (b) parity
- (c) striping only, no redundancy
- (d) two-disk failure safety

Answer: (c)

Q11. A 5-disk RAID 5 array tolerates how many simultaneous disk failures?

- (a) 0
- (b) 1
- (c) 2
- (d) 5

Answer: (b)

Q12. Usable space in an n-disk RAID 6 array is:

- (a) n disks' worth
- (b) $(n - 1)$
- (c) $(n - 2)$
- (d) $n/2$

Answer: (c)

Q13. Parity in RAID 5 is:

- (a) a full copy of every disk
- (b) stored distributed across disks, computed by XOR
- (c) kept on tape
- (d) the boot block

Answer: (b)

Q14. Logical formatting of a partition writes:

- (a) sectors
- (b) a file system
- (c) the BIOS
- (d) parity

Answer: (b)

Q15. Disk scheduling matters least for:

- (a) HDD under load
- (b) HDD idle
- (c) SSD/NVMe
- (d) a busy file server with HDDs

Answer: (c) — no arm, no seek.

Section B — Short-Answer Questions (with model answers)

Q1. Sector, track, cylinder — one line each. A sector is the smallest physical unit of a track (traditionally 512 B). A track is one concentric ring of sectors on a surface. A cylinder is the set of tracks at one arm position across all surfaces — all reachable without moving the arm.

Q2. Why is rotational latency quoted as an average? When a request arrives, the wanted sector can be anywhere in its rotation — just passing the head (near 0 wait) or just missed (a full rotation). The expected value over random positions is half a rotation.

Q3. Host-attached storage vs NAS, two points. Host-attached: disks wired to one computer's bus (SATA/NVMe), seen only by it. NAS: an appliance on the LAN serving files to many clients, who never see raw blocks.

Q4. FCFS vs SSTF for disks, two points. FCFS serves in arrival order — fair, but the arm zigzags (640 in the chapter's queue). SSTF serves the nearest pending request — much less travel (236 on the same queue) but extremes can starve under a steady middle load.

Q5. What is the elevator idea in SCAN? Serve everything in the direction of travel, reverse only at the end (or at the last request, in LOOK) — no request waits more than one full sweep, so travel savings do not buy starvation.

Q6. Swap partition vs swap file — one line each. Partition: a raw dedicated disk region — fastest, but fixed in size. File: a file inside a normal file system — easy to create/resize, slightly slower.

Q7. Why is RAID 1's efficiency exactly 50%? Every data block is written twice, once on each of a mirrored pair; of $2n$ disks' capacity, n disks' worth is copies, so half the raw space is usable.

Q8. "RAID is not backup" — explain in two lines. RAID protects against disk hardware failure only. Deletion, corruption, ransomware, theft and fire strike all copies at once — a backup is a separate copy, ideally elsewhere and offline.

Section C — Long-Answer Questions (with model answers)

Q1. Describe the structure of a hard disk with a labelled description (platters, heads, tracks, sectors, cylinders) and compute the capacity of a disk with 8 surfaces, 1024 tracks per surface, 64 sectors per track and 512 bytes per sector. Key points: anatomy (coated platters, one arm with heads moving together, concentric tracks, sectors as the smallest unit, cylinder = same-position tracks); capacity = $8 \times 1024 \times 64 \times 512 = 268,435,456 \text{ B} = 256 \text{ MB}$; note modern disks hide this geometry behind logical block numbering.

Q2. Define the three components of disk access time. Which does scheduling attack, and why can it not attack the other two? Key points: seek (arm to track — depends on request order, hence scheduling); rotational latency (platter spin — set by RPM and chance position, not order); transfer (data passing the head — set by size and spin speed). Conclusion: the scheduling lever works only on seek, which is luckily the dominant and most variable term.

Q3. Compare FCFS, SSTF, SCAN and C-SCAN on the chapter's queue (requests 98, 183, 37, 122, 14, 124, 65, 67; head 53). Give orders and totals, and discuss starvation. Key points: FCFS 640 (arrival order, no starvation, zigzag); SSTF 236 (greedy nearest, starvation of extremes with a demonstrating request pattern); SCAN 331 (sweep to end 199, no starvation); C-SCAN 382 counting the return jump (uniform waiting; convention stated); summary table and the fairness-vs-travel trade-off.

Q4. Explain RAID levels 0, 1, 5, 6 and 10 with technique, minimum disks, space efficiency and failure tolerance. A small business protects its accounts server: choose between RAID 5 and RAID 10 and defend the choice. Key points: the Chapter 11 table; parity and XOR rebuild described; choice defended either way with reasons (RAID 5: more usable space, one-disk tolerance, slower rebuilds and write penalty;

RAID 10: half space gone, faster rebuilds and writes, tolerates one per mirror pair) — a verdict tied to the business's data value and downtime cost is what earns marks, not the letter of the level.

Section D — Worked Numericals

N1. (Chapter 11 practice set, solved.) Queue: 55, 120, 18, 88, 160, 42; head at 40; disk 0–199.

- **FCFS:** 40 → 55 (15) → 120 (65) → 18 (102) → 88 (70) → 160 (72) → 42 (118). **Total = 442.**
- **SSTF:** from 40, nearest is 42 (2) → 55 (13) → 88 (33) → 120 (32) → 160 (40) → 18 (142). **Total = 262.**
- **SCAN (moving up, to the end):** 42, 55, 88, 120, 160, then end 199, back to 18. Movement 2 + 13 + 33 + 32 + 40 + 39 + 181 = **340.**
- **C-LOOK (moving up):** order 42, 55, 88, 120, 160, then jump straight to 18: 2 + 13 + 33 + 32 + 40 + 142 = **262** — here tying SSTF's total without its starvation risk; totals alone never tell the whole story.

N2. (Chapter 10 practice set, solved.) Disk: 10 surfaces, 4096 tracks per surface, 256 sectors per track, 512 bytes per sector. Sectors per surface = $4096 \times 256 = 1,048,576$; bytes per surface = $1,048,576 \times 512 = 512 \text{ MiB}$; total = $10 \times 512 \text{ MiB} = 5120 \text{ MiB} = \mathbf{5 \text{ GiB}}$.

N3. RAID arithmetic. Six disks of 3 TB each:

- **RAID 5:** usable = $(6 - 1) \times 3 = \mathbf{15 \text{ TB}}$; tolerates **1** failure.
- **RAID 6:** usable = $(6 - 2) \times 3 = \mathbf{12 \text{ TB}}$; tolerates **2** failures.
- **RAID 10:** usable = $6 \times 3 / 2 = \mathbf{9 \text{ TB}}$; tolerates **1 failure per mirrored pair** (up to 3 if luck spreads them, guaranteed only 1).
- **RAID 0:** usable = **18 TB**; tolerates **0** — and loses the whole array on the first failure.



Question Bank — Unit 5: File Management

(BTEUP CSE 3rd Semester · Subject 3.1 · Unit weight in the paper: about 18%)

Section A — Objective Questions (MCQ)

Q1. The OS identifies a file internally by:

- (a) its name
- (b) its unique identifier/inode number
- (c) its extension
- (d) its icon

Answer: (b)

Q2. Which is NOT a standard file operation?

- (a) create
- (b) reposition
- (c) compile
- (d) truncate

Answer: (c)

Q3. Tape storage supports which access method?

- (a) direct
- (b) indexed
- (c) sequential only
- (d) hashed

Answer: (c)

Q4. A database fetching record 5000 directly uses:

- (a) sequential access
- (b) direct access
- (c) tape access
- (d) append only

Answer: (b)

Q5. In a two-level directory, name collisions are avoided:

- (a) across all users
- (b) only between different users' directories
- (c) never
- (d) by longer names

Answer: (b)

Q6. The path /home/student/notes.txt is:

- (a) relative
- (b) absolute
- (c) a link
- (d) a mount point

Answer: (b)

Q7. Mounting attaches a file system:

- (a) to the CPU
- (b) at a mount point in the existing directory tree
- (c) to the network only
- (d) inside a file

Answer: (b)

Q8. In UNIX permissions `-rwxr-x---`, the group class may:

- (a) read and write
- (b) read and execute
- (c) only execute
- (d) do nothing

Answer: (b)

Q9. An ACL gives, compared to rwx classes:

- (a) less precision
- (b) per-user precision at the cost of longer lists
- (c) no protection
- (d) faster checks always

Answer: (b)

Q10. In UNIX the FCB is known as the:

- (a) inode
- (b) FAT
- (c) PCB
- (d) TLB

Answer: (a)

Q11. Which allocation method stores start block + length only?

- (a) linked
- (b) indexed
- (c) contiguous
- (d) hashed

Answer: (c)

Q12. The worst weakness of linked allocation is:

- (a) external fragmentation
- (b) very poor direct access
- (c) file growth
- (d) no sequential access

Answer: (b)

Q13. FAT improves linked allocation by:

- (a) removing pointers
- (b) collecting all pointers into one table that can be cached
- (c) making blocks contiguous
- (d) adding parity

Answer: (b)

Q14. A bit vector for free-space management uses:

- (a) one byte per block
- (b) one bit per block
- (c) one block per file
- (d) parity bits

Answer: (b)

Q15. The maximum file size on FAT32 is:

- (a) 2 GB
- (b) 4 GB
- (c) 16 GB
- (d) unlimited

Answer: (b)

Section B — Short-Answer Questions (with model answers)

Q1. Name vs identifier — why does the OS keep both? The name serves humans (changeable, per-directory, not globally unique). The identifier (inode number) is the OS's own stable key to the file's record, used by every internal structure; names resolve to identifiers through directories.

Q2. What happens at `open()` and at `close()`? Open loads the file's FCB into the system open-file table, creates a per-process entry with its own pointer at the start, and returns a handle. Close removes the per-process entry; when the last closer leaves, the FCB copy returns to disk, with any buffered writes flushed.

Q3. Sequential access — the pointer's behaviour and two natural workloads. The pointer starts at the beginning and advances automatically with every read/write; there is no jumping. Natural for CCTV footage recording and reading a log file top to bottom.

Q4. Relative path — define using .. with an example. A path interpreted from the current working directory instead of the root. From /home/student, the file /home/teacher/marks/cs.xlsx is ../teacher/marks/cs.xlsx — .. climbs to /home first.

Q5. Session semantics in one short paragraph. Under the model used by most network file systems, changes a user makes to an open file become visible to others when the file is closed — the session's work appears as a whole at the end, not byte by byte during editing.

Q6. Directory permission subtlety: readable file, unreachable file. A file with read permission inside a directory that lacks execute (search) permission for you cannot be opened — the OS cannot traverse the path to reach it. Directory rights guard the road, file rights guard the destination.

Q7. Grouping (free-space) in two lines. The first free block stores the addresses of a group of free blocks; the last address in the group points to the next such index block. Long free lists are thus fetched a chunk at a time, avoiding the linked list's block-by-block crawl.

Q8. Journaling — what problem does it solve? A crash mid-update can leave a file system half-written and inconsistent. A journaling system writes the intended changes to a log first; after a crash it replays or discards the log entry, returning the volume to a consistent state quickly.

Section C — Long-Answer Questions (with model answers)

Q1. List the file attributes and operations. Then compare sequential and direct access with one workload each, explaining why tape decides the question for one of them. Key points: attributes (name, identifier, type, location, size, protection, timestamps, owner); operations (create, write, read, reposition, delete, truncate, open/close); sequential (pointer order, tape/logs/media) vs direct (block-number positioning, databases on disk) with the position formula (relative block $\times$ record size); tape supports sequential only — a database on tape is therefore impossible in practice, however cheap the medium.

Q2. Trace directory structures from single-level to the acyclic graph: structure, the problem each step solved, and the new problem each created. Key points: single-level (all files together — collisions and clutter); two-level (per-user spaces — collisions solved, no internal organisation or sharing); tree (unlimited nesting, absolute/relative paths, working directory — deletion policy questions); acyclic graph (links and shared files — the same file under two names; deletion needs reference counting; cycles still banned to keep searches finite).

Q3. Compare contiguous, linked and indexed allocation on: record contents, sequential access, direct access, external fragmentation, growth, and extra cost. Choose for (i) a CCTV video archive (ii) a student-record database, defending both. Key points: Chapter 13's comparison table in prose; FAT's repair of linked allocation explained; choices — CCTV: contiguous (written once, watched sequentially, growth planned poorly is its one risk — or linked for painless growth; either defended beats an unargued pick); database: indexed (direct access is the workload).

Q4. Explain protection: the six access types, ACL vs the owner/group/others scheme (precision vs compactness), and decode `-rwxr-x---` completely. Why must directories be protected too? Key

points: read/write/execute/append/delete/list; ACL (per-user lists, precise, bulky) vs rwx (9 bits, three classes, cannot name one extra user); decode: owner rwx (all), group r-x (read+execute), others --- (nothing), leading '-' marks an ordinary file; directories: execute/search permission gates the path itself (B6), and list permission controls seeing names at all.

Section D — Worked Numericals and Exercises

N1. (Chapter 13 practice set, solved.) Free blocks on a disk: {2, 3, 4, 9, 15, 16, 17, 30}.

- **Linked list:** the free-list head points to 2, whose tail stores 3, then 4, then 9, then 15, 16, 17, and 30 ends the chain (null). Eight pointers travelled to enumerate eight blocks — the scheme's known slowness.
- **Counting:** runs described as (first, count) pairs: (2, 3), (9, 1), (15, 3), (30, 1) — four small entries describe the same space exactly, because free space arrives in runs.

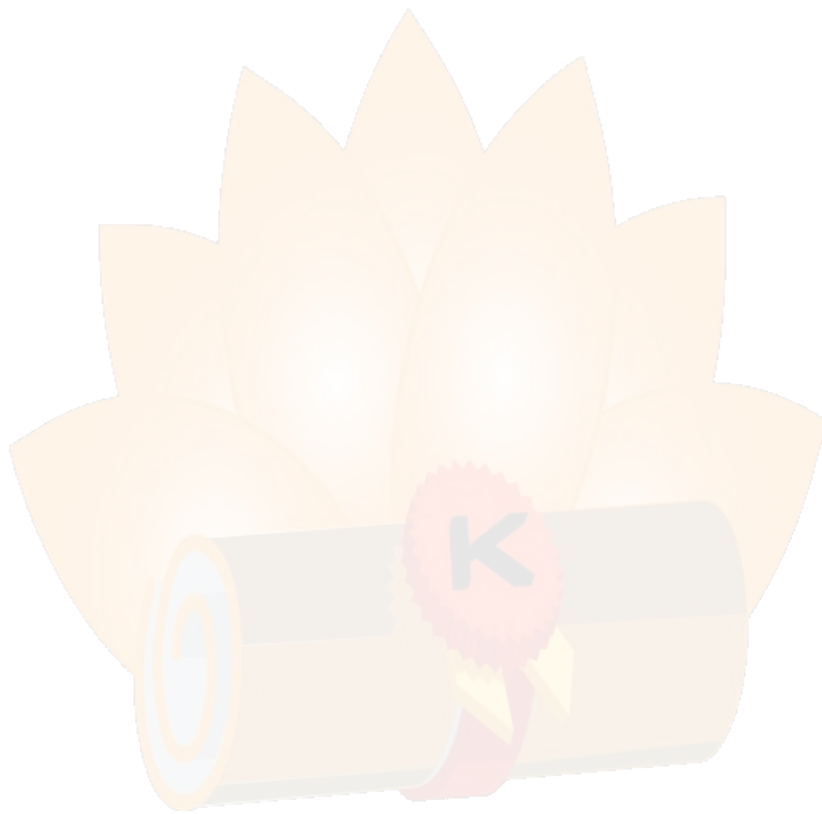
N2. Bitmap size (worked). A 2 TB disk (2^{41} bytes) with 4 KB blocks (2^{12} bytes): number of blocks = $2^{41} \div 2^{12} = 2^{29}$. One bit each $\rightarrow 2^{29}$ bits = 2^{26} bytes = **64 MiB** of bitmap — fixed by geometry alone, unchanged whether the disk is empty or nearly full.

N3. Paths (Chapter 12 practice set, solved). Absolute path /home/teacher/marks/cs.xlsx; working directory /home/student. Relative path: **../teacher/marks/cs.xlsx** (climb from student to home, descend into teacher/marks). And the reverse skill: from /home/student, the relative path **notes/os.txt** names the absolute **/home/student/notes/os.txt**.

N4. Permissions decisions. (a) **-rw-r-----**: owner reads+writes; group reads only; others nothing.

(b) A teacher wants: she can read-write her marks file, her department group can read it, nobody else anything, and the file must not be executed by anyone. Required string: **-rw-r-----** — exactly (a).

(c) A shared script that the owner edits and everyone (owner, group, others) runs but nobody else edits: owner needs rwx (7), group r-x (5), others r-x (5) → **-rwxr-xr-x**.





Model Papers — Operating System (Subject 3.1)

(Practice papers on the BTEUP pattern for CSE 3rd Semester: 60 marks, 3 hours. Each paper has 5 questions, one per unit; every question has three parts (a), (b), (c) of 6 marks each — attempt any TWO parts of each question. Answer hints follow each paper.)

Model Paper 1

Time: 3 Hours • Maximum Marks: 60 Attempt any two parts of each question. All parts of a question carry 6 marks.

Q1. (Unit 1 — Overview)

- (a) Define an operating system. Explain its two goals with one example of each.
- (b) Describe batch, time-sharing and real-time operating systems with one application of each.
- (c) What is a system call? List its five types with one example of each.

Q2. (Unit 2 — Process Management)

- (a) Draw the process state diagram and explain every transition.
- (b) Processes arrive as P1 (AT 0, BT 7), P2 (AT 2, BT 4), P3 (AT 4, BT 1), P4 (AT 5, BT 4). Compute average waiting time under FCFS and under SRTF, showing both Gantt charts.

(c) State the four necessary conditions for deadlock. For any two, give a prevention method and its cost.

Q3. (Unit 3 — Memory Management)

(a) Free holes in order are 120, 480, 260, 340, 200 KB; requests are 300, 140, 420 in that order. Show placements (or failure) under First Fit and Best Fit.

(b) With page size 256 and the page table entry page 3 → frame 6, translate logical address 1000 step by step. Also state how many pages a 5000-byte process needs and its internal fragmentation.

(c) Reference string 2, 3, 2, 1, 5, 2, 4, 5, 3, 2 with 3 frames: count page faults under LRU and under Optimal.

Q4. (Unit 4 — I/O System)

(a) A disk request queue is 86, 147, 91, 177, 48, 150, 102, 175; the head is at 100 and cylinders run 0–199. Compute total head movement under FCFS and under SCAN (head moving towards larger numbers, going to the end).

(b) Explain RAID levels 0, 1 and 5 with technique, minimum disks and failure tolerance. For four 1 TB disks, give usable space under each of the three.

(c) What is swap space? Compare a swap partition and a swap file, and explain why the OS manages swap outside the file system.

Q5. (Unit 5 — File Management)

(a) List the attributes of a file. Explain sequential and direct access methods with one suitable application each.

(b) Compare contiguous, linked and indexed file allocation on any four points, and name the repair that FAT applies to linked allocation.

(c) Decode the UNIX permission string `-rwxr-x---` completely. Then state the string for: owner read-write, group read, others nothing.

Answer Hints — Model Paper 1

Q1. (a) Ch 1 §1.1 definition; convenience (hiding hardware — double-click opens a file) and efficiency (CPU works during another job's I/O wait).

(b) Ch 1 §1.3 rows: batch/payroll; time-sharing/college server; real-time hard-flight control, soft-streaming.

(c) Ch 2 §2.4: user-mode → kernel-mode doorway; process control (fork), file (read), device (ioctl), information (getpid), communication (pipe).

Q2. (a) Ch 3 §3.2, Figure 3.1, all five arrows including waiting→ready on I/O completion.

(b) FCFS order P1, P2, P3, P4: CT 7, 11, 12, 16; WT: 0, 5, 7, 7 → **avg 4.75**. SRTF: P1 runs 0–2; P2 (4) vs P1 remaining 5 → P2 2–4; P3 (1) arrives at 4 → runs 4–5; then P2 remaining 2 → 5–7; P4 (4) vs P1 (5) → P4 7–11; P1 11–16. CT: P3 5, P2 7, P4 11, P1 16; WT = CT – AT – BT: P1 9, P2 1, P3 0, P4 2 → **avg 3**.

(c) Ch 6 §6.2 list; prevention pairs from §6.3 (spooling breaks mutual exclusion in effect; request-all-at-once breaks hold and wait — idle-held resources; resource ordering breaks circular wait — one global order for everyone).

Q3. (a) First Fit: 300 → 480 (leftover 180); 140 → 180 (leftover 40); 420 → no hole fits (holes 120, 40, 260, 340, 200) — failure. Best Fit: 300 → 340 (leftover 40); 140 → 200 (leftover 60); 420 → 480 (leftover 60) — all placed.

(b) $p = 1000 \div 256 = 3$, $d = 1000 - 768 = 232$; physical = $6 \times 256 + 232 = \mathbf{1768}$.

Pages needed = $\lceil 5000/256 \rceil = 20$ (19.53...); internal fragmentation = $20 \times 256 - 5000 = \mathbf{120 \text{ bytes}}$.

(c) LRU: only the repeated 2 (third reference) hits; the other nine references fault → **9 faults**. (Trace it: after 2, 3, 2, 1 the frames hold {2, 1}; 5 evicts 2, then 2 evicts 1, 4 evicts 5, 5 evicts 2, 3 evicts 4, 2 evicts 5 — the string keeps asking for exactly what LRU just removed.) Optimal: faults on 2, 3, 1, 5, 4, 3, 2 → **7 faults** (the references to 2 at positions 3 and 6, and to 5 at position 8, hit). The gap between 9 and 7 on one short string is LRU's price for not knowing the future.

Q4. (a) FCFS: 100→86 (14) →147 (61) →91 (56) →177 (86) →48 (129) →150 (102) →102 (48) →175 (73) = **569**. SCAN (up, to end 199): serve 102, 147, 150, 175, 177, then 199, then down: 91, 86, 48. Movement: 2 + 45 + 3 + 25 + 2 + 22 + 108 + 5 + 38 = **250**.

(b) Ch 11 §11.3 table rows; 4 × 1 TB: RAID 0 → 4 TB usable, tolerates 0; RAID 1 (two mirrored pairs in a 4-disk mirror set is RAID 10 in strict naming — with plain RAID 1 across 4 disks as two mirrors) → 2 TB usable, tolerates one per pair; RAID 5 → 3 TB usable, tolerates 1.

(c) Ch 11 §11.2: definition; partition = raw/fixed/fast, file = flexible/slower; bypass file system for speed and because contents are temporary (no names, sharing or backup).

Q5. (a) Ch 12 §12.1 attribute list; §12.2: sequential — CCTV/log; direct — database record fetch with the block-number idea.

(b) Ch 13 §13.5 table (any four rows); FAT collects the pointers into one cacheable table.

(c) Owner rwx (read, write, execute), group r-x (read, execute), others --- (none); leading '-' = ordinary file. Required string: **-rw-r-----**.

Model Paper 2

Time: 3 Hours · Maximum Marks: 60 Attempt any two parts of each question. All parts of a question carry 6 marks.

Q1. (Unit 1 — Overview)

(a) What is the kernel? Compare monolithic and microkernel designs on four points.

(b) Explain the layered architecture of UNIX/Linux, naming all layers from hardware to user.

(c) "A time-sharing system is a logical extension of multiprogramming." Justify in about eight lines.

Q2. (Unit 2 — Process Management)

(a) What is a semaphore? Define wait and signal precisely, and solve the producer-consumer problem using three semaphores, stating each one's initial value and job.

(b) A system has 4 processes and 3 resource types with totals $A = 9$, $B = 6$, $C = 6$. Allocation: $P_0 (1, 1, 0)$, $P_1 (2, 0, 1)$, $P_2 (1, 0, 2)$, $P_3 (0, 2, 0)$. Max: $P_0 (5, 3, 2)$, $P_1 (3, 2, 2)$, $P_2 (4, 1, 3)$, $P_3 (2, 4, 2)$. Find Available and Need, and determine whether the state is safe, giving a safe sequence if one exists.

(c) Explain the race condition with the shared-counter example, and state the three requirements of a critical-section solution.

Q3. (Unit 3 — Memory Management)

(a) Distinguish internal and external fragmentation with one example of each, and name one allocation scheme suffering each.

(b) Explain segmentation: the address parts, the segment table's contents, and the limit check. Translate (segment 1, offset 350) given segment 1: base 4200, limit 600 — and state what happens for offset 700.

(c) What is virtual memory? Explain thrashing, its cause in terms of working sets, and two remedies.

Q4. (Unit 4 — I/O System)

(a) Describe the physical structure of a hard disk (platter, track, sector, cylinder) and compute the capacity of a disk with 16 surfaces, 2048 tracks per surface, 128 sectors per track and 512 bytes per sector, in GiB.

(b) Distinguish host-attached storage, NAS and SAN by what each exports and over what path.

(c) A request queue is 30, 90, 60, 120, 10 with head at 50 (cylinders 0–199). Compute total head movement under SSTF and under C-SCAN (moving up; count the return jump).

Q5. (Unit 5 — File Management)

(a) Trace the evolution of directory structures from single-level to tree-structured, stating the problem each step solved. Include absolute vs relative paths in your answer.

(b) Explain the four free-space management schemes briefly. For free blocks {5, 6, 7, 20, 21, 40}, write the record under the counting scheme.

(c) What is file system mounting? Explain with a pen drive example, including mount point and safe removal.

Answer Hints — Model Paper 2

Q1. (a) Ch 2 §2.3 definition + the four-row comparison table (inside services, speed, reliability, example Linux/MINIX).

(b) Hardware → kernel → system-call interface → libraries/shell/utilities → users and applications (Ch 2 §2.5).

(c) QB Unit 1, Section C Q3's key points: same multi-job base, timer-based switching added, interactivity as the new property.

Q2. (a) Ch 5 §5.4 definition (atomicity stated) and §5.5: mutex = 1 (guard the buffer), empty = N (free slots), full = 0 (items); producer: wait(empty), wait(mutex), add, signal(mutex), signal(full); consumer mirrors with full/empty swapped.

(b) Allocated totals: A = 4, B = 3, C = 3 → **Available = (5, 3, 3)**. Need = Max – Allocation: P0 (4, 2, 2), P1 (1, 2, 1), P2 (3, 1, 1), P3 (2, 2, 2). Safety from Work (5, 3, 3): every Need fits already at the first pass — e.g., P1 (1,2,1) → Work (7, 3, 4); P0 (4,2,2) → (8, 4, 4); P2 (3,1,1) → (9, 4, 6); P3 (2,2,2) → (9, 6, 6). **Safe**, one sequence P1, P0, P2, P3 (others exist).

(c) Ch 5 §5.2 counter story (4 → 5 instead of 6) and §5.3's three requirements, one line each.

Q3. (a) Ch 9 §9.3 with Ch 7 examples: internal — 60 KB process in a 250 KB fixed partition / last page tail (fixed partitions, paging); external — 426 KB request failing amid 959 KB free (variable partitions, segmentation).

(b) Ch 8 §8.3: address = segment number + offset; table holds base + limit; check offset < limit: 350 < 600 ✓ → physical = 4200 + 350 = **4550**; offset 700 ≥

600 → **fault** (protection working).

(c) Ch 9 §9.4: definition (demand paging over a virtual space); thrashing when combined working sets exceed frames → fault storms and CPU collapse; remedies — guarantee frames per working set, or reduce the degree of multiprogramming.

Q4. (a) Ch 10 §10.2 anatomy; capacity = $16 \times 2048 \times 128 \times 512$ bytes = 2,147,483,648 bytes = **2 GiB**.

(b) Ch 10 §10.3: host-attached exports nothing (wired bus to one host); NAS exports files over the ordinary LAN; SAN exports raw blocks over a dedicated storage network.

(c) SSTF from 50: 60 (10), 90 (30), 120 (30), 30 (90), 10 (20) = **180**. C-SCAN up: serve 60, 90, 120, go to 199, jump to 0, serve 10, 30: movement $10 + 30 + 30 + 79$ (to 199) + 199 (jump) + 10 + 20 = **378**.

Q5. (a) QB Unit 5, Section C Q2's key points (single → two-level → tree), with /usr/bin/python3 as the absolute example and a relative form from a stated working directory.

(b) Ch 13 §13.6 one line each (bitmap, linked list, grouping, counting); counting record: **(5, 3), (20, 2), (40, 1)**.

(c) Ch 12 §12.4: mounting attaches a file system's root at a mount point (an existing directory) in the live tree; pen drive at /media/usb or a drive letter; unmount flushes pending writes — pulling early can corrupt, hence safe removal.



Previous Years' Papers — What the Board Actually Asked

(Read this first. This book is for the **BTEUP Polytechnic Diploma — Computer Science & IT** students. In the years up to 2025 this subject ran in the board's older scheme as **Operating System, Code 2207, 4th Semester** of that diploma: **50 marks in 2½ hours** — five questions, any two parts each, 5 marks a part — with a seventh unit on Linux that today's syllabus has dropped. From the current scheme it is **Subject 3.1, 3rd Semester, 60 marks, 3 hours**. Below are this diploma's real papers, exactly as asked. Every question carries its tag first — the year, the semester and the question number — and our answer hint follows each paper.)

Questions repeated across this diploma's papers

Two genuine papers of this diploma subject (Code 2207) are available — May 2025, and one more 2207 paper whose available copy does not carry its year. These questions appear in **both** of them — the board's proven favourites of this subject:

- **[May 2025 · Q1(b)] + [Year not printed · Q1(a)]** — Explain various functions of operating system. Answer: Chapter 2, §2.2.
- **[May 2025 · Q3(b)] + [Year not printed · Q3(b)]** — Explain paging, with example / as a note. Answer: Chapter 8, §8.2.
- **[May 2025 · Q4(b)] + [Year not printed · Q3(b)]** — Note / explanation on segmentation. Answer: Chapter 8, §8.3.

- **[May 2025 · Q4(a)] + [Year not printed · Q4(b)]** — Spooling: a note / buffering and spooling in detail. Answer: Paper 1, Q4(a) hint below.
- **[May 2025 · Q3(c)] + [Year not printed · Q2(c)]** — Deadlock: prevention / required conditions and handling. Answer: Chapter 6, §§6.2–6.3.
- **[May 2025 · Q5(a)] + [Year not printed · Q5(b)]** — Any five Linux commands with examples / functions. Old-syllabus Linux unit.
- **[May 2025 · Q5(c)] + [Year not printed · Q5(c)]** — Shell scripts: what are they / explain with examples. Old-syllabus Linux unit.
- **A named scheduling algorithm, examined every time** — Round Robin with example **[May 2025 · Q2(b)]**; the types of scheduling algorithms and an FCFS numerical **[Year not printed · Q1(b), Q2(b)]**.

Paper 1 — Operating System (Code 2207), May 2025, 4th Semester ✓ verified from the board's paper

Time Allowed: 2:30 Hours · Maximum Marks: 50 · Attempt ALL questions.

Q1 — Answer any two parts [2×5=10]:

- **[2025 · Sem 4 · Q1(a)]** Define operating system and explain its types.
- **[2025 · Sem 4 · Q1(b)]** Explain various functions of operating system.
- **[2025 · Sem 4 · Q1(c)]** Differentiate between system software and application software.

Q2 — Answer any two parts [2×5=10]:

- **[2025 · Sem 4 · Q2(a)]** What is a process state? Explain different process states.
- **[2025 · Sem 4 · Q2(b)]** Explain Round Robin (RR) scheduling algorithm with suitable example.
- **[2025 · Sem 4 · Q2(c)]** Discuss inter-process communication in detail.

Q3 — Answer any two parts [2×5=10]:

- [2025 · Sem 4 · Q3(a)] Differentiate between internal and external fragmentation.
- [2025 · Sem 4 · Q3(b)] Explain paging with example.
- [2025 · Sem 4 · Q3(c)] Explain the methods for deadlock prevention.

Q4 — Write notes on any two of the following [2×5=10]:

- [2025 · Sem 4 · Q4(a)] Spooling
- [2025 · Sem 4 · Q4(b)] Segmentation
- [2025 · Sem 4 · Q4(c)] Types of file system

Q5 — Answer any two parts [2×5=10] (old-syllabus Linux unit; NOT in today's Subject 3.1 — kept because the real paper contains it and back-paper students of the old scheme still need it):

- [2025 · Sem 4 · Q5(a)] Discuss any five Linux Commands with example.
- [2025 · Sem 4 · Q5(b)] What are Linux system requirements?
- [2025 · Sem 4 · Q5(c)] What do you mean by shell scripts?

Answer hints — Paper 1 (May 2025)

- **Q1(a)** [2025 · Sem 4 · Q1(a)] Chapter 1, §1.1 (the complete definition) and §1.3 (types table). **Q1(b)** Chapter 2, §2.2: the five functions. **Q1(c)** Chapter 1's system-software family vs the application programs of Figure 1.1.
- **Q2(a)** Chapter 3, §3.2 with Figure 3.1 — draw the diagram; boards reward it. **Q2(b)** Chapter 4, §4.4 (Dataset A's RR run is ready-made). **Q2(c)** Chapter 5, §5.1.
- **Q3(a)** Chapter 9, §9.3. **Q3(b)** Chapter 8, §8.2 (753 → 453). **Q3(c)** Chapter 6, §6.3.

- **Q4(a) Spooling:** print jobs are first written to a disk queue (the spool) and printed in turn, so programs never wait on the slow printer and the printer is shared without being held — Chapter 6, §6.3; the everyday picture is Chapter 1's print queue. **Q4(b)** Chapter 8, §8.3. **Q4(c)** Chapter 13, §13.7.
- **Q5 (Linux, old syllabus):** (a) any five of `ls`, `cd`, `mkdir`, `cp`, `cat`, `rm`, `pwd`, `chmod` with a one-line job and example, as practised in Lab Experiment 1.
(b) A typical Linux distribution installs on almost any modern PC: 64-bit processor, about 2 GB or more of RAM, roughly 20–25 GB of free disk for common desktop distributions (minimal installs need far less) — the board's point: Linux runs well on modest hardware.
(c) A shell script is a plain text file of shell commands in order, executed one by one by the shell (`chmod +x file.sh`, then `./file.sh`).

Paper 2 — Operating System (Code 2207), year not printed on the available copy ✓ genuine board paper

(This paper carries the printed **Code No. 2207** of this diploma subject and the same 50-marks / 2½-hour pattern. The copy available publicly is a one-page preview on which the year and semester are not visible, so no year is claimed for it. Its questions are printed exactly as asked, and its numerical is solved below.)

Time: 2:30 Hours • Maximum Marks: 50 • Minimum Marks: 17 • Attempt all questions, any two parts of each.

Q1:

- **[Year not printed • Q1(a)]** Explain the functions of an operating system.

- **[Year not printed · Q1(b)]** Explain the different types of process scheduling algorithms.
- **[Year not printed · Q1(c)]** Write short notes on: i) Context switch ii) Process Control Block (PCB).

Q2:

- **[Year not printed · Q2(a)]** Differentiate between Pre-emptive and Non pre-emptive algorithms with examples of each.
- **[Year not printed · Q2(b)]** Numerical: find the average waiting time and average turnaround time for the following processes using the FCFS algorithm — P0: Arrival 1, Burst 4; P1: Arrival 0, Burst 12; P2: Arrival 2, Burst 3 (burst time in ms).
- **[Year not printed · Q2(c)]** Explain the required conditions for deadlock and methods to handle deadlocks.

Q3:

- **[Year not printed · Q3(a)]** Explain in detail the memory management in operating system. Also, discuss logical address space and physical address space.
- **[Year not printed · Q3(b)]** Write short notes on: i) Paging ii) Segmentation.
- **[Year not printed · Q3(c)]** Explain virtual memory in detail.

Q4:

- **[Year not printed · Q4(a)]** Discuss system call and types of system calls.
- **[Year not printed · Q4(b)]** Discuss Buffering and spooling in detail.
- **[Year not printed · Q4(c)]** Explain I/O management functions.

Q5:

- **[Year not printed · Q5(a)]** Write an overview of Linux operating system. Explain its structure and features.

- **[Year not printed · Q5(b)]** Write the functions of following Linux commands: i) rmdir ii) pwd iii) grep iv) chmod v) kill.
- **[Year not printed · Q5(c)]** Explain shell scripts and shell commands with suitable examples.

Paper 2's numerical, solved

FCFS serves in arrival order: P1 arrives first (time 0), then P0 (time 1), then P2 (time 2). Gantt: **P1 runs 0–12, P0 runs 12–16, P2 runs 16–19.**

Completion times: P1 = 12, P0 = 16, P2 = 19. Turnaround = completion – arrival: P1 = 12, P0 = 15, P2 = 17 → **average turnaround time = $44/3 \approx$**

14.67 ms. Waiting = turnaround – burst: P1 = 0, P0 = 11, P2 = 14 → **average waiting time = $25/3 \approx 8.33$ ms.** The lesson, as always with FCFS: P2's tiny 3 ms job waits 14 ms behind P1's long one — the convoy effect (Chapter 4, §4.2). Remaining hints for Paper 2: Q1 — Chapter 2 §2.2, Chapter 4 overview, Chapter 3 §§3.4–3.5; Q2(a) Chapter 4 §4.1, Q2(c) Chapter 6 §§6.2–6.3; Q3 — Chapter 7 §7.1, Chapter 8 §§8.2–8.3, Chapter 9 §9.4; Q4 — Chapter 2 §2.4 (system calls), buffering/spooling as in Paper 1 Q4(a), Chapter 2 §2.2's I/O function; Q5 — old Linux unit: `rmdir` removes an empty directory, `pwd` prints the current directory, `grep` searches text in files, `chmod` changes permissions, `kill` ends a process by its PID; shell scripts as in Paper 1 Q5(c).

The 2024 examinations — held, papers not public

The board's official examination schemes confirm this diploma's paper was held twice in 2024: **1 July 2024** (even-semester main examination, 4th Semester) and **16 January 2024** (Back & Special Back examination), both as Code 2207 in a 2½-hour slot. The question papers themselves are not available in any public repository we could reach, so — rather than guess at their contents — this bank prints nothing for 2024. If your institute's

library keeps board papers, those two dates are the ones to ask for. Papers for 2020, 2021 and 2022 could likewise not be verified from any reliable source, and none is printed.

What these papers teach you (the honest pattern)

Across this diploma's real papers, the same skeleton shows: one question on OS basics, types and functions; one on processes and scheduling — with a named algorithm every time; one on memory (paging, fragmentation) with deadlock beside it; short notes on the classic small topics (buffering, spooling, segmentation); file-system and system-call questions; and the old Linux unit to close. Prepare those pillars first — they are this bank's Units 1–3 and the worked numericals — and no paper of this subject, old scheme or new, can surprise you.

