



KAMALDNP

kamaldnp.com • IT Training, Varanasi

PDF created by Raj Kamal using AI

Model Practice Paper — Operating Systems

BTEUP Diploma in Computer Science & Engineering • 3rd Semester • Session 2026-27
Session 2026-27

Practice Questions / Model Paper — NOT a BTEUP board paper

Time: 3 Hours

Max. Marks: 60

General Instructions:

- All questions are compulsory. Each question has three parts (a), (b) and (c) — **answer any two** of the three parts in every question.
- Each part carries 6 marks; each question carries 12 marks.
- Draw neat, labelled diagrams wherever they are asked or wherever they support your answer.
- Write your answers in your own words, in the same order as the questions.

About this paper: This is an original practice (model) paper prepared by KAMALDNP Academy from the official BTEUP 2026-27 syllabus topics for this subject. It is NOT a BTEUP board paper, and no question in it is promised to appear in your examination. In the verified 2026-27 scheme this theory paper carries 60 marks (external) in 3 hours, and this model paper is scaled to that maximum; its structure follows the board's last published paper pattern (5 questions, each with three parts of which any two are to be answered). The 2026-27 scheme's own paper format has not been published yet — when your board paper is in your hand, its printed instructions govern.

Q1. (From: Unit 1 — Overview of Operating System) — Answer any two of the following:

- (a) Discuss the different types of operating systems. Give one example and one use of each type. **(6)**

- (b) Describe the main functions and services of an operating system. (6)
- (c) Explain the UNIX/Linux architecture with a neat diagram. In your answer, state the role of the kernel, a system call and system programs. (6)

Q2. (From: Unit 2 — Process Management) — Answer any two of the following:

- (a) Explain the concept of a process. Describe the main operations on processes, namely process creation and process termination. (6)
- (b) Three processes arrive at time 0 in the order P1, P2, P3, with CPU burst times 6, 2 and 4 units respectively. Draw the Gantt chart and calculate the average waiting time under (i) FCFS and (ii) non-preemptive SJF scheduling. (6)
- (c) Discuss deadlock in an operating system. State the conditions necessary for deadlock and explain how deadlock can be prevented and how it can be avoided. (6)

Q3. (From: Unit 3 — Memory Management) — Answer any two of the following:

- (a) Explain paging with a diagram. Differentiate between paging and segmentation, and state what happens on a page fault. (6)
- (b) Memory blocks in order are of 100 KB, 500 KB, 200 KB, 300 KB and 600 KB. Processes of 212 KB, 417 KB, 112 KB and 426 KB arrive in this order. Show which block each process gets, or that it cannot be allocated, under First Fit, Best Fit and Worst Fit. (6)
- (c) For the page reference string 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 with three page frames, count the page faults under FIFO and under LRU page replacement. Show the frame contents step by step. (6)

Q4. (From: Unit 4 — I/O System) — Answer any two of the following:

- (a) Describe the structure of a disk in brief. Differentiate between host-attached storage and network-attached storage. (6)
- (b) A disk queue has requests for cylinders 82, 170, 43, 140, 24, 16 and 190 in this order. The head is at cylinder 50 and the disk has cylinders 0 to 199. Calculate the total head movement under (i) FCFS and (ii) SCAN, taking SCAN to be moving first towards larger cylinder numbers. (6)
- (c) Write a short note on RAID. Explain any four RAID types in brief and state where swap space is kept and what it is used for. (6)

Q5. (From: Unit 5 — File Management) — Answer any two of the following:

- (a) Explain the concept of a file. Describe the sequential and direct file access methods, with one use of each. (6)
- (b) Describe the main directory structures in brief, from single-level to tree-structured directory. Also explain file system mounting in two or three lines. (6)
- (c) Discuss file sharing and file protection. How does an operating system control which user may read, write or execute a file? (6)

Model Answer Outline — Operating Systems

The key points a full-marks answer contains, question by question.

Practice Questions / Model Paper — NOT a BTEUP board paper

Q1(a)

- Batch OS: jobs collected and run in batches, no direct interaction during running; example: payroll processing.
- Multiprogramming OS: several jobs kept in memory, CPU switches to another job when one waits for I/O, so CPU idle time falls.
- Time-sharing OS: multiprogramming with rapid switching and a small time slice, so many users get interactive response; example: a UNIX terminal server.
- Real-time OS: result must come within a fixed deadline; used in process control and embedded devices.
- Distributed OS: work is spread over computers joined by a network and the system appears as one system to the user.

Q1(b)

- Program execution: loads a program, runs it and ends it normally or abnormally.
- I/O operations and file-system manipulation: programs cannot do I/O directly, so the OS does it for them; it also creates, reads, writes and deletes files.
- Resource allocation and accounting: CPU, memory and devices are given to processes fairly, and usage is recorded.
- Error detection and protection/security: errors in CPU, memory, devices and user programs are detected; passwords and permissions stop unauthorised use.
- Communication service: processes on the same or different computers exchange information through the OS.

Q1(c)

- Diagram in layers: users on the outside, then system and application programs, then the UNIX operating system (kernel), then hardware at the centre.
- Kernel: the core part that stays in memory; it manages CPU, memory, files and devices and is the only part that directly controls hardware.
- System call flow: a program asks the kernel through a system call, for example to open or read a file; the kernel checks the request, does the work and returns the result.
- System programs sit above the kernel: shells, compilers, editors and utilities such as cp, mv and mkdir, which the user actually runs.
- This layering keeps application programs away from direct hardware access, which gives safety and a uniform way of working.

Q2(a)

- A process is a program in execution: it includes the program code, current activity, stack, data section and heap; the same program can be several processes for different users.
- The OS keeps a process control block (PCB) for each process, holding its state, program counter, CPU registers and scheduling information.
- Creation: a parent process creates a child process, for example with fork in UNIX; the child gets resources from the OS or shares part of the parent's resources.
- Termination: a process ends after its last instruction and asks the OS to delete it; the OS frees its memory and other resources.
- A parent may also terminate a child, for example when the child has exceeded its allotted resources or its task is no longer needed.

Q2(b)

- FCFS Gantt chart: P1 runs 0-6, P2 runs 6-8, P3 runs 8-12. Waiting times: P1 = 0, P2 = 6, P3 = 8.
- FCFS average waiting time = $(0 + 6 + 8) / 3 = 14 / 3 = 4.67$ units.
- SJF order is by burst time: P2 (2), P3 (4), P1 (6). Gantt chart: P2 runs 0-2, P3 runs 2-6, P1 runs 6-12.
- SJF waiting times: P2 = 0, P3 = 2, P1 = 6. Average = $(0 + 2 + 6) / 3 = 8 / 3 = 2.67$ units.
- Conclusion: SJF gives the smaller average waiting time here, but it needs the burst time to be known or estimated in advance.

Q2(c)

- Deadlock: a set of processes is blocked for ever because each one holds a resource and waits for a resource held by another process in the set.
- Four necessary conditions, all together: mutual exclusion, hold and wait, no preemption, and circular wait.
- Prevention: break any one condition by design, for example by making a process request all resources at once (breaks hold and wait) or by imposing an order on resource types (breaks circular wait).
- Avoidance: before granting a request, the OS checks that the system will stay in a safe state, as in the Banker's algorithm idea; an unsafe request is made to wait.
- Prevention is a fixed rule applied in advance, while avoidance is a check made at each request using extra information about future needs.

Q3(a)

- Paging: physical memory is divided into fixed-size frames and the process is divided into pages of the same size; pages are placed in any free frames, so no contiguous block is needed.
- A page table maps each logical page number to a frame number; the offset part of the address is used unchanged.
- Page fault: if the needed page is not in memory, the OS loads it from disk into a free frame (replacing a page if needed) and restarts the instruction.

- Segmentation divides the program into variable-size logical segments such as code, data and stack; paging uses equal fixed sizes chosen by the OS.
- Paging has no external fragmentation but may have a small internal fragmentation in the last page; segmentation can suffer external fragmentation. Paging is the basis of virtual memory, where only needed pages stay in RAM.

Q3(b)

- First Fit (first block big enough): 212 KB goes into the 500 KB block, 417 KB into the 600 KB block, 112 KB into the remaining 288 KB of the 500 KB block; 426 KB cannot be allocated.
- Best Fit (smallest block big enough): 212 KB into the 300 KB block, 417 KB into the 500 KB block, 112 KB into the 200 KB block, 426 KB into the 600 KB block; all four are allocated.
- Worst Fit (largest block): 212 KB into the 600 KB block, 417 KB into the 500 KB block, 112 KB into the remaining 388 KB of the 600 KB block; 426 KB cannot be allocated.
- Rule summary: First Fit scans from the start, Best Fit leaves the smallest leftover, Worst Fit leaves the largest leftover.
- Result to note: in this classic set, Best Fit is the only one of the three that fits all four processes.

Q3(c)

- Count a fault whenever the referenced page is not already in one of the three frames; the string is 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2.
- FIFO replaces the page that entered first: faults occur on 2, 3, 1, 5, 2, 4, 3, 5 and 2, giving 9 page faults in total.
- LRU replaces the page not used for the longest time: for example, when 5 arrives the victim is 3, and when 4 arrives the victim is 1.
- LRU faults occur on 2, 3, 1, 5, 4, 3 and 2, giving 7 page faults in total.
- Conclusion: LRU gives fewer faults here because recent use is a good guide to near-future use; FIFO ignores use completely.

Q4(a)

- A disk is made of platters; each surface has circular tracks, and each track is divided into sectors, the smallest transfer unit.
- Cylinders: the set of tracks at the same arm position on all surfaces; a block is located by surface, track (cylinder) and sector.
- The arm moves the head to the track (seek), the platter rotates the sector under the head (rotational latency), then data is transferred.
- Host-attached storage is connected directly to the computer through an I/O bus, for example an internal or USB disk used by that one machine.
- Network-attached storage is accessed over the network through a remote file system, so several machines can share the same storage.

Q4(b)

- FCFS serves in queue order 82, 170, 43, 140, 24, 16, 190 from head 50: movements are $32 + 88 + 127 + 97 + 116 + 8 + 174$.

- FCFS total head movement = 642 cylinders.
- SCAN, moving first towards larger numbers, serves 82, 140, 170, 190, continues to the end cylinder 199, then reverses and serves 43, 24, 16.
- SCAN total = $(199 - 50) + (199 - 16) = 149 + 183 = 332$ cylinders.
- Conclusion: SCAN gives much less movement than FCFS here because it serves requests in the direction of travel instead of jumping back and forth.

Q4(c)

- RAID (Redundant Array of Independent Disks) combines several disks so the group gives better performance, larger capacity, or safety against one disk failing, or a mix of these.
- RAID 0: striping only, data split across disks; fast, but no safety copy, so one disk failure loses data.
- RAID 1: mirroring, the same data on two disks; safe against one failure, but usable capacity is halved. RAID 5: striping with distributed parity, tolerates one disk failure. RAID 6: striping with double parity, tolerates two disk failures.
- Swap space is an area on disk used as an extension of main memory for pages or whole processes temporarily moved out of RAM.
- Swap space management: the OS reserves and tracks this area separately from the normal file system, because swap access must be fast and simple.

Q5(a)

- A file is a named collection of related information stored on secondary storage, as seen by the user; the OS maps it onto physical blocks.
- A file has attributes such as name, type, location, size, protection and time/date, and operations such as create, write, read, delete and open/close.
- Sequential access: information is processed in order, one record after another, as in a tape or a log file being read from the start.
- Direct (random) access: any block can be read or written directly by its number, as needed by a database file on a disk.
- Sequential access is simple but slow for a middle record; direct access needs addressable storage like a disk.

Q5(b)

- Single-level directory: one directory holds all files for all users; simple, but file names must be unique and searching is slow.
- Two-level directory: each user gets a separate directory, so two users may use the same file name.
- Tree-structured directory: directories can contain sub-directories, forming a tree; users work with absolute and relative path names. This is the structure used in practice.
- Acyclic-graph and general graph structures additionally allow shared files and directories in more than one place, at the cost of harder deletion and bookkeeping.
- File system mounting: a file system must be attached (mounted) at a mount point in the existing directory tree before its files can be accessed.

Q5(c)

- File sharing lets several users work on the same file; the OS must control simultaneous access so updates do not clash, and remote sharing works through the network.
- Protection is needed because in a shared system a user should reach only the files he is allowed to reach.
- Access types controlled by the OS include read, write, execute, append, delete and list.
- A common method is an access list or group scheme: owner, group and others each get a set of allowed operations, as in the rwx permissions of UNIX/Linux.
- The owner sets the permissions; at every open operation, the OS checks the user's identity against the allowed access before granting it.

KAMALDNP • kamaldnp.com • support@kamaldnp.com • +91 8318013442 • Varanasi, UP
Prepared by KAMALDNP Academy from the official BTEUP 2026-27 syllabus. This is a practice/model paper only.

