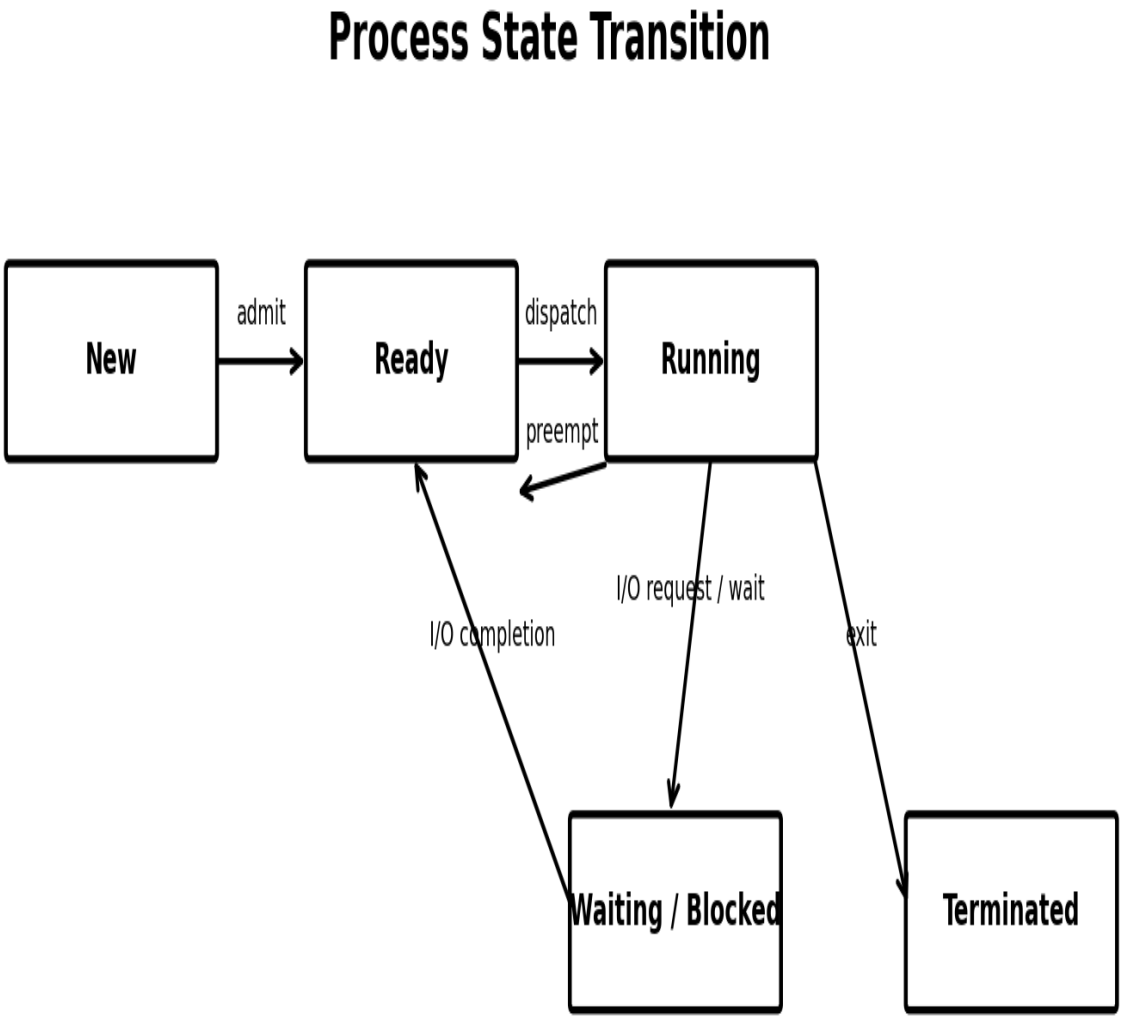
**OPERATING SYSTEM – UNIT II**

**PROCESS MANAGEMENT**

**Course Code: 315319 | Semester V | K Scheme | 14 Marks**

*Syllabus-aligned • Exam-linked • Diagram-rich • Publishing-ready*




Prepared from the supplied Unit-II notes, MSBTE syllabus and the Summer 2026 / Winter 2025 question papers.

**Dr. Ganesh Jorvekar**

## Unit Scope and Learning Outcomes

| Syllabus area | Coverage in these notes | TLO alignment |
|---|---|---|
| 2.1 Processes | Process concept, process states, PCB | TLO 2.1, 2.2 |
| 2.2 Process Scheduling | Scheduling queues, long/medium/short-term schedulers, context switch | TLO 2.2 |
| 2.3 IPC | Shared memory and message passing | TLO 2.3 |
| 2.4 Threads | Benefits, user/kernel threads, one-to-one, many-to-one, many-to-many | TLO 2.4 |
| 2.5 Linux commands | top, ps, kill, wait, sleep, exit, nice | TLO 2.2 / practical CO2 |

**Syllabus basis:** Unit II is Process Management (14 marks). The syllabus specifies TLO 2.1–2.4 and topics 2.1–2.5.

The specification table allocates Unit II 10 learning hours and 14 marks: R-level 4, U-level 4, A-level 6.

**EXAM LINK |** Across the two supplied papers, Unit II generated 12 sub-question opportunities (6 in Summer 2026 and 6 in Winter 2025). This is a past-paper observation, not a prediction.

## 1. Process Concept

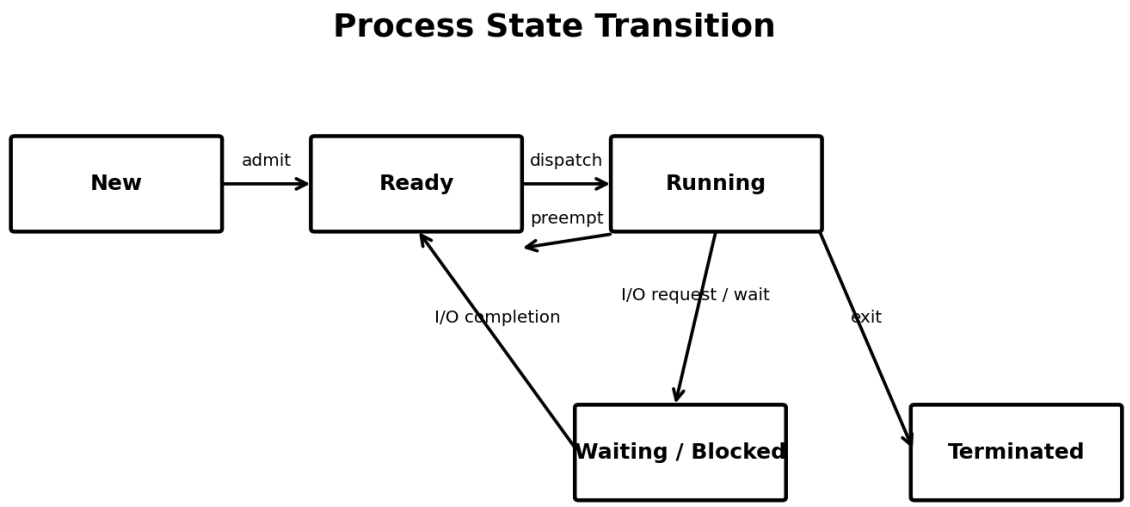
A process is a program in execution. It is an active entity with a current state, execution context, resources and associated program data. The supplied notes describe a process as the fundamental unit of computation and identify program, input, output and state as important aspects.

### 1.1 Process memory layout

- **Text section:** contains executable program code.
- **Data section:** contains global and static variables.
- **Heap:** used for dynamic memory allocation.
- **Stack:** stores local variables, parameters, return addresses and temporary data.
- **Program counter:** contains the address of the next instruction to be executed.

Exam focus: be able to define a process, identify its major memory sections and connect the program counter/registers with process state.

## 2. Process States



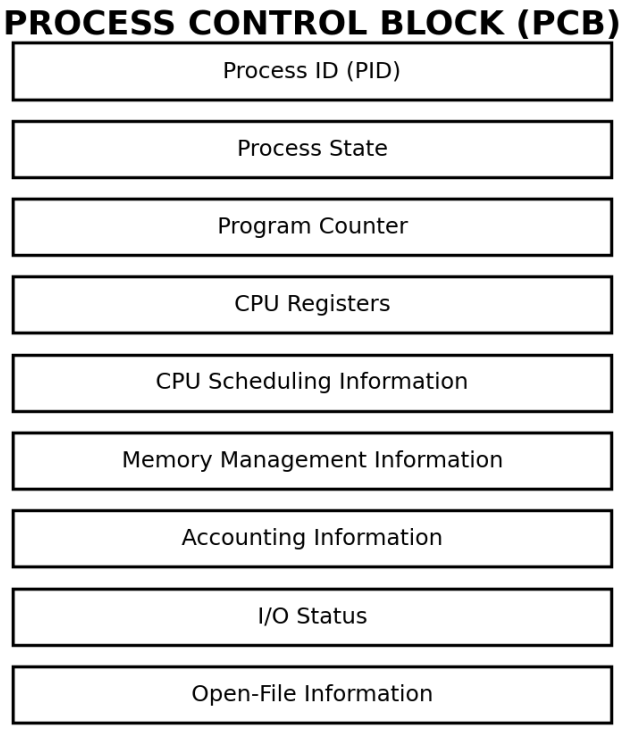

- **New:** the process is being created and has not yet been admitted to the ready state.
- **Ready:** the process is in main memory and waiting for CPU allocation.
- **Running:** the process currently has control of the CPU.
- **Waiting / Blocked:** the process is waiting for an I/O operation or event to complete.
- **Terminated:** the process has completed execution; the OS releases its resources.

**Typical transitions:** New → Ready → Running; Running → Waiting when I/O/event wait occurs; Waiting → Ready after completion; Running → Ready when preempted; Running → Terminated after completion.

**EXAM LINK |** Summer 2026 Q1(a): List the different states of a process. Winter 2025 Q3(a): Draw and explain the process-state diagram.

3. Process Control Block (PCB)

A Process Control Block is the OS data structure maintained for a process. It is created when the process is created and removed when the process terminates. The PCB stores the execution context and management information required by the OS.




Created with the process; maintained until termination.

### 3.1 Major PCB fields

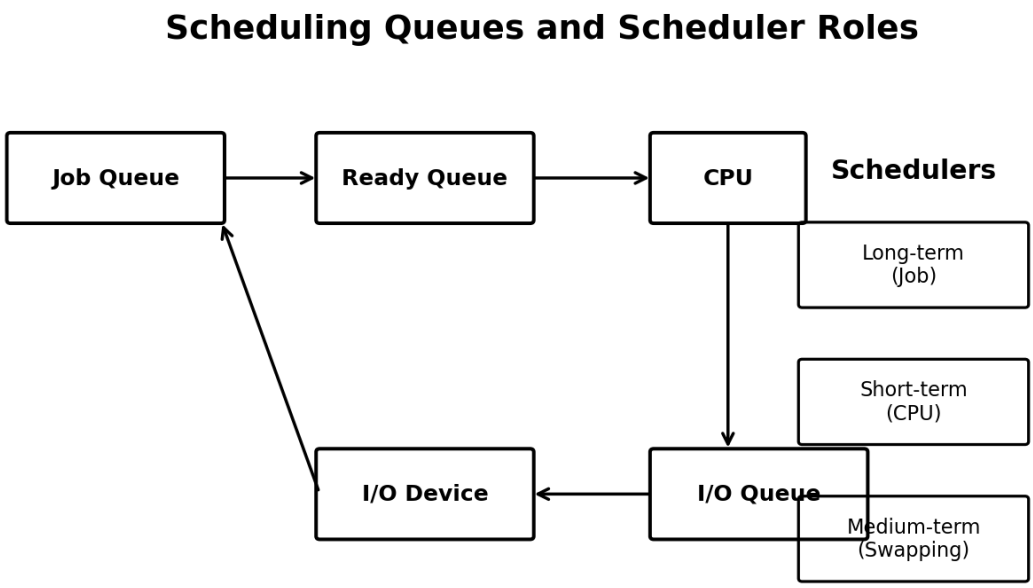
| PCB field | Purpose |
|---|---|
| **Process ID (PID)** | Unique identifier assigned to the process. |
| **Process state** | New, ready, running, waiting/blocked or terminated. |
| **Program counter** | Address of the next instruction to execute. |
| **CPU registers** | Saved register values needed to resume execution correctly. |
| **CPU scheduling information** | Priority, queue pointers and related scheduling parameters. |
| **Memory-management information** | Base/limit information or page/segment table information, depending on the memory system. |

| | |
|---|---|
| **Accounting information** | Resource/CPU usage information. |
| **I/O status** | Outstanding I/O requests and allocated/pending device information. |
| **File information** | Open-file information and access rights. |

**EXAM LINK |** Winter 2025 Q4(d): Explain process control block with a neat diagram.

4. Process Scheduling

Processes move through scheduling queues during their execution. The supplied notes identify the ready queue and I/O queue and describe the CPU–I/O cycle.




### 4.1 Scheduling queues

- **Job queue:** contains processes entering the system.
- **Ready queue:** contains processes in main memory that are ready and waiting for CPU allocation.
- **I/O queue:** contains processes waiting for an I/O device or I/O completion.

### 4.2 Types of schedulers

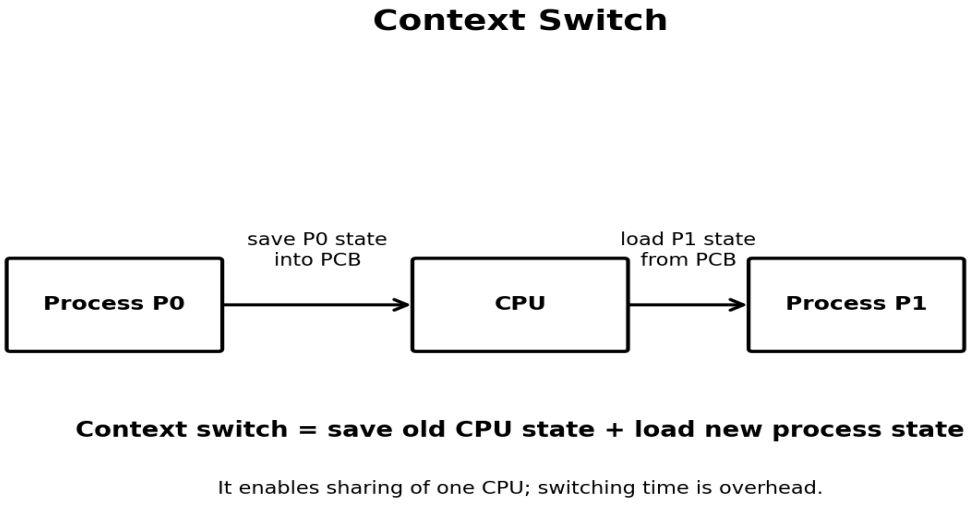
| Scheduler | Main function | Typical activity |
|---|---|---|
| Long-term scheduler (Job scheduler) | Selects jobs from the job pool and admits them into memory; controls the degree and mix of multiprogramming. | Relatively infrequent |
| Short-term scheduler (CPU scheduler) | Selects a ready process and allocates the CPU to it. | Very frequent / fast |
| Medium-term scheduler | Temporarily removes processes from main memory and later brings them back; associated with swapping and reducing degree of multiprogramming. | Intermediate frequency |

**Important boundary with Unit III:** Unit II covers the types/roles of schedulers and context switching. The detailed CPU-scheduling algorithms (FCFS, SJF, SRTN, RR, Priority, Multilevel Queue) are prescribed in Unit III.

**EXAM LINK |** Summer 2026 Q2(b): List types of scheduler and describe the function of each. Winter 2025 Q4(e): State the meaning of scheduler and explain any one type.

5. Context Switch

A context switch occurs when the CPU changes from one process to another. The OS saves the state of the old process and loads the saved state of the new process. The saved context is associated with the process's PCB.

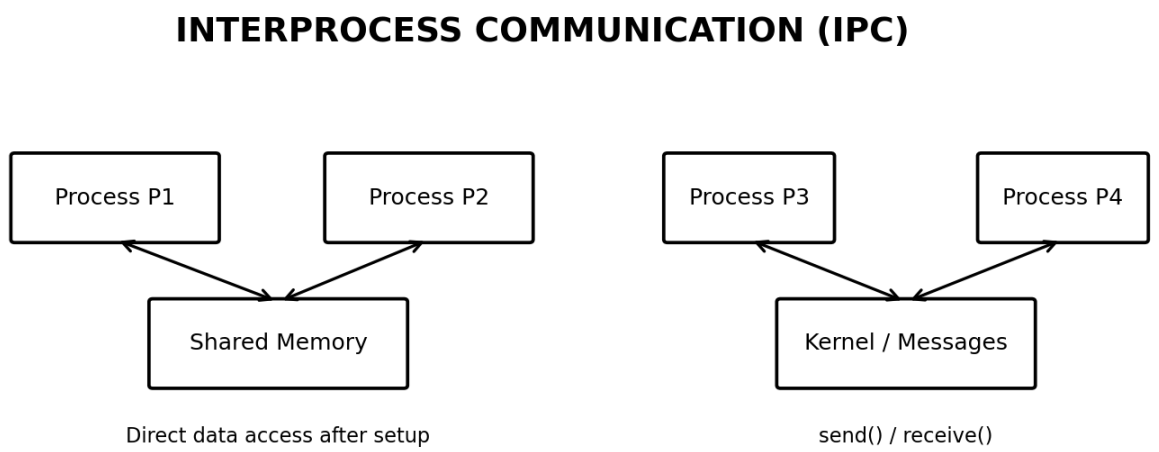


- **Purpose:** allows multiple processes to share a CPU.
- **Saved context:** includes processor-register information and the program counter needed to resume the process.
- **Performance effect:** context-switch time is overhead; it does not directly perform useful application work.

Exam-ready line: Context switching = save old process context + load new process context.

## 6. Interprocess Communication (IPC)

Interprocess Communication provides mechanisms that allow cooperating processes to exchange information and coordinate activity. The syllabus specifies two IPC approaches: shared memory and message passing.




### 6.1 Shared memory

- **Concept:** two or more processes communicate through a shared memory region.
- **Operation:** processes read and write data in the shared region after it is established.
- **Strength:** after setup, accesses can be performed as ordinary memory accesses with low communication overhead.
- **Design responsibility:** the communicating processes must coordinate access and maintain data consistency; synchronization may be required.

### 6.2 Message passing

- **Concept:** processes communicate by sending and receiving messages.
- **Kernel involvement:** message passing typically uses OS mechanisms to send/receive messages.
- **Use case:** particularly useful when processes do not share the same address space and in distributed environments.
- **Naming:** communication may be direct (process-to-process) or indirect (mailbox/port based).

- **Synchronization:** send/receive may be blocking or non-blocking.
- **Buffering:** messages may be held temporarily in buffers until received.

**EXAM LINK |** Summer 2026 Q6(a): Explain IPC and describe shared memory with advantages and disadvantages.

## 7. Threads

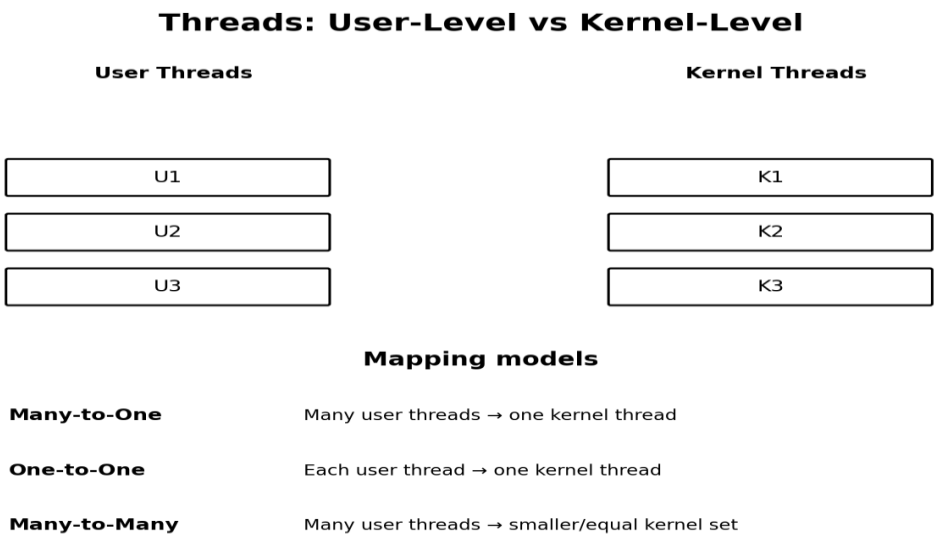
A thread is a basic unit of CPU utilization within a process. A process may contain one or more threads. Threads belonging to the same process share the process's code and resources while maintaining their own execution state.

### 7.1 Benefits of multithreading

| Benefit | Explanation |
|---|---|
| **Responsiveness** | One thread can continue while another is blocked or performing a lengthy operation. |
| **Resource sharing** | Threads share the process's address space and resources. |
| **Economy** | Creating and switching threads is generally less costly than creating and switching full processes. |
| **Utilization of multiprocessors** | Multiple threads can execute in parallel on multiple processors/cores. |

### 7.2 User-level and kernel-level threads

- **User-level threads:** managed by a thread library in user space; thread management can be fast because it does not require a kernel transition for every management operation.
- **Kernel-level threads:** managed and scheduled by the operating system kernel; the kernel can schedule individual threads and support concurrency on multiprocessor systems.




**EXAM LINK |** Summer 2026 Q3(b): Describe benefits of using threads. Winter 2025 Q1(e): Define thread and enlist two benefits.

## 8. Multithreading Models

The syllabus prescribes three models: One-to-One, Many-to-One and Many-to-Many. The model describes how user-level threads are mapped to kernel-level threads.

| Model | Mapping | Key points |
|---|---|---|
| Many-to-One | Many user-level threads are mapped to one kernel thread. | Simple and low management overhead; a blocking system call can block the entire process; no parallel execution of its threads on multiple processors. |
| One-to-One | Each user-level thread is mapped to one kernel thread. | More concurrency; one blocked thread need not block the others; creating many threads can increase overhead. |
| Many-to-Many | Many user-level threads are mapped to a smaller or equal number of kernel threads. | Provides flexibility in mapping and supports concurrency without requiring one kernel thread for every user thread. |

**EXAM LINK |** Summer 2026 Q4(b): Describe the many-to-one multithreading model with a suitable diagram. Winter 2025 Q3(d): Explain any two multithreading models with suitable diagram.

## 9. Linux Process-Related Commands

### Linux Process-Related Commands

| | |
|---|---|
| **top** | interactive view of running processes |
| **ps** | display process status |
| **kill** | send a signal to a process |
| **wait** | wait for a child/background process |
| **sleep** | pause execution for a specified time |
| **exit** | terminate the current shell |
| **nice** | run a process with a specified scheduling priority |

Lab extension in syllabus: renice, bg, fg; shared-memory tool: ipcs.

| Command | Purpose |
|---|---|
| **top** | Interactive display of running processes and system activity. |
| **ps** | Displays process status; options can show selected or detailed processes. |
| **kill** | Sends a signal to a process; commonly used to request termination. |
| **wait** | Waits for a child/background process and can return its completion status. |
| **sleep** | Suspends execution for a specified time. |
| **exit** | Exits the current shell and returns an exit status. |
| **nice** | Starts a process with an adjusted scheduling priority. |

**Practical extension from the syllabus laboratory section:** renice, bg, fg and ipcs are also listed for process/shared-memory related practice.

**EXAM LINK |** Summer 2026 Q1(b): state the use of ps and kill. Winter 2025 Q5(b): interpret sleep, ps -u and wait commands.

## 10. Supporting / Enrichment Content

The supplied notes also contain process creation/termination and the critical-section problem. These are useful supporting concepts, but they are not separately listed under the supplied Unit-II theory-content headings and were not directly represented in the two supplied question papers. They are therefore retained here as enrichment rather than treated as core syllabus headings.

### 10.1 Process creation and termination

- **Creation:** a parent process may create a child process; the OS creates/initializes the child's PCB and makes it eligible for execution.
- **Termination:** a process may terminate after completing execution or may be terminated by an appropriate OS mechanism; resources such as memory, files and I/O buffers are released.
- **wait():** the supplied notes associate wait() with a parent process waiting for a child process and receiving its completion status.

### 10.2 Critical section problem

- **Critical section:** part of a process where shared data/resources may be accessed.
- **Entry section:** code that requests permission to enter the critical section.
- **Exit section:** code that releases the shared resource after the critical section.
- **Remainder section:** the remaining code executed outside the critical section.

Teaching note: synchronization tools and detailed mutual-exclusion solutions belong naturally with later concurrency coverage; do not substitute them for the Unit-II IPC and thread outcomes.

## 11. Quick Revision Sheet

| Topic | One-line recall |
|---|---|
| **Process** | Program in execution; active entity with state and execution context. |
| **Process states** | New, Ready, Running, Waiting/Blocked, Terminated. |
| **PCB** | OS record containing PID, state, PC, registers, scheduling, memory, accounting, I/O and file information. |
| **Scheduling queues** | Job queue, Ready queue, I/O queue. |
| **Long-term scheduler** | Admits jobs to memory; controls degree/mix of multiprogramming. |
| **Short-term scheduler** | Selects a ready process and allocates CPU. |
| **Medium-term scheduler** | Handles swapping; temporarily removes/restores processes. |

| | |
|---|---|
| **Context switch** | Save old context + load new context; overhead. |
| **IPC** | Communication among processes via shared memory or message passing. |
| **Thread** | Basic unit of CPU utilization within a process. |
| **Thread benefits** | Responsiveness, resource sharing, economy, multiprocessor utilization. |
| **Models** | Many-to-One, One-to-One, Many-to-Many. |
| **Linux commands** | top, ps, kill, wait, sleep, exit, nice. |

## 12. Question-Paper Mapping: Unit II

The following mapping is based on the two supplied papers and classifies each Unit-II sub-question against the Unit-II syllabus. It is an exam-alignment tool, not a prediction of future questions.

| Paper | Mapped Unit-II questions | Opportunities |
|---|---|---|
| Summer 2026 | Q1(a) process states; Q1(b) ps/kill; Q2(b) types of schedulers; Q3(b) thread benefits; Q4(b) many-to-one model; Q6(a) IPC/shared memory. | 6 |
| Winter 2025 | Q1(e) thread + benefits; Q3(a) process-state diagram; Q3(d) multithreading models; Q4(d) PCB; Q4(e) scheduler/type; Q5(b) Linux process commands. | 6 |
| Total | Unit II opportunities across both supplied papers. | 12 |

### Exam-oriented question checklist

- Define a process and explain the process-state transitions with a neat diagram.
- Explain PCB and its major fields with a diagram.
- Explain job, ready and I/O scheduling queues.
- Differentiate long-term, medium-term and short-term schedulers.
- Explain context switching and its performance overhead.
- Explain IPC; compare shared memory and message passing.
- State the benefits of multithreading.
- Differentiate user-level and kernel-level threads.

- Explain Many-to-One, One-to-One and Many-to-Many models.
- State the purpose of top, ps, kill, wait, sleep, exit and nice.

## 13. References and Source Notes


1. [1] MSBTE, Operating System, Course Code 315319, Semester V, K Scheme, approved 24/02/2025. Unit II, TLO 2.1–2.4, CO2. Supplied syllabus copy.
2. [2] Abraham Silberschatz, Peter B. Galvin & Greg Gagne, Operating System Concepts, 10th Edition, Wiley. The companion site lists Chapter 3 (Processes) and Chapter 4 (Threads & Concurrency), supporting the process/thread organization used here.
3. [3] William Stallings, Operating Systems: Internals and Design Principles, 9th Edition, Pearson. Part II covers Process Description and Control, Threads, and Concurrency; Chapter 3 specifically covers process states and process control.
4. [4] D. M. Dhamdhere, Operating System: A Concept-Based Approach, 3rd Edition. Listed in the supplied syllabus as reference material.
5. [5] Richard Petersen, Linux: The Complete Reference, 6th Edition, McGraw Hill. Used as supporting reference for Linux command-line/process administration context.
6. [6] Richard Blum, Linux Command Line and Shell Scripting, Wiley India. Supporting reference for shell/process commands.
7. [7] Question-paper alignment: Summer 2026 QP-315319 and Winter 2025 QP-315319 supplied in the course materials.


**External verification notes:** Wiley's companion site lists Chapters 3–4 as Processes and Threads & Concurrency; Pearson's 9th-edition contents place Process Description and Control in Chapter 3 and Threads in Chapter 4. These references support the organization and terminology but do not replace the supplied MSBTE syllabus.
Wiley: https://bcs.wiley.com/he-bcs/Books?action=contents&bcsId=11227&itemId=1119320917
Pearson: https://www.pearson.com/en-us/subject-catalog/p/operating-systems%3A-internals-and-design-principles/P200000003349/9780137516742